

Update auf Version 5.3.57

Ein einfaches `<div>` im Rich-Text-Editor: einfügen, Klassen zuweisen, wieder auflösen. Seit dem Wechsel von CKEditor 4 auf 5 war das ohne Quelltextansicht nicht mehr möglich.

Eine Änderung, die bestehende Inhalte sichtbar betrifft: Das Werkzeug beansprucht beim Einlesen jedes `div`, das keinem anderen Werkzeug gehört. Im Editor erscheinen sie ab jetzt als umrandeter Block. Am gespeicherten Markup ändert sich nichts – Abschnitt 2 erklärt, warum das so sein muss und wie man es abschaltet.

Dazu zwei Dinge aus der Praxis: Die `h1` kann jetzt den Navigationstitel nehmen statt des Seitentitels. Und beim eigenen SCSS ist ab jetzt eindeutig der Konfigurations-Datensatz die Quelle – das Site-Setting `keepVariables` entfällt.

Dazu ein behobener Fehler mit großer Wirkung: Priorisiertes Inline-JavaScript fremder Extensions lief zweimal. Bei einem globalen `let` brach dadurch die gesamte erzeugte JavaScript-Datei ab — die Seite lud nie fertig.

VERSION

5.3.57

TYP03

^14.3

PHP

8.2 – 8.5

Neues Werkzeug	Toolbar-Eintrag Div , hinter <i>Klassen</i> . Einfügen umschließt die Markierung, <i>Div auflösen</i> nimmt die Hülle weg und behält den Inhalt.
Sichtbar im Backend	Das <code>div</code> ist ein Widget: Auswahlrahmen, graue Umrandung, die Klassen als Etikett in der Ecke. Im Frontend nichts davon.
Achtung	Beansprucht jedes <code>div</code> außer <code>alert</code> , <code>row</code> und <code>col-*</code> . Bestehende Texte sehen im Editor anders aus – das Markup bleibt gleich.
Klassen	Über das Werkzeug <i>Klassen</i> , das eine dritte Gruppe Div bekommen hat. Liste wie gehabt über <code>t3sbClassPicker</code> .
Verschachtelung	Zwei Ebenen tief. Darüber ist der Knopf aus.
Abschaltbar	<code>rteDiv</code> in der Extension-Konfiguration unter <i>RTE</i> , pro Seitenbaum über <code>RTE.t3sbootstrap.features</code> .
Navigationstitel	Neuer Schalter im Konfigurations-Datensatz unter <i>Page title (h1)</i> : die <code>h1</code> nimmt <code>nav_title</code> statt <code>title</code> . Ab Werk aus.
Eigenes SCSS	Der Konfigurations-Datensatz ist jetzt durchgängig die Quelle. <code>bootstrap.cdn.keepVariables</code> entfällt, und <code>t3sbootstrap:customScss</code> überschreibt vorhandenes SCSS nicht mehr.
Behoben	Priorisiertes Inline-JavaScript stand zweimal auf der Seite – einmal inline, einmal in der erzeugten Datei. Ein globales <code>let</code> beendete damit die ganze Datei.
Achtung	Eine neue Datenbankspalte <code>page_titlenavtitle</code> . Datenbankanalyse ausführen, danach den Konfigurations-Datensatz einmal speichern.

1

Warum es dafür ein Plugin braucht

General HTML Support hält ein `<div>` im Markup — es verschwindet beim Speichern nicht mehr, wie es CKEditor 5 ohne diese Erweiterung täte. Was GHS nicht tut: daraus ein Widget machen. Und ohne Widget gibt es keinen Auswahlrahmen, keine Kennzeichnung im Editor und nichts, woran ein Toolbar-Eintrag ansetzen könnte.

Praktisch heißt das: Ein Redakteur sieht nicht, dass dort ein Container steht, kann ihn nicht auswählen und ihm folglich auch keine Klasse zuweisen. Der einzige Weg führte über die Quelltextansicht — und das ist keiner, den man einem Redakteur zumuten will. In CKEditor 4 gab es dafür einen Knopf.

Das neue Werkzeug führt `div` deshalb als eigenes Modellelement `bsDiv` und zeichnet es im Editor als editierbares Widget aus. Damit ist es ein Element wie Alert oder Spalte: auswählbar, beschreibbar, mit Werkzeugen erreichbar.

2

Was sich an bestehenden Inhalten ändert

Damit ein `div` bedienbar wird, muss das Plugin es beim Einlesen beanspruchen. Es beansprucht jedes — bis auf die, die einem anderen Werkzeug gehören:

```
beansprucht:      jedes <div>
nicht beansprucht: .alert      (Alert-Werkzeug)
                  .row        (Spalten-Werkzeug)
                  .col-*      (Spalten-Werkzeug)
```

Für dich heißt das: Ein Text, in dem schon `div` stehen, sieht im Editor nach dem Update anders aus. Wo vorher ein unsichtbarer Block war, steht jetzt ein umrandeter Kasten mit seinen Klassen in der Ecke. Am gespeicherten Markup ändert sich nichts.

WARUM ALLE ATTRIBUTE MITGEFÜHRT WERDEN

Sobald ein Element beansprucht ist, hebt General HTML Support nichts mehr dafür auf. Ein `id`, ein `data-` Attribut oder ein inline gesetztes `style="column-count:2"` wären beim ersten Speichern verloren. Das Plugin trägt deshalb alle Attribute im Modell mit und schreibt sie unverändert zurück.

Wer das nicht will, schaltet `rteDiv` in der Extension-Konfiguration unter *RTE* aus. Dann verschwindet der Knopf, die `div` sind wieder unsichtbare Blöcke, und GHS hält sie wie bisher. Pro Seitenbaum geht es über `RTE.t3sbootstrap.features.div`.

Die Abgrenzung sitzt im Upcast-Callback und nicht in der Konverter-Priorität: Bei gleicher Priorität entscheidet die Reihenfolge, in der die Plugins registriert wurden, und darauf sollte sich nichts verlassen, was Markup beansprucht.

3 Bedienung

Einfügen	Mit Markierung: die markierten Blöcke werden umschlossen, sofern sie denselben Elternknoten haben. Ohne: ein leeres <code>div</code> mit einem Absatz, Cursor steht drin.
Auflösen	<i>Div auflösen</i> entfernt die Hülle, der Inhalt rückt an ihre Stelle. Nichts geht verloren.
Verschachteln	Zwei Ebenen. Tiefer ist der Eintrag ausgegraut — ein drittes <code>div</code> im <code>div</code> ist im Editor nicht mehr auseinanderzuhalten.
Raus kommen	<i>Enter</i> auf der letzten leeren Zeile springt hinter das <code>div</code> , <i>Backspace</i> auf der ersten davor — dieselbe Geste wie beim Alert.

Klassen

Die setzt das Werkzeug *Klassen* aus 5.3.56, das dafür eine dritte Gruppe bekommen hat. Steht der Cursor in einem `div`, erscheint neben *Zeile* und *Spalte* die Gruppe **Div**. Konfiguriert wird sie genauso:

```
t3sbClassPicker:
  div:
    padding: { class: 'p-3', label: 'Innenabstand' }
    border: { class: 'border', label: 'Rahmen' }
    rounded: { class: 'rounded', label: 'Ecken gerundet' }
    textCenter: { class: 'text-center', label: 'Text zentriert' }
    spalten: { class: 'columns-2', label: 'Zweispaltiger Text' }
```

Ein Sitepackage überschreibt einen Eintrag, entfernt ihn mit `null` oder ergänzt eigene; ein Eintrag darf mehrere Klassen tragen. Die letzte Zeile oben zeigt den Fall, für den das Ganze angefragt wurde: eine Projektklasse mit `column-count`, die vorher nur über den Quelltext zu setzen war.

4 Navigationstitel als Seitentitel

Die `h1` nahm bisher ausnahmslos den Seitentitel. Lange, für die Suche geschriebene Titel stehen damit auch über der Seite, obwohl im Menü die kurze Fassung steht. Der Konfigurations-Datensatz hat unter *Page title (h1)* jetzt den Schalter **Navigationstitel verwenden**.

An	Die <code>h1</code> nimmt das Seitenfeld <code>nav_title</code> .
Ohne <code>nav_title</code>	Die Seite behält ihren Seitentitel — eine Option darf keine leere Überschrift erzeugen.
Aus	Werkseinstellung. Bestehende Seiten ändern sich nicht.

NEUES DATENBANKFELD

`page_titlenavtitle` in `tx_t3sbootstrap_domain_model_config`. Nach dem Update die Datenbankanalyse ausführen und den Konfigurations-Datensatz einmal speichern — erst dabei wird die neue Konstante in `t3sbconstants.typoscript` geschrieben.

5 Eigenes SCSS: der Datensatz ist die Quelle

Das eigene SCSS hatte bisher zwei mögliche Quellen – den Konfigurations-Datensatz und die Datei unter `typo3temp/assets/t3sbootstrap/T3SB-SCSS/`. Welche von beiden galt, entschied das Site-Setting `bootstrap.cdn.keepVariables`. **Dieses Setting entfällt.**

Es hat sein Versprechen ohnehin nicht gehalten: Beim Speichern im Backend wurde die Datei immer aus dem Datensatz überschrieben, der Schalter wirkte nur gegen den Scheduler-Task. Er hielt die Datei also ausschließlich dann, wenn beide Felder im Datensatz leer blieben – und genau dann lag der Inhalt nur noch in `typo3temp/`.

WARUM DAS DIE LÜCKE WAR, DIE ES SCHLIESSEN SOLLTE

`typo3temp/` ist im Core ausdrücklich als jederzeit löschar deklariert. Jedes TYPO3-Update, jedes Deployment und jeder Cache-Flush leert es – gemanagte Hosts tun das bei jedem Update-Lauf. Was dort als einzige Quelle lag, war danach weg, ohne Meldung und ohne Weg zurück.

Was jetzt gilt

LAGE	VERHALTEN
Datensatz hat Inhalt	Die Datei wird daraus geschrieben. Der Inhalt bleibt, wie er ist.
Datensatz leer, Datei da	Die Datei wird einmalig in den Datensatz übernommen – der Altbestand aus <code>keepVariables = 1</code> .
Beides leer	Initialisierung aus dem Bootstrap-Theme oder dem Vorgabe-Kommentar, wie bisher.

`t3sbootstrap:customScss` setzt vorhandenes SCSS damit **nicht mehr zurück**. Bis 5.3.56 hat das Kommando den Inhalt bei jedem Lauf auf das Bootstrap-Theme bzw. den Vorgabe-Kommentar überschrieben – dafür waren die Zeitstempel-Kopien `<zeit>-custom-<pid>.scss` neben der Datei da. Beides ist weg: der Reset und die Kopien.

Wer das Bootstrap-Theme neu ziehen will, leert das Feld im Backend und lässt das Kommando laufen. Explizit, statt als stiller Nebeneffekt.

Sichern und übertragen

Dass der Datensatz die Quelle ist, ist die Voraussetzung dafür, dass die beiden vorhandenen Werkzeuge überhaupt greifen – eine Datei sieht keines von beiden:

Übertragen	Konfiguration übertragen im Backend-Modul. Der JSON-Export enthält <code>custom_scss</code> und <code>custom_variables_scss</code> und ist damit zugleich die Sicherung dieser Inhalte.
Zurückrollen	Die Datensatz-Historie von TYPO3 am Konfigurations-Datensatz – mit Diff und einzeln rücknehmbar.

ZU TUN BEIM UPDATE

Stand `keepVariables` auf `0` – der Werkseinstellung –, ist nichts zu tun. Stand es auf `1`, einmal `vendor/bin/typo3 t3sbootstrap:customScss <Rootseiten-ID>` laufen lassen, **bevor** der Konfigurations-Datensatz gespeichert wird. Dann wandert die Datei in den Datensatz. Das Setting in der Site-Konfiguration kann danach entfernt werden; es wird ignoriert.

Neu ist außerdem ein Hinweis im Backend-Modul, wenn bei aktiviertem SCSS die Bootstrap-Quellen unter `T3SB-Bootstrap/Bootstrap/scss/` fehlen. Bisher meldete das Modul nur fehlende CDN-Assets; fehlten die Quellen, blieb die Seite stumm unformatiert.

6

Behoben: priorisiertes Inline-JavaScript lief zweimal

Registriert eine Extension ihr Inline-JavaScript mit `priority` — Cookie-Consent-Lösungen tun das regelmäßig, weil ihr Code vor allem anderen laufen muss —, stand es danach zweimal auf der Seite: einmal inline im HTML und ein zweites Mal in der Datei, die der Asset-Renderer unter `typo3temp/assets/` erzeugt.

Bei `var` fällt das niemandem auf. Bei einem globalen `let`, `const` oder `class` wirft der zweite Durchlauf einen Redclaration-Error — und der beendet die gesamte erzeugte Datei, nicht nur die doppelte Stelle. Sichtbar wurde das als Seite, die nie fertig lädt, weil der Preloader stehen bleibt.

Woran es lag

`PageRenderer` arbeitet in dieser Reihenfolge:

```
renderInlineJavaScript(true)  Ereignis (inline, priority) -> Listener
                             danach: Core schreibt das priorisierte
                             Inline-JS ins HTML
renderInlineJavaScript(false) Ereignis (inline, non-priority) -> Listener
                             sammelt ein und schreibt die Datei
renderJavaScript(false)      bindet die erzeugte Datei ein
```

Das Rendern liest den Asset-Collector nur, es entfernt nichts. Der Listener holte die Skripte mit `getInlineJavaScripts()` — ohne Argument, und ohne Argument filtert die Methode nicht nach Priorität. Also kamen auch die Skripte mit, die eine Zeile zuvor bereits im HTML gelandet waren. Das `removeInlineJavaScript()` danach kam für sie zu spät.

Die Schleife liest jetzt `getInlineJavaScripts(false)`. Priorisiertes Inline-JavaScript bleibt damit inline — wofür `priority` da ist; bisher landete es in einer Datei im Footer, also genau dort, wo es nicht hingehört.

WER BISHER AUSGEWICHEN IST

Der verbreitete Workaround war `plugin.tx_t3sbootstrap_m1.settings.disableInlineJs = 1`. Der Schalter kann zurückgenommen werden. Danach Cache leeren und die alten `typo3temp/assets/t3sbootstrap_*.js` löschen — sie tragen den Hash ihres alten Inhalts und werden sonst weiter ausgeliefert.

Die eigenen Assets der Extension waren nie betroffen: Sie werden ohne Priorität registriert und wandern wie bisher in die erzeugte Datei.

7

Checkliste

• Pflicht

- ☐ **Paket aktualisieren** — über Composer, Befehle unten
- ☐ **Datenbank analysieren** — eine neue Spalte `page_titlenavtitle` in `tx_t3sbootstrap_domain_model_config`. Ohne sie lässt sich der Konfigurations-Datensatz nicht speichern
- ☐ **Caches leeren** — TCA, Sprachdateien und die kompilierten Fluid-Templates
- ☐ **Konfigurations-Datensatz einmal speichern** — erst dabei wird die neue Konstante nach `t3sbconstants.typoscript` geschrieben

```
composer update t3sbs/t3sbootstrap
vendor/bin/typo3 extension:setup
vendor/bin/typo3 cache:flush
```

• Nur wenn es dich betrifft

- ☐ `keepVariables` stand auf `1` — einmal `vendor/bin/typo3 t3sbootstrap:customScss <Rootseiten-ID>` laufen lassen, **bevor** der Datensatz gespeichert wird. Danach `bootstrap.cdn.keepVariables` aus der Site-Konfiguration entfernen, es wird ignoriert
- ☐ `disableInlineJs` war gesetzt — zurücknehmen, Cache leeren, alte `public/typo3temp/assets/t3sbootstrap_*.js` löschen
- ☐ **Das Div-Werkzeug ist nicht gewünscht** — `rteDiv` unter *RTE* ausschalten, Cache leeren

• Zum Prüfen

- ☐ **Ein Text mit vorhandenen `div`** — öffnen, speichern, danach das Feld `bodytext` vergleichen. Es sollte unverändert sein

✓ FERTIG!

Kein Upgrade-Wizard — eine neue Spalte und ein entfallenes Site-Setting, mehr nicht.