

Web Components

Custom Elements, Templates und Shadow Dom

Informatik – 3. Semester

1 Einführung

3 Custom Elements

5 Lifecycle-Callbacks

2 Templates & Shadow DOM

4 Slots

6 Übung: CustomProfileCard

Wiederverwendbare, gekapselte HTML-Bausteine

- Code wiederverwenden zu können spart Zeit
- Als Web-Developer:innen, haben wir uns vorgenommen, eine kleine Bibliothek mit fertig-gestylten HTML-Code-Snippets zu bauen, die wir häufig in Projekten wiederverwenden können.
- Als Übung wollen wir testweise einen Container erschaffen, der sämtlichen inneren Text blau einfärbt, unabhängig vom globalen Styling der Seite.

1 Einführung

3 Custom Elements

5 Lifecycle-Callbacks

2 **Templates & Shadow DOM**

4 Slots

6 Übung: CustomProfileCard

HTML-Schnipsel definieren und vor globalem CSS schützen

- Mit dem `<template>`-Element können Sie HTML-Schnipsel definieren, die vom Browser nicht gerendert werden
- Per JavaScript können Sie Templates auswählen und anschließend den Inhalt duplizieren und wiederverwenden

```
<template id="blue-div">
  <div>
    <style>
      div {color:blue;}
    </style>
    This is inner text;
    <p>This is a paragraph.</p>
  </div>
</template>
```

```
<div id="template-insert"></div>
<script>
  const blueDiv = document.querySelector(
    "#blue-div").content;
  const insert = document.querySelector(
    "#template-insert");
  for (var n = 0; n < 2; n++) {
    var clone = blueDiv.cloneNode(true);
    body.appendChild(clone);
  }
</script>
```

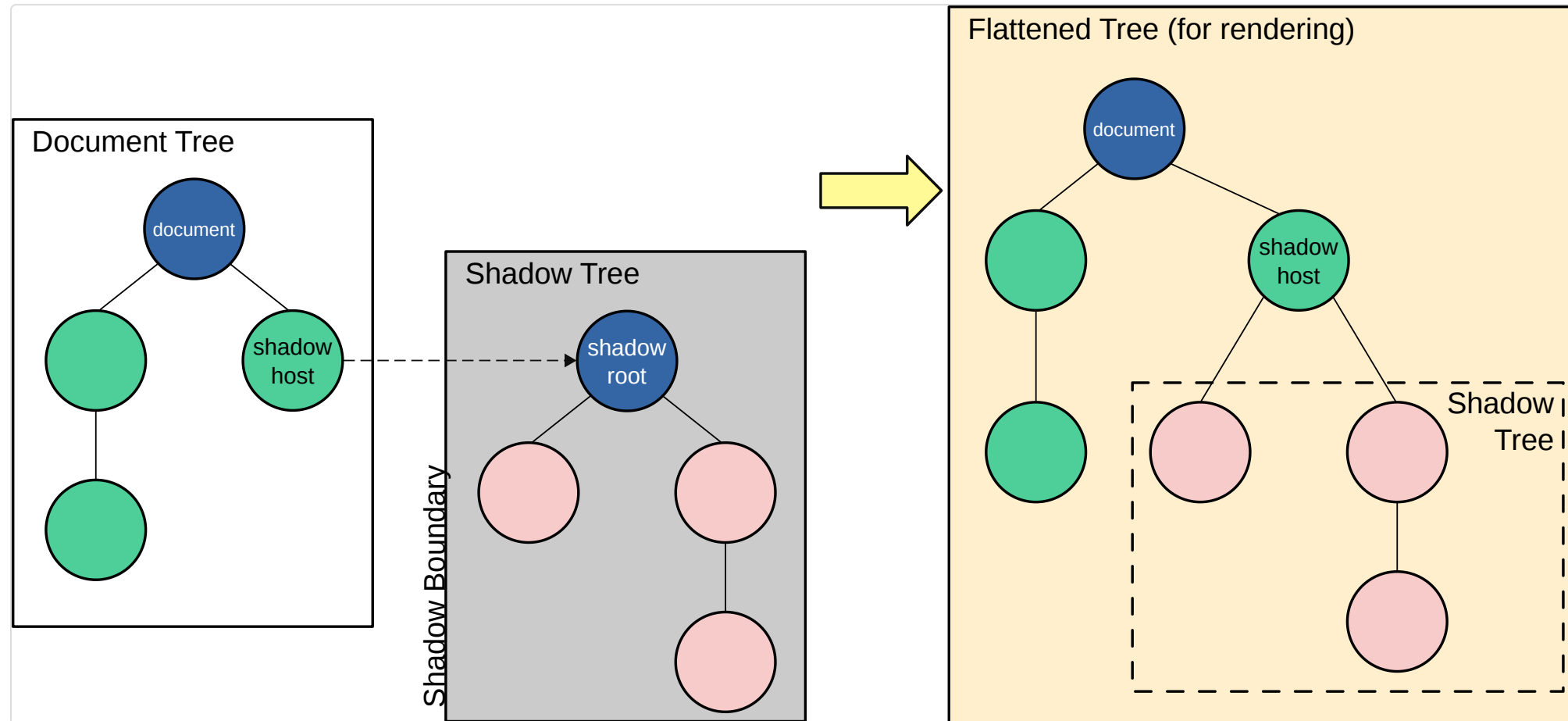
This is inner text;
This is a paragraph.
This is inner text;
This is a paragraph.

- Existierende CSS-Selektoren können mit den eingefügten Elementen überlappen
- Hier färbt der `<p>`-Selektor unsere eingefügten `<p>`-Element rot ein, obwohl wir sie im `<template>`-Element blau gefärbt haben

```
<style> p {color: red;} </style>  
<div id="template-insert"></div>
```

This is inner text;
This is a paragraph.
This is inner text;
This is a paragraph.

- Ein Shadow-DOM ist ein verkapselter DOM-Baum, der vor externem JavaScript und CSS versteckt ist
 - Ein Shadow-DOM wird der Haupt-DOM als verkapseltes Kind untergeordnet.
- Das ermöglicht das Wiederverwenden von HTML-Code Schnipseln, unabhängig von externem Styling



```
<div id="template-insert"></div>
<script>
  const blueDiv = document.querySelector(
    "#blue-div").content;
  const insert = document.querySelector(
    "#template-insert");
  // Hier passiert die Magie
  const shadow = insert.attachShadow(
    { mode: "open" })
  for (var n = 0; n < 2; n++) {
    var clone = blueDiv.cloneNode(true);
    shadow.appendChild(clone);
  }
</script>
```

This is inner text;
This is a paragraph.
This is inner text;
This is a paragraph.

1 Einführung

3 Custom Elements

5 Lifecycle-Callbacks

2 Templates & Shadow DOM

4 Slots

6 Übung: CustomProfileCard

Eigene HTML-Elemente als wiederverwendbare Klassen

```
class BlueDiv extends HTMLElement {  
  constructor() {  
    super();  
    const shadow = this.attachShadow({ mode: 'open' });  
    const div = document.createElement('div');  
    div.innerHTML = `  
      <style> div {color: blue} </style>  
      This is inner text;  
      <p>This is a paragraph.</p>  
    `;  
    shadow.append(div);  
  }  
}  
customElements.define('blue-div', BlueDiv);
```

```
<style> p {color: red;} </style>  
<blue-div> </blue-div>
```

This is inner text;
This is a paragraph.

- Unser eigenes HTML-Element in einer Seite
- Das `<p>`-Element von unserem `<blue-div>`-Element wird vom globalen Selektor dank der Shadow Root ignoriert

- 1 Einführung
- 3 Custom Elements
- 5 Lifecycle-Callbacks
- 2 Templates & Shadow DOM
- 4 Slots**
- 6 Übung: CustomProfileCard

Inhalte von außen in eine Komponente einspeisen

- Mit dem `<slot>`-Element können Sie HTML-Code in existierende Templates oder Custom Elements injizieren

```
<template id="info-card-tpl">
  <div>
    <slot>Kein Inhalt angegeben</slot>
  </div>
</template>
<script>
  customElements.define('info-card',
    class extends HTMLElement {
      constructor() {
        super();
        const root = this.attachShadow(
          { mode: 'open' });
        const tpl = document.querySelector(
          '#info-card-tpl');
        root.appendChild(
          tpl.content.cloneNode(true));
      }
    });
</script>
```

```
<info-card></info-card>
<info-card>Hallo Welt!</info-card>
```

Kein Inhalt angegeben

Hallo Welt!

```
<template id="info-card-tpl">
  <div>
    <h3><slot name="title">
      Titel</slot></h3>
    <slot>Kein Inhalt</slot>
  </div>
</template>
```

- Ein `slot="..."` -Attribut im Light-DOM verweist auf den passenden `name` im `<slot>` -Element
- So lassen sich mehrere Einfüge-Stellen in einer Komponente unterscheiden

```
<info-card>
  <span slot="title">
    Wichtiger Hinweis</span>
    Der Server ist um 22 Uhr
    offline.
</info-card>
```

Wichtiger Hinweis

Der Server ist um 22 Uhr offline.

- 1 Einführung
- 3 Custom Elements
- 5 Lifecycle-Callbacks**
- 2 Templates & Shadow DOM
- 4 Slots
- 6 Übung: CustomProfileCard

Wann wird ein Custom Element aktiv?

Callback	Wann?	Zweck
<code>constructor()</code>	Beim Erzeugen des Objekts, noch nicht im DOM	Shadow-Root anlegen, Grundzustand
<code>connectedCallback()</code>	Element wird in den DOM eingefügt	Attribute auslesen, Inhalt aufbauen
<code>disconnectedCallback()</code>	Element wird aus dem DOM entfernt	Aufräumen (Listener, Timer)
<code>attributeChangedCallback()</code>	Ein beobachtetes Attribut ändert sich	Auf die Änderung reagieren
<code>static get observedAttributes()</code>	Einmalig beim Registrieren der Klasse	Legt fest, welche Attribute beobachtet werden

Attribute erst im `connectedCallback` auslesen, nicht im Constructor: die Spezifikation garantiert dort noch keine vollständig geparsen Attribute.

```
class ColorBox extends HTMLElement {
  static get observedAttributes() {
    return ['color'];
  }

  connectedCallback() {
    this.style.display = 'block';
    this.style.backgroundColor =
      this.getAttribute('color');
  }

  attributeChangedCallback(name, oldVal, newVal) {
    if (name === 'color') {
      this.style.backgroundColor = newVal;
    }
  }

  disconnectedCallback() {
    console.log('color-box entfernt');
  }
}
customElements.define('color-box', ColorBox);
```

- 1 Einführung
- 3 Custom Elements
- 5 Lifecycle-Callbacks
- 2 Templates & Shadow DOM
- 4 Slots
- 6 Übung: CustomProfileCard**

Die Profilkarte selbst bauen

- Implementieren Sie das HTML-Element "Profile Card", dass ein Bild, einen Namen und eine Rolle als Attribut annimmt und ein HTML-Gerüst für eine Profilkarte zurückgibt
- Die Profilkarte soll Slot Element für einen Beschreibungstext enthalten
- Binden Sie das mitgelieferte Stylesheet in die Profilkarte ein
- Welche Elemente die Karte enthält entnehmen Sie bitte diesem Stylesheet
- **Lernraumkurs → Zusatzmaterial → Übung: CustomProfileCard**



20 Min

- **Schritt 1:** Stylesheet als `<link>`-Element erzeugen und in den Shadow-DOM einhängen
- **Schritt 2:** Card-Grundgerüst (`div.card`) bauen
- **Schritt 3:** Attribute (`name` , `img` , `role`) auslesen und in die passenden Elemente schreiben
- **Schritt 4:** `<slot name="description">` für den mitgelieferten Beschreibungstext ergänzen
- Alle vier Schritte passieren im `connectedCallback` , nicht im Constructor

```
connectedCallback() {  
  const shadow = this.shadowRoot;  
  
  const link = document.createElement('link');  
  link.setAttribute('rel', 'stylesheet');  
  link.setAttribute('href', 'profile-card.css');  
  shadow.appendChild(link);  
  
  const card = document.createElement('div');  
  card.classList.add('card');  
  
  const img = document.createElement('img');  
  img.setAttribute('src', this.getAttribute('img'));  
  card.appendChild(img);  
}
```

```
const name = document.createElement('h2');
name.textContent = this.getAttribute('name');
card.appendChild(name);

const role = document.createElement('small');
role.textContent = this.getAttribute('role');
card.appendChild(role);

const slot = document.createElement('slot');
slot.setAttribute('name', 'description');
card.appendChild(slot);

shadow.appendChild(card);
}
```

```
<profile-card name="Ada Lovelace"  
  role="Mathematician"  
  img="ada.jpg">  
  <p slot="description">  
    Erste Programmiererin.  
  </p>  
</profile-card>
```

