

CKEditor 5

Plugins schreiben für die neue Version

Gliederung

- CKEditor 5
- Plugins in CKEditor 4 vs. CKEditor 5
- Architektur
- UI
- Editing Engine
- DevTools

CKEditor 5

- Erstes Announcement am 15.10.2015
- Developer Preview 1 am 14.07.2016 (🎉)
- Stable Release (CKEditor 5 v10.0.0) am 26.4.2018

CKEditor 5 ist kein einfacher Editor mehr, es ist ein

✨ Framework ✨

CKEditor 4 vs. CKEditor 5

Scrollbar for scale!

Case Lux CKEditor-Plugin “email4link”

A screenshot of a code editor showing 172 lines of code for CKEditor 4. The code is color-coded and includes comments. A vertical scrollbar is visible on the right side of the editor window.

CKEditor 4
172 Zeilen Code

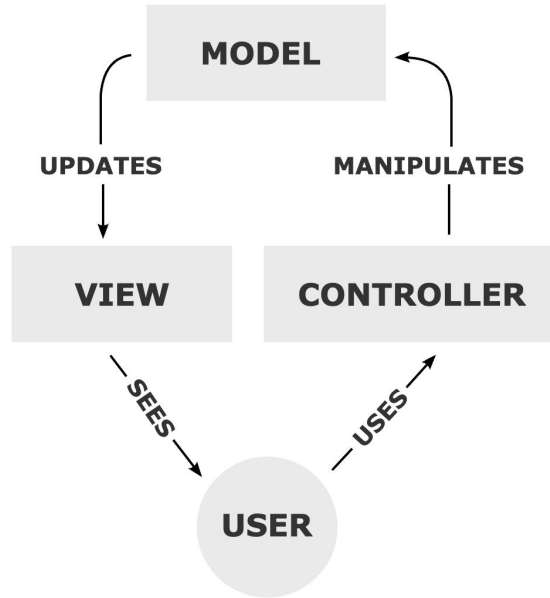
CKEditor 5
753 Zeilen Code

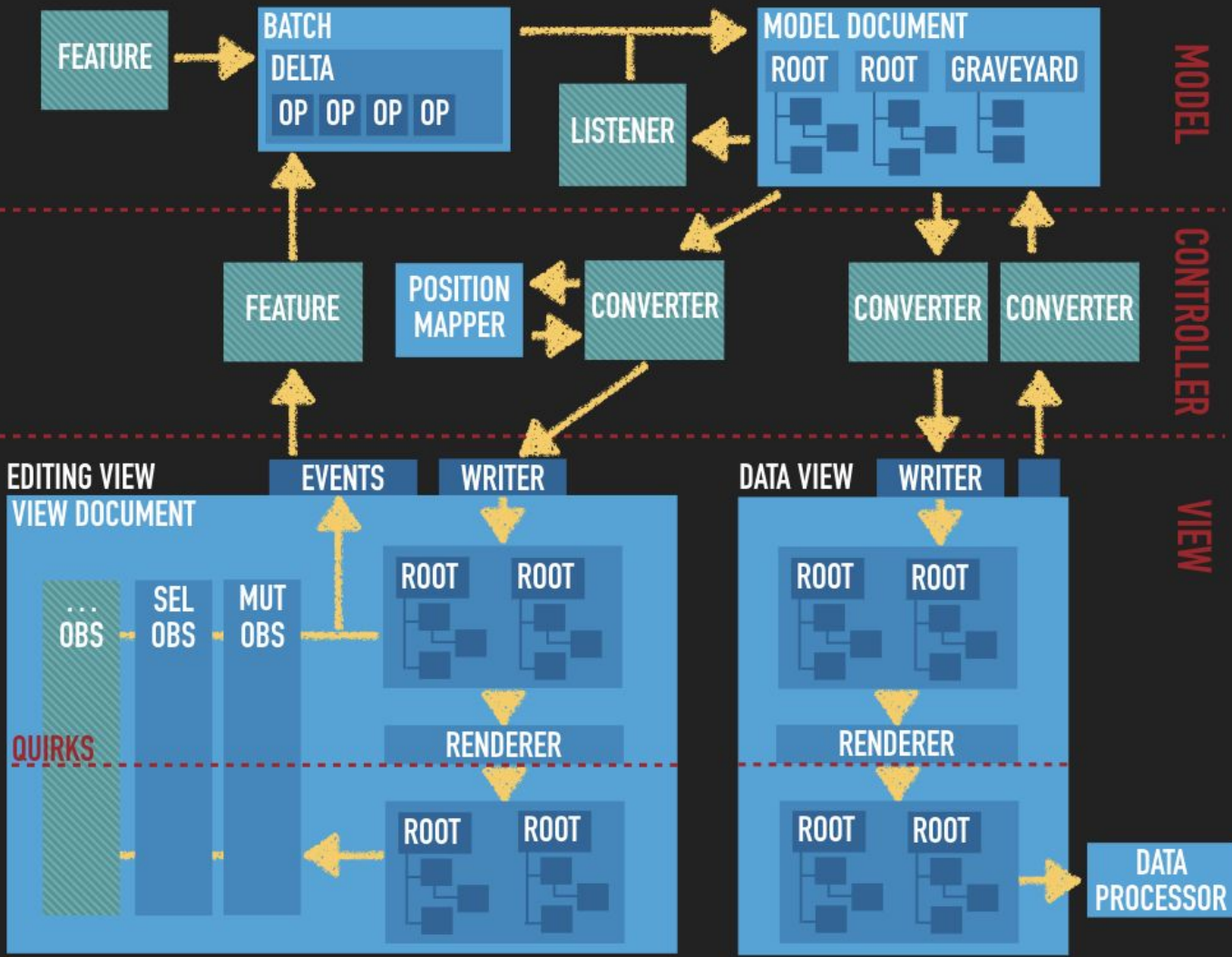
A screenshot of a code editor showing 753 lines of code for CKEditor 5. The code is color-coded and includes comments. A vertical scrollbar is visible on the right side of the editor window.

Der Screenshot passt nicht
mal in die Folie 😭

Architektur

Klassisches MVC







Architektur

- **Core:** Kern der Anwendung und Schnittstelle für andere Anwendungsteile
- **Editing Engine:** CKEditor Model <-> HTML Konvertierung
- **UI:** Buttons und Dialoge

Alles sind Plugins!

```
export default class Email4Link extends Core.Plugin {  
  static get requires() {  
    return [ Email4LinkEditing, Email4LinkUI ];  
  }  
}
```

UI und Editing-Engine für unser Plugin sind also nochmal selbst Plugins.

Alles sind Plugins!

Ein Plugin kann andere Plugins, innerhalb der statischen Methode `requires()` requiren.

Soll ein Plugin direkt Code ausführen, macht man das in einer `init()`-Methode innerhalb eines Plugins.

UI

Code-Demonstration!

UI:

<https://github.com/in2code-de/lux/blob/develop/Resources/Public/JavaScript/Static/CkEditorPlugins/luxEmail4Link/email4link/ui.js>

FormView:

<https://github.com/in2code-de/lux/blob/develop/Resources/Public/JavaScript/Static/CkEditorPlugins/luxEmail4Link/email4link/view.js>

TextArea:

<https://github.com/in2code-de/lux/blob/develop/Resources/Public/JavaScript/Static/CkEditorPlugins/luxEmail4Link/ui/textareaview.js>

Editing Engine - CKEditor 4

In CKEditor 4 war der Editor ein iFrame in dem man mithilfe von Hilfsmethoden HTML-Elemente bearbeitet/eingesetzt hat:

```
var getParentElement = function (elementType, currentSelection) {
    var parentElements = currentSelection.getParents();
    for (var i = 0; i < parentElements.length; i++) {
        if (parentElements[i].getName() === elementType) {
            return parentElements[i];
        }
    }
    return null;
};
```

```
// ...
```

```
var parent = getParentElement('a', editor.getSelection().getStartElement());
parent.setAttribute('data-lux-email4link-title', data.title);
parent.setAttribute('data-lux-email4link-text', data.text);
parent.setAttribute('data-lux-email4link-sendEmail', data.sendEmail);
```

Editing Engine - CKEditor 5

In CKEditor 5 bearbeitet man kein HTML mehr, sondern das “CKEditor Model” bei dem Attribute auf Selection-Nodes gesetzt werden.

```
_getSelectedLink() {
  const model = this.editor.model;
  const selection = model.document.selection;

  if (selection.hasAttribute('linkHref')) {
    return selection;
  }

  return null;
}

// ...

this.editor.model.change( writer => {
  const link = this._getSelectedLink();

  if (link) {
    const linkRange = Typing.findAttributeRange(link.getFirstPosition(), 'linkHref', link.getAttribute('linkHref'), editor.model);

    writer.setAttribute('sendEmail', String(sendEmail), linkRange)
    writer.setAttribute('emailTitle', String(title), linkRange)
    writer.setAttribute('emailText', String(text), linkRange)
  }
});
```

Editing Engine - Model

Einige Block-Tags existieren noch, aber alle Inline-Elemente sind in CKEditor 5 Attribute die auf Textabschnitten stehen, diese haben auch noch andere Namen, Beispiel linkHref für Links.

```
<paragraph>  
  "Hello world!"  
  "This is a link" linkHref  
</paragraph>
```

Editing Engine - Converter

Das CKEditor-Model wird mithilfe von Convertern zu HTML oder von HTML zurück in das Model umgewandelt.

Hierfür definiert man einen upcast und downcast für Attribute.

downcast = CKEditor-Model -> HTML

upcast = HTML -> CKEditor-Model

Editing Engine

Code-Demonstration!

Editing Engine:

<https://github.com/in2code-de/lux/blob/develop/Resources/Public/JavaScript/Static/CkEditorPlugins/luxEmail4Link/email4link/editing.js>

Command:

<https://github.com/in2code-de/lux/blob/develop/Resources/Public/JavaScript/Static/CkEditorPlugins/luxEmail4Link/email4link/command.js>

DevTools

Als Plugin-Entwickler ein Muss: Der CKEditor-Inspektor.

Diesen kann man im Plugin-Code mitinstallieren oder per JS-Booklet einsetzen. Das Booklet funktioniert im TYPO3-Backend wegen der Frames aber nicht.


Das JS kann man trotzdem kopieren und in der Browserkonsole im richtigen Frame ausführen.

Dokumentation:

<https://ckeditor.com/docs/ckeditor5/latest/framework/development-tools.html#ckeditor-5-inspector>

Text [bodytext]

Styles ▾ Paragraph ▾ **B** *I* \times_2 \times^2 (-) \therefore $\frac{1}{2} =$ “ ≡ ▾ ↺ ▾ @ *I*_x ↶ ↷ 🗒 ▾

Ω ▾ —  Source

Dieser [Testparcours](#) bildet alle gängigen Frontend-Funktionen der Extension Powermail ab.

Über Behat Tests (mit Mink und Selenium) wird vor **jedem** Release (GIT Tag oder TER upload) überprüft, ob die Extension und deren Frontend-Plugins wie erwartet funktioniert.

Model View Commands Schema

Instance: editor-1 ▾ >_ 📄 📁 ✂ 🗑 ▾

Root: main ▾ Compact text Show markers

Inspect Selection Markers

```
<$root>
<paragraph>
  "Dieser "
  linkHref "Testparcours"
  " bildet alle gängigen Frontend-Funktionen der Extension Powermail ab."
</paragraph>
<paragraph>
  "Über Behat Tests (mit Mink und Selenium) wird vor "
  bold "jedem"
  " Release (GIT Tag oder TER upload) überprüft, ob die Extension und deren F
rontend-Plugins wie erwartet funktioniert."
</paragraph>
<paragraph>
  "Fraaen zu Powermail? "
```

Text: "Testparcours"

Attributes

linkHref: "t3://page?uid=4"

Properties

startOffset: 7
endOffset: 19
offsetSize: 12
path: [0, 7]