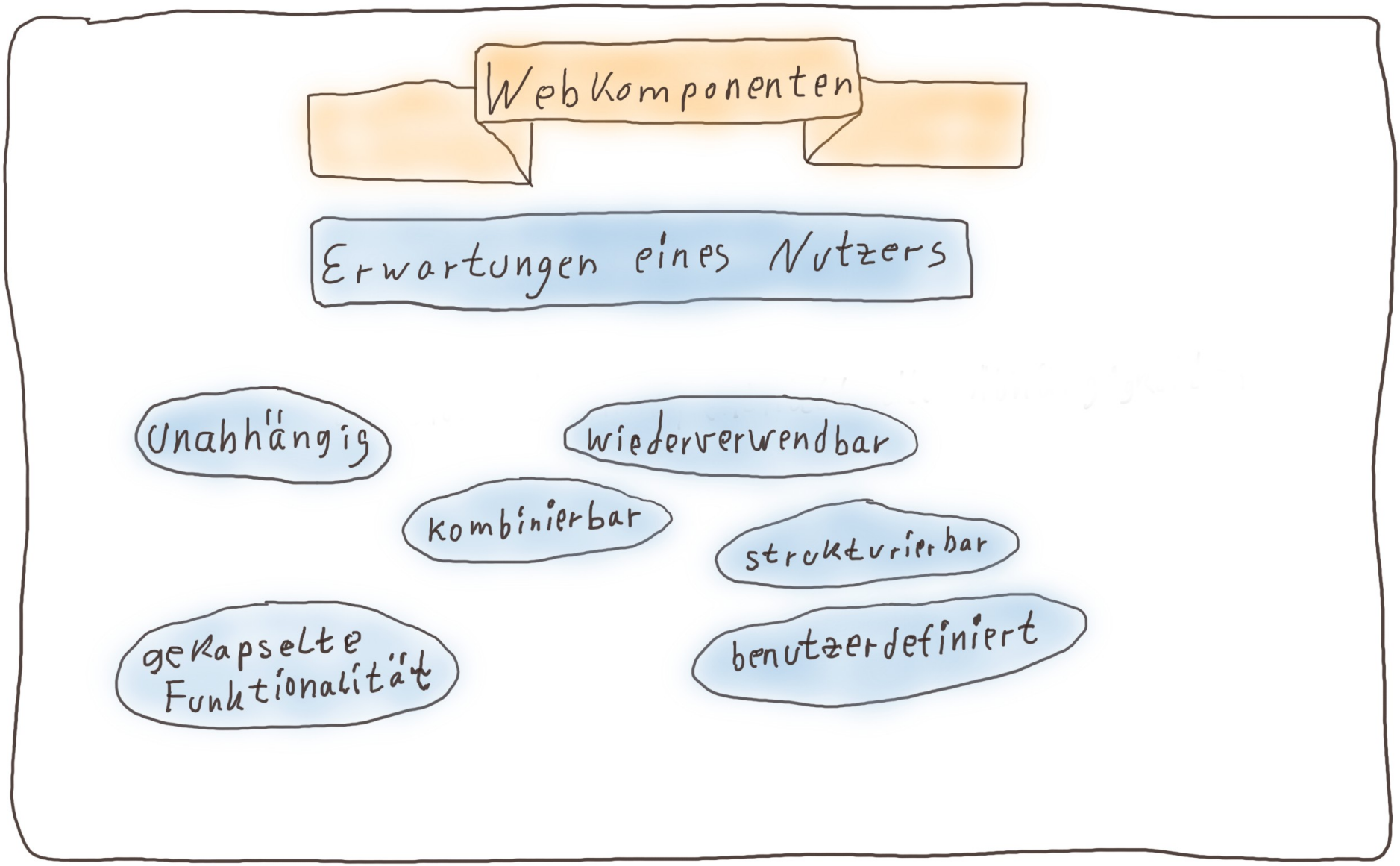
Webkomponenten mit Stencil bauen

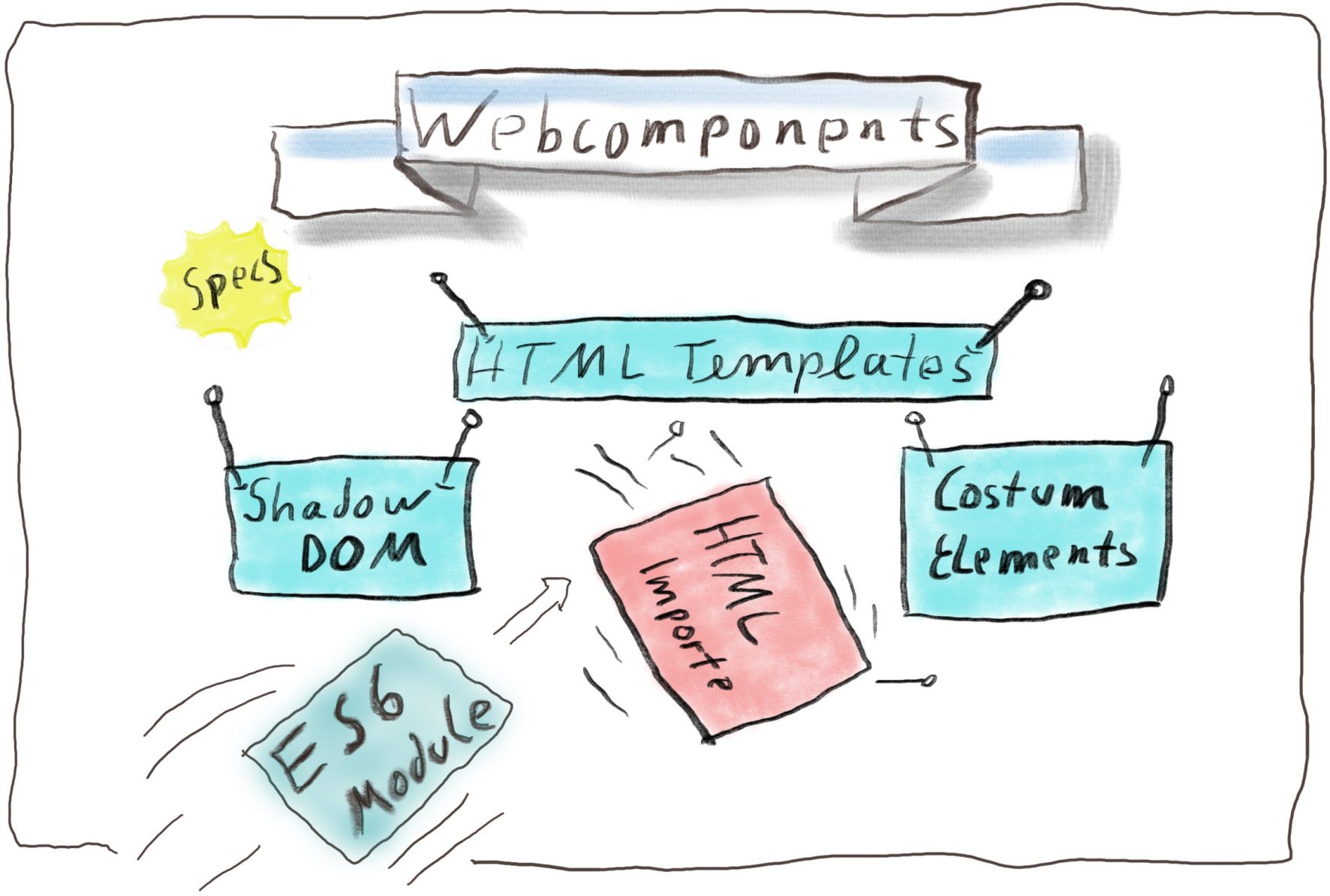
Autor: Thomas Schubert

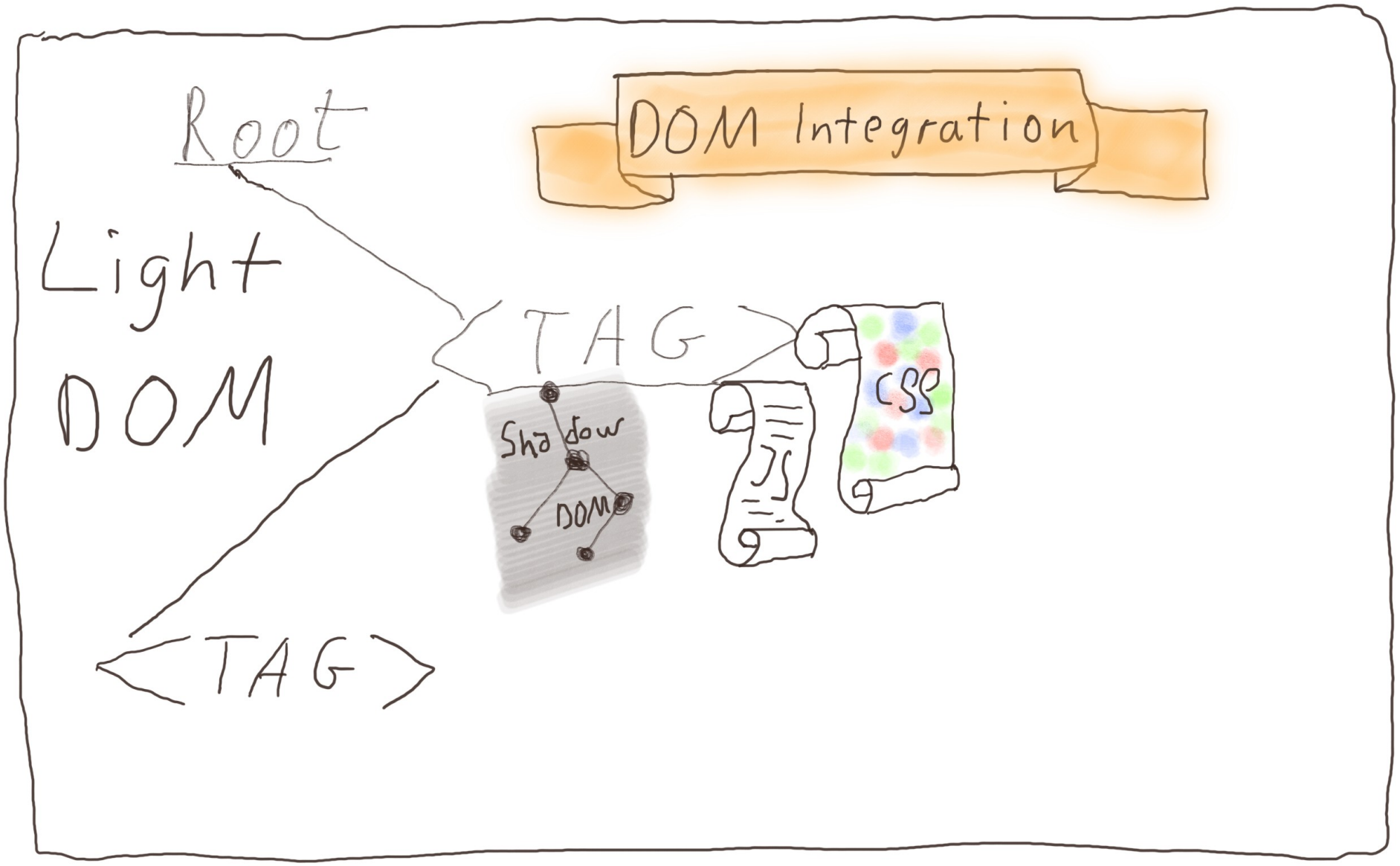
Beispiel Anwendung
- https://github.com/Huluvu424242/honey-chucknorris-jokes/tree/geparkt/stencil-slides

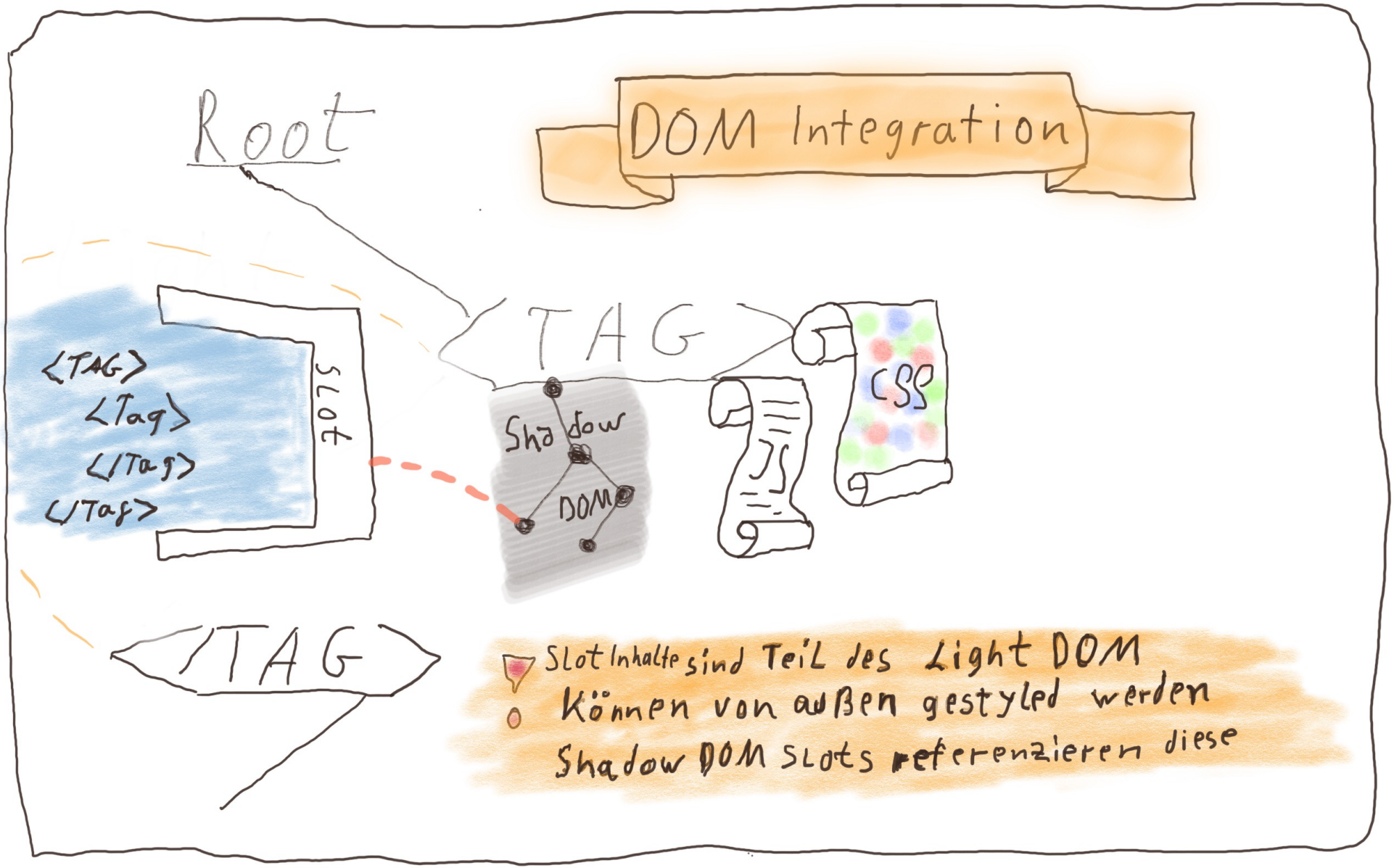
Bildquellen
- https://flickr.com/photos/huluvu424242/albums/72157719354993995
- https://flickr.com/photos/huluvu424242/albums/72157719063932639

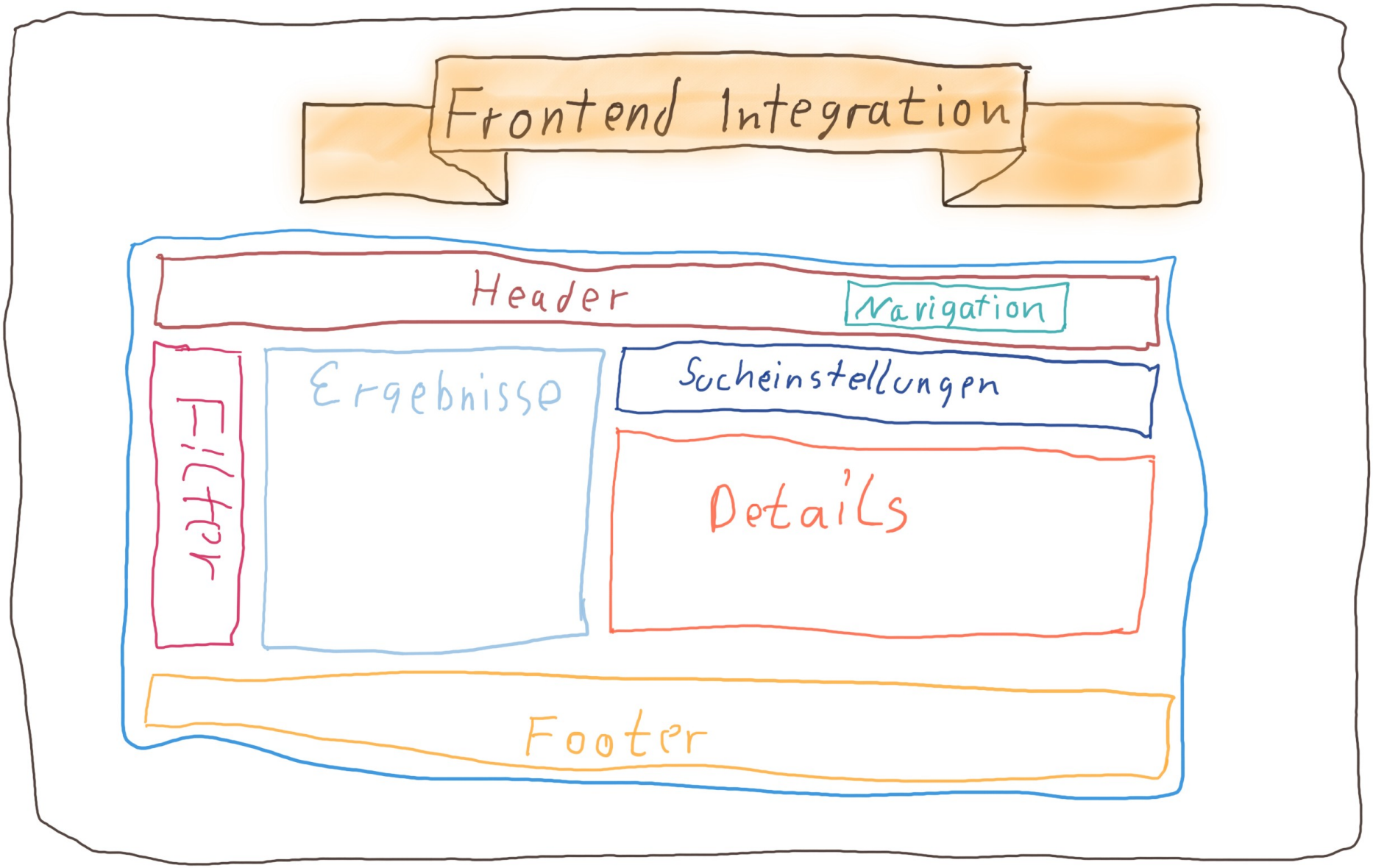
Slides zum Nachlesen und Hören
- https://huluvu424242.github.io/foile-pile/explainations/stenciljs/index.html

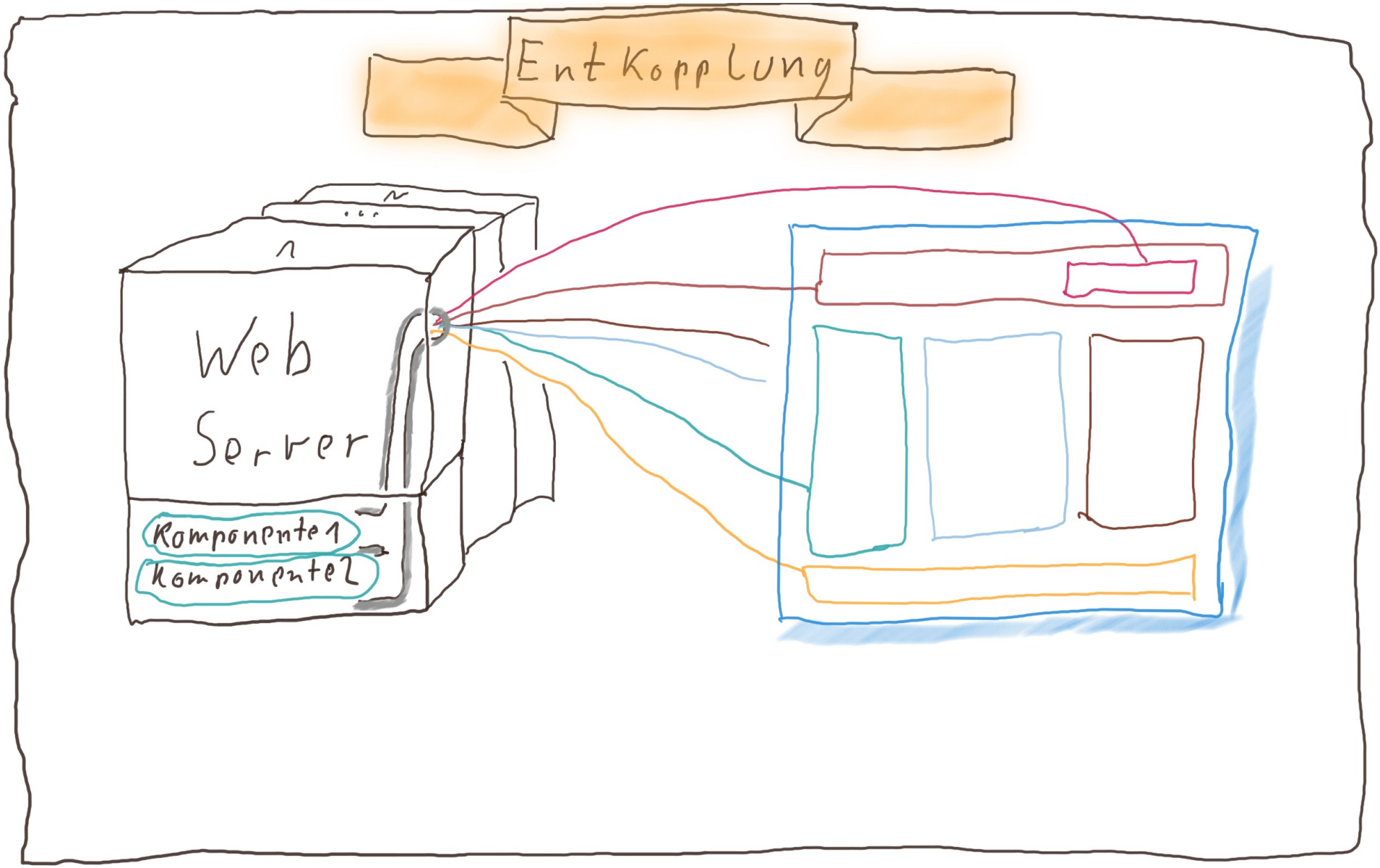

Web Komponenten
Erwartungen eines Nutzers
Unabhängig
wiederverwendbar
kombinierbar
strukturierbar
gekapselte Funktionalität
benutzerdefiniert


Webcomponents
Specs
HTML Templates
Shadow DOM
HTML Imports
Costum Elements
ES6 Module


Root
DOM Integration
Light
DOM
<TAG>
Shadow
DOM
JS
CSS
<TAG>


Root
DOM Integration
<TAG>
<Tag>
</Tag>
</TAG>
slot
<TAG>
Shadow
DOM
CSS
JS
</TAG>
Slot Inhalte sind Teil des Light DOM
Können von außen gestyled werden
Shadow DOM Slots referenzieren diese


Frontend Integration
Header
Navigation
Filter
Ergebnisse
Scheinstellungen
Details
Footer


Entkopplung
Web
Server
Komponente1
Komponente2

Pure Javascript Implementierung

```
const template = document.createElement('template');
template.innerHTML = `
<style>
p {
  font-family: Roboto, sans-serif;
  font-weight: bold;
}
<p><slot></slot></p>
`;

export class WCShadowDom extends HTMLElement {

  constructor() {
    super();
    this.attachShadow({mode: 'open'});
    this.shadowRoot.appendChild(document.importNode(template.content, true));
    // Initialize variables here
  }

  async connectedCallback() {
    // Use querySelector to grab reference to an element in the Shadow DOM
    this.shadowRoot.querySelector('p').firstChild().innerHTML = 'Insert some stuff';
    // This runs when the element is rendered
  }

}

customElements.define('wc-shadow-dom', WCShadowDom);
```

← Template erzeugen

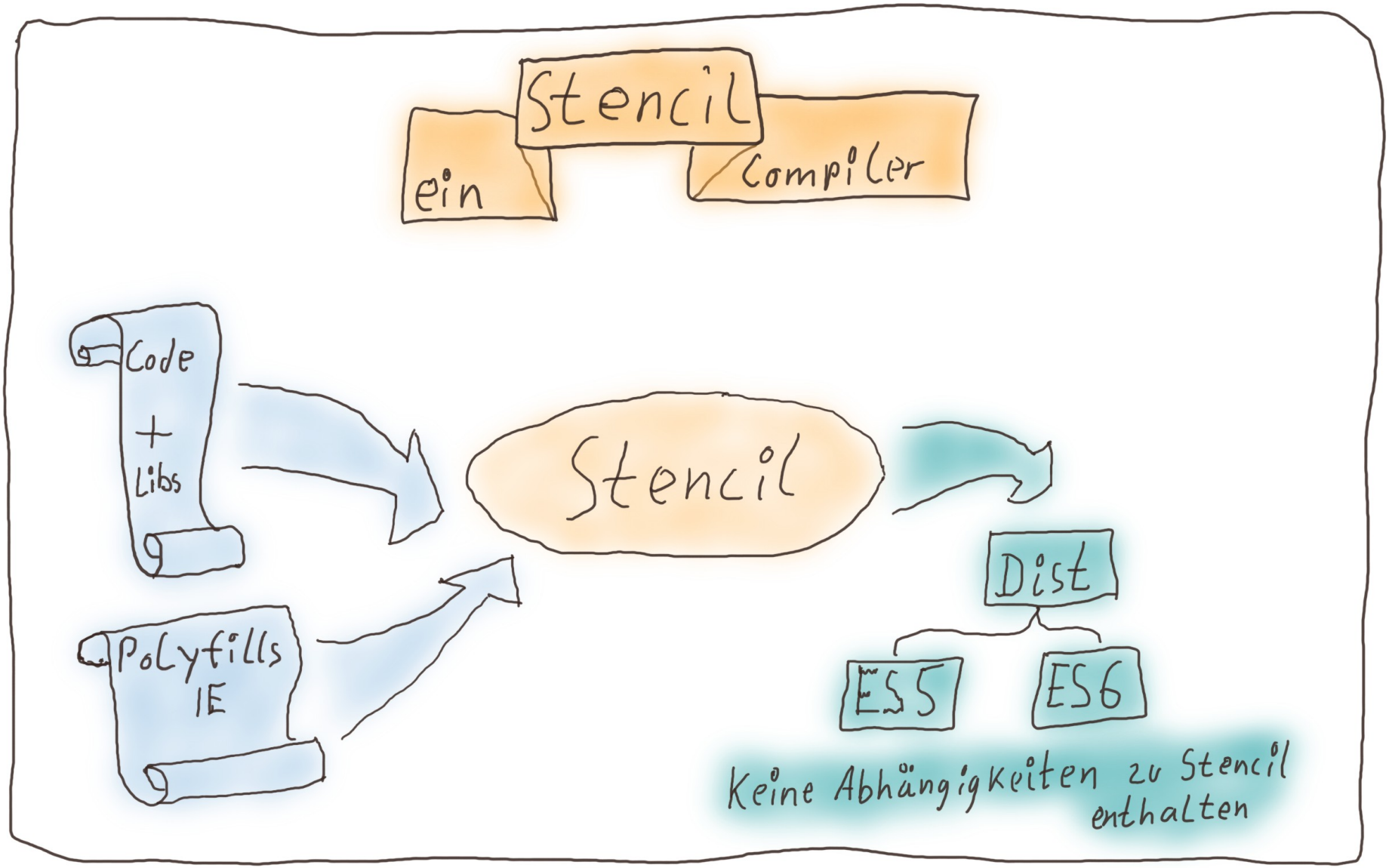
← Style definieren

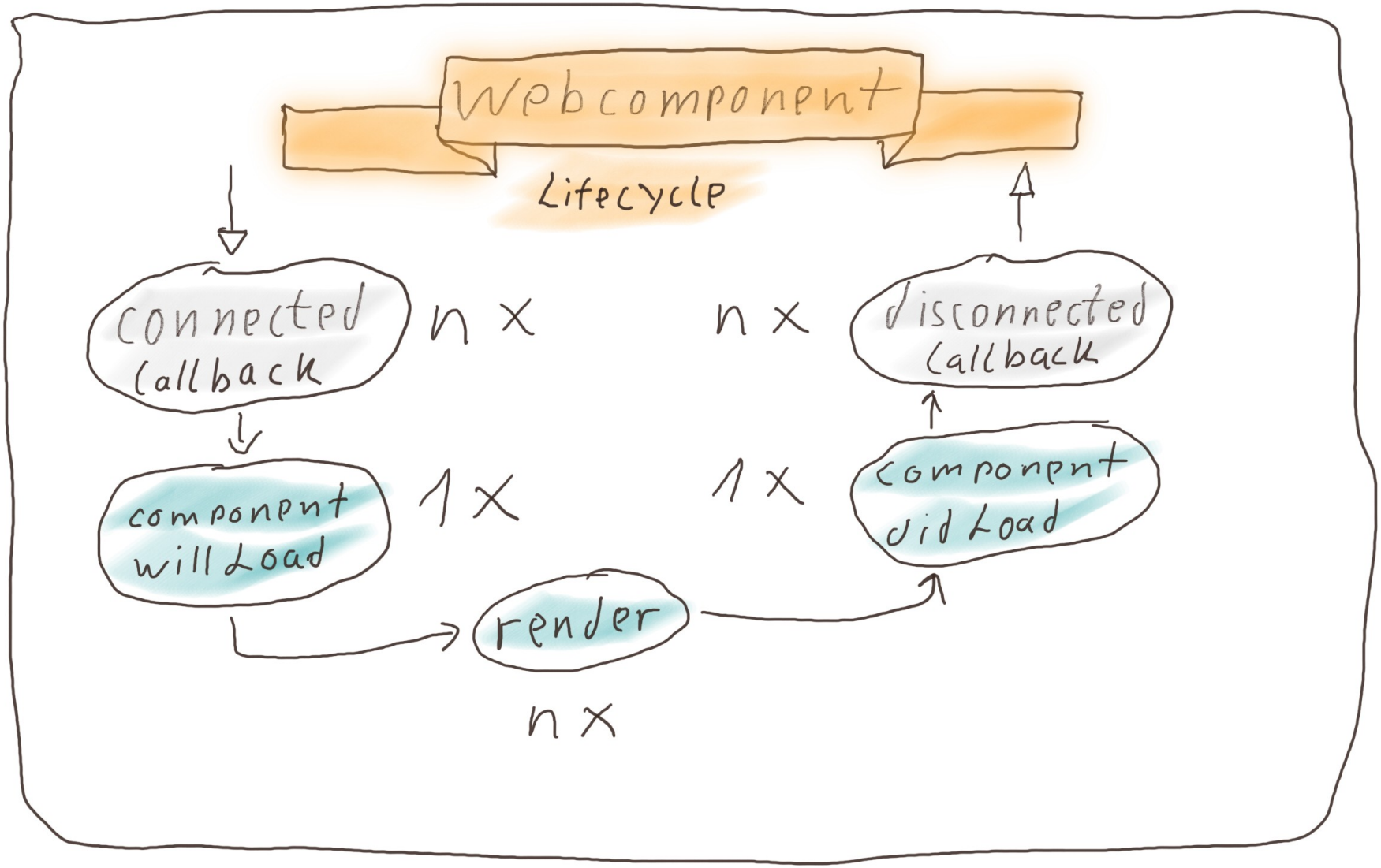
← Custom Element ableiten & implementieren

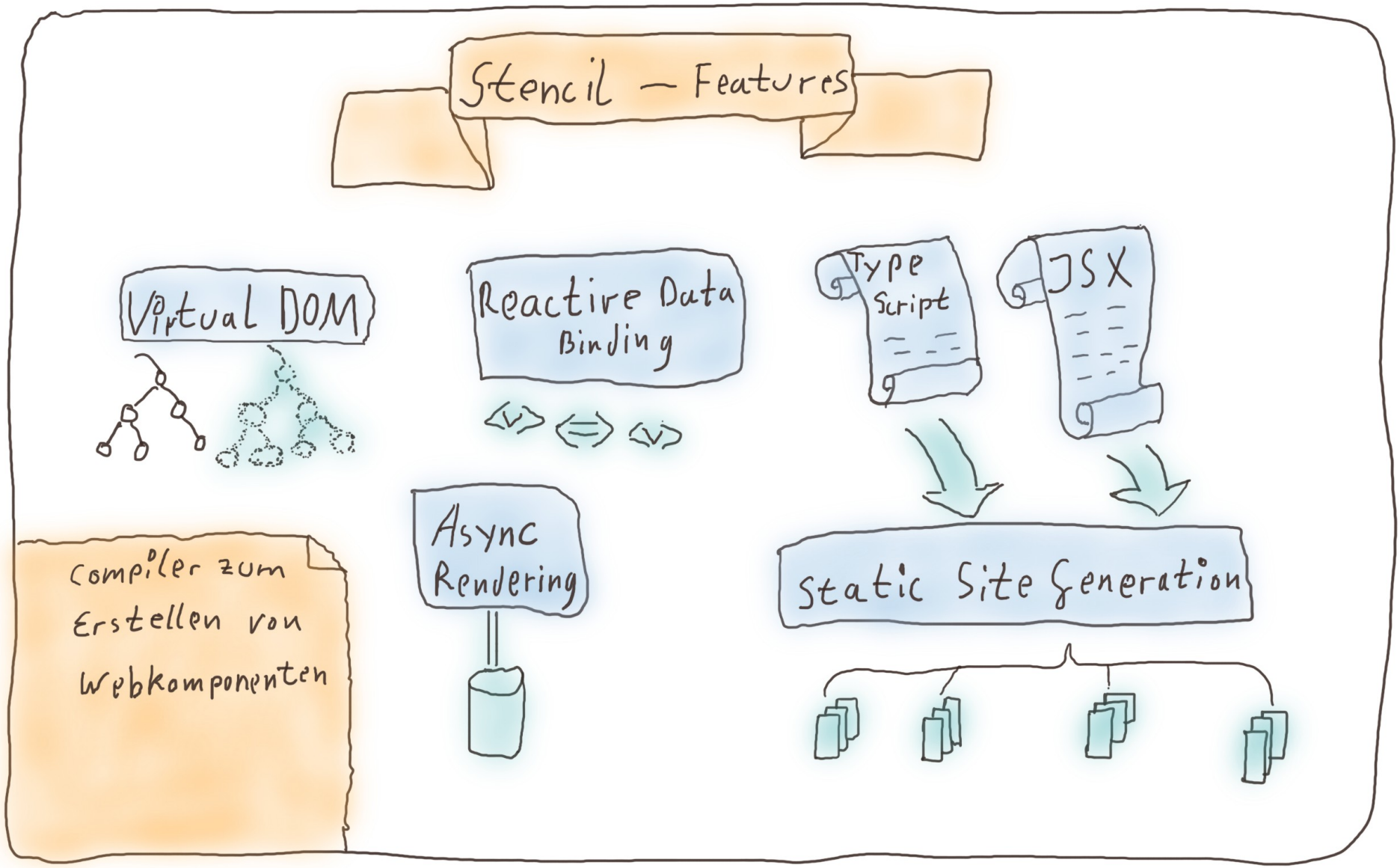
← CustomElement registrieren

Quelle:

https://github.com/vanillawc/vanillawc/blob/main/demo/basic-shadow-dom.js


Stencil
ein
Compiler
Code
+
Libs
Polyfills
IE
Stencil
Dist
ES5
ES6
Keine Abhängigkeiten zu Stencil
enthalten


Webcomponent
Lifecycle
connected callback
n x
component will Load
1 x
render
n x
n x
disconnected callback
1 x
component did Load

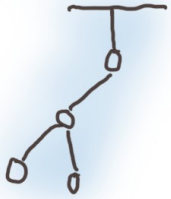

Stencil – Features
Virtual DOM
Reactive Data Binding
Type Script
JSX
Async Rendering
Static Site Generation
compiler zum Erstellen von Webkomponenten

DOM Bearbeitungs Konzepte

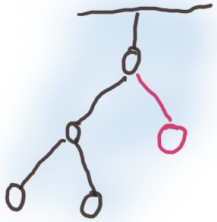
Browser DOM

Live DOM

Kopie Browser DOM



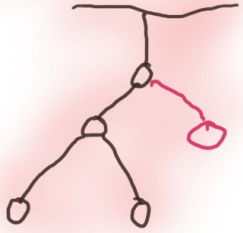
Veränderte Kopie DOM



Interpreter

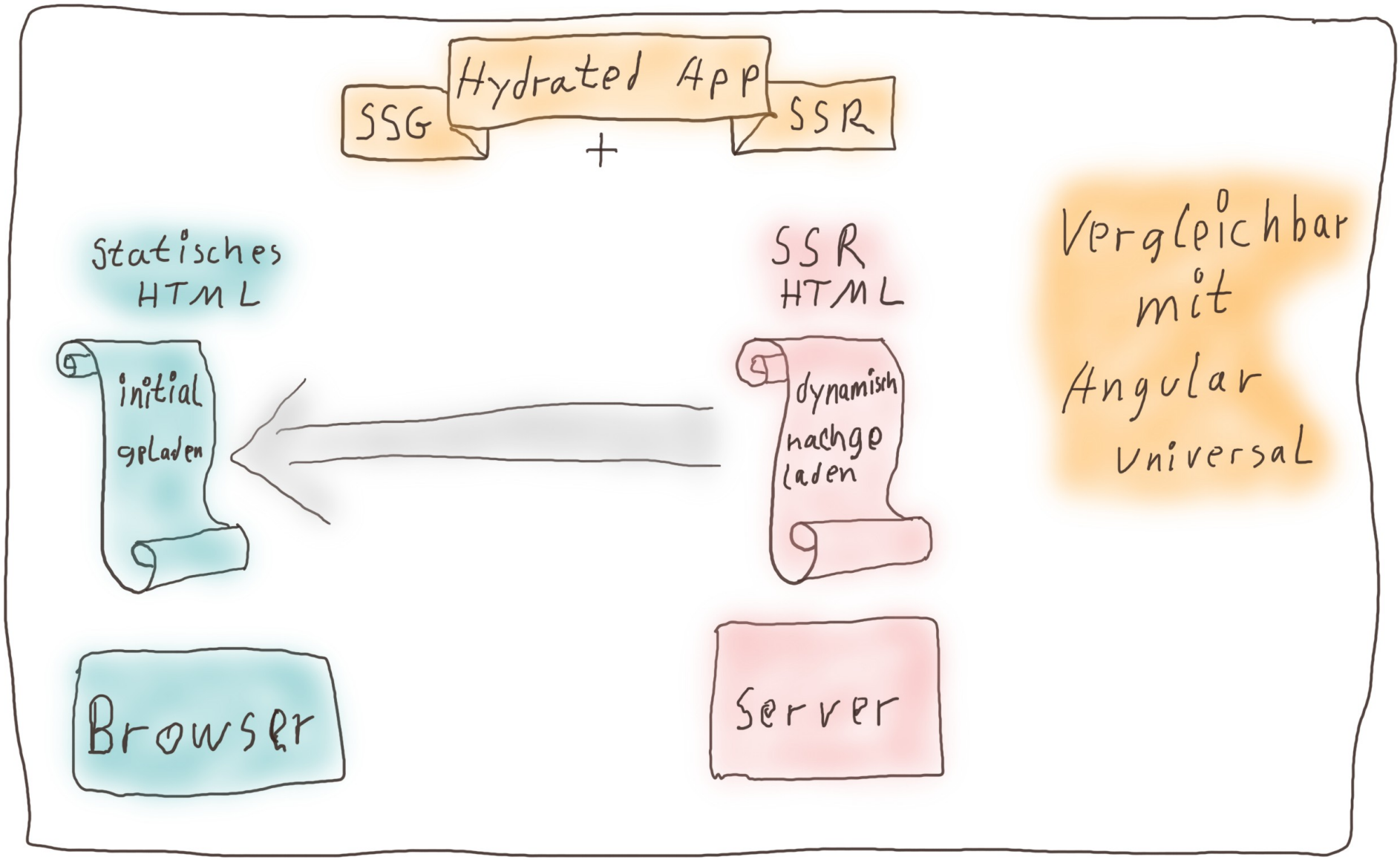
Virtual DOM
z.B. stencil

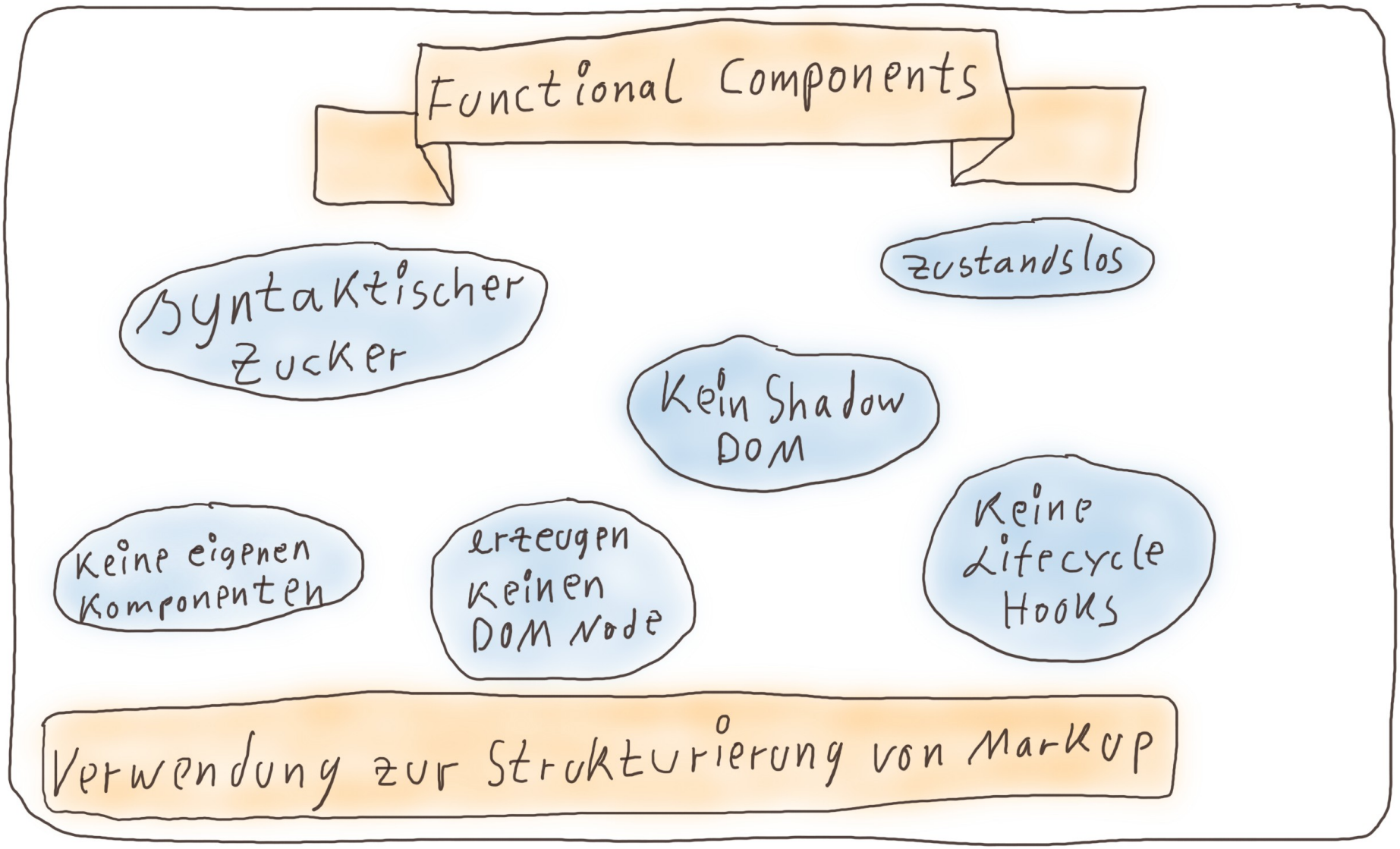
Kopie vom Browser DOM mit Veränderungen

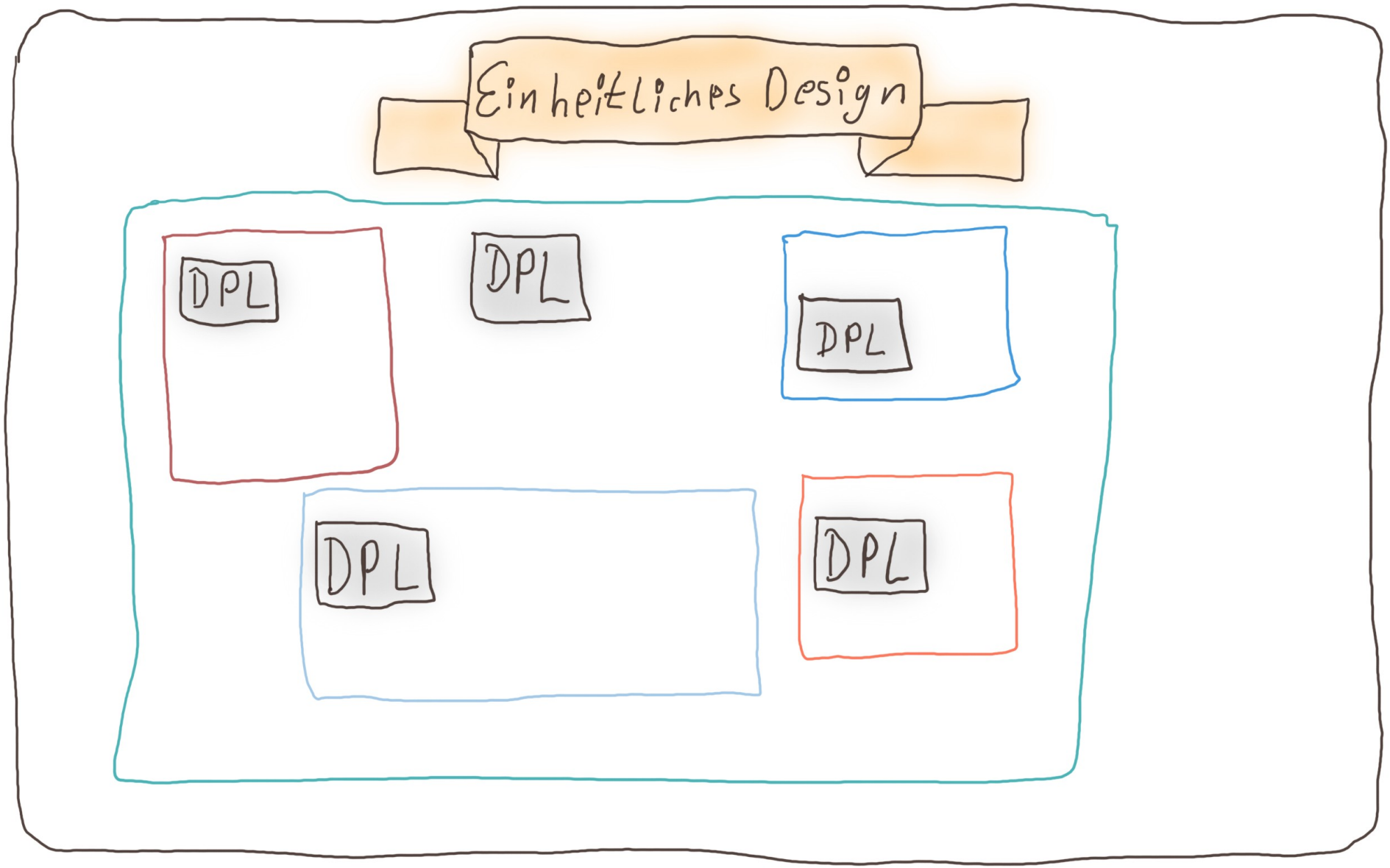


Incremental DOM
over
Virtual DOM

z.B. Angular


SSG
Hydrated App
+
SSR
Statisches HTML
initial geladen
SSR HTML
dynamisch nachgeladen
Vergleichbar mit Angular Universal
Browser
Server


Functional Components
Syntaktischer Zucker
Zustandslos
Kein Shadow DOM
Keine eigenen Komponenten
Erzeugen keinen DOM Node
Keine Lifecycle Hooks
Verwendung zur Strukturierung von Markup


Einheitliches Design
DPL
DPL
DPL
DPL
DPL

DPL Einbinden

1

Refs

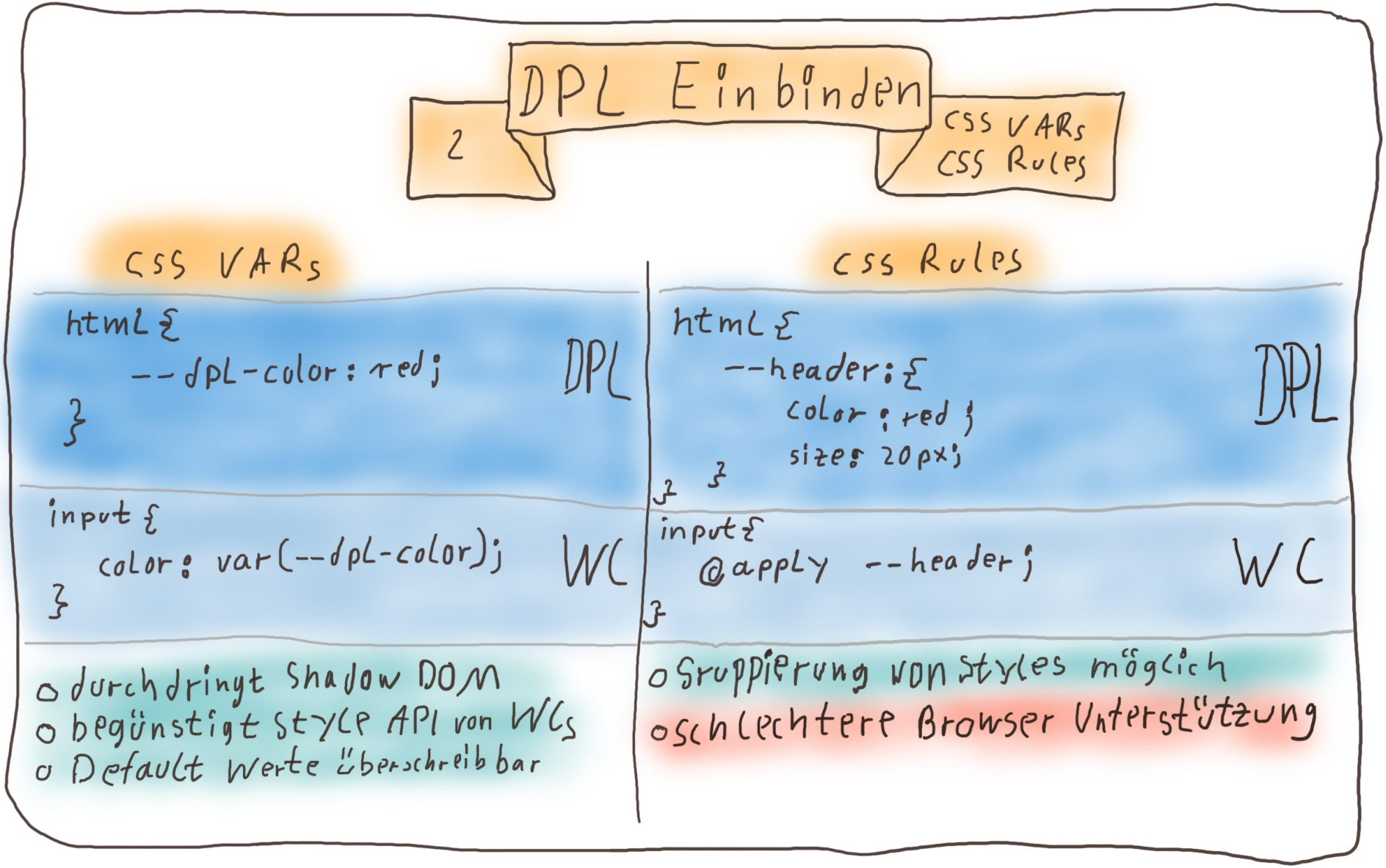
im HTML Template

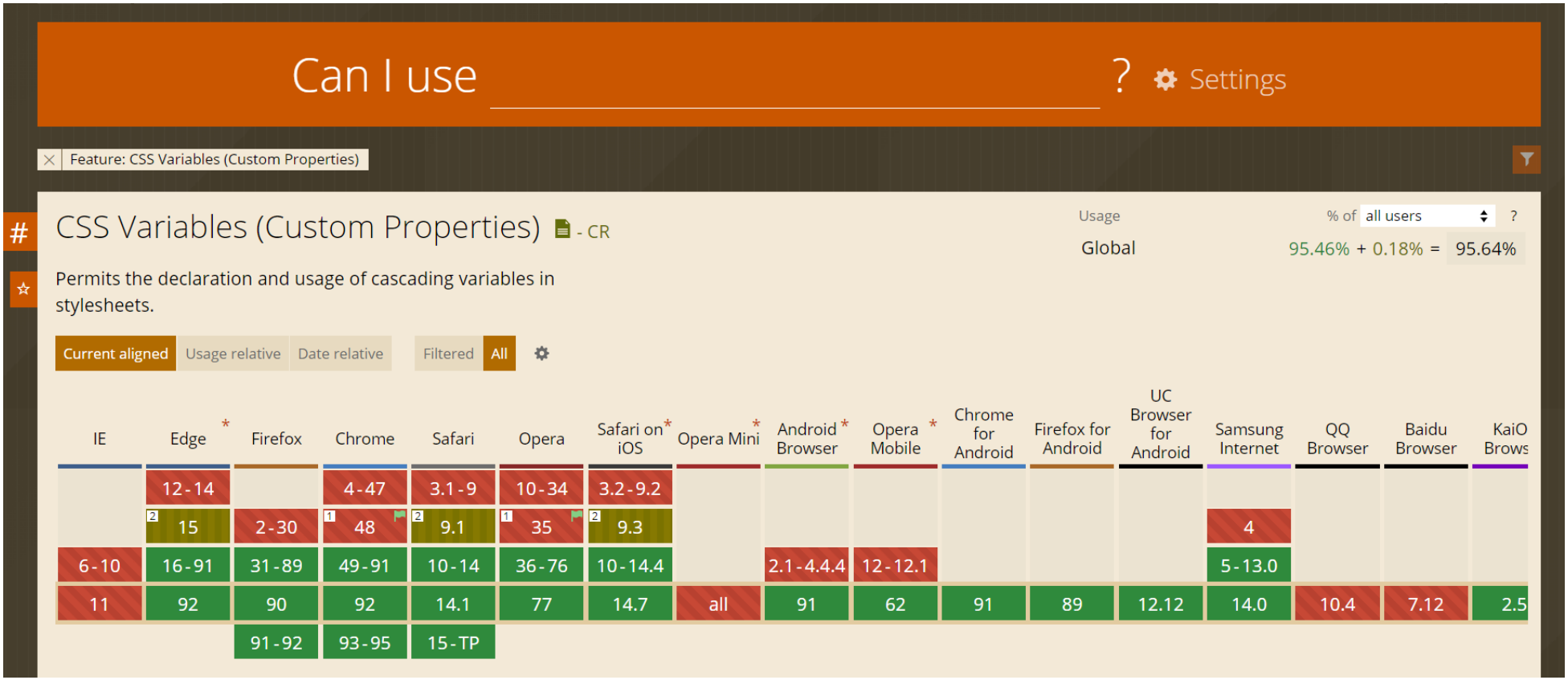
```
<style>
  @import url(...);
</style>
```

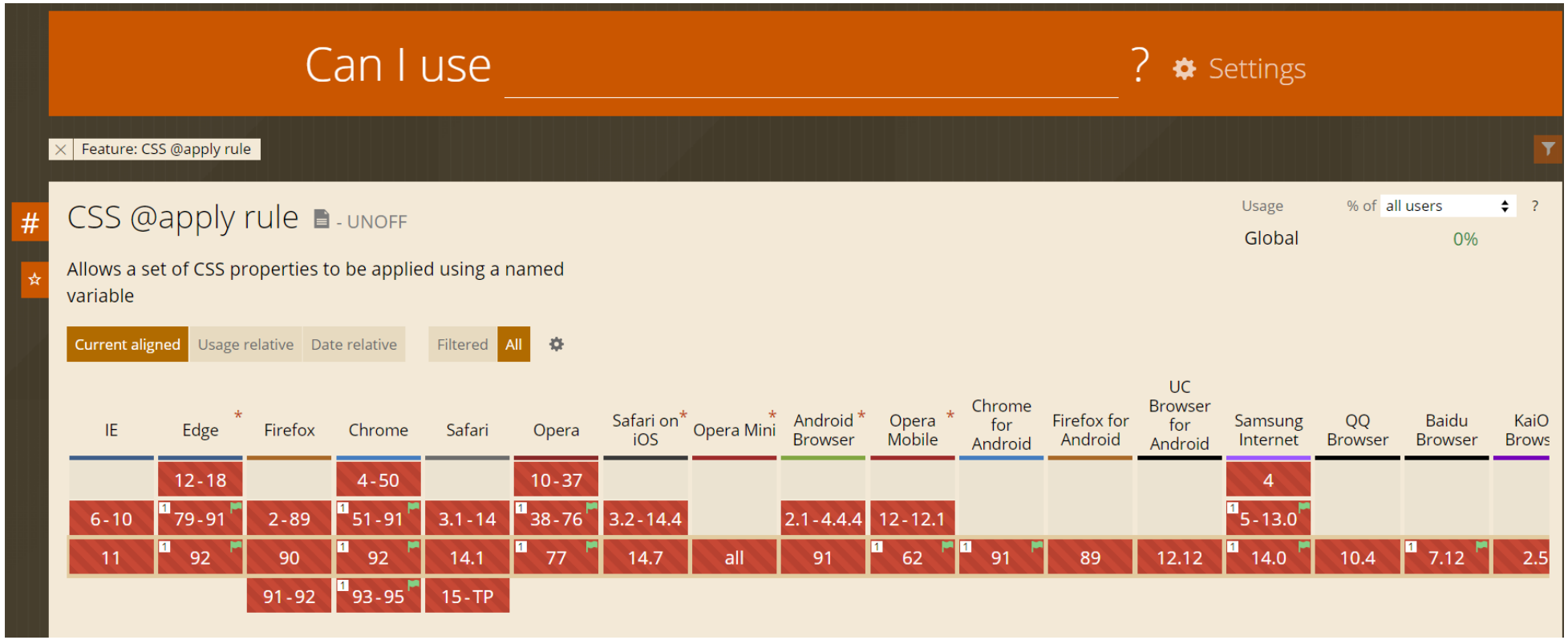
- Anti Pattern
- Führt zu Performance Problemen

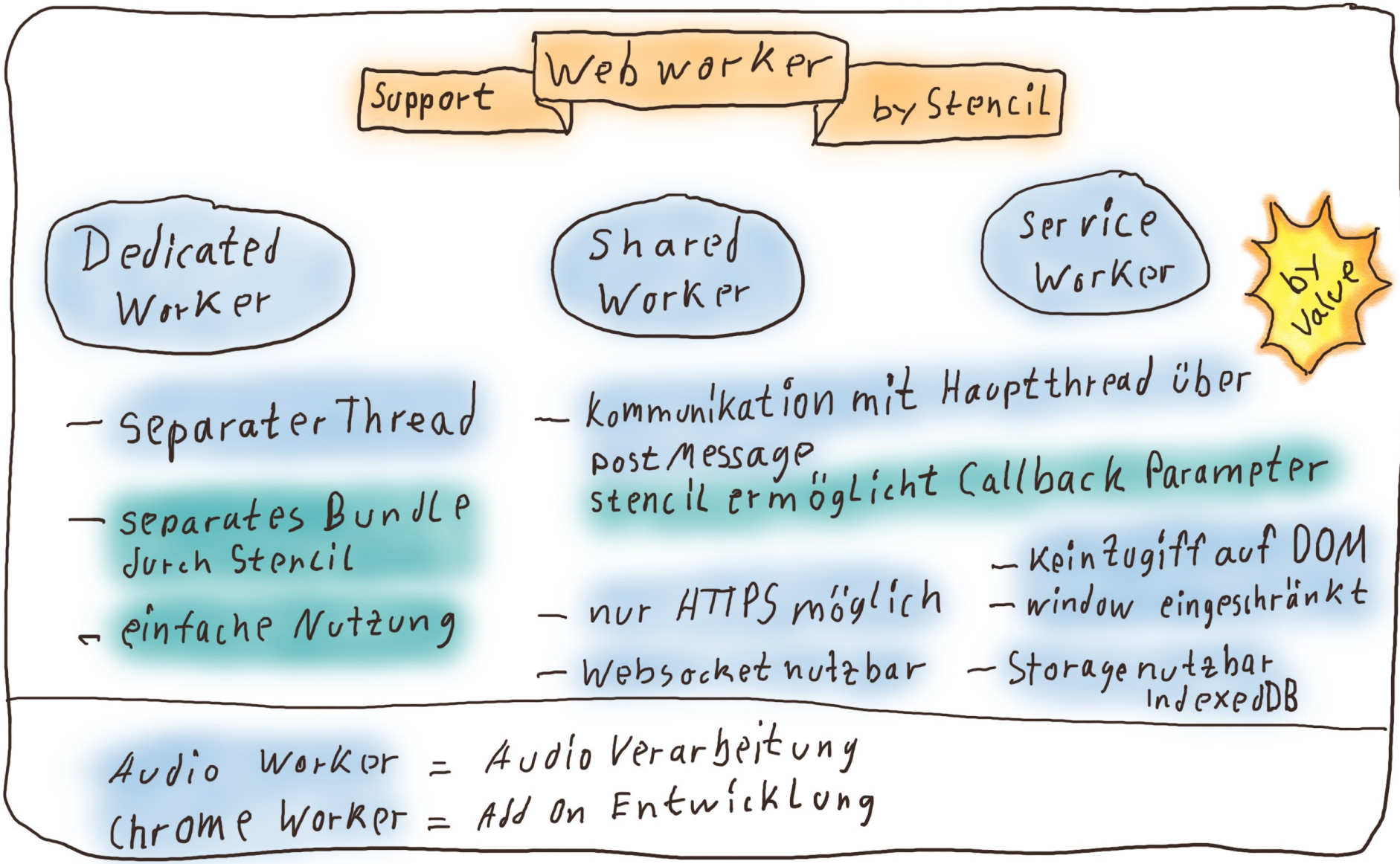
```
<link rel="stylesheet" ... >
<style>
  :          <- spezific Styles
</style>
```

- temporäre Lösung wird von der Spec nur kurzfristig erlaubt


DPL Einbinden
2
CSS VARs
CSS Rules
CSS VARs
CSS Rules
html {
--dpl-color: red;
}
DPL
input {
color: var(--dpl-color);
}
WC
o durchdringt Shadow DOM
o begünstigt style API von WCs
o Default Werte überschreibbar
html {
--header: {
color: red;
size: 20px;
}
}
DPL
input {
@apply --header;
}
WC
o Gruppierung von styles möglich
o schlechtere Browser Unterstützung


Can I use
?
Settings
Feature: CSS Variables (Custom Properties)
CSS Variables (Custom Properties) - CR
Permits the declaration and usage of cascading variables in stylesheets.
Usage
% of all users
?
Global
95.46% + 0.18% = 95.64%
Current aligned
Usage relative
Date relative
Filtered
All
IE
Edge
Firefox
Chrome
Safari
Opera
Safari on iOS
Opera Mini
Android Browser
Opera Mobile
Chrome for Android
Firefox for Android
UC Browser for Android
Samsung Internet
QQ Browser
Baidu Browser
KaiO Brows
12-14
4-47
3.1-9
10-34
3.2-9.2
15
2-30
48
9.1
35
9.3
4
6-10
16-91
31-89
49-91
10-14
36-76
10-14.4
2.1-4.4.4
12-12.1
5-13.0
11
92
90
92
14.1
77
14.7
all
91
62
91
89
12.12
14.0
10.4
7.12
2.5
91-92
93-95
15-TP

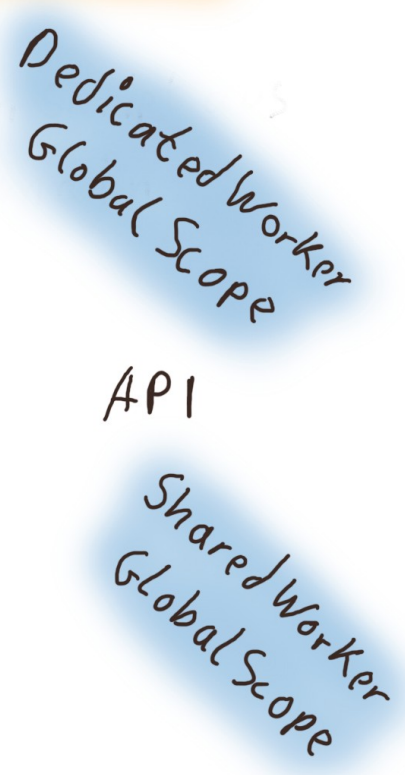

Can I use
?
Settings
Feature: CSS @apply rule
CSS @apply rule
- UNOFF
Allows a set of CSS properties to be applied using a named variable
Usage
% of all users
?
Global
0%
Current aligned
Usage relative
Date relative
Filtered
All
IE
Edge *
Firefox
Chrome
Safari
Opera
Safari on iOS *
Opera Mini *
Android Browser *
Opera Mobile *
Chrome for Android
Firefox for Android
UC Browser for Android
Samsung Internet
QQ Browser
Baidu Browser
KaiO Brows
12-18
4-50
10-37
4
6-10
79-91
2-89
51-91
3.1-14
38-76
3.2-14.4
2.1-4.4.4
12-12.1
5-13.0
11
92
90
92
14.1
77
14.7
all
91
62
91
89
12.12
14.0
10.4
7.12
2.5
91-92
93-95
15-TP

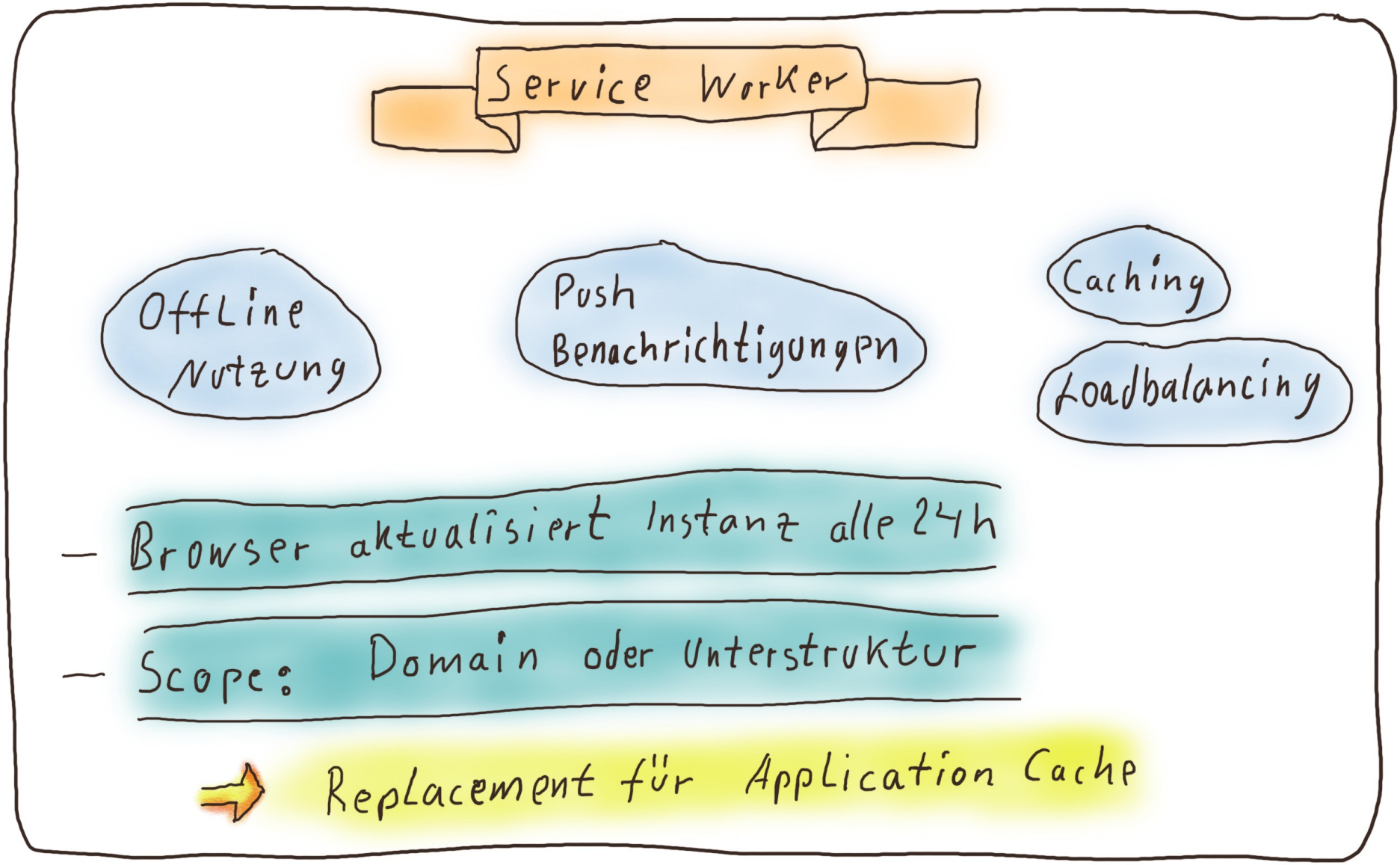

Support
Web worker
by Stencil
Dedicated Worker
Shared Worker
Service Worker
by Value
- Separater Thread
- separates Bundle durch Stencil
- einfache Nutzung
- Kommunikation mit Hauptthread über postMessage
stencil ermöglicht Callback Parampter
- nur HTTPS möglich
- Websocket nutzbar
- Keinzugiff auf DOM
- window eingeschränkt
- Storage nutzbar IndexedDB
Audio Worker = Audio Verarbeitung
Chrome Worker = Add on Entwicklung

Dedicated vs. Shared Worker

Beim Dedicated Worker kann nur vom erzeugenden Skript aus zugegriffen werden

Beim Shared Worker kann von jedem Skript aus der Domain zugegriffen werden




Service Worker
Offline Nutzung
Push Benachrichtigungen
Caching
Loadbalancing
- Browser aktualisiert Instanz alle 24h
- Scope: Domain oder Unterstruktur
Replacement für Application Cache

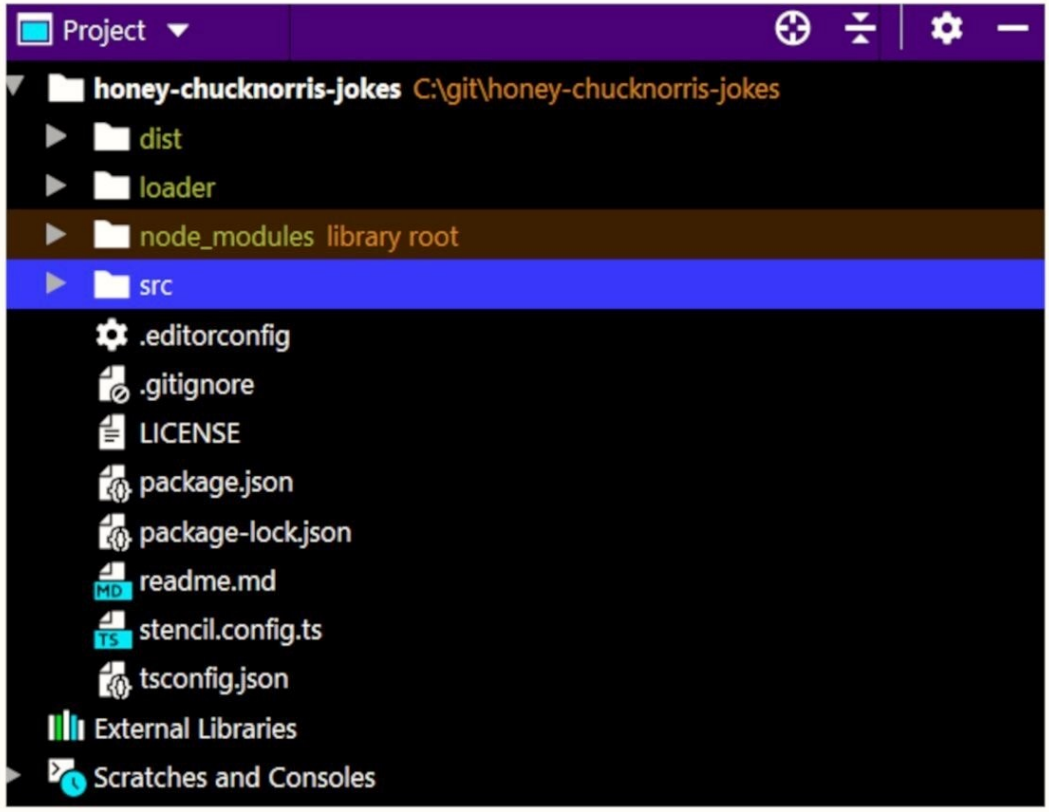
Stencil Features für Webworker

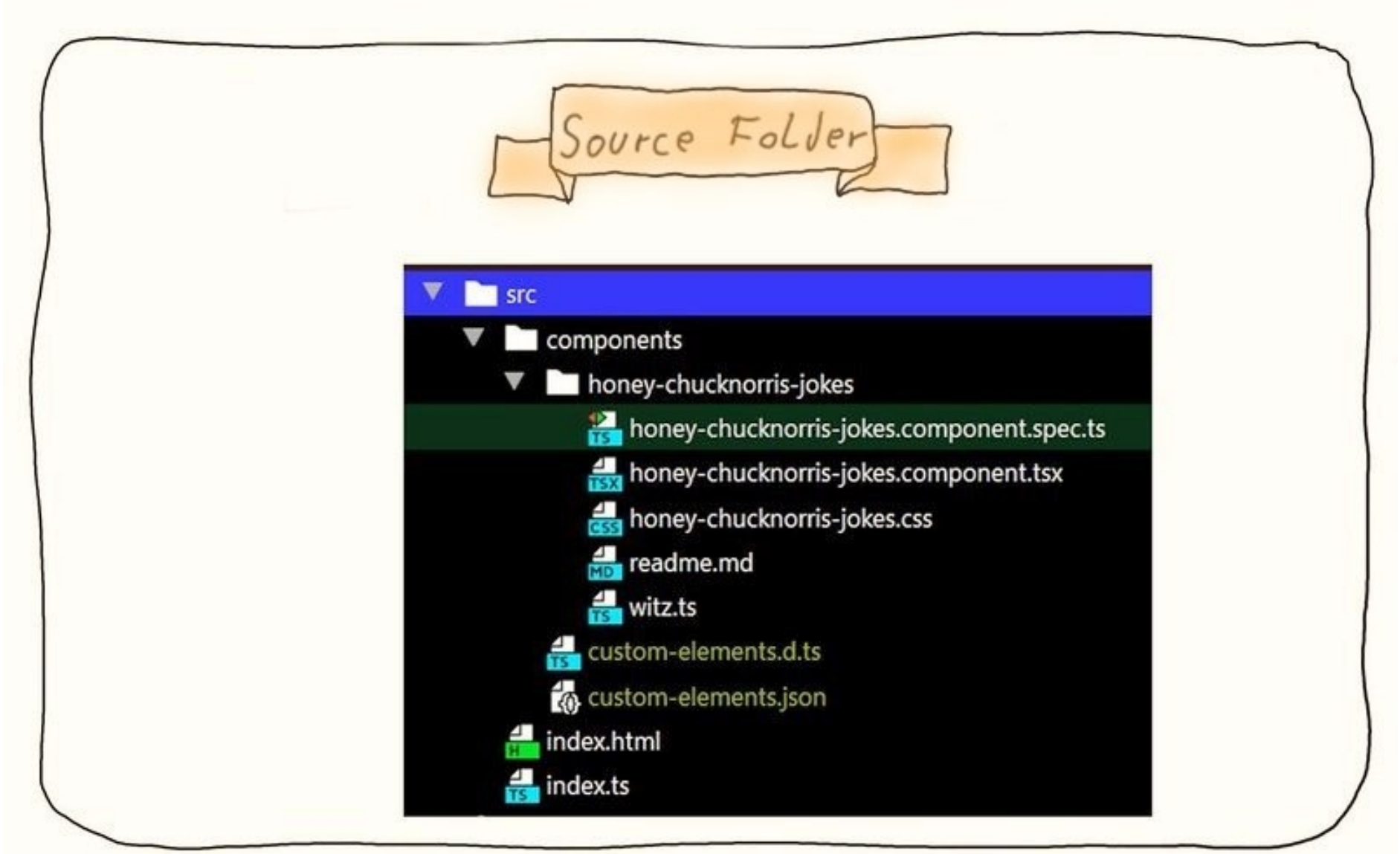
- Vereinfachte Kommunikation mit Hauptthread
- Worker Callbacks möglich
- Extra Bundle für Worker
- Importierte Libs werden beim Bundeling berücksichtigt
- ESM Importe werden unterstützt
- Dynamische Importe möglich

Webseite https://github.com/Huluvu424242/honey-chucknorris-jokes/tree/geparkt/stencil-slides

Lizenz MIT

Project struktur




Source Folder
src
components
honey-chucknorris-jokes
honey-chucknorris-jokes.component.spec.ts
honey-chucknorris-jokes.component.tsx
honey-chucknorris-jokes.css
readme.md
witz.ts
custom-elements.d.ts
custom-elements.json
index.html
index.ts

Definition – Komponente

```
@Component( opts: {
    tag: "honey-chucknorris-jokes",
    styleUrl: "honey-chucknorris-jokes.css",
    shadow: true
})
export class HoneyChucknorrisJokes {
```

Definition – State

```
/**
 * Host Element
 */
@Element() hostElement: HTMLElement;

/**
 * ID vom Host Tag
 */
ident: string;

fetcherSubscription: Subscription;

// Falls fachlich möglich Defaults hinterlegen um Logik einfach zu halten
@State() witz: Witz;

/**
 * Zeitintervall nachdem ein neuer Witz abgerufen wird (in Sekunden).
 */
@Prop( opts: {attribute: "period", reflect: false, mutable: true}) changePeriod: number = 20;
```

Render Method

```
public render() {
  this.printMessage("rendering");
  return (
    <Host
      id={this.ident}
    >
      <a href={this.witz.website} class={"container"} target={"blank"}>
        <img alt="Chuck" title="Funny icon of Chuck Norris" src={this.witz.imgurl} class={"item logo"}/>
        <p class={"item text"}>{this.witz.text}</p>
      </a>
    </Host>
  );
}
```

Hook – connectedCallback

```
public connectedCallback() {
  // attribute initialisieren wenn defaults notwendig
  this.ident = this.hostElement.id ? this.hostElement.id : Math.random().toString( radix: 36).substring(7);
  this.fetcherSubscription = this.subscribePeriodicFetcher();
  this.printMessage("DOM connected");
}
```

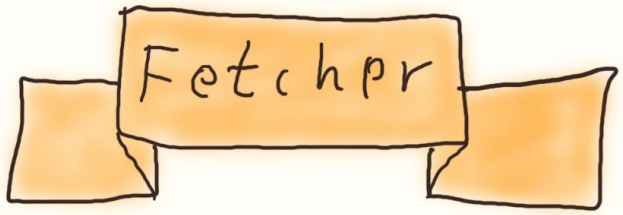
Hook – DisconnectedCallback

```
public disconnectedCallback() {
  this.fetcherSubscription.unsubscribe();
  this.printMessage("DOM disconnected");
}
```

Definition - Watcher

```
@Watch( propName: "changePeriod")
periodWatcher(newValue: number, oldValue: number) {
  this.printMessage("period changed old:" + oldValue + " new:" + newValue);
  if (newValue && oldValue !== newValue) {
    this.changePeriod = newValue;
    this.fetcherSubscription.unsubscribe();
    this.fetcherSubscription = this.subscribePeriodicFetcher();
    this.printMessage("period changed to:" + this.changePeriod);
  }
}
```


```
public async componentWillLoad() {
  // async damit vor Rendering auf das Laden der Daten gewartet wird
  this.printMessage("Lade Daten");
  // Fehler behandeln -> sonst dauerhaft kein Rendering
  await lastValueFrom(this.fetchWitz$()).catch(() => {
    this.setWitz(HoneyChucknorrisJokes.FALLBACK_WITZ)
  });
  this.printMessage("Daten geladen");
}
```




```
protected fetchWitz$(): Observable<Response> {
  return fromFetch(HoneyChucknorrisJokes.CHUCK_NORRIS_API_URL).pipe(
    switchMap(
      project: (response: Response) => response.json()
    ),
    tap(
      next: (data: any) => this.setWitz(data)
    ),
    catchError( selector: () => EMPTY)
  )
}
```


Periodic Fetcher
protected subscribePeriodicFetcher(): Subscription {
const timerPeriod: number = this.changePeriod * 1000;
const fetcher$: Observable<Response> = timer(timerPeriod, timerPeriod).pipe(
tap(
next: () => this.printMessage("neuen Witz angefordert")
),
switchMap
(
project: () => this.fetchWitz$()
),
)
;
return fetcher$.subscribe();
}

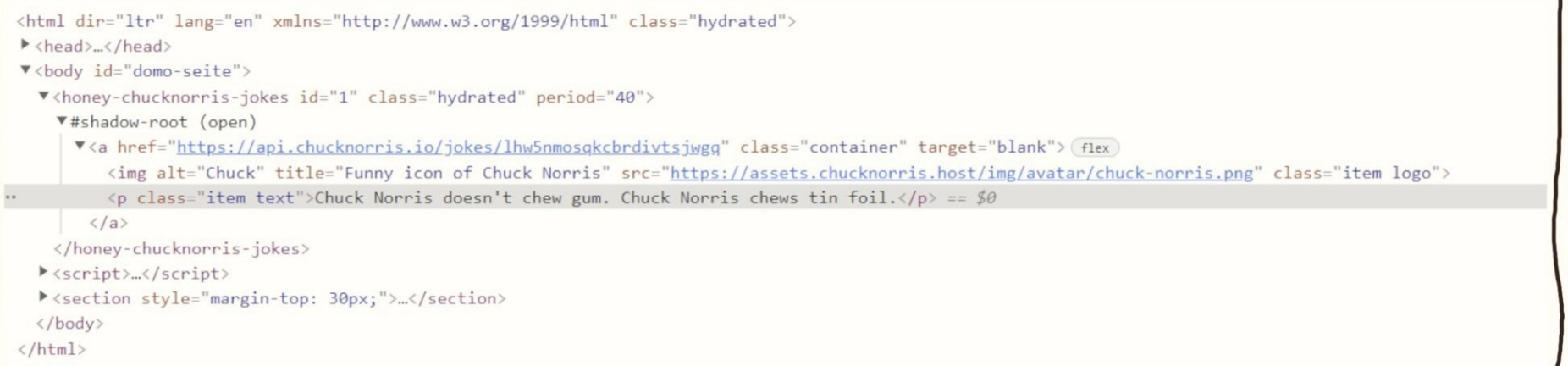
Definition - Setter

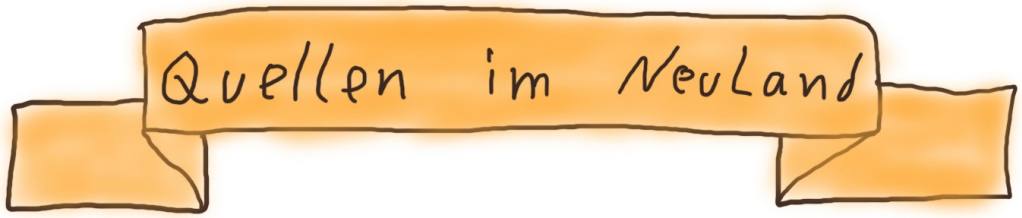
```
protected setWitz(data: any): void {
  this.printMessage("setze neuen  Witz");
  if (data) {
    // trigger rendering nur wenn ref changed
    this.witz = {
      id: data.id,
      imgurl: data.icon_url,
      website: data.url,
      text: data.value
    };
  } else {
    this.printMessage("konnte Witz nicht setzen")
  }
}
```

Einbinden der Webkomponente

```html
<!doctype html>
<html dir="ltr" lang="de" xmlns="http://www.w3.org/1999/html">
<head>
    <meta charset="UTF-8">
    <title>Honey Chuck Norris Jokes Beispiel</title>
    <script type="module" src="./build/honey-chucknorris-jokes.esm.js"></script>
    <script nomodule="" src="./build/honey-chucknorris-jokes.js"></script>
</head>
<body id="demo-seite">
    <honey-chucknorris-jokes id="1"></honey-chucknorris-jokes>
</body>
</html>
```

DOM Integration





Stencil Doku
* Stencil Home: https://stenciljs.com/docs/introduction
Rendering
* Browser Engine: https://valerii-udodov.com/posts/web-browser-anatomy/
* SGR,SSR,Hydrated: https://developers.google.com/web/updates/2019/02/rendering-on-the-web
* Virtual vs. Incremental DOM:
 Neutral: https://blog.bitsrc.io/incremental-vs-virtual-dom-eb7157e43dca
 ProGoogle: https://blog.nrwl.io/understanding-angular-ivy-incremental-dom-and-virtual-dom-243be844bf36
** Diskurs https://svelte.dev/blog/virtual-dom-is-pure-overhead
Shared Styles
* https://www.smashingmagazine.com/2016/12/styling-web-components-using-a-shared-style-sheet/
* https://developers.google.com/web/updates/2019/02/constructable-stylesheets