

The page header field is missing.

A fresh look at the backend will show that the page header field has been hidden. TCEFORM comes up when deciding which pages a user may add. If, for example, you want to prevent users from creating

a search form, use the following syntax:

```
TYPOSCRIPT
TCEFORM.tt_content.CType.removeItems = search
```

Listing 6

You will see on the 'add page content' page that the search form option is no longer shown. The following table lists all the entries that relate to this point:

Element	TS Key
Image	image
HTML	html
Separator	div
Header	header
Text	text
Text with image	textpic
Links to files	uploads
Multimedia	multimedia
Add data record	shortcut
Add plugin	list
Script	script
Menu/site map	menu
Table	table
Enumeration	bullets
Form	mailform
Search	search
Login	login
Text box	splash

Configuring system tables with TCEMAIN

With TCEMAIN you can set options for the system tables. This includes the possibility of presetting the rights allocation for a page-tree in such a way that it is dependent on the user settings for the addition of new pages. The following syntax shows a typical example of this:

```
TYPOSCRIPT
TCEMAIN.permissions.groupid = 5
TCEMAIN.permissions.user = show, editcontent, new, edit, delete
TCEMAIN.permissions.group = show, editcontent, new, edit, delete
TCEMAIN.permissions.everybody = show, edit, delete
```

Listing 7

Newly added pages are hereby automatically allocated to the user group with the ID 5. The values available are 'show' (view in the backend), 'editcontent' (edit contents), 'new' (add new pages), 'edit' (edit page header) and 'delete' (delete). In the present example, the members of the user group with the ID 5 obtain the following rights:

- View pages
- Delete pages
- Add new pages
- Edit page contents

Other users may only:

- View pages

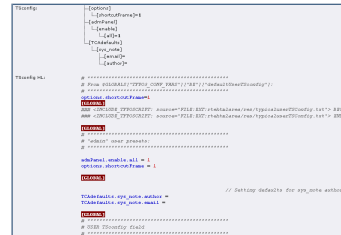
- Edit pages

A further example of the use of TCEMAIN concerns a typical (often annoying) TYPO3 phenomenon: when you copy a page, 'copy' is automatically added to it. Thus About Us becomes About Us ('copy'). This can easily be eliminated with:

```
TYPOSCRIPT
TCEMAIN.default.disablePrependAtCopy = 0
```

Listing 8

User TSConfig



An overview of the user configuration.

User TSConfig may be defined for individual users as well as for user groups. The configurations for individual users are based on those of the groups to which they belong. The configuration for an individual user can be overwritten with relevant entries into his or her user profile. If you wish to view the configurations of particular

users, open the module 'tools/user' and click on the user names you are after.

The domains 'setup', 'admPanel' and 'options' are listed within the user TSConfig tree (what these domains refer to will be explained below). Beneath the tree you can see the particular TypoScript instructions, which are augmented with comments.

Setup

All the properties found in the module 'user/settings' can be adapted via the properties of 'setup'. These include, for example, the maximum title length, help functions, and whether the RTE should be shown.

It is possible to set the default properties (setup – default) that are to apply to every new user. When the user exercises the option to restore the standard configuration, the values set here will be loaded automatically.

It is possible to overwrite the presettings with the parameter 'override'. The settings defined by means of 'setup.override' cannot be removed by users simply deleting the corresponding entry. Instead, the value must be overwritten afresh, or else emptied by entering an empty string.

If, for example, you want to deny a user the viewing of the RTE, customize the TSConfig field as follows:

```
TYPOSCRIPT
setup.default.edit.RTE = 0
```

Listing 9

To customize the admin panel, use the TLO 'admPanel'. You can hide individual modules of the admin panel via 'admPanel.enable'. If, for example, you wish to prevent the info module from appearing in the frontend, use the following syntax:

```
TYPOSCRIPT
admPanel.enable.info = 0
```

Listing 10

Of course, you can hide not only the info module, but also: all, preview, cache, publish, edit and tsdebug.

Incidentally, for administrators, the standard setting for all modules is '1'. Note that there is no point in making these settings unless the admin panel is actually supposed to appear in the frontend. For this purpose, you have to set the admin panel to be visible in the template setup with 'config.admPanel = 1'.

options

Global settings for the backend are set via the TLO 'options'. In doing so, you can (for example) show or hide RTE buttons for users, or define the time span for which the click menu appears.

Here is an example for 'options': the following syntax allows an editor to create folders in the element browser:

```

TYPOSCRIPT
options.createFoldersInEB = 1

```

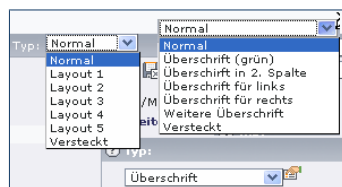
Listing 11

Practical solutions for backend customization

The following pages present several typical applications which you will repeatedly encounter during your daily work.

Header types in the customize header domain

The option fields in the backend can be customized without any problems. Here the content of a type field for headers will serve as an example. The following illustration shows the affected field in two different versions.



The headers in the right window are easier to allocate.

The left field is shown in the standard form. But what do 'Layout 1', 'Layout 2' and so on actually mean? Things become clearer in the right menu, which contains 'speaking' descriptions which you can understand immediately and which you will be able to understand

even a few weeks later.

To customize the menu, the TSConfig field of the corresponding page is customized as follows:

```

TYPOSCRIPT
TCEFORM.tt_content.header_layout.altLabels.1 = Überschrift (grün)
TCEFORM.tt_content.header_layout.altLabels.2 = Überschrift in 2. Spalte
TCEFORM.tt_content.header_layout.altLabels.3 = Überschrift für links
TCEFORM.tt_content.header_layout.altLabels.4 = Überschrift für rechts
TCEFORM.tt_content.header_layout.altLabels.5 = Weitere Überschrift

```

Listing 12

Assign appropriate headers so that, later on, they can still be easily allocated.

Removing obsolete page-types

You can explicitly determine which entries are to appear as options in the type field.

In the left menu, all page types are available. In the field next to it, however, some page types have been hidden. This means, for example, that the editor cannot add a page of the type 'not in menu'.

This type restriction has been achieved with the following TypoScript in the TSConfig field:

```

TYPOSCRIPT
TCEFORM.pages.doktype {
    removeltms=5,6,7,199,254
}

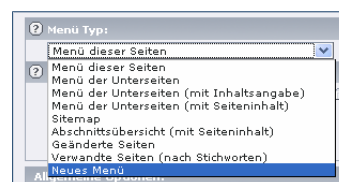
```

Listing 13

The following table shows how fields in the list are addressed:

Addressed via	Field
1	Standard
2	Expanded
3	External URL
4	Shortcut
5	Not in menu
6	Backend user domain
7	Mount page
199	Space
254	Sys Folder
255	Recycler
--div--	Border

Adding a new menu-entry to the backend



A new menu has been created.

When adding new page content, you can click on the type 'menu/sitemap' and choose the menu you are after from the menu type field. If the menus provided are insufficient, you can add your own. In this example, 'new menu' has

been selected from the menu list and implemented with the following line of TypoScript:

```

TYPOSCRIPT
TCEFORM.tt_content.menu_type.addItem.2 = Neues Menü

```

Listing 14

The following table lists the possible menu types. The values matching the numbers are items to be added.

Number	Label
0	Menu of these pages
1	Menu of subpages
2	Site map
3	Section overview with page contents
4	Menu of subpages with table of contents
5	Altered pages
6	Related pages by keyword
7	Menu of subpages with page contents

For the menu to actually work, it has to be modelled properly. This could look as follows:

```

TYPOSCRIPT
tt_content.menu.20.x = HMENU
tt_content.menu.20.x.1 = TMENU
tt_content.menu.20.x.1.NO.linkWrap = ||*||*||
tt_content.menu.20.x.1.NO.allWrap = |
tt_content.menu.20.x.1.target = _top
tt_content.menu.20.x.special = directory
tt_content.menu.20.x.special.value.field = pages

```

Listing 15

THE AUTHOR

Daniel Koch works as a freelance developer in the field of CM systems and is the author of several technical works about web technologies. He prefers to develop large web-projects on a TYPO3 foundation, which he can tailor to the needs of his clients with TypoScript. The present article is an excerpt from the new edition of „TYPO3 und TypoScript“, published by Hanser.