

NILS HARTMANN | @NILSHARTMANN

Das JavaScript Ökosystem

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

NILS HARTMANN

Programmierer und Architekt aus Hamburg

**Java
JavaScript
Trainings und Workshops**

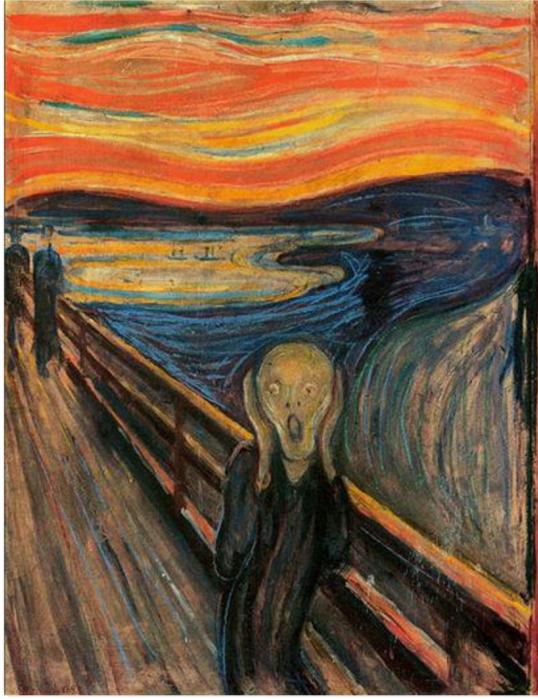
KONTAKT: NILS@NILSHARTMANN.NET

 **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

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

Einleitung

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!?"



I Am Developer
@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

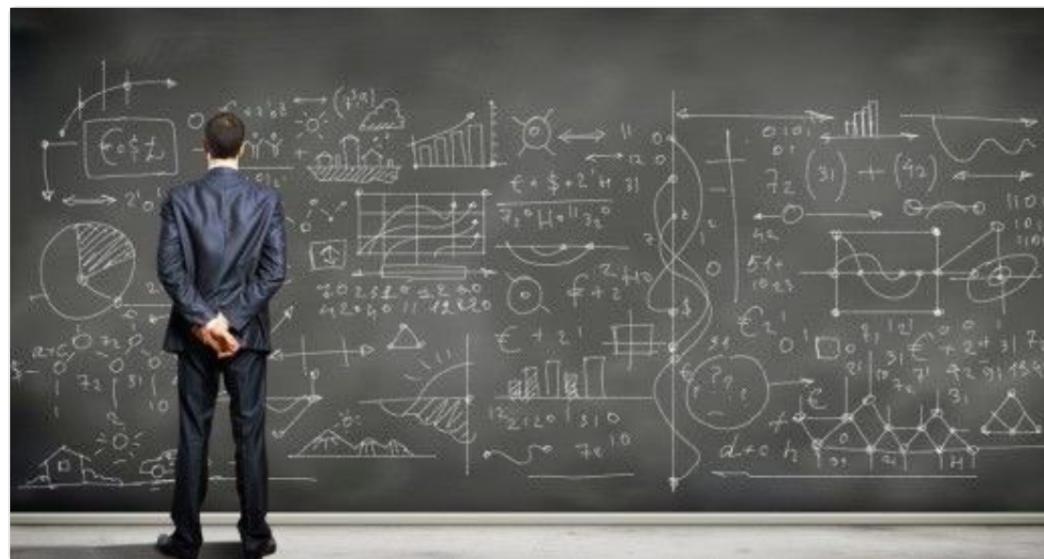
@thomasfuchs

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>

Warum ist das so?

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 Language		java	javac	javadoc	jar	javap	jdeps	Scripting					
Tools & Tool APIs		Security	Monitoring	JConsole	VisualVM	JMC	JFR						
		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											
Other Base Libraries		Swing		Java 2D		AWT	Accessibility						
lang and util		Drag and Drop		Input Methods		Image I/O	Print Service	Sound					
Base Libraries		IDL	JDBC	JNDI	RMI	RMI-IIOP		Scripting					
Java Virtual Machine		Beans	Security		Serialization		Extension Mechanism						
		JMX	XML JAXP		Networking		Override Mechanism						
		JNI	Date and Time		Input/Output		Internationalization						
lang and util													
		Math	Collections	Ref Objects		Regular Expressions							
		Logging	Management	Instrumentation		Concurrency Utilities							
		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
- Kein Threading
- Wenig Konventionen (keine klassische Maven Projektstruktur zB)

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

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

Einerseits...

- Sehr hohes Innovationstempo
 - erfordert stetig neue Lösungen für neue Probleme

...andererseits

- Für Tools, Bibliotheken etc gibt es keine zentrale Instanz
- Probleme werden "dezentral" gelöst



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?

Sample File - Figma

Sicher | https://www.figma.com/file/pQLiW7fU1VQwYIJjs2epDI/Sample-File?node-id=0%...

Instructions

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

Log-In Page

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

DESIGN PROTOTYPE CODE

Bring Forward ⌘]

Bring to Front ⌘]

Send Backward ⌘[

Send to Back ⌘[

Group Selection ⌘G

Ungroup Selection ⇧⌘G

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

CONSTRANTS

LAYER

FILL

STROKE

EFFECTS

EXPORT

Click + to add an export setting

The screenshot shows the Figma application interface. On the left, there's a sidebar with a tree view of components: 'Instructions' (with 8 items), 'Log-In Page' (with 5 items: 'Status Bar', 'App Info' (containing '151 3rd St San Fran...', 'YOUR ART MUSEUM'), 'Log-In Fields', 'Log-In Button'), and 'Background Image'. The 'Background Image' item is selected and highlighted in blue. The main canvas area shows a blurred background image of two people, with a foreground login form. The form includes fields for 'Email address' and 'Password', a 'Forgot your password?' link, a red 'Log In' button, and a 'Don't have an account?' link. A context menu is open over the background image, with the 'Send Backward' option highlighted in blue. The right side of the screen has various panels for 'DESIGN', 'PROTOTYPE', and 'CODE', along with settings for 'CONSTRAINTS', 'LAYER', 'FILL', 'STROKE', 'EFFECTS', and 'EXPORT'. The URL in the browser bar is 'https://www.figma.com/file/pQLiW7fU1VQwYIJjs2epDI/Sample-File?node-id=0...'. The top right corner shows the user's name 'Nils'.

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

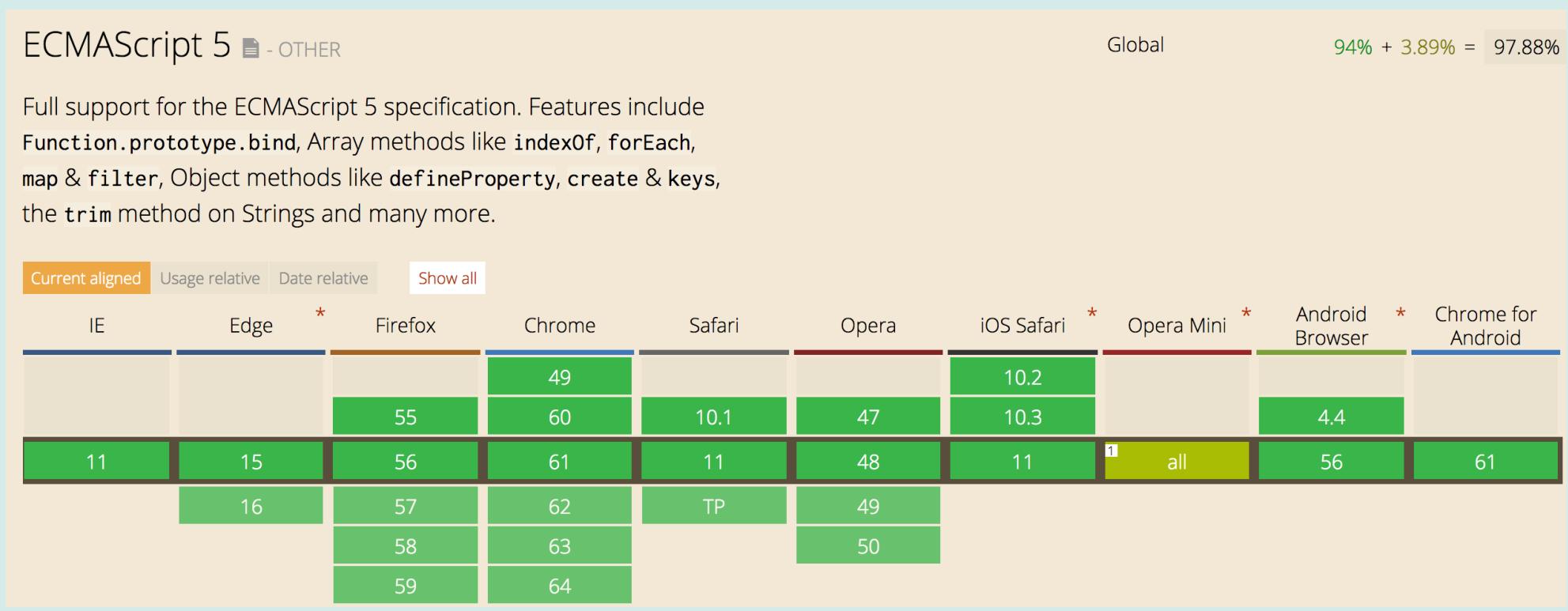
Die Sprache

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%	97%	97%	97%	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	7/7	4/7	0/7	5/7	4/7	0/7	5/7	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		
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	15/15	13/15	11/15	4/15	15/15	0/15	12/15	0/15	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	6/6	6/6	6/6	6/6	6/6	6/6	4/6	0/5	0/5		
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	9/9	7/9	9/9	8/9	3/9	9/9	7/9	6/9	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	4/4	2/4	4/4	4/4	4/4	2/4	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	0/5	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	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	24/24	21/24	19/24	23/24	24/24	21/24	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	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	2/2	1/2	2/2	0/2	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	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	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	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	46/46	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	19/19	14/19	17/19	14/19	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	19/19	14/19	17/19	14/19	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	11/12	11/12	12/12	11/12	6/12	11/12	9/12	0/12	0/2	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	10/11	10/11	11/11	10/11	5/11	10/11	8/11	0/11	0/1	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	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	15/20	0/20	15/20	0/20	0/20	0/20	14/20	14/20	0/20	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	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	9/12	12/12	8/12	4/12	10/12	2/12	12/12	2/2	0/12	2/12	0/12		
well-known symbols	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	26/26	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	17/17	17/17	17/17	17/17	8/17	14/17	3/17	0/17	6/17	0/17	7/17	0/7	0/17	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	7/10	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	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	7/11	0/11	0/11	7/11			

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, ...

Unterschiedliche Anforderungen für den Build!

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 (z.B. eine Datei)

- Nicht direkt vergleichbar mit Java9/OSGi Modulen

```
module.exports = function isObject(val) {  
    return val != null && typeof val === 'object' && Array.isArray(val) === false;  
};
```

Beispiel: isobject Modul

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)

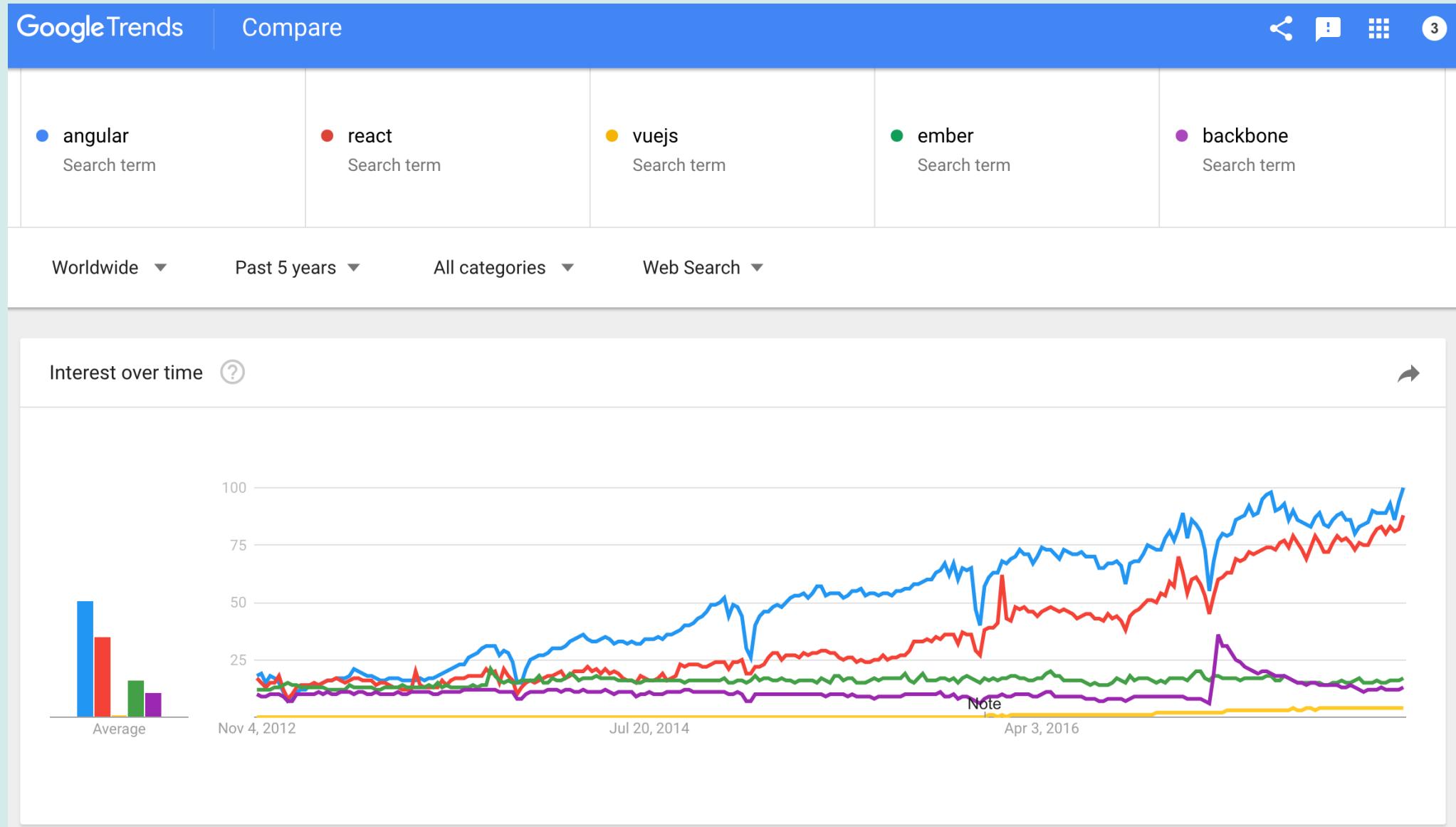
Prominente, aktuelle Vertreter:

- 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



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
- Create React App: <https://github.com/facebookincubator/create-react-app>
- Angular CLI: <https://cli.angular.io/>
- 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/>

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) möglich
- 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)!

PACKAGE MANAGER / EXTERNE ABHÄNGIGKEITEN

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** sidebar: Shows the project structure under "REACT-EXAMPLE-APP". It includes folders for "app", "components" (containing "CoreComponents.tsx", "PasswordForm.jsx", "PasswordView.tsx"), "model" (containing "Restrictions.ts"), and "node_modules" (containing ".bin", "@types", "react", "react-dom", "react-hot-api", "react-hot-loader", and "read-pkg"). A file named "app.tsx" is selected in the "node_modules/react" folder.
- OPEN EDITORS**: One editor tab is open, titled "app.tsx — react-example-app".
- Content of app.tsx:**

```
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:** Displays "typescript*" with 0 errors and 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

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 file named "example.js". The code contains a function "identical" that checks if two variables are equal using the triple equals operator ("==="). ESLint has flagged three issues in this code:

- [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)

The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with the PROBLEMS tab currently selected.

```
1
2  function identical(a, b) {
3    if (a === b) {
4      return
5      true;
6    }
7  }
8
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	Filter by t
▲ JS example.js 3				
✖ [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)				

TYPE CHECKER

Fehler schon während der Entwicklung finden

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

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

At the bottom of the editor, the `PROBLEMS` tab is selected, showing the following error list:

- `[ts] 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)`

TypeScript und Flow

- Beide syntaktisch sehr ähnlich
- Typ-Angaben sind optional (nur da, wo man sie braucht/will)

TYPE CHECKER

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

- Entwickelt von Microsoft
- Mehr als nur Type Checker
 - Spracherweiterungen, z.B. Enums, private Felder und Methoden
 - Compiler für ES6 Code nach ES5
- Angular2 ist mit TypeScript gebaut
- Sehr guter IDE Support (Visual Studio Code, WebStorm, Eclipse)
- Sehr viele Typ-Definitionen für gängige JavaScript Libraries
(<http://definitelytyped.org/>)

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

- Entwickelt von Facebook
- Bessere Typ-Inferenz als in TypeScript
- Weniger invasiv, deswegen einfacher für den Einstieg
- Wird viel im React-Umfeld genutzt

Empfehlungen:

- In größeren Projekten Type Checker unabdingbar
- Wenn Angular, dann auf jeden Fall TypeScript
- Wenn viele 3rd-Party-Libs verwendet werden, prüfen ob für Flow Deklarationen existieren / benötigt werden
- (Persönliche Präferenz: TypeScript)

Testen in JavaScript

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



TESTEN: DIE KLASSIKER

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/>)

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

- Babel (evtl TypeScript)

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

- Webpack

Verwalten von Package-Abhängigkeiten und Automatisierung von Tasks

- npm oder yarn

Type Checker

- Flow oder TypeScript

FAZIT UND ZUSAMMENFASSUNG

Empfehlungen

- So einfach wie möglich anfangen
- Erst das Problem verstehen, dann Lösung dafür suchen
- Frameworks und Tools einsetzen, wenn sie wirklich benötigt werden
- Nicht jedem Trend hinterher rennen
- Tip: Google "a vs b"
 - react vs angular
 - hsv vs bayern



ONE MORE THING...



I Am Developer

@iamdeveloper

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



<https://twitter.com/iamdeveloper/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 von **npm**: 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