08.08.2013

Aktuell zu alpha5

NEOS

TYPO3 Neos

Das Kompendium

typovision

Patrick Lobacher

Geschäftsführer

typovision

Feedback erwünscht

- Lieber Neos-Enthusiast!

  Ich versuche das TYPO3 Neos Kompendium stets aktuell zu halten und ständig zu ergänzen. Aber dafür brauche ich Deinen Input! Wenn Du Ideen hast, Code-Beispiele, FAQ-Themen oder schlicht Lob (oder Kritik), dann schreibe mir bitte an die folgende Adresse:

  patrick.lobacher [AT] typovision.de

  Viel Spaß bei Kompendium!
  Patrick Lobacher

typovision

Was ist

TYPO3 Neos?

NEOS

Take a peek on the next generation of content management.


nieuw
ny
новый
新た
jauns
νέος
new
neu
nuevo
nový

typovision

Die Geschichte von TYPO3 Neos beginnt bei TYPO3 CMS

TYPO3

- TYPO3 CMS ist ein „Enterprise Open Source Content Management Framework“
- TYPO3 CMS existiert seit 1998 / Erfunden vom Dänen Kaspar Skårhøj
- ca. 500.000 Installationen weltweit / > 5 Mio Downloads
- Einsatz in DE z.B. bei > 50% aller DAX 500 Unternehmen, > 50% aller Bundesliga-Vereinen, Discounter, Autovermieter, Öffentliche Träger
- > 6.000 Extensions
- > 100.000 Entwickler weltweit
- > 1500 Agenturen weltweit
- Finanziert und unterstützt von der TYPO3 Association

typovision

Die Geschichte von TYPO3 Neos: TYPO3 Phoenix

- Auf den ersten T3DD (TYPO Developer Days) im Jahr **2006** wurde der Entschluss gefasst, TYPO3 von Grund auf neu zu schreiben
- Codename: TYPO3 Phoenix (bzw. TYPO3 5.0)
- Einige benötigte Features gab es seinerseits in PHP noch nicht
- Das Projekt wurde von dem Chefarchitekten Robert Lemke begonnen und nach wenigen Monaten von Karsten Dambekalns flankiert
- Mit dem „Berlin Manifesto“ wurde **2008** der Rahmen und die Abgrenzung zum TYPO3 CMS festgelegt

  `http://typo3.org/roadmap/berlin-manifesto/`

Die Geschichte von TYPO3 Neos: TYPO3 Flow und Neos

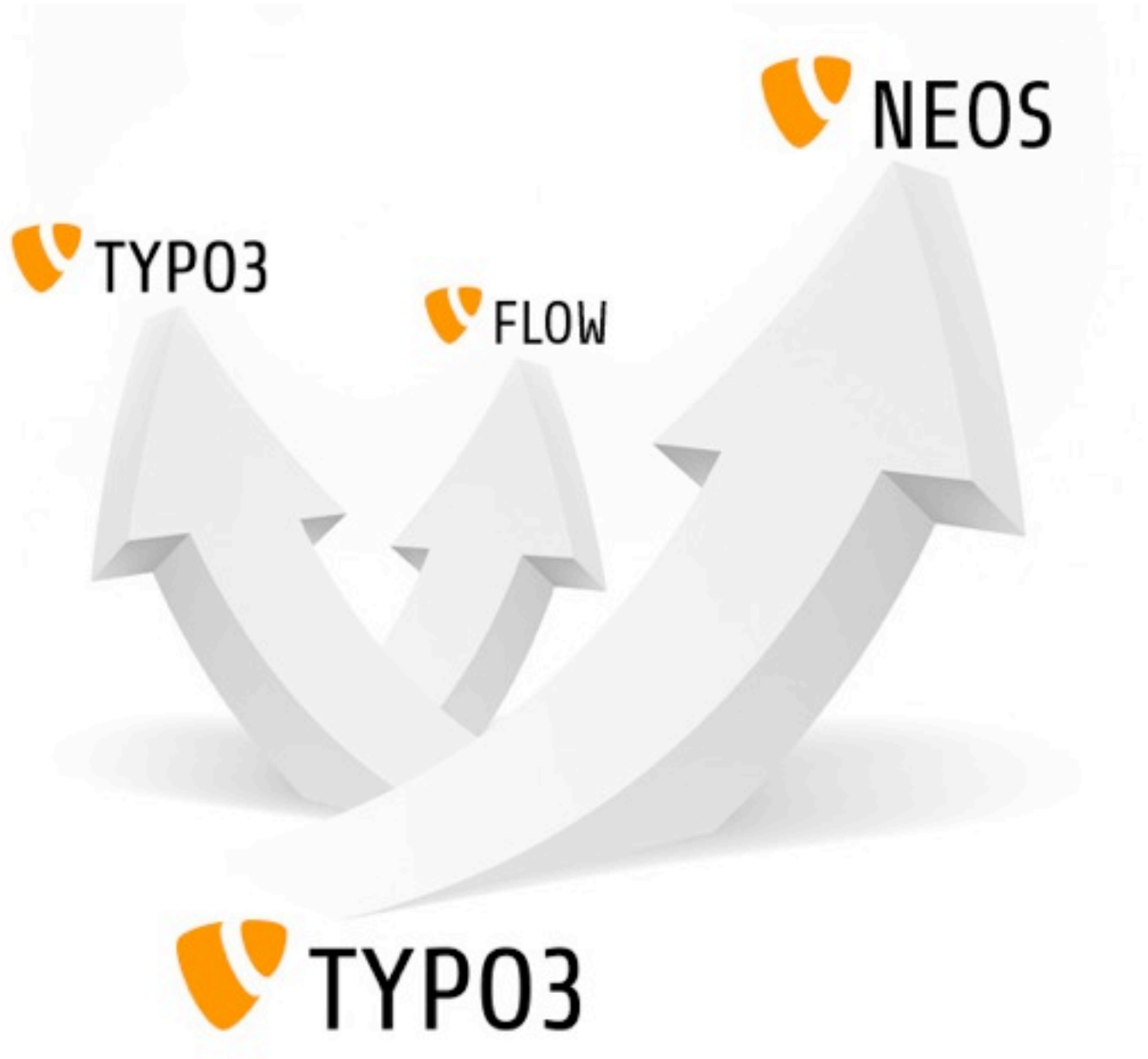
- Viele Grundfunktionen eines CMS sind nicht CMS-spezifisch (Session-Handling, Datenbank-Handling, Templating, ...) => daher Trennung dieser Funktionalitäten in ein eigenes Framwork TYPO3 Flow
- Durch die Einführung von Extbase im Jahr **2009** wurde es möglich, bereits in TYPO3 CMS Extensions zu schreiben, die in TYPO3 Flow mit geringen Änderungen lauffähig sind
- Am 20. Oktober **2011** wurde das Application Framework **TYPO3 Flow** (ehemals FLOW3) als Final veröffentlicht
- **TYPO3 Neos** ist eine Applikation die auf TYPO3 Flow basiert
- TYPO3 Neos Alpha 5 im August **2013**
- Erste finale Version im Oktober **2013** (Geplant)

FLOW

NEOS

Die TYPO3 Welt - seit Oktober 2012

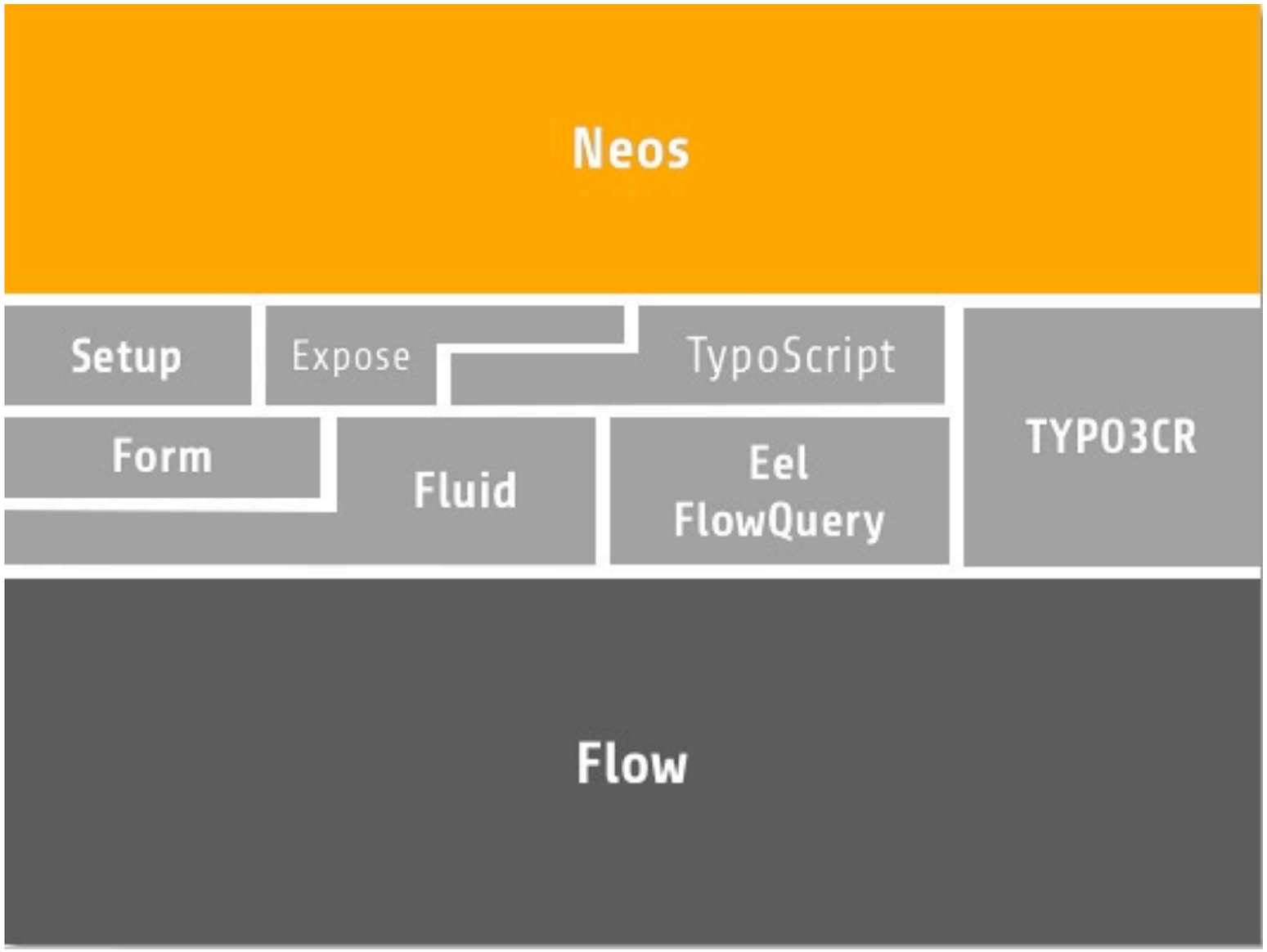
- TYPO3 CMS
- TYPO3 Flow
- TYPO3 Neos
- TYPO3 Surf



Die Architektur **von TYPO3 Neos**

typovision

Die Architektur von TYPO3 Neos - Backend



Fluid
Modern Templating Engine

TYPO3CR
Content Repository (JCR / Sling)

TypoScript
TypoScript 2.0 - next Generation

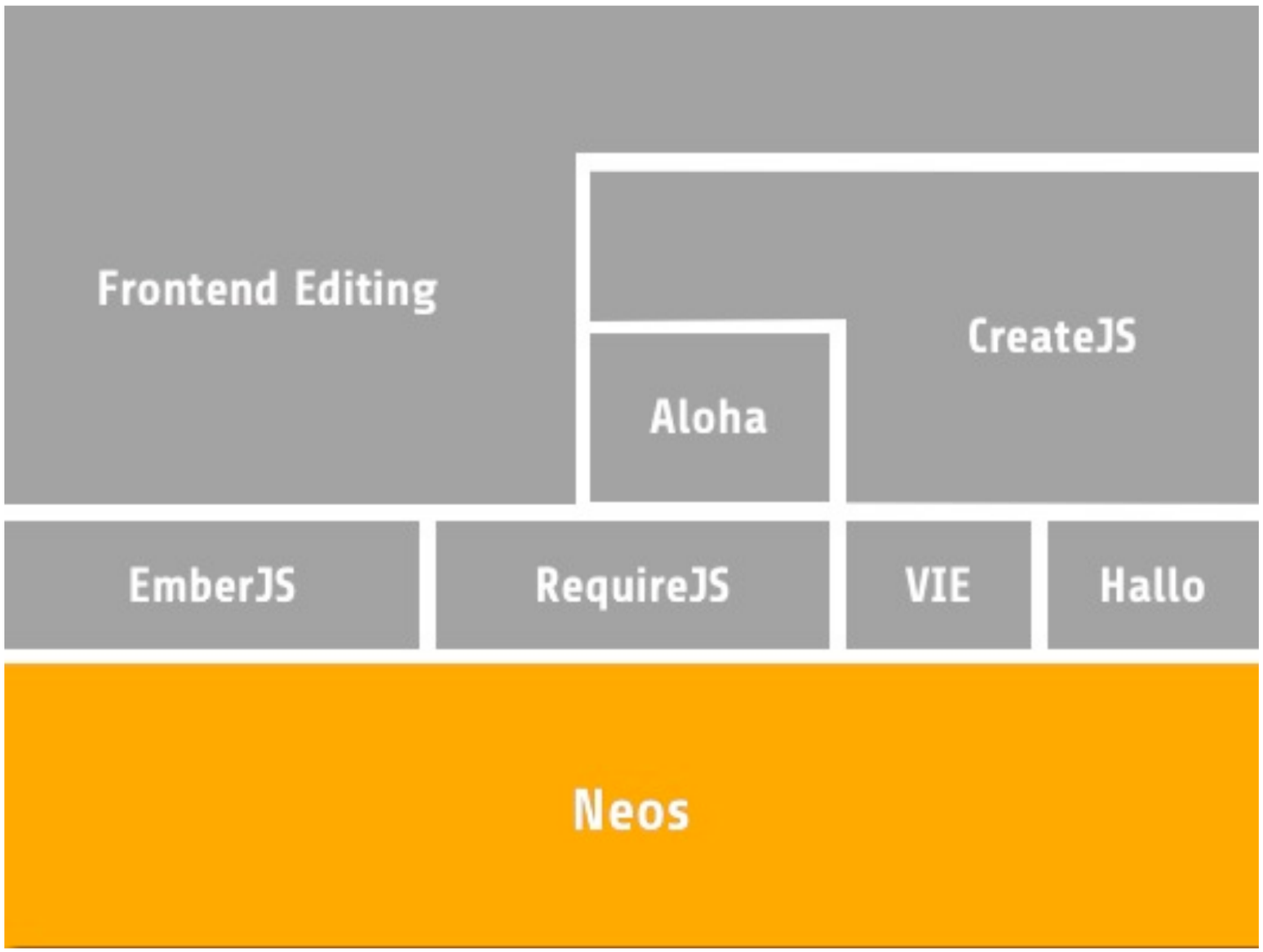
Forms
Form API & Form Builder

Expose
Extensible admin interface

Eel
Embedded Expression Language

typovision

Die Architektur von TYPO3 Neos - Frontend



EmberJS
JavaScript web application framework

Create.js
Web Editing Interface

Aloha / Hallo
HTML5 WYSIWYG Editor

VIE = viejs.org
Semantic Interaction Framework

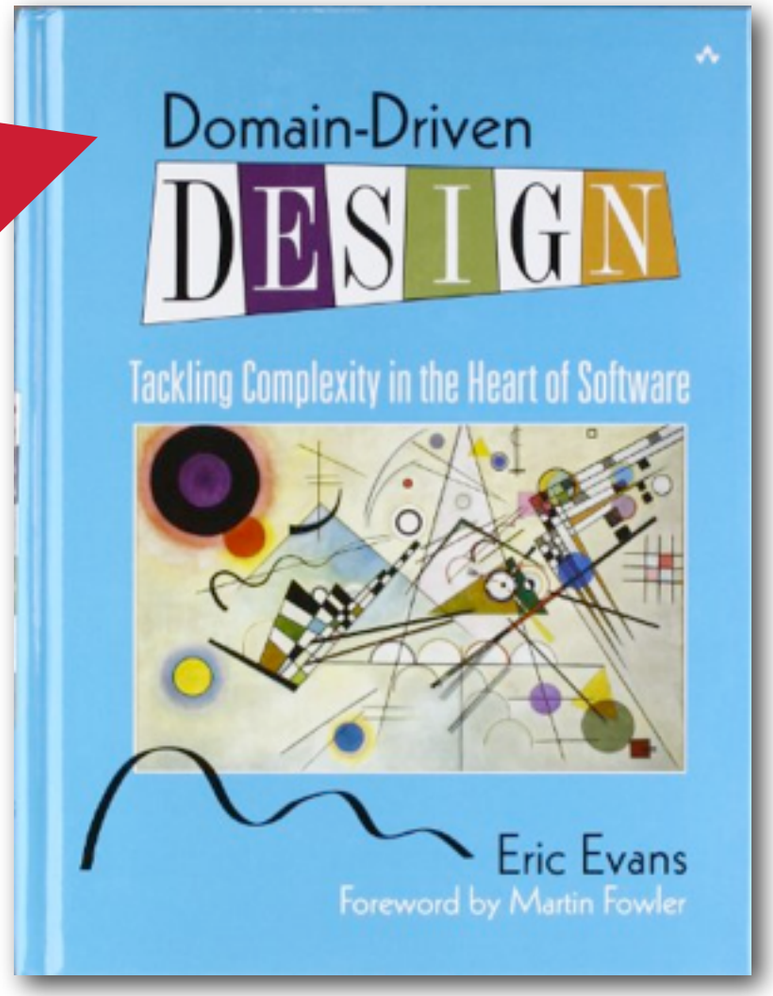
RequireJS
JavaScript file and module loader

typovision

Die Grundlagen von **TYPO3 Flow & Fluid**

TYPO3 Flow - Basis Design

Achtung: Buzzword-Bingo :-)

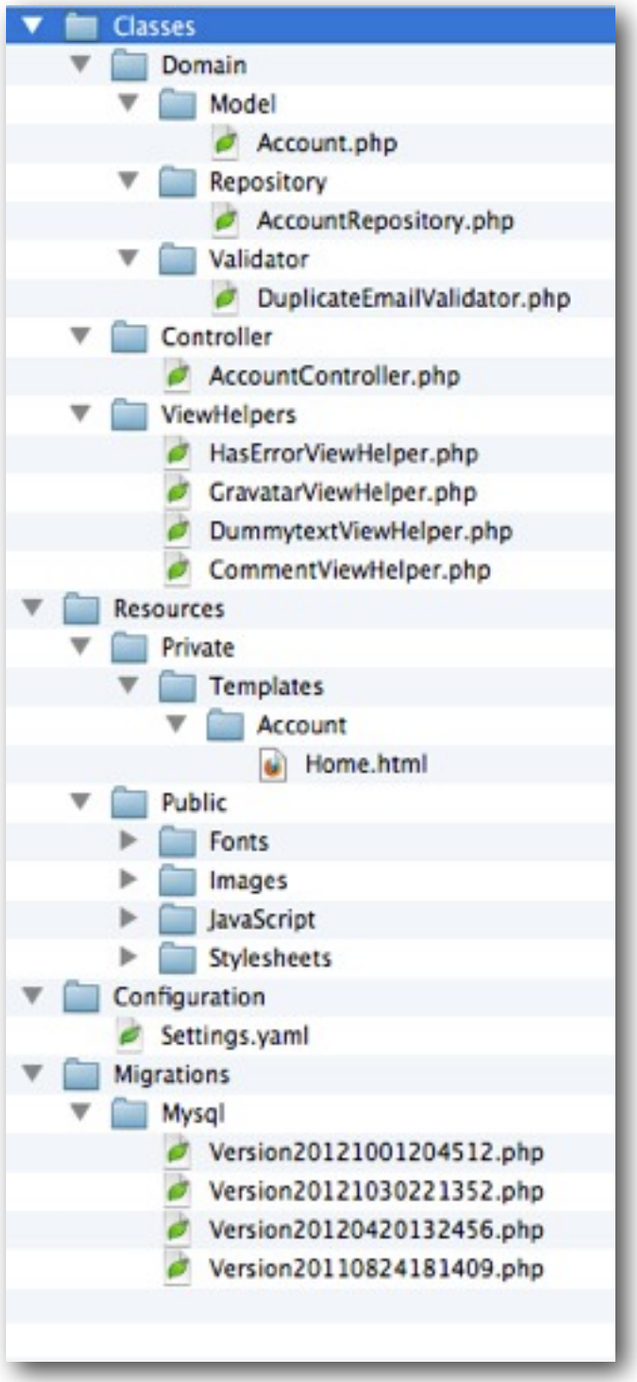


- **OOP** - Vollständig Objektorientiert
- **TDD** - Test Driven Development
- **DDD** - Domain Driven Design (Eric Evans)
- **MVC** - Model, View, Controller
- **AOP** - Aspect Oriented Programming (Separation of Concerns, Cross cutting concerns)
- **DI** - Dependency Injection
- **ORM** - Basiert auf Doctrine 2 (eigenes ORM möglich)

typovision

TYPO3 Flow - Paradigmen + Struktur

- Packages erweitern das Grundsystem
- Convention over Configuration
- Verzeichnis- und Dateibenennung gemäß Ubiquitous Language (z.B. Models)
- Schreibweise von Verzeichnissen, Dateien und Klassen ist UpperCamelCase
- Schreibweise von Proerties ist lowerCamelCase



TYPO3 Flow - Beispiel Model

- Entity oder Value Object (DDD)
- Reflection durch „PHPDoc Annotations“
- Validierung am Model
- Relationen über Annotations
- Steuerung des ORM ist ebenfalls über Annotations möglich
- Getter und Setter für den Zugriff

```php
namespace TYPO3\Blog\Domain\Model;

/**
 * A Blog object
 *
 * @Flow\Entity
 */
class Blog {

    /**
     * @var string
     * @Flow\Validate(type="Text")
     * @Flow\Validate(type="StringLength", options={ "minimum"=1, "maximum"=80 })
     * @ORM\Column(length=80)
     */
    protected $title;

    /**
     * @var \Doctrine\Common\Collections\ArrayCollection<\TYPO3\Blog\Domain\Model\Post>
     * @ORM\OneToMany(mappedBy="blog")
     * @ORM\OrderBy({"date" = "DESC"})
     */
    protected $posts;

    ...

}
```

TYPO3 Flow - Beispiel Repository

- Magic-Methoden bereits out-of-the-box vorhanden
  - findAll()
  - findBy*propertyName*()
  - findOneBy*propertyName*()
  - ...
- Query Manager liefert Interface zu Query zurück
- Ausgeführt wird der Query erst bei „Benutzung“ der Daten

```php
/**
 * A PostRepository
 */
class PostRepository extends \TYPO3\Flow\Persistence\Repository {

    /**
     * Finds posts by the specified tag and blog
     *
     * @param \TYPO3\Blog\Domain\Model\Tag $tag
     * @param \TYPO3\Blog\Domain\Model\Blog $blog The blog the post must refer to
     * @return \TYPO3\Flow\Persistence\QueryResultInterface The posts
     */
    public function findByTagAndBlog(\TYPO3\Blog\Domain\Model\Tag $tag,
      \TYPO3\Blog\Domain\Model\Blog $blog) {
        $query = $this->createQuery();
        return $query->matching(
            $query->logicalAnd(
                $query->equals('blog', $blog),
                $query->contains('tags', $tag)
            )
        )
        ->setOrderings(array(
            'date' => \TYPO3\Flow\Persistence\QueryInterface::ORDER_DESCENDING)
        )
        ->execute();
    }
}
```

typovision

TYPO3 Flow - Beispiel Controller

- Dependency Injection über @Inject Annotation
- Persistierung ist automatisiert (am Ende der Action)
- Validierung ist automatisiert (am Anfang der Action)
- Slim-Controller
- Es gibt Methoden, die bei Anwesendheit automatisch aufgerufen werden:
  `initializeAction()`
  `initialize[ActionName]Action()`
  `errorAction()`

```php
<?php
namespace TYPO3\Blog\Controller;

use TYPO3\Flow\Annotations as Flow;

// ...

class SetupController extends \TYPO3\Flow\Mvc\Controller\ActionController {

	/**
	 * @Flow\Inject
	 * @var \TYPO3\Blog\Domain\Repository\BlogRepository
	 */
	protected $blogRepository;

	/**
	 * @Flow\Inject
	 * @var \TYPO3\Blog\Domain\Repository\PostRepository
	 */
	protected $postRepository;

	/**
	 * Sets up a fresh blog and creates a sample post.
	 *
	 * @return void
	 */
	public function indexAction() {
		$this->blogRepository->removeAll();
		$this->postRepository->removeAll();

		$blog = new \TYPO3\Blog\Domain\Model\Blog();
		$blog->setTitle('My Blog');
		$blog->setDescription('A blog about Foo, Bar and Baz.');
		$this->blogRepository->add($blog);
```

typovision

TYPO3 Fluid - modernes Templating

- Templating ist Objektorientiert
- XML-Valider Code
- View-Logik im View!
- ViewHelper (Klassen) unterstützen den View und sind beliebig wieder-verwendbar
- ViewHelper: Formulare, Links, Security, Schleifen, If, Formatierung, Widgets, ...,

```
$this->view->assign('blogs',$this->blogRepository->findAll());
```

```
<f:form>
    <f:form.textfield name="search" value="{search}" />
    <f:form.submit value="suchen!" />
</f:form>

<f:for each="{blogs}" as="blog">

    <li>{blog.title}
        [
        <f:link.action action="show" arguments="{blog:blog}">SHOW</f:link.action>
        |
        <f:link.action action="edit" arguments="{blog:blog}">EDIT</f:link.action>
        |
        <f:link.action action="delete" arguments="{blog:blog}">DELETE</f:link.action>
        ]
    </li>
</f:for>
</ul>

<f:link.action action="new">Einen neuen Blog anlegen</f:link.action>
```

Installation

von TYPO3 Neos

typovision

Systemvoraussetzungen von TYPO3 Neos

- Webserver (empfohlen ist Apache 2.x mit aktiviertem mod_rewrite Modul)
- PHP 5.3.2 - 5.4.x
- Folgende Funktionen müssen in PHP aktiviert sein: system(), shell_exec(), escapeshellcmd() und escapeshellarg()
- php.ini: memory_limit = 256M oder höher
- php.ini: xdebug.max_nesting_level = 500 (sofern xdebug verwendet wird)
- php.ini: Fügen sie die folgende Optionen ans Ende hinzu: detect_unicode = Off
- php.ini: Zudem muss Magic_Quotes ausgeschaltet werden: magic_quotes_gpc = Off
- php.ini: Die Kommandozeile von Flow benötigt ferner noch eine Zeitzonen-Einstellung: date.timezone= "Europe/Berlin"
- MySQL 5.1.50 - 5.x.x (zum Beispiel - grundsätzlich kann jede zum Doctrine DBAL kompatible Datenbank verwendet werden)
- Zugang zur Konsole

typovision

Installation von TYPO3 Neos - Composer

- Die Installation erfolgt über „Composer“
  (Dependency Manager für PHP) - Dafür ist Zugang zur Konsole nötig

```
cd /pfad/zum/webserver/

curl -sS https://getcomposer.org/installer | php
```

- Dies legt die Datei `composer.phar` im aktuellen Verzeichnis an
- Will man den Composer zentral verwenden, kann man ihn auch verschieben

```
mv composer.phar /usr/local/bin/composer
```

typovision

Installation von TYPO3 Neos - Composer

- Laden von TYPO3 Neos via Composer:

```
php /pfad/zu/composer.phar create-project -s alpha
--dev typo3/neos-base-distribution TYPO3-Neos
```

- Dies sorgt für die Installation von TYPO3 Flow, Neos und den benötigten Modulen (inkl. 3rd Party wie Doctrine 2, Aloha, ...)
- Anschließend erhält man ein Verzeichnis TYPO3-Neos, welches die letzte Version von Neos enthält
- Zur Installation von Composer unter Windows gibt es hier Infos:
  `http://getcomposer.org/doc/00-intro.md#installation-windows`

typovision

[Alternative:] GIT-Version von TYPO3 Neos verwenden

- Laden der aktuellsten GIT-Version von TYPO3 Neos:

```
git clone git://git.typo3.org/Neos/Distributions/
Base.git TYPO3-Neos && cd TYPO3-Neos
```

- Anschließend müssen noch die Abhängigkeiten geladen werden:

```
composer install --dev
```

typovision

Installation von TYPO3 Neos - Rechte und VirtualHost

- Anschließend werden die Datei-Rechte in der Konsole gesetzt:

```
sudo ./flow flow:core:setfilepermissions shelluser wwwuser wwwgroup
```

(Weitere Infos: http://docs.typo3.org/flow/TYPO3FlowDocumentation/TheDefinitiveGuide/PartII/Installation.html#file-permissions)

- `shelluser`
  Dies ist der User, mit dem man in der Konsole eingeloggt ist - kann mittels `whoami` herausgefunden werden

- `wwwuser`
  Der User, unter dem der Webserver läuft (steht in der Datei `httpd.conf`) - unter Mac OS X z.B. `_www`

- `wwwgroup`
  Die Gruppe, unter dem der Webserver läuft (steht in der Datei `httpd.conf`) - unter Mac OS X z.B. `_www`

typovision

Installation von TYPO3 Neos - VirtualHost

- Virtual Host Eintrag (z.B. Apache)

```
NameVirtualHost *:80
<VirtualHost *:80>
    DocumentRoot "/pfad/zum/webserver/TYPO3-Neos/"
    # Während der Entwicklung sollte die folgende Zeile
    # auskommentiert bleiben, denn dies stellt den Context auf
    # „Production“ - dies bedeutet: kein Logging, mit Caching, ...
    #Setenv FLOW_CONTEXT Production

    ServerName neos.demo
    AllowOverride FileInfo Options=MultiViews
</VirtualHost>
```

- Eintrag in /etc/hosts

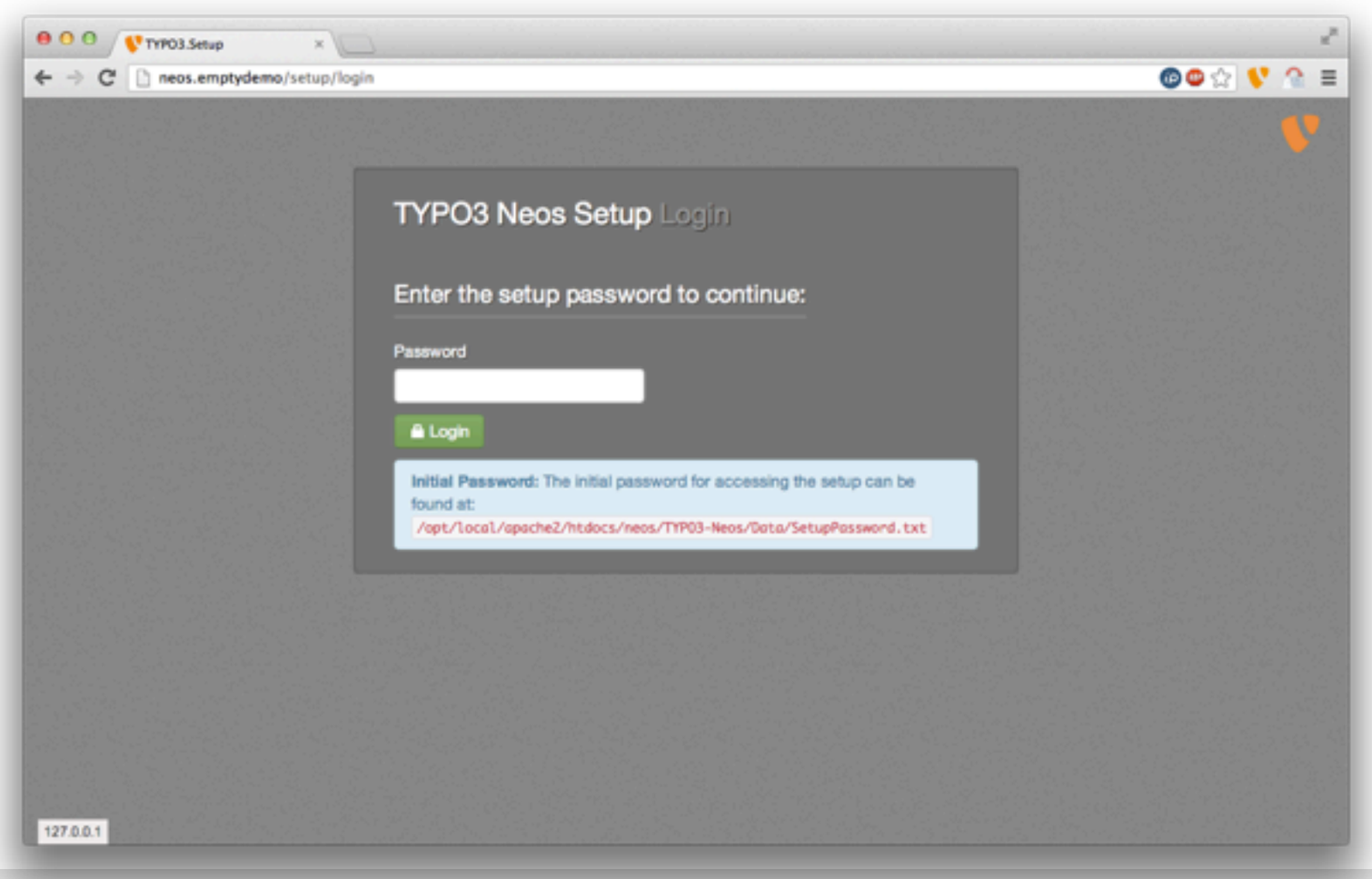
```
127.0.0.1 neos.demo
```

typovision

Installation von TYPO3 Neos - Setup

- Aufruf der Setup-Routine durch

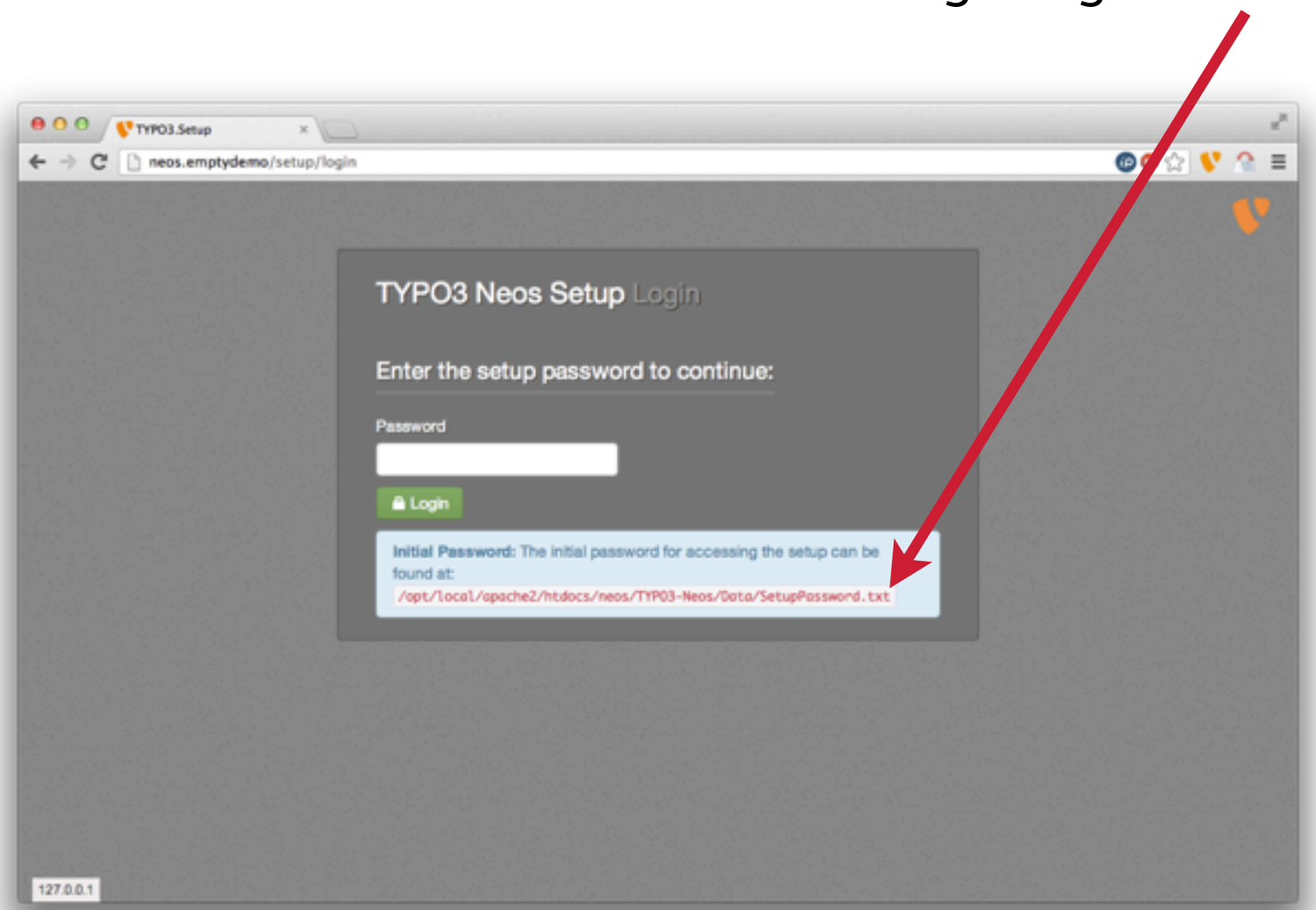
  `http://neos.demo/setup/`



typovision

Installation von TYPO3 Neos - Setup

- Das Password befindet sich unter angezeigten URL



Die Datei mit dem Password wird anschließend wieder gelöscht. Hat man das Password vergessen, muss man die Datei

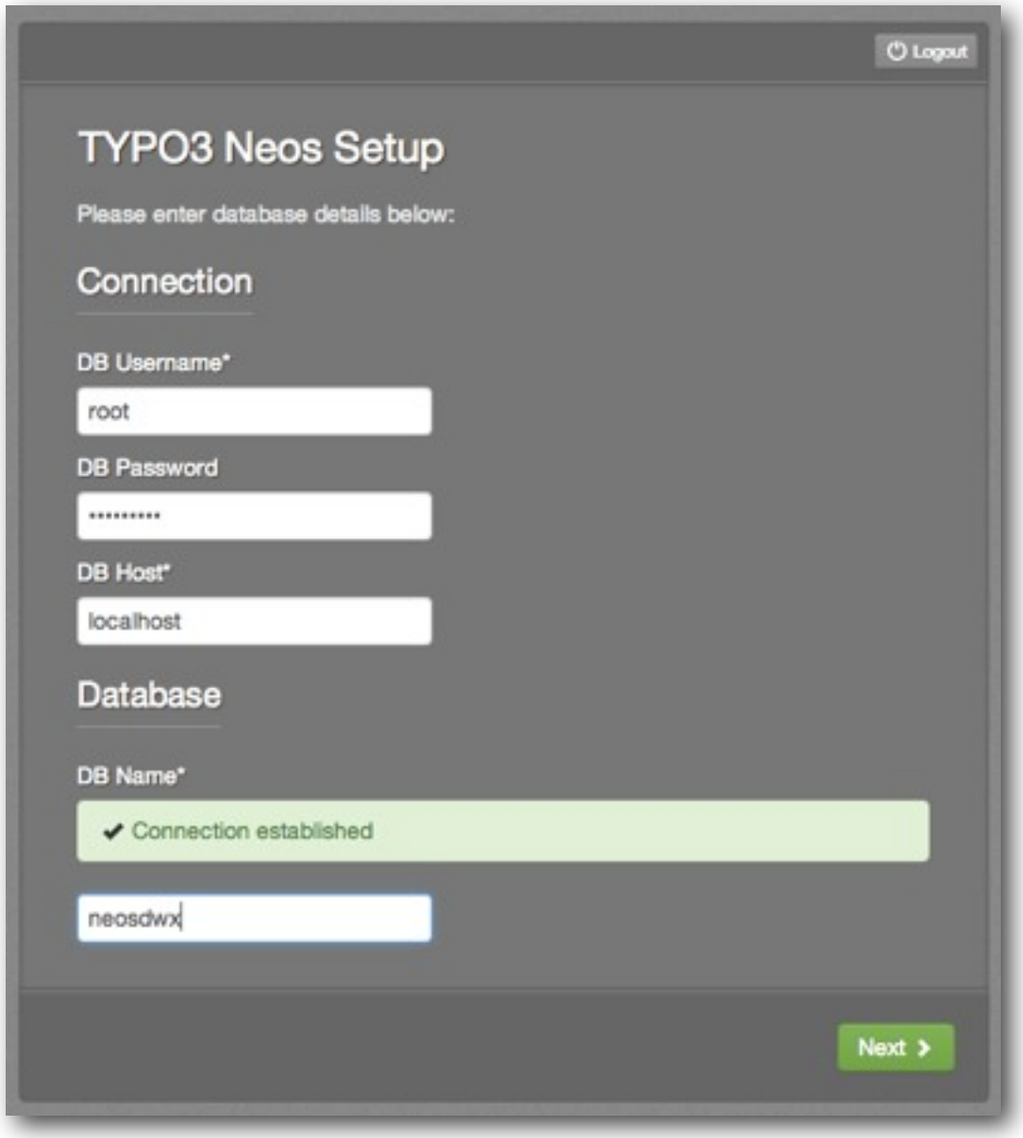
```
/pfad/zum/webserver/
TYPO3-Neos/Data/
Persistent/
FileBasedSimpleKeyService
/SetupKey
```

löschen und das Setup erneut aufrufen.

typovision

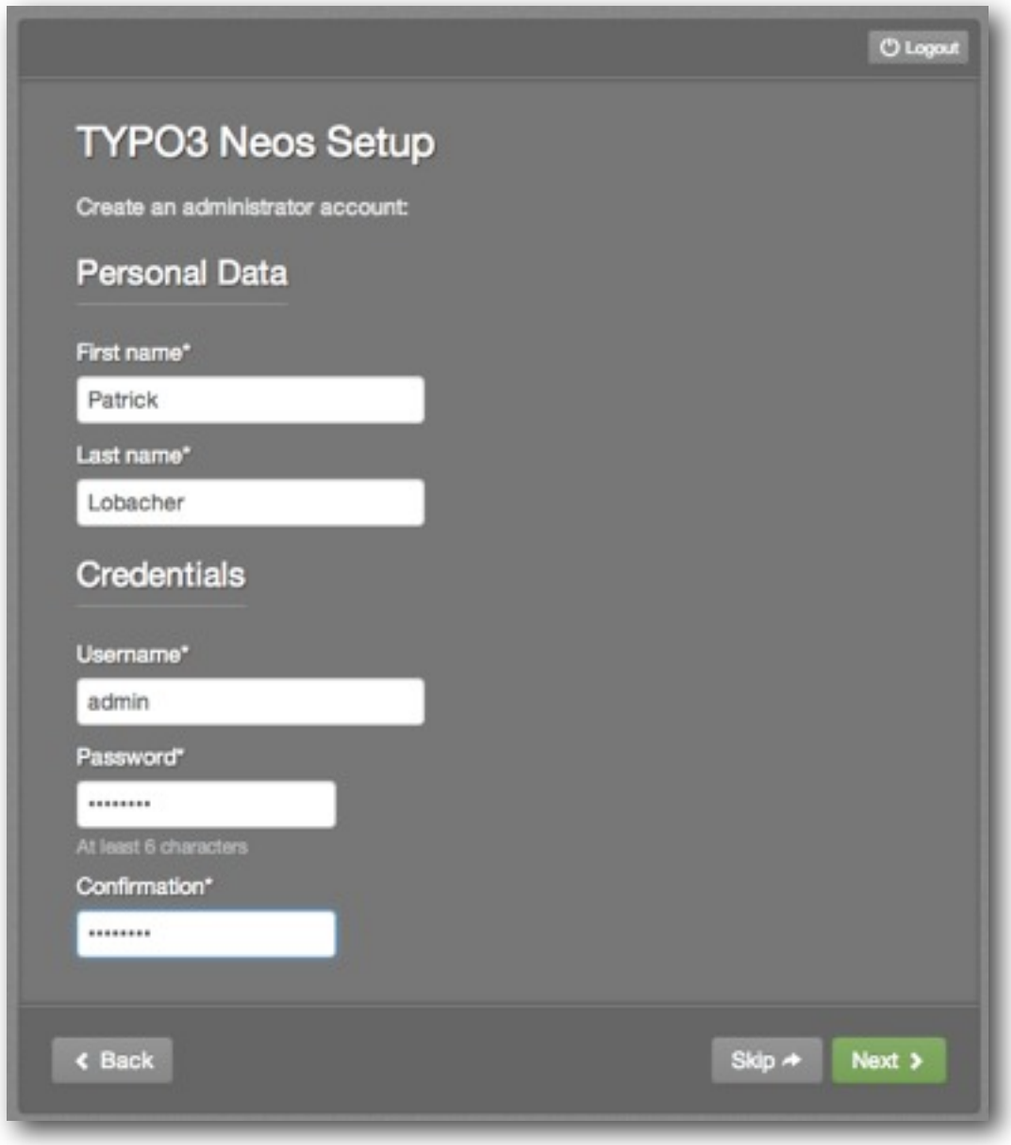
Installation von TYPO3 Neos - Setup

- Datenbank-Setup
- Voreinstellung ist MySQL
- Änderung des Drivers durch Editieren der Datei:
  `Configuration/Settings.yaml`
- Sollte als DB-Host `127.0.0.1` nicht funktionieren, so kann man probieren stattdessen `localhost` dort einzutragen



typovision

Installation von TYPO3 Neos - Setup



- Anlegen eines Administrators
- Weitere User können später in der Userverwaltung angelegt werden
- Zusätzliche User-Daten können ebenfalls später in der Userverwaltung zugefügt werden
- Manuell kann man einen User ebenfalls anlegen - in der Konsole:

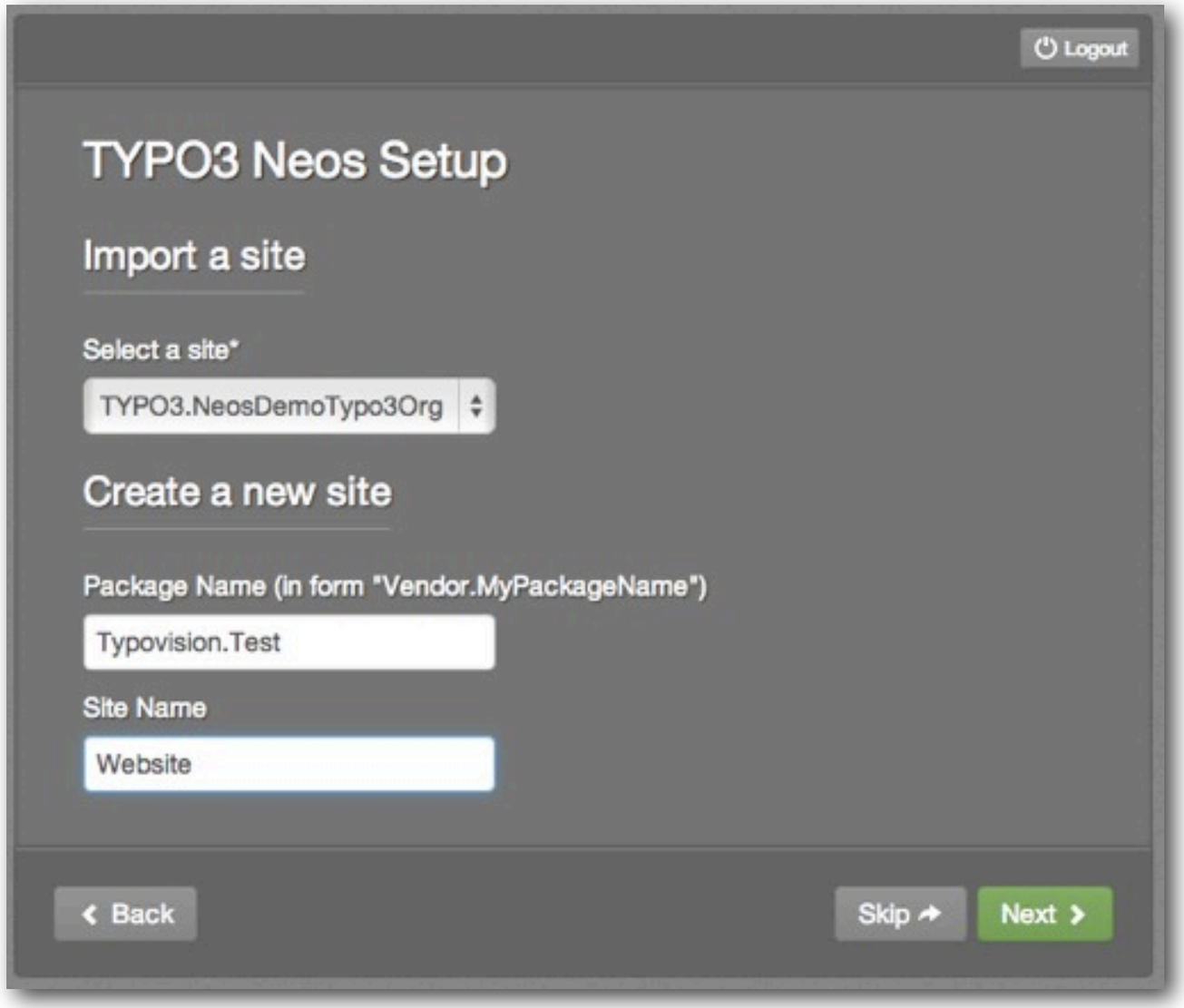
```
./flow typo3.neos:user:create username password firstname lastname
```

typovision

Installation von TYPO3 Neos - Setup

- Nun kann man entweder eine Demo-Site importieren (empfohlen)

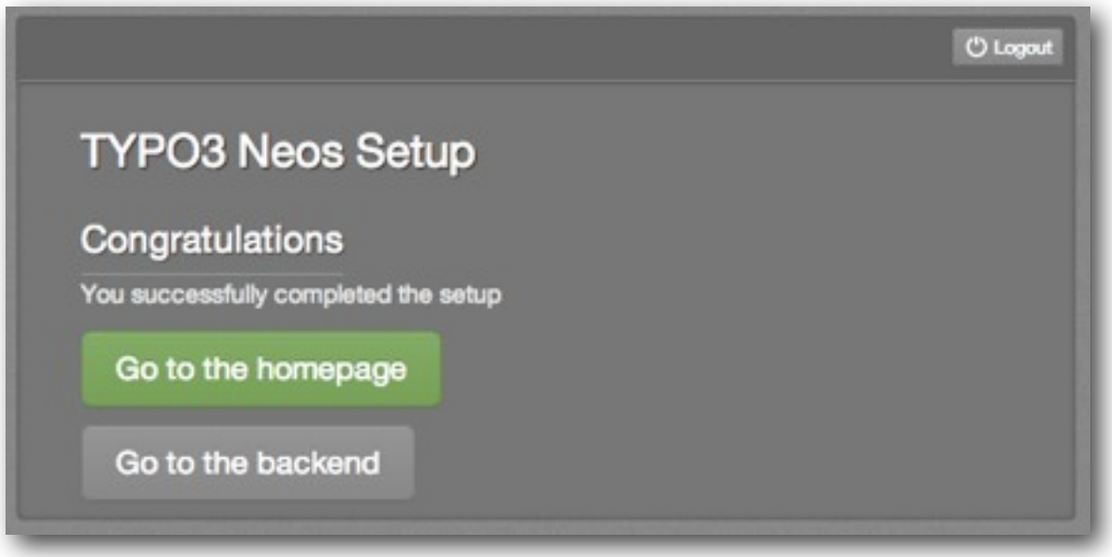
- Oder mit einer leeren Website starten
- Sobald in diese beide Formular-Felder etwas eingetragen ist, wird eine neue Site angelegt



typovision

Installation von TYPO3 Neos - Setup

- Wenn die Installation erfolgreich durchgelaufen ist, erscheint der entsprechende Hinweis

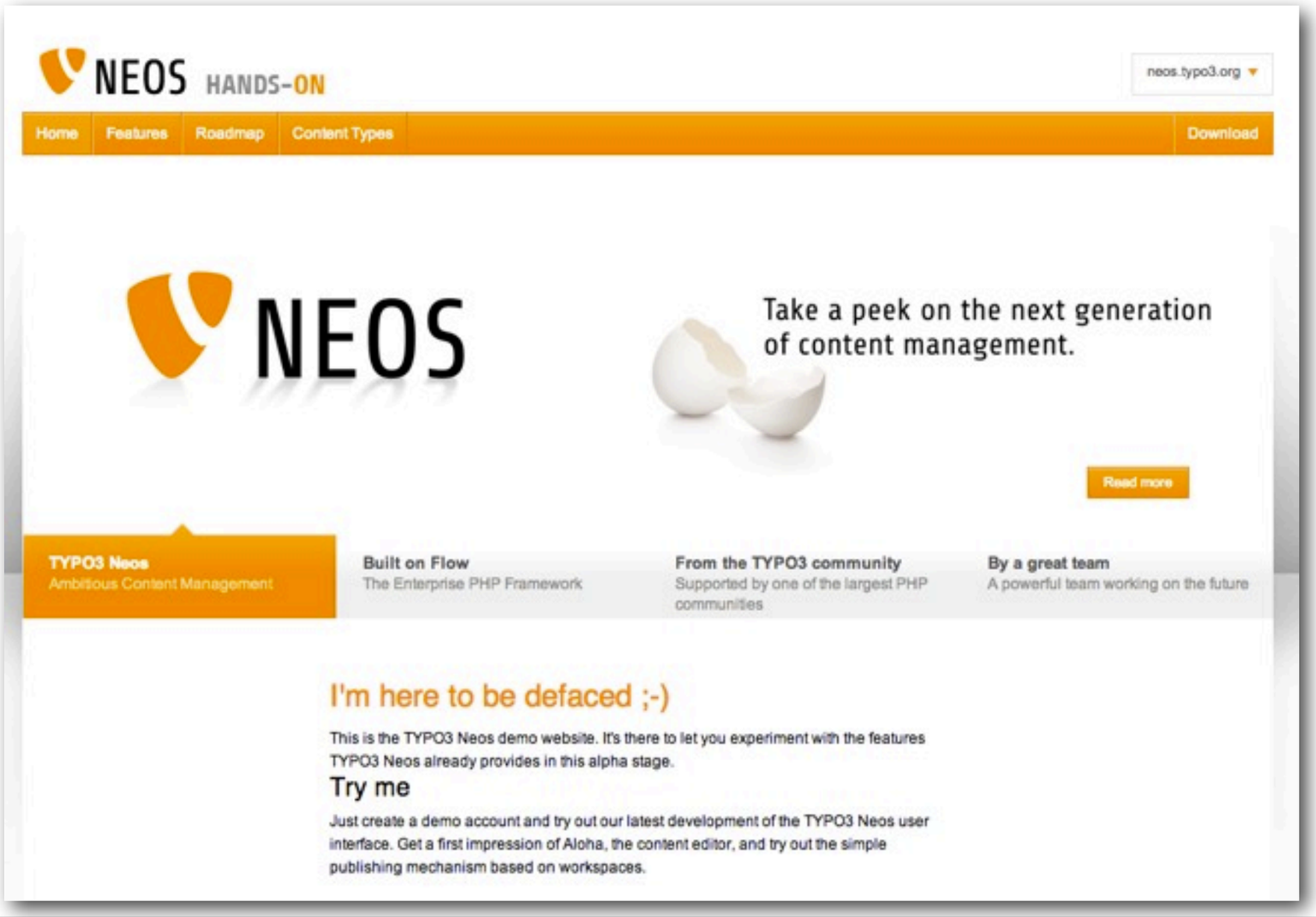


- Ist dies nicht der Fall, dann bitte ich Euch mir einen Screenshot des Fehlers und Eure Systemkonfiguration an folgende Email zu schicken, damit ich die Lösung hier in einem FAQ einfügen kann:

```
patrick.lobacher [AT] typovision.de
```

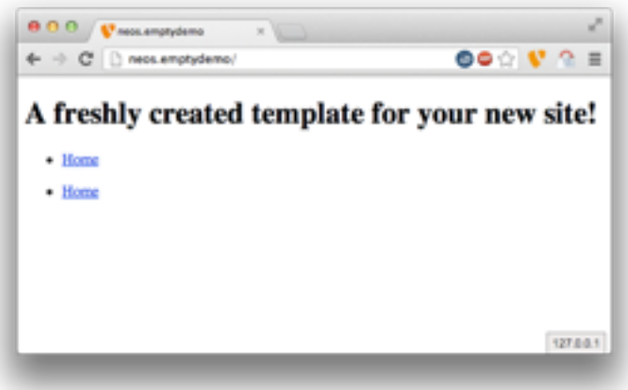
typovision

Installation von TYPO3 Neos - Setup



So wie links sieht das Frontend aus, wenn man die Demo-Site TYPO3.NeosDemoTypo3Org importiert hat.

Wenn man eine leere Site angelegt hat, erhält man den folgenden Screen:



typovision

Installation von TYPO3 Neos - Setup

Zugang zur Installation:

- Frontend
  `http://neos.demo/`

- Backend
  `http://neos.demo/neos/`

- Setup
  `http://neos.demo/setup/`

typovision

Installation auf einem

domainFACTORY Server

typovision

Installation auf einem domainFACTORY Server

Auf domainFatory Server muss man einer leicht abweichenden Installationsanleitung folgen

- Credits gehen an: **Christian Schwerdt** (die Medienagenten oHG)
- Die Anleitung findet sich auch hier:

  https://github.com/cschwerdt/TYPO3-NEOS-Install

typovision

Installation auf einem domainFACTORY Server

- Composer laden

```
curl -s https://getcomposer.org/installer | /usr/local/bin/php5-53STABLE-CLI
```

- TYPO3 Neos alpha 5 laden:

```
/usr/local/bin/php5-53LATEST-CLI composer.phar create-project --dev --
stability alpha typo3/neos-base-distribution TYPO3-Neos-1.0-alpha5
```

- In den domainFACTORY-Settings muss nun die Domain so konfiguriert werden, dass sie auf `/Web` zeigt

Installation auf einem domainFACTORY Server

- PHP-Pfad im Flow-CLI ändern

```
cd TYPO3-Neos-1.0-alpha5
vi flow
```

Die erste Zeile muss nun von

```
#!/usr/bin/env php
```

in die folgende geändert werden:

```
#!/usr/local/bin/php5-53STABLE-CLI
```

Anschließend abspeichern mit

```
:!wq
```

typovision

Installation auf einem domainFACTORY Server

- Settings.yaml in Configuration/ anpassen

```
cd Configuration/
cp Settings.yaml.example Settings.yaml
vi Settings.yaml

//set database host
db: 'mysql5.<yourdomain.com>

//uncomment core: & phpBinaryPathAndFilename
//and change phpBinaryPathAndFilename to:
core:
    phpBinaryPathAndFilename: '/usr/local/bin/php5-53STABLE-CLI'
```

Nun Abspeichern via:

```
:wq!
```

typovision

Installation auf einem domainFACTORY Server

- Settings.yaml in Development/ anpassen

```
cd Development/
cp Settings.yaml.example Settings.yaml
vi Settings.yaml


//set dbname, dbuser, dbpassword:
dbname: '<dbname>'
user: '<dbuser>'
password: '<password>'
```

Nun Abspeichern via:

```
:wq!
```

typovision

Installation auf einem domainFACTORY Server

- Flow testen

  `./flow help`

- Datenbank migrieren

  `./flow doctrine:migrate`

- Site kickstarten

  `./flow site:kickstart Your.Demopage Your.Demopage`

- Sites auflisten

  `:./flow site:list`

typovision

Installation auf einem domainFACTORY Server

- Neos Backend User anlegen

  ```
  ./flow user:create <username> <password> <firstname> <lastname>
  ```

- Admin Userrolle zum User zufügen

  ```
  ./flow user:addrole <username> Administrator
  ```

- Fertig :-)

typovision

Upgrade

von TYPO3 Neos

typovision

Upgrade von TYPO3 Neos

Wenn bereits ein installiertes Neos existiert, kann dies ganz bequem mittels Composer auf die neuste Version aktualisiert werden:

```
cd /pfad/zum/webserver/
composer require "typo3/neos:1.0.0-alpha5"
composer require "typo3/neosdemotypo3org:1.0.0-alpha5"
composer require "typo3/sitekickstarter:1.0.0-alpha5"
# Cache loeschen!
./flow flow:cache:flush --force
```

Gegebenenfalls muss man **`php composer.phar`** verwenden!

typovision

Verwenden des Development Master

Will man allerdings ein absolut aktuelles TYPO3 Neos verwenden, so kann man direkt auf den Development Master „umschalten“:

```
cd /pfad/zum/webserver/
composer update
# Cache loeschen!
./flow flow:cache:flush --force
```

Gegebenenfalls muss man **`php composer.phar`** verwenden!

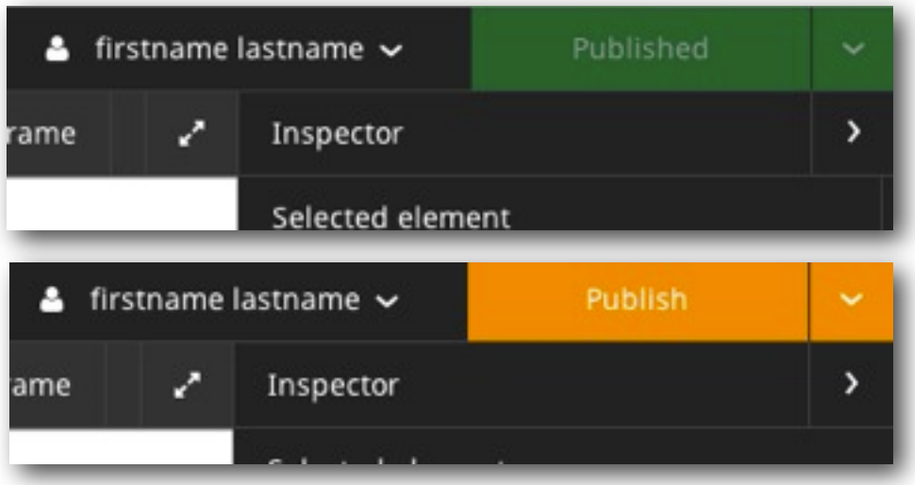
typovision

Release

Notes

Release Notes TYPO3 Neos alpha5 (06.08.2013)

- Live-Validierung von Inspektor-Feldern
- Publishing-State wird nun angezeigt (grün, wenn alles publiziert ist und orange, wenn publiziert werden muss)
- Zugriffsberechtigung für Module wurde verbessert
- Fehlerbehebungen:
  - Multi-Site funktioniert nun
  - Redirect zu nicht vorhandener Seite behoben
- Kosmetische Änderungen



Features

von TYPO3 Neos

typovision

Features von TYPO3 Neos

- Maximal erweiterbares Enterprise CMF - Content Management Framework
- Basis ist das stabile TYPO3 Flow Framework, welches bereits in Version 2.0 vorliegt
- Einfacher und flexibler Content Export/Import
- Multi-Domain Support
- Multi-Language Support (Version 1.1)
- Modernste Technologie und Paradigmen unter der Haube (DDD, AOP, RequireJS, EmberJS, ...)
- Intuitives Benutzer-Interface
- Wireframe Mode - Content-Editing ohne Template
- Webbasiertes Package Management

typovision

Features von TYPO3 Neos

- In-place Content-Editing
- Eigene Content-Elemente (inkl. In-place Content-Editing) leicht möglich
- Integration von TYPO3 Flow Applikationen als Plugins
- TypoScript2 / EEL / FlowQuery
- Workspaces
- Cloud-Ready
- Custom Single-Sign-On / Custom Authentification
- Audit Logging
- Soap/REST/JSON out-of-the-box
- TYPO3 Surf für das automatische Deployment (Integration mit CI Server wie Jenkins)

Aufbau der Neos

Oberfläche

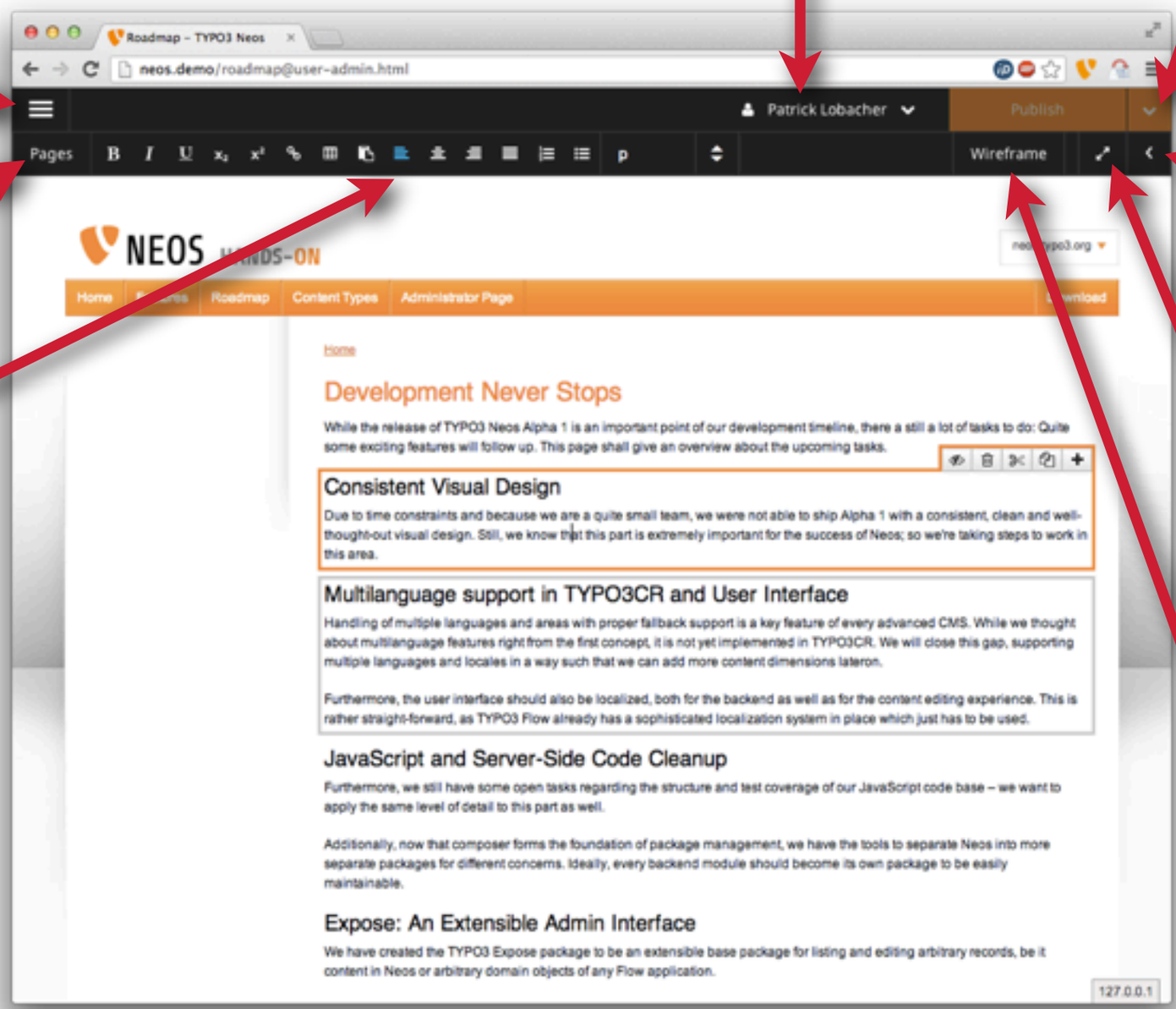
Aufbau der Admin-Oberfläche

Menu
z.B. Content, Workspaces, ...

Page-Browser (Seitenbaum)

Editor-Auszeichnungen
z.B. Ausrichtung, Link, Bold, ...

User-Administration
Logout und Settings



Publish
Publizieren, bei Klick auf Pfeil Möglichkeit zu „Auto-Publish“

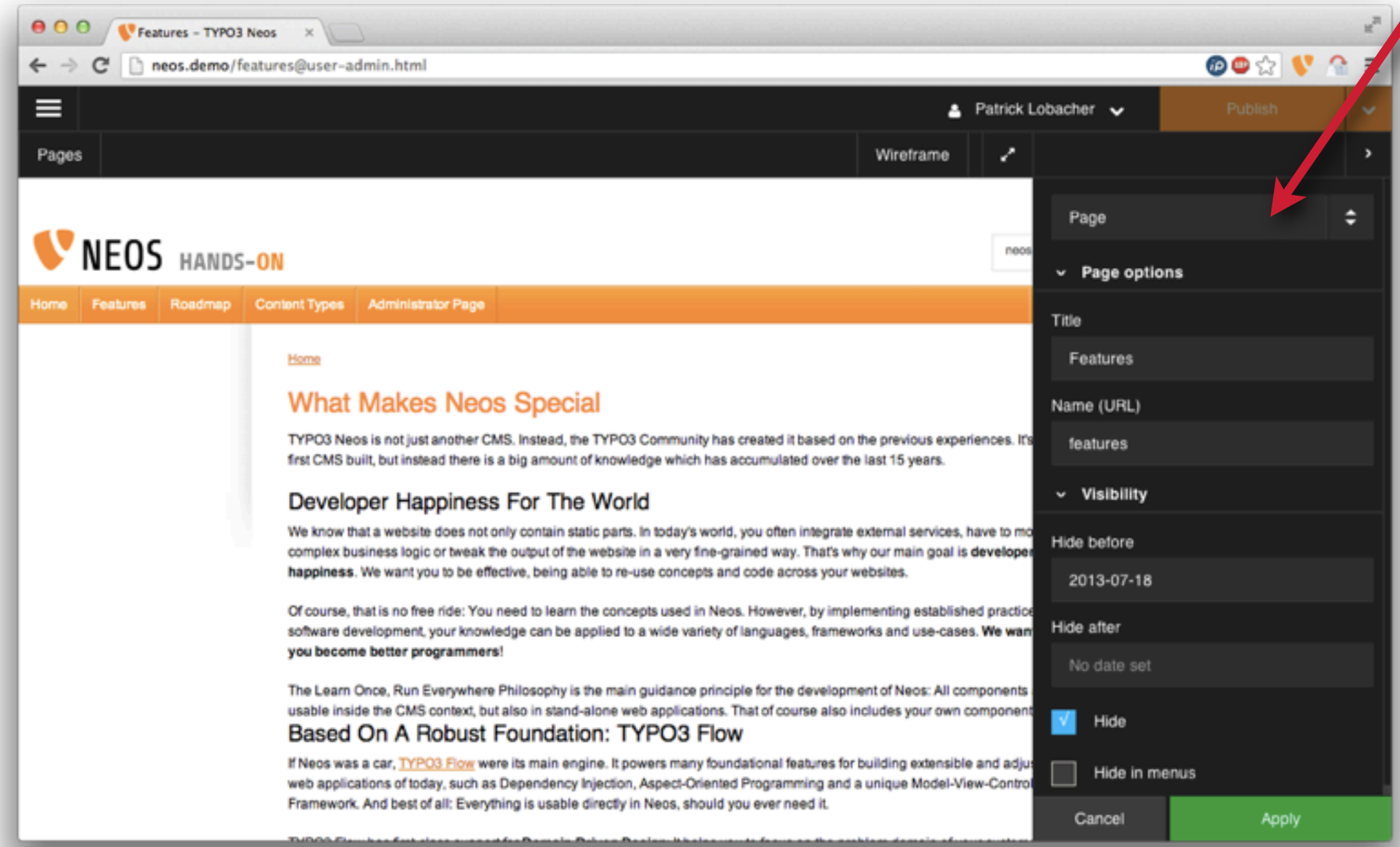
Properties
Content-bezogene Eigenschaften, z.B. Seiten-eigenschaften, o.ä.

Preview-Mode
Anzeige, wie die Seite live aussehen würde

Wireframe-Mode
Anzeige ohne Design

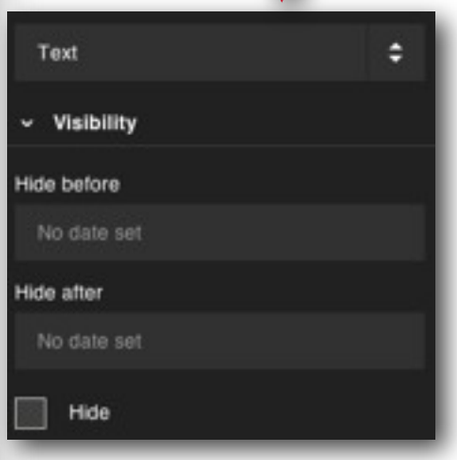
typovision

Eigenschaftspanel



Content-Type
z.B. Headline, Content-Collection, Page, ...

Properties
Content-bezogene Eigenschaften, z.B. Seiten-eigenschaften, o.ä.



Funktionen-Menü

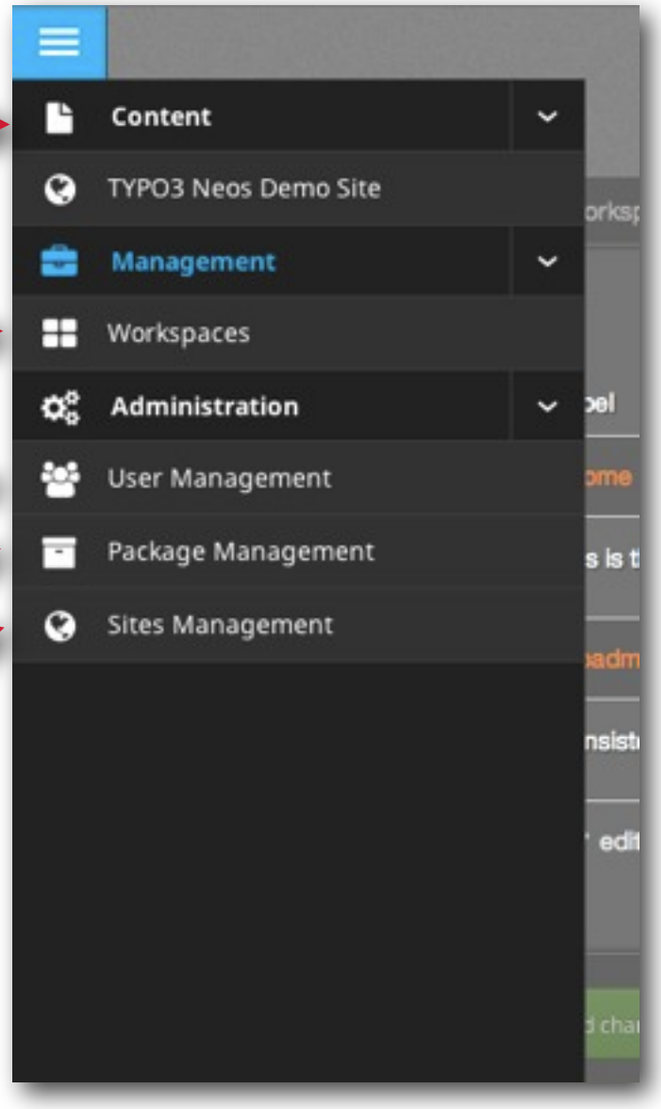
Content
Hier können die verschiedenen Sites umgeschaltet werden

Workspaces
Verwaltung der Workspaces

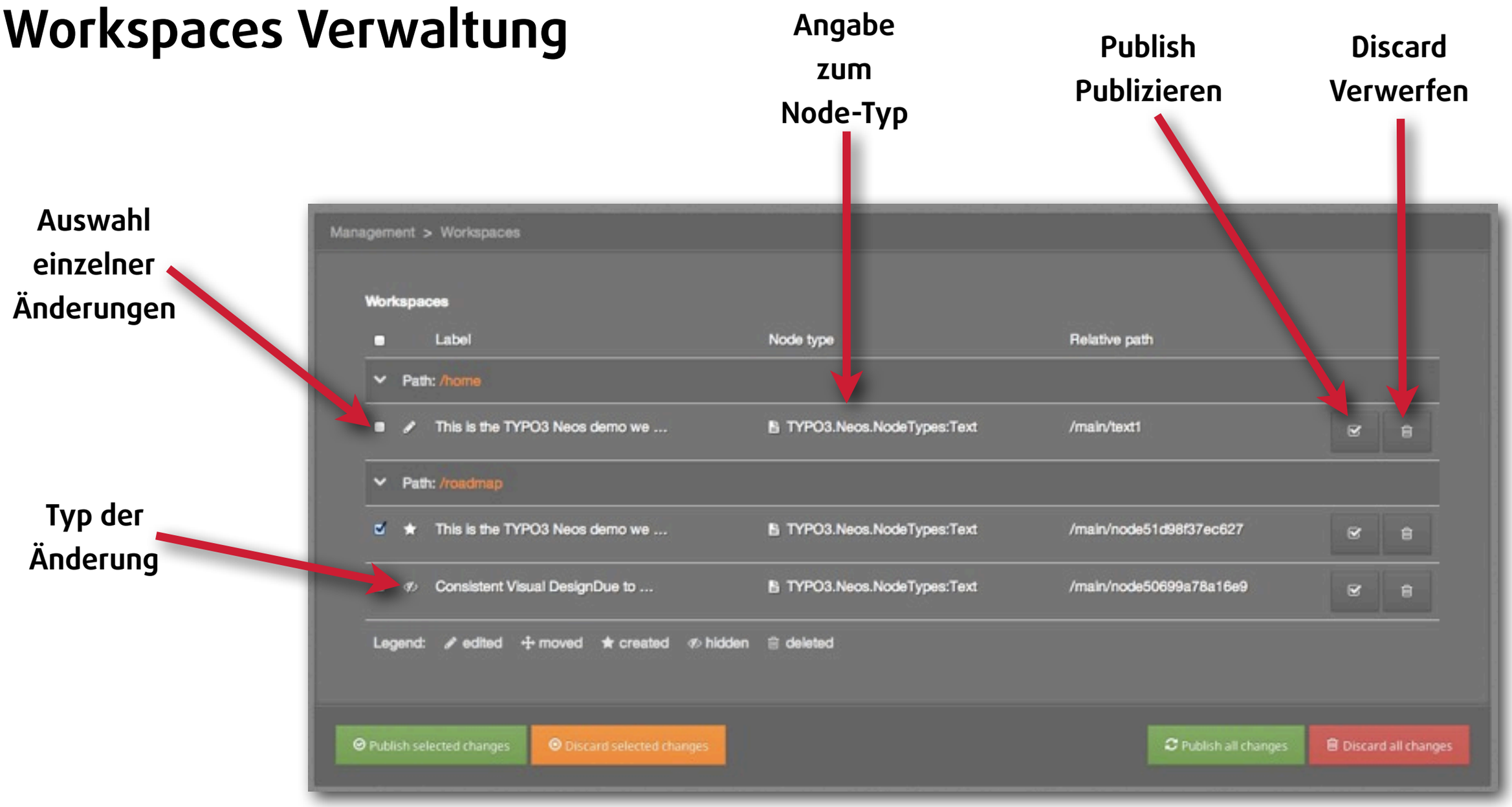
User Management
User-Verwaltung

Package Management
Package Verwaltung

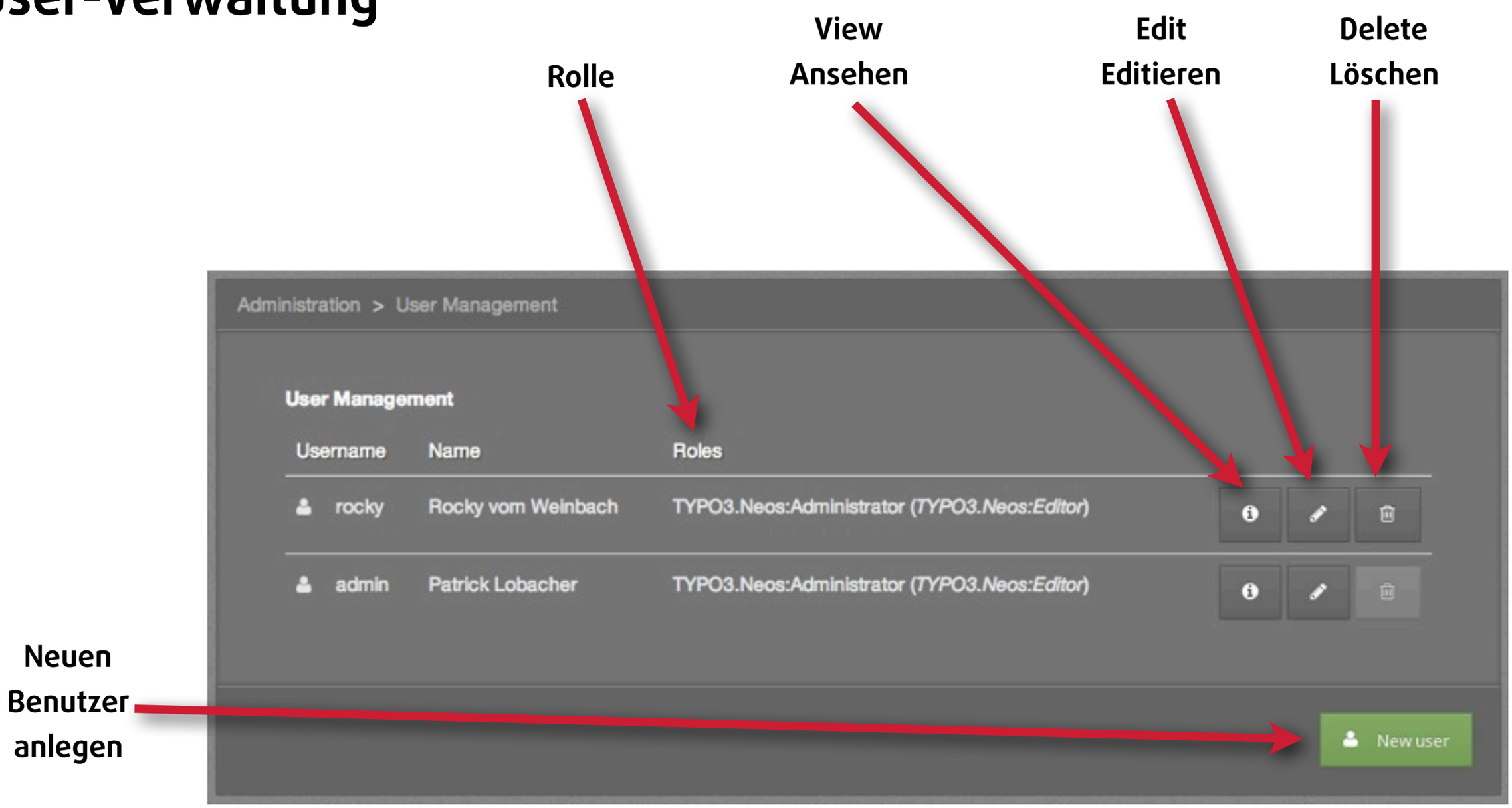
Sites Management
Verwaltung der Websites



Workspaces Verwaltung

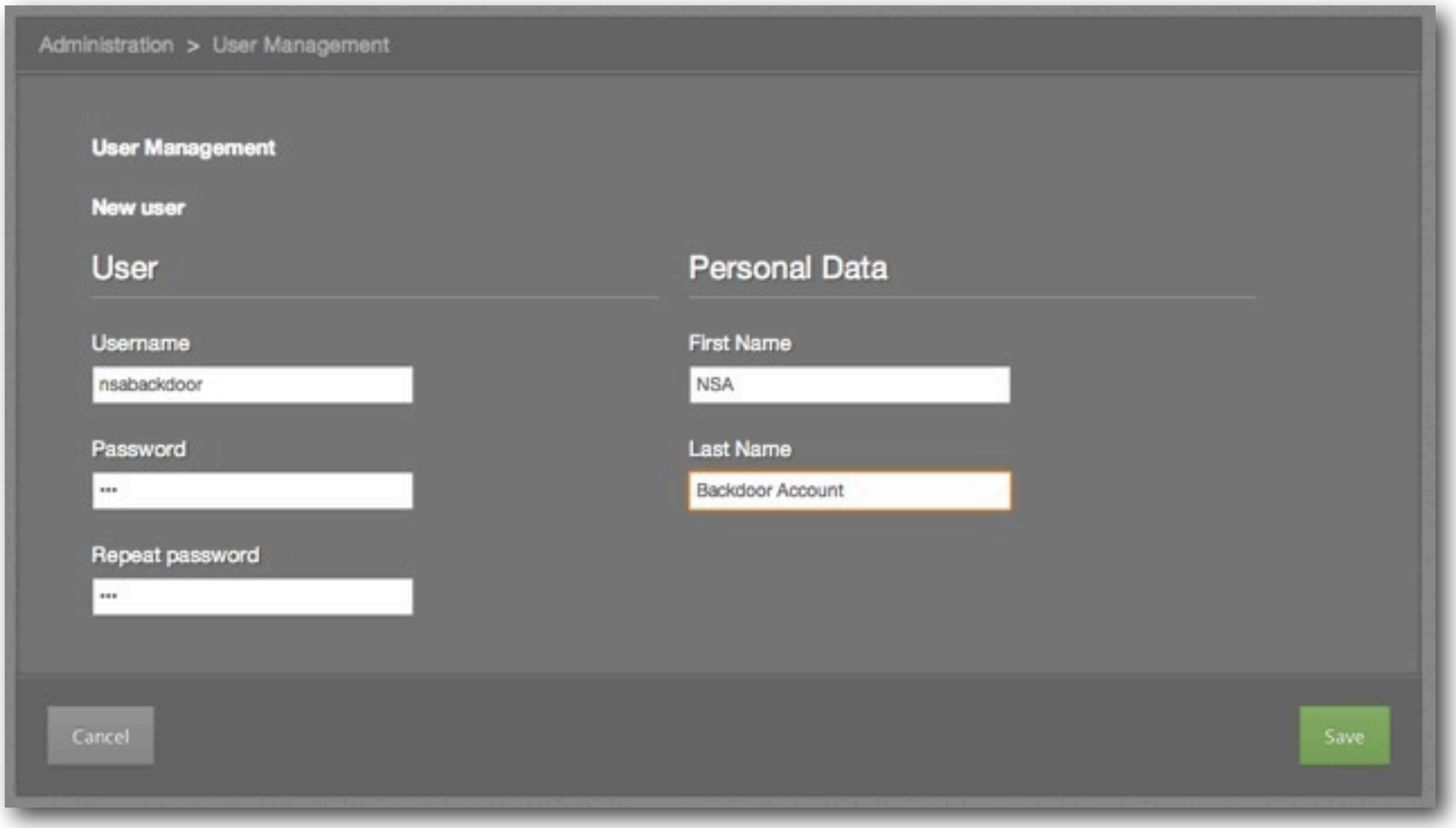


User-Verwaltung



User-Verwaltung - Neuer User

Einen neuen User anlegen - es muss lediglich ein Username, das Passwort und die persönlichen Daten angegeben werden

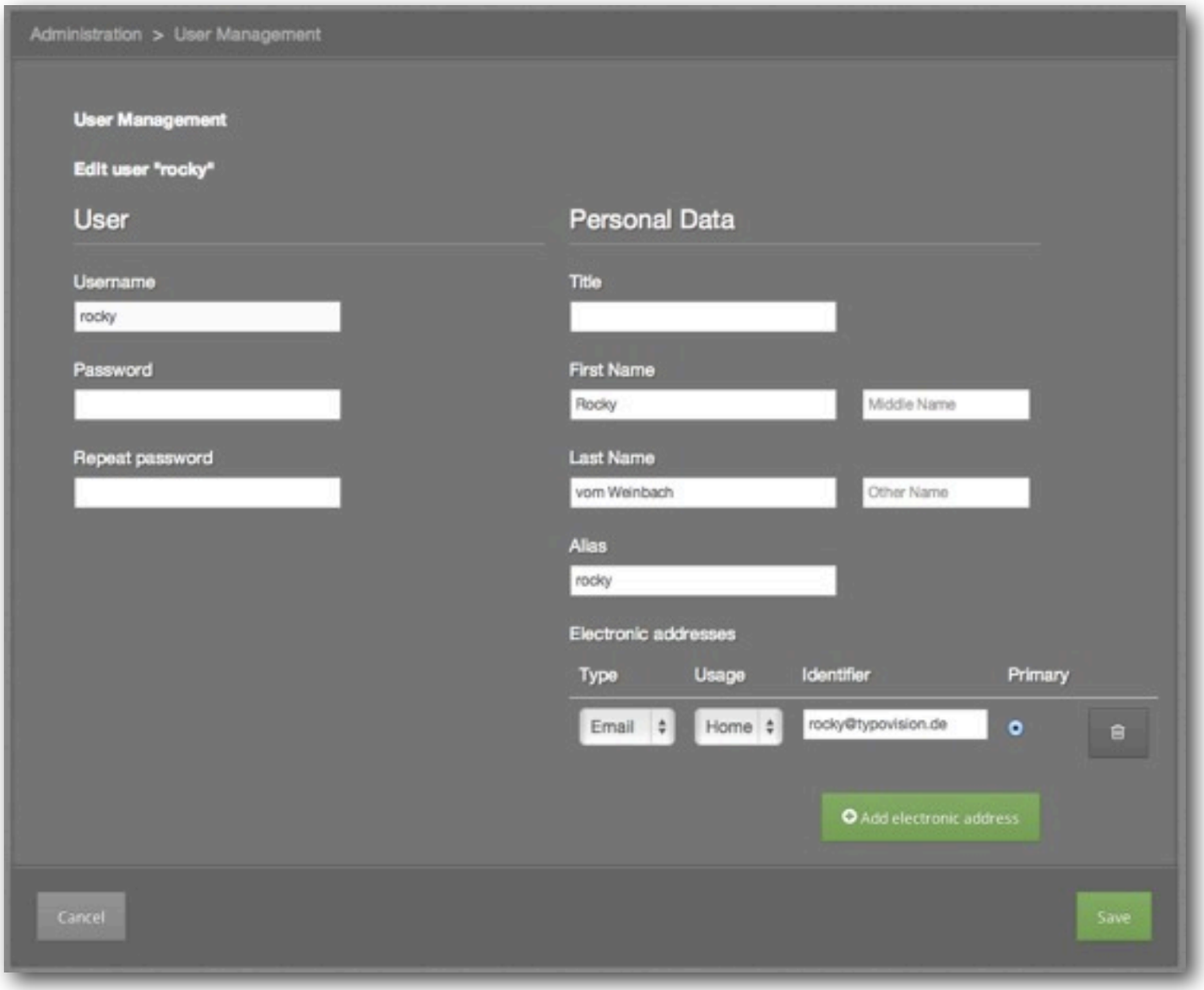


User-Verwaltung - Editieren

Hier können die Userdaten editiert werden und zudem auch weitere Daten angeben werden - z.B. Title

Zusätzlich kann man hier beliebig viele „Electronic Addresses“ anlegen - also z.B. Email-Adressen oder Skype

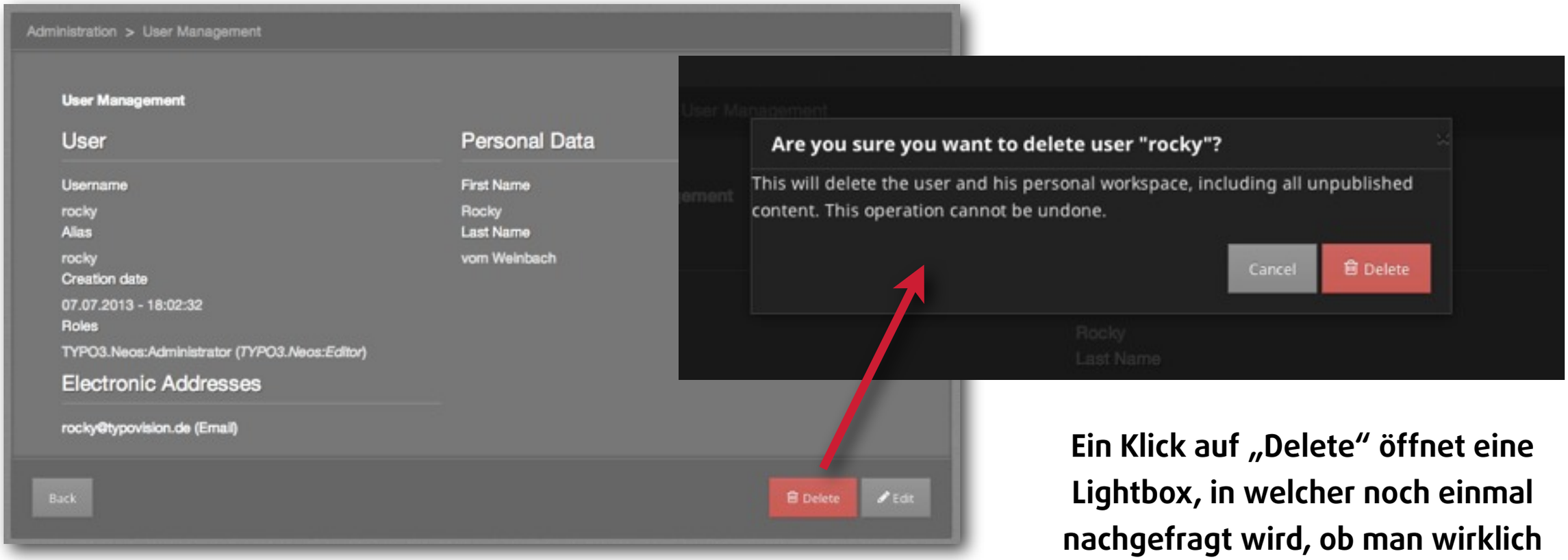
Durch die Angabe „Primary“ wählt man die Haupt-Adresse aus



typovision

User-Verwaltung - Info

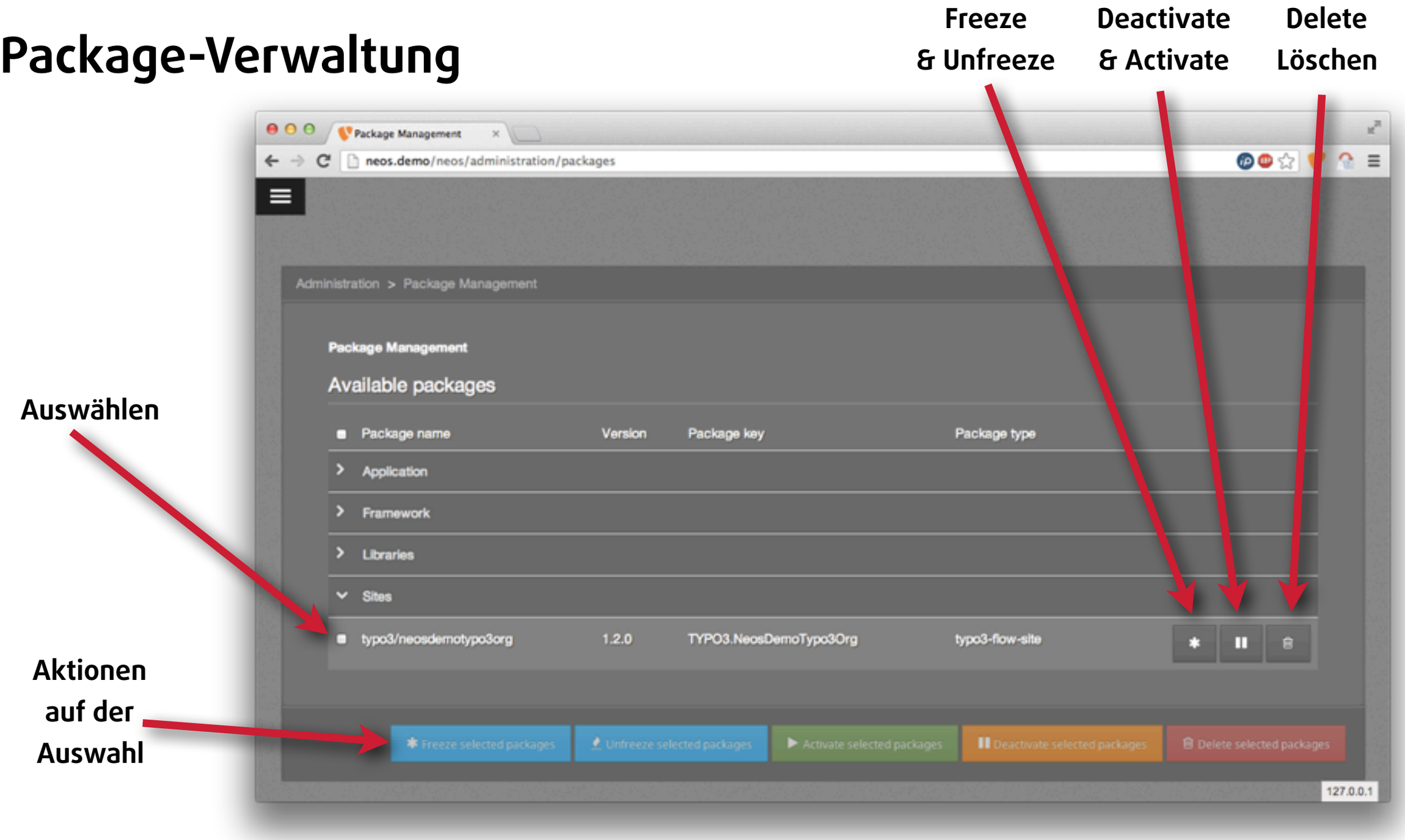
Die Funktion „Info“ zeigt eine Übersicht über alle eingegebenen Daten an



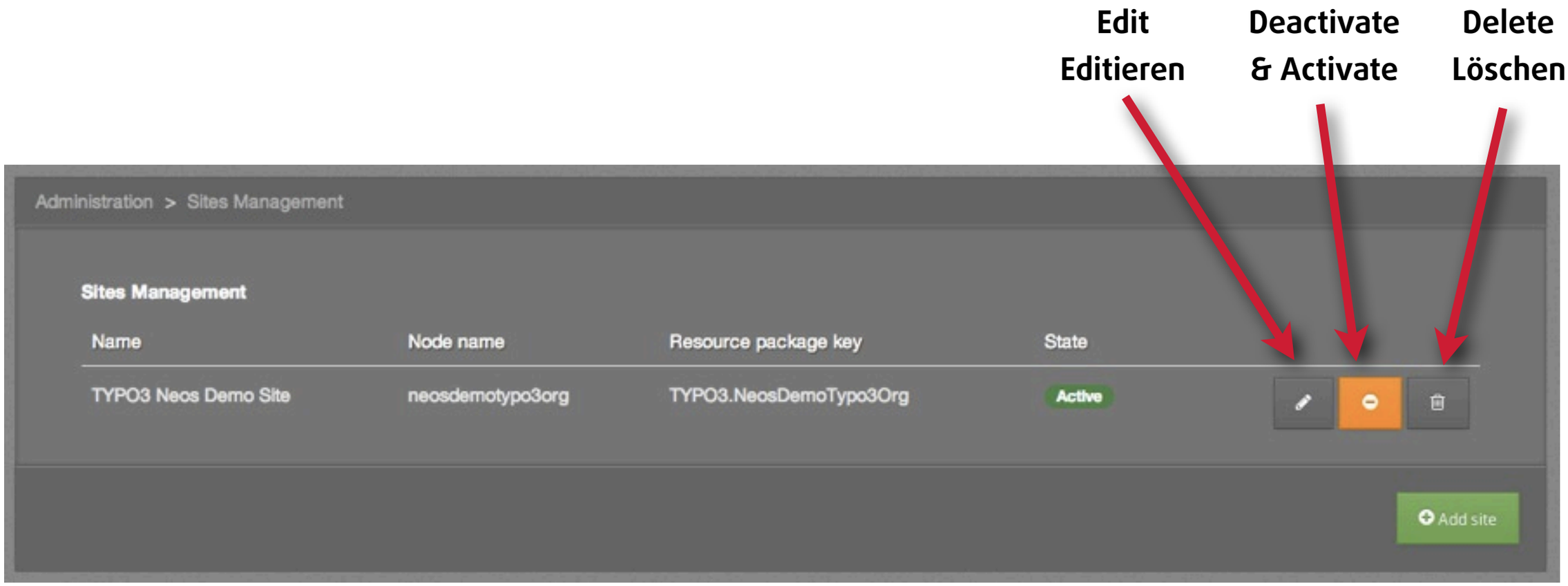
Ein Klick auf „Delete“ öffnet eine Lightbox, in welcher noch einmal nachgefragt wird, ob man wirklich löschen möchte

typovision

Package-Verwaltung

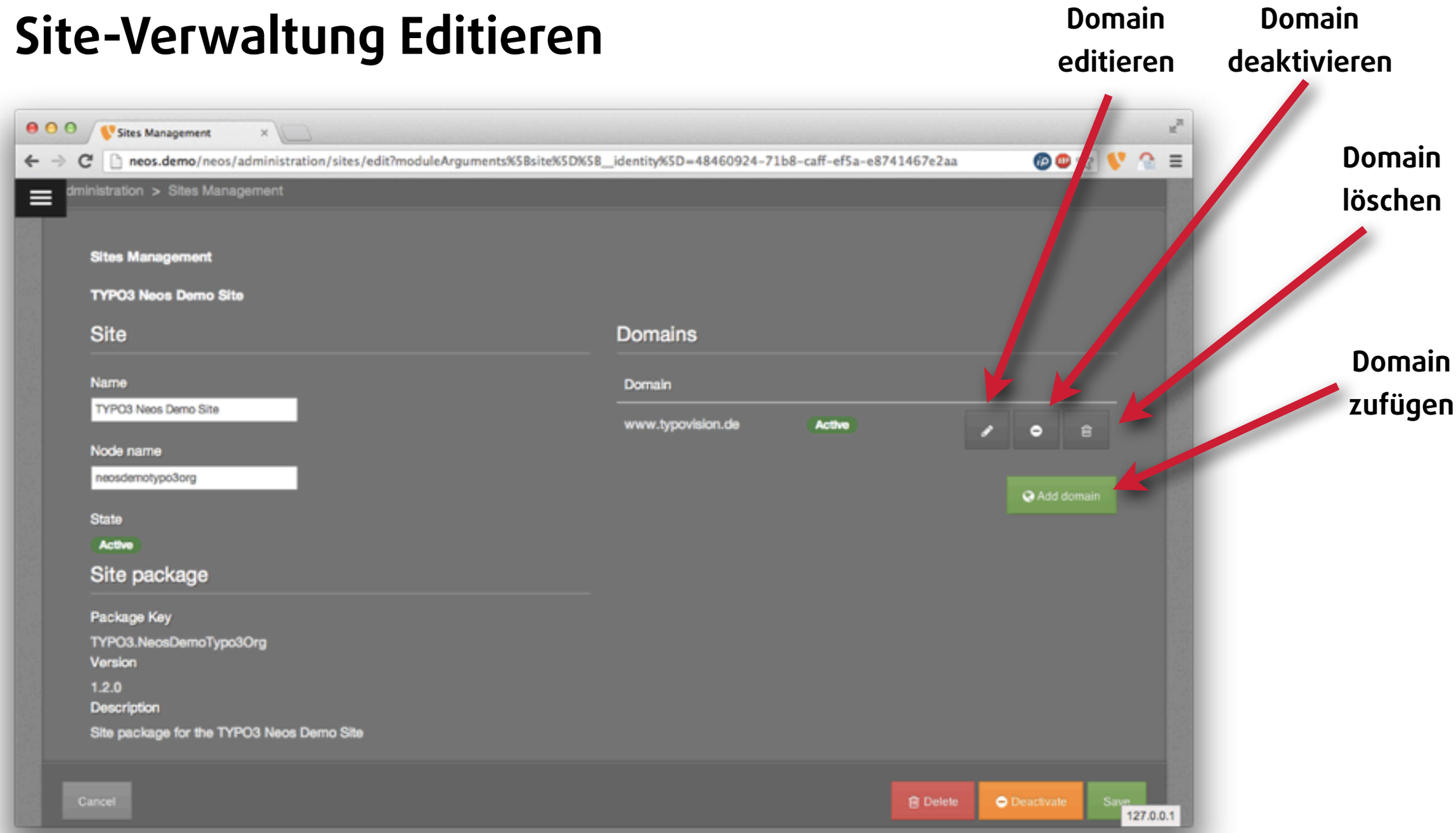


Site-Verwaltung Übersicht

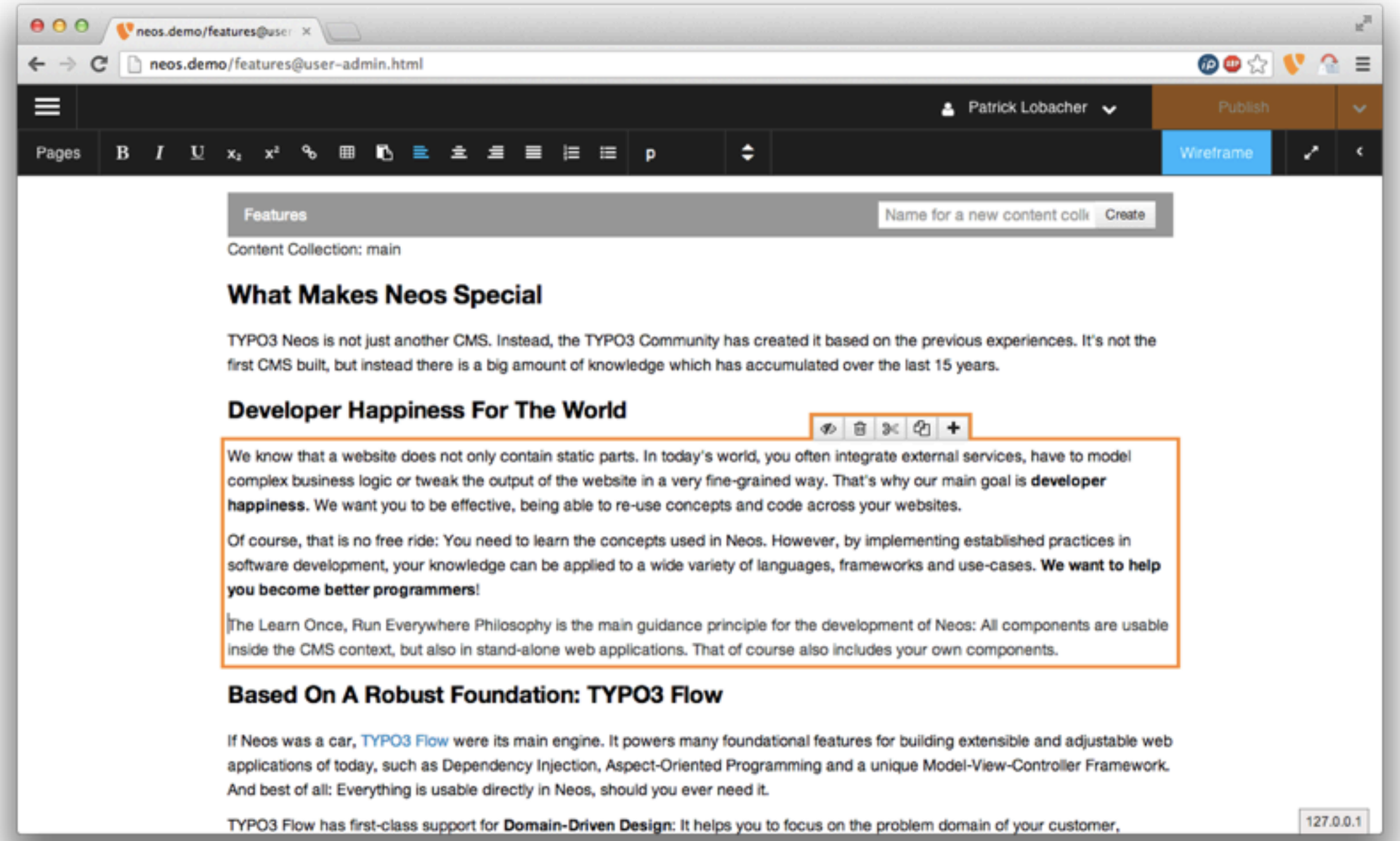


typovision

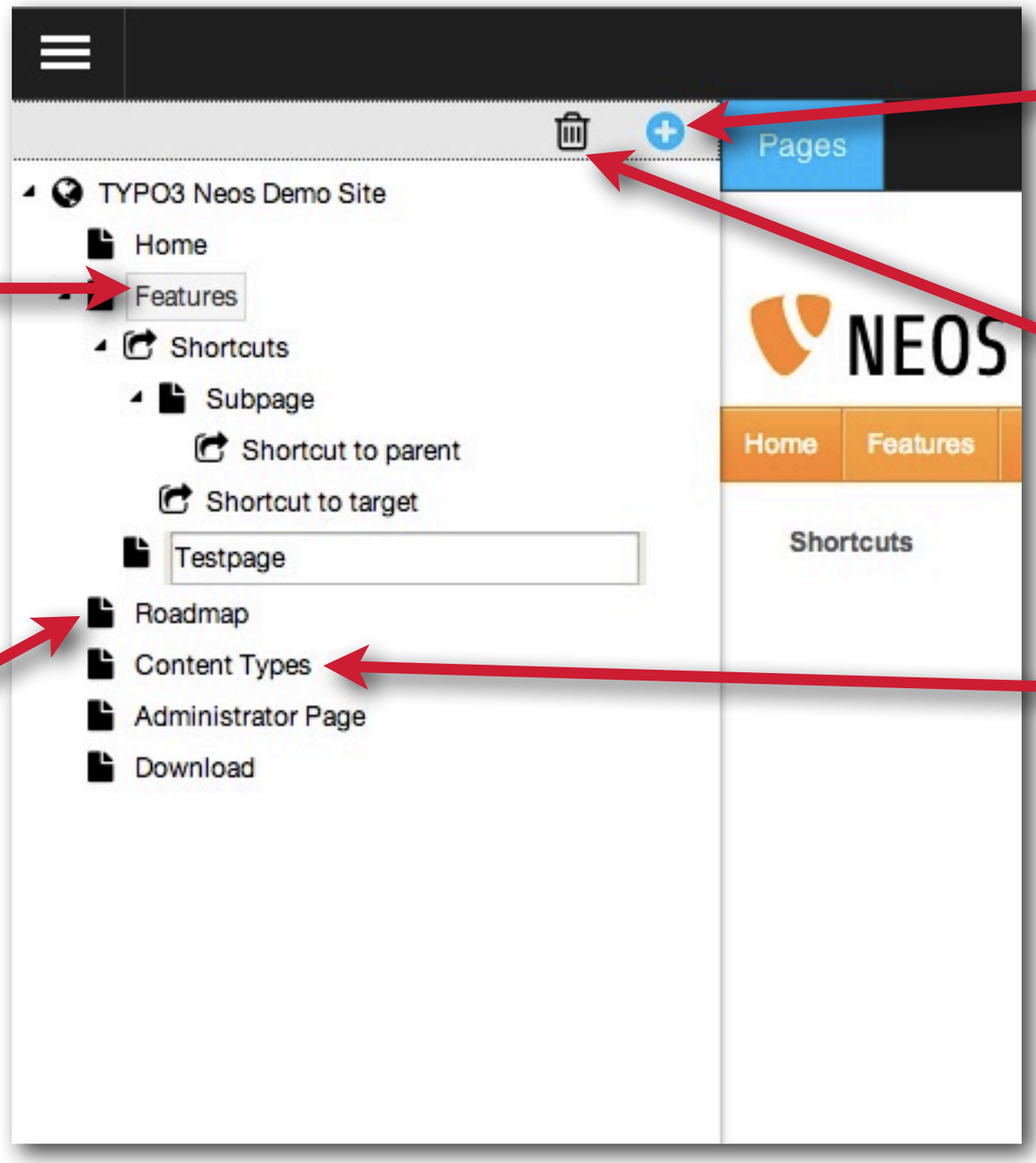
Site-Verwaltung Editieren



Wireframe-Mode



Seitenbaum



Klick auf das Plus-Symbo
legt neue Seite an

Doppelklicken um den
Page-Title zu editieren

Löschen einer Seite

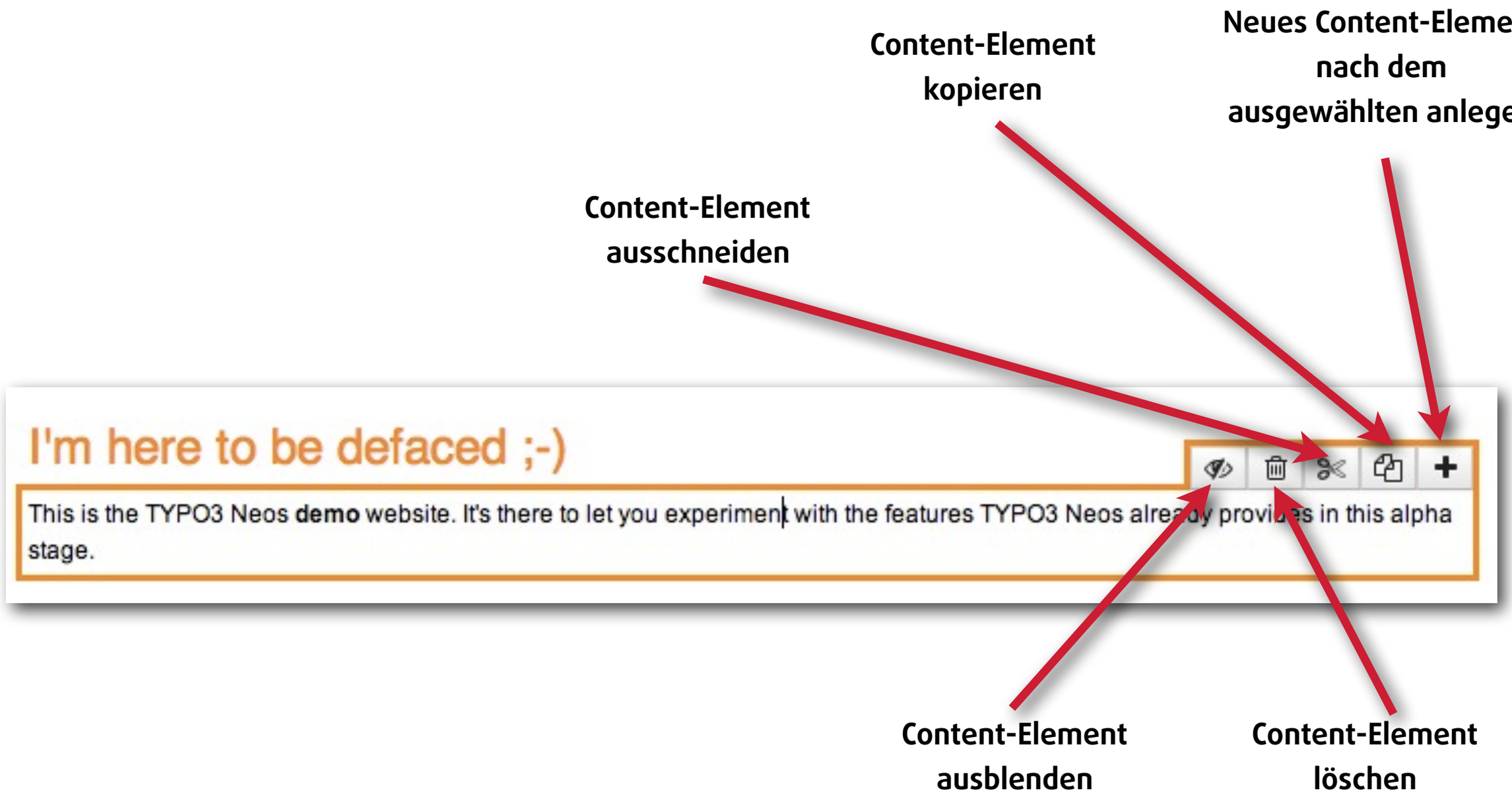
Drag&Drop um Seite
zu verschieben

Klick auf Page-Title um
Seite in der Preview
rechts zu sehen

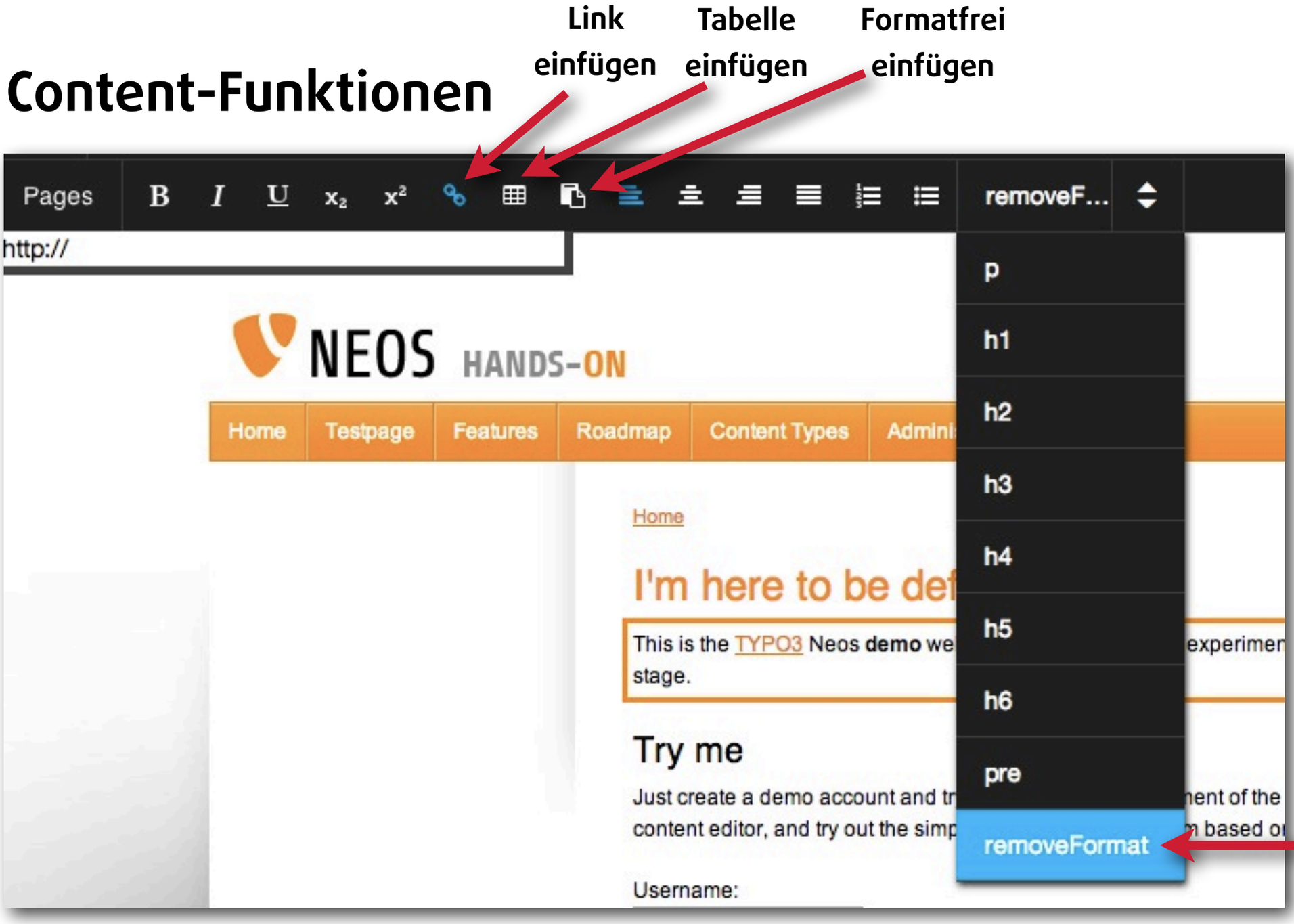
Durch einen Bug muss
man (momentan) manuell
aktualisieren um
Änderungen zu
übernehmen!

typovision

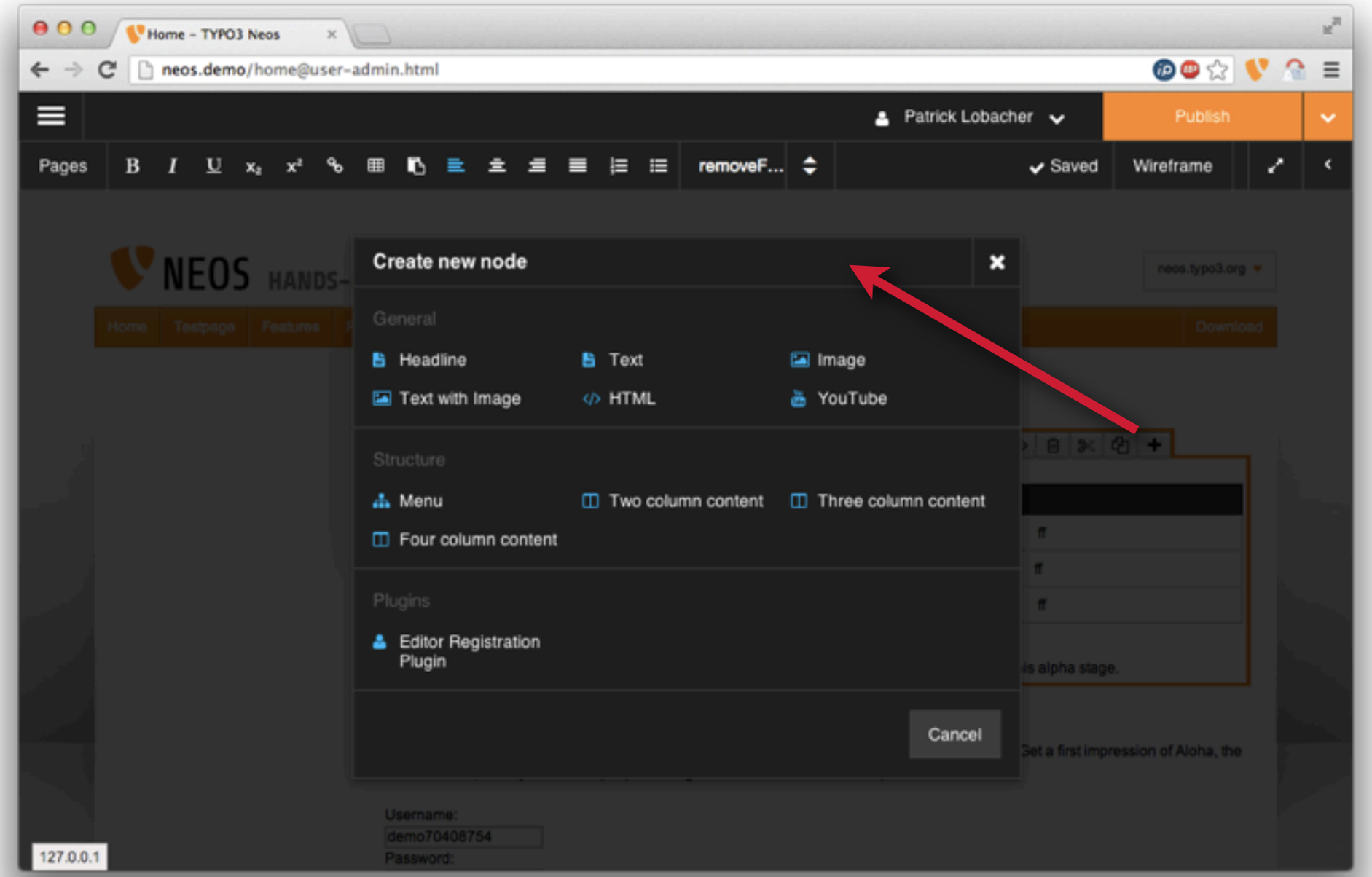
Content-Funktionen



Content-Funktionen



Content Element einfügen



typovision

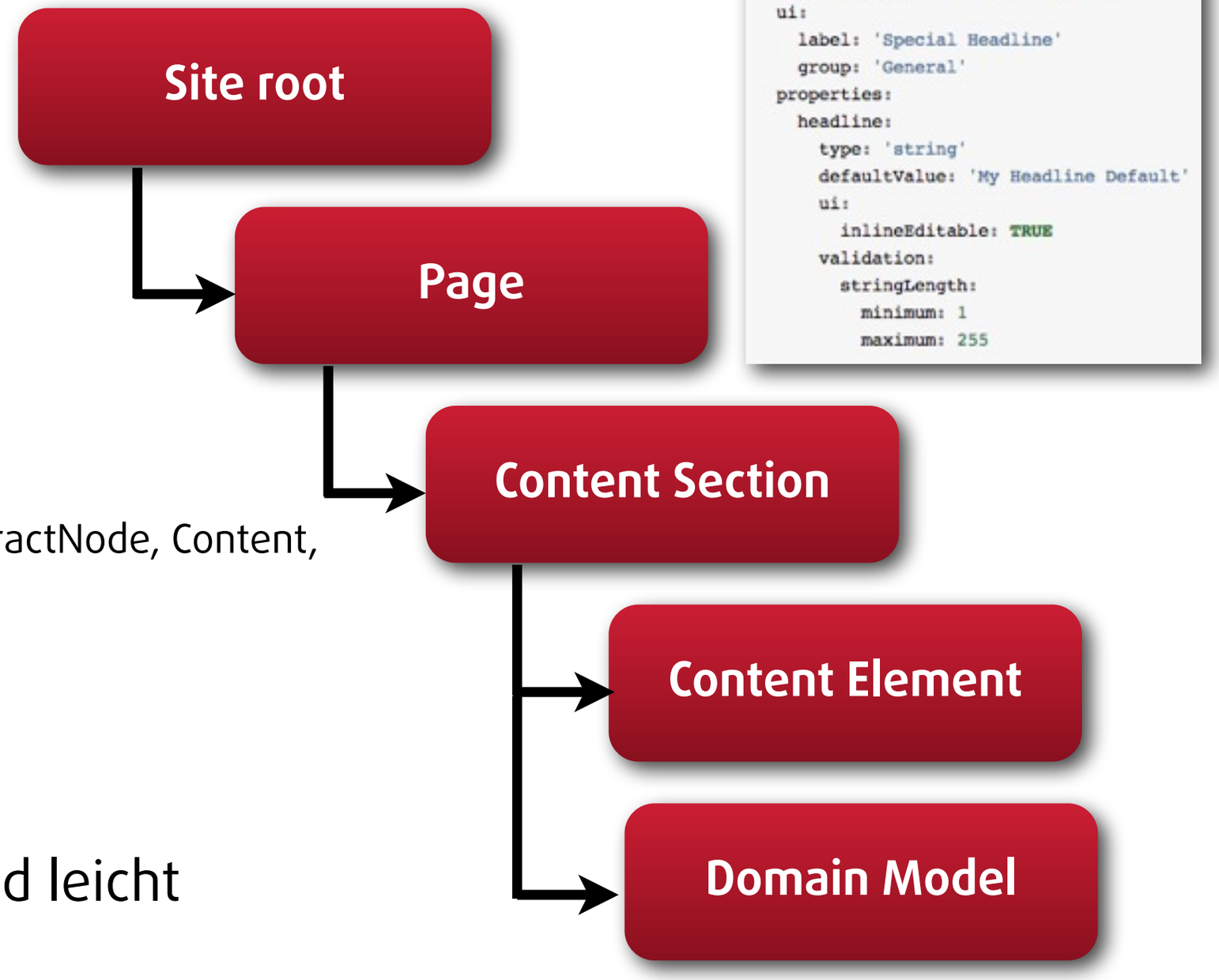
Internas von

TYPO3 Neos

Node Structure

- **TYPO3CR Node**
  - Node Name
    (dient der Identifikation)
  - Node Type
    (z.B. Document, Folder, AbstractNode, Content, ContentCollection...)
  - Properties
    (Abhängig vom NodeType)

- Einfach erweiterbar und leicht konfigurierbar

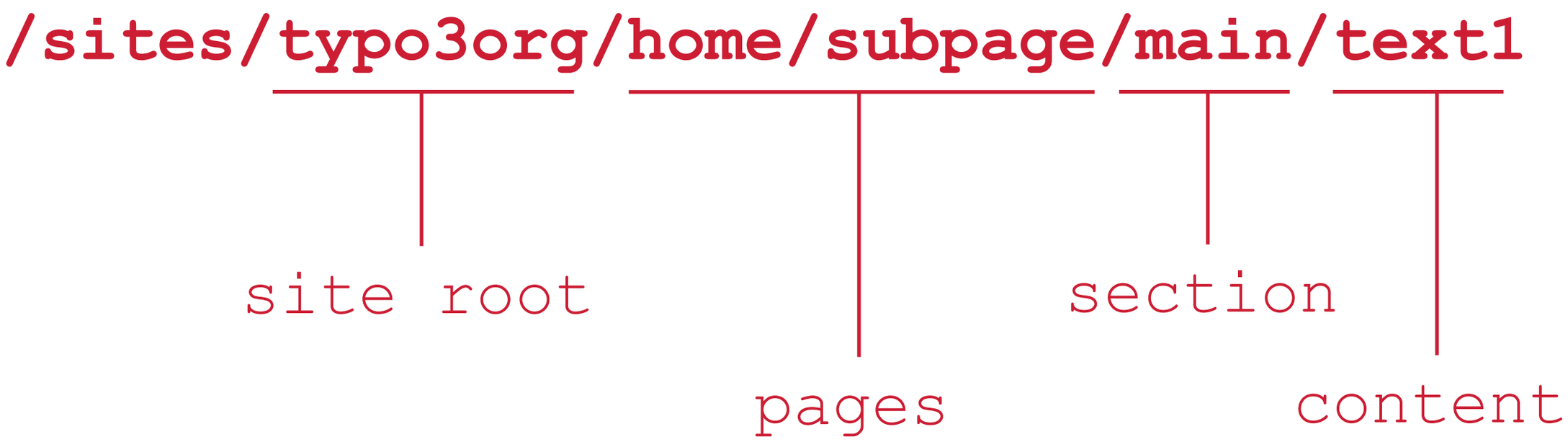



```
'My.Package:SpecialHeadline':
  superTypes: ['TYPO3.Neos:Content']
  ui:
    label: 'Special Headline'
    group: 'General'
  properties:
    headline:
      type: 'string'
      defaultValue: 'My Headline Default'
      ui:
        inlineEditable: TRUE
      validation:
        stringLength:
          minimum: 1
          maximum: 255
```

typovision

Node Zugriff

- Der Zugriff auf Nodes erfolgt durch „NodePaths“



typovision

TypoScript

- TypoScript ist eine hierarchische , objektorientierte und Prototypen-basierte Verarbeitungssprache
- Wird von Neos verwendet, um den Content flexibel zu rendern
- Objekte sind z.B. Array, Collection, Case, Menu, Page, Template, Plugin, Value, ...
- Objekte haben Eigenschaften, die das Objekt „konfigurieren“
- TypoScript hat Zugriff auf den jeweiligen „Context“ (z.B. Seitenbaum im Objekt „menu“ oder Seiteneigenschaften im Objekt „page“)
- Es gibt „Prozessoren“ die die Eigenschaftswerte verändern können (ähnlich stdWrap-Funktionen in TYPO3 CMS)

typovision

TypoScript: Fluid-Template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- {namespace typo3=TYPO3\Neos\ViewHelpers}{namespace typoScript=TYPO3\TypoScript\ViewHelpers
-->
    <meta charset="utf-8">
    <f:base/>
    <meta name="generator" content="TYPO3 Flow" />
    <title>Default Template</title>
    <f:section name="stylesheets">
       <link rel="stylesheet" href="../../../Public/Stylesheets/reset.css" media="all" />
    </f:section>
    <f:section name="javascripts">
       <script src="../../../Public/JavaScript/main.js"></script>
   </f:section>
</head>

<body class="js-off page-landing" id="typo3org">
<f:section name="body">
    <div class="p t3-reloadable-content" id="page">
        <!-- ### content ### -->
        <div class="c cl" id="content">
            <!-- ### aside ### -->
            <div class="a" id="aside">
                <typoScript:renderTypoScript path="parts/subMenu" />
```

typovision

TypoScript: Beispiel - Teil 1

```
include: TypoScripts/Library/ContentElements.ts2
namespace: TypoScript=TYPO3.TypoScript

page = Page
page.headerData {
    stylesheets = Template
    stylesheets {
        templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
        sectionName = 'stylesheets'
    }
    javascripts = Template
    javascripts {
        templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
        sectionName = 'javascripts'
    }
    title = Template
    title.templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/
TitleMenu.html'
    title.items = ${q(node).add(q(node).parents())}
    title << 1.wrap(prefix:'<title>', suffix: '</title>')
}
```

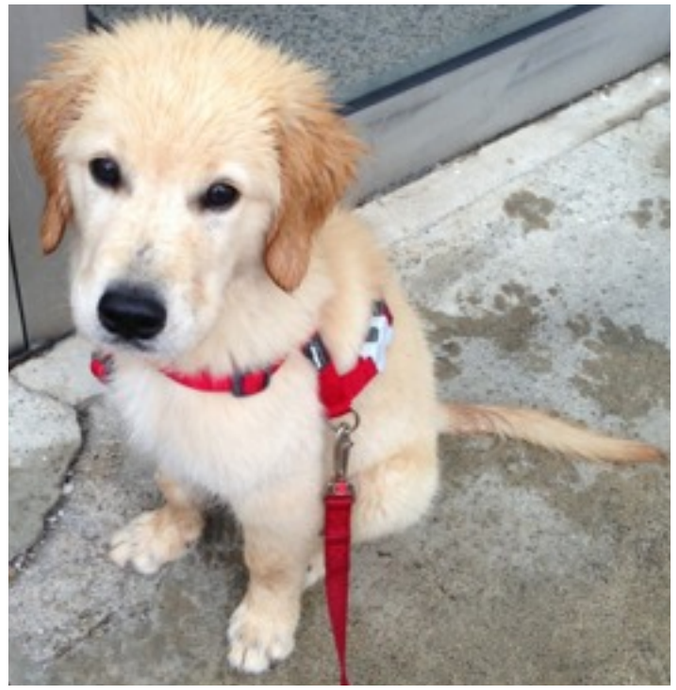
TypoScript: Beispiel - Teil 2

```
page.body {
    templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
    sectionName = 'body'
    parts {
        mainMenu = Menu
        mainMenu {
            entryLevel = 2
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/
MainMenu.html'
            maximumLevels = 2
        }
        subMenu = Menu
        subMenu {
            entryLevel = 3
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/
SubMenu.html'
            maximumLevels = 3
        }
        breadcrumb = Template
        breadcrumb {
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/
BreadcrumbMenu.html'
            items = ${q(node).add(q(node).parents())}
```

TypoScript: Prozessoren

- Beispiel:

```
myObject = MyObject {
    value = 'Rocky'
    value << 1.wrap(prefix: 'Der ', suffix: ' ist der Beste!')
}
# Resultat ist 'Der Rocky ist der Beste!</h1>'
```



- Weitere Prozessoren
  - crop
  - date
  - if, ifEmpty, ifBlank
  - multiply
  - override
  - replace
  - round
  - shiftCase
  - substring
  - toInteger
  - trim
  - wrap

Eel - Embedded Expression Language

- Während TypoScript Zuweisungen und Prozessoren beinhaltet, kann man mit Eel Ausdrücke der Art `myObject.foo = ${q(node).property('bar')}` formulieren
- Die Embedded Expression Language (Eel) ist ein Baustein um Domain Specific Languages (DSL) zu erstellen.
- Eel stellt eine reichhaltige Syntax zur Verfügung um beliebige Ausdrücke zu erstellen, damit sich der Autor der DSL auf die Semantik konzentrieren kann

```
${foo.bar}          // Traversal
${foo.bar()}        // Method call
${foo.bar().baz()} // Chained method call

${foo.bar("arg1", true, 42)} // Method call with arguments
${12 + 18.5}          // Calculations are possible
${foo == bar}       // ... and comparisons

${foo.bar(12+18.5, foo == bar)} // and of course also use it inside arguments
${[foo, bar]}             // Array Literal
${{foo: bar, baz: test}} // Object Literal
```

typovision

FlowQuery

- FlowQuery stellt eine Art jQuery für TYPO3 Flow dar
- FlowQuery stellt damit einen Weg dar, um Content (der ja eine TYPO3CR Node in Neos ist) im Eel zu verarbeiten
- In FlowQuery gibt es Operationen:
  - **property**
    Zugriff auf alle Eigenschaften einer Node
  - **filter**
    Filterausdrücke in „Fizzle“
  - **children**
    Gibt alle Kinder der TYPO3CR Node zurück
  - **parents**
    Gibt alle Eltern der TYPO3CR Node zurück
  - **Weitere Operationen: add, count, first, get, is, last, ...**

FlowQuery - Beispiele

- Anzahl der Kommentare = Anzahl der Kinder der aktuellen Node mit dem Namen „comments“, deren Eigenschaft „spam“ auf dem Wert „false“ steht.

```
numberOfComments = ${q(node).children('comments').children("[spam = false]").count()}
```

- Breadcrumb-Menü = Aktuelle Seite + alle Elternseiten

```
breadcrumb = Template
breadcrumb {
   templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/
TypoScriptObjects/BreadcrumbMenu.html'
   items = ${q(node).add(q(node).parents())}
}
{namespace neos=TYPO3\Neos\ViewHelpers}
<f:if condition="{items}">
   <ul class="breadcrumbs">
   <f:for each="{items}" as="item" reverse="TRUE">
      <f:if condition="{item.hiddenInIndex} == 0">
      <li>
         <neos:link.node node="{item}">{item.label}</neos:link.node>
      </li>
      </f:if>
   </f:for>
```

Technische Details

TypoScript

TypoScript 2.0 - Übersicht

- Nachdem bereits in TYPO3 CMS TypoScript verwendet wurde, bezeichnet man TypoScript in TYPO3 Neos (bzw. in TYPO3 Flow) mit der Versionsnummer 2.0
- TypoScript wird ausschließlich zum Rendern von Inhalten im Frontend verwendet (keine Konfiguration mehr von Usern im Backend beispielsweise)
- TypoScript ist hierarchisch, weil es auch hierarchischen Inhalt rendert
- TypoScript ist Prototypen-basiert (ähnlich wie JavaScript), da es beispielsweise erlaubt die Eigenschaften von allen Instanzen gleichzeitig zu ändern
- TypoScript ist eine Verarbeitungssprache, da es die Werte in einem Kontext verarbeitet und in einem eizigen Ausgabe-Wert überführt

TypoScript: Objekte

- TypoScript ist eine Sprache um **TypoScript Objekte** zu beschreiben
- Ein TypoScript Objekt besitzt Eigenschaften - sogenannte **Properties**
- TypoScript Objekte haben Zugriff zu einem „Kontext“, welcher letztlich eine Liste von Variablen ist
- TypoScript überführt diesen Kontext mit Hilfe der Properties in eine Ausgabe
- Intern kann TypoScript auch diesen Kontext verändern, sowie das Rendern von „verschachtelten“ Objekten (TypoScript-Tree) anstoßen
- TypoScript-Objekte werden durch PHP-Klasses realisiert, welche zur Laufzeit instanziert wird. Dabei kann eine Klasse die Basis von mehreren verschiedenen Objekten sein

TypoScript: Objekte

- Gültiges TypoScript wäre beispielsweise wie folgt
- Dabei werden TypoScript-Pfade immer in lowerCamelCase notiert und Objekte (Prototypen) in UpperCamelCase

```
foo = Page
my.object = Text
my.image = TYPO3.Neos.ContentTypes:Image
```

- Wertezuweisungen

```
foo.myProperty1 = 'Some Property which Page can access'
my.object.myProperty1 = "Some other property"
my.image.width = ${q(node).property('foo')}
```

- Werte die Strings darstellen müssen mit Anführungszeichen umschlossen werden (einzelne oder doppelte). Als Werte sind auch EEL-Ausdrücke zugelassen.

TypoScript: Syntax

- Man kann TypOScript auch mittels geschweiften Klammern notieren - dies bedeutet, dass der Pfad vor der öffnenden Klammer immer allen Pfaden innerhalb der Klammern vorangestellt wird.

```
my {
   image = Image
   image.width = 200
   object {
     myProperty1 = 'some property'
   }
}
```

- Dies ist identisch mit dem folgenden Code

```
my.image = Image
my.image.width = 200
my.object.myProperty1 = 'some property'
```

TypoScript: Objekt Instanziierung

- Es ist zudem möglich, Werte direkt bei der Instanziierung zu vergeben, wie das dritte Beispiel zeigt (alle Beispiele sind in der Wirkung identisch):

```
someImage = Image
someImage.foo = 'bar'

someImage = Image
someImage {
  foo = 'bar'
}

someImage = Image {
  foo = 'bar'
}
```

TypoScript Objekte sind SiteEffect-Free

- Obwohl TypoScript Objekte ihren Kontext verändern können, sind sie ohne „Seiteneffekte“.
- Dafür wird der Kontext nach der Benutzung eines TypOScript-Objektes wieder „aufgeräumt“, auch wenn dieser vorher verändert wurde
- TypoScript Objekte können nur andere TypoScript-Objekte ändern, die verschachtelt sind - aber nicht Objekte welche davor oder danach kommen
- Das sorgt dafür, dass sich ein TypoScript-Pfad zusammen mit seinem Kontext immer gleich verhält - egal an welcher Stelle dieser aufgerufen wird

TypoScript: Prototypen

- Wenn ein TypoScript Objekt instanziiert wird (weil z.B. jemand `someImage = Image` schreibt), dann wird der Prototyp für dieses Objekt kopiert und als Basis für dieses neue Objekt (someImage) verwendet.
- Ein Prototyp wird wie folgt definiert:

```
prototype(MyImage) {
   width = '500px'
   height = '600px'
}
```

- Nun kann man das Objekt verwenden:

```
# Das Objekt someImage hat eine Breite von 500px und eine Höhe
von 600 px
someImage = MyImage
# Nun ist die Breite 100px (und die Höhe nach wie vor 600px)
someImage.width = '100px'
```

TypoScript: Prototypen

- Prototypen sind veränderbar:

```
prototype(MyYouTube) {
  width = '100px'
  height = '500px'
}

# Hiermit wird die Breite für alle Instanzen
# auf 400px geändert
prototype(MyYouTube).width = '400px'

# Man kann auch neue Eigenschaften (Properties) definieren
prototype(MyYouTube).showFullScreen = ${true}

# Man kann einen Prototypen auch „vererben“
prototype(MyImage) < prototype(TYPO3.TypoScript:Template)
```

TypoScript: Prototypen

- Über Verberbung bleiben Prototypen aneinander „gebunden“. Ändert sich eine Eigenschaft in einer Instanz, wird diese auch in der anderen geändert.

```
prototype(TYPO3.TypoScript:Template).fruit = 'apple'
prototype(TYPO3.TypoScript:Template).meal = 'dinner'

# MyImage hat nun auch die Eigenschaften "fruit = apple" and "meal = dinner"
prototype(MyImage) < prototype(TYPO3.TypoScript:Template)

# Da MyImage das Objekt Template *erweitert*,
# hat MyImage.fruit nun ebenfalls den Wert 'Banana'
prototype(TYPO3.TypoScript:Template).fruit = 'Banana'
```

-
```
# Da die Eigenschaft „meal“ nun in der Kind-Klasse überschrieben wurde,
# hat ein Überschreiben in der ElternKklasse keine Auswirkung mehr
prototype(MyImage).meal = 'breakfast'
prototype(TYPO3.TypoScript:Template).meal = 'supper'
```

TypoScript: Prototypen

- Prototypen-Verberbung ist nur auf globaler Ebene möglich

```
prototype(Foo) < prototype(Bar)
```

- Nicht möglich wären daher folgende Anweisungen

```
prototype(Foo) < some.prototype(Bar)
other.prototype(Foo) < prototype(Bar)
prototype(Foo).prototype(Bar) < prototype(Baz)
```

TypoScript: Prototypen

- Prototypen-Verberbung ist nur auf globaler Ebene möglich

```
# Setze die Eigenschaft „bar“ (bzw. „some.thing“) für alle Objekte vom Typ „Foo“
prototype(Foo).bar = 'baz'
prototype(Foo).some.thing = 'baz2'

# Für alle Objekte vom Typ „Foo“, welche innerhalb des Pfades „some.path“ a
# auftauchen, setze die Eigenschaft „some“
some.path.prototype(Foo).some = 'baz2'

# Für alle Objekte vom Typ „Bar“ die innerhalb von Objekten vom Typ „Foo“
# auftauchen, setze die Eigenschaft „some“
prototype(Foo).prototype(Bar).some = 'baz2'

# Kombination aus allen Möglichkeiten zuvor
prototype(Foo).left.prototype(Bar).some = 'baz2'
```

TypoScript: Namespaces

- Namespaces können bei der Deklaration verwendet werden

```
# Definiert einen Namespace „Acme.Demo“ für den Prototyp „YouTube“
prototype(Acme.Demo:YouTube) {
   width = '100px'
   height = '500px'
}
```

- Der Namespace ist per Konvention der Package-Key, des Packages wo sich das TypoScript befindet

- Man kann voll qualifizierte Names spaces verwenden:

```
prototype(TYPO3.Neos:ContentCollection.Default) <
prototype(TYPO3.Neos:Collection)
```

typovision

TypoScript: Namespaces

- Verwendet man keinen Namespace, wird per Default immer `TYPO3.Neos` verwendet.
- Man kann aber auch eine Namespace-Direktive verwenden um einen eigenen Namespace zu verwenden:

```
namespace Foo = Acme.Demo

# Die folgenden beiden Anweisungen sind identisch
video = Acme.Demo:YouTube
video = Foo:YouTube
```

TypoScript: Eigenschaften

- Auch wenn TypoScript-Objekte direkt auf den Kontext zugreifen können, sollte man dennoch Eigenschaften (Properties) dafür verwenden

```
# Wir nehmen an, dass es eine Eigenschaft "foo=bar"
# im aktuellen Kontext an dieser Stelle gibt
myObject = MyObject

# Explizites Zuweisen des Wertes der Variable „foo“
# aus dem aktuellen Kontext und Zuweisen an die
# „foo“ Eigenschaft des Objektes „myObjekt“
myObject.foo = ${foo}
```

- Objekte sollten ausschließlich eigene Eigenschaften verwenden, um den Output zu generieren
- Lediglich bei der Prototypen-Definition kann man direkt auf den Kontext zugreifen: `prototype(MyObject).foo = ${foo}`

TypoScript: Manipulation des Kontext

- Der TypoScript-Kontext kann direkt manipuliert werden, indem man die `@override` Meta-Eigenschaft verwendet:

```
myObject = MyObject
myObject.@override.bar = ${foo * 2}
```

- Der obige Code erzeugt nun ein weitere Kontext-Variable mit dem Namen `bar` und dem doppelten Wert von `foo`.

TypoScript: Prozessoren

- Prozessoren erlauben es, TypoScript Eigenschaften zu manipulieren:

```
myObject = MyObject {
    value = 'some value'
    value << 1.wrap(prefix: 'before ', suffix: ' after')
}
# Das Ergebnis ist nun 'before some value after'
```

- Es können mehrere Prozessoren verwendet werden
- Die Reihenfolge ergibt sich aus der numerische Position im TypoScript - im obigen Beispiel würde der `2.wrap` das Ergebnis aus `1.wrap` weiterverarbeiten

TypoScript: Prozessoren

- Prozessoren sind PHP-Klassen, welche das `\TYPO3\TypoScript\ProcessorInterface` implementieren
- Dort muss eine Methode `process($subject)` realisiert sein.
- Der dort zurückgegebene Wert wird als Wert des Prozessors verwendet.
- Man kann entweder den FQON (Fully Qualified Object Name) angeben (dies erlaubt die Verwendung von eigenen Prozessoren) oder die „Kurzform“ (z.B. `\TYPO3\TypoScript\Processors\WrapProcessor` oder `wrap`)
- Gibt man die Kurzform an, hängt wandelt Neos den Anfangsbuchstaben in einen Großbuchstaben um und hängt „Processor.php“ an den Namen. Nun wird im Verzeichnis `Packages\Application\TYPO3.TypoScript\Classes\TYPO3\TypoScript\Processors` nach einer derart benannten Datei gesucht
- Folgende beiden Angaben sind daher identisch:

```
value << 1.wrap(prefix: 'before ', suffix: ' after')
value << 1.TYPO3\TypoScript\Processors\WrapProcessor(prefix: 'before ',
suffix: ' after')
```

TypoScript: Prozessoren - crop

- Schneidet einen Teil eines Strings ab und ersetzt den abgeschnittenen Teil durch einen weiteren String
- **Argumente:**
  - `maximumCharacters` (integer): Länge, nach der abgeschnitten werden soll
  - `preOrSuffixString` (string): Angabe ob der ersetze String vorne oder hinten zugefügt wird
  - `options` (integer): Bitmask-Kombination aus folgenden Konstanten:
    - 1 = CROP_FROM_BEGINNING: Abschneiden vom Anfang des String anstatt vom Ende
    - 2 = CROP_AT_WORD: Abschneiden an der nächsten Wortgrenze.
    - 4 = CROP_AT_SENTENCE: Abschneiden an der nächsten Satzgrenze.
- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'Lorem ipsum dolor sit amet!'
test.value << 1.crop(maximumCharacters: 7, preOrSuffixString:'...',
options: 3)
# Resultat
... dolor sit amet!
```

TypoScript: Prozessoren - date

- Wandelt einen Unix-Timestamp in ein date() Datum um
- **Argumente:**
  - `format` (string): Setzt das Format (siehe date()-Funktion)
  - `timezone` (string): Setzt die Zeitzone, siehe `http://php.net/manual/en/timezones.php`.
- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = '1374733358'
test.value << 1.date(format: 'd.m.Y',timezone:'Europe/Berlin')

# Resultat
25.07.2013
```

TypoScript: Prozessoren - if

- Gibt den „trueValue“ zurück, wenn die Bedingung TRUE ist, ansonsten den „falseValue“
- **Argumente:**
  - `condition` (boolean): Die Bedingung (oder einfach TRUE/FALSE)
  - `trueValue` (string): Der Rückgabewert, falls die Bedingung TRUE ist
  - `falseValue` (string): Der Rückgabewert falls die Bedingung FALSE ist

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value << 1.if(condition:'Hallo',trueValue:'TRUE',falseValue:'FALSE')

# Resultat
TRUE
```

TypoScript: Prozessoren - ifBlank

- Gibt den „trueValue“ zurück, wenn die Bedingung TRUE ist, ansonsten den „falseValue“

- **Argumente:**
  - `replacement` (string): Mit diesem String wird der Wert überschrieben, wenn der Wert leer ist

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = ''
test.value << 1.ifBlank(replacement:'Blindtext!')

# Resultat
Blindtext
```

typovision

TypoScript: Prozessoren - ifBlank

- Überschreibt den Wert, wenn dieser leer ist (ohne trimmen)
- **Argumente:**
  - `replacement` (string): Mit diesem String wird der Wert überschrieben, wenn der Wert leer ist
- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = ''
test.value << 1.ifBlank(replacement:'Blindtext!')

# Resultat
Blindtext
```

typovision

TypoScript: Prozessoren - ifEmpty

- Überschreibt den Wert, wenn dieser leer ist (mit trimmen)
- **Argumente:**
  - `replacement` (string): Mit diesem String wird der Wert überschrieben, wenn der Wert leer ist

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = '        '
test.value << 1.ifEmpty(replacement:'Blindtext!')

# Resultat
Blindtext
```

typovision

TypoScript: Prozessoren - override

- Überschreibt einen Wert mit einem String, wenn der String nicht leer ist

- **Argumente:**
  - `replacement` (string): Mit diesem String wird der Wert überschrieben, wenn der String nicht leer ist

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'Hallo'
test.value << 1.override(replacement:'Welt!')

# Resultat
Welt!
```

TypoScript: Prozessoren - replace

- Ersetzt einen Teil des Wertes (verwendet `str_replace()` im Hintergrund)
- **Argumente:**
  - `search` (string): Suchstring, der ersetzt werden soll
  - `replace` (string): Ersetzender String

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'Hallo Welt!'
test.value << 1.replace(search:'Welt', replace:'TypoScript')

# Resultat
Hallo TypoScript!
```

TypoScript: Prozessoren - round

- Rundet den Wert, wenn es sich bei diesem um eine Fließkommazahl handelt - ist die Zahl ein Integer, passiert nichts

- **Argumente:**
  - `precision` (integer): Anzahl der Nachkommastellen

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 3.14159265359
test.value << 1.round(precision:4)

# Resultat
3.1416
```

TypoScript: Prozessoren - shiftCase

- Ändert die Schreibweise eines Strings (Groß- oder Kleinschreibung)
- **Argumente:**
  - `direction` (string): Art der Schreibweise und zwar
    - `SHIFT_CASE_TO_UPPER (upper)` - Großschreibweise
    - `SHIFT_CASE_TO_LOWER (lower)` - Kleinschreibweise
    - `SHIFT_CASE_TO_TITLE (title)` - Erster Buchstabe groß, Rest klein

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'HalloWelt'
test.value << 1.shiftCase(direction:'upper')

# Resultat
HALLOWELT
```

TypoScript: Prozessoren - substring

- Gibt einen Teilstring eines Strings zurück
- **Argumente:**
  - `start` (integer): Linke Grenze des Teilstrings (0 = 1. Position, 1 = 2. Position, ...)
  - `length` (integer): Länge des Teilstrings

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'HalloWelt!'
test.value << 1.substring(start:5,length:4)

# Resultat
Welt
```

typovision

TypoScript: Prozessoren - toInteger

- Wandelt den Wert in einen Integer um
- **Argumente:**
  - keine
- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = '127.12'
test.value << 1.toInteger()

# Resultat
127
```

typovision

TypoScript: Prozessoren - trim

- Entfernt Whitespaces rechts und links von einem Wert
- **Argumente:**
  - keine
- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = '     127.12 '
test.value << 1.trim()

# Resultat
127.12
```

typovision

TypoScript: Prozessoren - wrap

- Hängt vorne und hinten an einen Wert jeweils einen String an
- **Argumente:**
  - `prefix` (string): String, der vorne angehängt wird
  - `suffix` (string): String, der hinten angehängt wird

- **Beispiel:**

```
test = TYPO3.TypoScript:Value
test.value = 'Welt'
test.value << 1.wrap(prefix:'Hallo ',suffix:'!')

# Resultat
127.12
```

Technische Details

Eel

Eel - Embedded Expression Language

- Neben einfachen TypoScript-Zuweisungen wie `myObject.foo = 'bar'` ist es mittels Eel möglich Ausdrücke wie `myObject.foo = ${q(node).property('bar')}` zu formulieren.
- Jeder Eel-Ausdruck wird mittels `${...}` notiert
- Eel ist JavaScript sehr ähnlich
- Eel unterstützt keine Variablen-Zuweisungen oder Kontrollstrukturen
- Eel unterstützt die üblichen JavaScript Operatoren für Arithmetik und Vergleiche
- Eel unerstützt den Dreifach-Operator für Vergleiche: <condition> ? <ifTrue> : <ifFalse>
- Sobald Eel auf eine Objekt-Eigenschaft zugreift, wird der entsprechende Getter aufgerufen
- Objekt-Zugriff über die Offset-Notation wird wie folgt unterstützt: foo['bar']

Eel - Beispiele

- **Folgende Ausdrücke sind beispielsweise gültig:**
- Eel selbst definiert keine Funktion oder Variablen, sondern verwendet das Eel Context Array. Funktionen und Objekte, auf die man zugreifen will, können dort definiert werden.
- Daher eignet sich EEl perfekt als "domain-specific language construction kit", welches die Syntax, nicht aber die Semantik der DSL (Domain Specific Language) zur Verfügung stellt.
- Für Eel innerhalb von TypoScript ist die Sematik wie folgt definiert:
  - Alle Variablen des TypoScript Kontextes sind im EEl Kontext zugänglich
  - Die spezielle Variable **this** zeigt immer auf die aktuelle TypoScript Objekt Implementierung
  - Es gibt eine Funktion **q()**, die ihre Argumente in einem FlowQuery-Objekt einhüllt

Eel - Sematik

- **Folgende Ausdrücke sind beispielsweise gültig:**

```
${foo.bar}                          // Traversal
${foo.bar()}                        // Methoden-Aufruf
${foo.bar().baz()}                  // Chained method call
${foo.bar("arg1", true, 42)}        // Methoden-Aufruf mit Argumenten
${12 + 18.5}                        // Kalkulation
${foo == bar}                       // Vergleiche
${foo.bar(12+7, foo == bar)}        // Alles kombiniert
${[foo, bar]}                       // Array Literal
${{foo: bar, baz: test}}            // Object Literal
```

Technische Details

FlowQuery

typovision

FlowQuery - jQuery für Flow

- FlowQuery stellt eine Art jQuery für Flow dar und wurde auch von jQuery stark inspiriert
- FlowQuery ist ein Weg um Inhalte (Content = TYPO3CR Node innerhalb von Neos) im Eel Kontekt zu verarbeiten
- FlowQuery Operationen werden durch PHP-Klassen implementiert
- Für jede FlowQuery-Operation, muss das Package installiert werden, welches die Operation (Klasse) enthält
- Jedes Packages kann seine eigenen FlowQuery Operationen hinzufügen
- Das TYPO3.Eel Package beinhaltet bereits ein Set an Basis-Operationen

typovision

FlowQuery - add Operation (TYPO3.Eel)

- Die add Operation fügt ein weiteres flowQuery Objekt zum aktuellen hinzu
- **Beispiel:**

```
items = ${q(node).add(q(node).parents())}
```

Hier wird die aktuelle Node ermittelt und dazu alle Eltern-Nodes hinzuaddiert. Dies wird beispielsweise verwendet, um eine Breadcrumb-Navigation zu erstellen. Hierfür wird die Rootline der Nodes benötigt, die mit Hilfe des obigen Statements ermittelt wird

typovision

FlowQuery - count Operation (TYPO3.Eel)

- Die count Operation gibt die Anzahl der Objekte (oder Array-Items) zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents()).count()}
```

Hier wird zunächst die Rootline ermittelt und dann mittels count() durchgezählt. Die Zahl wird zurückgegeben.

FlowQuery - first Operation (TYPO3.Eel)

- Die first Operation gibt das erste Objekt zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents())
              .first().property('title')}
```

Hier wird zunächst die Rootline ermittelt und dann mittels first() das erste Elemente ausgewählt. Von diesem wird der Titel zurückgegeben.

Wenn er zwei Seiten gibt: **Home > Kompendium**, dann wird nun „**Kompendium**“ zurück gegeben.

FlowQuery - get Operation (TYPO3.Eel)

- Wenn FlowQuery verwendet wird, ist das Ergebnis ebenfalls wieder FlowQuery. Die get Operation „befreit“ das Resultat vom „FlowQuery“
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents()).first().get(0)}
```

Hier wird zunächst die Rootline ermittelt und dann mittels first() das erste Elemente ausgewählt. Dieses wird nun mittels get() umgewandelt in ein Array (ohne wäre es ein FlowQuery Objekt) und dort wird das erste Element ausgewählt und zurückgegeben.

Wenn es zwei Seiten gibt: **Home > Kompendium**, dann wird folgendes zurück gegeben:

```
Node /sites/demo/homepage/Kompendium[TYPO3.Neos:Page]
```

typovision

FlowQuery - if Operation (TYPO3.Eel)

- Überprüft, ob mindestens eines der Elemente im Kontext einem gegebenen Filter entspricht
- **Beispiel:**

```
test.value = ${q(node).is('[instanceof TYPO3.Neos:Page]')
              ? 'Seite' : 'Keine Seite'}
```

Der if Operator überprüft zunächst, ob die aktuelle Node eine Instanz der Nodetypes TYPO3.Neos:Page (also eine normale Seite) ist. Wenn dies der Fall ist, wird der Wert „Seite“ zurückgegeben, ansonsten „Keine Seite“

FlowQuery - last Operation (TYPO3.Eel)

- Die last Operation gibt das letzte Objekt zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents())
              .last().property('title')}
```

Hier wird zunächst die Rootline ermittelt und dann mittels last() das letzte Elemente ausgewählt. Von diesem wird der Titel zurückgegeben.

Wenn er zwei Seiten gibt: **Home > Kompendium**, dann wird nun „**Home**“ zurück gegeben.

typovision

FlowQuery - property Operation (TYPO3.Eel)

- Die property Operation gibt die Eigenschaften eines Objekts zurück
- Wenn die Eigenschaft mit _ beginnt, werden interne Werte ausgelesen, z.B.:
  - `_path` (Node Pfad)
  - `_nodeType.name` (Node Typ)

- **Beispiel:**

```
test.value = ${q(node).property('title')}
```

Gibt die Eigenschaft „title“ der aktuellen Node zurück.

typovision

FlowQuery - children Operation (TYPO3.Eel)

- Die children Operation gibt die Kind-Objekte des aktuellen Objekts zurück. Dies sind bei einer Seite beispielsweise die Sektionen und dort die Content-Bereiche.
- Es ist möglich eine Filter-Option anzugeben: `...children('main')...`

- **Beispiel:**

```
test.value = ${q(node).children().property('_path')}
```

Ergebnis:
/sites/demo/homepage/Kompendium/kompendium1/main

FlowQuery - filter Operation (TYPO3.Eel)

- Die filter Operation limitiert das Ergebnis-Set der Objekte. Der Filter-Ausdruck wird mittels Fizzle notiert und hat die folgende Syntax:
  "[" [<value>] <operator> <operand> "]" und unterstützt die folgenden Operatoren:
  - `=` (Gleichheit)
  - `$=` (Wert endet mit dem Operanden)
  - `^=` (Wert beginnt mit dem Operanden)
  - `*=` (Wert enthält den Operanden)
  - `instanceof` (Prüft ob der Wert eine Instanz des Operanden ist)

- **Beispiel:**

```
test.value = ${q(node).children().filter('main').
             first().property('_path')}
```

  Gibt die Eigenschaft „_path“ der ersten Kind-Node im Pfad „main“ zurück.

Technische Details

Fizzle

Fizzle

- Filter Operationen (z.B. filter() in FlowQuery) werden in Fizzle notiert
- Property Name Filters
  - Dies kann der erste Teil des Filters sein, wie `foo`, `foo.bar` oder `foo.bar.baz`
- Attribut Filter
  - `baz[foo]`
  - `baz[answer = 42]`
  - `baz[foo = "Bar"]`
  - `baz[foo = 'Bar']`
  - `baz[foo ^= "Bar"]`
  - `baz[foo $= "Bar"]`
  - `baz[foo *= "Bar"]`
- Operatoren
  - `=` (Gleichheit)
  - `$=` (Wert endet mit dem Operator)
  - `^=` (Wert beginnt mit dem Operator)
  - `*=` (Wert enthält den Operator)
  - `instanceof` (Prüft ob der Wert eine Instanz des Operator ist)

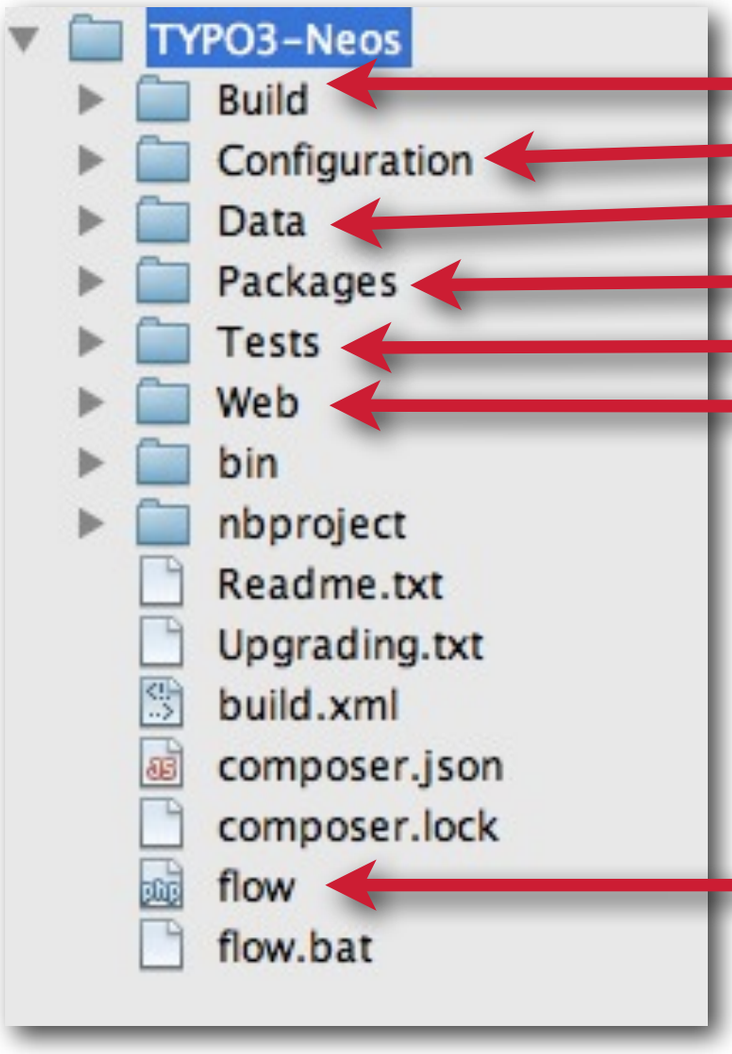
typovision

Aufbau des

TYPO3 Neos Renderings

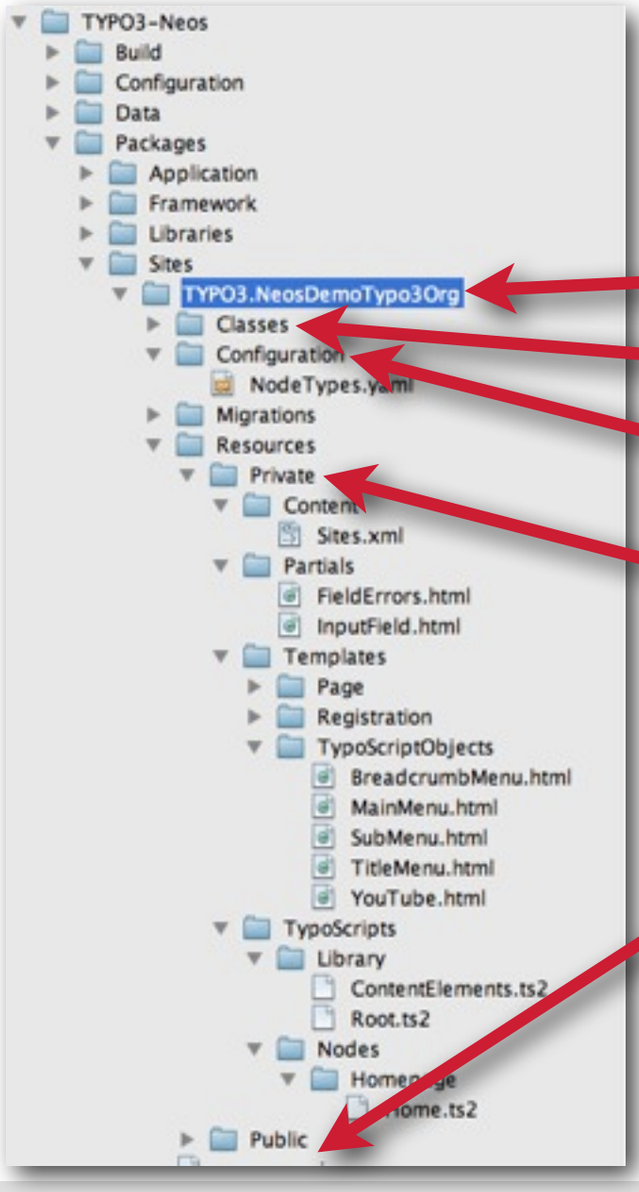
typovision

Die Verzeichnisstruktur der TYPO3 Flow Basis:



typovision

Die Verzeichnisstruktur einer TYPO3 Neos Site:



Startpunkt der TYPO3 Neos Site

Klassen-Verzeichnis (z.B. für Registrierungs-Klassen)

Konfiguration (z.B. der Node-Typen)

Private Ressourcen (z.B. Content, Templates, TypoScript, ...)

Öffentliche Ressourcen (Assets, z.B. CSS, Fonts, Bilder, JS, ...)

typovision

TypoScript - Lade-Reihenfolge - Root.ts2

- Das Rendering in TYPO3 Neos wird komplett über TypoScript 2 realisiert
- Dafür wird zunächst die folgende Datei geladen und der Inhalt verwertet:

  Packages/Application/TYPO3.Neos/Resources/Private/TypoScript/Root.ts2

  Achtung: In Pfadangaben wird das „Resources“ weggelassen, z.B.

```
$mergedTypoScriptCode = $this->readExternalTypoScriptFile('resource://
TYPO3.Neos/Private/TypoScript/Root.ts2') . $siteRootTypoScriptCode;
```

- Dann wird als nächstes die folgende Datei geladen - diese beinhaltet den TypoScript Code der eigenen Site:

  Packages/Sites/[Vendor.Sitename]/Resources/Private/Typoscripts/Library/Root.ts2

TypoScript: Root.ts2 (TYPO3.Neos)

- Die Datei Root.ts2 im TYPO3.Neos Resources-Verzeichnis lädt anschließend folgende Dateien nach:

```
include: resource://TYPO3.TypoScript/Private/TypoScript/Root.ts2
include: resource://TYPO3.Neos/Private/TypoScript/DefaultTypoScript.ts2
include: resource://TYPO3.Neos/Private/TypoScript/WireframeMode.ts2
include: resource://TYPO3.Neos.NodeTypes/Private/TypoScript/Root.ts2
```

TypoScript: Root.ts2 (TYPO3.TypoScript)

- Die Datei `Root.ts2` im TYPO3.TypoScript Resources-Verzeichnis definiert nun 6 Klassennamen für die jeweiligen Objekte Array, Template, Collection, Case, Matcher und Value. Die Klassen befinden sich in **Packages/TYPO3.TypoScript/Classes/TYPO3/TypoScript/TypoScriptObjects**:

```
prototype(TYPO3.TypoScript:Array).@class = 'TYPO3\\TypoScript\
\TypoScriptObjects\\ArrayImplementation'
prototype(TYPO3.TypoScript:Template).@class = 'TYPO3\\TypoScript\
\TypoScriptObjects\\TemplateImplementation'
prototype(TYPO3.TypoScript:Collection).@class = 'TYPO3\\TypoScript\
\TypoScriptObjects\\CollectionImplementation'
prototype(TYPO3.TypoScript:Case).@class = 'TYPO3\\TypoScript\\TypoScriptObjects
\\CaseImplementation'
prototype(TYPO3.TypoScript:Matcher).@class = 'TYPO3\\TypoScript\
\TypoScriptObjects\\MatcherImplementation'
prototype(TYPO3.TypoScript:Value).@class = 'TYPO3\\TypoScript\
\TypoScriptObjects\\ValueImplementation'
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das TYPO3.Neos Default-TypoScript geladen und verarbeitet
- Hier wird das Objekt TYPO3.Neos:ContentCase vom Objekt TYPO3.TypoScript:Case vererbt und die Eigenschaften @position, condition und type innerhalb des Schlüssels default gesetzt.
- Das "case" TypoScript Objekt rendert seine Kind-Elemente

```
# TYPO3.Neos:ContentCase inherits TYPO3.TypoScript:Case and overrides
# the default case with a catch-all condition for the default case,
# setting the type variable to the name of the current nodes' node type
#
prototype(TYPO3.Neos:ContentCase) < prototype(TYPO3.TypoScript:Case) {
    default {
        @position = 'end'
        condition = ${true}
        type = ${q(node).property('_nodeType.name')}
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das Objekt TYPO3.Neos:Template vom Objekt TYPO3.TypoScript:Template vererbt und die Eigenschaften `node` auf den aktuellen Node gesetzt.

```
# TYPO3.Neos:Template makes the current node available in addition to the
# default TYPO3.TypoScript:Template properties
#
prototype(TYPO3.Neos:Template) < prototype(TYPO3.TypoScript:Template) {
   node = ${node}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das Objekt ContentCollection initialisisert, welches wiederum später Content-Elemente aufnehmen kann.

```
# Case TS Object for ContentCollection
#
# Using a Case object allows for defining custom rendering of content
# collections for specific node types.
#
# The default case is to render all children of the current content collection
# node. This is the case for regular pages: A page contains a content
# collection which contains content elements.
#
prototype(TYPO3.Neos:ContentCollection) {
    @class = 'TYPO3\\Neos\\TypoScript\\ContentCollectionCaseImplementation'

    default {
        condition = ${true}
        type = 'TYPO3.Neos:ContentCollection.Default'
        @position = 'end'
    }
}
prototype(TYPO3.Neos:PrimaryContentCollection) < prototype(TYPO3.Neos:ContentCollection)
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Hier wird der Default-Fall für die Content-Collection festgelegt

```
# Default case for ContentCollection TS Object
#
prototype(TYPO3.Neos:ContentCollection.Default) < prototype(TYPO3.TypoScript:Collection)
{
    @class = 'TYPO3\\Neos\\TypoScript\\DefaultContentCollectionImplementation'

    # The node path is set by the user on TYPO3.Neos:ContentCollection. This
    # path needs to be copied from the context to the local variable of
    # ContentCollection.Default
    nodePath = ${nodePath}

    # To enable direct rendering of a ContentCollection we check if the current node is a
    # content collection already or if the child nodes of a content collection specified
    # by nodePath should be rendered
    collection = ${q(node).is('[instanceof TYPO3.Neos:ContentCollection]') ?
q(node).children() : q(node).children(this.getNodePath()).children()}
    itemName = 'node'
    itemRenderer = TYPO3.Neos:ContentCase
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das TYPO3.Neos:Page Objekt aufgebaut (vererbt von TYPO3.Neos:Template)
- Gesetzt wird zudem ein Template für das Rendering, das body, head und backendHeader Objekt

```
    # TYPO3.Neos:Page is the default object used for rendering in most sites
    #
    prototype(TYPO3.Neos:Page) < prototype(TYPO3.Neos:Template)
    prototype(TYPO3.Neos:Page) {
        templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
    PageTemplate.html'
        htmlAttributes = ''

        body = TYPO3.Neos:Template
        body.title = ${q(node).property('title')}
        body.nodePath = ${q(node).property('_path')}

        head = TYPO3.TypoScript:Array

        backendHeader = TYPO3.Neos:Template
        backendHeader.templatePath = 'resource://TYPO3.Neos/Private/Templates/
TypoScriptObjects/PageHead.html'
    }
```

Template: TYPO3.Neos/Private/Templates/TypoScriptObjects/PageTemplate.html

- Aufbau des gesamten Seitentemplates (PageTemplate.html)

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<!DOCTYPE html>
<html version="HTML+RDFa 1.1"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:typo3="http://www.typo3.org/ns/2012/Flow/Packages/Neos/Content/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      {htmlAttributes -> f:format.raw()}>
    <!--
        This website is powered by TYPO3 Neos, the next generation CMS, a free Open
        Source Enterprise Content Management System licensed under the GNU/GPL.

        TYPO3 Neos is based on Flow, a powerful PHP application framework licensed under the GNU/LGPL.

        More information and contribution opportunities at http://neos.typo3.org and http://flow.typo3.org
    -->
    <head>
        <f:base />
        <meta charset="UTF-8" />
        <ts:render path="head" />

        <ts:render path="backendHeader" />
    </head>
    <body<f:if condition="{bodyAttributes}"><f:for each="{bodyAttributes}" key="attribute" as="value"> {attribute}="{value ->
f:format.raw()}"</f:for></f:if>>

        <f:security.ifAccess resource="TYPO3_Neos_Backend_BackendController">
            <neos:contentElement node="{node}" page="TRUE" />
        </f:security.ifAccess>

        <ts:render path="body" />
        <neos:backend.container node="{node}"/>
    </body>
</html>
```

typovision

Template: TYPO3.Neos/Private/Templates/TypoScriptObjects/PageHead.html

- Aufbau des PageHeader-Templates (PageHead.html) - Teil 1
- Das JavaScript dient dazu, um direkt im Editor auf der aktuellen Seite herauszukommen, wenn man `/neos` an die URL hängt um damit die Admin-Oberfläche aufzurufen.

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ext=TYPO3\ExtJS\ViewHelpers}
{namespace aloha=TYPO3\Aloha\ViewHelpers}
<script>
    try {
        with (window.location) {
            sessionStorage.setItem(
                'TYPO3.Neos.lastVisitedUri',
                [protocol, '//', host, pathname, (pathname.charAt(pathname.length - 1) ===
'/' ? '{node.name}.html' : '')].join('')
            );
        }
    } catch(e) {}
</script>
```

typovision

Template: TYPO3.Neos/Private/Templates/TypoScriptObjects/PageHead.html

- Aufbau des PageHeader-Templates (PageHead.html) - Teil 2

```
<f:security.ifAccess resource="TYPO3_Neos_Backend_BackendController">
    <f:if condition="{0: 'live'} != {0: node.context.workspace.name}">
        <script type="text/javascript">
                    // TODO Get rid of those global variables
                var ExtDirectInitialization = function() {
                    <ext:extDirect.provider />;
                };
                {neos:backend.javascriptConfiguration()}
                var Aloha,
                    alohaBaseUrl = '{aloha:baseUri()}';
            </script>

            <script src="{f:uri.resource(path: 'JavaScript/ext-direct.js', package: 'TYPO3.ExtJS')}"></script>
            <f:if condition="{neos:backend.shouldLoadMinifiedJavascript()}">
                <f:then>
                    <script src="{f:uri.resource(path: 'JavaScript/require.js', package: 'TYPO3.Neos')}" data-
main="{f:uri.resource(path: 'JavaScript/ContentModule-built.js', package: 'TYPO3.Neos')}"></script>
                    <link rel="stylesheet" type="text/css" href="{f:uri.resource(path: 'Styles/Includes-built.css',
package: 'TYPO3.Neos')}" />
                </f:then>
                <f:else>
                    <script src="{f:uri.resource(path: 'JavaScript/require.js', package: 'TYPO3.Neos')}" data-
main="{f:uri.resource(path: 'JavaScript/ContentModuleBootstrap.js', package: 'TYPO3.Neos')}"></script>
                    <link rel="stylesheet" type="text/css" href="{f:uri.resource(path: 'Styles/Includes.css', package:
'TYPO3.Neos')}" />
                </f:else>
            </f:if>
        </f:if>
    </f:security.ifAccess>
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das TYPO3.Neos:Shortcut Objekt aufgebaut (vererbt von TYPO3.Neos:Template)
- Dieses ist aber im Backend (Stand alpha4) noch nicht benutzbar!

```
# TYPO.Neos.Shortcut is given a representation for editing purposes
#
prototype(TYPO3.Neos:Shortcut) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos:Shortcut) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
Shortcut.html'

    targetMode = ${q(node).property('targetMode')}
    targetNode = ${q(node).property('targetNode')}
    firstChildNode = ${q(node).children('[instanceof TYPO3.Neos:Document]').get(0)
    parentNode = ${q(node).parents().get(0)}
}
prototype(TYPO3.Neos:ContentCollection).shortcut {
    condition = ${q(node).is('[instanceof TYPO3.Neos:Shortcut]')}
    type = 'TYPO3.Neos:Shortcut'
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das TYPO3.Neos:Breadcrumb Objekt aufgebaut (vererbt von TYPO3.Neos:Template), welches eine Breadcrumb erzeugt

```
# TYPO3.Neos:Breadcrumb provides a basic breadcrumb navigation
#
prototype(TYPO3.Neos:Breadcrumb) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos:Breadcrumb) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
BreadcrumbMenu.html'
    items = ${q(node).add(q(node).parents())}
}
```

- **Dazugehöriges Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<f:if condition="{items}">
    <ul class="breadcrumbs">
    <f:for each="{items}" as="item" reverse="TRUE">
        <f:if condition="{item.hiddenInIndex} == 0">
            <li>
                <neos:link.node node="{item}">{item.label}</neos:link.node>
            </li>
        </f:if>
    </f:for>
    </ul>
</f:if>
```

TypoScript: WireframeMode.ts2 (TYPO3.Neos)

- Hier werden nun verschiedene Einstellungen getroffen, die nur für die Neos Wireframe-Mode im „Backend“ notwendig sind
- Dies betrifft vor allem das Markup und die Klassen
- Zusätzlich wird jQuery geladen

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - zuerst kommt das **Menu-Objekt**:

```
# Menu TS Object - extends TYPO3.Neos:Menu and is rendering menus inserted as content elements
prototype(TYPO3.Neos.NodeTypes:Menu) < prototype(TYPO3.Neos:Menu) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/Menu.html'
    entryLevel = ${q(node).property('startLevel')}
    entryLevel << 1.toInteger()
    maximumLevels = ${q(node).property('maximumLevels')}
    maximumLevels << 1.toInteger()
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}" reloadable="TRUE">
    <ul>
        <f:render section="itemsList" arguments="{items: items}" />
    </ul>
</neos:contentElement>

<f:section name="itemsList">
    <f:for each="{items}" as="item">
        <li class="{item.state}">
            <neos:link.node node="{item.node}">{item.label}</neos:link.node>
            <f:if condition="{item.subItems}">
                <ul>
                    <f:render section="itemsList" arguments="{items: item.subItems}" />
                </ul>
            </f:if>
        </li>
    </f:for>
</f:section>
```

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Html-Objekt**:

```
# HTML TS Object
prototype(TYPO3.Neos.NodeTypes:Html) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:Html) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
Html.html'
    source = ${q(node).property('source')}
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <div property="typo3:source">{source -> f:format.raw()}</div>
</neos:contentElement>
```

typovision

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Headline-Objekt**:

```
# Headline TS Object
prototype(TYPO3.Neos.NodeTypes:Headline) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:Headline) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
Headline.html'
    title = ${q(node).property('title')}
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <neos:contentElement.editable property="title">{title -> f:format.raw()}</
neos:contentElement.editable>
</neos:contentElement>
```

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Image-Objekt** (1/2):

```
# Image TS Object
prototype(TYPO3.Neos.NodeTypes:Image) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:Image) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/Image.html'
    image = ${q(node).property('image')}
    alignment = ${q(node).property('alignment')}
    alternativeText = ${q(node).property('alternativeText')}
    title = ${q(node).property('title')}
    link = ${q(node).property('link')}
    hasCaption = ${q(node).property('hasCaption')}
    caption = ${q(node).property('caption')}
    captionAlignment = ${q(node).property('captionAlignment')}
    maximumWidth = 2560
    maximumHeight = 2560
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <div class="typo3-image{f:if(condition: alignment, then: ' typo3-image-alignment-{alignment}')}">
        <f:render partial="Image" arguments="{
            image: image, alternativeText: alternativeText,
            title: title, link: link,
            hasCaption: hasCaption, caption: caption,
            captionAlignment: captionAlignment,
            maximumWidth: maximumWidth,
            maximumHeight: maximumHeight
        }" />
    </div>
</neos:contentElement>
```

typovision

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Image-Objekt** (2/2):
- **Partial HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace media=TYPO3\Media\ViewHelpers}
<figure>
     <f:if condition="{image}">
          <f:then>
               <f:if condition="{link}">
                    <f:then>
                         <a href="{link}"><media:image image="{image}" alt="{alternativeText}" title="{title}"
maximumWidth="{maximumWidth}" maximumHeight="{maximumHeight}" /></a>
                    </f:then>
                    <f:else>
                         <media:image image="{image}" alt="{alternativeText}" title="{title}"
maximumWidth="{maximumWidth}" maximumHeight="{maximumHeight}" />
                    </f:else>
               </f:if>
          </f:then>
          <f:else>
               <img src="{f:uri.resource(package: 'TYPO3.Neos', path: 'Images/dummy-image.png')}" title="Dummy image"
alt="Dummy image" />
          </f:else>
     </f:if>
     <f:if condition="{hasCaption}">
          <figcaption{f:if(condition: captionAlignment, then: ' class="alignment-{captionAlignment}"')}
><neos:contentElement.editable property="caption">{caption -> f:format.raw()}</neos:contentElement.editable></
figcaption>
     </f:if>
</figure>
```

typovision

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Text-Objekt**:

```
# Text TS Object
prototype(TYPO3.Neos.NodeTypes:Text) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:Text) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
Text.html'
    text = ${q(node).property('text')}
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <neos:contentElement.editable property="text">{text -> f:format.raw()}</
neos:contentElement.editable>
</neos:contentElement>
```

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **Text/Bild-Objekt**:

```
# TextWithImage TS Object
prototype(TYPO3.Neos.NodeTypes:TextWithImage) < prototype(TYPO3.Neos.NodeTypes:Image)
prototype(TYPO3.Neos.NodeTypes:TextWithImage) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
TextWithImage.html'
    text = ${q(node).property('text')}
    position = ${q(node).property('position')}
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <div class="typo3-image{f:if(condition: alignment, then: ' typo3-image-alignment-{alignment}')}">
        <f:render partial="Image" arguments="{
            image: image, alternativeText: alternativeText,
            title: title, link: link,
            hasCaption: hasCaption, caption: caption,
            captionAlignment: captionAlignment,
            maximumWidth: maximumWidth,
            maximumHeight: maximumHeight
        }" />
    </div>
    <neos:contentElement.editable property="text">{text -> f:format.raw()}</
neos:contentElement.editable>
</neos:contentElement>
```

typovision

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **MultiColumn-Objekt**:

```
# Basic implementation of a flexible MultiColumn element, not exposed directly but inherited by all
specific MultiColumn content elements
prototype(TYPO3.Neos.NodeTypes:MultiColumn) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:MultiColumn) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
MultiColumn.html'
    layout = ${q(node).property('layout')}
    columns = TYPO3.TypoScript:Collection
    columns {
        collection = ${q(node).children('[instanceof TYPO3.Neos:ContentCollection]')}
        itemRenderer = TYPO3.Neos.NodeTypes:MultiColumnItem
        itemName = 'node'
    }
}
```

- **HTML-Template**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<neos:contentElement node="{node}">
    <div class="container columns-{layout}">
        <ts:render path="columns" />
    </div>
</neos:contentElement>
```

typovision

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **MultiColumnItem-Objekt**:

```
# Abstract render definition for a single content column in a multi column element
prototype(TYPO3.Neos.NodeTypes:MultiColumnItem) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos.NodeTypes:MultiColumnItem) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/TypoScriptObjects/
MultiColumnItem.html'
    columnContent = TYPO3.Neos:ContentCollection
    columnContent {
        nodePath = '.'
    }
}
```

- **HTML-Template**

```
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<div class="column">
    <ts:render path="columnContent" />
</div>
```

TypoScript: Root.ts2 (TYPO3. Neos.NodeTypes)

- Schließlich werden die TS-Objekte konkretisiert - dann kommt das **TwoColumn-, ThreeColumn- und Four-Column-Objekt**:

```
# Two Column TS Object
prototype(TYPO3.Neos.NodeTypes:TwoColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)

# Three Column TS Object
prototype(TYPO3.Neos.NodeTypes:ThreeColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)

# Four Column TS Object
prototype(TYPO3.Neos.NodeTypes:FourColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)
```

TypoScript: Root.ts2 & Template der Site (1/3)

- Nun wird das TypoScript der Site ausgewertet
- Begonnen wird damit mit der Datei Root.ts2 in folgendem Verzeichnis

```
Sites/[Vendor].[Sitename]/Resources/Private/TypoScripts/Library/
```

- Per Default wird dort folgender Code angelegt:

```
/**
 * Root TypoScript template for the Website Site
 */
page = Page
page.head {
    stylesheets = TYPO3.TypoScript:Template
    stylesheets {
        templatePath = 'resource://Typovision.Demo/Private/Templates/Page/Default.html'
        sectionName = 'stylesheets'
    }
    scripts = TYPO3.TypoScript:Template
    scripts {
        templatePath = 'resource://Typovision.Demo/Private/Templates/Page/Default.html'
        sectionName = 'scripts'
    }
}
```

TypoScript: Root.ts2 & Template der Site (2/3)

- Per Default wird dort folgender Code angelegt:

```
...
page.body {
    templatePath = 'resource://Typovision.Demo/Private/Templates/Page/Default.html'
    sectionName = 'body'
    parts {
        menu = Menu
        breadcrumb =  Breadcrumb
    }
    // These are your content areas, you can define as many as you want, just name them
and the nodePath.
    content {
        main = ContentCollection
        main.nodePath = 'main'
    }
}
```

TypoScript: Root.ts2 & Template der Site (3/3)

- Das Default HTML-Template wird wie folgt angelegt:

```
<!DOCTYPE html>
{namespace typo3=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<html>
<head>
    <f:section name="stylesheets">
        <!-- put your stylesheet inclusions here, they will be included in your website by
TypoScript -->
    </f:section>
    <f:section name="scripts">
        <!-- put your scripts inclusions here, they will be included in your website by
TypoScript -->
    </f:section>
</head>
<body>
<f:section name="body">
    <h1>A freshly created template for your new site!</h1>
    <nav class="menu"><ts:render path="parts.menu" /></nav>
    <nav class="breadcrumb"><ts:render path="parts.breadcrumb" /></nav>
    <div class="content"><ts:render path="content.main" /></div>
</f:section>
</body>
</html>
```

Neuaufbau einer

TYPO3 Neos Website

typovision

Startpunkt

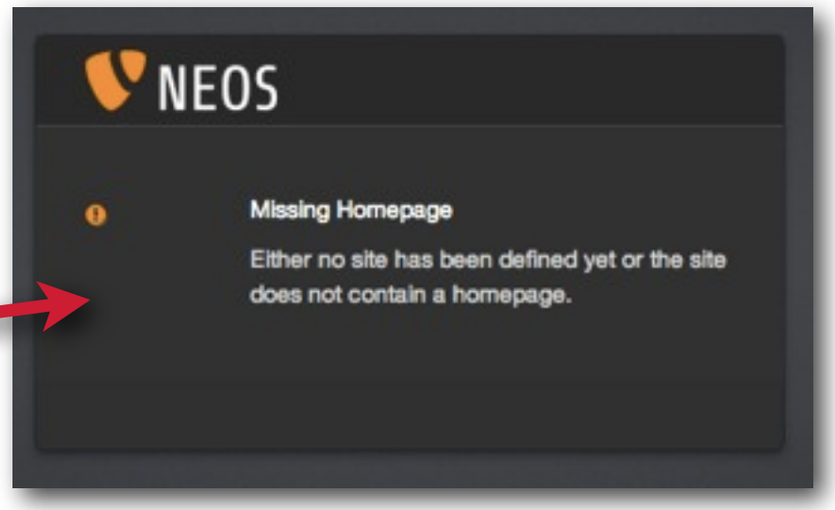
- Entweder installiert man TYPO3 Neos komplett neu
- Oder (wenn man ein Testsystem „zurücksetzen“ will): Man löscht die Datenbank und legt diese erneut an
- Anschließend führt man den folgenden Befehl aus, der alle notwendigen Datenbank-Tabellen (und Daten) wieder herstellt:

```
./flow doctrine:migrate
```

- Schließlich sollte man den Cache löschen

```
./flow flow:cache:flush
```

- Ein Aufruf im Frontend sollte folgendes Ergebnis bringen:



typovision

User anlegen

- Damit man sich später einloggen kann, brauchen wir einen Benutzer
- Diesen legen wir über die Kommandozeile an

- `./flow user:create --roles Administrator admin password firstname lastname`

- SYNTAX:
  ./flow user:create [<options>] <username> <password> <first name> <last name>

  ARGUMENTE:
  --username Username des Users, der erstellt wird
  --password Passwort des Users, der erstellt wird
  --first-name Vorname des Users, der erstellt wird
  --last-name Nachname des Users, der erstellt wird

  OPTIONEN:
  --roles Kommaseparierte Liste der Rollen, die der User bekommen soll

Site anlegen

- Ein Website kann auf zwei Arten initialisiert werden
  - Über den Site-Mangager innerhalb von Neos
  - Über die TYPO3 Flow Kommandozeile

```
cd /pfad/zur/Neos-Installation/
./flow site:kickstart Typovision.Demo Website
./flow site:import --package-key Typovision.Demo
```

- Syntax zum Anlegen

  `./flow site:kickstart <PackageKey> <SiteName>`

- Syntax zum Importieren der Site in TYPO3 Neos

  `./flow site:import --package-key <PackageKey>`

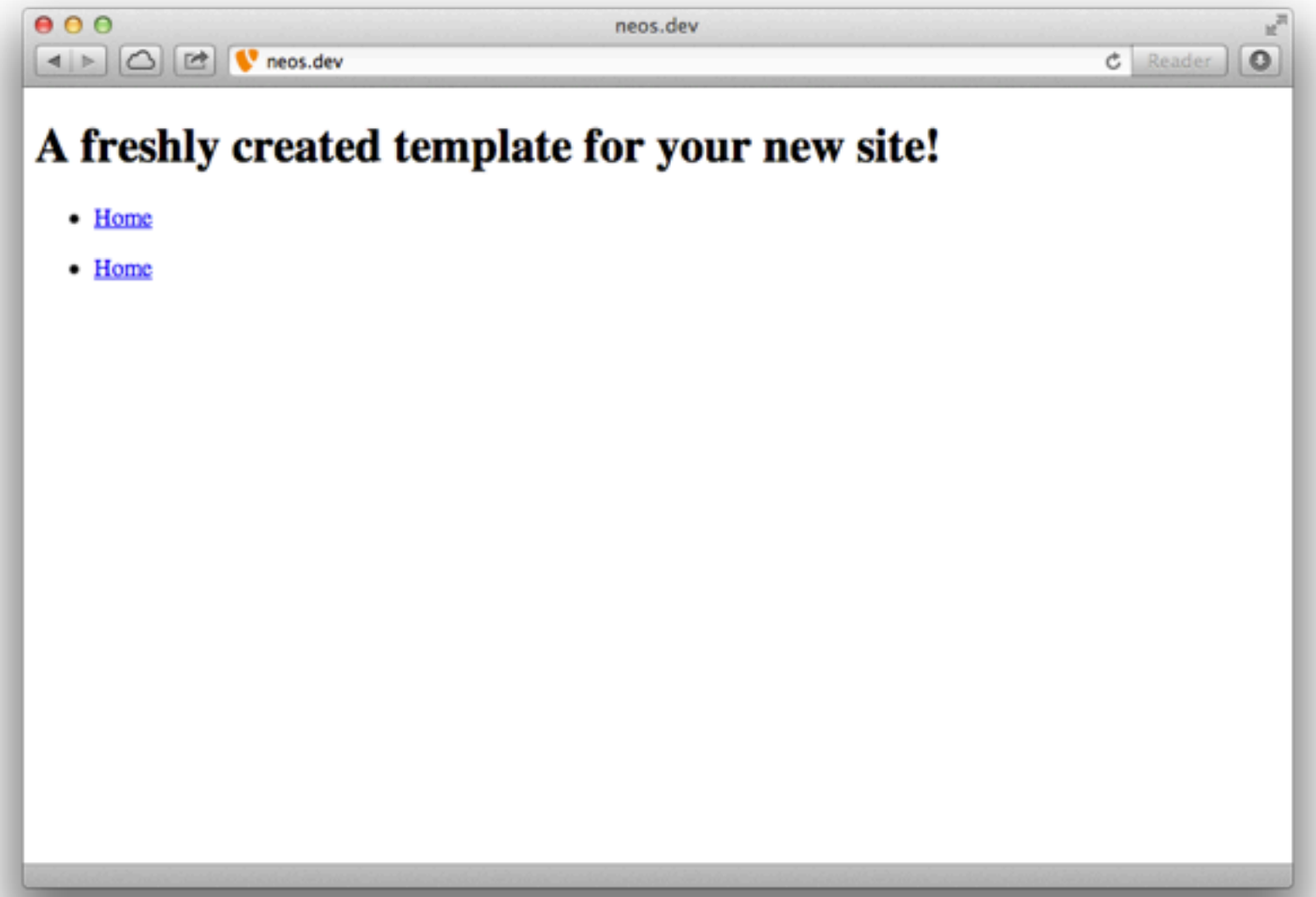
  (Import führt zu einem Datensatz in der Tabelle `typo3_neos_domain_model_site`)

- Syntax zum Anzeigen aller Sites

  `./flow site:list`

Site anlegen - Frontend

- Ein Auruf der Website sollte dann folgenden Output liefern:



typovision

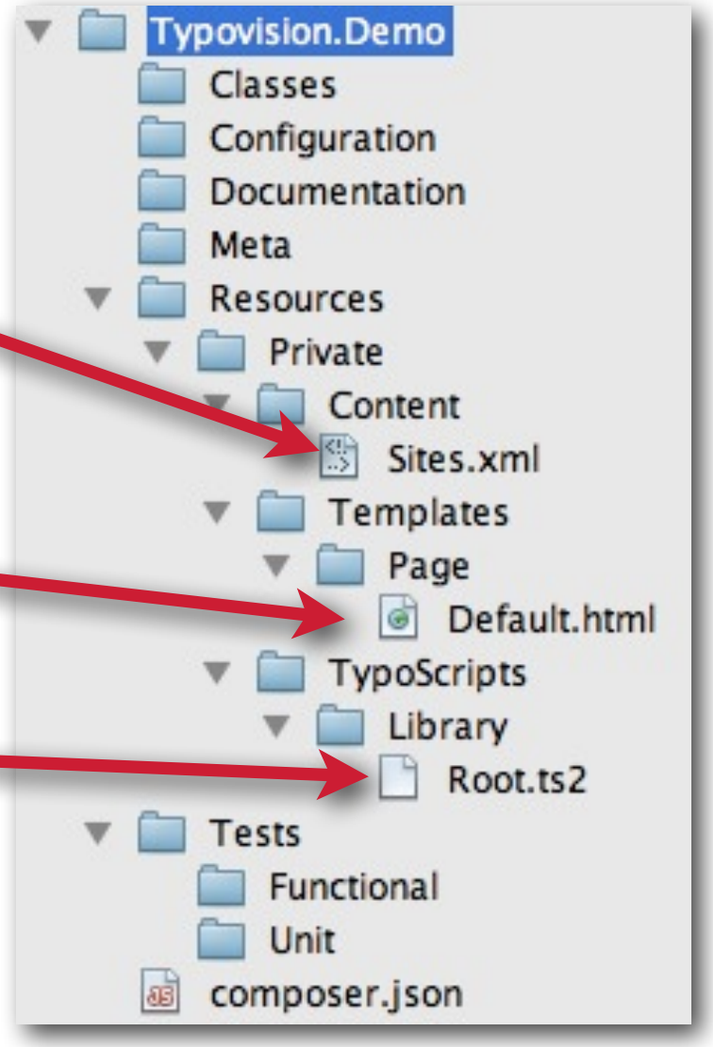
Site anlegen - angelegte Dateien

- Nun wird im Verzeichnis `Packages/Sites` folgende Struktur inkl. 3 Dateien angelegt

- `Sites.xml`
  Enthält die Root-Node Definition der Website

- `Default.html`
  Standard-Template

- `Root.ts2`
  Default TypoScript Code

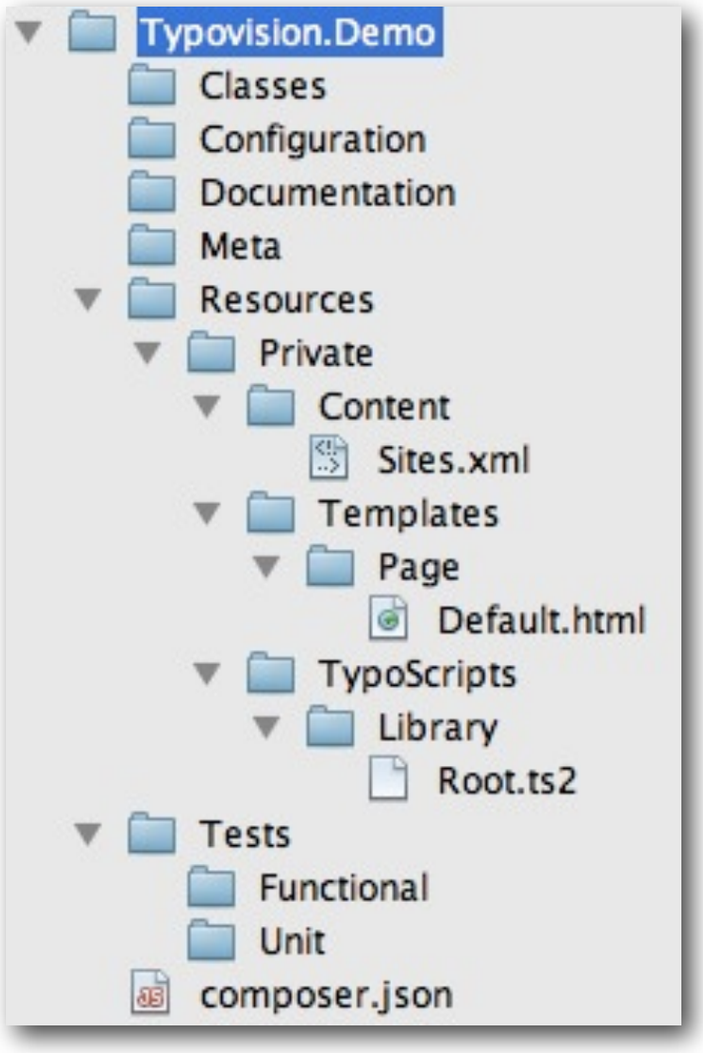


typovision

Site anlegen - angelegte Dateien - Sites.xml

- Die Sites.xml enthält die Root-Node Definition

- ```
  <?xml version="1.0" encoding="UTF-8"?>
  <root>
     <site nodeName="demo">
        <properties>
           <name>Website</name>
           <state>1</state>
           <siteResourcesPackageKey>Typovision.Demo
           </siteResourcesPackageKey>
        </properties>
        <node identifier="" type="TYPO3.Neos:Page"
           nodeName="homepage" locale="en_EN">
           <properties>
               <title>Home</title>
           </properties>
        </node>
     </site>
  </root>
  ```


  ```

typovision

Site anlegen - MultiSite

- Wenn das System nun lediglich eine Site enthält, kann man diese bereits über die URL aufrufen
- Sind jedoch mehrere Site enthalten, so müssen diese mit Hilfe von Domain-Einträgen konfiguriert werden
- Zunächst ermitteln Sie mittels `./flow site:list` die Liste aller Sites

```
pats-macbook-air-2:TYPO3-Neos patricklobacher$ ./flow site:list

 Name                    Node name               Resources package
------------------------------------------------------------------
 Website2                test                    Typovision.Test
 Website                 demo                    Typovision.Demo
```

Site anlegen - MultiSite

- Nun fügen Sie für jeden NodeName (deren Site Sie später ansprechen wollen) einen Domain-Eintrag hinzu

```
./flow domain:add test neos.dev
```

- Syntax
  `./flow domain:add <node name> <domain>`
- Auflisten kann man die Domains mittels `./flow domain:list`
- Löschen Sie anschließend den Cache
  `./flow flow:cache:flush --force`

Minimales TypoScript

- Minimal benötigt wird folgendes TypoScript - dieses sorgt dafür, dass die Template-Datei eingelesen und als (leere) HTML-Website ausgegeben wird

```
page = Page
page.body. templatePath =
        'resource://Typovision.Demo/Private/Templates/Page/Default.html'
```

- Nun kann man im Template z.B. `{title}` verwenden - dies wird durch den Titel der aktuellen Seite ersetzt
- Oder man verwendet den ViewHelper `<ts:render path="parts.menu" />`, welcher dafür sorgt, dass der TypoScript-Pfad `page.body.parts.menu` gerendert wird.

typovision

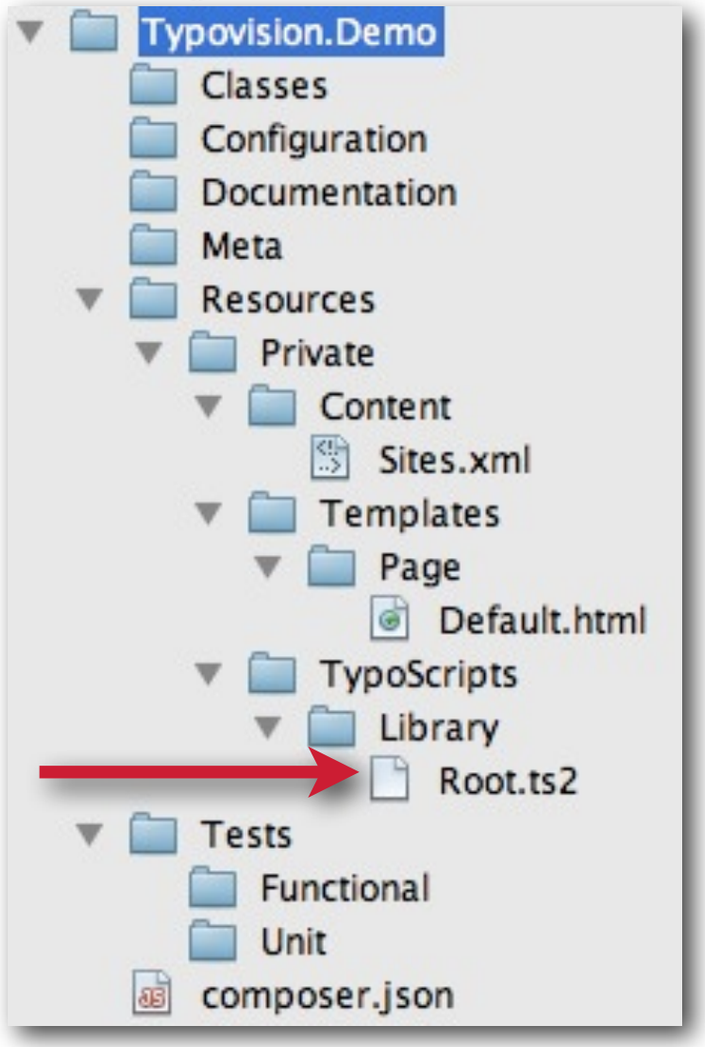
Analyse Root.ts2

- Die Datei Root.ts2 (TypoScript 2.0) wird nun als erstes geparst und ausgeführt
- Zuerst wird ein Objekt mit dem Namen „page“ erstellt, welchem das Page-Objekt zugewiesen wird

```
page = Page
```

- Das Objekt Page hat unter anderem zwei Eigenschaften:
  - head (vom Typ TYPO3.TypoScript:Array)
  - body (vom Typ TYPO3.Neos:Template)

- Diese wurden in `TYPO3.Neos/Resources/Private/TypoScript/DefaultTypoScript.ts2` festgelegt



typovision

Analyse Root.ts2 - Header-Bereich

- Als erstes wird der Header-Bereich der Website gefüllt, einmal mit den Stylesheets und einmal mit den benötigten JavaScripts.
- Dazu wird in beiden Fällen das Template geladen und die entsprechende Sektion angesprungen

```
page.head {

    stylesheets = TYPO3.TypoScript:Template
    stylesheets {
        templatePath = 'resource://Typovision.Demo/Private/Templates/Page/
Default.html'
        sectionName = 'stylesheets'
    }
    scripts = TYPO3.TypoScript:Template
    scripts {
        templatePath = 'resource://Typovision.Demo/Private/Templates/Page/
Default.html'
        sectionName = 'scripts'
    }
}
```

typovision

Analyse Root.ts2 - Header-Bereich - Template

- Im Template (`resource://Typovision.Demo/Private/Templates/Page/Default.html`) sieht das dann wie folgt aus:

```
<head>
    <f:section name="stylesheets">
        <!-- put your stylesheet inclusions here, they will be included in
your website by TypoScript -->
    </f:section>
    <f:section name="scripts">
        <!-- put your scripts inclusions here, they will be included in your
website by TypoScript -->
    </f:section>
</head>
```

- Ausgegeben wird dann letztendlich nur der Code, der zwischen den beiden `<f:section>`-ViewHelpern steht. Das `<head>`-Tag wird von TYPO3 Neos selbst ausgegeben (aufgrund von `<ts:render path="head" />` in `TYPO3.Neos/Private/Tamplates/TypoScriptObjects/PageTemplate.html`)

typovision

Analyse Root.ts2 - Body-Bereich & Template

- Weiter geht es mit der Befüllung des <body>-Bereichs - dafür wird ein Template geladen und dort die Section „body“ aufgerufen:

```
page.body {
    templatePath =
            'resource://Typovision.Demo/Private/Templates/Page/Default.html'
    sectionName = 'body'
```

- Das zugehörige Template an dieser Stelle sieht wie folgt aus:

```
<body>
<f:section name="body">
    ...
</f:section>
</body>
```

- Auch hier wird nur das gerendert, was sich zwischen dem `<f:section>`-ViewHelper befindet

Analyse Root.ts2 - Body-Bereich & Template

- Innerhalb von `page.body` sind nun zwei Objekte definiert - einerseits `menu` vom Objekttyp `Menu` und andererseits `breadcrumb` vom Objekttyp `Breadcrumb`:

```
page.body {
    parts {
        menu = Menu
        breadcrumb =  Breadcrumb
    }
```

- Im Template greift man dann über den `<ts:render>`-ViewHelper und dem relativen Pfad `parts.menu` bzw. `parts.breadcrumb` zu:

```
<h1>A freshly created template for your new site!</h1>
<nav class="menu"><ts:render path="parts.menu" /></nav>
<nav class="breadcrumb"><ts:render path="parts.breadcrumb" /></nav>
<div class="content"><ts:render path="content.main" /></div>
```

Analyse Root.ts2 - Body-Bereich & Template - Menu

- Für den Rendern des Menüs via `menu = Menu` ist letzlich das TypoScript unter `TYPO3.Neos/Private/Templates/TypoScript/DefaultTypoScript.ts2` und in Folge dessen das Template `templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/Menu.html'` zuständig:

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}" reloadable="TRUE">
    <ul>
        <f:render section="itemsList" arguments="{items: items}" />
    </ul>
</neos:contentElement>
<f:section name="itemsList">
    <f:for each="{items}" as="item">
    <li class="{item.state}">
        <neos:link.node node="{item.node}">{item.label}</neos:link.node>
        <f:if condition="{item.subItems}">
            <ul>
                <f:render section="itemsList" arguments="{items: item.subItems}" />
            </ul>
        </f:if>
    </li>
    </f:for>
</f:section>
```

typovision

Analyse Root.ts2 - Body-Bereich & Template - Menu

- Das TypoScript für den Prototyp Menu sieht wie folgt aus:

```
prototype(TYPO3.Neos:Menu) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos:Menu) {
    @class = 'TYPO3\\Neos\\TypoScript\\MenuImplementation'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
Menu.html'
}
```

- Die Menu Klasse sieht (auszugsweise) wie folgt aus:

```
class MenuImplementation extends \TYPO3\TypoScript\TypoScriptObjects
\TemplateImplementation {

    const MAXIMUM_LEVELS_LIMIT = 100;

    protected $templatePath = 'resource://TYPO3.Neos/Private/Templates/
TypoScriptObjects/Menu.html';
    /* 1=1.Level, 2=2.Level, 0=Aktuelles Level, -1=Elternlevel
    protected $entryLevel = 1;
    protected $lastLevel;
    protected $maximumLevels = 1;
    protected $items;
    protected $startingPoint;
    protected $currentNode;
```

typovision

Analyse Root.ts2 - Body-Bereich & Template - Breadcrumb

- Für den Rendern des Menüs via `menu = Menu` ist letzlich das TypoScript unter `TYPO3.Neos/Private/Templates/TypoScript/DefaultTypoScript.ts2` und in Folge dessen das Template `templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/BreadcrumbMenu.html'` zuständig:

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<f:if condition="{items}">
    <ul class="breadcrumbs">
    <f:for each="{items}" as="item" reverse="TRUE">
        <f:if condition="{item.hiddenInIndex} == 0">
            <li>
                <neos:link.node node="{item}">{item.label}</neos:link.node>
            </li>
        </f:if>
    </f:for>
    </ul>
</f:if>
```

typovision

Analyse Root.ts2 - Body-Bereich & Template - Breadcrumb

- Das TypoScript für den Prototyp Breadcrumb sieht wie folgt aus:

```
prototype(TYPO3.Neos:Breadcrumb) < prototype(TYPO3.Neos:Template)
prototype(TYPO3.Neos:Breadcrumb) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
BreadcrumbMenu.html'
    items = ${q(node).add(q(node).parents())}
}
```

- Hier gibt es keine eigene Menü-Klasse in PHP

typovision

Analyse Root.ts2 - Body-Bereich & Template

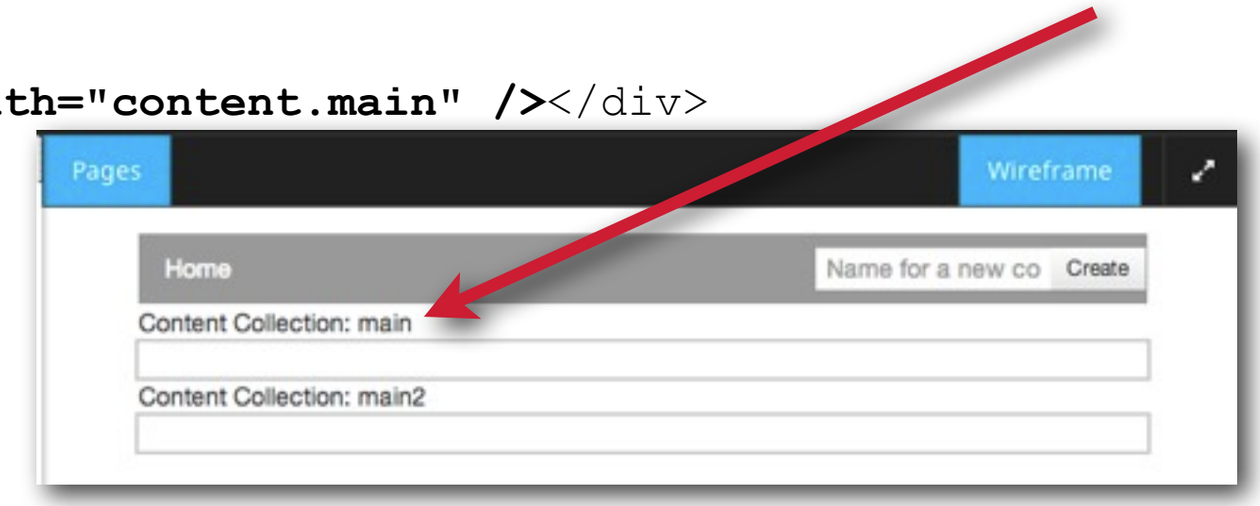
- Innerhalb von `page.body` wird nun auch noch das Content-Objekt definiert:

```
page.body {
    content {
        main = ContentCollection
        main.nodePath = 'main'
    }
```

- Im Template greift man dann ebenfalls über den `<ts:render>`-ViewHelper und dem relativen Pfad `content.main` zu:

```
<div class="content"><ts:render path="content.main" /></div>
```

- Die erste Content Collection wird automatisch mit **`main`** benannt - alle weitere kann man selbst festlegen.



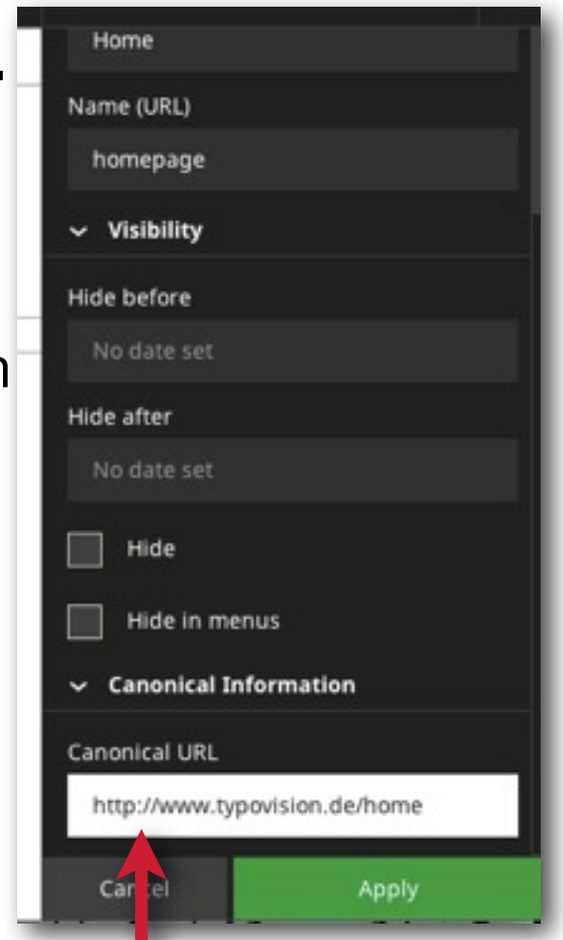
typovision

Erweiterung der Site

Einfaches Plugin 1

typovision

Plugin: Meta-Tag zur Angabe einer Canonical URL



- **Ziel:** TYPO3 Neos soll so erweitert werden, dass es möglich ist, im Backend eine Canonical URL anzugeben, welche dann wie folgt im Frontend gerendert wird:

```
<link rel="canonical" href="http://www.typovision.de/canonical" />
```

- Dafür benötigen wir ein Flow Package, welches wir als Neos Plugin verwenden

- **Schritt 1: Anlegen eines Flow Package** - der Vendor ist hier „Typovision“, der Name des Packages ist „Canonical“

```
./flow kickstart:package Typovision.Canonical
```

typovision

Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Schritt 2: Zufügen einer Datei „NodeTypes.yaml“** in `Packages/Application/Typovision.Canonical/Configuration/`

```
# Hinzufuegen des Feldes zu den Seiteneigenschaften
'TYPO3.Neos:Page':
  superTypes:
    - 'TYPO3.Neos:Document'
  ui:
    inspector:
      groups:
        metaOptions:
          label: 'Canonical Information'
          priority: 2
  properties:
    canonicalUrl:
      type: string
      ui:
        label: 'Canonical URL'
        inspector:
          group: metaOptions
```

WICHTIG:
Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Schritt 3: Anlegen des TypoScript Verzeichnisses unterhalb von `Resources`**

```
mkdir Packages/Application/Typovision.Canonical/Resources/Private/TypoScripts
```

- **Schritt 4: Dort wird die Datei `Root.ts2` angelegt, welche den Head-Bereich der Website mit einem Template erweitert, in welchem die CanonicalUrl angezeigt wird**

```
page.head {
    canonicalTag = Template
    canonicalTag {
        templatePath = 'resource://Typovision.Canonical/Private/Templates/
TypoScript/CanonicalTag.html'
        canonicalUrl = TYPO3.TypoScript:Value
        canonicalUrl.value = ${q(node).property('canonicalUrl')}
    }
}
```

Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Schritt 5: Anlegen des TypoScript Verzeichnisses unterhalb von Templates**

```
mkdir Packages/Application/Typovision.Canonical/Resources/Private/Templates/
TypoScript
```

- **Schritt 6: Dort wird die Datei `CanonicalTag.html` angelegt**

```
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<link rel="canonical" href="{ts:render(path: 'canonicalUrl')}" />
```

- **Übung für den Leser: Das Tag nur dann ausgeben lassen, wenn auch eine Canonical URL eingegeben wurde**

Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Schritt 7: Einfügen des TypoScripts in der Datei `Packages/Sites/Typovision.Demo/Resources/Private/TypoScripts/Library/Root.ts2`**

```
include: resource://Typovision.Canonical/Private/TypoScripts/Root.ts2
```

- **Hintergrundwissen:**

  Die Seiteneigenschaften werden in der Tabelle
  `typo3_typo3cr_domain_model_nodedata`
  im Feld
  `properties`
  gespeichert:
  `a:2:{s:5:"title";s:4:"Home";s:12:"canonicalUrl";s:29:"http://www.typovision.de/home";}`

Erweiterung der Site

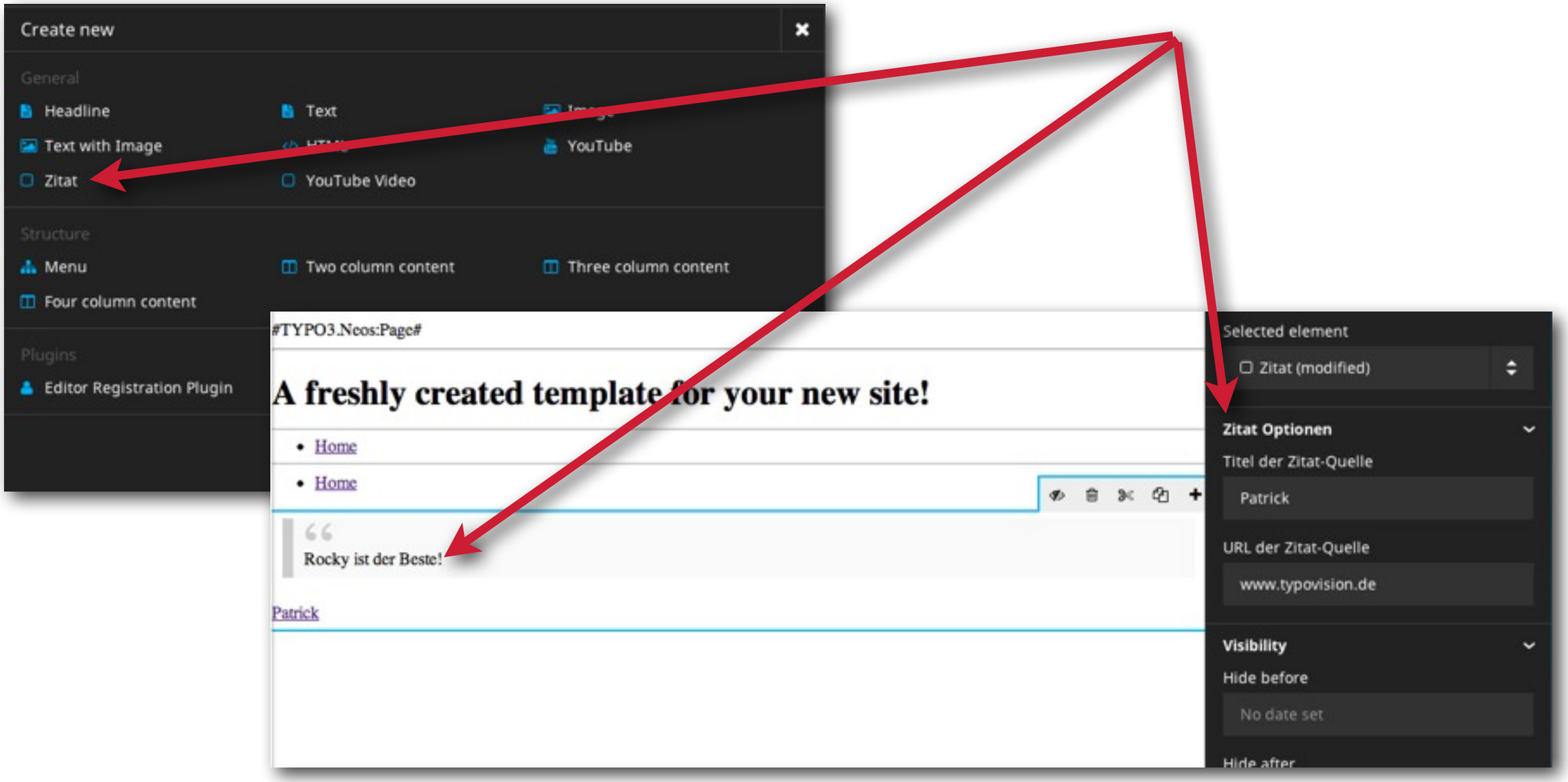
Einfaches Plugin 2

Plugin: Quote-FCE (Flexible Content Element)

- **Ziel:** TYPO3 Neos soll so erweitert werden, dass es möglich ist, ein Zitat-Content-Element einzufügen - bestehend aus einem Zitat-Text (Inline editierbar), einer Quelle und einer URL (beide über den Inspektor pflegbar)
- Dafür verwenden wir ein Flow Package, welches wir als Neos Plugin verwenden
- **Schritt 1: Anlegen eines Flow Package** - der Vendor ist hier „Typovision", der Name des Packages ist „Quote"

```
./flow kickstart:package Typovision.Quote
```

Plugin: Quote-FCE (Flexible Content Element)



Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 2: Zufügen einer Datei „`NodeTypes.yaml`“** in `Packages/Application/Typovision.Quote/Configuration/`

```
'Typovision.Quote:Quote':
  superTypes: ['TYPO3.Neos:Content']
  ui:
    label: 'Zitat'
    group: 'General'
    inspector:
      groups:
        quoteproperties:
          label: 'Zitat Optionen'
          position: 5
```

- **Weiter auf nächster Seite...**

WICHTIG:
Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 2: ...weiterer Inhalt („properties“ ist 2 Leerzeichen eingerückt):**

```
properties:
  blockquote:
    type: string
    ui:
      label: 'Zitat'
      inlineEditable: true
      reloadIfChanged: true
sourcetitle:
  type: string
  ui:
    label: 'Titel der Zitat-Quelle'
    inspector:
      group: 'quoteproperties'
    defaultValue: 'Titel der Zitat-Quelle'
    reloadOnChange: true
```

WICHTIG:
Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 2: ...weiterer Inhalt („sourceurl“ ist 4 Leerzeichen eingerückt):**

```
    sourceurl:
      type: string
      ui:
        label: 'URL der Zitat-Quelle'
        inspector:
          group: 'quoteproperties'
        reloadOnChange: true
groups:
  quoteproperties:
    ui:
      label: 'Zitat Optionen'
      priority: 10
inlineEditableProperties: ['blockquote']
```

WICHTIG:
Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 3: Anlegen des TypoScript Verzeichnisses unterhalb von `Resources`**

```
mkdir Packages/Application/Typovision.Quote/Resources/Private/TypoScripts
```

- **Schritt 4: Dort wird die Datei `Root.ts2` angelegt, welche das Quote-TS-Objekt definiert:**

```
# Quote TypoScript Object
prototype(Typovision.Quote:Quote) < prototype(TYPO3.Neos:Template)
prototype(Typovision.Quote:Quote) {
    templatePath = 'resource://Typovision.Quote/Private/Templates/
TypoScriptObjects/Quote.html'
    blockquote = ${q(node).property('blockquote')}
    sourceurl = ${q(node).property('sourceurl')}
    sourcetitle = ${q(node).property('sourcetitle')}
}
```

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 5: Anlegen des TypoScriptObjects Verzeichnisses unterhalb von Templates**

  `mkdir Packages/Application/Typovision.Quote/Resources/Private/Templates/TypoScriptObjects`

- **Schritt 6: Dort wird die Datei `Quote.html` angelegt**

```
{namespace t=TYPO3\Neos\ViewHelpers}
<t:contentElement node="{node}">
   <blockquote cite="{f:uri.external(uri: '{sourceurl}', defaultScheme: 'http')}">
      <t:contentElement.editable property="blockquote">
        {blockquote}
      </t:contentElement.editable>
   </blockquote>
   <f:if condition="{sourceurl}">
      <f:then>
         <f:link.external uri="{sourceurl}" defaultScheme="http">{sourcetitle}</
f:link.external>
      </f:then>
      <f:else>
         {sourcetitle}
      </f:else>
   </f:if>
</t:contentElement>
```

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 7: Einfügen des TypoScripts in der Datei `Packages/Sites/Typovision.Demo/Resources/Private/TypoScripts/Library/Root.ts2`**

```
include: resource://Typovision.Quote/Private/TypoScripts/Root.ts2
```

- **Schritt 7: Anlegen von zwei Verzeichnissen in der Demo-Site**

```
mkdir Packages/Sites/Typovision.Demo/Resources/Public
mkdir Packages/Sites/Typovision.Demo/Resources/Public/Stylesheets
```

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 8: Anlegen einer Datei Quote.css im Verzeichnis:**
  `Packages/Sites/Typovision.Demo/Resources/Public/Stylesheets/`

```
blockquote {
    background:#f9f9f9;
    border-left:10px solid #ccc;
    margin:1.5em 10px;
    padding:.5em 10px;
    quotes:"\201C""\201D""\2018""\2019";
}
blockquote:before {
    color:#ccc;
    content:open-quote;
    font-size:4em;
    line-height:.1em;
    margin-right:.25em;
    vertical-align:-.4em;
}
blockquote p {
    display:inline;
}
```

Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 9: Referenz auf das Stylesheet in der Datei:**
  `Packages/Sites/Typovision.Demo/Resources/Private/Templates/Page/Default.html`

```
<f:section name="stylesheets">
    <!-- put your stylesheet inclusions here, they will be included in your
website by TypoScript -->

    <link rel="stylesheet" href="../../../Public/Stylesheets/Quote.css"
media="all" />

</f:section>
```

Erweiterung der Site

Einfaches Plugin 3

typovision

Plugin: Youtube-Content Plugin

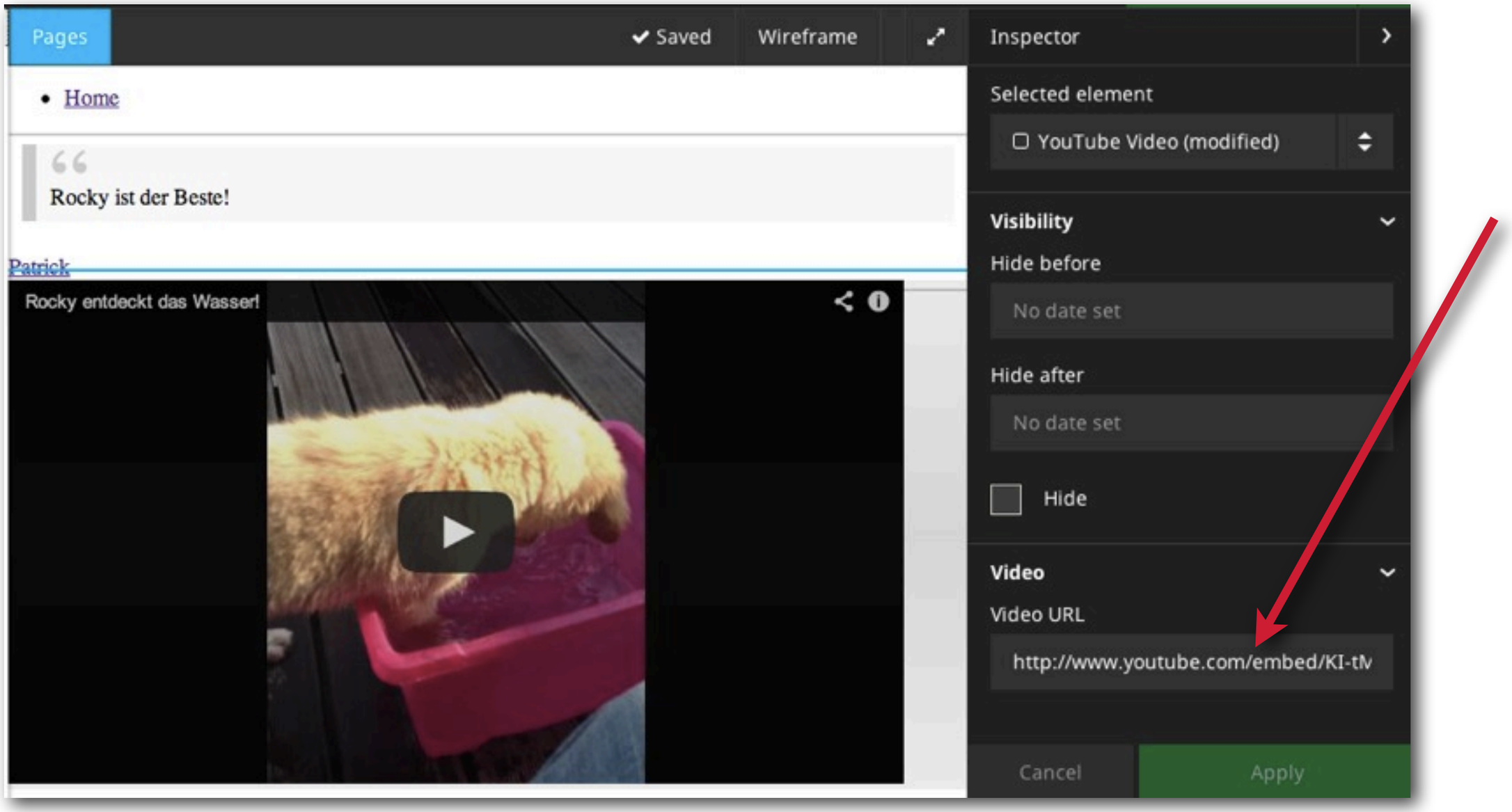
- **Ziel:** TYPO3 Neos soll so erweitert werden, dass es möglich ist, im Backend Content-Element einzufügen, welches eine YouTube-URL aufnimmt und dann einen Video-Player zur Verfügung stellt

- Dafür verwenden wir ein Flow Package, welches wir als Neos Plugin verwenden

- **Schritt 1: Anlegen eines Flow Package** - der Vendor ist hier „Typovision“, der Name des Packages ist „Youtube“

```
./flow kickstart:package Typovision.Youtube
```

Plugin: Youtube-Content Plugin



typovision

Plugin: Youtube-Content Plugin

- **Schritt 2: Zufügen einer Datei „NodeTypes.yaml“** in `Packages/Application/Typovision.Youtube/Configuration/`

```
'Typovision.Youtube:YouTube':
  superTypes: ['TYPO3.Neos:Content']
  ui:
    group: 'General'
    label: 'YouTube Video'
      inspector:
        groups:
          video:
            label: 'Video'
  properties:
    videoUrl:
      type: string
      ui:
        label: 'Video URL'
        inspector:
          group: 'video'
          reloadIfChanged: TRUE
```

WICHTIG:
Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Youtube-Content Plugin

- **Schritt 3: Anlegen des TypoScript Verzeichnisses unterhalb von `Resources`**

```
mkdir Packages/Application/Typovision.Youtube/Resources/Private/TypoScripts
```

- **Schritt 4: Dort wird die Datei `Root.ts2` angelegt, in welcher das Youtube-Content-Element deklariert wird:**

```
prototype(Typovision.Youtube:YouTube) < prototype(TYPO3.Neos:Template) {
    templatePath = 'resource://Typovision.Youtube/Private/Templates/
TypoScriptObjects/YouTube.html'
    videoUrl = ${q(node).property('videoUrl')}
    width = '640'
    height = '360'
}
```

Plugin: Youtube-Content Plugin

- **Schritt 5: Anlegen des TypoScriptObjects Verzeichnisses unterhalb von Templates**

```
mkdir Packages/Application/Typovision.Youtube/Resources/Private/Templates/
TypoScriptObjects
```

- **Schritt 6: Dort wird die Datei `Youtube.html` angelegt**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
<neos:contentElement node="{node}">
    <iframe width="{width}" height="{height}" src="{videoUrl}" frameborder="0"
allowfullscreen></iframe>
</neos:contentElement>
```

typovision

Neos

Kommandozeile

Neos Kommandozeile - Domain

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:domain:add` (Fügt eine Domain hinzu)**
- Argumente
  - **`--site-node-name`**
    Node-Name der Root-Node z.B. `neostypo3org`
  - **`--host-pattern`**
    Host-Pattern z.B. `neos.typo3.org`

- **`typo3.neos:domain:delete` (Löscht eine Domain)**
- Argumente
  - **`--host-pattern`**
    Host-Pattern, welcher entfernt werden soll z.B. `neos.typo3.org`

- **`typo3.neos:domain:list` (Listet alle Domains auf)**
- Argumente
  - **`--host-pattern`**
    Optionales Host-Pattern für die Suche z.B. `neos.typo3.org`

Neos Kommandozeile - Site

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:site:export`** **(Exportieren einer Site in ein XML Format)**
- Argumente
  - **`--site-name`**
    Name der Site, die exportiert werden soll - gibt man nichts an, werden alle Site exportiert

- **`typo3.neos:site:import`**
- Argumente
  - **`--package-key`**
    Package-Key welcher den Seiten-Inhalt importiert bekommen soll
  - **`--file-name`**
    Dateinamen der XML-Datei, die den Inhalt hat

Neos Kommandozeile - Site

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:site:list` (Auflisten aller Sites)**
- Argumente
  - **keine**

- **`typo3.neos:site:prune` (Löschen des Inhalts einer Site)**
- Optionen
  - **`--confirmation`**
    Frägt vor dem Löschen, ob man dies wirklich durchführen will

typovision

Neos Kommandozeile - User/Role

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:user:addrole` (Einem User eine Rolle hinzufügen)**
- Argumente
  - **`--username`**
    Username zu dem man die Rolle zufügen will
  - **`--role`**
    Rolle, die man zum User zufügen will: "TYPO3.Neos:Editor" oder "TYPO3.Neos:Administrator"

- **`typo3.neos:user:removerole` (Eine User eine Rolle entfernen)**
- Argumente
  - **`--username`**
    Username, dessen Rolle man entfernen will
  - **`--role`**
    Rolle, die man entfernen will: "TYPO3.Neos:Editor" oder "TYPO3.Neos:Administrator"

typovision

Neos Kommandozeile - User

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:user:create` (Einen User anlegen)**
  - Argumente
    - **`--username`**
      Username
    - **`--password`**
      Password
    - **`--first-name`**
      Vorname
    - **`--last-name`**
      Nachname
  - Optionen
    - **`--roles`**
      Kommaseparierte Liste von Rollen, die der User bekommen soll

typovision

Neos Kommandozeile - User

- Die CLI (Command line interface) Kommandos werden immer zusammen mit `./flow` ausgeführt - also z.B. `./flow typo3.neos:domain:add ...`

- **`typo3.neos:user:setpassword` (Passwort für einen User festlegen)**
- Argumente
  - **`--username`**
    Username, für den man das Passwort setzen will
  - **`--password`**
    Das neue Passwort

Mitarbeit

bei TYPO3 Neos

typovision

We need you!!

- Komme ins TYPO3 Neos und Flow Team!
- Jeder wird gebraucht: Programmier, Architekten, UX-Experten, Designer, Doku, Tester, JS-Spezialisten, ...
- Meldet Euch bei robert[AT]typo3.org

typovision

Kunden gesucht!

- **Direkte Mitarbeit**
  - Gebe frühes Feedback zu neuen Funktionen, User Interfaces, ...
  - Direkter Kontakt und Diskussion mit den Entwickler
  - Sofern sinnvoll, wird das Feedback unmittelbar umgesetzt

- **Vorteile**
  - Gestalte das WCMS der Zukunft direkt mit
  - Nimm Einfluss auf die Entwicklung
  - Zugang zu Entwicklern, Architekten und UX'lern
  - Networking mit anderen Neos-Kunden

- **Kontakt**
  - rasmus[at]typo3.org

typovision

Quellen und **Informationen**

typovision

Quellen und Informationen

- **TYPO3 Neos Website**
  `http://neos.typo3.org/`

- **TYPO3 Neos Download**
  `http://neos.typo3.org/download.html`

- **TYPO3 Flow Website**
  `http://flow.typo3.org/`

- **TYPO3 Neos Dokumentation**
  `http://docs.typo3.org/neos/TYPO3NeosDocumentation/Index.html`

- **TYPO3 Flow Dokumentation**
  `http://flow.typo3.org/documentation.html`

Quellen und Informationen

- **TYPO3 Neos Projekt bei forge**
  `http://forge.typo3.org/projects/typo3neos`

- **TYPO3 Flow Projekt bei forge**
  `http://forge.typo3.org/projects/flow3`

- **Cheatsheet für Fluid (und demnächst für Flow)**
  `http://www.typovision.de/de/kompetenzen/typo3/`

typovision

Quellen und Informationen

- **GitHub Account von Lelesys (Pankaj Lele / http://www.lelesys.com/)**
  `https://github.com/lelesys`

typovision

Fragen?

Oder komplett verwirrt?

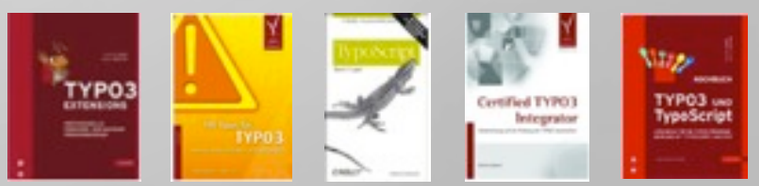
typovision

Patrick Lobacher

Geschäftsführer typovision GmbH

- 42 Jahre, glücklich verheiratet, wohnhaft in München
- Autor von 9 Fachbüchern und > 40 Fachartikeln zum Thema TYPO3 und Webentwicklung
- Selbständig im Bereich Webentwicklung seit 1994
- Certified TYPO3 Integrator seit 2009
- Mitglied bis 2012 in den TYPO3 Core-Teams: Certification, Extbase und Content Editorial
- Mitglied bis 2012 im Expert Advisory Board der TYPO3 Association (EAB)
- Mit-Organisator des TYPO3camp Munich 2008-2013 und der T3DD12
- Speaker auf nationalen und internationalen Kongressen
- Dozent für führende Schulungsinstitute und die MVHS

Veröffentlichungen:



typovision

typovision

- Münchner Fullservice-Agentur für digitale Kommunikation
- 39 festangestellte Mitarbeiter (+ 14 aus festem Freelancer Pool)
- Geschäftsführer: Sebastian Böttger, Patrick Lobacher
- Hochspezialisiert auf TYPO3 seit 11 Jahren und Solr

TYPO3 PLATINUM MEMBER

- **Agenturpräsentation unter: www.typovision.de/dieagentur**

- Über 400 TYPO3-Projekte jeglicher Größenordnung - für Kunden wie:

FTI TOURISTIK Einfach FairReisen · vodafone · SwissLife · arvato BERTELSMANN · MÖVENPICK · SCOUT 24 · apotheken.de Immer für Sie da!

Vielen Dank für Eure Aufmerksamkeit