SINGLE PAGE APPLICATION (SPA)

LERNZIELE

- Was ist eine Single Page Application (SPA)?
- Wie funktioniert einer SPA (in der Theorie)?
- Was sind Vorteile und Nachteile einer SPA?

“A single-page application is exactly what its name implies: a JavaScript-driven web application that requires only a single page load.”

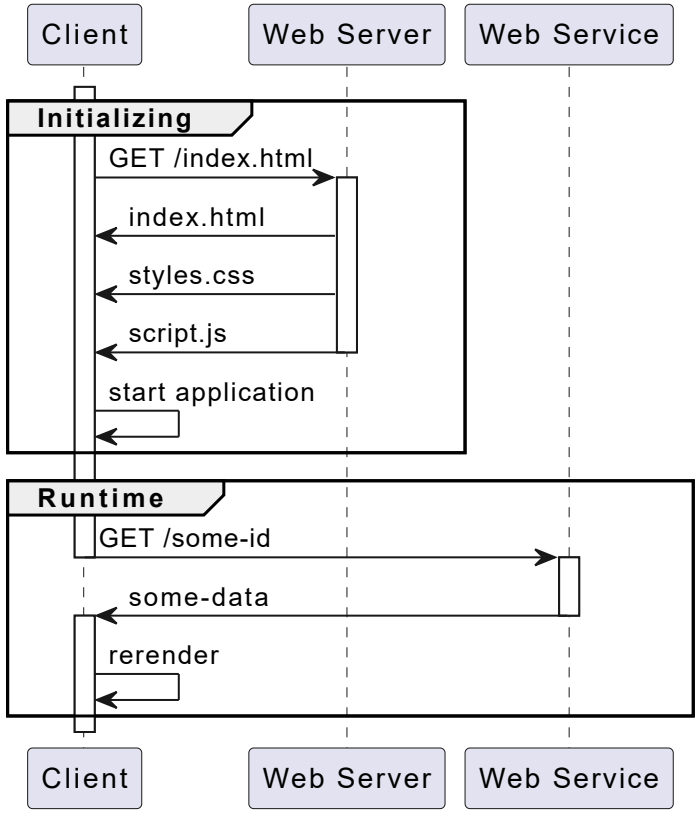
JavaScript - The Definitive Guide

5th ed., O'Reilly, Sebastopol, CA, 2006

Speaker notes

- Wir haben uns bereits Ajax beschäfitgt.
- Ajax ist die Grundlage für SPA's.
- SPA's sind quasi die stärkste Auspräung von Ajax.
- Anders als bei einer Multi Page Anwendung haben wir nur einen Seitenaufruf und laden danach nur noch Daten, aber zu den Details kommen wir gleich.

SINGLE PAGE APPLICATION KONZEPT



Speaker notes

- Terminologie
  - Client -> PC, bzw. Browser des Nutzers.
  - Web Server -> liefert HTML, CSS, JavaScript, Bilder, etc. aus.
  - Web Service -> kümmert sich um die Verwaltung von dynamischen Daten der Anwendung (User, Warenkorb, etc.).
- Eine Single Page Application hat grundlegend zwei Phasen.
  - Phase 1 ist die Initialisierung.
    - Der Client startet eine Anfrage an den Web Server, um die Frontend Application abzurufen.
    - Anschließend liefert der Web Server die statischen Daten für die Application aus.
    - Dies passiert oftmals gleichzeitig in mehreren Schritten. (HTTP 2 Multiplexing)
    - Nachdem/während die statischen Daten geladen werden, startet der Browser die Application.
  - Phase 2 ist die Laufzeit.
    - Zur Laufzeit läd der Client nur noch dynamische Daten vom Web Service.
    - Sobald er dynamische Daten geladen hat, löst er ein rerender aus und die Seite wird im Browser neu gezeichnet, nicht neu geladen!

WIESO GIBT ES SPA'S?

Findet ihr Gründe dafür, warum man sowas machen sollte?

Speaker notes

- Auf alle der hier genannten Punkte gehen wir jetzt noch mal genauer ein.

WIESO GIBT ES SPA'S?

Findet ihr Gründe dafür, warum man sowas machen sollte?

- Reduktion der übertragenen Daten
- Bessere User Experience
- Weniger Serverressourcen
- Session Clientseitig (Server ist Stateless)
- Hybride Anwendung auch mobile einsetzbar

WIESO GIBT ES SPA'S?

- *Reduktion der übertragenen Daten*
- Bessere User Experience
- Weniger Serverressourcen
- Session Clientseitig (Server ist Stateless)
- Hybride Anwendung auch mobile einsetzbar

REDUKTION DER ÜBERTRAGENEN DATEN

```
 1 <html>
 2 <head>
 3   <link rel="stylesheet" href="style.css">
 4   <script>{javascript}</script>
 5 </head>
 6 <body>
 7   <div>
 8     // some data
 9   </div>
10 </body>
11 </html>
```

Speaker notes

- Könnt ihr an diesem Beispiel einer HTML Seite erläutern welche Daten nur einmal ausgeliefert werden müssen?
- Diese Daten würden bei einer Multi Page Application bei jedem Seitenaufruf erneut mitgeschickt werden.

STYLESHEETS

```
 1 <html>
 2 <head>
 3   <link rel="stylesheet" href="style.css">
 4   <script>{javascript}</script>
 5 </head>
 6 <body>
 7   <div>
 8     // some data
 9   </div>
10 </body>
11 </html>
```

Speaker notes

- Wie ist das z.B. mit Stylesheets?
- Müssten sie wirklich immer neu augeliefert werden?
- Normalerweise wollen wir, dass Buttons, Inputfelder, etc. überall in unserer Application gleich aussehen.

STYLESHEETS

- enthalten häufig ähnliche Informationen
- könnten einmalig ausgeliefert werden

```
<style>
label {
 font-size: 12pt;
 color: blue;
}

input {
 font-size: 10pt;
 color: green;
 height: 10px;
 width: 20px;
}
</style>
```

JAVASCRIPT

```
 1 <html>
 2 <head>
 3   <link rel="stylesheet" href="style.css">
 4   <script>{javascript}</script>
 5 </head>
 6 <body>
 7   <div>
 8     // some data
 9   </div>
10 </body>
11 </html>
```

Speaker notes

- Was ist mit dem Javascript Code?
- Der Javascript Code einer Multi Page Application wird sich wahrscheinlich auf Dropdowns oder ähnliches Beschränken.
- Schließlich passiert das eigentliche Rendering der Seite bereits im Backend.
- Aber auch Frontend funktionalität wie Dropdowns oder Animationen sollten auf jeder Seite gleich funktionieren.

JAVASCRIPT

- ebenfalls repetitiv
- auf mehreren HTML Seiten braucht es gleiche Funktionalität

```
<script>
function openDropdown() {
  // do it
}

function doSomeFancyAnimation() {
  // do it
}
</script>
```

HTML STRUKTUR

```
<html>
// head
<body>
  <header></header>
  <nav></nav>
  <div>
    // some data
  </div>
  <footer></footer>
</body>
</html>
```

Speaker notes

- Was ist mit der restlichen HTML Struktur?
- Können wir noch mehr einspaaren?
- Header, Navigation und Footer sind meist auch überall gleich.
- Brauchen teilweise nicht mal neu gerendert werden.

HTML STRUKTUR

- dynamischer Anteil der Seite beschränkt sich auf Informationen

```
<html>
// head
<body>
  <header></header>
  <nav></nav>
  <div>
    // some data
  </div>
  <footer></footer>
</body>
</html>
```

Speaker notes

- Am Ende beschränkt sich der dynamische Teil auf die "business" Daten der Application.
- Das sind dann Nutzerdaten, Warenkörbe, etc.

WIESO GIBT ES SPA'S?

- Reduktion der übertragenen Daten
- ***Bessere User Experience***
- Weniger Serverressourcen
- Session Clientseitig (Server ist Stateless)
- Hybride Anwendung auch mobile einsetzbar

BESSERE USER EXPERIENCE

- kürzere Response Time
- weniger BE Request notwendig
- Seite ist während eines BE Requests benutzbar
- asynchrones Nachladen der Daten

Speaker notes

- Kürzere Response Time
  - durch weniger Daten die übertragen werden müssen
- weniger BE Requests notwendig
  - Fehlermeldungen etc. können bereits ohne BE Requests angezeigt werden
  - Auch Geschäftslogik kann direkt im Frontend ausgeführt werden
- Seite ist während eines BE Requests benutzbar
  - Durch Loadingspinner etc. bekommt der Nutzer ein direktes Feedback auf seine Action
  - Nutzer fordert Daten an und kann sich dann mit etwas anderem beschäftigen, bis die Daten geladen sind.
  - In der Realität wird die Seite meistens nicht benutzt während eines BE Requests
- asynchrones Nachladen der Daten
  - der Nutzer kann bereits mit ersten Daten interagieren, während andere noch geladen werden.

WIESO GIBT ES SPA'S?

- Reduktion der übertragenen Daten
- Bessere User Experience
- ***Weniger Serverressourcen***
- Session Clientseitig (Server ist Stateless)
- Hybride Anwendung auch mobile einsetzbar

WENIGER SERVERRESSOURCEN

- Rendering läuft auf dem Client
- Geschäftslogik kann auf dem Client laufen
  - weniger BE Requests notwendig
- Service kümmert sich nur um die Daten
- nicht um das Rendering

WIESO GIBT ES SPA'S?

- Reduktion der übertragenen Daten
- Bessere User Experience
- Weniger Serverressourcen
- ***Session Clientseitig (Service ist Stateless)***
- Hybride Anwendung auch mobile einsetzbar

SESSION CLIENTSEITIG (SERVICE IST STATELESS)

- Service kann Stateless sein
- Skalierbarkeit des Service
- Loadbalancing ist einfacher

Speaker notes

- Skalierbarkeit bedeutet, dass wir mehrere Instanzen eines Services laufen lassen können.
- Bei hoher Last können die Zugriffe auf mehrere Instanzen verteilen.
- Loadbalancing beschreibt den Prozess der Lastverteilung.
- Wenn ein Nutzer immer dem gleichen Service zugeordnet werden muss, macht dies das Loadbalancing schwieriger.
- Wir gehen auf diese Punkte noch mal genauer in der Backend Vorlesung ein.

WIESO GIBT ES SPA'S?

- Reduktion der übertragenen Daten
- Bessere User Experience
- Weniger Serverressourcen
- Session Clientseitig (Service ist Stateless)
- ***Hybride Anwendung auch mobile einsetzbar***

HYBRIDE ANWENDUNG AUCH MOBILE EINSETZBAR

- SPA's sind ähnlich wie App's
- Doppelentwicklungen können gespart werden
- React Native oder Flutter zeigen wie es geht
- ist nicht immer eine gute Entscheidung!

Speaker notes

- Statische Daten werden bei einer App, wie bei einer SPA initial ausgelierert. Danach werden nur noch "business" Daten ausgetauscht.
- Mit React Nativ oder Flutter kann man Apps schreiben, die als Webseite und gleichzeitig als App genutzt werden können.
- Dies kann eine Menge Zeit bei der Entwicklung sparen.
- Allerdings kann dies auch dazu führen, dass App und Web nicht entsprechend der Client Plattform nutzerfreundlich ist.
- Das heißt eine App bedient sich teilweise anders als ein Webbrowser.
- Allein Apps zwischen Apple und Android unterscheiden sich, da die Betriebssysteme und Geräte anders funktionieren.

NACHTEILE EINER SPA

Was könnten Nachteile einer SPA sein?

Speaker notes

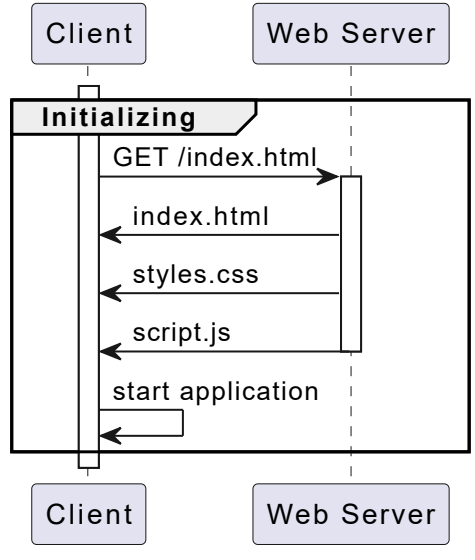
- Jetzt habe ich euch die Vorteile von SPA's gezeigt.
- Das heißt nicht, dass eine SPA immer die beste Wahl ist.
- Multi Page Anwendungen haben durchaus ihre Berechtigung.

NACHTEILE EINER SPA

Was könnten Nachteile einer SPA sein?

- initiale Response ist groß
- Client ist nicht Vertrauenswürdig
- duplizierter Code
- höherer Entwicklungsaufwand

INITIALE RESPONSE IST GROSS



Speaker notes

- Wir erinnern uns an das SPA Laufzeitdiagramm.
- Zu Anfang müssen erstmal alle Daten geladen werden.

NACHTEILE EINER SPA

- initiale Response ist groß
- ***Client ist nicht Vertrauenswürdig***
- duplizierter Code
- höherer Entwicklungsaufwand

CLIENT IST NICHT VERTRAUENSWÜRDIG

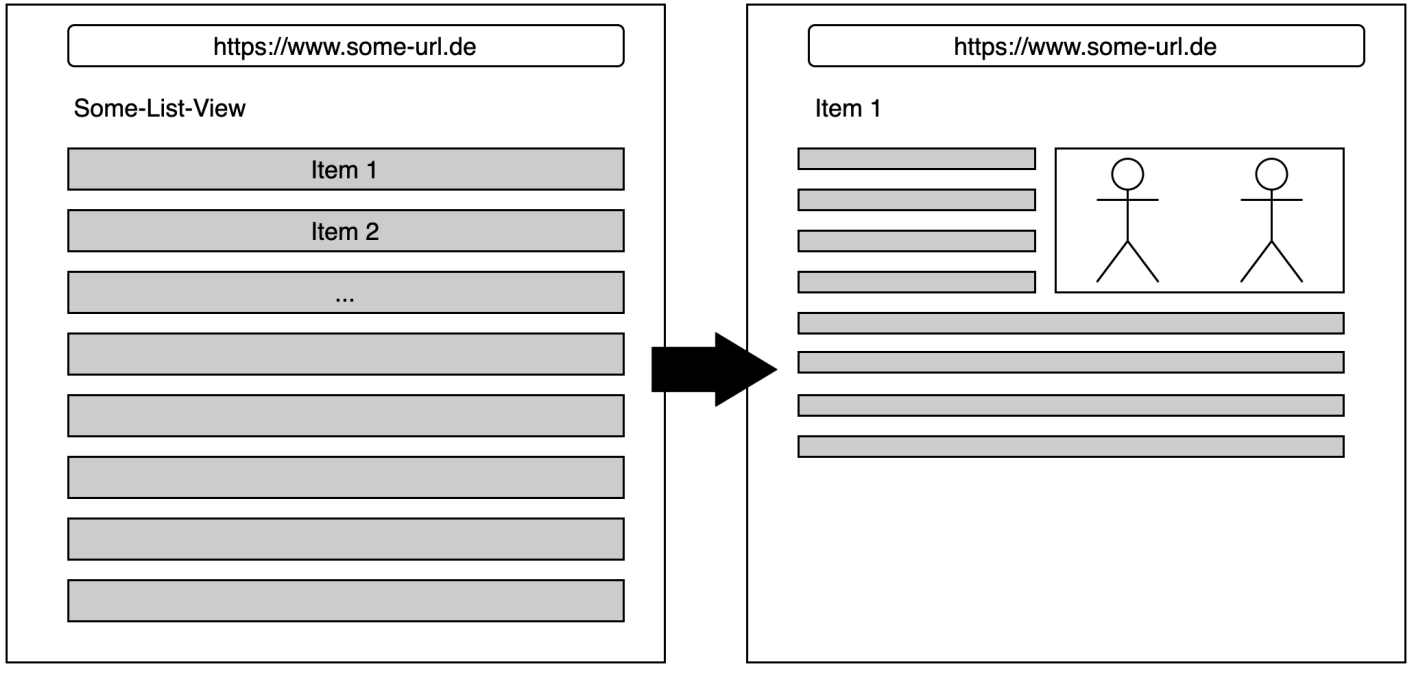
- JavaScript Code auf dem Client kann manipuliert werden
- erneute Validierung im BE notwendig
- Validierungen sind meist duplizierter Code
- dadurch entsteht mehr Entwicklungsaufwand
- hierfür gibt es Abhilfe:
  - Multiplattform Libraries

Speaker notes

- Ein versierter Nutzer kann den JavaScript Code in seinem Browser verändern.
  - Wir sprechen hier noch nicht mal von XSS.
- Daten die im BE gespeichert werden, müssen daher noch mal validiert werden.
- Validierungen, aber auch andere Geschäftslogik sind häufig dupliziert.
  - Das kann gewünscht sein. Vielleicht möchte man Frontend und Backend voneinander entkoppeln.
  - Andererseits kann man Code auch übers BE und Frontend sharen.
  - Multiplattform Libraries können hier helfen.

ROUTING?

- ist theoretisch nicht notwendig
- Anwendung macht einfach ein rerender

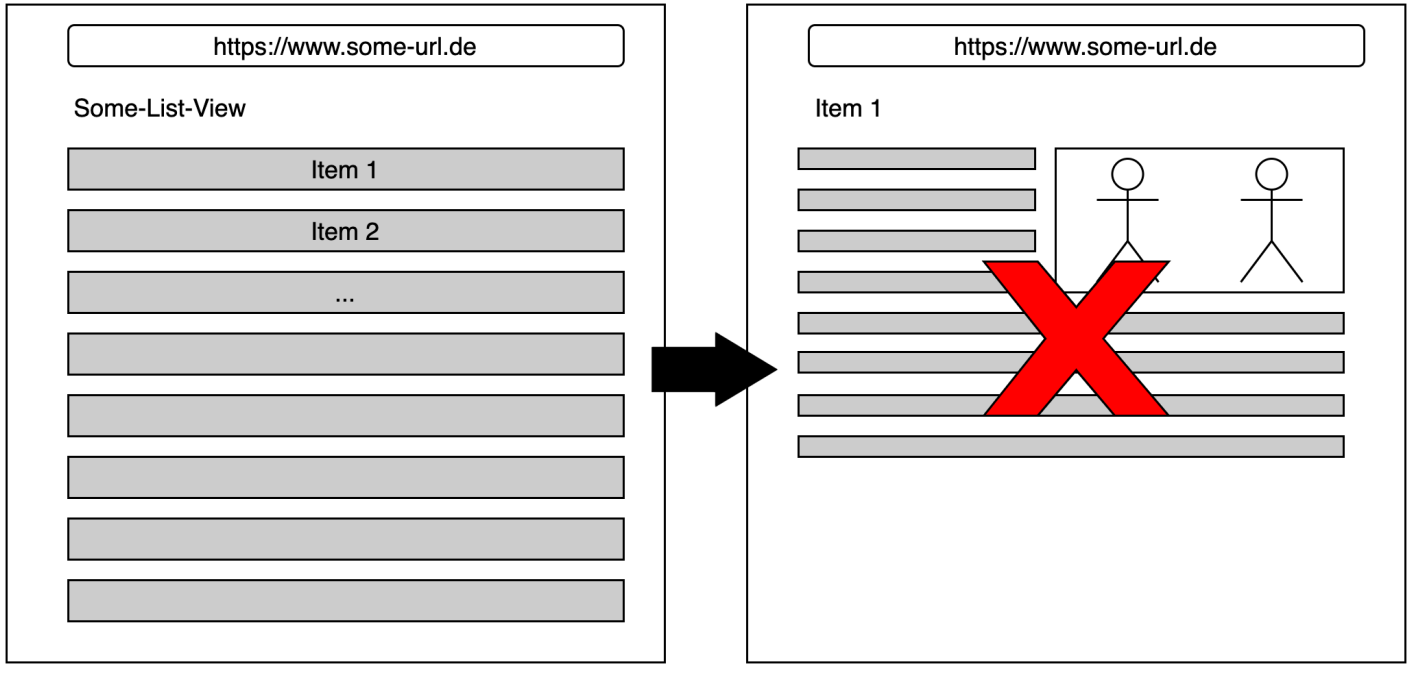


Speaker notes

- Im Gegensatz zu einer Multi Page Application machen wir nun nach dem ersten Seitenaufruf keine weiteren Aufrufe mehr.
- Theoretisch könnten wir uns Routing eigentlich sparen.
- Dynamische Daten vom Backend lösen einfach ein rerender des HTML Contents aus.

ALSO KEIN ROUTING?

- URL bleibt über die Laufzeit gleich
- teilen eines Links einer bestimmten Ressource?

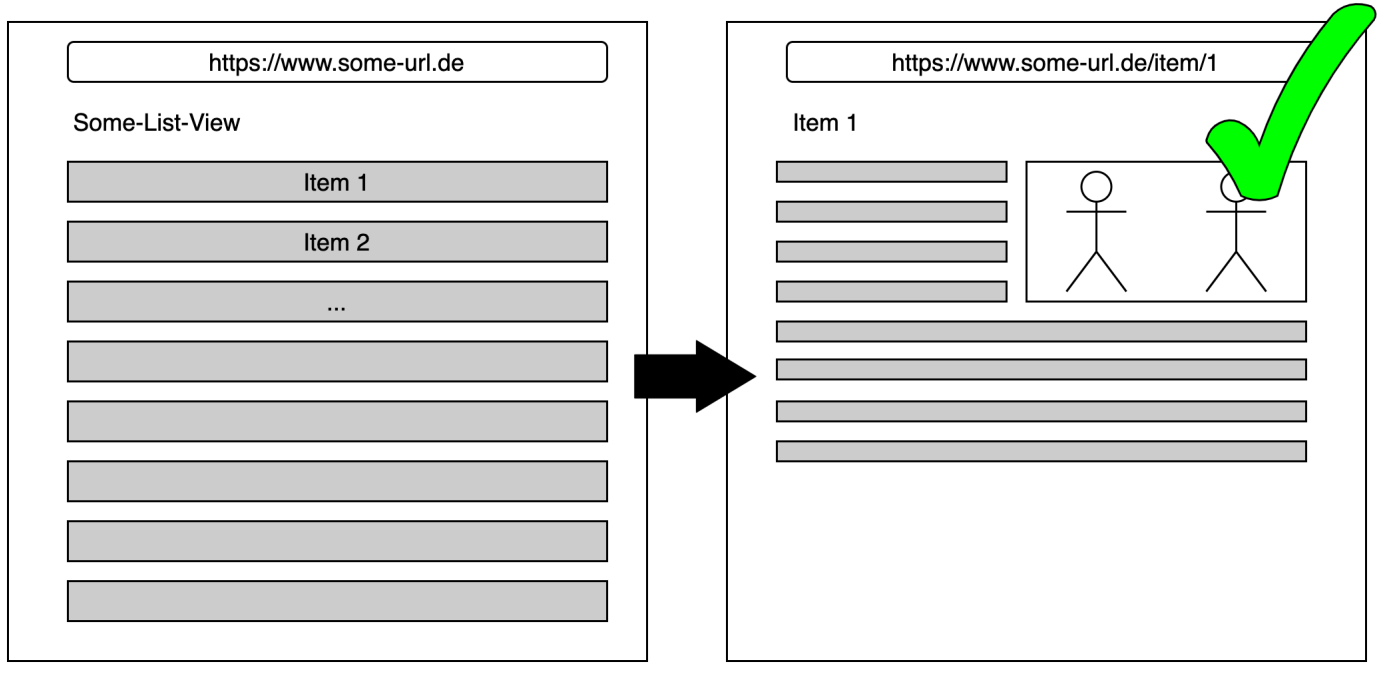


Speaker notes

- Oftmals ist es durchaus Sinnvoll Routing zu unterstützen.
- Link teilen, Lesezeichen hinzufügen, etc.

ROUTING

- wir brauchen Routing in SPA's doch!
- es passiert ein pseudo Routing
- SPA Frameworks liefern Routing mit oder es gibt Libraries
- dazu später mehr ...



Speaker notes

- Wenn ein rerender des HTML Contents aufgerufen wird, kann die URL ausgetausch werden.
- Dies macht natürlich nicht immer Sinn. Wenn sich nur ein Overlay öffnet, brauchen wir keine URL Änderung (zumindest meistens).
- Wir teilen mit der URL Änderung die "eine" Seite wieder in verschiedene Bereiche ein.
- Das macht es für den Nutzer, aber auch den Entwickler wieder einfacher und stärkt die Struktur.

LERNZIELE

- Was ist eine Single Page Application (SPA)?
- Wie funktioniert einer SPA (in der Theorie)?
- Was sind Vorteile und Nachteile einer SPA?

Speaker notes

- SPA ist eine Application, bei der nur ein Seitenaufruf erfolgt und danach nur noch Daten per Ajax geladen werden.
- Initialisierungsphase -> Laufzeitphase, auch Routing ist wichtig.
- SPA's sind kein Allerheilsmittel, es gibt viele Vorteile, aber auch Nachteile. Multi Page Applications haben ihre Daseinsberechtigung.

DAS NÄCHSTE MAL ...

- Backend für eine SPA
- Wie baut man einen Web Service?

Speaker notes

- Wir haben bereits gesehen, wie man mit plain Javascript und Ajax daten von einem bestehenden Backend abfragt.
- Wir haben ebenfalls eine Multi Page Application gebaut und damit auch schon ein Server, kombiniert mit Client, aufgesetzt.
- Eine getrennte Form von Server und Client, wie bei einer SPA bauen wir das nächste Mal.