Single Page Webapps

Web Engineering I

ANGULARJS by Google

BACKBONE.JS

Inhalt

- Einleitung
- Was sind Single Page Webapps?
- Web Technologien
- AngularJS by Google
- Backbone.js
- Vergleich
- andere Frameworks
- Alternativen zu Single Page Webapps
- Fazit

Einleitung

Was sind Single Page Webapps?

Welche Frameworks gibt es?

Welche Alternativen gibt es zu Single Page Webapps?

Wann sollen welche Frameworks verwendet werden?

Was sind Single Page Webapps?

- Webseiten, die aus **einem HTML-Dokument** bestehen
- **dynamisches nachladen** von Komponenten
- trotzdem mehrere **URL‘s**
- **intuitive Bedienung** durch mehr angezeigten Inhalt auf einer Seite (scrollen statt klicken)
- Realisierung nur mithilfe von **Frameworks**

facebook

VICE
RIDECHANNEL
GOOGLEDEVELOPERS
YouTube
App Engine Office Hours
YouTube Developers Live
Google Maps Developers Office Hours
THRASHERMAGAZINE
SURFER

code school

COURSES SCREENCASTS DISCUSS ENROLL CREATE FREE ACCOUNT SIGN IN

Learn By Doing

No setup. No hassle. Just learning.

Code School teaches web technologies in the comfort of your browser with video lessons, coding challenges, and screencasts.

VIEW OUR COURSES

LEARN: GIT BACKBONE.JS SASS RAILS JQUERY

Learn where to start and what to take next with **Paths**

Ruby Path
Master your Ruby skills and increase your Rails street cred by learning to build dynamic, sustainable applications for the web.
View Ruby Path

JavaScript Path
Spend some time with this powerful scripting language and learn to build lightweight applications with enhanced user interfaces.
View JavaScript Path

HTML/CSS Path
Learn the fundamentals of design, front-end development, and crafting user experiences that are easy on the eyes.
View HTML/CSS Path

iOS Path
Try your hand at building iOS applications for iPhone and iPad mobile devices. Learn the basics of iOS development and bring your app ideas to life.
View iOS Path

Neuer Tab

https://www.google.com

Apps

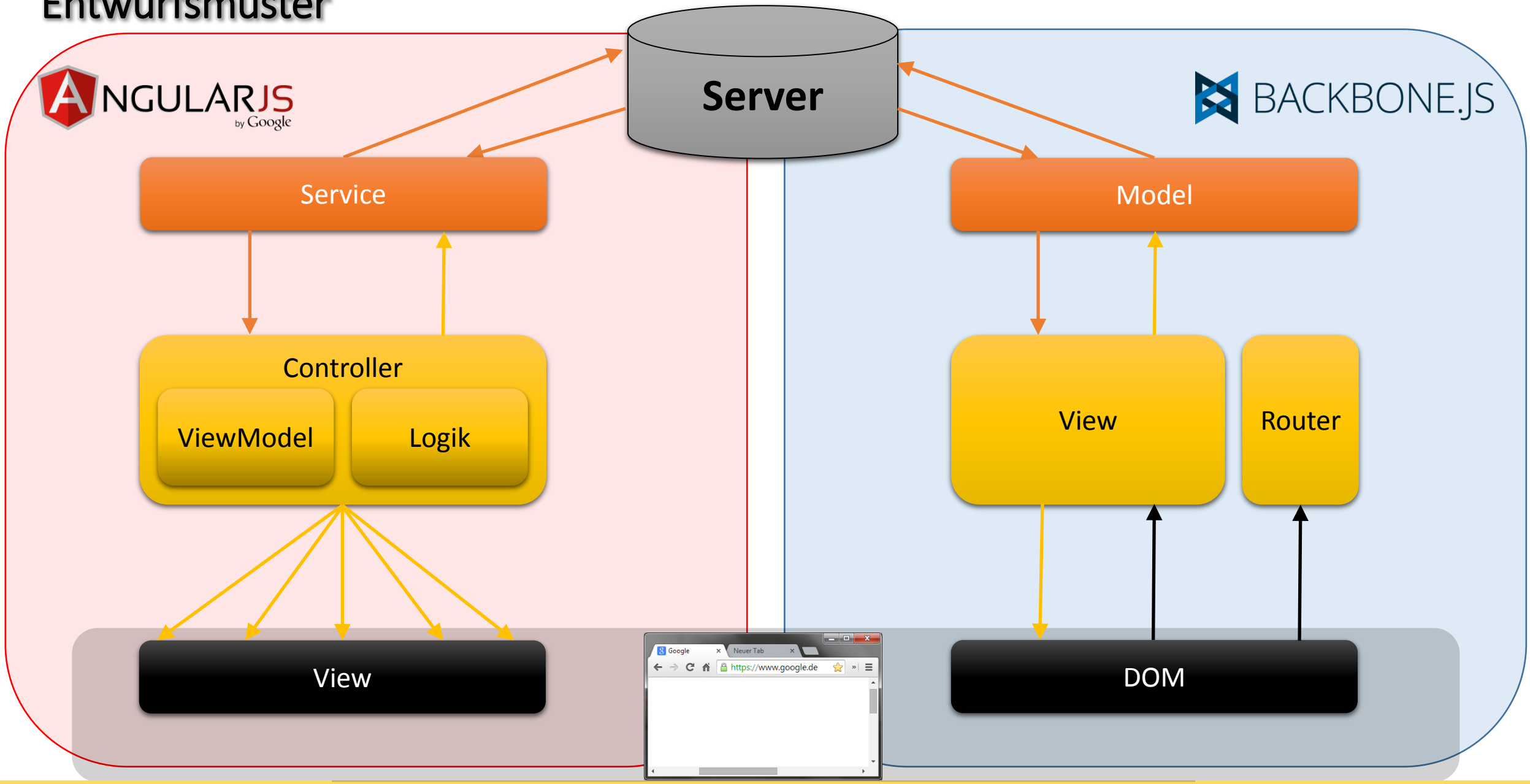
+Marcel Gmail Bilder

Google Deutsch

Google-Suche Auf gut Glück!

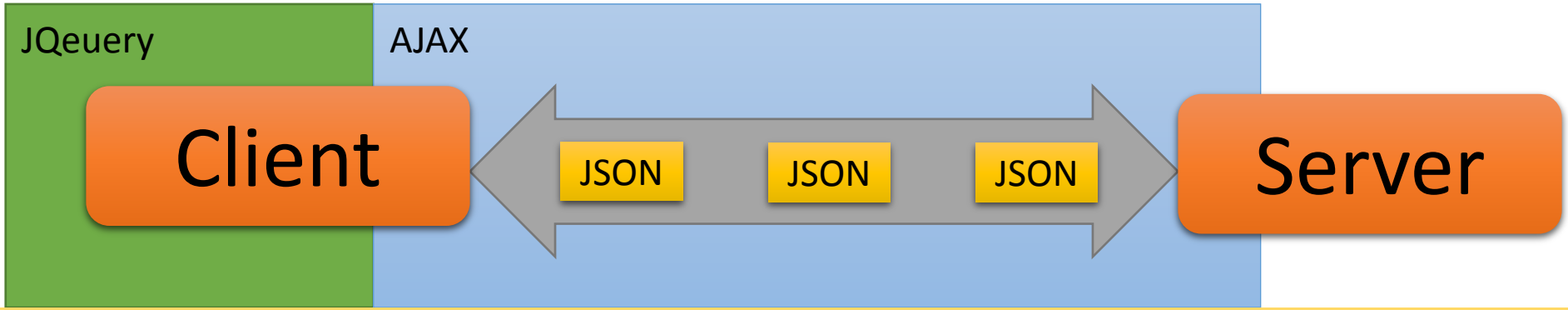
Werbeprogramme Unternehmen Über Google Datenschutzerklärung & Nutzungsbedingungen Einstellungen Google.de verwenden

Entwurfsmuster



Web Technologien

- **JSON**
  - als Datenformat
  - z.B.: `{"technologie": "json", "Herausgeber": "Douglas Crockford", "object": {"a": "value", "b": „null"}}`
- **AJAX**
  - asynchrone Datenübertragung zur Kommunikation zwischen Server & Client
- **JQuery**
  - JavaScript Erweiterung
  - Animationen
  - DOM Manipulation



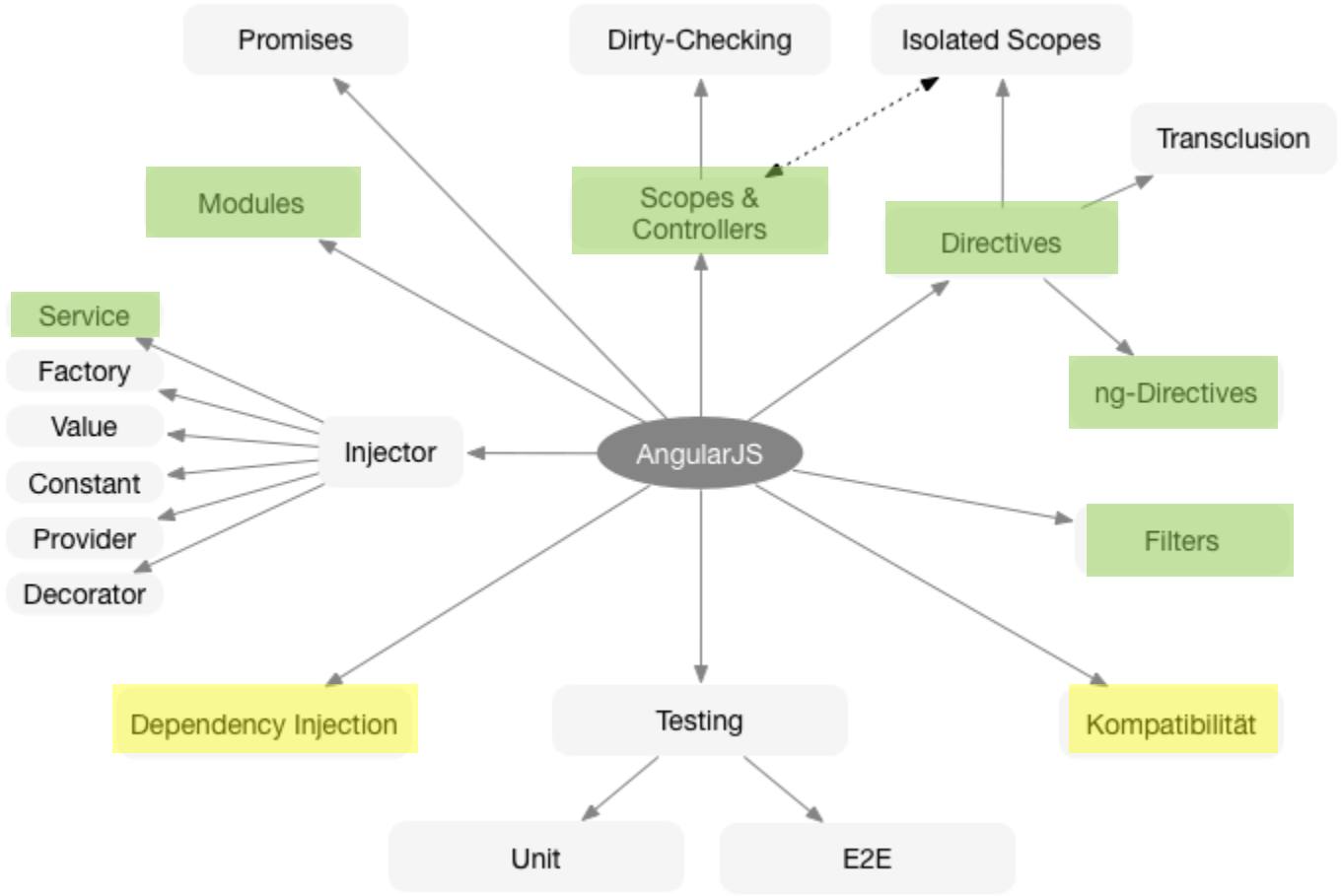

FLATGUITARS
exnovo
KICK MU
ANGULARJS
by Google

- Keyfacts
- Komponentenüberblick
- Modules
- Directives
- Controller
- Expressions
- Filter
- Eigenentwickelte Direktiven
- Services

Keyfacts

- Open-Source-Framework
- JavaScript
- Erscheinungsjahr: 2009
- Entwickler: Google Inc.
- Aktuelle Version 1.2.16 (04.04.2014)

Komponentenüberblick



http://angularjs.de/assets/figures/angularjs-overview-2c3d0ef6f094a5edd3d234c82ad6b886.png

Modules

- Container der Applikation
- sammeln Controller, Services, Filter, Direktiven
- deklarativer Programmieransatz
- optionale Abhängigkeit
- erhöhen Wiederverwendbarkeit
- hohe Wartbarkeit

app.js

```
1  var app = angular.module('shop', [ ]);
2
```

Modulname

angular.js

Abhängigkeiten

application.html.erb

```
1  <!DOCTYPE html>
2  <html lang="en" ng-app="shop">
3    <head>
```

Directive

app.js

Directives

- sind Marker im DOM
- geben dem Element ein Verhalten
- AngularJS Direktiven mit „ng-“-Präfix
- Beispiele:
  - ng-app: Einbinden von Angular
  - ng-repeat: Schleifeniteration
  - ng-show: bedingte Anzeige
  - ng-controller: Verwendung eines Controllers
- lassen sich mit Hilfe von Modules erstellen
  - ➢erhöht die Semantik des HTMLs

index.html.erb

```
1  <div ng-controller="TabletController as tabletCtrl">
2
3    <table ng-show="tabletCtrl.areTabletsAvailable()">
4
5      <tr>
6        <th>Name</th>
7        <th>Hersteller</th>
8        <th>Release</th>
9      </tr>
10
11     <tr ng-repeat="tablet in tabletCtrl.tablets">
12       <td>{{tablet.name}}</td>
13       <td>{{tablet.manufacture}}</td>
14       <td>{{tablet.release}}</td>
15     </tr>
16   </table>
17 </div>
```

Shop

localhost:3000

Name	Hersteller	Release
iPad Air	Apple	01.11.2013
Galaxy Tab 3 8.0	Samsung	07.07.2013

Controller

- grundsätzlich JavaScript-Funktionen
- Besonderheit: Übergabe der $scope Variable
- ermöglicht Datenaustausch mit View
- bestimmen Verhalten
- sollten nur Geschäftslogik enthalten
- hoher Grad an Wiederverwendbarkeit

app.js

```
1  var app = angular.module('shop', [ ]);
2
3  var devices = [
4    {name: 'iPad Air',
5     manufacture: 'Apple',
6     release: '01.11.2013'
7    }
8    {name: 'Galaxy Tab 3 8.0',
9     manufacture: 'Samsung',
10    release: '07.07.2013'
11   }
12 ];
13
14 app.controller('TabletController', function(){
15   this.tablets = devices;
16
17   this.areTabletsAvailable = function(){
18     return this.tablets.length > 0;
19   }
20
21 });
```

index.html.erb

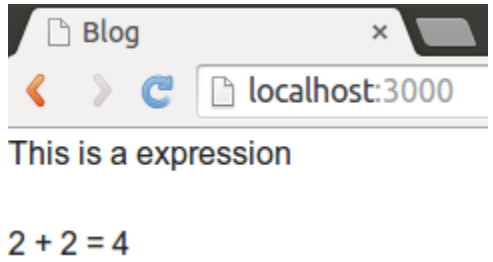
```
1  <div ng-controller="TabletController as tabletCtrl">
2
3    <table ng-show="tabletCtrl.areTabletsAvailable()">
4
5      <tr>
6        <th>Name</th>
7        <th>Hersteller</th>
8        <th>Release</th>
9      </tr>
10
11     <tr ng-repeat="tablet in tabletCtrl.tablets">
12       <td>{{tablet.name}}</td>
13       <td>{{tablet.manufacture}}</td>
14       <td>{{tablet.release}}</td>
15     </tr>
16   </table>
17 </div>
```

Expressions

- dient zu Darstellung von Ausdrücken
- Funktionalität wie eval() Funktion, aber:
  - keine Boolschen Ausdrücke
  - keine Schleifen
  - keine Exceptions
- doppelt geschweifte Syntax
- Verwendung von Filtern mögliche

index.html.erb

```
1  <p>This is a {{ "expression" }}</p>
2  <p>2 + 2 = {{ 2 + 2 }}</p>
```

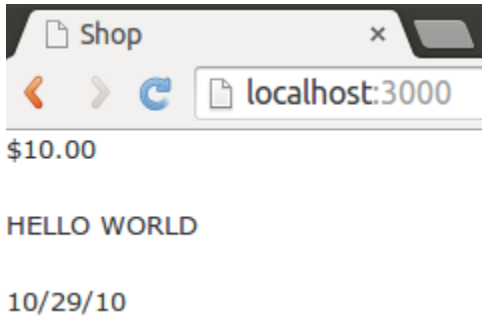


Filter

- formatiert Ausdrücke
- Einsatzbereich in Views, Controllern und Services
- Verwendung durch Pipe-Symbol (|)
- Analogie: Methoden
- können auch selbst erstellt werden
- Beispiele:
  - Währung
  - Großschrift
  - Datumsformatierung

index.html.erb

```
1  <p>{{ 10 | currency }}</p>
2  <p>{{ "hello world" | uppercase }}</p>
3  <p>{{ 1288323623006 | date:'shortDate' }}</p>
```



Eigenentwickelte Direktiven

- Deklaration durch directive() Methode
- werden mittels AJAX geladen
- ergeben HTML Elemente
- Verwendung von Templates
- hohe Wiederverwendbarkeit
- optionaler Controller
- erhöhte Semantik des HTML

index.html.erb

```
1  <product-table></product-table>
2
3
```

Shop

localhost:3000

Name	Hersteller	Release
iPad Air	APPLE	01.11.2013
Galaxy Tab 3 8.0	SAMSUNG	07.07.2013

Beispiel

app.js

```
1  var app = angular.module('shop', [ ]);
2
3  var tablets = [
4    {name: 'iPad Air', manufacture: 'Apple', release: '01.11.2013' },
5    {name: 'Galaxy Tab 3 8.0', manufacture: 'Samsung', release: '07.07.2013' },
6  ];
7
8  app.directive('productTable', function(){
9    return {
10     restrict: 'E',
11     templateUrl: 'welcome/product_table.html',
12     controller: function(){
13         this.products = tablets;
14     },
15     controllerAs: 'productCtrl'
16   }
17 });
```

product_table.html

```
1  <table>
2    <tr>
3      <th>Name</th>
4      <th>Hersteller</th>
5      <th>Release</th>
6    </tr>
7    <tr ng-repeat="product in productCtrl.products">
8      <td>{{product.name}}</td>
9      <td>{{product.manufacture}}</td>
10     <td>{{product.release}}</td>
11   </tr>
12 </table>
```

index.html.erb

```
1  <product-table></product-table>
2
3
```

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
localhost	GET	304 Not Modified	text/html	Other	772 B 2.2 KB	34 ms 33 ms	
product_table.html /welcome	GET	200 OK	text/html	angular.js?body=1:8381 Script	3.3 KB 2.5 KB	31 ms 30 ms	

Shop

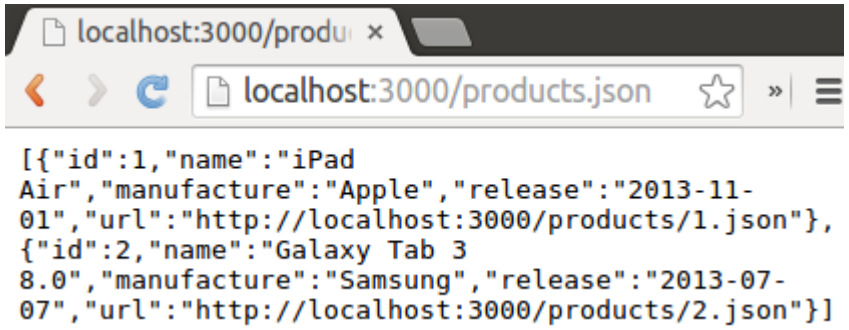
localhost:3000

Name	Hersteller	Release
iPad Air	APPLE	01.11.2013
Galaxy Tab 3 8.0	SAMSUNG	07.07.2013

Services

- komplexe Logik definiert in Services
- Dependency Injection
- kennzeichnen sich durch voranstehendes $-Symbol
- Vielzahl vorhandener Services
  - $http
  - $log
  - …
- $http ermöglicht Kommunikation mit Web Servern
  - Datenaustausch von JSON über AJAX
  - get(), put(), post(), delete()

Beispiel

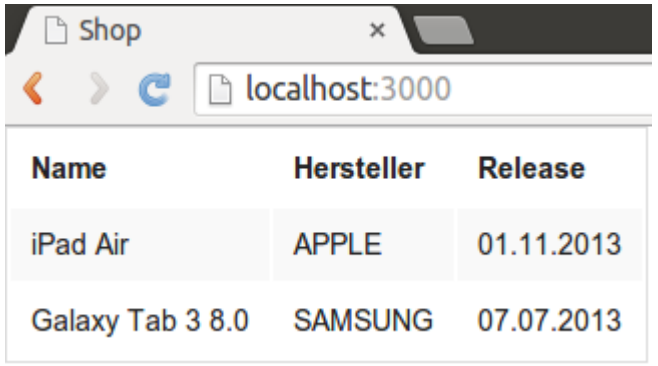


index.html.erb

```
1  <div ng-controller="ProductController as productCtrl">
2      <table>
3        <tr>
4          <th>Name</th>
5          <th>Hersteller</th>
6          <th>Release</th>
7        </tr>
8        <tr ng-repeat="product in productCtrl.products">
9          <td>{{product.name}}</td>
10         <td>{{product.manufacture}}</td>
11         <td>{{product.release}}</td>
12       </tr>
13     </table>
14 </div>
15 <product-table></product-table>
```

app.js

```
1  var app = angular.module('shop', [ ]);
2
3  app.controller('ProductController',[ '$http' , function($http){
4      var store = this;
5      store.products = [ ];
6
7    $http.get('/products.json').success(function(data){
8      store.products = data;
9    });
```



BACKBONE.JS

BACKBONE.JS - Agenda

- Keyfacts
- Backbone-Komponenten:
  - Backbone.`Model`
  - Backbone.`Collection`
  - Backbone.`View` | Templating (underscore.js)
  - Backbone.`Router`
- Außerdem
- Vorteile
- Nachteile

BACKBONE.JS - Keyfacts

- eine Bibliothek für die Entwicklung von Single-Page-Webapps
- entwickelt in 2010 von Jeremy Ashkenas (CoffeeScript)
- war ursprünglich Teil der DocumentCloud-Codebasis
- findet mittlerweile Anwendung in vielen, auch kommerziell erfolgreichen, Webapplikationen
  - z. B. SoundCloud

BACKBONE.JS - Backbone.Model

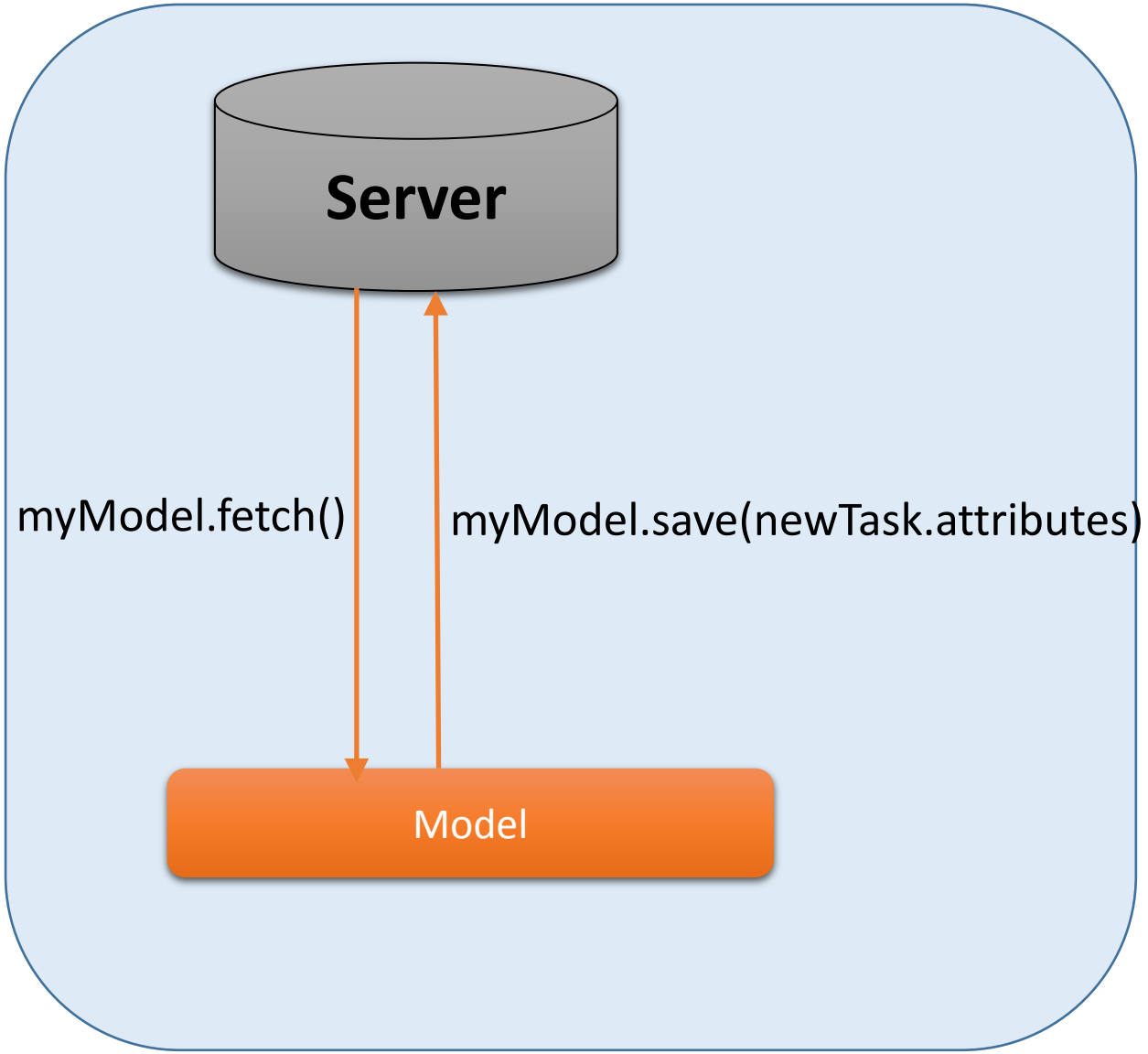
- Klassen und Objekte
- Models werden im JSON-Format mit dem Server ausgetauscht
- Model erhält eine ID

Beispiel: Task

```
{
        "id":98,"name":"Putzen",
        "description":"Alles  sauber machen!",
        "prio":1,
        "date":"2014-06-06 19:00„
}
```

Attribute

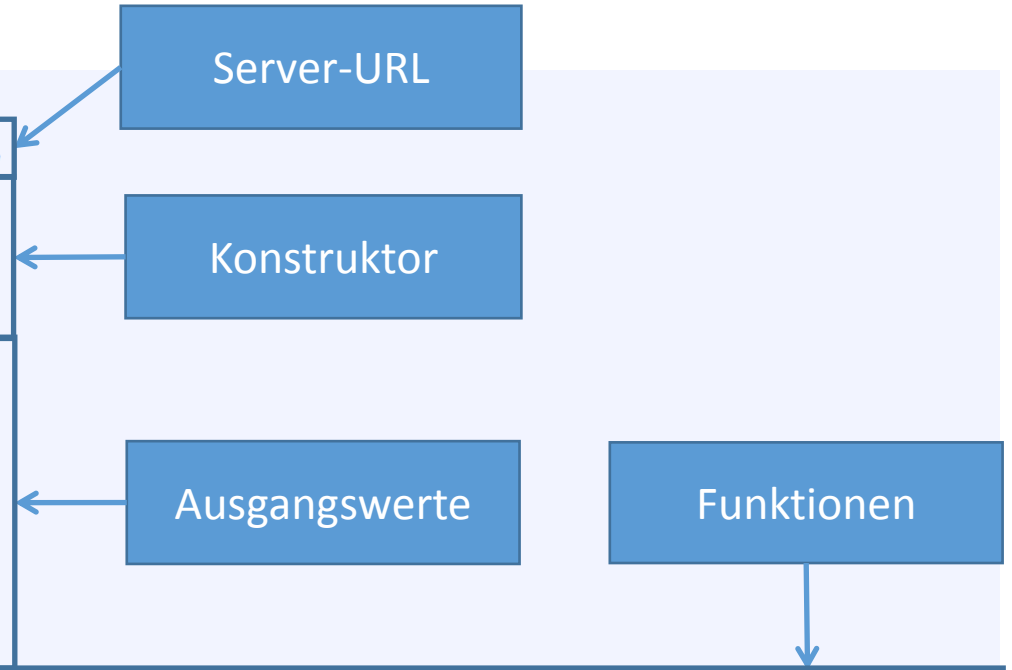
Werte



BACKBONE.JS - Ein Beispiel-Model

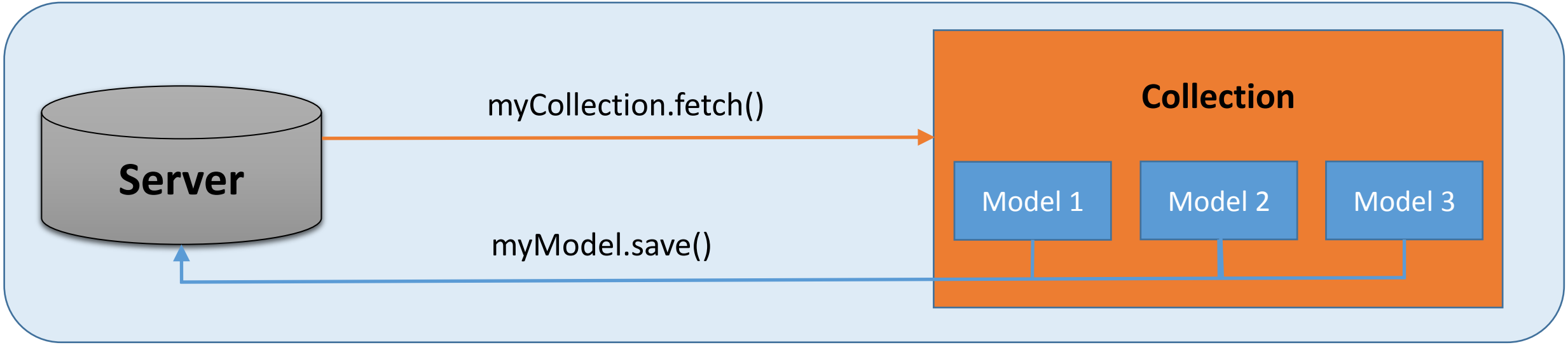
- ein Model erstellen

```
1  Task = Backbone.Model.extend({
2      urlRoot: 'http://localhost:8080/tasks/',
3      initialize: function() {
4          console.log('a new Task');
5      },
6      defaults: {
7          name: 'new task',
8          description: 'No description',
9          prio: '0',
10         date: new Date()
11     },
12     printData: function() {
13         console.log('Name: '+this.get('name')+', desc: '+this.get('description'));
14     }
15 });
```



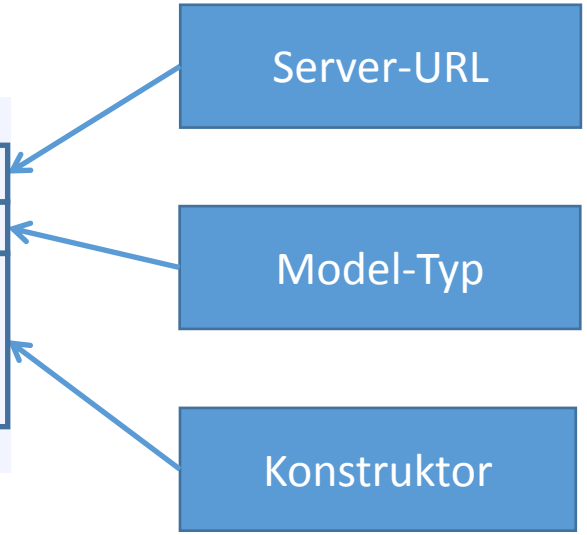
BACKBONE.JS - Backbone.Collection

- eine Collection ist eine Sammlung von Models
- Analogie: ArrayList in Java
- erleichtert Umgang und Verwaltung von Models
- eine Collection kann als Ganzes vom Server empfangen werden
- die Models in einer Collection müssen dennoch einzeln gespeichert werden



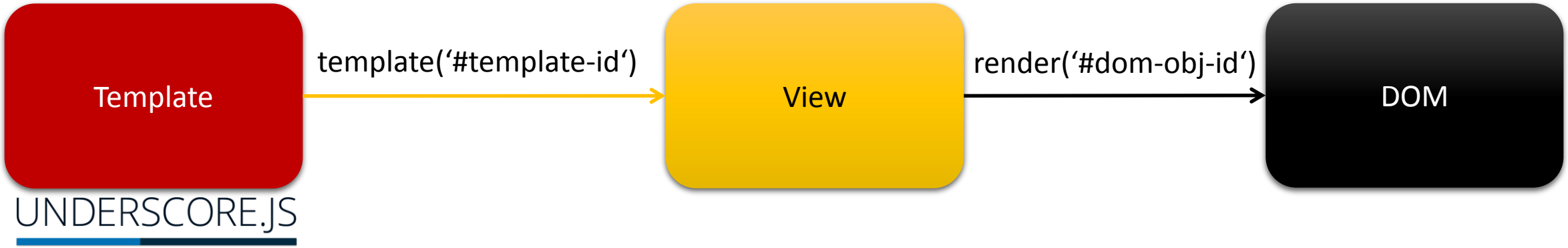
BACKBONE.JS - eine Beispiel-Collection

```
1  TaskList = Backbone.Collection.extend({
2      url: 'http://localhost:8080/tasks/',
3      model: Task,
4      initialize: function() {
5          console.log('Creating a new TaskList');
6      }
7  });
```

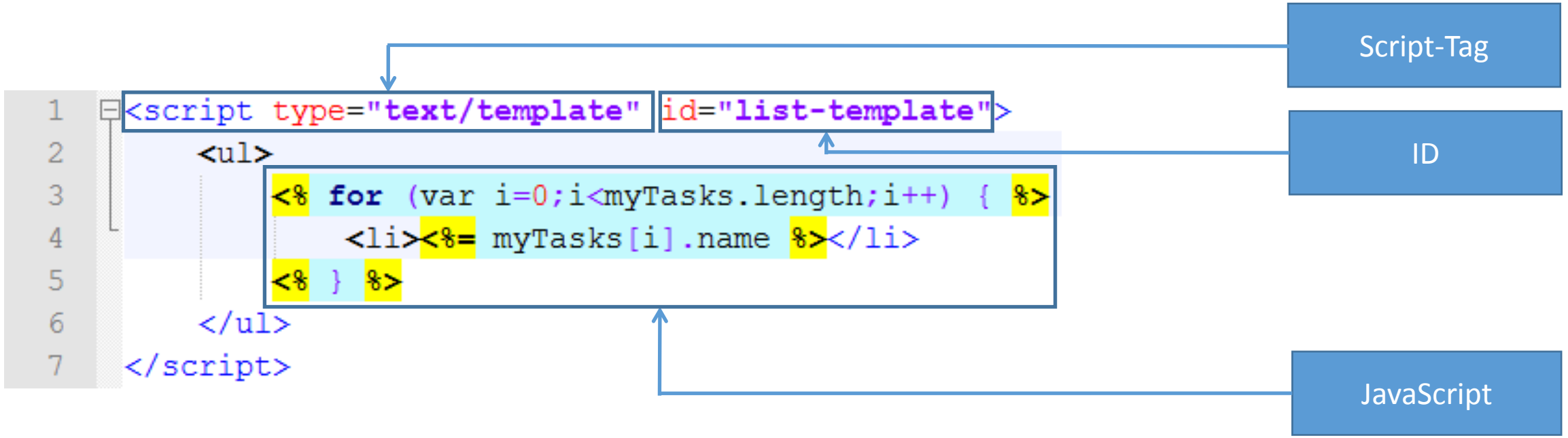


BACKBONE.JS - Backbone.View

- Backbone.js arbeitet mit Templating-Frameworks zusammen
  - nicht notwendig, aber sehr zu empfehlen
  - es kann das gewünschte Templating-Framework eingesetzt werden
  - UNDERSCORE.JS empfehlenswert



BACKBONE.JS - ein Template



BACKBONE.JS - das Template darstellen

```
1  ListView = Backbone.View.extend({
2      initialize: function() {
3                      console.log('temp init');
4                      this.render();
5                  },
6      template:   _.template($('#list-template').html()),
7      render:     function() {
8                      var self = this;
9                      this.$el.html(self.template({'myTasks': self.collection.toJSON()}));
10                     return this;
11                 }
12 });
```

Konstruktor

UNDERSCORE.JS

Darstellung

```
1  <!--Main-Div-->
2  <div id="main"></div>
3  <script>
4      var myListView = new ListView({
5          el: '#main'
6          collection: myTasks,
7      });
8  </script>
```

Ziel

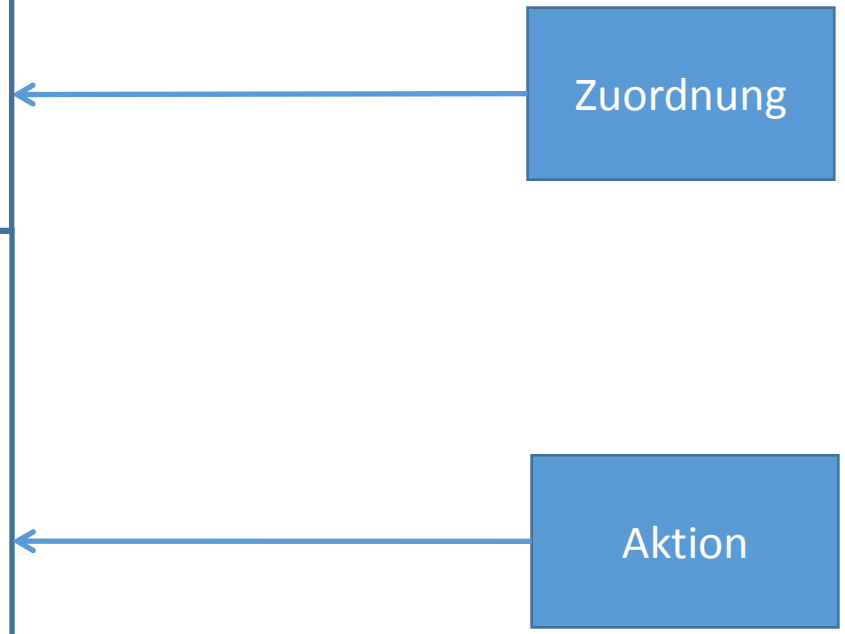
Werte-Übergabe

BACKBONE.JS - Backbone.Router

- URLs den Funktionen der Webapp zuordnen
- ermöglicht den Aufruf oder Speichern eines bestimmten Seitenzustands (z.B. Lesezeichen)
- hauptsächlich genutzt um Views anzuzeigen
- kann Controller-Funktionen erfüllen
  - Models updaten, wenn eine bestimmte URL aufgerufen wird

BACKBONE.JS - ein Router-Beispiel

```
1  var TodoListRouter = Backbone.Router.extend({
2      routes: {
3          '': 'home',
4          'todo/add': 'add',
5          'todo/:id': 'display'
6      },
7      home: function() {
8          var myListView = new ListView({
9                      collection: myTasks,
10                     el: '#main'
11                 });
12     },
13     add: function() {
14         //...
15     },
16     display: function(id) {
17         //...
18     }
19 });
```



BACKBONE.JS - Außerdem

- Backbone.`History`:
  - Backbone.js speichert die aufgerufenen URLs und ermöglicht „zurückspringen“
  - Backbone.history.start()
- Backbone.`Events`:
  - kann auf Benutzereingaben reagieren

BACKBONE.JS - Vorteile

- Backbone.js ist eine Bibliothek!
- Backbone.js lässt Entwickler viele Freiheiten
- Backbone.js ist sehr leicht einzusetzen
- sehr flexibel
- gut geeignet für umfangreiche Webapplikationen

BACKBONE.JS - Nachteile

- Backbone.js ist eine Bibliothek!
- für Neulinge können andere Lösungen besser sein
- nicht geeignet für kleine Web-Pages

Vergleich

- Framework
- Module
- Service
- Template
- Model View ViewModel (MVVM)
- Directives
- Expressions
- Filter

- Bibliothek
- zusätzliche Plugins
- Models & Collection
- Template
- Model View * (MV*)
- Router

andere Frameworks

- 

- ASP.NET
- Canjs
- Knockout.
- Knockback.js
- AGILITY.JS
- …

Alternativen vor SPA

- Java-Applet
  - Ende 1990
  - werden nicht von Suchmaschinen erfasst
- Flash
  - früher Flash sehr beliebt
  - viele Webseiten mit Flash
  - Flash im Niedergang

Fazit & Ausblick

- größere Anwendungen
- Umfangreich
- komplex
- große Community

- große und kleine Anwendungen
- einfaches Verständnis einer App
- einfache Wartung
- gute Unit Tests

- weitere Verbreitung von SPA
- immer mehr Webseiten mit SPA
- großes Zukunftspotential
- plattformübergreifend

Vielen Dank für die Aufmerksamkeit