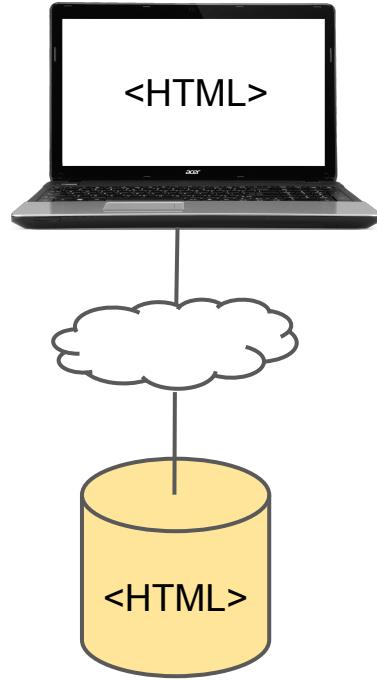


WebApps: Das WWW als App-Plattform

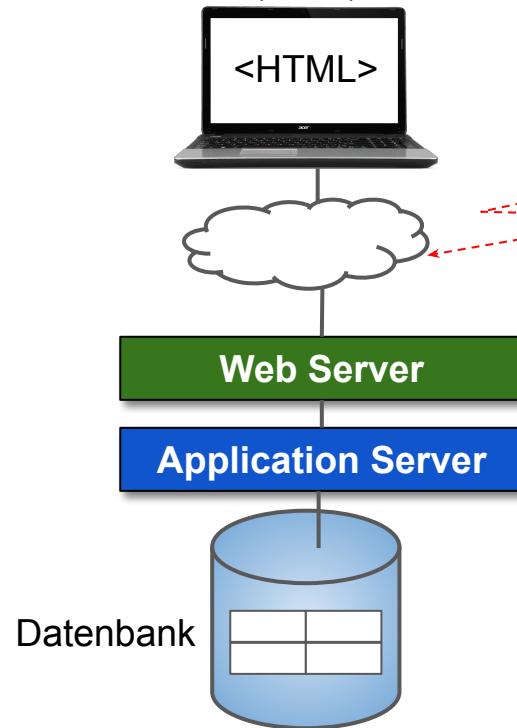
UX - User Experience
DX - Developer Experience

Das WWW war als Hypertext gedacht ...

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



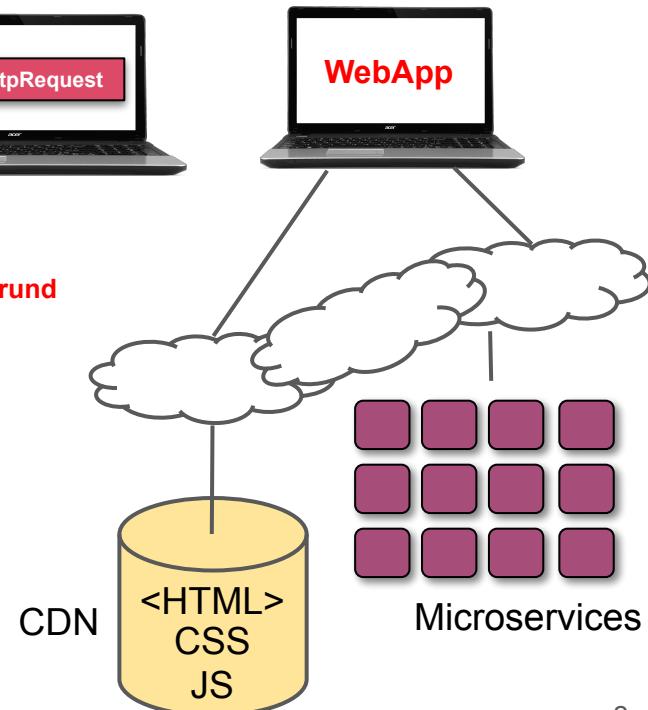
ab 1997
Server-side Applications
mit >3-Schichten-Architektur
(MVC)



ab 2000
AJAX

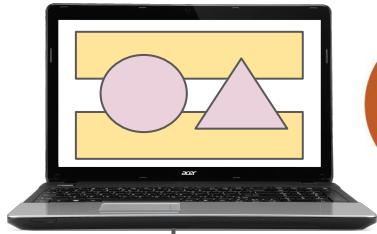


heute
Client-side Applications
mit Microservices

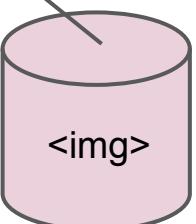
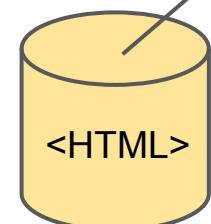


Der Kategorienumbruch des WWW vom Hypertext zur WebApp

Digital HyperMedia Library



Beruf: Medien Designer



Beruf: Software-Ingenieur

CDN

Events

Verteilte, Event-getriebene
Software

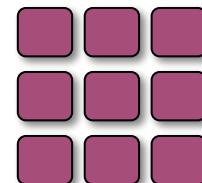
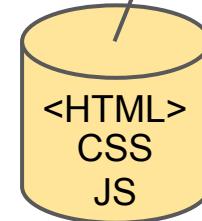


Events

Events

Events

Initialisierung
mit statischen
Dateien



Microservices

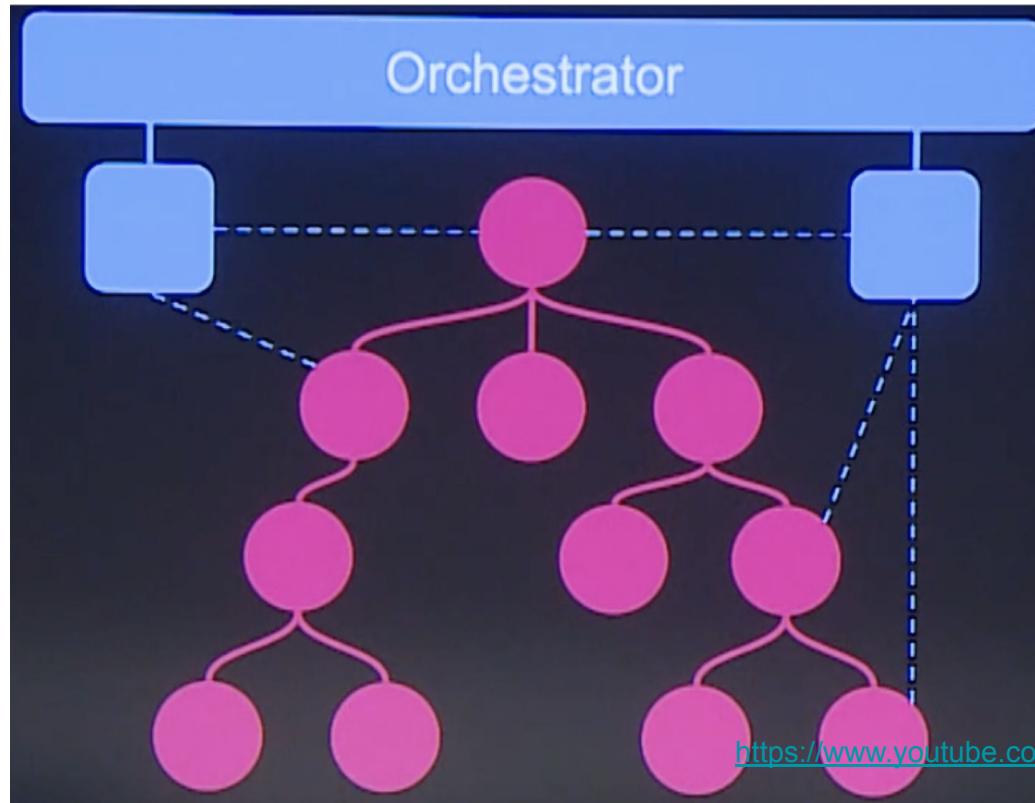
Was fehlt dem WWW zur App-Plattform?

- 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?



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>
```

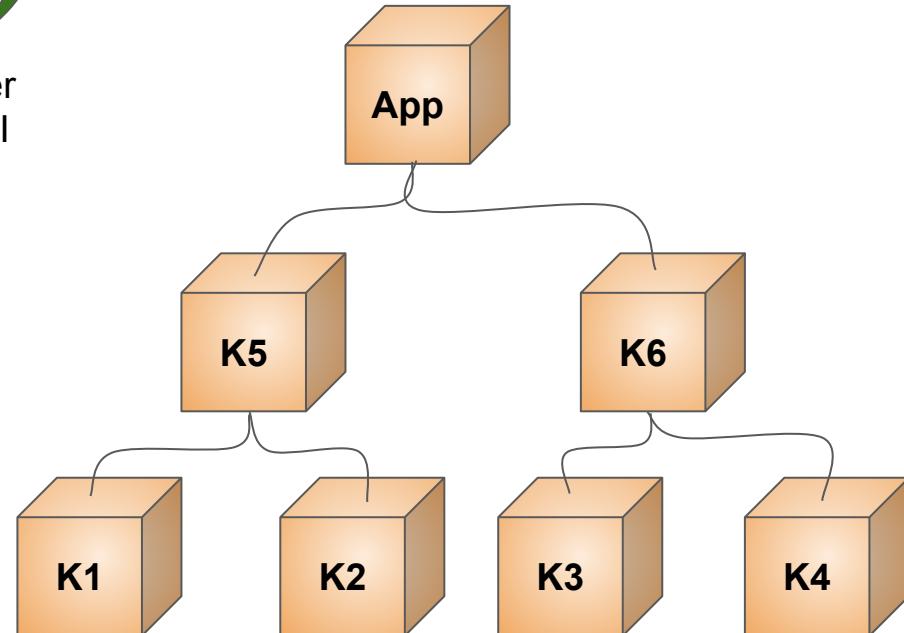
<https://www.youtube.com/watch?v=x9YDQUJx2uw>

Komponenten-Ansatz: App als Komponente

$$\text{UI} = f(\text{data})$$

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);
```

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);
```

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;
  }

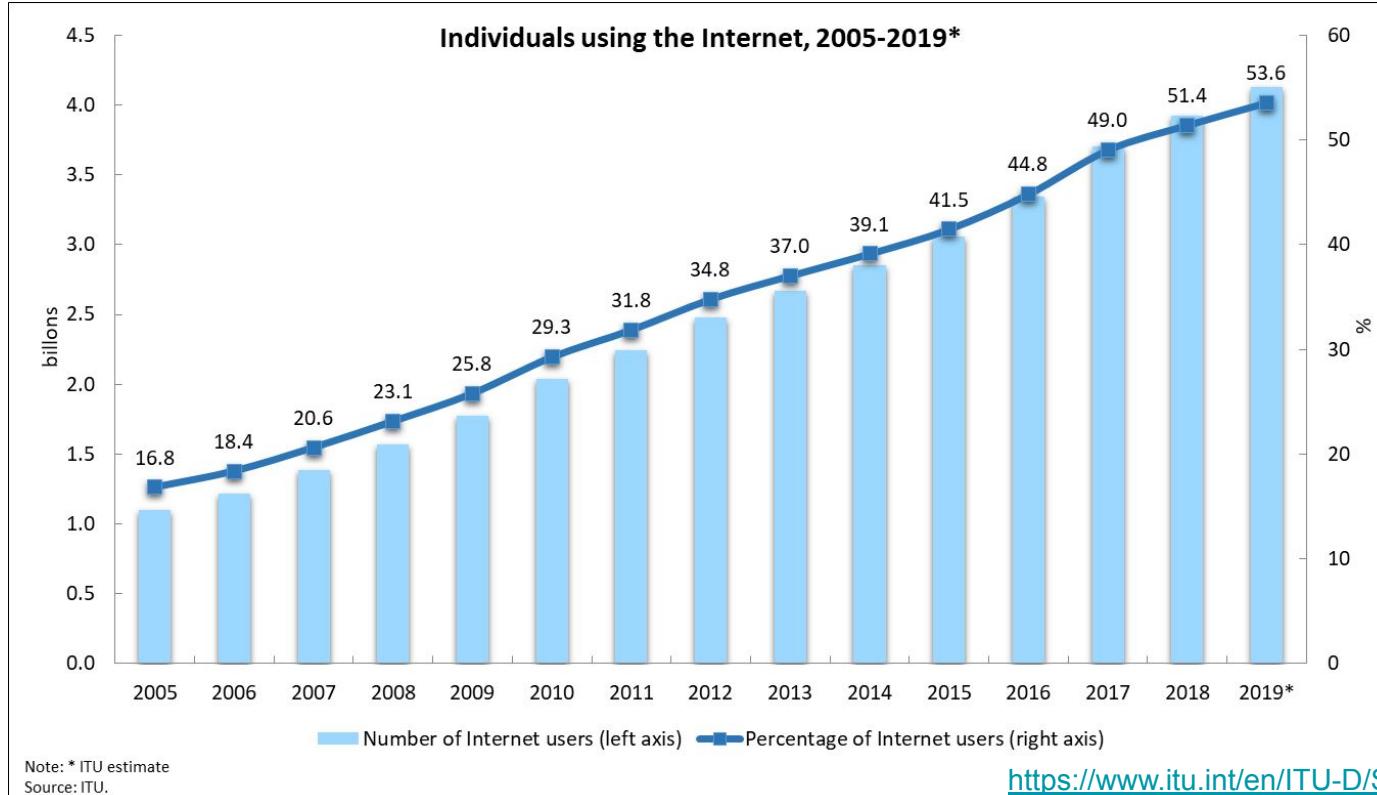
  get counter() {
    return this._counter;
  }

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

Comlink.expose(MyClass);
```

Performanz ist das nächste Großprojekt im Web! Warum? Internet Users 2005 - 2019.

Heute ~ 53%



Next Billion Users (NBU) on the Internet



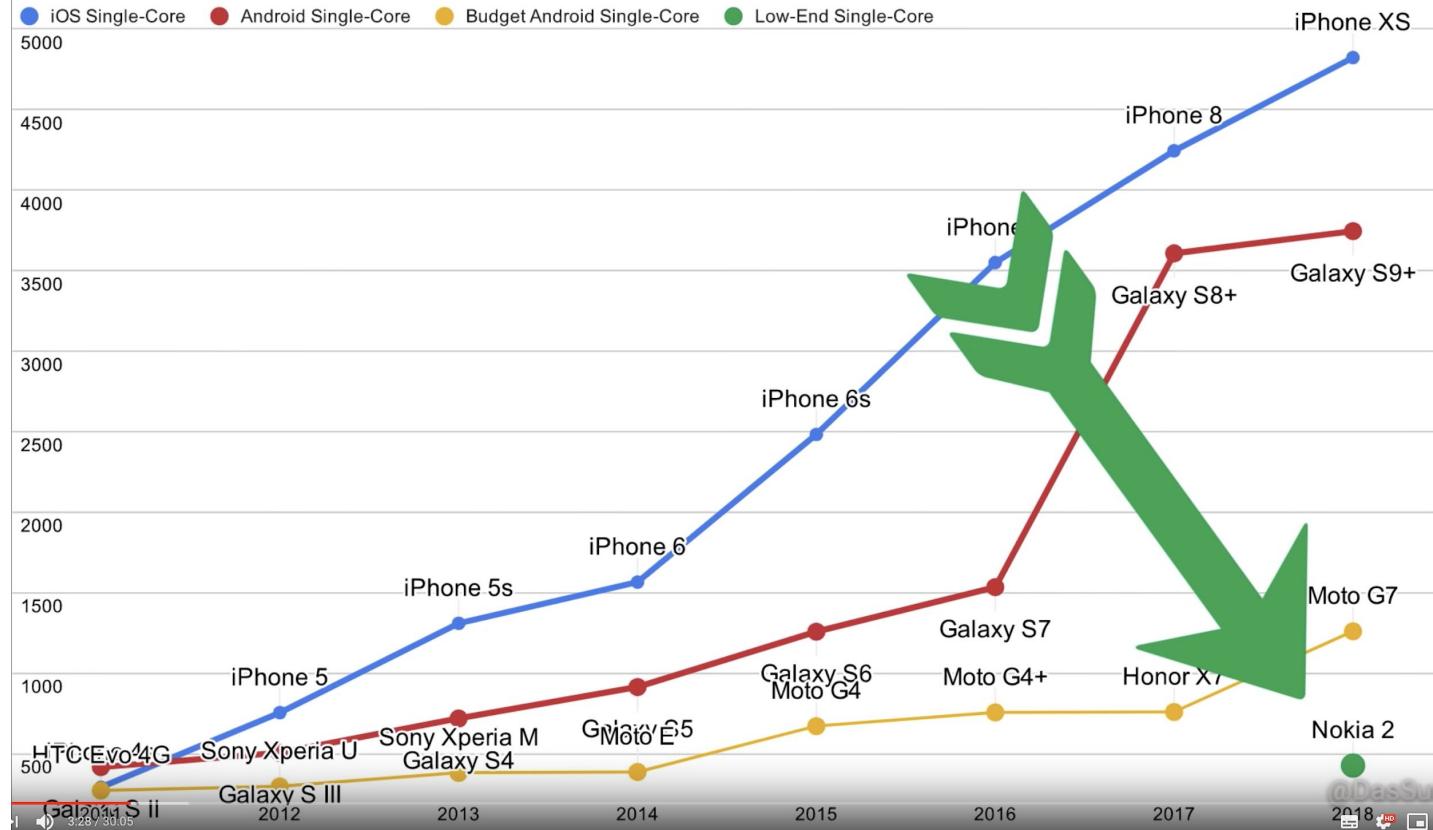
Feature Phones: 45€

90€ Nokia 2

A screenshot of a web page from the Aldi Talk online store. The top navigation bar includes the Aldi logo and links for "Tarifdetails", "Handys & Tablets", "Hilfe & Service", and "SIM-Karte aktivieren". Below the navigation, a breadcrumb trail shows the product path: "Startseite" > "Handys & Tablets" > "Smartphones" > "Nokia Smartphones" > "2.2 Schwarz, 16 GB". On the right side, there is a large image of the Nokia 2 smartphone in black, showing its front-facing camera and screen. To the left of the main image, there is a vertical thumbnail gallery showing other angles of the phone. At the bottom right, there is a summary box for the "Nokia 2.2" model, stating "Schwarz / 16 GB" and "Farbe ● Spez. verfügbar". There is also a small note about "Gratis ALDI TALK".

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

Geekbench Single-Core Scores



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

LitElement doppelt so schnell wie React, Vue

React To Do App

Performance

Metrics

First Contentful Paint	1,200 ms	✓	First Meaningful Paint	1,560 ms	✓
Speed Index	1,400 ms	✓	First CPU Idle	1,560 ms	✓
Time to Interactive	1,530 ms	✓	Estimated Input Latency	16 ms	✓

[View Trace](#)

Values are estimated and may vary.



Vue To Do App

Performance

Metrics

First Contentful Paint	1,190 ms	✓	First Meaningful Paint	1,500 ms	✓
Speed Index	1,420 ms	✓	First CPU Idle	1,500 ms	✓
Time to Interactive	1,470 ms	✓	Estimated Input Latency	16 ms	✓

[View Trace](#)

Values are estimated and may vary.



No HTTP2 was used in the running of the Lighthouse Audit, however at 5 total requests, there shouldn't have been much gains to be had there.

LitElement To Do App

Performance

Metrics

First Contentful Paint	740 ms	✓	First Meaningful Paint	790 ms	✓
Speed Index	760 ms	✓	First CPU Idle	790 ms	✓
Time to Interactive	760 ms	✓	Estimated Input Latency	16 ms	✓

[View Trace](#)

Values are estimated and may vary.



No HTTP2 was used in the running of the Lighthouse Audit, however at 1 total requests, there shouldn't have been much gains to be had there.

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

Wer verwendet LitElement?

Who's Using LitElement & lit-html?

- IBM
- Microsoft
- ING
- BBVA
- SAP
- Rabobank
- Nordea Bank
- Lyft
- Netflix
- Adobe
- Maersk
- GitHub
- Chrome OS
- Williams Sonoma
- Material Design
- British Gas
- New Google Projects
- And More...

Material Design Web Components Gallery

The image shows two views of the Material Design Web Components gallery. On the left is the main homepage with sections for 'Design', 'Components', and 'Documentation'. It features a large 'Web' heading, a 'Build beautiful, usable products using Material Components for the web' tagline, and three large cards illustrating components: a card with a globe icon, a card with a grid icon, and a card with a plus sign icon. Below these are links to 'Web components', 'Web GitHub', and 'Web tutorial'. On the right is a detailed view of 'MATERIAL COMPONENTS FOR THE WEB' showing various components like Button, Card, Checkbox, Chips, Data Table, Dialog, Drawer, and Elevation.

Design Components Documentation

Web

Build beautiful, usable products using Material Components for the web

POPULAR

- Web components
- Web GitHub
- Web tutorial

Button

Card

Checkbox

Chips

Data Table

Dialog

Drawer

Elevation

OVERLINE

Headline 5

Greyhound divisively hello coldly...wonderfully...

fully

BUTTON

BUTTON

Enabled Entity

Dialog Header

Document placeholder text

GLITCH

Build a Material theme on the web

Use this interactive project to customize the color, shape, and typography of Material Components

DOCUMENTATION

Web quick start guide

First time using Material Components? Here's an overview of how to customize and implement our code to easily create web apps

\$mdc-shape

Material Design Web Component (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>
```

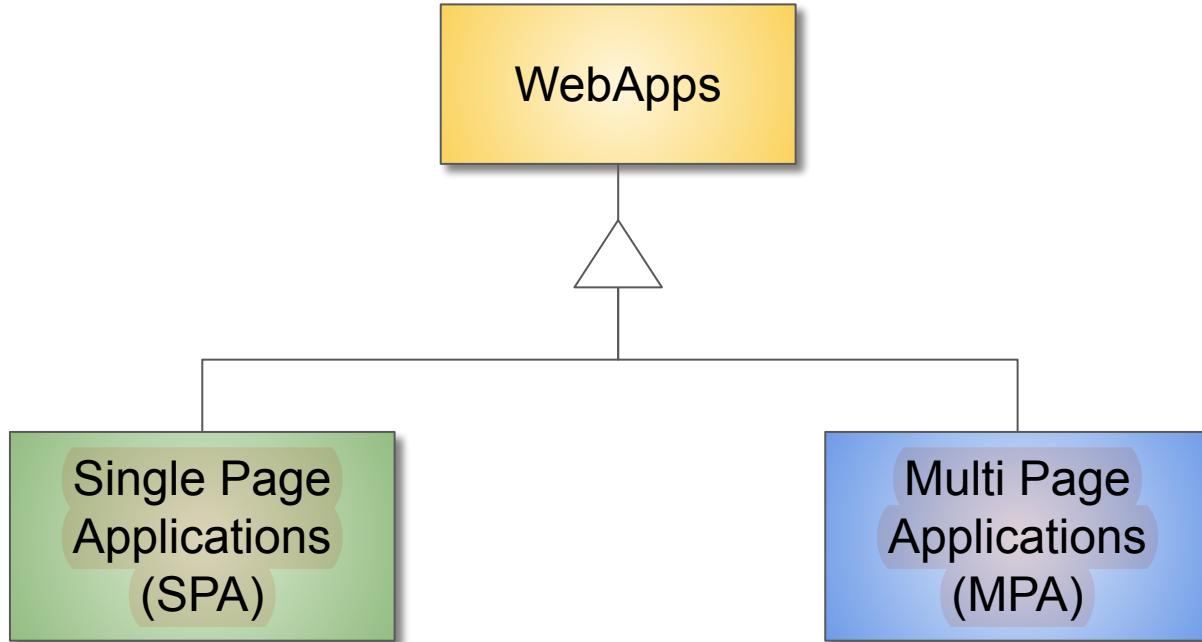
import

MARK AS READ

use

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

WebApps

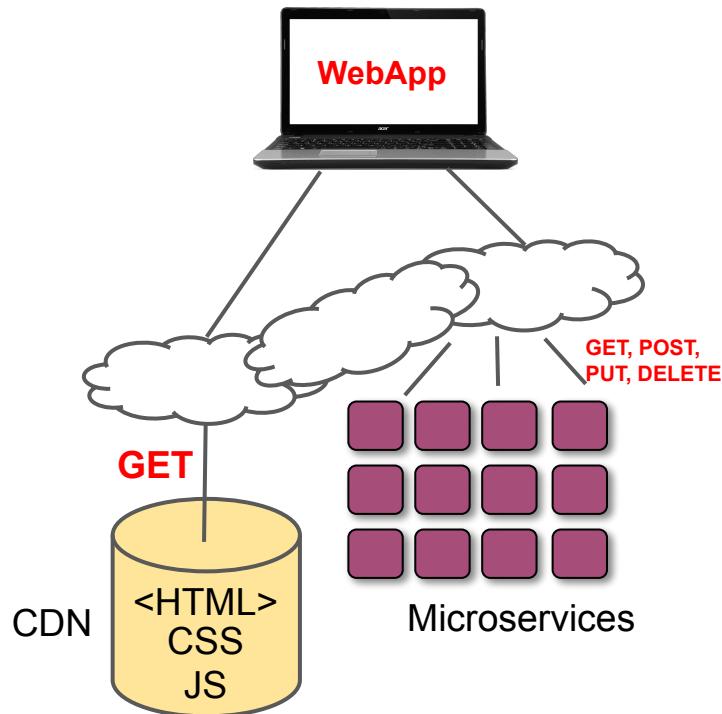


Routing in SPAs und MPAs

*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,...

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}”.

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}

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.



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

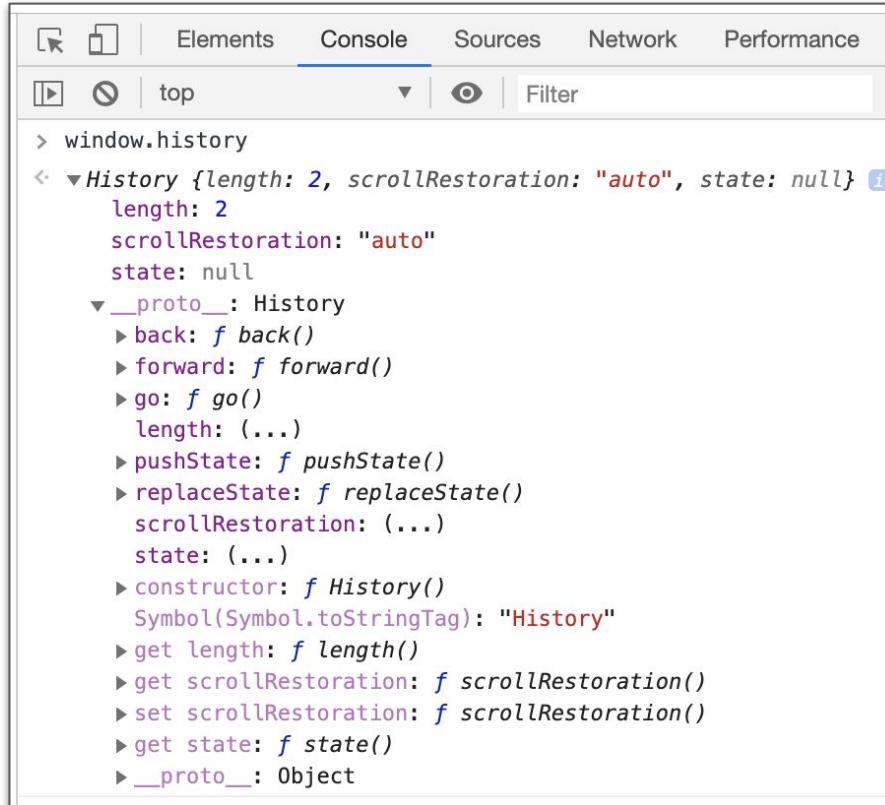
History

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



The screenshot shows the Chrome DevTools interface with the "Console" tab selected. The console output displays the state of the `window.history` object.

```
> window.history
< - History {length: 2, scrollRestoration: "auto", state: null} ⓘ
  length: 2
  scrollRestoration: "auto"
  state: null
  __proto__: History
    ▶ back: f back()
    ▶ forward: f forward()
    ▶ go: f go()
    length: (...)
    ▶ pushState: f pushState()
    ▶ replaceState: f replaceState()
    scrollRestoration: (...)
    state: (...)
    ▶ constructor: f History()
    Symbol(Symbol.toStringTag): "History"
    ▶ get length: f length()
    ▶ get scrollRestoration: f scrollRestoration()
    ▶ set scrollRestoration: f scrollRestoration()
    ▶ get state: f state()
    ▶ __proto__: Object
```

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.

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`
      <slot></slot>
    `;
  }
  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`
```

<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.

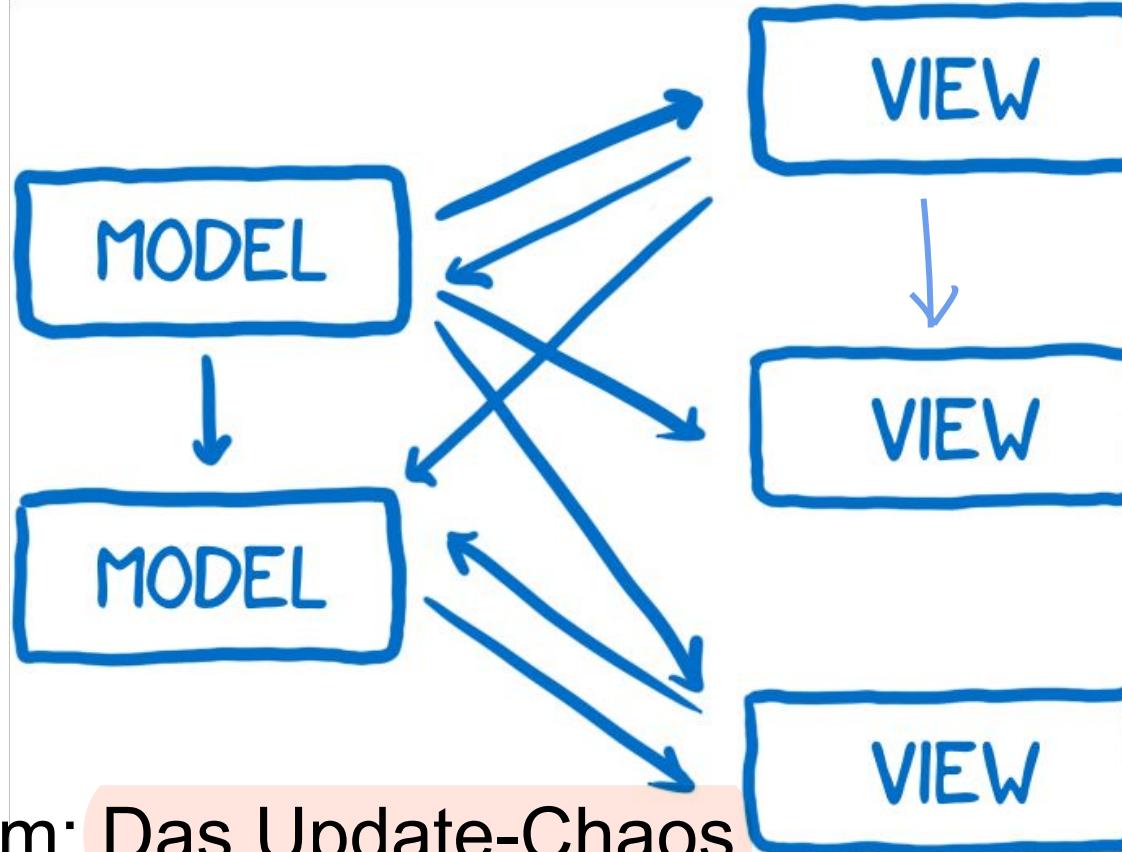
⇒ Eine WebApp besteht aus einem Netzwerk vieler Komponenten, die miteinander kommunizieren

2. Das State-Action-Model (SAM) - Pattern

3. Hooks

4. Redux

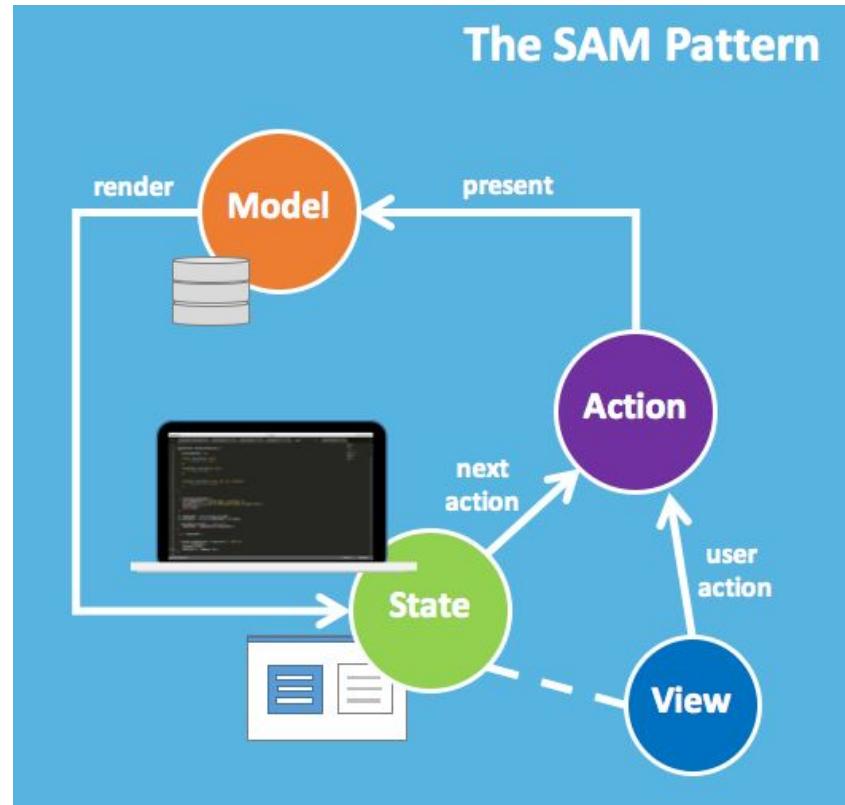
"Freies" dezentrales State Management



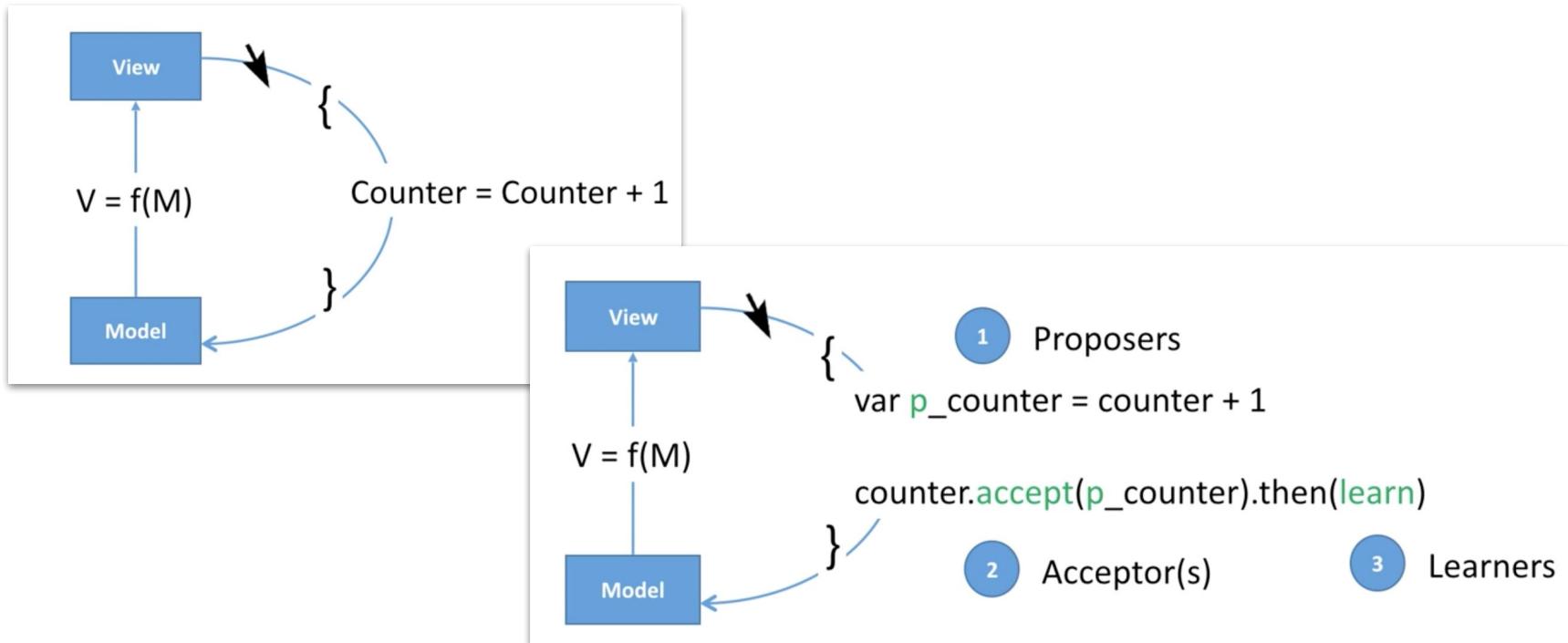
Das Problem: Das Update-Chaos

Das State-Action-Model (SAM) - Pattern

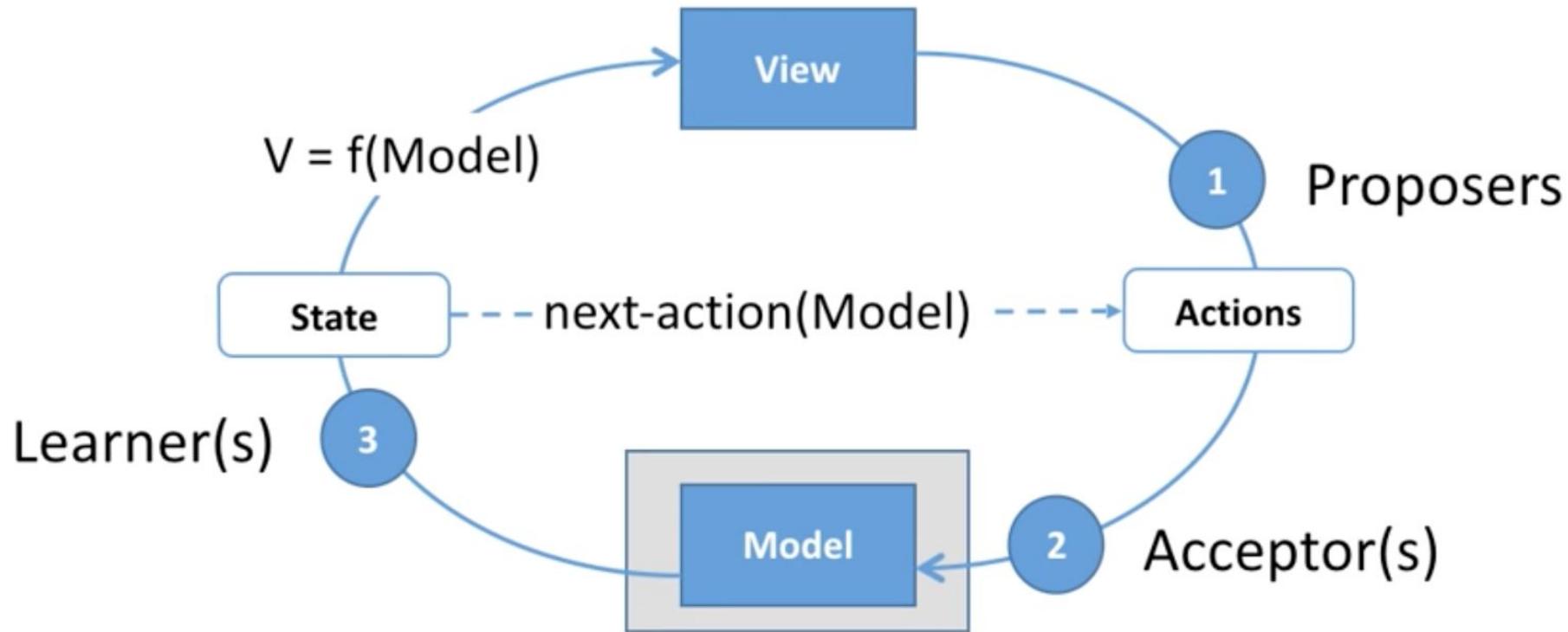
- Entkopplung von
 - State
 - Action
 - View
 - Model
- Separation of Concerns
 - Action Proposals (z.B. von Views)
 - Model accepts
 - "learn" from new state
- Prinzip: State Mutations are first-class objects



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

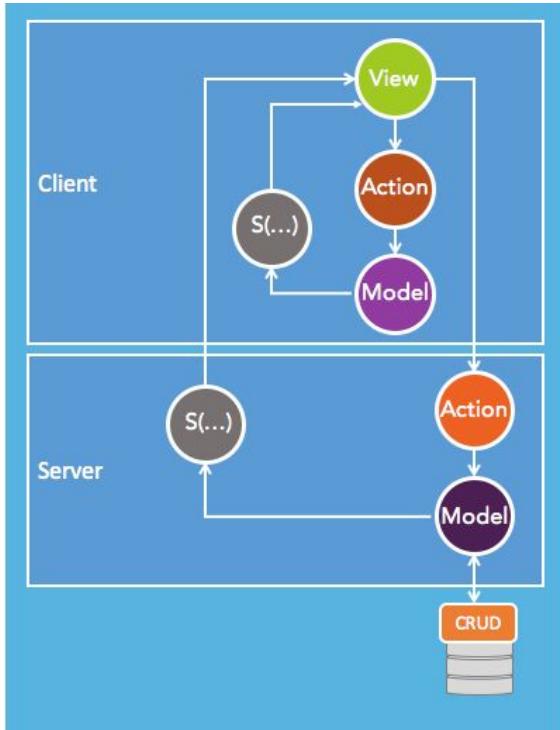


(1.) Proposers (2.) Acceptors (3.) Learners

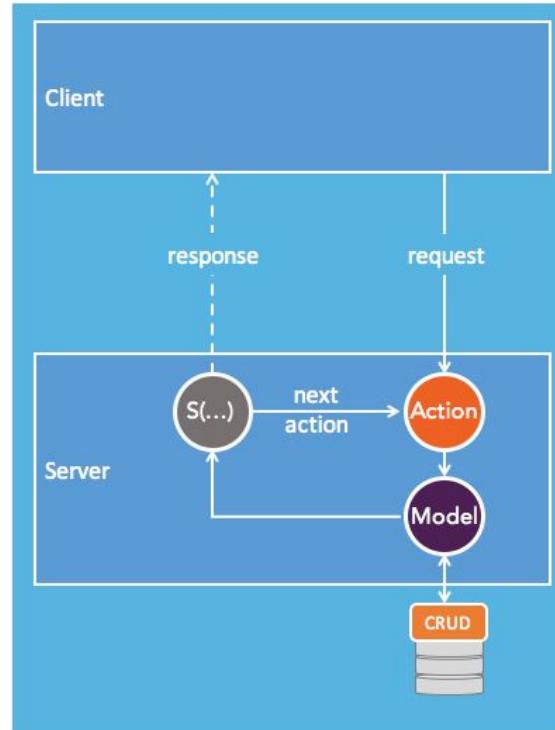


Varianten von SAM

Isomorphic JavaScript



Headless SAM



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)

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

$$\text{UI} = f(\text{data})$$

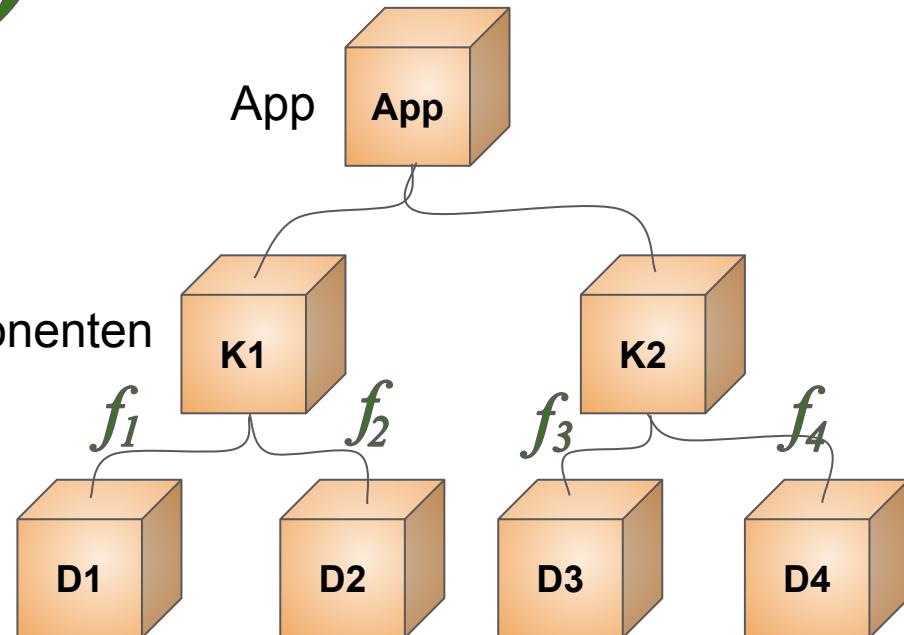
Die App ist eine Funktion.

Sie liefert mit Daten bestücktes HTML.

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

Daten

Komponenten



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>
```

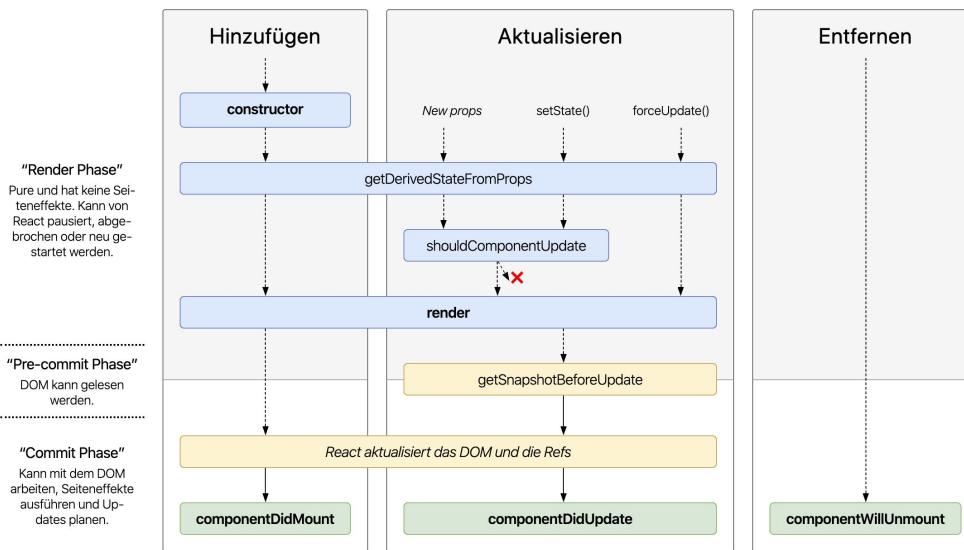
wandelt eine Funktion in eine
Komponente um

Function Components und React Hooks sind eine Neuerung in React 16.8

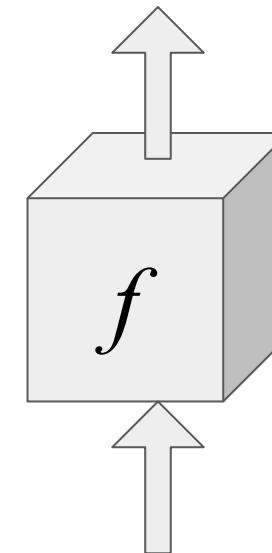
React Components beruhen auf einem komplexen Lifecycle Modell.

Weniger gebräuchliche Lifecycles anzeigen

React version 16.4 Language de-DE



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



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));
```

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`

<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`

${count}


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

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

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

0 is the initial value of `count`

ES6 Array Destructuring Assignment

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.

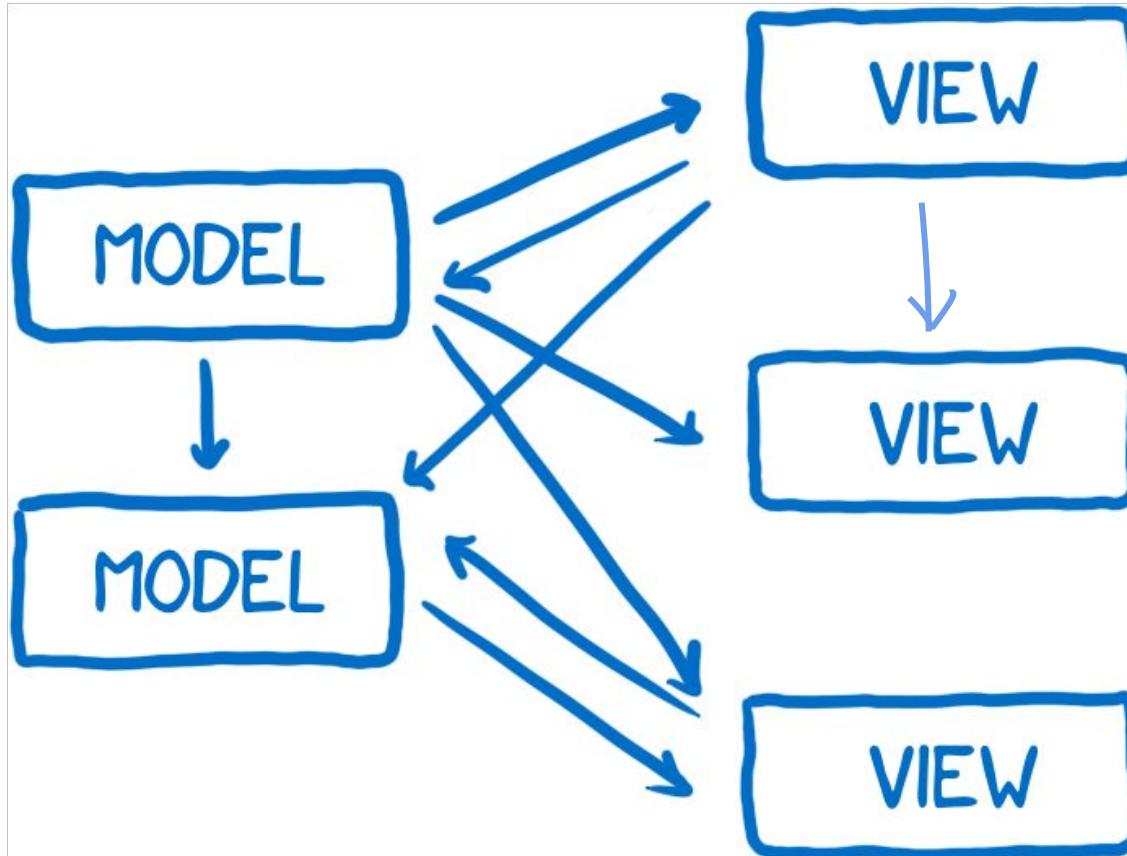
⇒ Eine WebApp besteht aus einem Netzwerk vieler Komponenten, die miteinander kommunizieren

2. Das State-Action-Model (SAM) - Pattern

3. Hooks

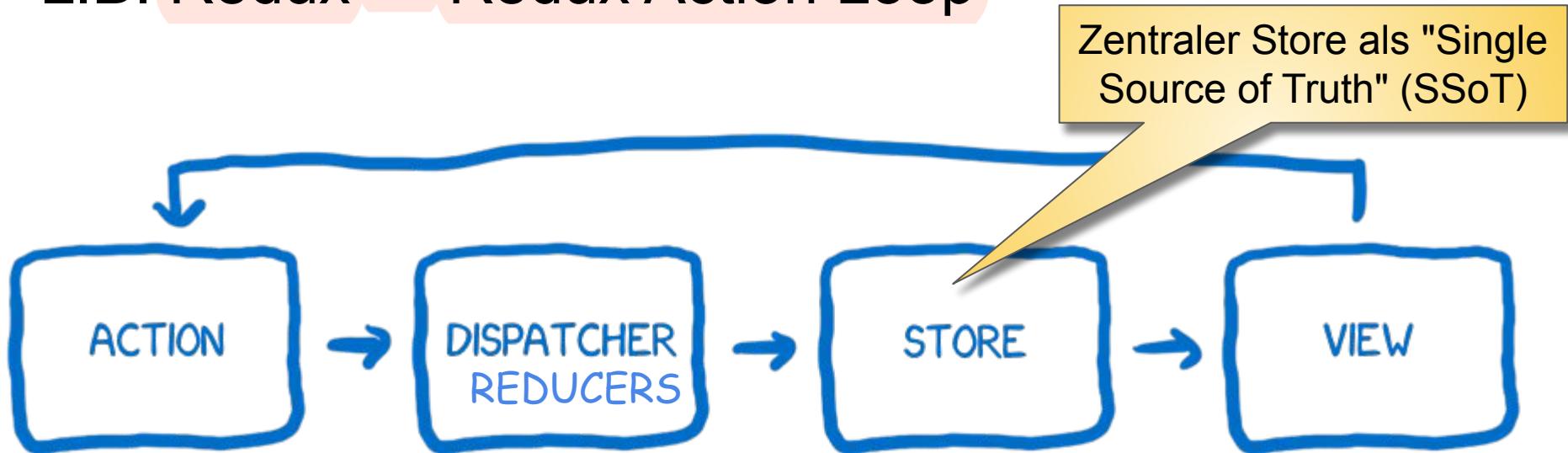
4. Redux

State Management als Lösung für das Update-Chaos-Problem



Die Lösung: Zentralisierung + Unidirektionaler Datenfluss

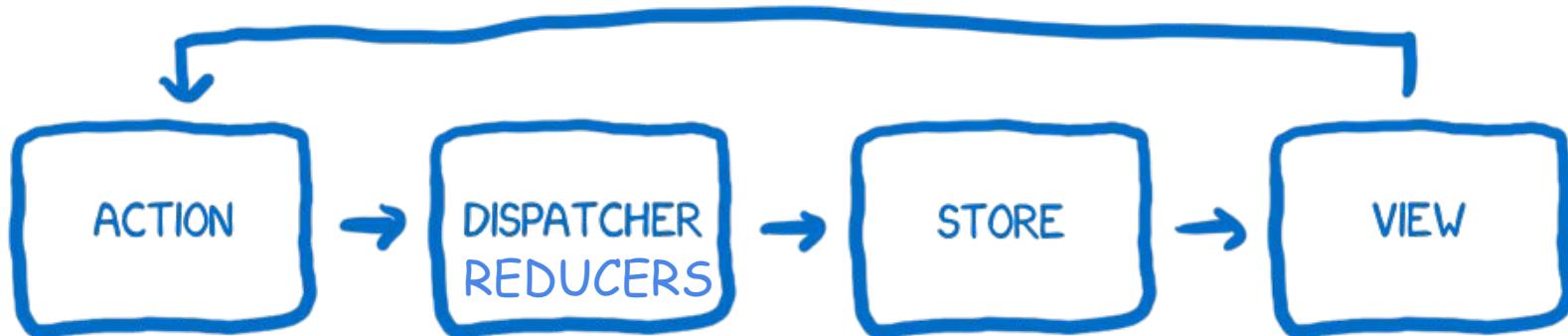
z.B. Redux ⇒ Redux Action Loop



- 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

Redux - Beispiel 1: Counter

my app

View One View Two View Three

Redux example: simple counter

2

Mehrfache konsistente Anzeige

This page contains a reusable <counter-element>. The element is not built Redux-y way (you can think of it as being a third-party element you obtained from someone else), but this page is connected to the Redux store. When this element updates its counter, this page updates the values in the Redux store and you can see the current value of the counter reflected in the bubble.

Clicked: 6 times. Value is 2

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

Custom Element <counter-element>

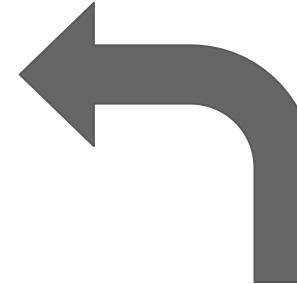
```
class CounterElement extends LitElement {  
  static get properties() {  
    return {  
      /* The total number of clicks you've done. */  
      clicks: { type: Number },  
      /* The current value of the counter. */  
      value: { type: Number }  
    }  
  }  
  
  render() {  
    return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

1. Redux Actions: Aus Methodenaufrufen werden Objekte

Action



View

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

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

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

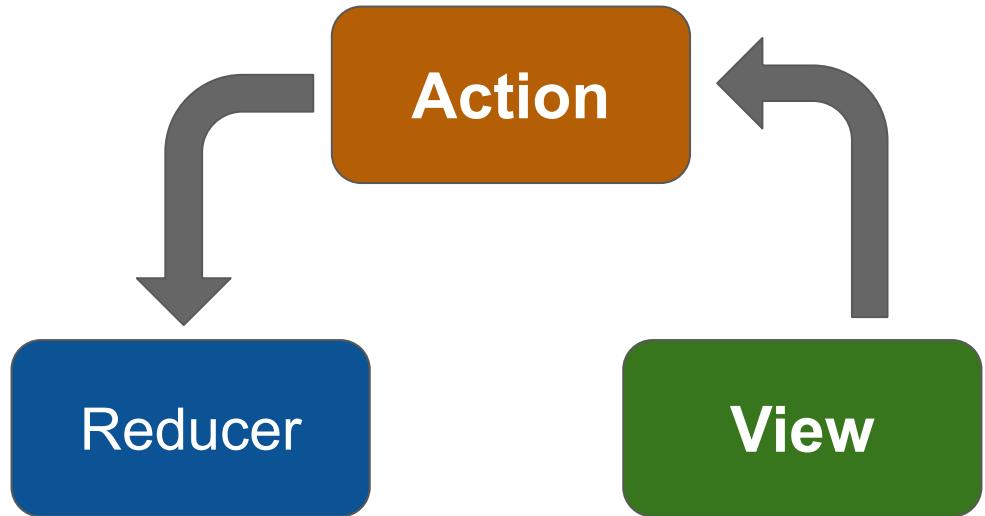
<https://github.com/Polymer/pwa-starter-kit/blob/master/src/actions/counter.js>

2. Redux Reducer: Old State → New State

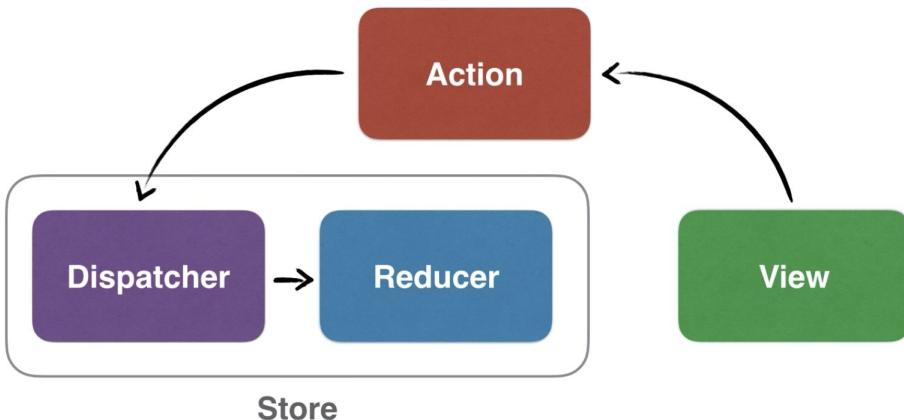
```
import { INCREMENT, DECREMENT } from './actions/counter.js';

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;
  }
};
```



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



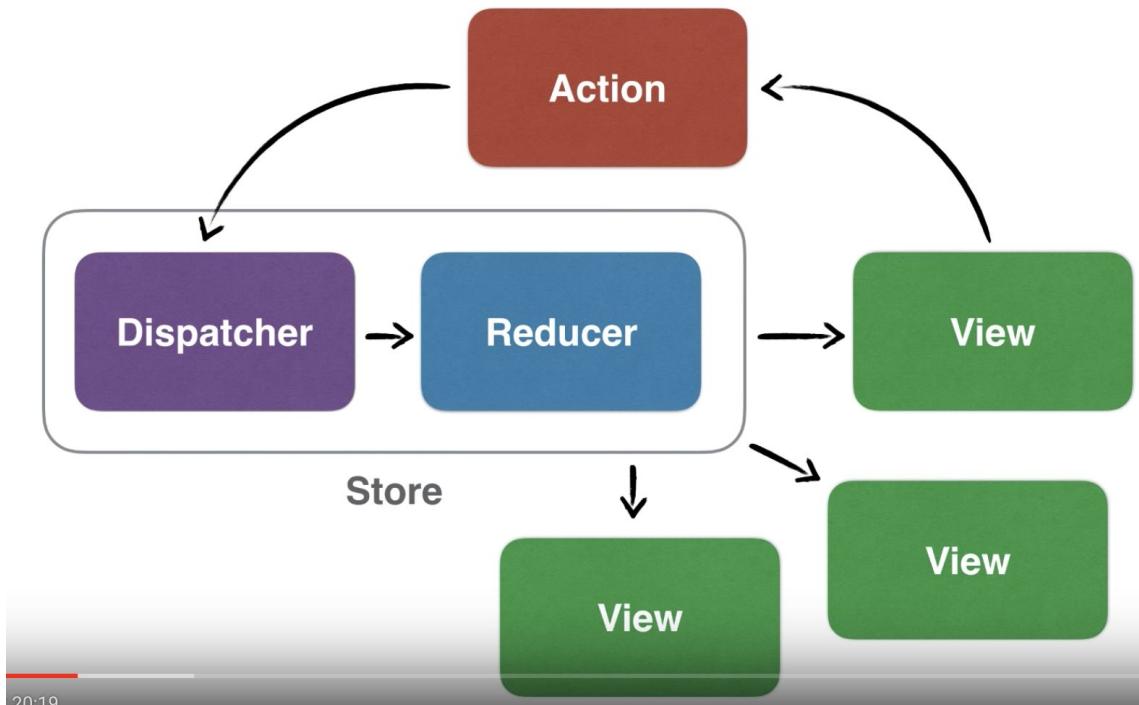
```
const increment = () => {
  return {
    type: INCREMENT
  };
};

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

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

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

4. Redux Store: Views updaten mit neuem State



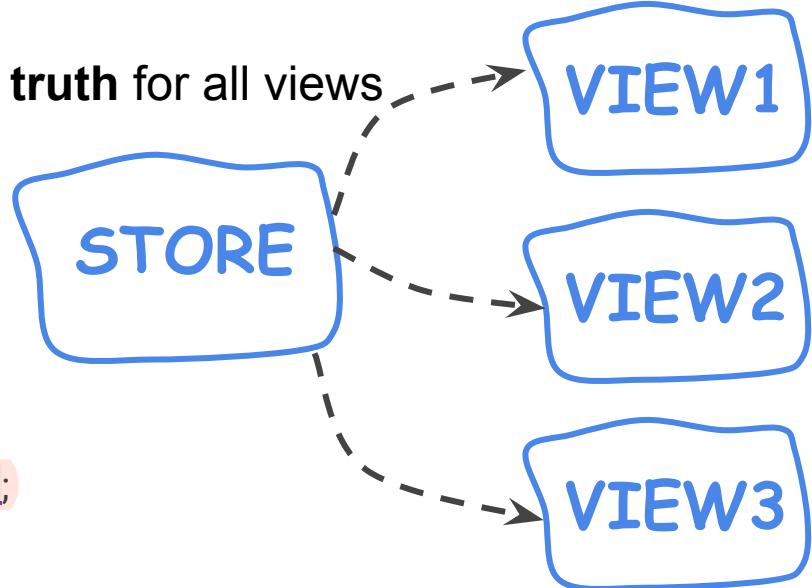
// This is called every time something is updated in the store.

```
stateChanged(state) {  
  this._clicks = state.counter.clicks;  
  this._value = state.counter.value;  
}
```

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)`.



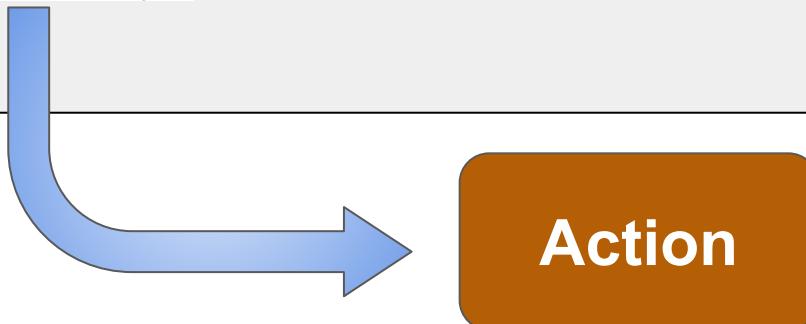
You'll only have **a single store** in a Redux application.

Observer-Pattern

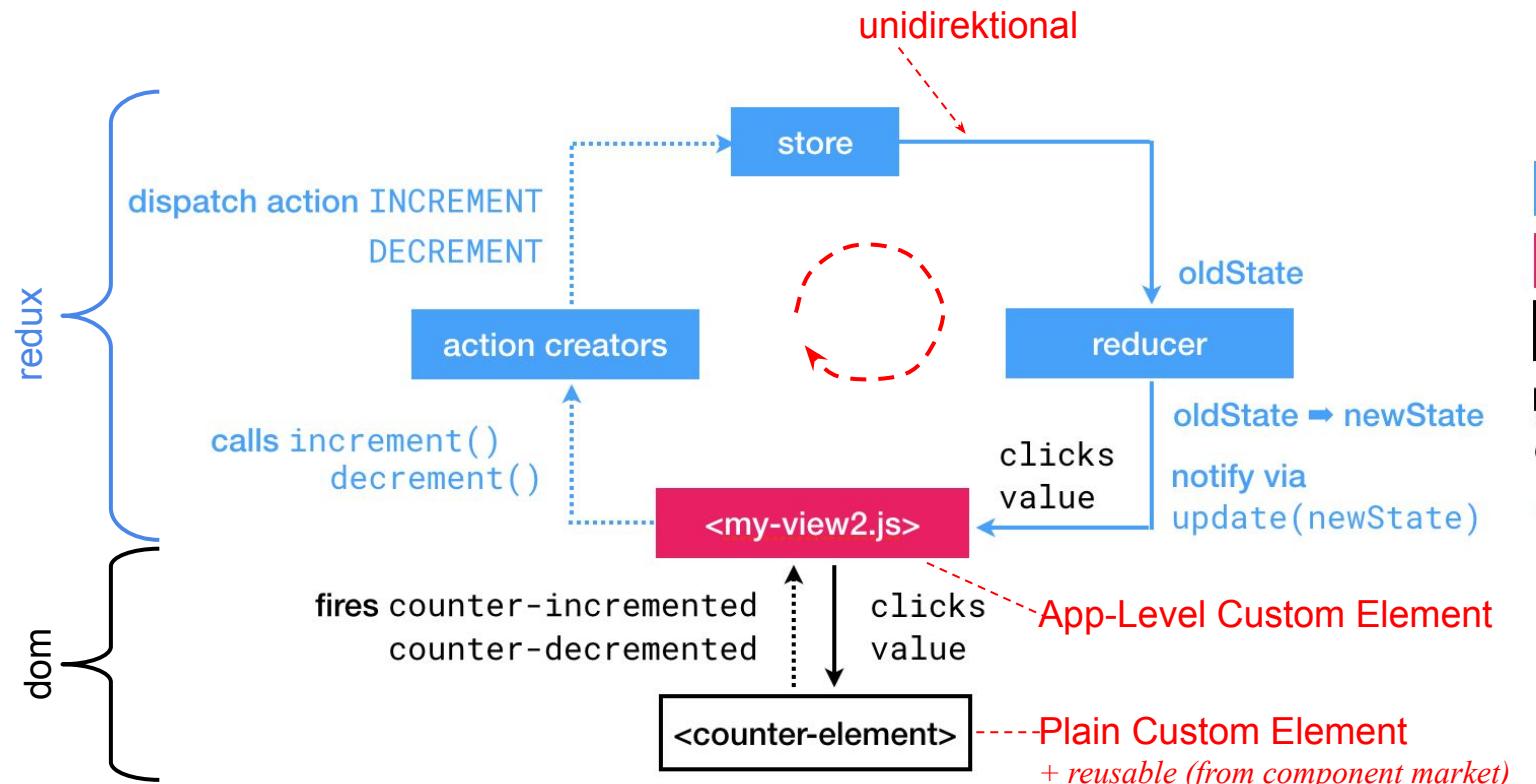
Store ⇒ dispatch(Action) ⇒ Reducer ⇒ update



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



redux Action Loop: Beispiel Counter



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.

Chrome Extension: Redux DevTools

my app

View One View Two **View Three**

Redux example: simple counter

2

This page contains a reusable <counter-element>. The element is not built in a Redux-y way (you can think of it as being a third-party element you got from someone else), but this page is connected to the Redux store. When the element updates its counter, this page updates the values in the Redux store, and you can see the current value of the counter reflected in the bubble above.

Clicked: 6 times. Value is 2 . + -

Inspector

filter...

INCREMENT

DECREMENT

DECREMENT

INCREMENT

Diff Action State Diff Test

Tree Raw

counter (pin)

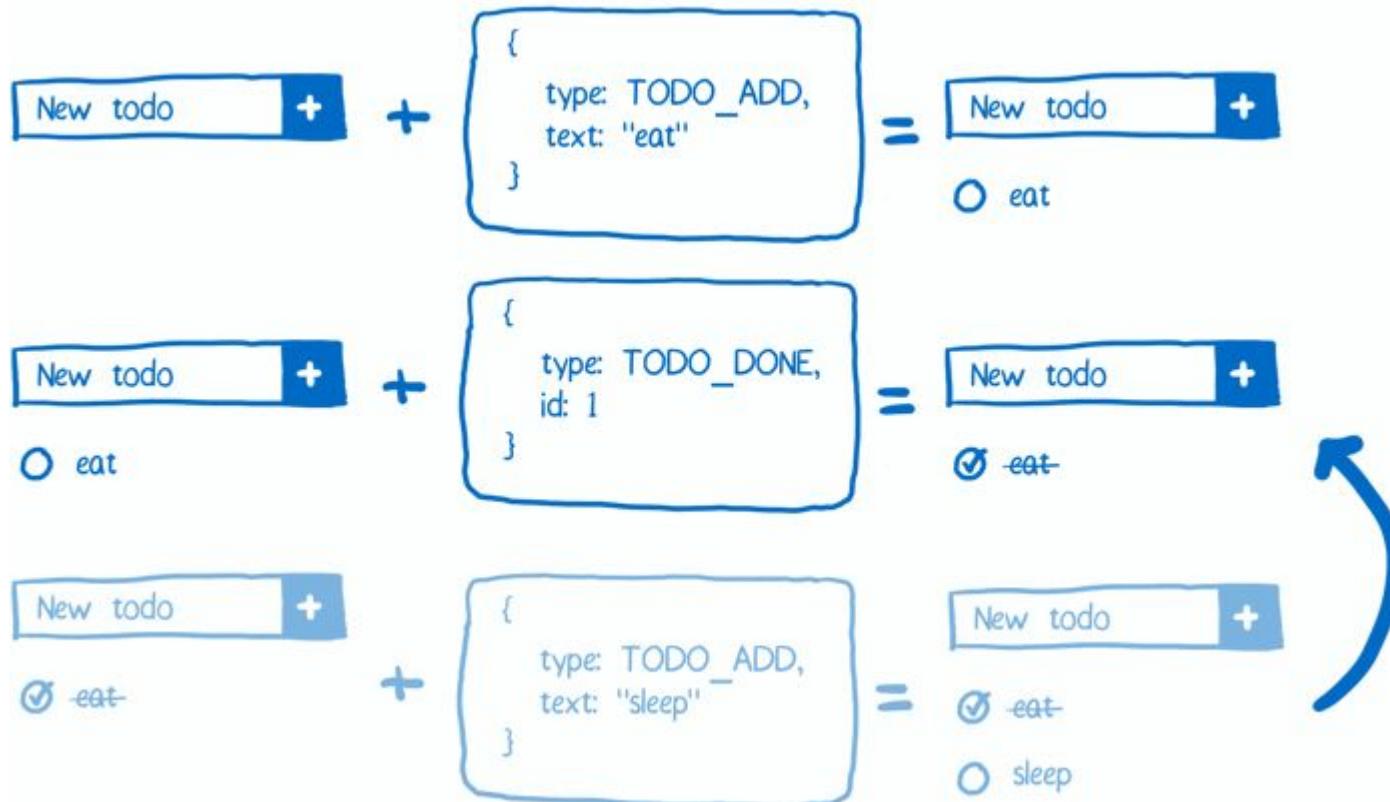
clicks (pin): 5 => 6

value (pin): 1 => 2

History Playback

The screenshot shows the Redux DevTools extension running in a browser. On the left, a sidebar titled 'Inspector' lists actions: 'INCREMENT', 'DECREMENT', 'DECREMENT', and 'INCREMENT'. Below this is a 'Diff' table with tabs for 'Action', 'State', 'Diff', and 'Test'. The 'State' tab is selected, showing a tree view with a single node 'counter (pin)'. Under 'counter', two state changes are listed: 'clicks (pin)' changing from 5 to 6, and 'value (pin)' changing from 1 to 2. A large button labeled 'History Playback' is visible at the bottom right of the sidebar. At the very bottom of the browser window, there's a row of small icons.

Time Travel Debugging



Wie verbindet man seine Komponenten mit dem Redux Store?

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 eine 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)) { ... }
```

Wdh. Multiple Inheritance

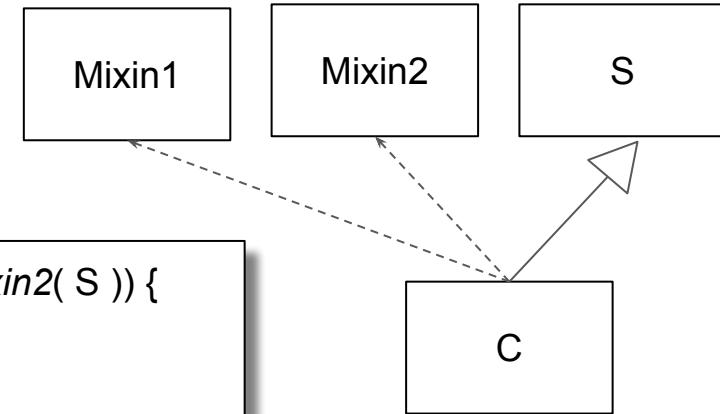
Mixins: Klassen als Parameter & Return Value

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

```
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  
))))))))))))))));
```

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?

Ja

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>