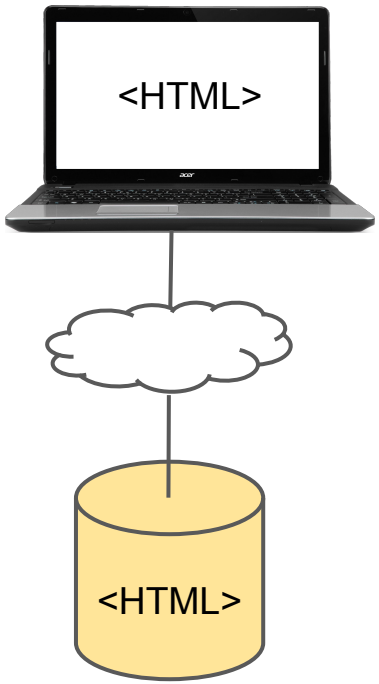
WebApps: Das WWW als App-Plattform

UX - User Experience
DX - Developer Experience

Das WWW war als Hypertext gedacht ...

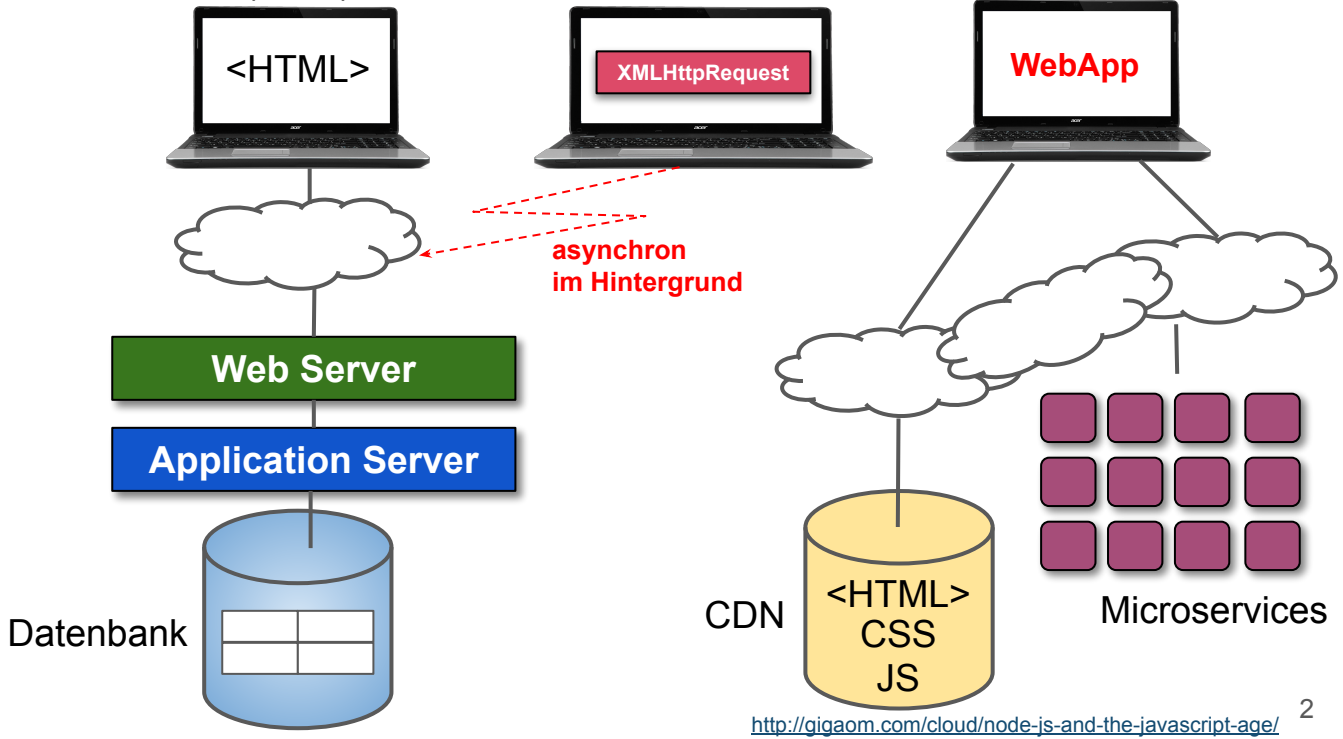
bis 1997
statisches HTML
Paradigma: **Digital Library**



ab 1997
Server-side Applications
mit >3-Schichten-Architektur
(MVC)

ab 2000
AJAX

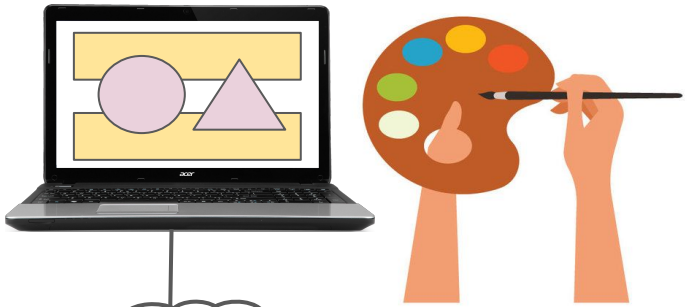
heute
Client-side Applications
mit Microservices



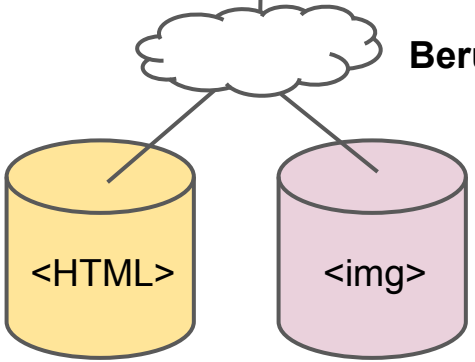
http://gigaom.com/cloud/node-js-and-the-javascript-age/

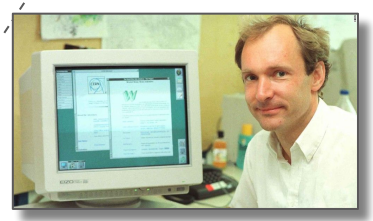
Der Kategorienumbruch des WWW
vom Hypertext zur WebApp

Digital HyperMedia Library



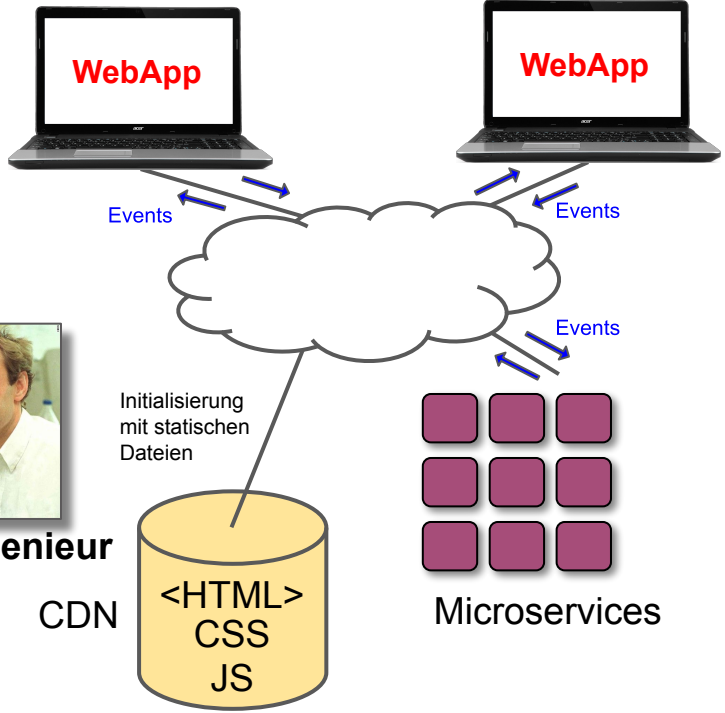
Beruf: Medien Designer





Beruf: Software-Ingenieur

Verteilte, Event-getriebene Software



https://de.wikipedia.org/wiki/Kategorienfehler

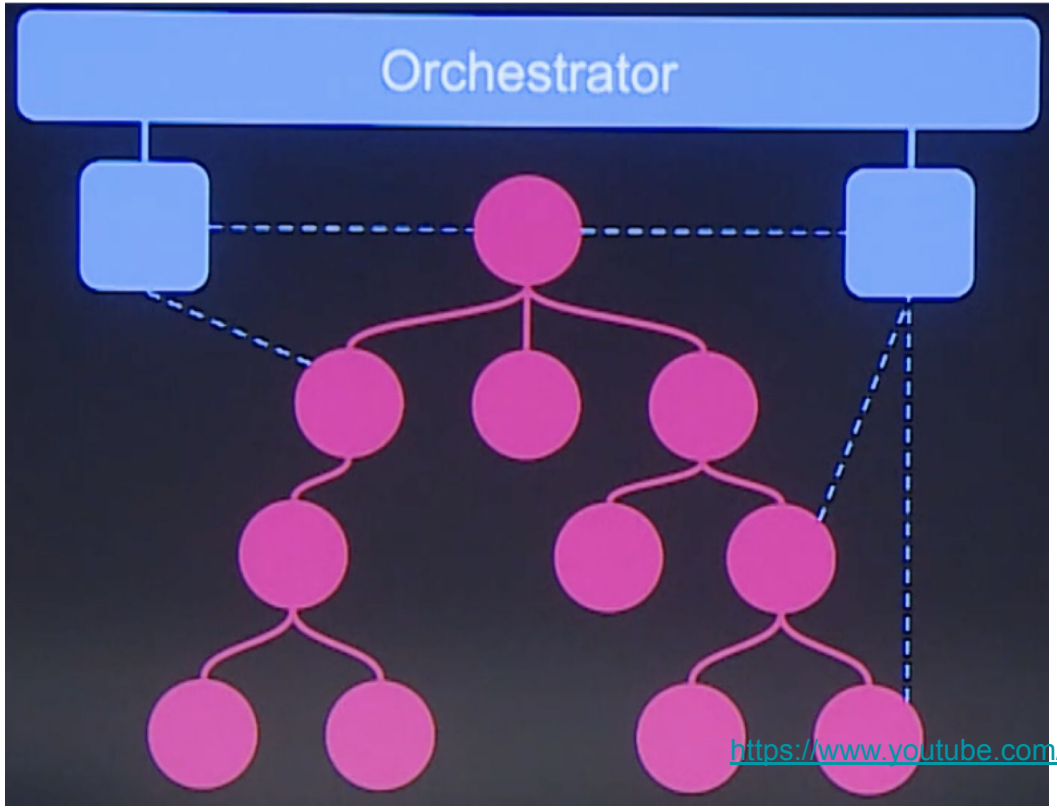
Was fehlt dem WWW zur App-Plattform?

- Application Architecture
- Multi-Threading
- Routing
- State Management
- Dependency Injection, ...

Kann man das mit den Mitteln des WWW nachbauen?

https://overreacted.io/the-elements-of-ui-engineering/

Application Architecture:
Muss es eine "Master"-Komponente geben?



https://www.youtube.com/watch?v=x9YDQUJx2uw

2 Web Application Architectures

1. Many components in the page

```
<html>
  <body>
    <app-top-bar>…</app-top-bar>
    <app-nav>…</app-nav>
    <main>
      <app-home>…</app-home>
    </main>
  </body>
</html>
```

2. The app *is* a component

```
<html>
  <body>
    <my-app></my-app>
  </body>
</html>
```

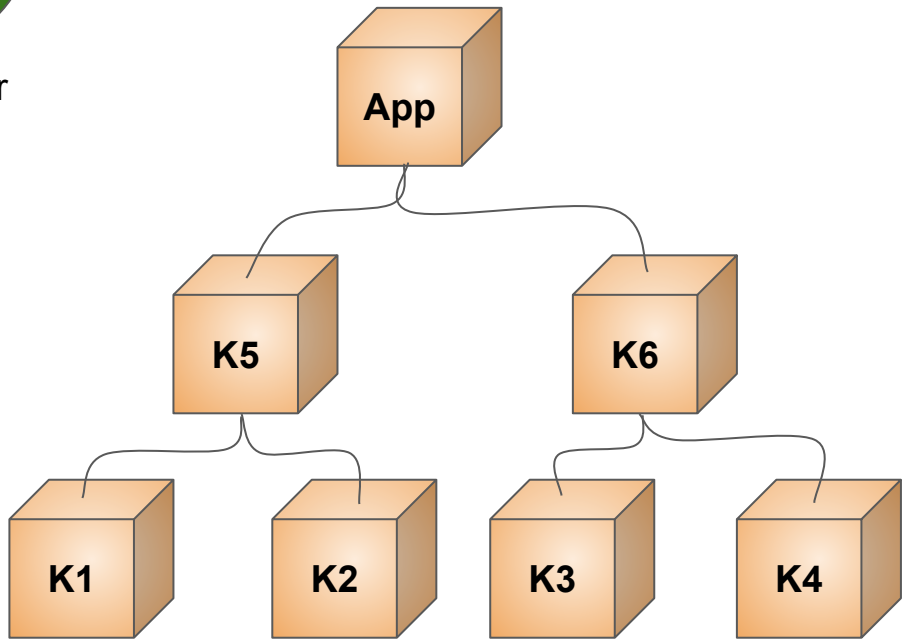
https://www.youtube.com/watch?v=x9YDQUJx2uw

Komponenten-Ansatz: App als Komponente

$$UI = f(data)$$

Ziel ist, sich als Entwickler auf die Aktualisierung der Daten beschränken zu können. Die Updates des UI sollen automatisch erfolgen. Re-rendering soll daten-getrieben bei jedem Update so sparsam wie möglich inkrementell erfolgen.

Rekursiver Aufbau der App aus Komponenten



Die Web-Plattform ist single-threaded. Das Web ist die einzige App-Plattform, die Multi-Threading vernachlässigt hat

```
let label = UILabel()

DispatchQueue.global(qos: .background).async {
  let text = loadArticleText()
  DispatchQueue.main.async {
      label.text = text
  }
}
```

Apple Swift-Code für die Anzeige asynchron geladener Texte im UI-Thread

https://youtu.be/7Rrv9qFMWNM?t=208

Web-Lösung: WebWorker + Abstraktion (z.B. Comlink)

Comlink makes WebWorkers enjoyable. Comlink is a tiny library (1.1kB), that removes the mental barrier of thinking about `postMessage` and hides the fact that you are working with workers. At a more abstract level it is an **RPC** implementation for `postMessage` and ES6 Proxies.

```
<script type="module">
 import * as Comlink from "https://unpkg.com/comlink@alpha/dist/esm/comlink.mjs";

 async function init() {
   const worker = new Worker("worker.js");
   // WebWorkers use `postMessage` and therefore work with Comlink.
   const obj = Comlink.wrap(worker);

   alert(`Counter: ${await obj.counter}`);
   await obj.inc();
   alert(`Counter: ${await obj.counter}`);
 }
 init();
</script>
```

worker.js

```
importScripts("https://unpkg.com/comlink@alpha/dist/umd/comlink.js");

const obj = {
 counter: 0,
 inc() {
   this.counter++;
 }
};

Comlink.expose(obj);
```

https://github.com/GoogleChromeLabs/comlink

Callback in einem anderen Thread

```
<script type="module">
 import * as Comlink from
"https://unpkg.com/comlink@alpha/dist/esm/comlink.mjs";
 // import * as Comlink from "../../../dist/esm/comlink.mjs";

 function callback(value) {
   alert(`Result: ${value}`);
 }

 async function init() {
   const remoteFunction = Comlink.wrap(new Worker("worker.js"));
   await remoteFunction(Comlink.proxy(callback));
 }

 init();
</script>
```

```
importScripts("https://unpkg.com/comlink@alpha/dist/umd/comlink.js");

async function remoteFunction(cb) {
 await cb("A string from a worker");
}

Comlink.expose(remoteFunction);
```

https://github.com/GoogleChromeLabs/comlink

Classes in einem anderen Thread

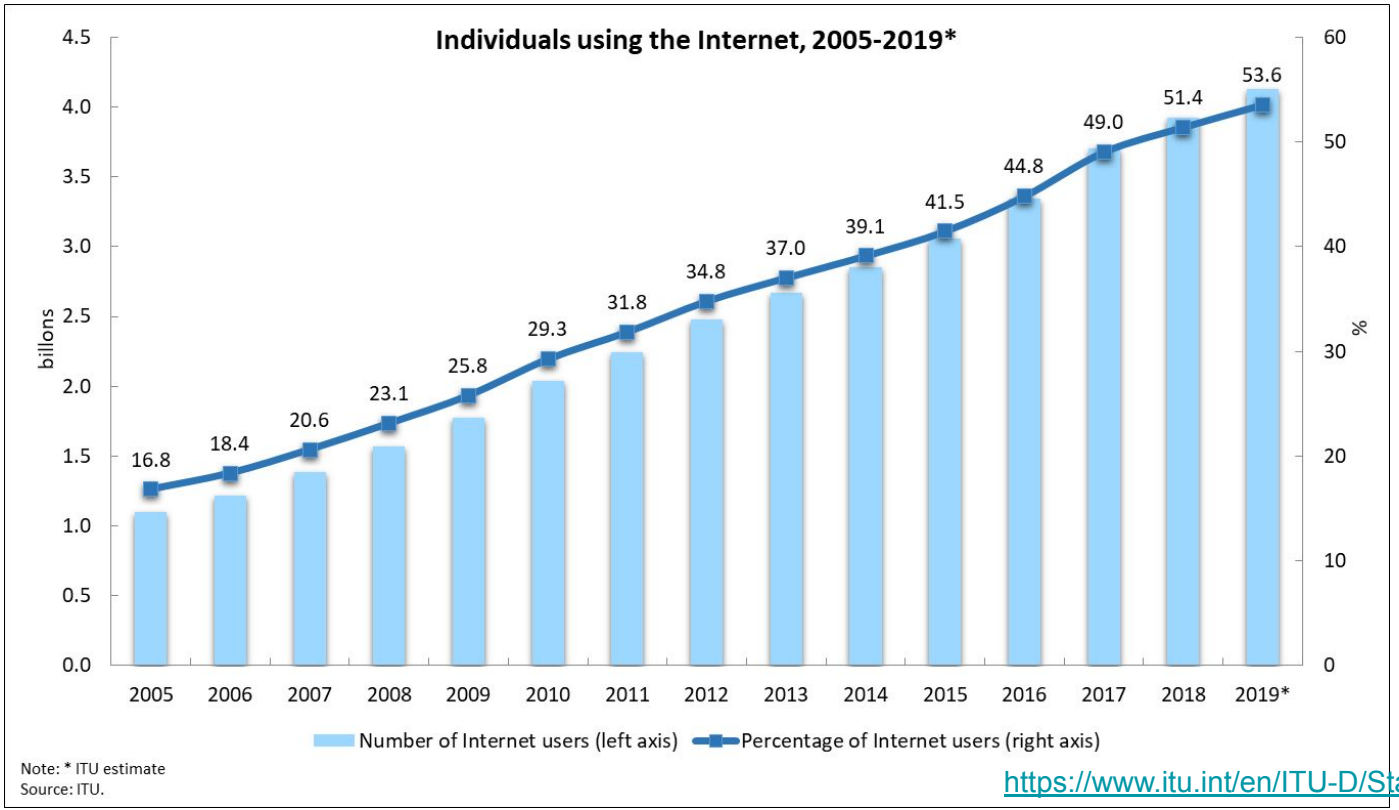
```
<script type="module">
 import * as Comlink from "https://unpkg.com/comlink@alpha/dist/esm/comlink.mjs";

 let instance1, instance2;
 async function showState() {
   alert(`instance1.counter = ${await instance1.counter},
     instance2.counter = ${await instance2.counter}`);
 }

 async function init() {
   const MyClass = Comlink.wrap(new Worker("worker.js"));
   instance1 = await new MyClass();
   instance2 = await new MyClass(42);
   await showState();
   await instance1.increment();
   await instance2.increment(23);
   await showState();
 }
 init();
</script>
```

```
importScripts("https://unpkg.com/comlink@alpha/dist/umd/comlink.js");

class MyClass {
 constructor(init = 0) {
   console.log(init);
   this._counter = init;
 }

 get counter() {
   return this._counter;
 }

 increment(delta = 1) {
   this._counter += delta;
 }
}

Comlink.expose(MyClass);
```

https://github.com/GoogleChromeLabs/comlink

Performanz ist das nächste Großprojekt im Web! Warum?

Internet Users 2005 - 2019. Heute ~ 53%



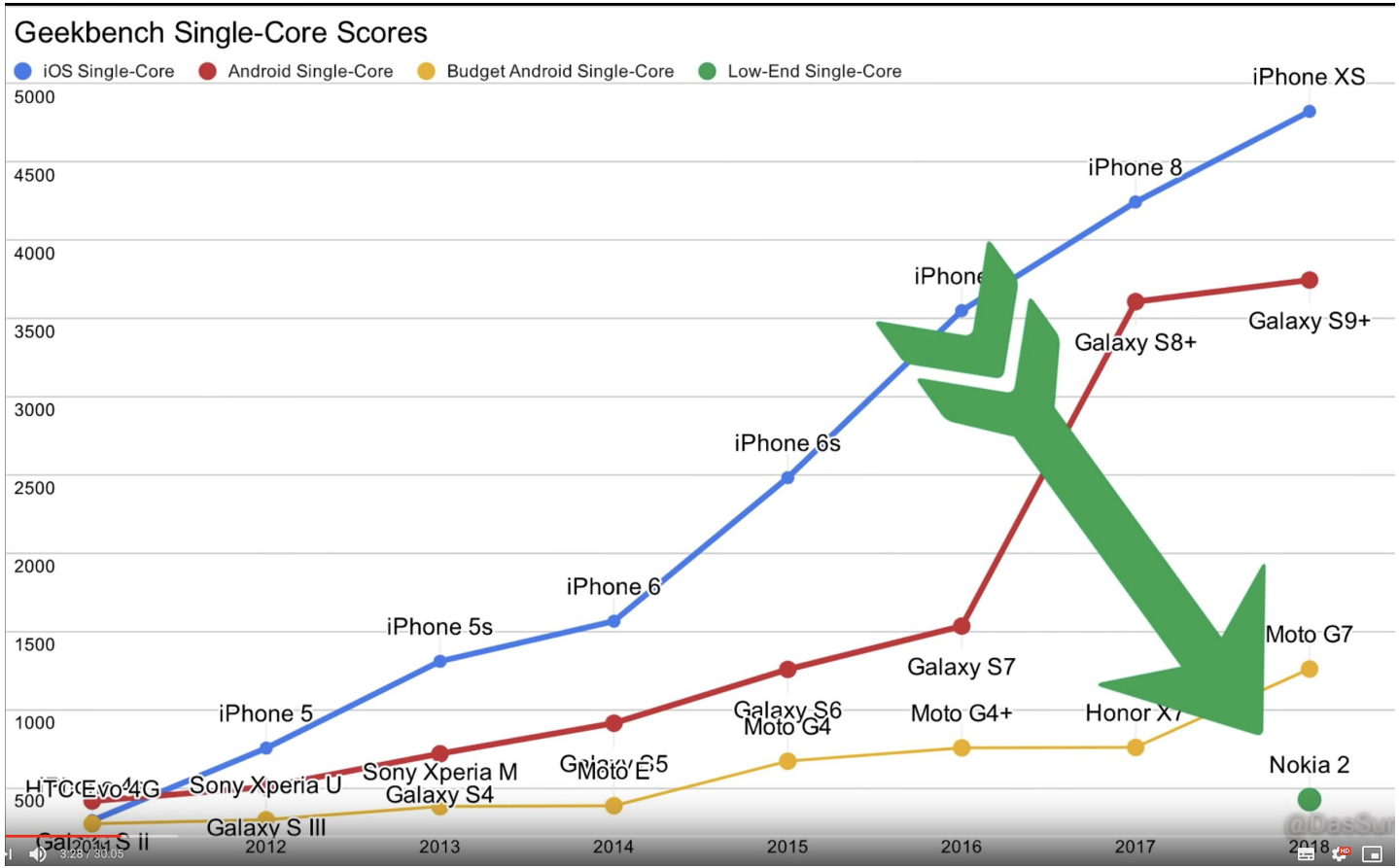
https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx

Next Billion Users (NBU) on the Internet

Feature Phones: 45€

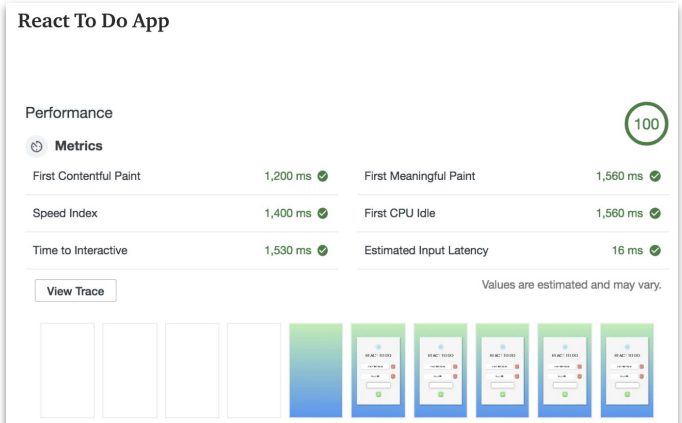
90€ Nokia 2

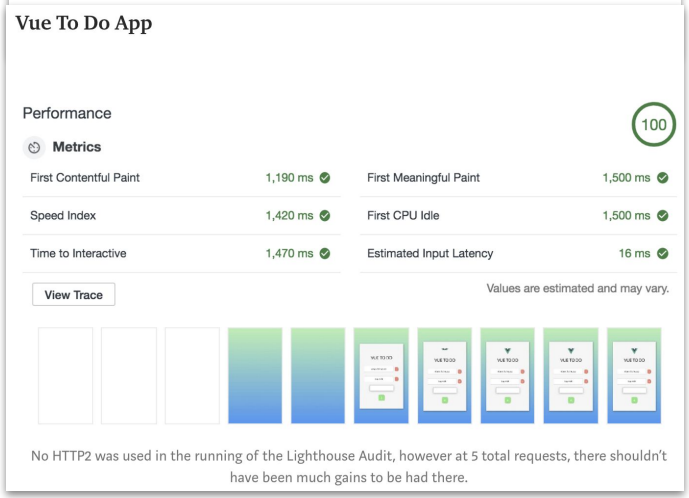
https://youtu.be/7Rrv9qFMWNM?t=208

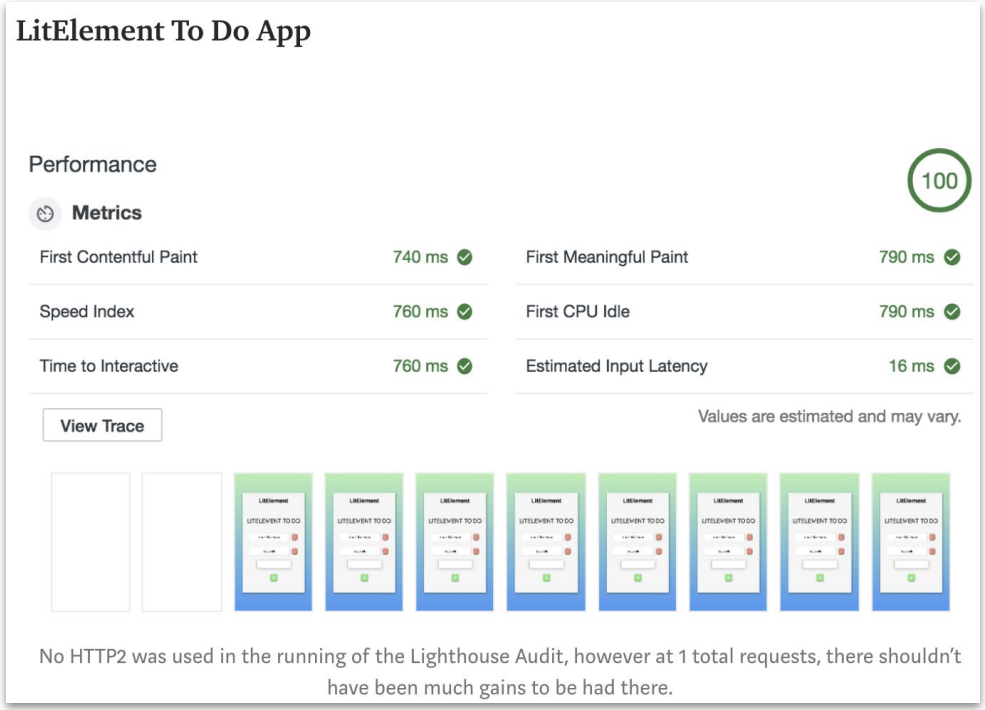


https://youtu.be/7Rrv9qFMWNM?t=208

LitElement doppelt so schnell wie React, Vue

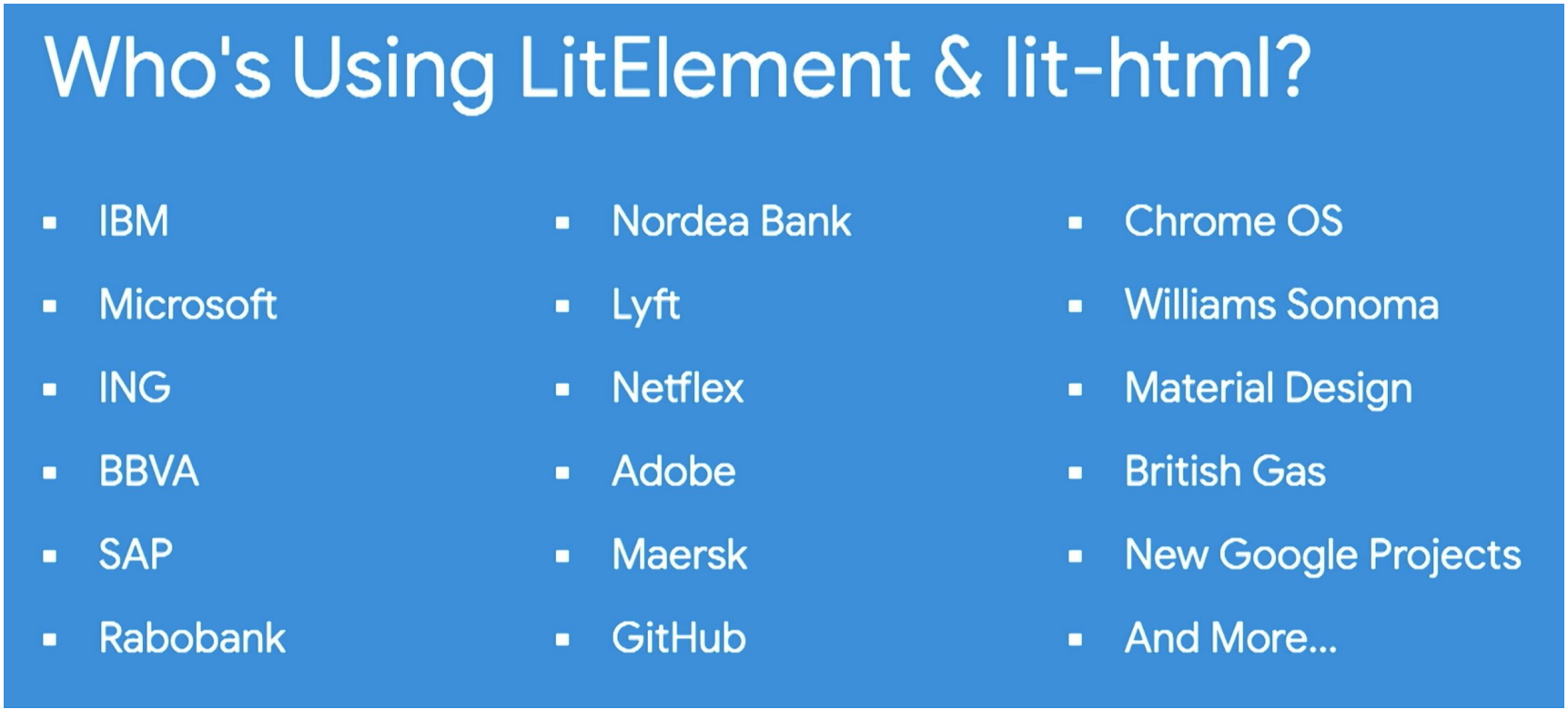






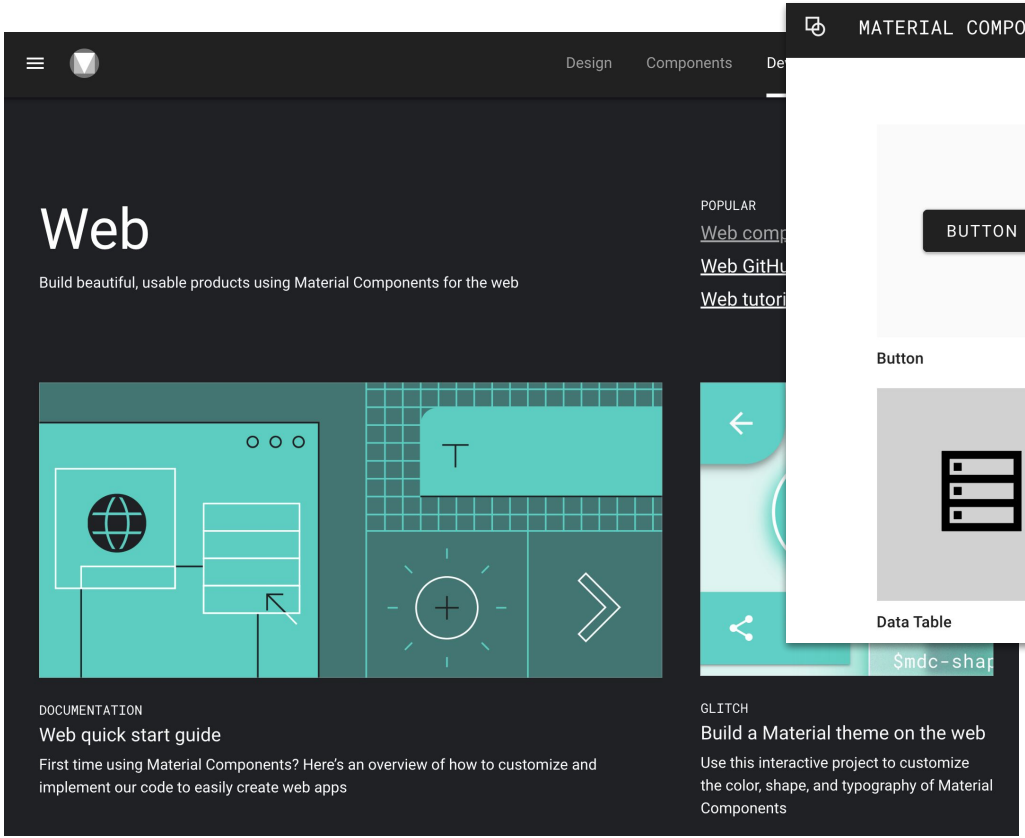
https://medium.com/@westbrook/litelement-to-do-app-1e08a31707a4

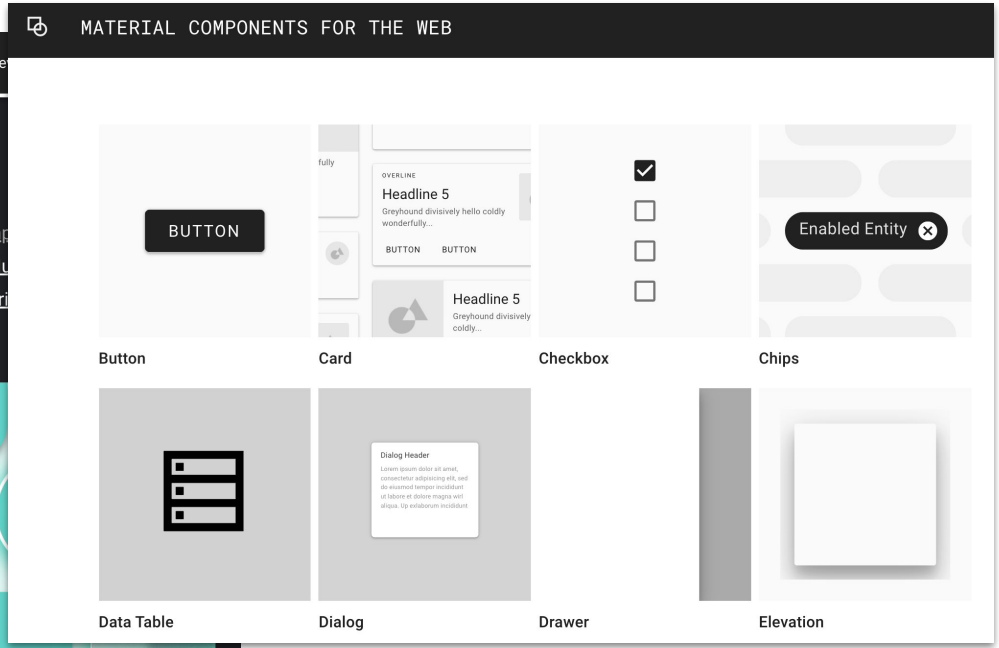
Wer verwendet LitElement?



https://www.youtube.com/watch?v=9FB0GSOAESo&t=2s

Material Design Web Components Gallery





https://material.io/develop/web/

Material Design **W**eb **C**omponent (MWC) Button

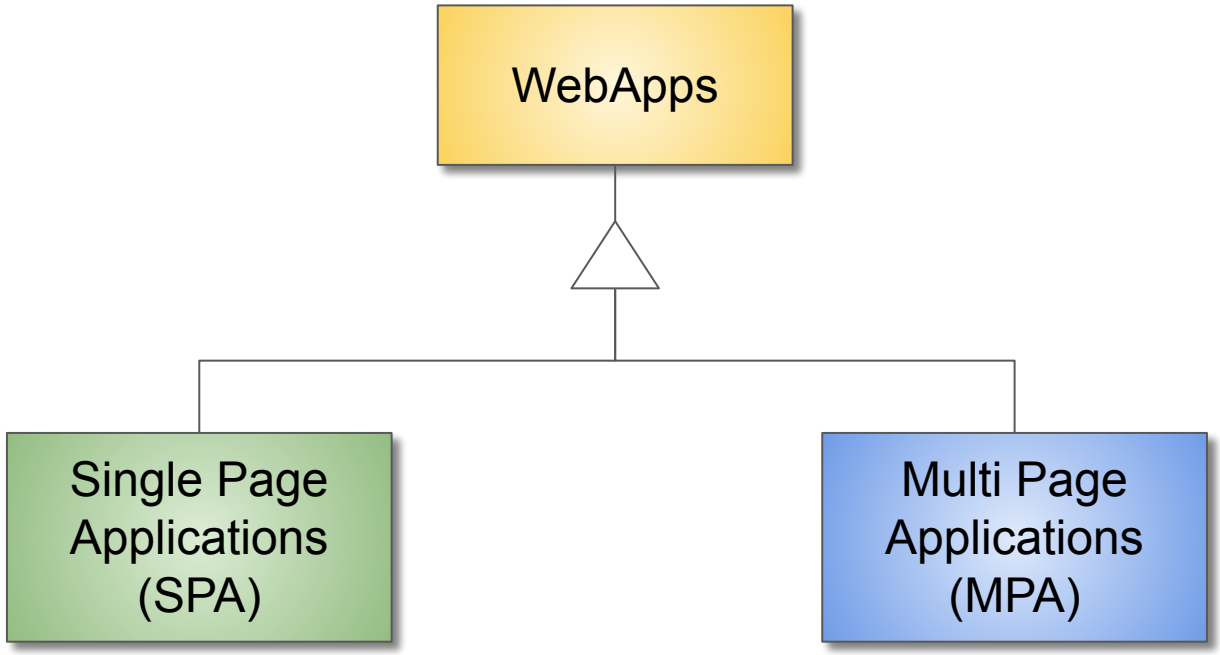
```
<show-case></show-case>
<script type="module">
  import { LitElement, html } from 'https://unpkg.com/lit-element?module';
  import 'https://unpkg.com/@material/mwc-button?module';

  class ShowCase extends LitElement {
    static get properties(){
      return {
        active: { type: String }
      }
    }
    toggle(){
      this.active = ! this.active;
    }
    render(){
      return html`
        <mwc-button raised outlined @click=${this.toggle}>
          Mark as ${this.active ? 'unread' : 'read'}
        </mwc-button>
      `;
    }
  }

  customElements.define('show-case', ShowCase);
</script>
```

import

use

https://material-components.github.io/material-components-web-catalog/#/component/button

WebApps

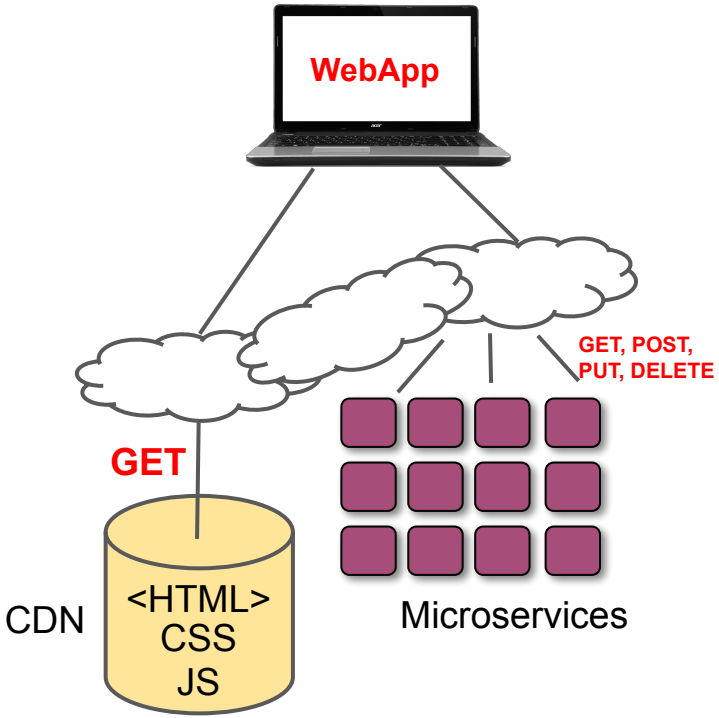

WebApps
Single Page Applications (SPA)
Multi Page Applications (MPA)

Routing in SPAs und MPAs

Der Router bestimmt,
welche Views mit welchen Daten
bei welcher URL anzuzeigen ist.

1. RESTful Routing
2. Location API
3. History API
    a. before 2012 very limited
    b. 2012 HTML5 History API added
        i. pushState
        ii. replaceState

RESTful Routing



Action	HTTP Method	Result (controller#action)	Url
SHOW	GET	profile#show	/profile
NEW	GET	profile#new	/profile/new
CREATE	POST	profile#create	/profile
EDIT	GET	profile#edit	/profile/edit
UPDATE	PUT	profile#update	/profile
DESTROY	DELETE	profile#destroy	/profile

http://restfulrouting.com/#resources

RESTful Routing

Action	HTTP Method	Result (controller#action)	Url Template
INDEX	GET	products#index	/products
SHOW	GET	products#show	/products/{id}
NEW	GET	products#new	/products/new
CREATE	POST	products#create	/products
EDIT	GET	products#edit	/products/{id}/edit
UPDATE	PUT	products#update	/products/{id}
DESTROY	DELETE	products#destroy	/products/{id}

Platzhalter für Identifier: 1,2,3,...

http://restfulrouting.com/#resources

REST Resource Naming Guide

In REST, primary data representation is called **Resource**. Having a strong and **consistent REST resource naming strategy** – will definitely prove one of the best design decisions in the long term.

> *The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g., a person), and so on. In other words, any concept that might be the target of an author's hypertext reference must fit within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.*
>
> Roy Fielding's dissertation

A resource can be a singleton or a **collection**. For example, "`customers`" is a collection resource and "`customer`" is a singleton resource (in a banking domain). We can identify "`customers`" collection resource using the URI "`/customers`". We can identify a single "`customer`" resource using the URI "`/customers/{customerId}`".

A resource may contain **sub-collection** resources also. For example, sub-collection resource "`accounts`" of a particular "`customer`" can be identified using the URN "`/customers/{customerId}/accounts`" (in a banking domain). Similarly, a singleton resource "`account`" inside the sub-collection resource "`accounts`" can be identified as follows: "`/customers/{customerId}/accounts/{accountId}`".

https://restfulapi.net/resource-naming/

Sub-Collection Resources

Action	HTTP Method	Result (controller#action)	Url Template
INDEX	GET	reviews#index	/products/{productId}/reviews
SHOW	GET	reviews#show	/products/{productId}/reviews/{id}
NEW	GET	reviews#new	/products/{productId}/reviews/new
CREATE	POST	reviews#create	/products/{productId}/reviews
EDIT	GET	reviews#edit	/products/{productId}/reviews/{id}/edit
UPDATE	PUT	reviews#update	/products/{productId}/reviews/{id}
DESTROY	DELETE	reviews#destroy	/products/{productId}/reviews/{id}

http://restfulrouting.com/#resources

REST Resource Naming Best Practices

- Use nouns to represent resources
- Use forward slash (/) to indicate hierarchical relationships
- Do not use trailing forward slash (/) in URIs
- Use hyphens (-) to improve the readability of URIs
- Do not use underscores (_)
- Use lowercase letters in URIs
- Do not use file extensions
- Never use CRUD function names in URIs
- Use query component to filter URI collection

https://restfulapi.net/resource-naming/

Location API

https://developer.mozilla.org/en-US/docs/Web/API/Location

Location

Web technology for developers > Web APIs > Location

The **`Location`** interface represents the location (URL) of the object it is linked to. Changes done on it are reflected on the object it relates to. Both the `Document` and `Window` interface have such a linked `Location`, accessible via `Document.location` and `Window.location` respectively.

https://de.wikipedia.org/wiki/Uniform_Resource_Identifier

Routing mit der Location API

```
<button id="a">A</button><button id="b">B</button>
<div id="content"></div>
<script>
  const a = document.getElementById('a');
  const b = document.getElementById('b');
  const content = document.getElementById('content');
  ['a', 'b'].forEach( hash => {
    document.getElementById( hash ).addEventListener( 'click', () => {
      location.hash = hash;
    });
  });
  window.onhashchange = function() {
    content.innerHTML = contentIndex[ location.hash ];
  };
  contentIndex = {
    '#a': `<h1>Header A</h1>
       <p>Text for A</p>`,
    '#b': `<h1>Header B</h1>
       <p>Text for B</p>`,
    '': `<h1>Please press a button</h1>`,
  };
</script>
```

In der Browser-URL wird nur das Fragment verändert.

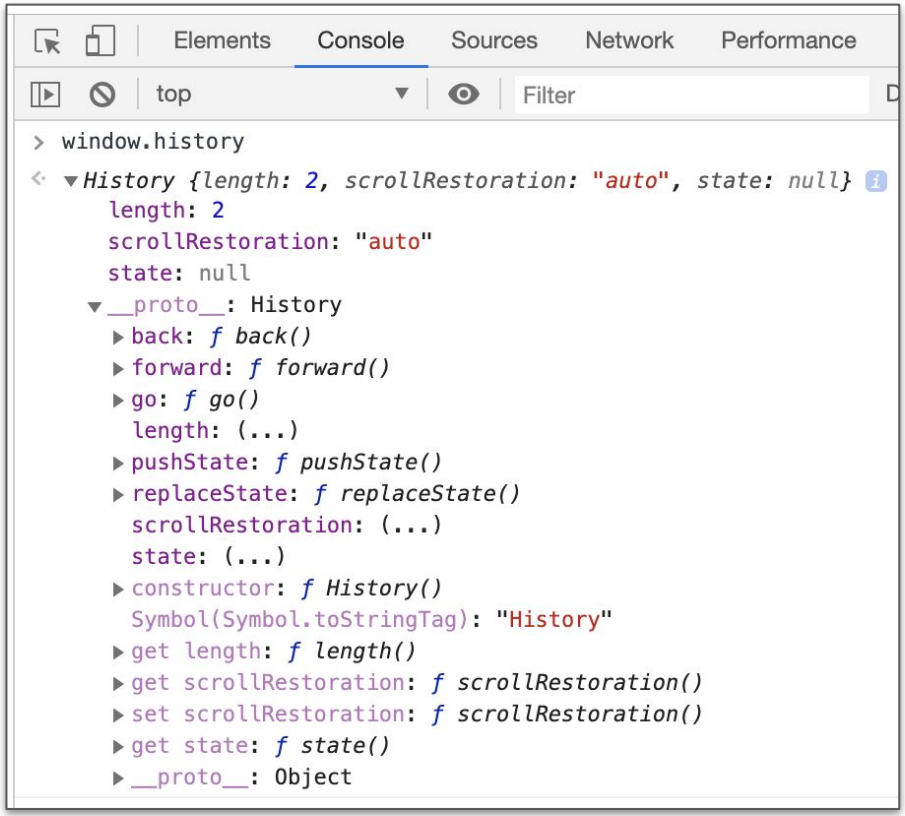
History API

History

Web technology for developers > Web APIs > History

The **`History`** interface allows manipulation of the browser *session history*, that is the pages visited in the tab or frame that the current page is loaded in.

DevTools Console



History API (erst seit 2012 mit HTML5 durch WhatWG)

Suppose http://mozilla.org/foo.html executes the following JavaScript:

Warum?

```
var stateObj = { konto: "bar" };
history.pushState( stateObj, "Banking", "bank24.de.html");
```

This will cause the URL bar to display http://mozilla.org/bank24.de.html, but won't cause the browser to load bank24.de.html or even check that bank24.de.html exists. Then suppose http://mozilla.org/bar.html executes the following JavaScript:

```
history.replaceState( stateObj, "page 3", "bar2.html");
```

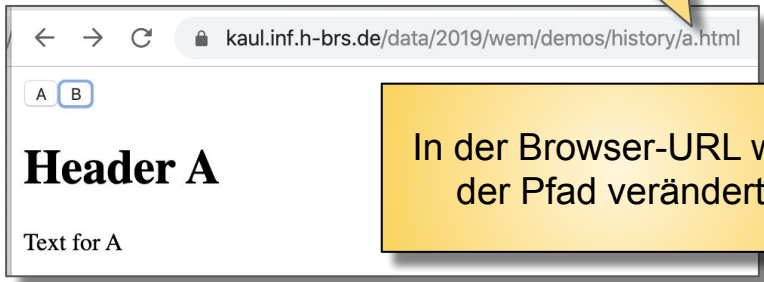
This will cause the URL bar to display http://mozilla.org/bar2.html, but won't cause the browser to load `bar2.html` or even check that `bar2.html` exists.

https://developer.mozilla.org/en-US/docs/Web/API/History_API/Working_with_the_History_API
https://developer.mozilla.org/en-US/docs/Web/API/History/replaceState

Routing mit der History API

```
<button id="a">A</button><button id="b">B</button>
<div id="content"><h1>Please press a button</h1></div>
<script>
 const a = document.getElementById('a');
 const b = document.getElementById('b');
 const content = document.getElementById('content');
 ['a', 'b'].forEach( id  => {
   document.getElementById( id ).addEventListener( 'click', () => {
     history.pushState( { id }, `Title: ${id}`, `${id}.html` );
     setPage( id );
   });
 });
 const contentIndex = {
   'a': `<h1>Header A</h1><p>Text for A</p>`,
   'b': `<h1>Header B</h1><p>Text for B</p>`,
   'other': '<h1>Please press a button</h1>'
 };
 window.addEventListener('popstate', () => {
   setPage( history.state ? history.state.id : 'other' );
 });
 function setPage( index ){
   content.innerHTML = contentIndex[ index ];
 }
</script>
```

a.html wird angezeigt, obwohl diese Seite nie geladen wurde und nie existierte.



In der Browser-URL wird der Pfad verändert.

Router page.js (1kB, ES6 import)

```
<h1>Page.js Demo</h1>
<p></p>
<ul>
  <li><a href="./">/</a></li>
  <li><a href="#whoop">#whoop</a></li>
  <li><a href="./about">/about</a></li>
  <li><a href="./contact">/contact</a></li>
  <li><a href="./contact/me">/contact/me</a></li>
  <li><a href="./not-found?foo=bar">/not-found</a></li>
</ul>
<script type="module">
 import page from "//unpkg.com/page/page.mjs";

 const basePath = window.location.pathname.split('/').slice(0,-1).join('/');
 page.base( basePath );

 const p = document.querySelector('p');

 page('/', index);
 page('/about', about);
 page('/contact', contact);
 page('/contact/:contactName', contact);
 page('*', notfound)
 page();
```

```
function index() {
  p.textContent = 'viewing index';
}

function about() {
  p.textContent = 'viewing about';
}

function contact(ctx) {
  p.textContent = 'viewing contact '
                + (ctx.params.contactName || '');
  // see Issue https://github.com/visionmedia/page.js/issues/494
}

function notfound(ctx) {
  p.textContent = 'Not found ' + location.href;
}
</script>
```

https://github.com/visionmedia/page.js
http://visionmedia.github.io/page.js/

LitElement Router

```
class App extends routerMixin( LitElement ) {
 static get properties() {
  return {
   route: { type: String },
   params: { type: Object },
   query: { type: Object },
   data: { type: Object }
  };
 }
 static get routes() {
  return [{
   name: 'home',
   pattern: '',
   data: { title: 'Home' }
  },
  {
   name: 'info',
   pattern: 'info'
  }];
 }
 constructor() {
  super();
  this.route = '';
  this.params = {};
  this.query = {};
  this.data = {};
 }
```

```
router(route, params, query, data) {
  this.route = route;
  this.params = params;
  this.query = query;
  this.data = data;
  console.log(route, params, query, data);
}
```

```
export let routerMixin = (superclass) => class extends superclass {
    static get properties() { … }
    connectedCallback(...args) { … }
    routing(routes, callback) { … }
}
```

```
render() {
  return html`
  <app-link href="/">Home</app-link>
  <app-link href="/info">Info</app-link>
  <app-link href="/info?data=12345">Info?data=12345</app-link>
  <app-link href="/user/14">user/14</app-link>

  <app-main active-route=${this.route}>
     <h1 route='home'>Home</h1>
     <h1 route='info'>Info ${this.query.data}</h1>
     <h1 route='user'>User ${this.params.id} </h1>
     <h1 route='not-found'>Not Found</h1>
  </app-main>
 `;
 }
}
customElements.define('my-app', App);
```

https://github.com/hamedasemi/lit-element-router

LitElement Router Links: Umleitung für HyperLinks

```
<script type="module">
 import { LitElement, html, css } from 'https://unpkg.com/lit-element/lit-element.js?module';
 import { routerMixin, linkMixin, outletMixin } from 'https://unpkg.com/lit-element-router/lit-element-router.js?module';

class Link extends linkMixin( LitElement ) {
 static get properties() {
  return { href: { type: String } };
 }
 constructor() {
  super(); this.href = '';
 }
 render() {
  return html`
      <a href='${this.href}' @click='${this.linkClick}'>
        <slot></slot>
      </a>
    `;
 }
 linkClick(event) {
  event.preventDefault();
  this.navigate(this.href);
 }
}
customElements.define('app-link', Link);
```

```
export let linkMixin = (superclass) => class extends superclass {
  navigate(href) {
    window.history.pushState({}, null, href);
    window.dispatchEvent(new CustomEvent('route'));
  }
};
```

```
render() {
 return html`
 <app-link href="/">Home</app-link>
 <app-link href="/info">Info</app-link>
 <app-link href="/info?data=12345">Info?data=12345</app-link>
```

https://github.com/hamedasemi/lit-element-router
https://www.npmjs.com/package/lit-element-router

SPA - Routing - Prinzipien

- URL codiert öffentlichen Teil des App-Zustands
  - Geheime Daten in URL verboten
  - Ausweg: Geheime Daten verhashen
- URL-Änderungen erzeugen Eintrag im History-Stack des Browsers
  - ⇒ UNDO- und REDO-Fähigkeit für die App
- Data driven Routing
  - Router parses URL ⇒ Pfad + Parameter
    - Pfad bestimmt View
    - Parameter bestimmen Zustand
    - … oder Kombination aus Beidem

Zusammenfassung Routing

1. RESTful Routing
2. Location API
3. History API
    a. before 2012 very limited
    b. 2012 HTML5 History API added
        i. pushState
        ii. replaceState

Routing Libs

#All Roads Lead To Rome

There are many roads that lead to Rome, and different apps have different requirements. For this reason, it's hard to recommend a single router for your project. Instead, we'll provide you with a list of different routers for different use cases that work well with your web component projects.

In alphabetical order:

- lit-element-router
- lit-router
- pwa-helpers router
- redux first routing
- simplr-router
- vaadin-router
- page.js

https://open-wc.org/developing/routing.html

Einen Router schreiben ist nicht schwer:
A modern JavaScript router in 100 lines

Requirements

The router should ...

- be less then 100 lines
- supports hash typed URLs like *http://site.com#products/list*
- work with the **History API**
- provide easy-to-use API
- not run automatically
- listen for changes only if we want to

https://krasimirtsonev.com/blog/article/A-modern-JavaScript-router-in-100-lines-history-api-pushState-hash-url

State Management in WebApps

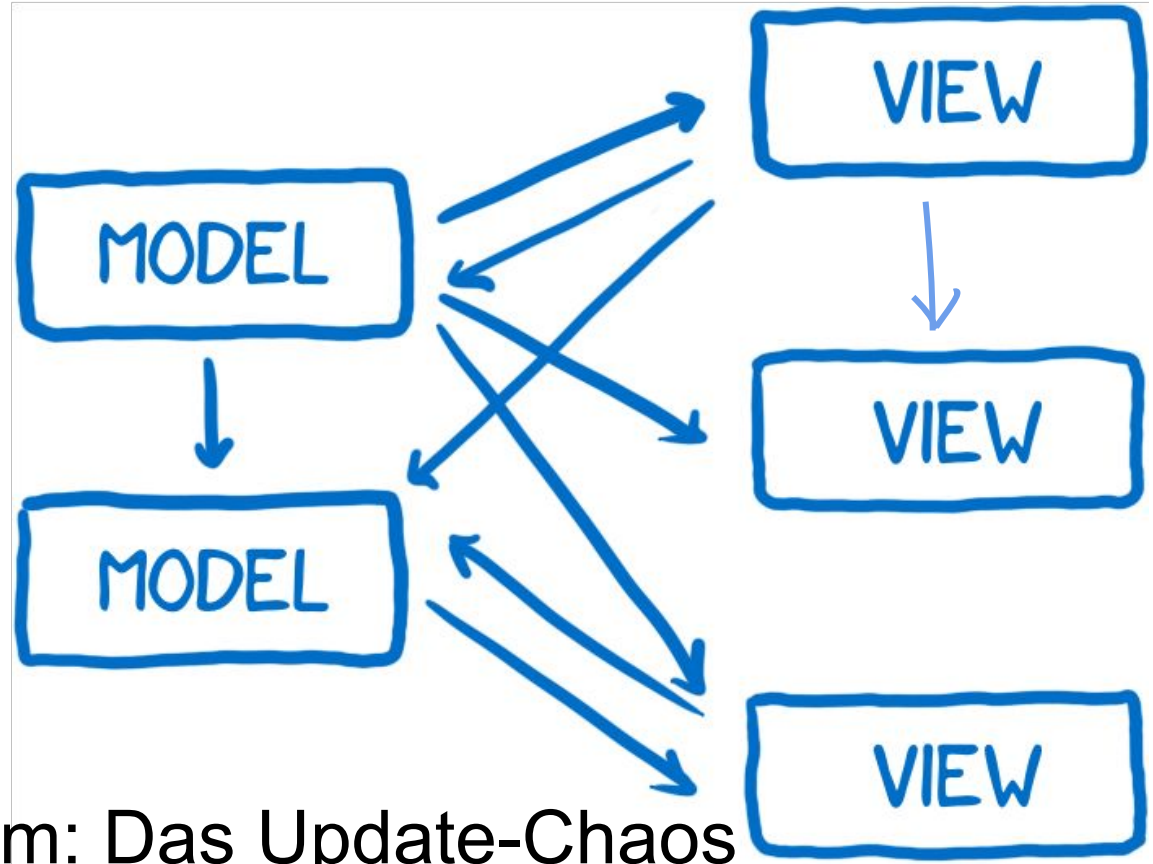
1. "Freies" dezentrales State Management

   Jede Komponente ist eine Instanz einer Klasse und kann ihren Zustand selbst verwalten.

   ⇒ Eine WebApp besteht aus einem Netzwerk vieler Komponenten, die miteinander kommunizieren

2. Das State-Action-Model (SAM) - Pattern
3. Hooks
4. Redux

"Freies" dezentrales State Management

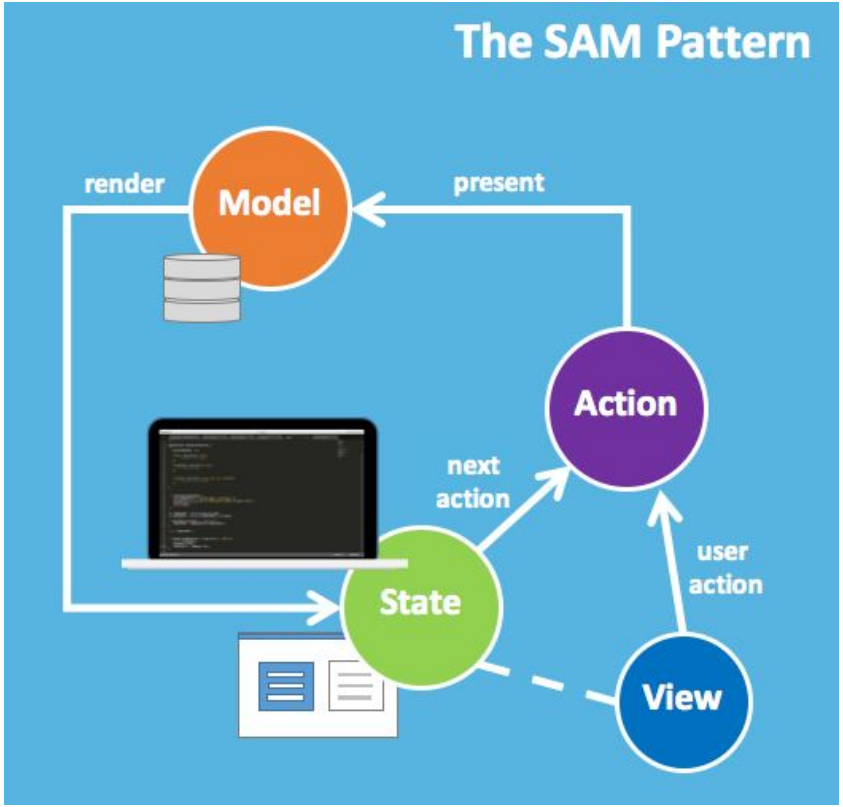


Das Problem: Das Update-Chaos

https://code-cartoons.com/a-cartoon-guide-to-flux-6157355ab207

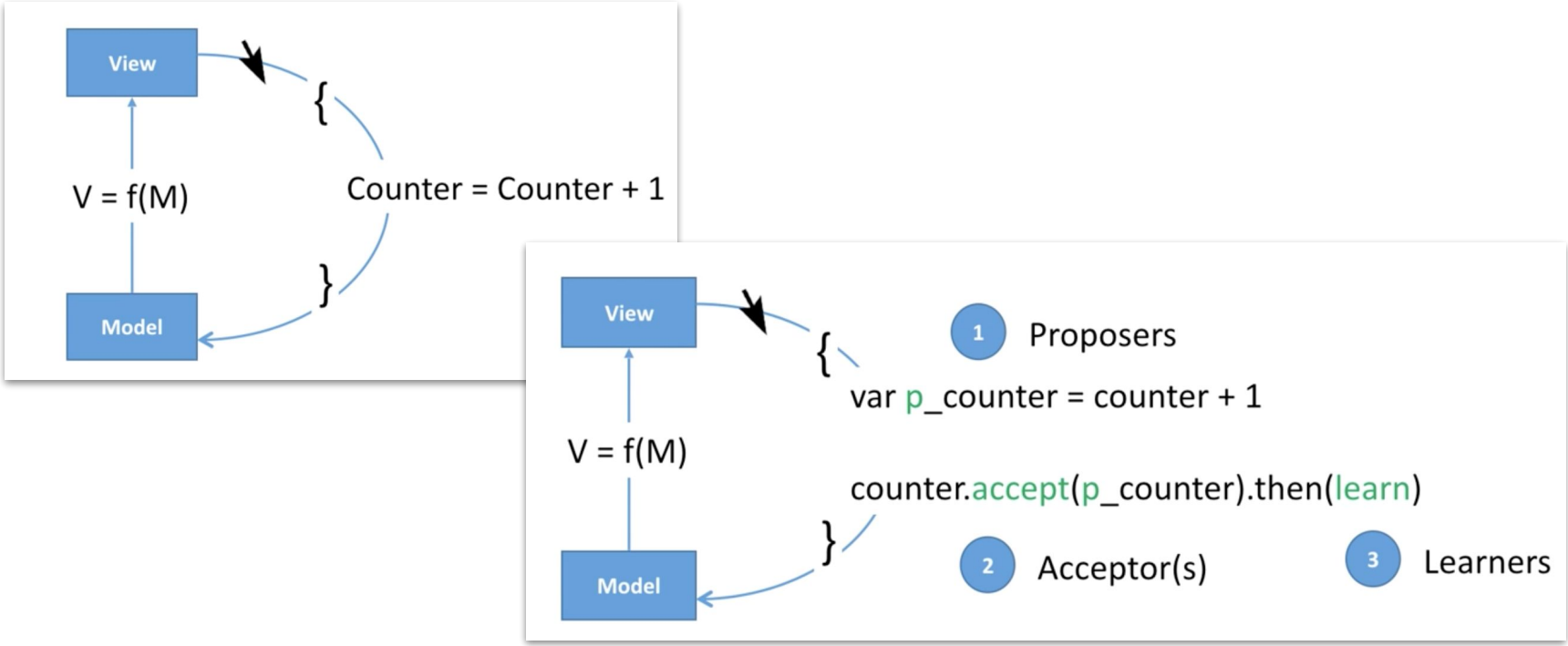
Das State-Action-Model (SAM) - Pattern

- Entkopplung von
  - State
  - Action
  - View
  - Model
- Separation of Concerns
  - Action Proposals (z.B. von Views)
  - Model accepts
  - "learn" from new state
- Prinzip: State Mutations are first-class objects



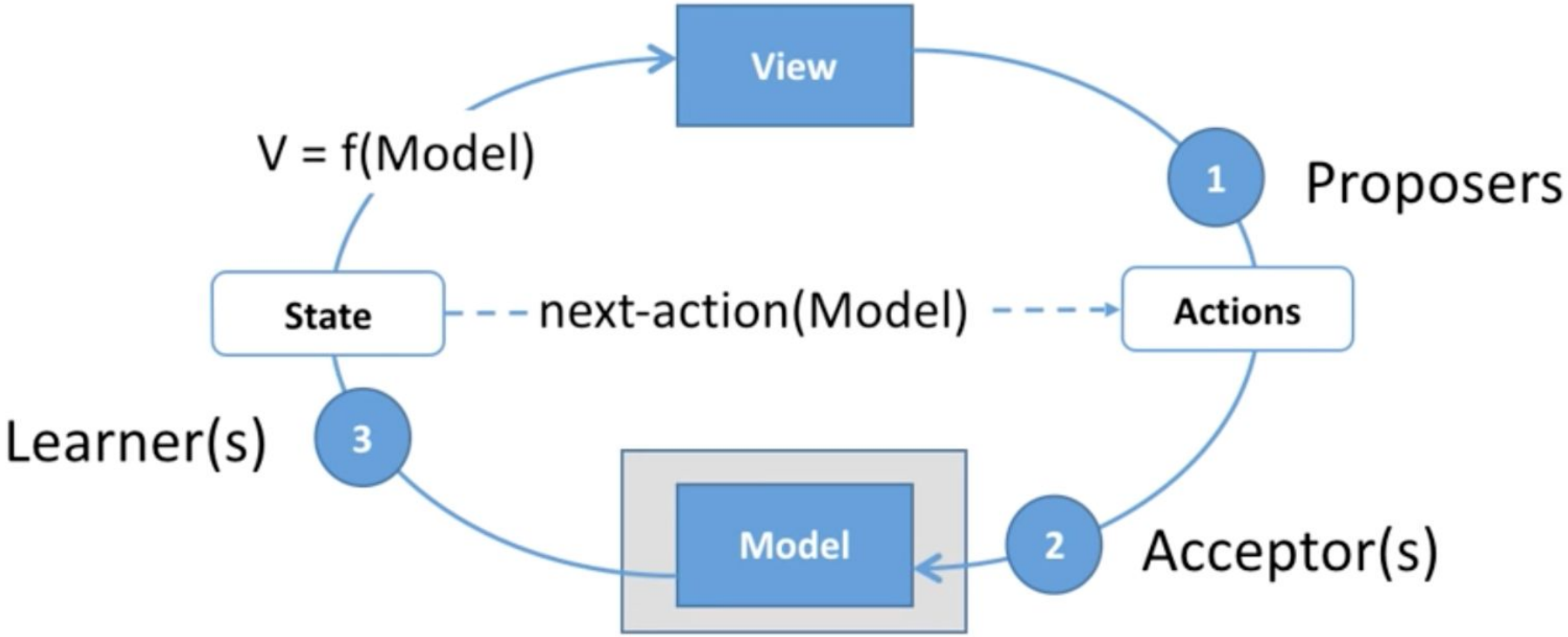
https://sam.js.org/

Update zerlegen in (1.) Proposer (2.) Acceptor (3.) Learner



https://www.youtube.com/watch?v=lMgSS_b98S8

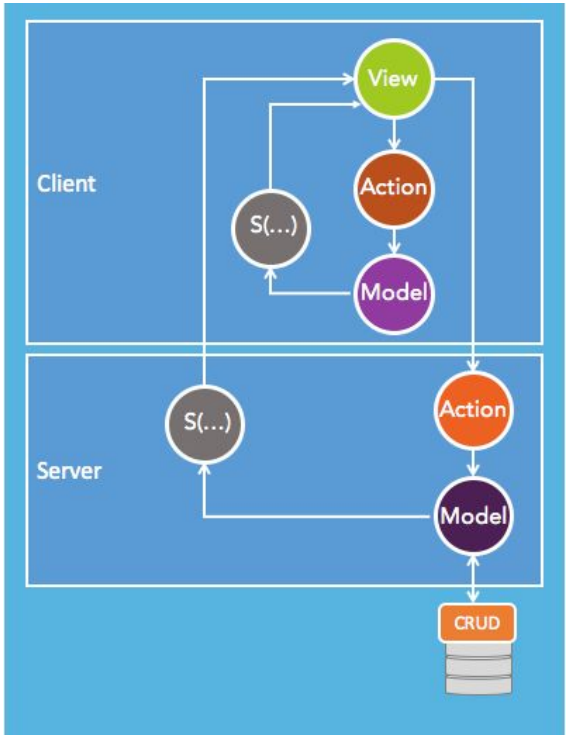
(1.) Proposers (2.) Acceptors (3.) Learners



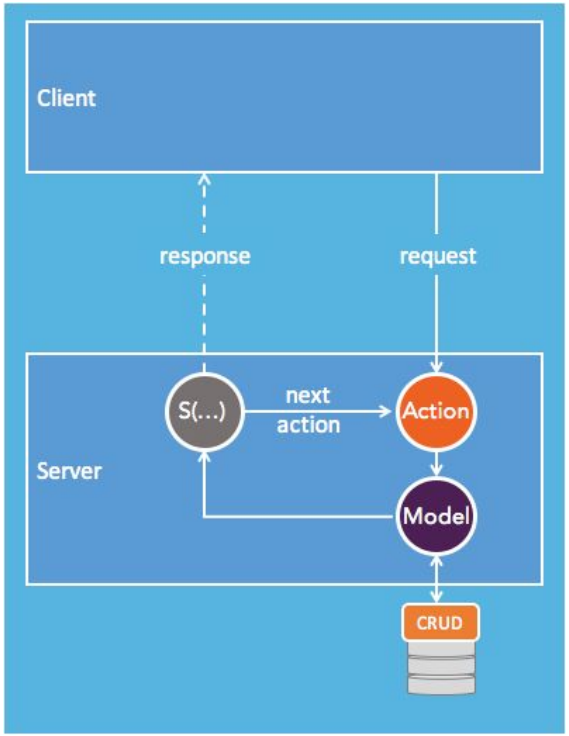
https://www.youtube.com/watch?v=IMgSS_b98S8

Varianten von SAM

Isomorphic JavaScript



Headless SAM



https://sam.js.org/

Vorteile des SAM-Patterns

- **Programming model**
  - Centered on Mutation, not immutability
  - True Single State Tree, no Sagas/Stateful components
  - Focus on "what's allowed", not subscriptions
  - View Components are 100% decoupled from the application
  - Functional UI/HTML (code generation), not templates
- **Architecture**
  - Wiring agnostic
  - Side-effects friendly
  - Action "Hang back" / Cancellations
  - Truly Isomorphic
  - 3rd party Actions (OAuth)

https://www.youtube.com/watch?v=IMgSS_b98S8

State Management in WebApps

1. "Freies" dezentrales State Management

   Jede Komponente ist eine Instanz einer Klasse und kann ihren Zustand selbst verwalten.

   ⇒ Eine WebApp besteht aus einem Netzwerk vieler Komponenten, die miteinander kommunizieren

2. Das State-Action-Model (SAM) - Pattern
3. **Hooks**
4. Redux

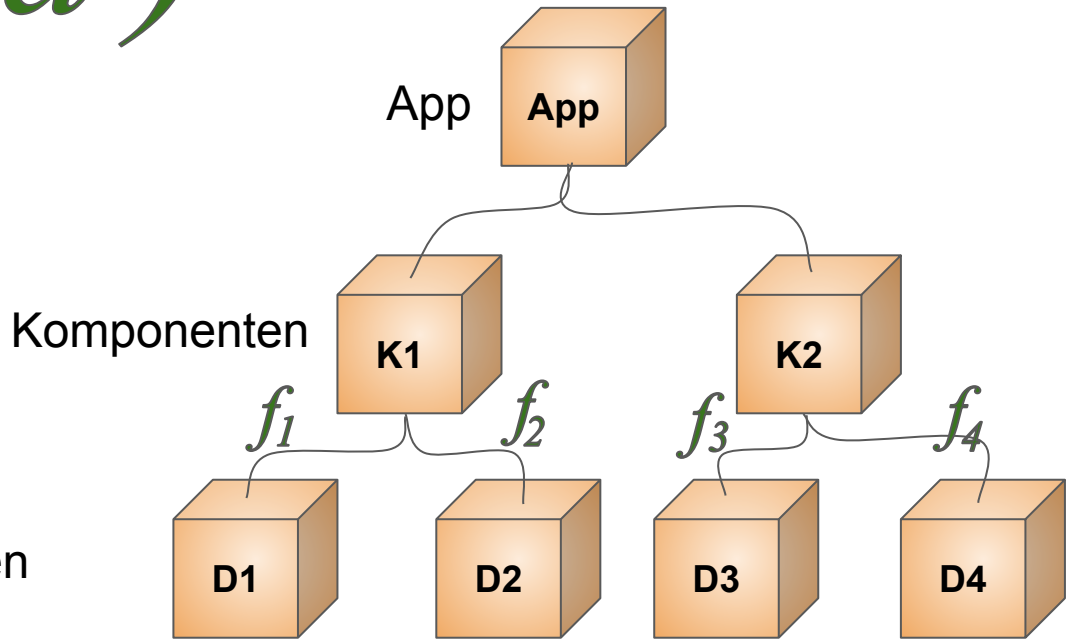
Die Hook-Lösung für State Management

Funktionale Lösung: User Interface als Funktion

$$UI = f(data)$$

Die App ist eine Funktion.

Sie liefert mit Daten bestücktes HTML. Darin können Komponenten zum Einsatz kommen, die selbst wiederum nur Funktionen sind, die mit Daten bestücktes HTML liefern.



Funktionen als Komponenten: Function Component

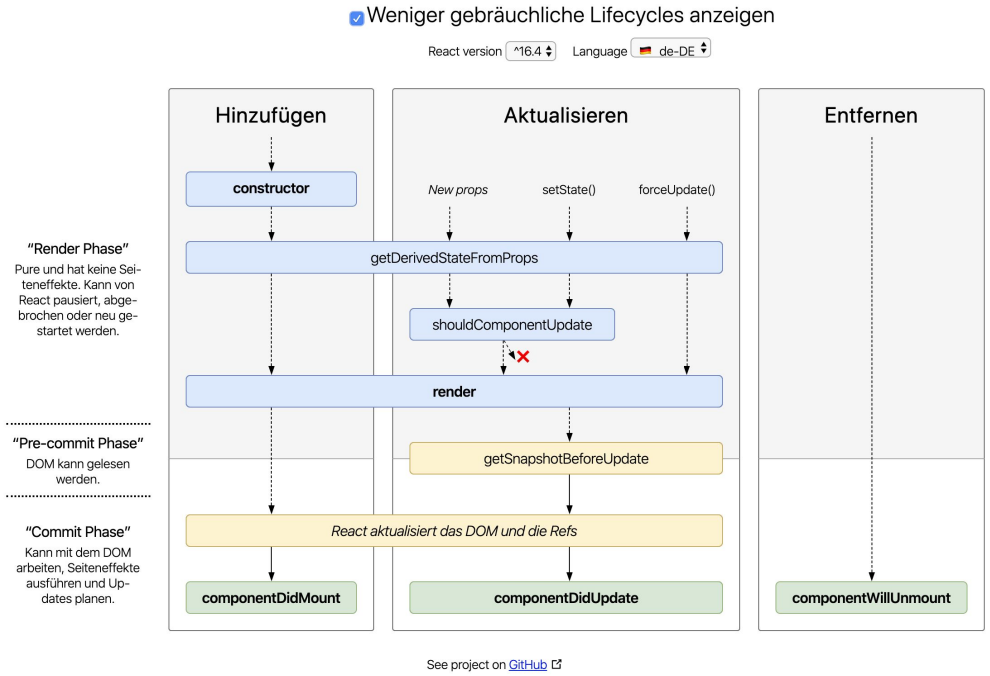
```
<my-comp></my-comp>

<script type="module">

 import { html } from 'https://unpkg.com/lit-html/lit-html.js';
 import { component } from 'https://unpkg.com/haunted/haunted.js';

 customElements.define('my-comp',
   component( () => html `<button @click=${() => alert('Thx!')}>Click Me!</button>` ));

</script>
```

wandelt eine Funktion in eine Komponente um

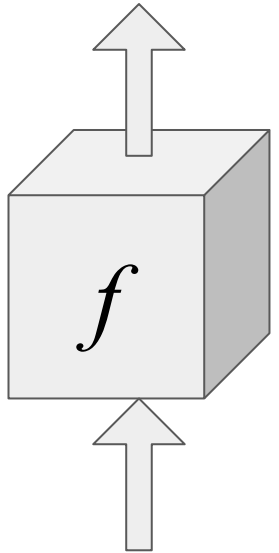
https://github.com/matthewp/haunted

Function Components und React Hooks sind eine Neuerung in **React 16.8**

React Components beruhen auf einem komplexen Lifecycle Modell.



React Function Component sind viel einfacher: Sie werden bei Bedarf einfach neu aufgerufen.



https://serverless-stack.com/chapters/understanding-react-hooks.html

Ziel: User Interface als Funktion

$$UI = f(\ data\ )$$

Die App ist eine Funktion.

Sie liefert mit Daten bestücktes HTML. Darin können Komponenten zum Einsatz kommen, die selbst wiederum nur Funktionen sind, die mit Daten bestücktes HTML liefern.

Dieser Ansatz wird z.Zt. mit React Hooks populär.

```
import {
 component,
 useState,
 html
} from "https://unpkg.com/haunted/haunted.js";

import "./FullName.js";

function App() {
 const [name, setName] = useState("");

 return html`
   <h2>User Page</h2>

   <h3>${name}</h3>

   <p>Change name:</p>
   <full-name @change="${ev => setName(ev.detail)}"> </full-name>
 `;
}

customElements.define("my-app", component(App));
```

https://github.com/matthewp/haunted-starter-app

Komponenten als Funktionen

```
import { component, html, useState } from "https://unpkg.com/haunted/haunted.js";

function dispatch(el, first, last) {
 let event = new CustomEvent("change", {
   detail: first + " " + last
 });
 el.dispatchEvent(event);
}

function FullName(el) { … }

customElements.define("full-name", component(FullName));
```

```
function FullName(el) {
 const [first, setFirst] = useState("Happy");
 const [last, setLast] = useState("Halloween 🎃");

 dispatch(el, first, last);

 return html`
   <div class="container">
    <label for="first">First</label>
    <input
     value="${first}"
     @keyup="${ev => setFirst(ev.target.value)}"
     type="text"
     name="first"
    />

    <label for="last">Last</label>
    <input
     value="${last}"
     @keyup="${ev => setLast(ev.target.value)}"
     type="text"
     name="last"
    />
   </div>
 `;
}
```

https://github.com/matthewp/haunted-starter-app

State Management with Hooks in Function Components

```
<my-counter></my-counter>

<script type="module">
 import { html } from 'https://unpkg.com/lit-html/lit-html.js';
 import { component, useState } from 'https://unpkg.com/haunted/haunted.js';

 function Counter() {
   const [count, setCount] = useState(0);

   return html`
     <div id="count">${count}</div>
     <button type="button" @click=${() => setCount(count + 1)}>Increment</button>
   `;
 }

 customElements.define('my-counter', component( Counter ));
</script>
```

0 is the initial value of count

ES6 Array Destructuring Assignment

useState(init) ist ein Hook, mit dem State Management in Funktionskomponenten eingeführt wird.

Mit setCount(newValue) wird der Zustand verändert.

https://reactjs.org/docs/hooks-reference.html#usestate

https://github.com/matthewp/haunted

State Management in WebApps

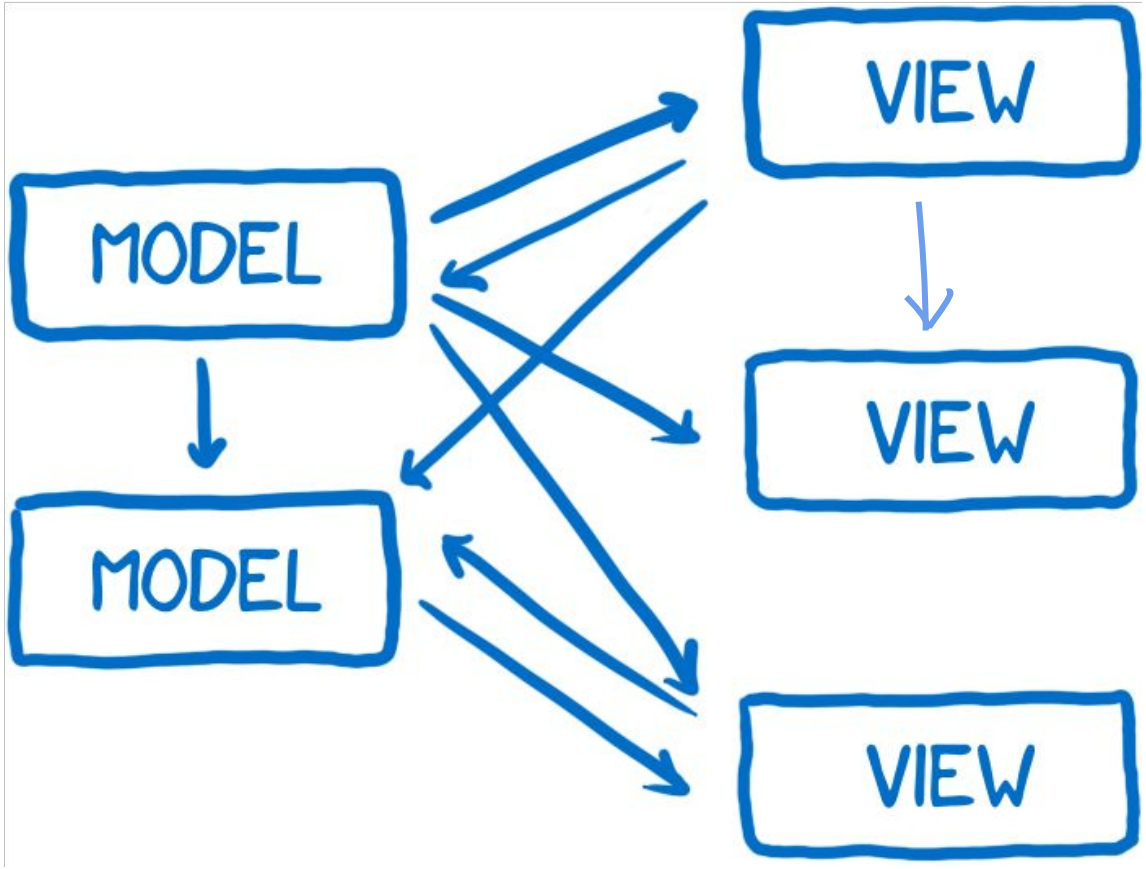
1. "Freies" dezentrales State Management

   Jede Komponente ist eine Instanz einer Klasse und kann ihren Zustand selbst verwalten.

   ⇒ Eine WebApp besteht aus einem Netzwerk vieler Komponenten, die miteinander kommunizieren

2. Das State-Action-Model (SAM) - Pattern
3. Hooks
4. Redux

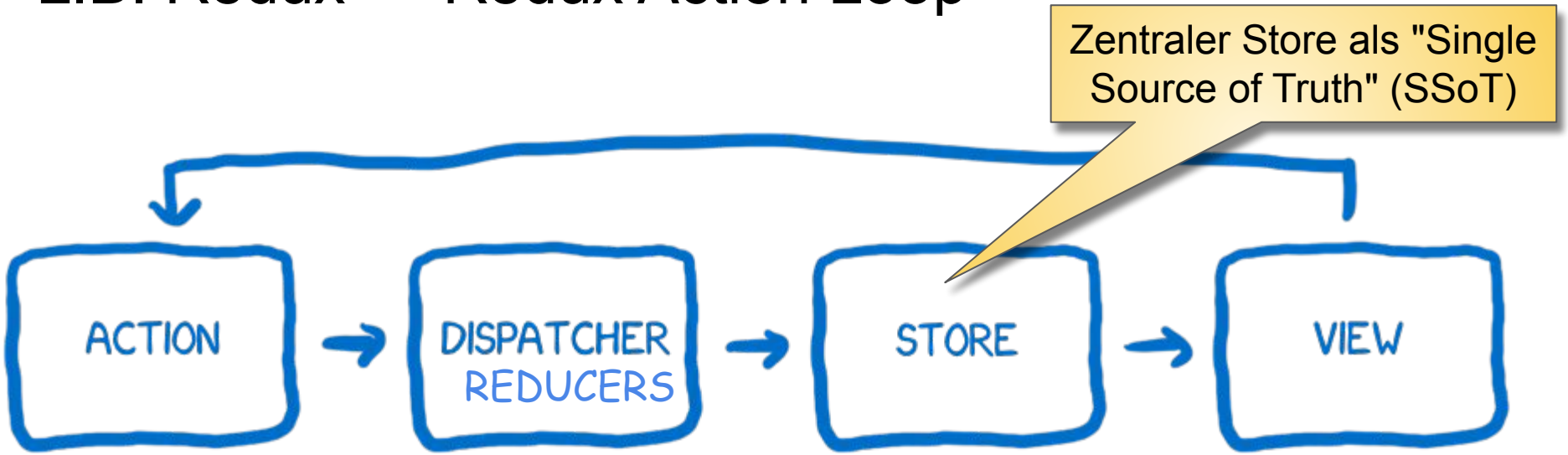
State Management als Lösung für das Update-Chaos-Problem



https://code-cartoons.com/a-cartoon-guide-to-flux-6157355ab207

Die Lösung: Zentralisierung + Unidirektionaler Datenfluss

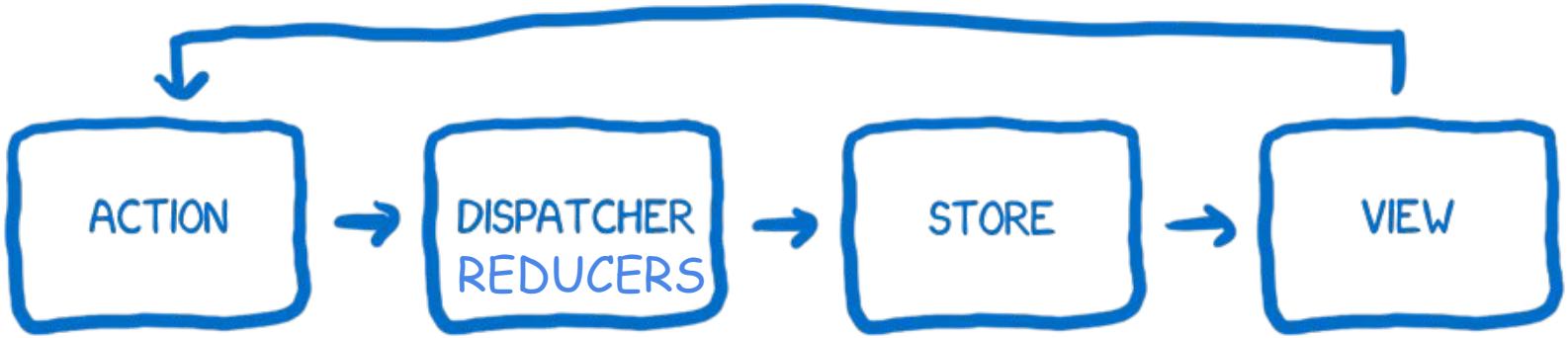
z.B. Redux ⇒ Redux Action Loop



- Zustand der WebApp nicht mehr auf viele Komponenten und Views verteilt, sondern zentral im Store gespeichert
- Die Views und Models kommunizieren nicht mehr direkt, sondern geben alle Kommunikation an den Dispatcher weiter, der einen Reducer wählt, mit dem der neue Zustand berechnet und im Store gespeichert wird.

https://code-cartoons.com/a-cartoon-intro-to-redux-3afb775501a6

Redux Bestandteile

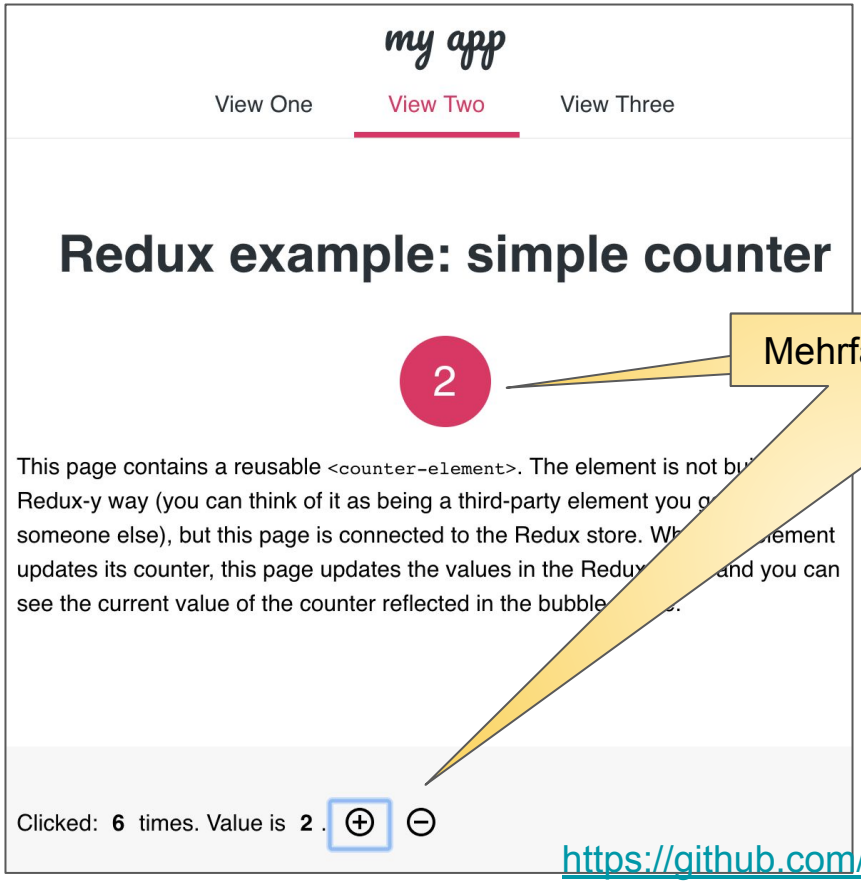


- Actions: What happened?
- Reducers: How to compute new state?
- Store: holds application state. `const store = createStore(todoApp)`
- Data Flow: unidirectional ⇒ one-way data binding

reducers

https://code-cartoons.com/a-cartoon-intro-to-redux-3afb775501a6

Redux - Beispiel 1: Counter



https://github.com/Polymer/pwa-starter-kit

Custom Element <counter-element>

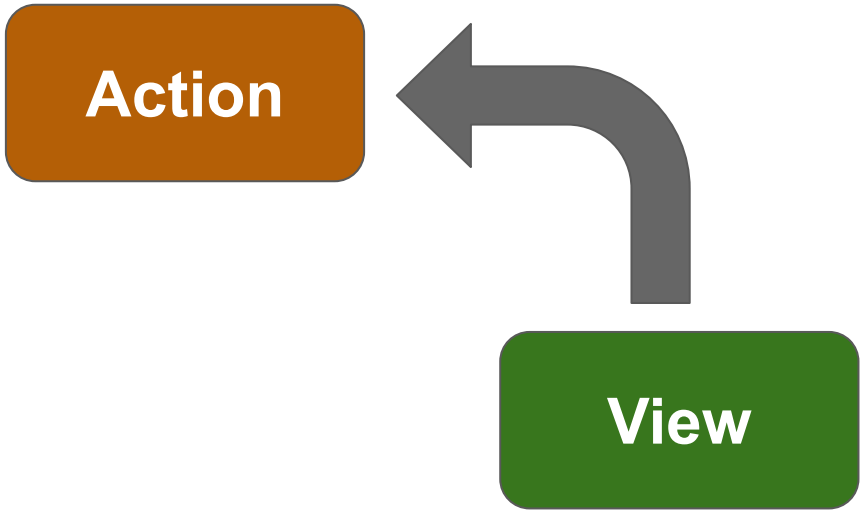
```
class CounterElement extends LitElement {
  static get properties() {
   return {
     /* The total number of clicks you've done. */
     clicks: { type: Number },
     /* The current value of the counter. */
     value: { type: Number }
  }
}

render() {
 return html`
   <div>
     <p>
       Clicked: <span>${this.clicks}</span> times.
       Value is <span>${this.value}</span>.
       <button @click="${this._onIncrement}"
            title="Add 1">${plusIcon}</button>
       <button @click="${this._onDecrement}"
           title="Minus 1">${minusIcon}</button>
     </p>
   </div>
 `;
}
```

```
constructor() {
 super();
 this.clicks = 0;
 this.value = 0;
}

 _onIncrement() {
   this.value++;
   this.clicks++;
   this.dispatchEvent(new CustomEvent('counter-incremented'));
 }
 _onDecrement() {
   this.value--;
   this.clicks++;
   this.dispatchEvent(new CustomEvent('counter-decremented'));
 }
}
window.customElements.define('counter-element', CounterElement);
```

Benachrichtigung der App über Events

https://github.com/Polymer/pwa-starter-kit/blob/master/src/components/counter-element.js

1. Redux Actions: Aus Methodenaufrufen werden Objekte

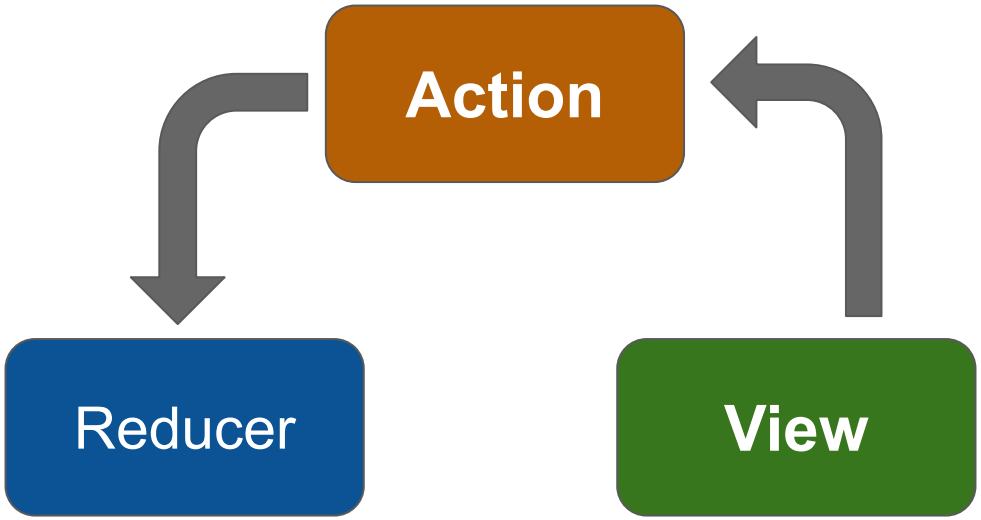



```
export const INCREMENT = 'INCREMENT';
export const DECREMENT = 'DECREMENT';

export const increment = () => {
 return {
   type: INCREMENT
 };
};

export const decrement = () => {
 return {
   type: DECREMENT
 };
};
```

https://github.com/Polymer/pwa-starter-kit/blob/master/src/actions/counter.js

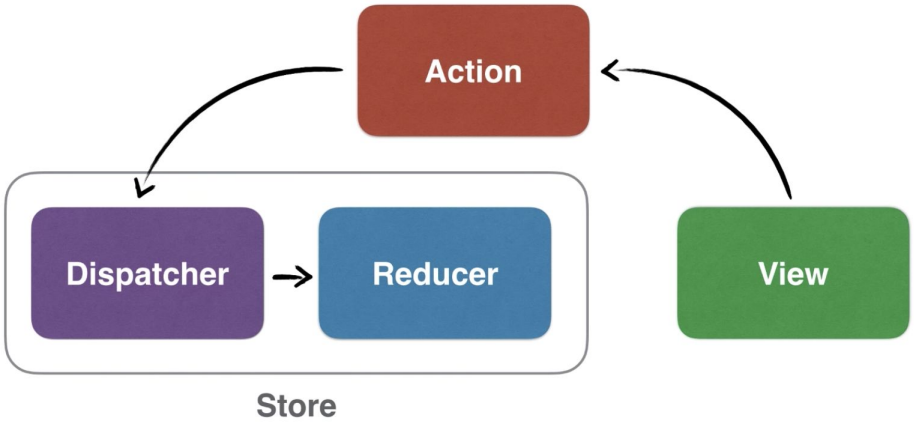
2. Redux Reducer: Old State → New State

```
import { INCREMENT, DECREMENT } from '../actions/counter.js';

const INITIAL_STATE = { clicks: 0, value: 0 };

const counter = (state = INITIAL_STATE, action) => {
 switch (action.type) {
  case INCREMENT:
   return {
    clicks: state.clicks + 1,
    value: state.value + 1
   };
  case DECREMENT:
   return {
    clicks: state.clicks + 1,
    value: state.value - 1
   };
  default:
   return state;
 }
};
```



https://github.com/Polymer/pwa-starter-kit/blob/master/src/reducers/counter.js

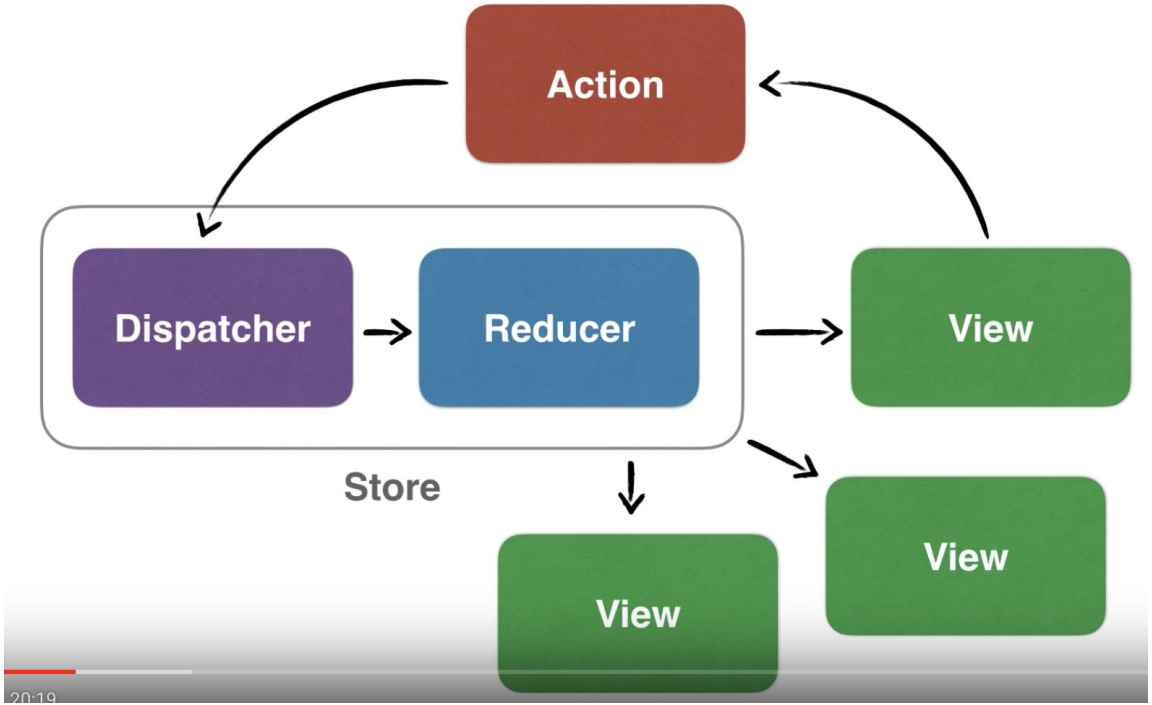
3. Redux Dispatcher: transforms View Action into Store Action, i.e. Append new State




```
const increment = () => {
 return {
   type: INCREMENT
 };
};

const decrement = () => {
 return {
   type: DECREMENT
 };
};

_counterIncremented() {
 store.dispatch( increment() );
}

_counterDecremented() {
 store.dispatch( decrement() );
}
```

https://github.com/Polymer/pwa-starter-kit/blob/master/src/components/my-view2.js

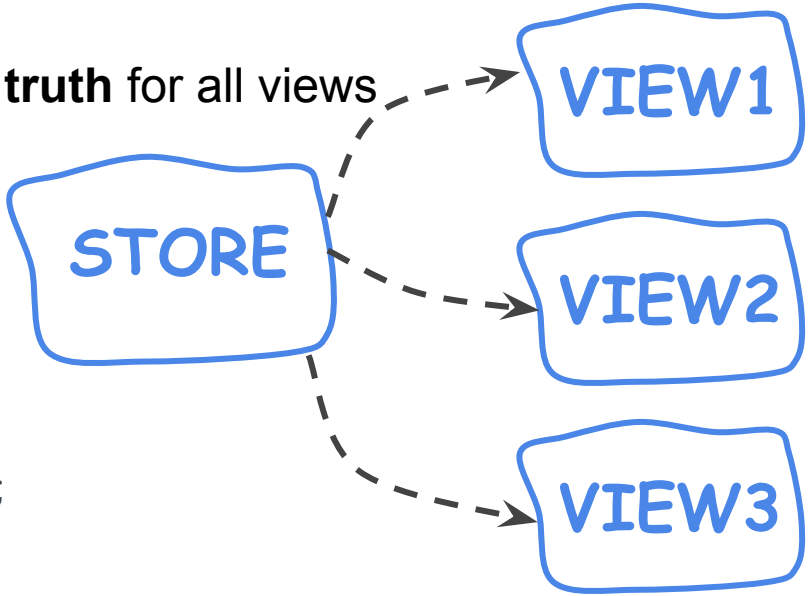
4. Redux Store: Views updaten mit neuem State




```
// This is called every time something is updated in the store.
stateChanged(state) {
 this._clicks = state.counter.clicks;
 this._value = state.counter.value;
}
```


https://www.youtube.com/watch?v=PahsgJn0sgU&t=218s

Redux Store is the **single source of truth** for all views



The **Store** has the following responsibilities:

- Holds application state;
- Allows access to state via `getState()`;
- Allows state to be updated via `dispatch(action)`;
- Registers listeners via `subscribe(listener)`;
- Handles unregistering of listeners via the function returned by `unsubscribe(listener)`.

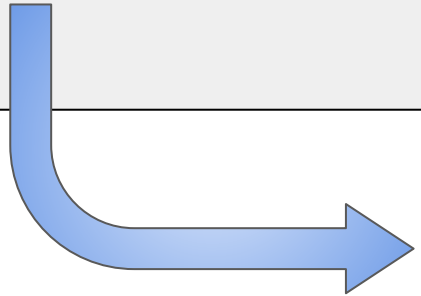
Observer-Pattern

You'll **only have a single store in a Redux application**.

https://redux.js.org/basics/store

Store ⇒ dispatch(Action) ⇒ Reducer ⇒ update

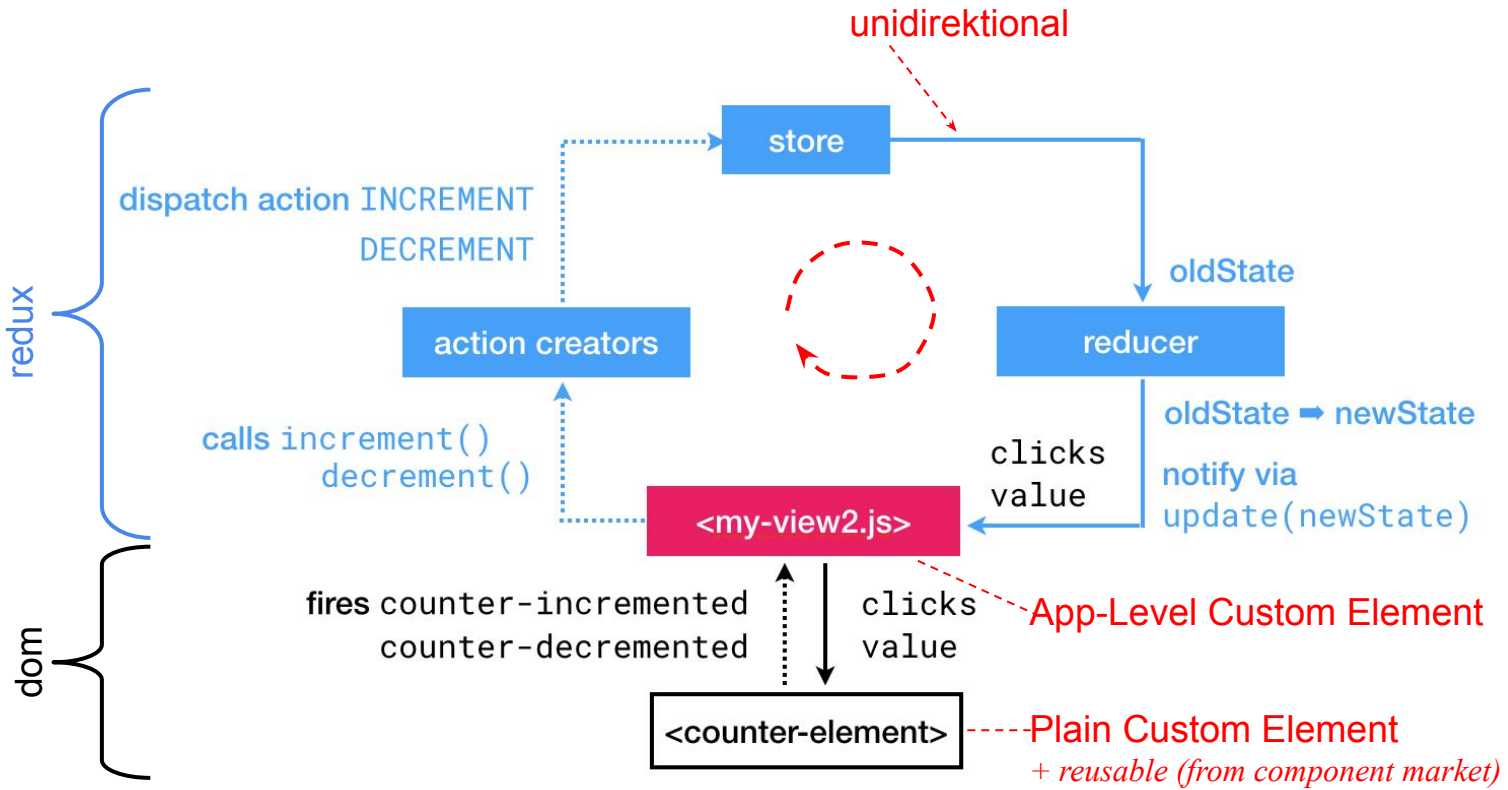

```
<counter-element value="${store.getState().counter.value}" clicks="${store.getState().counter.clicks}"
        @click="${() => store.dispatch( increment() )}"
        @click="${() => store.dispatch( decrement() )}">
</counter-element>
```



Action

https://pwa-starter-kit.polymer-project.org/redux-and-state-management

redux Action Loop: Beispiel Counter



https://github.com/Polymer/pwa-starter-kit/wiki/4.-Redux-and-state-management

Der Redux Reducer erzeugt immer neue Objekte

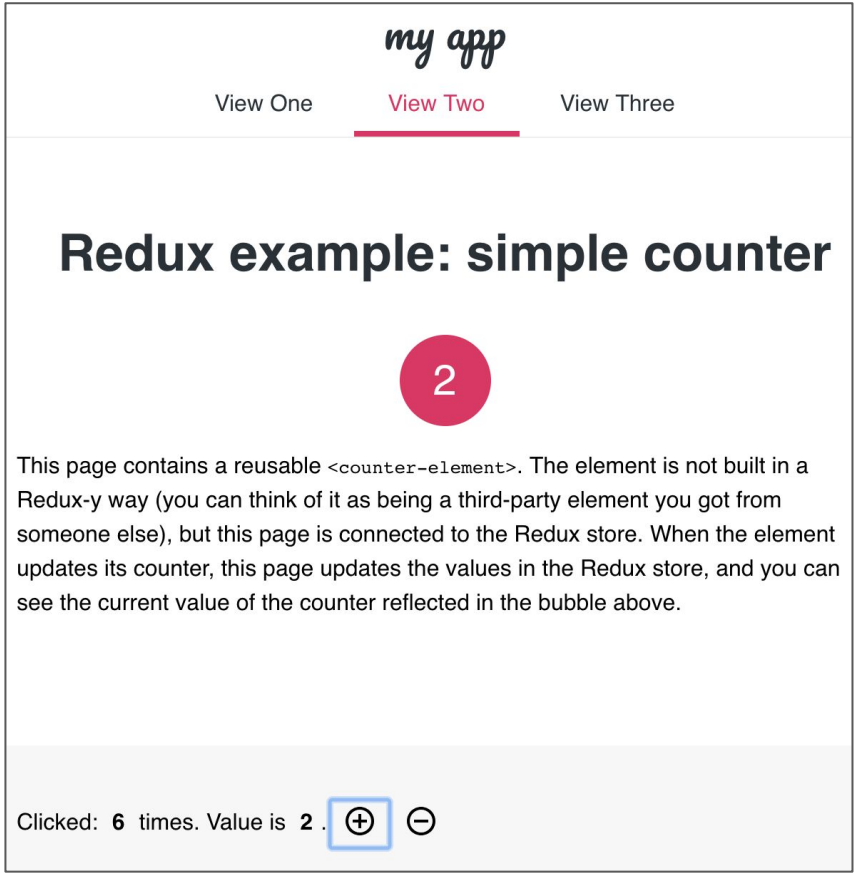
```
const counter = (state = {clicks: 0, value: 0}, action) => {
 switch (action.type) {
  case INCREMENT:
   return {
    'clicks': Object.assign({}, state, { clicks: state.clicks + 1 }),
    'value': Object.assign({}, state, { value: state.value + 1 })
   }; …….
  default:
   return state;
 }
}
```

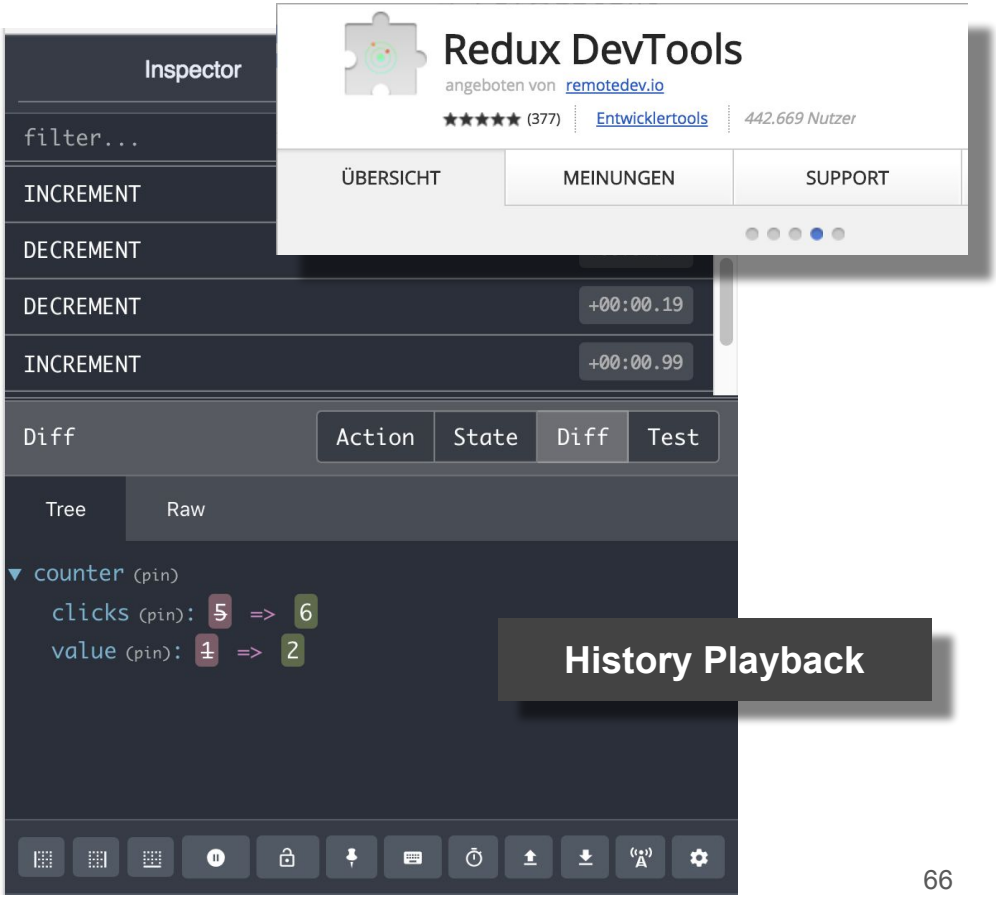
1. **We don't mutate the** `state`**.** We create a copy with `Object.assign()`. `Object.assign(state, { clicks: state.clicks + 1 })` is also wrong: it will mutate the first argument. You **must** supply an empty object as the first parameter. You can also enable the object spread operator proposal to write `{ ...state, ...newState }` instead.

2. **We return the previous** `state` **in the** `default` **case.** It's important to return the previous `state` for any unknown action.

https://redux.js.org/basics/reducers

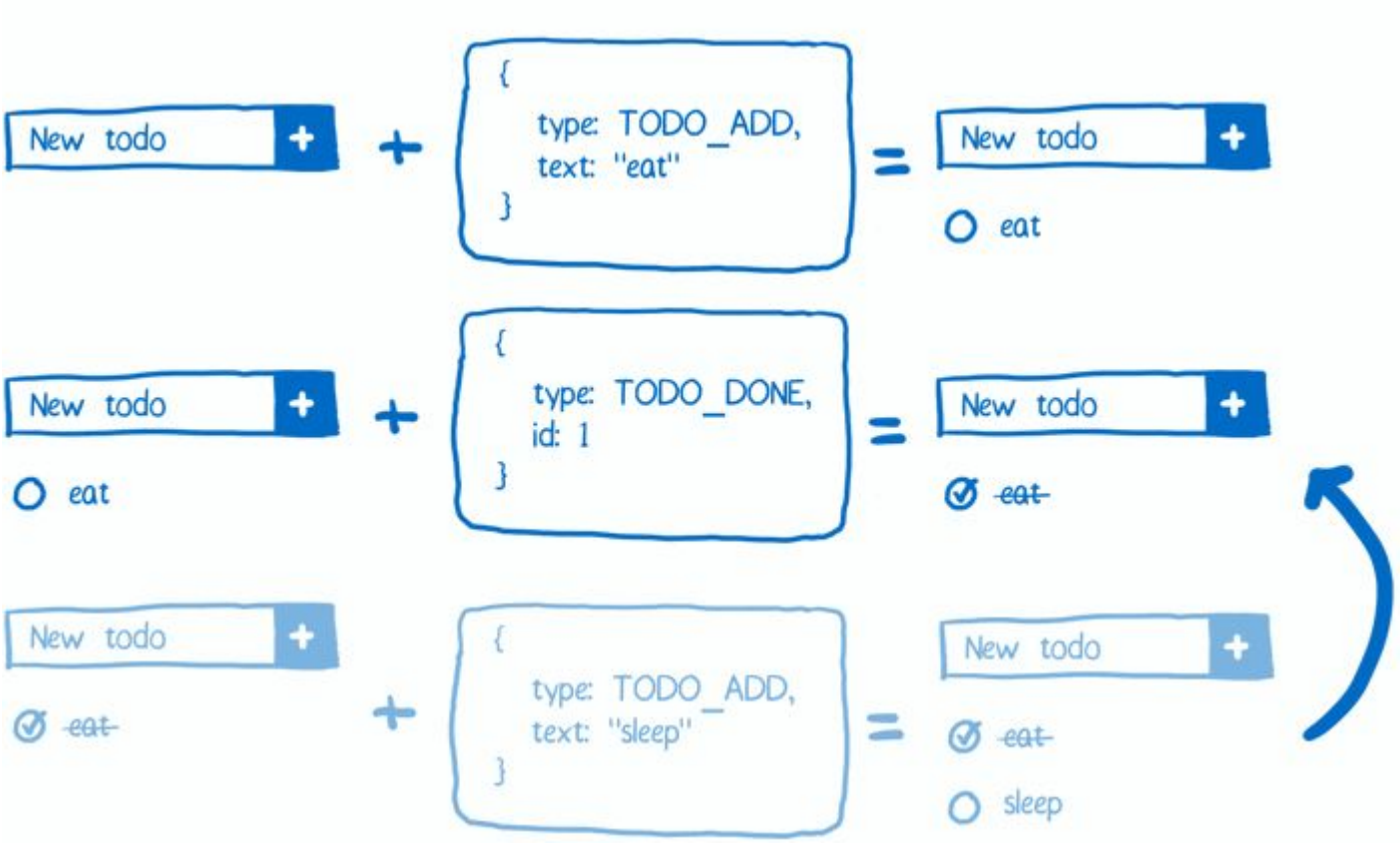
Chrome Extension: Redux DevTools





https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=de

Time Travel Debugging



https://code-cartoons.com/hot-reloading-and-time-travel-debugging-what-are-they-3c8ed2812f35

Detail: store connect mixin

```
import { PageViewElement } from './page-view-element.js';
import { connect } from 'pwa-helpers/connect-mixin.js';
// This element is connected to the Redux store.
import { store } from '../store.js';
class MyView2 extends connect(store)(PageViewElement) {
 // ...
}
```

app-level element

ES6 mixin
with parameter

redux store

Exkurs: ES6 Mix-ins

- Ein Mixin ist ein Funktion, deren Rückgabewert eine Klasse ist, die sich aus einem Klassenausdruck berechnet
- Flexible Komposition von Verhalten
- Teil von ES6
- kein Framework erforderlich
- nicht an Vererbungshierarchien gebunden

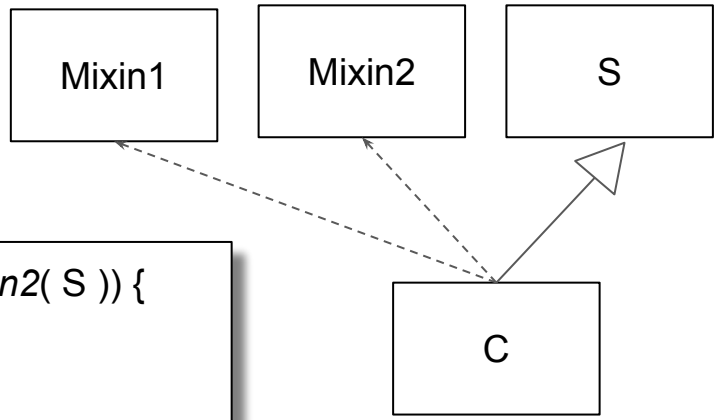
```
const Storage = Sup => class extends Sup {
    save(database) { ··· }
};
const Validation = Sup => class extends Sup {
    validate(schema) { ··· }
};
```

```
class Person { ··· }
class Employee extends Storage(Validation(Person)) { ··· }
```

http://2ality.com/2016/05/six-nifty-es6-tricks.html
http://exploringjs.com/es6/ch_classes.html#_simple-mixins

Wdh. Multiple Inheritance
Mixins: Klassen als Parameter & Return Value

```
let Mixin1 = (superclass) => class extends superclass {
   foo() {
     console.log('foo from Mixin1');
     if (super.foo) super.foo();
   }
 };

let Mixin2 = (superclass) => class extends superclass {
 foo() {
   console.log('foo from Mixin2');
   if (super.foo) super.foo();
 }
};

class S {
 foo() {
   console.log('foo from S');
 }
}
```




```
class C extends Mixin1( Mixin2( S )) {
 foo() {
   console.log('foo from C');
   super.foo();
 }
}

new C().foo();

// foo from C
// foo from Mixin1
// foo from Mixin2
// foo from S
```

http://justinfagnani.com/2015/12/21/real-mixins-with-javascript-classes/

```
// Functional mixin
function GreetMixin(Base) {
  return class Greet extends Base {
    greet() {
      alert('Hello');
    }
  };
}

// Define a web component that uses the mixin.
class GreetElement extends GreetMixin(HTMLElement) {}
customElements.define('greet-element', GreetElement);

// Instantiate the component.
const element = new GreetElement();
element.greet(); // "Hello"
```

ELIX

Elix is our open source collection of common user interface patterns like lists, menus, popups, and carousels.

Microsoft-Projekt Elix
https://component.kitchen/

```
const ListBoxBase =
  AriaListMixin(
  AttributeMarshallingMixin(
  ComposedFocusMixin(
  ContentItemsMixin(
  DirectionSelectionMixin(
  FocusVisibleMixin(
  ItemsTextMixin(
  KeyboardDirectionMixin(
  KeyboardMixin(
  KeyboardPagedSelectionMixin(
  KeyboardPrefixSelectionMixin(
  LanguageDirectionMixin(
  ReactiveMixin(
  RenderUpdatesMixin(
  SelectedItemTextValueMixin(
  SelectionInViewMixin(
  ShadowTemplateMixin(
  SingleSelectionMixin(
  SlotContentMixin(
  TapSelectionMixin(
    HTMLElement
  ))))))))))))))))))));
```

```
const MenuBase =
  AriaMenuMixin(
  AttributeMarshallingMixin(
  ContentItemsMixin(
  DelegateFocusMixin(
  DirectionSelectionMixin(
  FocusVisibleMixin(
  ItemsTextMixin(
  KeyboardDirectionMixin(
  KeyboardMixin(
  KeyboardPagedSelectionMixin(
  KeyboardPrefixSelectionMixin(
  LanguageDirectionMixin(
  ReactiveMixin(
  RenderUpdatesMixin(
  SelectedItemTextValueMixin(
  SelectionInViewMixin(
  ShadowTemplateMixin(
  SingleSelectionMixin(
  SlotContentMixin(
  TapSelectionMixin(
    HTMLElement
  ))))))))))))))))))));
```

Microsoft-Projekt Elix
https://component.kitchen/

Zusammenfassung:
Was fehlt dem WWW zur App-Plattform?

- Application Architecture
- Multi-Threading
- Routing
- State Management
- Dependency Injection

Kann man das mit den Mitteln des WWW nachbauen?

Referenzen Redux

- https://redux.js.org/
- https://codesandbox.io/s/github/reduxjs/redux/tree/master/examples/todos
- https://code-cartoons.com/a-cartoon-intro-to-redux-3afb775501a6
- https://github.com/Polymer/pwa-starter-kit/wiki/4.-Redux-and-state-management
- https://github.com/reduxjs/redux
- https://egghead.io/lessons/react-redux-describing-state-changes-with-actions
- https://github.com/arqex/freezer