

NILS HARTMANN | @NILSHARTMANN

Das JavaScript Öko-System

Slides: <http://bit.ly/wjax2017-javascript>

NILS HARTMANN

Programmierer aus Hamburg

**Java
JavaScript
Trainings und Workshops**

@NILSHARTMANN



El Matulā 🤝 ❤️💛

@mwessendorf

Folge ich



@nilshartmann Dude! Das klingt nach nem Tageswörkshop - mindestens ! ;-)



JAXCON @jaxcon

"Schauen wir uns moderne #JavaScript - Stacks an, um zu überprüfen, was davon heutzutage noch stimmt." @nilshartmann
ow.ly/NMle30fVGh2

12:08 - 17. Okt. 2017



1



ACHTUNG!

(Vor-)Urteile



I Am Devloper

@iamdevloper

Folgen

steps to writing js in 2017:

1. install node
2. configure babel

...

9. slowly beat the eggs into the flour and butter

...

35. open index.js

Original (Englisch) übersetzen

20:57 - 10. Okt. 2017

2.130 Retweets **5.031** „Gefällt mir“-Angaben



61



2,1 Tsd.



5,0 Tsd.



<https://twitter.com/iamdevloper/status/917826490443628544>



Thomas Fuchs

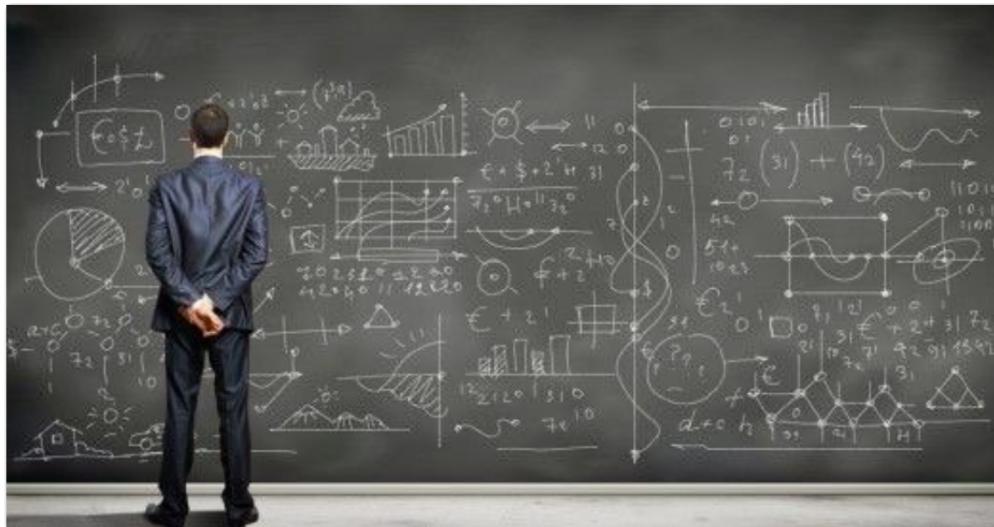
@thomasfuchs

Folgen



Marc was almost ready to implement his "hello world" React app

Original (Englisch) übersetzen



16:24 - 12. März 2016

4.285 Retweets 5.409 „Gefällt mir“-Angaben



53



4,3 Tsd.



5,4 Tsd.



<https://twitter.com/thomasfuchs/status/708675139253174273>



tom robinson

@tlrobinson

Folgen



Before ~2015: “JavaScript is a terrible language!”

After ~2016: “Stop improving JavaScript, I can’t keep up!”

🌐 Original (Englisch) übersetzen

12:34 - 29. Mai 2017

109 Retweets 197 „Gefällt mir“-Angaben



11



109



197



<https://twitter.com/tlrobinson/status/869139917137248260>

How jQuery Works

jQuery: The Basics

This is a basic tutorial, designed to help you get started using jQuery. If you don't have a test page setup yet, start by creating the following HTML page:

```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Demo</title>
6 </head>
7 <body>
8   <a href="http://jquery.com/">jQuery</a>
9   <script src="jquery.js"></script>
10  <script>
11
12    // Your code goes here. ←
13
14  </script>
15 </body>
16 </html>
```

<https://learn.jquery.com/about-jquery/how-jquery-works/>

"FRÜHER WAR ALLES BESSER!?"

Motivation

WIR BAUEN "ECHTE" ANWENDUNGEN

"ECHTE" ANWENDUNGEN

Aus Benutzersicht: bestes UI/UX

- Gewohntes Verhalten von Desktop Anwendungen
 - Einheitliches Layout und Design
 - Konsistentes Verhalten in der ganzen Anwendung
 - Konsistente Darstellung der Daten
-
- Kurze Reaktionszeiten

Für Entwicklung

- Einfach und schnell
- Saubere und verständliche Architektur
- Wartbar auch bei großer und langlebiger Code-Basis

Voice Of Rebellion · Pro-Pain

Sicher | https://open.spotify.com/artist/56F64pmwSSCcmS1CxAnPk8

Suchen

Start

Deine Musik

ZULETZT ABGESPIELT

Pro-Pain

KÜNSTLER

Pro-Pain — Voice O...

PLAYLIST

Iron Maiden HH 2017

PLAYLIST

Haukur Tomasson

PLAYLIST

App installieren

nils_hartmann

10.892 FOLLOWER

PAUSE

FOLGEN

ÜBERSICHT

ÄHNLICHE KÜNSTLER

INFORMATIONEN

Beliebt

Voice Of Rebellion 4:01

Deathwish 2:45

Foul Taste Of Freedom 3:41

One Shot One Kill 2:55

Make War (Not Love) 4:54

Radio starten

In „Deine Musik“ speichern

Zu Playlist hinzufügen

Songlink kopieren

Voice Of Rebellion

Pro-Pain

1:15 4:01

SPOTIFY WEB PLAYER

Sample File – Figma

Sicher | https://www.figma.com/file/pQLIW7fU1VQwYIJjs2epDl/Sample-File?node-id=0%...

Log-In Page

9:42 AM 42%

YOUR ART MUSEUM

151 3rd St
San Francisco, CA 94103

Email address

Password

Forgot your password?

Log In

Don't have an account?

Background Image

Instructions

- Intro Text
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6
- Step 7
- Step 8

Log-In Page

- Status Bar
- App Info
 - 151 3rd St San Fran...
 - YOUR ART MUSEUM
- Log-In Fields
- Log-In Button

Home

Menu

Exhibition

Shop

Share

DESIGN PROTOTYPE CODE

Bring Forward ⌘]

Bring to Front ⌘[

Send Backward ⌘[

Send to Back ⌘]

Group Selection ⌘G

Ungroup Selection ⌘ShiftG

Flatten Selection ⌘E

Use as Mask ⌘M

Outline Stroke ⌘O

Create Component ⌘K

Go to Master Component

Reset Instance

Detach Instance ⌘B

Show/Hide Selection ⌘H

Lock/Unlock Selection ⌘L

Flip Horizontal ⌘H

Flip Vertical ⌘V

Copy Style as CSS

Copy as SVG

Copy Style ⌘C

Paste Style ⌘V

X 0 Y 0

W 375 H 667

0° 0

CONSTRANTS

Left Top

LAYER

Pass Through 100%

FILL

Image 100%

STROKE

EFFECTS

Layer Blur

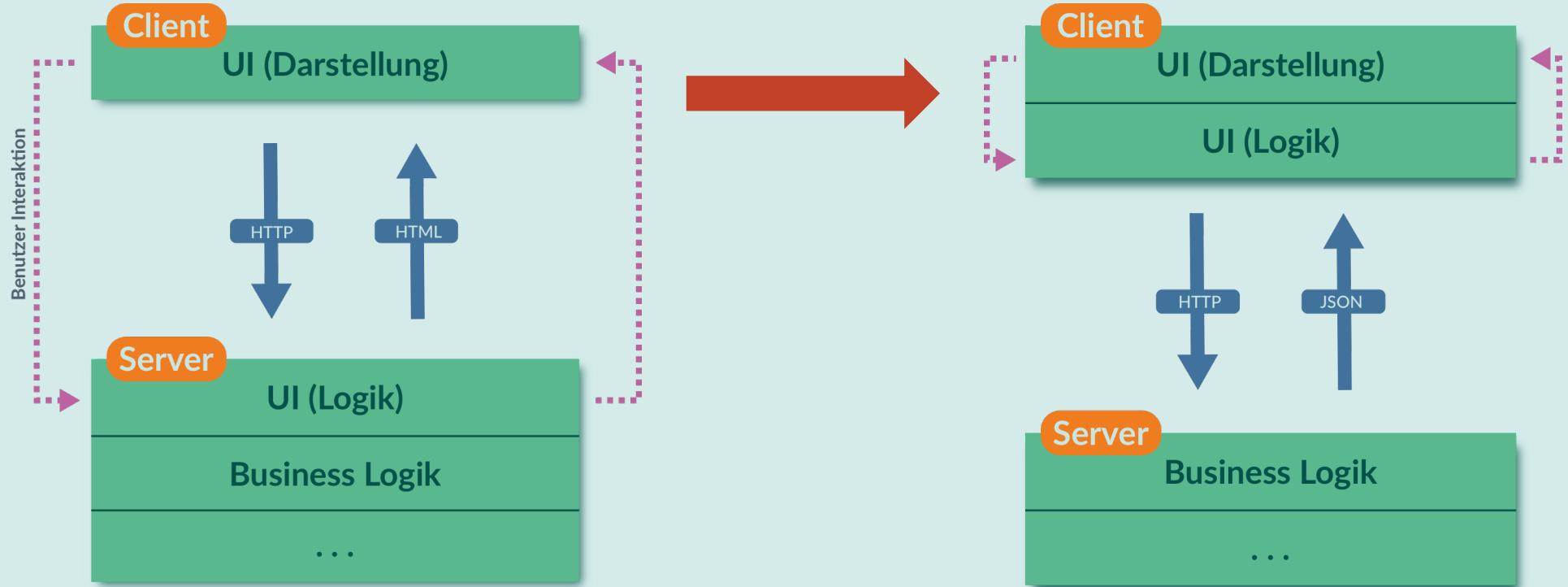
EXPORT

Click + to add an export setting

?

HTTPS://WWW.FIGMA.COM

SINGLE-PAGE-APPLICATION



Technologie

- JSP, Thymeleaf, JSF
- jQuery (anteilig)

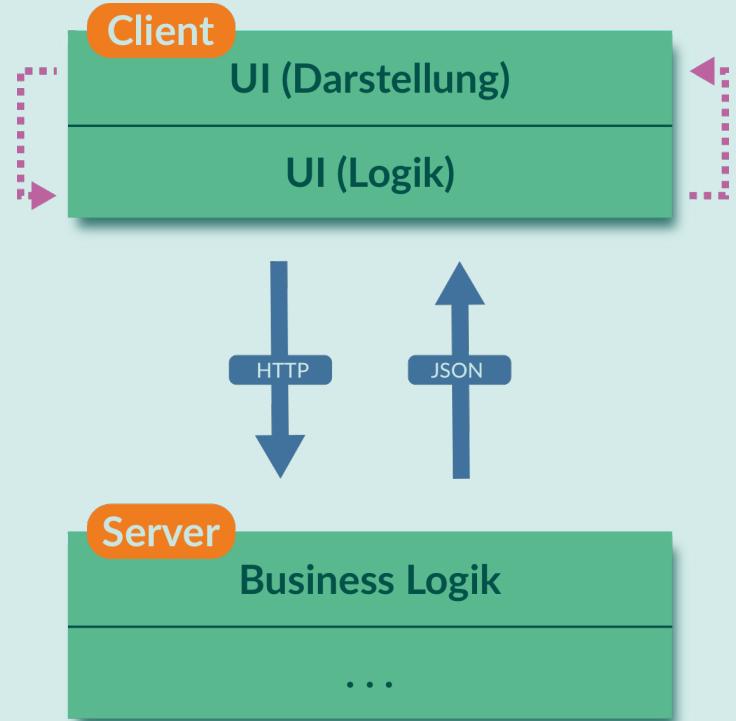
Technologie

- REST API
- React, Angular, Vue

SINGLE-PAGE-APPLICATION

JavaScript First Class Citizen, aber:

- kein Java
- kein Script
- 😞



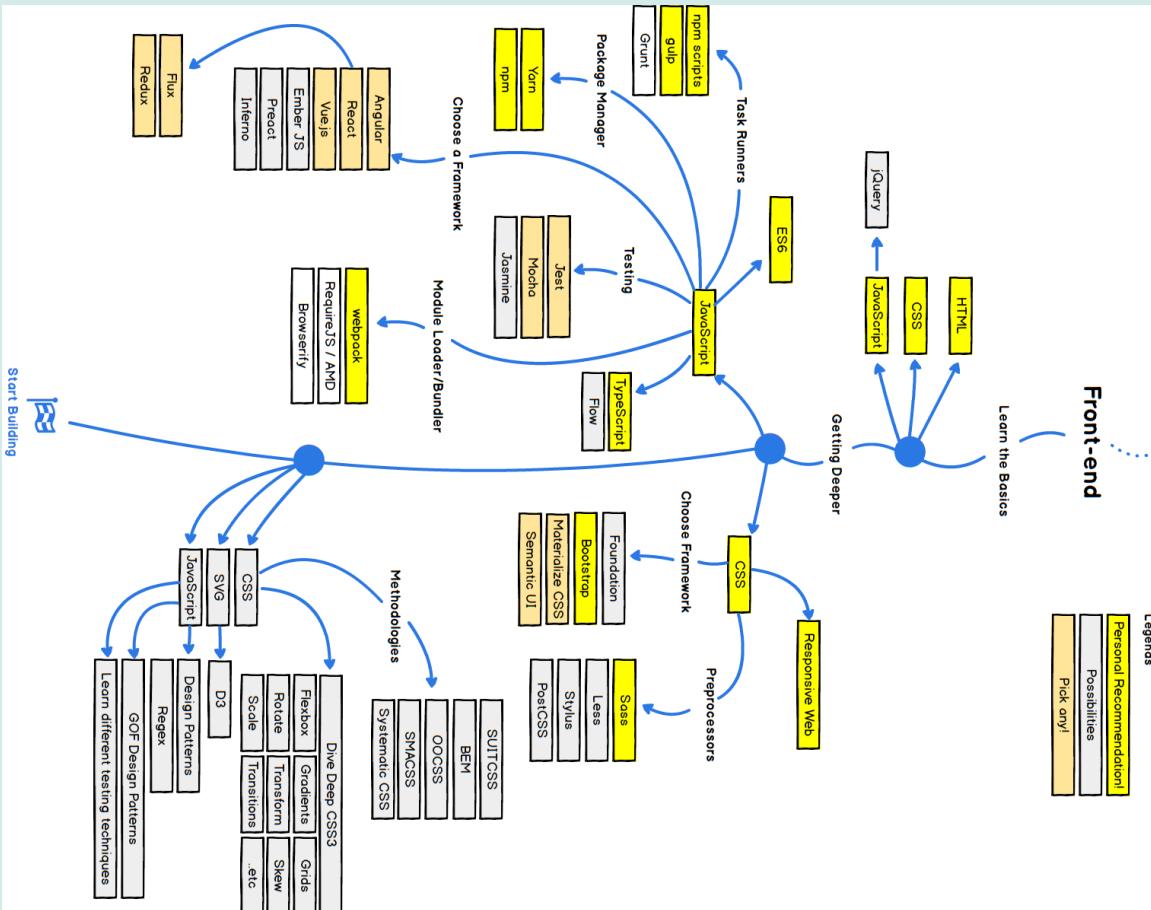
Technologie

- REST API
- React, Angular, Vue

Es gelten die "üblichen" Anforderungen

- Verständliche Architektur
- Wartbarer und langlebiger Code
- Refaktorisierbarkeit
- Testbarkeit
- Skalierbarkeit der Entwicklung

<https://medium.freecodecamp.org/a-roadmap-to-becoming-a-web-developer-in-2017-b6ac3ddd0cf>



UNSER FAHRPLAN

Bevor es losgeht:

- Vergleich mit Java schwierig
 - Unterschiedliche Konzepte etc
 - Ich mache trotzdem Vergleiche, um einzuordnen
- Empfehlungen: meine subjektive Meinung
- Grundsätzlich lieber Problemstellungen merken, als die konkreten Tools/Frameworks
 - Tools habe ich verlinkt
 - Im Anhang noch eine Link-Liste mit Lese-Empfehlungen

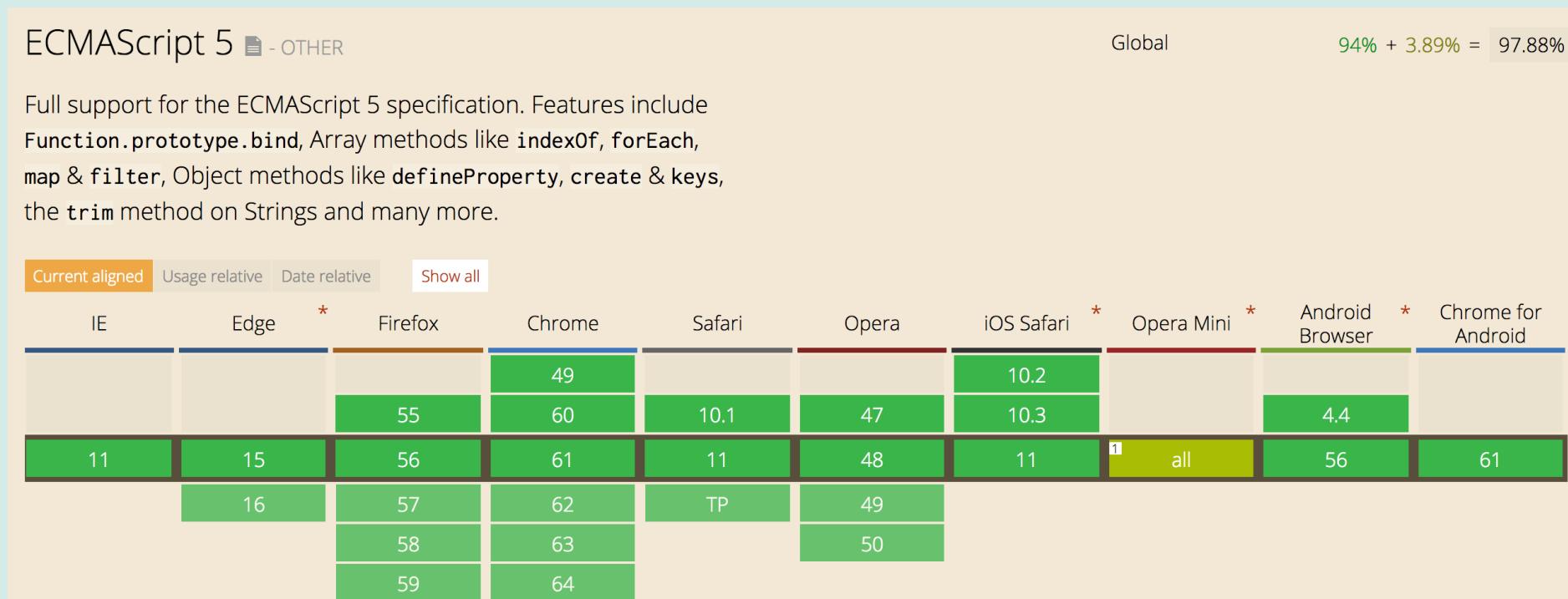
Die Sprache

JAVASCRIPT / ECMASCIPT

DIE SPRACHE

ECMAScript 5: Veröffentlicht 2009

- Unterstützung von praktisch allen Browsern
- Die "Referenz"-Version



Quelle: <https://caniuse.com/#search=es5>

- **JavaScript: Implementierung | ECMAScript: Spezifikation**

ECMAScript 2015: Veröffentlicht 2015

- Aliase: ES6, ES2015, JavaScript6, Harmony
- Künftig eine neue Version pro Jahr (ES2015, ES2016, ...)

Sehr viele Neuerungen:

- Block Scope mit let und const
- Klassen
- Module
- Arrow Funktionen
- ...

...aber leider noch kein vollständiger Browser Support!

Babel: Der "Standard" Compiler

- <https://babeljs.io/>
- Compiliert ES2015+ nach ES5
- Unterstützt auch experimentelle Sprachfeatures

TypeScript: Sprache von Microsoft inklusive Compiler

- <http://www.typescriptlang.org/>
- Typ-System für JavaScript
- Bringt Sprach-Erweiterungen mit
 - z.B. private Felder, Enum

HINTERGRUND: POLYFILLS

- Compiler / Transpiler übersetzen "nur" die Sprache
 - Retrotranslator ☺
- Polyfills sind JS Bibliotheken, die fehlende APIs implementieren
 - Stellen Abwärtskompatibilität für ältere Browser her
 - Zum Beispiel LocalStorage oder HTML5 History API

Browser: Der Klassiker

- Nahezu alle Browser implementieren ES5
- JavaScript-Support wird besser und einheitlicher
- Wettbewerb um beste Developer Tools

NodeJS: Serverseitiges JavaScript

- Basiert auf der JS Engine V8 von Chrome
 - Ermöglicht zusätzlich Zugriff auf File-System, Konsole etc
- Grundlage auch für diverses Tooling
 - Package Manager, Build, Test, ...

JavaScript Modul Systeme

- CommonJS: NodeJS Modul-System
- AMD: Asynchrone Module (für Browser)
- ES2015 spezifiziert natives Modul System

Module sind sehr fein-granular (zB eine Datei)

- Nicht direkt vergleichbar mit Java9/OSGi Modulen

"Bundler" machen aus dem Modul-Code ausführbares JavaScript

- Webpack / Browserify

Empfehlung: ES2015 Modulsystem

- In neuen Projekten mit import/export beginnen

STRUKTURIERUNG VON ANWENDUNGEN: MODULE

Beispiel ES6 Modul System

UserService.js

```
export default class UserService {  
    constructor() { . . . }  
    loadUser(id) { . . . }  
}
```

```
// Nur Modul-intern sichtbar  
const database = ...;
```

App.js

```
import UserService from "./UserService";  
  
const userService = new UserService();  
userService.loadUser(1);
```

Frameworks

JQUERY: DER KLASSIKER

jQuery (<https://jquery.com/>): Abstrahiert Zugriff auf den DOM

- Abstrahiert Verhalten unterschiedlicher Browser
- Extrem weit verbreitet und bekannt
- Gute Möglichkeit, um statischen Websites mit Logik zu "ergänzen"
 - Zum Beispiel Validierung von Eingabefeldern
 - Eher einfache Use-Cases
- Für "echte" Anwendungen nicht gut geeignet
 - Code wird schnell unübersichtlich
 - Zu Low-Level
- Empfehlung: (trotzdem) ansehen
 - Nach wie vor hohe Verbreitung (man kommt nicht drum rum)
 - Lernen, welche Probleme es mit dem DOM gibt und wie sie gelöst werden

SPA Frameworks

- Zentrales Element: Komponenten (statt DOM)
- Teilweise mit Template-Sprache, teilweise nur JavaScript (React)
- Unterschiedlich großer Funktionsumfang
 - Angular trifft sehr viele Entscheidungen (erinnert häufig an Java)
 - React sehr minimales API, wenig intrusiv
- Insgesamt sehr stabil (Angular ab Version 2)

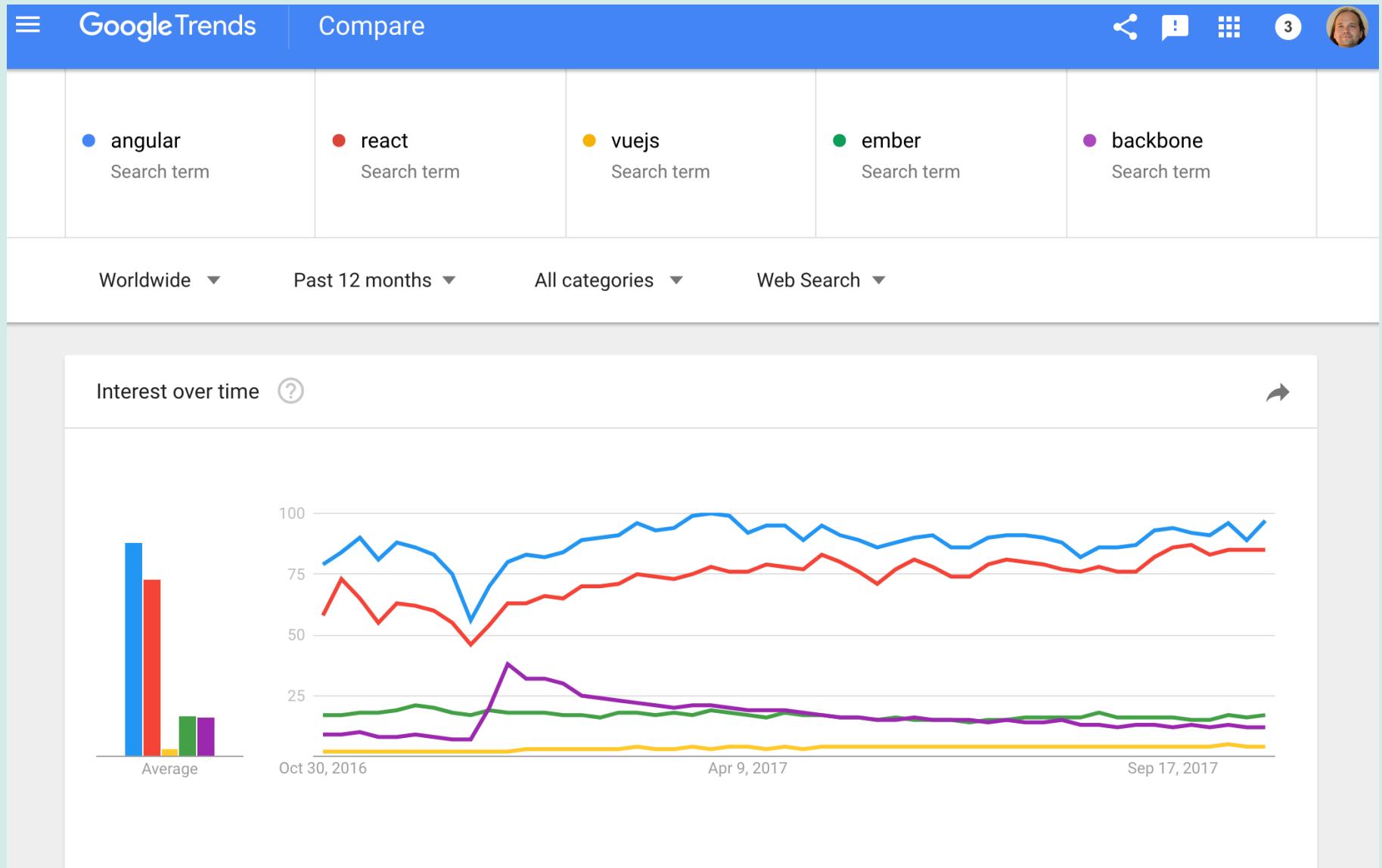
Prominente, aktuelle Vertreter:

- Angular2, React, Vue, (Ember)
- Empfehlung 1: ausprobieren, was einem am besten gefällt
- Empfehlung 2: Erst mit JavaScript (und ggf jQuery) vertraut machen

Prominente ältere Vertreter:

- Backbone, AngularJS (Angular 1)

SPA FRAMEWORKS: TRENDS



Quickstart SPA Frameworks

- Aufsetzen einer Anwendung kann sehr komplex sein
- Für alle SPA Frameworks gibt es Tools zum Aufsetzen von Projekten
 - (als npm Pakete)
- Zum schnellen Ausprobieren sehr gut geeignet
 - Nachteil: man weiß nicht, was im Hintergrund passiert
- Angular CLI: <https://cli.angular.io/>
- Create React App: <https://github.com/facebookincubator/create-react-app>
- Create Vue App: <https://github.com/vue-land/create-vue-app>
- Ember CLI: <https://ember-cli.com/>

SPA ARCHITEKTUR PATTERN

Exkurs: Architektur Pattern **Flux** und **Redux**

- **Flux** ursprünglich zur Strukturierung von React Anwendungen entwickelt
- Alternative z.B. zu MVC Pattern
- Diverse Implementierungen
- **Redux** mittlerweile sehr populär
 - Pattern und Implementierung
 - Verfügbar für React, Angular, Vue, ...
 - Sehr viele neue Konzepte, inspiriert aus funktionaler Programmierung
 - Empfehlung: erst verwenden, wenn man es wirklich braucht

Package Manager & Bundler

EXTERNE ABHÄNGIGKEITEN VERWALTEN

npm: node package manager (<https://npmjs.com>)

- Standard Package Manager für Node-Entwicklung
- Beschreibung externer Abhängigkeiten in package.json-Datei
 - Ähnlich wie in einer POM.xml
- Eigene Packages können publiziert werden
 - In zentrale Registry (analog maven-central)
 - Private Registry (z.B. Nexus)
- Über npm werden üblicherweise auch notwendige **Tools** deklariert und installiert
 - Zum Beispiel für Compiler und Build-Tools

PACKAGE MANAGER / EXTERNE ABHÄNGIGKEITEN

Alternative zu NPM: **yarn (<https://yarnpkg.com/>)**

- Verwendet ebenfalls NPM Pakete und NPM Registry
- Gleiche Beschreibung der Dependencies (package.json)
- Version Locking
- Höhere Performance
- Empfehlung: **Ausprobieren (statt npm)!**

Aus vergangenen Tagen: **Bower (<https://bower.io>)**

- War in erster Linie für Frontend Bibliotheken gedacht
- Bower-Team empfiehlt mittlerweile, npm zu verwenden
(<https://bower.io/blog/2017/how-to-migrate-away-from-bower/>)

MODULE VERWENDEN

The screenshot shows the VS Code interface with the following details:

- EXPLORER View:** Shows the project structure under "REACT-EXAMPLE-APP".
 - app folder:** Contains "components" (with "CoreComponents.tsx", "PasswordForm.jsx", "PasswordView.tsx"), "model" (with "Restrictions.ts"), and "app.tsx".
 - node_modules folder:** Contains ".bin", "@types", "react", "react-dom", "react-hot-api", "react-hot-loader", and "read-pkg".
- app.tsx Editor:** The active editor tab. The code imports styles from "styles.css" and uses internal components like "PasswordView" and external modules like "React" and "ReactDOM". It also uses the "document.getElementById" method.
- Bottom Status Bar:** Displays "typescript*" as the language, line 13, column 1, tab size 2, encoding UTF-8, line separator LF, and version 2.5.2 of the TypeScript React extension. It also shows Prettier status and a smiley face icon.

Modularisierte Anwendung: Interne und externe Module

Problem: Keine Unterstützung für Module im Browser

- Verwendete Module wurden mit <script> eingebunden
- Inhalt der Module war in der Regel global sichtbar
- Probleme:
 - (Namens)kollisionen
 - Manuelle Pflege der Abhängigkeiten (Reihenfolge!)
 - CommonJS-Module (aus Node) nicht im Browser nutzbar
- ES2015-Module im Browser noch nicht überall unterstützt

Webpack: Zentrales Build-Werkzeug

- <https://webpack.github.io/>
- Erstellt lauffähiges JavaScript-Modul ("Bundle"), quasi ein "Fat Jar"
- Unterstützung für alle Modul-Systeme
- Kann mit diversen Dateitypen umgehen (nicht nur JavaScript)

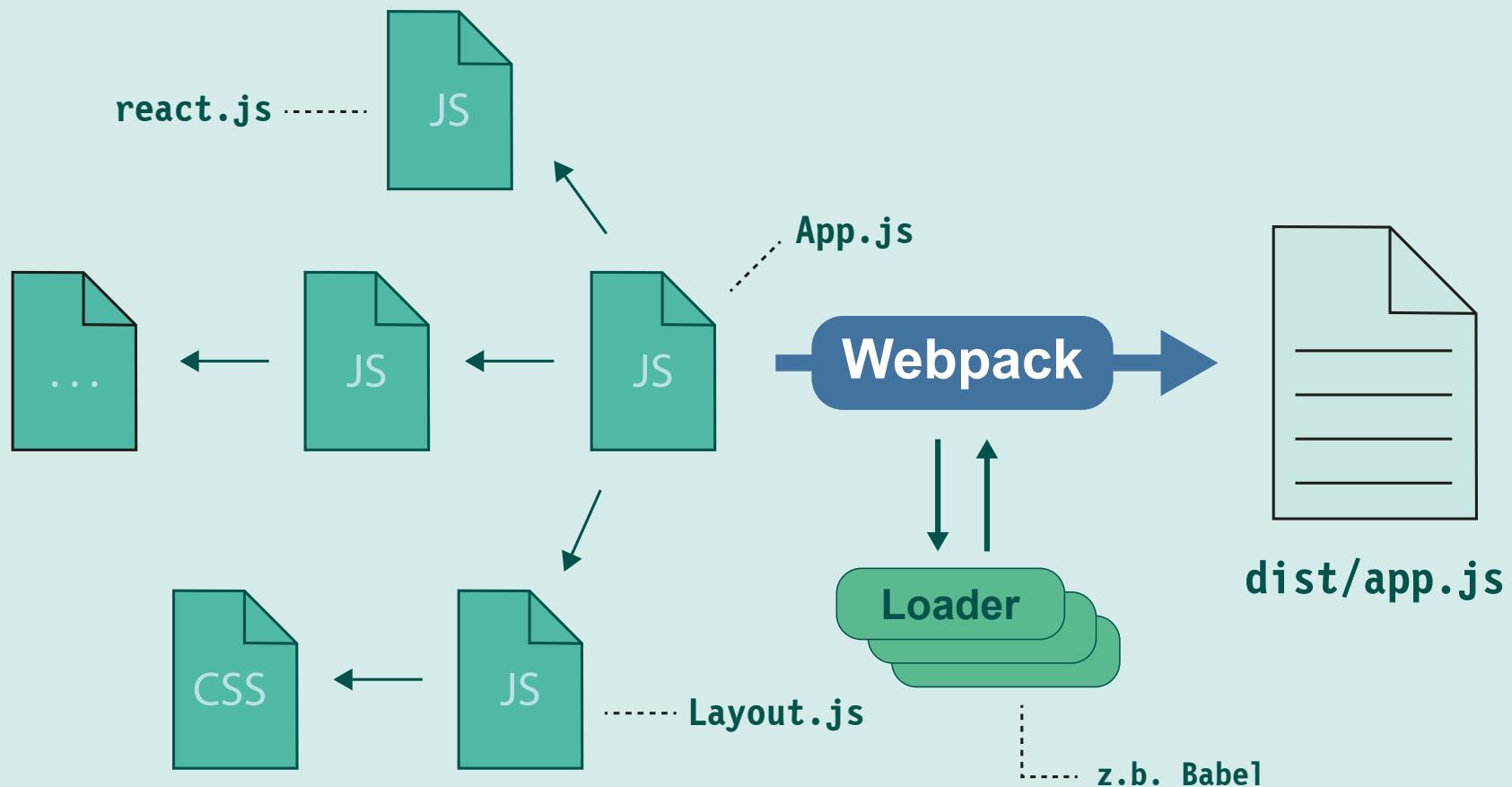
Alternativen: Browserify, Rollup

- Empfehlung: Webpack ist "quasi-standard", deswegen benutzen

VERWENDUNG EXTERNER MODULE

Webpack: Zentrales Build-Werkzeug

- Basiert auf statischer Code-Analyse



Webpack Dev Server: Webserver für Entwicklung

- Integration mit Webpack
- Änderungen im Code werden automatisch gebaut
- Änderungen werden automatisch im Browser aktualisiert
- State bleibt erhalten
- Empfehlung: für lokale Entwicklung verwenden; ideal für kurze Roundtrips

Automatisierung & Build

Wofür Automatisierung?

- Compilieren / Transpilieren / Bundlen
- Tests ausführen
- Releases erstellen und publizieren
 - Zum Beispiel Code minifizieren
- Server zum Entwickeln starten

npm scripts: Ausführen von (Shell) Scripts und Node-Apps

- Oftmals müssen nur einfache Tasks erledigt werden
- Installierte Node-Tools lassen sich direkt aufrufen
- Scripts werden in package.json eingetragen

```
package.json  {
    "name": "...",
    "scripts": {
        "clean": "rm -rf public/dist/",
        "dist": "webpack --dist",
        "test": "mocha test",
        "all": "npm run clean && npm run dist && npm test"
    },
    "dependencies": { . . . }
}
```

Kommandozeile \$ **npm run all**

grunt und gulp: Komplexe Task-Runner

- Erlauben das Schreiben von kompletten Abläufen (in JavaScript)
- Benötigte Tools (z.B. Webpack, Babel etc) werden als Plug-ins eingebunden
- Oftmals overkill und zu viele Abstraktionen
- Empfehlung: npm scripts verwenden
 - Why I left Gulp and Grunt for npm scripts:
<https://medium.freecodecamp.org/why-i-left-gulp-and-grunt-for-npm-scripts-3d6853dd22b8>
 - How to use npm as a Build Tool:
<https://www.keithcirkel.co.uk/how-to-use-npm-as-a-build-tool/>

Qualitätssicherung und Testen

Statische Code-Analyse: ESLint (<https://eslint.org/>)

- Findet typische JavaScript Programmierfehler
- Achtet auf Einhaltung von Konventionen (z.B. Semikolon ja/nein)

The screenshot shows a code editor window for a file named `example.js`. The code contains a function `identical` that checks if two variables are equal. ESLint has flagged several issues:

```
1
2  function identical(a, b) {
3      if (a == b) {
4          return
5          true;
6      }
7  }
```

Below the code editor, the `PROBLEMS` tab is selected in the bottom navigation bar. The list of problems shows three ESLint errors:

- `[eslint] Expected '===' and instead saw '=='. (eqeqeq) (3, 8)`
- `[eslint] Expected an assignment or function call and instead saw 'true;'. (no-unreachable) (5, 3)`
- `[eslint] Unreachable code. (no-unreachable) (5, 3)`

Typ-System für JavaScript

- Fehler schon zur Build-Zeit finden
- Zwei prominente Vertreter, TypeScript und Flow
- Beide syntaktisch sehr ähnlich
- Typ-Angaben sind optional (nur da, wo man sie braucht/will)

Typ-System für JavaScript

- Fehler schon zur Build-Zeit finden
- Zwei prominente Vertreter, TypeScript und Flow
- Beide syntaktisch sehr ähnlich
- Typ-Angaben sind optional (nur da, wo man sie braucht/will)

TypeScript (<http://www.typescriptlang.org/>):

- Entwickelt von Microsoft
- Bringt auch eigene Sprach-Erweiterungen mit (z.B. Enums)
- Angular2 ist mit TypeScript gebaut
- Sehr guter IDE Support (insb Visual Studio Code, WebStorm)

Typ-System für JavaScript

- Fehler schon zur Build-Zeit finden
- Zwei prominente Vertreter, TypeScript und Flow
- Beide syntaktisch sehr ähnlich
- Typ-Angaben sind optional (nur da, wo man sie braucht/will)

TypeScript (<http://www.typescriptlang.org/>):

- Entwickelt von Microsoft
- Bringt auch eigene Sprach-Erweiterungen mit (z.B. Enums)
- Angular2 ist mit TypeScript gebaut
- Sehr guter IDE Support (insb Visual Studio Code, WebStorm)

Flow (<https://flow.org/>):

- Entwickelt von Facebook
- Wird viel im React-Umfeld genutzt

Typ-System für JavaScript

- Fehler schon zur Build-Zeit finden
- Zwei prominente Vertreter, TypeScript und Flow
- Beide syntaktisch sehr ähnlich
- Typ-Angaben sind optional (nur da, wo man sie braucht/will)

TypeScript (<http://www.typescriptlang.org/>):

- Entwickelt von Microsoft
- Bringt auch eigene Sprach-Erweiterungen mit (z.B. Enums)
- Angular2 ist mit TypeScript gebaut
- Sehr guter IDE Support (insb Visual Studio Code, WebStorm)

Flow (<https://flow.org/>):

- Entwickelt von Facebook
- Wird viel im React-Umfeld genutzt

Empfehlung: Wenn Angular, dann auf jeden Fall TypeScript, ansonsten nach Geschmack (persönliche Präferenz: TypeScript)

BEISPIEL: TYPESCRIPT

The screenshot shows a code editor window for a file named `basic.ts`. The code contains several TypeScript statements and a warning message:

```
1 let count = 7;
2
3 // Type Inference: x ist eine Zahl
4 const x = count.toUpperCase();
5 count = "Geht nicht";
6
7 function sayHello(name: string) {
8   console.log(`Hello, ${name}`);
9 }
10
11 [ts] Argument of type '666' is not assignable to parameter
12   of type 'string'.
13 sayHello(666);
14
15
16
17 sayHello("Welt");
18
```

A tooltip is displayed over the line `sayHello(666);`, showing the error message: `[ts] Argument of type '666' is not assignable to parameter of type 'string'.`

At the bottom of the screen, the VS Code interface is visible, including the `PROBLEMS` tab which lists the following errors:

- `basic.ts (3)`
 - `[ts] Property 'toUpperCase' does not exist on type 'number'. (4, 17)`
 - `[ts] Type ""Geht nicht"" is not assignable to type 'number'. (5, 1)`
 - `[ts] Argument of type '666' is not assignable to parameter of type 'string'. (13, 10)`

Testen in JavaScript

- Sehr viele Ansätze, Tools und Frameworks
- Eigener Talk
- Gute Übersicht über den aktuellen Stand:
<https://medium.com/powtoon-engineering/a-complete-guide-to-testing-javascript-in-2017-a217b4cd5a2a>



Mocha: Modularer Ansatz

- Testrunner: Mocha (<https://mochajs.org/>)
- Assertions: Chai (<http://chaijs.com/>)
- Mocking Bibliothek: Sinon (<http://sinonjs.org/>)
- Code Coverage: Istanbul (<https://istanbul.js.org/>)

Jest: Alles-inkusive-Lösung

- Enthält alles was man zum testen braucht, inkl JSDom
- Entstanden im React-Umfeld

Jasmine: Batteries included (<https://jasmine.github.io>)

- Testrunner
- Assertions
- Mocking Bibliothek

Jest: "Delightful JavaScript Testing" (<https://facebook.github.io/jest/>)

▲ jcoffland 327 days ago [-]

There's no such thing as painless testing.

<https://news.ycombinator.com/item?id=13128146#13128900>

- All-inclusive-Lösung
 - Testrunner, Assertions, Mocks, Coverage, JSDom
 - Integration mit Babel und TypeScript
- Besonderheit: Snapshot-Testing
- Entstanden im React-Umfeld

Empfehlung: Jasmine oder Jest. Wenn mit React entwickelt wird, auf jeden Fall Jest.

Code-Beispiel Jest

```
sum.js    export function sum(a,b) {  
            return a+b  
};  
  
sum.test.js import {sum} from '../sum.js';  
  
test('sum of 2 and 2 is 4', function() {  
    expect(sum(2, 2)).toBe(4);  
});  
  
test('sum of 2 and 2 is not 3', function() {  
    expect(sum(2, 2)).not.toBe(3);  
});
```

Fazit

...UND ZUSAMMENFASSUNG

FAZIT UND ZUSAMMENFASSUNG

...alle Tools nochmal auf einen Schlag und Vergleich mit Java

FAZIT UND ZUSAMMENFASSUNG

Wir bauen "echte" ("Enterprise") Anwendungen

- **JavaScript != Java**
- **JavaScript != Script**
- Entsprechend komplex der Tool-Stack (wie in Java)
- Sprache seit ES2015 deutlich brauchbarer
- Typ-System für echte System unabdingbar
- Empfehlung: Konzepte und Probleme verstehen, dann erst konkrete Tools

WEITERFÜHRENDE LINKS

State of the JavaScript Landscape - A Map for Newcomers

<https://www.infoq.com/articles/state-of-javascript-2016>

Modern JavaScript Explained For Dinosaurs

<https://medium.com/@peterxjang/modern-javascript-explained-for-dinosaurs-f695e9747b70>

Grab Front End Guide

<https://github.com/grab/front-end-guide/blob/master/README.md>

Die große JavaScript Erschöpfung

<https://jaxenter.de/die-grosse-javascript-erschoepfung-36278>

Zugabe

Lebenszyklen im Vergleich

Empfehlung: Traue keiner Statistik, die Du nicht selbst gefälscht hast!

Sprache

- ES5 (12/2009) bis ES2015: (6/2015): 5,5 Jahre
- Zum Vergleich: Java8 auf Java9: 3,5 Jahre
- Ab 2015: jährliche Releases von JavaScript, Java ab 2018: halb-jährliche Releases
- Wer ist hier hektisch? ☺

jQuery

- Erste Version Januar 2006, aktuelles jQuery ist noch weitgehend API-kompatibel
- Zum Vergleich: JSF erste Version 2004

Node Package Manager

- 1. Release: 12. Januar 2010.
- Zum Vergleich: Maven 3.0 erschienen Oktober 2010

SPA Frameworks

Angular 1: 2009 | Angular 2: 2016 | React: Open-Source seit 2013 | VueJS: 2013

Vielen Dank!

<http://bit.ly/wjax2017-javascript>

Fragen?

[HTTPS://NILSHARTMANN.NET](https://nilshartmann.net) | @NILSHARTMANN