

NILS HARTMANN | @NILSHARTMANN

Das JavaScript Öko-System

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

W-JAX MÜNCHEN | NOVEMBER 2017

NILS HARTMANN

Programmierer und Architekt aus Hamburg

**Java
JavaScript
Trainings und Workshops**

KONTAKT: NILS@NILSHARTMANN.NET

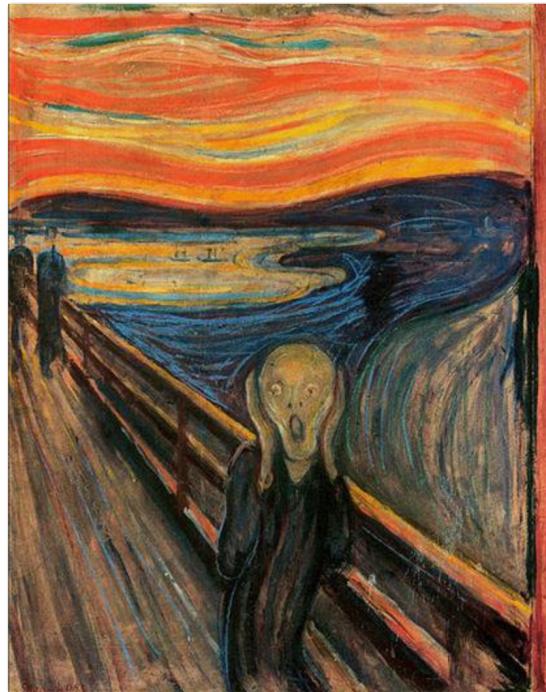
Lukas Eder
@lukaseder

Folgen



Still one of my favourite paintings:
Edvard Munch: The JavaScript, 1893

Original (Englisch) übersetzen



11:00 ~ 15. Okt. 2016

285 Retweets 442 „Gefällt mir“-Angaben



7



285



442



Einleitung

<https://twitter.com/lukaseder/status/787216648642109441>



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>



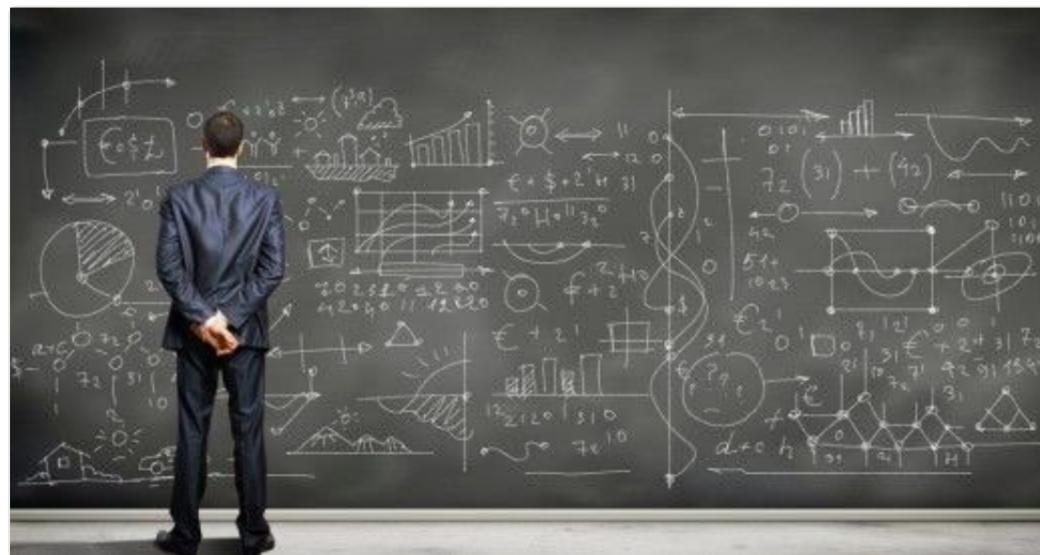
Thomas Fuchs

@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!?"

JAVA - EINE KOMPLETTE PLATTFORM

Java: Bringt alles mit was wir zum Entwicklung brauchen

Tools, Bibliotheken, Laufzeitumgebung, ...

Alles aus einer Hand

Java Language												
Java Language	java	javac	javadoc	jar	javap	jdeps	Scripting					
Tools & Tool APIs	Security	Monitoring	JConsole	VisualVM	JMC	JFR						
Deployment	JPDA	JVM TI	IDL	RMI	Java DB	Deployment						
Deployment	Internationalization		Web Services		Troubleshooting							
User Interface Toolkits	Java Web Start			Applet / Java Plug-in								
Integration Libraries	JavaFX											
Integration Libraries	Swing		Java 2D		AWT	Accessibility						
Integration Libraries	Drag and Drop		Input Methods		Image I/O	Print Service	Sound					
Other Base Libraries	IDL	JDBC	JNDI	RMI	RMI-IIOP		Scripting					
Other Base Libraries	Beans	Security		Serialization		Extension Mechanism						
Other Base Libraries	JMX	XML JAXP		Networking		Override Mechanism						
lang and util Base Libraries	JNI	Date and Time		Input/Output		Internationalization						
lang and util												
lang and util Base Libraries	Math	Collections		Ref Objects		Regular Expressions						
lang and util Base Libraries	Logging	Management		Instrumentation		Concurrency Utilities						
lang and util Base Libraries	Reflection	Versioning		Preferences API		JAR	Zip					
Java HotSpot Client and Server VM												

Grafik: <https://docs.oracle.com/javase/8/docs/index.html>

JavaScript: Nur Sprache

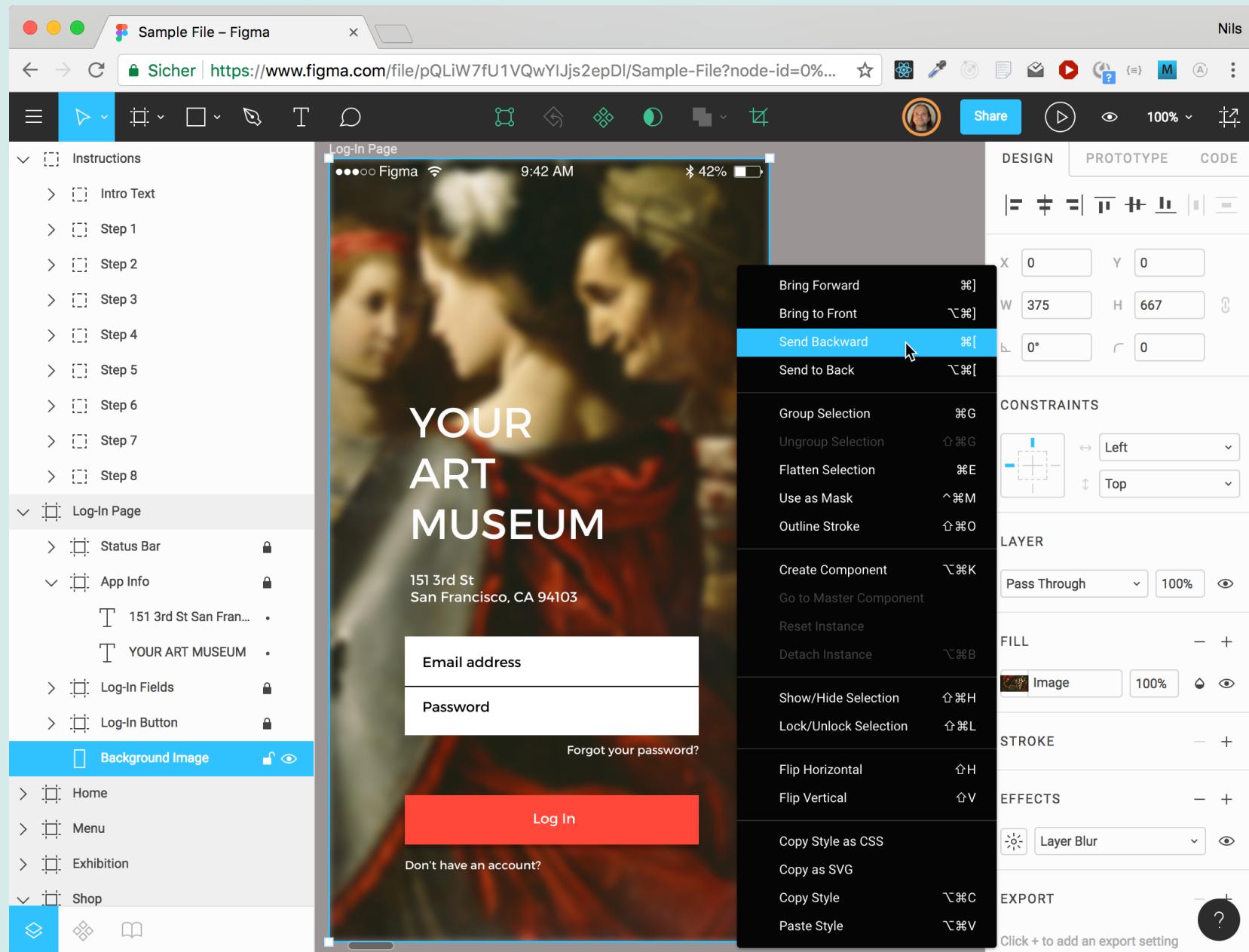
- Keine zentrale Organisation (abgesehen vom Sprachstandard)
- Keine Standard Bibliothek (nur minimal)
- Kein Typ-System
- Kein Compiler
- Keine einheitliche Laufzeitumgebung (analog zum JRE)
- Kein Modul-System

Warum dann überhaupt JavaScript?

- Browser / Web als **die** zentrale Plattform für Applikationen
- Kein Deployment notwendig
- Browser sind sehr schnell und leistungsfähig
 - Laufen auf diversen Geräten
- Sehr hohes Innovationstempo
 - erfordert stetig neue Lösungen
- Fehlende Tools, Bibliotheken werden "dezentral" entwickelt

The screenshot shows the Spotify Web Player interface. At the top, there's a navigation bar with icons for back, forward, search, and user profile ('Nils'). The URL in the address bar is <https://open.spotify.com/artist/56F64pmwSSCcmS1CxAnPk8>. The main content area displays the artist 'Pro-Pain' with a black and white photo of a man with a beard. The artist has 10.892 FOLLOWER. Below the photo are three buttons: PAUSE (green), FOLGEN (white), and a more options menu (...). To the left of the photo is a sidebar with a 'Suchen' search bar, a 'Start' button, and a 'Deine Musik' section. Under 'Deine Musik', it shows 'ZULETZT ABGESPIELT' with items: 'Pro-Pain' (KÜNSTLER), 'Pro-Pain — Voice O...' (PLAYLIST), 'Iron Maiden HH 2017' (PLAYLIST), and 'Haukur Tomasson' (PLAYLIST). At the bottom of the sidebar are buttons for 'App installieren' and a user profile for 'nils_hartmann'. The main content area also includes tabs for 'ÜBERSICHT' (selected), 'ÄHNLICHE KÜNSTLER', and 'INFORMATIONEN'. A 'Beliebt' section lists five songs: 1. 'Voice Of Rebellion' (4:01), 2. 'Deathwish' (2:45), 3. 'Foul Taste Of Freedom' (3:41), 4. 'One Shot One Kill' (2:55), and 5. 'Make War (Not Love)' (4:54). A context menu is open over the fourth song, listing options: 'Radio starten', 'In „Deine Musik“ speichern' (with a cursor arrow pointing here), 'Zu Playlist hinzufügen', and 'Songlink kopieren'. The bottom of the screen shows the playback controls: a track list with 'Voice Of Rebellion' by 'Pro-Pain' currently playing, a progress bar at 1:15 of 4:01, and standard play/pause, skip, and volume controls.

SPOTIFY WEB PLAYER



[HTTPS://WWW.FIGMA.COM](https://www.figma.com)



Thomas Fuchs

@thomasfuchs

Folgen

▼

1997: Let's make a website!

fires up vi

2007: Let's make a website!

downloads jQuery

fires up vi

2017: Let's make a website!



Original (Englisch) übersetzen

This is a starter boilerplate app I've put together using the following technologies:

- [Isomorphic Universal](#) rendering
- Both client and server make calls to load data from separate API server
- [React](#)
- [React Router](#)
- [Express](#)
- [Babel](#) for ES6 and ES7 magic
- [Webpack](#) for bundling
- [Webpack Dev Middleware](#)
- [Webpack Hot Middleware](#)
- [Redux](#)'s futuristic [Flux](#) implementation
- [Redux Dev Tools](#) for next generation DX (developer experience). Watch [Dan Abramov's talk](#).
- [React Router Redux](#) Redux/React Router bindings.
- [ESLint](#) to maintain a consistent code style
- [redux-form](#) to manage form state in Redux
- [Iru-memoize](#) to speed up form validation
- [multireducer](#) to combine single reducers into one key-based reducer
- [style-loader](#), [sass-loader](#) and [less-loader](#) to allow import of stylesheets in plain css, sass and less,
- [bootstrap-sass-loader](#) and [font-awesome-webpack](#) to customize Bootstrap and FontAwesome
- [react-helmet](#) to manage title and meta tag information on both server and client
- [webpack-isomorphic-tools](#) to allow require() work for statics both on client and server
- [mocha](#) to allow writing unit tests for the project.

20:13 - 22. Feb. 2017

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

WEBSITE ODER WEB-ANWENDUNG?

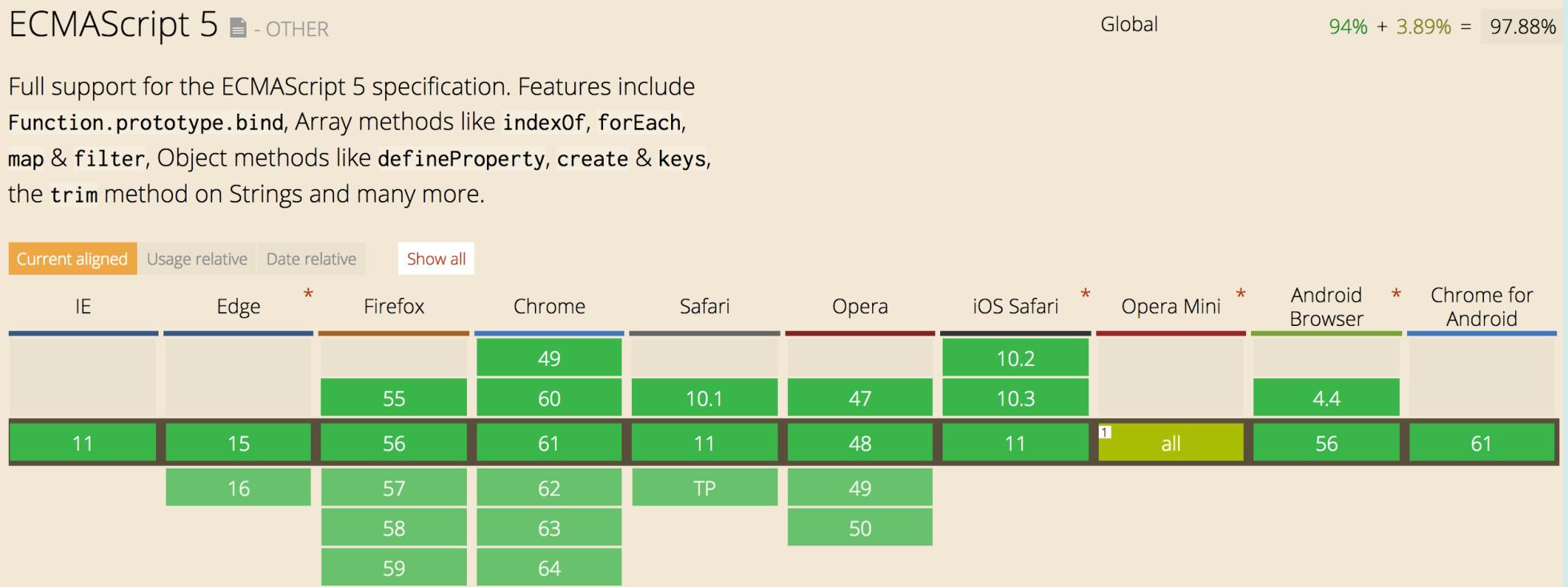
Die Sprache

JAVASCRIPT / ECMASCIPT

DIE SPRACHE

ECMAScript 5: Veröffentlicht 2009

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



- **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 und Module
- Arrow Funktionen
- Map, Set, WeakMap, WeakSet

Löst viele "klassische" JavaScript-Probleme

- (teilweise) Sichtbarkeiten, Hoisting, this-Binding
- <https://github.com/you-dont-need/You-Dont-Need-Lodash-Underscore>

ES6 SUPPORT

Feature name	Current browser	97%	99%	99%	99%	99%	Node ≥8.7 <9 ^[5]	CH 62, OP 49 ^[1]	CH 61, OP 48 ^[1]	97%	97%	97%	96%	96%	95%	94%	71%	59%	59%	56%	52%	48%	24%	17%	11%	
Syntax																										
default function parameters	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	6/7	4/7	0/7	5/7	4/7	0/7	5/7	0/7	0/7	0/7	0/7
rest parameters	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	3/5	0/5	4/5	4/5	0/5	2/5	0/5	0/5	0/5	0/5
spread (...) operator	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	13/15	11/15	4/15	15/15	0/15	12/15	0/15	0/5	0/5	0/15	
object literal extensions	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	5/6	6/6	6/6	6/6	4/6	4/6	0/5	0/5	0/6	
for..of loops	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	7/9	9/9	8/9	3/9	9/9	7/9	6/9	0/9	0/9	0/9	
octal and binary literals	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4
template literals	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	4/5	5/5	3/5	4/5	5/5	3/5	0/5	0/5	0/5	0/5
RegExp "y" and "u" flags	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	3/5	0/5	0/5	3/5	0/5	0/5	0/5	0/5	0/5	0/5
destructuring declarations	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	21/22	21/22	19/22	15/22	20/22	0/22	20/22	0/22	0/2	0/22	
destructuring assignment	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	23/24	24/24	21/24	19/24	23/24	0/24	21/24	0/24	0/4	0/24	
destructuring parameters	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	21/24	18/24	16/24	19/24	0/24	18/24	0/24	0/4	0/24		
Unicode code point escapes	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2
new.target	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	0/2	0/2	
Bindings																										
const	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	14/16	10/16	14/16	14/16	9/16	14/16	2/16	0/6	12/16		
let	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	10/12	0/12	10/12	10/12	6/12	10/12	0/12	0/2	10/12		
block-level function declaration ^[14]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No	Yes		
Functions																										
arrow functions	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	9/13	0/13	9/13	11/13	9/13	10/13	0/13	0/3	0/13		
class	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	19/24	18/24	19/24	17/24	0/24	13/24	0/24	0/4	0/24		
super	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	4/8	7/8	7/8	7/8	0/8	6/8	0/8	0/3	0/8		
generators	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	25/27	24/27	0/27	0/27	24/27	20/27	16/27	0/27	0/7	0/27	
Built-ins																										
typed arrays	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	45/46	45/46	0/46	43/46	0/46	20/46	0/6	16/46			
Map	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	18/19	19/19	18/19	19/19	14/19	17/19	0/19	15/19	8/19		
Set	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	18/19	19/19	18/19	19/19	14/19	17/19	0/19	15/19	8/19		
WeakMap	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	11/12	12/12	11/12	12/12	6/12	11/12	9/12	0/12	6/12		
WeakSet	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	10/11	11/11	10/11	11/11	5/11	10/11	8/11	0/11	0/11		
Proxy	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	0/34	0/34	0/34	0/34	0/34	0/34	15/34	0/4	0/34		
Reflect	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	16/20	20/20	15/20	0/20	15/20	0/20	14/20	14/20	14/20	0/20	
Promise	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	0/8	8/8	4/8	7/8	7/8	0/8	7/8	0/8	0/8	
Symbol	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	9/12	12/12	8/12	4/12	10/12	2/12	12/12	2/2	0/12		
well-known symbols ^[21]	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	25/26	26/26	22/26	15/26	1/26	3/26	1/26	1/26	0/6	0/26	
Built-in extensions																										
Object static methods	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	3/4	4/4	3/4	3/4	4/4	2/4	4/4	2/4	1/4		
function "name" property	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	16/17	16/17	17/17	12/17	8/17	14/17	3/17	0/17	6/17	0/17	7/17	0/7	0/17		
String static methods	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	1/2	1/2	2/2	0/2	
String.prototype methods	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	9/10	10/10	9/10	8/10	10/10	7/10	7/10	7/0	0/10		
RegExp.prototype properties	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	5/6	6/6	0/6	0/6	1/6	1/5	0/6			
Array static methods	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	8/11	9/11	10/11	9/11	5/11	0/11	7/1	0/11	0/11		

COMPILER: WENN DER BROWSER SUPPORT NICHT AUSREICHT

Babel: Der "Standard" Compiler

- <https://babeljs.io/>
- Compiliert ES2015+ nach ES5
- Durch Plug-ins erweiterbar (eigenes Ökosystem ...)
- 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, ...

STRUKTURIERUNG VON ANWENDUNGEN: MODULE

JavaScript Modul Systeme

- CommonJS: NodeJS Modul-System ("require ..." / "module.exports")
- 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

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 und Bibliotheken

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

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)

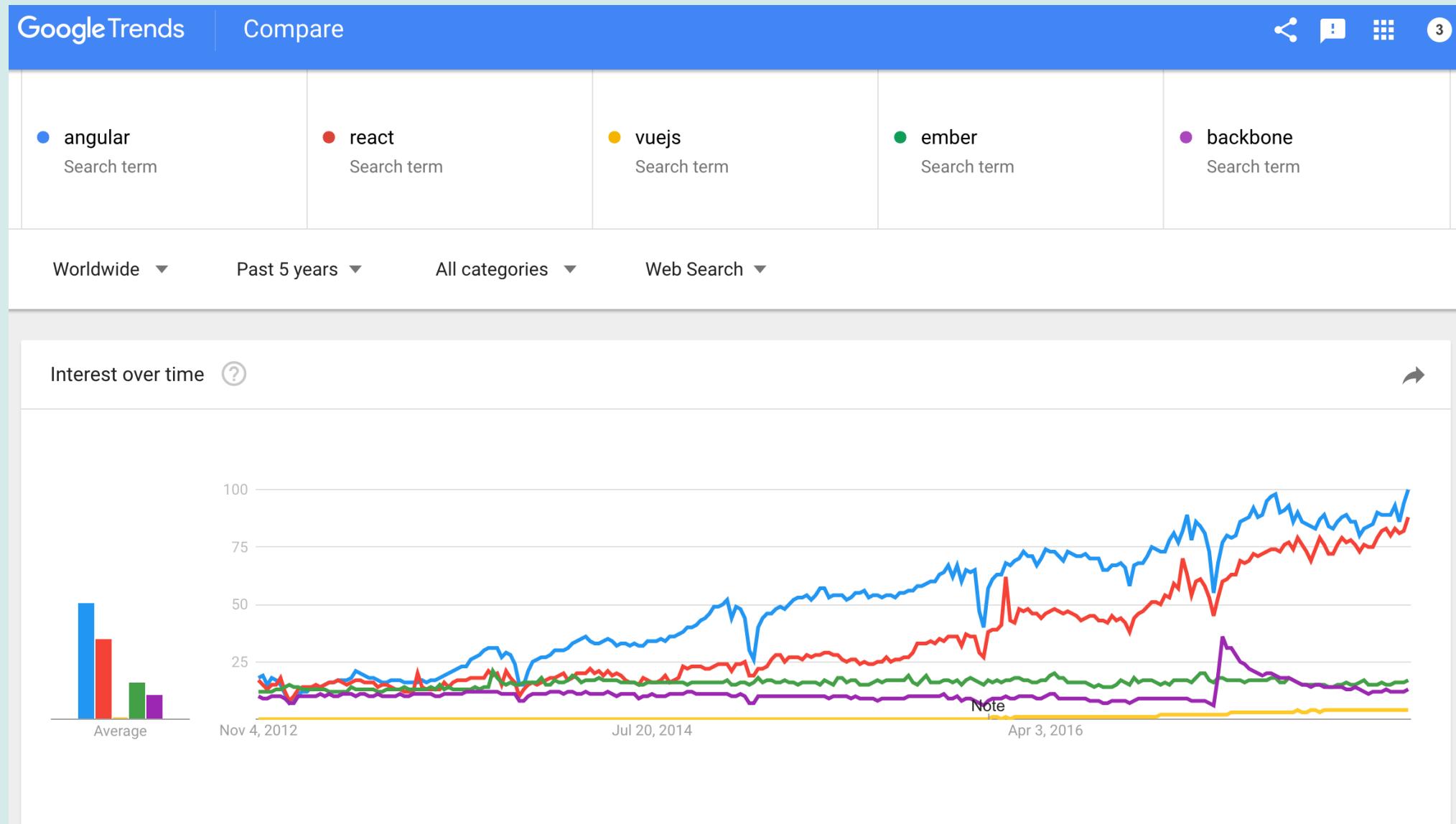
Prominente, aktuelle Vertreter:

- 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



SPA FRAMEWORKS

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
- 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/>
- Hilfe bei der Auswahl eines Frameworks: <http://todomvc.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

PACKAGE MANAGER / EXTERNE ABHÄNGIGKEITEN

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
 - components folder containing CoreComponents.tsx, PasswordForm.jsx, and PasswordView.tsx
 - model folder containing Restrictions.ts
 - app.tsx (selected)
 - node_modules folder containing .bin, @types, react, react-dom, react-hot-api, react-hot-loader, and read-pkg
- app.tsx Editor:** Displays the following TypeScript code:

```
1 import "./styles/styles.css";
2
3 import * as React from "react";
4 import * as ReactDOM from "react-dom";
5 import PasswordView from "./components/PasswordView";
6
7 ReactDOM.render(
8   <div className="ApplicationView">
9     <PasswordView />
10  </div>,
11  document.getElementById("mount")
12);
13
```
- Bottom Status Bar:** Shows the following information: TypeScript*, 0 down 1 up, 0 errors 0 warnings, Ln 13, Col 1, Tab Size: 2, UTF-8, LF, TypeScript React 2.5.2, Prettier: ✓, 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 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

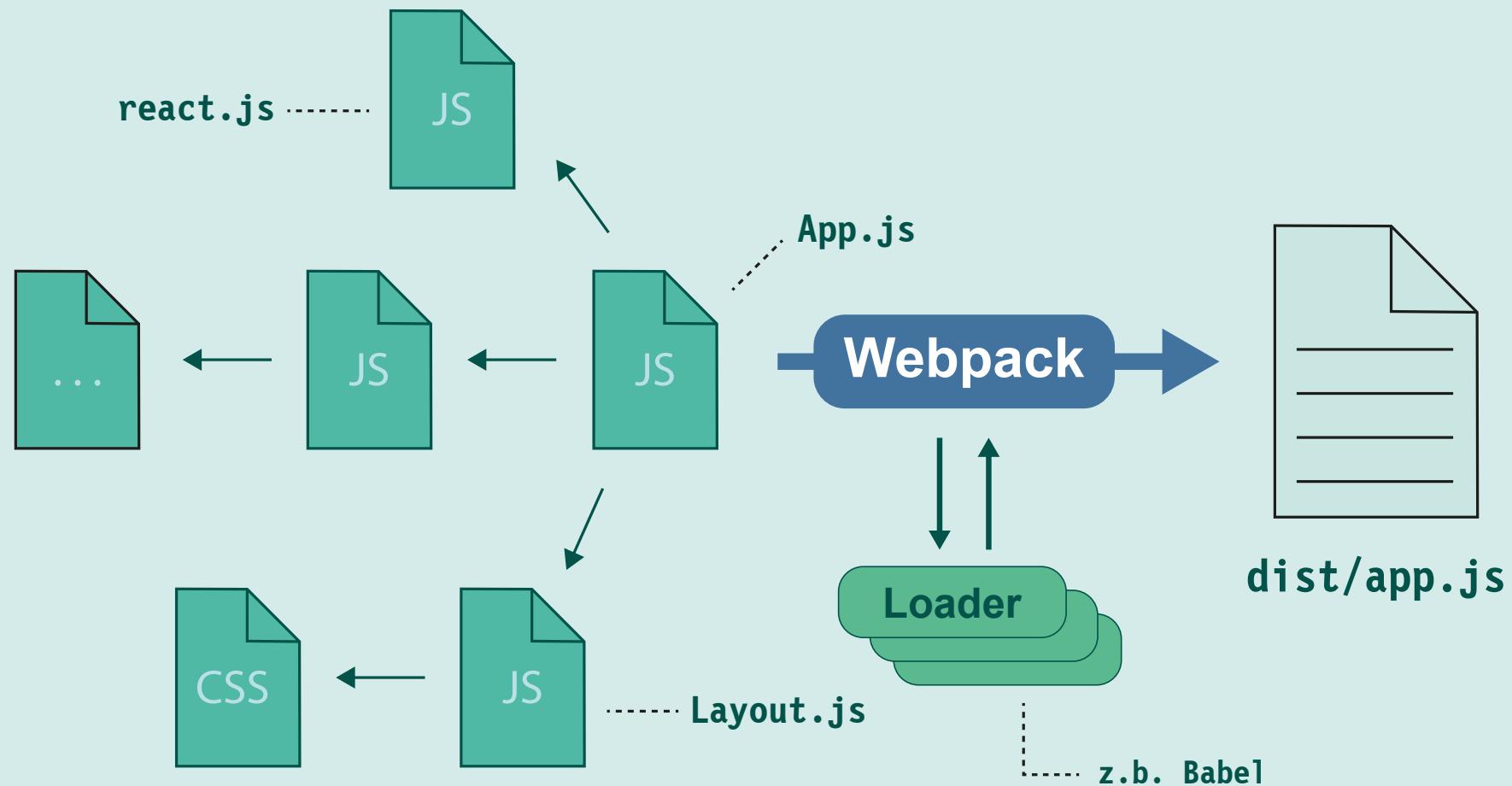
- <https://webpack.github.io/>
- Erstellt lauffähiges JavaScript-Modul ("Bundle"), quasi ein "Fat Jar"
- Unterstützung für alle Modul-Systeme
- Kann mit diversen Dateitypen umgehen (nicht nur JavaScript)
- Ein eigenes Ökosystem...

Alternativen: Browserify, Rollup

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

VERWENDUNG EXTERNER MODULE

Webpack: Zentrales Build-Werkzeug



Automatisierung & Build

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
- Funktioniert auch mit yarn

```
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)
- Kann in den CI-Build eingebunden werden

The screenshot shows a code editor window with a tab labeled "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  }
8
```

The "PROBLEMS" tab is selected in the bottom navigation bar. The list of errors is as follows:

- Line 3: [eslint] Expected '===' and instead saw '=='. (eqeqeq) (3, 8)
- Line 5: [eslint] Expected an assignment or function call and instead saw 'true'. (no-unreachable) (5, 3)
- Line 5: [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 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 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 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-specific features and errors:

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

At the bottom, the `PROBLEMS` tab is active, showing three errors:

- [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/>)

TESTEN

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

ECMAScript 2015 ("ES6") bringt viele gute Neuerungen, vereinfacht die Entwicklung

- Browser Support wird immer besser

Keine zentrale Instanz, die für uns Tools, Libs etc erstellt

- Ökosystem entwickelt sich sehr schnell (und in viele Richtungen)
- Lösungen werden in "Eigenregie" entwickelt

Compiler und Polyfills für Abwärtskompatibilität

- TypeScript und Babel

Bundler um (u.a.) Module im Browser zu nutzen

- Webpack

Verwalten von Package-Abhängigkeiten

- npm oder yarn



I Am Developer

@iamdevloper

Folgen



I think I've had milk last longer than some
JavaScript frameworks.

Original (Englisch) übersetzen

13:22 - 4. Dez. 2014

1.790 Retweets 1.162 „Gefällt mir“-Angaben



33



1,8 Tsd.



1,2 Tsd.



<https://twitter.com/iamdevloper/status/540481335362875392>

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

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>

Vielen Dank!

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

Fragen?

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