Unternehmenskritische Anwendungen mit JavaScript

Oliver Zeigermann / @DJCordhose

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

Joe McCann
@joemccann

Folgen

"By 2017, JavaScript will be the most in-demand language skill in application development (AD)."

— Forrester Research 2014

Übersetzt aus dem Englisch von bing Translator

"Bis zum Jahr 2017 werden JavaScript die am meisten gefragte Sprachfertigkeit in der Anwendungsentwicklung (AD)".

— Forrester Research 2014

Antworten Retweetet Favorisiert ••• Mehr

RETWEETS	FAVORITEN
264	125

20:29 - 16. Apr. 2014

Was ist eine Unternehmenskritische Anwendungen?

der Erfolg des Unternehmens ist von der Anwendung abhängig

die Anwendung lebt für viele Jahre

Halbwertszeit von Code oft ebenfalls mehrere Jahre

die Anwendung ist ein unendliches Spiel (im Gegensatz zu einem endlichen Spiel in der Spieltheorie)

die Anwendung wird von einem Team entwickelt (mehr als ein Mitglied)

es gibt kein konstantes Team,
Mitglieder kommen und gehen

Fragen wir uns erst einmal selbst...

Warum nehmen wir JavaScript für unternehmenskritische Anwendungen?

- JavaScript macht doch jetzt jeder!
- Ich möchte schnellere Entwicklung
- Die Turn-Around-Zeiten von JavaScript machen das Entwickeln zur Freude
- Java / C# langweilen uns und wir wollen mal etwas neues ausprobieren
- JavaScript ist echt eine tolle und aufregende Sprache

Oder eher ...

... JavaScript ist die einzige Sprache, die in jedem Browser läuft

Egal ob wir wollen oder nicht, wir kommen nicht an JavaScript vorbei...

… wenn wir moderne Single Page-Applikationen für den Browser haben wollen

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

Welche Problemstellungen ergeben sich daraus?

Wie halte ich meine Code-Basis wartbar?

Was mache ich auf der Server-Seite?

Wie gehe ich mit den Komplexitäten der Browser-Entwicklung um?

Problemstellung #1: Wie halte ich meine Code-Basis wartbar?

Code Qualität, Module, statische Analyse

Untersuchung der Fragen: Sind JavaScript-Projekte von Anfang an unwartbar und können nur im kleinen Rahmen funktionieren?

Provokantere Frage: Selbst wenn ich schnell in JavaScript entwickeln kann, wird mein Code in einem Jahr noch wartbar sein?

Und in 5 Jahren?

Und in 10 Jahren?

Fragestellungen der Wartbarkeit

- Wie modularisiere ich meinen Code, um
  - ihn handhabbar zu halten
  - mit mehreren Entwicklern daran arbeiten zu können
- Wie halte ich meinen Code einfach lesbar?
- Wie definiere ich ein Maß für Code-Qualität und wie erreiche ich diese?
- Wie ermögliche ich eine statische Analyse?
- Wie teste ich meinen Code?
- Wie automatisiere ich Analyse und Test?

Erfahrung: Module und Typen-Hierarchien fördern strukturiertes Arbeiten

Modularisierung

- Revealing Module Pattern: Sichtbarkeit
- AMD und RequireJS: Externe Abhängigkeiten, asynchron
- CommonJS und Browserify: Externe Abhängigkeiten, synchron
- Module in ECMAScript 6

Klassische Typen und Vererbung mit JavaScript

```
/** @constructor */
function Person(name) {
    this.name = name;
}
// Methode
Person.prototype.getName = function() {
    return this.name;
};
```

```
var olli = new Person('Olli');
olli.getName() === 'Olli';
```

Deklarierte Typen

- Grundlage der Überlegung
  - Verlässliche Toolunterstützung für Analyse und Refactoring sind ohne statische Typ-Information nicht möglich
  - Ohne verlässliche Analyse- und Refactoring-Möglichkeiten wird Enterprise-Code schnell untwartbar
- Mögliche Techniken
  - Google Closure Compiler
  - TypeScript

Google Closure Compiler

```
/**
 *
 * @param name {string}
 * @param alter {number}
 * @param geschlecht {string=}
 * @constructor
 * @implements {HasName}
 */
function Person(name, alter, geschlecht) {
    this.name = name;
    this.alter = alter;
    this.geschlecht = geschlecht;
}
```

Google Closure Compiler: Bewertung

- Vorteile
  - Lesbarkeit für Menschen verbessert
  - Verbesserter IDE-Support durch deklarierte Typen
  - Kein Compilierungsschritt zur Ausführung notwendig
  - Compiler nutzt Typ-Informationen auch zur Optimierung
- Nachteile
  - Technologie-Stack wächst
  - Es sieht nicht sonderlich schön aus
  - Abhängigkeit von Google-Technologie (Rückkehr zu einfach JavaScript möglich)
  - JavaScript-Konstrukte (z.B. für Interfaces oder Varargs) teilweise etwas überraschend

TypeScript

- Mehr oder weniger ECMAScript 6 mit optionalen deklarierten Typen
- Compiliert zu JavaScript, das man so auch von Hand schreiben würde
- TypeScript-Playground
- Externe Deklarationen erlauben nachträgliche Typisierung von existierendem JavaScript-Code
- Externe Deklarationen für sehr viele JavaScript-Biblitotheken
- Talk zum 1.0 Release von Anders Hejlsberg

Beispiel-Code

```
interface HasName {
    getName(): string;
}

class Person implements HasName {
    constructor(private name: string, private alter: number,
    private geschlecht: string = 'F') {
    }
    getName() {
        return this.name;
    }
}

var olli: HasName = new Person('Olli', 43);
```

TypeScript: Bewertung

- Vorteile
  - Lesbarkeit für Menschen verbessert
  - Stark verbesserter IDE-Support durch deklarierte Typen
  - Typinformation sogar für JSON-Objekte
- Nachteile
  - Technologie-Stack wächst
  - Compilierungsschritt immer notwendig
  - Abhängigkeit von MicroSoft-Technologie (Rückkehr zu JavaScript möglich)
  - Optionale Typen erfordern viel Eigenverantwortung der Entwickler

Code Qualität mit JSHint

- Analysiert JavaScript-Files und macht konfigurierbare Prüfungen
- http://www.jshint.com/
- Deckt potentielle Fehler auf
- Kann die Code-Conventions des Teams unterstützen
- Aufruf von der Kommandozeile oder IDE-Integration

Was könnten hier die Probleme sein?

```
function olli() {
    console.log("aha")
    func();
    variable = 10;
 var myVar;
    10 == "10";
    if (true) console.log('Yo');
}
```

- Anhand eine Konfigurationsfiles können *alle* Optionen an oder ausgeschaltet bzw. konfiguriert werden
- In jeder JavaScript-Datei können zudem Warnungen ausgeschaltet werden (vgl. @SuppressWarnings in Java)

Weitere Tools

- Yo: Scaffolding
- WebStorm / IntelliJ / Visual Studio: IDE
- Grunt.js: Build-Tool
- Bower: Abhängigkeitsmanagement
- Jasmine: BDD-/Unit-Test-Framework

Fragen?

Problemstellung #2: 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

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

Fragen?

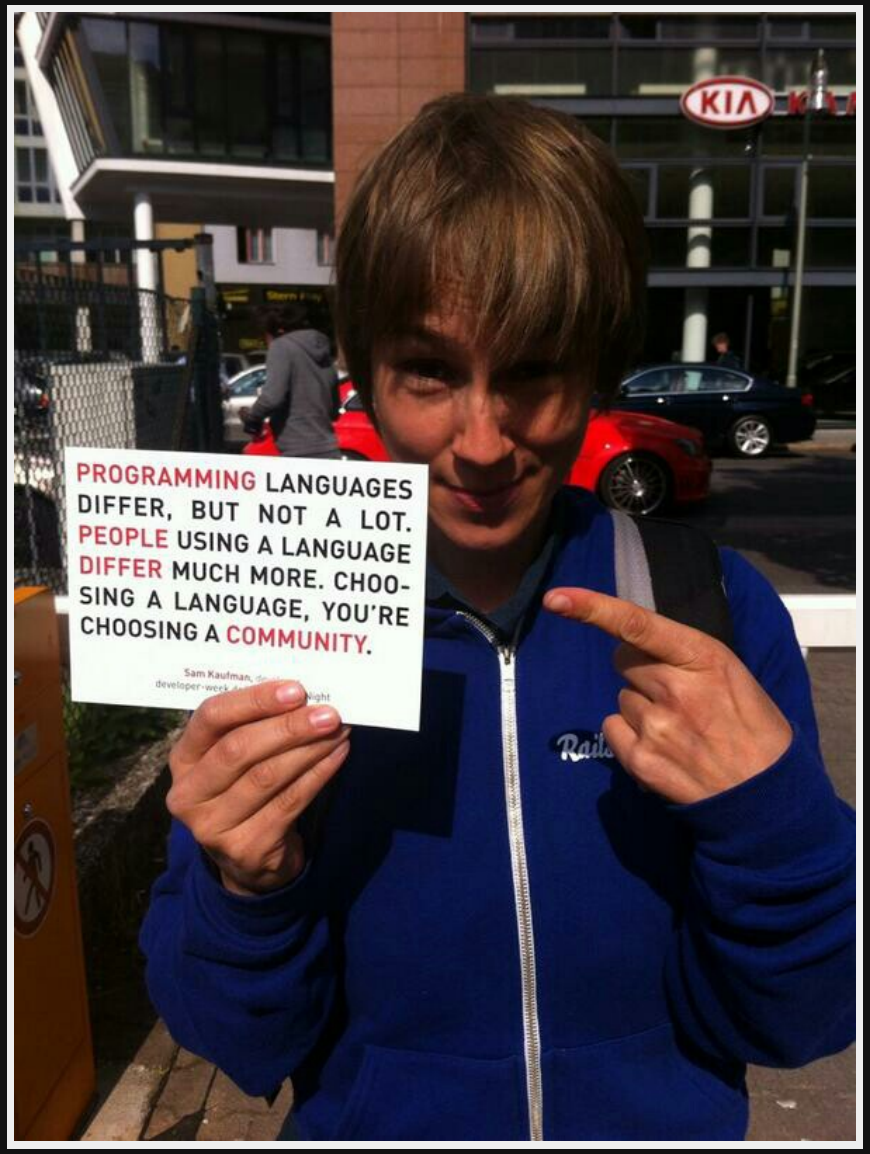
Problemstellung #3: Wie gehe ich mit den Komplexitäten der Browser-Entwicklung um?

Wozu ein Framework für die Browser-Entwicklung?

- Es gibt immer noch eine Reihe von Browser-Unterschieden bzgl. DOM und JavaScript (insbesondere pre IE9)
- Das DOM ist kein "ideales" API
- Selbst idealerweise ist das DOM immer noch sehr low-level

Anforderungen an ein Browser-Framework für Enterprise-Projekte

- Es muss zu meiner Anwendung passen
- Es muss über viele Jahre gewartet bleiben
- Rückwärtskompatibilität



https://twitter.com/kilaulena/status/463317989648248832

Kultur-Clash Java-/C# vs JavaScript-Frameworks

- kleinere Module dominieren die JavaScript-Framework-Welt
  - npm und bower mit anarchischer Organisationsstruktur
  - Wenn überhaupt dann Defacto-Standards
  - Jeder muss sich seinen eigenen Satz an Komponenten auswählen
  - u.a. yeoman und Bootstrap-Pakete geben zumindest etwas Richtung
- Java und C# kommen mit kompletten Bibliotheken und Standards
  - bietet Orientierung
  - allerdings auch weniger Freiheit (bzw. andere Auswahl muss auch vertreten werden)

Auswahl von Frameworks, die sich als Enterprise-würdig erwiesen haben

- jQuery
- AngularJS
- React
- Ext.js
- Kein Framework

Zum Vergleich: Hello World ohne Framework

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>
    <body>
      <div id="log"></div>
      <script>
          var element = document.getElementById("log");
          element.innerHTML = "<h1>Hello World</h1>";
      </script>
    </body>
</html>
```

Run

Option: jQuery

- Standard-JavaScript-Bibliothek
- Fast überall zu finden
- Adressiert Probleme bei der Programmierung des DOMs
- Abstrahiert nicht von der Ebene der DOM-Manipulation
- Funktionalität unterteilbar in "Auswahl" und "Manipulation"

Hello World jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World jQuery!</title>
    <body>
      <div id="log"></div>
      <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.0/jquery.min.js"></script>

      <script>
        $(document).ready(function(){
            $("#log").html("<h1>Hello World</h1>");
        });
      </script>
    </body>
</html>
```

Run

Option: AngularJS

HTML enhanced for web apps!

http://angularjs.org

Konzepte

- Client-Seitige Templates
- MVC (Modell ist eher ViewModel)
- Die drei Ds
  - Data Binding
  - Dependency Injection
  - Directives

Hello World AngularJS

```
<html ng-app>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.11/angular.min.js"></scr
    <script src="controller.js"></script>
</head>
<body 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>
</html>
```

```
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
- Templates können in reinem JavaScript oder mit der Template-Sprache JSX geschrieben werden
  - JSX kann "on the fly" gerendert oder
  - für Produktion nach JavaScript vorkompiliert werden
- Basiert auf Komponenten
- der View wird als "virtuelles DOM" repräsentiert
- 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

```
<div id="example"></div>
<script type="text/jsx">
/** @jsx React.DOM */
var mountNode = document.getElementById('example');
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

React.renderComponent(<HelloMessage name="Olli" />, mountNode);
</script>
```

Run

- `React.createClass`: Erzeugt eine neue Komponenten-Klasse
- `render`: erzeugt das virtuelle DOM
- `this.props` / input: Parameter für die Komponentnen-Instanz

Option: ExtJs

- hat alle Komponenten für eine Business-Anwendung
- folgt weniger der Web-Philosophy als der Desktop-Metapher
- ExtJS 5 unterstützt alle Devices
- Unterstützt weiterhin IE8
- Unterschiedliche Themes durch austauschen von CSS
- existiert seit vielen Jahren, Einsatz kommerziell

Hello World ExtJS

```
Ext.application({
    name: 'Hello World',

    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: [
                {
                    xtype: 'panel',
                    title: 'Hello World',
                    html : 'Hello!'
                }
            ]
        });
    }
});
```

Run

Letzte Option: Kein Framework

- Selbstbetrug ausschließen: Was ist die Motivation?
- Passt wirklich keines der Frameworks zu den Anforderungen?
- Glauben wir wirklich, dass unser eigenes Framework besser sein wird als ein Standard-Framework?
- Wenn ja, wer in dem Team hat schon einmal ein vergleichbares Framework gebaut?
- Wer wird das Framework warten?
- Wird es ein Framework- und ein Applikations-Team geben?

Wrapup

- Kein Framework: Hybris, gängige Frameworks passen nicht oder Anwendung ist sehr einfach
- jQuery: DOM als Abstraktion ist ausreichend, Projekt evtl. nicht sehr komplex
- Angular.js / Ember / Knockout: Sehr praktisch für SPAs und größere Projekte
- React: Höchste Anforderungen an Performanz, server-seitiges Rendering möglich, fehlende Teile evtl. über Backbone
- Ext.js: Für Business-Anwendungen, wie man sie auch mit JSF oder JavaFX bauen würde

Fragen?

Was haben wir gesehen?

- Es gibt brauchbare Strukturen, Prozesse und Tools für JavaScript-Projekte
- Gute Frameworks machen das DOM und die Browser-Vielfalt erträglich
- Die Sprache JavaScript und Werkzeuge um die Sprache herum ermöglichen Wartbarkeit auch für große Projekte
- JavaScript ist auf Client und auf Server möglich

Allerdings

- Herangehensweise anders als bei Java / .NET
- Vieles muss man sich selbst erarbeiten
- Es gibt wenige echte Standards
- Die meisten Entscheidungen muss man selbst treffen (und verantworten)
- Langlebigkeit von Bibliotheken teilweise fraglich
- Einstieg daher oft mit Angst und Unsicherheit verbunden

Vielen Dank

Fragen / Diskussion

Oliver Zeigermann / @DJCordhose