

WEB-ENGINEERING II

Fachbereich Informatik

Friedhelm Koch

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

Bei Fragen

Friedhelm Koch



Mobil: +49 172 8697922

Mail: dhbw@koch.team

Zoom-Vorlesung: <https://koch.team/vorlesung>

Scripts/Snippets: <https://dhbw.koch.team>

GitHub: <https://github.com/FriedhelmKoch/DHBW>

Vernetzen über: [linkedin.com/in/friedhelmkoch](https://www.linkedin.com/in/friedhelmkoch)

Bitte Feedback

Sollte sich der Fehlerteufel in diesem Script eingeschlichen haben oder Links ungültig sein, dann lassen Sie es mich bitte wissen!

Zoom - Online Session

Möglichkeiten während einer Zoom-Session:

Zoom Help-Center:

<https://support.zoom.us/hc/de>

Zusätzliche Infos?

- Online-Vorlesung, Script und Beispiele:
<https://dhw.koch.team>
- Programm-Snippets im Quellcode auf meinem GitHub-Repository:
<https://github.com/FriedhelmKoch/DHBW>
- Kontakt:
 - Zoom-Videokonferenz: <https://koch.team/vorlesung>
 - Email: dhw@koch.team
 - Phone: +49 172 8697922

EINFÜHRUNG - WAS HABEN WIR VOR?

Worum geht es bei Web-Engineering II ?

Themen dieses Kurses

- Anforderungen an Web-Entwickler und Technologien
- Agiles Projektmanagement, Methodenkompetenz (**Scrum**)
- Progressive-Web-Application (**Web-Technologie** / **Web-Manifest** / **Service Worker** ...)
- Skalierbare und strukturierte SPA's und PWA's (**React**)
- Logik und Markup zusammenführen (**JSX**)
- Asynchrone lokale noSQL-Database API (**IndexedDB**)
- Predictable-State-Container (**Redux**) // wenn genügend Zeit vorhanden?
- App-like Anwendungsframeworks (**Framework7**)
- Anhang
 - Developer Frameworks (**Lighthouse** / **Node.js** / **npm** ...)
 - Sicherung gegen Änderungen oder Löschungen (**Git**)
 - Code-Verzeichnis für Verteilung (**GitHub**)
- Literaturverzeichnis

Ziel: Erstellung eines Programmentwurfs für eine PWA

Questions

&



Answers

ANFORDERUNGEN AN EINE MODERNE SOFTWAREENTWICKLUNG

Gestern - Heute - Morgen (Profil - Umgebung - Sprachen)

Job Description

Über IT Services

Wir sind die treibende Kraft bei der Digitalisierung der Logistik und entwickeln innovative Software für die Zukunft von DP DHL. Das Projektportfolio umfasst unter Herausforderungen in der Vermarktung, Sortierung, der Zustellung und des Transports von Brief und Paket. Unsere Arbeitsweise ist durch agile Methoden in interdisziplinären Teams geprägt. Wir nutzen aktuelle Werkzeuge und Technologie und investieren in die Weiterbildung. Die Vision "smart application build" bedeutet für uns, Geschäfts- und IT Prozesse schnell zur produktiven Umsetzung zu bringen, um deren Wirksamkeit zu erproben und neue Erkenntnisse zu gewinnen. Wir setzen auf SCRUM als Vorgehensmodell und arbeiten eng mit unseren Product-Ownern zusammen. Methoden-, Technologie- und Toolentscheidungen treffen wir gemeinschaftlich in unseren Teams. Getreu dem Motto "you build it, you run it" schließt unsere Verantwortung als DevOps Team die kontinuierliche Verbesserung und die Pflege der Software ein.

Ihre Aufgaben:

- Konzeption und Implementierung von innovativen Anwendungen in der Logistik
- Mitwirkung bei Architektur und Design
- Entwicklung und Testen von Softwaresystemen mit technischen Schwerpunkten im Frontend (Angular, React)
- Mitwirkung im SCRUM-Team und Beteiligung an Anforderungsanalyse, Design, Entwicklung und Test
- Sukzessive Erweiterung des Aufgabenfelds mit Blick auf die technische Leitung und Architektur des Gesamtsystems



Quelle: <https://www.dpdhl.jobs/job/bonn/frontend-entwickler-m-w-d-angular-react/1886/3199299488>

Ihr Profil:

- Abgeschlossenes Studium, vorzugsweise mit Informatik/ Wirtschaftsinformatik
- Praxiserprobtes Know-how in der Webentwicklung (TypeScript, JavaScript, HTML, CSS, AngularJS/Angular, React, Redux) bei microservice-basierter Architektur
- Gute Kenntnisse und Erfahrung in modernen Entwicklungsmethoden und -tools, wie Webdesign, MicroServices, APIs, Buildpipelines.
- Kommunikativer Teamplayer mit Spaß an technischen Herausforderungen
- Interesse in spannenden und innovativen Projekten die weitere Digitalisierung der Deutschen Post / DHL technische Verantwortung zu übernehmen
- Gutes Deutsch und Englisch

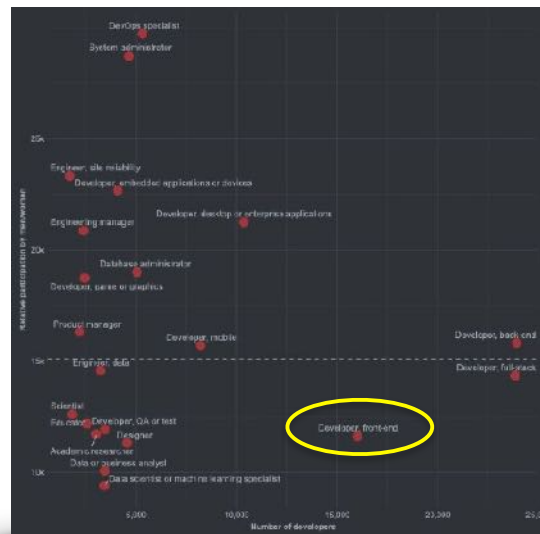
Persönliche Anforderungen:

- Kreativ und lösungsorientiert
- Teamfähigkeit, insbesondere auch in agilen (SCRUM, DevOps) oder verteilten Teams
- Sicheres, verbindliches Auftreten gegenüber Kunden, Kollegen, Lieferanten
- Stark ausgeprägte Leistungs- und Lernbereitschaft.

Profil des Softwareentwicklers



Quelle:
<https://javascript.plainenglish.io/are-you-a-good-front-end-engineer-3b2a3078a4be>



Quelle: <https://insights.stackoverflow.com/survey/2020#most-popular-technologies>

- Methodenkompetenz
- Internationale Mobilität
- Fähigkeit in Projektteams eine hohe Produktivität zu entwickeln
- Know-how Erweiterung und ständige Anpassung an neueste Technologie
- Erfahrung mit Standards und Tools/Frameworks
- Möglichst Branchenerfahrung

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

11

Entwicklungsumgebung

- Eclipse (OpenSource, der fast Alleskönner)
- Netbeans (bietet ein Plugin „nbts“)
- WebStorm (aus dem PHP Umfeld mit guter TS-Unterstützung)
- Atom (von GitHub entwickelt, aus dem Web-Umfeld mit guter Unterstützung von TS)
- Sublime (proprietär - Biz.-Version kostenpflichtig)
- Xcode (Apple, ein muss für iOS-App Entwickler)
- Visual Studio Code (von Microsoft entwickelt, OpenSource, sehr leistungsfähig für TS, eBook: [VS Tipps und Tricks](#))

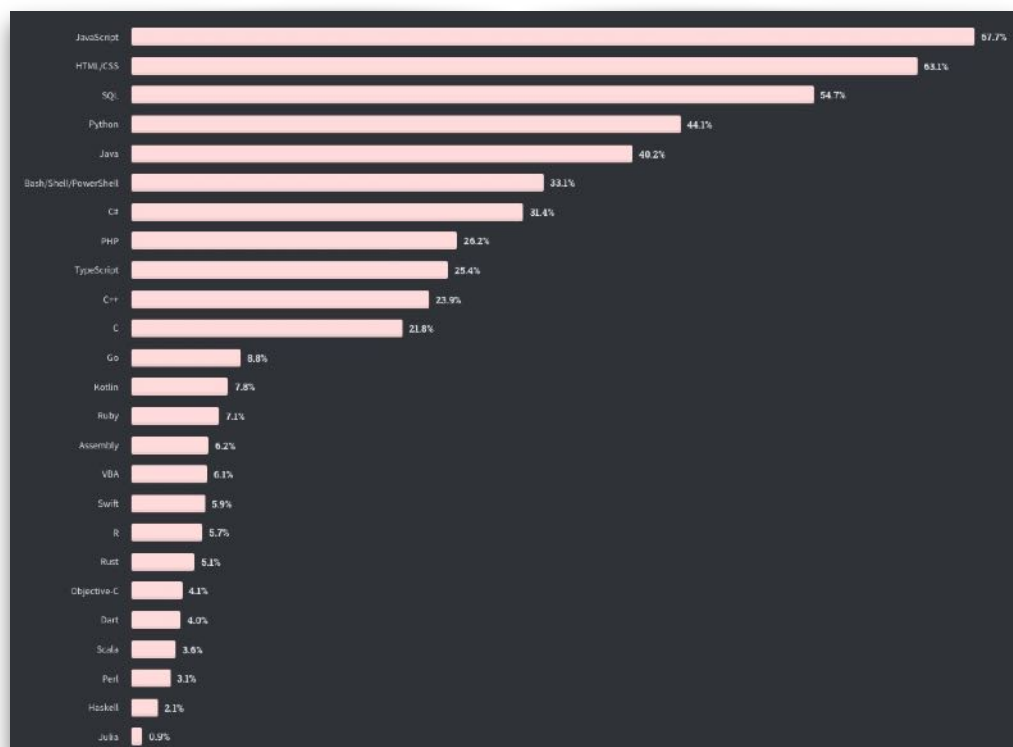
© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

12

Programmiersprachen

- **Assembler** : Low-Level CPU-hardwarenah
- **C** : Ideal um Betriebssysteme zu entwickeln
- **C#** : .Net Sprache, meist im Windows Umfeld
- **C++** : Hardwarenah, hochperformant
- **Swift** : Nachfolger von Objective-C für iOS
- **Python** : KI, Neuronale Netze z.B. Raspberry Pi
- **Java** : Unternehmensanwendungen
- **Perl** : Scriptsprache - „es gibt fast nichts, was nicht geht“
- **PHP** : geliebt und gehasst, jedoch die verbreitetste Web-Programmiersprache
- **Ruby** : universell wird meistens mit Framework „Ruby on Rails“ verwendet
- **TypeScript** : ein SuperSet von JavaScript (Java/C# like)
- **JavaScript** : "die" Sprache des Internets

Marktübersicht



<https://insights.stackoverflow.com/survey/2020>

Websites und deren Programmierung

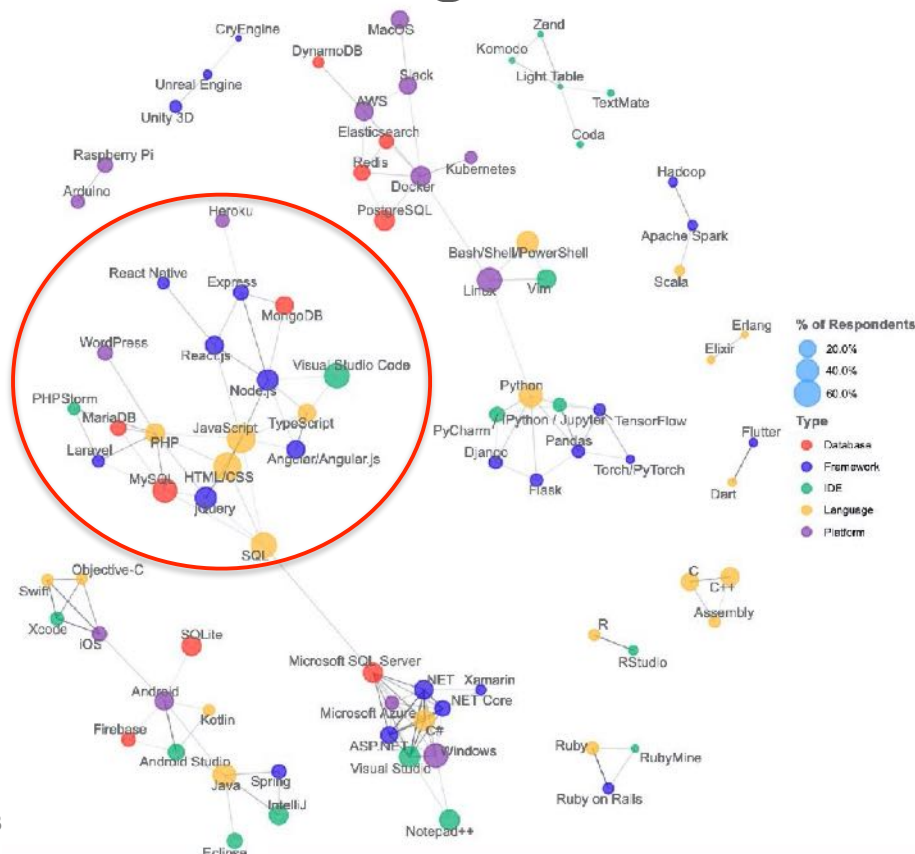
Websites	Popularity (unique visitors per month) ^[1]	Front-end (Client-side)	Back-end (Server-side)	Database	Notes
Google ^[2]	1,600,000,000	JavaScript, TypeScript	C, C++, PHP, Go ^[3] Java, Python, Node	Bigtable, ^[4] MariaDB ^[5]	The most used search engine in the world
Facebook	1,120,000,000	JavaScript, Flow ⁶	Hack, PHP (HHVM), Python, C++, Java, Erlang, D, ^[6] XHP ^[7] Haskell ^[8]	MariaDB, MySQL, ^[9] HBase, Cassandra ^[10]	The most visited social networking site
YouTube	1,100,000,000	JavaScript	C, C++, Python, PHP, Java, ^[11] Go ^[12]	Vitess, BigTable, MariaDB ^{[5][13]}	The most popular video sharing site [YouTube is also owned by Google, the first listing (can be seen above)]
Yahoo	750,000,000	JavaScript	PHP	PostgreSQL, HBase, Cassandra, MongoDB, ^[14]	
Etsy	516,000,000 ^[15]	JavaScript	PHP ^{[16][17]}	MySQL, Redis ^[18]	E-commerce website.
Amazon	500,000,000	JavaScript	Java, PHP, C++, Perl ^[19]	DynamoDB, RDS/Aurora, Redshift ^[20]	The most used e-commerce site in the world
Wikipedia	475,000,000	JavaScript	PHP	MariaDB ^[21]	A free online encyclopedia based on MediaWiki, which is programmed in PHP
Fandom	315,000,000 ^[22]	JavaScript	PHP	MySQL	Wiki hosting service.
Twitter	290,000,000	JavaScript	C++, Java, ^[23] Scala, ^[24] Ruby	MySQL ^[25]	Popular social network
Bing	285,000,000	JavaScript	C++, C#	Microsoft SQL Server, Cosmos DB	Search engine from Microsoft.
eBay	285,000,000	JavaScript	Java, ^[26] JavaScript, ^[27] Scala ^[28]	Oracle Database	Online auction house.
MSN	280,000,000	JavaScript	C#	Microsoft SQL Server	An email client, for simple use. Previously known as "messenger", not to be confused with Facebook's messaging platform.
LinkedIn	260,000,000	JavaScript	Java, JavaScript, ^[29] Scala	Venice ^{[30][31]}	World's largest professional network.
Pinterest	250,000,000	JavaScript	Python (Django), ^[32] Erlang	MySQL, Redis ^[33]	Search engine for ideas.
WordPress.com	240,000,000	JavaScript	PHP ^[34]	MariaDB ^[35]	Website manager software.

Quelle: https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

15

Welche Technologien sind connected



<https://insights.stackoverflow.com/survey/2019>

© 2022, DHB

16

Marktübersicht

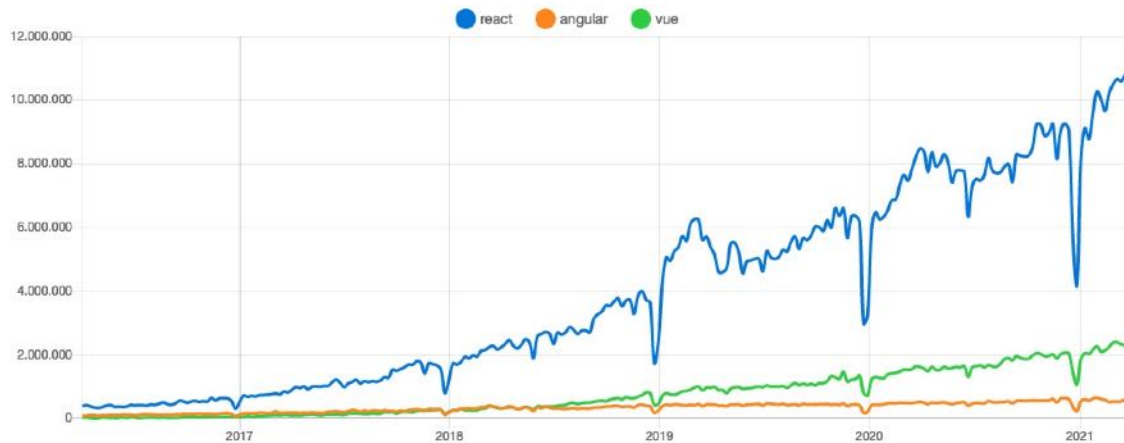
 npm trends

react vs angular vs vue

Enter an npm package...

react x angular x vue x + @angular/core + ember-source

Downloads in past 5 Years



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

<https://www.npmtrends.com/react-vs-angular-vs-vue>

17

Frage

In welcher Programmiersprache haben Sie schon einmal programmiert?

<https://pingo.coactum.de/462993> (start)

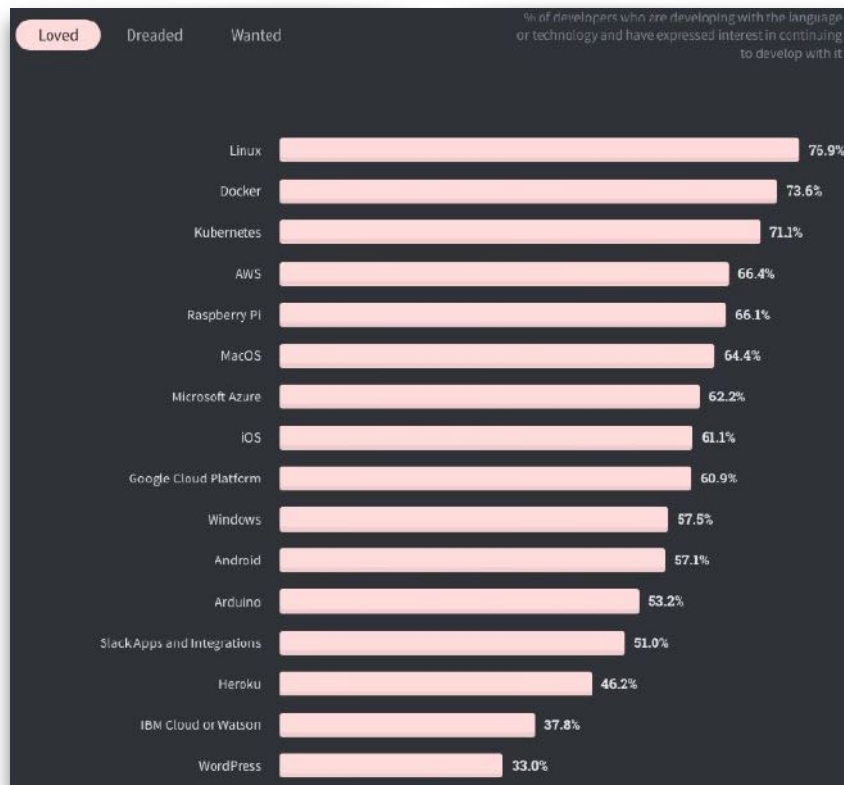


© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

18

Marktübersicht

Most Loved, Dreaded, and Wanted Platforms



<https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted>

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

19

Frage

Welches Betriebssystem nutzen Sie auf Ihren mobilen Devices für Ihr Studium?

<https://pingo.coactum.de/462993> (start)



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

20

Wie gehen Sie vor?

Sie sollen eine anspruchsvolle, interaktive und skalierbare Web-Applikation auf unterschiedlichen Devices für einen Kunden programmieren.



Frage:

Welche Web-Technologien werden Sie verwenden und warum?

Mit welcher Vorgehensweise werden Sie ihn überzeugen ?

WEB-TECHNOLOGIEN

Kriterien für die Auswahl

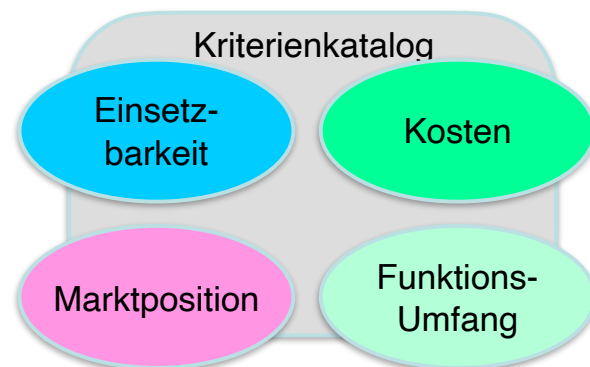
Web-Technologien

Für die Entwicklung leistungsfähiger Web-Applications ist die Entscheidung der Web-Technologie essenziell.

Was raten Sie Ihrem Kunden?

Ein Kriterienkatalog soll dabei helfen, die technologischen Entscheidungen zu treffen:

- Einsetzbarkeit
- Kosten
- Marktposition
- Funktionsumfang



Web-Technologien

Einsetzbarkeit

▪ Kriterium 1: Notwendigkeit von Frameworks

- Ein Framework ist ein Werkzeug zur Verallgemeinerung von Strukturen, um den Quellcode einfacher schreiben zu können, zu entwickeln und auch beliebig wiederzuverwenden. Dieser Code besteht meist aus Blöcken, man könnte diese auch Bausteine nennen, die nach Notwendigkeit verwendet werden.

▪ Kriterium 2: Softwarekompatibilität

- Beschreibt die Eigenschaft einer Web-Technologie, die auf mehreren unterschiedlichen Software-Plattformen (z. B. Windows, Android, iOS, ...) eingesetzt werden kann.

▪ Kriterium 3: Browserkompatibilität

- Unter Browserkompatibilität versteht man die Darstellung und Verarbeitung von Web-Applications in unterschiedlichen Browsern. Obwohl das W3C viele Techniken des WWW standardisiert hat, werden nach wie vor viele Standards von Browser-Herstellern oft nicht, oder nur teilweise eingehalten.

▪ Kriterium 4: Hardwarekompatibilität

- Dieses Kriterium untersucht den Grad der Einsetzbarkeit einer Anwendung auf verschiedenen (auch mobilen) Endgeräten. Konkret geht es um die Frage, wie gut eine Web-Technologie die Entwicklung von Anwendungen für unterschiedliche Ausgabegeräte unterstützt.

■ Kriterium 5: Dokumentation

- Für den effizienten und effektiven Einsatz einer Web-Technologie benötigen Entwickler Informationen, die eine gute Hilfestellung, gerade bei Problemen, bieten. Eine qualitativ hochwertige Dokumentation wirkt sich daher positiv auf den Entwicklungsaufwand und damit auf die Entwicklungskosten aus. Dabei sind eine sinnvolle Strukturierung der Informationen und die Qualität und Aktualität der Informationen von besonderer Bedeutung.

■ Kriterium 6: Abwärtskompatibilität

- Es ist wichtig, dass neue Versionen einer Web-Technologie so kompatibel wie möglich zu den jeweiligen Vorgängerversionen sind. Andernfalls führt dies zu hohen Update- und Schulungskosten, was sich wiederum negativ auf die Entwicklungskosten auswirkt.

■ Kriterium 7: Customizing

- Aufgabe eines Entwicklers ist es, das gleiche Aussehen der entwickelten Anwendung auf verschiedenen Browsern zu garantieren und alle Informationen und Interaktionselemente, auch in älteren Browserversionen, für Benutzer verfügbar zu machen. Es geht um die Frage, wie aufwendig und damit kostenintensiv entsprechende Maßnahmen und Aktivitäten zur Sicherstellung von Browserkompatibilität sind.

■ Kriterium 8: Integrationsaufwand

- Frameworks müssen in Webseiten oder Webanwendungen integriert werden, um bestimmte Formate in Browsern auszuführen. Dies macht zwar Browseranpassungen überflüssig, benötigt aber je nach Web-Technologie auch Zeit und ggf. Kosten.

■ Kriterium 9: Programmiersprache

- Die von einer Web-Technologie unterstützte Programmiersprache ist ein sehr wichtiger Faktor. Eine komplexe Programmiersprache (wie z.B. Java) erschwert den Entwicklungsprozess, kann jedoch gleichzeitig einen größeren Funktionsumfang bieten.

■ Kriterium 10: Verbreitung

- Die Verbreitung eines Frameworks gibt – zumindest im Falle von Technologien wie „FrameWork7“ oder „Angular Material“ – Aufschluss über die Akzeptanz der Web-Technologie bei Internetnutzern. Die Verbreitung eines Frameworks wird nachhaltig und stark die Marktanteile einer Web-Technologie beeinflussen.

■ Kriterium 11: Technologieunterstützung

- Browser-Hersteller können den Erfolg einer Web-Technologie sehr stark beeinflussen. Denn nur durch die Implementierung einer Web-Technologie in einem Browser kann diese tatsächlich verbreitet werden. Bsp.: Service Worker oder Web-Manifest.

■ Kriterium 12: Marktentwicklung

- Relevant ist die Frage danach, ob eine Web-Technologie in den letzten Jahren an Bedeutung gewonnen hat, d.h. Marktanteile gewonnen hat. Dieses Kriterium untersucht, wie sich die Marktanteile einer Web-Technologie entwickelt haben (vgl. XMLHttpRequest vs. Service Worker, XML vs. JSON).

■ Kriterium 13: Support

- Um den sich wandelnden Anforderungen der Internetnutzer gerecht zu werden, muss eine Web-Technologie sich selbstständig weiterentwickeln. Relevant ist insbesondere, wie viel Zeit ein Anbieter einer Web-Technologie in deren Weiterentwicklung investiert. Dazu kann beispielsweise die Menge bzw. Regelmäßigkeit neuer Versionen gezählt werden.

■ Kriterium 14: Reaktionsfähigkeit

- Eine zunehmend wichtige Anforderung an Oberflächen von Web-Applications ist die Unterstützung reaktionsfähiger Oberflächenkonzepte (Touch, Maus, etc.).

■ Kriterium 15: Performance

- Unter Performance versteht man die Geschwindigkeit und Qualität, mit der eine Web-Application (z.B. eine PWA) ausgeführt werden kann. Die Berücksichtigung von Performanceaspekten bei der Auswahl einer Web-Technologie ist deswegen wichtig.

■ Kriterium 16: Hardwarezugriff

- Der Zugriff auf unterschiedliche Hardwarekomponenten erweitert die Möglichkeiten zur Interaktivität und kann zu einer Personalisierung von Webanwendungen führen. Durch Zugriff z.B. auf Webcam, Mikrofon oder GPS-Modul können Anwendungen viel präziser an die Bedürfnisse des Benutzers angepasst werden.

■ Kriterium 17: Multimedialität

- Eine Web-Application enthält typischerweise viele Grafiken, Fotos und/oder Bewegtbilder. Eine Web-Technologie muss in der Lage sein, mit möglichst vielen Medien umzugehen.

Questions

&



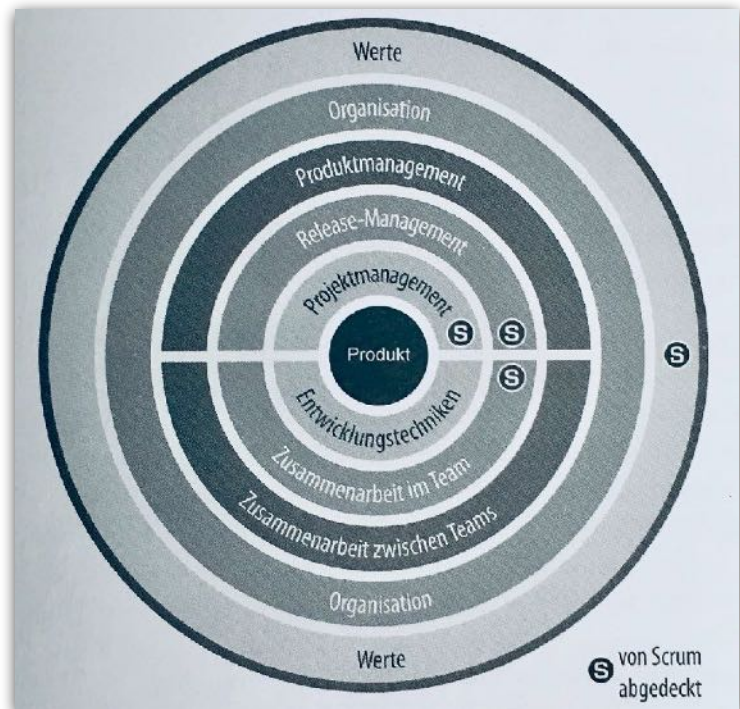
Answers

SCRUM - EINE GRUNDSÄTZLICHE EINFÜHRUNG

Methodenkompetenz - Agiles Projektmanagement

Scrum - „das Gedränge“

- Jeffrey Sutherland und Ken Schwaber entwickelten 1994 Scrum als Projektmanagement Framework. Auf der OOPSLA (Object-Oriented-Programming Languages and Applications Konferenz) wurde Scrum 1995 zum ersten Mal der Öffentlichkeit vorgestellt.
- Scrum akzeptiert, dass der Entwicklungsprozess nicht vorherzusehen ist.



Quelle: Rolf Dräther, Holger Koschek, Carsten Sabling (2013), S. 26, Scrum - kurz & gut, Heidelberg: O'Reilly Verlag

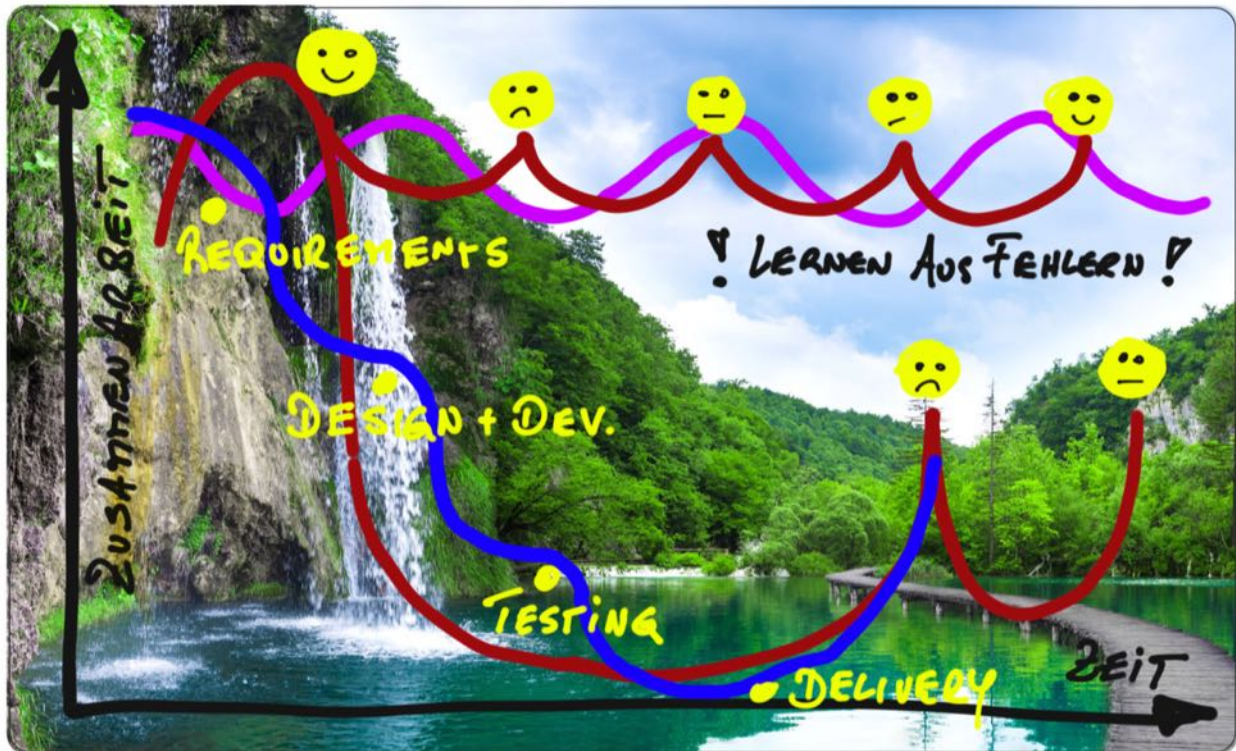
Frage

Sind Sie mit Scrum, als agile Projekt- und Softwareentwicklungs-Methode, vertraut?

<https://pingo.coactum.de/462993> ([start](#))



Wasserfall versus Scrum

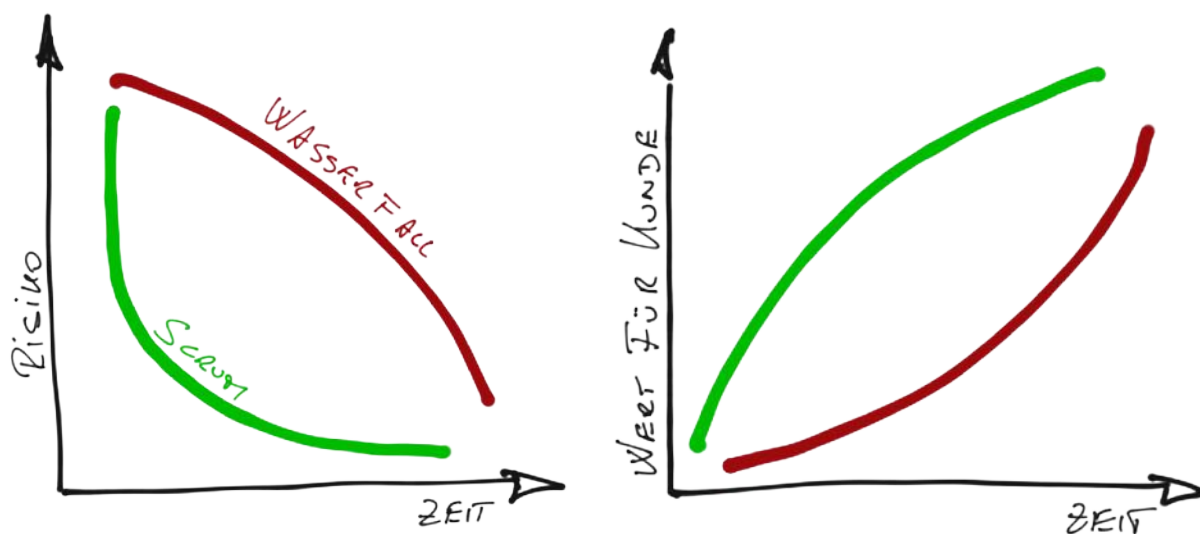


© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

33

Wasserfall versus Scrum

cont'd

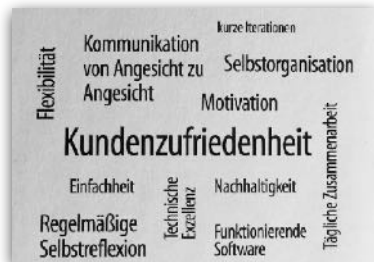


© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

34

Scrum - Agiles Manifest

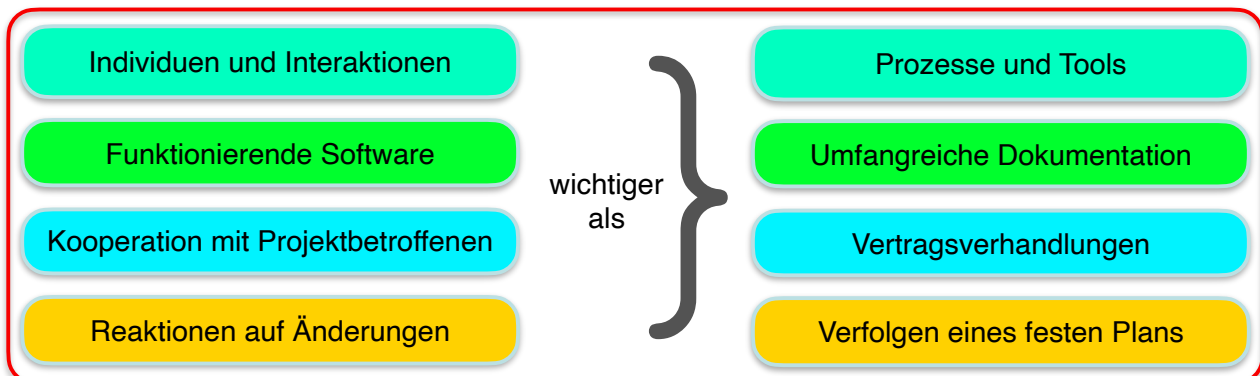
„Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt.“ (<https://agilemanifesto.org/iso/de/manifesto.html>)



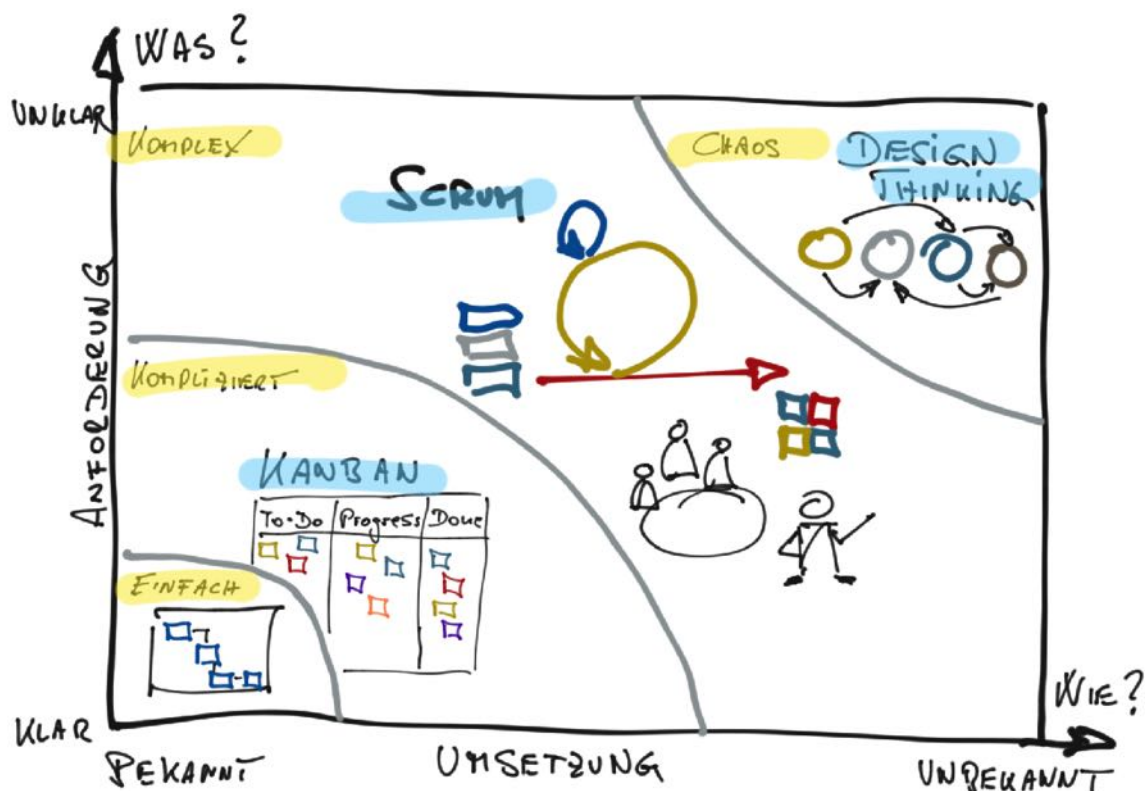
Quelle: Rolf Dräther, Holger Koschek, Carsten Sabling (2013), S. 18, Scrum - kurz & gut, Heidelberg: O'Reilly Verlag

Werte bei Scrum:

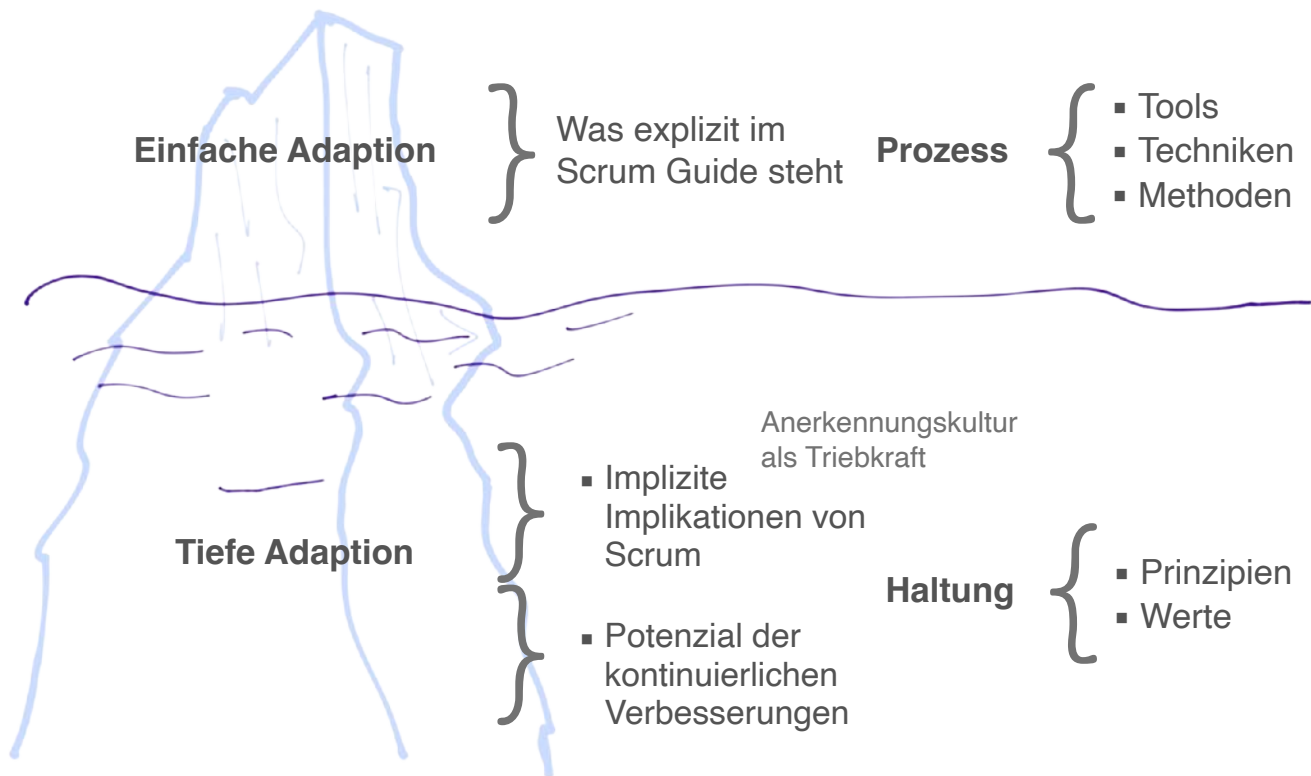
- Selbstverpflichtung
- Mut
- Fokus
- Offenheit
- Respekt



Scrum - Stacey Matrix

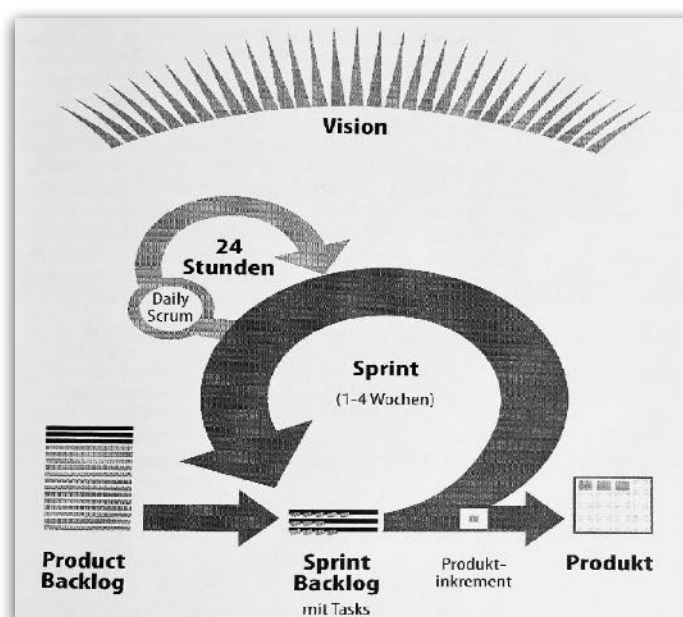


Eisberg der Agilen Transformation



Scrum - Prozess

Eine inkrementelle Vorgehensweise



Quelle: Rolf Dräther, Holger Koschek, Carsten Sabling (2013), S. 45, Scrum - kurz & gut, Heidelberg: O'Reilly Verlag

Scrum - User Story

User Stories sind Nutzergeschichten die Anforderungen formulieren. Sie transportieren nicht nur Erwartungen an zukünftige IT-Systeme, sondern auch Anforderungen an den Liefergegenstand der Vision und der Roadmap (Sprints).

- User Stories bestehen aus wenigen Sätzen. Sie sind kurz, knapp und verständlich, aus Kundensicht jedoch spezifisch und detailliert.
- Sie konzentrieren sich auf das Wesentliche und vermeiden verwirrende oder zweideutige Begriffe.
- User Stories definieren den Anforderungsraum, nicht den Lösungsraum und geben keine technischen Lösungen vor.
- „User Story“ werden im Scrum Guide nicht erwähnt. Sie sind damit nicht Kerns der Methode, sind allerdings Arbeitstechniken in agilen Prozessen.

Scrum - Rollen

Product Owner

- Entwickelt aus einer Vision eine Roadmap mit mehreren Epics (Story auf höchster Abstraktionsstufe) und User-Stories für das Produkt. Er ist für das optimale Produkt verantwortlich, stellt die fachlichen Anforderungen und priorisiert sie. Er ist verantwortlich für das Product-Backlog und organisiert mit den Stakeholder und dem Scrum-Team das Sprint-Review.

Scrum Master

- Organisiert den Prozess und beseitigt Hindernisse. Er hat dafür zu sorgen, dass alle Beteiligten ein Bewusstsein für die Ziele, Regeln und Inhalte haben. Er moderiert den Prozess und hat das Team und den Owner zu schützen (Servant Leadership).

Scrum-Team

- Entwickelt das Produkt/Lösung und erstellen das Product-Inkrement und soll selbstorganisierend strukturiert sein. Das Sprint-Backlog liegt alleinig in der Verantwortung des Scrum-Teams. Es muss dem Product-Owner und Scrum-Master verdeutlichen können, wie das Sprint-Ziel (Product-Inkrement) zu realisieren ist. Dabei muss das Sprint-Backlog vom Team täglich upgedatet werden. Für alle Ergebnisse ist immer das ganze Team verantwortlich!

Stakeholder (Kunden, Anwender, Management)

- Sie beobachten und sind Ratgeber

Scrum - Artefakte

Artefakte dienen der empirischen Prozesssteuerung. Sie sollen die „Transparenz“, „Überprüfbarkeit“ und „Anpassung“ gewährleisten! Sie sollen aktuell, detailliert und richtig sein. Der Scrum-Master ist dafür verantwortlich. Einem entsprechendem Optimum kann sich nur angenähert werden!

Product Backlog

- In diesem werden durch den Product Owner die Anforderungen an das Produkt (oder die Dienstleistung) beschrieben. Zudem werden diese Anforderungen nach ihrer Wichtigkeit für das Endergebnis priorisiert.

Sprint Backlog

- Für jeden Sprint wird aus dem gesamten Anforderungskatalog des Product-Backlogs eine Auswahl von Anforderungen getroffen. Diese werden innerhalb des entsprechenden Sprints bearbeitet.

Product Inkrement

- Jeder Sprint endet mit einem funktionsfähigen Zwischenprodukt (Produkt-Inkrement).

Scrum - Aktivitäten

Sprint Planning

- Der Product Owner stellt dem Team die wichtigste Anforderung an das Produkt (oder die Dienstleistung) vor. Der zur Umsetzung nötige Aufwand muss geschätzt und die „Definition of Done“, also wann gilt die Anforderung als erfolgreich umgesetzt gilt. Danach entscheidet „nur“ das Team, ob weitere Product Backlog Item bearbeitet werden sollen. Falls ja, wird der obere Prozesswiederholt.

Daily Scrum

- Die Teammitglieder (und meist auch der Scrum Master und ggf. Product Owner) treffen sich jeden Morgen zu einem 15-minütigem „Gedränge“. Fragen bzw. Probleme, die nicht geklärt werden können, werden an den Scrum Master gegeben.

Sprint Review

- Jeder Sprint wird mit einem Sprint Review durch das Entwicklungsteam abgeschlossen. Dort wird das Zwischenprodukt überprüft und anschließend ggf. der Product Backlog angepasst. Parallel wird Feedback vom Product Owner und den Stakeholdern eingeholt und die nächsten Schritte besprochen.

Sprint Retrospective
















- Die Arbeitsweise des Entwicklungsteams wird evaluiert um für die nächsten Sprints eine Verbesserung zu erzielen.

Product Backlog Refinement

- Im Rahmen dieser Aktivität organisiert und aktualisiert der Product Owner den Product Backlog.

Scrum - Kanban Taskboard

Beispiel

Sprint - User Stories	Tasks To-Do	In Progress	Done
			 
	 		
	 	 	

Scrum - Workflow

- Der **Scrum-Owner** beschreibt sein Ziel (**Vision**) und führt mit dem **Team** und den **Stakeholdern** den **Product Backlog**.
- Der **Scrum-Master** ist der Coach des Prozesses, schaut das die Regeln eingehalten werden und beseitigt Störungen (**Impediment Backlog**). Diese werden ggf. auf einem **Taskboard** (Kanban-Tafel: „To-Do, In Progress, Done“) visualisiert.
- Auf Basis eines **Product Backlogs Refinements** legt der Owner und das Team einzelne **Sprints** (Arbeitsabschnitte) fest und definieren dessen Ziele (**Definition of Done [DoD]**, ggf. über „**Planung-Poker**“). Daraufhin werden die **Product Backlogs** für die Sprints mit ihren beschreibenden **User Stories** und die Anforderungen festgelegt.
- Der detaillierte Sprint wird in den **Sprint-Backlogs** mit den entsprechenden **Tasks** auf dem **Taskboard** festgehalten.
- Kurze tägliche „**Daily Scrums**“ dienen dem Informationsaustausch und Fragen/Problem werden vom Master festgehalten, der für eine Lösungsfindung verantwortlich ist.
- Das **Product-Inkrement** ist die Summe aller **Product-Backlogs** in dem der nutzbare Zustand der Sprints mit seinen DoD's aufgeführt ist. Ein **Burn-Down Diagramm** visualisiert den Zustand der Aufgaben.
- Das Ende jedes Sprints (fest definierter Zeitpunkt) wird mittels eines **Sprint-Review** durch den Owner und Stakeholder das Ergebnis beurteilt.
- Über eine **Sprint-Retrospektive**, nach dem letzten Sprint, erarbeitet das Team mit dem Master Verbesserungsmaßnahmen.

ScrumButs

... sind Gründe, warum Teams nicht den vollen Nutzen aus Scrum ziehen können.

Wir machen Scrum aber...	Antipatterns
Rollen und Events werden nicht eingehalten.	Wir sind so agil, dass wir planlos sind.
Es werden die alten Begrifflichkeit verwendet.	Der Product Owner steht meist nicht zur Verfügung.
Wir passen es so an, wie wir es wollen.	Wir nennen den Projektleiter einfach Scrum Master (kein Servan Leadership).
Ein Daily in der Woche reicht aus.	Wir testen spät und viel manuell.
Wir machen keine Retrospektiven.	Das Daily wird als Status-Meeting abgehalten und es wird dabei lange diskutiert.
Unser Sprint dauert 3 Monate.	Falsche Erwartungshaltung: Scrum ist keine Methode, die man einfach einführt. Es bedarf einen tiefgreifenden Kulturwandel und das Einlassen auf immer währenden Wandel.

Planning Poker - Aufwandsschätzung

- Es gilt für User-Stories und Epics den Aufwand des Product-Backlogs zu schätzen.
- Jeder Mitspieler gibt eine Schätzung für die Komplexität mittels der Karten ab, oder legt eine Sonderkarte ab. Dies geschieht verdeckt.
- Haben alle Mitspieler entschieden, werden nach Aufforderung alle Karten gleichzeitig aufgedeckt.
- Liegen die Werte der Spieler zu weit auseinander, so müssen die Spieler mit der höchsten und der niedrigsten Karte eine Erklärung abgeben.
- Dies wird so lange wiederholt, bis sich alle Werte ohne großen Ausreißer grundsätzlich ähneln.
- Das Resultat zeigt die Verhältnismäßigkeit der Aufgaben untereinander.

Siehe auch Blogeintrag: <https://blog.mayflower.de/8443-planning-poker.html>



0	Aufgabe wurde bereits erledigt
1/2	geringer Aufwand
1 - 5	ist eine sehr kleine Aufgabe
8	mittlere Aufgaben
13	ist gut für die maximale Komplexität für eine Aufgabe
20 & 40	sind zu umfangreich für einen Sprint und müssen gegebenenfalls noch kleinschrittiger formuliert werden
100	steht für eine sehr umfangreiche und komplexe Aufgabe
?	signalisiert, dass keine Einschätzung möglich ist und mehr Informationen nötig sind
Tasse	bedeutet eine Pause

Fibonacci

- Leonardo Fibonacci 1170-1250
- Aus seinem Buch: Liber Abaci

Ein Kaninchenpaar lebt abgegrenzt von der Umwelt. Wenn ein Kaninchenpaar jeden Monat ein weiteres Paar "züchten" kann, wieviele Paare können in einem Jahr entstehen?
- Es entsteht folgende Zahlenfolge:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

und insgesamt 376 Kaninchenpaare.
- Die Folge entsteht, indem man die beiden vorhergehenden Zahlen addiert (rekursiv).
- Die n-te Zahl der Folge ergibt sich aus der Rechenvorschrift:

$$a_{n+1} = a_n + a_{n-1}$$



Scrum - Bierdeckel



Übung - Scrum

- Verschaffen Sie sich einen generellen Überblick und schauen Sie sich das offizielle Scrum-Guide (deutsch) an:
<https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf>
- Teilen Sie sich in 4 möglichst gleich große Gruppen auf (Scrum-Teams, *bei Bedarf eigene online Sessions*).
- Formulieren Sie eine User-Story für die Entwicklung einer Web-App die folgende Bedingungen erfüllen soll:
„Mobile Web-App um Events auf einem Marktplatz zu vermarkten“
- Zerlegen Sie die User-Story in kleinere Stories die die Grundlage für einzelne Sprints sein können.
- Diskutieren Sie auf Basis der Stories die Sprints in ihren Scrum-Teams (*wenn online Teams*) und schätzen sie den Aufwand und die Realisierung der einzelnen Sprints (für die Übung nicht mehr als 3 Stories). Nutzen Sie dazu die Planning-Poker Karten (ggf. Smartphone-App „Scrum Time“ von Endava aus den Apple/Google App/Play-Store):
<https://itunes.apple.com/de/app/scrums-time-planning-poker/id844162336?mt=8>
- Teilen Sie dem Product-Owner die Ergebnisse mit.

Übung - Scrum - Lösung

- **Anforderung des Kunden:**
 „Ich benötige eine App um meine Veranstaltungen in einem Portal zu vermarkten“
- **EPIC (höchste Abstraktionsstufe):**
 „Wir entwickeln eine mobile Web-App um Events auf einem Marktplatz zu vermarkten“
- **User Stories:**
 1. Der Marktplatz soll einen Kalender zeigen in dem alle Events aufgeführt sind.
 - Akzeptanzkriterium: Innerhalb einer PWA soll ein Tages-, Wochen- und Monatskalender dargestellt werden
 2. Die Events können über ein Ticket-System gebucht und bezahlt werden.
 - Akzeptanzkriterium: Das Ticketsystem soll selbst programmiert werden, der Bezahlvorgang sollte möglichst über ein online Payment (evtl. PayPal) abgewickelt werden.
 3. Ist das Event gebucht, soll es mit anderen Personen geteilt werden können.
 - Akzeptanzkriterium: Das Teilen des Termins mit Datum etc. soll über den lokalen Kalender (Outlook, iCal etc.) erfolgen.
 4. Sollte sich das Datum des Events verändern, muss der User informiert werden.
 - Akzeptanzkriterium: Die Information über die Änderung des Termins soll über Email, SMS und/oder WhatsApp erfolgen.
 5. Ist das Event beendet, soll der User über seine Zufriedenheit mit dem Event, Organisation etc. befragt werden (Survey).
 - Akzeptanzkriterium: Es soll eine Schnittstelle zu einem online Survey-Tool (z.B. SurveyMonkey) programmiert werden. Dies schafft für den User eine höhere Akzeptanz bezüglich Geheimhaltung und Datensicherheit. Die Ergebnisse sollen allerdings danach anonym in unsere PWA einfließen.

Questions

&

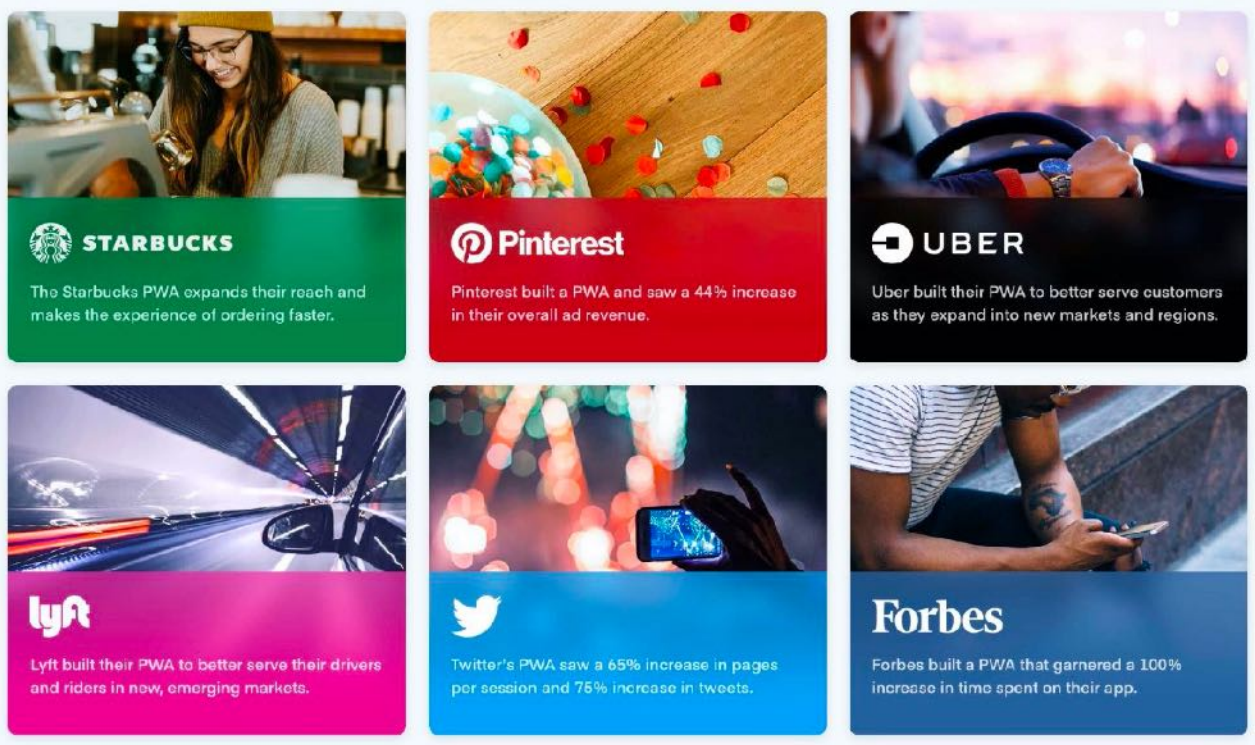


Answers

PROGRESSIVE-WEB-APP (PWA)

Merkmale & Kriterien

PWA - Beispiele

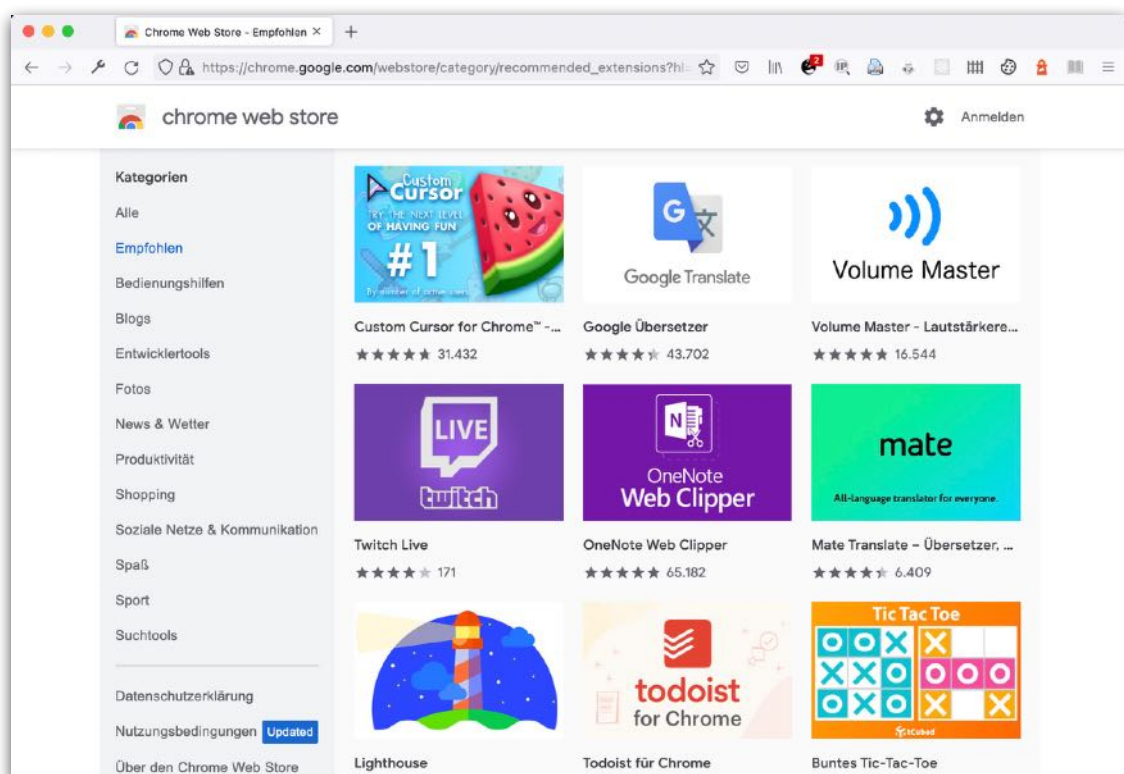


Quelle: <https://ionicframework.com/pwa>

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

53

PWA - Chrome Web Store



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

54

Progressive-Web-Application

Was ist eine PWA?

Eigenschaften nach Definition von Alex Russel (Software Engineer bei Google/Chrome Team):

1. **Progressive:** Anwender älterer Browser sollen nicht ausgeschlossen werden.
2. **Responsive:** Verfügbare Bildschirmabmessungen sollen bestmöglich genutzt werden.
3. **Connectivity Independent:** Die App läuft auch offline oder bei schwacher Verbindung.
4. **App-like:** PWA's erscheinen und verhalten sich wie native Applikationen.
5. **Fresh:** Die Applikation ist trotz Offlinekopie immer aktuell.
6. **Safe:** Die Ressource wird über HTTPS ausgeliefert.
7. **Discoverable:** Die PWA ist von einer klassischen Web-Seite unterscheidbar.
8. **Re-engageable:** Die Applikation kann den Benutzer dank Push-Benachrichtigungen zur Wiederverwendung der App bewegen.
9. **Installable:** Die PWA läßt sich installieren.
10. **Linkable:** Die Applikation ist über eine URL vereinbar, sogar bis hin zu einem bestimmten Zielzustand (Deep Link).

Quelle: Liebel Christian (2019), S. 99-100, Progressive Web Apps, Das Praxisbuch, Bonn: Rheinwerk Verlag

11. **Sensorik:** Zugriff auf native Schnittstellen, wie Kamera, GPS etc.

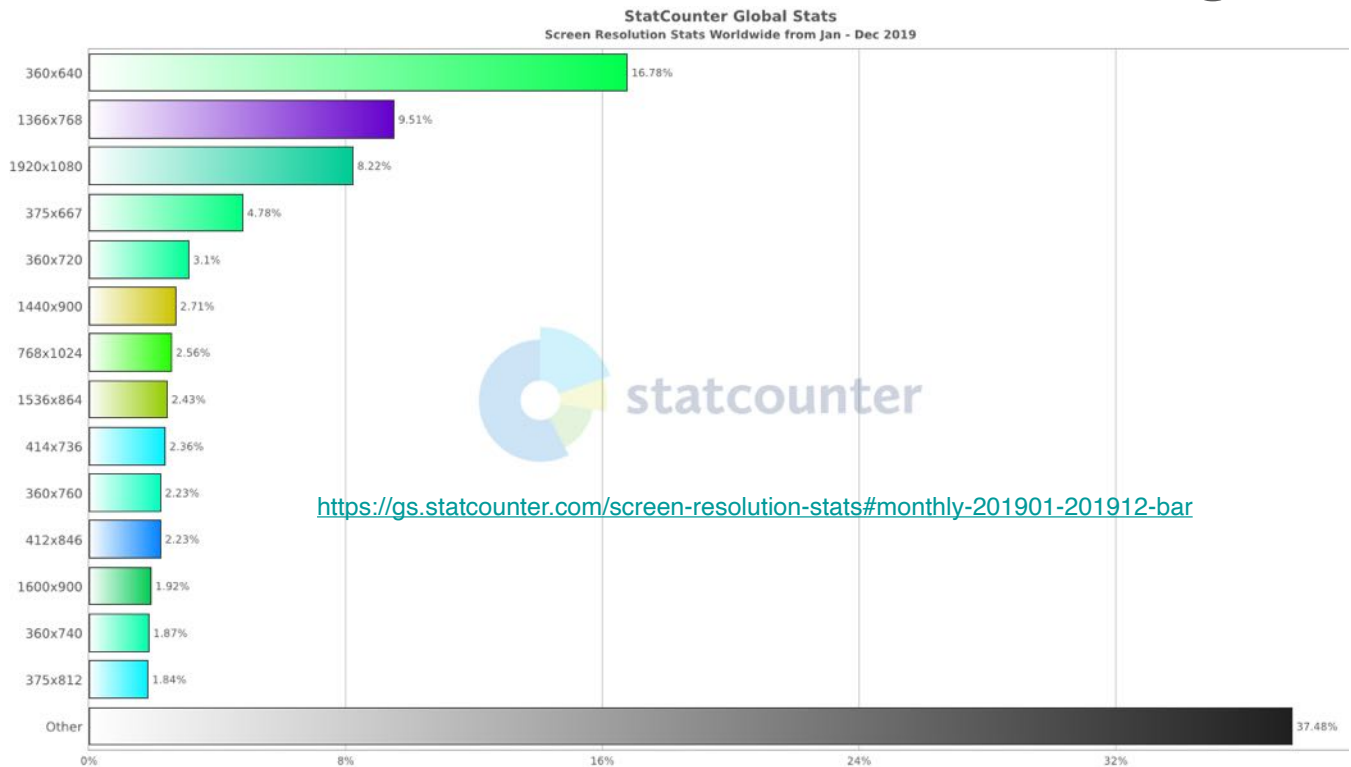
Frage

Progressive Web Applications (PWA's)?

<https://pingo.coactum.de/462993> ([start](#))



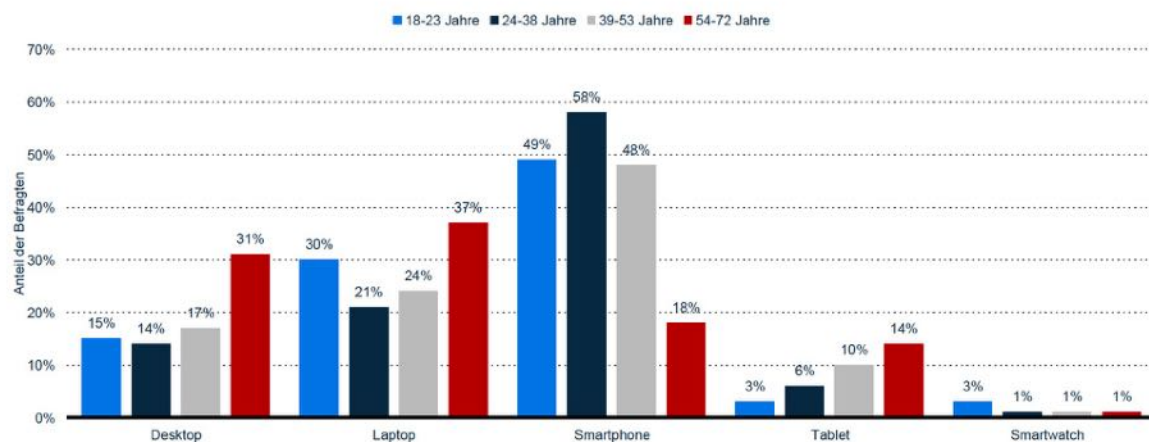
Market Share - Bildschirmauflösung



Endgeräte nach Altersgruppen

Welche Endgeräte nutzen Sie für Online-Shopping?

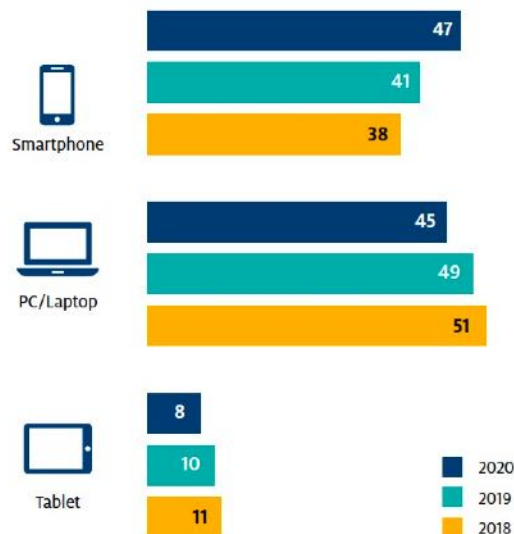
Umfrage zur Nutzung von Endgeräten für Online-Einkäufe weltweit nach Alter 2020



Hinweis(e): Weltweit; 2020; ab 18 Jahre; 4.050 Befragte
Weitere Angaben zu dieser Statistik, sowie Erläuterungen zu Fußnoten, sind auf [Seite 100](#) zu finden.
Quelle(n): Epistat; ID 1118015

Online Shopping mit dem Smartphone

Verteilung der Onlineumsätze auf Endgeräte
in Prozent



Wo wird mit dem Smartphone eingekauft?
2020 in Prozent, Vergleich zu 2019 in Klammern



Quelle: Online Monitor 2021 - <https://einzelhandel.de/online-monitor>

PWA - 1. Progressive

- Eine PWA beginnt ihr Leben im Browser-Tab.
- Sie soll zur Installation auffordern -> Home-Icon.
- Und möglichst im Vollbildmodus ausgeführt werden.
- Die PWA entwickelt sich schrittweise, also progressive in Richtung einer nativen App.
- Sie soll tolerant gegenüber Fehlern sein („Graceful Degradation“: es soll eine Teilfunktionalität immer erhalten bleiben und möglichst nie komplett abstürzen!).
- Schnittstellen zu Sensoren und Ein-/Ausgabeeinheiten sollen im Vorfeld geprüft werden („Feature Detection“).

Links:

- <https://developer.microsoft.com/en-us/windows/pwa>
- https://developer.mozilla.org/de/docs/Web/Progressive_web_apps/Introduction
- <https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=de> (Lighthouse PWA Analyse Tool für Google Chrome)

PWA - 2. Responsive Web-Design

- Mit einer Code-Basis soll sich die App an alle Devices von Smartphone bis Fernseher anpassen.
- Und dies nicht nur in Bezug auf die Bildschirmabmessungen, sondern auch bezüglich der Eingabemethoden (Maus, Trackpad, Finger, Sprache etc.)
- CSS3: Media Queries fragen Eigenschaften des Monitors ab (Barrierefreiheit).
- Konzipieren über Ansatz: „Mobile First“ - vom kleinsten bis zum größten Bildschirm.

Links:

- <https://docs.google.com/presentation/d/1uG20FaNBMQGNjMZGN1SeIJ05HVBm5u6fRSCj5ldFtg/edit#slide=id.p4>
- <https://developers.google.com/web/fundamentals/design-and-ux/responsive/> (Google, Responsive Webdesign - Grundlagen, deutsch)

PWA - 3. Connectivity Independent

- Soll auch bei fehlender oder schwacher Internetverbindung funktionieren.
- Der „Service Worker“: antwortet stellvertretend für den Web-Server. Es geht es u.a darum, Anwendungsquelldateien offline verfügbar zu machen.
- Caching Strategien:
 - Bei „Cache-First“ wird die zwischengespeicherte Version ausgeliefert.
 - Bei „Network-First“ wird die Anfrage versucht über das Netzwerk zu bedienen.
- Offline-Daten benötigen eine Synchronisationslogik. Es muss eine Strategie zur Konfliktauflösung implementiert sein.
- Paradigma: „Offline-First“.
- Service Worker sind aus Sicherheitsgründen auf HTTPS beschränkt.

Links:

- <https://serviceworker.rs/> (Service Worker Cookbook)
- <https://www.w3.org/TR/service-workers/>

PWA - 3. Connectivity Independent

- Asynchrone Operationen werden bei einem Service Worker von „Promises“ gesteuert („Versprechen“, also ein Wrapper der eine asynchrone Operation kapselt und deren Rückgabewert steuert).

```
self.addEventListener('fetch', event => event.respondWith(
  caches.open('pwa')
    .then(cache => cache.match(event.request))
    .then(response => response || fetch(event.request))
));
```

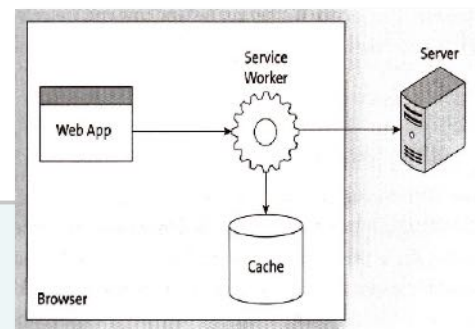
- Funktionsweise:

Die Fetch API ist die modernere Schnittstelle zur „alten“ AJAX-Schnittstelle „XMLHttpRequest“. Sie integriert durchgängig Promises (<https://fetch.spec.whatwg.org>). Auf das Ereignis „event“ wird die Methode „respondWith“ aufgerufen, wodurch der Service Worker übernimmt und ein Promise entgegen nimmt. Dann wird der Cache geöffnet und der Eigenschaft „request“ zur Verfügung gestellt. Die Methode „match“ überprüft die Inhalte gegen den Cache. Entweder wird die Anfrage über den Cache-Inhalt beantwortet oder „fetch“ startet eine Netzwerkabfrage.

PWA - 3. Connectivity Independent

- Die Cache API bietet die Möglichkeit ihr mehrere Ressourcen über die Methode „addAll()“ hinzuzufügen.

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('pwa-cache',)
      .then(cache => cache.addAll([
        './',
        './index.html',
        './favicon.ico', ... ]))
      .then(() => self.skipWaiting())
  );
});
```



- Funktionsweise:

Die Installation des Service Workers wird solange verzögert (waitUntil), bis die Inhalte des Cache zugeführt wurden (addAll).

Das Service Worker Script wird nicht im Cache gespeichert, sondern vom Browser selbst offline gehalten.

PWA - 4. App-like

- PWA soll sich möglichst „anfühlen“ wie eine native App
- Konzept der „App-Shell“: Der Anwendungsrahmen mit Teilen der Benutzeroberfläche und Bedienelemente wie Menüs, Titel- und Statuszeilen soll offline vorgehalten werden
- Dynamische Inhalte sollen „Connectivity Independent“ gehalten werden
- Die „User Experience“ ist entscheidend für den Erfolg einer PWA

Übungen hierzu werden wir in den kommenden Vorlesungen zu Framework7 haben.

Links:

- <https://www.w3.org/TR/mobile-bp/> (Mobile Web Best Practices)
- <http://mobile-frameworks-comparison-chart.com/> (Mobile Frameworks Vergleich)

PWA - 5. Fresh

- Der Vorteil einer zentralen Wartbarkeit darf gleichzeitig aber keine Inkonsistenzen zwischen unterschiedlichen Versionen (Website-Versionen) werden
- Achtung bei Datenbank- und Datensatz-Veränderungen

PWA - 6. Safe

- Sicherstellung einer geschützten Auslieferung der Quelldateien und Übertragung der Anwenderdaten
- Hypertext Transfer Protocol Secure (HTTPS), basierend auf „Transport Layer Security“ (TLS) oder älter „Secure Socken Layer“ (SSL)
- HTTP5: Geolocation-API (GPS) setzt HTTPS voraus!
- Verhindern von Attacken:
 - Cross-Site Scripting (XSS) - „Buffer Overflow“. Laden fremder Inhalte in einen vertrauenswürdigen Kontext
 - Cross-Site Request Forgery (CSRF) - Unterschieben einer kompromittierenden URL, um Daten umzulenken

Bemerkung:

- kostenlose Zertifikate sind meistens nur 90 Tage gültig (z.B. Let's Encrypt)
- HTTPS Server haben Einfluss auf Google ranking
- Beispiel einer Location Based Service Web-App: <https://viaSailing.koch.team>

PWA - 7. Discoverable

- PWA soll sich von regulären Webseiten unterscheiden
- „Web-App-Manifest“ (JSON-Format): stellt sicher, dass die PWA auch mit entsprechenden PWA Funktionen ausgestattet ist und auch entsprechend von Suchmaschinen-Crawler gefunden und kategorisiert wird
- Dieses „Anwendungsbündel“ kann dazu führen, dass PWA Stores (z.B. Microsoft Store) die PWA aufnehmen und registrieren

Links:

- <https://developers.google.com/web/fundamentals/web-app-manifest/>
- <https://developer.mozilla.org/de/docs/Web/Manifest>

PWA - 8. Re-engageable

- Einbindung von Benutzer-Push-Notifications
 - Aktualisierungen der PWA
 - Neue Inhalte
 - Breaking News
 - Nachrichten von Kontakten



- Notification API des Service Workers durch Zustimmung des Nutzers
- Push-Dienste über Protokoll: „Web-Push“:
 - Chrome: Firebase Cloud Messaging
 - Windows: Push Notification Server
 - Firefox: Mozilla Web Push
 - Safari/Mobile: steht derzeit nicht zur Verfügung
- Mögliche Drittanbieter: OneSignal, Pushpad, PushAlert, Pusher (<https://pusher.com/>)

Weiterführende Links:

- <https://support.mozilla.org/de/kb/push-benachrichtigungen-firefox>
- <https://developers.google.com/web/fundamentals/codelabs/push-notifications/>

PWA - 9. Installable

- PWA soll als Icon auf dem Home-Screen installiert werden können, um dann auch dem Merkmal „1. Progressive“ (z.B. Vollbildmodus bei Single Page Apps) zu genügen
- Davon bleibt die manuelle Installation unberührt.
Menüpunkt: „Add to Homescreen“
- Damit ist die PWA auch im App-Switcher verfügbar, sichtbar und umschaltbar
- Ggf. muss eine lokale offline Datenbankinstanz installiert werden (unter Berücksichtigung von 3. Connectivity Indenpendencies)

Links:

- <https://github.com/knadh/localStorageDB> (Key/Value based locale Database)

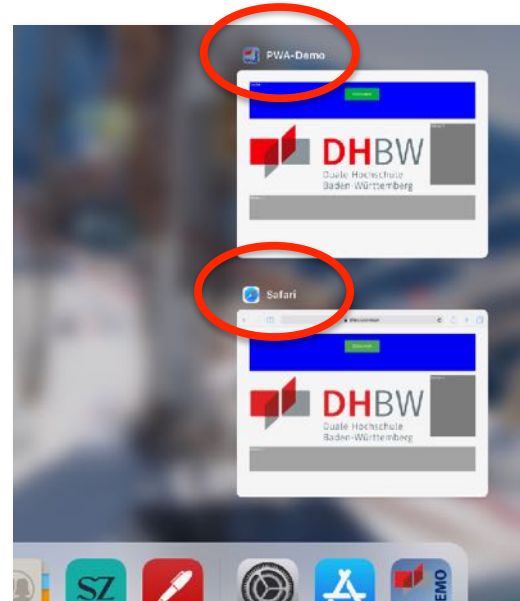
PWA - 9. Installable

- Eigenes Home-Icon



- Task-Switcher

Die PWA steht, obwohl sie eine Browser-App ist, als eigenständige Applikation mit eigenem Icon im Task-Switcher (am Beispiel eines iPads).



PWA - 10. Linkable

- Auf eine PWA kann mittels URL verwiesen werden, ggf. direkt auf einen Zielzustand
- Da eine PWA browserbasiert arbeitet ist dies direkt gegeben
- Über ein „Deep Linking“ ist ein Zielzustand einer Single Page Web App direkt ansteuerbar:
 - Hash-basiertes Routing:
Bsp.: https://my-web.test/#!/product
 - Pfad-basiertes Routing:
Bsp.: https://my-web.test/product

PWA - 11. Sensorik & Multimedia

- PWAs haben üblicherweise keinen Zugriff auf native Schnittstellen. Mit HTML 5 haben sich jedoch schon die Geolocation API etabliert oder auch der Zugriff auf die Kamera.
- Weitere Schnittstellen befinden sich in der Spezifikationsphase:
 - *Shape Detection API*: zur Erkennung von Strichcodes oder Gesichtern in Bilddateien (<https://developers.google.com/web/updates/2019/01/shape-detection>).
 - *Web Bluetooth API*: zur Kommunikation mit Bluetooth Geräten (<https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web>).
 - *WebXR Device API*: zur Umsetzung von Augmented und Virtual Reality Anwendungen (<https://www.w3.org/TR/webxr/>).

Beispiele - Links:

<https://viaSailing.koch.team> (Location Based Services, Geolocation Web-App zur Wegpunktnavigation von segelgängigen Schiffen).

<https://dhw.koch.team/demo/kamera> (ein einfacher Zugriff auf die Media Capture and Stream API, siehe auch: <https://www.w3.org/TR/mediacapture-streams/>).

<https://dhw.koch.team/demo/sprache> (ein einfacher Zugriff auf die Speech Synthesis API, um Texte vorlesen zu lassen, siehe auch: <https://developer.mozilla.org/de/docs/Web/API/SpeechSynthesis>).

Questions

&



Answers

PWA - ANALYSE

Performance, Architektur, Kompatibilität & Implementierung

Performance einer PWA

- Der größte Prozentanteil des Internet-Traffics sind Bilder.
 - Check => Format und Größe?
- Wieviele Schriften werden geladen? Sie können ein großes Bottleneck darstellen.
 - Check => Nutzen vs. Performance?
- Was wird zuerst geladen - welchen Einfluss hat das auf die Performance?
 - Check => Werden Skripte asynchron geladen?
 - Check => Wieviele Stylesheets sind im HTML?
- Die Priorisierung von Ressourcen hat einen großen Einfluss auf die PWA-Performance.
 - Der Browser interpretiert das HTML-Dokument Zeile für Zeile. Steht das <script>-Tag im <head>-Tag der Seite, wird der Browser das Javascript an Ort und Stelle ausführen.
 - Die meisten Funktionen können allerdings erst ausgeführt werden, wenn bestimmte Elemente der Webseite geladen sind (Abhängigkeit vom DOM). Der Browser muss die angesprochenen Elemente bereits geladen haben, sonst entsteht ein Javascript-Fehler.
 - Wenn das Skript relativ klein ist und/oder vom Aufbau des DOM abhängig ist, kann es sinnvoll sein, es inline zu definieren, obwohl die Inline-Definition das Parsen des HTML-Dokuments blockiert, sollte dies bei einer geringen Größe keinen wesentlichen Performance-Engpass darstellen.

Performance einer PWA

- Einfache Regel:
 - CSS-Dateien im <head>, Javascript am Ende der Webseite einbinden.
 - Wenn es gute Gründe gibt, Javascript bereits im <Head> zu laden, rangiert das Script nach den CSS-Dateien.
- Asynchrones Laden: <script async src="script.js">
 - Das Attribut "async" ist nur für extern befindliche Skripte verfügbar. Dann wird das Script heruntergeladen, während das HTML-Dokument noch geparkt wird. Nach dem Herunterladen wird das Parsen angehalten und das Skript ausgeführt (besonders nützlich, wenn es keine Abhängigkeiten vom DOM existieren!).
- Defer-Attribut: <script defer src="script.js">
 - "defer" weist den Browser an, das Skript erst dann auszuführen, wenn das Script vollständig geladen und das HTML-Dokument vollständig geladen und geparkt wurde.

Bsp.: index.html

```
<html lang="de">

<head>

  <meta charset="UTF-8">
  <link rel="stylesheet" href="style.css">
  ...

</head>

<body>

  <div id="header">...</div>

  ...

  <script async src="/script1.js"></script>
  <script defer src="/script2.js"></script>

  <script>
    alert ("Hallo World!");
    document.getElementById("header");
  </script>

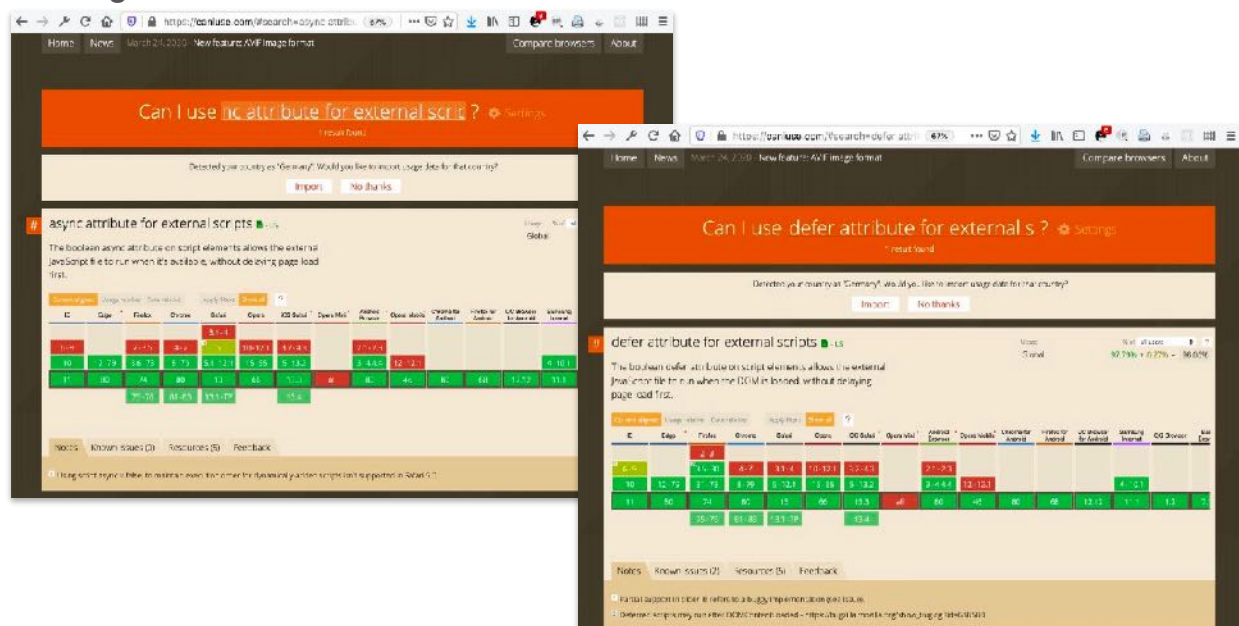
</body>

</html>
```

Übung 1 - PWA Analyse

Finden Sie heraus, welcher Browser die Script-Funktion "async" und "defer" unterstützen (https://caniuse.com/). Und wie ist das Resultat zu interpretieren?

Lösung:



Übung 1 - PWA Analyse

Installieren Sie Google Lighthouse und analysieren Sie die PWA:

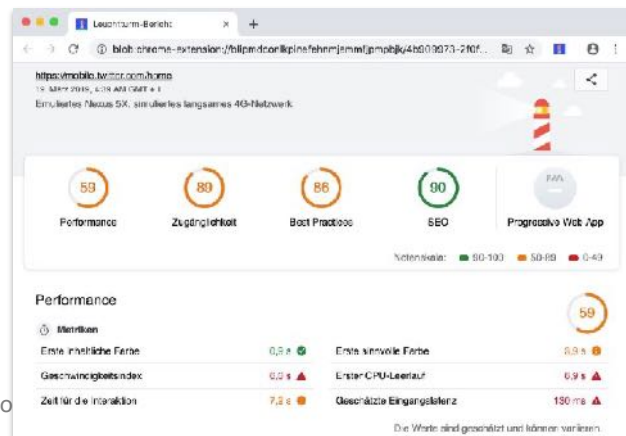
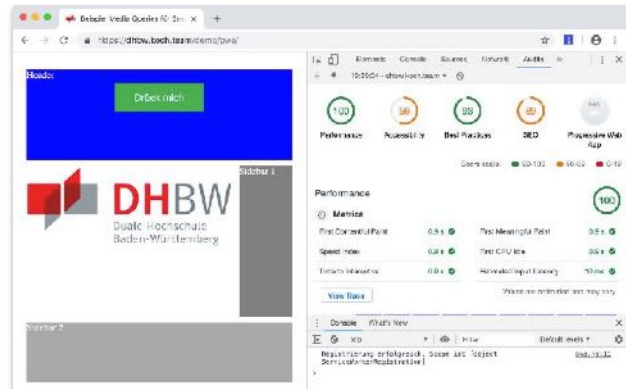
- <https://mobile.twitter.com>

Vergleichen Sie die Analyse mit einer Analyse von

- <https://dhw.koch.team/demo/pwa>

Welche Merkmale fallen Ihnen auf? Was könnte man verbessern?

© 222, DHBW - Fachbereich Informatik - Friedhelm Koch

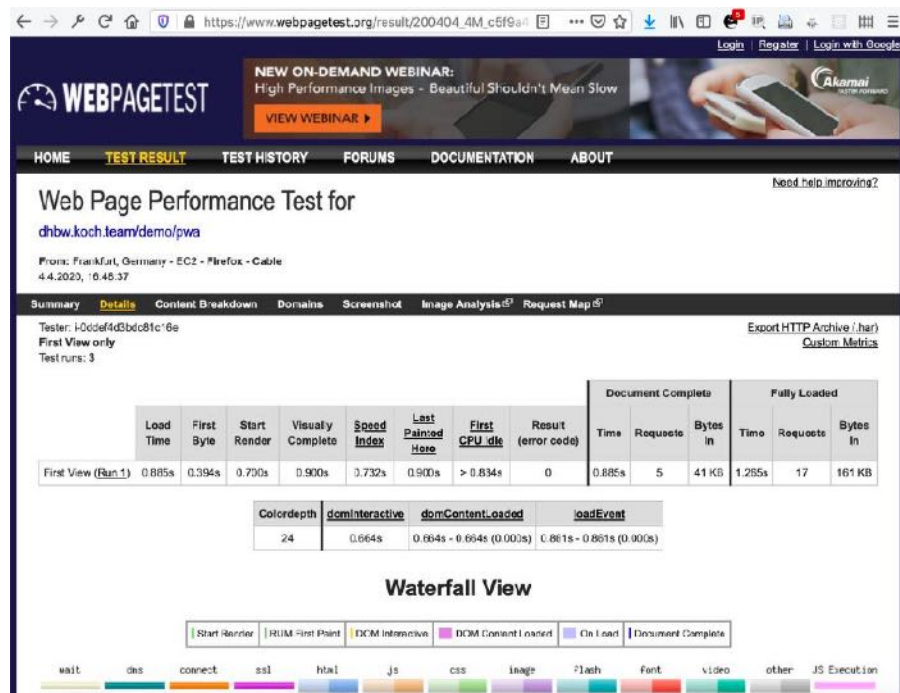


Analyse von: dhw.koch.team/demo/pwa

Analyse von: mobile.twitter.com

Übung 1 - PWA Analyse

Welche Unterschiede der Analyse sehen Sie zwischen Google "Lighthouse" und "webpagetest.org"?



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

80

Übung 2 - PWA

Den Screen auf Devicegröße zwingen:

```
<meta name = "viewport" content = "width = device-width,  
  initial-scale = 1.0,  
  maximum-scale = 1.0,  
  user-scalable = no">
```

- „viewport“ ist der Bereich des Browserfensters, der für die Darstellung der Inhalte zur Verfügung steht.
- „width = device-width“ heißt "nicht zoomen" und setzt die Breite der Ausgabe auf maximale Auflösung der Bildschirmbreite.
- „initial-scale = 1“ setzt den anfänglichen Zoom-Grad auf 100%.
- „maximum-scale = 1“ setzt den Zoom-Grad auf Maximum 100%. Würde Zoom zugelassen, dann würde ein Scale von 2 das Zoomen auf die 2-fache Größe zulassen.
- „user-scalable = no“ verhindert, dass der Benutzer die Seite vergrößern oder verkleinern kann.

Übung 2 - PWA

Responsive Style Sheet - CSS Anweisungen

Ausgabe auf dem Bildschirm:

```
@media screen {  
  /* CSS-Anweisungen für die Ausgabe auf Bildschirmen  
  ...  
  */  
}
```

Erweiterte Anweisungen über die Breite des Bildschirms:

(„only“ für ältere Browser, die dieses Merkmal nicht unterstützen)

```
@media only screen and ( max-width: 500px ) {  
  /* CSS-Anweisungen für Bildschirmbreiten bis maximal 500 Pixel  
  ...  
  */  
}
```

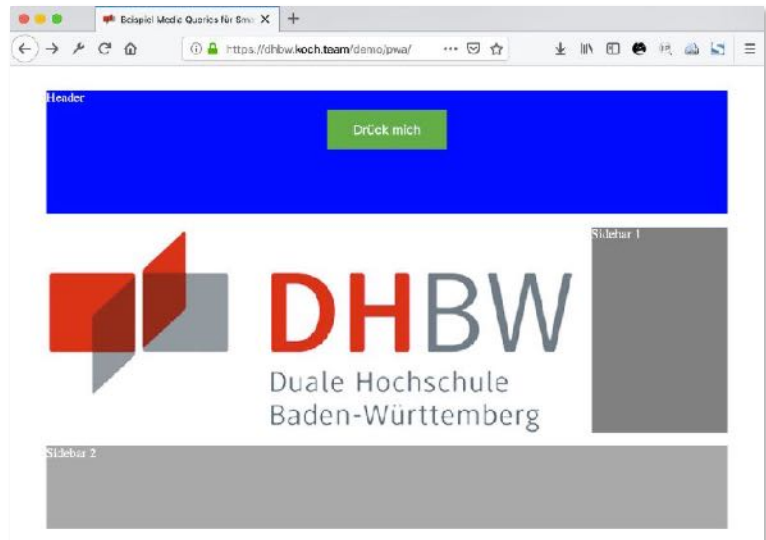
Beispiel für Smartphones und Tablets:

```
@media only screen and ( max-width: 440px ) {  
  /* mobile Geräte, wie Smartphones */  
}  
@media only screen and ( min-width: 440px ) and ( max-width: 950px ) {  
  /* Tablets und mittlere Bildschirme */  
}
```


Übung 2 - PWA

- Kopieren oder klonen Sie sich aus meinem GitHub Repository die Datei:
 „/PWA/index.html und pwa.css“

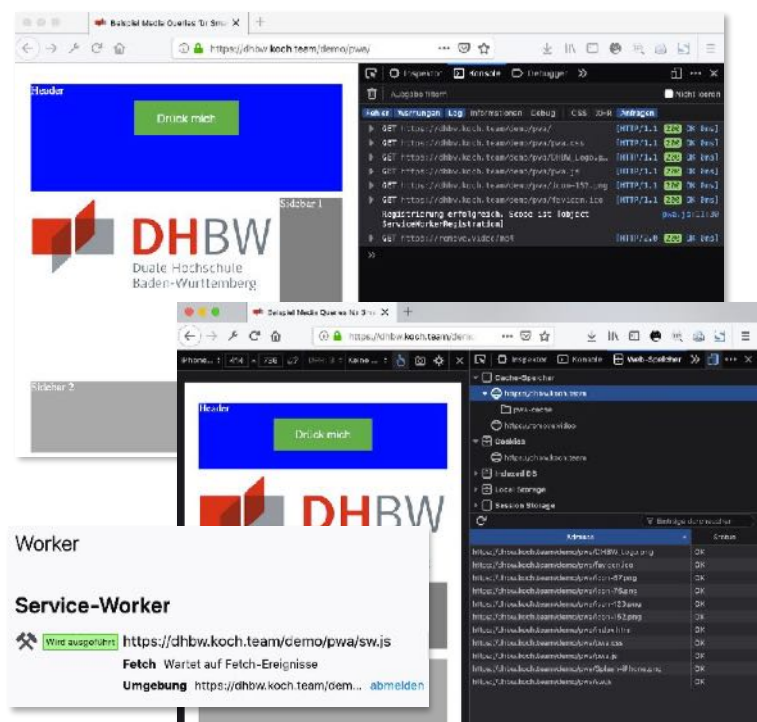
- In „pwa.css“:
 Machen Sie Änderungen am Viewport und den Media Queries für Tablet und Smartphone und verfolgen Sie die Auswirkungen.



Übung 3 - PWA

Öffnen Sie im Browser (Chrome, FireFox Developer) die Seite „dhw.koch.team/demo/pwa“. Über die Tastenkombination <F12> (Win/Linux) oder <cmd><option><i> (Mac) gelangen Sie in den Entwicklermodus. Kontrollieren Sie die Konsole, ob der Service Worker initialisiert wurde und alle Dateien wie in der Datei sw.js beschrieben im Cache aufgenommen wurden.

Lösen Sie die Internetverbindung Ihres Devices und rufen die Seite aus dem Cache auf.



Übung 4 - PWA

- Einbinden eines Web-App-Manifest in HTML-Datei:

```
<link rel="manifest" href="//webmanifest.json">
```

- Beispiel:

```
{
  "name": „Web_Engineering_II“,
  "short_name": „PWA Demo“,
  "start_url": „.“,
  "display": "standalone",
  "background_color": "#fff",
  "description": „PWA Demo“,
  "icons": [{
    "src": „./img/icon-76.png“,
    "sizes": "76x76",
    "type": „./img/png“
  }, {
    "src": „./img/icon-120.png“,
    "sizes": "120x120",
    "type": „./img/png“
  }],
  "related_applications": [{ // Alternative zu nativen App im Store
    "platform": „Web“
  }, {
    "platform": "play",
    "url": „https://play.google.com/store/apps/details?id=test.test“
  }]
}
```

Übung 4 - PWA

cont'd

- Integrieren Sie das Web-Manifest in die Datei:

„/PWA/index.html“

- Safari nutzt derzeit für die Home-Icons folgende Auflösungen:

- 120x120px - iPhone
- 152x152px - iPad und iPad mini
- 180x180px - iPhone Plus



- und läßt die Home-Icons nur folgendermaßen integrieren:

- <link rel = "apple-touch-icon" sizes = „120x120“ href = „icon_120x120.png“>
- ...

- Erweitern und spielen Sie mit der obige HTML-Datei. Lesen Sie dazu die Ausführungen von Google und Mozilla.

WEB COMPONENTS

<https://www.w3.org/TR/components-intro/>

Web Components

Mit der Erstellung von Web-Komponenten soll das Ziel verfolgt werden:

1. sie anderweitig **wieder zu verwenden** und ggf.
2. **fremde Komponenten** einbinden zu können und
3. eine **übersichtlichere Struktur** zu schaffen.

Die Web Component Erstellung basiert auf drei Elementen um das Vorhergesagte zu erreichen:

- **Custom Elements**
 - Ein Satz von JavaScript APIs, die es ermöglichen, benutzerdefinierte Elemente sowie deren Verhalten zu definieren, die dann anschließend beliebig auf Ihrer Benutzeroberfläche verwendet werden können.
- **Shadow DOM**
 - Ein Satz von JavaScript APIs, um einen Baum aus darin gekapselten "Schatten"-DOM-Elementen an ein Element anzufügen, der unabhängig vom DOM des Hauptdokuments gerendert wird, sowie um die dazugehörige Funktionalität zu steuern.
- **HTML Templates**
 - Diese <template>-Elemente gestatten es, Markup-Vorlagen zu schreiben, die nicht auf der dargestellten Seite abgebildet werden. Diese können dann als Grundlage für benutzerdefinierte Elemente mehrere Male wiederverwendet werden.

Siehe auch Spezifikation des W3C: <https://www.w3.org/TR/components-intro/>

Web Components

Ein Marktplatz für Komponenten:

[webcomponents.org](https://www.webcomponents.org)

Man kann schnell, einfach und kostensparend auf Komponenten zurückgreifen, die von einer großen Anzahl an Entwicklern benutzt werden.

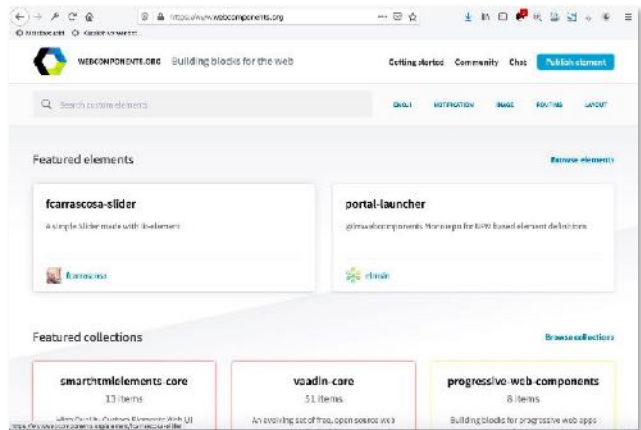
Der hauptsächliche Anwendungsfall für Web Components liegt darin,

Framework-unabhängige UI-Komponenten zu erstellen - jedoch wird in der Praxis dennoch sehr oft zu einem Framework von Google gegriffen: **Polymer!**

Auch führen sehr spezifische Anforderungen für jedes einzelne Projekt dazu, dass sehr individuelle Komponenten geschaffen werden, die nicht oder nur sehr schwer in anderen Projekten weiterverwendet werden können. Auch der Vorteil, dass bestehende Web Components aus Dritthänden weiterentwickelt werden, versiegt schnell, wenn für das Projekt relevante Änderungen an der Komponente durchgeführt werden müssen, die ein späteres Update nicht mehr möglich machen.

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

89



Web Components

Von der Theorie zur Praxis - 3 wesentliche Aspekte:

- Eine Wiederverwendbarkeit und das dahinterliegende Konzept ist verlockend!
- Ist jedoch der bisherige Standard von 2014 des W3C Konsortium genug ausgereift?
- Ist es sinnvoll ein Web Component zu schaffen, dass meistens wiederum mittels eines weiteren Framework, wie z.B. Polymer erstellt werden muss?

Es stellt sich die Frage?

Warum nicht mit einem der Standard Frameworks arbeiten? React - Angular - Vue



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

90

REACT

Eine grundsätzliche Einführung in das Konzept

Frage

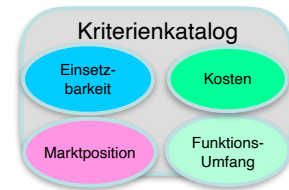
Warum React?

<https://pingo.coactum.de/462993> (start)



React - Single Page Application

- Mit welcher Technologie soll heute eine Webanwendung umgesetzt werden (siehe auch Kapitel: Web-Technologien - Kriterien für die Auswahl)?
- Angular (2010/16), React (2013) oder Vue.js (2014)?



- Die Entwicklung von Webanwendungen in Form von Multi Page Applications (MPAs) mittels HTML, CSS und JavaScript wird zunehmend durch Single Page Applications (SPA's) verdrängt.
- Netflix, Paypal oder Google Maps/Email bedienen sich der Technologie von SPA's.

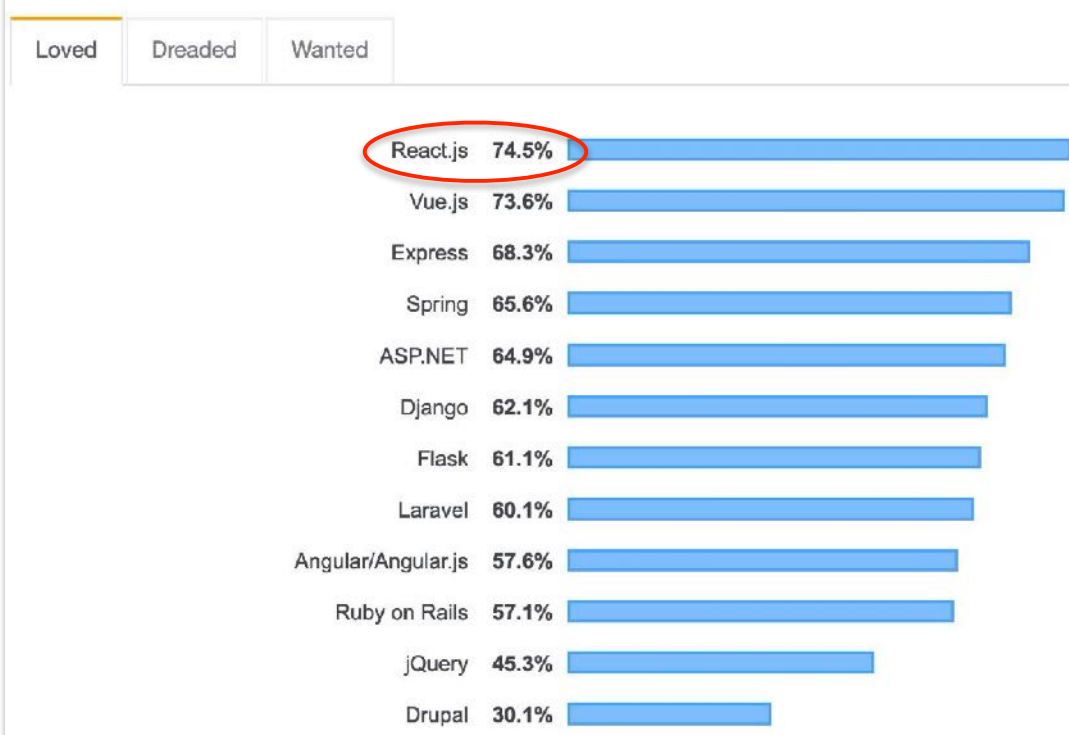
Definition SPA:

"Ist eine webbasierte Anwendung oder Website, die statt wie üblich auf mehrere Seiten verteilt, komplett auf einer Seite realisiert ist, um den Nutzern den Eindruck und das Gefühl einer Desktop-Anwendung zu vermitteln. Die Inhalte der Anwendung werden entweder zu Beginn einmalig geladen, oder dynamisch während der Laufzeit, sobald Nutzer eine Aktion ausführen möchten."

(Quelle: Siepermann, Markus (2017), in: Kompakt-Lexikon Wirtschaftsinformatik, Wiesbaden: Springer/Gabler Fachmedien)

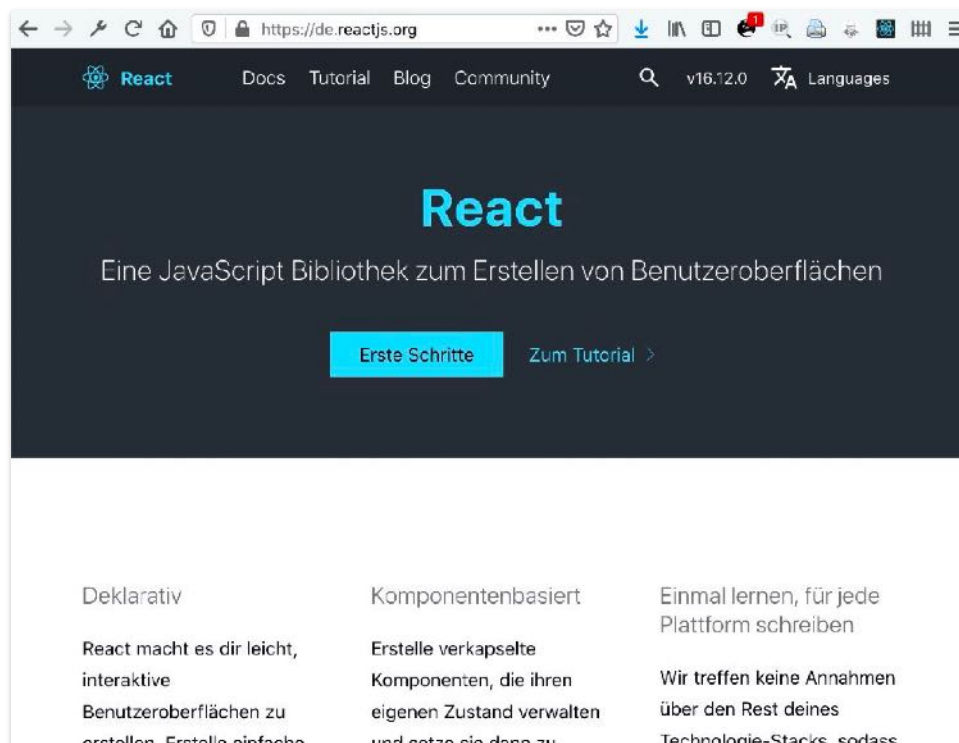
React - Marktposition

Most Loved, Dreaded, and Wanted Web Frameworks



<https://insights.stackoverflow.com/survey/2019>

React - Website



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

95

React - standalone - installieren

Node.js (Node Package Manager- npm)
installieren:

<https://nodejs.org/de/>

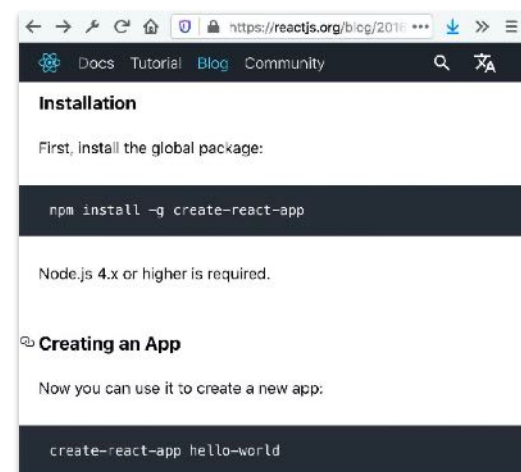
Im Terminal:

React CLI installieren:

\$ npm install -g create-react-app

Projekt im entsprechenden Verzeichnis
initialisieren:

\$ create-react-app <projekt-name>



© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

96

React - Installation

```
Freddy's Web-Build-Server $ create-react-app react-state
```

Creating a new React app in `/Users/freddy/Documents/Web-Build-Server/react-state`.

Installing packages. This might take a couple of minutes.

Installing `react`, `react-dom`, and `react-scripts` with `cra-template`...

Success! Created react-state at `/Users/freddy/Documents/Web-Build-Server/react-state`
 Inside that directory, you can run several commands:

`npm start`

Starts the development server.

`npm run build`

Bundles the app into static files for production.

`npm test`

Starts the test runner.

`npm run eject`

Removes this tool and copies build dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

```
cd react-state
```

```
npm start
```

Happy hacking!

```
Freddy's Web-Build-Server $
```

React - (Verzeichnisse nach Installation)

```
Freddy's React-State $ tree -I node_*
```

```
.
├── README.md
├── package-lock.json
├── package.json
├── node_modules
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── serviceWorker.js
    └── setupTests.js
```

- "node_modules" ist das Verzeichnis, an dem sich die von NPM oder Yarn installierten Pakete befinden werden.
- "public" ist der Ort, an dem sich die statischen Dateien befinden. Sie behalten in im Produktion-Build denselben Dateinamen bei.
- "src" ist der Ort, an dem sich die dynamischen Dateien befinden.
- "public/index.html" stellt das Page-Template dar.
- "src/index.js" ist der JavaScript entry point.

React (package.json - Versioning)

```
{
  "name": "React-Training1",
  "version": "1.0.0",
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.5.0",
    "react": "^16.13.1",
    "react-dom": "^16.13.1",
    ...
  },
  ...
}
```

Beispiel: ^9.5.0

Format: Caret MAJOR.MINOR.PATCH

- PATCH: Fehlerbehebung, um so das reibungslose Funktionieren der App sicherzustellen. Auch „Bug Fix“ oder „Patch Release“ benannt.
- MINOR: Es sind einzelne Erweiterungen und Änderungen enthalten, mit umfangreicher Verbesserung der bestehenden Funktionen und wenigen neuen Funktionen.
- MAJOR: Es beinhaltet umfangreiche neue Funktionen und ist eine Neuentwicklung auf Basis der Vorgängerversion.
- Caret: Entweder ist die angeforderte Version zu installieren ist, oder eine dazu kompatible neuere Version.

(Node.js benutzt das Konzept der Semantic Versioning - siehe auch: <https://semver.org/>)

React - Codekit



- Bevor der "React-Code" im Browser ausgeführt werden kann, muss er geändert werden.
- Babel (<https://babeljs.io/>) ist ein JavaScript-Transpiler, der die Möglichkeit bietet, JSX/ES5+ in reguläres (Vanilla-)JavaScript zu transpilieren.
- Eine App in React ermöglicht es unter ES5+ und JSX mittels Webpack (<https://webpack.js.org/>) und Babel zu schreiben. Während Webpack alle JavaScript-Quelldateien in einem Bundle als SPA bündelt, ermöglicht Babel die Nutzung von JSX und neueren ECMAScript-Funktionen (ES5+), die von vielen Browsern noch nicht unterstützt werden.
- Über das React Command-Line-Interface (CLI) "create-react-app", für die Initialisierung einer neuen App, ergibt sich für den Entwickler die Möglichkeit sich nur auf die Programmierung unter React und JSX zu konzentrieren.
- Mit Webpack, welches ebenfalls das zentrale Tool im React-Ecosystem ist, handelt es sich um einen "Modul-Bundler", der modulbasierte Entwicklung ermöglicht. Dadurch kann Code in einzelne Dateien verteilt werden, die bezüglich unterschiedlicher Abhängigkeiten über import oder require() in ihren eigenen Modul-Scope geladen und innerhalb des Moduls verfügbar gemacht werden. Am Ende wird dann eine einzelne Vanilla-JavaScript Datei erstellt (SPA).
- Neben dem Module-Bundling kann Webpack mit Babel konfiguriert werden. Dabei wird der Code transpiliert, aber auch ebenfalls Bilder, Stylesheets oder andere Assets in einen "build"-Ordner kopiert, der später auf einen Server deployed werden kann.

React - Bauplan

- Deklarativ: Der Programmierer beschreibt, was das Programm mit einer Eingabe macht, also wie mit welcher Eingabe umzugehen ist und nicht wie bei einer imperativen Sprache (Pascal, C++, C, Java, etc.), wobei in einem Algorithmus die einzelnen Schritte nacheinander und im Einzelnen, abgelaufen werden müssen.

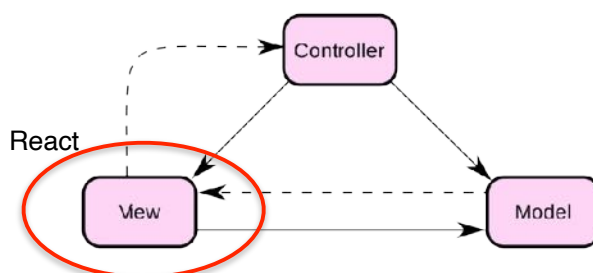
Imperativ

```
var array = [0,1,2,3,4];
for(let i = 0; i < array.length; i++) {
  array[i] = Math.pow(array[i], 2);
}
array; // -> [0,1,4,9,16]
```

Deklarativ

```
[0,1,2,3,4].map(num => Math.pow(num, 2));
```

- Komponentenbasiert: Besteht aus Komponenten, die ihren eigenen Zustand verwalten und sich zu komplexeren UI's zusammensetzen.



Model-View-Controller (MVC) Software Design Pattern:

- Model: Datenmodell
- View: Visualisierung / UI
- Controller: Logik / Server

React - Codestruktur (Funktionskomponenten)

```
// React wird in React/ES6 als Modul importiert.
import React from 'react';
import ReactDOM from 'react-dom';

// Definitionen von React Funktionen.
function Title(props) {
  return <h1{props.label}<h1>
}

...
function Header(props) {
  return <div className='header'>
    <Title label='Hello World!' />
  </div>
}

...
// die Methode ReactDOM.render() hängt die Repräsentanz eines HTML-Elements oder
// eine eigene Komponente in den nativen DOM ein.
ReactDOM.render(
  <Header />
  // im nativen DOM muss ein Element mit der Id: "element" vorhanden sein!
  // Bsp.: <div id='element'></div>
  Document.getElementById('element');
);
```

React - Codestruktur (ES6 klassenbasiert)

```
// React wird in React/ES6 als Modul importiert.
import React, { Component } from 'react';
// Eine ES6-Objekt-Klasse die von React.Component erbt.
class App extends Component { // extends gibt die Klasse von der geerbt wird an.
  // Die Constructor-Methode wird zuerst aufgerufen und definiert alle Eigenschaften
  // des Objekts.
  constructor(props) {
    super(props); // Über super() kommt man an den Konstrukten, Methoden und
                  // Properties der Oberklasse (geerbte Klasse).

    ...
    // Deklaration von Variablen & Funktionen und mögl.Übergabe von "Properties".
    // Dient dazu den lokalen State (Status) der Komponente zu initialisieren und
    // Callbacks/Event-Händler müssen an "this" dieser Komponente gebunden werden.
  }
  // Die "render"-Funktion gibt das JSX-Objekt in der "return"-Funktion zurück,
  // welches in HTML übersetzt wird.
  // Es können innerhalb einer Komponente mehrere Render-Funktionen genutzt werden.
  render() {
    let display = this.state... ? ... : ...; // ggf. Variablendeklaration etc.
    return (
      /* ... JSX-Code...
      Es darf nur ein gekapseltes Objekt zurückgegeben werden ansonsten
      muss es über DIV- oder leeren Container eingeschlossen werden.
      <div></div> oder <></> */
    );
  }
}
export default App;
```

React - Klassen

React enthält eine auf Klassen basierte Objektorientierung. Eine Klasse durch die Eigenschaft/Methode der Klasse innerhalb des Konstruktor (constructor) definiert. Dieser dient dazu Initialisierungen auszuführen, etwa Variablen zuzuweisen. Enthält die Klasse keinen Konstruktor ist sie zustandslos.

Wenn "App" an "render()" gegeben wird, ruft React den Konstruktor der Klasse auf und initialisiert den State des Objekts.

In Javascript/ECMAScript weist "super" immer auf den Konstruktor der Basisklasse den sie erbt und greift damit auf dessen Properties zu. (In dem voran gegangenen Beispiel bezieht es sich auf die Implementation von React.Component.)

Das Prinzip:

Für OO Neulinge: Eine Klasse definiert den Bauplan eines Hauses, aber noch nicht das Haus selber - sie ist die Beschreibung. Erst wenn ein Objekt dieser Klasse erzeugt wird und damit die Klasse instanziiert wird, gibt es ein Objekt/Instanz dieser Klasse.

Beispiel: Klasse SayHello

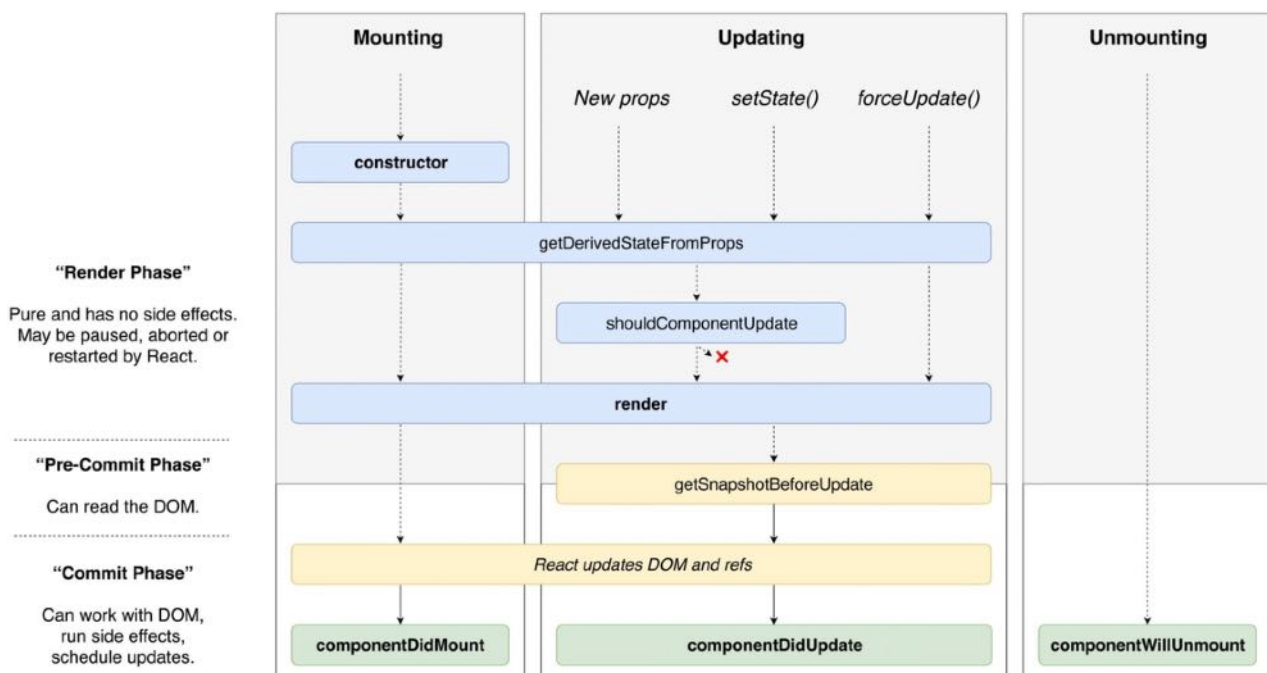
```
class SayHello {
  msg: string = "leer";
  constructor(message: string) {
    this.msg = message;
  }
  Output() {
    return "Hallo " + this.msg;
  }
}
let ToDo = new SayHello('DHBW');
console.log(ToDo.Output()); //Hallo DHBW
```

React - Lebenszyklus einer Component

- Die Methode "**componentDidMount()**" wird aufgerufen, nachdem die Komponente gerendert wurde. Hier werden die Anweisungen ausgeführt, die erforderlich sind, wenn die Komponente bereits im DOM platziert ist.
- Die nächste Phase im Lebenszyklus ist die Aktualisierung einer Komponente. Eine Komponente wird immer dann aktualisiert, wenn es eine Änderung des State oder der Properties der Komponente gibt.
- Bei Updates wird die "**getDerivedStateFromProps(props, state)**"-Methode aufgerufen. Dies ist die erste Methode, die aufgerufen wird, wenn eine Komponente aktualisiert wird.
- Mittels **shouldComponentUpdate()**-Methode (true/false) wird angegeben, ob React mit dem Rendern fortfahren soll oder nicht.
- Die **render()**-Methode wird dann aufgerufen, wenn eine Komponente aktualisiert wird, sie muss den HTML-DOM mit den neuen Änderungen rendern.
- Die Methode "**getSnapshotBeforeUpdate(prevProps, prevState)**" gibt Zugriff auf die Properties und den State vor der Aktualisierung, d.h. es ist auch nach der Aktualisierung noch möglich zu prüfen, ob und welche Werte vor der Aktualisierung vorhanden waren.
- Die Methode **componentDidUpdate()** wird aufgerufen, nachdem die Komponente im DOM aktualisiert wurde. Hierdurch können weitere Entscheidungswege getroffen werden.
- Eine weitere Phase im Lebenszyklus ist, wenn eine Komponente aus dem DOM entfernt (unmounted) werden soll. Die Methode **componentWillUnmount()** wird dann aufgerufen und entfernt die Komponente aus dem DOM.

(Siehe auch: <https://reactjs.org/docs/state-and-lifecycle.html>)

React - Lebenszyklus einer Component



(Quelle: <https://lernen.react-js.dev/die-grundlagen/state-und-lifecycle-methods>)

React - Begriffsdefinitionen

Klasse

Eine Klasse definiert den Bauplan eines Hauses, aber noch nicht das Haus selber - sie ist die Beschreibung. Erst wenn ein Objekt dieser Klasse erzeugt wird und damit die Klasse instanziiert wird, gibt es ein Objekt/Instanz dieser Klasse.

Objekt

Eine Instanz einer Klasse.

Eigenschaft

Eine Eigenschaft eines Objekts, wie zum Beispiel eine Farbe.

Methode

Ist die Fähigkeit eines Objekts, wie z.B. laufen. Sie ist eine Unterroutine oder Funktion, die mit der Klasse assoziiert ist.

Konstruktor

Eine spezielle Methode, die zur Instanziierung eines Objekts aufgerufen wird. Sie hat gewöhnlicherweise denselben Namen wie die Klasse, zu der sie gehört.

Vererbung

Eine Klasse kann die Eigenschaften und Fähigkeiten von anderen Klassen erben und somit für sich verwenden.

Datenkapselung

Die Datenkapselung ist eine Technik, um die Daten und die Methoden, die diese Daten verwenden, miteinander zu bündeln.

Abstraktion

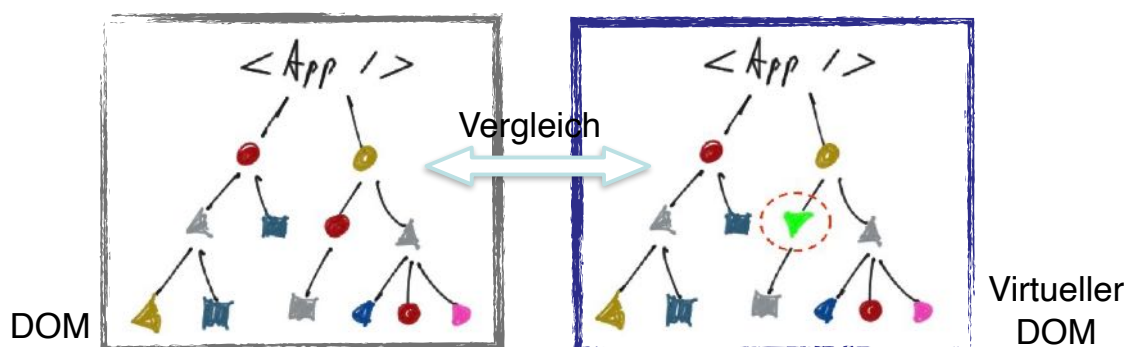
Die Verknüpfung von komplexer Vererbung, Methoden und Eigenschaften eines Objekts müssen ein Modell der Realität simulieren können (Kfz - LKW, PKW - Bus, Transporter, Limousine etc.).

Polymorphismus

Poly bedeute "*viele*" und Morphismus bedeutet "*Formen*". Verschiedene Klassen können dieselbe Methode oder dieselbe Eigenschaft definieren. (Bsp.: ES6-Snippets auf GitHub - polymorph.js)

React - Virtueller DOM

- React unterhält eine interne Darstellung des gerenderten UI ("virtuelles DOM"), welches nur die geänderten Elemente wiedergibt.
- React überprüft beide Bäume: Top-Down – Level für Level, Knoten für Knoten, Komponente für Komponente.
- Sobald zwei unterschiedliche Komponenten gefunden werden, wird ab da nicht mehr weitergesucht.
- Anstatt diesen ganzen Teilbaum Knoten für Knoten weiter zu vergleichen, wird der ganze Bereich einfach neu gerendert.
- Bei reinen <div>-Repräsentationen arbeiten, wären solche Optimierungen natürlich nicht so einfach möglich.



React - State (controlled components)

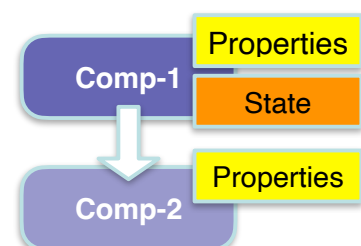
- Der "State" ist eine Instanz einer Komponenten-Klasse und kann als Objekt von "Observables" verstanden werden, die das Verhalten der Komponente steuern und ist "Single Source of Truth" (die einzige Quelle der Wahrheit).
- Unter Observable ist der Zustand eines Objekts zu verstehen, das sich während der Laufzeit (Lebensdauer der Komponente) jederzeit ändern kann.
- Wenn sich, während der Laufzeit der Komponente, der State ändert, wird diese Komponente mit ihren Child-Komponenten re-rendert.
- Inkonsistenzen durch redundante Datenhaltung werden durch den zentralen State vermieden.
- Die Child-Komponenten (siehe unten...) stellen die Daten nur noch dar.

Beispiel: State und Controlled Component

```
class InputState extends Component {
  constructor() {
    super()
    this.state = {
      input: ""
    }
    this.handleChange = this.handleChange.bind(this)
    // Erklärung hierzu bei Event-Handler...
  }
  handleChange() {
    this.setState({
      input: event.target.value
    })
  }
  render() {
    return (
      ...
      <div>
        <input
          type='text'
          value={this.state.input}
          onChange={this.handleChange} />
        </div>
      )
    )
  }
}
```

React - Properties - Children

- Der unidirektionale Datenfluss ist eines der wichtigsten Konzepte Reacts. Wie in diesem Beispiel sind die beiden Komponenten "Comp-1" und "Comp-2" hierarchisch aufgebaut. Der Datenfluss findet immer „Top-Down“ von der Komponente Comp-1 in Richtung Comp-2 statt.
- Properties sind anders als der State, unveränderlich (immutable).
- Eine React "Parent"-Komponente kann an eine andere "Child"-Komponente "Properties" weitergeben, die innerhalb des Rendering-Prozesses die eingebundenen Daten an die Child-Komponente übergeben kann.



Comp-1.jsx

```
import React, { Component } from 'react';
import Comp2 from '../component/Comp-2';

class Comp1 extends Component {
  constructor(props) {
    super(props);
    this.state = { name: "Peter" }
  }
  render() {
    return (
      ...
      <Comp2 name={this.state.name} />
      ...
    );
  }
  ...
}
```

Comp-2.jsx

```
class Comp2 extends Component {
  render() {
    return (
      ...
      <h1>Hallo {this.props.name}</h1>
      ...
    );
  }
}
```

React - CSS Module

Externe Stylesheets

Beispiel: Button Komponente mit Stylesheet Integration

```
import React, { Component } from 'react';
import styles from './button.css'; // Import css modules stylesheet
import './weitere-styles.css'; // Import regular stylesheet

class Button extends Component {
  render() {
    // Reference als Object
    return <button className={styles.wichtig}>ACHTUNG Button</button>;
  }
}
```

! nicht class

./button.css

```
...
.wichtig {
  color: red;
}
```

Beispiel: Inline Styles

Inline Stylesheets

```
import React, { Component } from 'react';
const Btn-form = {
  background-color: #f44336;
  color: white;
  padding: 15px 32px;
  text-align: center;
};

class Button extends Component {
  render() {
    return <button style={Btn-form}>ACHTUNG Button</button>;
  }
}
```

React - Bedingtes Rendering

```
class Ausgabe extends React.Component {
  constructor(props) {
    super(props);
    this.state = { isLoggedIn: true }
  }
  render() {
    return (
      <div>
        {this.state.isLoggedIn ? (
          <h1>Du bist angemeldet</h1> // true!
        ) : (
          <h1>Du bist nicht angemeldet!</h1>
        )}
      </div>
    );
  }
}

ReactDOM.render(
  <Ausgabe />, // Du bist angemeldet!
  document.getElementById('root')
);
```

Ein "If-Else" mit dem bedingten Operator ("ternäre Operator" wie bei Java) ist per (... ? ... : ...) einzubinden.

Für umfangreichere Ausdrücke sollte ggf. eine eigene Komponente genutzt werden.

React - Events (globaler Event-Listener)

- In React sind "Event-Handler" als Instanzen von "SyntheticEvent" zu übergeben
- Mit SyntheticEvents sind React-eigene Event gemeint und keine nativen DOM-Events!
- Die Schnittstellen zwischen SyntheticEvents und nativen DOM-Events sind allerdings gleich und können mit den bekannten Methoden aufgerufen und Properes abgefragt werden.

Beispiel: Event-Listener

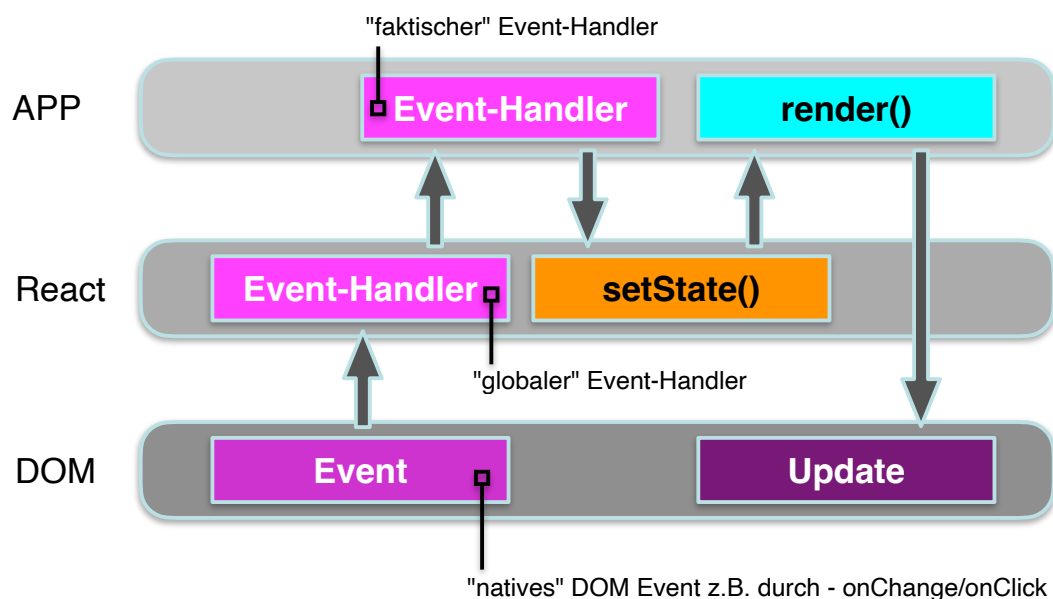
```

constructor() {
  this.handleChange = this.handleChange.bind(this)
}
handleChange() {
  this.setState({ input: event.target.value })
}
render() {
  return (
    ...
    <input
      type='text'
      value={this.state.input}
      onChange={this.handleChange} />
    )
  )
}

```

- React stellt dabei sicher, dass sich Event-Objekte browserübergreifend gleich verhalten!
- Da kein DOM-Event Händler erzeugt wird, muss React sicherstellen, dass SyntheticEvents über einen globalen Event-Händler genau am obersten Element/Knoten des nativen DOMs registriert werden.
- Dieser globale Event-Handler verteilt die Events an die registrierten "fachlichen" Händler, die in den einzelnen Komponenten registriert wurden (siehe nächste Folio).
- Über den Event-Händler wird der Zustand der Komponente per "setState()" verändert, was zur Folge hat, dass die Komponente re-rendert wird.

React - Datenfluss



React - Methoden für Event-Handler

- Mittels "bind"-Methode wird eine neue Methode erzeugt, die fest an die Klasse der Komponente gebunden ist.
- "this" zeigt dabei auf die Instanz der Klasse (Beispiel: State und Controlled Component).
- Im nebenstehenden Beispiel können unterschiedliche Parameter direkt an den Event-Händler übergeben werden. Dadurch kann die Methode nicht über den Konstruktor gebunden werden,
- und deshalb direkt beim Registrieren des Event-Handlers gebunden werden.
- Dabei wird dann nicht nur "this", sondern auch die übergebenen "Parameter" fest gebunden.

Beispiel: Instanzmethode mit bind

```
class InputState extends Component {
  constructor() {
    super()
    this.state = {counter: 0}
  }
  onCount(increment) {
    this.setState(
      {counter: this.state.counter + increment}
    )
  }
  render() {
    return (
      <>
        // Referenz auf Funktion 'onCount'
        <button onClick={this.onCount.bind(this, 1)}>
          {counter}
        </button>
      </>
    )
  }
}
```

Beispiel: Alternative Arrow / Lambda Funktion

```
<button onClick={() => this.onCount(1)} />
```

- Innerhalb der Lambda-Funktion wird das this and das this des umschließenden Scopes (Instanz der Komponente) gebunden und damit das this der Handler-Methode korrekt gesetzt.
- Ein bind innerhalb des Konstruktes erübrigt sich natürlich.

React - Übung Components

- Erstellen Sie eine eigene Komponente, die einen Button für das vorhergehende Beispiel darstellt und damit wiederverwendbar wird.
- Der Button soll die
 - Hintergrundfarbe rot,
 - Schriftfarbe weiß, Textgröße 16px haben und
 - es soll der Counter auf dem Button angezeigt werden..

Übung: Button Component erstellen

```
import React, { Component } from 'react';
import Button from '../components/button';

...

<Button onClick={() => this.onCount(1)} />

// oder

<Button onClick={this.onCount.bind(this, 1)}>
  {counter}
</Button>

...
```

React - Übung Components

Lösung:

Übung: index.jsx

```
import React, { Component } from 'react';
import "../css/Button.css";

class Button extends Component {
  render() {
    return (
      <div>
        <button className="Btn-form">
          <strong>{this.props.label}</strong>
        </button>
      </div>
    )
  }
}

export default Button;
```

Übung: ../css/button.css

```
.Btn-form {
  background-color: #f44336; /* red */
  color: white; /* white */
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
}
```

React - Object Spread-Operator

- Innerhalb von ES6 ist der Spread-Operator nur auf Arrays anwendbar.
- In React kann er auf Objekte angewandt werden.

Beispiel: ES6 Spread-Operator

```
const str = 'String';
const char = [...str];
console.log(char); // ['S','t','r','i','n','g']
```

Beispiel: React Object Spread-Operator

```
const obj = { auto: 'BMW', farbe: 'rot' };
const newObj = {...obj};
console.log(newObj); // { auto: 'BMW', farbe: 'rot' }
```

- Mittels Spread-Operator können Properties (props) per Attribut an Komponenten und Funktionen übergeben werden. Hierzu müssen sie nicht explizit benannt werden.
- Damit ist es hilfreich Properties, in der Hierarchie, von einer "oberen" and eine "untere" Komponente durchzureichen, ohne, dass die dazwischenliegenden Komponenten den Inhalt der Properties explizit kennen.
- Der Spread-Operator geht ggf. zu Lasten der Nachvollziehbarkeit des Quellcodes!

Beispiel: React Spread-Operator - Weitergabe von props

```
funktion Seite(props) {
  return <div><Header {...props} /></div>
}

funktion Header(props) {
  Return <Title {...props} />
}

funktion Title(props) {
  Return <h1>{props.name}</h1>
}

ReactDOM.render(
  <Seite name="Max Muster" />
  Document.getElementById(ausgabe)
);
```

React - Listen

Beispiel: Listen ohne Key

```
...
this.state = {
  list: [
    {title: 'List_1'},
    {title: 'List_2'}
  ];
}
...
const Listen = () => (
  <div>
    <ul>
      {this.state.list.map((item) => (
        <li>{item.title}</li>
      ))}
    </ul>
  </div>
);
...
```

Beispiel: Listen mit Key

```
import { v4 as uuidv4 } from 'uuid';
// https://www.npmjs.com/package/uuid
...
this.state = {
  list: [
    {id: uuidv4(), title: 'List_1'},
    {id: uuidv4(), title: 'List_2'}
  ];
}
...
const Listen = () => ( // Kurzschreibweise
  <div>                // als Arrow-Funktion
    <ul>
      {this.state.list.map((item) => (
        <li key={list.id}>, {list.title}</li>
      ))}
    </ul>
  </div>
);
...
```

- Folgende Warnung wird ausgegeben: *'Each child in a list should have a unique "key" prop.'*
- Die Warnung besagt, dass ein eindeutiges Schlüsselattribut zu jedem der Listenelemente hinzugefügt werden muss.
- Im Hintergrund verwendet React das Key-Attribut, um das gerenderte Element mit einem festen Platz in der gerenderten Liste zu verbinden. Dies stellt die eindeutige Kennung für das Element in der Liste dar.
- Im zweiten Beispiel wird den Listen ein eindeutiger Key mittels UUID V4 (siehe auch: [RFC 4122](https://tools.ietf.org/html/rfc4122)) zugewiesen und hilft React zu erkennen, welche Elemente geändert, hinzugefügt oder gelöscht wurden.

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

119

Questions

&



Answers

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

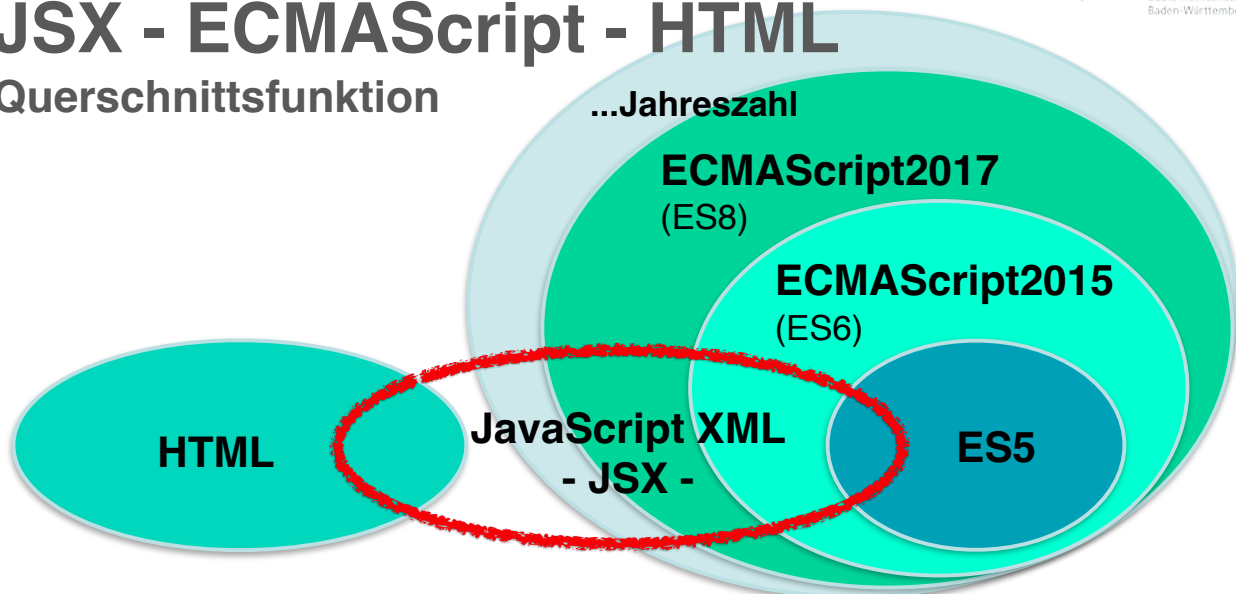
120

JSX (JAVAScript XML)

Eine Einführung

JSX - ECMAScript - HTML

Querschnittsfunktion



JavaScript (European Computer Manufacturers Association)

- <https://www.w3resource.com/javascript/introduction/ECMA-and-javascript.php>

ECMAScript Language Specification

- <https://tc39.github.io/ecma262/>

JSX - JavaScript XML Specification

- <https://facebook.github.io/jsx/>

JavaScript und ECMAScript

- JavaScript nur eine von weiteren möglichen Implementierung einer ideellen Sprachdefinition von ECMAScript.
- ECMAScript ist der Realisierung von JavaScript-Engines immer etwas voraus (das "Feature-Set" wird jährlich festgelegt). ECMAScript wird für 2020 wohl im Juni des Jahres geprüft und genehmigt (Vorschlag: <https://github.com/tc39/proposals>). Herkömmliche Browser unterstützen lediglich ES6 (2015) zu 100% (siehe auch <https://caniuse.com/>).
- JavaScript ist ursprünglich eine interpretierte Sprache. Moderne Engines verfolgen jedoch aus Performancegründen den Compiler-Ansatz.
- Normalerweise wird JavaScript im Browser ausgeführt, wozu die Browserhersteller ihre eigene JavaScript-Engines (JavaScript-Engines und nicht Browser-Engines wie: Gecko "Firefox", WebKit "Safari" und Blink "Chrome") entwickelt haben:
 - Chrome: V8
 - Firefox: SpiderMonkey
 - Safari: JavaScriptCore

JSX

- Ist eine syntaktische Erweiterung zu JavaScript, die HTML und JavaScript insoweit vermischt, dass JS nur benutzt wird, um den DOM zu manipulieren.
- Innerhalb der "render-Funktion" ist es möglich HTML schreiben zu können? React verwendet dazu JSX, eine Syntaxerweiterung für JavaScript, mit der es leichter fällt Markup zu schreiben.
- Ein populäres Styleguide (Codestyling: Benennen von Komponenten, Variablen und Dateien) ist der Airbnb React/JSX Style Guide:
["https://github.com/airbnb/javascript/tree/master/react"](https://github.com/airbnb/javascript/tree/master/react)

JSX-Component:

```
import React from 'react';
import ReactDOM from 'react-dom';

function Willkommen()
{
  return <h1>Hello World!</h1>
}

ReactDOM.render(
  <Willkommen />,
  document.getElementById("root")
  /* Wird im "root"-DOM-Knoten innerhalb der HTML-Datei gerendert:
    <div id="root"></div> */
);
```

JSX - Variablen und Ausdrücke

```
import React from 'react';
import ReactDOM from 'react-dom';

const tit1 = "Hello"; // string
const tit2 = "World";

let title1 = `01: <h1>${tit1} ${tit2}</h1>`;
console.log(title1);

let title2 = "02: <h1>" + tit1 + " " + tit2 + "</h1>";
console.log(title2);

function getTitle() {
  const tit1 = "Hello"; // ist außerhalb von getTitle()
  const tit2 = "World"; // nicht sichtbar!
  return "03: <h1>" + tit1 + " " + tit2 + "</h1>";
}
console.log(getTitle());

ReactDOM.render(
  document.getElementById('root')
);
```

- Variablendeklaration mit "let", "const" unterscheiden sich mit denen von "var" durch den Scope der Sichtbarkeit durch den sie definiert wurden:
 - var: ist eine globale Variable
 - let: Gültigkeit bezieht sich nur auf einen lokalen Block (Funktion, Schleife).
 - const: Gültigkeit wie bei let, jedoch kann der Wert nicht verändert werden.
- Jeder valide JavaScript-Audruck kann in geschweifte Klammern "{}" gesetzt und dargestellt werden.

JSX - Objekte

```
import React from 'react';
import ReactDOM from 'react-dom';

const contact = {
  vorName: 'Max',
  nachName: 'Mustermann'
}

function combineName(name) {
  return name.vorName + ' ' + name.nachName;
}

const gruss = 'Ich heiße: ' + combineName(contact);
console.log(gruss);

ReactDOM.render(
  gruss,
  document.getElementById('root')
);
```

Hier wird der Aufruf einer JavaScript-Funktion bezüglich eines Objektes zurück gegeben.

JSX - Destructuring

- Destructuring wurde in ES6 eingeführt. Es handelt sich dabei um eine JavaScript-Funktion, die es ermöglicht, mehrere Daten aus einem Array oder Objekt zu extrahieren und sie eigenen Variablen zuzuordnen.
- Destructuring wird in React intensiv genutzt, um den Umgang mit Properties wesentlich zu vereinfachen.

Beispiel: Kontakt-Objekt

```
const contact = {
  person: {
    name: 'Max',
    adresse: {
      plz: '22357',
      ort: 'Hamburg',
    }
  },
  email: 'max@domain.ext'
};
```

Beispiel: Destructuring

```
const {per} = contact; //Auspacken des Objekts
const {adr} = per; //in eigene Variablen
console.log(`Adresse=${adr.plz} ${adr.ort}`);
//Adresse=22357 Hamburg

// Die Namen der Properties können sich auch
// unterscheiden:

const {name: firstName} = per; //umbenennen
console.log(`Vorname=${firstName}`); //Vorname=Max
```

Beispiel: Destructuring in beliebiger Tiefe

```
const {person: {adresse: {plz, ort: origin}}} = contact; // ort in origin umbenennen
console.log(`Type of Adresse=${typeof adresse}`); // Type of Adresse=undefined

// ACHTUNG:
// 'adresse' ist keine Variable, sondern nur als Pfad zu tieferen Properties definiert!

console.log(`plz=${plz}`); // plz=22357
console.log(`ort=${origin}`); // ort=Hamburg
```

JSX - Destructuring

- Wir sind bereits damit vertraut, Objekt-Eigenschaften der ersten Ebene zu extrahieren und sie in ihre eigenen Variablen einzubinden:
 - `const {name} = contact;`
- Wenn wir von einer tiefer verschachtelten Objekteigenschaft wie 'plz' oder 'ort' Gebrauch machen wollen, können wir dies in einer speziellen Einzelextraktion (single-line extraction) tun:
 - `const {person: {adresse: {plz, ort: origin}}} = contact;`
- Wir müssen dann nicht mehr auf "contact.person.adresse.plz" referenzieren, sondern können direkt auf die Variable zugreifen,
 - `<p>PLZ = {plz}</p>`
 - `<p>Ort = {ort}</p>`
 wodurch der Code leichter lesbar wird.
- Eine weitere Destructuring-Möglichkeit gibt uns die "Rest-Syntax", die durch drei Punkte (...) gekennzeichnet ist (nicht zu verwechseln mit dem Spread-Operator!).
- In der Rest-Zuweisung können alle verbleibenden Eigenschaften des Objects in einer eigenen Variablen gruppiert werden:
 - `const { person, ...rest } = contact;`
 - `console.log(person);` // Object {vorName: "Max", nachName: "Mustermann", adresse: {...}}
 - `console.log(rest);` // Object {email: "max@domain.ext"}

JSX - Destructuring

Destructuring bei Funktionen:

Beispiel: Kontakt-Objekt

```
const contact = {
  person: {
    name: 'Max',
    adresse: {
      plz: '22357',
      ort: 'Hamburg',
    }
  },
  email: 'max@domain.ext'
};
```

Beispiel: Destructuring bei Funktionen

```
function schreibe({email: mailAdresse}) {
  console.log(`Email-Contact=${mailAdresse}`);
}
schreibe(contact); // Email-Contact=max@domain.ext
```

Destructuring bei Arrays:

Beim Destructuring von Arrays gibt die Position der Variablen die Zuordnung an!

Beispiel: Destructuring bei Arrays

```
const [vorName, nachName] = ['Max', 'Mustermann'];
console.log(`Vorname=${vorName}`); // Vorname=Max
console.log(`Nachname=${nachName}`); // Nachname=Mustermann
```

Übung - Objekte verknüpfen

1. **Objekt** - activityList: ["625aa16e1fadc70311b605ef","625aa16e1fadc70311b605f0"]
2. **Objekt** - groupedOrder:

 {"625aa16e1fadc70311b605ef":

 {

 "ownerId":"bfe1257c-48f1-4cce-95f7-92f80c579f6d","activityId":"625aa16e1fadc70311b605ef","activityText":"HipHop Coole Moves","activityCurrency":"EUR","ownerId":"000000021fadc70311b605db","ownerName":"tanzschule","ticketsIndex":0,"ticketsCount":2,"ticketsPrice":0,"ticketsValue":0,"ticketsText":"Normalpreis"

 },

 "625aa16e1fadc70311b605f0":

 {

 "ownerId":"fe4c7ca6-cd1e-4a97-b6a7-59041382a1f8","activityId":"625aa16e1fadc70311b605f0","activityText":"Tango Argentino","activityCurrency":"EUR","ownerId":"000000021fadc70311b605db","ownerName":"tanzschule","ticketsIndex":1,"ticketsCount":3,"ticketsPrice":0,"ticketsValue":0,"ticketsText":"Auszubildende, Schüler, Studierende"

 }

 }

Übung - Objekte - ZielObjekt

```
paymentIntent:
  { ownerId: "IdDesAnbieters",
    tickets: [{
      activityId: "IdDerErstenAktivität",
      tickets: [
        { ticketIndex: Index },
        { ticketIndex: Index },
        ...
      ]},
    {
      activityId: "IdDerZweitenAktivität",
      tickets: [
        { ticketIndex: Index },
        { ticketIndex: Index },
        ...
      ]}
  ]
}
```

Übung - Objekte - Lösung

Beispiel: Objekt konvertieren

```
let tickets = [];
let perProvider = [];
activityList.map((order) => {
  activityId = groupedOrder[order][0].activityId;

  groupedOrder[order].map((singleOrder) => {
    for (let i=0; i<singleOrder.ticketsCount; i++ ) {
      tickets.push({ ticketIndex: singleOrder.ticketsIndex });
    }
  });

  perProvider.push({ activityId: activityId, tickets:tickets });
  activityId = "";
  tickets = [];
});

const paymentIntent = {
  ownerId: ownerId,
  tickets: perProvider
};
```


JSX - Template Strings

- Ein Template String (kurz: Template) ähnelt einem Zeichenketten-Literal und einem Literal für reguläre Ausdrücke.
- Templates erlauben das Einbetten von Ausdrücken/JSX Code in einen String.
- Dies kann auch genutzt werden, um einen String mit Zeilenumbrüchen zu verarbeiten.
- Ein Template-String wird nicht mit normalen Anführungszeichen, sondern mit Backticks (`...`) umschlossen.

Beispiel: Template Strings

```
const contact = {
  vorName: 'Max',
  nachName: 'Mustermann',
  adresse: {
    plz: '22357',
    ort: 'Hamburg',
  },
  email: 'max@domain.ext'
};
console.log(`${contact.vorName} // mehrzeile Ausgabe möglich
  kann ${759 * 0.30} EUR Fahrtkosten steuerlich absetzen!`);
```

JSX - map and reduce

- "map" und "reduce" iterieren über ein Array/Objekt und führen eine Transformation oder Berechnung durch. (map und reduce ist nicht mit "MapReduce", dem von Google eingeführten "Data-Processing Programming Model" auf Computerclustern zu verwechseln: <https://research.google/pubs/pub62/>)
- Die map()-Methode wird zum Erstellen eines neuen Arrays aus einem vorhandenen Array verwendet, wobei auf jedes der Elemente des ersten Arrays eine Funktion angewendet wird.
- Die reduce()-Methode reduziert ein Array von Werten auf nur einen Wert. Um den Ausgabewert abzurufen, wird für jedes Element des Arrays eine Reduzierungsfunktion ausgeführt.

Beispiel: map / reduce

```
const iso = new Date().toISOString(); // ISO Datum Format: 2021-11-05T11:34:12.000Z
let dat = new Date(iso);
dat.toISOString(); // konvertiert in lokales Datum

const objArr = [
  {dat: iso, text: "Prod-1", preis: "11.50", anzahl: 20},
  {dat: iso, text: "Prod-2", preis: "12.00", anzahl: 30},
  {dat: iso, text: "Prod-3", preis: "13.50", anzahl: 10}
];

let umsatzProd = [0.00, 0.00, 0.00];
objArr.map((item, index) => {
  umsatzProd[index] = parseFloat(item.preis) * parseInt(item.anzahl);
});
// [230,360,135]

let totalUmsatz = umsatzProd.reduce((prev, current) => prev + current);
// 725
```

JSX - Promises (ECMAScript 2015)

- Ein "Promise" (Versprechen) ist eine Art Platzhalter Objekt für ein zukünftiges Ergebnis (asynchrone Aktion). Das Ergebnis kann nun nach einer bestimmten Zeit erfolgreich eintreffen (fulfilled), noch schwebend (pending), oder es kann ein Fehler auftreten (rejected).
- Das Ergebnis lässt sich durch "chainen" (Aufruf von zwei oder mehr asynchronen Funktionen nacheinander in Sequenz)
 - „.then((result) => {})“ bzw.
 - „.catch((error) => {})“ weiter verarbeiten.

Beispiel: Promise - Abfrage bei Wikipedia

```
const url=`https://en.wikipedia.org/w/api.php?action=query
&list=search&prop=info&inprop=url&utf8=&format=json
&origin=*&srlimit=20&srsearch=Albert Einstein`;

fetch(url)
  .then (response => response.json())
  .then (json => console.log("Albert Einstein: " + JSON.stringify(json)));
  .catch (response => response.json());
};
```

JSX - Async/Await (ECMAScript 2017)

- Der Aufruf einer asynchronen Funktion gibt ein Versprechen/Promise zurück.
- Um das Ergebnis, wenn das Versprechen erfüllt ist, tatsächlich nutzen zu können, wird ein ".then()" - Block verwendet, der das Versprechen zurückgibt.
- Das Schlüsselwort "async" wird zur Funktion hinzugefügt, um anzuweisen, dass ein Versprechen anstatt das Ergebnis direkt zurückzugeben.
- Der Vorteil einer asynchronen Funktion wird erst deutlich, wenn er mit dem Schlüsselwort "await" kombiniert wird.
- "await" kann vor jede asynchrone, auf Versprechen basierenden Funktion gestellt werden, um den Code in dieser Zeile anzuhalten. Und dies solange, bis das Versprechen erfüllt ist. Danach wird das resultierenden Ergebnis zurückzugeben.

Beispiel: Asynchrone Funktionen (Promises) synchronisieren (Bsp. In React-Training 1)

```
let hello = async function() { return "Hello" };
// oder
let hello = async () => { return "Hello" };

hello().then((value) => console.log(value))
// oder
hello().then(console.log)

async function hello() {
  await Promise.resolve("Hello") //resolved - ist erfüllt!
  .then (res => console.log("Ich sage " + res))
};
hello();
```

JSX - Operatoren

- (ausdruck1 || ausdruck2)
- Wenn ein Wert zu "true" konvertiert werden kann, wird er "truthy" genannt. Wenn ein Wert zu "false" konvertiert werden kann, wird er "falsy" genannt.
- Gibt "ausdruck1" zurück, wenn er zu "true" kommt. Anderenfalls wird "ausdruck2" zurückgegeben. Ist einer der Operanden "true", gibt || "true" zurück.
- Gibt "false" zurück, wenn der Operand zu "true" konvertiert werden kann, andernfalls wird "true" zurückgegeben.
- (Fall_A && Fall_B)
- Wenn "Fall_A" wahr ist, dann wird "Fall_B" ausgeführt
- Sind beide Operanden "true", gibt && "true" zurück, andernfalls "false".

Beispiel: Boolesche Operator-1

```
const a = 5;
const b = -1;

console.log(a > 0 && b > 0);
// false
console.log(a > 0 || b > 0);
// true
console.log(!(a > 0 || b > 0));
// false
```

Beispiel: Short Circuit Evaluation

```
// && Operator
{Fall_A && Fall_B}

// gleich
if(Fall_A) {
  Fall_B
}
```

Beispiel: Logisches Nicht!

```
x = !true // x wird zu false
y = !false // y wird zu true
```

Beispiel: Konvertierung

```
x || y // ist equivalent !(x && !y)
!!x // ist äquivalent x
```

JSX - Operatoren

Aritmetische	Bedeutung	Beispiel
+, +=	Addition	x += 3
-, -=	Subtraktion	x = x - 3
*, *=	Multiplikation	x = y * z
/, /=	Division	x = y / z
%	Modulus	x = 10 % 3 = 3, Rest 1
++, --	Inkrement, Dekrement	x++ oder y--

Relational	Bedeutung	Beispiel
>, >=	Größer (gleich)als	(A > B) ist false
<, <=	Kleiner (gleich)als	(A <= B) ist true
==	Gleich	(A == B) ist false
!=	Ungleich	(A != B) ist true

Bit-Weise	Bedeutung	Beispiel
& (bitweise UND)	logische UND-Verknüpfung	1 & 2 = 0
(bitweise ODER)	logische ODER-Verknüpfung	1 2 = 3
^ (bitweise XOR)	Exklusiv-ODER-Verknüpfung	(A ^ B) ist 1
~ (bitweise NICHT)	Aus 0 wird 1 und umgekehrt	(~ B) ist -4

Übung - Logische Operatoren

Boolsche Logikfunktion - George Boole (1815-1864) war Mathematiker und Philosoph.

UND: Du darfst deinen Pudding nur essen, wenn du dein Gemüse gegessen hast und wenn du trotzdem noch hungrig bist

- Wenn ich mein Gemüse gegessen habe und nicht mehr hungrig bin, darf ich den Pudding essen -> Wahr/Falsch
- Wenn ich das Gemüse gegessen habe und noch hungrig bin, darf ich den Pudding essen -> Wahr/Falsch

Oder: Du darfst im Park spazieren gehen, wenn Wochenende ist oder du Urlaub genommen hast

- Wenn Wochenende ist, darf ich im Park spazieren gehen -> Wahr/Falsch
- Wenn ich Urlaub habe und es ist Wochenende, darf ich im Park spazieren gehen -> Wahr/Falsch

XOR: Du darfst zum Training gehen, entweder wenn du alles erledigt hast oder du dein Zimmer aufgeräumt hast

- Wenn ich alles erledigt habe und mein Zimmer aufgeräumt ist, darf ich zum Training gehen -> Wahr/Falsch
- Ich habe alles erledigt, es aber nicht geschafft mein Zimmer aufzuräumen, dann darf ich zum Training gehen -> Wahr/Falsch

Übung - Frage

Logische Operatoren?

<https://pingo.coactum.de/462993> ([start](#))



Übung - Logische Operatoren (Lösung)

Boolsche Logikfunktion - George Boole (1815-1864) war Mathematiker und Philosoph.

UND: Du darfst deinen Pudding nur essen, wenn du dein Gemüse gegessen hast und wenn du trotzdem noch hungrig bist

- Wenn ich mein Gemüse gegessen habe und nicht mehr hungrig bin, darf ich den Pudding essen -> **Wahr/Falsch**
- Wenn ich das Gemüse gegessen habe und noch hungrig bin, darf ich den Pudding essen -> **Wahr/Falsch**

Oder: Du darfst im Park spazieren gehen, wenn Wochenende ist oder du Urlaub genommen hast

- Wenn Wochenende ist, darf ich im Park spazieren gehen -> **!!! Wahr/Falsch ???**
- Wenn ich Urlaub habe und es ist Wochenende, darf ich im Park spazieren gehen -> **Wahr/Falsch**

XOR: Du darfst zum Training gehen, entweder wenn du alles erledigt hast oder du dein Zimmer aufgeräumt hast

- Wenn ich alles erledigt habe und mein Zimmer aufgeräumt ist, darf ich zum Training gehen -> **Wahr/Falsch**
- Ich habe alles erledigt, es aber nicht geschafft mein Zimmer aufzuräumen, dann darf ich zum Training gehen -> **Wahr/Falsch**

Übung - Logische Operatoren

Wahrheitstabelle

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Questions

&

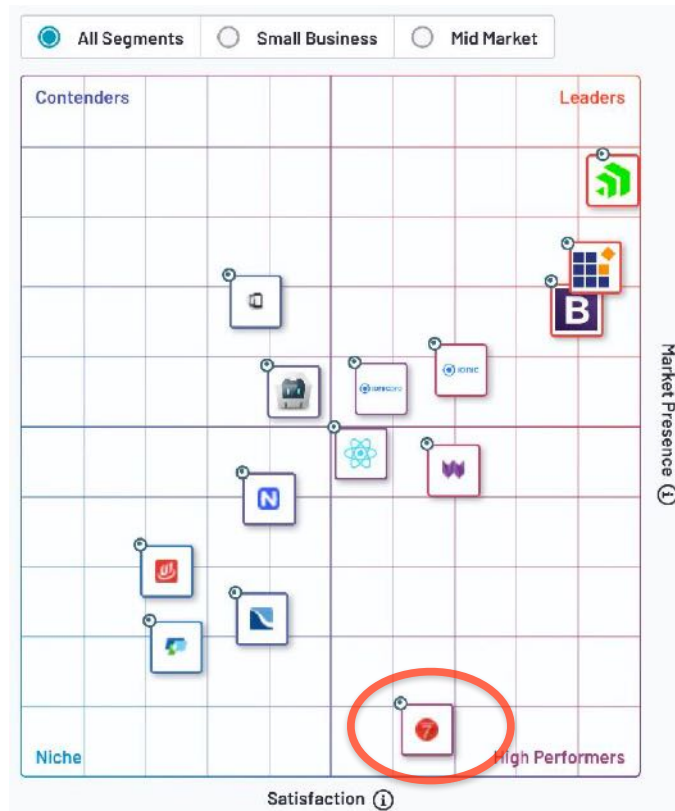


Answers

MOBILE FRAMEWORKS

Framework für hybride und mobile Web-Apps

Marktübersicht



<https://www.q2.com/categories/mobile-development-frameworks?segment=all>

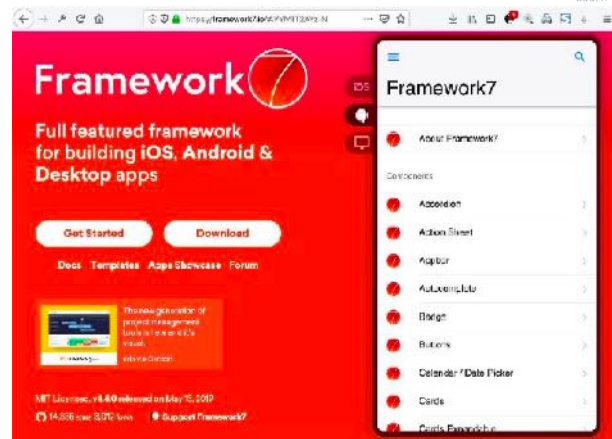
Framework7 und React

- React ist eine JavaScript-Bibliothek zum Erstellen und manipulieren von HTML-basierten Benutzeroberflächen (ähnlich jQuery). React wurde 2013 von Facebook unter BSD-Lizenz veröffentlicht.
 - React ermöglicht eine "Komponenten"-Architektur, die als Basis für eine modulare und leicht nachzuvollziehende Frontend-HTML- und JavaScript-Entwicklung dient.
 - React versteht sich als Bibliothek und nicht als Framework.
-
- Framework7 React kombiniert die Leistungsfähigkeit des React-Frameworks mit Flexibilität und UI (Side Panels, Popups, List Views, Action Sheets, Tabs, etc.).
 - Dem Entwickler für iOS-, Android-Apps mit HTML, CSS und JavaScript erschliessen sich die üblichen mobile UI-Elemente, um dadurch einfach und sehr schnell Apps gemäß dem "mobile first" Ansatz zu erstellen.
 - Framework7 ist ein MIT lizenziertes Open-Source-Projekt welches ständig weiterentwickelt wird.

Framework7

Framework7 lässt sich einsetzen, um hybride mobile Apps (PWA's) zu entwickeln.

Zu den Elementen und Widgets von Framework7 gehören etwa Action-Sheets, Tabs, Pop-ups und Side-Panels etc.



Installation Command Line Interface (CLI):

<https://framework7.io/cli/>

\$ npm install framework7-cli -g (Node Package Manager)

App-Template (im jeweiligen Verzeichnis) initialisieren:

\$framework7 create

Beispiel Applikation (Get Started):

<https://framework7.io/docs/kitchen-sink.html>

Handbuch (englisch):

https://www.academia.edu/33673842/Framework7_tutorial?email_work_card=view-paper

Projekt initialisieren

Nach der Installation von Framework7, die App im entsprechenden Projekt-Verzeichnis initialisieren und ausführen:

\$ framework7 create

\$ npm start

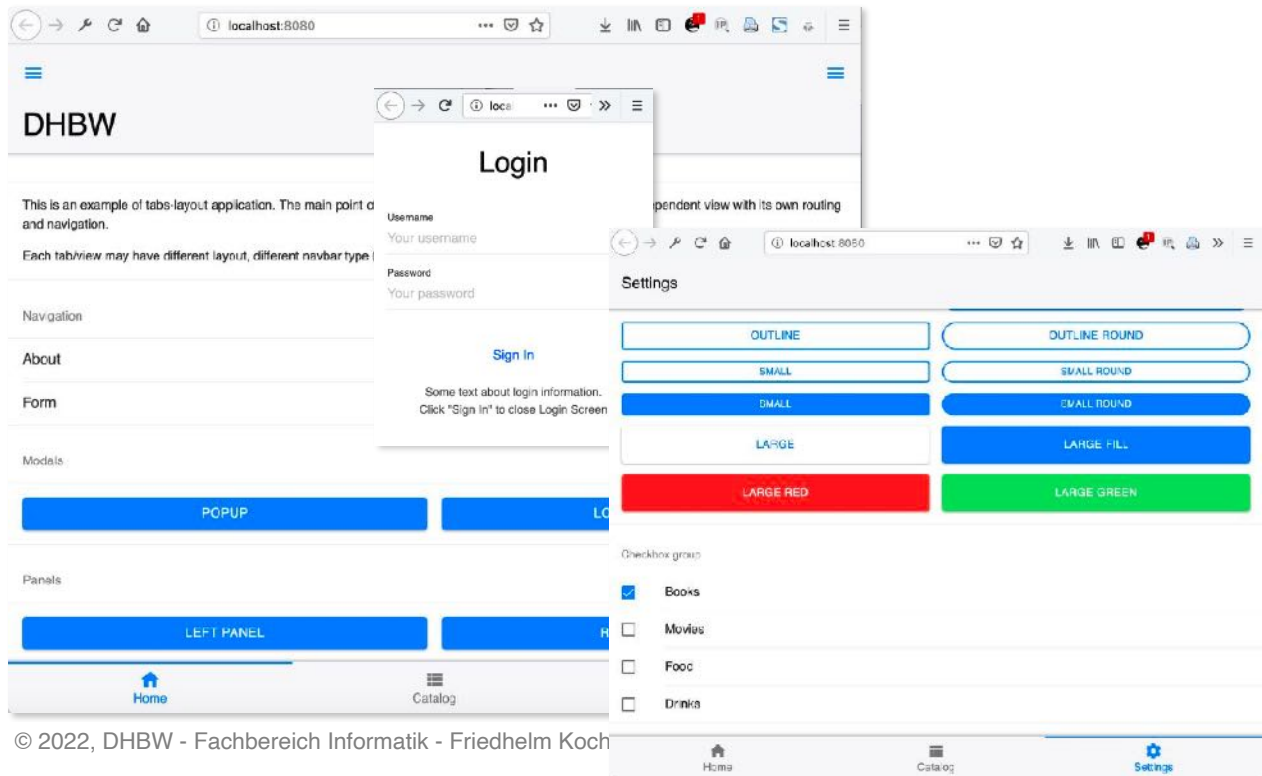
```

Freddy's React-Example $ framework7 create
✓ All good, you have latest framework7-cli version.
? What type of the app are you targeting? Simple web app, PWA (Progressive Web App)
? App (project) name: React-Examples
? What type of framework do you prefer? Framework7 with React
? Choose starter template: Tabbed Views (Tabs)
? Do you want to setup CSS Pre-Processor No, I am good with CSS
? Do you want to specify custom theme color? No, use default color theme
? Do you want to include Framework7 Icons and Material Icons icon fonts? Yes, include icon fonts
✓ Generating package.json
✓ Creating required folders structure
✓ Installing NPM Dependencies
✓ Installing NPM Dev Dependencies
✓ Executing NPM Scripts
✓ Creating project files
✓ Done! 🎉

i Next steps:
- 🔥 Run "npm run start" - run development server
- 🛠 Run "npm run dev" - run development server
- 🛠 Run "npm run build-dev" - build web app using development mode (faster build without minification and optimization)
- 🛠 Run "npm run build-prod" - build web app for production
- 📖 Visit documentation at https://framework7.io/docs/
- 📖 Check README.md in project root folder with further instructions
Love Framework7? Support project by donating or pledging on patreon:
https://patreon.com/vladimirkharlampidi
Freddy's React-Example $
  
```

Projekt-Template

Template mit den gängigsten Design UI-Elementen



Framework7 - API

Beispiel - Radio Group/List

Radio Group

```
<div class="list">
  <ul>
    <li>
      <label class="item-radio item-content">
        <input type="radio" name="demo-radio" value="Books" checked />
        <i class="icon icon-radio"></i>
        <div class="item-inner">
          <div class="item-title">Books</div>
        </div>
      </label>
    </li>
    <li>
      <label class="item-radio item-content">
        <input type="radio" name="demo-radio" value="Movies" />
        <i class="icon icon-radio"></i>
        <div class="item-inner">
          <div class="item-title">Movies</div>
        </div>
      </label>
    </li>
    <li>
      <label class="item-radio item-content">
        <input type="radio" name="demo-radio" value="Food" />
        <i class="icon icon-radio"></i>
        <div class="item-inner">
          <div class="item-title">Food</div>
        </div>
      </label>
    </li>
  </ul>
</div>
```



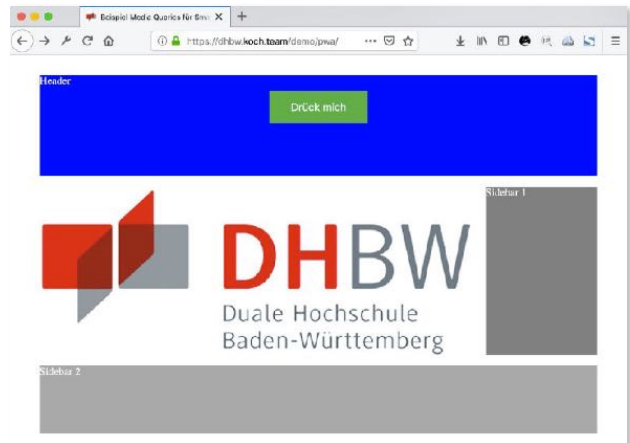
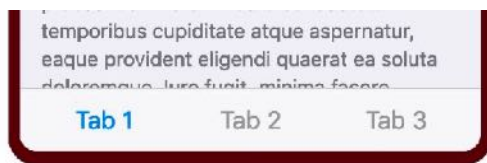
Übung

- Installieren Sie Framework7
- Erstellen Sie mittels der CLI ein Projekt in einem Projekt-Verzeichnis.
- Nehmen Sie das PWA-Beispiel aus

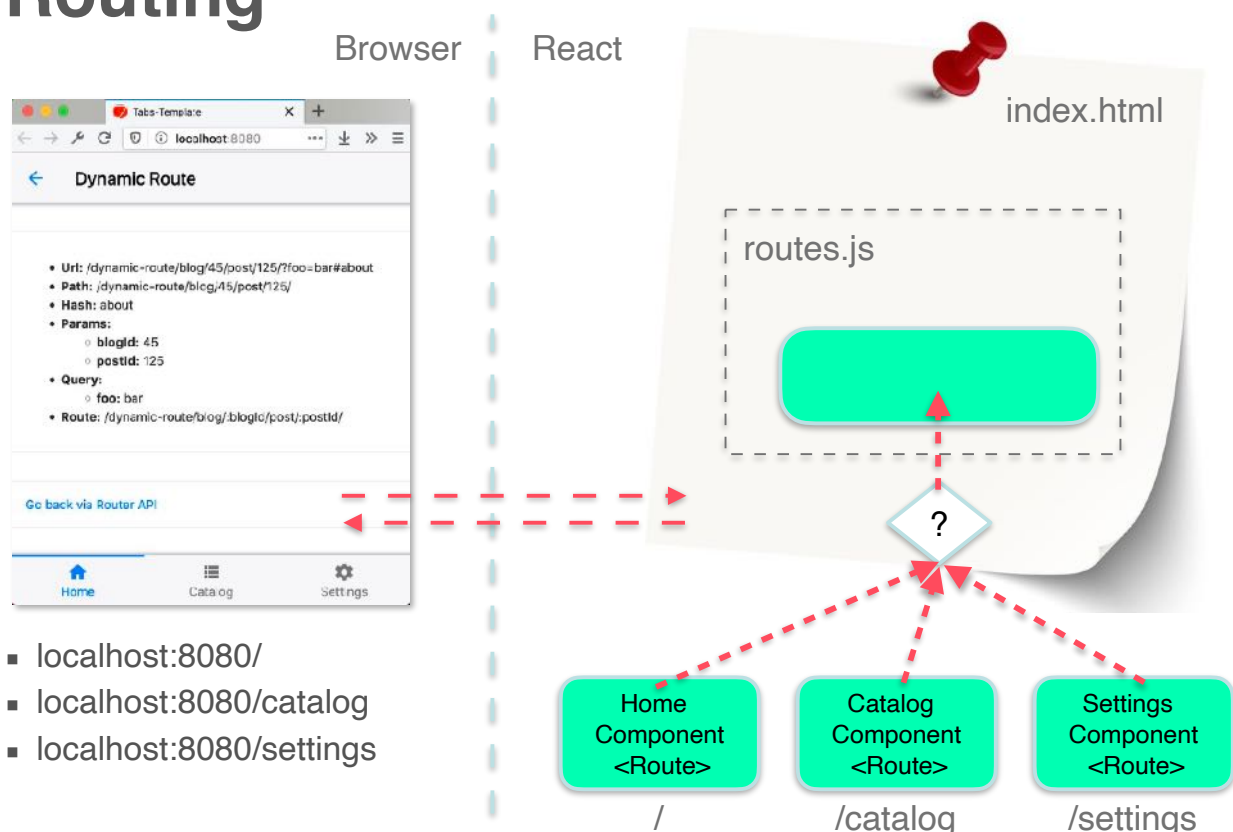
<https://github.com/FriedhelmKoch/DHBW>

- Erweitern Sie das Beispiel mit einem Tab-Layout

<https://framework7.io/docs/tabs.html>



Routing



Framework7 - Routing

routes.jsx

```
// Import routing pages components
import HomePage from '../pages/home.jsx';
import CatalogPage from '../pages/catalog.jsx';
import SettingsPage from '../pages/settings.jsx';
import NotFoundPage from '../pages/404.jsx';

...

var routes = [ // specify routes for app
  { path: '/', component: HomePage, },
  { path: '/catalog/', component: CatalogPage, },
  { path: '/settings/', component: SettingsPage, },
  { path: '(.*)', component: NotFoundPage, }
]
export default routes;
```

home.jsx

```
...
<BlockTitle>Navigation</BlockTitle>
<List>
  <ListItem link="/" title="Home"/>
  <ListItem link="/catalog/" title="Catalog"/>
  <ListItem link="/settings/" title="Settings"/>
</List>
...
```

- Grundsätzlich ermöglicht Routing verschiedene Komponenten, basierend auf der URL der Seite, zu rendern.
- Beim Routing geht es um das Zuordnen einer URL in der Anwendung zu einer bestimmten Funktion.
- Die Navigation der React-Komponenten werden über den Router gesteuert, so dass die Seiten (Components) den Routen zugeordnet sein müssen.
- Der Router stellt das Interface der Applikation mit seinen Komponenten dar und hält es synchron mit der URL des Browsers.

Questions

&



Answers

INDEXEDDB



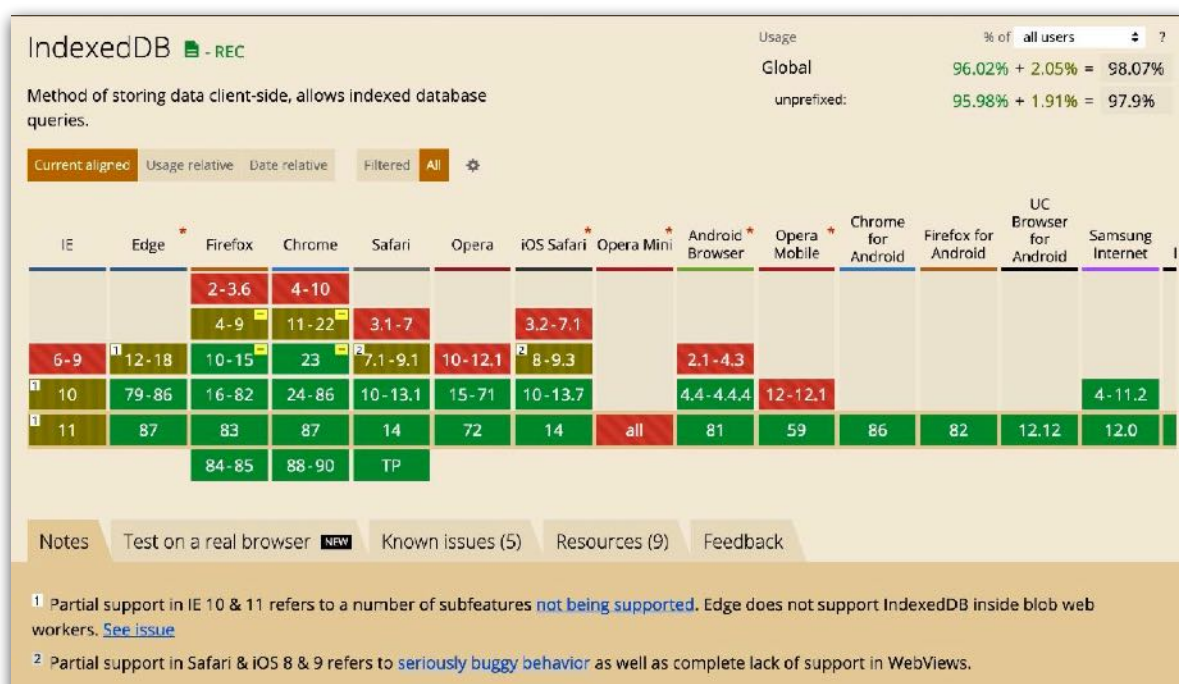
Connectivity Independent - PWA (3)

Asynchrone lokale noSQL-Database API

<https://w3c.github.io/IndexedDB/>

Browser Unterstützung

<https://caniuse.com/indexeddb> (abgerufen am 26.11.2020)



Arten von Datenspeichern

Im Allgemeinen gibt es zwei verschiedene Arten von Datenbanken:

- Relationale Datenbanken (SQL) und
 - Dokumentdatenbanken (auch als NoSQL oder auch als Objektspeicher bekannt).
-
- Relationale Datenbanken wie der Microsoft SQL-Server, MySQL und die Oracle Datenbank speichern Daten in Tabellen und Records.
 - Dokumentdatenbanken wie MongoDB, CouchDB und Redis speichern Daten als einzelne Objekte in Collections und Dokumente.
 - Zu den verschiedenen Speichertypen im Browser gehören LocalStorage, SessionStorage, Cookies, Web SQL und IndexedDB. IndexedDB ist eine Dokumentendatenbank auf Basis eines Key/Value Stores (object based).

Lokale Speicher im Browser

Cookies:

- Sind ein Key-Value-Store, in dem Daten als Zeichenfolge gegen einen Schlüssel gespeichert werden können.
Bsp.: `document.cookie="key=value"`
- Cookies sind mit einer Verfallszeit verbunden. Ist keine Verfallszeit angegeben ist, wird das Cookie am Ende der Browser-Sitzung verfallen.
- In den meisten Browsern können nur bis zu 4 kByte Daten im Cookie gespeichert werden.
- Cookie-Daten können nur von dem Server verwendet werden, von dem sie geschrieben wurden. 'First-Party-Cookies' folgen der Politik des 'gleichen Ursprungs' und können nicht domainübergreifend genutzt werden.
- 'Third Party-Cookies' hingegen werden durch Dritte gesetzt. Dies wird dadurch ermöglicht, dass die aufgerufene Webseite Inhalte dritter Server beinhaltet (z.B. ein Pixel-Image). Die Cookie-Policy lässt zu, dass damit alle diese involvierten Server zusätzlich Zugriffsrechte auf die gesetzten Cookies erhält.

Lokale Speicher im Browser cont'd

LocalStorage:

- Ist ein Key-Value-Store, der Werte als 'strings' speichert.
Bsp.: `localStorage.setItem("key", "value")`
- Die Daten, die im lokalen Speicher gespeichert werden, haben kein Verfallsdatum und bleiben persistent im Browser, bis die Daten im Browser manuell mit Hilfe von Einstellungen oder JavaScript gelöscht werden.
- In den meisten Browsern können bis zu 10 MByte an Daten gespeichert werden.

SessionStorage:

- Alternativ zu den Eigenschaften des LocalStorage speichert der SessionStorage die Daten nur für die Dauer der Browser-Session.

Lokale Speicher im Browser

- **IndexedDB** ist ein asynchrones NoSQL („Not only SQL“) Speichersystem. Damit können im Browser Daten im Form von Key/Value gespeichert werden. Neben den üblichen Aktionen zum Suchen, Abrufen und Setzen unterstützt IndexedDB auch Transaktionen.
- Jede IndexedDB-"Datenbank" ist in ihrem Zugriff eindeutig (Same-Origin-Policy - Top-Level-Domäne, Subdomäne, Port), was bedeutet, dass von einer anderen Domäne kein Zugriff darauf erfolgen kann.

Siehe auch: <https://www.w3.org/TR/IndexedDB/#idl-def-IDBEnvironment>

- Database: Ist die höchste Ebene von IndexedDB (Datenbank in SQL).
- Collection: Ist eine Sammlung von Dokumenten in den Daten gespeichert sind (Tabelle in SQL).
- Document: Ist die Einheit zum Speichern von Daten im JSON-Format (Record in SQL).

The screenshot shows the Redux DevTools application state. The left sidebar displays the component tree, with 'vi@Linked - https://via...' selected. The main panel shows the state of the 'vi@Linked' component, which is an array of objects. The first object is highlighted, showing its 'id', 'account', 'address', 'communication', 'indexes', 'personal', 'provider', 'search', and 'status' properties.

#	Key	Value
0	"activeUser"	<pre>{ id: "0000000273d35b2103e71a44", userName: "tanzschule", status: Array(3), account: { bank: "Sparkasse Musterstadt", bic: "GE00CNBP", iban: "DE92 7205 1200 0000 1111 22", address: { streetNo: "SchulstraÙe 9", zip: "85726", city: "Pfaffenhofen", ... }, communication: { phone: "+49 8252 8946 13", mobile: "+49 123 5678901244", ... }, id: "0000000273d35b2103e71a44", indexes: { cart: "01234566789", voucher: "", tickets: Array(1), contacts: ... }, personal: { salutation: "Herr", title: "", firstName: "Thomas", middleName: ... }, provider: { cardType: "H1", banner: "data:image/jpeg;base64,/9j/4QAYRhpZg=...", search: { hashTags: Array(9), subscriptions: Array(2), radiusSearch: 100, ... }, status: { "provider": true, "2021-04-27T17:45:33.000Z" }, userName: "tanzschule" } } }</pre>
1	"activity0w..."	{0000000073d35b2103e71a42: {...}}
2	"ownersActi..."	{6087af4a73d35b2103e71a4a: {...}}
3	"settings"	{appName: "vi@Linked", vi@LinkedLogo: "data:image/png;base64,iVBORw0KGQAAA="}

Total entries: 6

Ein einfach zu benutzende Bibliothek: <https://github.com/jakearchibald/idb-keyval>

IndexedDB Packages

react-indexed-db

- <https://www.npmjs.com/package/react-indexed-db>

idb

- <https://www.npmjs.com/package/idb>

dexie

- <https://dexie.org/docs/API-Reference#quick-reference>

idb-keyval

Eine winzige Bibliothek, die IndexedDB mittels IDB-Request-Objekte durch Promises (asynchron) nutzbar macht, plus ein paar kleinen Erweiterungen. Für die simple Nutzung ist dies völlig ausreichend.

- <https://www.npmjs.com/package/idb-keyval>

idb-keyval - Beispiel

app.jsx

```
// Default Database name ist: "keyval-store" und Store name ist: "keyval"
//
import { set, get, del, clear, keys } from 'idb-keyval';
...
set('hello', 'DHBW');    // key = hello, value = DHBW
...
get('hello').then((val) => console.log('value: ' + val)); // value: DHBW
...
keys().then((keys) => console.log('keys: ' + keys));      // keys: hello
...

// Benutzerspezifizierter Datenbank und Store name
import { createStore } from 'idb-keyval';
...
const custom_db_name = 'Hochschule';
const custom_store_name = 'DHBW';
const customStore = createStore(custom_db_name, custom_store_name);
...
set('hello', 'DHBW', customStore);
...

// Löscht alle values im entsprechenden store
del('hello');
del('hello', customStore);
...
```

Questions

&



Answers

REDUX

A Predictable State Container

Redux - installieren

Redux:

<https://redux.js.org/>

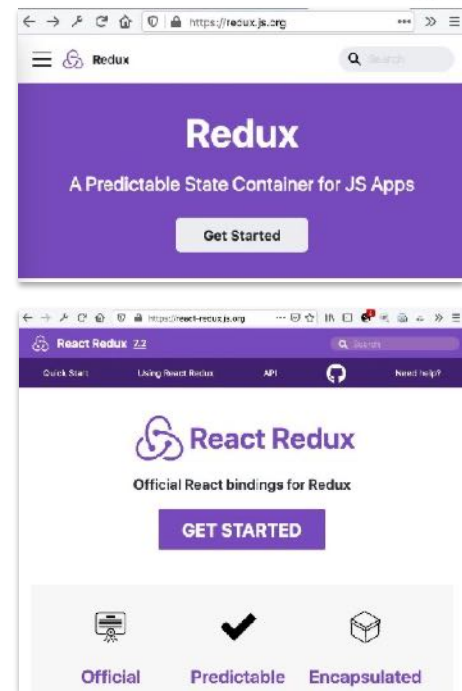
Im Terminal:

Redux installieren:

\$ npm install redux --save

Eine React Redux App installieren:

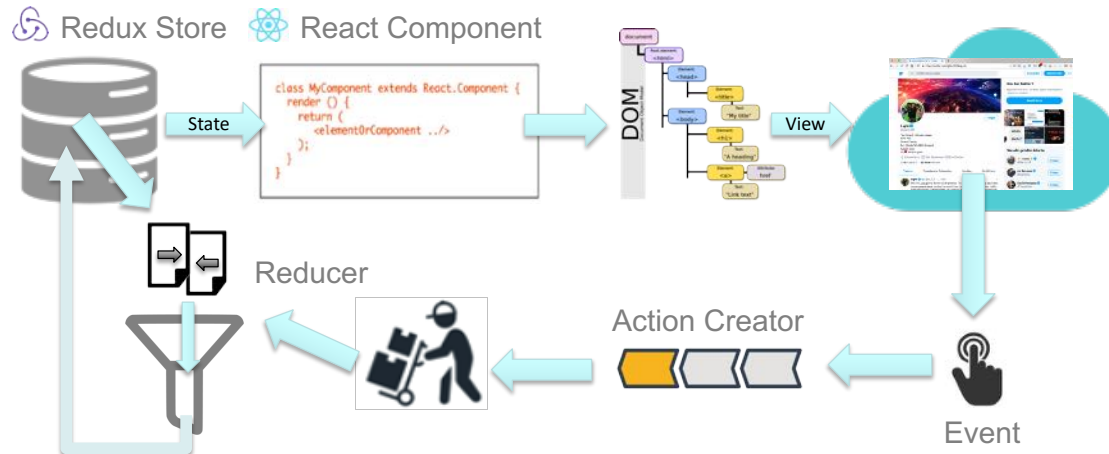
\$ npx create-react-app <appname> --template redux



Redux

- "Redux" ist ein "State-Handling-System".
- Redux stellt eine Wortkombination aus "Reduce" und "Flux" dar. Flux (<https://facebook.github.io/flux/>) ist eine Architekturidee, während Redux (<https://redux.js.org/>) eine Implementierung dieser Architektur ist.
(Dan Abramov von Facebook, legte das Konzept 2015 offen)
- Es stellt im Wesentlichen einen "Predictable-State-Container" zur Verfügung.
 - Ein Command löst einen "Event-Handler" aus, der dieses Event an den zentralen Store "dispatched" und ein "Reducer" steuert welche Änderungen am "State" erfolgen.
 - Redux besteht aus drei wesentlichen Elementen: "State", "Action" und "Reducer".
- Über den State wird eine Veränderung kontrolliert. Dabei ist der State als "read-only" zu betrachten und kann nur über "Actions" verändert werden!
- In der App ist jederzeit der State erkennbar und wie er sich ggf. verändert hat.
- Anders als bei React, in dem der State sich auf eine Component bezieht, stellt Redux einen "Application State" zentral für die gesamte Applikation dar!
- Über dieses Konzept wird eine App vorhersehbarer, strukturierter und leichter wartbar.

Redux / React Architektur



Single-Source-of Truth

Die "einzige Quelle der Wahrheit" bedeutet, dass es nur eine Möglichkeit gibt Daten innerhalb einer Benutzeroberfläche zu ändern. Dies kann nur durch eine "Action" innerhalb einer App ausgelöst werden. Der Zustand kann mittels eines "Reducers" und Vergleich des vorhandenen mit dem veränderten State verändert werden kann. Die React-Component reagiert eigenständig auf die Veränderung des zentralen "State" und ändert automatisch entsprechend den virtuellen und physischen DOM und damit die Benutzeroberfläche. Dieser Ablauf funktioniert nur in einer Richtung (in diesem Fall rechts herum), denn der "Redux-Store" ist die einzige Quelle der Wahrheit.

Redux - Provider

Index.js - Redux Provider

```
import { Provider } from 'react-redux';
import { createStore } from 'redux';
import counter from './reducers/index';

let store = createStore(counter);
...
<Provider store={store}>
  <div>
    <h1 className="App">React Redux Training</h1>
    <App />
    <TaskList />
  </div>
</Provider>,
...
```

- Die Redux-Component "Provider" wird in der Top-level Komponente abgelegt.
- Der mit "CreateStore" erzeugte Store wird als Property dem Provider mitgegeben, der diesen im React Kontext hält und seinen eingeschlossenen Child-Components (<App />, <TaskList /> etc.) zur Verfügung stellt.
- Diese eingeschlossenen Components können dann auf den Store zugreifen und ihn für den Zustand des "Action-Creators" nutzen.

Redux - Action Creator

./actions/index.js - Action Creator

```
export function incrementCounter() {  
  return {type: "INCREMENT"};  
}  
export function addTask(title) {  
  return {type: "TASK_ADD", title: title};  
}
```

- Ebenfalls wie die Idee von Redux einen zentralen State zur Verfügung zu stellen, wird dies auch auf die "Actions" übertragen, die ebenfalls zentral alle Aktionen für die gesamte Applikation zusammenführen.
- Eine Action ist dabei ein Objekt, welches mindestens aus einem Property mit dem Namen "type" besteht.
- Mittels dieses Properties wird der Typ der Aktion festgelegt.
- Zusätzlich können weitere Objekte eingefügt werden, die eine zusätzliche Verarbeitung erwirken sollen.
- Funktionen die ein Action-Objekt erzeugen, werden Action-Creator genannt (Bsp: incrementCounter(), addTask(title)).
- Wichtig: Der Action-Creator verändert nicht den Zustand der Anwendung, sondern definiert nur eine Aktion, die andere Teile der Anwendung (Reducers) dann ausführen müssen.

Redux - Reducer

./reducers/counter.js - Reducers

```
let initialCount = 0; //Default, Anfangszustand  
  
function counter(state = initialCount, action) {  
  if (action.type === "INCREMENT") {  
    return state + 1;  
  } else {  
    return state;  
  }  
}  
  
export default counter;
```

- "Reducer" haben die Aufgabe anhand einer "Action" einen bestehenden Zustand in einen neuen Zustand zu überführen.
- Im obigen Beispiel ist der Zustand des InitialCount zu verändern.
 - Der neue Zustand der Count-Variable ist zu inkrementieren.
- Ein Reducer nutzt dazu zwei Übergabe-Parameter:
 - 1. den derzeitigen Zustand sowie
 - 2. die Aktion die über den Action-Creator ausgelöst wurde.
- Der Reducer wird als Funktion implementiert und gibt den neuen Zustand zurück.

Redux - Actions - Reducer

- Eine App reagiert auf Benutzerinteraktion, in dem mit dem Action-Creator eine Aktion erzeugt wird und ggf. eine Zustandsänderung über einen Reducer eingeleitet wird.
- Eine Action erhält alle relevanten Informationen um den Application-State zu verändern.
- Diese fachlichen Actions wandern in die Funktion eines Action-Creators in dem bestimmt wird, wie sich der Zustand der Anwendung ändern soll.
- Der Action-Creator dient auch als Stelle, um asynchrone Aufgaben zu erledigen, da ggf. gewartet werden muss, um z.B. eine Serveranfrage zu beantworten.
- Mittels eindeutiger ID (Bsp.: "INCREMENT") und entsprechenden Funktionsparametern wird eine Action übergeben.
- In einer realen Anwendung kann es mehrere Action-, als auch Reducer-Funktionen und Dateien geben.
- Alle Zustände werden in einem zentralen strukturierten Store gehalten, in dem es für jeden Bereich und Teilzustand einen Reducer gibt.
- Mehrere Reducer können unter "combineReducer" zu einem Reducer zusammengefasst werden.

Redux - Reducer kombinieren

./reducers/index.js - combineReducer

```
import { combineReducers } from 'redux';  
  
import counter from './counter';  
import tasks from './tasks';  
  
let reduce = combineReducers({  
  counter: counter, //Incrementieren eines Counters  
  tasks: tasks      //Taskliste anzeigen und erweitern  
})
```

- Die "combineReducer" Funktion erwartet ein Objekt, das den Namen eines Teilzustands (counter, tasks) auf die entsprechende Reducer-Funktion (counter, tasks) abbildet.
- Der zentrale Zustand/State wird über Properes an die Components weitergereicht.
- Da jedoch eine Component nicht mit dem gesamten Zustand/State arbeitet, sondern nur mit einem Teilzustand, wird dieser mittels einer Map-Funktion "mapStateToProps()" festgelegt.

Redux - Mapping Funktion

./components/App.js - Component

```
...
<Card title="Counter">
  <h1>Ergebnis: {this.props.value}</h1>
  <button onClick={this.props.onIncrement}>Addiere</button>
</Card>

...

let mapStateToProps = function(state) {
  return {
    value: state.counter
  }
}
...
```

- Der Mapping-Funktion wird der komplette zentral gehaltene Zustand über den Parameter "state" übergeben.
- Innerhalb dieser Funktion wird der Teilzustand, der nur zuständig für diese Component ist (state.counter), hineingereicht.
- Der Name des Properties (Bsp.: this.props.value) steht dann der Component uneingeschränkt zur Verfügung.
- Umgekehrt soll eine Aktion mit ggf. veränderten Zuständen ebenfalls an den zentralen Store weitergegeben werden

Redux - Dispatch Funktion

./components/App.js - Component

```
import { incrementCounter } from '../actions/index';

...
<Card title="Counter">
  <h1>Ergebnis: {this.props.value}</h1>
  <button onClick={this.props.onIncrement}>Addiere</button>
</Card>

...
let mapDispatchToProps = () => {
  onIncrement: incrementCounter
}
...
```

- Da Aktionen/actions alleine nichts bewirken, müssen sie mittels eine Dispatch-Methode (mapDispatchToProps) weitergereicht werden, um eine Zustandsänderung erwirken zu können.
- In Redux ist dies die einzige Möglichkeit, den Zustand, respektive den Store zu ändern.
- Hierzu wird der Action-Creator an das Property der Component gebunden. Der gebundene Action-Creator kümmert sich dann auch um die Verteilung der Actions.

Redux - Integration mit connect

./components/App.js - Component

```
import { connect } from 'react-redux';
...
class App extends Component {
  ...
  let mapStateToProps = function(state) {
    return {
      value: state.counter
    }
  }
  let mapDispatchToProps = () => {
    onIncrement: incrementCounter
  }
  ...
  let AppContainer = connect(mapStateToProps, mapDispatchToProps)(App);
  export default AppContainer;
```

- Bisher wurde der Zustand und die Logik aus der Component in den zentralen Store und umgekehrt in die Properties verschoben.
- Nun müssen jedoch noch die Zustandsänderungen der Component mit dem Redux-Store verknüpft werden.
- Dazu stellt Redux die Bibliothek "connect" aus der Komponente "react-redux" zur Verfügung.
- Über connect werden "mapStateToProps" und "mapDispatchToProps" an Redux gebunden.
- Technisch betrachtet, erzeugt React eine neue Komponente (App), die sich als Wrapper (Higher-Order-Component) um die Komponente legt und ihr zusätzlich Fähigkeiten verleiht.

Questions

&



Answers

ANHANG

GITHUB

Datei Repository

Frage

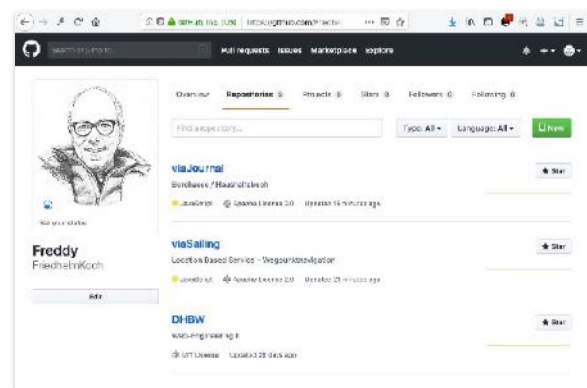
Sind Sie mit einem Instrument der Datei
Versionsverwaltung vertraut?

<https://pingo.coactum.de/462993> (start)



GitHub - öffentliches Dateirepository

- Entwickelt 2008 von Tom Preston-Werner, PJ Hyett und Chris Wanstrath.
- GitHub ist der Host für Ihre Git-Projekte.
- GitHub soll das Teilen von Codes vereinfachen und hilft, die unterschiedlichen Versionen der Codes von Software-Projekten zu organisieren.
- Die Plattform macht es möglich, dass man auch bei fremden Projekten mitentwickeln kann.

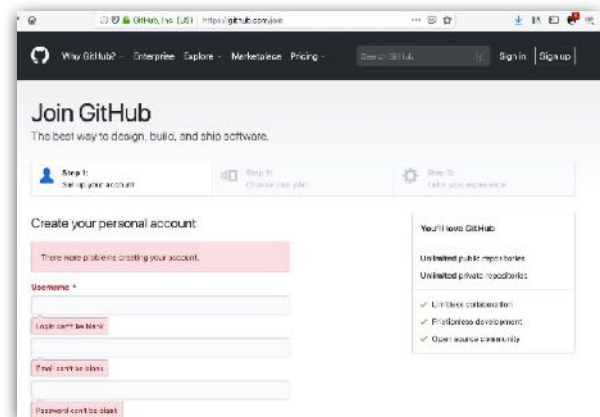


Bsp.: Mein GitHub-Repository:

<https://github.com/FriedhelmKoch?tab=repositories>

Übung - GitHub

- Erstellen Sie ein eigenes öffentliches GitHub Repository
 - <https://github.com>
 - <https://help.github.com/en/articles/creating-a-new-repository>
- Erstellen Sie dazu einen SSH-Key für den Zugriff auf GitHub (siehe nächstes Slide).



Übung - GitHub - SSH Key für GitHub erstellen

Feststellen, ob im Verzeichnis .ssh Keys existieren.

```
$ ls -al ~/.ssh
```

Generieren eines neuen SSH-Keys

```
$ ssh-keygen -t rsa -b 4096 -C 'mailname@domain.ext'
```

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/name/.ssh/id_rsa): newkey

Enter passphrase (empty for no passphrase): password

Enter same passphrase again: password

Your identification has been saved in newkey.

Your public key has been saved in newkey.pub.

The key fingerprint is:

SHA256:z0RNaG4iK1SGVLE9+45Zau7LM8rW2x0ONoZeUmfvFr4 mailname@domain.ext

The key's randomart image is:

```
+---[RSA 4096]-----+
| ..oo.  ..          |
|  . oo  oo          |
| o. oo.  .          |
| . . .+o           |
| . oSoo o          |
| . . B o ..         |
| . o @ ....         |
| ..O=% = oo         |
| .o=@=o o.E.        |
+---[SHA256]-----+
```

```
$ ls -al ~/.ssh
```

```
-rw----- 1 name staff 3434 20 Feb 19:38 newkey
```

```
-rw-r--r-- 1 name staff 749 20 Feb 19:38 newkey.pub
```

Key speichern (z.B. in MacOS Keychain)

```
$ ssh-add -K ~/.ssh/id_rsa
```


GIT

Versionsverwaltung

Datensicherung vs. Versionierung

Es gibt drei Arten um Daten zu sichern:

- **Archivierung**
- **Backup**
- **Versionierung**



Archiv: 1 x Woche

t(1) = 1.1.01, 12:00 Uhr

2 x Backup/Tag (12:00 und 18:00 Uhr)

t(1) = 1.1.01, 12:00 Uhr

t(2) = 1.1.01, 18:00 Uhr

t(3) = 2.1.01, 12:00 Uhr

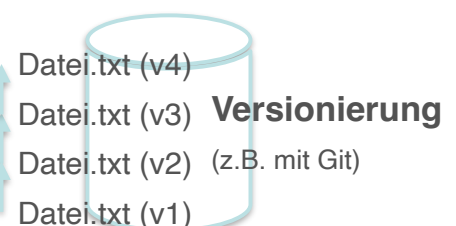
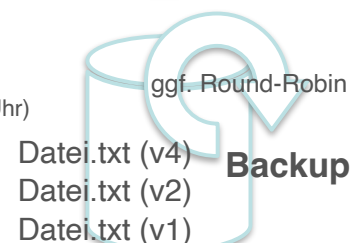
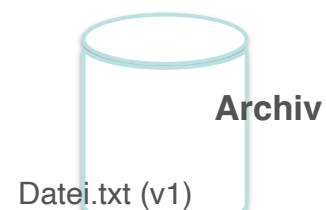
Bei jeder Änderung "commit"

commit t(1) = 1.1.01, 11:50 Uhr

commit t(2) = 1.1.01, 12:30 Uhr

commit t(3) = 1.1.01, 18:30 Uhr

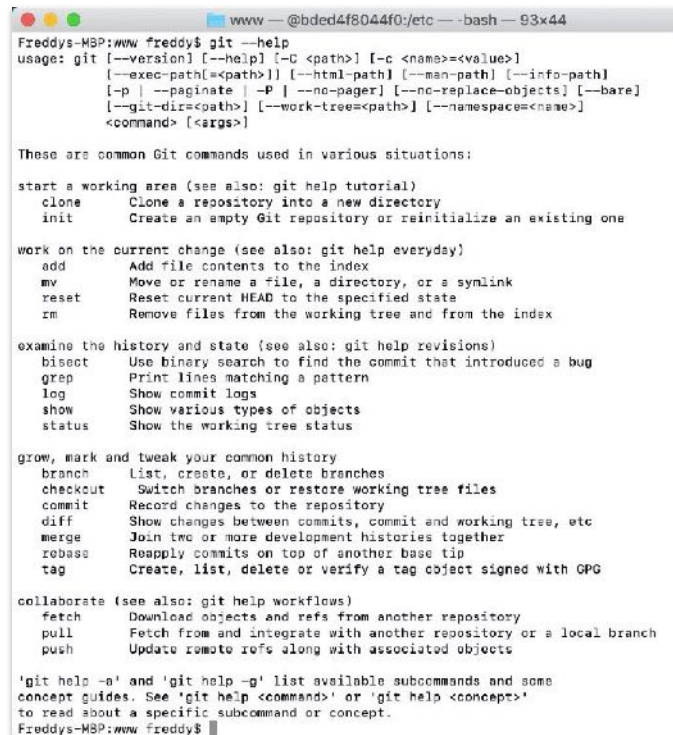
commit t(4) = 2.1.01, 10:30 Uhr



Git - Versionsverwaltung

- Git ist nicht GitHub.
- Git wurde 2005 von Linus Torvald entwickelt, um den Programmiercode für Linux zu verwalten.
- Es ist ein System zur Verwaltung von Dateien, die einer ständigen Änderung unterworfen sind.
- Git ist eine dezentrale Versionsverwaltung.

Git dient nicht nur der Softwareentwicklung sondern auch zur Versionsverwaltung von Studienarbeiten oder der Bachelor/Master-Thesis in Word, Pages und Co.



```

Freddys-MBP:www freddys$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone Clone a repository into a new directory
  init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add Add file contents to the index
  mv Move or rename a file, a directory, or a symlink
  reset Reset current HEAD to the specified state
  rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect Use binary search to find the commit that introduced a bug
  grep Print lines matching a pattern
  log Show commit logs
  show Show various types of objects
  status Show the working tree status

grow, mark and tweak your common history
  branch List, create, or delete branches
  checkout Switch branches or restore working tree files
  commit Record changes to the repository
  diff Show changes between commits, commit and working tree, etc
  merge Join two or more development histories together
  rebase Reapply commits on top of another base tip
  tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch Download objects and refs from another repository
  pull Fetch from and integrate with another repository or a local branch
  push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
Freddys-MBP:www freddys$
  
```

Git - Wie Wo Wer

Git - Dokumentation

- <https://services.github.com/on-demand/downloads/de/github-git-cheat-sheet/> (deutsch)
- <https://git-scm.com/book/de/v1> (Online-Buch: Pro Git, deutsch)
- <https://riptutorial.com/de/ebook/git> (eBook/pdf, Learning Git, deutsch)
- <https://www.amazon.de/Daily-Git-kompetenter-Kollege-erkl%C3%A4ren/dp/1500432571> (Daily Git: Wie ein kompetenter Kollege Ihnen Git erklären würde)

Git installieren (Linux, Windows, Mac)

- <https://git-scm.com/downloads>

GitHub persönlichen Account anlegen

- <https://github.com>

Git - Directories und Repositories

Working Directory (Arbeitsverzeichnis)

- Hier werden ggf. die Source-Dateien verändert. Die Änderungen werden dann für ein „commit“ in der Staging Area per „add“ zusammengestellt.

Staging Area (Index)

- Mittels „commit“ werden die geänderten Dateien in das „Local Repository“ transportiert (Ein commit basiert auf einer eindeutigen SHA-1-Hash-ID).

Local Repository (lokaler Datenspeicher)

- Über einen „push“ werden das lokale und remote Repository synchronisiert.

Remote Repository (öffentlicher Datenspeicher)

- GitHub

Git - Clones und Branches

Clone

- Ein Projekt wird in einem Repository abgelegt.
- Mehrere Projekte können auf einem Server zur Verfügung gestellt werden (dezentral).
- Um auf ein veröffentlichten Projekt zuzugreifen, muss eine Kopie (Clone) kopiert werden.
- Änderungen werden an diesem Clone vorgenommen.
- Danach werden sie dann mit dem Clone abgeglichen/synchronisiert.

Branch (<https://git-scm.com/book/de/v1/Git-Branching-Was-ist-ein-Branch%3F>)

- Ein Branch ist ein Zeiger auf einen Commit und stellt einen parallelen Zweig eines Hauptzweiges (Master) dar.
- Branches sind sinnvoll, wenn eine "eingefrorene" Version ggf. mit Patches etc. weiter gepflegt werden muss.
- Über „git checkout <branchname>“ und „git checkout master“ kann zwischen den Zweigen gewechselt werden.
- Mittels „git merge“ können Branches auf einen Master wieder zusammengeführt werden.

Übung - Git

- Verschaffen Sie sich einen kurzen Überblick über die angegebene Dokumentation zu Git.
- Installieren Sie Git.
- Erstellen Sie eine HTML-Datei „HelloWorld.html“.
- Legen Sie im „Hello World“-Verzeichnis ein Git Repository an (<https://git-scm.com/book/de/v1/Git-Grundlagen-Ein-Git-Repository-anlegen>)
- Verändern Sie die HTML-Datei „HelloWorld.html“ mit einer geänderten Ausgabe „Hello DHBW-World“.
- Verfahren Sie beim Upload wie auf den nachfolgenden Slides beschrieben und erstellen Sie „Hello World“ auf Ihrem öffentlichen GitHub Repository.
- Fragen Sie den Status ab. Kontrollieren Sie den Upload im Browser auf GitHub.

Übung - Git - Upload des lokalen Repositories nach GitHub

Das Git-Repository im entsprechenden lokalen Verzeichnis aktivieren.

```
$ git init
```

Das lokale Repository mit GitHub verbinden.

```
$ git remote add origin git@github.com:GitName/Repo.git
```

Die Repositories abgleichen.

```
$ git pull origin master
```

Dateien und Unterverzeichnisse in Staging Area hochladen

```
$ git add . //aktuelles Verzeichnis mit allen Unterverzeichnissen
```

```
$ git add index.html //eine einzelne Datei hochladen
```

Die so hochgeladenen Daten/Verzeichnisse im sog. Stash committen.

(ACHTUNG: Es werden nur Dateien committet, die auch per „add“ in die Staging Area geladen wurden. Gibt es nach dem „add“ noch Änderungen, werden die nicht berücksichtigt!

```
$ git commit -m „Beschreibung“
```

Aktualisieren des Repositories auf GitHub (hier als Master-Branch).

```
$ git push --set-upstream origin master
```

Status abfragen.

```
$ git status
```

Übung - Git - Upload / Download cont'd

Wenn ein Repository von mehreren Personen genutzt wird, sollte das Repo vorher mit dem lokalen Repo aktualisiert werden.

```
$ git pull
```

Weitere lokale Änderungen aktualisieren.

```
$ git add .
```

```
$ git commit -m "Nachricht"
```

```
$ git push
```

Ein entferntes Repository downloaden

```
$ git clone https://github.com/GitName/Repo.git
```

Dateien und Verzeichnisse die nicht auf GitHub hochgeladen werden sollen, werden in der hidden Datei „.gitignore“ definiert:

```
$ echo -e "geheim.html \nconfig/passwd \nininclude/verz" > .gitignore
geheim.html
config/passwd
include/verz
```

Übung - Git - Standard Workflow

Aktuellen master-Branch vom Server downloaden (vorher mit git status sicherstellen, dass man sich auf dem master-Branch befindet. Wenn nicht: git checkout master)

```
$ git pull
```

Einen neuen Branch für das kommende Feature anlegen

```
$ git checkout -b meinfeature master
```

[Änderungen am Code durchführen]

Geänderte und neue Dateien stagen

```
$ git add .
```

Änderungen committen

```
$ git commit -m „Changes xyz“
```

Jetzt den Branch auf den Server laden, wenn gewünscht

```
$ git push -u origin meinfeature
```

... oder direkt in dem master wechseln

```
$ git checkout master
```

(nochmals aktuellen Code ziehen – zur Sicherheit)

```
$ git pull
```

Eigenen Code mit master zusammenführen

```
$ git merge meinfeature
```

Nicht mehr benötigten Branch löschen

```
$ git branch -d meinfeature
```

Aktualisierten master auf den Server pushen

```
$ git push
```

Done!

Questions

&



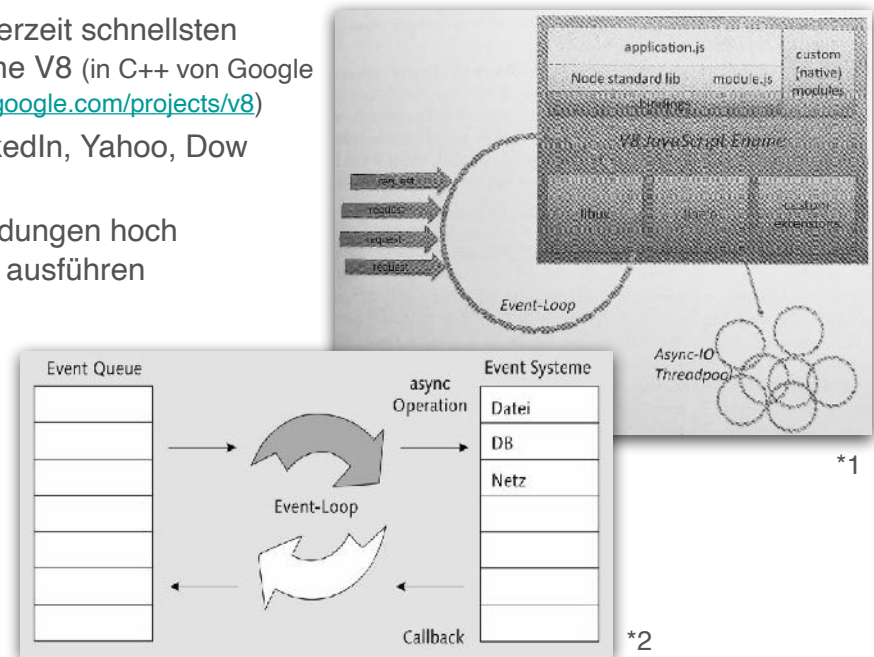
Answers

NODE.JS

Grundlagen - Package Manager, Web-Server und mehr...

Node.js - Funktionalität

- Node.js basiert auf dem derzeit schnellsten JavaScript Compiler Engine V8 (in C++ von Google entwickelt, <https://opensource.google.com/projects/v8>)
- Entwicklungen: eBay, LinkedIn, Yahoo, Dow Jones
- Node.js lässt Web-Anwendungen hoch skalierbar auf dem Server ausführen
- Hohe Performance bei hoher asynchroner I/O Last (libeio-Framework)
- Event-Loops über libuv-Framework



*1: Robert Prediger, Ralph Winzinger (2015), S. 299, Node.js, Professionell hochperformante Software entwickeln, München: Carl Hanser Verlag

*2: Springer Sebastian (2016), S. 49, Node.js: Das umfassende Handbuch. Serverseitige Webapplikationen mit JavaScript entwickeln, 2. aktualisierte Auflage, Bonn: Rheinwerk Verlag GmbH

© 2022, DHBW - Fachbereich Informatik - Friedhelm Koch

195

Node.js - Installation & Dokumentation

Packages

- <https://nodejs.org/de/download/>

Dokumentation

- <https://nodejs.org/api/>

OpenJS Foundation

- <https://openjsf.org/>

Gebündelt mit: „npm“ - Node Package Manager

- # **npm init** (initiiert package.json und enthält wichtige Daten zu einem installieren Modul)
- <https://docs.npmjs.com/packages-and-modules> (Packages & Modules)

Node.js - Laufzeitumgebung

Node.js eine Umgebung, mit der man u.a. JavaScript ohne einen Browser ausführen kann. Dabei kommt die V8-JavaScript-Engine zum Einsatz, die von Google entwickelt wurde und eine Laufzeitumgebung zur Verfügung stellt, in der JavaScript auf dem Server ausgeführt wird.

Beispiel:

```
$ node  
> var user = "Peter"  
undefined  
> 'hallo ' + user + '!!'  
'hallo Peter!!'  
>
```

oder:

```
> console.log(3+4)  
7  
undefined  
> .exit  
$
```

Questions

&



Answers

LITERATURVERZEICHNIS

Literaturverzeichnis

- **Ackermann, Philip (2019):** Node.js, Rezepte und Lösungen, 1. Auflage, Bonn: Rheinwerk Verlag
- **Cherny, Boris (2019):** Programming TypeScript: Making Your JavaScript Applications Safe, London: O'Reilly Media, Inc.
- **Deru, Matthieu; Ndiaye, Alassane (2019):** Deep Learning mit Tensorflow, Keras und Tensorflow.js, 1. korrigierter Nachdruck, Bonn: Rheinwerk Verlag
- **Drähter, Rolf; Koschek, Holger; Salbung, Carsten (2013):** Srum, kurz & bündig, Heidelberg: O'Reilly Verlag GmbH & Co. KG c/o dpunkt verlag GmbH
- **Haverbeke, Marijn (2020):** JavaScript, Richtig gut programmieren lernen - Von der ersten Codezeile bis zum eigenen Projekt, 1. Auflage, Heidelberg: dpunkt verlag GmbH
- **Höller, Christopher (2019):** Angular, Das umfassende Handbuch, 2. aktualisierte und überarbeitete Auflage, Bonn: Rheinwerk Verlag
- **Krause, Jörg (2019):** Einführung in TypeScript, Sprachgrundlagen und Techniken, 4. Auflage, Leanpub-Buch, texxtoor Verlag (ISBN: 978-198059737-7)
- **Liebel, Christian (2019):** Progressive Web Apps, Das Praxisbuch, 1. Auflage, Bonn: Rheinwerk Verlag
- **Lim, Greg (2019):** Beginning Node.js, Express & MongoDB Development, Printed eBook, Wroclaw: Amazon Fulfillment
- **Münz, Stefan; Gull, Clemens (2014):** HTML5 Handbuch, 10. Auflage, München: Franzis Verlag
- **Öggl, Bernd; Koller, Michael (2018):** Docker, Das Praxisbuch für Entwickler und DevOps-Teams, 1. Auflage, Bonn: Rheinwerk Verlag
- **Prediger, Robert; Winzinger, Ralph (2015),** Node.js, Professionell hochperformante Software entwickeln, München: Carl Hanser Verlag
- **Rashid, Tariq (2017):** Neuronale Netze selbst programmieren, 1. Auflage, Heidelberg: O'Reilly Verlag GmbH & Co. KG c/o dpunkt verlag GmbH
- **Rubin, Kenneth (2014):** Essential Scrum, Umfassendes Scrum Wissen aus der Praxis, München/Heidelberg: mitp - Verlagsgruppe Hüthig Fehle Rehm GmbH
- **Springer, Sebastian (2016):** Node.js, Das umfassende Handbuch. Serverseitige Webapplikationen mit JavaScript entwickeln, 2. aktualisierte Auflage, Bonn: Rheinwerk Verlag GmbH
- **Springer, Sebastian (2020):**, React, Das umfassende Handbuch, 1. Auflage, Bonn: Rheinwerk Verlag
- **Woiwode, Gregor; Malcher, Ferdinand; Koppenhagen, Danny; Hoppe, Johannes (2018):** Angular, Grundlagen, fortgeschrittene Technologien und Best Practices mit TypeScript, 1. korrigierter Nachdruck, Heidelberg: dpunkt verlag GmbH
- **Zeigermann, Oliver; Hartmann, Nils (2020):** Grundlagen, fortgeschrittene Techniken und Praxistipps - mit TypeScript und Redux, 2. überarbeitete und erweiterte Auflage, Heidelberg: dpunkt verlag GmbH
- **o. V. (2017) [a]:** React, Erste Schritte mit React, <https://riptutorial.com/de/reactjs> (letzter Zugriff: 12.07.2019)