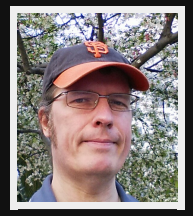
Single Page Applications mit JavaScript

Oliver Zeigermann / @DJCordhose

Online-Version: http://djcordhose.github.io/serious-javascript/slides/spa_basta_2014.html

Oliver Zeigermann

- Entwickler bei http://graylog2.org/
- @DJCordhose
- https://github.com/DJCordhose
- javatojavascript.blogspot.de
- JavaScript und TypeScript für C#-Entwickler



Signierstunde

Mittwoch, 10:00 Uhr

„JavaScript und TypeScript für C#-Entwickler“

mit **Oliver Zeigermann**

entwickler.press

Single Page Applications (SPA)

- Web-Anwendungen, laufen im Browser
- Nur eine einzige Seite wird an den Browser vom Server ausgeliefert
- Alle weiteren Aktionen werden von dem JavaScript kontrolliert, das mit der Seite ausgeliefert wurde
- Zugriffe auf den Server nicht zur Darstellungen, sondern zur Übertragung von Daten
- Erlaubt höchste Interaktivität und beste Reaktion auf Benutzereingaben
- Komfort vergleichbar mit Desktop-Anwendungen
- Offline-Fähigkeit

Inhalt

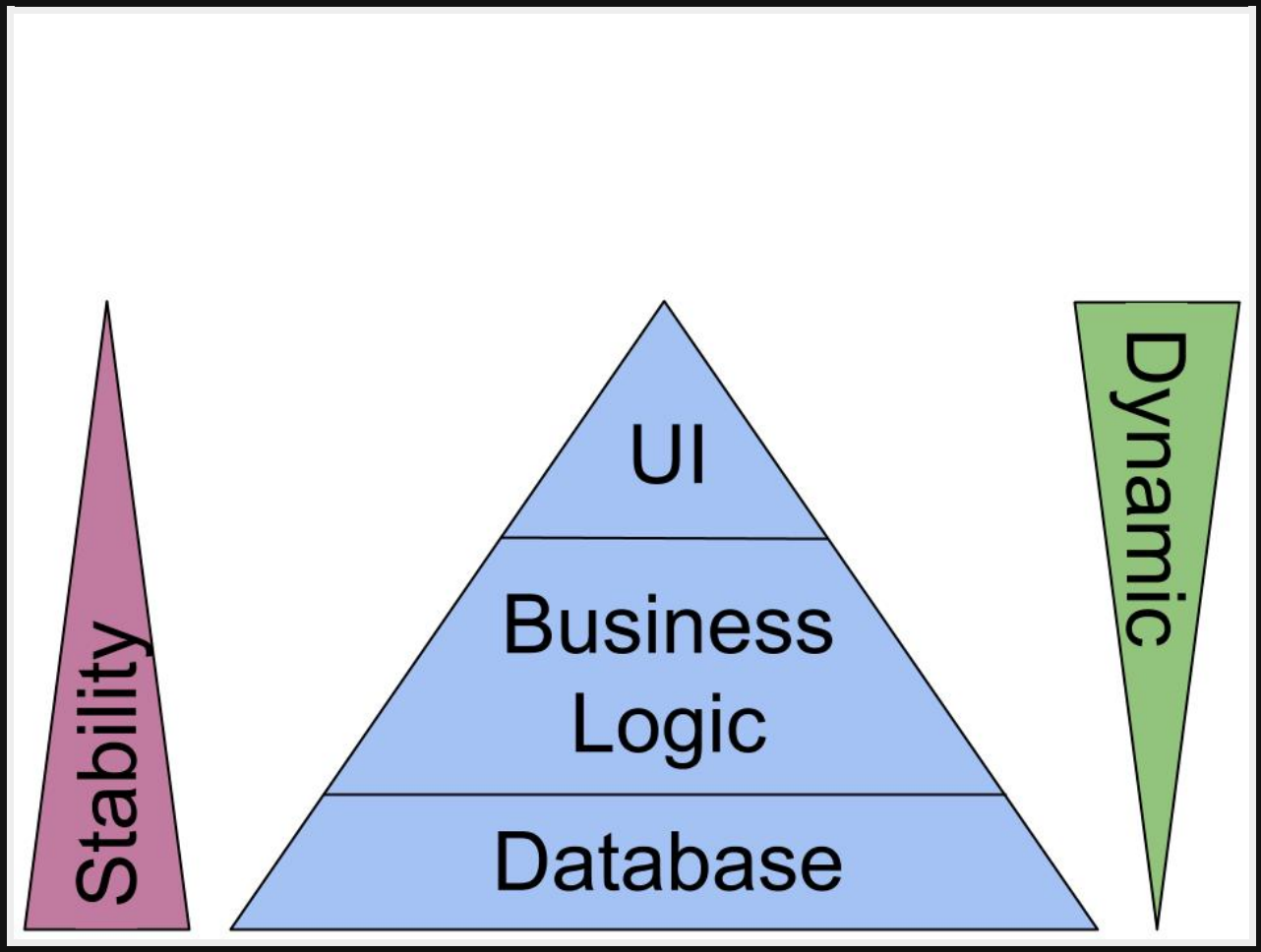
- Architektur
- Server-Seite und Client-/Server-Kommunikation
- Frameworks
- Fragen und Diskussion

Architektur einer (SPA)

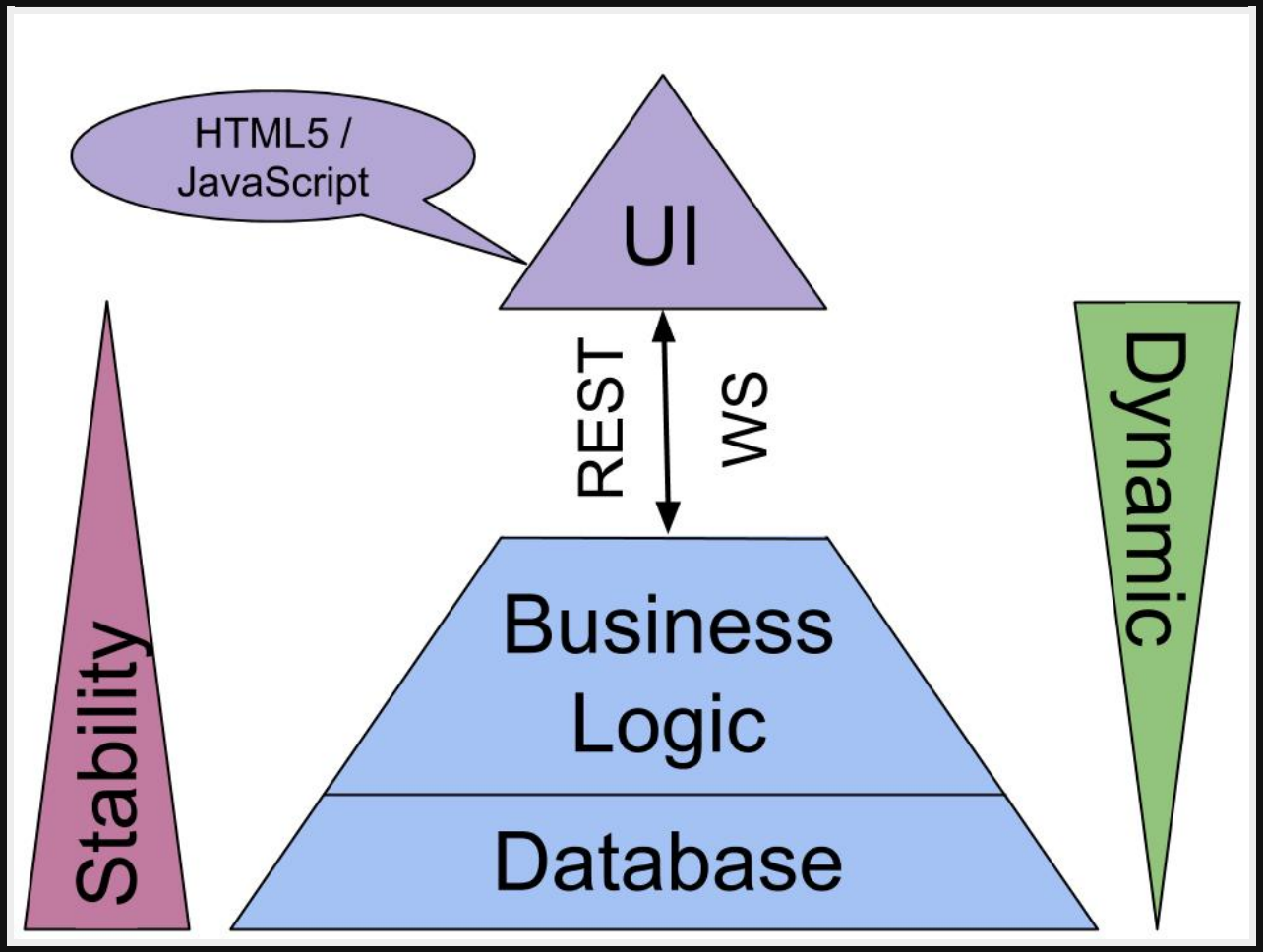
- Server wird nur noch zur Kommunikation mit dem Client benötigt
- Client macht REST/JSON calls zum Server
- Client wird entweder komplett geladen oder in Modulen nachgeladen
- Reload, Links, Bookmarks, Forward-/Backward-Navigation über Deep-Links (#)
- Lokale und Offline-Speicherung über LocalStorage oder IndexedDB
- Kommunikation mit mehreren API-Servern über CORS möglich

Architektur-Varianten

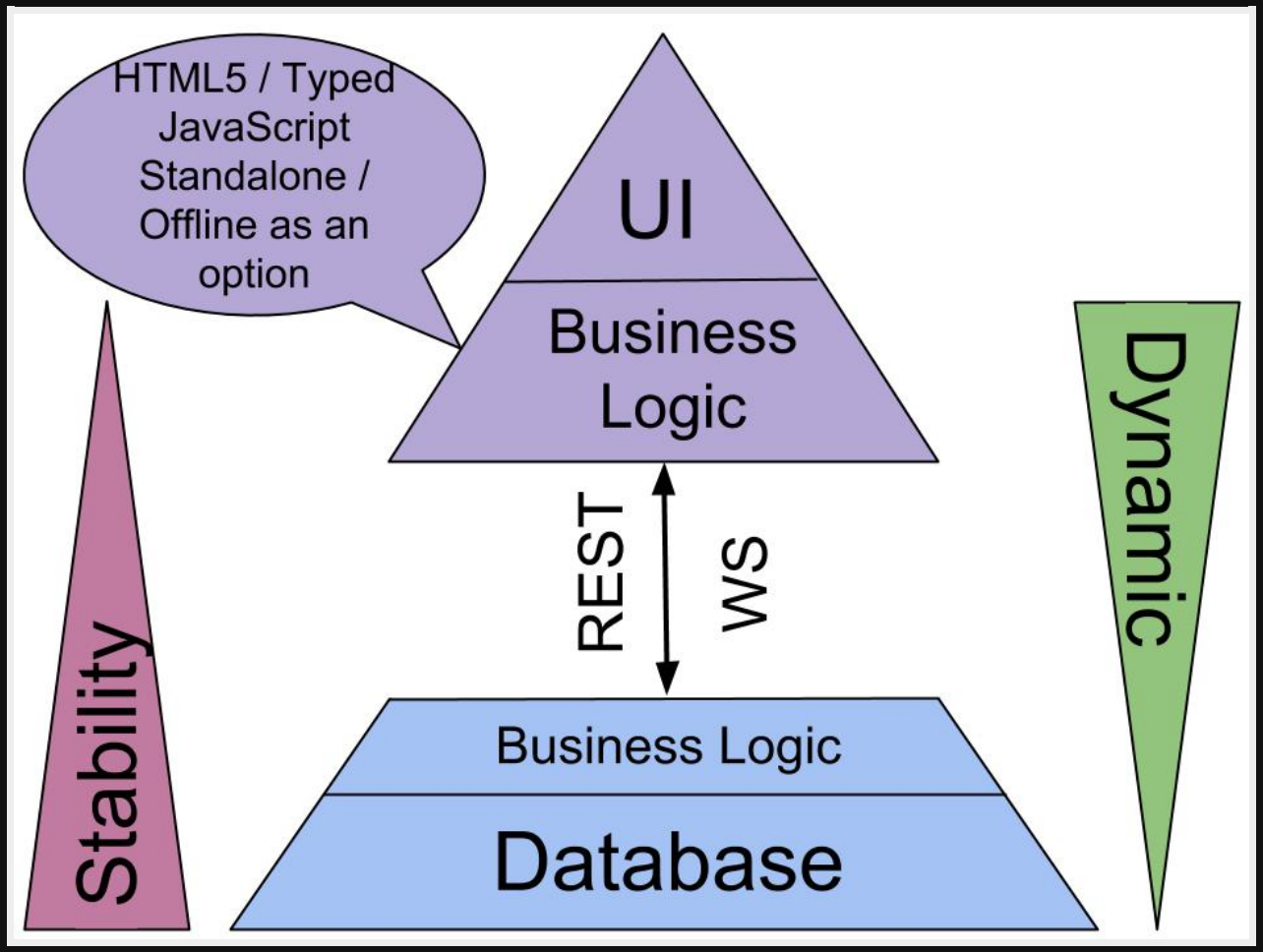
Anforderungen an Dynamik und Stabilität



SPA Variante #1: Lean



SPA Variante #2: Fat



Technische Anforderungen an den Browser

- SPAs laufen länger (teilweise viel länger) als eine klassiche Webseite
- Manche SPAs schließt man fast nie
- Bei mir sind das Gmail, Facebook und Twitter
- Die Anforderungen für Speicher-Management vergleichbar wie bei einer Desktop-Applikation
- Reaktionszeiten der Anwendungen sind ebenso wichtig
- Moderne Browser bieten Mittel des Profilings und der Analyse
- Chrome sticht hervor, aber auch Firefox und die neuesten IE (ab 11) erlauben Profiling
- Mehr dazu: https://speakerdeck.com/addyosmani/javascript-memory-management-masterclass

Was mache ich auf der Server-Seite?

Fragestellung

- JavaScript im Browser ist als Zielsprache gesetzt
- Meist braucht man auch Code, der auf dem Server läuft
- Wie setzt man diesen um?
- JEE/Spring/.NET/Rails/Python?
- Oder ebenfalls in JavaScript?

Option #1: Server in klassicher Server-Technik (JEE/Spring/.NET/Rails/Python) bzw. existierender Server bleibt wie er ist

- Naheliegende Lösung
- Kommunikation über REST/JSON
- Mapping von Objekten auf JSON
- Polyglotte Programmierung

RESTFul Web Services

REST = Representational State Transfer

Eigenschaften

Es gibt keine klare Definition, aber dies sind Eigenschaften, die die meisten darunter subsumieren

- basiert auf HTTP / HTTPS
- Stateless
- Cachebar
- HTTP-Methoden PUT / GET / POST / DELETE werden für CRUD genommen
- Wenn möglich werden Aufrufparameter in der URL kodiert
- Oft URL Pfad auf eine bestimmte Ressource
- JSON bevorzugtes Format für komplexe Daten, XML geht auch

Beispiel

RESTFul Aufruf mit `GET`, z.B.

```
https://mortgage-demo.appspot.com/mortgage_calculator/rs/200000/10/7.5/30
```

Aufrufparameter sind in die URL codiert, Service zustandslos.

Antwort als JSON

```
{"principle": "199990.00", "total": "503409.60", "payments": "360", "monthly": "1398.36"}
```

Aufruf mit Pfad auf Ressource, z.B. `DELETE`

```
https://accounting.com/user/4711
```

Antwort Status Code 204 (No content)

Option #2: JavaScript auch auf dem Server?

- Code kann wieder verwendet werden
- Dieselben Tools für Frontend und Backend
- Kleinerer Technologiestack
- Vertikale Teams einfacher zu realisieren
- Einheitliche Entwicklungsphilosophie
- JSON als natürliches Datenformat

Node.js

- Erlaubt die Ausführung von JavaScript auch auf dem Server
- Bestandteile
  - Chrome V8 JavaScript-Engine
  - Asynchrone IO-Bibliothek (libuv), die auf allen Plattformen läuft
- Asynchrones Programmiermodell, kein Multithreading
- Kommt mit Abhängigkeits-Manager `npm`
- Forken für Auslastung aller Kerne mit "Clustering" als Kernmodul
- Skalierung über viele Maschinen möglich
- Skaliert sehr gut bei hoher Last (non-blocking)
- Express als Modul für klassische Webanwendungen
- Auch ideal als API-Server

Beispiel-Server für node

```
var http = require('http');

function handleRequest (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}
http.createServer(handleRequest).listen(1337);
```

Starten

```
node server.js
```

Wer nutzt Node.js in Produktion?

- Paypal
- Walmart
- Ebay
- Linkedin
- Viele weitere

SPAs in vier Frameworks

- jQuery
- AngularJS
- React
- Kein Framework (vanilla.js)

Zum Vergleich: Hello World ohne Framework

```
<input id="in" onkeyup="setModel(this.value)">
<p><span id="log"></span>, World</p>
<input type="button" value="Clear" onclick="reset()">
<script>
    var model;
    function setModel(value) {
        model = value;
        document.getElementById("log").innerHTML = model;
        document.getElementById("in").value = model;
    }
    function reset() {
        setModel("");
    }
    window.onload = function() {
        setModel("Hello");
    };
</script>
```

Run

Option: jQuery

- Standard-JavaScript-Bibliothek
- Fast überall zu finden
- Abstrahiert nicht von der Ebene der DOM-Manipulation
- Typischerweise "unobtrusive", d.h. bestehendes HTML wird nicht verändert

Hello World jQuery

```
<input id="in">
<p><span id="log"></span>, World</p>
<input id="btn" type="button" value="Clear">
<script>
    var model;
    function setModel(value) {
        model = value;
        $("#log").html(model);
        $("#in").val(model);
    }
    $(document).ready(function () {
        $("#in").on("keyup", function (event) {
            setModel($("#in").val());
        });
        $("#btn").on("click", function () {
            setModel("");
        });
        setModel("Hello");
    });
</script>
```

Run

Option: AngularJS

- HTML enhanced for web apps!
- HTML wird erweitert (Directives)
- Eigene Directives (== Komponenten) sind möglich
- 2-Wege-Databinding

Hello World AngularJS

```
<body np-app ng-controller="HelloController as helloController">
<input ng-model="helloController.greeting.text">
<p>{{helloController.greeting.text}}, World!</p>
<input type="button" value="Clear" ng-click="helloController.clear()">
</body>
```

```
function HelloController() {
    this.greeting = {
        text: 'Hello'
    };
}
HelloController.prototype.clear = function() {
    this.greeting.text = '';
};
```

Run

Option: React

- Framework von Facebook und Instagram
- Nur das V in MVC
- Basiert auf Komponenten
- 1-Wege-Databinding: Änderungen des Modells führen automatisch zum Neurendern von Komponenten
- Ein schlauer Algorithmus macht minimale Änderungen am eigentlich DOM => schnell
- React läuft im Browser und auf dem Server

Hello World React

```
var HelloMessage = React.createClass({
  getInitialState: function() {
    return { model: this.props.greeting};
  },
  updateModel: function(event) {
    this.setState({model: event.target.value});
  },
  reset: function() {
    this.setState({model: ""});
  },
  render: function() {
    return (<div>
        <input onChange={this.updateModel} value={this.state.model} />
        <p>{this.state.model}, World</p>
        <input type="button" value="Clear" onClick={this.reset} />
     </div>);
  }
});
var mountNode = document.getElementById('example');
React.renderComponent(<HelloMessage greeting="Hello" />, mountNode);
```

Run

Wrapup

- SPAs erlauben einen Bedienkomfort, der Desktop-Anwendungen in nichts nachstehen muss
- JavaScript ist auf Client und auf Server möglich
- Die Auswahl des Frameworks ist essentiell
  - Kein Framework: Hybris, gängige Frameworks passen nicht oder Anwendung ist sehr einfach
  - jQuery: Der Standard, keine Abstraktion vom DOM, bei vielen Ausdrücken wird es unübersichtlich, andere Frameworks basieren darauf
  - Angular.js: Trifft viele Entscheidungen für uns, viele davon gut, Java-/C#-Programmierer fühlen sich abgeholt
  - React: minimales, einfaches API bei bester Performanz, server-seitiges Rendering möglich, fehlende Teile evtl. über Backbone oder Flux

Signierstunde

Mittwoch, 10:00 Uhr

„JavaScript und TypeScript für C#-Entwickler“

mit **Oliver Zeigermann**

entwickler.press

Vielen Dank

Fragen / Diskussion

Oliver Zeigermann / @DJCordhose