FRONTEND DEVELOPMENT

WINTERSEMESTER 2022

WEBCOMPONENTS

- Introduced 2011
- Set of features developed to allow the creation of reusable widgets and components
- Bring component based software engineering to the WWW

WEBCOMPONENTS
FEATURES

- HTML Templates
  - reuse HTML fragments
- Shadow DOM
  - encapsulate DOM and styling
- Custom Elements
  - APIs to create new HTML elements

WEBCOMPONENTS
FIRST COMPONENT

```
class MyComponent extends HTMLElement {
}
//                    ^^^^^^^^^^^^
// a webcomponent needs to extend from HTMLElement

customElements.define('fhs-component', MyComponent)
// 1) ^^^^^^^^^^^^^^^
// 2)                  ^^^
// 3)                                ^^^^^^^
// 1) define a new custom component
// 2) custom elements need to contain a dash. Usually prefixed with project name
// 3) use component
```

WEBCOMPONENTS

FIRST COMPONENT

```
class MyComponent extends HTMLElement {
    connectedCallback() {
//  ^^^^^^^^^^^^^^^^^
// called when component gets rendered

        this.innerHTML = 'hello world!';
//      ^^^^^^^^^^^^^^
// add some content to the body of the component
    }
}
```

Webcomponents

Render the component

- simply render the component in the HTML file
- webcomponents need to have a closing tag

```
<fhs-component></fhs-component>
```

WEBCOMPONENTS
NESTED COMPONENTS

```
class MyComponent extends HTMLElement {
    connectedCallback() {
        this.innerHTML = '<my-nested-component></my-nested-component>';
        //                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        // render nested components as you would in an HTML page
    }
}
```

WEBCOMPONENTS
ATTACH EVENT LISTENERS

```
class MyComponent extends HTMLElement {
    connectedCallback() {
        this.innerHTML = `
           <a href="/test">hallo</a>
        `;

        const aElement = this.querySelector('a')
        //                    ^^^^^^^^^^^^^^^^^^
        // query selector is only scoped to elements
        // in this Webcomponent

        // attach event listener
        aElement.addEventListener('click', (evt) => {
            evt.preventDefault()
            console.log("hallo")
        })
    }
}
```

WEBCOMPONENTS

ATTACH EVENT LISTENERS

```
class MyComponent extends HTMLElement {
    connectedCallback() {
        this.innerHTML = `
           <a href="/test">hallo</a>
        `;

        const aElement = this.querySelector('a')
        //                    ^^^^^^^^^^^^^^^^^^
        // query selector is only scoped to elements
        // in this Webcomponent

        // attach event listener
        aElement.addEventListener('click', (evt) => {
            evt.preventDefault()
            console.log("hallo")
        })
    }
}
```

WEBCOMPONENTS

ATTACH EVENT LISTENERS

```
class MyComponent extends HTMLElement {
    someAttribute = ''
//  ^^^^^^^^^^^^^^
// define a default value

    static get observedAttributes() {
        return ['some-attribute']
//             ^^^^^^^^^^^^^^^^^^
// define attributes which can be provided
    }

    attributeChangedCallback(property, oldValue, newValue) {
//  ^^^^^^^^^^^^^^^^^^^^^^^^
// whenever attribute changes this callback will be executed
        if (property === 'some-attribute') {
          this.someAttribute = newValue
        }
        this.connectedCallback()
// rerender component
    }

    connectedCallback() {
        this.innerHTML = `attribute value: ${this.someAttribute}`;
    }
}
```

WEBCOMPONENTS

RENDER A LIST OF COMPONENTS

```
class MyComponent extends HTMLElement {
    connectedCallback() {
        const someArray = [1, 2, 3, 4]
        this.innerHTML = `
          <section>
              <ul>
                ${someArray.map((value) => `<li>${value}</li>`).join('')}
              </ul>
          </section>
        `;
    }
}
```

WEBCOMPONENTS

FETCH ASYNC DATA

```
class MyComponent extends HTMLElement {
    async connectedCallback() {
//  ^^^^^
// mark function as async

        const someArray = await fetch('/whatever')
//                        ^^^^^
// await the result

        this.innerHTML = `
          <section>
              <ul>
                ${someArray.map((value) => `<li>${value}</li>`).join('')}
              </ul>
          </section>
        `;
    }
}
```

HOMEWORK

- see wiki

Feedback

- Questions: tmayrhofer.lba@fh-salzburg.ac.at
- Feedback Link