Web-Programmieren 3

TSBE Frühlingssemester 2016

http://smlz.github.io/tsbe-2016fs/web/

Marco Schmalz

marco.schmalz@gibb.ch

Kursübersicht

1. HTML und CSS, Bootstrap
2. Repetition HTML/CSS, JavaScript Einführung
3. AngularJS Browser-Applikation
4. (Auffahrt)
5. Backend mit Elasticsearch

Heute

1. Test
2. Repetition JavaScript-Funktionen
3. Tutorial
4. Theorie Angular
5. Projektarbeit

Funktionen definieren

Funktionen können auf zwei Arten definiert werden.

Mit der Funktions-Anweisung:

```
function cube(x) {
    return x * x * x;
}
```

oder als annonymer Funktions-Ausdruck, dessen Ergebnis man dann in einer Variable abspeichern kann:

```
var cube = function(x) {
    return x * x * x;
}
```

First-Class Functions

Man kann eine Funktion als Argumente übergeben:

```
// Schreibt hello nach 1000ms
function sayHello() {
    console.log('hello');
}

setTimeout(sayHello, 1000);
```


First-Class Functions (cont.)

... oder aus einer Funktion zurückgeben:

```
// Gibt eine Funktion zurück
function makeAdder(initial) {
    // Der Wert von 'initial' geht nicht verloren, wenn makeAdder
    // fertig ist.
    return function(x) {
        return initial + x;
    }
}
var f = makeAdder(5);
console.log(f(2));      // -> 7
```

Neue Objekte erstellen mit Constructor-Funktionen

Normale Funktionen können, wenn mit `new` aufgerufen, als Konstruktoren dienen.

```
function Point(x, y) {
    var self = this; // 'this' zeigt auf die neue Instanz
    self.x = x;
    self.y = y;
    self.dist = function() {
        return Math.sqrt(self.x * self.x + self.y * self.y);
    }
}

// Anwendung mit 'new'
var p = new Point(3, 4);
console.log(p.dist());  // -> 5
```

- Konstruktor-Funktionen werden per Konvention gross geschrieben.

Probleme mit this

Warum die folgende Zeile?

```
var self = this
```

`this` kann verloren gehen, zum beispiel in call-back-Funktionen.

Variablen-Referenzen innerhalb von verschachtelten Funktionen sind dahingegen immer eindeutig.

"Module"

Self-evaluating anonymous functions

```
(function() {
    function sayHello(name) {
        console.log("Hello " + name);
    }
    var name = "Wilfriede";
    sayHello(name);
})();
```

Hilfvariablen und Hilfsfunktionen werdem so nicht zu globalen Variablen.

Codeacademy Tutorial zu Funktionen

https://www.codecademy.com/courses/javascript-beginner-en-6LzGd/0/1

Was ist AngularJS

- Ein JavaScript Programmier-Framework, um im Browser dynamische Webseiten zu erstellen.
- Open Source (Von Google unerstützt)
- https://angularjs.org/

Wieso AngularJS

- Standard JavaScript
- DataBinding (Daten werden automatisch aktualisiert)
- Routing (Mehrere Seiten simulieren)
- Leichgewichtig
- Grosse Community (60'000 Fragen auf StackOverflow)

Angular: Los gehts!

`angular.js` in Webseite einbinden:

```
<script
   src="https://code.angularjs.org/1.5.1/angular.js"
   data-require="angular.js@1.5.x"
   data-semver="1.5.1">
</script>
```

Erstes Beispiel (`ng-app` Attribut zum `body`-Tag hinzufügen):

```
<body ng-app>
    <h1>{{2+2}}</h1>
</body>
```


Eigenes Modul schreiben

Jede Angular-App besteht mindestens aus einem Haupt-Modul:

```
// Neues Modul erstellen
var app = angular.module('meineApp', []);

// Neues Modul brauchen um einen Controller zu erstellen
app.controller("MeinController", function() {
    ...
});
```

Neues Modul im HTML angeben:

```
<body ng-app="meineApp">
   ...
</body>
```

Controller

Der Controller ist das '''Bindeglied''' von JavaScript zum HTML.

```
// Neues Modul erstellen
var app = angular.module('meineApp', []);

// Neues Modul brauchen um einen Controller zu erstellen
app.controller("MeinController", MeinController);

function MeinController() {}
```

Controller im HTML mit `ng-contoller` verwenden:

```
<body ng-app="meineApp">
   <div ng-controller="MeinController as ctrl">
     ...
   </div>
</body>
```


Datenaustausch

Variabeln welche im Controller definiert sind ...

```
function MeinController() {
  var ctrl = this;
  ctrl.name = "Marco";
}
```

... können in der View (HTML) verwendet werden:

```
<div ng-controller="MeinController as ctrl">
    <p ng-bind="ctrl.name"></p>
    <p>{{ctrl.name}}</p>
</div>
```

Funktionen aufrufen

Funktionen welche im Controller definiert werden ...

```
function MeinController() {
  var ctrl = this;
  ctrl.name = "Marco";
  ctrl.sagHallo = function() {
    alert('Hallo ' + ctrl.name);
  };
}
```

... können vom HMTL aus mit `ng-click` aufgerufen werden:

```
<div ng-controller="MeinController as ctrl">
    <button ng-click="ctrl.sagHallo()">Sag Hallo</button>
</div>
```


Datenaustausch 2

Daten können in beide Richtungen augetauscht werden.

```
function MeinController() {
  var ctrl = this;
  ctrl.name = "Marco";
  ctrl.sagHallo = function() {
    alert('Hallo ' + ctrl.name);
  };
}
```

In der View (HTML) das Attribut `ng-model` verwenden:

```
<div ng-controller="MeinController as ctrl">
    <input ng-model="ctrl.name"/>
    <p>{{ctrl.name}}</p>
</div>
```

Angular template

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>AngularJS Plunker</title>
  <script data-require="angular.js@1.5.x" data-semver="1.5.1"
   src="https://code.angularjs.org/1.5.1/angular.js"></script>
  <script>
    var app = angular.module('app', []);
    app.controller('MeinController', MeinController);
    function MeinController() {
      var ctrl = this;
      ctrl.name = "Hans";
    }
  </script>
</head>
<body ng-app="app">
  <div ng-controller="MeinController as ctrl">
    <p ng-bind="ctrl.name"></p>
    <p>{{ctrl.name}}</p>
  </div>
</body>
</html>
```


LocalStorage

- Werte im Browser über den Neustart hinaus speichern
- Key-Value Store
- Zugriff mit

```
// lesen
console.log(window.localStorage.name);

// Wert schreiben
window.localStorage.name = "Marco";
```

- Referenz-Dokumentation:

  https://developer.mozilla.org/en/docs/Web/API/Window/localStorage

Komplettes Beispiel

http://plnkr.co/edit/hnKyqx92a8N1YFAUFm3M