

Frontend ohne Framework – nativ, nicht naiv!

ObjektForum Karlsruhe, September 2023




andrena
OBJECTS

andrena objects

Jan Müller






Frontend ohne Framework – nativ, nicht naiv!

- Unsere Erfahrungen
- Frameworks
- WebComponents als Alternative
 - Custom Elements 
 - Shadow DOM  
 - Mit Lit
 - Weitere Aspekte
- Fazit



Frontend ohne Framework – nativ, nicht naiv!

- **Unsere Erfahrungen**
- Frameworks
- WebComponents als Alternative
 - Custom Elements 
 - Shadow DOM  
 - Mit Lit
 - Weitere Aspekte
- Fazit






Erfahrungen aus unserem Projekt

- Website, statisches HTML aus CMS
- Multi-Page Application in angularJS
- Modularer Aufbau, Baukastenprinzip
- angularJS ablösen, aber wie? Angular2+? React?
 - Frontend modernisieren, aber lieferfähig bleiben

→ Erfolgreiche Migration auf WebComponents mit Lit

Frontend ohne Framework – nativ, nicht naiv!

- Unsere Erfahrungen
- **Frameworks**
- WebComponents als Alternative
 - Custom Elements 
 - Shadow DOM  
 - Mit Lit
 - Weitere Aspekte
- Fazit

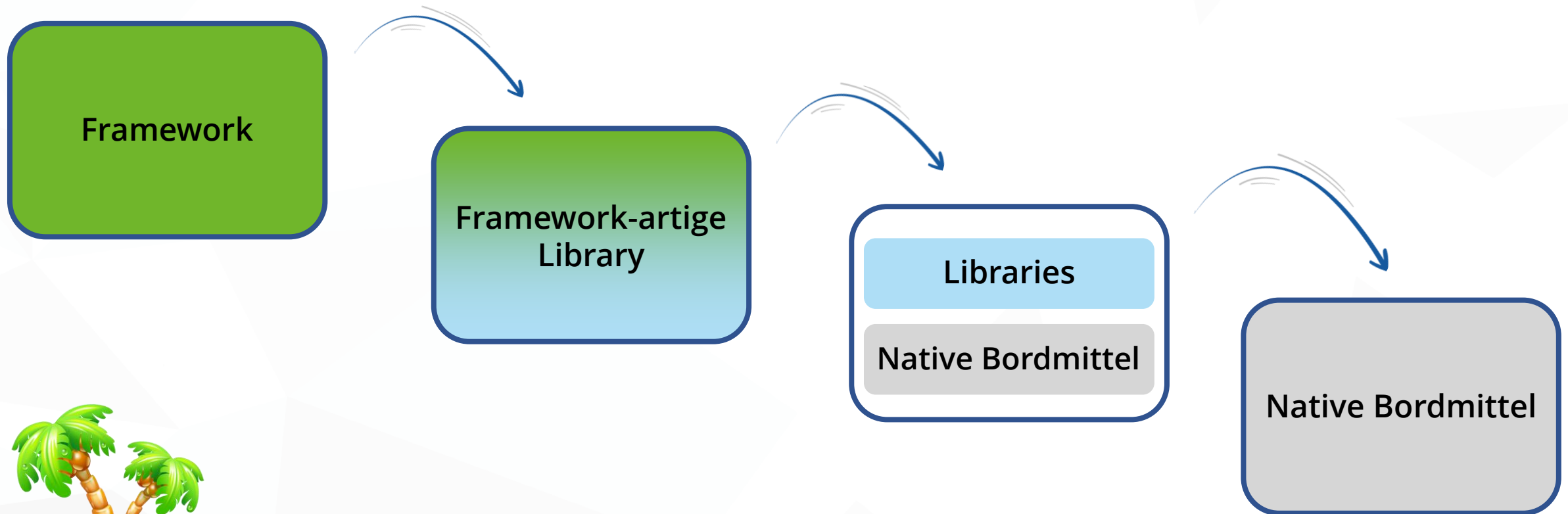


Was ist eigentlich ein Framework?



Was ist eigentlich ein Framework?




Verschiedene Abstufungen möglich



Was spricht gegen ein Framework?

- Framework-Lock-In
 - Wechsel zu anderer Technologie sehr teuer / unmöglich
 - Updates / Breaking Changes betreffen oft viele Ebenen
- Man kauft das Komplettpaket samt Nachteilen und Beschränkungen
 - Workarounds
 - Was ist das jeweils „richtige“ Framework? Gibt es das überhaupt?
- Frameworks sind schwergewichtig

Frontend ohne Framework – nativ, nicht naiv!

- Unsere Erfahrungen
- Frameworks
- **WebComponents als Alternative**
 - Custom Elements 
 - Shadow DOM  
 - Mit Lit
 - Weitere Aspekte
- Fazit



Custom Elements

```
<my-component></my-component>
```

HTML Template

```
<template>  
<span>My Template!</span>  
</template>
```

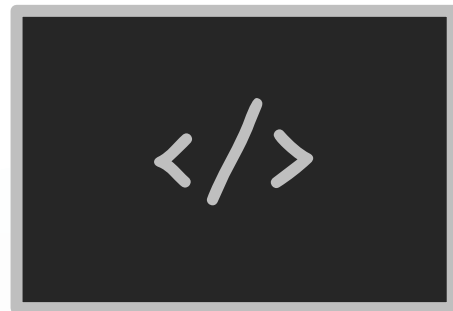
WebComponents

Shadow DOM

```
<my-component>  
#shadow-root (open)  
  <p>Shadow DOM content</p>  
</my-component>
```

ES Modules

```
<script type="module" src="myScript.js"/>
```



Code-Beispiel

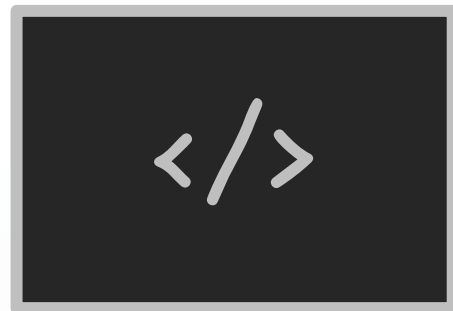
Custom Element

Shadow DOM

- „privater“ DOM einer Komponente
- Erlaubt u. a. Scoping von Styles
- Wird zur Laufzeit erzeugt

```
▼ <reise-basic reise-id="r1" name="Ans Meer" descriptio
  ▼ #shadow-root (open) == $0
    ▼ <div class="reise-tile"> flex
      ▶ <div class="tile-top"> ... </div>
      ▶ <div class="tile-bottom"> ... </div>
    </div>
  </reise-basic>
▼ <reise-basic reise-id="r2" name="In die Berge" descri
  ▼ #shadow-root (open)
    <!-->
    ▶ <div class="reise-tile"> ... </div> flex
  </reise-basic>
```

→ Isolierte Komponenten



Code-Beispiel

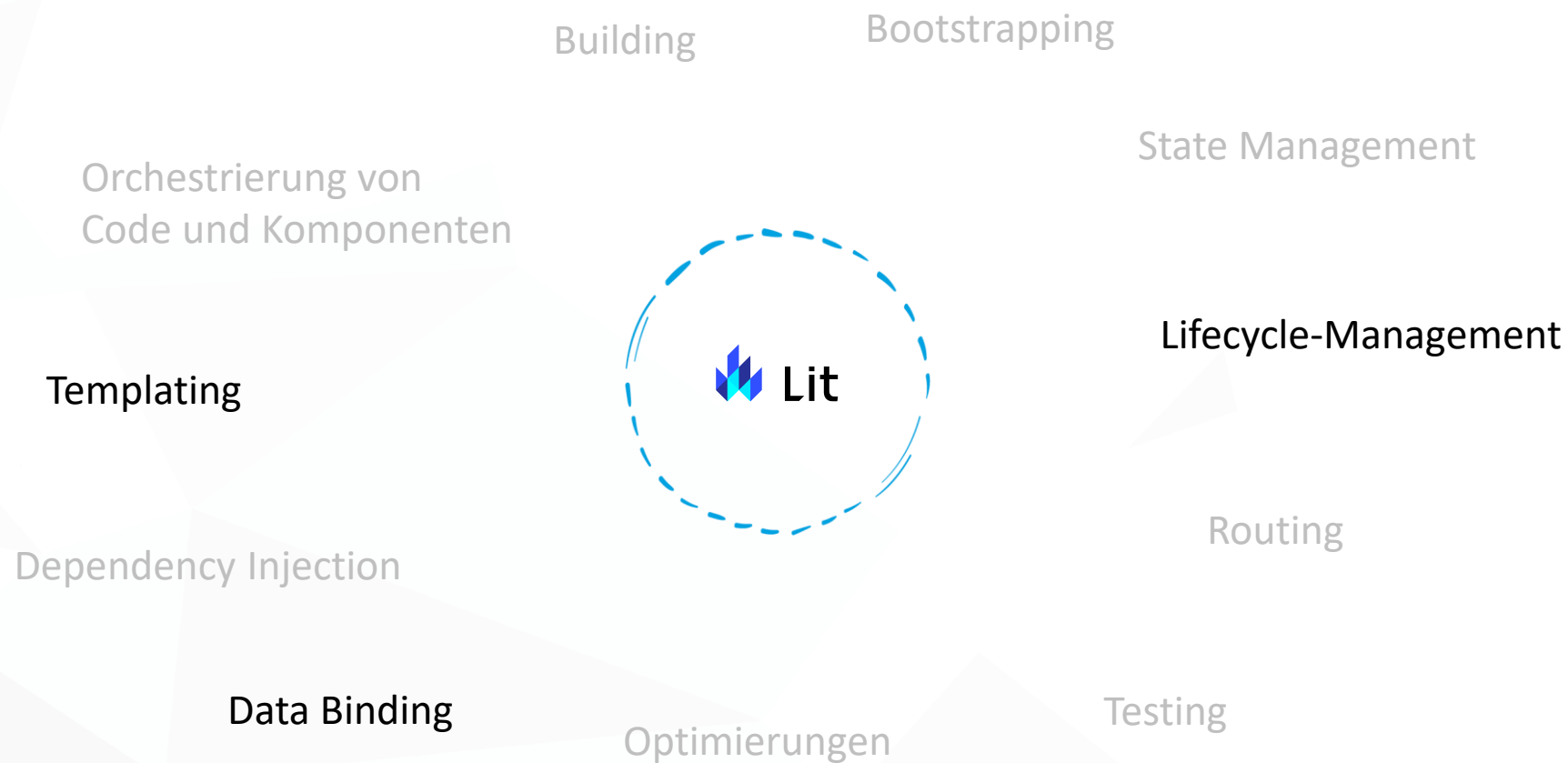
WebComponents mit Shadow DOM

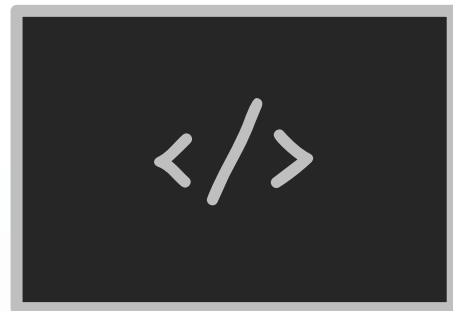
Herausforderungen ohne Framework

- Code wird sehr schnell unübersichtlich
- Framework-Aspekte müssen durch Libraries ersetzt werden
 - Mehr Freiheiten, aber auch Qual der Wahl
- Eine Option zum Bauen von WebComponents: Lit

WebComponents mit Lit

- Bibliothek zum Erstellen von WebComponents





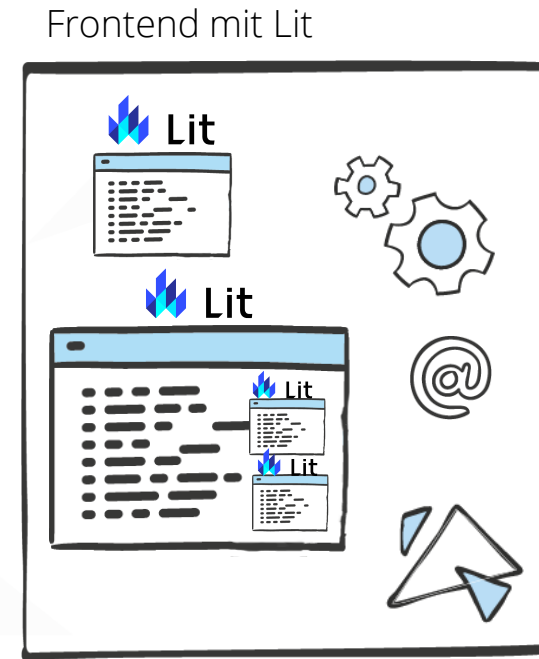
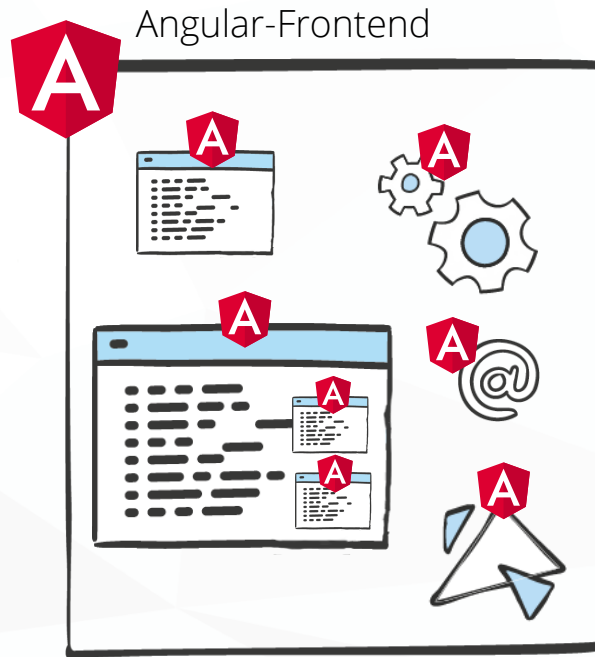
Code-Beispiel

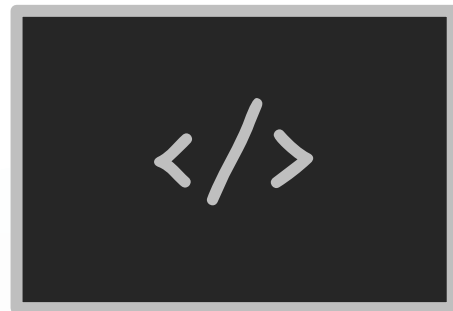
Lit

WebComponents mit Lit

- Bibliothek zum Erstellen von WebComponents
- Abstufung zu einem Framework:

Auswirkung beschränkt sich nur auf meine Komponente



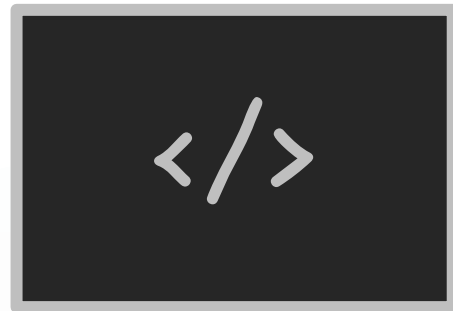


Code-Beispiel

Kleine Applikation

Weitere Aspekte




- Dependency Injection: tsyringe
- HTTP: fetch-API
- Testing: open-wc testing helpers
- Routing: router.js
- ...



Code-Beispiel

Kleine Applikation revisited

Frontend ohne Framework – nativ, nicht naiv!

- Unsere Erfahrungen
- Frameworks
- WebComponents als Alternative
 - Custom Elements 
 - Shadow DOM  
 - Mit Lit
 - Weitere Aspekte
- Fazit



WebComponents mit Libraries gegenüber Frameworks

- Vorteile:
 - leichtgewichtig
 - nah am Browser / Standard
 - langlebig
 - individuell gestaltbar, viel Freiheit
- Nachteile:
 - mehr, um das man sich kümmern muss

Zusammenfassung

- Frameworks sind wie All-inclusive-Pakete, mit allen Vor- und Nachteilen
- Direkter Ansatz mit WebComponents kann eine valide Alternative sein
- Man kann mit einer WebComponents-Lösung
 - frisch anfangen
 - aber auch schrittweise migrieren

Vielen Dank fürs Zuhören!

- WebComponents: <https://www.webcomponents.org/>
- Lit: <https://lit.dev/>
- Code: <https://github.com/andrena/webfrontend-ohne-framework>



Quellen

- WebComponents: <https://www.webcomponents.org/>
- Lit: <https://lit.dev/>

Icons

- <https://angular.io/assets/images/logos/angular/angular.svg>
- <https://angular.io/assets/images/logos/angularjs/AngularJS-Shield.svg>
- <https://raw.githubusercontent.com/reactjs/reactjs.org/main/src/icons/logo.svg>
- <https://github.com/vuejs/art/blob/9f8ee218d32b52c2d150212f22798a7a61689361/logo.svg>
- <https://github.com/sveltejs/branding/blob/master/svelte-logo.svg>
- <https://lit.dev/images/logo.svg>
- https://www.iconfinder.com/icons/993791/beach_leisure_palm_travel_vacation_icon