

# TypoScript Reference

Extension Key: doc\_core\_tsref

Language: en

Version: 4.7.0

Keywords: forAdmins, forIntermediates

Copyright 2000-2012, Documentation Team, <documentation@typo3.org>

This document is published under the Open Content License  
available from <http://www.opencontent.org/opl.shtml>

The content of this document is related to TYPO3  
- a GNU/GPL CMS/Framework available from [www.typo3.org](http://www.typo3.org)

## Official documentation

This document is included as part of the official TYPO3 documentation. It has been approved by the TYPO3 Documentation Team following a peer-review process. The reader should expect the information in this document to be accurate - please report discrepancies to the Documentation Team (documentation@typo3.org). Official documents are kept up-to-date to the best of the Documentation Team's abilities.

## Core Manual

This document is a Core Manual. Core Manuals address the built in functionality of TYPO3 and are designed to provide the reader with in-depth information. Each Core Manual addresses a particular process or function and how it is implemented within the TYPO3 source code. These may include information on available APIs, specific configuration options, etc.  
Core Manuals are written as reference manuals. The reader should rely on the Table of Contents to identify what particular section will best address the task at hand.

# Table of Contents

<b>TypoScript Reference.....</b>	<b>1</b>		
<b>Introduction.....</b>	<b>4</b>		
About this document.....	4		
What's new.....	4		
Credits.....	4		
Feedback.....	4		
General information.....	4		
<b>Data types.....</b>	<b>6</b>		
Introduction.....	6		
Data types reference.....	6		
<b>Objects and properties.....</b>	<b>13</b>		
Introduction.....	13		
<b>Conditions.....</b>	<b>18</b>		
Condition reference.....	18		
<b>Functions.....</b>	<b>32</b>		
stdWrap.....	32		
imgResource.....	42		
imageLinkWrap.....	44		
numRows.....	45		
select.....	45		
split.....	47		
replacement.....	48		
if.....	49		
typolink.....	50		
textStyle.....	55		
encapsLines.....	56		
tableStyle.....	58		
addParams.....	58		
filelink.....	59		
round.....	61		
numberFormat.....	61		
parseFunc.....	62		
makelinks.....	65		
tags.....	66		
HTMLparser.....	67		
HTMLparser_tags.....	67		
cache.....	68		
<b>Setup.....</b>	<b>70</b>		
Top-level objects.....	70		
The "plugin" TLO.....	71		
"CONFIG".....	72		
"CONSTANTS".....	96		
"PAGE".....	96		
"FE_DATA".....	102		
"FE_TABLE".....	102		
"FRAMESET".....	103		
"FRAME".....	103		
"META".....	104		
"CARRAY".....	104		
<b>Content Objects (cObject).....</b>	<b>106</b>		
HTML.....	108		
TEXT.....	108		
COBJ_ARRAY (COA, COA_INT).....	109		
		FILE.....	109
		IMAGE.....	110
		IMG_RESOURCE.....	111
		CLEARGIF.....	111
		CONTENT.....	111
		RECORDS.....	113
		HMENU.....	114
		CTABLE.....	125
		OTABLE.....	126
		COLUMNS.....	126
		HRULER.....	127
		IMGTEXT.....	127
		CASE.....	131
		LOAD_REGISTER.....	132
		RESTORE_REGISTER.....	132
		FORM.....	133
		SEARCHRESULT.....	140
		USER and USER_INT.....	142
		TEMPLATE.....	143
		FLUIDTEMPLATE.....	147
		MEDIA.....	148
		SWFOBJECT.....	150
		QTOBJECT.....	151
		MULTIMEDIA.....	152
		SVG.....	153
		EDITPANEL.....	154
		<b>GIFBUILDER.....</b>	<b>156</b>
		GIFBUILDER.....	156
		Object names in this section.....	158
		NON-GifBuilderObj.....	166
		<b>MENU Objects.....</b>	<b>167</b>
		Common properties.....	167
		Common item states for TMENU, GMENU and	
		IMGMENU series:.....	170
		[menuObj].sectionIndex.....	171
		GMENU.....	172
		GMENU_LAYERS / TMENU_LAYERS.....	174
		GMENU_FOLDOUT.....	177
		TMENU.....	180
		TMENUITEM.....	180
		IMGMENU.....	182
		IMGMENUITEM.....	184
		JSMENU.....	184
		JSMENUITEM.....	185
		<b>Appendix A - media/scripts/ Plugins.....</b>	<b>186</b>
		media/scripts/ in general.....	186
		fe_adminLib.inc.....	186
		tipafriendLib.inc.....	196
		plaintextLib.inc.....	197
		<b>Appendix B - Standard Templates.....</b>	<b>200</b>
		static_template.....	200
		Media.....	200
		<b>Appendix C - PHP include scripts.....</b>	<b>201</b>
		Introduction.....	201

TypoScript Configuration.....	201	Introduction.....	209
Including your script.....	203	Submitting data to index.php.....	209
Case story.....	205	Search.....	209
Storing user-data or session-data.....	206	Emailforms.....	210
Using the built in "shopping basket".....	207	Database-submit.....	210
<b>Appendix D – index.php.....</b>	<b>209</b>		

# Introduction

## About this document

This document is a complete reference to all objects and properties of TypoScript as used in TYPO3 templates (and not in TSconfig).

For explanations about the syntax of TypoScript itself, please refer to the "TypoScript Syntax and In-Depth Study" manual.

This version is updated for TYPO3 version 4.7.

## What's new

The main changes include new `config.stat_*` options which allow anonymized storage of log information. For `page.includeCSS` and `page.includeJS*` conditions are now available.

The new `stdWrap` properties "cache" and "orderedStdWrap" were added. `stdWrap` has been added to the `HMENU` options "maxItems", "minItems" and "begin". The option `config.htmlTag_stdWrap`, which makes more modifications of the html tag possible, has been appended.

New properties of `filelink.icon` were appended. The option `config.pageTitleSeparator` has been added allowing further customizations of the website title. The meta object now offers the new subproperty "httpEquivalent", which makes handling of meta tags more flexible.

Additionally various descriptions were improved and many smaller mistakes were fixed.

For more details about changes in the various TYPO3 versions please refer to the links below.



### More information about changed properties

You can find a list of changes for more recent TYPO3 versions here:

TYPO3 4.2: [http://wiki.typo3.org/Documentation\\_changes\\_in\\_4.2](http://wiki.typo3.org/Documentation_changes_in_4.2)

TYPO3 4.3: [http://wiki.typo3.org/Documentation\\_changes\\_in\\_4.3](http://wiki.typo3.org/Documentation_changes_in_4.3)

TYPO3 4.4 and 4.5: [http://wiki.typo3.org/Documentation\\_changes\\_in\\_4.4\\_and\\_4.5](http://wiki.typo3.org/Documentation_changes_in_4.4_and_4.5)

TYPO3 4.6: [http://wiki.typo3.org/Documentation\\_changes\\_in\\_4.6](http://wiki.typo3.org/Documentation_changes_in_4.6)

TYPO3 4.7: [http://forge.typo3.org/projects/typo3v4-doc\\_core\\_tsref/versions/1454](http://forge.typo3.org/projects/typo3v4-doc_core_tsref/versions/1454)

## Credits

The manual was originally written by Kasper Skårhøj. Over the years it has been maintained and updated successively by Michael Stucki, François Suter and Christopher Stelmaszyk.

## Feedback

For general questions about the documentation get in touch by writing to [documentation@typo3.org](mailto:documentation@typo3.org).

If you find a bug in this manual, please file an issue in this manual's bug tracker:

[http://forge.typo3.org/projects/typo3v4-doc\\_core\\_tsref/issues](http://forge.typo3.org/projects/typo3v4-doc_core_tsref/issues)

Maintaining quality documentation is hard work and the Documentation Team is always looking for volunteers. If you feel like helping please join the documentation mailing list ([typo3.projects.documentation@lists.typo3.org](mailto:typo3.projects.documentation@lists.typo3.org)).

## General information

### Case sensitivity

All names and references in TypoScript are **case sensitive!** This is very important to notice. That means that:

```
myObject = TEXT  
myObject.value = <strong>Some HTML code</strong>
```

is not the same as

```
myObject = text  
myObject.Value = <strong>Some HTML code</strong>
```

While the first will be recognized as the content-object "TEXT" and will produce the desired output, the latter will not be recognized and will not output anything. Even if you wrote "TEXT" in uppercase in the second example, it would still not work, because the property "value" is misspelled.

Always remember: In this manual the case of objects is important.

## Version numbers

For new features TSref includes a note in which TYPO3 version the feature was added. If such a note is missing, the feature is part of TYPO3 since version 4.5 at least.

# Data types

## Introduction

The values you assign to properties in TypoScript are often of a specific format. The following table describes these formats.

E.g. if a value is defined as the type "<tag>", you're supposed to supply HTML-code. If it is of the type "resource", it's a reference to a file from the resource-field in the template. If the type is "GraphicColor" a color-definition is expected and you should supply an HTML-valid color-code or RGB-values comma-separated.

## Data types reference

Data type:	Examples:	Comment:	Default:
<b>&lt;tag&gt;</b>	<code>&lt;BODY bgcolor="red"&gt;</code>		
<b>align</b>	right	<b>right / left / center</b> Decides alignment, typically in HTML-tags	left
<b>VHalign</b>	Horizontal alignment = right and Vertical alignment = center: r , c	Pair of values separated by a comma. The first value determines the horizontal alignment, the second one the vertical alignment.  <b>Possible values:</b> <b>r/c/l , t/c/b</b>  Horizontal values standing for: right, center, left Vertical values standing for: top, center, bottom	l , t
<b>resource</b>	<i>From the resourcefield:</i> <code>toplogo*.gif</code>  <i>Reference to filesystem:</i> <code>fileadmin/picture.gif</code>	<ol style="list-style-type: none"> <li>1. A reference to a file from the resource-field in the template. You can write the exact filename or you can include an asterisk (*) as wildcard. It's recommended to include a "*" before the file extension (see example to the left). This will ensure that the file is still referenced correct even if the template is copied and the file will have it's name prepended with numbers!!</li> <li>2. If the value contains a "/" it's expected to be a reference (absolute or relative) to a file on the file-system instead of the resource-field. No support for wildcards.</li> </ol>	
<b>imgResource</b>	Here "file" is an imgResource: <code>file = toplogo*.gif</code> <code>file.width = 200</code>  GIFBUILDER: <code>file = GIFBUILDER</code> <code>file {</code> <code>... (GIFBUILDER-</code> <code>properties here)</code> <code>}</code>	<ol style="list-style-type: none"> <li>1. A "resource" (see above) + imgResource-properties (see example to the left and object-reference below) Filetypes can be anything among the allowed types defined in the configuration variable \$TYP03_CONF_VARS['GFX']['imagefile_ext']. Standard is pdf, gif, jpg, jpeg, tif, bmp, ai, pcx, tga, png.</li> <li>2. GIFBUILDER-object</li> </ol>	
<b>HTML-code</b>	<code>&lt;b&gt;Some text in bold&lt;/b&gt;</code>	pure HTML-code	
<b>target</b>	<code>_top</code> <code>_blank</code>	target in <A>-tag. This is normally the same value as the name of	

Data type:	Examples:	Comment:	Default:
	content	the root-level object that defines the frame.	
<b>imageExtension</b>	jpg web ( <i>gif or jpg ..</i> )	Image extensions can be anything among the allowed types defined in the global variable \$TYP03_CONF_VARS[ 'GFX' ] [ 'imagefile_ext' ]. Standard is pdf, gif, jpg, jpeg, tif, bmp, ai, pcx, tga, png. The value " <b>web</b> " is special. This will just ensure that an image is converted to a web image format (gif or jpg) if it happens not to be already!	
<b>degree</b>		-90 to 90, integers	
<b>posint / int+</b>		Positive integer	
<b>int</b>		integer (sometimes used generally though another type would have been more appropriate, like "pixels")	
<b>str / string / value</b>		string. (sometimes used generally though another type would have been more appropriate, like "align")	
<b>boolean</b>	1	boolean non-empty strings (but not zero) are "true"	
<b>rotation</b>		integer, degrees from 0 - 360	
<b>x,y,w,h</b>	10,10,5,5	x,y is the offset from the upper left corner. w,h is the width and height	
<b>HTML-color</b>	red #ffecc	HTML-color codes:  Black = "#000000" Silver = "#C0C0C0" Gray = "#808080" White = "#FFFFFF" Maroon = "#800000" Red = "#FF0000" Purple = "#800080" Fuchsia = "#FF00FF" Green = "#008000" Lime = "#00FF00" Olive = "#808000" Yellow = "#FFFF00" Navy = "#000080" Blue = "#0000FF" Teal = "#008080" Aqua = "#00FFFF"	
<b>GraphicColor</b>	red ( <i>HTML-color</i> ) #ffecc ( <i>HTML-color</i> ) 255,0,255 ( <i>RGB-integers</i> )  <i>Extra:</i> red : *0.8 ( <i>"red" is darkened by factor 0.8</i> ) #ffecc : +16 ( <i>"ffecc" is going to #fffedc because 16 is added</i> )	The color can be given as HTML-colors or as a comma-separated list of RGB-values (integers) You can add an extra parameter that will modify the color mathematically: Syntax: [colordef] : [modifier] where modifier can be and integer which is added/subtracted to the three RGB-channels or a floating point with an "*" before, which will then multiply the values with that factor.	
<b>page_id</b>	this 34	A page id (int) or "this" (=current page id)	
<b>pixels</b>	345	pixel-distance	
<b>list</b>	item,item2,item3	list of values	
<b>margins</b>	<i>This sets leftmargin to 10 and</i>	l,t,r,b	

Data type:	Examples:	Comment:	Default:
	<i>bottom-margin to 5. Top and right is not set (zero)</i> 10,0,0,5	left, top, right, bottom	
<b>wrap</b>	<i>This will cause the value to be wrapped in a font-tag coloring the value red:</i> <font color="red">   </font>	<...>   </...> Used to wrap something. The part on the left and right of the vertical line is placed on the left and right side of the value.	
<b>linkWrap</b>	<i>This will make a link to the root-level of a website:</i> <a href="?id={0}">   </a>	<.. {x}.>   </...> {x}; x is an integer (0-9) and points to a key in the PHP-array rootLine. The key is equal to the level the current page is on measured relatively to the root of the website. If the key exists the uid of the level that key pointed to is inserted instead of {x}. Thus we can insert page_ids from previous levels.	
<b>case</b>	upper	Case-conversion.  Possible keywords: <b>upper</b> : Convert all letters of the string to uppercase. <b>lower</b> : Convert all letters of the string to lowercase. <b>capitalize</b> : (Since TYPO3 4.6) Uppercase the first character of each word in the string. <b>ucfirst</b> : (Since TYPO3 4.6) Convert the first letter of the string to uppercase. <b>lcfirst</b> : (Since TYPO3 4.6) Convert the first letter of the string to lowercase.	
<b>space</b>	5   5	"before   after" Used for content and sets space "before   after".	
<b>date-conf</b>	d-m-y      (dd-mm-yy format)	See PHP function Date()!  a - "am" or "pm" A - "AM" or "PM" d - day of the month, numeric, 2 digits (with leading zeros) D - day of the week, textual, 3 letters; e.g. "Fri" F - month, textual, long; e.g. "January" h - hour, numeric, 12 hour format H - hour, numeric, 24 hour format i - minutes, numeric j - day of the month, numeric, without leading zeros l (lowercase 'L') - day of the week, textual, long; i.e. "Friday" m - month, numeric M - month, textual, 3 letters; e.g. "Jan" s - seconds, numeric S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd" U - seconds since the epoch Y - year, numeric, 4 digits w - day of the week, numeric, 0 represents Sunday y - year, numeric, 2 digits z - day of the year, numeric; e.g. "299"	
<b>strftime-conf</b>	Date "DD-MM-YY" = %e:%m:%y  Time "HH:MM:SS" =	%a - abbreviated weekday name according to the current locale %A - full weekday name according to the current locale	



Data type:	Examples:	Comment:	Default:
	<p><code>%H:%M:%S</code></p> <p>or just</p> <p><code>%T</code></p>	<p><code>%b</code> - abbreviated month name according to the current locale</p> <p><code>%B</code> - full month name according to the current locale</p> <p><code>%c</code> - preferred date and time representation for the current locale</p> <p><code>%C</code> - century number (the year divided by 100 and truncated to an integer, range 00 to 99)</p> <p><code>%d</code> - day of the month as a decimal number (range 00 to 31)</p> <p><code>%D</code> - same as <code>%m/%d/%y</code></p> <p><code>%e</code> - day of the month as a decimal number, a single digit is preceded by a space (range ' 1' to '31')</p> <p><code>%h</code> - same as <code>%b</code></p> <p><code>%H</code> - hour as a decimal number using a 24-hour clock (range 00 to 23)</p> <p><code>%I</code> - hour as a decimal number using a 12-hour clock (range 01 to 12)</p> <p><code>%j</code> - day of the year as a decimal number (range 001 to 366)</p> <p><code>%m</code> - month as a decimal number (range 01 to 12)</p> <p><code>%M</code> - minute as a decimal number</p> <p><code>%n</code> - newline character</p> <p><code>%p</code> - either 'am' or 'pm' according to the given time value, or the corresponding strings for the current locale</p> <p><code>%r</code> - time in a.m. and p.m. notation</p> <p><code>%R</code> - time in 24 hour notation</p> <p><code>%S</code> - second as a decimal number</p> <p><code>%t</code> - tab character</p> <p><code>%T</code> - current time, equal to <code>%H:%M:%S</code></p> <p><code>%u</code> - weekday as a decimal number [1,7], with 1 representing Monday</p> <p><code>%U</code> - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week</p> <p><code>%V</code> - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week.</p> <p><code>%W</code> - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week</p> <p><code>%w</code> - day of the week as a decimal, Sunday being 0</p> <p><code>%x</code> - preferred date representation for the current locale without the time</p> <p><code>%X</code> - preferred time representation for the current locale without the date</p> <p><code>%y</code> - year as a decimal number without a century (range 00 to 99)</p> <p><code>%Y</code> - year as a decimal number including the century</p> <p><code>%Z</code> - time zone or name or abbreviation</p> <p><code>%%</code> - a literal '%' character</p>	
<b>UNIX-time</b>	<i>Seconds to 07/04 2000 23:58:955144722</i>	Seconds since 1/1 1970...	
<b>path</b>	<i>fileadmin/stuff/</i>	path relative to the directory from which we operate.	
<b>&lt;tag&gt;-data</b>	<i>&lt;frameset&gt;-data: row could be '150,*'</i>		
<b>&lt;tag&gt;-params</b>	<i>&lt;frameset&gt;-params</i>		

Data type:	Examples:	Comment:	Default:
	<i>could be 'border="0" framespacing="0"'</i>		
<b>getText</b>	<p><b>= field : header</b> <i>get content from the \$cObj-&gt;data-array[header]</i></p> <p><b>= parameters : color</b> <i>get content from the \$cObj-&gt;parameters-array[color]</i></p> <p><b>= register : color</b> <i>get content from the \$GLOBALS['TSFE']-&gt;register[color]</i></p> <p><b>= leveltitle : 1</b> <i>get the title of the page on the first level of the rootline</i></p> <p><b>= leveltitle : -2 , slide</b> <i>get the title of the page on the level right below the current page AND if that is not present, walk to the bottom of the rootline until there's a title</i></p> <p><b>= leveluid : 0</b> <i>get the id of the root-page of the website (level zero)</i></p> <p><b>= levelfield : -1 , user_myExtField , slide</b> <i>get the value of the user defined field "user_myExtField" in the root line (requires additional configuration in \$TYPO3_CONF_VARS to include field!)</i></p> <p><b>= global : HTTP_COOKIE_VARS   some_cookie</b> <i>get the env variable \$HTTP_COOKIE_VARS[some_cookie]</i></p> <p><b>= date : d-m-y</b> <i>get the current time formatted dd-mm-yy</i></p> <p><b>= page : title</b> <i>get the current page-title</i></p> <p><b>= current : 1</b> <i>get the current value</i></p> <p><b>= level : 1</b> <i>get the rootline level of the current page</i></p>	<p>This returns a value from somewhere in a PHP-array, as defined by the type. The syntax is "type : pointer". The type is case-insensitive.</p> <p><b>field:</b> [field name from the current \$cObj-&gt;data-array in the cObj.] As default the \$cObj-&gt;data-array is \$GLOBALS['TSFE']-&gt;page (record of the current page!) In TMENU: \$cObj-&gt;data is set to the page-record for each menu item. In CONTENT/RECORDS \$cObj-&gt;data is set to the actual record In GIFBUILDER \$cObj-&gt;data is set to the data GIFBUILDER is supplied with.</p> <p><b>parameters:</b> [field name from the current \$cObj-&gt;parameters-array in the cObj.] See -&gt;parseFunc!</p> <p><b>register:</b> [field name from the \$GLOBALS['TSFE']-&gt;register] See cObject "LOAD_REGISTER"</p> <p><b>leveltitle, leveluid, levelmedia:</b> [levelTitle, uid or media in rootLine, 0- , negative = from behind, " , slide" parameter forces a walk to the bottom of the rootline until there's a "true" value to return. Useful with levelmedia.]</p> <p><b>levelfield:</b> Like "leveltitle" et al. but where the second parameter is the rootLine field you want to fetch. Syntax: [pointer, integer], [field name], ["slide"]</p> <p><b>global:</b> [GLOBAL-var, split with   if you want to get from an array! DEPRECATED, use GP, TSFE or getenv]</p> <p><b>date:</b> [date-conf]</p> <p><b>page:</b> [current page record]</p> <p><b>current:</b> 1 (gets 'current' value)</p> <p><b>level:</b> 1 (gets the rootline level of the current page)</p>	

Data type:	Examples:	Comment:	Default:
	<p>= <b>GP</b> : <b>stuff</b>  <i>get input value from query string, (&amp;stuff=)</i></p> <p>= <b>GP</b> : <b>stuff</b>   <b>key</b>  <i>get input value from query string, (&amp;stuff[key]=)</i></p> <p>= <b>getenv</b> : <b>HTTP_REFERER</b>  <i>get the env var HTTP_REFERER</i></p> <p>= <b>getIndpEnv</b> : <b>REMOTE_ADDR</b>  <i>get the client IP</i></p> <p>= <b>DB</b> : <b>tt_content:234:header</b>  <i>get the value of the header of record with uid 234 from table tt_content</i></p> <p>= <b>fullRootLine</b> : <b>-1, title</b>  <i>get the title of the page right before the start of the current website</i></p> <p>= <b>LLL:EXT:css_styled_content/pil/locallang.x:login.logout</b>  <i>get localized label for logout button</i></p> <p>= <b>path:EXT:ie7/js/ie7-standard.js</b>  <i>get path to file relative to siteroot possibly placed in an extension</i></p> <p>= <b>cObj</b> : <b>parentRecordNumber</b>  <i>get the number of the current cObject record</i></p> <p>= <b>debug</b> : <b>rootLine</b>  <i>output the current root-line visually in HTML</i></p>	<p><b>GP</b>: Value from GET or POST method. Use this instead of global  <b>GPvar: usage of "GPvar" is deprecated. Use "GP" instead</b></p> <p><b>getenv</b>: Value from environment variables</p> <p><b>getIndpEnv</b>: Value from  t3lib_div::getIndpEnv()</p> <p><b>DB</b>: Value from database, syntax is [table name] : [uid] : [field]. Any record from a table in TCA can be selected here. Only marked-deleted records does not return a value here.</p> <p><b>fullRootLine</b>: syntax is [pointer, integer], [field name], ["slide"]  This property can be used to retrieve values from "above" the current page's root. Take the below page tree and assume that we are on the page "Here you are!". Using the "levelfield" property described above, it is possible to go up only to the page "Site root", because it is the root of a new (sub-)site. With "fullRootLine" it is possible to go all the way up to page tree root. The numbers between square brackets indicate to which page each value of <i>pointer</i> would point to:</p> <ul style="list-style-type: none"> <li>- Page tree root    [-2]</li> <li>   - 1. page before   [-1]</li> <li>       - Site root (root template here!)   [0]</li> <li>           - Here you are!   [1]</li> </ul> <p>A "slide" parameter can be added just as for the "levelfield" property above.</p> <p><b>LLL</b>: Reference to a locallang (php or xml) label. Reference consists of [fileref]:[labelkey]</p> <p><b>path</b>: path to a file, possibly placed in an extension, returns empty if the file doesn't exist.</p> <p><b>cObj</b>: [internal variable from list: "parentRecordNumber"]: For CONTENT and RECORDS cObjects that are returned by a select query, this returns the row number (1,2,3,...) of the current cObject record.</p> <p><b>debug</b>: Returns HTML formatted content of PHP variable defined by keyword. Available keys are "rootLine", "fullRootLine", "data"</p> <p><b>Getting array/object elements</b>  You can fetch the value of an array/object by splitting it with a pipe " ".  Example:  = TSFE:fe_user user username</p>	

Data type:	Examples:	Comment:	Default:
		<b>Getting more values</b> By separating the value of getText with "/" (double slash) you let getText fetch the first value. If it appears empty (" " or zero) the next value is fetched and so on. Example: = <i>field:header // field:title // field:uid</i> This gets "title" if "header" is empty. If "title" is also empty it gets field "uid"	
<b>dir</b>	<i>returns a list of all pdf, gif and jpg-files from fileadmin/files/ sorted by their name reversely and with the full path (with "fileadmin/files/" prepended)</i> <b>fileadmin/files/   pdf.gif.jpg   name   r   true</b>	[path relative to the web root of the site]   [list of valid extensions]   [sorting: name, size, ext, date]   [reverse: "r"]   [return full path: boolean] Files matching is returned in a comma-separated string. <b>Note:</b> The value of config-option "lockFilePath" must equal the first part of the path. Thereby the path is locked to that folder.	
<b>function name</b>	Function: user_reverseString Method in class: user_stringReversing ->reverseString	Indicates a function or method in a class to call. See more information at the USER cObject. Depending on implementation the class or function name (but not the method name) should probably be prefixed with "user_". This can be changed in the \$TYPO3_CONF_VARS config though. Also the function / method is normally called with 2 parameters, \$conf (TS config) and \$content (some content to be processed and returned) Also if you call a method in a class, it is checked (when using the USER/USER_INT objects) whether a class with the same name, but prefixed with "ux_" is present and if so, this class is instantiated instead. See "Inside TYPO3" document for more information on extending the classes in TYPO3!	

[tsref:(datatypes)]

## Data types: Object types

These are some "data-types" that might be mentioned and valid values are shown here below:

Data type:	Comment:
<b>cObject</b>	"cObjects" are also called "Content Objects". See the section "Content Objects" later in this manual.  <b>Examples:</b> TEXT / IMAGE / MEDIA ....
<b>frameObj</b>	FRAMESET / FRAME
<b>menuObj</b>	See the section "Menu Objects" later in this manual.  <b>Examples:</b> GMENU / TMENU / IMGMENU / JSMENU
<b>GifBuilderObj</b>	See the section "GIFBUILDER" later in this manual.  <b>Examples:</b> TEXT / SHADOW / OUTLINE / EMBOSS / BOX / IMAGE / EFFECT

# Objects and properties

## Introduction

### Reference to objects

Whenever you see `->[objectname]` in the tables it means that the property is an object "*objectname*" with properties from object *objectname*. You don't need to define the objecttype.

### Calculating values (+calc)

Sometimes a data type is set to "something +calc". "+calc" indicates that the value is calculated with "+-/\*". *Be aware that the operators have no "weight"*. The calculation is just done from left to right.

#### Example:

`45 + 34 * 2 = 158` (which is the same as this in ordinary arithmetic: `(45+34)*2=158`)

### "... /stdWrap"

When a data type is set to "*type* /stdWrap" it means that the value is parsed through the stdWrap function with the properties of the value as parameters.

#### Example:

pixels /stdWrap: Here the value should be set to pixels and parsed through stdWrap.

In a real application we could do like this:

```
.pixels.field = imagewidth
.pixels.intval = 1
```

This example imports the value from the field "imagewidth" of the current \$cObj->data-array. But we don't trust the result to be an integer so we parse it through the intval()-function.

### optionSplit

optionSplit is a very tricky function. It's primarily used in the menu-objects where you define properties of a whole bunch of items at once. Here the value of properties would be parsed through this function and depending on your setup you could e.g. let the last menu-item appear with another color than the others.

The syntax is like this:

|\*| - splits the value in parts *first*, *middle*, *last*.  
 || - splits each of the *first*, *middle*, *last* in subparts

1. The priority is *last*, *first*, *middle*.
2. If the *middle*-value is empty (""), the last part of the first-value is repeated.
3. If the *first*- and *middle* value are empty, the first part of the last-value is repeated before the last value
4. The *middle* value is rotated.

ex: first1 || first2 |\*| middle1 || middle2 || middle3 |\*| last1 || last 2

#### Examples:

This is very complex and you might think that this has gone too far. But it's actually useful.

Now consider a menu with five items:

Introduction

Who are we?

Business

Contact

Links

... and a configuration like this (taken from the example-code on the first pages):

```
temp.topmenu.1.NO {
    backColor = red
    ....
}
```

If you look in this reference (see later) at the linkWrap-property of the GMENU-object, you'll discover that all properties of *.NO* are parsed through *optionSplit*. This means that before the individual menuitems are generated, the properties are split by this function. Now lets look at some examples:

## Subparts ||

### Example:

All items take on the same value. Only the *first*-part is defined and thus it's repeated to all elements

```
TS: backColor = red
```

Introduction	(red)
Who are we?	(red)
Business	(red)
Contact	(red)
Links	(red)

### Example:

Here the *first*-part is split into subparts. The third subpart is repeated because the menu has five items.

```
TS: backColor = red || yellow || green
```

<b>Introduction</b>	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(green)	first, subpart 3
Contact	(green)	first, subpart 3 (repeated)
Links	(green)	first, subpart 3 (repeated)

## Parts |\*|

### Example:

Now a *middle*-value is also defined ("*white*"). This means that after the first two menu-items the *middle*-value is used.

```
TS: backColor = red || yellow |*| white
```

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
<b>Business</b>	(white)	middle
Contact	(white)	middle
Links	(white)	middle

**Example:**

Now a *last*-value is also defined ("*blue* || *olive*"). This means that after the first two menu-items the *middle*-value is used.

TS: `backColor = red || yellow |*| white |*| blue || olive`

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
<b>Business</b>	(white)	middle
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we expand the menu a bit (*middle*-value is repeated!)

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
<b>Business</b>	(white)	middle
....	(white)	middle
....	(white)	middle
....	(white)	middle
....	(white)	middle
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we contract the menu to only four items (the *middle*-value is discarded as it's priority is the least)

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we contract the menu to only 3 items (the last subpart of the *first*-value is discarded as it's priority is less than the *last*-value)

Introduction	(red)	first, subpart 1
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

### "1: The priority is *last*, *first*, *middle*"

Now the last two examples showed that the *last*-value has the highest priority, then the *first*-value and then the *middle*-value.

### "2: If the *middle*-value is empty, the last part of the first-value is repeated"

#### Example:

The *middle*-value is left out now. Then subpart 2 of the first value is repeated. *Please observe that no space must exist between the two |\*|\*|!*

```
TS:  backColor = red || yellow |*|*| blue || olive
```

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
<b>Business</b>	(yellow)	first, subpart 2 (repeated)
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

### "3: If the *first*- and *middle* value are empty, the first part of the last-value is repeated before the last value"

#### Example:

The *middle*-value and *first*-value are left out now. Then the subpart 1 of the last value is repeated. *Please observe that no space must exist between the two |\*|\*|!*

```
TS:  backColor = |*|*| blue || olive
```

Introduction	(blue)	last, subpart 1 (repeated)
Who are we?	(blue)	last, subpart 1 (repeated)
<b>Business</b>	(blue)	last, subpart 1 (repeated)
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

### "4: The *middle* value is rotated"

#### Example:

```
TS:  backColor = red |*| yellow || green |*|
```

<b>Introduction</b>	(red)	first
Who are we?	(yellow)	middle, subpart 1
<b>Business</b>	(green)	middle, subpart 2
....	(yellow)	middle, subpart 1
....	(green)	middle, subpart 2
....	(yellow)	middle, subpart 1
....	(green)	middle, subpart 2
<b>Contact</b>	(yellow)	middle, subpart 1



**Links**

(green)

middle, subpart 2

# Conditions

## Condition reference

### General syntax

Each condition is encapsulated by square brackets. For a list of available conditions see below.

"[ELSE]" is available as else operator. It is a condition, which will return TRUE, if the previous condition returned FALSE.

Each condition block is ended with "[GLOBAL]".

#### Example:

```
[browser = msie]
# TypoScript Code for users of Internet Explorer.
[ELSE]
# TypoScript Code for users of other browsers.
[GLOBAL]
```

### General notes

Values are normally trimmed before comparison, so blanks are not taken into account.

Note that conditions cannot be used inside of curly brackets.

You may combine several conditions with two operators: && (and), || (or)

Alternatively you may use "AND" and "OR" instead of "&&" and "||". The AND operator has always higher precedence over OR. If no operator has been specified, it will default to OR.

#### Examples:

This condition will match if the visitor opens the website with Internet Explorer on Windows (but not on Mac)

```
[browser = msie] && [system = win]
```

This will match with either Opera or Firefox browsers

```
[browser = opera] || [browser = firefox]
```

This will match with either Firefox or Internet Explorer. In case of Internet Explorer, the version must be above 8.

```
[browser = firefox] || [browser = msie] && [version => 8]
```

For full explanations about conditions, please refer to "TypoScript Syntax and In-depth Study".

## browser

#### Syntax:

```
[browser = browser1,browser2,...]
```

#### Values and comparison:

Browser:	Identification:
<b>Amaya</b>	amaya
<b>AOL</b>	aol
<b>Avant</b>	avant
<b>Camino</b>	camino
<b>Google Chrome</b>	chrome

Browser:	Identification:
<b>Mozilla Firefox</b>	firefox
<b>Flock</b>	flock
<b>Gecko</b>	gecko
<b>Konqueror</b>	konqueror
<b>Lynx</b>	lynx
<b>NCSA Mosaic</b>	mosaic
<b>Microsoft Internet Explorer</b>	msie
<b>Navigator</b>	navigator
<b>Netscape Communicator</b>	netscape
<b>OmniWeb</b>	omniweb
<b>Opera</b>	opera
<b>Safari</b>	safari
<b>SeaMonkey</b>	seamonkey
<b>Webkit</b>	webkit
<b>?? (if none of the above was found in the user agent)</b>	unknown

The condition works with the user agent string. The user agent is parsed with a regular expression, which searches the string for matches with the identifications named above. If there are multiple matches, the rightmost match is finally used, because it mostly is the most correct one.

An example user agent could look like this:

```
Mozilla/5.0 (Windows NT 5.1; rv:11.0) Gecko/20100101 Firefox/11.0
```

This string contains the identifications "Gecko" and "Firefox". The condition

```
[browser = firefox]
```

evaluates to true.



### Older TYPO3 versions

Until TYPO3 4.2 the user agent was determined differently: Each value was compared with the (\$browsername.\$browserversion, e.g. "netscape4.72") using strstr(). So if the value was "netscape" or just "scape" or "net" all netscape browsers would match. If the value was "netscape4" all Netscape 4.xx browsers would match. If any value in the list matched the current browser, the condition returned true.

TYPO3 version 4.2 or older does not detect all the browsers listed above.

#### Examples:

This will match with Chrome and Opera-browsers:

```
[browser = chrome, opera]
```

## version

#### Syntax:

```
[version = value1, >value2, =value3, <value4, ...]
```

#### Comparison:

Values are floating-point numbers with "." as the decimal separator.

The values may be preceded by three operators:

Operator:	Function:
<b>[nothing]</b>	The value must be part of the beginning of the version as a string. This means that if the version is "4.72" and the value is "4" or "4.7" it matches. But "4.73" does not match. Example from syntax: "value1"
=	The value must match exactly. Version "4.72" matches only with a value of "4.72"
>	The version must be greater than the value
<	The version must be less than the value

### Examples:

This matches with exactly "4.03" browsers

```
[version= =4.03]
```

This matches with all 4+ browsers and Netscape 3 browsers

```
[version= >4][browser= netscape3]
```

## system

### Syntax:

```
[system= system1,system2]
```

### Values and comparison:

System:	Identification:
<b>Linux</b>	linux
<b>Android</b>	android
<b>OpenBSD/NetBSD/FreeBSD</b>	unix_bsd
<b>SGI / IRIX</b>	unix_sgi
<b>SunOS</b>	unix_sun
<b>HP-UX</b>	unix_hp
<b>Chrome OS</b>	chrome
<b>iOS</b>	iOS
<b>Macintosh</b>	mac
<b>Windows 7</b>	win7
<b>Windows Vista</b>	winVista
<b>Windows XP</b>	winXP
<b>Windows 2000</b>	win2k
<b>Windows NT</b>	winNT
<b>Windows 98</b>	win98
<b>Windows 95</b>	win95
<b>Windows 3.11</b>	win311
<b>Amiga</b>	amiga

Comparison with the operating system, which the website visitor uses. The system is extracted out of the useragent string.

Values are strings and a match happens if one of these strings is the first part of the system-identification.

For example if the value is "win9" this will match with "win95" and "win98" systems.

**Examples:**

This will match with windows and mac -systems only

```
[system= win,mac]
```



**Older TYPO3 versions and backwards compatibility**

TYPO3 version 4.4 or older does not detect all the systems listed above.

For backwards compatibility, some systems are also matched by more generic strings.

It is recommended to use the new identifiers documented above, but the following are valid, too:

System:	Generic identification:
<b>Android</b>	linux
<b>Chrome OS</b>	linux
<b>iOS</b>	mac
<b>Windows 7</b>	winNT
<b>Windows Vista</b>	winNT
<b>Windows XP</b>	winNT
<b>Windows 2000</b>	winNT

## device

**Syntax:**

```
[device= device1, device2]
```

**Values and comparison:**

Device:	Identification:
<b>HandHeld</b>	pda
<b>WAP phones</b>	wap
<b>Grabbers:</b>	grabber
<b>Indexing robots:</b>	robot

Values are strings and a match happens if one of these strings equals the type of device

**Examples:**

This will match WAP-phones and PDA's

```
[device = wap, pda]
```

## useragent

**Syntax:**

```
[useragent = agent]
```

**Values and comparison:**

This is a direct match on the useragent string from getenv("HTTP\_USER\_AGENT")

You have the options of putting a "\*" at the beginning and/or end of the value *agent* thereby matching with this wildcard!

**Examples:**

If the HTTP\_USER\_AGENT is "Mozilla/4.0 (compatible; Lotus-Notes/5.0; Windows-NT)" this will

match with it:

```
[useragent = Mozilla/4.0 (compatible; Lotus-Notes/5.0; Windows-NT)]
```

This will also match with it:

```
[useragent = *Lotus-Notes*]
```

... but this will also match with a useragent like this: "Lotus-Notes/4.5 ( Windows-NT )"

A short list of user-agent strings and a proper match:

HTTP_USER_AGENT:	Agent description:	Matching condition:
<b>Nokia7110/1.0+(04.77)</b>	Nokia 7110 WAP phone	[useragent= Nokia7110*]
<b>Lotus-Notes/4.5 ( Windows-NT )</b>	Lotus-Notes browser	[useragent= Lotus-Notes*]
<b>Mozilla/3.0 (compatible; AvantGo 3.2)</b>	AvantGo browser	[useragent= *AvantGo*]
<b>Mozilla/3.0 (compatible; WebCapture 1.0; Auto; Windows)</b>	Adobe Acrobat 4.0	[useragent= *WebCapture*]

## WAP-agents:

These are some of the known WAP agents:

HTTP_USER_AGENT:	HTTP_USER_AGENT (continued):
ALAV UP/4.0.7 Alcatel-BE3/1.0 UP/4.0.6c AUR PALM WAPPER Device V1.12 EricssonR320/R1A fetchpage.cgi/0.53 Java1.1.8 Java1.2.2 m-crawler/1.0 WAP Materna-WAPPreview/1.1.3 MC218 2.0 WAP1.1 Mitsu/1.1.A MOT-CB/0.0.19 UP/4.0.5j MOT-CB/0.0.21 UP/4.0.5m Nokia-WAP-Toolkit/1.2 Nokia-WAP-Toolkit/1.3beta Nokia7110/1.0 () Nokia7110/1.0 (04.67) Nokia7110/1.0 (04.67) Nokia7110/1.0 (04.69) Nokia7110/1.0 (04.70) Nokia7110/1.0 (04.71) Nokia7110/1.0 (04.73) Nokia7110/1.0 (04.74) Nokia7110/1.0 (04.76) Nokia7110/1.0 (04.77) Nokia7110/1.0 (04.80) Nokia7110/1.0 (30.05) Nokia7110/1.0	PLM's WapBrowser QWAPPER/1.0 R380 2.0 WAP1.1 SIE-IC35/1.0 SIE-P35/1.0 UP/4.1.2a SIE-P35/1.0 UP/4.1.2a UP.Browser/3.01-IG01 UP.Browser/3.01-QC31 UP.Browser/3.02-MC01 UP.Browser/3.02-SY01 UP.Browser/3.1-UPG1 UP.Browser/4.1.2a-XXXX UPG1 UP/4.0.7 Wapalizer/1.0 Wapalizer/1.1 WapIDE-SDK/2.0; (R320s (Arial)) WAPJAG Virtual WAP WAPJAG Virtual WAP WAPman Version 1.1 beta:Build W2000020401 WAPman Version 1.1 Waptor 1.0 WapView 0.00 WapView 0.20371 WapView 0.28 WapView 0.37 WapView 0.46 WapView 0.47 WinWAP 2.2 WML 1.1 wmlb YourWap/0.91 YourWap/1.16 Zetor

## language

### Syntax:

```
[language = lang1, lang2, ...]
```

### Comparison:

The values must be a straight match with the value of `getenv("HTTP_ACCEPT_LANGUAGE")` from PHP. Alternatively, if the value is wrapped in `"*"` (eg. `"*en-us"`) then it will split all languages found in the `HTTP_ACCEPT_LANGUAGE` string and try to match the value with any of those parts of the string. Such a string normally looks like `"de,en-us;q=0.7,en;q=0.3"` and `"*en-us"` would match with this string.

## IP

### Syntax:

```
[IP = ipaddress1, ipaddress2, ...]
```

### Comparison:

The values are compared with the `getenv("REMOTE_ADDR")` from PHP.

You may include `"*"` instead of one of the parts in values. You may also list the first one, two or three parts and only they will be tested.

### Examples:

These examples will match any IP-address starting with "123":

```
[IP = 123.*.*.*]
```

or

```
[IP = 123]
```

These examples will match any IP-address ending with "123" or being "192.168.1.34":

```
[IP = *.*.*.123][IP = 192.168.1.34]
```

## hostname

### Syntax:

```
[hostname = hostname1, hostname2, ...]
```

### Comparison:

The values are compared to the fully qualified hostname of getenv("REMOTE\_ADDR") retrieved by PHP.

Value is comma-list of domain names to match with. \*-wildcard allowed but cannot be part of a string, so it must match the full host name (eg. myhost.\*.com => correct, myhost.\*domain.com => wrong)

## hour

### Syntax:

```
[hour = hour1, > hour2, < hour3, ...]
```

**Note:** The first "=" sign directly after the word "hour" is always needed and is no operator. After that follow the operator and then the hour.

### Comparison:

Possible values are 0 to 23 (24-hours-format). The values in floating point are compared with the current hour of the server time.

As you see in the section "Syntax" above, you can separate multiple conditions in one with a comma. The comma will then connect them with a logical disjunction (OR), that means the whole condition will be true, when *one or more* of its operands are true.

Operator:	Function:
[none]	Requires an exact match with the value.
>	The hour must be greater than the value.
<	The hour must be less than the value.
<=	The hour must be less than or equal to the value.
>=	The hour must be greater than or equal to the value.
!=	The hour must be not equal to the value.

### Examples:

This will match, if it is between 9 and 10 o'clock (according to the server time):

```
[hour = 9]
```

This will match, if it is before 7 o'clock:

```
[hour = < 7]
```

This will match, if it is before 15 o'clock:

```
[hour = <= 14]
```

The following examples will demonstrate the usage of the comma inside the condition:

This will match, if it is between 8 and 9 o'clock (the hour equals 8) or after 16 o'clock (the hour is



bigger than or equal to 16):

```
[hour = 8, >= 16]
```

This will match between 16 and 8 o'clock (remember that the comma acts as an OR):

```
[hour = > 15, < 8]
```

In contrast a condition matching for 8 until 16 o'clock would be:

```
[hour = > 7] && [hour = < 16]
```

## minute

See "Hour" above. Uses the same syntax!

### Syntax:

```
[minute = ...]
```

### Comparison:

Minute of hour, possible values are 0-59.

Apart from that this condition uses the same way of comparison as hour.

## month

See "Hour" above. Uses the same syntax!

### Syntax:

```
[month = ...]
```

### Comparison:

Month, from January being 1 until December being 12.

Apart from that this condition uses the same way of comparison as hour.

## year

See "Hour" above. Uses the same syntax! For further information look at the date() function in the PHP manual, format string Y.

### Syntax:

```
[year = ...]
```

### Comparison:

Year, as a 4-digit number.

Apart from that this condition uses the same way of comparison as hour.

## dayofweek

See "Hour" above. Uses the same syntax!

### Syntax:

```
[dayofweek = ...]
```

### Comparison:

Day of week, starting with Sunday being 0 until Saturday being 6.

Apart from that this condition uses the same way of comparison as hour.

## dayofmonth

See "Hour" above. Uses the same syntax!

### Syntax:

```
[dayofmonth = ...]
```

### Comparison:

Day of month, possible values are 1-31.

Apart from that this condition uses the same way of comparison as hour.

## dayofyear

See "Hour" above. Uses the same syntax! For further information look at the date() function in the PHP manual, format string z.

### Syntax:

```
[dayofyear = ...]
```

### Comparison:

Day of year, 0-364 (or 365 in leap years). That this condition begins with 0 for the first day of the year means that e.g. [dayofyear = 7] will be true on the 6<sup>th</sup> of January.

Apart from that this condition uses the same way of comparison as hour.

## usergroup

### Syntax:

```
[usergroup = group1-uid, group2-uid, ...]
```

### Comparison:

The comparison can only return true if the grouplist is not empty (global var "gr\_list").

The values must either exists in the grouplist OR the value must be a "\*".

### Example:

This matches all logins:

```
[usergroup = *]
```

This matches logins from users members of groups with uid's 1 and/or 2:

```
[usergroup = 1,2]
```

## loginUser

### Syntax:

```
[loginUser = fe_users-uid, fe_users-uid, ...]
```

### Comparison:

Matches on the uid of a logged in frontend user. Works like 'usergroup' above including the \* wildcard to select ANY user.

**Example:**

This matches any login (use this instead of "[usergroup = \*]" to match when a user is logged in!):

```
[loginUser = *]
```

Additionally it is possible to check if no FE user is logged in.

**Example:**

This matches when no user is logged in:

```
[loginUser = ]
```

## page

**Syntax:**

```
[page|field = value]
```

**Comparison:**

This condition checks values of the current page record. While you can achieve the same with TSFE: [field] conditions in the frontend, this condition is usable in both frontend and backend.

**Example:**

This condition matches, if the layout field is set to 1:

```
[page|layout = 1]
```

## treeLevel

**Syntax:**

```
[treeLevel = levelnumber, levelnumber, ...]
```

**Comparison:**

This checks if the last element of the rootLine is at a level corresponding to one of the figures in "treeLevel". Level = 0 is the "root" of a website. Level=1 is the first menu level.

**Example:**

This changes something with the template, if the page viewed is on level either level 0 (basic) or on level 2

```
[treeLevel = 0,2]
```

## PIDinRootline

**Syntax:**

```
[PIDinRootline = pages-uid, pages-uid, ...]
```

**Comparison:**

This checks if one of the figures in "treeLevel" is a PID (pages-uid) in the rootline.

**Example:**

This changes something with the template, if the page viewed is or is a subpage to page 34 or page 36

```
[PIDinRootline = 34,36]
```

## PIDupinRootline

**Syntax:**

```
[PIDupinRootline = pages-uid, pages-uid, ...]
```

### Comparison:

Do the same as PIDinRootline, except the current page-uid is excluded from check.

## compatVersion

### Syntax:

```
[compatVersion = x.y.z]
```

### Comparison:

Require a minimum compatibility version. This version is not necessary equal with the TYPO3 version, it is a configurable value that can be changed in the Upgrade Wizard of the Install Tool.

"compatVersion" is especially useful if you want to provide new default settings but keep the backwards compatibility for old versions of TYPO3.

## globalVar

### Syntax:

```
[globalVar = var1 = value1, var2 > value2, var3 < value3, var4 <= value4, var5 >= value5, var6 != value6, ...]
```

### Comparison:

The values in floating point are compared to the global variables "var1", "var2" ... from above.

You can use multiple conditions in one by separating them with a comma. The comma then acts as a logical disjunction, that means the whole condition evaluates to true, whenever *one or more* of its operands are true.

Operator:	Function:
=	Requires an exact match.
>	The var must be greater than the value.
<	The var must be less than the value.
<=	The var must be less than or equal to the value.
>=	The var must be greater than or equal to the value.
!=	The var must be not equal to the value.

### Examples:

This will match with a URL like "...&print=1":

```
[globalVar = GP:print > 0]
```

This will match, if the page-id is higher than or equal to 10:

```
[globalVar = TSFE:id >= 10]
```

This will match, if the page-id is not equal to 316:

```
[globalVar = TSFE:id != 316]
```

This will match the non-existing GET/POST variable "style":

```
[globalVar = GP:style = ]
```

This will match, if the GET/POST variable "L" equals 8 or the GET/POST variable "M" equals 2 or both:

```
[globalVar = GP:L = 8, GP:M = 2]
```

This will match with the pages having the layout field set to "Layout 1":

```
[globalVar = TSFE:page|layout = 1]
```

If the constant {\$constant\_to\_turnSomethingOn} is "1" then this matches:

```
[globalVar = LIT:1 = {$constant_to_turnSomethingOn}]
```

## globalString

### Syntax:

```
[globalString = var1=value, var2= *value2, var3= *value3*, ...]
```

### Comparison:

This is a direct match on global strings.

You have the options of putting a "\*" as a wildcard or using a PCRE style regular expression (must be wrapped in "/" ) to the value.

### Examples:

If the HTTP\_HOST is "www.typo3.com" this will match with:

```
[globalString = IENV:HTTP_HOST = www.typo3.com]
```

This will also match with it:

```
[globalString = IENV:HTTP_HOST = *typo3.com]
```

... but this will also match with an HTTP\_HOST like this: "demo.typo3.com"

### IMPORTANT NOTE ON globalVar and globalString:

You can use values from global arrays and objects by dividing the var-name with a "|" (vertical line).

### Examples:

The global var \$HTTP\_POST\_VARS['key']['levels'] would be retrieved by "HTTP\_POST\_VARS|key|levels"

Also note that it's recommended to program your scripts in compliance with the php.ini-optimized settings. Please see that file (from your distribution) for details.

Caring about this means that you would get values like HTTP\_HOST by getenv() and you would retrieve GET/POST values with t3lib\_div::GP(). Finally a lot of values from the TSFE object are useful. In order to get those values for comparison with "globalVar" and "globalString" conditions, you prefix that variable's name with either "IENV:"/"ENV:" , "GP:" , "TSFE:" or "LIT:" respectively. Still the "|" divider may be used to separate keys in arrays and/or objects. "LIT" means "literal" and the string after ":" is trimmed and returned as the value (without being divided by "|" or anything)

**Notice:** Using the "IENV:" prefix is highly recommended to get server/environment variables which are system-independent. Basically this will get the value using t3lib\_div::getIndpEnv(). With "ENV:" you get the raw output from getenv() which is NOT always the same on all systems!

### Examples:

This will match with a remote-addr beginning with "192.168."

```
[globalString = IENV:REMOTE_ADDR = 192.168.*]
```

This will match with the user whose username is "test":

```
[globalString = TSFE:fe_user|user|username = test]
```

## userFunc

### *Syntax:*

```
[userFunc = user_match(checkLocalIP)]
```

### **Comparison:**

This call the function "user\_match" with the first parameter "checkLocalIP". You write that function. You decide what it checks. Function result is evaluated as true/false.

**Example:**

Put this function in your localconf.php file:

```
function user_match($cmd) {
    switch($cmd) {
        case 'checkLocalIP':
            if (strstr(getenv('REMOTE_ADDR'), '192.168')) {
                return TRUE;
            }
            break;
        case 'checkSomethingElse':
            // ....
            break;
    }
}
```

This condition will return true if the remote address contains "192.168" - which is what your function finds out.

```
[userFunc = user_match(checkLocalIP)]
```

# Functions

## stdWrap

This function is often added as a property to values in TypoScript.

Example with the property "value" of the content-object, "TEXT":

```
10 = TEXT
10.value = some text
10.case = upper
```

Here the content of the object "10" is uppercased before it's returned.

stdWrap properties are executed in the order they appear in the table below. If you want to study this further please refer to `typo3/sysext/cms/tslib/class.tslib_content.php`, where you will find the function `stdWrap()` and the array `$stdWrapOrder`, which represents the exact order of execution.

Note that the stdWrap property "orderedStdWrap" allows you to execute multiple stdWrap functions in a freely selectable order.

### Content-supplying properties of stdWrap

The properties in this table are parsed in the listed order. The properties "data", "field", "current", "cObject" (in that order!) are special as they are used to import content from variables or arrays. The above example could be rewritten to this:

```
10 = TEXT
10.value = some text
10.case = upper
10.field = header
```

Now the line "10.value = some text" is obsolete, because the whole value is "imported" from the field called "header" from the `$cObj->data`-array.

Property:	Data type:	Description:	Default:
<b>Get data:</b>			
<b>setContentToCurrent</b>	boolean	Sets the current value to the incoming content of the function.	
<b>setCurrent</b>	string /stdWrap	Sets the "current"-value. This is normally set from some outside routine, so be careful with this. But it might be handy to do this	
<b>lang</b>	Array of language keys	<p>This is used to define optional language specific values. If the global language key set by the <code>-&gt;config</code> property <code>.language</code> is found in this array, then this value is used instead of the default input value to stdWrap.</p> <p><b>Example:</b></p> <pre>config.language = de page.10 = TEXT page.10.value = I am a Berliner! page.10.lang.de = Ich bin ein Berliner!</pre> <p>Output will be "Ich bin..." instead of "I am..."</p>	
<b>data</b>	getText		



Property:	Data type:	Description:	Default:
<b>field</b>	Field name	Sets the content to the value <code>\$cObj-&gt;data[<i>field</i>]</code>  <b>Example:</b> Set content to the value of field "title": <code>".field = title"</code> <code>\$cObj-&gt;data</code> changes. See the description for the data type "getText"/field!  <b>Note:</b> You can also divide field names by <code>//</code> . Say, you set <code>"nav_title // title"</code> as the value, then the content from the field <code>nav_title</code> will be returned unless it is a blank string, in which case the <code>title</code> -field's value is returned.	
<b>current</b>	boolean	Sets the content to the "current"-value (see <code>-&gt;split</code> )	
<b>cObject</b>	cObject	Loads content from a content-object	
<b>numRows</b>	<code>-&gt;numRows</code>	Returns the number of rows resulting from the select	
<b>filelist</b>	dir /stdWrap	Reads a directory and returns a list of files. The value is exploded by <code>" "</code> into parameters: 1: The path 2: comma-list of allowed extensions (no spaces between); if empty all extensions goes. 3: sorting: name, size, ext, date, mdate (modification date) 4: reverse: Set to <code>"r"</code> if you want a reversed sorting 5: <code>fullpath_flag</code> : If set, the filelist is returned with complete paths, and not just the filename	
<b>preUserFunc</b>	Function name	Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. See <code>.postUserFunc</code>	
<b>Override / Conditions:</b>			
<b>override</b>	string /stdWrap	if "override" returns something else than <code>""</code> or zero (trimmed), the content is loaded with this!	
<b>preIfEmptyListNum</b>	(as "listNum" below)	(as "listNum" below)	
<b>ifEmpty</b>	string /stdWrap	if the content is empty (trimmed) at this point, the content is loaded with "ifEmpty". Zeros are treated as empty values!	
<b>ifBlank</b>	string /stdWrap	Same as "ifEmpty" but the check is done using <code>strlen()</code> .	

Property:	Data type:	Description:	Default:
<b>listNum</b>	int +calc +"last" +"rand"	<p>Explodes the content with "," (comma) and the content is set to the item[<i>value</i>].</p> <p><b>Special keyword:</b> "last" is set to the last element of the array!</p> <p>(Since TYPO3 4.6) <b>Special keyword:</b> "rand" returns a random item out of a list.</p> <p><b>.splitChar</b> (string): Defines the string used to explode the value. If splitChar is an integer, the character with that number is used (eg. "10" to split lines...).</p> <p>Default: "," (comma)</p> <p><b>.stdWrap</b> (stdWrap properties): stdWrap properties of the listNum...</p> <p><b>Examples:</b> We have a value of "item 1, item 2, item 3, item 4": This would return "item 3":  <pre>.listNum = last - 1</pre> <p>That way the subtitle field to be displayed is chosen randomly upon every reload:  <pre>page.5 = COA_INT page.5 {     10 = TEXT     10 {         field = subtitle         stdWrap.listNum = rand     } }</pre></p> </p>	
<b>trim</b>		PHP-function trim(); Removes whitespace around value	
<b>stdWrap</b>	->stdWrap	Recursive call to stdWrap function	
<b>required</b>	boolean	This flag requires the content to be set to some value after any content-import and treatment that might have happened now (data, field, current, listNum, trim). Zero is NOT regarded as empty! Use "if" instead! If the content is empty, "" is returned immediately.	
<b>if</b>	->if	If the if-object returns false, stdWrap returns "" immediately	
<b>fieldRequired</b>	Field name	value in this field MUST be set	
<b>Parse data:</b>			
<b>csConv</b>	string	Convert the charset of the string from the charset given as value to the current rendering charset of the frontend (renderCharset).	
<b>parseFunc</b>	object path reference / ->parseFunc	<p>Processing instructions for the content.</p> <p><b>Notice:</b> If you enter a string as value this will be taken as a reference to an object path globally in the TypoScript object tree. This will be the basis configuration for parseFunc merged with any properties you add here. It works exactly like references does for content elements.</p> <p><b>Example:</b>  <pre>parseFunc = &lt; lib.parseFunc_RTE parseFunc.tags.myTag = TEXT parseFunc.tags.myTag.value = This will be inserted when &amp;lt;myTag&amp;gt; is found!</pre></p>	
<b>HTMLparser</b>	boolean / ->HTMLparser	<p>This object allows you to parse the HTML-content and make all kinds of advanced filterings on the content. Value must be set and properties are those of -&gt;HTMLparser.</p> <p>(See "Core API" for -&gt;HTMLparser options)</p>	

Property:	Data type:	Description:	Default:
<b>split</b>	->split		
<b>replacement</b>	->replacement	(Since TYPO3 4.6) Performs an ordered search/replace on the current content with the possibility of using PCRE regular expressions. An array with numeric indices defines the order of actions and thus allows multiple replacements at once.	
<b>prioriCalc</b>	boolean	<p>Calculation of the value using operators -+*/%^ plus respects priority to + and - operators and parenthesis ().</p> <p>. (period) is decimal delimiter.</p> <p>Returns a doublevalue.</p> <p>If .prioriCalc is set to "intval" an integer is returned.</p> <p>There is no error checking and division by zero or other invalid values may generate strange results. Also you use a proper syntax because future modifications to the function used may allow for more operators and features.</p> <p><b>Examples:</b></p> <pre> 100%7 = 2 -5*.4 = 20 +6^2 = 36 6^(1+1) = 36 -5*-4+6^2-100%7 = 54 -5 * (-4+6) ^ 2 - 100%7 = 98 -5 * ((-4+6) ^ 2) - 100%7 = -22                     </pre>	
<b>char</b>	int	<p>Content is set to the chr(<i>value</i>).</p> <p>PHP: <code>\$content = chr(intval(\$conf['char']));</code></p>	
<b>intval</b>	boolean	<p>PHP function intval(); Returns an integer.</p> <p>PHP: <code>\$content = intval(\$content);</code></p>	
<b>hash</b>	string /stdWrap	<p>(Since TYPO3 4.6) Returns a hashed value of the current content. Set to one of the algorithms which are available in PHP. For a list of supported algorithms see <a href="http://www.php.net/manual/en/function.hash-algos.php">http://www.php.net/manual/en/function.hash-algos.php</a>.</p> <p><b>Example:</b></p> <pre> page.10 = TEXT page.10 {     value = test@example.com     hash = md5     wrap = &lt;img     src="(http://www.gravatar.com/avatar/ " /&gt; }                     </pre>	
<b>round</b>	->round	(Since TYPO3 4.6) Round the value with the selected method to the given number of decimals.	
<b>numberFormat</b>	->numberFormat	Format a float value to any number format you need (e.g. useful for prices).	
<b>date</b>	date-conf	<p>The content should be data-type "UNIX-time". Returns the content formatted as a date.</p> <p>PHP: <code>\$content = date(\$conf['date'], \$content);</code></p> <p>Properties:</p> <p><b>.GMT:</b> If set, the PHP function gmdate() will be used instead of date().</p> <p><b>Example</b> where a timestamp is imported:</p> <pre> .value.field = tstamp .value.date =                     </pre>	

Property:	Data type:	Description:	Default:
<b>strftime</b>	strftime-conf	<p>Exactly like "date" above. See the PHP-manual (strftime) for the codes, or datatype "strftime-conf". This formatting is useful if the locale is set in advance in the CONFIG-object. See this.</p> <p>Properties:</p> <p><b>.charset:</b> Can be set to the charset of the output string if you need to convert it to renderCharset. Default is to take the intelligently guessed charset from t3lib_cs.</p> <p><b>.GMT:</b> If set, the PHP function gmstrftime() will be used instead of strftime().</p>	
<b>age</b>	boolean or string	<p>If enabled with a "1" (number, integer) the content is seen as a date (UNIX-time) and the difference from present time and the content-time is returned as one of these eight variations:</p> <p>"xx min" or "xx hrs" or "xx days" or "xx yrs" or "xx min" or "xx hour" or "xx day" or "year"</p> <p>The limits between which layout is used are 60 minutes, 24 hours and 365 days.</p> <p>If you set this property with a non-integer, it is used to format the eight units. The first four values are the plural values and the last four are singular. This is the default string:</p> <pre>" min  hrs  days  yrs  min  hour  day  year"</pre> <p>Set another string if you want to change the units. You may include the "-signs. They are removed anyway, but they make sure that a space which you might want between the number and the unit stays.</p> <p><b>Example:</b></p> <pre>lib.ageFormat = TEXT lib.ageFormat.data = page:tstamp lib.ageFormat.age = " Minuten   Stunden   Tage   Jahre   Minute   Stunde   Tag   Jahr"</pre>	
<b>case</b>	case	<p>Converts case</p> <p>Uses "renderCharset" for the operation.</p>	
<b>bytes</b>	boolean	<p>Will format the input (an integer) as bytes: bytes, kb, mb</p> <p>If you add a value for the property "labels" you can alter the default suffixes. Labels for bytes, kilo, mega and giga are separated by vertical bar ( ) and possibly encapsulated in "". Eg: "   K  M  G" (which is the default value)</p> <p>Thus:</p> <pre>bytes.labels = "   K  M  G"</pre>	
<b>substring</b>	[p1], [p2]	<p>Returns the substring with [p1] and [p2] sent as the 2nd and 3rd parameter to the PHP substring function.</p> <p>Uses "renderCharset" for the operation.</p>	
<b>removeBadHTML</b>	boolean	<p>Removes "bad" HTML code based on a pattern that filters away HTML that is considered dangerous for XSS bugs.</p>	
<b>cropHTML</b>		<p>Crops the content to a certain length. In contrast to stdWrap.crop it respects HTML tags. It does not crop inside tags and closes open tags. Entities (like "&gt;") are counted as one char. See stdWrap.crop below for a syntax description and examples.</p> <p>Note that stdWrap.crop should not be used if stdWrap.cropHTML is already used.</p>	

Property:	Data type:	Description:	Default:
<b>stripHtml</b>	boolean	Strips all html-tags.	
<b>crop</b>		<p>Crops the content to a certain length.  Syntax: +/- (chars) = from left / from right   [string]   [boolean: keep whole words]</p> <p><b>Examples:</b>  20   ... =&gt; max 20 characters. If more, the value will be truncated to first 20 chars and prepended with "..."  -20   ... =&gt; max 20 characters. If more, the value will be truncated to last 20 chars and appended with "..."  20   ...   1 =&gt; max 20 characters. If more, the value will be truncated to last 20 chars and appended with "...". If the division is in the middle of a word, the remains of that word is removed.</p> <p>Uses "renderCharset" for the operation.</p>	
<b>rawUrlEncode</b>	boolean	Passes the content through rawurlencode()-PHP-function.	
<b>htmlSpecialChars</b>	boolean	<p>Passes the content through htmlspecialchars()-PHP-function.  Additional property ".preserveEntities" will preserve entities so only non-entity chars are affected.</p>	
<b>doubleBrTag</b>	string	All double-line-breaks are substituted with this value.	
<b>br</b>	boolean	PHP function nl2br(); converts line breaks to  -tags.	
<b>brTag</b>	string	All ASCII-codes of "10" (CR) are substituted with <i>value</i> .	
<b>encapsLines</b>	->encapsLines	Lets you split the content by chr(10) and process each line independently. Used to format content made with the RTE.	
<b>keywords</b>	boolean	Splits the content by characters ",", ";" and chr(10) (return), trims each value and returns a comma-separated list of the values.	
<b>innerWrap</b>	wrap /stdWrap	Wraps the content.	
<b>innerWrap2</b>	wrap /stdWrap	Same as .innerWrap (but watch the order in which they are executed).	
<b>fontTag</b>	wrap		
<b>addParams</b>	->addParams	Lets you add tag-parameters to the content <i>if</i> the content is a tag!	
<b>textStyle</b>	->textStyle	Wraps content in font-tags	
<b>tableStyle</b>	->tableStyle	Wraps content with table-tags	
<b>filelink</b>	->filelink	Used to make lists of links to files.	
<b>preCObject</b>	cObject	cObject prepended the content	
<b>postCObject</b>	cObject	cObject appended the content	
<b>wrapAlign</b>	align /stdWrap	Wraps content with <div style=text-align:[ <i>value</i> ];">   </div> <i>if</i> align is set	
<b>typolink</b>	->typolink	Wraps the content with a link-tag	
<b>TCAselectItem</b>	Array of properties	<p>Resolves a comma-separated list of values into the TCA item representation.</p> <p><b>.table</b> (string): <i>The Table to look up</i>  <b>.field</b> (string): <i>The field to resolve</i>  <b>.delimiter</b> (string): <i>Delimiter for concatenating multiple elements.</i></p> <p><b>Notice:</b> Currently this works only with TCA fields of type "select" which are not database relations.</p>	
<b>spaceBefore</b>	int /stdWrap	Pixels space before. Done with a clear-gif; <img ...> 	

Property:	Data type:	Description:	Default:
<b>spaceAfter</b>	int /stdWrap	Pixels space after. Done with a clear-gif; <img ...> 	
<b>space</b>	space /stdWrap	[spaceBefore]   [spaceAfter]  <b>Additional property:</b> .useDiv = 1 If set, a clear gif is not used but rather a <div> tag with a style-attribute setting the height. (Affects spaceBefore and spaceAfter as well).	
<b>wrap</b>	wrap /+.splitChar	.splitChar defines an alternative splitting character (default is " " - the vertical line)	
<b>noTrimWrap</b>	"special" wrap	This wraps the content with the values val1 and val2 in the example below - including surrounding whitespace! - without trimming the values. Note that this kind of wrap requires a " " character to begin and end the wrap.  <b>Example:</b>   val1   val2	
<b>wrap2</b>	wrap /+.splitChar	<i>same as .wrap (but watch the order in which they are executed)</i>	
<b>dataWrap</b>		The content is parsed for sections of {...} and the content of {...} is of the type getText and substituted with the result of getText.  <b>Example:</b> This will produce a tag around the content with an attribute that contains the number of the current page: <div id="{tsfe : id}">   </div>	
<b>prepend</b>	cObject	cObject prepended to content (before)	
<b>append</b>	cObject	cObject appended to content (after)	
<b>wrap3</b>	wrap /+.splitChar	<i>same as .wrap (but watch the order in which they are executed)</i>	
<b>orderedStdWrap</b>	Array of numeric keys with /stdWrap each	(Since TYPO3 4.7) Execute multiple stdWrap statements in a freely selectable order. The order is determined by the numeric order of the keys. This allows to use multiple stdWrap statements without having to remember the rather complex sorting order in which the stdWrap functions are executed.  <b>Example:</b> 10 = TEXT 10.value = a 10.orderedStdWrap { 30.wrap =  .  10.wrap = is   working 10.innerWrap = &nbsp; &nbsp;  20.wrap = This solution 20.stdWrap.wrap = &nbsp; &nbsp; } In this example orderedStdWrap is executed on the value "a". 10.innerWrap is executed first, followed by 10.wrap. Then the next key is processed which is 20. Afterwards 30.wrap is executed on what already was created. This results in "This is a working solution."	
<b>outerWrap</b>	wrap /stdWrap	<i>Wraps the complete content</i>	

Property:	Data type:	Description:	Default:
<b>insertData</b>	boolean	<p>If set, then the content string is parsed like .dataWrap above.</p> <p><b>Example:</b>  Displays the page title:  <pre>10 = TEXT 10.value = This is the page title: {page:title} 10.insertData = 1</pre> </p>	
<b>offsetWrap</b>	x,y	<p>This wraps the input in a table with columns to the left and top that offsets the content by the values of x,y. Based on the cObject OTABLE.</p> <p><b>.tableParams / .tdParams /stdWrap</b>  - used to manipulate tableParams/tdParams (default width=99%) of the offset. Default: See OTABLE.</p> <p><b>.stdWrap</b>  - stdWrap properties wrapping the offsetWrap'ed output</p>	
<b>postUserFunc</b>	function name	<p>Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. Please see the description of the cObject USER for in-depth information.</p> <p><b>Example:</b>  You can paste this example directly into a new template record.</p> <pre>page = PAGE page.typeNum=0 includeLibs.something = media/scripts/example_callfunction.php  page.10 = TEXT page.10 {     value = Hello World     postUserFunc = user_reverseString     postUserFunc.uppercase = 1 }  page.20 = TEXT page.20 {     value = Hello World     postUserFunc = user_various- &gt;reverseString     postUserFunc.uppercase = 1     postUserFunc.typolink = 11 }</pre>	
<b>postUserFuncInt</b>	function name	<p>Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. The result will be rendered non-cached, outside the main page-rendering. Please see the description of the cObject USER_INT and the cObject PHP_SCRIPT_INT (which you find in the appendix "PHP include scripts") for in-depth information. Supplied by Jens Ellerbrock</p>	

Property:	Data type:	Description:	Default:
<b>prefixComment</b>	string	<p>Prefixes content with an HTML comment with the second part of input string (divided by " ") where first part is an integer telling how many trailing tabs to put before the comment on a new line. The content is parsed through insertData.</p> <p><b>Example:</b></p> <pre>prefixComment = 2   CONTENT ELEMENT, uid: {field:uid}/{field:CType}</pre> <p>Will indent the comment with 1 tab (and the next line with 2+1 tabs) (Added in TYPO3 &gt;3.6.0RC1)</p>	
<b>editIcons</b>	string	<p>If not empty, then insert an icon linking to the typo3/alt_doc.php with some parameters to build and backend user edit form for certain fields. The value of this property is a list of fields from a table to edit. It's assumed that the current record of the cObj is the record to be edited. Syntax: <i>optional tablename : comma list of field names[list of palette-field names separated by  ]</i></p> <p><b>.beforeLastTag</b> (1,0,-1): If set (1), the icon will be inserted before the last HTML tag in the content. If -1 the icon will be prepended to the content. If zero (0) the icon is appended in the end of the content.</p> <p><b>.styleAttribute</b> (string): Adds a style-attribute to the icon image with this value. For instance you can set "position:absolute" if you want a non-destructive insertion of the icon. Notice: For general styling all edit icons has the class "frontEndEditIcons" which can be addressed from the stylesheet of the site.</p> <p><b>.iconTitle</b> (string): The title attribute of the image tag.</p> <p><b>.iconImg</b> (HTML): Alternative HTML code instead of the default icon shown. Can be used to set another icon for editing (for instance a red dot or otherwise... :-)</p> <p><b>Example:</b> This will insert an edit icon which links to a form where the header and bodytext fields are displayed and made available for editing (provided the user has access!).</p> <pre>editIcons = tt_content : header, bodytext</pre> <p>Or this line that puts the header_align and date field into a "palette" which means they are displayed on a single line below the header field. This saves some space.</p> <pre>editIcons = header[header_align date], bodytext</pre>	
<b>editPanel</b>	boolean / editPanel	See cObject EDITPANEL.	
<b>cache</b>	->cache	(Since TYPO3 4.7) Caches rendered content in the caching framework.	
<b>debug</b>	boolean	<p>Prints content with htmlspecialchars() and &lt;PRE&gt;&lt;/PRE&gt;: Useful for debugging which value stdWrap actually ends up with, if you're constructing a website with TypoScript. Should be used under construction only.</p>	



Property:	Data type:	Description:	Default:
<b>debugFunc</b>	boolean	Prints the content directly to browser with the debug() function. Should be used under construction only. Set to value "2" the content will be printed in a table - looks nicer.	
<b>debugData</b>	boolean	Prints the current data-array, \$CObj->data, directly to browser. This is where ".field" gets data from. Should be used under construction only.	

[tsref:->stdWrap]

# imgResource

imgResource contains the properties that are used with the data type imgResource.

## Example:

This scales the image toplogo.gif to the width of 200 pixels.

```
file = toplogo.gif
file.width = 200
```

Property:	Data type:	Description:	Default:
<b>ext</b>	imageExtension /stdWrap		web
<b>width</b>	pixels /stdWrap	<p>If both the width and the height are set and one of the numbers is appended by an "m", the proportions will be preserved and thus width/height are treated as maximum dimensions for the image. The image will be scaled to fit into width/height rectangle.</p> <p>If both the width and the height are set and at least one of the numbers is appended by a "c", crop-scaling will be enabled. This means that the proportions will be preserved and the image will be scaled to fit <u>around</u> a rectangle with width/height dimensions. Then, a centered portion from <u>inside</u> of the image (size defined by width/height) will be cut out.</p> <p>The "c" can have a percentage value (-100 ... +100) after it, which defines how much the cropping will be moved off the center to the border.</p> <p>Notice that you can only use "m" or "c" at the same time!</p> <p><b>Examples:</b></p> <p>This crops 120x80px from the center of the scaled image:</p> <pre>.width = 120c .height = 80c</pre> <p>This crops 100x100px; from landscape-images at the left and portrait-images centered:</p> <pre>.width = 100c-100 .height = 100c</pre> <p>This crops 100x100px; from landscape-images a bit right of the center and portrait-images a bit upper than centered:</p> <pre>.width = 100c+30 .height = 100c-25</pre>	
<b>height</b>	pixels /stdWrap	see ".width"	
<b>params</b>	Until TYPO3 4.5: string Since TYPO3 4.6: string /stdWrap	ImageMagick command-line: fx. "-rotate 90" or "-negate"	
<b>sample</b>	boolean	If set, -sample is used to scale images instead of -geometry. Sample does not use antialiasing and is therefore much faster.	

Property:	Data type:	Description:	Default:
<b>noScale</b>	boolean /stdWrap	If set, the image itself will never be scaled. Only width and height are calculated according to the other properties, so that the image is <i>displayed</i> resizedly, but the original file is used. Can be used for creating PDFs or printing of pages, where the original file could provide much better quality than a rescaled one.  <b>Example:</b> <pre>// test.jpg could e.g. have 1600 x 1200 pixels file = test.jpg file.width = 240m file.height = 240m file.noScale = 1</pre> This example results in an image tag like the following. Note that src="test.jpg" is the <i>original</i> file: <pre>&lt;img src="test.jpg" width="240" height="180" /&gt;</pre>	0
<b>alternativeTempPath</b>	string	Enter an alternative path to use for temp images. Must be found in the list in \$TYPO3_CONF_VARS['FE']['allowedTempPaths'].	
<b>frame</b>	int	Chooses which frame in an gif-animation or pdf-file. "" = first frame (zero)	
<b>import</b>	path /stdWrap	<i>value</i> should be set to the path of the file with stdWrap you get the filename from the data-array  <b>Example:</b> This returns the first image in the field "image" from the data-array: <pre>.import = uploads/pics/ .import.field = image .import.listNum = 0</pre>	
<b>maxW</b>	pixels /stdWrap	Max width	
<b>maxH</b>	pixels /stdWrap	Max height	
<b>minW</b>	pixels /stdWrap	Min width (overrides maxW/maxH)	
<b>minH</b>	pixels /stdWrap	Min height (overrides maxW/maxH)	
<b>stripProfile</b>	boolean	If set, IM-command will use a stripProfile-command which shrinks the generated thumbnails. See Install Tool for options and details. If im_useStripProfileByDefault is set in the install tool, you can deactivate it by setting stripProfile=0.  <b>Example:</b> <pre>10 = IMAGE 10.file = fileadmin/images/image1.jpg 10.file.stripProfile = 1</pre>	
<b>Masking:</b> (Black hides, white shows)			
<b>m.mask</b>	imgResource	The mask by which the image is masked onto "m.bgImg". Both "m.mask" and "m.bgImg" <b>is scaled to fit</b> the size of the imgResource image! <b>NOTE:</b> Both "m.mask" and "m.bgImg" must be valid images.	
<b>m.bgImg</b>	imgResource	<b>NOTE:</b> Both "m.mask" and "m.bgImg" must be valid images.	
<b>m.bottomImg</b>	imgResource	An image masked by "m.bottomImg_mask" onto "m.bgImg" before the imgResources is masked by "m.mask". Both "m.bottomImg" and "m.bottomImg_mask" <b>is scaled to fit</b> the size of the imgResource image! This is most often used to create an underlay for the imgResource. <b>NOTE:</b> Both "m.bottomImg" and "m.bottomImg_mask" must be valid images.	
<b>m.bottomImg_mask</b>	imgResource	(optional) <b>NOTE:</b> Both "m.bottomImg" and "m.bottomImg_mask" must be valid images.	

[tsref:-&gt;imgResource]

## imageLinkWrap

This object wraps the input (an image) with a link ready for calling up the eID "tx\_cms\_showpic" script with parameters that define such things as the size of the image, the background color of the new window and so on.

An md5-hash of the parameters is generated. The hash is also generated in the "tx\_cms\_showpic" script and the hashes **MUST** match in order for the image to be shown. This is a safety feature in order to prevent users from changing the parameters in the URL themselves.

Since TYPO3 4.5 it is also possible to display the image in a lightbox instead of using showpic.php. See the property "linkParams" below for a short instruction.

Property:	Data type:	Description:	Default:
<b>file</b>	stdWrap	Override the path of the image which is displayed	
<b>width</b>	int (1-1000) /stdWrap	If you add "m" to either the width or height, the image will be held in proportions and width/height works as max-dimensions	
<b>height</b>	int (1-1000) /stdWrap	see ".width"	
<b>effects</b>	see GIFBUILDER / effects. (from stdgraphics- library) /stdWrap	<b>Example:</b> <code>gamma=1.3   sharpen=80   solarize=70</code>	
<b>sample</b>	boolean /stdWrap	If set, -sample is used to scale images instead of -geometry. Sample does not use antialiasing and is therefore much faster.	
<b>alternativeTempPath</b>	path /stdWrap	Enter an alternative path to use for temp images. Must be found in the list in \$TYPO3_CONF_VARS['FE']['allowedTempPaths'].	
<b>title</b>	string /stdWrap	page title of the new window (HTML)	
<b>bodyTag</b>	<tag> /stdWrap	Body tag of the new window	
<b>wrap</b>	wrap /stdWrap	Wrap of the image, which is output between the body-tags	
<b>target</b>	<A>- data:target /stdWrap	NOTE: Only if "JSwindow" is set	
<b>JSwindow</b>	boolean /stdWrap	If set to "1", the image will be opened in a new window which is fitted to the dimensions of the image! You can also use stdWrap here.	
<b>JSwindow.exp and</b>	x,y /stdWrap	x and y is added to the window dimensions.	
<b>JSwindow.newWindow</b>	boolean /stdWrap	Each picture will open in a new window!	
<b>JSwindow.altUrl</b>	string /stdWrap	If this returns anything, the URL shown in the JS-window is NOT tx_cms_showpic but the url given here!	
<b>JSwindow.altUrl_noDefaultParams</b>	boolean	If this is set, the image parameters are not appended to the altUrl automatically. This is useful if you want to create them with a user function instead.	
<b>typolink</b>	->typolink	NOTE: This overrides the imageLinkWrap if it returns anything!!	
<b>directImageLink</b>	boolean /stdWrap	If true, a link to the generated image file will be returned directly (which means that showpic.php will not be used).	0

Property:	Data type:	Description:	Default:
<b>linkParams</b>	->typolink	Allows manipulation of the generated typolink, if JSwindow is not used.  <b>Example:</b> <pre>JSwindow = 0 directImageLink = 1 linkParams.ATagParams.dataWrap = class="{ \$styles.content.imgtext.linkWrap.lightboxCss Class}" rel="{ \$styles.content.imgtext.linkWrap.lightboxRelAt tribute}"</pre> <p>With these options it is easy to use a lightbox of your choice to display resizable images in the frontend: You only need to integrate the lightbox by including its JS and CSS files and to activate it for certain links (e.g. for links with the class "lightbox").</p>	
<b>stdWrap</b>	->stdWrap	Enable stdWrap for the image	
<b>enable</b>	boolean /stdWrap	<b>The image is linked ONLY if this is true!!</b>	0

[tsref:->imageLinkWrap]

#### Example:

```
1.imageLinkWrap = 1
1.imageLinkWrap {
    enable = 1
    bodyTag = <BODY bgColor=black>
    wrap = <A href="javascript:close();"> | </A>
    width = 800m
    height = 600

    JSwindow = 1
    JSwindow.newWindow = 1
    JSwindow.expand = 17,20
}
```

## numRows

This object returns the number of rows.

Property:	Data type:	Description:	Default:
<b>table</b>	Table name		
<b>select</b>	->select	Select query for the operation.  The property "selectFields" is overridden internally with "count(*)".	

[tsref:->numRows]

## select

This object generates an SQL-select statement needed to select records from the database.

Some records are hidden or timed by start and end-times. This is automatically added to the SQL-select by looking in the \$TCA (enablefields).

Also, if the "pidInList" feature is used, any page in the pid-list that is not visible for the user of the website IS REMOVED from the pidlist. Thereby no records from hidden, timed or access-protected pages are selected! Nor records from recyclers.

**Note:** Be careful if you are using GET/POST data (for example GPvar) in this object! You could introduce SQL injections!

Always secure input from outside, for example with intval.

Property:	Data type:	Description:	Default:
<b>uidInList</b>	Until TYPO3 4.5: <i>list of page_id</i> Since TYPO3 4.6: <i>list of page_id</i> /stdWrap		
<b>pidInList</b>	<i>list of page_id</i> /stdWrap		this
<b>recursive</b>	Until TYPO3 4.5: int Since TYPO3 4.6: int /stdWrap	Recursive levels for the pidInList	0
<b>orderBy</b>	<i>SQL-orderBy</i> /stdWrap	Without "order by"! E.g. "sorting, title"	
<b>groupBy</b>	<i>SQL-groupBy</i> /stdWrap	Without "group by"! E.g. "CType"	
<b>max</b>	Until TYPO3 4.5: int +calc +"total" Since TYPO3 4.6: int +calc +"total" /stdWrap	Max records  <b>Special keyword:</b> "total" is substituted with count(*)	
<b>begin</b>	Until TYPO3 4.5: int +calc +"total" Since TYPO3 4.6: int +calc +"total" /stdWrap	Begin with record number <i>value</i>  <b>Special keyword:</b> "total" is substituted with count(*)	
<b>where</b>	Until TYPO3 4.5: <i>SQL-where</i> Since TYPO3 4.6: <i>SQL-where</i> /stdWrap	Without "where"!, E.g. " (title LIKE '%SOMETHING%' AND NOT doktype) "	
<b>andWhere</b>	<i>SQL-where</i> /stdWrap	Without "AND"!, E.g. "NOT doktype".	
<b>languageField</b>	Until TYPO3 4.5: string Since TYPO3 4.6: string /stdWrap	If set, this points to the field in the record which holds a reference to a record in sys_language table. And if set, the records returned by the select-function will be selected only if the value of this field matches the \$GLOBALS['TSFE']->sys_language_uid (which is set by the config.sys_language_uid option)	
<b>selectFields</b>	Until TYPO3 4.5: string Since TYPO3 4.6: string /stdWrap	List of fields to select, or "count(*)". If the records need to be localized, please include the relevant localization-fields (uid,pid,languageField,transOrigPointerField). Otherwise the TYPO3 internal localization will not succeed.	*
<b>join</b> <b>leftjoin</b> <b>rightjoin</b>	Until TYPO3 4.5: string Since TYPO3 4.6: string /stdWrap	Enter tablename for JOIN , LEFT OUTER JOIN and RIGHT OUTER JOIN respectively.	

Property:	Data type:	Description:	Default:
<b>markers</b>	array of markers	<p>The markers defined in this section can be used, wrapped in the usual <code>###markername###</code> way, in any other property of select. Each value is properly escaped and quoted to prevent SQL injection problems. This provides a way to safely use external data (e.g. database fields, GET/POST parameters) in a query.</p> <p><code>&lt;markername&gt;.value (value)</code> Sets the value directly.</p> <p><code>&lt;markername&gt;.commaSeparatedList (bool)</code> If set, the value is interpreted as a comma-separated list of values. Each value in the list is individually escaped and quoted.</p> <p>(stdWrap properties ...) All stdWrap properties can be used for each markername.</p> <p><b>Example:</b></p> <pre> page.60 = CONTENT page.60 {   table = tt_content   select {     pidInList = 73     where = header != ###whatever###     orderBy = ###sortfield###     markers {       whatever.data = GP:first       sortfield.value = sor       sortfield.wrap =  ting     }   } }</pre>	

[tsref:-&gt;select]

## split

This object is used to split the input by a character and then parse the result onto some functions.

For each iteration the split index starting with 0 (zero) is stored in the register key SPLIT\_COUNT.

### Example:

This is an example of TypoScript-code that imports the content of field "bodytext" from the \$cObj->data-array (ln 2). The content is split by the linebreak-character (ln 4). The items should all be treated with a stdWrap (ln 5) which imports the value of the item (ln 6). This value is wrapped in a tablerow where the first column is a bullet-gif (ln 7). Finally the whole thing is wrapped in the proper table-tags (ln 9)

```

1      20 = TEXT
2      20.field = bodytext
3      20.split {
4        token.char = 10
5        cObjNum = 1
6        1.current = 1
7        1.wrap = <TR><TD valign="top"><IMG src="dot.gif"></TD><TD valign="top"> |
</TD></TR>
8      }
9      20.wrap = <TABLE border="0" cellpadding="0" cellspacing="3" width="368"> |
</TABLE><BR>
```

Property:	Data type:	Description:	Default:
<b>token</b>	str /stdWrap	string or character (token) used to split the value	
<b>max</b>	int /stdWrap	max number of splits	
<b>min</b>	int /stdWrap	min number of splits.	

Property:	Data type:	Description:	Default:
<b>returnKey</b>	int /stdWrap	Instead of parsing the split result, just return this element of the index immediately.	
<b>cObjNum</b>	<i>cObjNum</i> +optionSplit /stdWrap	This is a pointer the array of this object ("1,2,3,4"), that should treat the items, resulting from the split.	
<b>1,2,3,4</b>	->CARRAY /stdWrap	The object that should treat the value. <b>NOTE:</b> The "current"-value is set to the value of current item, when the objects are called. See "stdWrap" / current.  <b>Example (stdWrap used):</b> 1.current = 1 1.wrap = <B>   </B>  <b>Example (CARRAY used):</b> 1 { 10 = TEXT 10.current = 1 10.wrap = <B>   </B> 20 = CLEARGIF 20.height = 20 }	
<b>wrap</b>	wrap +optionSplit /stdWrap	Defines a wrap for each item.	

[tsref:-&gt;split]

## replacement

(Since TYPO3 4.6) This object performs an ordered search and replace operation on the current content with the possibility of using PCRE regular expressions. An array with numeric indices defines the order of actions and thus allows multiple replacements at once.

Property:	Data type:	Description:	Default:
<b>search</b>	string /stdWrap	Defines the string that shall be replaced.	
<b>replace</b>	string /stdWrap	Defines the string to be used for the replacement.	
<b>useRegExp</b>	boolean /stdWrap	Defines that the search and replace strings are considered as PCRE regular expressions.  <b>Example:</b> 10 { search = #(a )CAT#i replace = \lcat useRegExp = 1 }	0

[tsref:-&gt;replacement]

### Example:

```

20 = TEXT
20 {
value = There_are_a_cat,_a_dog_and_a_tiger_in_da_hood!_Yeah!
stdWrap.replacement {
10 {
search = _
replace.char = 32
}
20 {
search = in da hood
replace = around the block
}
30 {
search = #a (Cat|Dog|Tiger)#i
replace = an animal
useRegExp = 1
}
}

```



```
}
}
```

This returns: "There are an animal, an animal and an animal around the block! Yeah!"

## if

This function returns true if ALL of the present conditions are met (they are AND'ed). If a single condition is false, the value returned is false.

The returned value may still be negated by the ".negate"-property.

Property:	Data type:	Description:	Default:
<b>isTrue</b>	str /stdWrap	If the content is "true".... (not empty string and not zero)	
<b>isFalse</b>	str /stdWrap	If the content is "false"... (empty or zero)	
<b>isPositive</b>	int /stdWrap + calc	returns false if content is not positive	
<b>isGreaterThan</b>	value /stdWrap	returns false if content is not greater than ".value"	
<b>isLessThan</b>	value /stdWrap	returns false if content is not less than ".value"	
<b>equals</b>	value /stdWrap	returns false if content does not equal ".value"	
<b>isInList</b>	value /stdWrap	returns false if content is not in the comma-separated list ".value". The list in ".value" may not have spaces between elements!!	
<b>value</b>	value /stdWrap	"value" (the comparison value mentioned above)	
<b>negate</b>	boolean	This negates the result just before it exits. So if anything above returns true the overall returns ends up returning false!!	
<b>directReturn</b>	boolean	If this property exists the true/false of this value is returned. Could be used to set true/false by TypoScript constant	

[tsref:->if]

## Explanation

The "if"-function is a very odd way of returning true or false! Beware!

"if" is normally used to decide whether to render an object or return a value (see the cObjects and stdWrap)

Here is how it works:

The function returns true or false. Whether it returns true or false depends on the properties of this function. Say if you set "isTrue = 1" then result is true. If you set "isTrue.field = header" the function returns true if the field "header" in \$cObj->data is set!

If you want to compare values, you must load a base-value in the ".value"-property. Example:

```
.value = 10
.isGreaterThan = 11
```

This would return true because the value of ".isGreaterThan" is greater than 10, which is the base-value.

More complex is this:

```
.value = 10
.isGreaterThan = 11
.isTrue.field = header
.negate = 1
```

There are two conditions - isGreaterThan and isTrue. If they are both true, the total is true (AND) BUT(!) the result if the function in total is false because the ".negate"-flag inverts the result!

### Example:

This is a GIFBUILDER object that will write "NEW" on a menu-item if the field "newUntil" has a date

less than the current date!

```
...
30 = TEXT
30.text = NEW!
30.offset = 10,10
30.if {
    value.data = date: U
    isLessThan.field = newUntil
    negate = 1
}
...
```

## typolink

Wraps the incoming value with link.

If this is used from parseFunc the \$cObj->parameters-array is loaded with the link-parameters (lowercased)!

Property:	Data type:	Description:	Default:
<b>extTarget</b>	target /stdWrap	Target used for external links	_top
<b>fileTarget</b>	target /stdWrap	Target used for file links	
<b>target</b>	target /stdWrap	Target used for internal links	
<b>no_cache</b>	boolean /stdWrap	Adds a "&no_cache=1"-parameter to the link	
<b>useCacheHash</b>	boolean	If set, the additionalParams list is exploded and calculated into a hash string appended to the url, like "&cHash=ae83fd7s87". When the caching mechanism sees this value, it calculates the same value on the server based on incoming values in HTTP_GET_VARS, excluding id,type,no_cache,ftu,cHash,MP values. If the incoming cHash value matches the calculated value, the page may be cached based on this. The \$TYPO3_CONF_VARS['SYS']['encryptionKey'] is included in the hash in order to make it unique for the server and non-predictable.	
<b>additionalParams</b>	string /stdWrap	This is parameters that are added to the end of the url. This must be code ready to insert after the last parameter.  <b>Example:</b> '&print=1' '&sword_list[]=word1&sword_list[]=word2'  <b>Applications:</b> This is very useful – for example – when linking to pages from a search result. The search words are stored in the register-key SWORD_PARAMS and can be insert directly like this: <pre>.additionalParams.data =     register:SWORD_PARAMS</pre> <b>NOTE:</b> This is only active for internal links!	
<b>addQueryString</b>	boolean	Add the QUERY_STRING to the start of the link. Notice that this does not check for any duplicate parameters! This is not a problem (only the last parameter of the same name will be applied), but enable "config.uniqueLinkVars" if you still don't like it.  <b>.method:</b> If set to GET or POST then then the parsed query arguments (GET or POST data) will be used. This settings are useful if you use URL processing extensions like Real URL, which translate part of the path into query arguments. It's also possible to get both, POST and GET data, on setting	

Property:	Data type:	Description:	Default:
		<p>this to "POST,GET" or "GET,POST". The last method in this sequence takes precedence and overwrites the parts that are also present for the first method.</p> <p><b>.exclude:</b> List of query arguments to exclude from the link (eg L or cHash).</p>	
<b>jumpurl</b>	boolean	<p>Decides if the link should call the script with the jumpurl parameter in order to register any clicks in the statistics. This works the same way as "filelink.jumpurl" does – for more details see the description there.</p> <p><b>Example:</b></p> <pre>lib.parseFunc_RTE.tags.link {     typolink.jumpurl = 1     typolink.jumpurl.secure = 1     typolink.jumpurl.secure.mimeType = pdf=application/pdf, doc=application/msword, png=image/png, gif=image/gif, jpg=image/jpg }</pre> <p>These settings in the TS template will make any link to an internal file inserted in the RTE be rendered as a secure file download.</p>	0
<b>wrap</b>	wrap /stdWrap	Wraps the links.	
<b>ATagBeforeWrap</b>	boolean	If set, the link is first wrapped with ".wrap" and then the <A>-tag.	
<b>parameter</b>	string /stdWrap	<p>This is the main data that is used for creating the link. It can be the id of a page, the URL of some external page, an e-mail address or a reference to a file on the server. On top of this there can be additional information for specifying a target, a class and a title. Below are a few examples followed by full explanations.</p> <p><b>Examples:</b></p> <pre>parameter = 51</pre> <p><i>Most simple. Will create a link page 51.</i></p> <pre>parameter = 51 _blank specialLink "Very important information"</pre> <p><i>A full example. A link to page 51 that will open in a new window. The link will a class attribute with value "specialLink" and a title attribute reading "Very important information". So the result will be the following:</i></p> <pre>&lt;a href="?id=51" target="_blank" class="specialLink" title="Very important information"&gt;</pre> <pre>parameter = http://typo3.org/ - specialLink</pre> <p><i>An external link with a class attribute. Note the dash (-) that replaces the second value (the target). This makes it possible to define a class (third value) without having to define a class.</i></p> <pre>parameter = info@typo3.org - - "Send a mail to main TYP03 contact"</pre> <p><i>Create a mailto link with a title attribute (but no target and no class)</i></p> <p>As you can see from the examples, each significant part of the parameter string is separated by a space. Values that can themselves contain spaces must be enclosed in double quotes. Each of these values are described in more detail below.</p>	

Property:	Data type:	Description:	Default:
		<p><b>Destination</b></p> <p>The first value is the destination of the link. If there's a @ it will be considered to be a mail address and a mailto link will be created. If the value contains a dot (.) before the first slash (/) or a double slash (//) or if a scheme (like http) is found inside it, the link will be considered to be an external one. If there's a slash but not a dot before it, it is considered to be a path to a file and link is made to it (even if it doesn't exist as it must consider that it might be a speaking URL). In all other cases it is assumed that the value is either a page id and a page alias and a link is made to that page, if it exists.</p> <p>In the case of a link to a page, the value can be more complex than just a number or an alias. There can be three "sub-values" separated by commas. Here's an example:</p> <pre>typolink.parameter = 51,100,&amp;test=1 - - "RSS Feed"</pre> <p>The first value is the page id, the second is the type, the third will override the "additionalParams" property. It's also possible to specify a section that will override the section property. If the section mark is an integer, it will be considered as a pointer to a tt_content record. If not, it's used as is. If there's only a section mark, the link is made to the current page.</p> <p><b>Examples:</b></p> <pre>typolink.parameter = 51#345</pre> <p>Create a link to page 51 with an anchor to tt_content element number 345</p> <pre>typolink.parameter = #top</pre> <p>Create a link to the current page with an anchor called "top".</p> <p>It's also possible to direct the typolink to use a custom function (a "link handler") to build the link. This is described in more details below this table.</p> <p><b>Target or popup settings</b></p> <p>Targets are normally defined the properties described above (extTarget, fileTarget and target) but it is possible to override them by explicitly defining a target in the parameter property. It's possible to use a dash (-) to skip this value when one wants to define a third or fourth value, but no target (see examples above).</p> <p>Instead of a target, this second value can be used to define the parameters of a JavaScript popup window into which the link will be opened (using window.open). The height and width of the window can be defined, as well as additional parameters to be passed to the JavaScript function. Also see property "Jswindow".</p> <p><b>Examples:</b></p> <pre>typolink.parameter = 51 400x300</pre> <p>Open page 51 in a popup window measuring 400 by 300 pixels</p> <pre>typolink.parameter = 51 400x300:resizable=0,location=1</pre> <p>Same as above, but window will not be resizable and will show the location bar</p> <p><b>Class</b></p> <p>The third value can be used to define a class name for the</p>	

Property:	Data type:	Description:	Default:
		<p>link tag. This class is inserted in the tag before any other value from the "ATagParams" property. Beware of conflicting class attributes. It's possible to use a dash (-) to skip this value when one wants to define a fourth value, but no class (see examples above).</p> <p><b>Title</b> The standard way of defining the title attribute of the link would be to use the "title" property or even the "ATagParams" property. However it can also be set in this fourth value, in which case it will override the other settings. Note that the title should be wrapped in double quotes (") if it contains blanks.</p> <p><b>Note:</b> When used from parseFunc, the value should not be defined explicitly, but imported using:</p> <pre>typolink.parameter.data = parameters : allParams</pre>	
<b>forceAbsoluteUrl</b>	boolean	<p>Forces links to internal pages to be absolute, thus having a proper URL scheme and domain prepended.</p> <p>Additional property: .scheme: Defines the URL scheme to be used (https or http). http is the default value.</p> <p><b>Example:</b></p> <pre>typolink {     parameter = 13     forceAbsoluteUrl = 1     forceAbsoluteUrl.scheme = https }</pre>	0
<b>title</b>	string /stdWrap	Sets the title parameter of the A-tag.	
<b>JSwindow_params</b>	string	<p>Preset values for opening the window. This example lists almost all possible attributes: status=1,menubar=1,scrollbars=1,resizable=1,location=1,directories=1,toolbar=1</p>	
<b>returnLast</b>	string	<p>If set to "url" then it will return the URL of the link (\$this-&gt;lastTypoLinkUrl)</p> <p>If set to "target" it will return the target of the link.</p> <p>So, in these two cases you will not get the value wrapped but the url or target value returned!</p>	
<b>section</b>	string /stdWrap	<p>If this value is present, it's prepended with a "#" and placed after any internal url to another page in TYPO3.</p> <p>This is used create a link, which jumps from one page directly the section on another page.</p>	
<b>ATagParams</b>	<A>-params /stdWrap	<p>Additional parameters</p> <p>Example: class="board"</p>	
<b>linkAccessRestrictedPages</b>	boolean	If set, typolinks pointing to access restricted pages will still link to the page even though the page cannot be accessed.	
<b>userFunc</b>	function name	<p>This passes the link-data compiled by the typolink function to a user-defined function for final manipulation.</p> <p>The \$content variable passed to the user-function (first parameter) is an array with the keys "TYPE", "TAG", "url", "targetParams" and "aTagParams".</p> <p>TYPE is an indication of link-kind: mailto, url, file, page</p> <p>TAG is the full &lt;A&gt;-tag as generated and ready from the typolink function.</p>	

Property:	Data type:	Description:	Default:
		<p>The latter three is combined into the 'TAG' value after this formula:</p> <pre>&lt;a href=" ' . \$finalTagParts['url'] . ' ' .           \$finalTagParts['targetParams'] .           \$finalTagParts['aTagParams'] . '&gt;</pre> <p>The userfunction must return an &lt;A&gt;-tag.</p>	


[tsref:->typolink]

## Using link handlers

A feature (added in TYPO3 4.1) allows you to register a link handler for a keyword you define. For example, you can link to a page with id 34 with "<link 34>" in a typical bodytext field which converts <link> tags with "->typolink". But what if you have an extension, "pressrelease", and wanted to link to a press release item displayed by a plugin on some page you don't remember? With this feature it's possible to create the logic for this in that extension.

So, in a link field (the "parameter" value for ->typolink) you could enter "pressrelease:123":

Link



Some TypoScript will usually transfer this value to the "parameter" attribute of the ->typolink call. When "pressrelease:123" enters ->typolink as the "parameter" it will be checked if "pressrelease" is a keyword with which a link handler is associated and if so, that handler is allowed to create the link.

Registering the handler for keyword "pressrelease" is done like this:

```
$TYPO3_CONF_VARS['SC_OPTIONS']['tslib/class.tslib_content.php']['typolinkLinkHandler']  
['pressrelease'] = 'EXT:pressrelease/class.linkHandler.php:&tx_linkHandler';
```

The class file "pressrelease/class.linkHandler.php" contains the class "tx\_linkHandler" which could look like this:

```
class tx_linkHandler {  
    function main($linktxt, $conf, $linkHandlerKeyword, $linkHandlerValue, $link_param,  
    &$pObj) {  
        $lconf = array();  
        $lconf['useCacheHash'] = 1;  
        $lconf['parameter'] = 34;  
        $lconf['additionalParams'] =  
        '&tx_pressrelease[showUid]='.rawurlencode($linkHandlerValue);  
        return $pObj->typoLink($linktxt, $lconf);  
    }  
}
```

In this function, the value part after the keyword is set as the value of a GET parameter, "&tx\_pressrelease[showUid]" and the "parameter" value of a new ->typolink call is set to "34" which assumes that on page ID 34 a plugin is put that will display pressrelease 123 when called with &tx\_pressrelease[showUid]=123. In addition you can see the "userCacheHash" attribute for the typolink function used in order to produce a cached display.

The link that results from this operation will look like this:

```
<a href="index.php?id=34&tx_pressrelease[showUid]=123%3A456&chash=c0551fead6" >
```

The link would be encoded with RealURL and respect config.linkVars as long as ->typolink is used to

generate the final URL.

## textStyle

This is used to style text with a bunch of standard options + some site-specific.

Property:	Data type:	Description:	Default:
<b>align.field</b>	align	Set to field name from the \$cObj->data-array	
<b>face.field</b>	string	Set to field name from the \$cObj->data-array  [1] = "Times New Roman"; [2] = "Verdana,Arial,Helvetica,Sans serif"; [3] = "Arial,Helvetica,Sans serif";	
<b>face.default</b>	string /stdWrap	[default] = User defined	
<b>size.field</b>	string	Set to field name from the \$cObj->data-array  [1] = 1; [2] = 2; [3] = 3; [10] = "+1"; [11] = "-1";	
<b>size.default</b>	string /stdWrap	[default] = User defined	
<b>color.field</b>	string	Set to field name from the \$cObj->data-array  See "content.php" for the colors available	
<b>color.default</b>	string /stdWrap	[default] = User defined	
<b>color.1</b> <b>color.2</b>	string	[1],[2] = User defined	
<b>properties.field</b>	int	Set to field name from the \$cObj->data-array  The property values goes like this: bit 0: <B> bit 1: <I> bit 2: <U> bit 3: (uppercase)  Thus a value of 5 would result in bold and underlined text	
<b>properties.default</b>	int /stdWrap	[default] = User defined (This value will be used whenever ".field" is false!)	
<b>altWrap</b>	wrap	If this value is set, the wrapping with a font-tag based on font,size and color is NOT done. Rather the element is wrapped with this value. Use it to assign a stylesheet by setting this value to eg.  <div class="text">   </div>	

[tsref:->textStyle]

## encapsLines

Property:	Data type:	Description:	Default:
<b>encapsTagList</b>	list of strings	<p>List of tags which qualify as encapsulating tags. Must be lowercase.</p> <p><b>Example:</b></p> <pre>encapsTagList = div, p</pre> <p>This setting will recognize the red line below as encapsulated lines:</p> <pre>First line of text Some &lt;div&gt;text&lt;/div&gt; &lt;p&gt;Some text&lt;/p&gt; &lt;div&gt;Some text&lt;/div&gt; &lt;B&gt;Some text&lt;/B&gt;</pre>	
<b>remapTag.</b> <b>[tagname]</b>	string	<p>Enter a new tag name here if you wish the tagname of any encapsulation to be unified to a single tag name.</p> <p>For instance, setting this value to "remapTags.P=DIV" would convert:</p> <pre>&lt;p&gt;Some text&lt;/p&gt; &lt;div&gt;Some text&lt;/div&gt;</pre> <p>to</p> <pre>&lt;div&gt;Some text&lt;/div&gt; &lt;div&gt;Some text&lt;/div&gt;</pre> <p>([tagname] is in uppercase.)</p>	
<b>addAttributes.</b> <b>[tagname]</b>	array of strings	<p>Attributes to set in the encapsulation tag.</p> <p><b>Example:</b></p> <pre>addAttributes.P {     style=padding-bottom:0px; margin-top:1px;     margin-bottom:1px;     align=center }</pre> <p>([tagname] is in uppercase.)</p> <p><b>.setOnly =</b>  exists : This will set the value ONLY if the property does not already exist  blank : This will set the value ONLY if the property does not already exist OR is blank ("")</p> <p>Default is to always override/set the attributes value.</p>	
<b>removeWrapping</b>	boolean	<p>If set, then all existing wrapping will be removed.</p> <p>This:</p> <pre>First line of text Some &lt;div&gt;text&lt;/div&gt; &lt;p&gt;Some text&lt;/p&gt; &lt;div&gt;Some text&lt;/div&gt; &lt;B&gt;Some text&lt;/B&gt;</pre> <p>becomes this:</p> <pre>First line of text Some &lt;div&gt;text&lt;/div&gt; Some text Some text &lt;B&gt;Some text&lt;/B&gt;</pre>	



Property:	Data type:	Description:	Default:
<b>wrapNonWrappedLines</b>	wrap	Wrapping for non-encapsulated lines  <b>Example:</b> <pre>.wrapNonWrappedLines = &lt;P&gt; &lt;/P&gt;</pre> <p>This:</p> <pre>First line of text &lt;p&gt;Some text&lt;/p&gt;</pre> <p>becomes this:</p> <pre>&lt;P&gt;First line of text&lt;/P&gt; &lt;p&gt;Some text&lt;/p&gt;</pre>	
<b>innerStdWrap_all</b>	->stdWrap	Wraps the content inside all lines, whether they are encapsulated or not.	
<b>encapsLinesStdWrap.[tagname]</b>	->stdWrap	Wraps the content inside all encapsulated lines. ([tagname] is in uppercase.)	
<b>defaultAlign</b>	string /stdWrap	If set, this value is set as the default "align" value of the wrapping tags, both from .encapsTagList, .bypassEncapsTagList and .nonWrappedTag	
<b>nonWrappedTag</b>	tagname	For all non-wrapped lines, you can set here which tag it should be wrapped in. Example would be "P". This is an alternative to .wrapNonWrappedLines and has the advantage that it's attributes are set by .addAttributes as well as defaultAlign. Thus you can easier match the wrapping tags used for non-wrapped and wrapped lines.	

[tsref:->encapsLines]

**Example:**

```
encapsLines {
    encapsTagList = div,p
    remapTag.DIV = P
    wrapNonWrappedLines = <P>|</P>
    innerStdWrap_all.ifEmpty = &nbsp;
}
```

This example shows how to handle content rendered by TYPO3 and stylesheets where the <P> tag is used to encapsulate each line.

Say, you have made this content with the Rich Text Editor:

```
This is line # 1

[Above is an empty line!]
<DIV align=right>This line is right-aligned</DIV>
```

After being processed by encapsLines with the above configuration, the content looks like this:

```
<P>This is line # 1 </P>
<P>&nbsp;</P>
<P>[Above is an empty line!] </P>
<P align="right">This line is right-aligned</P>
```

Each line is nicely wrapped with <P> tags. The line from the database which was *already* wrapped (but in <DIV>-tags) has been converted to <P>, but keeps it's alignment. Overall, notice that the Rich Text Editor ONLY stored the line which was in fact right-aligned - every other line from the RTE was stored without any wrapping tags, so that the content in the database remains as human readable as possible.

**Example:**

```
# Make sure nonTypoTagStdWrap operates on content outside <typolist> and <typohead> only:
tt_content.text.20.parseFunc.tags.typolist.breakoutTypoTagContent = 1
tt_content.text.20.parseFunc.tags.typohead.breakoutTypoTagContent = 1
# ... and no <BR> before typohead.
tt_content.text.20.parseFunc.tags.typohead.stdWrap.wrap >
```

```
# Setting up nonTypoTagStdWrap to wrap the text with P-tags
tt_content.text.20.parseFunc.nonTypoTagStdWrap >
tt_content.text.20.parseFunc.nonTypoTagStdWrap.encapsLines {
    encapsTagList = div,p
    remapTag.DIV = P
    wrapNonWrappedLines = <P style="margin:0 0 0;">|</P>

    # Forcing these attributes onto the encapsulation-tags if any
    addAttributes.P {
        style=margin:0 0 0;
    }
    innerStdWrap_all.ifEmpty = &nbsp;
    innerStdWrap_all.textStyle < tt_content.text.20.textStyle
}
# finally removing the old textstyle formatting on the whole bodytext part.
tt_content.text.20.textStyle >
# ... and <BR>-tag after the content is not needed either...
tt_content.text.20.wrap >
```

This is an example of how to wrap traditional tt\_content bodytext with <P> tags, setting the line-distances to regular space like that generated by a <BR> tag, but staying compatible with the RTE features such as assigning classes and alignment to paragraphs.

## tableStyle

This is used to style a table-tag. The input is wrapped by this table-tag

Property:	Data type:	Description:	Default:
<b>align</b>	align /stdWrap		
<b>border</b>	int /stdWrap		
<b>cellspacing</b>	int /stdWrap		
<b>cellpadding</b>	int /stdWrap		
<b>color.field</b>	string	Set to field name from the \$cObj->data-array	
<b>color.default</b> <b>color.1</b> <b>color.2</b>	string	[default],[1],[2] = User defined	
<b>params</b>	<TABLE>-params		

[tsref:->tableStyle]

### Example:

```
styles.content.tableStyle {
    align.field = text_align
    border.field = table_border
    cellspacing.field = table_cellspacing
    cellpadding = 1

    color.field = table_bgColor
    color.default = {$styles.content.tableStyle.color}
    color.1 = {$styles.content.tableStyle.color1}
    color.2 = {$styles.content.tableStyle.color2}
}
```

## addParams

Property:	Data type:	Description:	Default:
<b>_offset</b>	int	Use this to define which tag you want to manipulate. 1 is the first tag in the input, 2 is the second, -1 is the last, -2 is the second last	1

Property:	Data type:	Description:	Default:
<b>(array of strings)</b>	string /stdWrap	This defines the content of each added property to the tag. If there is a tag-property with this name already (case-sensitive!) that property will be overridden! If the returned value is a blank string (but not zero!) then the existing (if any) property will not be overridden.	

[tsref:->addParams]

**Example:**

```
page.13 = TEXT
page.13.value = <tr><td valign=top>
page.13.addParams.bgcolor = {$menuCol.bgColor}
page.13.addParams._offset = -1
```

Result example:

```
<tr><td valign="top" bgcolor="white">
```

(This example adds the 'bgColor' property to the value of the TEXT cObject, if the content is not "". (zero counts as a value here!))

## filelink

Input is a filename in the path "path".

icon, size and file are rendered in the listed order.

Property:	Data type:	Description:	Default:
<b>path</b>	path /stdWrap	<b>Example:</b> path = "uploads/media/"	
<b>icon</b>	boolean /stdWrap	Set, if an icon should be shown.  The filename of the icon used is the one of the filetype of the file given in "path" (see above) plus extension (by default gif). E.g. for CSS files the icon file "css.gif" will be used by default. If for a certain filetype no icon file is found in icon.path, the file "default" plus extension (e.g. "default.gif") will be used.  Since TYPO3 4.7 the following sub-properties are available: <b>path:</b> Path to the icon set (default: typo3/sysex/cms/tslib/media/fileicons/) <b>ext:</b> File extension of icons (default: gif) <b>widthAttribute:</b> Width of the icons in pixels (default: 18) <b>heightAttribute:</b> Height of the icons in pixels (default: 16) These sub-properties all have stdWrap available.	
<b>icon_image_ext_list</b>	<i>list of image extensions</i> /stdWrap	This is a comma separated list of those file extensions that should render as thumbnails instead of icons.	
<b>icon_thumbSize</b>	string /stdWrap	Defines the size of the thumbnail in pixels.  "icon" needs to be set for the option to take effect and the file extension of the image file must be part of "icon_image_ext_list".  You can set one or two values, see the examples. If you set two values, the first value will define the max width and the second one the max height. The aspect ratio of the original image will be preserved.  <b>Examples:</b> icon_thumbSize = 150 icon_thumbSize = 40x40	
<b>iconCObject</b>	cObject	Enter a cObject to use alternatively for the icons, e.g. IMAGE type. If this is set, it'll substitute the use of the thumbs-script for display of thumbnails.	

Property:	Data type:	Description:	Default:
<b>icon_link</b>	boolean	If the icon should be linked also	
<b>labelStdWrap</b>	->stdWrap	stdWrap options for the label (by default the label is the filename) before being wrapped with the A-tags. Use this to eg. import another label from a database field or such.	
<b>wrap</b>	wrap /stdWrap	Wraps the links.	
<b>ATagBeforeWrap</b>	boolean	If set, the link is first wrapped with ".wrap" and then the <A>-tag.	
<b>file</b>	->stdWrap	stdWrap of the label (by default the label is the filename) after having been wrapped with A-tag!	
<b>size</b>	boolean /stdWrap	Set if size should be shown	
<b>jumpurl</b>	boolean	Decides if the link should call the script with the jumpurl parameter in order to register any clicks in the stat. This has the advantage that any clicks on the file will register in the stat. The disadvantage is, that users cant right-click and select "Save Target As" in the browser.  <b>Extra properties:</b> .secure = [boolean] If set, then the file pointed to by jumpurl is NOT redirected to, but rather it's read from the file and returned with a correct header. This option adds a hash and locationData to the URL and there MUST be access to the record in order to download the file. If the file position on the server is furthermore secured by a .htaccess file preventing ANY access, you've got secure download here!  .secure.mimeTypes = list of mimetypes Syntax: [ext] = [mimetype]  .parameter = [string/stdWrap] By default the jumpurl link will use the current pid and typeNum. If you need alternative values (e.g. for logging) you can specify them here. For options see typolink.parameter.  <b>Example:</b> jumpurl.secure = 1 jumpurl.secure.mimeTypes = pdf=application/pdf, doc=application/msword	
<b>target</b>	target /stdWrap		
<b>stdWrap</b>	->stdWrap		
<b>ATagParams</b>	<A>- params /stdWrap	Additional parameters  <b>Example:</b> class="board"	
<b>removePrependNumbers</b>	boolean	if set, any 2-digit prepended numbers ("eg _23") in the filename is removed.	
<b>altText</b> <b>titleText</b>	string /stdWrap	For icons (image made with "iconCObject" must have their own properties)  If no alttext is specified, it will use an empty alttext	
<b>emptyTitleHandling</b>	string /stdWrap	Value can be "keepEmpty" to preserve an empty title attribute, or "useAlt" to use the alt attribute instead.	useAlt

Property:	Data type:	Description:	Default:
<b>longdescURL</b>	string /stdWrap	For icons (image made with "iconCObject" must have their own properties)  "longdesc" attribute (URL pointing to document with extensive details about image).	

[tsref:->filelink]

**Example:**

```
1.filelink {
    path = uploads/media/
    icon = 1
    icon.wrap = <td> | </td>
    size = 1
    size.wrap = <td> | </td>
    file.fontTag = {$styles.content.uploads.wrap}
    file.wrap = <td> | </td>
    jumpurl = 1
    target = _blank
    stdWrap = <tr> | </tr>
}
```

## round

(Since TYPO3 4.6) With this property you can round the value up, down or to a certain number of decimals. For each roundType the according PHP function will be used.

The value will be converted to a float value before applying the selected round method.

Property:	Data type:	Description:	Default:
<b>roundType</b>	string /stdWrap	Round method which should be used.  Possible keywords: - <b>ceil</b> : Round the value up to the next integer. - <b>floor</b> : Round the value down to the previous integer. - <b>round</b> : Round the value to the specified number of decimals.	round
<b>decimals</b>	integer /stdWrap	Number of decimals the rounded value will have. Only used with the roundType "round". Defaults to 0, so that your input will in that case be rounded up or down to the next integer.	0

[tsref:->round]

**Examples:**

```
lib.number = TEXT
lib.number {
    value = 3.14159
    round.roundType = round
    round.decimals = 2
}
```

This returns 3.14.

## numberFormat

With this property you can format a float value and display it as you want, for example as a price. It is a wrapper for the number\_format() function of PHP.

You can define how many decimals you want and which separators you want for decimals and thousands.

Since the properties are finally used by the PHP function number\_format(), you need to make sure that they are valid parameters for that function. Consult the PHP manual, if unsure.

Property:	Data type:	Description:	Default:
<b>decimals</b>	integer	Number of decimals the formatted number will have. Defaults to 0,	0

Property:	Data type:	Description:	Default:
	/stdWrap	so that your input will in that case be rounded up or down to the next integer.	
<b>dec_point</b>	string /stdWrap	Character that divides the decimals from the rest of the number. Defaults to ".".	.
<b>thousands_sep</b>	string /stdWrap	Character that divides the thousands of the number. Defaults to ","; set an empty value to have no thousands separator.	,

[tsref:->numberFormat]

### Examples:

```
lib.myPrice = TEXT
lib.myPrice {
    value = 0.8
    numberFormat {
        decimals = 2
        dec_point.cObject = TEXT
        dec_point.cObject {
            value = .
            lang.de = ,
        }
    }
    noTrimWrap = || &euro;|
}
# Will basically result in "0.80 €", but for German in "0,80 €".

lib.carViews = CONTENT
lib.carViews {
    table = tx_mycarext_car
    select.pidInList = 42
    renderObj = TEXT
    renderObj {
        field = views
        # By default use 3 decimals or
        # use the number given by the Get/Post variable precisionLevel, if set.
        numberFormat.decimals = 3
        numberFormat.decimals.override.data = GP:precisionLevel
        numberFormat.dec_point = ,
        numberFormat.thousands_sep = .
    }
}
# Could result in something like "9.586,007".
```

## parseFunc

This object is used to parse some content for stuff like special typo tags, the "makeLinks"-things and so on...

### Example:

This example takes the content of the field "bodytext" and parses it through the makelinks-functions and substitutes all <LINK> and <TYPOLIST>-tags with something else.

```
tt_content.text.default {
    20 = TEXT
    20.field = bodytext
    20.wrap = | <BR>
    20.brTag = <br>
    20.parseFunc {
        makelinks = 1
        makelinks.http.keep = path
        makelinks.http.extTarget = _blank
        makelinks.mailto.keep = path
        tags {
            link = TEXT
            link {
                current = 1
                typolink.extTarget = _blank
                typolink.target=${<LinkTagTarget>}
                typolink.wrap = <B><FONT color=red>|</FONT></B>
                typolink.parameter.data = parameters : allParams
            }
        }
    }
}
```

```

    }

    typolist < tt_content.bullets.default.20
    typolist.trim = 1
    typolist.field >
    typolist.current = 1
  }
}

```

Property:	Data type:	Description:	Default:
<b>externalBlocks</b>	list of tag names/ +properties	<p>This allows you to pre-split the content passed to parseFunc so that only content outside the blocks with the given tags is parsed.</p> <p><b>Extra properties:</b></p> <p><b>[tagname]</b> {</p> <ul style="list-style-type: none"> <li><b>callRecursive</b> = [boolean]; If set, the content of the block is directed into parseFunc again. Otherwise the content is just passed through with no other processing than stdWrap (see below)</li> <li><b>callRecursive.dontWrapSelf</b> = [boolean]; If set, the tags of the block is <i>not</i> wrapped around the content returned from parseFunc.</li> <li><b>callRecursive.alternativeWrap</b> = Alternative wrapping instead of the original tags.</li> <li><b>callRecursive.tagStdWrap</b> = -&gt;stdWrap processing of the block-tags.</li> <li><b>stdWrap</b> = -&gt;stdWrap processing of the whole block (regardless of whether callRecursive was set.)</li> <li><b>stripNLprev</b> = [boolean]; Strips off last linebreak of the previous outside block</li> <li><b>stripNLnext</b> = [boolean]; Strips off first linebreak of the next outside block</li> <li><b>stripNL</b> = [boolean]; Does both of the above.</li> </ul> <p><b>HTMLtableCells</b> = [boolean]; If set, then the content is expected to be a table and every table-cell is traversed.</p> <p># Below, default is all cells and 1,2,3... overrides for specific cols.</p> <p><b>HTMLtableCells.[default/1/2/3/...] {</b></p> <ul style="list-style-type: none"> <li><b>callRecursive</b> = [boolean]; The content is parsed through current parseFunc</li> <li><b>stdWrap</b> = -&gt;stdWrap processing of the content in the cell</li> <li><b>tagStdWrap</b> = -&gt; The &lt;TD&gt; tag is processed by -&gt;stdWrap</li> </ul> <p><b>HTMLtableCells.addChr10BetweenParagraphs</b> = [boolean]; If set, then all &lt;/P&gt;&lt;P&gt; appearances will have a chr(10) inserted between them</p> <p>}</p> <p><b>Example:</b></p> <p>This example is used to split regular bodytext content so that tables and blockquotes in the bodytext are processed correctly. The blockquotes are passed into parseFunc again (recursively) and further their top/bottom margins are set to 0 (so no apparent line breaks are seen)</p> <p>The tables are also displayed with a number of properties of the cells overridden.</p> <pre> tt_content.text.20.parseFunc.externalBlocks {     blockquote.callRecursive=1     blockquote.callRecursive.tagStdWrap.HTMLparser = 1     blockquote.callRecursive.tagStdWrap.HTMLparser {         tags.blockquote.fixAttrib.style.list = margin- bottom:0;margin-top:0;         tags.blockquote.fixAttrib.style.always=1     }     blockquote.stripNLprev=1     blockquote.stripNLnext=1      table.stripNL=1     table.stdWrap.HTMLparser = 1     table.stdWrap.HTMLparser {         tags.table.overrideAttribs = border=0 </pre>	

Property:	Data type:	Description:	Default:
		<pre> cellpadding=2 cellspacing=1 style="margin-top: 10px; margin-bottom: 10px;" tags.tr.allowedAttribs=0 tags.td.overrideAttribs = valign="top" bgcolor="#eeeeee" style="font-family: Verdana, Geneva, Arial, Helvetica, sans-serif; font-size: 10px;" } } </pre>	
<b>constants</b>	boolean	<p>The top-level defined constants will be substituted in the text. The constant-name is wrapped in "###".</p> <p><b>Example:</b></p> <pre>constants.EMAIL = email@email.com</pre> <p>(NOTE: This is top-level TypoScript!)</p> <p>All cases of the string ###EMAIL### will be substituted in the text. The constants are defined as a top-level object.</p>	
<b>short</b>	<i>array of strings</i>	<p>Like constants above, but local.</p> <p><b>Example:</b></p> <p>This substitutes all occurrences of "T3" with "TYPO3 CMS" and "T3web" with a link to typo3.com.</p> <pre> short {     T3 = TYPO3 CMS     T3web = &lt;a href="http://typo3.com"&gt;typo3&lt;/a&gt; } </pre>	
<b>plainTextStdWrap</b>	->stdWrap	This is stdWrap properties for all non-tag content.	
<b>userFunc</b>	function name	<p>This passes the non-tag content to a function of your own choice. Similar to e.g. .postUserFunc in stdWrap.</p> <p>Remember the function name must possibly be prepended "user_"</p>	
<b>nonTypoTagStdWrap</b>	->stdWrap	<p>Like .plainTextStdWrap. Difference:</p> <p>.plainTextStdWrap works on ALL non-tag pieces in the text. .nonTypoTagStdWrap is post processing of all text (including tags) between special TypoTags (unless .breakoutTypoTagContent is not set for the TypoTag)</p>	
<b>nonTypoTagUserFunc</b>	function name	<p>Like .userFunc. Differences is (like nonTypoTagStdWrap) that this is post processing of all content pieces around TypoTags while .userFunc processes all non-tag content. (Notice: .breakoutTypoTagContent must be set for the TypoTag if it's excluded from nonTypoTagContent)</p>	
<b>sword</b>	wrap	<p>Marks up any words from the GET-method send array sword_list[] in the text. The word MUST be at least two characters long!</p> <p><b>NOTE:</b> works only with \$GLOBALS['TSFE']-&gt;no_cache==1</p>	<font color="red"></font>
<b>makelinks</b>	boolean / ->makelinks	Convert webaddresses prefixed with "http://" and mail-adresses prefixed with "mailto:" to links.	
<b>tags</b>	->tags	Here you can define <b>custom tags</b> that will parse the content to something.	
<b>allowTags</b>	list of strings	<p>List of tags, which are allowed to exist in code!</p> <p>Highest priority: If a tag is found in allowTags, denyTags is ignored!!</p>	
<b>denyTags</b>	list of strings	<p>List of tags, which may NOT exist in code! (use "*" for all.)</p> <p>Lowest priority: If a tag is NOT found in allowTags, denyTags is checked. If denyTags is not "*" and the tag is not found in the list, the tag may exist!</p> <p><b>Example:</b></p> <p>This allows &lt;B&gt;, &lt;I&gt;, &lt;A&gt; and &lt;IMG&gt; -tags to exist</p> <pre> .allowTags = b,i,a,img .denyTags = * </pre>	
<b>if</b>	->if	if "if" returns false the input value is not parsed, but returned directly.	



[tsref:->parseFunc]

## makelinks

makelinks substitutes all appearances of

`http://www.webaddress.rld`

`mailto:name@email.rld`

... to a real linktag.

Property:	Data type:	Description:	Default:
<b>http.extTarget</b>	target	The target of the link	_top
<b>http.wrap</b>	wrap /stdWrap	wrap around the link	
<b>http.ATagBeforeWrap</b>	boolean	If set, the link is first wrapped with <i>http.wrap</i> and then the <A>-tag.	
<b>http.keep</b>	list: "scheme","path","query"	As default the link-text will be the full domain-name of the link. <b>Examples:</b> <code>http://www.webaddress.rld/test/doc.php?id=3</code> <code>"": www.webaddress.rld</code> <code>"scheme": http://www.webaddress.rld</code> <code>"scheme,path": http://www.webaddress.rld/test/doc.php</code> <code>"scheme,path,query": http://www.webaddress.rld/test/doc.php?id=3</code>	
<b>http.ATagParams</b>	<A>-params /stdWrap	Additional parameters <b>Example:</b> <code>class="board"</code>	
<b>mailto.wrap</b>	wrap /stdWrap	wrap around the link	
<b>mailto.ATagBeforeWrap</b>	boolean	If set, the link is first wrapped with <i>mailto.wrap</i> and then the <A>-tag.	
<b>mailto.ATagParams</b>	<A>-params /stdWrap	Additional parameters <b>Example:</b> <code>class="board"</code>	

[tsref:->makelinks]

## tags

Used to create custom tags and define how they should be parsed. This is used in conjunction with *parseFunc*.

Property:	Data type:	Description:	Default:
<b>Array...</b>	cObject +stripNL +breakoutTypoTagContent	<p>Every entry in the <i>Array...</i> corresponds to a tag, that will be parsed. The elements MUST be in lowercase.</p> <p>Every entry must be set to a content-object.</p> <p>"current" is set to the content of the tag, eg &lt;TAG&gt;content&lt;/TAG&gt;: here "current" is set to "content".</p> <p><b>Parameters:</b></p> <p>Parameters of the tag is set in \$cObj-&gt;parameters (key is lowercased):</p> <p>&lt;TAG COLOR="red"&gt;content&lt;/TAG&gt;</p> <p>=&gt; \$cObj-&gt;parameters[color] = red</p> <p><b>Special added properties to the content-object:</b></p> <p>\$cObj-&gt;parameters[allParams]: this is automatically set to the whole parameter-string of the tag, eg ' color="red"'</p> <p>[cObject].stripNL: is a boolean option, which tells <i>parseFunc</i> that NewLines before and after content of the tag should be stripped.</p> <p>[cObject].breakoutTypoTagContent: is a boolean option, which tells <i>parseFunc</i> that this block of content is breaking up the nonTypoTag content and that the content after this must be re-wrapped.</p> <p><b>Examples:</b></p> <pre>tags.bold = TEXT tags.bold {     current = 1     wrap = &lt;B&gt;   &lt;/B&gt; } tags.bold.stripNL = 1</pre>	

[tsref:->tags]

### Example:

This example creates 4 custom tags. The <LINK>-, <TYPOLIST>-, <GRAFIX>- and <PIC>-tags

<LINK> is made into a typolink and provides an easy way of creating links in text

<TYPOLIST> is used to create bullet-lists

<GRAFIX> will create a gif-file 90x10 pixels where the text is the content of the tag.

<PIC> lets us place an image in the text. The content of the tag should be the image-reference in "fileadmin/"

```
tags {
    link = TEXT
    link {
        current = 1
        typolink.extTarget = _blank
        typolink.target={$cLinkTagTarget}
        typolink.wrap = <B><FONT color=red>|</FONT></B>
        typolink.parameter.data = parameters : allParams
    }

    typolist < tt_content.bullets.default.20
    typolist.trim = 1
    typolist.field >
    typolist.current = 1

    grafix = IMAGE
    grafix {
        file = GIFBUILDER
        file {
            XY = 90,10
            100 = TEXT
            100.text.current = 1
            100.offset = 5,10
        }
    }
}
```

```

        100.nicetext = 1
    }
}
pic = IMAGE
pic.file.import = fileadmin/
pic.file.import.current = 1
}

```

## HTMLparser

Property:	Data type:	Description:
<b>allowTags</b>	list of tags	Default allowed tags
<b>tags.[tagname]</b>	boolean/->HTMLparser_tags	Either set this property to 0 or 1 to allow or deny the tag. If you enter ->HTMLparser_tags properties, those will automatically overrule this option, thus it's not needed then. [tagname] in lowercase.
<b>localNesting</b>	list of tags, must be among preserved tags	List of tags (among the already set tags), which will be forced to have the nesting-flag set to true
<b>globalNesting</b>	(ibid)	List of tags (among the already set tags), which will be forced to have the nesting-flag set to "global"
<b>rmTagIfNoAttrib</b>	(ibid)	List of tags (among the already set tags), which will be forced to have the rmTagIfNoAttrib set to true
<b>noAttrib</b>	(ibid)	List of tags (among the already set tags), which will be forced to have the allowedAttribs value set to zero (which means, all attributes will be removed).
<b>removeTags</b>	(ibid)	List of tags (among the already set tags), which will be configured so they are surely removed.
<b>keepNonMatchedTags</b>	boolean / "protect"	If set (true=1), then all tags are kept regardless of tags present as keys in \$tags-array. If "protect", then the preserved tags have their <> converted to &lt; and &gt; Default is to REMOVE all tags, which are not specifically assigned to be allowed! So you might probably want to set this value!
<b>htmlSpecialChars</b>	-1 / 0 / 1 / 2	This regards all content which is NOT tags: "0" means "disabled" - nothing is done "1" means the content outside tags is htmlspecialchars()'ed (PHP-function which converts &"<> to &...;) "2" is the same as "1" but entities like "&quot;" or "&#234;" are untouched. "-1" does the opposite of "1" - converts &lt; to <, &gt; to >, &quot; to "& etc.
<b>xhtml_cleaning</b>	boolean	Cleans up the content for XHTML compliance. Still slightly experimental and supports only some clean up operations (like conversion tags and attributes to lower case).

[page:->HTMLparser; tsref:->HTMLparser]

## HTMLparser\_tags

Property:	Data type:	Description:
<b>overrideAttribs</b>	string	If set, this string is preset as the attributes of the tag.
<b>allowedAttribs</b>		'0' (zero) = no attributes allowed, '[commalist of attributes]' = only allowed attributes. If blank/not set, all attributes are allowed.
<b>fixAttrib.[attribute].set</b>	string	Force the attribute value to this value.
<b>fixAttrib.[attribute].unset</b>	boolean	If set, the attribute is unset.
<b>fixAttrib.[attribute].default</b>	string	If no attribute exists by this name, this value is set as default value (if this value is not blank)

Property:	Data type:	Description:
<code>fixAttrib.[attribute].always</code>	boolean	If set, the attribute is always processed. Normally an attribute is processed only if it exists
<code>fixAttrib.[attribute].trim</code> <code>fixAttrib.[attribute].intval</code> <code>fixAttrib.[attribute].upper</code> <code>fixAttrib.[attribute].lower</code>	boolean	If any of these keys are set, the value is passed through the respective PHP-functions.
<code>fixAttrib.[attribute].range</code>	[low],[high]	Setting integer range.
<code>fixAttrib.[attribute].list</code>	list of values, trimmed	Attribute value must be in this list. If not, the value is set to the first element.
<code>fixAttrib.[attribute].removeIfFalse</code>	boolean/"blank" string	If set, then the attribute is removed if it is "false". If this value is set to "blank" then the value must be a blank string (that means a "zero" value will not be removed)
<code>fixAttrib.[attribute].removeIfEquals</code>	string	If the attribute value matches the value set here, then it is removed.
<code>fixAttrib.[attribute].casesensitiveComp</code>	boolean	If set, the comparison in <code>.removeIfEquals</code> and <code>.list</code> will be case-sensitive. At this point, it's insensitive.
<code>fixAttrib.[attribute].prefixLocalAnchors</code>	integer	If the first char is a "#" character (anchor of fx. <a> tags) this will prefix either a relative or absolute path. If the value is "1" you will get the absolute path ( <code>t3lib_div::getIndpEnv("TYPO3_REQUEST_URL")</code> ) If the value is "2" you will get the relative path (stripping of <code>t3lib_div::getIndpEnv("TYPO3_SITE_URL")</code> )  <b>Example:</b>  <pre>...fixAttrib.href.prefixLocalAnchors = 1</pre>
<code>fixAttrib.[attribute].prefixRelPathWith</code>	string	If the value of the attribute seems to be a relative URL (no scheme like "http" and no "/" as first char) then that value of this property will be prefixed the attribute.  <b>Example:</b>  <pre>...fixAttrib.src.prefixRelPathWith = http://192.168.230.3/typo3/32/dummy/</pre>
<code>fixAttrib.[attribute].userFunc</code>	function reference	User function for processing of the attribute.  <b>Example:</b>  <pre>...fixAttrib.href.userFunc = tx_realurl-&gt;test_urlProc</pre>
<b>protect</b>	boolean	If set, the tag <> is converted to &lt; and &gt;
<b>remap</b>	string	If set, the tagname is remapped to this tagname
<b>rmTagIfNoAttrib</b>	boolean	If set, then the tag is removed if no attributes happend to be there.
<b>nesting</b>		If set true, then this tag must have starting and ending tags in the correct order. Any tags not in this order will be discarded. Thus '</B><B><I></B></I></B>' will be converted to '<B><I></B></I>'. Is the value "global" then true nesting in relation to other tags marked for "global" nesting control is preserved. This means that if <B> and <I> are set for global nesting then this string '</B><B><I></B></I></B>' is converted to '<B></B>'

[page:->HTMLparser\_tags; tsref:->HTMLparser\_tags]

## cache

(Since TYPO3 4.7) Stores the rendered content into the caching framework and reads it from there. This allows you to reuse this content without prior rendering. The presence of "cache.key" will trigger this feature. It is evaluated twice:

- Content is read from cache directly after the `stdWrapPreProcess` hook and before

"setContentToCurrent". If there is a cache entry for the given cache key, stdWrap processing will stop and the cached content will be returned. If no cache content is found for this key, the stdWrap processing continues as usual.

- Writing to cache happens at the end of rendering, directly before the stdWrapPostProcess hook is called and before the "debug\*" functions. The rendered content will be stored in the cache, if cache.key was set. The configuration options cache.tags and cache.lifetime allow to control the caching.

**Note:** This feature relies on the caching framework, which needs to be enabled for this feature to work. Otherwise content will not be cached, but rendered on every call.

Property:	Data type:	Description:	Default:
<b>key</b>	string /stdWrap	The cache identifier that is used to store the rendered content into the cache and to read it from there.  <b>Note:</b> Make sure to use a valid cache identifier. Also take care to choose a cache key that is accurate enough to distinguish different versions of the rendered content while being generic enough to stay efficient.	
<b>lifetime</b>	mixed /stdWrap	Lifetime of the content in cache. Allows you to determine the lifetime of the cached object independently of the lifetime of the cached version of the page on which it is used.  Possible values are any positive integer and the keywords "unlimited" and "default": <b>integer:</b> Lifetime in seconds. <b>"unlimited":</b> Cached content will not expire unless actively purged by id or by tag or if the complete cache is flushed. <b>"default":</b> The default cache lifetime as configured in config.cache_period is used.	default
<b>tags</b>	string /stdWrap	Can hold a comma-separated list of tags. These tags will be attached to the cached content into the cache_hash storage (not into cache_pages) and can be used to purge the cached content.	

[tsref:->cache]

#### Examples:

```
5 = TEXT
5 {
    cache.key = mycurrenttimestamp
    cache.tags = tag_a,tag_b,tag_c
    cache.lifetime = 3600
    data = date : U
    strftime = %H:%M:%S
}
```

In the above example the current time will be cached with the key "mycurrenttimestamp". This key is fixed and does not take the current page id into account. So if you add this to your TypoScript, the cObject will be cached and reused on all pages (showing you the same timestamp).

```
5 = TEXT
5 {
    cache.key = mycurrenttimestamp_{page:id}_{TSFE:sys_language_uid}
    cache.key.insertData = 1
}
```

Here a dynamic key is used. It takes the page id and the language uid into account making the object page and language specific.

# Setup

## Top-level objects

Property:	Data type:	Description:	Default:
<b>types</b>	readonly	Types (internal) type=99 reserved for plaintext display	
<b>resources</b>	readonly	Resources in list (internal)	
<b>sitetitle</b>	readonly	SiteTitle (internal)	
<b>config</b>	->CONFIG	Global configuration. These values are stored with cached pages which means they are also accessible when retrieving a cached page.	
<b>constants</b>	->CONSTANTS	Site-specific constants, eg. a general email-adresse. These constants may be substituted in the text throughout the pages. The substitution is done by parseFunc. (Option: constants=1)	
<b>FEData</b>	->FE_DATA	Here you can configure how data submitted from the front-end should be processed, which script and so on.	
<b>includeLibs</b>	<i>Array of strings</i>	With this you can include php-files with function libraries for use in your includescript in TYPO3. Please see the PAGE-object, which has the same property.	
<b>Other reserved TLO's:</b>  <b>plugin</b> <b>tt_*</b> <b>temp</b> <b>styles</b> <b>lib</b> <b>_GIFBUILDER</b>		These top-level object names are reserved. That means you can risk static_templates to use them: " <b>plugin</b> " is used for rendering of special content like boards, e-commerce solutions, guestbooks and so on. Normally set from static_templates. <i>Please see separate description below!</i> " <b>tt_*</b> ", eg tt_content (from "content (default)") is used to render content from tables. " <b>temp</b> " and " <b>styles</b> " are used for conde-libraries you can copy during parse-time, but they are not saved with the template in cache. <i>"temp" / "styles" are unset</i> before the template is cached! Therefore use these names to store temporary data. " <b>lib</b> " can be used for a "library" of code, you can reference in TypoScript (unlike "styles" which is unset)	
...	->PAGE	Start a new page.  <b>Example:</b> page = PAGE page.typeNum = 1  <b>Guidelines:</b> Good, general PAGE object names to use are such as: <i>page</i> for the main page with content <i>frameset</i> , <i>frameset2</i> for framesets. <i>top</i> , <i>left</i> , <i>menu</i> , <i>right</i> , <i>bottom</i> , <i>border</i> for top and menu frames etc. These are just recommendations. Especially the name 'page' for the content bearing page is very common.	
...	<i>(whatever)</i>	If a top-level object is not a PAGE-object it could be used as a temporary repository for setup. In this case you should use the "temp" or "styles" objects. "tt_..." is normally used to define the setup of content-records. Eg. "tt_content" would be used for the tt_content-table as default. See the "CONTENT"-cObject	

[tsref:(TLO)]

## The "plugin" TLO

This is used for extensions in TYPO3 set up as frontend plugins. Typically you can set configuration properties of the plugin here. Say you have an extension with the key "myext" and it has a frontend plugin named "tx\_myext\_pi1" then you would find the TypoScript configuration at the position "plugin.tx\_myext\_pi1" in the object tree!

Most plugins are USER or USER\_INT objects which means that they have at least 1 or 2 reserved properties. Furthermore this table outlines some other default properties. Generally system properties are prefixed with an underscore:

Property:	Data type:	Description:	Default:
<i>userFunc</i>		<i>Property setting up the USER / USER_INT object of the plugin</i>	
<i>includeLibs</i>		<i>Property setting up the USER / USER_INT object of the plugin</i>	
<b>_CSS_DEFAULT_STYLE</b>	string	Use this to have some default CSS styles inserted in the header section of the document. Most likely this will provide a default acceptable display from the plugin, but should ideally be cleared and moved to an external stylesheet. This value is for all plugins read by the pagegen script when making the header of the document.	
<b>_DEFAULT_PI_VARS.</b> <b>[piVar-key]</b>	string	Allows you to set default values of the piVars array which most plugins are using (and should use) for data exchange with themselves. This works only if the plugin calls \$this->pi_setPiVarDefaults().	
<b>_LOCAL_LANG.</b> <b>[lang-key].</b> <b>[label-key]</b>	string	Can be used to override the default locallang labels for the plugin.	

[tsref:plugin]

## "CONFIG"

In typo3/sysexst/cms/tslib/ this is known as \$GLOBALS['TSFE']->config['config'], thus the property "debug" below is accessible as \$GLOBALS['TSFE']->config['config']['debug'].

Property:	Data type:	Description:	Default:
<b>defaultGetVars</b>	array	<p>Allows to set default values for GET parameters. Default value is taken only if the GET parameter isn't defined. Array notation is done with dots, e.g.: test[var1] will be written as text.var1</p> <p><b>Example:</b></p> <pre>config.defaultgetVars {     test.var1.var2.p3 = 15     L = 3 }</pre>	
<b>linkVars</b>	list	<p>HTTP_GET_VARS, which should be passed on with links in TYPO3. This is compiled into a string stored in \$GLOBALS['TSFE']-&gt;linkVars</p> <p>The values are rawurlencoded in PHP.</p> <p>You can specify a range of valid values by appending a () after each value. If this range does not match, the variable won't be appended to links. This is very important to prevent that the cache system gets flooded with forged values.</p> <p>The range may contain one of these values:</p> <ul style="list-style-type: none"> <li>• <b>[a]-[b]</b> - A range of allowed integer values</li> <li>• <b>int</b> - Only integer values are allowed</li> <li>• <b>[a][b][c]</b> - A list of allowed strings (whitespaces will be removed)</li> <li>• <b>/[regex]/</b> - Match against a regular expression (PCRE style)</li> </ul> <p><b>Example:</b></p> <pre>config.linkVars = L, print</pre> <p>This will add "&amp;L=[L-value]&amp;print=[print-value]" to all links in TYPO3.</p> <pre>config.linkVars = L(1-3), print</pre> <p>Same as above, but "&amp;L=[L-value]" will only be added if the current value is 1, 2 or 3.</p> <p><b>Note:</b> Do <b>not</b> include the "type" parameter in the linkVars list, as this can result in unexpected behavior.</p>	
<b>uniqueLinkVars</b>	boolean	It might happen that TYPO3 generates links with the same parameter twice or more. This is no problem because only the last parameter is used, thus the problem is just a cosmetic one.	1
<b>MP_defaults</b>	string	<p>Allows you to set a list of page id numbers which will always have a certain "&amp;MP=..." parameter added.</p> <p><b>Syntax:</b> [id],[id],... : [MP-var]   [id],[id],... : [MP-var]   ...</p> <p><b>Example:</b></p> <pre>config.MP_defaults = 36,37,48 : 2-207</pre> <p>This will by default add "&amp;MP=2-207" to all links pointing to pages 36,37 and 48</p>	



Property:	Data type:	Description:	Default:
<b>MP_mapRootPoints</b>	list of PIDs/string	Defines a list of ID numbers from which the MP-vars are automatically calculated for the branch. The result is used just like MP_defaults are used to find MP-vars if none has been specified prior to the call to t3lib_tstemplate::linkData(). You can specify "root" as a special keyword in the list of IDs and that will create a map-tree for the whole site (but this may be VERY processing intensive if there are many pages!). The order of IDs specified may have a significance; Any ID in a branch which is processed already (by a previous ID root point) will not be processed again.	
<b>MP_disableTypolinkClosestMPvalue</b>	boolean	If set, the typolink function will not try to find the closest MP value for the id.	
<b>renderCharset</b>	string	Charset used for the internal rendering of the page content. It is highly recommended that this value is the same as the charset of the content coming from the main data source (eg. the database). Thus you don't need to do any other conversion. All strings from locallang files and locale strings are (and should be) converted to "renderCharset" during rendering.  If you need another output charset than the render charset, see "metaCharset" below.  Until TYPO3 4.7 you can set \$TYPO3_CONF_VARS['BE']['forceCharset']. If you do, its value is used for "renderCharset" by default. It is highly recommended to use \$TYPO3_CONF_VARS['BE']['forceCharset'] = "utf-8" for multilingual websites in TYPO3. If you set this, you don't have to worry about renderCharset and metaCharset - the same charset is used in the whole system.  <b>Note:</b> In TYPO3 4.7 \$TYPO3_CONF_VARS['BE']['forceCharset'] has been removed. Since this version TYPO3 internally always uses UTF-8 by default.	Until TYPO3 4.7: The value of \$TYPO3_CONF_VARS['BE']['forceCharset'] if set, otherwise "iso-8859-1" Since TYPO3 4.7: "utf-8"
<b>metaCharset</b>	string	Charset used for the output document. For example in the meta tag: <code>&lt;meta charset=... /&gt;</code>  It is used for a) HTML meta tag, b) HTTP header (unless disabled with .disableCharsetHeader) and c) xhtml prologues (if available).  If renderCharset and metaCharset are different, the output content is automatically converted to metaCharset before output and likewise are values posted back to the page converted from metaCharset to renderCharset for internal processing. This conversion takes time of course so there is another good reason to use the same charset for both.	value of ".renderCharset"
<b>disableCharsetHeader</b>	boolean	By default a header "content-type:text/html; charset..." is sent. This option will disable that.	

Property:	Data type:	Description:	Default:
<b>sendCacheHeaders</b>	boolean	<p>If set, TYPO3 will output cache-control headers to the client based mainly on whether the page was cached internally. This feature allows client browsers and/or reverse proxies to take load off TYPO3 websites.</p> <p>The conditions for allowing client caching are:</p> <ul style="list-style-type: none"> <li>• page was cached</li> <li>• No *_INT or *_EXT objects were on the page (eg. USER_INT)</li> <li>• No frontend user is logged in</li> <li>• No backend user is logged in</li> </ul> <p>If these conditions are met, the headers sent are:</p> <ul style="list-style-type: none"> <li>• Last-Modified [SYS_LASTCHANGED of page id]</li> <li>• Expires [expire time of page cache]</li> <li>• Etag [md5 of content]</li> <li>• Cache-Control: max-age: [seconds til expiretime]</li> <li>• Pragma: public</li> </ul> <p>In case caching is not allowed, these headers are sent to avoid client caching:</p> <ul style="list-style-type: none"> <li>• Cache-Control: private</li> </ul> <p>Notice that enabling the browser caches means you have to consider how log files are written. Because when a page is cached on the client it will not invoke a request to the webserver, thus not writing the request to the log. There should be ways to circumvent these problems but they are outside the domain of TYPO3 in any case.</p> <p><b>Tip:</b> Enabling cache-control headers might confuse editors seeing old content served from the browser cache. "Shift-Reload" will bypass both browser- and reverse-proxy caches and even make TYPO3 regenerate the page. Teach them that trick!</p> <p>Thanks to Ole Tange, <a href="http://www.forbrug.dk">www.forbrug.dk</a> for co-authoring this feature.</p>	
<b>sendCacheHeaders_onlyWhenLoginDeniedInBranch</b>	boolean	<p>If this is set, then cache-control headers allowing client caching is sent only if user-logins are disabled for the branch. This feature makes it easier to manage client caching on sites where you have a mixture of static pages and dynamic sections with user logins.</p> <p>The background problem is this: In TYPO3 the same URL can show different content depending on whether a user is logged in or not. If a user is logged in, cache-headers will never allow client caching. But if the same URL was visited without a login prior to the login (allowing caching) the user will still see the page from cache when logged in (and so thinks he is not logged in anyway)! The only general way to prevent this is to have a different URL for pages when users are logged in (which the extension "realurl" can accomplish).</p> <p>Another way to solve the problem is using this option in combination with disabling and enabling logins in various sections of the site. In the page records ("Advanced" page types) you can disable frontend user logins for branches of the page tree. Since many sites only needs the login in a certain branch of the page tree, disabling it in all other branches makes it much easier to use cache-headers in combination with logins; Cache-headers should simply be sent when logins are not allowed and never be send when logins are allowed! Then there will never be problems with logins and same-URLs.</p>	

Property:	Data type:	Description:	Default:
<b>additionalHeaders</b>	strings divided by " "	<p>This property can be used to define additional HTTP headers. Separate each header with a vertical line " ".</p> <p><b>Examples:</b>  Content-type: text/vnd.wap.wml  (this will send a content-header for a WAP-site)</p> <p>Content-type: image/gif   Expires: Mon, 25 Jul 2011 05:00:00 GMT  (this will send a content-header for a GIF-file and an Expires header)</p> <p>Location: www.typo3.com  (This redirects the page to <a href="http://www.typo3.com">www.typo3.com</a>)</p> <p>By default TYPO3 sends a "Content-Type" header with the defined encoding, unless this is disabled using <code>config.disableCharsetHeader</code> (see above). It then sends cache headers, if configured (see above). Then come the additional headers, plus finally a "Content-Length" header, if enabled (see below).</p>	
<b>enableContentLengthHeader</b>	boolean	<p>If set, a header "content-length: [bytes of content]" is sent.</p> <p>If a PHP_SCRIPT_EXT object is detected on the page or if the Backend user is logged in, this is disabled. The reason is that the content length header cannot include the length of these objects and the content-length will cut of the length of the document in some browsers.</p>	
<b>doctype</b>	string	<p>If set, then a document type declaration (and an XML prologue) will be generated. The value can either be a complete doctype or one of the following keywords:</p> <p>"<b>xhtml_trans</b>" for the XHTML 1.0 Transitional doctype.  "<b>xhtml_frames</b>" for the XHTML 1.0 Frameset doctype.  "<b>xhtml_strict</b>" for the XHTML 1.0 Strict doctype.  "<b>xhtml_basic</b>" for the XHTML basic doctype.  "<b>xhtml_11</b>" for the XHTML 1.1 doctype.  "<b>xhtml+rdfa_10</b>" for the XHTML+RDFa 1.0 doctype.  "<b>xhtml_2</b>" for the XHTML 2 doctype.  "<b>html5</b>" for the HTML5 doctype.  "<b>none</b>" for NO doctype at all.</p> <p><b>Note:</b> In TYPO3 4.4 the keyword for HTML5 was "html_5". This spelling was deprecated since TYPO3 4.5 and has been removed in TYPO3 4.7.</p> <p>Note that the keywords also change the way TYPO3 generates some of the XHTML tags to ensure valid XML. If you set doctype to a string, then you must also set <code>config.xhtmlDoctype</code> (see below).</p> <p>See "<code>config.htmlTag_setParams</code>" and "<code>config.htmlTag_langKey</code>" for more details on the effect on the html tag.</p> <p>Default is the HTML 4 Transitional doctype:</p> <pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"&gt;</pre>	

Property:	Data type:	Description:	Default:
<b>doctypeSwitch</b>	boolean / string	<p>If set, the order of <code>&lt;?xml...&gt;</code> and <code>&lt;!DOCTYPE...&gt;</code> will be reversed. This is needed for MSIE to be standards compliant with XHTML.</p> <p><b>Background:</b> By default TYPO3 outputs the XML/DOCTYPE in compliance with the standards of XHTML. However a browser like MSIE will still run in "quirks-mode" unless the <code>&lt;?xml&gt;</code> and <code>&lt;DOCTYPE&gt;</code> tags are ordered opposite. But this breaks CSS validation... With this option designers can decide for themselves what they want then.</p> <p>If you want to check the compatibility-mode of your webbrowser you can do so with a simple JavaScript that can be inserted on a TYPO3 page like this:</p> <pre>page.headerData.1 = TEXT page.headerData.1.value = &lt;script&gt;alert(document.compatMode);&lt;/script&gt;</pre> <p>If your browser has detected the DOCTYPE correctly it will report "CSSCompat" If you are not running in compliance mode you will get some other message. MSIE will report "BackCompat" for instance - this means it runs in quirks-mode, supporting all the old "browser-bugs".</p>	
<b>xhtmlDoctype</b>	string	<p>Sets the document type for the XHTML version of the generated page.</p> <p>If <code>config.doctype</code> is set to a string then <code>config.xhtmlDoctype</code> must be set to one of these keywords:</p> <p>"<b>xhtml_trans</b>" for XHTML 1.0 Transitional doctype.  "<b>xhtml_frames</b>" for XHTML 1.0 Frameset doctype.  "<b>xhtml_strict</b>" for XHTML 1.0 Strict doctype.  "<b>xhtml_basic</b>" for XHTML basic doctype.  "<b>xhtml_11</b>" for XHTML 1.1 doctype.  "<b>xhtml_2</b>" for XHTML 2 doctype.</p> <p>This is an example to use MathML 2.0 in an XHTML 1.1 document:</p> <pre>config.doctype ( &lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN" "http://www.w3.org/Math/DTD/mathml2/xhtml- math11-f.dtd"&gt; ) config.xhtmlDoctype = xhtml_11</pre> <p>Default: same as <code>config.doctype</code> if set to a keyword</p>	

Property:	Data type:	Description:	Default:
<b>xmlprologue</b>	string	<p>If empty (not set) then the default XML 1.0 prologue is set, when the doctype is set to a known keyword (eg xhtml_11):</p> <pre>&lt;?xml version="1.0" encoding="[config.renderCharset]"&gt;</pre> <p>If set to one of the know keywords then a standard prologue will be set:  <b>"xml_10"</b> XML 1.0 prologue (see above)  <b>"xml_11"</b> XML 1.1 prologue</p> <p>If <b>"none"</b> then the default XML prologue is not set. Any other string is used as the XML prologue itself.</p>	
<b>htmlTag_setParams</b>	string	<p>Sets the attributes for the &lt;html&gt; tag on the page. If you set "config.doctype" to a keyword enabling XHTML then some attributes are already set. This property allows you to override any preset attributes with your own content if needed.</p> <p><b>Special:</b> If you set it to "none" then no attributes will be set at any event.</p> <p><b>Example:</b></p> <pre>config.htmlTag_setParams = xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"</pre>	
<b>htmlTag_stdWrap</b>	->stdWrap	(Since TYPO3 4.7) Modify the whole <html> tag with stdWrap functionality. This can be used to extend or override this tag.	
<b>namespaces</b>	array of strings	<p>This property enables you to add xml namespaces (xmlns) to the &lt;html&gt; tag. This is especially useful if you want to add RDFa or microformats to your html.</p> <p><b>Example:</b></p> <pre>config.namespaces.dc = http://purl.org/dc/elements/1.1/ config.namespaces.foaf = http://xmlns.com/foaf/0.1/</pre> <p>This configuration will result in an &lt;html&gt; tag like</p> <pre>&lt;html xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:foaf="http://xmlns.com/foaf/0.1/"&gt;</pre>	
<b>htmlTag_langKey</b>	string	<p>Allows you to set the language value for the attributes "xml:lang" and "lang" in the &lt;html&gt; tag (when using "config.doctype = xhtml*").</p> <p>The values must follow the format specified in IETF RFC 3066</p> <p><b>Example:</b></p> <pre>config.htmlTag_langKey = en-US</pre>	en
<b>htmlTag_dir</b>	string	<p>Sets text direction for whole document (useful for display of Arabic, Hebrew pages).</p> <p>Basically the value becomes the attribute value of "dir" for the &lt;html&gt; tag.</p> <p><b>Values:</b>  rtl = Right-To-Left (for Arabic / Hebrew)  ltr = Left-To-Right (Default for other languages)</p> <p><b>Example:</b></p> <pre>config.htmlTag_dir = rtl</pre>	

Property:	Data type:	Description:	Default:
<b>disableImgBorderAttr</b>	boolean	Returns the 'border' attribute for an <img> tag only if the doctype is not xhtml_strict, xhtml_11 or xhtml_2 or if the config parameter 'disableImgBorderAttr' is not set	
<b>ATagParams</b>	<A>- params	Additional parameters to all links in TYPO3 (excluding menu-links)  <b>Example:</b> To blur links, insert: onFocus="blurLink(this)"	
<b>setJS_openPic</b>	boolean	If set, the openPic JavaScript function is forced to be included	
<b>setJS_mouseOver</b>	boolean	If set, the over() and out() JavaScript functions are forced to be included	
<b>removeDefaultJS</b>	boolean / string	If set, the default JavaScript in the header will be removed. The default JavaScript is the blurLink function and browser detection variables.  <b>Special case:</b> if the value is "external" then the default JavaScript is written to a temporary file and included from that file. See "inlineStyle2TempFile" below.  Depends on the compatibility mode (see Tools>Install>Update wizard): compatibility mode < 4.0: 0 compatibility mode >= 4.0: 1  <b>Examples:</b> config.removeDefaultJS = external config.removeDefaultJS = 1	
<b>removeDefaultCss</b>	boolean	(Since TYPO3 4.6) Remove CSS generated by _CSS_DEFAULT_STYLE configuration of extensions.	
<b>minifyJS</b>	boolean	If set, inline or externalized (see removeDefaultJS above) JavaScript will be minified. Minification will remove all excess space and will cause faster page loading. Together with removeDefaultJS = external it will significantly lower web site traffic.  The default value depends on the compatibility mode (see Tools>Install>Update wizard): compatibility mode < 4.0: 0 compatibility mode >= 4.0: 1  <b>Example:</b> config.minifyJS = 1  <b>Note:</b> JavaScript in external files in the FE will only be minified, if a compression handler is registered using \$GLOBALS['TYPO3_CONF_VARS']['FE']['jsCompressHandler'].  <b>Example:</b> \$GLOBALS['TYPO3_CONF_VARS']['FE'] ['jsCompressHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_jsCompressHandler.php' :tx_myext_jsCompressHandler->minifyJs';  <b>Note:</b> This property is deprecated and will be removed with TYPO3 6.0! Use config.compressJs instead.	

Property:	Data type:	Description:	Default:
<b>compressJs</b>	boolean	<p>(Since TYPO3 4.6) Enabling this option together with \$TYPO3_CONF_VARS['FE']['compressionLevel'] in the Install Tool delivers Frontend JavaScript files using GZIP compression.</p> <p>This can significantly reduce file sizes of linked JavaScript files and thus decrease loading times.</p> <p>Please note that this requires .htaccess to be adjusted, as otherwise the files will not be readable by the user agent. Please see the description of \$TYPO3_CONF_VARS['FE']['compressionLevel'] in the Install Tool.</p> <p><b>Example:</b></p> <pre>config.compressJs = 1</pre> <p><b>Note:</b> TYPO3 comes with a built-in compression handler, but you can also register your own one using \$GLOBALS['TYPO3_CONF_VARS']['FE']['jsCompressHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['jsCompressHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_jsCompressHandler.php :tx_myext_jsCompressHandler-&gt;compressJs';</pre>	
<b>minifyCSS</b>	boolean	<p>Setting this option will activate CSS minification.</p> <p><b>Example:</b></p> <pre>config.minifyCSS = 1</pre> <p><b>Note:</b> CSS in external files in the FE will only be minified, if a compression handler is registered using \$GLOBALS['TYPO3_CONF_VARS']['FE']['cssCompressHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['cssCompressHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_cssCompressHandler.php :tx_myext_cssCompressHandler-&gt;minifyCss';</pre> <p><b>Note:</b> This property is deprecated and will be removed with TYPO3 6.0! Use config.compressCss instead.</p>	

Property:	Data type:	Description:	Default:
<b>compressCss</b>	boolean	<p>(Since TYPO3 4.6) If set, CSS files will be minified and compressed.</p> <p>Minification will remove all excess space. The more significant compression step (using gzip compression) requires \$TYPO3_CONF_VARS['FE']['compressionLevel'] to be enabled in the Install Tool. For this to work you also need to activate the gzip-related compressionLevel options in .htaccess, as otherwise the compressed files will not be readable by the user agent.</p> <p><b>Example:</b></p> <pre>config.compressCss = 1</pre> <p><b>Note:</b> TYPO3 comes with a built-in compression handler, but you can also register your own one using \$GLOBALS['TYPO3_CONF_VARS']['FE']['cssCompressHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['cssCompressHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_cssCompressHandler.ph p:tx_myext_cssCompressHandler-&gt;compressCss';</pre>	
<b>concatenateJsAndCss</b>	boolean	<p>Setting config.concatenateJsAndCss bundles JS and CSS files in the FE.</p> <p><b>Example:</b></p> <pre>config.concatenateJsAndCss = 1</pre> <p><b>Note:</b> There are no default concatenation handlers, which could do the concatenation. A custom concatenation handler must be provided and registered using \$GLOBALS['TYPO3_CONF_VARS']['FE']['concatenateHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['concatenateHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_concatenateHandler.ph p:tx_myext_concatenateHandler- &gt;concatenateFiles';</pre> <p><b>Note:</b> This property is deprecated and will be removed with TYPO3 6.0! Use config.concatenateJs and config.concatenateCss instead.</p>	0
<b>concatenateJs</b>	boolean	<p>(Since TYPO3 4.6) Setting config.concatenateJs merges JavaScript files referenced in the Frontend together.</p> <p><b>Example:</b></p> <pre>config.concatenateJs = 1</pre> <p><b>Note:</b> TYPO3 comes with a built-in concatenation handler, but you can also register your own one using \$GLOBALS['TYPO3_CONF_VARS']['FE']['jsConcatenateHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['jsConcatenateHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_jsConcatenateHandler. php:tx_myext_jsConcatenateHandler- &gt;concatenateJs';</pre>	



Property:	Data type:	Description:	Default:
<b>concatenateCss</b>	boolean	<p>(Since TYPO3 4.6) Setting config.concatenateCss merges Stylesheet files referenced in the Frontend together.</p> <p><b>Example:</b></p> <pre>config.concatenateCss = 1</pre> <p><b>Note:</b> TYPO3 comes with a built-in concatenation handler, but you can also register your own one using \$GLOBALS['TYPO3_CONF_VARS']['FE']['cssConcatenateHandler'].</p> <p><b>Example:</b></p> <pre>\$GLOBALS['TYPO3_CONF_VARS']['FE'] ['cssConcatenateHandler'] = t3lib_extMgm::extPath(\$_EXTKEY) . 'Classes/class.tx_myext_cssConcatenateHandler- .php:tx_myext_cssConcatenateHandler- &gt;concatenateCss';</pre>	
<b>inlineStyle2TempFile</b>	boolean	<p>If set, the inline styles TYPO3 controls in the core are written to a file, typo3temp/stylesheet_[hashstring].css, and the header will only contain the link to the stylesheet.</p> <p>The file hash is based solely on the content of the styles.</p> <p>Depends on the compatibility mode (see Tools&gt;Install&gt;Update wizard):</p> <p><i>compatibility mode &lt; 4.0: 0</i></p> <p><i>compatibility mode &gt;= 4.0: 1</i></p> <p><b>Example:</b></p> <pre>config.inlineStyle2TempFile = 1</pre>	
<b>meaningfulTempFilePrefix</b>	integer	<p>If &gt; 0 TYPO3 will try to create a meaningful prefix of the given length for the temporary image files.</p> <p>This works with GIFBUILDER files (using content from the GIFBUILDER TEXT objects as a base for the prefix), menus (using the title of the menu item) and scaled images (using the original filename base).</p>	
<b>ftu</b>	boolean	<p>If set, the "&amp;ftu=..." GET-fallback identification is inserted. "&amp;ftu=[hash]" is always inserted in the links on the first page a user hits. If it turns out in the next hit that the user has cookies enabled, this variable is not set anymore as the cookies does the job. If no cookies is accepted the "ftu" remains set for all links on the site and thereby we can still track the user.</p> <p><b>You should not set this feature if grabber-spiders like Teleport are going to grab your site!</b></p> <p><b>You should not set this feature if you want search-engines to index your site (in conjunction with the simulateStaticDocuments feature!)</b></p> <p>You can also ignore this feature if you're certain, website users will use cookies.</p> <p>"ftu" means fe_typo_user ("fe" is "frontend").</p>	false
<b>mainScript</b>	string	<p>This lets you specify an alternative "mainScript" which is the document that TYPO3 expects to be the default doc. This is used in form-tags and other places where TYPO3 needs to refer directly to the main-script of the application</p>	index.php

Property:	Data type:	Description:	Default:
<b>pageGenScript</b>	resource	<p>Alternative page generation script for applications using index_ts.php for initialization, caching, stating and so on. This script is included in the global scope of index_ts.php-script and thus you may include libraries here. Always use include_once for libraries.</p> <p>Remember not to output anything from such an included script. <b>All content must be set into \$TSFE-&gt;content.</b> Take a look at typo3/sysex/cms/tslib/pagegen.php</p> <p><b>NOTE:</b> This option is ignored if</p> <pre>\$TYPO3_CONF_VARS['FE']['noPHPscriptInclude'] = 1;</pre> <p>is set in localconf.php.</p>	typo3/sysex/cms/tslib/pagegen.php
<b>debug</b>	boolean	If set any debug-information in the TypoScript code is output. Currently this applies only to the menu-objects	
<b>message_page_is_being_generated</b>	string	<p>Alternative HTML message that appears if a page is being generated.</p> <p>Normally when a page is being generated a temporary copy is stored in the cache-table with an expire-time of 30 seconds.</p> <p>It is possible to use some keywords that are replaced with the corresponding values. Possible keywords are: ###TITLE###, ###REQUEST_URI###</p>	
<b>message_preview</b>	string	Alternative message in HTML that appears when the preview function is active!	
<b>message_preview_workspace</b>	string	<p>Alternative message in HTML that appears when the preview function is active in a draft workspace. You can use sprintf() placeholders for Workspace title (first) and number (second).</p> <p><b>Examples:</b></p> <pre>config.message_preview_workspace = &lt;div class="previewbox"&gt;Displaying workspace named "%s" (number %s)!&lt;/div&gt; config.message_preview_workspace = &lt;div class="previewbox"&gt;Displaying workspace number %2\$s named "%1\$s"!&lt;/div&gt;</pre>	
<b>disablePreviewNotification</b>	boolean	Disables the "preview" notification box completely.	0
<b>locale_all</b>	string	<p>PHP: setlocale("LC_ALL", [value]);</p> <p>value-examples: deutsch, de_DE, danish, portuguese, spanish, french, norwegian, italian. See www.php.net for other value.</p> <p>Also on linux, look at /usr/share/locale/</p> <p>TSFE-&gt;localeCharset is intelligently set to the assumed charset of the locale strings. This is used in stdWrap.strftime to convert locale strings to the renderCharset of the frontend.</p> <p><b>Example:</b></p> <p>This will render dates in danish made with stdWrap/strftime:</p> <pre>locale_all = danish locale_all = da_DK</pre>	
<b>sword_standAlone</b>	boolean	Used by the parseFunc-substitution of search Words (sword): If set, the words MUST be surrounded by whitespace in order to be marked up.	
<b>sword_noMixedCase</b>	boolean	Used by the parseFunc-substitution of search Words (sword): If set, the words MUST be the exact same case as the search word was.	
<b>intTarget</b>	target	Default internal target. Used by typolink if no target is set	
<b>extTarget</b>	target	Default external target. Used by typolink if no extTarget is set	_top
<b>fileTarget</b>	target	Default file link target. Used by typolink if no fileTarget is set.	

Property:	Data type:	Description:	Default:
<b>spamProtectEmailAddresses</b>	"ascii" / -10 to 10	<p>If set, then all email addresses in typolinks will be encrypted so spam bots cannot detect them.</p> <p>If you set this value to a number, then the encryption is simply an offset of character values. If you set this value to "-2" then all characters will have their ASCII value offset by "-2". To make this possible, a little JavaScript code is added to every generated web page! (It is recommended to set the value in the range from -5 to 1 since setting it to <math>\geq 2</math> means a "z" is converted to "I" which is a special character in TYPO3 tables syntax - and that might confuse columns in tables. Now hardcoded range)</p> <p>Alternatively you can set this value to the keyword "ascii". This way every character of the "mailto:" address will be translated to a Unicode HTML notation. Have a look at the example to see how this works.</p> <p>Example: mailto:a@b.c will be converted to mailto:&amp;#97;&amp;#64;&amp;#98;&amp;#46;&amp;#99; The big advantage of this method is that it doesn't need any JavaScript!</p>	
<b>spamProtectEmailAddresses_atSubst</b>	string	Substitute label for the at-sign (@).	(at)
<b>spamProtectEmailAddresses_lastDotSubst</b>	string	Substitute label for the last dot in the email address. Example: (dot)	Default: . ( $\leq$ just a simple dot)
<b>forceTypeValue</b>	int	Force the &type value of all TYPO3 generated links to a specific value (except if overruled by local forceTypeValue values). Useful if you run a template with special content at - say &type=95 - but still wants to keep your targets neutral. Then you set your targets to blank and this value to the type value you wish.	
<b>frameReloadIfNotInFrameset</b>	boolean	If set, then the current page will check if the page object name (e.g. "page" or "frameset") exists as "parent.[name]" (e.g. "parent.page") and if not the page will be reloaded in top frame. This secures that links from search engines to pages inside a frameset will load the frameset. Works only with type-values different from zero.	
<b>jumpurl_enable</b>	boolean	jumpUrl is a concept where external links are redirected from the index_ts.php script, which first logs the URL. This feature is only interesting if "config.sys_stat" is used.	0
<b>jumpurl_mailto_disable</b>	boolean	Disables the use of jumpUrl when linking to email-adresses.	0
<b>compensateFieldWidth</b>	double	<p>this floating point value will be used by the FORMS cObject to compensate the length of the form fields text and input. This feature is useful, if the page-option "smallFormFields" is set. In that case Netscape renders form fields much longer than IE. If you want the two browsers to display the same size form fields, use a value of approx "0.6" for netscape-browsers.</p> <p><b>Example:</b>  <pre>[browser = netscape]     config.compensateFieldWidth = 0.6 [global]</pre> </p> <p>This option may be overridden in the FORMS-cObject.</p>	

Property:	Data type:	Description:	Default:
<b>includeLibrary</b>	resource	This includes a PHP file.	
<b>incT3Lib_htmlmail</b>	boolean	Include t3lib/class.t3lib_htmlmail.php	
<b>lockFilePath</b>	string	This is used to lock paths to be "inside" this path. Used by "filelist" in stdWrap	fileadmin/
<b>noScaleUp</b>	boolean	Normally images are scaled to the size specified via TypoScript. This also forces small images to be scaled to a larger size. This is not always a good thing. If this property is set, images are <b>not</b> allowed to be scaled up in size. This parameter clears the \$this->mayScaleUp var of the class t3lib_stdgraphics (often "gifbuilder").	
<b>USERNAME_substToken</b>	string	The is the token used on the page, which should be substituted with the current username IF a front-end user is logged in! If no login, the substitution will not happen.	<!-- ###USERNAM E###-->
<b>USERUID_substToken</b>	string	The is the token used on the page, which should be substituted with the current users UID IF a front-end user is logged in! If no login, the substitution will not happen. This value has no default value and only if you specify a value for this token will a substitution process take place.	
<b>cache_period</b>	int, seconds	The number of second a page may remain in cache. This value is overridden by the value set in the page-record (field="cache_timeout") if this value is greater than zero.	86400 (=24H)
<b>cache</b>	array	<p>(Since TYPO3 4.6) Determine the maximum cache lifetime of a page.</p> <p>The maximum cache lifetime of a page can not only be determined by the start and stop times of content elements on the page itself, but also by arbitrary records on any other page. However, the page has to be configured so that TYPO3 knows the start and stop times of which records to include. Otherwise, the cache entry will be used although a start/stop date already passed by.</p> <p>To include records of type &lt;tablename&gt; on page &lt;pid&gt; into the cache lifetime calculation of page &lt;page-id&gt;, add the following TypoScript: config.cache.&lt;page-id&gt; = &lt;tablename&gt;:&lt;pid&gt;</p> <p>Multiple record sources can be added as comma-separated list, see the examples.</p> <p>You can use the keyword "all" instead of a &lt;page-id&gt; to consider records for the cache lifetime of all pages.</p> <p><b>Examples:</b></p> <pre>config.cache.10 = fe_users:2</pre> <p>This includes the fe_users records on page 2 in the cache lifetime calculation for page 10.</p> <pre>config.cache.10 = fe_users:2,tt_news:11</pre> <p>This includes records from multiple sources, namely the fe_users records on page 2 and the tt_news records on page 11.</p> <pre>config.cache.all = fe_users:2</pre> <p>Consider the fe_user records on page 2 for the cache lifetime of all pages.</p>	
<b>cache_clearAtMidnight</b>	boolean	With this setting the cache always expires at midnight of the day, the page is scheduled to expire.	false
<b>no_cache</b>	boolean	If this is set to true, the page will not be cached. If set to false, it's ignored. Other parameters may have set it to true of other reasons.	-

Property:	Data type:	Description:	Default:
<b>disableAllHeaderCode</b>	boolean	If this is set, none of the features of the PAGE-object is processed and the content of the page will be the result of the cObject array (1,2,3,4...) of the PAGE-object. This means that the result of the cObject should include everything from the <HTML> ... to the </HTML> tag! Use this feature in templates supplying other content-types than HTML. That could be an image or a WAP-page!	false
<b>disablePageExternalUrl</b>	boolean	If set, pages with doktype "External Url" will not trigger jumpUrl in TSFE. This may help you to have external urls open inside you framesets.	
<b>stat</b>	boolean	Enable stat logging at all.	true
<b>stat_typeNumList</b>	int/list	List of pagetypes that should be registered in the statistics table, sys_stat. If no types are listed, all types are logged. Default is "0,1" which normally logs all hits on framesets and hits on content keeping pages. Of course this depends on the template design.	0,1
<b>stat_excludeBEuserHits</b>	boolean	If set a page hit is not logged if a user is logged in into TYPO3.	false
<b>stat_excludeIPList</b>	list of strings	If the REMOTE_ADDR is in the list of IP-addresses, it's also not logged. Can use wildcard, e.g. "192.168.1.*"	
<b>stat_mysql</b>	boolean	Enable logging to the MySQL table sys_stat.	false
<b>stat_apache</b>	boolean	Enable logging to the log file "stat_apache_logfile"	false
<b>stat_apache_logfile</b>	filename	This defines the name of the log file where TYPO3 writes an Apache-style logfile to. The location of the directory is defined by \$TYPO3_CONF_VARS['FE']['logfile_dir'] which must exist and be writable. It can be relative (to PATH_site) or absolute, but in any case it must be within the regular allowed paths of TYPO3 (meaning for absolute paths that it must be within the "lockRootPath" set up in \$TYPO3_CONF_VARS).  It is also possible to use date markers in the filename as they are provided by the PHP function strftime(). This will enable a natural rotation of the log files.  <b>Example:</b> <pre>config.stat_apache_logfile = typo3_%Y%m%d.log</pre> This will create daily log files (e.g. typo3_20060321.log).	
<b>stat_apache_pagename</b>	string	The "pagename" simulated for apache. Default: "[path][title]--[uid].html" Codes: [title] = inserts title, no special characters and shortened to 30 chars. [uid] = the id [alias] = any alias [type] = the type (typeNum) [path] = the path of the page [request_uri] = inserts the REQUEST_URI server value (useful with RealUrl for example)	
<b>stat_apache_notExtended</b>	boolean	If true the log file is NOT written in Apache extended format	
<b>stat_apache_noHost</b>	boolean	If true the HTTP_HOST is - if available - NOT inserted instead of the IP-address	

Property:	Data type:	Description:	Default:
<b>stat_apache_niceTitle</b>	boolean / string	<p>If set, the URL will be transliterated from the renderCharset to ASCII (e.g. ä =&gt; ae, à =&gt; a, &amp;#945; "alpha" =&gt; a), which yields nice and readable page titles in the log. All non-ASCII characters that cannot be converted will be changed to underscores.</p> <p>If set to "utf-8", the page title will be converted to UTF-8 which results in even more readable titles, if your log analyzing software supports it.</p>	
<b>stat_apache_noRoot</b>	boolean	If set, the root part (level 0) of the path will be removed from the path. This makes a shorter name in case you have only a redundant part like "home" or "my site".	
<b>stat_titleLen</b>	int 1-100	The length of the page names in the path written to log file/database	20
<b>stat_pageLen</b>	int 1-100	The length of the page name (at the end of the path) written to the log file/database.	30
<b>stat_IP_anonymize</b>	boolean	(Since TYPO3 4.7) Set to 1 to activate anonymized logging. Setting this to 1 will log an empty hostname and will enable anonymization of IP addresses.	0
<b>stat_IP_anonymize_mask_ipv4</b>	int	<p>(Since TYPO3 4.7) Prefix-mask 0..32 to use for anonymisation of IP addresses (IPv4). Only used, if stat_IP_anonymize is set to 1.</p> <p>Recommendation for Germany:</p> <pre>config.stat_IP_anonymize_ipv4 = 24</pre>	24
<b>stat_IP_anonymize_mask_ipv6</b>	int	<p>(Since TYPO3 4.7) Prefix-mask 0..128 to use for anonymisation of IP addresses (IPv6). Only used, if stat_IP_anonymize is set to 1.</p> <p>Recommendation for Germany:</p> <pre>config.stat_IP_anonymize_ipv6 = 64</pre>	64
<b>stat_logUser</b>	boolean	(Since TYPO3 4.7) Configure whether to log the username of the Frontend user, if the user is logged in in the FE currently. Setting this to 0 allows to anonymize the username.	1

Property:	Data type:	Description:	Default:
<b>simulateStaticDocuments</b>	boolean / string	<p>If set TYPO3 makes all links in another way than usual. This can be used with <b>Apache compiled with mod_rewrite and configured in httpd.conf for use of this in the ".htaccess"-files.</b></p> <p>Include this in the .htaccess file</p> <pre>RewriteEngine On RewriteRule ^([^\.]*)\.html\$ index.php</pre> <p>This means that any "*.html"-documents should be handled by index.php. Now if is done, TYPO3 will interpret the url of the html-document like this:</p> <p>[title].[id].[type].html</p> <p>Title is optional and only useful for the entries in the apache log-files. You may omit both [title] and [type] but if title is present, type must also be there!.</p> <p><b>Example:</b> TYPO3 will interpret this as page with uid=23 and type=1 : Startpage.23.1.html</p> <p>TYPO3 will interpret this as the page with alias = "start" and the type is zero (default): start.html</p> <p><b>Alternative option (PATH_INFO):</b> Instead of using the rewrite-module in apache (eg. if you're running Windows!) you can use the PATH_INFO variable from PHP. It's very simple. Just set simulateStaticDocuments to "PATH_INFO" and you're up and running!</p> <p><b>Also:</b> See below, .absRefPrefix</p> <p><b>Example (put in Setup-field of your template):</b> config.simulateStaticDocuments = PATH_INFO</p>	<p>default is defined by a configuration option in localconf.php. It's</p> <pre>\$TYPO3_CONF_VARS['FE']['simulateStaticDocuments'] = 1;</pre> <p>This affects all sites in the database. You can also set this value to the string "PATH_INFO"</p>
<b>simulateStaticDocuments_addTitle</b>	int	<p>If not zero, TYPO3 generates urls with the title in, limited to the first [simulateStaticDocuments_addTitle] number of chars.</p> <p><b>Example:</b> Startpage.23.1.html instead of the default, "23.1.html", without the title.</p>	
<b>simulateStaticDocuments_noTypeIfNoTitle</b>	boolean	<p>If set, then the type-value will not be set in the simulated filename if the type value is zero anyways. However the filename must be without a title.</p> <p><b>Example:</b> "Startpage.23.0.html" would <i>still</i> be "Startpage.23.0.html" "23.0.html" would be "23.html" (that is without the zero) "23.1.html" would <i>still</i> be "23.1.html"</p>	
<b>simulateStaticDocuments_replacementChar</b>	string	<p>Word separator for URLs generated by simulateStaticDocuments. If set to hyphen, this option allows search engines to index keywords in URLs. Before TYPO3 4.0 this character was hard-coded to underscore.</p> <p>Depends on the compatibility mode (see Tools&gt;Install&gt;Update wizard):  <i>compatibility mode</i> &lt; 4.0: underscore "_"  <i>compatibility mode</i> &gt;= 4.0: hyphen "-"</p>	

Property:	Data type:	Description:	Default:
<b>simulateStaticDocuments_dontRedirectPathInfoError</b>	boolean	Regarding PATH_INFO mode: When a page is requested by "PATH_INFO" method it must be configured in order to work properly. If PATH_INFO is not configured, the index_ts.php script sends a location header to the correct page. However if you better like an error message outputted, just set this option.	
<b>simulateStaticDocuments_pEnc</b>	string	<p>Allows you to also encode additional parameters into the simulated filename.</p> <p><b>Example:</b> You have a news-plugin. The main page has the url "Page_1.228.0.html" but when one clicks on a news item the url will be "Page_1.228.0.html?&amp;tx_mininews_pil[showUid]=2&amp;cHash=b8d239c224" instead. Now, this URL will not be indexed by external search-engines because of the query-string (everything after the "?" mark). This property avoids this problem by encoding the parameters. These are the options:</p> <p><b>Value set to "base64":</b> This will transform the filename used to this value: "Page_1.228+B6JnR4X2lpbmluZXdzX3BpMVtzaG93VWlkXT0yJmNIYXNoPWl4ZDIzOWMyMjQ_.0.html". The query string has simply been base64-encoded (and some more...) and added to the HTML-filename (so now external search-engines will find this!). The really great thing about this that the filename is self-reliant because the filename contains the parameters. The downside to it is the very very long filename.</p> <p><b>Value set to "md5":</b> This will transform the filename used to this value: "Page_1.228+M5786720If4a.0.html". Now, what a lovely, short filename! Now all the parameters has been hashed into a 10-char string inserted into the filename. At the same time an entry has been added to a cache table in the database so when a request for this filename reaches the frontend, then the REAL parameter string is found in the database! The really great thing about this is that the filename is very short (opposite to the base64-method). The downside to this is that IF you clear the database cache table at any time, the URL here does NOT work until a page with the link has been generated again (re-inserting the parameter list into the database).</p> <p><b>NOTICE:</b> From TYPO3 3.6.0 the encoding will work only on parameters that are manually entered in the list set by .simulateStaticDocuments_pEnc_onlyP (see right below) or those parameters that various plugins might allow in addition. This is to limit the run-away risk when many parameters gets combined.</p>	
<b>simulateStaticDocuments_pEnc_onlyP</b>	string	<p>A list of variables that may be a part of the md5/base64 encoded part of a simulate_static_document virtual filename (see property in the row above).</p> <p><b>Example:</b>  <pre>simulateStaticDocuments_pEnc_onlyP = tx_maillisttofaq_pil[pointer], L, print</pre> </p> <p>-&gt; this will allow the "pointer" parameter for the extension "maillisttofaq" to be included (in addition to whatever vars the extension sets itself) and further the parameter "L" (could be language selection) and "print" (could be print-version).</p>	



Property:	Data type:	Description:	Default:
<b>content_from_pid_allowOutsideDomain</b>	boolean	Using the "Show content from this page instead" feature allows you to insert content from the current domain only. Setting this option will allow content included from anywhere in the page tree!	
<b>absRefPrefix</b>	string	<p>If this value is set, then all relative links in TypoScript are prepended with this string. Used to convert relative paths to absolute paths.</p> <p><b>Note:</b> This value is automatically set to the dirname of the index.php script in case simulateStaticDocuments is set to "PATH_INFO".</p> <p>If you're working on a server where you have both internal and external access, you might do yourself a favor and set the absRefPrefix to the url and path of your site, e.g. http://www.typo3.com/. If you do not, you risk to render pages to cache from the internal network and thereby prefix image-references and links with a non-accessible path from outside.</p>	
<b>pageRendererTemplateFile</b>	string	<p>Sets the template for page renderer class (t3lib_PageRenderer).</p> <p><b>Example:</b></p> <pre>pageRendererTemplateFile = fileadmin/test_pagerender.html</pre>	
<b>noPageTitle</b>	integer	<p>If you only want to have the site name (from the template record) in your &lt;title&gt; tag, set this to 1. If the value is 2 then the &lt;title&gt; tag is not printed at all.</p> <p>Please take note that this tag is required for (X)HTML compliant output, so you should only disable this tag if you generate it manually already.</p>	0
<b>pageTitleFirst</b>	boolean	<p>TYPO3 by default prints a title tag in the format "website: page title".</p> <p>If pageTitleFirst is set (and if the page title is printed), then the page title will be printed IN FRONT OF the template title. So it will look like "page title: website".</p>	0
<b>pageTitleSeparator</b>	string	(Since TYPO3 4.7) The signs which should be printed in the title tag between the website name and the page title.	:
<b>titleTagFunction</b>	function name	Passes the default <title> tag content to this function. No TypoScript parameters are passed though.	
<b>moveJsFromHeaderToFooter</b>	boolean	If set, all JavaScript (includes and inline) will be moved to the bottom of the HTML document, which is after the content and before the closing body tag.	
<b>headerComment</b>	string	The content is added before the "TYPO3 Content Management Framework" comment in the <head> section of the page. Use this to insert a note like that "Programmed by My-Agency".	

Property:	Data type:	Description:	Default:
<b>language</b>	string	<p>Language key. See stdWrap.lang for more information. Select between:  English (default) = [empty]  Danish = dk  German = de  Norwegian = no  Italian = it  etc...</p> <p>Value must correspond with the key used for backend system language if there is one. See inside config_default.php or look at the translation page on TYPO3.org for the official 2-byte key for a given language. Notice that selecting the official key is important if you want labels in the correct language from "locallang" files.</p> <p>If the language you need is not yet a system language in TYPO3 you can use an artificial string of your choice and provide values for it via the TypoScript template where the property "_LOCAL_LANG" for most plugins will provide a way to override/add values for labels. The keys to use must be looked up in the locallang-file used by the plugin of course.</p>	
<b>language_alt</b>	string	<p>If "config.language" (above) is used, this can be set to another language key which will be used for labels if a label was not found for the main language. For instance a brazil portuguese website might specify "pt" as alternative language which means the portuguese label will be shown if none was available in the main language, brazil portuguese. This feature makes sense if one language is incompletely translated and close to another language.</p>	
<b>sys_language_uid</b>	int	<p>This value points to the uid of a record from the "sys_language" table and if set, this means that various parts of the frontend display code will select records which are assigned to this language. See -&gt;SELECT</p> <p>Internally, the value is depending on whether an Alternative Page Language record can be found with that language. If not, the value will default to zero (default language) except if "sys_language_mode" is set to a value like "content_fallback".</p>	

Property:	Data type:	Description:	Default:
<b>sys_language_mode</b>	string	<p>Setting various modes of handling localization. The syntax is "[keyword] ; [value]".</p> <p>Possible keywords are:</p> <p>[default] - The system will look for a translation of the page (from "Alternative Page Language" table) and if it is not found it will fall back to the default language and display that.</p> <p><b>content_fallback</b> - [ Recommended ] The system will always operate with the selected language even if the page is not translated with a page overlay record. This will keep menus etc. translated. However, the <i>content</i> on the page can still fall back to another language, defined by the value of this keyword, e.g. "content_fallback ; 1,0" to fall back to the content of sys_language_uid 1 and if that is not present either, to default (0)</p> <p><b>strict</b> - The system will report an error if the requested translation does not exist. Basically this means that all pages with gray background in the Web&gt;Info / Localization overview module will fail (they would otherwise fall back to default language in one or another way)</p> <p><b>ignore</b> - The system will stay with the selected language even if the page is not translated and there's no content available in this language, so you can handle that situation on your own then.</p>	
<b>sys_language_overlay</b>	boolean / keyword	<p>If set, records from certain tables selected by the CONTENT cObject using the "languageField" setting will select the default language (0) instead of any language set by sys_language_uid / sys_language_mode. In addition the system will look for a translation of the selected record and overlay configured fields.</p> <p>The requirements for this is that the table is configured with "languageField" and "transOrigPointerField" in the [ctrl] section of \$TCA. Also, exclusion of certain fields can be done with the "ll0n_mode" directive in the field-configuration of \$TCA.</p> <p>For backend administration this requires that you configure the "Web&gt;Page" module to display content elements accordingly; That each default element is shown and next to it any translation found. This configuration can be done with Page TSconfig for a section of the website using the object path "mod.web_layout.defLangBinding = 1".</p> <p>Keyword: <b>hideNonTranslated</b> : If this keyword is used a record that has no translation will not be shown. The default is that records with no translation will show up in the default language.</p>	

Property:	Data type:	Description:	Default:
<b>sys_language_softMergeIfNotBlank</b>	string	<p>Setting additional "mergeIfNotBlank" fields from TypoScript.</p> <p><b>Background:</b> In TCA you can configure "l10n_mode" - localization mode - for each field. Two of the options affect how the frontend displays content; The values "exclude" and "mergeIfNotBlank" (see "TYPO3 Core API" document for details). The first ("exclude") simply means that the field when found in a translation of a record will not be overlaid the default records field value. The second ("mergeIfNotBlank") means that it will be overlaid <i>only</i> if it has a non-blank value.</p> <p>Since it might be practical to set up fields for "mergeIfNotBlank" on a per-site basis this options allows you to override additional fields from tables.</p> <p><b>Syntax:</b> [table]:[field], [table]:[field], [table]:[field], ...</p> <p><b>Example:</b> <code>config.sys_language_softMergeIfNotBlank = tt_content:image , tt_content:header</code></p> <p>This setting means that the header and image field of content elements will be used from the translation only if they had a non-blank value. For the image field this might be very practical because it means that the image(s) from the default translation will be used unless other images are inserted!</p>	
<b>sys_language_softExclude</b>	string	<p>Setting additional "exclude" flags for l10n_mode in TCA for frontend rendering. Works exactly like sys_language_softMergeIfNotBlank (see that for details - same Syntax!).</p> <p>Fields set in this property will override if the same field is set for "sys_language_softMergeIfNotBlank".</p>	
<b>typolinkCheckRootline</b>	boolean	<p>If set, then every "typolink" is checked whether it's linking to a page within the current rootline of the site.</p> <p>If not, then TYPO3 searches for the first found domain record (without redirect) in that rootline from out to in.</p> <p>If found (another domain), then that domain is prepended the link, the external target is used instead and thus the link jumps to the page in the correct domain.</p>	

Property:	Data type:	Description:	Default:
<b>typolinkEnableLinksAcrossDomains</b>	boolean	<p>This option enables to create links across domains using current domain's linking scheme.</p> <p>If this option is not set, then all cross-domain links will be generated as "http://domain.tld/index.php?id=12345" (where 12345 is page id). If this option is set and current site uses, for example, simulateStatic, then links will be generated as "http://domain.tld/PageTitle.12345.html" (includes RTE links too). Setting this option requires that domains, where pages are linked, have the same configuration for:</p> <ul style="list-style-type: none"> <li>- linking scheme (i.e. all use simulateStatic or RealURL or CoolURI but not any mixture)</li> <li>- all domains have identical localization settings (config.sys_language_XXX directives)</li> <li>- all domains have the same set of languages defined</li> </ul> <p>This option implies "config.typolinkCheckRootline=1", which will be activated automatically. Setting value of "config.typolinkCheckRootline" inside TS template will have no effect.</p> <p>Disclaimer: it must be understood that while link is generated to another domain, it is still generated in the context of current domain. No side effects are known at the time of writing of this documentation but they may exist. If any side effects are found, this documentation will be updated to include them.</p>	0
<b>typolinkLinkAccessRestrictedPages</b>	integer (page id) / keyword "NONE"	<p>If set, typolinks pointing to access restricted pages will still link to the page even though the page cannot be accessed. If the value of this setting is an integer it will be interpreted as a page id to which the link will be directed.</p> <p>If the value is "NONE" the original link to the page will be kept although it will generate a page-not-found situation (which can of course be picked up properly by the page-not-found handler and present a nice login form).</p> <p>See "showAccessRestrictedPages" for menu objects as well (similar feature for menus)</p> <p><b>Example:</b></p> <pre>config.typolinkLinkAccessRestrictedPages = 29 config.typolinkLinkAccessRestrictedPages_addParams = &amp;return_url=###RETURN_URL###&amp;pageId=###PAGE_ID###</pre> <p>Will create a link to page with id 29 and add GET parameters where the return URL and original page id is a part of it.</p>	
<b>typolinkLinkAccessRestrictedPages_addParams</b>	string	See "typolinkLinkAccessRestrictedPages" above	
<b>notification_email_urlmode</b>	string	<p>This option allows you to handle URL's in plain text emails so long URLs of more than 76 chars are not broken. This option can be either empty or "76" or "all".</p> <p>If the string is blank, all links in plaintext emails are untouched.</p> <p>If it's set to 76 then all links longer then 76 characters are stored in the database and a hash is sent in the GET-var ? RDCT=[md5/20] to the index.php script which finds the proper link in the database and issues a location header (redirection).</p> <p>If the value is "all" then ALL "http://" links in the message are converted.</p>	

Property:	Data type:	Description:	Default:
<b>notification_email_encoding</b>	string	This sets the encoding of plaintext emails (notification messages). The default encoding is "quoted-printable". But setting this to eg. "base64" will encode the content with base64 encoding.  <b>Values possible:</b> base64 quoted-printable 8bit	
<b>notification_email_charset</b>	string	Alternative charset for the notification mails.	Until TYPO3 4.7: ISO-8859-1 Since TYPO3 4.7: utf-8
<b>admPanel</b>	boolean	If set, the admin panel appears in the bottom of pages.  <b>NOTE:</b> In addition the panel must be enabled for the user as well, using the TSconfig for the user! See the TSconfig reference about additional admin panel properties.	
<b>beLoginLinkIPList</b>	[IP-number]	If set and REMOTE_ADDR matches one of the listed IP-numbers (Wild-card, *, allowed) then a link to the typo3/login scrip with redirect pointing back to the page is shown.  <b>NOTE:</b> beLoginLinkIPList_login and/or beLoginLinkIPList_logout (see below) must be defined if the link should show up!	
<b>beLoginLinkIPList_login</b>	HTML	HTML code wrapped with the login link, see 'beLoginLinkIPList'  <b>Example:</b> <hr /><b>LOGIN</b>	
<b>beLoginLinkIPList_logout</b>	HTML	HTML code wrapped with the logout link, see above	
<b>index_enable</b>	boolean	Enables cached pages to be indexed.	
<b>index externals</b>	boolean	If set, external media linked to on the pages is indexed as well.	
<b>index_descrLgd</b>	int	This indicates how many chars to preserve as description for an indexed page. This may be used in the search result display.	200
<b>index_metatags</b>	boolean	This allows to turn on or off the indexing of metatags. It is turned on by default.	true

Property:	Data type:	Description:	Default:
<b>xhtml_cleaning</b>	string	<p>Tries to clean up the output to make it XHTML compliant and a bit more. THIS IS NOT COMPLETE YET, but a "pilot" to see if it makes sense anyways. For now this is what is done:</p> <p><b>What it does at this point:</b></p> <ul style="list-style-type: none"> <li>- All tags (img,br,hr) is ended with "&gt;" - others?</li> <li>- Lowercase for elements and attributes</li> <li>- All attributes in quotes</li> <li>- Add "alt" attribute to img-tags if it's not there already.</li> </ul> <p><b>What it does NOT do (yet) according to XHTML specs.:</b></p> <ul style="list-style-type: none"> <li>- Wellformedness: Nesting is NOT checked</li> <li>- name/id attribute issue is not observed at this point.</li> <li>- Certain nesting of elements not allowed. Most interesting, &lt;PRE&gt; cannot contain img, big,small,sub,sup ...</li> <li>- Wrapping scripts and style element contents in CDATA - or alternatively they should have entities converted.</li> <li>- Setting charsets may put some special requirements on both XML declaration/ meta-http-equiv. (C.9)</li> <li>- UTF-8 encoding is in fact expected by XML!!</li> <li>- stylesheet element and attribute names are NOT converted to lowercase</li> <li>- ampersands (and entities in general I think) MUST be converted to an entity reference! (&amp;);. This may mean further conversion of non-tag content before output to page. May be related to the charset issue as a whole.</li> <li>- Minimized values not allowed: Must do this: selected="selected"</li> </ul> <p>Please see the class t3lib_parsehtml for details. You can enable this function by the following values:</p> <p><b>all</b> = the content is always processed before it may be stored in cache. <b>cached</b> = only if the page is put into the cache, <b>output</b> = only the output code just before it's echoed out.</p>	
<b>prefixLocalAnchors</b>	string keyword	<p>If set to one of the keywords, the content will have all local anchors in links prefixed with the path of the script. Basically this means that &lt;a href="#"&gt; will be transformed to &lt;a href="path/path/script?params#"&gt;. This procedure is necessary if the &lt;base&gt; tag is set in the script (eg. if "realurl" extension is used to produce Speaking URLs).</p> <p>Keywords are the same as for "xhtml_cleaning", see above.</p>	
<b>disablePrefixComment</b>	boolean	<p>If set, the stdWrap property "prefixComment" will be disabled, thus preventing any revealing and space-consuming comments in the HTML source code.</p>	
<b>baseURL</b>	string	<p>This writes the &lt;base&gt; tag in the header of the document. Set this to the value that is expected to be the URL and append a "/" to the end of the string.</p> <p><b>Example:</b></p> <pre>config.baseURL = http://typo3.org/sub_dir/</pre>	
<b>tx_[extension key with no underscores]_[*]</b>	-	<p>Configuration space for extensions. This can be used – for example – by plugins that need some TypoScript configuration, but that don't actually display anything in the frontend (i.e. don't receive their configuration as an argument from the frontend rendering process).</p> <p><b>Example:</b></p> <pre>config.tx_realurl_enable = 1</pre>	

[tsref:config/->CONFIG]

## "CONSTANTS"

Property:	Data type:	Description:	Default:
<b>Array...</b>	<i>string</i>	Constants.  <b>Examples:</b> .EMAIL = <i>email@email.com</i> Now if parseFunc anywhere is configured with constants=1 then all cases of the string ###EMAIL### will be substituted in the text. see ->parseFunc	

[tsref:constants]

## "PAGE"

Pages are referenced by two main values. The "id" and "type".

The "id" points to the uid of the page (or the alias). Thus the page is found.

The "type" is used to define how the page should be rendered. This is primarily used with framesets. Here the frameset normally has the type=0 (or not set) and the documents in the frameset would be defined with another type, e.g. type=1 for the content-page.

You should explore the framesets of the TYPO3-sites around. Also look in the standard-templates for framesets.

It's a good habit to use type=1 for the main-page of a website with frames. With no-frames sites type is normally zero.

Another good habit is to use "page" as the top-level object name for the content-page on a website.

Most of this codes is executed in the PHP-script *typo3/sysext/cms/tslib/class.tslib\_pagegen.php*.

Property:	Data type:	Description:	Default:
<b>typeNum</b>	int	This decides the the typeId of the page. The value defaults to 0 for the first found PAGE object, but it MUST be set and be unique as soon you use more than one such object (watch this if you use frames on your page)!	0
<b>1,2,3,4...</b>	cObject		
<b>wrap</b>	wrap	Wraps the content of the the cObject array	
<b>stdWrap</b>	->stdWrap	Wraps the content of the the cObject array with stdWrap options	
<b>bodyTagCObject</b>	cObject	This is default bodytag overridden by ".bodyTag" if that is set.	
<b>bodyTag</b>	<tag>	Body tag on the page  <b>Example:</b> <body bgcolor="{ \$bgCol }">	<body bgcolor="#FFFFFF">
<b>headTag</b>	<tag>	Head-tag if alternatives are wanted	<head>
<b>bodyTagMargins</b>	int	margins in the body tag.  <b>Property:</b> .useCSS = 1 (boolean) - will set a "BODY {margin: ...}" line in the in-document style declaration - for XHTML compliance.  <b>Example:</b> value 4 adds <i>leftmargin="4" topmargin="4" marginwidth="4" marginheight="4"</i> to the bodyTag.	
<b>bodyTagAdd</b>	string	This content is added to the end of the bodyTag.	
<b>bgImg</b>	imgResource	Background image on the page. This is automatically added	



Property:	Data type:	Description:	Default:
		to the body-tag.	
<b>frameSet</b>	->FRAMESET	if any properties is set to this property, the page is made into a frameset.	
<b>meta</b>	->META		
<b>shortcutIcon</b>	resource	Favicon of the page. Create a reference to an icon here! Browsers that support favicons display them in the browser's address bar, next to the site's name in lists of bookmarks, and next to the page's title in the tab.  <b>Note:</b> This must be a valid ".ico"-file (iconfile)	
<b>headerData</b>	->CARRAY	Inserts content in the header-section. Could be JavaScripts, meta-tags, other stylesheet references. By default, gets inserted after all the style definitions.	
<b>footerData</b>	->CARRAY	Same as headerData above, except that this block gets included at the bottom of the page (just before the closing body tag).	
<b>config</b>	->CONFIG	configuration for the page. Any entries override the same entries in the toplevel-object "config".	
<b>includeLibs</b>	<i>array of strings</i>	With this you may include php-files. This does the same as "includeLibrary" in ->CONFIG but this can include more than one file. These files are included <i>after</i> the file of includeLibrary.  <b>NOTE:</b> The toplevel object "includeLibs" and the scripts defined with this property is added to each other. Script-keys (that is the "array of strings"-value, like below "tx_myext") from this property of the page overrides any scripts-keys from the toplevel "includeLibs" property! The script-filenames are of the datatype "resource".  <b>Example:</b> includeLibs.tx_myext = lib_filename.php	
<b>JavaScript:</b>			
<b>javascriptLibs</b>	<i>array of strings</i>	This allows to include the JavaScript libraries that are shipped with the TYPO3 Core.  <pre> javascriptLibs {     # include prototype     Prototype = 1      # include Scriptaculous     Scriptaculous = 1     # adds modules dragdrop and controls to Scriptaculous     Scriptaculous.modules = dragdrop,controls      # include ExtCore     ExtCore = 1     # include ExtCore debug file     (uncompressed)     ExtCore.debug = 1      # includes ExtJS     ExtJs = 1     # include ext-all.css     ExtJs.css = 1     # include default theme     ExtJs.theme = 1     # load specific adapter (jquery prototype yui)     ExtJs.adapter = ... </pre>	

Property:	Data type:	Description:	Default:
		<pre> # initialize QuickTips ExtJs.quickTips = 1 # includes ExtJS debug file (uncompressed) ExtJs.debug = 1  # include SVG library SVG = 1 # include SVG debug file SVG.debug = 1 #force rendering with flash SVG.forceFlash = 1 } </pre> <p><b>Note:</b> If both ExtCore and ExtJS are requested, the only superset ExtJS will be loaded. This will also affect any options set. They will only come from ExtJS.</p> <p><b>Note:</b> In TYPO3 4.5.2 and older you should either request ExtJS or ExtCore, but not both together. Requesting both at the same time will lead to errors.</p>	
<b>inlineLanguageLabel</b>	<i>array of strings</i>	<p>ExtJS specific, adds language labels to the page.</p> <p><b>Example:</b></p> <pre> inlineLanguageLabel {     label1 = 123     label2 = 456 } </pre> <p>will produce following source:</p> <pre> TYPO3.lang = {"label1":"123","label2":"456"}; </pre>	
<b>inlineSettings</b>	<i>array of strings</i>	<p>ExtJS specific, adds settings to the page.</p> <p><b>Example:</b></p> <pre> page.inlineSettings {     setting1 = Hello     setting2 = GoOnTop } </pre> <p>will produce following source:</p> <pre> TYPO3.settings = {"TS": {"setting1":"Hello","setting2":"GoOnTop"}}; </pre>	
<b>extOnReady</b>	->CARRAY	<p>ExtJS specific, adds inline JavaScript, wrapped in Ext.onReady.</p> <p><b>Example:</b></p> <pre> page.extOnReady {     10 = TEXT     10.value = Ext.Msg.alert("TypoScript Message","Hello World!"); } </pre> <p>will produce following source:</p> <pre> Ext.onReady(function() {Ext.Msg.alert("TypoScript Message","Hello World!"); }); </pre>	
<b>includeJSlibs. [array]</b>	resource	<p>Adds JS library files to head of page.</p> <p>The file definition must be a valid "resource" data type, otherwise nothing is inserted. This means that remote files cannot be referenced (i.e. using "http://..."), except by using the ".external" property.</p> <p>Each file has <i>optional properties</i>:</p> <p><b>.allWrap</b> - wraps the complete tag, useful for conditional comments.</p>	

Property:	Data type:	Description:	Default:
		<p><b>.disableCompression</b> - (Since TYPO3 4.6) If config.compressJs is enabled, this disables the compression of this file.</p> <p><b>.excludeFromConcatenation</b> - (Since TYPO3 4.6) If config.concatenateJs is enabled, this prevents the file from being concatenated.</p> <p><b>.external</b> - If set, there is no file existence check. Useful for inclusion of external files.</p> <p><b>.forceOnTop</b> - boolean flag. If set, this file will be added on top of all other files.</p> <p><b>.if</b> - (Since TYPO3 4.7) Allows to define conditions, which must evaluate to TRUE for the file to be included. If they do not evaluate to TRUE, the file will not be included. Extensive usage might cause huge numbers of temporary files to be created. See -&gt;if for details.</p> <p><b>Example:</b></p> <pre>includeJSlibs.twitter = http://twitter.com/javascripts/blogger.js includeJSlibs.twitter.external = 1</pre>	
<b>includeJSFooterlibs.[array]</b>	resource	Same as includeJSlibs above, except that this block gets included at the bottom of the page (just before the closing body tag).	
<b>includeJS.[array]</b>	resource	<p>Inserts one or more (Java)Scripts in &lt;script&gt; tags.</p> <p>The file definition must be a valid "resource" data type, otherwise nothing is inserted. This means that remote files cannot be referenced (i.e. using "http://..."), except by using the ".external" property.</p> <p>Each file has <i>optional properties</i>:</p> <p><b>.allWrap</b> - wraps the complete tag, useful for conditional comments.</p> <p><b>.disableCompression</b> - (Since TYPO3 4.6) If config.compressJs is enabled, this disables the compression of this file.</p> <p><b>.excludeFromConcatenation</b> - (Since TYPO3 4.6) If config.concatenateJs is enabled, this prevents the file from being concatenated.</p> <p><b>.external</b> - If set, there is no file existence check. Useful for inclusion of external files.</p> <p><b>.forceOnTop</b> - boolean flag. If set, this file will be added on top of all other files.</p> <p><b>.if</b> - (Since TYPO3 4.7) Allows to define conditions, which must evaluate to TRUE for the file to be included. If they do not evaluate to TRUE, the file will not be included. Extensive usage might cause huge numbers of temporary files to be created. See -&gt;if for details.</p> <p><b>.type</b> - setting the MIME type of the script (default: text/javascript).</p> <p><b>Example:</b></p> <pre>includeJS { file1 = fileadmin/helloworld.js file1.type = application/x-javascript # Include a second file, but only if myConstant is set in the TS constants field. file2 = javascript_uploaded_to_template*.js file2.if.isTrue = {\$myConstant} }</pre>	
<b>includeJSFooter.[array]</b>	resource	Same as includeJS above, except that this block gets included at the bottom of the page (just before the closing body tag).	
<b>jsInline</b>	->CARRAY	Use cObjects for creating inline JavaScript	

Property:	Data type:	Description:	Default:
		<p><b>Example:</b></p> <pre>page.jsInline {     10 = TEXT     10.dataWrap = var pageId = {TSFE:id}; }</pre> <p><b>Note:</b> with config.removeDefaultJS = external, the inlineJS is moved to external file. with config.minifyJS = 1, the jsInline will be minified as well.</p>	
<b>jsFooterInline</b>	->CARRAY	Same jsInline above, except that the JavaScript gets inserted at the bottom of the page (just before the closing body tag).	
<b>inlineJS</b>	->CARRAY	<p>Inserts inline JavaScript in the header-section. Don't use script-tags as they are added by TYPO3.</p> <p><b>Example:</b></p> <pre>page.inlineJS.10 = TEXT page.inlineJS.10.value = function a(val) { alert(val); }</pre> <p>with config.removeDefaultJS = external the inlineJS is moved to external file. with config.minifyJS = 1 the inlineJS will be minified as well.</p> <p><b>Note:</b> This option was deprecated and has been removed in TYPO3 4.3. Use jsInline instead.</p>	
<b>CSS Stylesheets:</b>			
<b>stylesheet</b>	resource	Inserts a stylesheet in the <HEAD>-section of the page; <link rel="stylesheet" href="[resource]">	
<b>includeCSS.</b> [array]	resource	<p>Inserts a stylesheet (just like the .stylesheet property) by allows to setting up more than a single stylesheet, because you can enter files in an array.</p> <p>The file definition must be a valid "resource" data type, otherwise nothing is inserted.</p> <p>Each file has <i>optional properties</i>:</p> <ul style="list-style-type: none"> <li><b>.allWrap</b> - wraps the complete tag, useful for conditional comments.</li> <li><b>.alternate</b> - If set (boolean) then the rel-attribute will be "alternate stylesheet".</li> <li><b>.disableCompression</b> - (Since TYPO3 4.6) If config.compressCss is enabled, this disables the compression of this file.</li> <li><b>.excludeFromConcatenation</b> - (Since TYPO3 4.6) If config.concatenateCss is enabled, this prevents the file from being concatenated.</li> <li><b>.external</b> - If set, there is no file existence check. Useful for inclusion of external files.</li> <li><b>.if</b> - (Since TYPO3 4.7) Allows to define conditions, which must evaluate to TRUE for the file to be included. If they do not evaluate to TRUE, the file will not be included. Extensive usage might cause huge numbers of temporary files to be created. See -&gt;if for details.</li> <li><b>.import</b> - If set (boolean) then the @import way of including a stylesheet is used instead of &lt;link&gt;</li> <li><b>.media</b> - setting the media attribute of the &lt;style&gt; tag.</li> <li><b>.title</b> - setting the title of the &lt;style&gt; tag.</li> </ul> <p><b>Example:</b></p> <pre>includeCSS {     file1 = fileadmin/mystylesheet1.css     file2 =</pre>	

Property:	Data type:	Description:	Default:
		<pre> stylesheet_uploaded_to_template*.css file2.title = High contrast file2.media = print ie6Style = fileadmin/css/style3.css ie6Style.allWrap = &lt;!--[if lte IE 7]&gt; &lt;![endif]--&gt; cooliris = http://www.cooliris.com/shared/ resources/css/global.css cooliris.external = 1 </pre>	
<b>cssInline</b>	->CARRAY	<p>Use cObjects for creating inline CSS</p> <p><b>Example:</b></p> <pre> cssInline {     10 = TEXT     10.value = h1 {margin:15px;}      20 = TEXT     20.value = h1 span {color: blue;} } </pre>	
<b>CSS_inlineStyle</b>	string	This value is just passed on as inline css (in-document css encapsulated in <style>-tags)	
<b>Other:</b>			
<b>insertClassesFromRTE</b>	boolean	<p>If set, the classes for the Rich Text Editor configured in Page TSconfig is inserted in as the first thing in the Style-section right after the setting of the stylesheet.</p> <p><b>.add_mainStyleOverrideDefs</b> = [ * / list of tags ] - will add all the "RTE.default. mainStyleOverride_add" - tags configured as well.</p> <p><i>Might be deprecated soon. Most likely the RTE should be configured by the stylesheet instead. Stay tuned...</i></p>	
<b>noLinkUnderline</b>	boolean	<p>Disables link-underlining. Uses in-document stylesheet.</p> <p><i>Deprecated. Use stylesheet instead.</i></p>	
<b>hover</b>	HTML-color	<p>The color of a link when the mouse moves over it! (only MSIE). Uses in-document stylesheet.</p> <p><i>Deprecated. Use stylesheet instead.</i></p>	
<b>hoverStyle</b>	string	<p>Additional style information to the hover-color.</p> <p><b>Example:</b></p> <pre> page.hoverStyle = font: bold; text-decoration: none; </pre> <p><i>Deprecated. Use stylesheet instead.</i></p>	
<b>smallFormFields</b>	boolean	<p>Renders formfields like textarea, input and select-boxes small with "verdana size 1" font. Uses in-document stylesheet.</p> <p><b>Tip:</b> Use this together with the config-option "compensateFieldWidth" set to "0.6" for netscape-browsers in order to render the small form fields in the same width!</p> <p><i>Deprecated. Use stylesheet instead.</i></p>	
<b>adminPanelStyles</b>	boolean	Will include CSS styles for the Admin Panel.	

[tsref:(page)]

## "FE\_DATA"

Property:	Data type:	Description:	Default:
<b>array of tableNames</b>	->FE_TABLE		

[tsref:FEData]

## "FE\_TABLE"

Property:	Data type:	Description:	Default:
<b>default.[field]</b>	string	<p>This property is in charge of which default-values is used for the table:</p> <p><b>Example:</b> This defines the default values used for new records. These values will be overridden with any value submitted instead (as long as the submitted fields are allowed due to "allowNew")</p> <pre>default {     subject = This is the default subject value!     hidden = 1     parent = 0 }</pre>	
<b>allowNew.[field]</b>	string	<p>This property is in charge of which fields that may be written from the frontend.</p> <p><b>Example:</b> This defines that subject is a field, that may be submitted from the frontend. If a value is not submitted, subject is filled with the default value (see above). The field "hidden" on the other hand cannot be changed from the frontend. "hidden" will gain the value from the default definition (see above). If fields are set to "0" (zero) it's the same as if they were not defined in this array.</p> <pre>allowNew {     subject = 1     hidden = 0 }</pre>	
<b>allowEdit.[field]</b>	string	<p>Same as above ("allowNew") but this controls which fields that may be written in case of an update of a record (and not a new submission) Please pay attention to the property below! ("overrideEdit")</p>	
<b>overrideEdit.[field]</b>	string	<p>This works like default-values above but is values inserted after the submitted values have been processed. This means that opposite to default-values overwritten by the submitted values, these values override the submitted values.</p> <p><b>Example:</b> In this case overrideEdit secures that if a user updates his record (if he "own" it) the "hidden"-field will be set no matter what.</p> <pre>overrideEdit {     hidden = 1 }</pre>	
<b>userIdColumn</b>	string (field)	This is a string that points to the column of a record where the user-id of the current fe_user should be inserted. This fe_user-uid is inserted/updated both by "new" and "edit"	
<b>autoInsertPID</b>	boolean	Works with new records: Insert automatically the PID of the page, where the submitted data is sent to. Any "pid" supplied from the submitted data will override. This is for convenience.	

Property:	Data type:	Description:	Default:
<b>processScript</b>	resource	Include-script to be used for processing of incoming data to the table. The script is included from a function in the class tslib_fetce This is the really important option, because whether or not you are going to utilize the "cleaning"/"authorization" features of the properties above depend on how you write your script to process data and put it in the database. A very good example is to look at "media/scripts/guest_submit.inc", included from static_template "plugin.tt_guest" (Used for the default guestbook feature)	
<b>separator</b>	string	Separator character used when the submitted data is an array from eg. a multiple selector box.	chr(10) (linebreak)
<b>doublePostCheck</b>	string (field name)	Specifies a field name (integer) into which an integer-hash compiled of the submitted data is inserted. If the field is set, then submissions are checked whether another record with this value already exists. If so, the record is NOT inserted, because it's expected to be a "double post" (posting the same data more than once)	

[tsref:FEData.(tablename)/->FE\_TABLE]

## "FRAMESET"

Property:	Data type:	Description:	Default:
<b>1,2,3,4...</b>	frameObj	Configuration of frames and nested framesets.	
<b>cols</b>	<frameset>-data:cols	Cols	
<b>rows</b>	<frameset>-data:rows	Rows	
<b>params</b>	<frameset>-params	<b>Example:</b> border="0" framespacing="0" frameborder="NO"	

[tsref:(page).frameSet/->FRAMESET]

## "FRAME"

Property:	Data type:	Description:	Default:
<b>obj</b>	<i>pointer to top-level object-name</i>	top-level object-name of a PAGE / FRAMESET <b>Example:</b> "left", "page", "frameset"	
<b>options</b>	<i>url-parameters</i>	<b>Example:</b> print=1&othervar=anotherthing would add '&print=1&othervar=anotherthing' to the ".src"-content (if not ".src" is set manually!!)	
<b>params</b>	<frame>-params	<b>Example:</b> scrolling="AUTO" noresize frameborder="NO"	
<b>name</b>	<frame>-data:name	Manually set name of frame  <b>NOTE:</b> Is set automatically and should not be overridden under normal conditions!	value of ".obj"
<b>src</b>	<frame>-data:src /stdWrap	Instead of using the "obj" destination, you can define a specific src for your frame with this setting. This overrides the default behavior of using the "obj" parameter!	typolink to id=[currentId] &type=[obj- >typeNum]

[tsref:(page).frameSet.(number)/->FRAMESET.(number)]

**Example of a simple frameset with a topframe and content-frame:**

```
frameset = PAGE
frameset.typeNum = 0

page = PAGE
page.typeNum = 1

top = PAGE
top.typeNum = 3

frameset.frameSet.rows = 150,*
frameset.frameSet.params = border="0" framespacing="0" frameborder="NO"
frameset.frameSet {
    1 = FRAME
    1.obj = top
    1.params = scrolling="NO" noresize frameborder="NO" marginwidth="0" marginheight="0"
    2 = FRAME
    2.obj = page
    2.params = scrolling="AUTO" noresize frameborder="NO"
}
```

## "META"

Property:	Data type:	Description:	Default:
<b>Array...</b>	string /stdWrap	<p>Allows you to define meta tags.</p> <p>Use the scheme meta.key = value. The "key" can be the name of any meta tag, e.g. "description" or "keywords". If the key is "refresh" (case insensitive), then the "http-equiv" attribute is used in the meta tag instead of the "name" attribute. If the "value" is empty (after trimming), the meta tag is not generated.</p> <p><b>Examples:</b></p> <pre>meta.description = This is the description of the content in this document. meta.keywords = These are the keywords. meta.refresh = [seconds]; [url, leave blank for same page]</pre> <p>For each key the following sub-property is available: httpEquivalent: (Since TYPO3 4.7) If set to 1, the http-equiv attribute is used in the meta tag instead of the "name" attribute. Default: 0.</p> <p><b>Example:</b></p> <pre>meta.X-UA-Compatible = IE=edge,chrome=1 meta.X-UA-Compatible.httpEquivalent = 1</pre> <p>This results in &lt;meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"&gt;.</p>	

[tsref:->META]

## "CARRAY"

Property:	Data type:	Description:	Default:
<b>1,2,3,4...</b>	cObject	<p>This is a numerical "array" of content-objects (cObjects). The order by which you specify the objects is not important as the array will be sorted before it's parsed!</p>	
<b>Occasional properties:</b>			



Property:	Data type:	Description:	Default:
<b>(stdWrap properties...)</b>		<p><b>NOTE:</b> This applies ONLY if "CARRAY /stdWrap" is set to be data type If you specify any non-integer properties to a CARRAY, stdWrap will be invoked with all properties of the CARRAY.</p> <p><b>Example:</b> This will return '<b>This will be rendered before</b> "10"testing'</p> <p>10 = TEXT 10.value = testing 5 = TEXT 5.value = This will be rendered before "10" wrap = <b> </b></p>	
<b>(TDParams)</b>	<TD>-params	<p><b>NOTE:</b> This applies ONLY if "CARRAY +TDParams" is set to be data type This property is used only in some cases where CARRAY is used. Please look out for a note about that in the various cases.</p>	

[tsref:-&gt;CARRAY]

# Content Objects (cObject)

## PHP information

The content objects (cObjects) are primarily controlled by the PHP-script "typo3/sysexst/cms/tslib/class.tslib\_content.php". The PHP-class is named "tslib\_cObj" and often this is also the variable-name of the objects (\$cObj)

The \$cObj in PHP has an array, \$this->data, which holds records of various kind. See data type "getText".

This record is normally "loaded" with the record from a table depending on the situation. Say if you are creating a menu it's often loaded with the page-record of the actual menuitem or if it's about content-rendering it'll be the content-record.

## REUSING cOBJECTS

When dealing with "cObjects", you're allowed to use a special syntax in order to reuse cObjects without actually creating a copy. This has the advantage of minimizing the size of the cached template. But on the other hand it doesn't give you the flexibility of overriding values.

This example will show you how it works:

```
#
# Temporary objects are defined:
#
lib.stdheader = COA
lib.stdheader {
    stdWrap.wrapAlign.field = header_position
    stdWrap.typolink.parameter.field = header_link
    stdWrap.fieldRequired = header

    1 = TEXT
    1.current = 1
    1.fontTag = {$content.wrap.header1}

    stdWrap.space = {$content.headerSpace}
}

#
# CType: header
#
tt_content.header = COA
tt_content.header {
    10 < lib.stdheader
    10.stdWrap.space >

    20 = TEXT
    20.field = subheader
    20.fontTag = {$content.wrap.subheader1}
}

#
# CType: bullet
#
tt_content.bullets = COA
tt_content.bullets {
    10 = < lib.stdheader
    20 < styles.content.bulletlist_gr
}
```

First lib.stdheader is defined. This is (and must be) a cObject! (In this case it is COA.)

Now *lib.stdheader* is copied to *tt\_content.header.10* with the "<" operator. This means that an actual copy of *lib.stdheader* is created at *parsetime*.

But this is not the case with *tt\_content.bullets.10*. Here lib.stdheader is just pointed to and lib.stdheader

will be used as the cObject at *runtime*.

The reason why lib.stdheader was copied in the first case is the fact that it's needed to unset ".stdWrap.space" inside the cObject ("10.stdWrap.space >"). This could NOT be done in the second case where only a pointer is created.

**NOTE:**

If *lib.stdheader* was *temp.stdheader* instead, the pointer would not work! This is due to the fact that the runtime-reference would find nothing in "temp." as this is unset before the template is stored in cache!

This goes for "temp." and "styles." (see the toplevel object definition elsewhere).

Overriding values anyway:

Although you cannot override values TypoScript-style (using the operators and all) the properties of the object which has the reference will be merged with the configuration of the reference.

**Example:**

```
page.10 = TEXT
page.10.value = kasper
page.10.case = upper

page.20 = < page.10
page.20.case = lower
page.20.value >
page.20.field = pages
```

The result is this config:

value	kasper
case	lower
field	pages

Notice that .value was not cleared (the red line), because it's simply two arrays which are joined:

value	kasper
case	upper

case	lower
field	pages

So hence the red line in the above example is useless.

# HTML

The content object "HTML" can be used to output static text or html. stdWrap is available for the cObject itself and for the property "value". See the examples.

**Note:** This content object is deprecated since TYPO3 4.6 and will be removed in TYPO3 6.0. Use the content object "TEXT" instead!

Property:	Data type:	Description:	Default:
<b>value</b>	HTML /stdWrap	Raw HTML-code.	
<b>stdWrap</b>	->stdWrap	(Executed after the stdWrap for the property ".value".)	

[tsref:(cObject).HTML]

**Example:**

```
10 = HTML
10.value = This is a text in uppercase
10.value.case = upper
```

**Example:**

```
10 = HTML
10.value.field = bodytext
10.value.br = 1
```

**Example:**

```
10 = HTML
10.stdWrap.field = title
10.stdWrap.wrap = <strong>|</strong>
```

# TEXT

The content object "TEXT" can be used to output static text or html. So it is very similar to the cObject "HTML". Note that the stdWrap properties are not available under the property "stdWrap" (as they are for the other cObjects), but on the very rootlevel of the object. This is non-standard! Check the examples.

Property:	Data type:	Description:	Default:
<b>value</b>	value /stdWrap	Text, which you want to output.	
<b>(stdWrap properties...)</b>	->stdWrap		

[tsref:(cObject).TEXT]

**Example:**

```
10 = TEXT
10.value = This is a text in uppercase
10.case = upper
```

**Example:**

```
10 = TEXT
10.field = bodytext
10.br = 1
```

**Example:**

```
10 = TEXT
10.field = title
10.wrap = <strong>|</strong>
```

## COBJ\_ARRAY (COA, COA\_INT)

This is a cObject, in which you can place several other cObjects using numbers to enumerate them.

It has the alias COA standing for "content object array". You can also call it "COA" instead of COBJ\_ARRAY.

You can also create this object as a COA\_INT in which case it works exactly like the USER\_INT object does: It's rendered non-cached! The COA\_INT provides a way to utilize this feature not only with USER cObjects but with any cObject.

Property:	Data type:	Description:	Default:
<b>1,2,3,4...</b>	cObject		
<b>if</b>	->if	if "if" returns false the COA is NOT rendered	
<b>wrap</b>	wrap /stdWrap		
<b>stdWrap</b>	->stdWrap		
<b>includeLibs</b>	<i>list of resource /stdWrap</i>	<b>(This property is used only if the object is COA_INT!, See introduction.)</b> This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	

[tsref:(cObject).COA/(cObject).COA\_INT/(cObject).COBJ\_ARRAY]

### Example:

```
temp.menutable = COBJ_ARRAY
temp.menutable {
    10 = TEXT
    10.value = <table border="0" cellpadding="0" cellspacing="0">

    20 = HMENU
    20.entryLevel = 0
    20.1 = GMENU
    20.1.NO {
        wrap = <tr><td> | </td></tr>
        XY = {$menuXY}
        backColor = {$bgCol}
        20 = TEXT
        20 {
            text.field = title
            fontFile = media/fonts/hatten.ttf
            fontSize = 23
            fontColor = {$menuCol}
            offset = |*| 5,18 || 25,18
        }
    }

    30 = TEXT
    30.value = </table>
}
```

## FILE

This object returns the content of the file specified in the property "file".

It is defined as PHP function fileResource() in /typo3/sysex/cms/tslib/class.tslib\_content.php.

Property:	Data type:	Description:	Default:
<b>file</b>	resource /stdWrap	The file whose content should be returned. If the resource is <b>jpg, jpeg, gif or png</b> the image is inserted as an image-tag. All other formats are read and inserted into the HTML-code. The maximum filesize of documents to be read is set to 1024 kb internally!	
<b>linkWrap</b>	linkWrap /stdWrap	(Executed before ".wrap" and ".stdWrap".)	
<b>wrap</b>	wrap /stdWrap	(Executed after ".linkWrap" and before ".stdWrap".)	
<b>stdWrap</b>	->stdWrap	(Executed after ".linkWrap" and ".wrap".)	
<b>altText</b> <b>titleText</b>	string /stdWrap	<b>For &lt;img&gt; output only!</b>  If no alttext is specified, it will use an empty alttext.	
<b>emptyTitleHandling</b>	string /stdWrap	Value can be "keepEmpty" to preserve an empty title attribute, or "useAlt" to use the alt attribute instead.	useAlt
<b>longdescURL</b>	string /stdWrap	<b>For &lt;img&gt; output only!</b>  "longdesc" attribute (URL pointing to document with extensive details about image).	

[tsref:(cObject).FILE]

### Example:

In this example a page is defined, but the content between the body-tags comes directly from the file "gs.html":

```
page.10 = FILE
page.10.file = fileadmin/gs/gs.html
```

## IMAGE

Returns an image tag with the image file defined in the property "file" and processed according to the properties set.

Defined as PHP function cImage() in /typo3/sysex/cms/tslib/class.tslib\_content.php.

The array \$GLOBALS['TSFE']->lastImageInfo is set with the info-array of the returning image (if any) and contains width, height and so on.

Property:	Data type:	Description:	Default:
<b>file</b>	imgResource		
<b>params</b>	<IMG>-params /stdWrap		
<b>border</b>	integer	Value of the "border" attribute of the image tag.	0
<b>altText</b> <b>titleText</b>  (alttext)	string /stdWrap	If no alt text is specified, an empty alt text will be used.  ("alttext" is the old spelling of this attribute. It was deprecated since TYPO3 4.3 and was used only if "altText" did not specify a value or properties. In TYPO3 4.6 "alttext" has been removed.)	
<b>emptyTitleHandling</b>	string /stdWrap	Value can be "keepEmpty" to preserve an empty title attribute, or "useAlt" to use the alt attribute instead.	useAlt
<b>longdescURL</b>	string /stdWrap	"longdesc" attribute (URL pointing to document with extensive details about image).	
<b>linkWrap</b>	linkWrap /stdWrap	(before ".wrap")	

Property:	Data type:	Description:	Default:
<b>imageLinkWrap</b>	boolean/ ->imageLinkWrap	<b>NOTE:</b> ONLY active if linkWrap is NOT set and file is NOT GIFBUILDER (as it works with the original imagefile)	
<b>if</b>	->if	if "if" returns false the image is not shown!	
<b>wrap</b>	wrap /stdWrap		
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).IMAGE]

**Example:**

```
10 = IMAGE
10.file = toplogo*.gif
10.params = hspace=5
10.wrap = |<BR>
```

## IMG\_RESOURCE

Returns only the image-reference, possibly wrapped with stdWrap. May be used for putting background images in tables or table-rows or to import an image in your own include-scripts.

The array \$GLOBALS['TSFE']->lastImgResourceInfo is set with the info-array of the resulting image resource (if any) and contains width, height and so on.

Property:	Data type:	Description:	Default:
<b>file</b>	imgResource		
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).IMG\_RESOURCE]

## CLEARGIF

Inserts a transparent gif-file.

Property:	Data type:	Description:	Default:
<b>height</b>	<img>- data:height /stdWrap	Height of the image.	1
<b>width</b>	<img>- data:width /stdWrap	Width of the image.	1
<b>wrap</b>	wrap /stdWrap		 
<b>stdWrap</b>	->stdWrap	(Executed after ".wrap".)	

[tsref:(cObject).CLEARGIF]

**Example:**

```
20 = CLEARGIF
20.height = 20
```

## CONTENT

This object is designed to generate content by making it possible to finely select records and rendering them.

The register-key SYS\_LASTCHANGED is updated with the tstamp-field of the records selected which has a higher value than the current.

Property:	Data type:	Description:	Default:
<b>select</b>	->select	The SQL-statement is set here!	
<b>table</b>	<i>TableName</i> /stdWrap	The table, the content should come from. In standard configuration this will be "tt_content". <b>Note:</b> Allowed tables are "pages" or tables prefixed with one of these: "pages_", "tt_", "tx_", "ttx_", "fe_", "user_" or "static_".	
<b>renderObj</b>	cObject		< [tablename]
<b>slide</b>	integer /stdWrap	If set and no content element is found by the select command, then the rootLine will be traversed back until some content is found.  Possible values are "-1" (slide back up to the siteroot), "1" (only the current level) and "2" (up from one level back).  Use -1 in combination with collect.  <b>.collect (integer /stdWrap):</b> If set, all content elements found on current and parent pages will be collected. Otherwise, the sliding would stop after the first hit. Set this value to the amount of levels to collect on, or use "-1" to collect up to the siteroot. <b>.collectFuzzy (boolean /stdWrap):</b> Only useful in collect mode. If no content elements have been found for the specified depth in collect mode, traverse further until at least one match has occurred. <b>.collectReverse (boolean /stdWrap):</b> Change order of elements in collect mode. If set, elements of the current page will be at the bottom.	
<b>wrap</b>	wrap /stdWrap	Wrap the whole content-story...	
<b>stdWrap</b>	->stdWrap	(Executed after ".wrap".)	

[tsref:(cObject).CONTENT]



**Example (of the CONTENT-obj):**

```
1 = CONTENT
1.table = tt_content
1.select {
    pidInList = this
    orderBy = sorting
}
```

**Example (of record-renderObj's):**

```
// Configuration for records with the typeField-value (often "CType") set to "header"
tt_content.header.default {
    10 = TEXT
    10.field = header
    .....
}

// Configuration for records with the typeField-value (often "CType") set to "bullets"
// If field "layout" is set to "1" or "2" a special configuration is used, else default
tt_content.bullets.subTypeField = layout
tt_content.bullets.default {
    .....
}
tt_content.bullets.1 {
    .....
}
tt_content.bullets.2 {
    .....
}

// This is what happens if the typeField-value does not match any of the above
tt_content.default.default {
    .....
}
```

## RECORDS

This object is meant for displaying lists of records from a variety of tables. Contrary to the CONTENT object, it does not allow for very fine selections of records (it has no "select" property)

The register-key SYS\_LASTCHANGED is updated with the tstamp-field of the records selected which has a higher value than the current.

**NOTE:** Records with parent ids (pid's) for non-accessible pages (that is hidden, timed or access-protected pages) are normally not selected. Pages may be of any type, except recycler. Disable the check with the "dontCheckPid"-option.

Property:	Data type:	Description:	Default:
<b>source</b>	<i>records-list</i> /stdWrap	List of record-id's, optionally with prepended table-names.  <b>Example:</b> source = tt_content_34, 45, tt_links_56	
<b>tables</b>	<i>list of tables</i> /stdWrap	List of accepted tables. If any items in the ".source"-list are not prepended with a table name, the first table in this list is assumed to be the table for such records. Also table names configured in .conf are allowed.  <b>Example:</b> tables = tt_content, tt_address, tt_links conf.tx_myexttable = TEXT conf.tx_myexttable.value = Hello world  This adds the tables tt_content, tt_address, tt_links and tx_myexttable.	

Property:	Data type:	Description:	Default:
<b>conf.</b> <b>[tablename]</b>	cObject	Config-array which renders records from table <i>tablename</i>	If this is NOT defined, the rendering of the records is done with the toplevel-object [tablename] - just like the cObject, CONTENT!
<b>dontCheckPid</b>	boolean /stdWrap	Normally a record cannot be selected, if its parent page (pid) is not accessible for the website user. This option disables that check.	
<b>wrap</b>	wrap /stdWrap		
<b>stdWrap</b>	->stdWrap	(Executed after ".wrap".)	

[tsref:(cObject).RECORDS]

**Example:**

```
20 = RECORDS
20.source.field = records
20.tables = tt_address
20.conf.tt_address < tt_address.default
```

## HMENU

Generates hierarchical menus.

Property:	Data type:	Description:	Default:
<b>(1 / 2 / 3 /...)</b>	menuObj	<b>Required!</b> Defines which menuObj that should render the menu items on the various levels. 1 is the first level, 2 is the second level, 3 is the third level, 4 is ....  <b>Example:</b> temp.sidemenu = HMENU temp.sidemenu.1 = GMENU	(no menu)
<b>cache_period</b>	int	The number of seconds a menu may remain in cache. If this value is not set, the first available value of the following will be used: 1) cache_timeout of the current page 2) config.cache_period defined globally 3) 86400 (= 1 day)	
<b>entryLevel</b>	int /stdWrap	Defines at which level in the rootLine the menu should start. Default is "0" which gives us a menu of the very first pages on the site. If the value is < 0, entryLevel is chosen from "behind" in the rootLine. Thus "-1" is a menu with items from the outermost level, "-2" is the level before the outermost...	0
<b>special</b>	"directory" / "list" / "updated" / "browse" / "rootline" / "keywords" / "language"	See section "The .special property" and the according tables below.	
<b>special.value</b>	list of page-uid's /stdWrap	See above	
<b>minItems</b>	Until TYPO3 4.6: int Since TYPO3 4.7: int /stdWrap	The minimum items in the menu. If the number of pages does not reach this level, a dummy-page with the title "..." and uid=[currentpage_id] is inserted.  <b>Notice:</b> Affects all sub menus as well. To set the value for each menu level individually, set the properties in the menu objects (see "Common properties" table).	

Property:	Data type:	Description:	Default:
<b>maxItems</b>	Until TYPO3 4.6: int Since TYPO3 4.7: int /stdWrap	The maximum items in the menu. More items will be ignored.  <b>Notice:</b> Affects all sub menus as well. (See "minItems" for notice)	
<b>begin</b>	Until TYPO3 4.6: int +calc Since TYPO3 4.7: int /stdWrap +calc	The first item in the menu.  <b>Example:</b> This results in a menu, where the first two items are skipped starting with item number 3: <code>begin = 3</code>  <b>Notice:</b> Affects all sub menus as well. (See "minItems" for notice)	
<b>excludeUidList</b>	list of integers /stdWrap	This is a list of page uid's to exclude when the select statement is done. Comma-separated. You may add "current" to the list to exclude the current page.  <b>Example:</b> The pages with these uid-number will NOT be within the menu!! Additionally the current page is always excluded too. <code>excludeUidList = 34,2,current</code>	
<b>excludeDoktypes</b>	list of integers	Enter the list of page document types (doktype) to exclude from menus. By default pages that are "not in menu" (5) are excluded and those marked for backend user access only (6).	5,6
<b>includeNotInMenu</b>	boolean	If set, pages with the checkbox "Not in menu" checked will be included in menus.	
<b>alwaysActivePIDlist</b>	list of integers /stdWrap	This is a list of page UID numbers that will always be regarded as active menu items and thereby automatically opened regardless of the rootline.	
<b>protectLvar</b>	boolean / keyword	If set, then for each page in the menu it will be checked if an Alternative Page Language record for the language defined in "config.sys_language_uid" (typically defined via &L) exists for the page. If that is not the case and the pages "Localization settings" have the "Hide page if no translation for current language exists" flag set, then the menu item will link to a non accessible page that will yield an error page to the user. Setting this option will prevent that situation by simply adding "&L=0" for such pages, meaning that they will switch to the default language rather than keeping the current language. The check is only carried out if a translation is requested ("config.sys_language_uid" is not zero).  <b>Keyword: "all"</b> When set to "all" the same check is carried out but it will not look if "Hide page if no translation for current language exists" is set - it always reverts to default language if no translation is found.  For these options to make sense, they should only be used when "config.sys_language_mode" is not set to "content_fallback".	
<b>addQueryString</b>	string	<i>see typolink.addQueryString</i>  <b>Notice:</b> This works only for <i>special=language</i> .	
<b>if</b>	->if	If "if" returns false, the menu is not generated	
<b>wrap</b>	wrap /stdWrap		
<b>stdWrap</b>	->stdWrap	(Executed after ".wrap".)	

[tsref:(cObject).HMENU]

**Example:**

```
temp.sidemenu = HMENU
temp.sidemenu.entryLevel = 1
temp.sidemenu.1 = TMENU
temp.sidemenu.1 {
    target = page
    NO.afterImg = media/bullets/dots2.gif |*||*| _
    NO.afterImgTagParams = hspace="4"
    NO.linkWrap = {$fontTag}
    NO.ATagBeforeWrap = 1

    ACT < .NO
    ACT = 1
    ACT.linkWrap = <b>{$fontTag}</b>
}
```

## The .special property

This property makes it possible to create menus that are not strictly reflecting the current page-structure, but rather creating menus with links to pages like "next/previous", "last modified", "pages in a certain page" and so on.

**Note:** .entryLevel generally is not supported together with the .special property! The only exception is special.keywords.

Also be aware that this property selects pages for the first level in the menu. Submenus by menuObjects 2+ will be created as usual.

### **special.directory**

A HMENU of type special = directory lets you create a menu listing the subpages of one or more parent pages. The parent pages are defined in the property ".value". It is usually used for sitemaps.

Mount pages are supported.

Property:	Data type:	Description:	Default:
<b>value</b>	list of page ids /stdWrap	This will generate a menu of all pages with pid = 35 and pid = 56. 20 = HMENU 20.special = directory 20.special.value = 35, 56	current page id

[tsref:(cObject).HMENU.special.directory]

### **special.list**

A HMENU of type special = list lets you create a menu that lists the pages you define in the property ".value".

Mount pages are supported.

Property:	Data type:	Description:	Default:
<b>value</b>	list of page ids /stdWrap	This will generate a menu with the two pages (uid=35 and uid=56) listed:  20 = HMENU 20.special = list 20.special.value = 35, 56  If .value is not set, the default uid is 0, so that only your homepage will be listed.	0

[tsref:(cObject).HMENU.special.list]

### **special.updated**

An HMENU with the property special = updated will create a menu of the most recently updated pages.

**A note on ordering:** The sorting menu is by default done in reverse order (desc) with the field specified by "mode", but setting "alternativeSortingField" for the menu object (e.g. TMENU or GMENU, see later) will override that.

Mount pages are supported.

Property:	Data type:	Description:	Default:
<b>value</b>	list of page ids /stdWrap	This will generate a menu of the most recently updated pages from the branches in the tree starting with the uid's (uid=35 and uid=56) listed. <code>20 = HMENU</code> <code>20.special = updated</code> <code>20.special.value = 35, 56</code>	
<b>mode</b>	string	The field in the database which should be used to get the information about the last update from.  The following values are possible: - <b>SYS_LASTCHANGED</b> : Is updated to the youngest tstamp of the records on the page when a page is generated. - <b>crdate</b> : Uses the "crdate"-field of the pagerecord. - <b>tstamp</b> : Uses the "tstamp"-field of the pagerecord, which is set automatically when the record is changed. - <b>manual</b> or <b>lastUpdated</b> : Use the field "lastUpdated", which can be set manually in the page-record. - <b>starttime</b> : Uses the starttime field.  Fields with empty values are generally not selected.	SYS_LASTCHANGED
<b>depth</b>	int	Defines the tree depth. The allowed range is 1-20. A depth of 1 means only the start id, depth of 2 means start-id + first level. <b>Notice</b> : "depth" is relative to "beginAtLevel".	20
<b>beginAtLevel</b>	int	Determines starting level for the page trees generated based on .value and .depth.  0 is default and includes the start id. 1 starts with the first row of subpages, 2 starts with the second row of subpages.  <b>Notice</b> : "depth" is relative to this property.	0
<b>maxAge</b>	int (seconds) +calc	Pages with update-dates older than the current time minus this number of seconds will not be shown in the menu no matter what. Default is "not used". You may use +*/ for calculations.	
<b>limit</b>	int	Maximal number of items in the menu. Default is 10, max is 100.	10
<b>excludeNoSearchPages</b>	boolean	If set, pages marked "No search" are not included.	0

[tsref:(cObject).HMENU.special.updated]

### Example for special = updated:

The following example will generate a menu of the most recently updated pages from the branches in the tree starting with the uid's (uid=35 and uid=56) listed. Furthermore the field "tstamp" is used (default is SYS\_LASTCHANGED) and the tree depth is 2 levels. Also a maximum of 8 pages will be shown and they must have been updated within the last three days (3600\*24\*3):

```
20 = HMENU
20.special = updated
20.special.value = 35, 56
20.special {
    mode = tstamp
    depth = 2
    maxAge = 3600*24*3
    limit = 8
```

```
}

```

### ***special.rootline***

The path of pages from the current page to the root page of the page tree is called "rootline".

A rootline menu is a menu which shows you these pages one by one in their hierarchical order.

An HMENU with the property `special = rootline` creates a rootline menu (also known as "breadcrumb trail") that could look like this:

Page level 1 > Page level 2 > Page level 3 > *Current page*

Such a click path facilitates the user's orientation on the website and makes navigation to a certain page level easier.

Mount pages are supported.

Property:	Data type:	Description:	Default:
<b>range</b>	string /stdWrap	[begin-level]   [end-level] (same way as you reference the .entryLevel for an HMENU). The following example will start at level 1 and not show the page the user is currently on: <pre>temp.breadcrumbs = HMENU temp.breadcrumbs.special = rootline temp.breadcrumbs.special.range = 1 -2</pre>	
<b>reverseOrder</b>	boolean	If set to true, the order of the rootline menu elements will be reversed.	false
<b>targets.[level number]</b>	string	For framesets. You can set a default target and a target for each level by using the level number as sub-property.  <b>Example:</b> Here the links to pages on level 3 will have target="page", while all other levels will have target="_top" as defined for the TMENU property .target.  <pre>page.2 = HMENU page.2.special = rootline page.2.special.range = 1 -2 page.2.special.targets.3 = page page.2.1 = TMENU page.2.1.target = _top page.2.1.wrap = &lt;HR&gt;   &lt;HR&gt; page.2.1.NO {     linkWrap =   &gt; }</pre>	

[tsref:(cObject).HMENU.special.rootline]

### **Example for special = rootline:**

The following example will generate an accessible rootline menu: It will be wrapped as an unordered list. The first page in the menu is the page on level 1, that is one level below the root page of the website. The last page in the menu is the current page.

After each link there is an image, which could contain a small arrow.

The current page is not linked, but wrapped in em tags. It does not have the image appended.

```
20 = HMENU
20.wrap = <ul>|</ul>
20.special = rootline
20.special.range = 1|-1

20 {
    1 = TMENU
    1.noBlur = 1
}
```

```

1.NO.wrapItemAndSub = <li>|</li>
1.NO.ATagTitle.field = description // subtitle // title
1.NO.afterImg = fileadmin/arrow.jpg

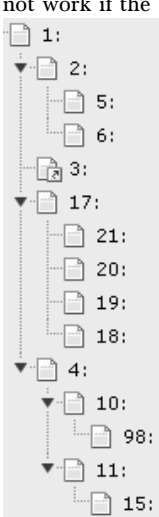
1.CUR = 1
1.CUR < .1.NO
1.CUR.doNotLinkIt = 1
1.CUR.wrapItemAndSub = <li><em>|</em></li>
1.CUR.afterImg >
}

```

### special.browse

**Warning:** Mount pages are not supported!

This menu contains pages which give your user the possibility to browse to the previous page, to the next page, to a page with the table of contents and so on. The menu is built of items given by a list from the property "items".

Property:	Data type:	Description:	Default:
<b>value</b>	int /stdWrap	Default is the current page id. Seldom you might want to override this value with another page-uid which will then act as the base point for the menu and the predefined items.	current page id
<b>items</b>	list of item names separated by " "	<p>Each element in the list (separated by " ") is either a reserved item name (see list) with a predefined function, or a user-defined name which you can assign a link to any page. Note that the current page cannot be the root-page of a site.</p> <p><i>Reserved item names:</i></p> <p><b>next / prev:</b> Links to the next page / the previous page. Next and previous pages are from the same "pid" as the current page id (or "value") - that is the next item in a menu with the current page. Also referred to as current level.</p> <p>If ".prevnextToSection" is set then next/prev will link to the first page of the next section / to the last page of the previous section, too.</p> <p><b>nextsection / prevsection:</b> Links to the next section / the previous section. A section is defined as the subpages of a page on the same level as the parent (pid) page of the current page. Will not work if the parent page of the current page is the root page of the site.</p> <p><b>nextsection_last / prevsection_last:</b> Where nextsection/prevsection links to the first page in a section, these link to the last pages. If there is only one page in the section that will be both first and last. Will not work if the parent page of the current page is the root page of the site.</p> <p><b>first / last:</b> First / last page on the current level. If there is only one page on the current level that page will be both first and last.</p> <p><b>up:</b> Links to the parent (pid) page of the current page (up 1 level). Will always be available.</p> <p><b>index:</b> Links to the parent of the parent page of the current page (up 2 levels). May not be available, if that page is out of the rootline.</p> <p><b>Examples:</b></p> <p>If id=20 is the current page then:  21= prev and first, 19 = next, 18 = last, 17 = up, 1=index, 10 = nextsection, 11 = nextsection_last</p> 	

Property:	Data type:	Description:	Default:
		<p>prevsection and prevsection_last is not present because id=3 has no subpages!</p> <p><b>TypoScript (only "browse"-part, needs also TMENU/GMENU):</b></p> <pre>xxx = HMENU xxx.special = browse xxx.special {     items = index up next prev     items.prevnextToSection = 1     index.target = _blank     index.fields.title = INDEX     index.uid = 8 }</pre>	
<b>items.prevnextToSection</b>	boolean	If set, the "prev" and "next" navigation will jump to the next section when it reaches the end of pages in the current section. That way "prev" and "next" will also link to the first page of the next section / to the last page of the previous section.	
<b>[itemname].target</b>	string	Optional/alternative target of the item.	
<b>[itemname].uid</b>	int	(uid of page) - optional/alternative page-uid to link to.	
<b>[itemname].fields.[field name]</b>	string	<p>Override field "field name" in pagerecord.</p> <p><b>Example:</b> This gives the link to the previous page the linktext "« zurück".</p> <pre>prev.fields.title = « zurück</pre>	

[tsref:(cObject).HMENU.special.browse]

### **special.keywords**

Makes a menu of pages, which contain one or more keywords also found on the current page.

**Ordering** is by default done in reverse order (desc) with the field specified by "mode", but setting "alternativeSortingField" for the menu object (e.g. for a GMENU, see later) will override that.

Mount pages are supported.

Property:	Data type:	Description:	Default:
<b>value</b>	int /stdWrap	<p>Page for which keywords to find similar pages</p> <p><b>Example:</b></p> <pre>20 = HMENU 20.special = keywords 20.special {     value.data = TSFE:id     entryLevel = 1     mode = manual } 20.1 = TMENU 20.1.NO {     ... }</pre>	
<b>mode</b>	string	<p>Which field in the pages-table to use for sorting.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> <li>- <b>SYS_LASTCHANGED</b>: Is updated to the youngest tstamp of the records on the page when a page is generated.</li> <li>- <b>manual</b> or <b>lastUpdated</b>: Use the field "lastUpdated", which can be set manually in the page-record.</li> <li>- <b>tstamp</b>: Uses the "tstamp"-field of the pagerecord, which is set automatically when the record is changed.</li> <li>- <b>crdate</b>: Uses the "crdate"-field of the pagerecord.</li> </ul>	SYS_LASTCHANGED



Property:	Data type:	Description:	Default:
		- <b>starttime</b> : Uses the starttime field.	
<b>entryLevel</b>	int	Where in the rootline the search begins. <i>See property entryLevel in the section "HMENU" above.</i>	
<b>depth</b>	int	(same as in section "special.updated")	20
<b>limit</b>	int	(same as in section "special.updated")	10
<b>excludeNoSearchPages</b>	boolean	(same as in section "special.updated")	
<b>begin</b>	boolean	(same as in section "special.updated")	
<b>setKeywords</b>	string /stdWrap	Lets you define the keywords manually by defining them as a comma-separated list. If this property is defined, it overrides the default, which is the keywords of the current page.	
<b>keywordsField</b>	string	Defines the field in the pages-table in which to search for the keywords. Default is the field name "keyword". No check is done to see if the field you enter here exists, so enter an existing field, OK?!	keywords
<b>keywordsField.sourceField</b>	string	Defines the field from the current page from which to take the keywords being matched. The default is "keyword". (Notice that ".keywordsField" is only setting the page-record field to <i>search in</i> !)	keywords

[tsref:(cObject).HMENU.special.keywords]

### special.language

Creates a language selector menu. Typically this is made as a menu with flags for each language a page is translated to and when the user clicks any element the same page id is hit but with a change to the "&L" parameter in the URL.

The "language" type will create menu items based on the current page record but with the language record for each language overlaid if available. The items all link to the current page id and only "&L" is changed.

Note on item states:

When "TSFE->sys\_language\_uid" matches the sys\_language uid for an element the state is set to "ACT", otherwise "NO". However, if a page is not available due to the pages "Localization settings" (which can disable translations) or if no Alternative Page Language record was found (can be disabled with ".normalWhenNoLanguage", see below) the state is set to "USERDEF1" for non-active items and "USERDEF2" for active items. So in total there are four states to create designs for. It is recommended to disable the link on menu items rendered with "USERDEF1" and "USERDEF2" in this case since they are disabled exactly because a page in that language does not exist and might even issue an error if tried accessed (depending on site configuration).

Property:	Data type:	Description:	Default:
<b>value</b>	comma list of sys_language uids /stdWrap	The number of elements in this list determines the number of menu items.	
<b>normalWhenNoLanguage</b>	boolean	If set to 1 the button for a language will be rendered as a non-disabled button even if no translation is found for the language.	

[tsref:(cObject).HMENU.special.language]

### Example:

Creates a language menu with flags (notice that some lines break):



```
lib.langMenu = HMENU
lib.langMenu.special = language
```

```
lib.langMenu.special.value = 0,1,2
lib.langMenu.1 = GMENU
lib.langMenu.1.NO {
    XY = [5.w]+4, [5.h]+4
    backColor = white
    5 = IMAGE
    5.file = typo3/sysex/cms/tslib/media/flags/flag_uk.gif ||
typo3/sysex/cms/tslib/media/flags/flag_fr.gif ||
typo3/sysex/cms/tslib/media/flags/flag_es.gif
    5.offset = 2,2
}

lib.langMenu.1.ACT < lib.langMenu.1.NO
lib.langMenu.1.ACT = 1
lib.langMenu.1.ACT.backColor = black

lib.langMenu.1.USERDEF1 < lib.langMenu.1.NO
lib.langMenu.1.USERDEF1 = 1
lib.langMenu.1.USERDEF1.5.file = typo3/sysex/cms/tslib/media/flags/flag_uk_d.gif ||
typo3/sysex/cms/tslib/media/flags/flag_fr_d.gif ||
typo3/sysex/cms/tslib/media/flags/flag_es_d.gif
lib.langMenu.1.USERDEF1.noLink = 1
```

### ***special.userdefined***

Lets you write your own little PHP-script that generates the array of menu items.

Property:	Data type:	Description:	Default:
<b>file</b>	resource	Filename of the php-file to include. (Just like cObject PHP_SCRIPT)	
<b>[any other key]</b>		Your own variables to your script. They are all accessible in the array \$conf in your script.	

[tsref:(cObject).HMENU.special.userdefined]

**Note:** The special type "userdefined" has been removed in TYPO3 4.6. Use the special type "userfunction" instead.

### **How-to:**

You must populate an array called \$menuItemsArray with page-records of the menu items you want to be in the menu.

It works like this:

```
$menuItemsArray[] = pageRow1;
$menuItemsArray[] = pageRow2;
$menuItemsArray[] = pageRow3;
...
```

A "pageRow" is a record from the table "pages" with all fields selected (SELECT \* FROM...)

If you create fake page rows, make sure to add at least "title" and "uid" field values.

Notice:

If you work with mount-points you can set the MP param which should be set for the page by setting the internal field "\_MP\_PARAM" in the page-record (xxx-xxx).

Overriding URLs:

You can also use the internal field "\_OVERRIDE\_HREF" to set a custom href-value (eg. "http://www.typo3.org") which will in any case be used rather than a link to the page that the page otherwise might represent. If you use "\_OVERRIDE\_HREF" then "\_OVERRIDE\_TARGET" can be used to override the target value as well (See example below).

Other reserved keys:

"\_ADD\_GETVARS" can be used to add get parameters to the URL, eg. "&L=xxx".

"\_SAFE" can be used to protect the element to make sure it is not filtered out for any reason.

Creating submenus:

You can create submenus for the next level easily by just adding an array of menu items in the internal field "\_SUB\_MENU" (See example below).

Presetting element state

If you would like to preset an element to be recognized as a SPC, IFSUB, ACT, CUR or USR mode item, you can do so by specifying one of these values in the key "ITEM\_STATE" of the page record. This setting will override the natural state-evaluation.

### *special.userfunction*

Calls a user function/method in class which should (similar to how "userdefined" worked above) return an array with page records for the menu.

Property:	Data type:	Description:	Default:
<b>userFunc</b>	string	Name of the function	

[tsref:(cObject).HMENU.special.userfunction]

### Example: Creating hierarchical menus of custom links

By default the HMENU object is designed to create menus from pages in TYPO3. Such pages are represented by their page-record contents. Usually the "title" field is used for the title and the "uid" field is used to create a link to that page in the menu.

However the HMENU and sub-menu objects are so powerful that it would be very useful to use these objects for creating menus of links which does not relate to pages in TYPO3 by their ids. This could be a menu reflecting a menu structure of a plugin where each link might link to the same page id in TYPO3 but where the difference would be in some parameter value.

First, this listing creates a menu in three levels where the first two are graphical items:

```

0: # *****
1: # MENU LEFT
2: # *****
3: lib.leftmenu.20 = HMENU
4: lib.leftmenu.20.special = userfunction
5: lib.leftmenu.20.special.userFunc = user_3dsplm_pi2->makeMenuArray
6: lib.leftmenu.20.1 = GMENU
7: lib.leftmenu.20.1.NO {
8:   wrap = <tr><td>|</td></tr><tr><td class="bckgdgrey1" height="1"></td></tr>
9:   XY = 163,19
10:   backColor = white
11:   10 = TEXT
12:   10.text.field = title
13:   10.text.case = upper
14:   10.fontColor = red
15:   10.fontFile = fileadmin/fonts/ARIALNB.TTF
16:   10.niceText = 1
17:   10.offset = 14,12
18:   10.fontSize = 10
19: }
20: lib.leftmenu.20.2 = GMENU
21: lib.leftmenu.20.2.wrap = | <tr><td class="bckgdwhite" height="4"></td></tr><tr><td
class="bckgdgrey1" height="1"></td></tr>
22: lib.leftmenu.20.2.NO {
23:   wrap = <tr><td class="bckgdwhite" height="4"></td></tr><tr><td>|</td></tr>
24:   XY = 163,16
25:   backColor = white
26:   10 = TEXT
27:   10.text.field = title
28:   10.text.case = upper
29:   10.fontColor = #666666
30:   10.fontFile = fileadmin/fonts/ARIALNB.TTF
31:   10.niceText = 1
32:   10.offset = 14,12
33:   10.fontSize = 11
34: }
35: lib.leftmenu.20.2.R0 < lib.leftmenu.20.2.NO
36: lib.leftmenu.20.2.R0 = 1

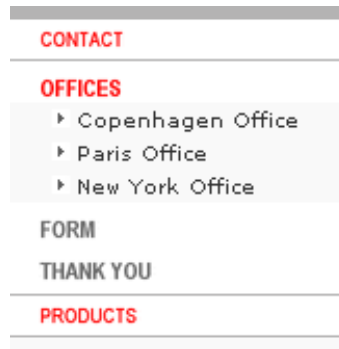
```

```

37: lib.leftmenu.20.2.R0.backgroundColor = #eeeeee
38: lib.leftmenu.20.2.ACT < lib.leftmenu.20.2.NO
39: lib.leftmenu.20.2.ACT = 1
40: lib.leftmenu.20.2.ACT.10.fontColor = red
41: lib.leftmenu.20.3 = TMENU
42: lib.leftmenu.20.3.NO {
43:     allWrap = <tr><td>|</td></tr>
44:     linkWrap (
45:         <table border="0" cellpadding="0" cellspacing="0" style="margin: 2px; 0px; 2px;
0px;">
46:             <tr>
47:                 <td></td>
48:                 <td></td>
49:                 <td>|</td>
50:             </tr>
51:         </table>
52:     )
53: }

```

The menu looks like this on a web page:



The TypoScript code above generates this menu, but the items do not link straight to pages as usual. This is because the *whole* menu is generated from this array, which was returned from the function "menuMenuArray" called in TypoScript line 4+5

```

1:     function makeMenuArray($content, $conf) {
2:         return array(
3:             array(
4:                 'title' => 'Contact',
5:                 '_OVERRIDE_HREF' => 'index.php?id=10',
6:                 '_SUB_MENU' => array(
7:                     array(
8:                         'title' => 'Offices',
9:                         '_OVERRIDE_HREF' => 'index.php?id=11',
10:                        '_OVERRIDE_TARGET' => '_top',
11:                        '_ITEM_STATE' => 'ACT',
12:                        '_SUB_MENU' => array(
13:                            array(
14:                                'title' => 'Copenhagen Office',
15:                                '_OVERRIDE_HREF' => 'index.php?id=11&officeId=cph',
16:                            ),
17:                            array(
18:                                'title' => 'Paris Office',
19:                                '_OVERRIDE_HREF' => 'index.php?id=11&officeId=paris',
20:                            ),
21:                            array(
22:                                'title' => 'New York Office',
23:                                '_OVERRIDE_HREF' => 'http://www.newyork-office.com',
24:                                '_OVERRIDE_TARGET' => '_blank',
25:                            )
26:                        )
27:                    ),
28:                    array(
29:                        'title' => 'Form',
30:                        '_OVERRIDE_HREF' => 'index.php?id=10&cmd=showform',
31:                    ),
32:                    array(
33:                        'title' => 'Thank you',
34:                        '_OVERRIDE_HREF' => 'index.php?id=10&cmd=thankyou',

```

```

35:         ),
36:     ),
37: ),
38:     array(
39:         'title' => 'Products',
40:         '_OVERRIDE_HREF' => 'index.php?id=14',
41:     )
42: );
43: }

```

Notice how the array contains "fake" page-records which has *no* uid field, only a "title" and "\_OVERRIDE\_HREF" as required and some other fields as it fits.

- The first level with items "Contact" and "Products" contains "title" and "\_OVERRIDE\_HREF" fields, but "Contact" extends this by a "\_SUB\_MENU" array which contains a similar array of items.
- The first item on the second level, "Offices", contains a field called "\_OVERRIDE\_TARGET". Further the item has its state set to "ACT" which means it will render as an "active" item (you will have to calculate such stuff manually when you are not rendering a menu of real pages!). Finally there is even another sub-level of menu items.

## CTABLE

Creates a table in which you can define the content of the the various cells.

A CTABLE is a little more feature packed than the simple OTABLE. It features a content column and four surrounding columns, which may be useful for putting menus into them.

Property:	Data type:	Description:	Default:
<b>offset</b>	x,y /stdWrap	Offset from upper left corner.	0,0
<b>tm</b>	->CARRAY +TDPparams /stdWrap	TopMenu The default value of TDPparams is: valign="top". stdWrap is available for the property TDPparams.	
<b>lm</b>	->CARRAY +TDPparams /stdWrap	LeftMenu The default value of TDPparams is: valign="top". stdWrap is available for the property TDPparams.	
<b>rm</b>	->CARRAY +TDPparams /stdWrap	RightMenu The default value of TDPparams is: valign="top". stdWrap is available for the property TDPparams.	
<b>bm</b>	->CARRAY +TDPparams /stdWrap	BottomMenu The default value of TDPparams is: valign="top". stdWrap is available for the property TDPparams.	
<b>c</b>	->CARRAY +TDPparams /stdWrap	Content-cell The default value of TDPparams is: valign="top". stdWrap is available for the property TDPparams.	
<b>cMargins</b>	margins /stdWrap	Distance around the content-cell "c".	0,0,0,0
<b>cWidth</b>	pixels /stdWrap	Width of the content-cell "c".	
<b>tableParams</b>	<TABLE>-params /stdWrap	Attributes of the table tag.	border="0" cellspacing="0" cellpadding="0"
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).CTABLE]

### Example:

```

page.10 = CTABLE
page.10 {
    offset = 5, 0
    tableParams = border="0" width="400"
    cwidth = 400
    c.1 = CONTENT
    c.1.table = tt_content
    c.1.select {
        pidInList = this
    }
}

```

```

        orderBy = sorting
    }

    tm.10 < temp.sidemenu
    tm.TDParams = valign=top

    stdWrap.wrap = <div id="mytable">|</div>
}

```

## OTABLE

Creates a simple table. You can set an offset and some parameters for the table tag.

Property:	Data type:	Description:	Default:
<b>offset</b>	x,y /stdWrap	Offset from upper left corner.  <b>Note:</b> Actually the data type is "x,y,r,b,w,h" and stdWrap: x,y is the offset from upper left corner. r,b is the offset (margin) to right and bottom. w is the required width of the content field. h is the required height of the content field.  All measures are in pixels.	
<b>1,2,3,4...</b>	cObject		
<b>tableParams</b>	<TABLE>-params /stdWrap	Attributes of the table tag.	border="0" cellspacing="0" cellpadding="0"
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).OTABLE]

### Example:

```

top.100 = OTABLE
top.100.offset = 310,8
top.100.tableParams = border="1" cellpadding="0" cellspacing="0"
top.100.1 < temp.topmenu

```

## COLUMNS

Inserts a table with several columns. Size and styling of the table tag can be defined with the according options.

Property:	Data type:	Description:	Default:
<b>tableParams</b>	<TABLE>-params /stdWrap	Attributes of the table tag.	border="0" cellspacing="0" cellpadding="0"
<b>TDparams</b>	<TD>-params /stdWrap	Attributes of the td tags.	valign="top"
<b>rows</b>	integer (Range: 2-20) /stdWrap	The number of rows in the columns.	2
<b>totalWidth</b>	integer /stdWrap	The total-width of the columns+gaps.	
<b>gapWidth</b>	integer /stdWrap +optionSplit	Width of the gap between columns. 0 = no gap	
<b>gapBgCol</b>	HTML-color /stdWrap +optionSplit	Background-color for the gap-tablecells.	
<b>gapLineThickness</b>	integer /stdWrap +optionSplit	Thickness of the divider line in the gap between cells. 0 = no line	

Property:	Data type:	Description:	Default:
<b>gapLineCol</b>	HTML-color /stdWrap +optionSplit	Line color of the divider line.	black
<b>[column-number]</b> 1,2,3,4...	cObject	This is the content-object for each column!!	
<b>after</b>	cObject	This is a cObject placed after the columns-table!!	
<b>if</b>	->if	If "if" returns false, the columns are not rendered!	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).COLUMNS]

## HRULER

This object inserts a table tag, which you can use as a horizontal divider.

Property:	Data type:	Description:	Default:
<b>lineThickness</b>	integer /stdWrap	Range: 1-50	1
<b>lineColor</b>	HTML-color /stdWrap	The color of the ruler.	black
<b>spaceLeft</b>	pixels /stdWrap	Space before the line (to the left).	
<b>spaceRight</b>	pixels /stdWrap	Space after the line (to the right).	
<b>tableWidth</b>	string /stdWrap	Width of the ruler ("width" attribute in a table).	99%
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).HRULER]

## IMGTEXT

This object is designed to align images and text. This is normally used to render text/picture records from the tt\_content table.

The image(s) are placed in a table and the table is placed before, after or left/right relative to the text.

See code examples.

Property:	Data type:	Description:	Default:
<b>text</b>	->CARRAY /stdWrap	Use this to import / generate the content, that should flow around the image block.	
<b>textPos</b>	int /stdWrap	Text position: bit[0-2]: 000 = center, 001 = right, 010 = left bit[3-5]: 000 = over, 001 = under, 010 text  0 - Above: Centre 1 - Above: Right 2 - Above: Left 8 - Below: Centre 9 - Below: Right 10 - Below: Left 17 - In Text: Right 18 - In Text: Left 25 - In Text: Right (no wrap) 26 - In Text: Left (no wrap)	
<b>textMargin</b>	pixels /stdWrap	Margin between the image and the content.	
<b>textMargin_out OfText</b>	boolean	If set, the textMargin space will still be inserted even if the image is placed above or below the text. This flag is only for a kind of backwards compatibility because this "feature" was recently considered a bug and thus corrected. So if anyone has depended on this way things are	

Property:	Data type:	Description:	Default:
		done, you can compensate with this flag.	
<b>imgList</b>	<i>list of imagefiles</i> /stdWrap	List of images from ".imgPath".  <b>Example:</b> This imports the list of images from tt_content's image-field. <code>imgList.field = image</code>	
<b>imgPath</b>	path /stdWrap	Path to the images.  <b>Example:</b> "uploads/pics/"	
<b>imgMax</b>	int /stdWrap	Maximum number of images.	
<b>imgStart</b>	int /stdWrap	Start with image-number ".imgStart".	
<b>imgObjNum</b>	<i>imgObjNum</i> +optionSplit	Here you define, which IMAGE-cObjects from the array "1,2,3,4..." in this object that should render the images. "current" is set to the image-filename.  <b>Example:</b> <code>imgObjNum = 1  *  *  2</code> This would render the first two images with "1. ..." and the last image with "2. ...", provided that the ".imgList" contains 3 images.	
<b>1,2,3,4</b>	->IMAGE (cObject)	Rendering of the images. The register "IMAGE_NUM" is set with the number of image being rendered for each rendering of an image-object. Starting with zero. The image-object should not be of type GIFBUILDER!  <b>Important:</b> "file.import.current = 1" fetches the name of the images!	
<b>caption</b>	->CARRAY /stdWrap	Caption.	
<b>captionAlign</b>	align /stdWrap	Caption alignment.	default = ".textPos"
<b>captionSplit</b>	boolean	If this is set, the caption text is split by the character (or string) from ".token" , and every item is displayed under an image each in the image block.  .token = (string /stdWrap) Character to split the caption elements (default is chr(10)) .cObject = cObject, used to fetch the caption for the split .stdWrap = stdWrap properties used to render the caption.	
<b>altText</b> <b>titleText</b>	string /stdWrap	Default altText/titleText if no alternatives are provided by the ->IMAGE cObjects.  If alttext is not specified, an empty alttext will be used.	
<b>emptyTitleHandling</b>	string /stdWrap	Value can be "keepEmpty" to preserve an empty title attribute, or "useAlt" to use the alt attribute instead.	useAlt
<b>longdescURL</b>	string /stdWrap	Default longdescURL if no alternatives are provided by the ->IMAGE cObjects  "longdesc" attribute (URL pointing to document with extensive details about image).	
<b>border</b>	boolean /stdWrap	If true, a border is generated around the images.	
<b>borderCol</b>	<i>HTML-color</i> /stdWrap	Color of the border, if ".border" is set	black
<b>borderThick</b>	pixels /stdWrap	Width of the border around the pictures	1
<b>cols</b>	int /stdWrap	Columns	



Property:	Data type:	Description:	Default:
<b>rows</b>	int /stdWrap	Rows (higher priority than "cols")	
<b>noRows</b>	boolean /stdWrap	If set, the rows are not divided by a table-rows. Thus images are more nicely shown if the height differs a lot (normally the width is the same!)	
<b>noCols</b>	boolean /stdWrap	If set, the columns are not made in the table. The images are all put in one row separated by a clear giffile to space them apart. If noRows is set, noCols will be unset. They cannot be set simultaneously.	
<b>colSpace</b>	int /stdWrap	Space between columns.	
<b>rowSpace</b>	int /stdWrap	Space between rows.	
<b>spaceBelowAbove</b>	int /stdWrap	Pixel space between content and images when position of image is above or below text (but not in text)	
<b>tableStdWrap</b>	->stdWrap	This passes the final <table> code for the image block to the stdWrap function.	
<b>maxW</b>	int /stdWrap	Maximum width of the image-table. This will scale images not in the right size! Takes the number of columns into account!  <b>NOTE:</b> Works ONLY if IMAGE-obj is NOT GIFBUILDER!	
<b>maxWInText</b>	int /stdWrap	Maximum width of the image-table, if the text is wrapped around the image-table (on the left or right side). This will scale images not in the right size! Takes the number of columns into account!  <b>NOTE:</b> Works ONLY if IMAGE-obj is NOT GIFBUILDER!	50% of maxW
<b>equalH</b>	int /stdWrap	If this value is greater than zero, it will secure that images in a row has the same height. The width will be calculated. If the total width of the images raise above the "maxW"-value of the table the height for each image will be scaled down equally so that the images still have the same height but is within the limits of the totalWidth. Please note that this value will override the properties "width", "maxH", "maxW", "minW", "minH" of the IMAGE-objects generating the images. Furthermore it will override the "noRows"-property and generate a table with no columns instead!	
<b>colRelations</b>	string /stdWrap	This value defines the width-relations of the images in the columns of IMGTEXT. The syntax is "[int] : [int] : [int] : ..." for each column. If there are more image columns than figures in this value, it's ignored. If the relation between two of these figures exceeds 10, this function is ignore. It works only fully if all images are downscaled by their maxW-definition.  <b>Example:</b> If 6 images are placed in three columns and their width's are high enough to be forcibly scaled, this value will scale the images in the to be e.g. 100, 200 and 300 pixels from left to right 1 : 2 : 3	
<b>image_compression</b>	int /stdWrap	Image Compression: 0= Default 1= Don't change! (removes all parameters for the image_object!!) (adds gif-extension and color-reduction command) 10= GIF/256 11= GIF/128 12= GIF/64 13= GIF/32 14= GIF/16	

Property:	Data type:	Description:	Default:
		<p>15= GIF/8 (adds jpg-extension and quality command)  20= IM: -quality 100  21= IM: -quality 90 &lt;=&gt; Photoshop 60 (JPG/Very High)  22= IM: -quality 80 (JPG/High)  23= IM: -quality 70  24= IM: -quality 60 &lt;=&gt; Photoshop 30 (JPG/Medium)  25= IM: -quality 50  26= IM: -quality 40 (JPG/Low)  27= IM: -quality 30 &lt;=&gt; Photoshop 10  28= IM: -quality 20 (JPG/Very Low)  (adds png-extension and color-reduction command)  30= PNG/256  31= PNG/128  32= PNG/64  33= PNG/32  34= PNG/16  35= PNG/8  39= PNG</p> <p>The default ImageMagick quality seems to be 75. This equals Photoshop quality 45. Images compressed with ImageMagick with the same visual quality as a Photoshop-compressed image seem to be largely 50% greater in size!!</p> <p><b>NOTE:</b> Works ONLY if IMAGE-obj is NOT GIFBUILDER</p>	
<b>image_effects</b>	int /stdWrap	<p>Adds these commands to the parameters for the scaling. This function has no effect if "image_compression" above is set to 1!!</p> <p>1 =&gt; "-rotate 90",  2 =&gt; "-rotate 270",  3 =&gt; "-rotate 180",  10 =&gt; "-colorspace GRAY",  11 =&gt; "-sharpen 70",  20 =&gt; "-normalize",  23 =&gt; "-contrast",  25 =&gt; "-gamma 1.3",  26 =&gt; "-gamma 0.8"</p> <p><b>NOTE:</b> Works ONLY if IMAGE-obj is NOT GIFBUILDER</p>	
<b>image_frames</b>	Array + .key /stdWrap	<p><b>Frames:</b>  .key points to the frame used.</p> <p>"image_frames.x" is imgResource-mask (".m")properties which will override to the [imgResource].m properties of the imageObjects. This is used to mask the images into a frame. See how it's done in the default configuration and IMGTEXT in the static_template-table.</p> <p><b>Example:</b></p> <pre> 1 {     mask = media/uploads/darkroom1_mask.jpg     bgImg = GIFBUILDER     bgImg {         XY = 100,100         backColor = {\$bgCol}     }     bottomImg = GIFBUILDER     bottomImg {         XY = 100,100         backColor = black     }     bottomImg_mask = media/uploads/darkroom1_bottom.jpg } </pre>	

Property:	Data type:	Description:	Default:
		<b>NOTE:</b> This cancels the jpg-quality settings sent as ordinary ".params" to the imgResource. In addition the output of this operation will always be jpg or gif! <b>NOTE:</b> Works ONLY if IMAGE-obj is NOT GIFBUILDER	
<b>editIcons</b>	string	(See stdWrap.editIcons)	
<b>noStretchAndMarginCells</b>	boolean /stdWrap	If set (1), the cells used to add left and right margins plus stretch out the table will not be added. You will loose the ability to set margins for the object if entered "in text". So it's not recommended, but it has been requested by some people for reasons.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).IMGTEXT]

**Example:**

```

tt_content.textpic.default {
    5 = IMGTEXT
    5 {
        text < tt_content.text.default
        imgList.field = image
        textPos.field = imageorient
        imgPath = uploads/pics/
        imgObjNum = 1
        1 {
            file.import.current = 1
            file.width.field = imagewidth
            imageLinkWrap = 1
            imageLinkWrap {
                bodyTag = <BODY bgColor=black>
                wrap = <A href="javascript:close();"> | </A>
                width = 800m
                height = 600m
                JSwindow = 1
                JSwindow.newWindow = 1
                JSwindow.expand = 17,20
            }
        }
        maxW = 450
        maxWinText = 300
        cols.field = imagecols
        border.field = imageborder
        caption {
            1 = TEXT
            1.field = imagecaption
            1.wrap = <font size="1"> |</font>
            1.wrap2 = {$cBodyTextWrap}
        }
        borderThick = 2
        colSpace = 10
        rowSpace = 10
        textMargin = 10
    }
    30 = TEXT
    30.value = <br>
}

```

## CASE

This is a very flexible object whose rendering can vary depending on a given key. The principle is similar to that of the "switch" construct in PHP.

The "key" property is expected to match one of the values found in the "Array". If none is found, the "default" property will be used. Any string can be used as value in the "Array" except for those that match another property. So the forbidden values are: "setCurrent", "key", "stdWrap" and "if". And of course, "default" has a special meaning.

Property:	Data type:	Description:	Default:
<b>setCurrent</b>	string /stdWrap	Sets the "current"-value.	
<b>key</b>	string /stdWrap	This is used to define the source of the value that will be matched against the values of "Array". It will generally not be a simple string, but use its stdWrap properties to retrieve a dynamic value from some specific source, typically a field of the current record (see example below).	default
<b>default</b>	cObject	Defines the rendering for all values of "key" that don't match any of the values of "Array".	
<b>Array...</b>	cObject	Defines the rendering for a number of values.	
<b>stdWrap</b>	->stdWrap	stdWrap around any object that was rendered no matter what the "key" value is.	
<b>if</b>	->if	If "if" returns false, nothing is returned.	

[tsref:(cObject).CASE]

### Example:

This example chooses between two different renderings of some content depending on whether the field "layout" is "1" or not ("default"). The result is in either case wrapped with "<br />". If the field "header" turns out not to be set ("false") an empty string is returned anyway.

```
stuff = CASE
stuff.key.field = layout
stuff.if.isTrue.field = header
stuff.stdWrap.wrap = |<br />

stuff.default = TEXT
stuff.default {
    ....
}
stuff.1 = TEXT
stuff.1 {
    ....
}
```

## LOAD\_REGISTER

This provides a way to load the array \$GLOBALS['TSFE']->register[] with values. It doesn't return anything! The usefulness of this is, that some predefined configurations (like the page-content) can be used in various places but use different values as the values of the register can change during the page-rendering.

Property:	Data type:	Description:	Default:
<b>Array...</b> <b>[field name]</b>	string /stdWrap	<b>Example:</b> (This sets "contentWidth", "label" and "head")  <pre>page.27 = LOAD_REGISTER page.27 {     contentWidth = 500      label.field = header      head = some text     head.wrap = &lt;b&gt;   &lt;/b&gt; }</pre>	

[tsref:(cObject).LOAD\_REGISTER]

## RESTORE\_REGISTER

This unsets the latest changes in the register-array as set by LOAD\_REGISTER.

Internally this works like a stack where the original register is saved when LOAD\_REGISTER is called. Then a RESTORE\_REGISTER cObject is called the last element is pulled of that stack the register is

replaced with it.

RESTORE\_REGISTER has no properties.

## FORM



### Note

The following only applies, if the system extension "form" (which comes with TYPO3 since version 4.6) is **not** installed. If it is, things work as described in the documentation inside the system extension.

This object provides a way to create forms:

```
textarea: Label | [* = required][field name =] textarea[,cols,rows,"wrap= [eg. "OFF"]" ] |
[defaulttdata] | Special evaluation configuration (see note below)
input:      Label | [* = required][field name =] input[,size,max] | [defaulttdata] |
Special evaluation configuration (see note below)
password:   Label | [* = required][field name =] input[,size,max] | [defaulttdata]
file:       Label | [* = required][field name (*1)=] file[,size]
check:      Label | [* = required][field name =]check | [checked=1]
select:     Label | [* = required][field name =]select[,size (int/"auto"),
"m"=multiple] | label [=value] , ...
radio:      Label | [* = required][field name =]radio | label [=value] , ...
hidden:     |[[field name =]hidden | value
submit:     Label |[field name =]submit | Caption
reset:      Label |[field name =]reset | Caption
label:      Label | label | Label value
property:   [Internal, see below]
```

### Preselected item with type "select" and "radio":

This is an example, where "Brown" is the preselected item of a selector box:

```
Haircolor: | *haircolor=select| Blue=blue , Red=red , *Brown=brown
```

You can enter multiple items to be preselected by placing an asterisk in front of each preselected item.

### Property override:

This can be done with the following properties from the table below:

type, locationData, goodMess, badMess, emailMess

syntax:

```
|[property] =property | value
```

(\*1) (field name for files)

In order for files to be attached the mails, you must use the field names:

*attachment, attachment1, ... , attachment10*

### Displaying the form:

**You must set the property "layout".** If you do not set it, the form will not be rendered! For more information see the example and the table below.

#### Example:

```
temp.mailform = FORM
temp.mailform {

    dataArray {
        10.label = Name:
        10.type = name=input

        20.label = Nachricht:
        20.type = nachricht=textarea,40,10

        100.type = submit=submit
        100.value = Absenden
    }
}
```

```

}
recipient = info@example.com
layout = <div class="some-class">###LABEL### ###FIELD###</div>
}

```

## Correct return-email:

In order for the mails to be attached with the email address of the people that submits the mails, please use the field name "email", e.g:

```
Email: | *email=input |
```

## Special evaluation

By prefixing a "\*" before the field name of most types you can have the value of the field required. The check is done in JavaScript; It will only submit the form if this field is filled in.

Alternatively you can evaluate a field value against a regular expression or as an email address for certain types (textarea, password, input).

This is done by specifying the "Special evaluation configuration" for those types as part 4 in the configuration line (see examples above).

The special evaluation types are divided by a semicolon (":").

The first part defines the evaluation **keyword**. Current options are "EREG" (for regular expression) and "EMAIL" (for evaluation to an email address).

If the "EREG" keyword is specified the 2<sup>nd</sup> and 3<sup>rd</sup> parts are error message and regular expression respectively.

### Examples:

```

Your address: | address=textarea,40,10 | | EREG : You can only enter the characters A to
Z : ^[a-zA-Z]*$
Your email: | *email=input | | EMAIL

```

Property:	Data type:	Description:	Default:
<b>data</b>	string /stdWrap	This is the data that sets up the form. See above. " " can be used instead of line breaks	
<b>dataArray</b>	[array of form elements]	<p>This is an alternative way to define the form-fields. Instead of using the syntax with vertical separator bars suggested by the .data property, you can define the elements in regular TypoScript style arrays.</p> <p>.dataArray is <i>added</i> to the input in .data if any.</p> <p>Every entry in the dataArray is numeric and has three main properties, <i>label</i>, <i>type</i>, <i>value</i> and <i>required</i>. All of them have stdWrap properties.</p> <p>There is an alternative property to .value, which is .valueArray. This is also an array in the same style with numeric entries which has properties <i>label</i>, <i>value</i> and <i>selected</i>. All three of these properties have stdWrap properties.</p> <p><b>Example:</b></p> <pre> dataArray {   10.label = Name:   10.type = name=input   10.value = [Enter name]   10.required = 1   20.label = Eyecolor   20.type = eyecolor=select   20.valueArray {     10.label = Blue     10.value = 1     20.label = Red     20.value = 2     20.selected = 1   }   40.type = submit=submit </pre>	

Property:	Data type:	Description:	Default:
		<pre> 40.value = Submit } </pre> <p>This is the same as this line in the .data property:</p> <pre> Name:   *name=input   [Enter name] Eyecolor:   eyecolor=select   Blue=1, *Red=2   submit=submit   Submit </pre> <p><b>Why do it this way?</b> Good question, but doing it this way has a tremendous advantage, because labels are all separated from the codes. In addition it's much easier to pull out or insert new elements in the form.</p> <p>Inserting an email-field after the name field would be like this:</p> <pre> dataArray { 15.label = Email: 15.type = input 15.value = your@email.com 15.specialEval = EMAIL } </pre> <p>Or translating the form to danish (setting config.language to 'dk'):</p> <pre> dataArray { 10.label.lang.dk = Navn: 10.value.lang.dk = [Indtast dit navn] 20.label.lang.dk = Øjenfarve 20.valueArray { 10.label.lang.dk = Blå 20.label.lang.dk = Rød } 40.value.lang.dk = Send } </pre>	
<b>radioWrap</b>	->stdWrap	Wraps the labels for radio buttons.	
<b>radioWrap.accessibilityWrap</b>	wrap /stdWrap	Defines how radio buttons are wrapped when accessibility mode is turned on (see below "accessibility" property).	<fieldset###RADIO_FIELD_ID###><legend>###RADIO_GROUP_LABEL###</legend></fieldset>
<b>radioInputWrap</b>	->stdWrap	Wraps the input element and label of a radio button.	
<b>type</b>	integer, string	<p>Type (action="" of the form):</p> <p><b>Integer:</b> this is regarded to be a page in TYPO3</p> <p><b>String:</b> this is regarded to be a normal URL (e.g. "formmail.php" or "fe_tce_db.php")</p> <p><b>Empty:</b> the current page is chosen.</p> <p><b>NOTE:</b> If type is integer/empty the form will be submitted to a page in TYPO3 and if this page has a value for target/no_cache, then this will be used instead of the default target/no_cache below.</p> <p><b>NOTE:</b> If the redirect-value is set, the redirect-target overrides the target set by the action-url</p> <p><b>NOTE:</b> May be overridden by the property override feature of the formdata (see above)</p>	
<b>target</b>	target /stdWrap	Default target of the form.	

Property:	Data type:	Description:	Default:
<b>method</b>	form-method /stdWrap	<b>Example:</b> GET	POST
<b>no_cache</b>	string /stdWrap	Default no_cache-option.	
<b>noValueInsert</b>	boolean /stdWrap	By default values that are submitted to the same page (and thereby same form, e.g. at search forms) are re-inserted in the form instead of any default-data that might be set up. This, however, applies ONLY if the "no_cache=1" is set! (a page being cached may not include user-specific defaults in the fields of course...) If you set this flag, "noValueInsert", the content will always be the default content.	
<b>compensateFieldWidth</b>	double /stdWrap	Overriding option to the config-value of the same name. See "CONFIG" above.	
<b>locationData</b>	boolean / string /stdWrap	If this value is true, then a hidden-field called "locationData" is added to the form. This field will be loaded with a value like this: [page id]:[current record table]:[current record id] For example, if a formfield is inserted on page with uid = "100", as a page-content item from the table "tt_content" with id "120", then the value would be "100:tt_content:120". The value is use by eg. the cObject SEARCHRESULT. If the value \$GLOBALS['HTTP_POST_VARS']['locationData'] is detected here, the search is done as if it was performed on this page! This is very useful if you want a search functionality implemented on a page with the "stye" field set to "LI" which means that the search is carried out from the first level in the rootline. Suppose you want the search to submit to a dedicated search page where ever. This page will then know - because of locationData - that the search was submitted from another place on the website. If "locationData" is not only true but also set to "HTTP_POST_VARS" then the value will insert the content of \$GLOBALS['HTTP_POST_VARS']['locationData'] instead of the true location data of the page. This should be done with search-fields as this will carry the initial searching start point with. <b>NOTE:</b> May be overridden by the property override feature of the formdata (see above)	
<b>redirect</b>	string /stdWrap	URL to redirect to (generates the hidden field "redirect")  <b>Integer:</b> this is regarded to be a page in TYPO3 <b>String:</b> this is regarded to be a normal url <b>Empty;</b> the current page is chosen.  <b>NOTE:</b> If this value is set, the target of this overrides the target of the "type".	
<b>recipient</b>	(list of) string /stdWrap	Email recipient of the formmail content (generates the hiddenfield "recipient")	No email
<b>goodMess</b>	string /stdWrap	Message for the form evaluation function in case of correctly filled form.  <b>NOTE:</b> May be overridden by the property override feature of the formdata (see above).	No message
<b>badMess</b>	string /stdWrap	Message for the form evaluation in case of missing required fields. This message is shown above the list of fields.  <b>NOTE:</b> May be overridden by the property override feature of the formdata (see above).	No message



Property:	Data type:	Description:	Default:
<b>emailMess</b>	string /stdWrap	Message if a field evaluated to be an email address did not validate.  <b>NOTE:</b> May be overridden by the property override feature of the formdata (see above).	
<b>image</b>	->IMAGE (cObject)	If this is a valid image the submit button is rendered as this image!!  <b>NOTE:</b> CurrentValue is set to the caption-label before generating the image.	
<b>layout</b>	string	This defines how the label and the field are placed towards each other.  <b>This property is mandatory; you must set it!</b> Otherwise the form will not be rendered.  <b>Example:</b> This substitutes the marker "###FIELD###" with the field data and the marker "###LABEL###" with label data.  <pre>layout = &lt;tr&gt;&lt;td&gt;###FIELD###&lt;/td&gt;&lt;td&gt;###LABEL###&lt;/td&gt;&lt;/tr&gt;</pre> You can also use the marker ###COMMENT### which is ALSO the label value inserted, but wrapped in .commentWrap stdWrap-properties (see below).	
<b>fieldWrap</b>	->stdWrap	Field: Wraps the fields	
<b>labelWrap</b>	->stdWrap	Labels: Wraps the label	
<b>commentWrap</b>	->stdWrap	Comments: Wrap for comments IF you use ###COMMENT###	
<b>REQ</b>	boolean /stdWrap	Defines if required-fields should be checked and marked up.	
<b>REQ.fieldWrap</b>	->stdWrap	Field: Wraps the fields, but for required fields	the "fieldWrap"-property
<b>REQ.labelWrap</b>	->stdWrap	Labels: Wraps the label, but for required fields	the "labelWrap"-property
<b>REQ.layout</b>	string /stdWrap	The same as "layout" above, but for required fields	the "layout"-property
<b>COMMENT.layout</b>	string /stdWrap	Alternative layout for comments.	the "layout"-property
<b>CHECK.layout</b>	string /stdWrap	Alternative layout for checkboxes	the "layout"-property
<b>RADIO.layout</b>	string /stdWrap	Alternative layout for radio buttons	the "layout"-property
<b>LABEL.layout</b>	string /stdWrap	Alternative layout for label types	the "layout"-property
<b>stdWrap</b>	->stdWrap	Wraps the whole form (before form tag is added)	
<b>hiddenFields</b>	[array of cObject]	Used to set hiddenFields from TS.  <b>Example:</b> <pre>hiddenFields.pid = TEXT hiddenFields.pid.value = 2</pre> This makes a hidden-field with the name "pid" and value "2".  Available sub-property: <b>stdWrap</b> , see ->stdWrap.	

Property:	Data type:	Description:	Default:
<b>params</b>	form-element tag parameters /stdWrap	<p>Extra parameters to form elements.</p> <p><b>Example:</b></p> <pre>params = style="width:200px;" params.textarea = style="width:300px;" params.check =</pre> <p>This sets the default to 200 px width, but excludes check-boxes and sets textareas to 300.</p> <p><b>stdWrap</b> is available for the sub-properties, e.g. params.tagname.</p>	
<b>wrapFieldName</b>	wrap /stdWrap	<p>This wraps the field names before they are applied to the form-field tags.</p> <p><b>Example:</b></p> <p>If value is <code>tx_myextension[input][   ]</code> then the field name "email" would be wrapped to this value: <code>tx_myextension[input][email]</code></p>	
<b>noWrapAttr</b>	boolean /stdWrap	<p>If this value is true then all wrap attributes of textarea elements are suppressed. This is needed for XHTML-compliance.</p> <p>The wrap attributes can also be disabled on a per-field basis by using the special keyword "disabled" as the value of the wrap attribute.</p>	
<b>arrayReturnMode</b>	boolean /stdWrap	<p>If set, the &lt;form&gt; tags and the form content will be returned in an array as separate elements including other practical values. This mode is for use in extensions where the array return value can be more useful.</p>	
<b>accessibility</b>	boolean /stdWrap	<p>If set, then the form will be compliant with accessibility guidelines (XHTML compliant). This includes:</p> <ul style="list-style-type: none"> <li>label string will be wrapped in &lt;label for="formname[field name-hash]"&gt; ... &lt;/label&gt;</li> <li>All form elements will have an id-attribute carrying the formname with the md5-hashed field name appended</li> </ul> <p><b>Notice:</b> In TYPO3 4.0 and later, CSS Styled Content is configured to produce accessible forms by default.</p>	
<b>formName</b>	string /stdWrap	<p>An alternative name for this form. Default will be a unique (random) hash.</p> <pre>&lt;form name="..."&gt;</pre>	
<b>fieldPrefix</b>	string /stdWrap	<p>Alternative prefix for the name of the fields in this form. Otherwise, all fields are prefixed with the form name (either a unique hash or the name set in the "formName" property). If set to "0", there will be no prefix at all.</p>	
<b>dontMd5FieldNames</b>	boolean /stdWrap	<p>The IDs generated for all elements in a form are md5 hashes from the field name. Setting this to true will disable this behavior and use a cleaned field name, prefixed with the form name as the ID, instead.</p> <p>This can be useful to style specifically named fields with CSS.</p>	

[tsref:(cObject).FORM]

### Example: Login

In order to create a login form, you would need to supply these fields:

- "username" = username
- "userid" = password
- "login\_status" = "logout" for logout, "login" for login.

If you insert "<!--###USERNAME###-->" somewhere in your document this will be substituted by the username if a user is logged in!

If you want the login-form to change into a logout form you should use conditions to do this. See this TS-example (extract from the static\_template *"styles.content (default)"*):

```
# loginform
styles.content.loginform {
    data = Username:|*username=input || Password:|*userid=password
}
[usergroup = *]
styles.content.loginform.data = Username: <!--###USERNAME###--> || |submit=submit| Logout
[global]
```

### Example: Mailform

This creates a simple mail form (this is not TypoScript, but the setup code that you should put directly into the "bodytext"-field of a pagecontent record of the type "FORMMAIL"):

```
Name: | *replyto_name= input | Enter your name here
Email: | *replyto_email=input |
Like TV: | tv=check |
| formtype_mail = submit | Send this!

| html_enabled=hidden | 1
| subject=hidden| This is the subject
| recipient_copy=hidden | copy@email.com
| auto_respond_msg=hidden| Hello / This is an automatic response. //We have received your
mail.
| from_name=hidden | Website XY
| from_email=hidden | noreply@website.com
| organization=hidden | Organization XY
| redirect=hidden | 16
| priority=hidden | 5
| tv=hidden | 0
```

- "replyto\_name": If the field is named like this the value is used as reply to name in the email software and will not be shown in the mail content. Choose another field name like the\_name to use the value as a normal field. Note the asterisk (\*) which means the field is required. and the field name will be "the\_name". Also a default value is set ("Enter your name here")
- "replyto\_email": If the field is named like this the value is used as reply to email address in the email software and will not be shown in the mail content. To get the value as sender address in the mail software use "email" as field name.
- "Like TV" is a checkbox. Default is "unchecked".
- "formtype\_mail" is the name of the submit button. It **must** be names so if you use the built-in form mail of TYPO3, at it will make TYPO3 react automatically on the input and interpret it as form mail input!
- "html\_enabled" will let the mail be rendered in nice HTML
- "use\_base64" will send the mail encoded as base64 instead of quoted-printable
- "subject": Enter the subject of your mail
- "recipient\_copy" : A copy is sent to this mail-address. You may supply more addresses by separating with a comma (,). The mail sent to recipient\_copy is the same, but a separate message from the one sent to the 'recipient' and furthermore the copy-mail is sent only if the 'recipient' mail is sent.
- "auto\_respond\_msg": This is an auto-responder message. This is sent if the email of the "submitter" is known (field: "email"). The value of this is the message broken up in to lines by a slash "/". Each slash is a new line in the email. The first line is used for the subject.
- "from\_name": With this option you can set the mail header from name, which will be shown in the mail software.
- "from\_email": With this option you can set the mail header from email, which will be shown in

the mail software as sender address.

- "organization": With this option you can set the mail header organization parameter, which won't be shown in the mail but in the mail header.
- "redirect": With this option you can define a TYPO3 page (page id) or external URL (www.example.com) as redirect url after submit. If this option isn't set the form will be shown again.
- "priority": With this option you can set the priority of the mail from 1 (not important) to 5 (very important). Default is 3.
- "tv" (again, but hidden). Repeating this field may be smart as the value "tv" is normally NOT submitted with the value "false" if not checked. Inserting this line will ensure a default value for "tv".

## SEARCHRESULT

This object can be used to display search results.

Search words are loaded into the register in a form ready for linking to pages:

### Example:

```
register:SWORD_PARAMS = '&sword_list[]=word1&sword_list[]=word2 ....'
```

See [typolink](#) for more info!

SEARCHRESULT returns results only from pages with of doktype "Standard" (1), "Advanced" (2) and "Not in menu" (5)

Property:	Data type:	Description:	Default:
<b>allowedCols</b>	string	List (separated by ".") of allowed table-cols.  <b>Example:</b> <code>pages.title:tt_content.bodytext</code>	
<b>layout</b>	string	This defines how the search content is shown.  <b>Example:</b> This substitutes the following fields: <pre> ###RANGELOW###:   The low result range, eg. "1" ###RANGEHIGH###:  The high result range, eg. "10" ###TOTAL###:      The total results ###RESULT###:     The result itself ###NEXT###:       The next-button ###PREV###:       The prev-button </pre>	
<b>next</b>	cObject	This cObject will be wrapped by a link to the next search result. This is the code substituting the "###NEXT###"-mark	
<b>prev</b>	cObject	This cObject will be wrapped by a link to the prev search result. This is the code substituting the "###PREV###"-mark	
<b>target</b>	target /stdWrap	target til next/prev links!	
<b>range</b>	integer /stdWrap	The number of results at a time!	20
<b>renderObj</b>	cObject	The cObject to render the search results \$cObj->data array is set to the resulting record from the search. Please note, that in all fields are named [tablename]_[fieldnam]. Thus the page title is in the field "pages_title". Apart from this, these fields from the pages-table are also present: <code>uid</code>	
<b>renderWrap</b>	wrap /stdWrap		
<b>resultObj</b>	cObject	The cObject prepended in the search results returns rows	
<b>noResultObj</b>	cObject	The cObject used if the search results in no rows.	

Property:	Data type:	Description:	Default:
<b>noOrderBy</b>	boolean /stdWrap	If this is set, the result is NOT sorted after lastUpdated, tstamp for the pages-table.	
<b>wrap</b>	wrap /stdWrap	Wrap the whole content...	
<b>stdWrap</b>	->stdWrap	Wrap the whole content...	
<b>addExtUrlsAndShortcuts</b>	boolean	If set, then the doktypes 3 and 4 (External URLs and Shortcuts) are added to the doktypes being searched. However at this point in time, no pages will be select if they do not have at least one tt_content record on them! That is because the pages and tt_content (or other) table is joined. So there must at least be one occurrence of a tt_content element on an External URL / Shortcut page for them to show up.	
<b>languageField.</b> <b>[2nd table]</b>	string	Setting a field name to filter language on. This works like the "languageField" setting in ->select  <b>Example:</b>  <code>languageField.tt_content = sys_language_uid</code>	

[tsref:(cObject).SEARCHRESULT]

**NOTE:** "sword" and "scols" MUST be set in order for the search to be engaged.

var "sword" = search word(s)

var "scols" = search columns separated by ":". E.g.: pages.title:pages.keywords:tt\_content.bodytext

var "styp" = the starting point of the search: false = current page, L-2 = page before currentPage, L-1 = current page, L0 = rootlevel, L1 = from first level, L2 = from second level

var \$GLOBALS['HTTP\_POST\_VARS']['locationData']: If this is set, the search is done as was it from another page in the website given by the value of "locationData" here. See the description at the cObject "FORMS".

Only if the page locationData is pointing to, is inside the real rootLine of the site, the search will take this into account.

internal:

var "scount": If this is set this is used as the searchCount - the total rows in the search. This way we don't need to reconstruct this number!

var "spointer": This points to the start-record in the search.

LATER:

var "alldomains" : boolean: If set the search will proceed into other domains

var "allsites" : boolean: If set the search will proceed into other sites (defined by the "root" setting of an active template.)

var "depth": The depth

## Search syntax

When you search, you can use three operator types

- AND: "+", "and" (UK), "og" (DK)
- OR: "or" (UK), "eller" (DK)
- NOT: "-", "not" (UK), "uden" (DK)

Default operator is AND. If you encapsulate words in "" they are searched for as a whole string. The search is case insensitive and matches parts of words also.

### Examples:

1. *menu backend* - will find pages with both 'menu' and 'backend'.
2. *"menu backend"* - will find pages with the phrase "menu backend".
3. *menu or backend* - will find pages with either 'menu' or 'backend'
4. *menu or backend not content* - will find pages with either 'menu' or 'backend' but not 'content'

### Queries to the examples

In this case "pagecontent" is chosen as the fields to search. That includes *tt\_content.header*, *tt\_content.bodytext* and *tt\_content.imagecaption*.

Prefixed to these queries is this:

```
SELECT pages.title AS pages_title, pages.subtitle AS pages_subtitle, pages.keywords AS
pages_keywords, pages.description AS pages_description, pages.uid, tt_content.header AS
tt_content_header, tt_content.bodytext AS tt_content_bodytext, tt_content.imagecaption AS
tt_content_imagecaption
FROM pages, tt_content
WHERE (tt_content.pid=pages.uid) AND (pages.uid IN (2,5,6,20,21,22,29,30,31,3,4,8,9,16,1) AND
pages.doktype in (1,2,5) AND pages.no_search=0 AND NOT tt_content.deleted AND NOT
tt_content.hidden AND (tt_content.starttime<=985792797) AND (tt_content.endtime=0 OR
tt_content.endtime>985792797) AND tt_content.fe_group IN (0,-1) AND NOT pages.deleted AND
NOT pages.hidden AND (pages.starttime<=985792797) AND (pages.endtime=0 OR
pages.endtime>985792797) AND pages.fe_group IN (0,-1)) ...
```

The part "... pages.uid IN (2,5,6,20,21,22,29,30,31,3,4,8,9,16,1)..." is a list of pages-uid's to search. This list is based on the page-ids in the website-branch of the pagetree and confines the search to that branch and not the whole page-table.

1. ... AND ((tt\_content.header LIKE '%menu%' OR tt\_content.bodytext LIKE '%menu%' OR
tt\_content.imagecaption LIKE '%menu%') AND (tt\_content.header LIKE '%backend%' OR
tt\_content.bodytext LIKE '%backend%' OR tt\_content.imagecaption LIKE '%backend%')) GROUP
BY pages.uid
2. ... AND ((tt\_content.header LIKE '%menu backend%' OR tt\_content.bodytext LIKE '%menu
backend%' OR tt\_content.imagecaption LIKE '%menu backend%')) GROUP BY pages.uid
3. ... AND ((tt\_content.header LIKE '%menu%' OR tt\_content.bodytext LIKE '%menu%' OR
tt\_content.imagecaption LIKE '%menu%') OR (tt\_content.header LIKE '%backend%' OR
tt\_content.bodytext LIKE '%backend%' OR tt\_content.imagecaption LIKE '%backend%')) GROUP
BY pages.uid
4. ... AND ((tt\_content.header LIKE '%menu%' OR tt\_content.bodytext LIKE '%menu%' OR
tt\_content.imagecaption LIKE '%menu%') OR (tt\_content.header LIKE '%backend%' OR
tt\_content.bodytext LIKE '%backend%' OR tt\_content.imagecaption LIKE '%backend%')) AND NOT
(tt\_content.header LIKE '%content%' OR tt\_content.bodytext LIKE '%content%' OR
tt\_content.imagecaption LIKE '%content%')) GROUP BY pages.uid

Notice that upper and lowercase does not matter. Also 'menu' as searchword will find 'menu', 'menus', 'menuitems' etc.

## USER and USER\_INT

This calls either a PHP-function or a method in a class. This is very useful if you want to incorporate you own data processing or content.

Basically this is a userdefined cObject, because it's just a call to a function or method you control!

An important thing to know is that if you call a method in a class (which is of course instantiated as an object) the internal variable 'cObj' of that class is set with a *reference* to the parent cObj. See the file `typo3/sysexst/cms/tslib/media/scripts/example_callfunction.php` for an example of how this may be useful for you. Basically it offers you an API of functions which are more or less relevant for you. Refer to the appendix "PHP include scripts" at the end of this document.

If you create this object as USER\_INT, it'll be rendered non-cached, outside the main page-rendering.

Property:	Data type:	Description:	Default:
<b>userFunc</b>	function name	<p>The name of the function. If you specify the name with a '-&gt;' in, it's interpreted as a call to a method in a class. Two parameters are sent: A content variable (which is empty in this case, but not when used from stdWrap function .postUserFunc and .preUserFunc) and the second parameter is an array with the properties of this cObject if any.</p> <p><b>Example:</b> This TypoScript will display all content element headers of a page in reversed order. Please take a look at typo3/sysexst/cms/tslib/media/scripts/example_callfunction.php!</p> <pre> page = PAGE page.typeNum=0 includeLibs.something = typo3/sysexst/cms/tslib/media/scripts/example_c allfunction.php  page.30 = USER page.30 {     userFunc = user_various- &gt;listContentRecordsOnPage     reverseOrder = 1 }</pre> <p><b>NOTE:</b> When using a function, the name of the function has to start with "user_". When using a class, the name of the class must start with "user_" (there are no conditions on the name of the method).</p>	
<b>includeLibs</b>	<i>list of resource</i> /stdWrap	<p><b>This property applies only if the object is created as USER_INT.</b></p> <p>This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included.</p>	

[tsref:(cObject).USER/(cObject).USER\_INT]

## TEMPLATE

With this cObject you can define a template (e.g. an HTML file) which should be used as a basis for your whole website. Inside the template file you can define markers, which will later be replaced with dynamic content by TYPO3.

Property:	Data type:	Description:	Default:
<b>template</b>	cObject	<p>This must be loaded with the template-code. If not, the object returns nothing.</p> <p><b>Example:</b></p> <pre> page.10 {     template = FILE     template.file = fileadmin/template.html }</pre> <p>This will use the file fileadmin/template.html as template for your website.</p>	

Property:	Data type:	Description:	Default:
<b>subparts</b>	<i>Array... of cObject</i>	<p>This is an array of subpart-markers (case-sensitive). A subpart is defined by <b>two</b> markers in the template. The markers must be wrapped by "###" on both sides. You may insert the subpart-markers inside HTML-comment-tags!!</p> <p><b>Example:</b> In the template there is the subpart "HELLO":  <pre>&lt;!-- start of subpart ###HELLO### --&gt; This is the HTML-code, that will be loaded in the register and will be replaced with the result... &lt;!-- end ###HELLO### --&gt;</pre> The following TypoScript code now replaces the subpart "HELLO" with the text given in "value": <pre>page.10.subparts {     HELLO = TEXT     HELLO.value = En subpart er blevet     erstattet!! }</pre> <b>NOTE:</b> Before the content-objects of each subpart are generated, all subparts in the array are extracted and loaded into the register so that you can load them from there later on. The register-key for each subparts code is "SUBPART_[theSubpartkey]". In addition the current-value is loaded with the content of each subpart just before the cObject for the subpart is parsed. That makes it quite easy to load the subpart of the cObject (eg: ".current=1") Eg. this subpart above has the register-key "SUBPART_HELLO". <i>This is valid ONLY if the property .nonCachedSubst is not set! (see below)</i></p>	
<b>relPathPrefix</b>	<i>string / properties</i>	<p>Finds all relative references (e.g. to images or stylesheets) and prefixes this value. If you specify properties (uppercase) these will match HTML tags and specify alternative paths for them. See example below. If the property is named "style" it will set alternative path for the "url()" wrapper that may be in &lt;style&gt; sections.</p> <p><b>Example:</b>  <pre>page.10 = TEMPLATE page.10 {     template = FILE     template.file = fileadmin/template.html     relPathPrefix = fileadmin/     relPathPrefix.IMG = fileadmin/img/ }</pre> In this example all relative paths found are prefixed "fileadmin/" unless it was the src attribute of an img tag in which case the path prefixed is "fileadmin/img/"</p>	



Property:	Data type:	Description:	Default:
<b>marks</b>	<i>Array... of cObject</i>	<p>This is an array of marks-markers (case-sensitive). A mark is defined by <b>one</b> marker in the template. The marker must be wrapped by "###" on both sides. Opposite to subparts, you may NOT insert the subpart-markers inside HTML-comment-tags! (They will not be removed.)</p> <p><b>Example:</b> In the template:  <pre>&lt;div id="copyright"&gt;   &amp;copy;; ###DATE### &lt;/div&gt;</pre> </p> <p>The following TypoScript code now dynamically replaces the marker "DATE" with the current year:  <pre>page.10.marks {   DATE = TEXT   DATE {     data = date : U     strftime = %Y   } }</pre> </p> <p>Marks are substituted by a <code>str_replace</code>-function. The subparts loaded in the register are also available to the cObjects of markers (only if <code>.nonCachedSubst</code> is not set!).</p>	
<b>wraps</b>	<i>Array... of cObject</i>	<p>This is an array of wraps-markers (case-sensitive). This is shown best by an example: <b>Example:</b> In the template there is the subpart "MYLINK":  <pre>This is &lt;!--###MYLINK###--&gt;a link to my&lt;!-- ###MYLINK###--&gt; page!</pre> </p> <p>With the following TypoScript code the subpart will be substituted by the wrap which is the content returned by the MYLINK cObject.  <pre>page.10.wraps {   MYLINK = TEXT   MYLINK.value = &lt;a href="#"&gt;   &lt;/a&gt; }</pre> </p>	
<b>workOnSubpart</b>	string /stdWrap	This is an optional definition of a subpart, that we decide to work on. In other words; if you define this value that subpart is extracted from the template and is the basis for this whole template object.	
<b>markerWrap</b>	wrap /stdWrap	<p>This is the wrap the markers are wrapped with. The default value is <code>###   ###</code> resulting in the markers to be presented as <code>###[marker_key]###</code>.</p> <p>Any whitespace around the wrap-items is stripped before they are set around the marker_key.</p>	<code>###   ###</code>
<b>substMarksSeparately</b>	boolean /stdWrap	<p>If set, then marks are substituted in the content AFTER the substitution of subparts and wraps.</p> <p>Normally marks are not substituted inside of subparts and wraps when you are using the default cached mode of the TEMPLATE cObject. That is a problem if you have marks inside of subparts! But setting this flag will make the marker-substitution a non-cached, subsequent process.</p> <p>Another solution is to turn off caching, see below.</p>	

Property:	Data type:	Description:	Default:
<b>nonCachedSubst</b>	boolean /stdWrap	If set, then the substitution mode of this cObject is totally different. Normally the raw template is read and divided into the sections denoted by the marks, subparts and wraps keys. The good thing is high speed, because this "pre-parsed" template is cached. The bad thing is that templates that depend on incremental substitution (where the order of substitution is important) will not work so well. By setting this flag, markers are first substituted by str_replace in the template - one by one. Then the subparts are substituted one by one. And finally the wraps one by one. Obviously you loose the ability to refer to other parts in the template with the register-keys as described above.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).TEMPLATE]

**Example:**

```

page.10 = TEMPLATE
page.10 {
    template = FILE
    template.file = fileadmin/test.tmpl
    subparts {
        HELLO = TEXT
        HELLO.value = This is the replaced subpart-code.
    }
    marks {
        Testmark = TEXT
        Testmark.value = This is replacing a simple marker in the HTML-code.
    }
    workOnSubpart = DOCUMENT
}

```

In this example a template named test.tmpl is loaded and used.

# FLUIDTEMPLATE

The TypoScript object FLUIDTEMPLATE works in a similar way to the regular "marker"-based TEMPLATE object. However, it does not use markers or subparts, but allows Fluid-style variables with curly braces.



## Note

The extensions "fluid" and "extbase" need to be installed for this to work.

Property:	Data type:	Description:	Default:
<b>file</b>	string /stdWrap	The fluid template file.	
<b>layoutRootPath</b>	filepath /stdWrap	Sets a specific layout path; usually it is Layouts/ underneath the template file.	
<b>partialRootPath</b>	filepath /stdWrap	Sets a specific partials path; usually it is Partials/ underneath the template file.	
<b>format</b>	keyword /stdWrap	Sets the format of the current request.	html
<b>extbase.pluginName</b>	string /stdWrap	Sets variables for initializing extbase.	
<b>extbase.controllerExtensionName</b>	string /stdWrap	Sets the extension name of the controller.	
<b>extbase.controllerName</b>	string /stdWrap	Sets the name of the controller.	
<b>extbase.controllerActionName</b>	string /stdWrap	Sets the name of the action.	
<b>variables</b>	Array... of cObjects	Sets variables that should be available in the fluid template. The keys are the variable names in Fluid. Reserved variables are "data" and "current", which are filled automatically with the current data set.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).FLUIDTEMPLATE]

## Example:

The Fluid template (in fileadmin/templates/MyTemplate.html) could look like this:

```
<h1>{data.title}<f:if condition="{data.subtitle}">, {data.subtitle}</f:if></h1>
<h3>{mylabel}</h3>
<f:format.html>{data.bodytext}</f:format.html>
```

You could use it with a TypoScript code like this:

```
page = PAGE
page.10 = FLUIDTEMPLATE
page.10 {
    file = fileadmin/templates/MyTemplate.html
    partialRootPath = fileadmin/templates/partial/
    variables {
        mylabel = TEXT
        mylabel.value = Label coming from TypoScript!
    }
}
```

As a result the page title and the label from TypoScript will be inserted as headlines.

## MEDIA

The Media content element is a dispatcher which gets its HTML output from one of the available render objects. By default, these render objects include SWFOBJECT (Flash driven by JavaScript), QTOBJECT (QuickTime driven by JavaScript) and Multimedia (the original Multimedia object rendered with EMBED tags).

The property "renderType" defines which object will be used for rendering. If set to its default value "auto", the Media content element uses the media file's extension to choose the right renderer. This auto-detection may not work as well for external URLs so setting the renderType manually is preferable in that case.

If one of the existing renderTypes does not meet your needs, new renderTypes can be registered and rendered with a custom extension.

The Media content element contains the following 3<sup>rd</sup> party files in typo3/contrib/flashmedia:

- qtobject/qtobject.js (JavaScript for QTOBJECT)
- swfobject/swfobject.js (JavaScript for SWFOBJECT)
- swfobject/expressInstall.swf (This is displayed if the client's Flash version is too low)
- flvplayer.swf (TYPO3 video player for flv, swf, mp4, m4u etc)
- player.swf (Audio player from lpixelout)
- player.txt (License for the audio player)

If you want to use a different player, it can be configured via TypoScript.



### Note

Files are treated as URLs. You need to set fully qualified URLs. Use config.baseURL and/or config.absRefPrefix to get fully qualified URLs automatically.

Property:	Data type:	Description:	Default:
<b>flexParams</b>	string /stdWrap	Used for Flexform configuration of the content element	flexParams.field = pi_flexform
<b>alternativeContent</b>	stdWrap	Alternative content, which is printed out, if the client deactivated JavaScript or has no Flash. Otherwise, the media will replace this content.	alternativeContent.field = bodytext
<b>type</b>	string /stdWrap	Defines media type: "video" or "audio".	video
<b>renderType</b>	string /stdWrap	Used to select the render object. Possible values are: auto, swf, qt, embed. Extensions may add a custom renderType as well. swf will be used, if renderType is "auto".  <b>Note:</b> renderType embed will be rendered by the cObject MULTIMEDIA, swf by SWFOBJECT and qt by QTOBJECT. For the according documentation see the sections on these cObjects.	auto
<b>allowEmptyURL</b>	boolean	If set to 0, you see a warning if no file/URL is configured. If you do some advanced setup such as configuring a JavaScript-driven player with a playlist, you may use the object without a URL and need to set the value to 1.	0

Property:	Data type:	Description:	Default:
<b>fileExtHandler</b>	array	<p>The mappings between file extensions and render types can be configured here and will be used when renderType = auto. Possible values are MEDIA, SWF, QT.</p> <p><b>Example:</b></p> <pre>fileExtHandler {     default = MEDIA     mp3 = SWF     mp4 = SWF     m4v = SWF     mov = QT     avi = MEDIA     asf = MEDIA     class = MEDIA     swa = SWF }</pre>	
<b>mimeConf.swfobject</b> <b>mimeConf.qtobject</b>	array	<p>Configuration for a specific renderType. For each media type you can set default values.</p> <p><b>Example:</b></p> <pre>mimeConf.swfobject.audio {     defaultWidth = 100     defaultHeight = 50 }</pre>	
<b>file</b>	string /stdWrap	URL of the media file.	
<b>parameter</b>	array	<p>There are some configuration values which are set via the media content element. They are used to override the default settings. It is not expected to use them directly via TypoScript.</p> <pre>parameter {     mmFile     mmRenderType     mmforcePlayer     mmType     mmWidth     mmHeight     mmMediaOptions     mmMediaOptionsContainer }</pre>	
<b>forcePlayer</b>	string /stdWrap	If the file is a URL and forcePlayer is not set, the URL will be called directly instead of using a player.	
<b>width</b>	int /stdWrap	Media width, will be overridden by parameter.mmWidth.	
<b>height</b>	int /stdWrap	Media height, will be overridden by parameter.mmHeight.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).MEDIA]

# SWFOBJECT

This object will insert a Flash player driven by JavaScript.

Property:	Data type:	Description:	Default:
<b>file</b>	string /stdWrap	Media file or URL.  <b>Note:</b> Files are treated as URLs. You need to set fully qualified URLs. Use config.baseURL and/or config.absRefPrefix to get fully qualified URLs automatically.	
<b>width</b>	int /stdWrap	Width of the swfObject. If it is not set, it will be filled with defaultWidth of the player configuration.	
<b>height</b>	int /stdWrap	Height of the swfObject. If it is not set, it will be filled with defaultHeight of the player configuration.	
<b>type</b>	string /stdWrap	Sets default for different media types. E.g. "audio" or "video". If value is "audio", the player configuration audio.player will be used.	
<b>[type].player</b>	string /stdWrap	Location of player	
<b>[type].player.default</b>	array	Default parameter for flashvars / params / attributes.  Usage: default { flashvars.allowFullScreen = true params.wmode = transparent attributes.align = center } flashvars are used for swf file configuration. There is no standard across players, but for flvplayer see description below. For detailed description of possible params/attributes visit this URL: <a href="http://livedocs.adobe.com/flash/9.0/UsingFlash/help.html?content=W5d60f23110762d6b883b18f10cblfelaf6-7ba7.html">http://livedocs.adobe.com/flash/9.0/UsingFlash/help.html?content=W5d60f23110762d6b883b18f10cblfelaf6-7ba7.html</a>	
<b>[type].player.defaultWidth</b>		Default media width.	
<b>[type].player.defaultHeight</b>		Default media height.	
<b>[type].player.mapping</b>		The audio player doesn't work with file, but instead expects the file with the flashvar soundFile. mapping does the rename of parameter for you by default.  <b>Example:</b> <pre>mapping {     flashvars.file = soundFile }</pre>	
<b>installUrl</b>	string /stdWrap		typo3/contrib/flashmedia/swfobject/expressInstall.swf
<b>forcePlayer</b>	string /stdWrap	If the file is a URL and forcePlayer is not set, the URL will be called directly instead of using a player.	
<b>flashvars</b>	array	Flash vars.	
<b>params</b>	array	Flash params.	
<b>attributes</b>	array	Flash attributes.	
<b>flashVersion</b>	string /stdWrap	Required flash version.	9
<b>alternativeContent</b>	stdWrap	Alternative content.	alternativeContent.field = bodytext

Property:	Data type:	Description:	Default:
<b>layout</b>	stdWrap	HTML Template for the Object. <code>###SWFOBJECT###</code> is replaced with the sfwobject, <code>###ID###</code> is replaced with the unique Id of the div/object.	<code>###SWFOBJECT###</code>
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).SWFOBJECT]

## QTOBJECT

This element inserts a QuickTime Player.

Property:	Data type:	Description:	Default:
<b>file</b>	stdWrap	Media file or URL.  <b>Note:</b> Files are treated as URLs. You need to set fully qualified URLs. Use <code>config.baseURL</code> and/or <code>config.absRefPrefix</code> to get fully qualified URLs automatically.	
<b>width</b>	int	Width of QTOBJECT. If it is not set, it will be filled with <code>defaultWidth</code> of the player configuration.	
<b>height</b>	int	Width of QTOBJECT. If it is not set, it will be filled with <code>defaultHeight</code> of the player configuration.	
<b>alternativeContent</b>	stdWrap	Alternative content.	<code>alternativeContent.field = bodytext</code>
<b>layout</b>	stdWrap	HTML Template for the Object. <code>###QTOBJECT###</code> is replaced with the qtobject, <code>###ID###</code> is replaced with the unique Id of the div/object.	<code>###QTOBJECT###</code>
<b>params</b>	array	Define some parameters which should be set for the QTOBJECT. These settings having precedence over player specific settings ( <code>[type].player.default.aprams</code> ).	
<b>type</b>	string /stdWrap	Sets default for different media types. E.g. "audio" or "video". If value is "audio", the player configuration <code>audio.player</code> will be used. Player specific settings are only used, if there is no general value set.	
<b>[type].player.default</b>	array	Player specific default parameters. You can override them via <code>params</code> setting (see above). Usage: <pre>default.params {     autoplay = true }</pre>	
<b>[type].player.defaultWidth</b>	int	Default width.	
<b>[type].player.defaultHeight</b>	int	Default height.	
<b>[type].player.mapping</b>	array	The mapping does the rename of a parameter for a specific player type. Player specific parameter mapping. See SWFOBJECT for an example.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).QTOBJECT]

## MULTIMEDIA

This element will insert a multimedia file. Text files will be output directly; for Java, Flash, Audio and Video files an embed tag will be used.

Property:	Data type:	Description:	Default:
<b>file</b>	resource /stdWrap	The multimedia file. Possible file types are: <b>txt, html, htm</b> : Will be inserted directly, of the following properties only ".stdWrap" can be used. <b>class</b> : Java-applet. <b>swf</b> : Flash animation. <b>swa, dcr</b> : ShockWave Animation. <b>au, wav, mp3</b> : Sound. <b>avi, mov, asf, mpg, wmv</b> : Movies (AVI, QuickTime, MPEG4).	
<b>params</b>	string /stdWrap	These are parameters for the multimedia-objects. Use this to enter stuff like autostart, type, width, height and so on. For each file type several parameters make sense. For an incomplete list see below this table.  <b>Example:</b> <pre> params (     type = application/x-shockwave-flash     width = 200     height = 300 ) </pre> This will generate a tag like <pre> &lt;embed .... type="application/x-shockwave-flash" width="200" height="300"&gt; </pre> For parameters which are set by default (see tables below) an empty string will remove the parameter from the embed-tag. <b>Example:</b> <pre> params (     height = ) </pre> <b>Note:</b> If you set a width or a height here, this will overwrite the width or the height which have been set using ".width" and ".height".	
<b>width</b>	integer /stdWrap	Width attribute of the embed tag. Not used for txt, html, htm and sound files.	200
<b>height</b>	integer /stdWrap	Height attribute of the embed tag. Not used for txt, html, htm and sound files.	200
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).MULTIMEDIA]

### Meaningful parameters for .params

For the different file types many different parameters can be set. This is an incomplete list of some of those parameters:

**au, wav, mp3:**

Parameter:	Description:	Default:
<b>width</b>	Width of the controls. If not set, the browser defaults to 200.	
<b>height</b>	Height of the controls. If not set, the browser defaults to 16.	
<b>loop</b>	Repeat the sound, when playing finished. Set to true or false.	
<b>autostart</b>	Automatically start the sound. Set to true or false.	



***avi, mov, asf, mpg, wmv:***

Parameter:	Description:	Default:
<b>width</b>	Width of the movie.	200
<b>height</b>	Height of the movie.	200
<b>autostart</b>	Automatically start the video. Set to true or false.  <b>Note:</b> Not for "mov", there the parameter is called "autostart". See example below.	

***swf, swa, dcr:***

Parameter:	Description:	Default:
<b>width</b>	Width of the object. If not set, the browser defaults to approx. 200.	200
<b>height</b>	Height of the object. If not set, the browser defaults to approx. 200.	200
<b>quality</b>	Quality of the video.	high

***class:***

Parameter:	Description:	Default:
<b>width</b>	Width of the object.	200
<b>height</b>	Height of the object.	200

***Example for QuickTime (mov):***

```

params (
    width = 256
    height = 208
    autoplay = true
    controller = true
    loop = false
    pluginspage = http://www.apple.com/quicktime/
)

```

## SVG

With this element you can insert a SVG. You can use XML data directly or reference a file. A flash fallback will be used for browsers which do not have native SVG support, so that it also works in e.g. IE 6/7/8.

Property:	Data type:	Description:	Default:
<b>width</b>	integer /stdWrap	Width of the SVG.	600
<b>height</b>	integer /stdWrap	Height of the SVG.	400
<b>src</b>	file resource /stdWrap	SVG file resource.  <b>Example:</b> src = fileadmin/svg/tiger.svg	
<b>value</b>	XML /stdWrap	Raw XML data for the SVG. Will be ignored, if "src" is defined.	
<b>noscript</b>	string /stdWrap	Output, if SVG output is not possible.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).SVG]

**Example:**

```
10 = SVG
10 {
    width = 600
    height = 600
    value (
        <rect x="100" y="100" width="500" height="200" fill="white" stroke="black" stroke-
width="5px"/>
        <line x1="0" y1="200" x2="700" y2="200" stroke="red" stroke-width="20px"/>
        <polygon points="185 0 125 25 185 100" transform="rotate(135 125 25)" />
        <circle cx="190" cy="150" r="40" stroke="black" stroke-width="2" fill="yellow"/>
    )
    noscript.cObject = TEXT
    noscript.cObject.value = No SVG rendering possible, please use a browser.
}
```

This example will show some geometric forms.

## EDITPANEL

This content object is inserted only if a backend user is logged in and if a FE-editing extension is installed and loaded. What gets displayed exactly may depend on which FE-editing extension is used. The reference below is related to the "feedit" system extension. In such a case the EDITPANEL also requires that the Admin Panel be displayed (config.admPanel = 1) and that the user has checked the "Display Edit Icons" option. Whenever the edit panel is inserted, page caching is disabled.

The edit panel inserts icons for moving, editing, deleting, hiding and creating records.

In conjunction with css\_styled\_content, an EDITPANEL will appear for each content element on the page. It is also possible to insert an EDITPANEL as cObject in the template, using TypoScript.

**Example**

```
page = PAGE
page.10 = EDITPANEL
page.10 {
    ...
}
```

In such a case, there's nothing to edit in the FE, but the panel can be used to create new records, for example.



**Note**

The extension "feedit" needs to be installed for this to work.

Property:	Data type:	Description:	Default:
<b>label</b>	string /stdWrap	Title for the panel. You can insert the record title with %s  <b>Example:</b> label = Section <b>%s</b>	
<b>allow</b>	string	Define which functions are accessible. Further this list may be reduced, if the BE_USER does not have permission to perform the action Values should be listed separated by comma. This is the options you can choose between: toolbar,edit,new,delete,move,hide (toolbar is a general list of icons regarding the page, so use this for page records only)	
<b>newRecordFrom Table</b>	string	Will display a panel for creation of new element (in the top of list) on the page from that table.	
<b>newRecordInPid</b>	int	Define a page ID where new records (except new pages) will be created.	
<b>line</b>	boolean / int	If set, a black line will appear after the panel. This value will indicate the distance from the black line to the panel	

Property:	Data type:	Description:	Default:
<b>edit.displayRecord</b>	boolean	If set, then the record edited is displayed above the editing form.	
<b>onlyCurrentPid</b>	boolean	If set, only records with a pid matching the current id (TSFE->id) will be shown with the panel.	
<b>innerWrap</b>	wrap /stdWrap	Wraps the edit panel	
<b>outerWrap</b>	wrap /stdWrap	Wraps the whole edit panel including the black line (if configured)	
<b>printBeforeContent</b>	boolean	<p>Normally the edit panel is displayed below the content element it belongs to. If this option is set, the panel is printed in front of the according element.</p> <p><b>Example:</b></p> <pre>tt_content.stdWrap.editPanel.   printBeforeContent = 1</pre> <p>This displays the edit panels in front of the according elements, if you use <code>css_styled_content</code>.</p>	0
<b>previewBorder</b>	boolean / int	<p>If set, the hidden/starttime/endtime/fe_user elements which are previewed will have a border around.</p> <p>The integer value denotes the thickness of the border</p>	
<b>previewBorder.innerWrap</b> <b>previewBorder.outerWrap</b> <b>previewBorder.color</b>	wrap / HTML color	<p><b>innerWrap</b> wraps the content elements (including the icons) inside the preview border (an HTML table).</p> <p><b>outerWrap</b> wraps the whole content element including the border.</p> <p><b>color</b> denotes the color of the border.</p>	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).EDITPANEL]

# GIFBUILDER

## GIFBUILDER

GIFBUILDER is an object, which is used in many situations for creating gif-files. Anywhere the ->GIFBUILDER object is mentioned, these are the properties that apply.

Using TypoScript you can define a "numerical array" of "GIFBUILDER OBJECTS" (like "TEXT", "IMAGE", etc.) and they will be rendered onto an image one by one.

The name "GIFBUILDER" comes from the time where GIF was the only file format supported. PNG and JPG are just as well to create today (configured with \$TYPO3\_CONF\_VARS['GFX']).

### NOTE (+calc)

Whenever the "+calc"-function is added to a value in the data type of the properties underneath, you can use the dimensions of TEXT and IMAGE-objects from the GifBuilderObj-array. This is done by inserting a tag like this: "[10.w]" or "[10.h]", where "10" is the GifBuilderObj-number in the array and "w"/"h" signifies either width or height of the object.

The special property "lineHeight" (e.g. "[10.lineHeight]") uses the height a single line of text would take.

On using the special function max(), the maximum of multiple values can be determined. Example:

```
XY: [10.w]+[20.w], max([10.h], [20.h])
```

Here's a full example (taken from "styles.content (default)"):

```
styles.header.gfx1 = IMAGE
styles.header.gfx1 {
    wrap = {$styles.header.gfx1.wrap}
    file = GIFBUILDER
    file {
        XY = [10.w]+10 , {$styles.header.gfx1.itemH}
        backColor = {$styles.header.gfx1.bgCol}
        reduceColors = {$styles.header.gfx1.reduceColors}
        10 = TEXT
        10 {
            text.current = 1
            text.crop = {$styles.header.gfx1.maxChars}
            fontSize = {$styles.header.gfx1.fontSize}
            fontFile = {$styles.header.gfx1.file.fontFile}
            fontColor = {$styles.header.gfx1.fontColor}
            offset = {$styles.header.gfx1.fontOffset}
        }
    }
}
```

As you see, the gif-image has a width defined as the width of the text printed onto it + 10 pixels. The height is fixed by the value of the constant {\$styles.header.gfx1.itemH}

### The "\_GIFBUILDER" Top Level Object

You can configure some global settings for GIFBUILDER by a top level object named "\_GIFBUILDER". One of the available properties of the global GIFBUILDER configuration is "charRangeMap".

.charRangeMap

By this property you can globally configure mapping of font files for certain character ranges. For instance you might need GIFBUILDER to produce gif files with a certain font for latin characters while you need to use another true type font for Japanese glyphs. So what you need is to specify the usage of another font file when characters fall into another range of Unicode values.

In the GIFBUILDER object this is possible with the "splitRendering" option but if you have hundreds of GIFBUILDER objects around your site it is not very efficient to add 5-10 lines of configuration for each time you render text. Therefore this global setting allows you to match the basename of the main font face with an alternative font.

Property:	Data type:	Description:	Default:
[array]	string	<p>Basename of font file to match for this configuration. Notice that only the <i>filename</i> of the font file is used - the path is stripped off. This is done to make matching easier and avoid problems when font files might move to other locations in extensions etc.</p> <p>So if you use the font file "EXT:myext/fonts/arial.ttf" or "t3lib/fonts/arial.ttf" both of them will match with this configuration.</p> <p><b>The key:</b> The value of the array key will be the key used when forcing the configuration into "splitRendering" configuration of the individual GIFBUILDER objects. In the example below the key is "123". Notice; If the key is already found in the local GIFBUILDER configuration the content of that key is respected and not overridden. Thus you can make local configurations which override the global setting.</p> <p><b>Example:</b></p> <pre>_GIFBUILDER.charRangeMap {   123 = arial.ttf   .... }</pre>	
[array].charMapConfig	TEXT / splitRendering. [array] configuration	<p>splitRendering configuration to set. See GIFBUILDER TEXT object for details.</p> <p><b>Example:</b></p> <pre>_GIFBUILDER.charRangeMap {   123 = arial.ttf   123 {     charMapConfig {       fontFile = t3lib/fonts/vera.ttf       value = -65       fontSize = 45     }     fontSizeMultipliator = 2.3   } }</pre> <p>This example configuration shows that GIFBUILDER TEXT objects with font faces matching "arial.ttf" will have a splitConfiguration that uses "t3lib/fonts/vera.ttf" for all characters that fall below/equal to 65 in Unicode value.</p>	
[array].fontSizeMultipliator	double	<p>If set, this will take the font size of the TEXT GIFBUILDER object and multiply with this amount (xx.xx) and override the "fontSize" property inside "charMapConfig".</p>	
[array].pixelSpaceFontSizeRef	double	<p>If set, this will multiply the four [x/y]Space[Before/After] properties of split rendering with the relationship between the fontsize and this value. In other words; Since pixel space may vary depending on the font size used you can simply specify by this value at what fontsize the pixel space settings are optimized and for other font sizes this will automatically be adjusted according to this font size.</p> <p><b>Example:</b></p> <pre>_GIFBUILDER.charRangeMap {   123 = arial.ttf   123 {     charMapConfig {       fontFile = t3lib/fonts/vera.ttf       value = 48-57       color = green     }   } }</pre>	

Property:	Data type:	Description:	Default:
		<pre> xSpaceBefore = 3 xSpaceAfter = 3 } pixelSpaceFontSizeRef = 24 } } </pre> <p>In this example xSpaceBefore and xSpaceAfter will be "3" when the font size is 24. If this configuration is used on a GIFBUILDER TEXT object where the font size is only 16, the spacing values will be corrected by "16/24", effectively reducing the pixelspace to "2" in that case.</p>	

[tsref: \_GIFBUILDER.charRangeMap]

## Object names in this section

Whenever you see a reference to anything named an "object" in this section it's a reference to a "GifBuilderObj" and not the "cObjects" from the previous section. Confusion could happen, because both "IMAGE" and "TEXT" is an object in both areas.

Property:	Data type:	Description:	Default:
<b>1,2,3,4...</b>	GifBuilderObj + .if (->if)	.if (->if) is a property of all gifbuilder-objects. If the property is present and NOT set, the object is NOT rendered! This corresponds to the functionality of ".if" of the stdWrap-function.	
<b>XY</b>	x,y +calc /stdWrap	Size of the gif-file. For the usage of "calc" see the according note on that at the beginning of the section "GIFBUILDER".	100,20
<b>format</b>	"gif" / "jpg"	Output type. "jpg"/"jpeg" = jpg-image	gif
<b>reduceColors</b>	posint (1-255) /stdWrap	Reduce the number of colors (if gif-file)	
<b>transparentBackground</b>	boolean /stdWrap	Set this flag to render the background transparent. TYPO3 makes the color found at position 0,0 of the image (upper left corner) transparent. If you render text, you should leave the niceText option OFF as the result will probably be more precise without the niceText antialiasing hack.	
<b>transparentColor</b>	<i>HTMLColor</i> /stdWrap	Specify a color that should be transparent  <b>Example-values:</b> #ffffcc red 255,255,127  <b>Option:</b> transparentColor.closest = 1 This will allow for the closest color to be matched instead. You may need this if you image is not guaranteed "clean".  <b>NOTE:</b> You may experience that this doesn't work if you use the reduceColors-option or render text with niceText-option.	
<b>quality</b>	posint (10-100)	JPG-quality (if ".format" = jpg/jpeg)	
<b>backColor</b>	GraphicColor /stdWrap	Background color for the gif.	white
<b>offset</b>	x,y +calc /stdWrap	Offset all objects on the gif.	0,0

Property:	Data type:	Description:	Default:
<b>workArea</b>	x,y,w,h + calc /stdWrap	Define the workarea on the giffile. All the GifBuilderObj's will see this as the dimensions of the gif-file regarding alignment, overlaying of images an so on. Only TEXT-objects exceeding the boundaries of the workarea will be printed outside this area.	
<b>maxWidth</b>	pixels /stdWrap	Maximal width of the gif-file.	
<b>maxHeight</b>	pixels /stdWrap	Maximal height of the gif-file.	

[tsref:-&gt;GIFBUILDER]

## TEXT

Property:	Data type:	Description:	Default:
<b>text</b>	->stdWrap	This is text text-string on the gif-file. The item is rendered only if this string is not empty. The cObj->data-array is loaded with the page-record, if for example the GIFBUILDER-object is used by GMENU or IMGMENU.	
<b>breakWidth</b>	integer /stdWrap	Defines the maximum width for an object, overlapping elements will force an automatic line break.	
<b>breakSpace</b>	float	Defines a value that is multiplied by the line height of the current element.	1.0
<b>textMaxLength</b>	int	The maximum length of the text. This is just a natural break that prevents incidental rendering of very long texts!	100
<b>maxWidth</b>	pixels /stdWrap	Sets the maximum width in pixels, the text must be. Reduces the fontSize if the text does not fit within this width.  Does not support setting alternative fontSizes in splitRendering options.  (By René Fritz <r.fritz@colorcube.de>)	
<b>doNotStripHTML</b>	boolean	If set, HTML-tags in the string inserted are NOT removed. Any other way HTML-code is removed by default!	0
<b>fontSize</b>	posint	Font size	12
<b>fontColor</b>	GraphicColor /stdWrap	Font color	black
<b>fontFile</b>	resource	Font face (truetype font you can upload!)	Nimbus (Arial-clone)
<b>angle</b>	degree	Rotation degrees of the text.  <b>Note:</b> Angle is not available if spacing/wordSpacing is set.	0 Range: -90 til 90
<b>align</b>	align	Alignment of the text	left
<b>offset</b>	x,y +calc /stdWrap	Offset of the text	0,0
<b>antiAlias</b>	boolean	FreeType antialiasing. Notice, the default mode is "on"!.  <b>Note:</b> This option is not available if .niceText is enabled.	1
<b>iterations</b>	posint	How many times the text should be "printed" onto it self. This will add the effect of bold text.  <b>Note:</b> This option is not available if .niceText is enabled.	1
<b>spacing</b>	posint	Pixel-distance between letters. This may render ugly!	0
<b>wordSpacing</b>	posint	Pixel-distance between words.	= ".spacing"*2
<b>hide</b>	boolean	If this is true, the text is NOT printed. This feature may be used if you need a shadow-object to base a shadow on the text, but do not want the text to print.	0

Property:	Data type:	Description:	Default:
<b>hideButCreateMap</b>	boolean	If this option is set, the text will not be rendered. Shadows and emboss will, though, so don't apply these!! But this feature is also meant only to enable a text to generate the imageMap coordinates without rendering itself.	
<b>emboss</b>	GifBuilderObj->EMBOSS		
<b>shadow</b>	GifBuilderObj->SHADOW		
<b>outline</b>	GifBuilderObj->OUTLINE		
<b>imgMap</b>	->IMGMAP  ->stdWrap properties for "altText" and "titleText" in this case		
<b>niceText</b>	boolean	<p>This is a very popular feature that helps to render small letters much nicer than the freetype library can normally do. But it also loads the system very much!</p> <p>The principle of this function is to create a black/white giffile in twice or more times the size of the actual gif-file and then print the text onto this in a scaled dimension. Afterwards ImageMagick (IM) scales down the mask and masks the font color down on the original gif-file through the temporary mask.</p> <p>The fact that the font is actually rendered in the double size and scaled down adds a more homogenous shape to the letters. Some fonts are more critical than others though. If you do not need the quality, then don't use the function.</p> <p><b>Some properties:</b>          .before = IM-params before scale          .after = IM-params after scale          .sharpen = sharpen-value for the mask (after scaling), integer 0-99 (this enables you to make the text crisper if it's too blurred!)          .scaleFactor = scaling-factor, int 2-5</p>	
<b>splitRendering.compX</b> <b>splitRendering.compY</b> <b>splitRendering.array</b>		<p>Split the rendering of a string into separate processes with individual configurations. By this method a certain range of characters can be rendered with another font face or size. This is very useful if you want to use separate fonts for strings where you have latin characters combined with e.g. Japanese and there is a separate font file for each. You can also render keywords in another font/size/color.</p> <p><b>Properties:</b>          splitRendering.compX = <i>Additional pixelspace between parts, x direction</i>          splitRendering.compY = <i>Additional pixelspace between parts, y direction</i>          splitRendering.array = <i>keyword [charRange, highlightWord]</i>          splitRendering.array {            fontFile = <i>Alternative font file for this rendering</i>            fontSize = <i>Alternative font size for this rendering</i>            color = <i>Alternative color for this rendering, works ONLY without "niceText"</i>            xSpaceBefore = <i>x-Space before this part</i>            xSpaceAfter = <i>x-Space after this part</i>            ySpaceBefore = <i>y-Space before this part</i>            ySpaceAfter = <i>y-Space after this part</i>          }  <b>Keyword: charRange</b></p>	



Property:	Data type:	Description:	Default:
		<p>splitRendering.[array].value = Commaseparated list of character ranges (eg. "100-200") given as Unicode character numbers. The list accepts optional starting and ending points, eg. " - 200" or " 200 -" and single values, eg. "65, 66, 67"</p> <p><b>Keyword: highlightWord</b> splitRendering.[array].value = Word to highlight, makes a case sensitive search for this.</p> <p><b>Limitations:</b></p> <ul style="list-style-type: none"> <li>The pixelcompensation values are not corrected for scale factor used with niceText. Basically this means that when niceText is used, these values will have only the half effect.</li> <li>When word spacing is used the "highlightWord" mode doesn't work.</li> <li>The color override works only without "niceText".</li> </ul> <p><b>Example:</b></p> <pre> 10.splitRendering.compX = 2 10.splitRendering.compY = -2 10.splitRendering.10 = charRange 10.splitRendering.10 {     value = 200-380 , 65, 66     fontSize = 50     fontFile = t3lib/fonts/nimbus.ttf     xSpaceBefore = 30 } 10.splitRendering.20 = highlightWord 10.splitRendering.20 {     value = TheWord     color = red } </pre>	

[tsref:->GIFBUILDER.(GBObj).TEXT]

## SHADOW

Property:	Data type:	Description:	Default:
<b>textObjNum</b>	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
<b>offset</b>	x,y	Shadow offset	
<b>color</b>	GraphicColor	Shadow color	
<b>blur</b>	posint (1-99)	<p>Blurring of the shadow. Above 40 only values of 40,50,60,70,80,90 mean something.</p> <p><b>Note:</b> Unfortunately the blurring capabilities of ImageMagick are not very mature in version 4.2.9. This is addressed in the later version 5.2.0 where a gaussian blur-function is added. BUT as we cannot use the latest ImageMagick development yet, this is not utilized so far.</p>	
<b>opacity</b>	posint (1-100)	Opacity (transparency <sup>-1</sup> ) 100% opacity = 0% transparency). Only active with a value for blur.	
<b>intensity</b>	posint(0-100)	How "massive" the shadow is. This value can - if it has a high value combined with a blurred shadow - create a kind of soft-edged outline.	

[tsref:->GIFBUILDER.(GBObj).SHADOW]

## EMBOSS

Emboss is actually two shadows offset in opposite directions and with different colors as to create an effect of light cast onto an embossed text.

Property:	Data type:	Description:	Default:
<b>textObjNum</b>	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
<b>offset</b>	x,y	Offset of the emboss	
<b>highColor</b>	GraphicColor	Upper border-color	
<b>lowColor</b>	GraphicColor	lower border-color	
<b>blur</b>	posint (1-99)	Blurring of the shadow. Above 40 only values of 40,50,60,70,80,90 means something.	
<b>opacity</b>	posint (1-100)	Opacity (transparency <sup>-1</sup> ) 100% opacity = 0% transparency). Only active with a value for blur.	
<b>intensity</b>	posint(0-100)	How "massive" the emboss is. This value can - if it has a high value combined with a blurred shadow - create a kind of soft-edged outline.	

[tsref:->GIFBUILDER.(GBObj).EMBOSS]

## OUTLINE

This outline normally renders quite ugly as it's done by printing 4 or 8 texts underneath the text in question. Try to use a shadow with a high intensity. That works better!

Property:	Data type:	Description:	Default:
<b>textObjNum</b>	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
<b>thickness</b>	x,y	Thickness in each direction, range 1-2	
<b>color</b>	GraphicColor	Outline color	

[tsref:->GIFBUILDER.(GBObj).OUTLINE]

## BOX

Property:	Data type:	Description:	Default:
<b>dimensions</b>	x,y,w,h +calc /stdWrap	Dimensions of a filled box. x,y is the offset. w,h are the dimensions. Dimensions of 1 will result in 1-pixel wide lines!	
<b>color</b>	GraphicColor	fill-color	black
<b>opacity</b>	pos-int (1-100)	Opacity (i.e. inverse of transparency, e.g. 100% opacity = 0% transparency)	100

Property:	Data type:	Description:	Default:
<b>align</b>	VHalign	<p>Pair of values, which defines the horizontal and vertical alignment.</p> <p><b>Values:</b> Horizontal alignment: r/c/l standing for right, center, left Vertical alignment: t/c/b standing for top, center, bottom</p> <p><b>Example:</b> Horizontally centered, vertically at the bottom: <code>align = c, b</code></p>	l, t

[tsref:->GIFBUILDER.(GBObj).BOX]

## ELLIPSE

Property:	Data type:	Description:	Default:
<b>dimensions</b>	x,y,w,h +calc /stdWrap	<p>Dimensions of a filled ellipse. x,y is the offset. w,h are the dimensions. Dimensions of 1 will result in 1-pixel wide lines!</p>	
<b>color</b>	GraphicColor	<p>fill-color</p> <p><b>Example:</b></p> <pre>file = GIFBUILDER file {   XY = 200,200   format = jpg   quality = 100   10 = ELLIPSE   10.dimensions = 100,100,50,50   10.color = red }</pre>	black

[tsref:->GIFBUILDER.(GBObj).ELLIPSE]

## IMAGE

Property:	Data type:	Description:	Default:
<b>file</b>	imgResource	The imagefile	
<b>offset</b>	x,y +calc /stdWrap	Offset of the image	0,0
<b>tile</b>	x,y	<p>tile x,y times. Maximum times is 20 each direction. If you need more, use a larger image.</p>	
<b>align</b>	VHalign	<i>See in the "Data types reference" at the beginning of this document or in the table "BOX".</i>	
<b>mask</b>	imgResource	Optional mask-image for the imagefile.	

[tsref:->GIFBUILDER.(GBObj).IMAGE]

## EFFECT

### Syntax:

.value = [Property] = [value] | [Property] = [value]

### Example:

```
lib.image = IMAGE
lib.image {
  file = GIFBUILDER
  file {
    XY = 1024,768
    format = jpg
    10 = IMAGE
```

```

10.file = fileadmin/image.jpg

20 = EFFECT
20.value = gamma=1.3 | flip | rotate=180
}
}

```

Property:	Data type:	Description:	Default:
<b>gamma</b>	0.5 - 3.0	Sets the gamma value.	1.0
<b>blur</b>	1-99	Blurs the edges inside the image.	0
<b>sharpen</b>	1-99	Sharpens the edges inside the image.	0
<b>solarize</b>	0-99	Color reduction.	
<b>swirl</b>	0-100	The image is swirled or spun from its center.	0
<b>wave</b>	amplitude, length	All horizontal edges are transformed by a wave with the given amplitude and length. Maximum value for amplitude and length is 100.  <b>Example:</b> 20 = EFFECT 20.value = wave=1,20	
<b>charcoal</b>	0-100	Makes the image look as if it had been drawn with charcoal and defines the intensity of that effect.	
<b>gray</b>	-	The image is converted to gray tones.  <b>Example:</b> This gives the image a slight wave and renders it in gray. 20 = EFFECT 20.value = wave=1,20   gray	
<b>edge</b>	0-99	Creates rounded edges.	
<b>emboss</b>	-	Creates a relief effect: Creates highlights or shadows that replace light and dark boundaries in the image.	
<b>flip</b>	-	Vertical flipping.	
<b>flop</b>	-	Horizontal flipping.	
<b>rotate</b>	0-360	Number of degrees for a clockwise rotation. Image dimensions will grow if needed, so that nothing is cut off from the original image.	0
<b>colors</b>	2-255	Defines the number of different colors to use in the image.	
<b>shear</b>	-90 - 90	Horizontal shearing.	
<b>invert</b>	-	Invert the colors.	

[tsref:->GIFBUILDER.(GBObj).EFFECT]

## WORKAREA

Sets another workarea.

Property:	Data type:	Description:	Default:
<b>set</b>	x,y,w,h + calc /stdWrap	Sets the dimensions of the workarea. x,y is the offset. w,h are the dimensions. For the usage of "calc" see the according note at the beginning of the section "GIFBUILDER".	
<b>clear</b>	(isset)	Sets the current to the default. Checked for using isset().	

[tsref:->GIFBUILDER.(GBObj).WORKAREA]

## CROP

**Note:** This object resets workArea to the new dimensions of the image!

Property:	Data type:	Description:	Default:
<b>backColor</b>	GraphicColor	See "Data types reference".	The original backColor
<b>align</b>	VHalign	Horizontal and vertical alignment of the crop frame. See "Data types reference".	l, t
<b>crop</b>	x,y,w,h + calc /stdWrap	x,y is the offset of the crop-frame from the position specified by "align". w,h are the dimensions of the frame. For the usage of "calc" see the according note at the beginning of the section "GIFBUILDER".	

```
[tsref:->GIFBUILDER.(GBObj).CROP]
```

## SCALE

**Note:** This object resets workArea to the new dimensions of the image!

Property:	Data type:	Description:	Default:
<b>width</b>	pixels + calc /stdWrap	Width of the scaled image.	
<b>height</b>	pixels + calc /stdWrap	Height of the scaled image.	
<b>params</b>	ImageMagickParams	Parameters to be used for the processing.	

```
[tsref:->GIFBUILDER.(GBObj).SCALE]
```

## ADJUST

This lets you adjust the tonal range like in the "levels"-dialog of Photoshop. You can set the input- and output-levels and that way remap the tonal range of the image. If you need to adjust the gamma value, have a look at the EFFECT-object.

### Example:

```
20 = ADJUST
20.value = inputLevels = 13, 230
```

Property:	Data type:	Description:	Default:
<b>inputLevels</b>	low, high	<p>With this option you can remap the tone of the image to make shadows darker, highlights lighter and increase contrast.</p> <p>Possible values for "low" and "high" are integers between 0 and 255, where "high" must be higher than "low". The value "low" will then be remapped to a tone of 0, the value "high" will be remapped to 255.</p> <p><b>Example:</b> This example will cause the tonal range of the resulting image to begin at 50 of the original (which is set as 0 for the new image) and to end at 190 of the original (which is set as 255 for the new image).</p> <pre>20 = ADJUST 20.value = inputLevels = 50, 190</pre>	

Property:	Data type:	Description:	Default:
<b>outputLevels</b>	low, high	<p>With this option you can remap the tone of the image to make shadows lighter, highlights darker and decrease contrast.</p> <p>Possible values for "low" and "high" are integers between 0 and 255, where "high" must be higher than "low". The beginning of the tonal range, which is 0, will then be remapped to the value "low", the end, which is 255, will be remapped to the value "high".</p> <p><b>Example:</b>  This example will cause the resulting image to have a tonal range, where there is no pixel with a tone below 50 and no pixel with a tone above 190 in the image.</p> <pre>20 = ADJUST 20.value = outputLevels = 50, 190</pre>	
<b>autoLevels</b>	-	Sets the levels automatically.	

[tsref:->GIFBUILDER.(GBObj).ADJUST]

## NON-GifBuilderObj

### IMGMAP

This is used by the GifBuilderObj "TEXT" to create an image-map for the gif-file. This is especially used with the IMGMENU menuobject.

Property:	Data type:	Description:	Default:
<b>url</b>	url	url to link	For IMGMENU menu objects provided automatically
<b>target</b>	target	target for link	For IMGMENU menu objects provided automatically
<b>noBlur</b>	Boolean	Normally graphical links are "blurred" if the browser is MSIE. This removes the ugly box around a link. If this property is set, the link is NOT blurred with "onFocus".	For IMGMENU menu objects provided automatically
<b>explode</b>	x,y	This "explodes" or "implodes" the image-map. Useful to let the hot area cover a little more than just the letters of the text.	
<b>altText</b>	string	Value of the alt-attribute. (Used from TEXT Gifbuilding objects, this has stdWrap properties. Otherwise not)	
<b>titleText</b>	string	Value of the title attribute. (Used from TEXT Gifbuilding objects, this has stdWrap properties. Otherwise not)	

[tsref:->IMGMAP]

# MENU Objects

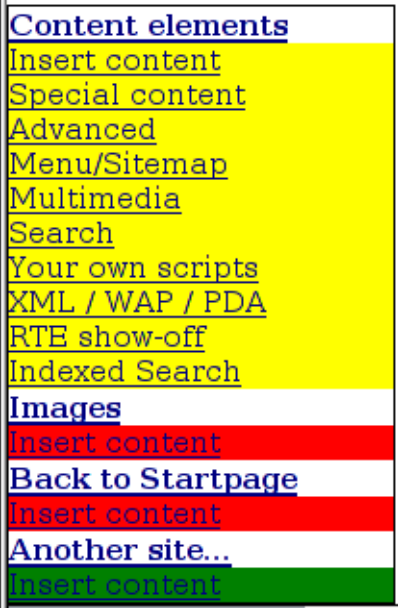
## Common properties

These properties are in common for *all* menu objects unless otherwise noted!

Property:	Data type:	Description:	Default:
<b>sectionIndex</b>		(see below)	
<b>alternativeSortingField</b>		<p>Normally the menuitems are sorted by the fields "sorting" in the pages- and tt_content-table. Here you can enter a list of fields that is used in the SQL- "ORDER BY" statement instead.</p> <p><b>Examples (for "pages" table):</b>  alternativeSortingField = title desc  (This will render the menu in reversed alphabetical order.)</p> <p><b>LIMITATIONS:</b>  This property works with normal menus, sectionsIndex menus and special-menus of type "directory".</p>	
<b>minItems</b>	int	<p>The minimum items in the menu. If the number of pages does not reach this level, a dummy-page with the title "..." and uid=[currentpage_id] is inserted.</p> <p>Takes precedence over HMENU.minItems.</p>	
<b>maxItems</b>	int	<p>The maximum items in the menu. More items will be ignored.</p> <p>Takes precedence over HMENU.maxItems.</p>	
<b>begin</b>	int +calc	<p>The first item in the menu.</p> <p><b>Example:</b>  This results in a menu, where the first two items are skipped starting with item number 3:  begin = 3</p> <p>Takes precedence over HMENU.begin.</p>	
<b>JSWindow</b>	boolean	<p>If set, the links of the menu-items will open by JavaScript in a pop-up window.</p> <p><b>.newWindow</b> boolean, that lets every menuitem open in its own window opposite to opening in the same window for each click.</p> <p><b>.params</b> is the list of parameters sent to the JavaScript open-window function, e.g.:  width=200,height=300,status=0,menubar=0</p> <p><b>Note:</b> Does not work with JSMENU's.</p>	
<b>imageNamePrefix</b>	string	prefix for the imagenames. This prefix is appended with the uid of the page.	"img"
<b>imageNameNotRandom</b>	boolean	<p>If set, the image names of menuitems is not randomly assigned. Useful switch if you're manipulating these images with some external JavaScript.</p> <p><b>Note:</b> Don't set this if you're working with a menu with sectionIndex! In that case you need special unique names of items based on something else than the uid of the parent page of course!</p>	

Property:	Data type:	Description:	Default:
<b>debugItemConf</b>		Outputs (by the debug()-function) the configuration arrays for each menuitem. Useful to debug optionSplit things and such... Applies to GMENU, TMENU and IMGMENU.	
<b>overrideId</b>	integer (page-id)	If set, then all links in the menu will point to this pageid. Instead the real uid of the page is sent by the parameter "&real_uid=[uid]". This feature is smart, if you have inserted a menu from somewhere else, perhaps a shared menu, but wants the menuitems to call the same page, which then generates a proper output based on the real_uid. Applies to GMENU, TMENU and IMGMENU.	
<b>addParams</b>	string	Additional parameter for the menu-links.  <b>Example:</b> "&some_var=some%20value" Must be rawurlencoded. Applies to GMENU, TMENU and IMGMENU.	
<b>showAccessRestrictedPages</b>	integer (page id) / keyword "NONE"	If set, pages in the menu will include pages with frontend user group access enabled. However the page is of course not accessible and therefore the URL in the menu will be linked to the page with the ID of this value. On that page you could put a login form or other message. If the value is "NONE" the link will not be changed and the site will perform page-not-found handling when clicked (which can be used to capture the event and act accordingly of course).  <b>Properties:</b> .addParam = Additional parameter for the URL, which can hold two markers; ###RETURN_URL### which will be substituted with the link the page would have had if it had been accessible and ###PAGE_ID### holding the page id of the page coming from (could be used to look up which fe_groups was required for access).  <b>Example:</b> <pre>showAccessRestrictedPages = 22 showAccessRestrictedPages.addParams = &amp;return_url=###RETURN_URL###&amp;pageId=###PAGE_ID###</pre> The example will link access restricted menu items to page id 22 with the return URL in the GET var "return_url" and the page id in the GET var "pageId".	
<b>itemArrayProcFunc</b>	function name	The first variable passed to this function is the "menuArr" array with the menuitems as they are collected based on the type of menu. You're free to manipulate or add to this array as you like. Just remember to return the array again!  <b>Note:</b> .parentObj property is <u>hardcoded</u> to be a reference to the calling tslib_menu object. Here you'll find e.g. ->id to be the uid of the menu item generating a submenu and such.  <b>Presetting element state</b> You can override element states like SPC, IFSUB, ACT, CUR or USR by setting the key ITEM_STATE in the page records. See cObject HMENU/special=userdefined for more information.	



Property:	Data type:	Description:	Default:
<b>submenuObjSuffixes</b>	string +optionSplit	<p>Defines a suffix for alternative sub-level menu objects. Useful to create special submenus depending on their parent menu element. See example below.</p> <p><b>Example:</b> This example will generate a menu where the menu objects for the second level will differ depending on the number of the first level item for which the submenu is rendered. The second level objects used are "2" (the default), "2a" and "2b" (the alternatives). Which of them is used is defined by "1.submenuObjSuffixes" which has the configuration "a  *   *  b". This configuration means that the first menu element will use configuration "2a" and the last will use "2b" while anything in between will use "2" (no suffix applied)</p> <pre> page.200 = HMENU page.200 {   1 = TMENU   1.wrap = &lt;div style="width:200px; border: 1px solid;"&gt; &lt;/div&gt;   1.expAll = 1   1.submenuObjSuffixes = a  *   *  b   1.NO.allWrap = &lt;b&gt; &lt;/b&gt;&lt;br/&gt;    2 = TMENU   2.NO.allWrap = &lt;div style="background:red;"&gt; &lt;/div&gt;    2a = TMENU   2a.NO.allWrap = &lt;div style="background:yellow;"&gt; &lt;/div&gt;    2b = TMENU   2b.NO.allWrap = &lt;div style="background:green;"&gt; &lt;/div&gt; } </pre> <p>The result can be seen in the image below (applied on the testsite package):</p>  <p>The image shows a menu structure with the following items:</p> <ul style="list-style-type: none"> <li><u>Content elements</u> (blue link)</li> <li><u>Insert content</u> (yellow background)</li> <li><u>Special content</u> (yellow background)</li> <li><u>Advanced</u> (yellow background)</li> <li><u>Menu/Sitemap</u> (yellow background)</li> <li><u>Multimedia</u> (yellow background)</li> <li><u>Search</u> (yellow background)</li> <li><u>Your own scripts</u> (yellow background)</li> <li><u>XML / WAP / PDA</u> (yellow background)</li> <li><u>RTE show-off</u> (yellow background)</li> <li><u>Indexed Search</u> (yellow background)</li> <li><u>Images</u> (blue link)</li> <li><u>Insert content</u> (red background)</li> <li><u>Back to Startpage</u> (blue link)</li> <li><u>Insert content</u> (red background)</li> <li><u>Another site...</u> (blue link)</li> <li><u>Insert content</u> (green background)</li> </ul> <p>Applies to GMENU, TMENU, GMENU_LAYERS, TMENU_LAYERS and GMENU_FOLDOUT on &gt;= 2<sup>nd</sup> level in a menu.</p>	

[tsref:(cObject).HMENU.(mObj)]

## Common item states for TMENU, GMENU and IMGMENU series:

These properties are in common for TMENU, GMENU and IMGMENU series. That means they are not used by for instance the JSMENU.

Property:	Data type:	Description:	Default:
<b>NO</b>	Boolean / (config)	<p>The default "Normal" state rendering of Item. This is required for all menus.</p> <p>If you specify properties for the "NO" property you do not have to set it "1". Otherwise with no properties setting "NO=1" will render the menu anyways (for TMENU this may make sense).</p> <p>The simplest menu TYPO3 can generate is then:</p> <pre>page.20 = HMENU page.20.1 = TMENU page.20.1.NO = 1</pre> <p>That will be pure &lt;a&gt; tags wrapped around page titles.</p>	1
<b>IFSUB</b> <b>IFSUBRO</b>	Boolean / (config)	Enable/Configuration for menu items which has subpages.	0
<b>ACT</b> <b>ACTRO</b>	Boolean / (config)	Enable/Configuration for menu items which are found in the rootLine.	0
<b>ACTIFSUB</b> <b>ACTIFSUBRO</b>	Boolean / (config)	Enable/Configuration for menu items which are found in the rootLine and have subpages.	0
<b>CUR</b> <b>CURRO</b>	Boolean / (config)	Enable/Configuration for a menu item if the item is the current page.	0
<b>CURIFSUB</b> <b>CURIFSUBRO</b>	Boolean / (config)	Enable/Configuration for a menu item if the item is the current page and has subpages.	0
<b>USR</b> <b>USRRO</b>	Boolean / (config)	Enable/Configuration for menu items which are access restricted pages that a user has access to.	0
<b>SPC</b>	Boolean / (config)	<p>Enable/Configuration for 'Spacer' pages.</p> <p>Spacers are pages of the doktype "Spacer". These are not viewable pages but "placeholders" which can be used to divide menuitems.</p> <p><b>Note:</b> Rollovers doesn't work with spacers, if you use GMENU!</p>	0
<b>USERDEF1</b> <b>USERDEF1RO</b>	Boolean / (config)	<p>Userdefined, see .itemArrayProcFunc for details on how to use this.</p> <p>You can set the ITEM_STATE values USERDEF1 and USERDEF2 (+...RO) from a script/userfunction processing the menu item array. See HMENU/special=userdefined or the property .itemArrayProcFunc of the menu objects.</p>	
<b>USERDEF2</b> <b>USERDEF2RO</b>	Boolean / (config)	(See above)	

[tsref:(cObject).HMENU.(mObj\_itemStates)]

Order of priority: USERDEF2, USERDEF1, SPC, USR, CURIFSUB, CUR, ACTIFSUB, ACT, IFSUB  
All \*RO states require the default "RO" configuration to be set up.

## [menuObj].sectionIndex

This is a property that all menuObj's share. If it's set, then the menu will not consist of links to pages on the "next level" but rather links to the parent page to the menu, but in addition "#" -links to the cObjects rendered on the page. In other words, the menu items will be links to the content elements (with colPos=0!) on the page. A section index.

.sectionIndex = [boolean]

If you set this, all content elements (from tt\_content table) of "Column" = "Normal" *and* the "Index"-checkbox clicked are selected. This corresponds to the "Menu/Sitemap" content element when "Section index" is selected as type.

.sectionIndex.type = "all" / "header"

If you set this additional property to "all", then the "Index"-checkbox is not considered and all content elements with colPos=0 is selected.

If this property is "header" then only content elements with a visible header-layout (and a non-empty 'header'-field!) is selected. In other words, if the header layout of an element is set to "Hidden" then the page will not appear in the menu.

### The data-record /Behind the scene

When the menu-records are selected it works like this: The parent page record is used as the "base" for the menu-record. That means that any "no\_cache" or "target"-properties of the parent page is used for the whole menu.

But of course some fields from the tt\_content records are transferred. This is how it mapped:

```
$temp[$row[uid]]=$basePageRow;
$temp[$row[uid]]['title']=$row['header'];
$temp[$row[uid]]['subtitle']=$row['subheader'];
$temp[$row[uid]]['starttime']=$row['starttime'];
$temp[$row[uid]]['endtime']=$row['endtime'];
$temp[$row[uid]]['fe_group']=$row['fe_group'];
$temp[$row[uid]]['media']=$row['media'];
$temp[$row[uid]]['header_layout']=$row['header_layout'];
$temp[$row[uid]]['bodytext']=$row['bodytext'];
$temp[$row[uid]]['image']=$row['image'];
$temp[$row[uid]]['sectionIndex_uid']=$row['uid'];
```

Basically this shows that

- the field "header" and "subheader" from tt\_content are mapped to "title" and "subtitle" in the pages-record. Thus you shouldn't need to change your standard menu-objects to fit this thing...
- the fields "starttime", "endtime", "fe\_group", "media" from tt\_content are mapped to the same fields in a pages-record.
- the fields "header\_layout", "bodytext" and "image" are mapped to non-existing fields in the page-record
- a new field, "sectionIndex\_uid" is introduced in the page record which is detected by the function t3lib\_tstemplate->linkData(). If this field is present in a page record, the linkData()-function will prepend a hash-mark and the number of the field.

#### Note:

You cannot create submenus to sectionIndex-menus. That doesn't make any sense as these elements are not pages and thereby have no children.

## GMENU

GMENU works as an object under the cObject "HMENU" and it creates graphical navigation, where each link is a separate gif-file.

Property:	Data type:	Description:	Default:
<b>RO</b>	Boolean	RollOver configuration enabled / disabled. If this is true, RO becomes a GIFBUILDER-object defining the layout of the menu item when the mouse rolls over it	0
<b>expAll</b>	Boolean	If this is true, the menu will always show the menu on the level underneath the menu item. This corresponds to a situation where a user has clicked a menu item and the menu folds out the next level. This can enable that to happen on all items as default.	
<b>collapse</b>	Boolean	If set, "active" menu items that has expanded the next level on the menu will now collapse that menu again.	
<b>accessKey</b>	Boolean	If set access-keys are set on the menu-links	
<b>noBlur</b>	Boolean	Normally graphical links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
<b>target</b>	target	Target of the menu links	self
<b>forceTypeValue</b>	int	If set, the &type parameter of the link is forced to this value regardless of target. Overrides the global equivalent in 'config' if set.	
<b>stdWrap</b>	->stdWrap	Wraps the whole item using stdWrap  <b>Example:</b> <pre> 2 = TMENU 2 {     stdWrap.dataWrap = &lt;ul class="{register : parentProperty}"&gt;   &lt;/ul&gt; NO {     ... } </pre>	
<b>wrap</b>	wrap	Wraps only if there were items in the menu!	
<b>applyTotalH</b>	objNumsList (offset)	This adds the total height of the previously generated menu items to the offset of the GifBuilderObj's mentioned in this list.  <b>Example:</b> This is useful if you want to create a menu with individual items but a common background image that extends to the whole area behind the menu. Then you should setup the background image in each GIFBUILDER-object and include the object-number in this list. Look at the implementation in static_template "styles.gmenu.bug"	
<b>applyTotalW</b>	objNumsList (offset)	This adds the total width of the previously generated menu items to the offset of the GifBuilderObj's mentioned in this list.	
<b>min</b>	x,y (calcInt)	Forces the menu as a whole to these minimum dimensions	
<b>max</b>	x,y (calcInt)	Forces the menu as a whole to these maximum dimensions	
<b>useLargestItemX</b>	boolean	If set, then the width of all menu items will be equal to the largest of them all.	
<b>useLargestItemY</b>	boolean	If set, then the height of all menu items will be equal to the largest of them all.	

Property:	Data type:	Description:	Default:
<b>distributeX</b>	int+	If set, the total width of all the menu items will be equal to this number of pixels by adding/subtracting an equal amount of pixels to each menu items width. Will overrule any setting for ".useLargestItemX"	
<b>distributeY</b>	int+	If set, the total height of all the menu items will be equal to this number of pixels by adding/subtracting an equal amount of pixels to each menu items height. Will overrule any setting for ".useLargestItemY"	
<b>removeObjectsOfDummy</b>	objNumsList	If the menu is forced to a certain minimum dimension, this is a list of objects in the gifbuilder-object that is removed for this last item. This is important to do if the menu items has elements that should only be applied if the item is actually a menu item!!	
<b>disableAltText</b>	boolean	If set, the alt-parameter of the images are not set. You can do it manually by "imgParams" (see below)	
<b>IProcFunc</b>	function name	The internal array "I" is passed to this function and expected returned as well. Subsequent to this function call the menu item is compiled by implode()'ing the array \$I[parts] in the passed array. Thus you may modify this if you need to. See example in typo3/sysect/cms/tslib/media/scripts/example_itemArrayProcFunc.php	
<b>[Common Item States, see above] + rollover version for all, except SPC</b>	->GIFBUILDER + Additional properties! See table below	This is the GIFBUILDER-options for each category of menu item that can be generated.  <b>Note:</b> For the GMENU series you can also define the RollOver configuration for the item states. This means that you define the GIFBUILDER object for the 'Active' state by ACT and the RollOver GIFBUILDER object for the 'Active' state by ACTRO. This pattern goes for ALL the states except the SPC state.  <b>SPECIAL:</b> The ->OptionSplit function is run on the whole GIFBUILDER-configuration before the items are generated.	

[tsref:(cObject).HMENU.(mObj).GMENU

## Additional properties for Menu item states

These properties are additionally available for the GMENU item states although the main object is declared to be GIFBUILDER.

It is evident that it is an unclean solution to introduce these properties on the same level as the GIFBUILDER object in a single situation like this. However this is how it irreversibly is and has been for a long time.

Property:	Data type:	Description:	Default:
<b>noLink</b>	boolean	If set, the item is NOT linked!	
<b>imgParams</b>	params	Parameters for the <img>-tag	
<b>altTarget</b>	string	Alternative target which overrides the target defined for the GMENU	
<b>altImgResource</b>	imgResouce	Defines an alternative image to use. If an image returns here, it will override any GIFBUILDER configuration.	
<b>ATagParams</b>	string /stdWrap	Additional parameters	
<b>ATagTitle</b>	string /stdWrap	which defines the title attribute of the a-tag. (See TMENUITEM also)	

Property:	Data type:	Description:	Default:
<b>additionalParams</b>	string /stdWrap	Define parameters that are added to the end of the URL. This must be code ready to insert after the last parameter.  For details, see typolink->additionalParams	
<b>wrap</b>	wrap	Wrap of the menu item.	
<b>allWrap</b>	wrap /stdWrap	Wraps the whole item.	
<b>wrapItemAndSub</b>	wrap /stdWrap	Wraps the whole item and any submenu concatenated to it.	
<b>subst_elementUid</b>	boolean	If set, "{elementUid}" is substituted with the item uid.	
<b>allStdWrap</b>	->stdWrap	stdWrap of the whole item	

[tsref:(cObject).HMENU.(mObj).GMENU.(itemState)]

## GMENU\_LAYERS / TMENU\_LAYERS

GMENU\_LAYERS / TMENU\_LAYERS works as an extension to GMENU/TMENU, which means the these properties underneath is additional properties to the ones above.

The purpose of xMENU\_LAYERS is to create 2-level (or more!) menus where the 2nd+ level is shown on a DHTML-layer. Most features works with modern browsers including Netscape, Microsoft Internet Explorer, Mozilla, Konqueror and Opera. You can cascade the menus as you like.

**Note:** You must include the library "typo3/sysex/cms/tslib/media/scripts/gmenu\_layers.php" (for GMENU\_LAYERS) and/or "typo3/sysex/cms/tslib/media/scripts/tmenu\_layers.php" (for TMENU\_LAYERS) and you must also expand the xMENU\_LAYERS to the next for the menu to make sense (use the expAll-flag).

**Compatibility:** MSIE 4+, Netscape 4+ and 6+, Opera 5+, Konqueror.

### Notes:

- Netscape 4 does not support mouseover on the layers.
- Opera seems to have problems with the mouseout event if you roll from an element to a layer. Then the event may not be fired before entering the layer. It happens only if the layer is placed very close to the trigger element. Problems from this may be that the rollover state of the items are not reset.
- Possible bug; It has been seen with cascaded layers that Opera may suddenly refuse any interaction on the page, even clicking normal links. It may be a JavaScript error that makes this happen, but as even normal links are not clickable anymore, I'm not really sure. Seems to be no problem with single-level menu.

Property:	Data type:	Description:	Default:
<b>layerStyle</b>	<DIV>-tag params	Parameters for the <DIV>-layer-tags in the HTML-document. You might probably not need change this.  <b>Example:</b> position: absolute; VISIBILITY: hidden;	position: absolute; visibility: hidden;
<b>lockPosition</b>	"x" / "y" / ""	If this is set to "x" or "y" the menu on the layers is locked and does not follow the mouse-cursor (which it does if this is not set). "x" or "y" defines respectively that the summed width (x) or height (y) is added to the x or y offset of the menu. That means that you should set this value to "x" if you have a horizontal GMENU_LAYERS and to "y" if you have a vertical menu.	
<b>dontFollowMouse</b>	boolean	If set and lockPosition is blank (so that the menu layer follows the mouse) then the menu will NOT follow the mouse but still it will appear where the mouse cursor hit the trigger-element. Useful if you don't know the exact positions of elements.	

Property:	Data type:	Description:	Default:
		<b>Warning:</b> You should not set displayActiveOnLoad for menus with this feature enabled (because the absolute position of the layer is not known).	
<b>lockPosition_adjust</b>	int	A number which is added to the width/height of the menu items in order to compensate for e.g. hspace or other things between the images in the GMENU_LAYERS	
<b>lockPosition_addSelf</b>	boolean	Normally the width and height of the items (+lockPosition_adjust) are summed up after the item has been rendered. This is good if the direction of the menu layers is right- og downwards. But if you use directionLeft/directionUp, you might want to add the width of the items before. If so, set this flag.	
<b>xPosOffset</b>	int	The offset of the menu from the point where it's "activated" (if lockPosition is false) / from top left page corner (if lockPosition is set)	
<b>yPosOffset</b>	int	As above, but for the y-dimension.	
<b>topOffset</b>	int	The offset of menu items from top of browser. Should be set rather than defining it in the .layerStyle property. Must be set in order to use directionUp. Used with either lockPosition=x or xPosOffset defined.	
<b>leftOffset</b>	int	The offset of menu items from left border of browser. Should be set rather than defining it in the .layerStyle property. Must be set in order to use directionLeft. Used with either lockPosition=y or yPosOffset defined.	
<b>blankStrEqFalse</b>	boolean	If set, then the properties topOffset,leftOffset, xPosOffset, yPosOffset are considered "blank" if they are really blank strings - not just "zero". You should enable this if you wish to be able to work with zero offsets. This is typically the case if you use relative positioning.	
<b>directionLeft</b>	boolean	Set this, if you want the items to be right-aligned (pop's out towards the left). Does not work with Opera at this time because I don't know how to make Opera read the width of each layer. If you set the width of the menu-layers in .layerStyles this might work no matter what.	
<b>directionUp</b>	boolean	Set this, if you want the items to be bottom-aligned (pop's out upwards instead of downwards).	
<b>setFixedWidth</b>	int	For GMENU_LAYERS the width and heights of the element is normally known from the graphical item. For TMENU_LAYERS this cannot be known in the same way. Therefore you can use .setFixedWidth and .setFixedHeight to set these values to a number you find reasonable. Of course this may be blasted by the browsers rendering if the font gets out of proportions etc.  Alternatively you may want to use the property "relativeToTriggerItem" which will position your menu layers relative to the item you roll over. This has some drawbacks though. A middle solution is to use a menu with lockPosition set to blank and dontFollowMouse set to true. Then you need only specify either an x or y coordinate to follow and the item will appear where the mouse hits the element. <b>Notice:</b> Active if value is NOT a blank str. Setting this value to zero means that no width is calculated for the items in GMENU_LAYERS.	
<b>setFixedHeight</b>	int	See "setFixedWidth". Same, but for height.	

Property:	Data type:	Description:	Default:
<b>bordersWithin</b>	l,t,r,b,l,t	Keep borders of the layer within these limits in pixels. Zero is 'not set' (Syntax: List of integers, evaluated clockwise: Left, Top, Right, Bottom, Left, Top)	
<b>displayActiveOnLoad</b>	boolean	If set, the submenu-layer of the active menu item is opened at page-load. If .freezeMouseover is also set and there is RO defined for the main menu items, the menu item belonging to the displayed submenu is also shown.  <b>Properties:</b> .onlyOnLoad (boolean) If set, then the display of the active item will happen only when the page is loaded. The display will not be restored on mouseout of other items.  <b>Warning:</b> If you are cascading GMENU_LAYER objects, make sure that all elements before this element (for which you set this attribute) also have this attribute set!	
<b>freezeMouseover</b>	boolean	If set, any mouseout effect of main menu items is removed not on roll-out but when another element is rolled over (or the layer is hidden/default layer restored)  <b>Properties:</b> .alwaysKeep (boolean) If set, the frozen element will always stay, even if the submenu is hidden.	
<b>hideMenuWhenNotOver</b>	int+	If set (> 1) then the menu will hide it self whenever a user moves the cursor away from the menu. The value of this parameter determines the width (pixels) of the zone around the element until the mouse pointer is considered to be far enough away to hide the layer.	
<b>hideMenuTimer</b>	int+	This is the number of milliseconds to wait before the submenu will disappear if hideMenuWhenNotOver is set.	
<b>dontHideOnMouseUp</b>	boolean	If set, the menu will not hide its layers when the mouse button is clicked. Useful if your menu items loads the pages in another frame.	
<b>layer_menu_id</b>	string	If you want to specifically name a menu on a page. Probably you don't need that!  <b>Warning:</b> Don't use underscore and special characters in this string. Stick to alpha-numeric characters.	[random 6 char hashstring]
<b>relativeToTriggerItem</b>	boolean	This allows you to position the menu layers relative to the item that triggers it. However you should be aware of the following facts: <ul style="list-style-type: none"> <li>• This does not work with Netscape 4 - the position of the trigger layer will be calculated to zero and thus the offset for all menu layers will be 0,0 + your values.</li> <li>• This feature will wrap the menu item in some &lt;div&gt;-tags right before the whole item is wrapped by the .wrap code (for GMENU_LAYERS) or .allWrap (for TMENU_LAYERS). The bottom line of this is: 1) If your menu is horizontal, always wrap your menu items in a table so line breaks does not appear because of the &lt;div&gt;-tags and 2) make sure the wrapping of the table cell is done with the .wrap/.allWrap properties respectively.</li> <li>• Works only effectively on the first xMENU_LAYER in a cascade. For succeeding xMENU_LAYERS</li> </ul>	



Property:	Data type:	Description:	Default:
		<p>items please use "relativeToParentLayer".  <i>If set, properties xPosOffset, yPosOffset and lockPosition* are not functional (properties directionLeft, directionUp, topOffset and leftOffset are still active)</i></p> <p><b>Additional Properties:</b>  <b>.addWidth</b> = Adds the width of the trigger element  <b>.addHeight</b> = Adds the height of the trigger element</p>	
<b>relativeToParentLayer</b>	boolean	<p>If set, then the layer will be positioned relative to the previous layer (parent) in a cascaded series of xMENU_LAYERS. Basically the relative position of the parent layer is just added to the offset of the current menu.</p> <p><b>Warning:</b> This property makes sense only if there really is a previous GMENU_LAYER to get position from! So you must have a cascaded menu!</p> <p><b>Additional Properties:</b>  <b>.addWidth</b> = Adds the width of the parent layer  <b>.addHeight</b> = Adds the height of the parent layer</p>	

[tsref:(cObject).HMENU.(mObj).GMENU\_LAYERS, (cObject).HMENU.(mObj).TMENU\_LAYERS]

**Example:**

```

page.includeLibs.gmenu_layers = media/scripts/gmenu_layers.php
page.10 = HMENU
page.10.1 = GMENU_LAYERS
page.10.1 {
    layerStyle = position: absolute; VISIBILITY: hidden;
    xPosOffset = -30
    lockPosition = x
    expAll=1
    leftOffset = 15
    topOffset = 30
}
page.10.1.NO {
    backColor = #cccccc
    XY = [10.w]+10, 14
    10 = TEXT
    10.text.field = title
    10.offset = 5,10
}
page.10.2 = GMENU
page.10.2.wrap = <nobr>|</nobr>
page.10.2.NO {
    backColor = #99cccc
    XY = [10.w]+10, 14
    10 = TEXT
    10.text.field = title
    10.offset = 5,10
}

```

## GMENU\_FOLDOUT

GMENU\_FOLDOUT works as an extension to GMENU, which means the these properties underneath is additional properties to the ones above.

The purpose of GMENU\_FOLDOUT is to create 2-level menus which are folded out dynamically.

It works with both Netscape, Mozilla, Microsoft internet Explorer and Opera. The menu on the first level is a GMENU because GMENU\_FOLDOUT is responsible for this, but the submenu on the next level (referred to as 2nd level) can be both TMENU and another GMENU.

**NOTE:** You must include the library "typo3/sysex/cms/tslib/media/scripts/gmenu\_foldout.php".

The script implemented is taken from [http://www9.ewebcity.com/skripts/foldoutmenu\\_move.htm](http://www9.ewebcity.com/skripts/foldoutmenu_move.htm)

**Compatibility:** MSIE 4+, Netscape 4+ and 6+, Opera 5+

Property:	Data type:	Description:	Default:
<b>dontLinkIfSubmenu</b>	boolean	If set, items that has a submenu is not linked. Items without a submenu are always linked in the regular ways.	
<b>foldTimer</b>	int	The timeout in the animation, these are milliseconds.	40
<b>foldSpeed</b>	int, range 1-100	How many steps in an animation? Choose 1 for no animation.	1
<b>stayFolded</b>	boolean	Stay open when you click a new topline? (Level 1)	
<b>bottomHeight</b>	int, pixels	Sets the height of the bottom layer. Is important if the bottom layer contains either content or a background color: Else the layer will be clipped.	100
<b>menuWidth</b>	int, pixels	Width of the whole menu main layer. Important to set, especially for the bottom layer as it is clipped by this value. Always try to set this to the width in pixels of the menu.	170
<b>menuHeight</b>	int	Height of the whole menu layer. Seems not to be not that important.	400
<b>subMenuOffset</b>	x,y	Offset of the submenu for each menu item. This is important because if you don't set this value the items will appear on top of their "parent".	
<b>menuOffset</b>	x,y	Offset of the menu main layer on the page. From upper left corner	
<b>menuBackColor</b>	HTML-color	Background color behind menu. If not set, transparent (which will not work very well in case .foldSpeed is set to something else than 1. But see for yourself)	
<b>dontWrapInTable</b>	boolean	By default every menu item on the first level is wrapped in a table: <code>&lt;TABLE cellSpacing=0 cellPadding=0 width="100%" border=0&gt;&lt;TR&gt;&lt;TD&gt;</code> [menu item HTML here..] <code>&lt;/TD&gt;&lt;/TR&gt;&lt;/TABLE&gt;</code> Doing this ensures that the layers renders equally in the supported browsers. However you might need to disable that which is what you can do by setting this flag. <b>Note:</b> Using <code>&lt;TBODY&gt;</code> in this tables seems to break Netscape 4+	0
<b>bottomContent</b>	cObject	Content for the bottom layer that covers the end of the menu.	
<b>adjustItemsH</b>	int	Adjusts the height calculation of the menulayers of the first level (called Top)  <b>Example:</b> - 10  This value will subtract 10 pixels from the height of the layer in calculations.	
<b>adjustSubItemsH</b>	int	Adjusts the height calculation of the menu layers of the second level (subitems, called Sub) See above	
<b>arrowNO</b> <b>arrowACT</b>	imgResource	If both arrowNO and arrowACT is defined and valid imgResources then these images are use as "traditional arrows" that indicates whether an item is expanded (active) or not. NO is normal, ACT is expanded The image is inserted just before the menu item. If you want to change the position, put the marker <code>###ARROW_IMAGE###</code> into the wrap of the item and the image will be put there instead.	
<b>arrowImgParams</b>	<img> params	Parameters to the arrow-image.  <b>Example:</b> hspace=5 vspace=7	

Property:	Data type:	Description:	Default:
<b>displayActiveOnLoad</b>	boolean	If set, the active menu items will fold out "onLoad".	

[tsref:(cObject).HMENU.(mObj).GMENU\_FOLDOUT]

### Example:

```
## GMENU_FOLDOUT
includelibs.gmenu_foldout = typo3/sysex/cms/tslib/media/scripts/gmenu_foldout.php

temp.foldoutMenu = HMENU
temp.foldoutMenu.1 = GMENU_FOLDOUT
temp.foldoutMenu.1.expAll = 1
temp.foldoutMenu.1.NO {
    wrap = | <br>
    XY = 150,20
    backColor = silver

    10 = TEXT
    10.text.field = title
    10.fontSize = 12
    10.fontColor = Blue
    10.offset = 2,10
}
temp.foldoutMenu.1.R0 < temp.foldoutMenu.1.NO
temp.foldoutMenu.1.R0 = 1
temp.foldoutMenu.1.R0 {
    10.fontColor = red
}
temp.foldoutMenu.2 = TMENU
temp.foldoutMenu.2.NO {
    linkWrap = <nobr><font face=verdana size=1 color=black><b>|</b></font></nobr><br>
    stdWrap.case = upper
}
temp.foldoutMenu.1 {
    dontLinkIfSubmenu = 1
    stayFolded=1
    foldSpeed = 6
    subMenuOffset = 10,18
    menuOffset = 100,20
    menuBackColor = silver
    bottomBackColor = silver
    menuWidth = 170

    arrowNO = typo3/sysex/cms/tslib/media/bullets/arrow_no.gif
    arrowACT = typo3/sysex/cms/tslib/media/bullets/arrow_act.gif
    arrowImgParams = hspace=4 align=top

    bottomContent = TEXT
    bottomContent.value = Hello World! Here is some content!
}
```

#### ▼ Content elements

[INSERT CONTENT](#)  
[SPECIAL CONTENT](#)  
[ADVANCEDASDF ASDF](#)  
[MENU/SITEMAP](#)  
[MULTIMEDIA!D SDSDF](#)  
[SEARCH](#)  
[YOUR OWN SCRIPTS](#)

- ▶ New, new page!
- ▶ Images
- ▶ Back to Startpage
- ▶ Another site...

Hello World! Here is some content!

This creates a menu like this (above). One important point is the line

```
temp.foldoutMenu.1.expAll = 1
```

If you don't set this (just like the GMENU\_LAYERS) then the second level is not generated!

## TMENU

Property:	Data type:	Description:	Default:
<b>expAll</b>	Boolean /stdWrap	If this is true, the menu will always show the menu on the level underneath the menu item. This corresponds to a situation where a user has clicked a menu item and the menu folds out the next level. This can enable that to happen on all items as default.	
<b>collapse</b>	boolean	If set, "active" menu items that has expanded the next level on the menu will now collapse that menu again.	
<b>accessKey</b>	boolean	If set access-keys are set on the menu-links	
<b>noBlur</b>	boolean	Normally links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
<b>target</b>	target	Target of the menu links	self
<b>forceTypeValue</b>	int	If set, the &type parameter of the link is forced to this value regardless of target.	
<b>stdWrap</b>	->stdWrap	Wraps the whole item using stdWrap  <b>Example:</b> see GMENU.stdWrap	
<b>wrap</b>	wrap	Wraps only if there were items in the menu!	
<b>IProcFunc</b>	function name	The internal array "I" is passed to this function and expected returned as well. Subsequent to this function call the menu item is compiled by implode()'ing the array \$I[parts] in the passed array. Thus you may modify this if you need to. See example in typo3/sysex/cms/tslib/media/scripts/example_itemArrayProcFunc.php	
<b>[Common Item States, see above]</b>	->TMENUITEM	This is the TMENUITEM-options for each category of menu item that can be generated.  <b>SPECIAL:</b> The ->OptionSplit function is run on the whole configuration before the items are generated.	

[tsref:(cObject).HMENU.(mObj).TMENU]

## TMENUITEM

The current record is the page-record of the menu item - just like you have it with GMENU/gifbuilder. Now, if you would like to get data from the current page record, use stdWrap.data = page : [field name]

Property:	Data type:	Description:	Default:
<b>allWrap</b>	wrap /stdWrap	Wraps the whole item.	
<b>wrapItemAndSub</b>	wrap /stdWrap	Wraps the whole item and any submenu concatenated to it.	
<b>subst_elementUid</b>	boolean	If set, all appearances of the string '{elementUid}' in the total element html-code (after wrapped in .allWrap) is substituted with the uid number of the menu item. This is useful if you want to insert an identification code	

Property:	Data type:	Description:	Default:
		in the HTML in order to manipulate properties with JavaScript.	
<b>RO_chBgColor</b>	string	<p>If property RO is set (see below) then you can set this property to a certain set of parameters which will allow you to change the background color of e.g. the table cell when the mouse rolls over you text-link.</p> <p><b>Syntax:</b></p> <pre>[over-color]   [out-color]   [id-prefix]</pre> <p><b>Example:</b></p> <pre>page = PAGE page.typeNum = 0 page.10 = HMENU page.10.wrap = &lt;table border=1&gt; &lt;/table&gt; page.10.1 = TMENU page.10.1.NO {   allWrap = &lt;tr&gt;&lt;td valign=top id="lmenu{elementUid}" style="background:#eeeeee;"&gt; &lt;/td&gt;&lt;/tr&gt; subst_elementUid = 1   RO_chBgColor = #cccccc   #eeeeee   lmenu   RO = 1 }</pre> <p>This example will start out with the table cells in #eeeeee and change them to #cccccc (and back) when rolled over. The "lmenu" string is a unique id for the menu items. You may not need it (unless the same menu items are more than once on a page), but the important thing is that the id of the table cell has the exact same label before the {elementUid} (red marks). The other important thing is that you DO set a default background color for the cell with the style-attribute (blue marking). If you do not, Mozilla browsers will behave a little strange by not capturing the mouseout event the first time it's triggered.</p>	
<b>before</b>	HTML /stdWrap		
<b>beforeImg</b>	imgResource		
<b>beforeImgTagParams</b>	<img>-params		
<b>beforeImgLink</b>	boolean	If set, this image is linked with the same <A> tag as the text	
<b>beforeROImg</b>	imgResource	If set, ".beforeImg" and ".beforeROImg" is expected to create a rollOver-pair.	
<b>beforeWrap</b>	wrap	wrap around the ".before"-code	
<b>linkWrap</b>	wrap		
<b>stdWrap</b>	->stdWrap	stdWrap to the link-text!	
<b>ATagBeforeWrap</b>	boolean		
<b>ATagParams</b>	<A>-params /stdWrap	<p>Additional parameters</p> <p><b>Example:</b></p> <pre>class="board"</pre>	
<b>ATagTitle</b>	string /stdWrap	<p>Allows you to specify the "title" attribute of the &lt;a&gt; tag around the menu item.</p> <p><b>Example:</b></p> <pre>ATagTitle.field = abstract // description</pre> <p>This would use the abstract or description field for the &lt;a title=""&gt; attribute.</p>	

Property:	Data type:	Description:	Default:
<b>additionalParams</b>	string /stdWrap	Define parameters that are added to the end of the URL. This must be code ready to insert after the last parameter.  For details, see typolink->additionalParams	
<b>doNotLinkIt</b>	boolean /stdWrap	If set, the linktext are not linked at all!	
<b>doNotShowLink</b>	boolean /stdWrap	If set, the text will not be shown at all (smart with spacers)	
<b>stdWrap2</b>	wrap /stdWrap	stdWrap to the total link-text and ATag. (Notice that the plain default value passed to the stdWrap function is " ".)	
<b>RO</b>	boolean	If set, rolOver is enabled for this link	
<b>after...</b>	[mixed]	The series of "before..." properties are duplicated to "after..." properties as well. The only difference is that the output generated by the .after.... properties are placed after the link and not before.	
<b>altTarget</b>	target	Alternative target overriding the target property of the TMENU if set.	
<b>allStdWrap</b>	->stdWrap	stdWrap of the whole item	

[tsref:(cObject).HMENU.(mObj).TMENUITEM]

## IMGMENU

Imagemaps are made by creating one large GIFBUILDER-object based on the GIFBUILDER-object ".main" and adding the properties of the GIFBUILDER-objects for each item (NO, ACT, SPC... and so on).

Property:	Data type:	Description:	Default:
<b>target</b>	target	Target of the menu links	self
<b>forceTypeValue</b>	int	If set, the &type parameter of the link is forced to this value regardless of target.	
<b>noBlur</b>	Boolean	Normally graphical links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
<b>wrap</b>	wrap		
<b>params</b>	<img>-params		
<b>main</b>	->GIFBUILDER	Main configuration of the image-map! This defines the "underlay"!	
<b>dWorkArea</b>	offset + calc	Main offset of the GIFBUILDER-items (also called the "distribution")	

Property:	Data type:	Description:	Default:
<b>[Common Item States, see above]</b>	- >IMGMENUITEM M + .distrib	<p>This is the TMENUITEM-options for each category of menu item that can be generated.</p> <p><b>SPECIAL:</b> The -&gt;OptionSplit function is run on the whole GIFBUILDER-configuration before the items are generated.</p> <p><b>.distrib</b> is (x,y,v,h +calc) of the distribution of the menu items. This provides a way to space each item from the other. The codes "textX" and "textY" can be used for the width (X) and height (Y) dimension of each link. This works by adding a WORKAREA-GifBuilderObj between each of the IMGMENUITEM ("subset" of a GIFBUILDER-object) and this work area defines where the text should be printed. As such the "x,y" defines the offset <i>the next item will have</i> (this should be the width of the previous in many cases!) and "v,h" defines the <i>dimensions of the current item</i>. Consider this example taken from the static_template "template: MM": NO.distrib = textX+10, 0, textX+10, textY+5 In the future TypoScript may provide better ways to position GIFBUILDER-objects on the image-maps!</p> <p><b>ImgMap</b> is automatically used on the links! (that is the ".imgMap" property of the text-objects in the GIFBUILDER-objects is set automatically, unless is already set.)</p>	
<b>imgMapExtras</b>	<area...>-tags	Extra <area...>tags for the image-map	
<b>debugRenumberedObject</b>	boolean	if set, the final GIFBUILDER object configuration is output in order for you to debug your configuration	

[tsref:(cObject).HMENU.(mObj).IMGMENU]

# IMGMENUITEM

Property:	Data type:	Description:	Default:
1,2,3,4...	->GifBuilderObj	<p><b>NOTE:</b></p> <p>The way an imagemap is made is this; All IMGMENUITEMS are included in one big Gifbuilderobj (and renumbered!!). Because of this, Gifbuilderobjects on the next level will not be able to access the data of each menuitem.</p> <p>Also the feature of using [##.w] and [##.h] with +calc is currently not supported by IMGMENUITEMs.</p> <p>Therefore all IMAGE-objects on the first level is checked; if "file" or "mask" for any IMAGE-objects are set to "GIFBUILDER", the Gifbuilder-object is parsed to see if any TEXT-objects are present and if so, the TEXT-object is "checked" - which means, that the stdWrap-function is called at a time where the \$cObj-&gt;data-array is set to the actual menuitem.</p> <p>In the example below, the text of each menuitem is rendered by letting the title be rendered on a mask instead of directly on the image. Please observe that the "NO.10"-object is present in order for the image-map coordinates to be generated!!</p> <pre> NO.6 = IMAGE NO.6.file = masked_pencolor*.gif NO.6.mask = GIFBUILDER NO.6.mask {   XY = 500, 200   backColor = black   10 = TEXT   10 {     text.field = title     fontFile = fileadmin/fonts/caflisch.ttf     fontSize = 34     fontColor = white     angle = 15     offset = 48,110   }   20 = EFFECT   20.value = blur=80 } NO.10 = TEXT NO.10 {   text.field = title   fontFile = fileadmin/fonts/caflisch.ttf   fontSize = 34   angle = 15   offset = 48,110   hideButCreateMap = 1 } </pre>	

[tsref:(cObject).HMENU.(mObj).IMGMENUITEM]

# JSMENU

Property:	Data type:	Description:	Default:
<b>levels</b>	int, 1-5	How many levels there are	1
<b>menuName</b>	string	JavaScript menu name. If you have more than one JSMENU on the page, you should set this value for each one.	
<b>target</b>	target	Decides target of the menu-links	
<b>forceTypeValue</b>	int	If set, the &type parameter of the link is forced to this value regardless of target.	
1,2,3,4...	JSMENUITEM	levels-config	
<b>wrap</b>	wrap	wrap around the selector-boxes	



Property:	Data type:	Description:	Default:
<b>wrapAfterTags</b>	wrap	wrap around the selector-boxes with wrap and form-tags og JS-code.	
<b>firstLabelGeneral</b>	string	General first label. May be overridden by the one set in each JSMENUITEM	
<b>SPC</b>	boolean	If set, spacer can go into the menu, else not.	

[tsref:(cObject).HMENU.(mObj).JSMENU]

## JSMENUITEM

Property:	Data type:	Description:	Default:
<b>noLink</b>	boolean	Normally the selection of a menu item in the selector box will update the selector on the next level (if there is a next level) and if there are no items for that selector (because there were no subpages), then the link jumps to the page of itself. If this flag is set, however, no menu items in the selector box will ever link to anything. Only update the content of the next selector box on next level.	
<b>alwaysLink</b>	boolean	If set an item in the menu selector will always link. This takes precedence over "noLink".	
<b>showFirst</b>	boolean	if set, the first link will be shown when the menu is updated.	
<b>showActive</b>	boolean	if set, the active level will be selected, if present	
<b>wrap</b>	wrap	wraps the selector box	
<b>width</b>	int+	Initial width of the boxes set by a number of _ (underscores)	14
<b>elements</b>	int+	Initial number of elements in the menu. This is of course overruled by the actual menu item texts.	5
<b>additionalParams</b>	string	Additional parameters to the <select> box. Eg, you could set the width with a style-parameter like this: style="width: 200px;"	
<b>firstLabel</b>	string	Firt label in top of the menu (default is blank)	

[tsref:(cObject).HMENU.(mObj).JSMENUITEM]

### Example:

```
# The menu:
temp.jsmenu = HMENU
temp.jsmenu.1 = JSMENU
temp.jsmenu.1 {
    levels = 2
    1.wrap = |<br>
    2.wrap = |<hr>
}

# Insert on page.
page = PAGE
page.typeNum =0
page.5 = TEXT
page.5.field = title
page.10 < temp.jsmenu
```

This draws a menu with two selector boxes.

# Appendix A – media/scripts/ Plugins

## media/scripts/ in general

The directory typo3/sysexst/cms/tslib/media/scripts (in older versions just media/scripts) primarily contains php-scripts which are meant as 'external modules' as opposed to features included in the typo3/sysexst/cms/tslib/ libraries. Although they are distributed with TYPO3 just like the rest of tslib/ they form a basis for externally developed frontend functionality. So for most of these scripts, be inspired by them to write your own code. Notice the word 'most'; because some are written long time ago and do not represent the state-of-the-day to do it.

### About 'example templates'

For each plugin script there is one or more example templates. These templates are a part of the documentation of the features in the plugin because they describe the features of the markers and subparts and present an example to learn from. Therefore the example templates may be changed e.g. when new features come along.

You should therefore *not* rely on using the default templates unless you'll accept the fact that they may change in the future! So make a copy, modify it for your own purpose if needed and set up the TypoScript of the plugin to use your own template file!

## fe\_adminLib.inc

### Files:

File:	Description:
<b>fe_adminLib.inc</b>	Main class used to display the frontend administration forms. Call it from a USER_INT cObject with 'userFunc = user_feAdmin->init'. See the static_templates for examples. <b>Note:</b> Using the USER_INT cObject allows the script to work regardless of the page-cache which is necessary!!
<b>fe_admin_dmailsubscription.tmpl</b>	Example template file for subscription to newsletters of users to the tt_address table. This template is used by the static_template 'plugin.feadmin.dmailsubscription'.
<b>fe_admin_fe_users.tmpl</b>	Example template file for creating new frontend users (fe_users). This template is used by the static_template 'plugin.feadmin.fe_users'.

### Description

This class is used to create forms for database-administration in the frontend *independently of the backend (TBE)*. Thus you may want to use this, if you like frontend users to edit database content.

Authentication either goes through fe\_user login in which case you can stamp the records with the fe\_user\_uid so a record belongs to a certain fe\_user. The other authentication option is email authentication. In this case you have access to the record if your email is found in a certain field. By fe\_user authentication you can get a menu of items to edit when you're logged in. With email-authentication, you can request an email to be sent to your email address. This email contains a list of the available records.

It's all based on HTML-template files which you have to design by yourself, so there's some design work to do. On the other hand you get total freedom to design your forms.

### Example:

See static\_templates 'plugin.feadmin.\*' for various examples. Test them configured on the TYPO3 test site.

## Static template

plugin.feadmin.\*

### Incoming GET or POST vars:

Name:	Description:
<b>cmd</b>	Command.
<b>preview</b>	Preview flag.
<b>backURL</b>	Back URL.
<b>rU</b>	Record UID.
<b>aC</b>	Authentication Code.
<b>fD</b>	Fixed Data (array of fields)
<b>FE</b>	Frontend Edit data array, syntax, FE[ <i>tablename</i> ][ <i>field name</i> ] = value

### fe\_adminLib.inc properties

Property:	Data type:	Description:	Default:
<b>templateFile</b>	resource	The template file, see examples in typo3/sysex/cms/tslib/media/scripts/fe_user_admin.tmpl	
<b>templateContent</b>	string	Alternatively you can set this property directly to the value of the template.	
<b>table</b>	tablename	The table to edit. Notice: The ultimate list of fields allowed to be edited for the table is defined in TCA with the key ["feInterface"] ["fe_admin_fieldList"] for each table in question. For an example, see the table definition for fe_users which is a good example.	
<b>defaultCmd</b>	string	Defines which action should be default (if &cmd= is not set when calling the page)	
<b>clearCacheOfPages</b>	[list of integers]	This is a list of page-ids for which to clear the cache on any successful operation be it EDIT, CREATE or DELETE.	
<b>debug</b>	boolean	If set, debug information will be output from fe_adminLib which helps to track errors.	
<b>Actions:</b>			
<b>edit</b>	boolean /actionObject	<p>If set, editing is basically allowed. But you need to specify:</p> <p><b>.fields</b> (list of field names) which determines the fields allowed for editing. Every field in this list must be found as well in the ["feInterface"]["fe_admin_fieldList"] found in the TCA array which ultimately determines which fields can be edited by the fe_adminLib.</p> <p><b>.overrideValues.[field name]</b> (value string) defines values for specific fields which will override ANY input from the form. Overriding values happens after the outside values has been parsed by the .parseValues-property of fe_adminLib but before the evaluation by .required and .evalValues below. For example this may be useful if you wish to hide a record which is being edited, because you want to preview it first.</p> <p><b>.required</b> (list of field names, subset of .fields) which determines which fields are required to return a true value. The valid fields entered here will have the subpart ###SUB_REQUIRED_FIELD_[field name]### removed from the templates if they evaluates to being true and thereby OK. See below for information about this subpart.</p>	

Property:	Data type:	Description:	Default:
		<p><b>.evalValues.[field name]</b> (list of eval-codes) defines specific evaluation forms for the individual fiels of the form. See below.</p> <p><b>.preview</b> (boolean) will enable the form submitted to be previewed first. This requires a template for preview to be found in the template file. See below for subpart marker names.</p> <p><b>.menuLockPid</b> (boolean will force the menu of editable items to be locked to the .pid (edit only)</p> <p><b>.userFunc_afterSave</b> (function name) is called after the record is saved. The content passed is an array with the current (and previous) record in.</p>	
<b>create</b>	boolean /actionObject	<p>The same as .edit above except where otherwise stated. Plus there is these additional properties:</p> <p><b>.noSpecialLoginForm</b> (boolean) - if set, fe_adminLib does NOT look for the subpart marker TEMPLATE_CREATE_LOGIN but always for TEMPLATE_CREATE</p> <p><b>.defaultValues.[field name]</b> (value string); Like .overrideValues but this sets the default values the first time the form is displayed.</p>	
<b>delete</b>	boolean	Whether or not records may be deleted. Still regular authentication (ownership or email authCode) is required. Setting the var "preview" lets you make a delete-preview before actually deleting the record.	
<b>infomail</b>	boolean	Infomails are plaintext mails based on templates found in the template file. They may be used for such as sending a forgotten password to a user, but what goes into the infomail is totally up to your design of the template. Normally you may have only a default infomail (infomail.default) for instance for sending the password. But you can use other keys also. See below.	
<b>infomail.[key]</b>	(configuration of infomail properties)	<p>In order to make fe_adminLib send an infomail, you must specify these vars in your GET vars or HTML-form.</p> <p><b>fetch</b> - if integer, it searches for the uid being the value of 'fetch'. If not, it searches for the email-field (defined by a property of fe_adminLib, see below).</p> <p><b>key</b> - points to the infomail.[key] configuration to use</p> <p><b>Properties:</b></p> <p><b>.dontLockPid</b> (boolean) - selects only records from the .pid of fe_adminLib.</p> <p><b>.label</b> (string) - The suffix for the markers, see 'Email Markers' beneath.</p>	
<b>setfixed</b>	boolean /properties	<p>Allows set-fixed input, probably coming from a link in an infomail or notification mail.</p> <p><b>Syntax:</b></p> <p><b>[fixkey].[field name] = fieldvalue</b> - is used to setup a setfixed-link insertable in the infomail by the SYS_SETFIXED_*-markers. See above (setfixed-property of fe_adminLib).</p> <p>Special fixkey 'DELETE' is just a boolean.</p> <p><b>.userFunc_afterSave</b> (function name) is called after the record is saved. The content passed is an array with the</p>	

Property:	Data type:	Description:	Default:
		<p>current (and previous) record in.</p> <p><b>Concept:</b>  The 'setfixed' concept is best explained by describing a typical scenario - in fact the most common situation of its use:  Imagine you have some users submitting information on your website. But before that information enters the database, you would like to moderate it - simply preview it and then either delete it or approve it. In the 'create' configuration of fe_adminLib, you set up the hidden field of the record to be overridden to 1. Thus the record is hidden by default. Then you configure a setfixed-fixkey to set the hidden field to 0. This set up generates a list of parameters for use in an URL and those parameters are finally inserted by a corresponding marker in the email template. The link includes all necessary authentication to perform the change of values and thus a single click on that link is enough to change the field values. So this will - by a single click of a link in a notification mail sent to an admin - enable the record! Or of course a similar link with a cmd=delete link will delete it...</p> <p>There is a special "field name" you can use, which is '_FIELDLIST' and that lets you specify a list of fields in the record to base the auth-code on. If nothing is specified the md5-hash is based on the whole record which means that any changes will disable the setfixed link. If on the other hand, you set _FIELDLIST = uid,pid then that record will be editable as long as the uid and pid values are intact.</p> <p><b>Example:</b>  This is a common configuration of the email-properties with a simple setfixed setting:</p> <pre>email.from = kasper@typo3.com email.fromName = Kasper Skårhøj email.admin = kasper@typo3.com setfixed.approve {     hidden = 0     _FIELDLIST = uid,pid } setfixed.DELETE = 1 setfixed.DELETE._FIELDLIST = uid</pre> <p>Now, if you insert this marker in your email template</p> <pre>###SYS_SETFIXED_approve###</pre> <p>it will get substituted with something like these parameters:</p> <pre>&amp;cmd=setfixed&amp;rU=9&amp;fD[hidden]=0&amp;aC=5c403d90</pre> <p>Now, all you need is to point that to the correct url (where fe_adminLib is invoked!), e.g.:</p> <pre>###THIS_URL#####FORM_URL#####SYS_SETFIXED_approve###</pre> <p>and for deletion:</p> <pre>...###SYS_SETFIXED_DELETE###</pre>	
<b>Others</b>			
<b>authcodeFields</b>	<i>[list of fields]</i>	Comma separated list of fields to base the authCode generation on. Basically this list would include "uid" only in most cases. If the list includes more fields, you should be aware that the authCode will change when the value of that field changes. And then the user will have to re-send an email to himself with a new code.	

Property:	Data type:	Description:	Default:
		<p><b>.addKey</b> (string) adds the string to the md5-hash of the authCode. Just enter any random string here. Point is that people from outside doesn't know this code and therefore are not able to reconstruct the md5-hash solely based on the uid</p> <p><b>.addDate</b> (date-config) You can use this to make the code time-disabled. Say if you enter "d-m-Y" here as value, the code will work until midnight and then a new code will be valid.</p> <p><b>.codeLength</b> (int) Defines how long the authentication code should be. Default is 8 characters. In any case \$TYPO3_CONF_VARS['SYS']['encryptionKey'] is prepended.</p> <p><b>Advice:</b> If you want to generate authCodes compatible with the standard authCodes (used by the direct mailer by t3lib_div::stdAuthCode()), please set \$TYPO3_CONF_VARS['SYS']['encryptionKey'] to a unique and secret key (like you should in any case) and add "uid" as authcodeField ONLY. This is secure enough.</p>	
<b>email</b>		<p><b>.from</b> (string, email) Defines the sender email address of mails sent out</p> <p><b>.fromName</b> (string) Defines the name of the sender. If set, this will be used on the form NAME &lt;EMAIL&gt;</p> <p><b>.admin</b> Email address of the administrator which is notified of changes.</p> <p><b>.field</b> (string/integer) Defines the field name of the record where the email address to send to is found. If the field content happens to be an integer, this is assumed to be the uid of the fe_user owning the record and the email address of that user is fetched for the purpose instead.</p>	
<b>pid</b>	int+	The pid in which to store/get the records.	Current page
<b>fe_userOwnSelf</b>	boolean	If set, fe_users created by this module has their fe_cruser_id-field set to their own uid which means they 'own' their own record and can thus edit their own data. All other tables which has a fe_cruser_id field configured in the 'ctrl' section of their \$TCA-configuration will automatically get this field set to the current fe_user id.	
<b>fe_userEditSelf</b>	boolean	If set, fe_users - regardless of whether they own themselves or not - will be allowed to edit himself.	
<b>allowedGroups</b>	[list of integers]	List of fe_groups uid numbers which are allowed to edit the records through this form. Normally only the owner fe_user is allowed to do that.	
<b>evalFunc</b>	function name	Function by which you can manipulate the dataArray before it's saved. The dataArray is passed to the function as \$content and MUST be returned again from the function. The property "parentObj" is a hardcoded reference to the fe_adminLib object.	
<b>formurl</b>	->typolink	Contains typolink properties for the URL (action tag) of the form.	
<b>parseValues.[field]</b>	[list of parseCodes]	<p><b>ParseCodes:</b></p> <p><b>int</b> - returns the integer value of the input</p> <p><b>lower</b> - returns lowercase version of the input</p>	

Property:	Data type:	Description:	Default:
		<p><b>upper</b> - returns uppercase version of the input</p> <p><b>nospace</b> - strips all space</p> <p><b>alpha</b>, <b>num</b>, <b>alphanum</b>, <b>alphanum_x</b> - only alphabetic (a-z) and/or numeric chars. alphanum_x also allows _ and -</p> <p><b>trim</b> - trims whitespace in the ends of the string</p> <p><b>setEmptyIfAbsent</b> - will make sure the field is set to empty if the value is not submitted. This ensures a field to be updated an is handy with checkboxes</p> <p><b>random[x]</b> - Returns a random number between 0 and x</p> <p><b>files[semicolon-list(!) of extensions, none=all][maxsize in kb, none=no limit]</b> - Defining the field to hold files. See below for details!</p> <p><b>multiple</b> - Set this, if the input comes from a multiple-selector box (remember to add ...[] to the field name so the values come in an array!)</p> <p><b>checkArray</b> - Set this, if you want several checkboxes to set bits in a single field. In that case you must prepend every checkbox with [x] where x is the bitnumber to set starting with zero. The default values of the checkbox form elements must be false.</p> <p><b>uniqueHashInt[semicolon-list(!) of other fields]</b> - This makes a unique hash (32 bit integer) of the content in the specified fields. The values of those fields are first converted to lowercase and only alphanum chars are preserved.</p>	
<b>userFunc_updateArray</b>	function name	Points to a user function which will have the value-array passed to it before the value array is used to construct the update-JavaScript statements.	
<b>evalErrors.[field]. [evalCode]</b>		This lets you specify the error messages inserted in the <code>###EVAL_ERROR_FIELD_[field name]###</code> markers upon an evaluation error. See description of evaluation below.	
<b>cObjects.[marker_name]</b>	cObject	<p>This is cObjects you can insert by markers in the template.</p> <p><b>Example:</b> Say, you set up a cObject like this:</p> <pre>cObject.myHeader = TEXT cObject.myHeader.value = This is my header</pre> <p>then you can include this cObject in most of the templates through a marker named <code>###CE_myHeader###</code> or <code>###PCE_myHeader###</code> (see below for details on the difference).</p>	
<b>wrap1</b>	->stdWrap	<p>Global Wrap 1. This will be split into the markers <code>###GWIB###</code> and <code>###GWIE###</code>. Don't change the input value by the settings, only wrap it in something.</p> <p><b>Example:</b> <code>wrap1.wrap = &lt;b&gt;  &lt;/b&gt;</code></p>	
<b>wrap2</b>	->stdWrap	Global Wrap 2 (see above)	
<b>color1</b>	string /stdWrap	Value for <code>###GC1###</code> marker (Global color 1)	
<b>color2</b>	string /stdWrap	Value for <code>###GC2###</code> marker (Global color 2)	
<b>color3</b>	string /stdWrap	Value for <code>###GC3###</code> marker (Global color 3)	

[tsref:(script),fe\_adminLib]

## Main subparts

There is a certain system in the naming of the main subparts of the template file. The markers below are used when an action results in "saving". The *[action]* code may be DELETE, EDIT or CREATE depending on the cmd value.

Subpart marker:	Description:
###TEMPLATE_ <i>[action]</i> _SAVED###	Used for HTML output
###TEMPLATE_SETFIXED_OK### (general) ###TEMPLATE_SETFIXED_OK_ <i>[fixkey]</i> ###	Used for a successful setfixed-link.
###TEMPLATE_SETFIXED_FAILED###	Used for an unsuccessful setfixed-link. Notice that if you click a setfixed link twice, the second time it will fail. This is because the setfixed link is bound to the original record and if that changes in any way the authentication code will be invalid!
###EMAIL_TEMPLATE_ <i>[action]</i> _SAVED###	Used for an email message sent to the website user
###EMAIL_TEMPLATE_ <i>[action]</i> _SAVED-ADMIN###	Used for an email message sent to the admin
###EMAIL_TEMPLATE_SETFIXED_ <i>[fixkey]</i> ###	Used for notification messages in the event of successful setfixed operations.
###EMAIL_TEMPLATE_SETFIXED_ <i>[fixkey]</i> -ADMIN###	Ditto, for admin email

Likewise there is a system in the subpart markers used for the EDIT and CREATE actions to display the initial forms:

####TEMPLATE\_*[action]*### or if a fe\_user is logged in (only CREATE):  
####TEMPLATE\_*[action]*\_LOGIN###

... and if the &preview-flag is sent as well (including DELETE)

####TEMPLATE\_*[action]*\_PREVIEW###

Must-have subparts:

These are subparts that should exist in any template.

Subpart marker:	Description:
###TEMPLATE_AUTH###	Displayed if the authentication - either of fe_user or email authentication code - failed. You must design the error display to correctly reflect the problem!
###TEMPLATE_NO_PERMISSIONS###	This error message is displayed if you were authenticated but did not possess the right to edit or delete a record due to other reasons (like wrong fe_user/group ownership).

## 'infomail' Email subparts

All email subparts can be sent as HTML. This is done if the first and last word of the templates is <html> and </html> respectively. In addition the t3lib\_htmlmail class must be loaded.

Subpart:	Description:
###EMAIL_TEMPLATE_NORECORD###	
###EMAIL_TEMPLATE_ <i>[infomail_key]</i> ###	
###SUB_RECORD###	



## 'infomail' Email markers

Marker:	Description:
###SYS_AUTHCODE###	
###SYS_SETFIXED_ <i>[fixkey]</i> ###	

## FORM conventions

The forms used with fe\_adminLib should be named after the table they are supposed to edit. For instance if you are going to edit records in the table 'fe\_users' you must use a FORM-tag like this:

```
<FORM name="fe_users_form" method="POST" action="....">
```

The fields used to submit data for the records has this syntax: FE[*tablename*][*field name*]. This means, if you want to edit the 'city' field of a tt\_address record, you could use a form element like this:

```
<INPUT name="FE[tt_address][city]">
```

Submit buttons can be named as you like except using the name "doNotSave" of a submit button will prevent saving. If you need a Cancel button, please resort to JavaScript in an onClick even to change document.location.

## Common markers

Marker:	Description:
###GW1B### / ###GW1E###	Global wrap 1, begin and end (headers).
###GW2B### / ###GW2E###	Global wrap 2, begin and end (bodytext).
###GC1### / ###GC2### / ###GC3###	Global color 1 through 3.
###FORM_URL###	The url used in the forms: <code>index.php?id=page-id&amp;type=page-type</code>
###FORM_URL_ENC###	As above, but rawurlencoded.
###BACK_URL###	The backUrl value. Set to the value of incoming "backURL" var.
###BACK_URL_ENC###	As above, but rawurlencoded.
###REC_UID###	The UID of the record edited. Set to the value of incoming "rU" var.
###AUTH_CODE###	The "aC" incoming var.
###THE_PID###	The "thePid" value - where the records are stored.
###THIS_ID###	Set to the current page id.
###THIS_URL###	Set to the current script url as obtained by t3lib_div::getThisUrl().
###HIDDENFIELDS###	A bunch of hiddenfields which are required to be inserted in the forms. These by default include 'cmd', 'aC' and 'backURL'.

In addition you can in most cases use markers like this

```
###FIELD_[field name]###
```

where [field name] is the name of a field from the record. All fields in the record are used.

Finally you can insert cObjects defined in TypoScript with this series of markers (see .cObject property in table above):

```
###CE_[cObjectName]###
###PCE_[cObjectName]###
```

(###PCE\_\* is different from the ###CE\_\* cObjects by the fact they are rendered with a newly created cObj (as opposed to the parent cObj of fe\_adminLib) where the data-array is loaded with the value of ->dataArr which is the array submitted into the script. This is useful for presenting preview data. Finally both PCE\_ and CE\_ types cObject markers may be used with each single element in an edit menu (list of available records) by prefixing the marker with 'ITEM\_', e.g.  
###ITEM\_PCE\_[cObjectName]###

## Evaluation of the form fields

Printing error messages for REQUIRED fields

When a form template is displayed all subparts with the markers

```
###SUB_REQUIRED_FIELDS_WARNING###
```

and

```
###SUB_REQUIRED_FIELD_[field name]###
```

are removed. If there is a simple "required"-error (a field is not filled in) then the SUB\_REQUIRED\_FIELDS\_WARNING is not removed and thus the error message contained herein is shown.

Let's say that more specifically it's the 'email' field in a form which is not filled in. Then you can put in a subpart named

```
###SUB_REQUIRED_FIELD_email###
```

This is normally removed, but it'll *not* be removed if the email field fails and thus you are able to give a special warning for that specific field.

Printing other error messages

However you may use other forms of evaluation than simple "required" check. This is specified for "create" and "edit" modes by the properties ".evalValues.[field name] = [list of codes]". In order to tell your website user *which* of the possible evaluations went wrong, you can specify error messages by the property .evalErrors which will be inserted as the marker named ###EVAL\_ERROR\_FIELD\_[field name]###.

Lets say that you have put the code 'uniqueLocal' in the list of evaluation code for the email field. You would do that if you want to make sure that no email address is put into the database twice. Then you may specify that as:

```
create.evalValues {
    email = uniqueLocal, email
}
```

Then you set the evaluation error messages like this:

```
evalErrors.email {
    uniqueLocal = Apparently you're already registered with this email address!
    email = This is not a proper email address!
}
```

If the error happens to be that the email address already exists, the field ###EVAL\_ERROR\_FIELD\_email### will be substituted with the error message "Apparently you're already registered with this email address!".

## Passing default values to a form

You can pass default values to a form by the same syntax as you use in the forms. For instance this would set the name and email address by default:

```
...?FE[tt_address][name]=Mike%20Tyson&FE[tt_address]
```

[email]=mike@trex.us&doNotSave=1&noWarnings=1

Notice the blue value names are the field values (must be rawurlencoded. In javascript this function is called escape()) and the red values are necessary if you want to NOT save the record by this action and NOT to display error messages if some fields which are required is not passed any value.

## List of eval-codes

Eval-code:	Description:
<b>uniqueGlobal</b>	This requires the value of the field to be globally unique, which means it must not exist in the same field of any other record in the current table.
<b>uniqueLocal</b>	This is like uniqueGlobal, but the value is required to be unique <i>only</i> in the PID of the record. Thus if two records has different pid values, they may have the same value of this field.
<b>twice</b>	This requires the value of the field to match the value of a secondary field name [field name]_again sent in the incoming formdata. This is useful for entering password. Then if your password field is name "user_pass" then you simple add a second field name "user_pass_again" and then set the 'twice' eval code.
<b>email</b>	Requires the field value to be an email address at least on the form [name]@[domain].[tld]
<b>required</b>	Just simple required (trimmed value). 0 (zero) will evaluate to false!
<b>atLeast</b> <b>atMost</b>	Specifies a minimum / maximum of characters to enter in the fields. <b>Example</b> , that requires at least 5 characters: atleast [5]
<b>inBranch</b>	inBranch requires the value (typically of a pid-field) to be among a list of page-id's (pid's) specified with the inBranch parameters. The parameters are given like [root_pid; depth; beginAt] <b>Example</b> , which will return a list of pids one level deep from page 4 (included): inBranch [4;1]
<b>unsetEmpty</b>	This evaluation does not result in any error code. Only it simply unsets the field if the value of the field is empty. Thus it'll not override any current value if the field value is not set.

[tsref:(script).fe\_adminLib.evalErrors.(field).(evalCode)]

## Uploading files

fe\_adminLib is able to receive files in the forms. However there currently are heavy restrictions on how that is handled. Ideally the proces would be handled by the t3lib\_tcemain class used in the backend. In fact this could have been deployed but is not at this stage. The good thing about tcemain.php is that it perfectly handles the copying/deletion of files which goes into a certain field and even handles it independent of the storing method be it a list of filenames or use MM-relations to records (see tables.php section in 'Inside TYPO3').

This is how files are handled by fe\_adminLib and the restrictions that apply currently:

- You can upload files ONLY using "create" mode of a record. In any case you cannot edit currently attached files (this may be improved in the future). You can however use 'delete' mode.
- However you can use PREVIEW mode with 'create'. Works like this: if the mode is preview the temporary uploaded file is copied to a unique filename (prepended with the tablename) in typo3temp/ folder. Then the field value is set to the filenames in a list. When the user approves the content of the preview those temporary files are finally copied to the uploads/\* folder (or wherever specified in TCA). Limitations are that the temporary files in typo3temp/ are NOT deleted when copied to the real upload-folder (this may be improved) and certainly not if the user aborts (can't be improved because the user may go anywhere). If the user cancels the preview in order to change values, the files will need to be uploaded again (this may be improved).
- The TCA extensions allowed for the field is ignored! However you can specify a list of extensions of allowed for the files in the .parseValues property of fe\_adminLib
- The TCA filesize limitation for the field is ignored! However you can specify a max file size in kb in the .parseValues property of fe\_adminLib
- Works only on fields configured for comma-list representation of the filenames (non-MM, see

"Inside TYPO3" document on MM relations for files).

It's recommended to use a dedicated folder for files administered by the fe\_adminLib. The TYPO3 testsite does that by using the uploads/photomarathon/ folder for images. This makes it much easier to clean up the mess if files and their relations to the records are broken.

field names for files

Lets say you have a field named "picture" of a table name "user\_cars", the form-element should look like this:

```
<input type="file" name="FE[user_cars][picture][>
```

If you wish to upload multiple files to that field, the form-elements should look like:

```
<input type="file" name="FE[user_cars][picture][>
<input type="file" name="FE[user_cars][picture][>
<input type="file" name="FE[user_cars][picture][>
```

Use blob-types for the file-fields and reserve a minimum of 32 characters pr. filename.

**Note:** Make sure to always add the last square brackets ('...[]') to the field name! Otherwise it will not work!

## tipafriendLib.inc

Tip a Friend!

Tip a friend:

You're now sending this link to an email recipient:

[http://192.168.1.4/typo3/32/testsite-32b3/index.php?127&type=1&backPID=58&tt\\_news=1](http://192.168.1.4/typo3/32/testsite-32b3/index.php?127&type=1&backPID=58&tt_news=1)

Your Name: \*

Your Email: \*

Recipient email: \*

(Separate many recipients by comma)

Message:

HTML message:

☐

(You must fill in fields with \* correctly!)

Send

## Files:

File:	Description:
<b>tipafriendLib.inc</b>	Main class used to display the Tip-a-Friend form. Call it from a USER cObject with 'userFunc = user_tipafriend->main_tipafriend'
<b>tipafriend_template.tmpl</b>	Example template file.

## Example:

(See static\_template 'plugin.tipafriend' for a working configuration)

## Static template

plugin.tipafriend

## tipafriendLib.inc properties

Property:	Data type:	Description:	Default:
<b>templateFile</b>	resource	The template-file. See example in 'media/scripts/tipafriend_template.tmpl'	
<b>code</b>	string /stdWrap	Code to define, what the script does. Case sensitive.	
<b>defaultCode</b>	string	The default code (see above) if the value is empty. By default it's not set and a help screen will appear	
<b>wrap1</b>	->stdWrap	Global Wrap 1. This will be split into the markers <b>###GWIB###</b> and <b>###GWIE###</b> . Don't change the input value by the settings, only wrap it in something.  <b>Example:</b> wrap1.wrap = <b>   </b>	
<b>wrap2</b>	->stdWrap	Global Wrap 2 (see above)	
<b>color1</b>	string /stdWrap	Value for <b>###GC1###</b> marker (Global color 1)	
<b>color2</b>	string /stdWrap	Value for <b>###GC2###</b> marker (Global color 2)	
<b>color3</b>	string /stdWrap	Value for <b>###GC3###</b> marker (Global color 3)	
<b>typolink</b>	->typolink	TypoLink configuration for the TIPLINK to the TIPFORM page. .additionalParams is added the parameter "&tipUrl="	
<b>htmlmail</b>	boolean	If set, the page is fetched as HTML and send in HTML (a plain text version is sent as well).	

[tsref:(script),tipafriend]

## plaintextLib.inc

## Files:

File:	Description:
<b>plaintextLib.inc</b>	Main class used to display plain text content. Call it from a USER cObject with 'userFunc = user_plaintext->main_plaintext'
<b>plaintext_content.tmpl</b>	Example template file.

## Example:

(See static\_template 'plugin.alt.plaintext' for a working configuration)

## Static template

plugin.alt.plaintext

### plaintextLib.inc properties

Property:	Data type:	Description:	Default:
<b>siteUrl</b>	url	Url of the site.	
<b>defaultOutput</b>	untrimmed string	Default output if CType is not rendered.	
<b>uploads.header</b>	untrimmed string	Header for uploads.	
<b>images.header</b>	untrimmed string	Header for images.	
<b>images.captionHeader</b>	untrimmed string	Header for image captions.	
<b>images.linkPrefix</b>	untrimmed string	Prefix for image-links.	
<b>.header</b>			
<b>defaultType</b>	int	Defines which type to use as default.	
<b>date</b>	date-config	For header date.	
<b>datePrefix</b>	untrimmed string	Prefix for header date.	
<b>linkPrefix</b>	untrimmed string	Prefix for header links.	
<b>[1-5].preLineLen</b>	int	Length of line before header.	
<b>[1-5].postLineLen</b>	int	Length of line after header.	
<b>[1-5].preBlanks</b>	int	Number of blank lines before header.	
<b>[1-5].postBlanks</b>	int	Number of blank lines after header.	
<b>[1-5].stdWrap</b>	->stdWrap	for header text.	
<b>[1-5].preLineChar</b>	string	Character to pre-line.	
<b>[1-5].postLineChar</b>	string	Character to post-line.	
<b>[1-5].preLineBlanks</b>	int	Number of blank lines between header and pre-line.	
<b>[1-5].postLineBlanks</b>	int	Number of blank lines between header and post-line.	
<b>[1-5].autonumber</b>	boolean	If set, a number is prepended every header. The number corresponds to the content element number in the select.	
<b>[1-5].prefix</b>	untrimmed string	Header string prefix.	
<b>bulletlist.[0-3].bullet</b>	untrimmed string	Bullet for bullet list, layout [0-3].	
<b>bulletlist.[0-3].secondRow</b>	untrimmed string	If set, this is used for lines on the second row of bullet-lists.	
<b>menu</b>	cObject	cObject to render menu. The output is stripped for tags and the links is extracted. Further all   chars are converted to chr(10).	
<b>shortcut</b>	cObject	cObject to render other elements. See config below which simply uses this object to render more tt_content elements as plaintext.	
<b>bodytext.stdWrap</b>	->stdWrap	stdWrap for body-text. See config example below.	
<b>userProc</b>	function name	Lets you process the output of each content element before it finally is returned. Property "parentObj" of the conf-array holds a references to the plainText object calling the function.	

[tsref:(script).plaintextLib]

Datatype 'untrimmed string' means that you can enter a string as usual, but if you enter a value between two vertical lines, that value will be used and NOT trimmed. Normally values are trimmed.

#### Example:

```
lib.renderObj = USER
```

```

lib.renderObj.userFunc = user_plaintext->main_plaintext
lib.renderObj {
    header.defaultType = 1
    header.date = D-m-Y
    header.datePrefix = |Date: |
    header.linkPrefix = | - Headerlink: |
    header.1.preLineLen = 76
    header.1.postLineLen = 76
    header.1.preBlanks = 1
    header.1.stdWrap.case = upper

    header.2 < .header.1
    header.2.preLineChar = *
    header.2.postLineChar = *

    header.3.preBlanks = 2
    header.3.postBlanks = 1
    header.3.stdWrap.case = upper

    header.4 < .header.1
    header.4.preLineChar= =
    header.4.postLineChar= =
    header.4.preLineBlanks= 1
    header.4.postLineBlanks = 1

    header.5.preBlanks = 1
    header.5.autonumber = 1
    header.5.prefix = |: >> |

    siteUrl = {$plugin.alt.plaintext.siteUrl}
    defaultOutput (
|
[Unrendered Content Element; ###CType### ]
|
)

uploads.header = |DOWNLOADS:|

images.header = |IMAGES:|
images.linkPrefix = | - Imagelink: |
images.captionHeader = |CAPTION:|

bulletlist.0.bullet = |* |
bulletlist.1.bullet = |# |
bulletlist.2.bullet = | - |
bulletlist.3.bullet = |> |
bulletlist.3.secondRow = |. |
bulletlist.3.blanks = 1

menu = <tt_content.menu.20
shortcut = <tt_content.shortcut.20
shortcut.0.conf.tt_content = <lib.renderObj
shortcut.0.tables = tt_content

bodytext.stdWrap.parseFunc.tags {
    link < styles.content.parseFunc.tags.link
    typolist = USER
    typolist.userFunc = user_plaintext->typolist
    typolist.siteUrl = {$plugin.alt.plaintext.siteUrl}
    typolist.bulletlist < temp.renderObj.bulletlist
    typohead = USER
    typohead.userFunc = user_plaintext->typohead
    typohead.siteUrl = {$plugin.alt.plaintext.siteUrl}
    typohead.header < temp.renderObj.header
    typocode = USER
    typocode.userFunc = user_plaintext->typocode
    typocode.siteUrl = {$plugin.alt.plaintext.siteUrl}
}
}

```

# Appendix B – Standard Templates

## static\_template

This section of the TypoScript reference is used to introduce the standard templates that come with TYPO3 in the static table "static\_template".

In newer versions of TYPO3 the static templates are an own system extension. Old records in the database table static\_template are NOT changed from version to version! Still changes may appear!

## Media

The standard templates use some standard media-files, like gif-images and fonts. These are situated in the folder "typo3/sysexst/cms/tslib/media/" (in older versions in "media/") relative to the root of the TYPO3-website.



# Appendix C – PHP include scripts

## Introduction

Although you can do very much with TypoScript itself, it can sometimes be a much more flexible solution to include a PHP-script you write on your own. But you must understand and respect some circumstances. For example the caching system: When a page is shown with TYPO3 it's normally cached afterwards in the SQL-database. This is done to ensure a high performance when delivering the same page the next time. But this also means that you can only make custom code from your include files if you differ your output based on the same conditions that the template may include! For example you cannot just return browser-specific code to TypoScript if not the template also distinguishes between the actual browsers. If you do, the cache will cache the page with the browser-specific HTML-code and the next hit by another browser will trigger the cache to return a wrong page. If the condition is correctly setup "another browser"-hit will instead render another page (which will also be cached but tagged with the other browser!) and the two browsers will receive different pages but still the pages will be cached.

## TypoScript Configuration

The following objects are related to the direct inclusion of PHP code inside templates.

### PHP\_SCRIPT

This includes a PHP-script. You should not name the script ".php" but rather ".inc" as it's meant to be included and not executed on it's own.

NOTE: This option is ignored if \$TYPO3\_CONF\_VARS['FE']['noPHPscriptInclude']=1; is set in localconf.php.

Property:	Data type:	Description:	Default:
<b>file</b>	resource /stdWrap	File that will be included. This file must be valid PHP-code! It's included with "include()";  <b>Directions:</b> <b>1) All content must be put into \$content.</b> No output must be echo'ed out!  2) Call \$GLOBALS['TSFE']->set_no_cache(), if you want to disable caching of the page. Set this during development! And set it, if the content you create may not be cached.  <b>NOTE:</b> If you have a parsing error in your include script the \$GLOBALS['TSFE']->set_no_cache() function is NOT executed and thereby does not disable caching. Upon a parse-error you must manually clear the page-cache after you have corrected your error! 3) the array \$conf contains the configuration for the PHP_SCRIPT cObject. Try debug(\$conf) to see the content printed out for debugging! <i>See the appendix later in this manual for an introduction to writing your own PHP include-scripts.</i>	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).PHP\_SCRIPT]

### PHP\_SCRIPT\_INT

(see PHP\_SCRIPT)

Property:	Data type:	Description:	Default:
<b>file</b>	resource /stdWrap	File that will be included. This file must be valid PHP-code! It's included with "include()";	

Property:	Data type:	Description:	Default:
		<p><b>Purpose:</b> This basically works like PHP_SCRIPT. But the vital difference is that inserting a PHP_SCRIPT_INT (internal opposed to external, see below) merely inserts a divider-string in the code and then serializes the current cObj and puts it in the \$GLOBALS['TSFE']-&gt;config['INTincScript']-array. This array is saved with the cached page-content. Now, the point is, that including a script like this lets you avoid disabling pagecaching. The reason is that the cached page contains the divider string and when a "static" page is fetched from cache, it's divided by that string and the dynamic content object is inserted. This is the compromise option of all three PHP_SCRIPT-cObjects, because the page-data is all cached, but still the pagegen.php script is included, which initializes all the classes, objects and so. What you gain here is an environment for your script almost exactly the same as PHP_SCRIPT because your script is called from inside a class tslib_cObj object. You can work with all functions of the tslib_cObj-class. But still all the "static" page content is only generated once, cached and only your script is dynamically rendered.</p> <p><b>Rules:</b> - calls to \$GLOBALS['TSFE']-&gt;set_no_cache() and \$GLOBALS['TSFE']-&gt;set_cache_timeout_default() makes no sense in this situation. - parsing errors do not interfere with caching - Be aware that certain global variables may not be set as usual and be available as usual when working in this mode. Most scripts should work out-of-the-box with this option though. - Dependence and use of LOAD_REGISTER is fragile because the PHP_SCRIPT_INT is not rendered until <i>after</i> the cached content and due to this changed order of events, use of LOAD_REGISTER may not work. - You can not nest PHP_SCRIPT_INT and PHP_SCRIPT_EXT in PHP_SCRIPT_INT. You may nest PHP_SCRIPT cObjects though.</p>	
<b>includeLibs</b>	<i>list of resource</i>	<p>This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.</p>	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).PHP\_SCRIPT\_INT]

## PHP\_SCRIPT\_EXT

(see PHP\_SCRIPT)

Property:	Data type:	Description:	Default:
<b>file</b>	resource /stdWrap	<p>File that will be included. This file must be valid PHP-code! It's included with "include()";</p> <p><b>Purpose:</b> This works like PHP_SCRIPT_INT, because a divider string is also inserted in the content for this kind of include-script. But the difference is that the content is divided as the very last thing before it's output to the browser. This basically means that PHP_SCRIPT_EXT (external, because it's included in the global space in index_ts.php file!!) can output data directly with echo-statements! This is a very "raw" version of PHP_SCRIPT because it's not included from inside an object and you have only very few standard functions from TYPO3 to call. This is the fastest option of all three PHP_SCRIPT-cObjects, because the page-data is all cached and your dynamic content is generated by a raw php-script.</p> <p><b>Rules:</b> - All content can be either 1) echo'ed out directly, or 2) returned in \$content. - calls to \$GLOBALS['TSFE']-&gt;set_no_cache() and \$GLOBALS['TSFE']-&gt;set_cache_timeout_default() makes no sense in this situation. - parsing errors do not interfere with caching - In the global name-space, the array \$REC contains the current record when the file was "inserted" on the page, and \$CONF-array contains the configuration for the script. - Don't mess with the global vars named \$EXTiS_*</p>	
<b>includeLibs</b>	<i>list of resource</i>	This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	
<b>stdWrap</b>	->stdWrap		

[tsref:(cObject).PHP\_SCRIPT\_EXT]

## Including your script

Your script is included by a function, PHP\_SCRIPT, inside the class "tslib\_cObj" in the "tslib\_content.php" script. Thereby your file is a part of this object (tslib\_cObj) and function. This is why you must return all content in the variable "\$content" and any TypoScript-configuration is available from the array "\$conf" (it may not be set at all though so check it with is\_array(!))

### \$conf

The array \$conf contains the configuration for the PHP\_SCRIPT cObject. Try debug(\$conf) to see the content printed out for debugging!

### \$content

Return all content in this variable.

Remember, don't output anything (but debug code) in your script!

### White spaces

Because nothing is sent off to the browser before everything is rendered and returned to index\_ts.php which originally set of the rendering process, you must ensure that there's no whitespace before and after your <?...?> tags in your include- or library-scripts!

## **\$GLOBALS['TSFE']->set\_no\_cache()**

Call the function `$GLOBALS['TSFE']->set_no_cache()`, if you want to disable caching of the page. Call this during development! And call it, if the content you create may not be cached.

**Note:** If you make a syntax error in your script that keeps PHP from executing it, then the `$GLOBALS['TSFE']->set_no_cache()` function is not executed and the page *is* cached! So in these situations, correct the error, clear the page-cache and try again. This is true only for `PHP_SCRIPT` and not for `PHP_SCRIPT_INT` and `PHP_SCRIPT_EXT` which are rendered *after* the cached page!

**Example:**

```
$GLOBALS['TSFE']->set_no_cache();
```

## **\$this->cObjGetSingle( value , properties )**

Gets a content-object from the `$conf`-array. (See the section below named "Case story" on how to use this!)

**Example:**

```
$content = $this->cObjGetSingle($conf['image'], $conf['image.']);
```

This would return any `IMAGE-cObject` at the property "image" of the `conf`-array for the include-script!

## **\$this->stdWrap( value, properties )**

`stdWrap`'s the content "value" due to the configuration of the array "properties".

**Example:**

```
$content = $this->stdWrap($content, $conf['stdWrap.']);
```

This will `stdWrap` the content with the properties of ".stdWrap" of the `$conf`-array!

## **Internal Vars in the main frontend object, TSFE (TypoScript Front End)**

There are some variables in the global object, `TSFE`, you might need to know about. These ARE ALL READ-ONLY!! (Read: Don't change them!) See the class `tslib_fe` for the full descriptions.

If you for instance want to access the variable "id", you can do so by writing: `$GLOBALS['TSFE']->id`

Var:	PHP-Type:	Description:	Default:
<b>id</b>	int	The page id	
<b>type</b>	int	The type	
<b>page</b>	array	The pagerecord	
<b>fe_user</b>	object	The current front-end user. Userrecord in <code>\$GLOBALS['TSFE']-&gt;fe_user-&gt;user</code> , if any login.	
<b>loginUser</b>	boolean	Flag indicating that a front-end user is logged in.	0
<b>rootLine</b>	array	The rootLine (all the way to tree root, not only the current site!). Current site root line is in <code>\$GLOBALS['TSFE']-&gt;tmpl-&gt;rootLine</code>	
<b>sys_page</b>	object	The object with pagefunctions (object) See <code>t3lib/page.php</code>	
<b>gr_list</b>	string (list)	The group list, sorted numerically. Group -1 = no login	
<b>beUserLogin</b>	boolean	Flag that indicates if a Backend user is logged in!	0

## Global vars

Var:	PHP-Type:	Description:	Default:
<b>BE_USER</b>	object	The back-end user object (if any).	not set
<b>TYPO3_CONF_VARS</b>	array	TYPO3 Configuration.	
<b>TSFE</b>	object	Main frontend object.	

## Case story

This is a case story of how to use include-scripts.

In this situation we would like to use some libraries of our very own, not part of TYPO3. Therefore we use the feature of including a library at the very beginning of the page-parsing.

First we put this TypoScript line in the "Setup"-field of the template:

```
config.includeLibrary = fileadmin/scripts/include.inc
```

The file **include.inc** is now included (in typo3/sysex/cms/tslib/pagegen.php). In this case it looks like this:

file: fileadmin/scripts/include.inc

```
<?
...
include('fileadmin/scripts/hello_world.inc');
include('fileadmin/scripts/other_library.inc');
...
?>
```

As you can see, this file includes our library "hello\_world" and some other libraries too!

The file **hello\_world.inc** looks like this:

file: fileadmin/scripts/hello\_world.inc

```
<?
class hello_world {
    function theMessage() {
        return "Hello World";
    }
}
?>
```

So far nothing has happened, except our libraries are included, ready for use.

Now we need to use the outcome of the class hello\_world somewhere on a page. So in the TypoScript code we setup a content-object that includes the third script:

```
page.100 = PHP_SCRIPT
page.100.file = fileadmin/scripts/surprise.inc
```

**surprise.inc** looks like this:

file: fileadmin/scripts/surprise.inc

```
<?
$hello_world_object = new hello_world; // New instance is created
$contentBefore = $this->cObjGetSingle($conf['cObj'], $conf['cObj.']);
$content = $contentBefore . $hello_world_object->theMessage();
$content = $this->stdWrap($content, $conf['stdWrap.']);
?>
```

Line 1: The PHP-object \$hello\_world\_object is created.

Line 2: This fetches the content of a cObject, "cObj", we defined

Line 3: The result of line 2 is concatenated with the result of the "theMessage"-function of the \$hello\_world\_object

Line 4: Finally the content is stdWrap'ed with the properties of ".stdWrap" of the \$conf-array.

The output:

With this configuration -

```
page.100 = PHP_SCRIPT
page.100.file = fileadmin/scripts/surprise.inc
```

- the output will look like this:

Hello World

With this configuration -

```
page.100 = PHP_SCRIPT
page.100 {
    file = fileadmin/scripts/surprise.inc
    cObj = TEXT
    cObj.value = Joe says:&nbsp;
}
```

- the output will look like this:

Joe says: Hello World

With this configuration -

```
page.100 = PHP_SCRIPT
page.100 {
    file = fileadmin/scripts/surprise.inc
    cObj = TEXT
    cObj.value = Joe says:&nbsp;
    stdWrap.wrap = <font color="red"> | </font>
    stdWrap.case = upper
}
```

- the output will look like this:

JOE SAYS: HELLO WORLD

End of lesson.

## Storing user-data or session-data

Doing so is quite simple with TYPO3.

Userdata is data, that follows login users. As soon as a user, who is logged in, logs out, this data is no more accessible and cannot be altered.

Session data is data, that follows the user currently browsing the site. This user may be a logged in user, but his session-data is bound to the "browsing-session" and not to the user-id of his. This means, that the very same person will carry this data still, even if he logs out. As soon as he closes his browser, his data will be gone though.

Also you should know, that session-data has a default expire-time of 24 hours.

Retrieving and storing user-/session-data is done by these functions:

### **\$GLOBALS['TSFE']->fe\_user->getKey(type, key)**

"type" is either "user" or "ses", which defines the data-space, user-data or session-data

"key" is the "name" under which your data is stored. This may be arrays or normal scalars.

Note that the key "recs" is reserved for the built-in "shopping-basket". As is "sys" (for TYPO3 standard modules and code)

**Example:**

```
if ($GLOBALS['TSFE']->loginUser) {
    $myData = $GLOBALS['TSFE']->fe_user->getKey('user', 'myData');
} else {
    $myData = $GLOBALS['TSFE']->fe_user->getKey('ses', 'myData');
}
```

This gets the stored data with the key "myData" from the user-data, but if no user is logged in, it's fetched from the session data instead.

### **\$GLOBALS['TSFE']->fe\_user->setKey(type, key, data)**

"type" is either "user" or "ses", which defines the data-space, user-data or session-data

"key" is the "name" under which your data is stored.

Note that the key "recs" is reserved for the built-in "shopping-basket". As is "sys" (for TYPO3 standard modules and code)

"data" is the variable, you want to store. This may be arrays or normal scalars.

**Example:**

```
$myConfig['name'] = 'paul';
$myConfig['address'] = 'Main street';
$GLOBALS['TSFE']->fe_user->setKey('ses', 'myData', $myConfig);
```

This stores the array \$myConfig under the key "myData" in the session-data. This lasts as long as "paul" is surfing the site!

## Using the built in "shopping basket"

TYPO3 features a shopping basket for the session-data.

When you submit data from forms (or by querystring) (post/get-method) in the array "recs" it's stored in the session-data under the key recs.

The syntax is like this:

```
recs[table_name][uid_of_record]
```

**Example:**

This form-element will change the registered value of record with uid=345 from the "tt\_products" table in typo3. Please note, that the record itself is NOT in any way modified, only the "counter" in the session-data indicating the "number of items" from the table is modified.

```
<input name="recs[tt_products][345]">
```

Note on checkboxes:

When you are creating forms with checkboxes, the value of the checkbox is sent by MSIE/Netscape

ONLY if the checkbox is checked! If you want a value sent in case of a disabled checkbox, include a hidden formfield of the same name just before the checkbox!

**Example:**

```
<INPUT type="hidden" name="recs[tt_content][345]" value="0">
<INPUT type="checkbox" name="recs[tt_content][345]" value="1">
```

## Clearing the "basket"

This will clear the basket:

```
<INPUT type="hidden" name="recs[clear_all]" value="1">
```



# Appendix D – index.php

## Introduction

index.php is the main script for showing pages with TYPO3 / TypoScript. This page provides some information about this script and how to use it.

Normally you request pages by setting a value for "id" and possibly for "type".

"id" refers to a page. This is an integer. If a string is supplied, it's regarded as an alias and the corresponding page is found.

"type" defines which "type" the page is of. It is always an integer (0-255). If "type" is not set, it's regarded to be zero. "type" is used to build framesets. For example the frameset could have "type=0" (or nothing) and the pages in the various frames could have "type=1" and "type=2" and "type=3". In TypoScript you define a PAGE-object for each type so TYPO3 renders different pages depending on the type-value. Normally the PAGE-object displaying the page content is named "page" and has the "type=1" value.

## Submitting data to index.php

You can submit data to index.php for several reasons. These are the standard features included in the script:

### Login/Logout:

Detected by class "t3lib\_userauth" looking for the var "logintype". If this is set, authentication is done.

Input may be of both GET and POST method.

Login:

logintype = "login"

pass = the password

user = the username

pid = the id of the page where the user-archive is found. You don't need this value if \$TYPO3\_CONF\_VARS['FE']['checkFeUserPid'] is set.

(redirect = Not used)

Logout:

logintype = "logout"

See the cObject FORMS for an in-depth description

## Search

Detected by the cObject SEARCHRESULT, which proceeds with a search if "sword" && "scols" are set. The search MUST submit to a page with such a content-object on it!

Input may be of both GET and POST method.

Search:

sword = the searchwords

`stype` = the search type  
`scols` = the tables/columns to search  
`locationData` = Reference to the record carrying the form. Used to look up the original startingpoint of the search (ONLY POST-method)  
`(redirect` = Not used)  
  
`scount` = Used by the searchresult to indicate the number of results  
`spointer` = Used by the searchresult to indicate the startingpoint for the next number of results.

See the cObject SEARCHRESULT for a complete description.

## Emailforms

Detected by the mainscript "index.php" looking for the var "formtype\_mail" to be set (could be the submit-button).

Input MUST be POST method. And the REFERER and HTTP\_HOST must match. Also the locationData var must be sent and at least point to the uid of a readable page.

## Database-submit

Detected by the mainscript "index.php" looking for the var "formtype\_db" to be set. (could be the submit-button)

Input MUST be POST method. And the REFERER and HTTP\_HOST must match. To setup a script to handle the input, refer to the FE\_DATA object.

See examples from the typo3/sysex/cms/tslib/media/scripts/ folder, e.g. "guest\_submit.inc".