The 18th International Conference on Advances in Databases, Knowledge, and Data Applications

DBKDA 2026

March 08, 2026 to March 12, 2026 - Valencia, Spain

A Short Web-Component Primer for Database Devolopers

Andreas Schmidt[1,3], Tobias Münch[2,4], Tizian Berger[3]

(1)
Karlsruhe Institute of Technology (KIT)
Germany

(2)
Münch Ges. für IT Solutions mbH,
Gewerbering 1, 49393 Lohne, Germany

(3)
Karlsruhe University of Applied Sciences
Germany

(4)
Chemnitz University of Technology
Germany

Web-Components

„Web Components are a W3C standard, consisting of different technologies that allow you to to extend HTML with your own, reusable elements, with dedicated functionality and apperance, which are encapsulated behind a declarative interface and can be used in your web apps.“

Web components are executed in the browser!!

Karlsruhe Institute of Technology

münch IT SOLUTIONS

UNIVERSITY OF TECHNOLOGY IN THE EUROPEAN CAPITAL OF CULTURE CHEMNITZ

Hochschule Karlsruhe University of Applied Sciences
Fakultät für Informatik und Wirtschaftsinformatik

Outline

- Introduction/Motivation
- Custom elements
- Shadow DOM
- Communication with a backend service
- Hands-on Exercise
- Summary & Outlook

Purpose of the Tutorial

After completing the tutorial (including exercise), the participants should be able to ...

- ... identify potential areas of application for web components.
- ... create their own web components.
- ... create web components that communicate with backend services.
- ... create declarative interfaces that enable the development of websites even for non-programmers.

Karlsruhe Institute of Technology

münch IT SOLUTIONS

UNIVERSITY OF TECHNOLOGY IN THE EUROPEAN CAPITAL OF CULTURE CHEMNITZ

Hochschule Karlsruhe University of Applied Sciences
Fakultät für Informatik und Wirtschaftsinformatik

Provided Materials

Website of the Tutorial:

https://www.smiffy.de/dbkda-2026/

- Presentation slides
- Exercise and Solution
- A bunch of example web-components from the slides (and some more ...)
- A simple development environment consisting of a Docker Container with a bind mount

Advantages of Web-Components

- Encapsulate programmatic logic behind a declarative Interface (HTML elements represents an intuitive and simple interface to the implemented program logic)
- Enable the development of domain-specific solutions that make it even easier for domain experts without programming knowledge to develop their own applications or adapt existing solutions.
- Low-Code solution (reducing the burden on IT infrastructure)

Advantages of Web-Components

- Encapsulate programmatic logic behind a declarative Interface (HTML elements represents an intuitive and simple interface to the implemented program logic)
- Enable the development of domain-specific solutions that make it even easier for domain experts without programming knowledge to develop their own applications or adapt existing solutions.
- Low-Code solution (reducing the burden on IT infrastructure)
-

support declarative web development

Advantages of Web-Components

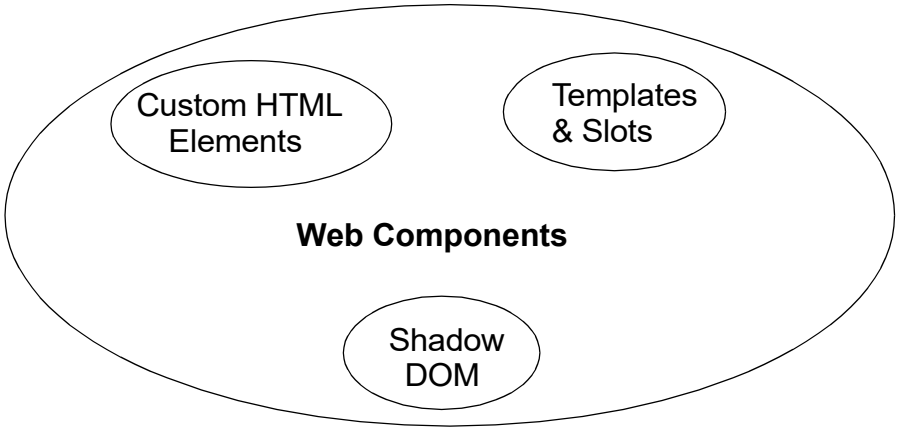
- Can be simply integrated in existing web applications (i.e. arbitrary ERP, CRM, ...)
- Independent of any specific web frameworks such as Vue, React, Angular, ... (but can also be used in conjunction with them[1])

1. see https://custom-elements-everywhere.com/ for some minor issues ...

Ingredients of Web-Components

- Web components are a suite of related technologies, bundled in a „Composite W3C standard“

- Ingreedients:
  - Custom Elements
  - Shadow DOM
  - Templates and slots



Custom Elements

Custom Elements

- Two types of Custom Elements:
  - Autonomous Custom Elements (inheriting from HTMLElement)
  - Customized built-in elements (inheriting from standard HTML-elements like HTMLParagraphElement (<p/>) , HTMLAnchorElement (<a/>), ...)

  Customized build-in elements are not supported by Safari and will not be covered in this tutorial

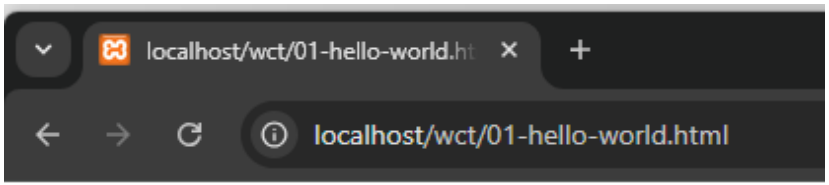
- Custom element name must have a dash (-) in its name (i.e. **`<hello-world>`**, **`<csv-table>`**, ...)
- Custom Elements must be registered for a page that uses it

A first minimal, but complete Example

```
<!DOCTYPE html>
<!-- file: 01-hello-world.html -->
<html>
  <head>
    <script>
      class Hello extends HTMLElement {

          connectedCallback() {
             var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
             this.innerHTML = `Hello World, it's ${day}`
          }
      }

      customElements.define('hello-world', Hello)
    </script>
  </head>
  <body>
    <hello-world></hello-world>
  </body>
</html>
```

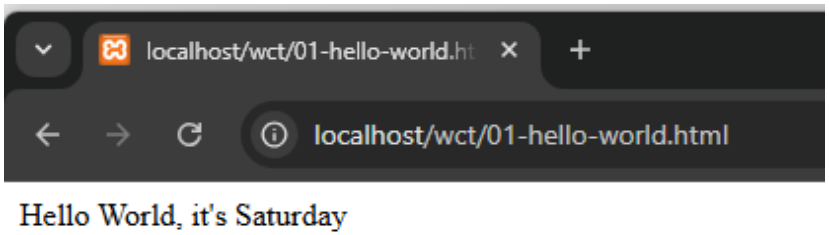
A first minimal, but complete Example

```
<!DOCTYPE html>
<!-- file: 01-hello-world.html -->
<html>
  <head>
    <script>
      class Hello extends HTMLElement {

          connectedCallback() {
            var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
            this.innerHTML = `Hello World, it's ${day}`
        }
      }

      customElements.define('hello-world', Hello)
    </script>
  </head>
  <body>
    <hello-world></hello-world>
  </body>
</html>
```



A first minimal, but complete Example

```
<!DOCTYPE html>
<!-- file: 01-hello-world.html -->
<html>
  <head>
    <script>
      class Hello extends HTMLElement {

          connectedCallback() {
             var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
             this.innerHTML = `Hello World, it's ${day}`
          }
      }

      customElements.define('hello-world', Hello)
    </script>
  </head>
  <body>
    <hello-world></hello-world>
  </body>
</html>
```



A first minimal, but complete Example

```
<!DOCTYPE html>
<!-- file: 01-hello-world.html -->
<html>
  <head>
    <script>
      class Hello extends HTMLElement {

          connectedCallback() {
            var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
            this.innerHTML = `Hello World, it's ${day}`
        }
      }

      customElements.define('hello-world', Hello)
    </script>
  </head>
  <body>
    <hello-world></hello-world>
  </body>
</html>
```

(1) Implement

(2) Register

(3) Use

A first minimal, but complete Example

```
<!DOCTYPE html>
<html>
  <head>
    <script type="module" src="./js/hello-world.js"></script>
  </head>
  <body>
    <hello-world></hello-word>
  </body>
</html>
```

```
class HelloWorld extends HTMLElement {

   connectedCallback() {
      var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
      this.innerHTML = `<h1>Hello World, today it's ${day}</h1>`
   }
}

customElements.define('hello-world', HelloWorld)
```

Custom Elements Lifecycle Callbacks [1]

- `constructor()`:
  - Call of constructor of parent class (`super()`)
  - Set up initial state/set of default values
  - Register event listeners
  - Create shadow root if needed
- `connectedCallback()`:
  - Called, when the element is added to the document (DOM-tree).
  - General custom element setup
  - At this stage, HTML-attribute values are visible
- `disconnectedCallback()`:
  - Called, when the element is removed from the document (DOM-tree).

Custom Elements Lifecycle Callbacks [1]

- `attributeChangedCallback(attributeName, oldValue, newValue)`:
  - called when a HTML-attribute is changed, added, removed
  - Only called for attributes, listed in the static property `observedAttributes`

- `connectedMoveCallback()`:
  - If existent, called, when element is moved to a different place inside the document (replaces calls to `disconnectedCallback`, `connectedCallback`)

- `adoptedCallback()`:
  - Called, when an element is moved to a new document

Working with Attributes

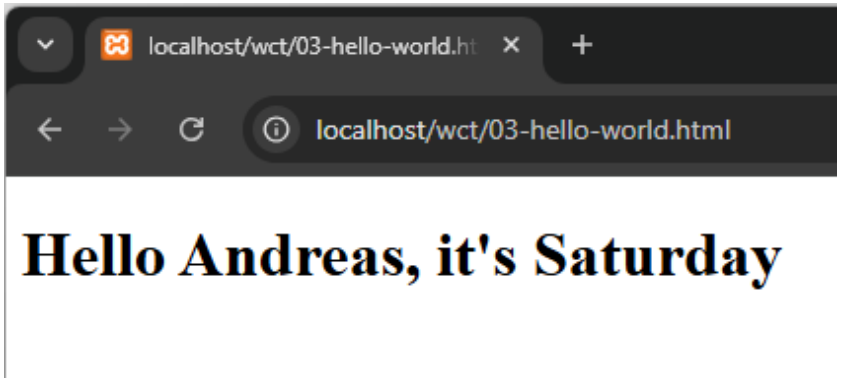
- Extension of the previous example by adding an HTML attribute that specifies the name of the person to be greeted.
- Add some HTML-formatting
- HTML-Code:

```
<!DOCTYPE html>
<html>
<head>
   <script type="module" src="./js/03-hello-world.js">
   </script>
</head>
<body>
   <hello-world name="Andreas"></hello-world>
</body>
</html>
```

Working with Attributes

- Extension of the previous example by adding an optional HTML attribute that specifies the name of the person to be greeted.
- Add some HTML-formatting
- HTML-Code:

```
<!DOCTYPE html>
<html>
<head>
   <script type="module" src="./js/03-hello-world.js">
   </script>
</head>
<body>
   <hello-world name="Andreas"></hello-world>
</body>
</html>
```



Custom Element Implementation

```
class Hello extends HTMLElement {

   static observedAttributes = ['name']

   constructor() {
      super()
      this['name'] = 'World'
   }
   connectedCallback() {
      var day = new Date().toLocaleDateString('en-US',{weekday: 'long'})
      this.innerHTML = `<h1>Hello ${this['name']}, it's ${day}</h1>`
   }
   attributeChangedCallback(property, oldValue, newValue) {
      if (oldValue === newValue)
         return
      this[ property ] = newValue
   }
}

customElements.define('hello-world', Hello)
```

So far ...

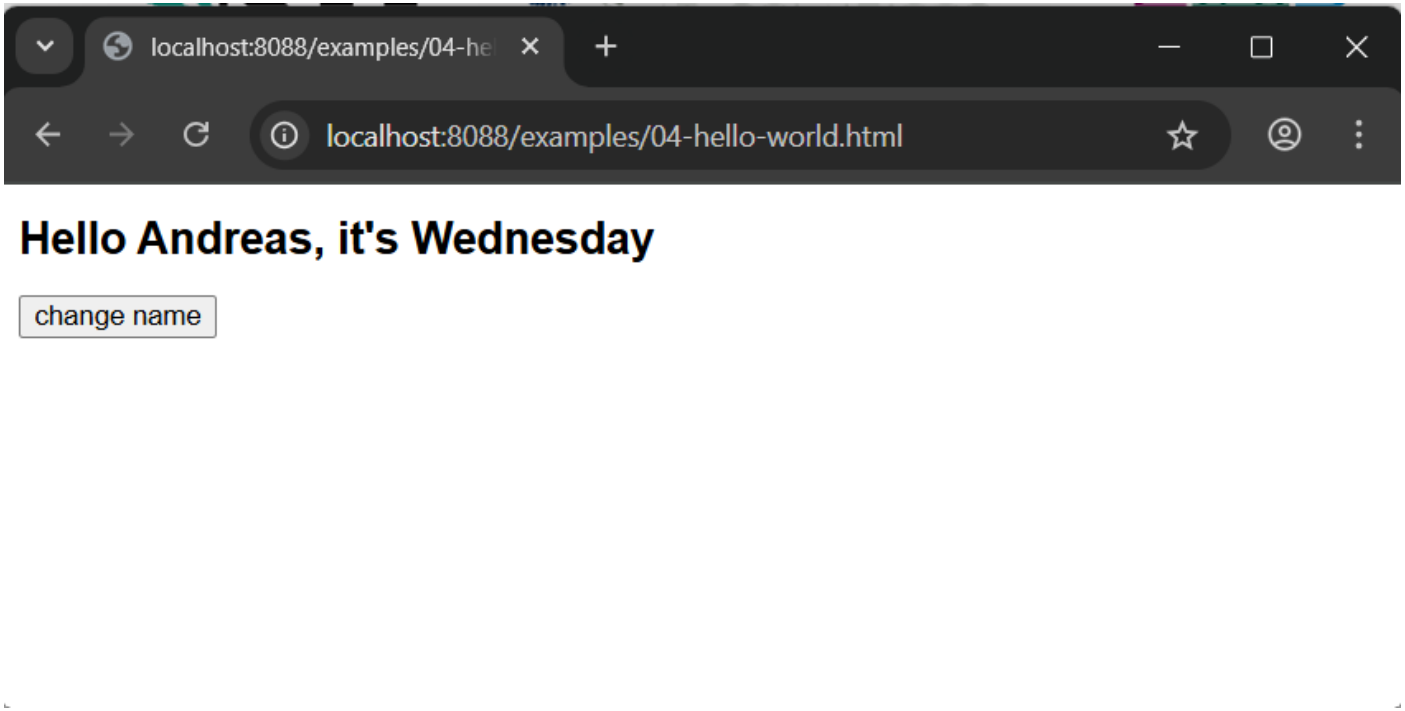
- A Web component consists of ...
  - Custom Element class, derived from class HTMLElement
  - Custom element class implements
    - ... a number of callback methods
    - ... the application logic
  - A mapping to an element-name (`customElements.define()` method)
- HTML Attributes and properties of the Custom Element classes are not the same (but typically mapped)
- Initial initialisation can be done in the in constructor
- General setup of web-components should be done in the `connectedCallback()` method

Dynamic Attribute Changes

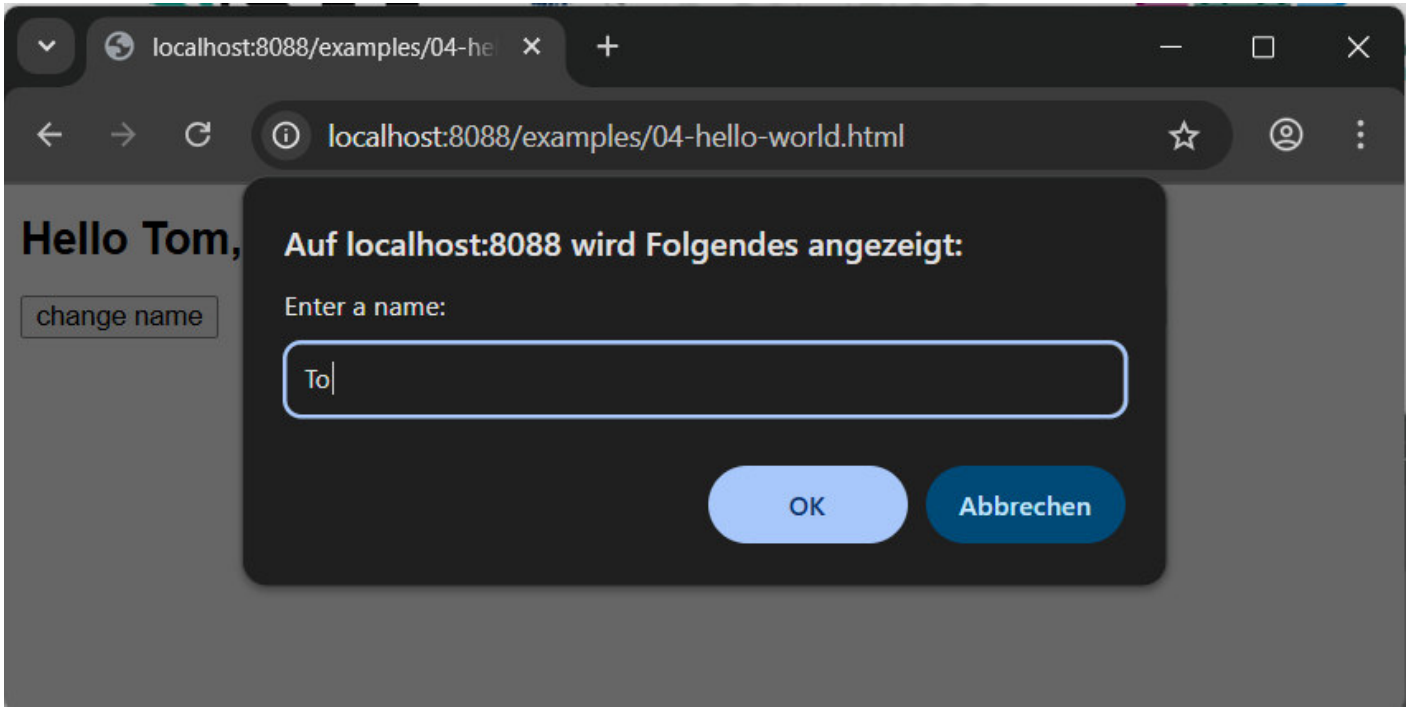
- The HTML attributes specified in `observedAttributes` are monitored for changes.
- Every time an attribute value is set, the method `attributeChangedCallback(property, oldValue, newValue)` is called.
- The attribute value of an HTML-element can be changed using the instance method `Element.setAttribute(name, new_value)`
- Example:

```
var new_name = prompt("Enter a name:")
const element = document.getElementsByTagName('HELLO-WORLD')[0]
element.setAttribute('name', new_name)
```

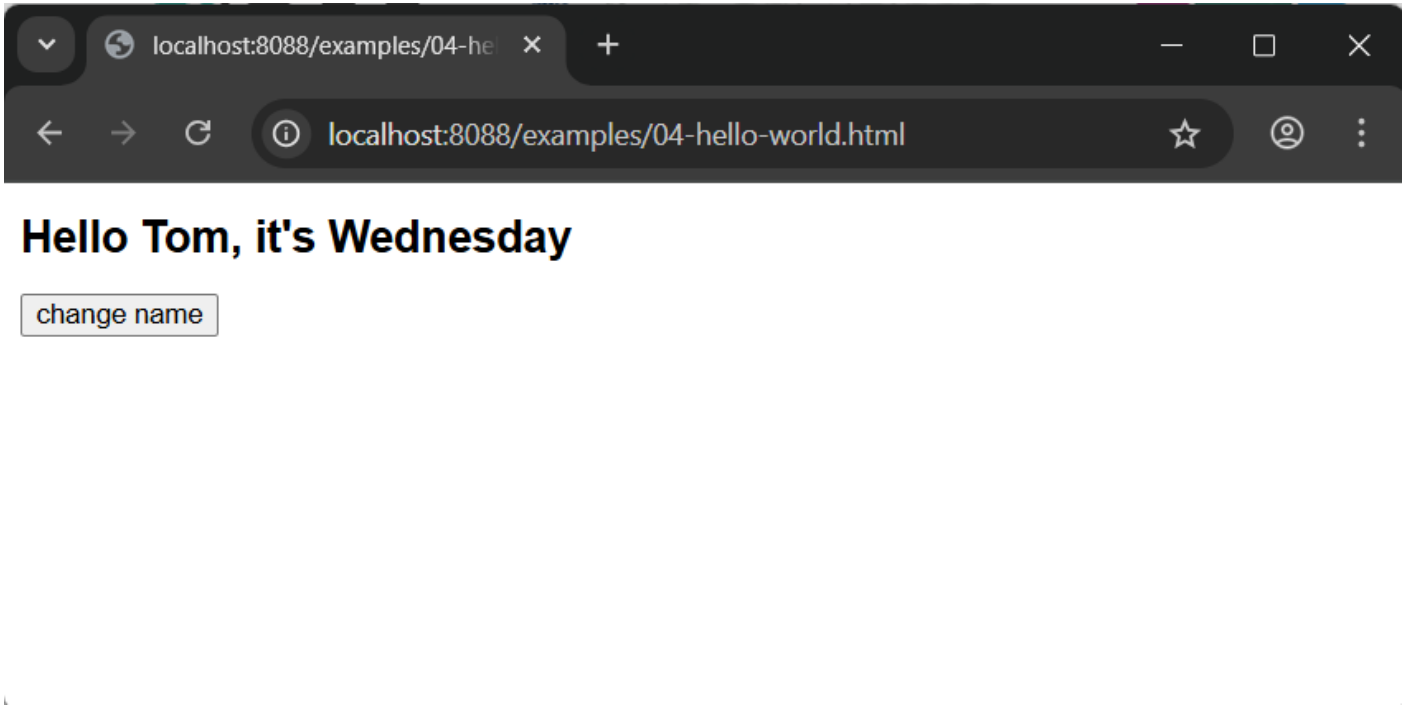
Dynamic Attribute Changes



Dynamic Attribute Changes



Dynamic Attribute Changes



Dynamic Attribute Changes

```
<!DOCTYPE html>
<html>
  <head>
    <script type="module" src="./js/Hello-v4.js"></script>
   <script>
       function change_name() {
          var new_name = prompt("Enter a name:")
          const element = document.getElementsByTagName('HELLO-WORLD')[0]
          element.setAttribute('name', new_name)
       }
   </script>
  </head>
  <body>
    <hello-world name="Andreas"></hello-world>
   <form>
       <input type="button" value="change name" onclick="change_name()"/>
   </form>
  </body>
</html>
```

Dynamic Attribute Changes

```
class Hello extends HTMLElement {

   static observedAttributes = ['name'];
   constructor() { ... }

   connectedCallback() {
       this.render()
   }
   render() {
      var day = new Date().toLocaleDateString('en-US',{weekday: 'long'});
      this.innerHTML = `<h1>Hello ${this.name}, it's ${day}</h1>`
   }
   attributeChangedCallback(property, oldValue, newValue) {
      if (oldValue === newValue)
         return;
      this[ property ] = newValue;
      this.render()
   }
}
customElements.define('hello-world', Hello);
```

Like the example on page 21, but with an explicit `render()` method, because we have multiple situations, where we have to update the GUI

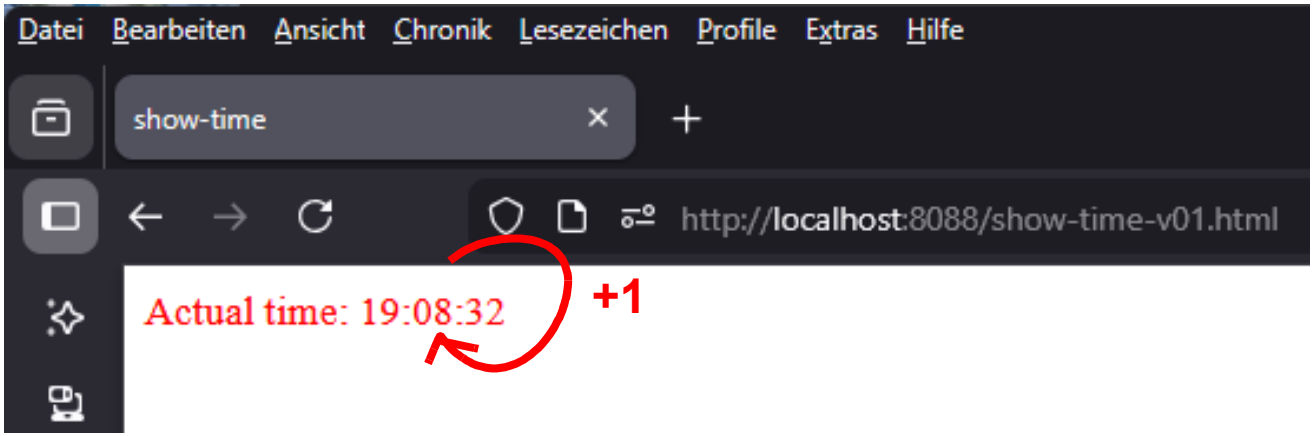
Adding some Dynamics

- Task 1: Display the current time

```
class ActualTime extends HTMLElement {
   connectedCallback() {
      setInterval(()=>this.render(),1000)
   }
   render() {
      this.innerHTML = `<span>
         ${new Date().toLocaleTimeString()}
      </span>`
   }
}
customElements.define('actual-time', ActualTime)
```

```
<!DOCTYPE html>
<html>
   <style>
   span {
      color: red;
   }
    </style>
<head>
   <script type="module"
src="./js/show-time-v01.js"></script>
</head>
<body>
<span>Actual time:</span> <actual-time></actual-time>
</body>
</html>
```

Output



- Method render() is called every second (1000 ms)

http://localhost:8088/examples/05-date-time.html

Karlsruhe Institute of Technology

münch IT SOLUTIONS

UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik

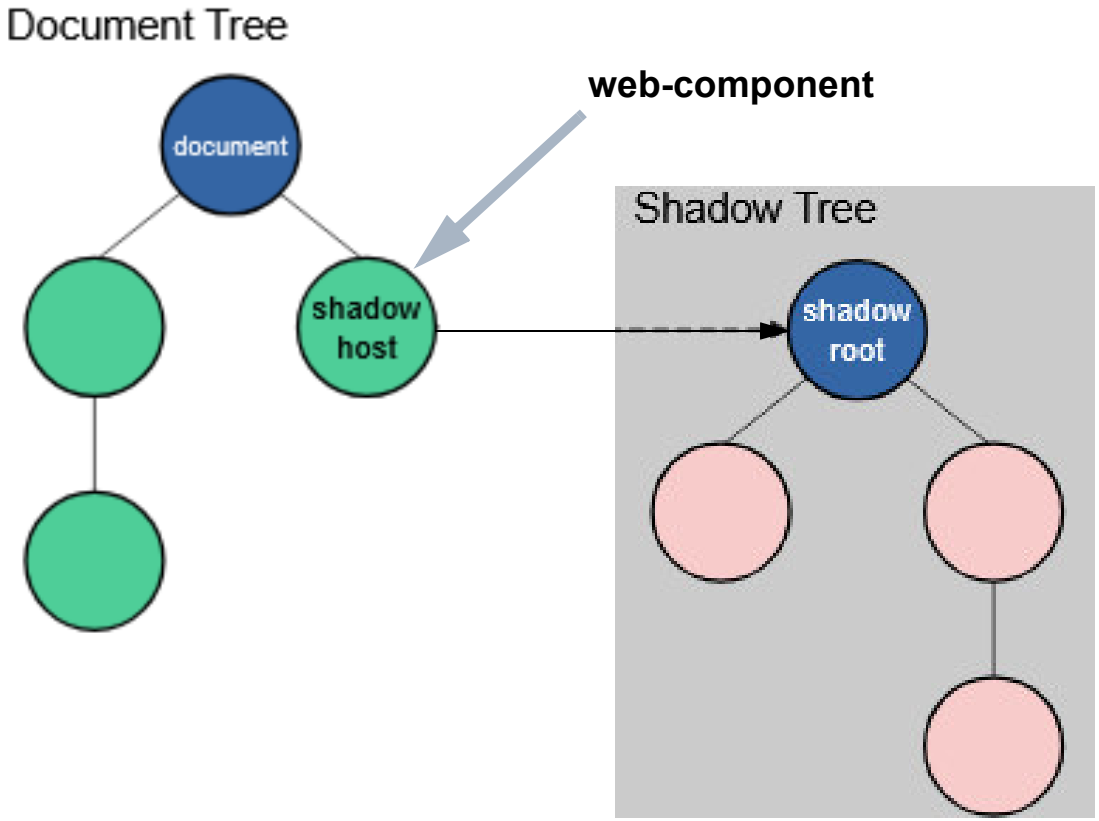
HKA

Shadow DOM

Shadow DOM

- What if I don't want stylesheets (and JavaScript code) to affect my web component nor the web component to affect the rest of the page?
- Solution: Shadow DOM
- Principle:
  - Separate DOM tree that is attached below a ShadowRoot, which acts as a boundary between the two parts.

Shadow DOM


Document Tree
web-component
document
shadow host
Shadow Tree
shadow root

Shadow DOM

```
constructor() {
    super();
    this.attachShadow({mode:'open'});
}

render() {
    console.log(`render()`)
    var now = new Date()
    this.shadowRoot.innerHTML =    `
        <style>
            span {
                color: green;
            }
        </style>
        <span>${now.toLocaleTimeString()}</span>`
}
```

Karlsruhe Institute of Technology

münch IT SOLUTIONS

UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Hochschule Karlsruhe
University of
Applied Sciences
Fakultät für
Informatik und
Wirtschaftsinformatik

HKA

Communication with a Backend

Communication with a Backend Service

- Simple Demo Backend Service at
  `https://www.smiffy.de/dbkda-2026/getTemperature.php`
  or local
  `http://localhost:8088/examples/REST/getTemperature.php`

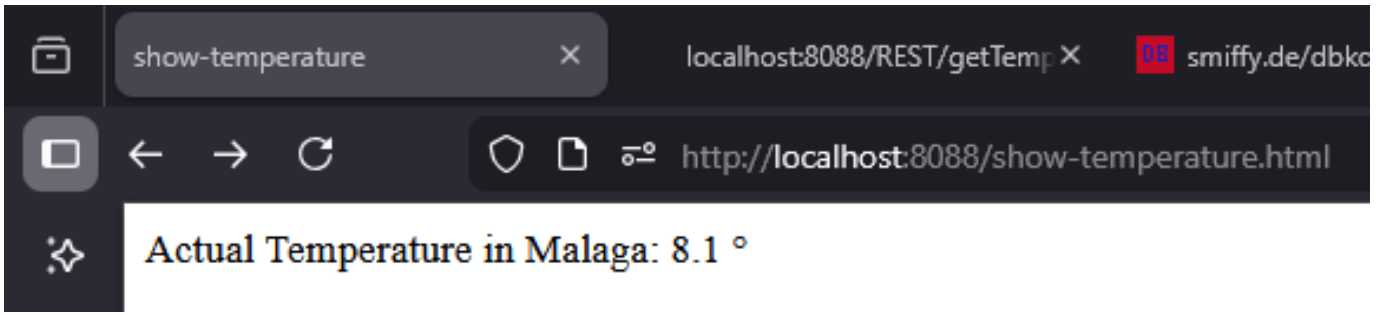
```
$ curl http://www.smiffy.de/webist-2025/getTemperature.php/Malaga
{
    "status": "OK",
    "timestamp": "2025-10-05 09:37:23",
    "location": "Malaga",
    "value": "17.8"
}
```

- Returns a JSON data set with the temperature at a location specified in the URL.
- Example using this backend service ...

Communication with a Backend Service

- Task: Create a web component that is passed the name of a city and displays the current temperature.
-
- usage:

```
<temperature-value city="Malaga"></temperature-value>
```



Communication with a Backend Service

- Implementation:

```
class CityTemperature extends HTMLElement {

    static observedAttributes = ['city', 'temperature'];

    async getTemperature() {
        const endpoint = "http://www.smiffy.de/dbkda-2026/getTemperature.php"
        const url = `${endpoint}/${this.city}`;
        try {
            const response = await fetch(url);
            if (!response.ok) {
                throw new Error(`Response status: ${response.status}`);
            }
            const result = await response.json();
            this.temperature = result.value
            this.render()
        } catch (error) {
            console.error(error.message);
        }
    }
```

when result is not in json-format:
`response.text()`

Communication with a Backend Service

```
    connectedCallback() {
        this.getTemperature();
        const interval_time = 10000 // reload every 10 sec.
        setInterval(() => this.getTemperature(), interval_time)
    }

    render() {
        if (this.temperature)
            this.innerHTML = `Actual Temperature in ${this.city}:
                                ${this.temperature} &deg;`
    }

    attributeChangedCallback(property, oldValue, newValue) {
        // as seen before
    }
}
customElements.define('temperature-value', CityTemperature);
```

SOP and CORS

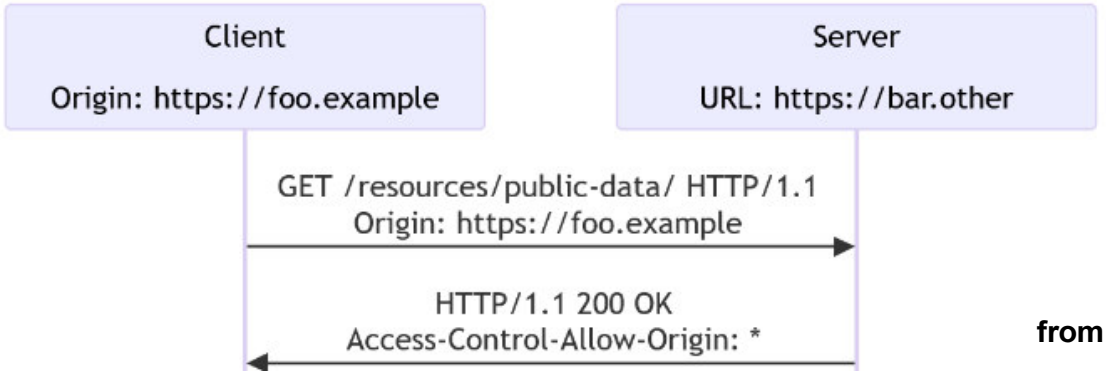
- Same-origin policy (SOP): „The same-origin policy is a critical security mechanism that restricts how a document or script loaded by one origin can interact with a resource from another origin.“ [2]
- For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, `fetch()` and `XMLHttpRequest` follow the same-origin policy. This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers. [3]

- Cross-Origin Resource Sharing (CORS): is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources [3].

[2] https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS

What does that mean for us?

- The response from a server (other than the server the main page was loaded), from were we request a resource must allow us to access this resource, indicated with a „Access-Control-Allow-Origin“-header



from [3]

- The sample resources from **`https://www.smiffy.de/dbkda-2026`**, which is used in the included examples, does this ...

Data Transformation

- Example:

```
<h1>Female Oscar price winners since 2000</h1>
<csv-table>
      Year, Age, Name, Movie
      2000, 25, Hilary Swank, Boys Don't Cry
      2001, 33, Julia Roberts, Erin Brockovich
      2002, 35, Halle Berry, Monster's Ball
      ...
</csv-table>
```

Data Transformation

- Example:

```
<h1>Female Oscar price winners since 2000</h1>
<csv-table>
      Year, Age, Name, Movie
      2000, 25, Hilary Swank, Boys Don't Cry
      2001, 33, Julia Roberts, Erin Brockovich
      2002, 35, Halle Berry, Monster's Ball
      ...
</csv-table>
```

Female Oscar price winners since 2000

Year	Age	Name	Movie
2000	25	Hilary Swank	Boys Don't Cry
2001	33	Julia Roberts	Erin Brockovich
2002	35	Halle Berry	Monster's Ball

Data Transformation

- Example:

```
<h1>Female Oscar price winners since 2000</h1>
<csv-table>
      Year, Age, Name, Movie
      2000, 25, Hilary Swank, Boys Don't Cry
      2001, 33, Julia Roberts, Erin Brockovich
      2002, 35, Halle Berry, Monster's Ball
      ...
</csv-table>
```

- Transform the inner structure of the csv-table into a HTML-Table

```
this.innerHTML = transform(this.innerHTML)
```

Practical Exercise

- Download the exercise from https://www.smiffy.de/dbkda-2026/exercise.pdf

- The exercise is about a web-component that visualizes tabular data in a HTML-table.
- Topics covered:
  - Accessing substructures of web-components
  - Customization via attributes
  - loading data from external sources

Environment for Practical Exercises

- Docker solution (requires Docker desktop)
- PHP's build-in webserver (requires PHP)
- An arbitrary web server on your system (requires apache, nginx, xampp, ...)

Karlsruhe Institute of Technology

münch IT SOLUTIONS

UNIVERSITY OF TECHNOLOGY IN THE EUROPEAN CAPITAL OF CULTURE CHEMNITZ

Hochschule Karlsruhe University of Applied Sciences
Fakultät für Informatik und Wirtschaftsinformatik

HKA

Summary

- WC allow you to extend HTML with your own component
- The components can be equipped with any functionality and appearance.
- W3C standard supported by all current browsers
  - Prevents vendor lock-in
  - Can be used with or without web frameworks such as React, Vue, Angular, etc.
  - Very well suited for building a platform-independent component library (Domain-specific solutions for a wide range of areas)
- Implementation of a series of callback methods in a class derived from HTMLElement
- Shadow DOM for encapsulating your own components (appearance, behavior) and preventing them from affecting the rest of the page

Summary

- The declarative interface also allows non-programmers to create simple web-based applications or better adapt existing applications to their needs.

- WC Interface:
  - HTML-Attributes
  - Events (dispatch, listen)
  - Subelements and text nodes (i.e. `csv-table` from exercise)
  - Communication with a backend

Further Readings/Investigations ...

- Shadow DOM
- Slots and Templates
- lit framework:
  - reduces boilerplate code
  - Declarative templates
  - reactive properties
  - Declarative event listeners (@click)
  - Concept of microbatches to reduce rerendering
- take a look at existing web-components ...

Contact

email: andreas.schmidt@kit.edu

Institute for Automation and Applied Informatics (IAI)

Karlsruhe Institute of Technology

Hermann-von-Helmholtz-Platz 1

76344 Eggenstein-Leopoldshafen

Germany

References

- [1] https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_custom_elements
- [2] https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- [3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS