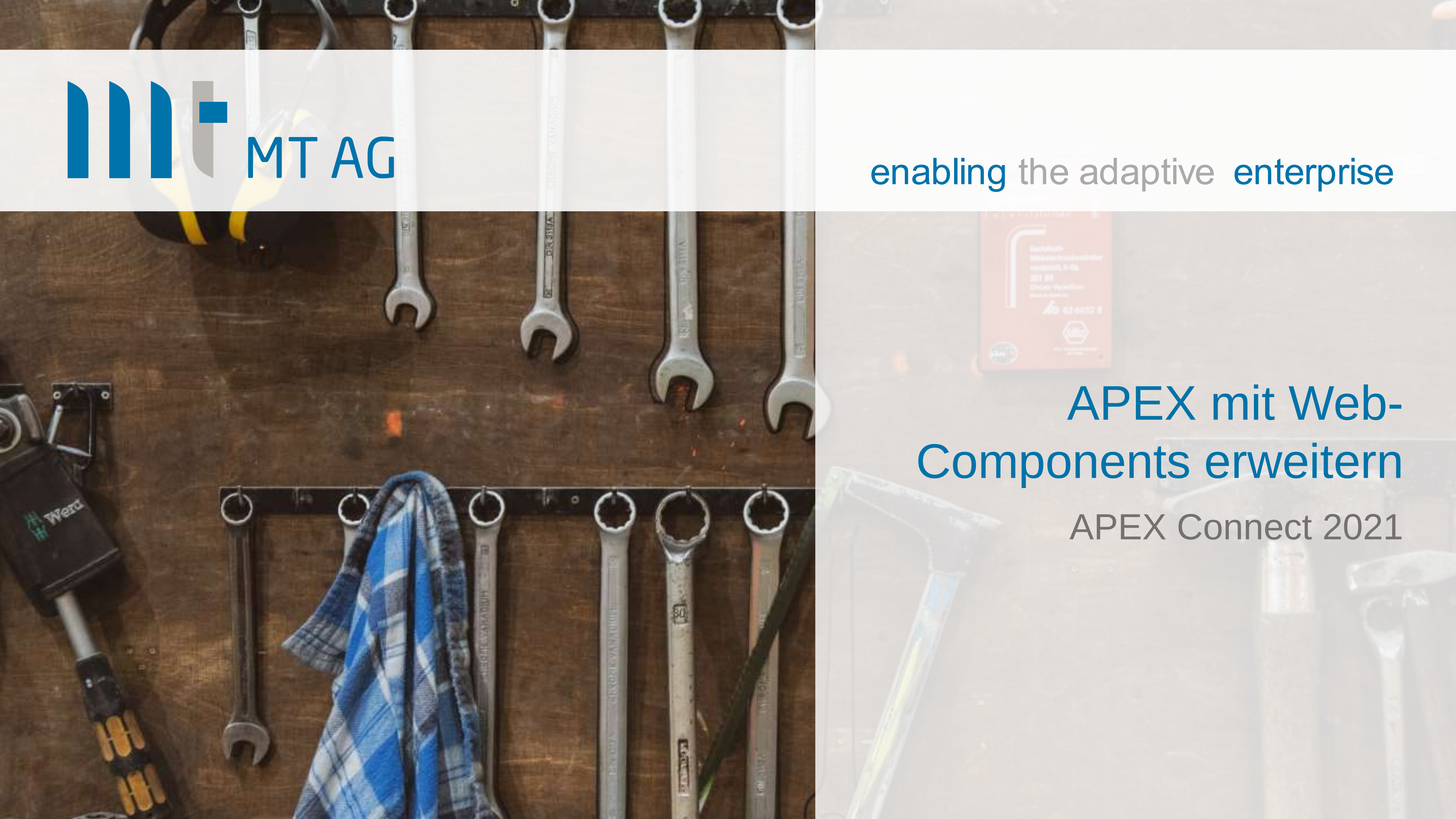
MT AG

enabling the adaptive enterprise



APEX mit Web-Components erweitern

APEX Connect 2021

Über die MT AG

Gründung 1994

Inhabergeführt

ca. 36 Mio. Euro Umsatz in 2020

>100 Kunden

Hauptsitz
Ratingen

Niederlassungen
Frankfurt am Main
Köln
München
Hamburg

280 Beschäftigte
45 APEX Berater

Ihr Partner für den digitalen Wandel

Individuelle IT-Lösungen aus einer Hand

Zertifizierter Partner führender Technologie-hersteller

Herstellerneutral

Branchen-übergreifend

kununu
TOP COMPANY
www.kununu.com
VON MITARBEITERN EMPFOHLEN!

Ausbildungsbetrieb, Partner im dualen Studium

Philipp Hartenfeller

Seit 2016 @ MT AG

APEX / DBs / Web / JavaScript

Aus Düsseldorf

Blog: https://hartenfeller.dev/blog/

@phartenfeller

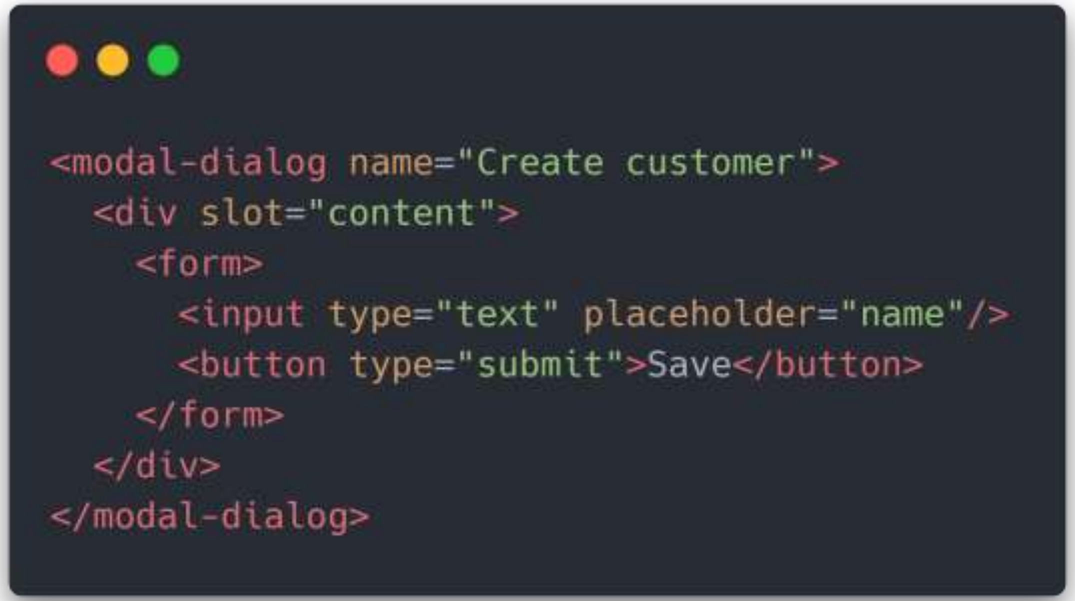
Agenda

MT AG

Was sind Web Components?

Was sind Web Components?

- Set von Web-APIs
- W3C Standard -> Kein Framework nötig
- Ermöglichen eigene HTML-Komponenten zu erstellen
  - Beinhalten Logik und Styles
  - Gekapselt
  - Wiederverwendbar
- Von allen aktuellen Browsern unterstützt
  (kein IE)



```
<modal-dialog name="Create customer">
  <div slot="content">
    <form>
      <input type="text" placeholder="name"/>
      <button type="submit">Save</button>
    </form>
  </div>
</modal-dialog>
```

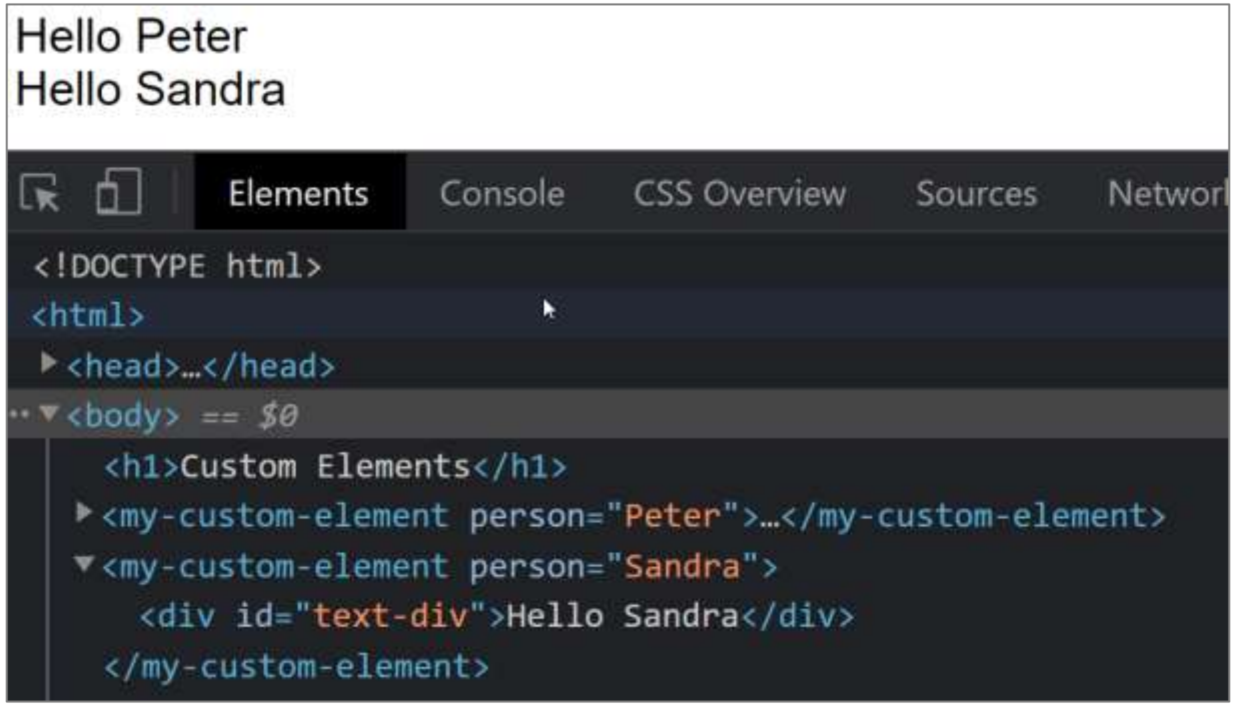
MT AG

Wie funktionieren Web Components?

Wie funktionieren Web Components?

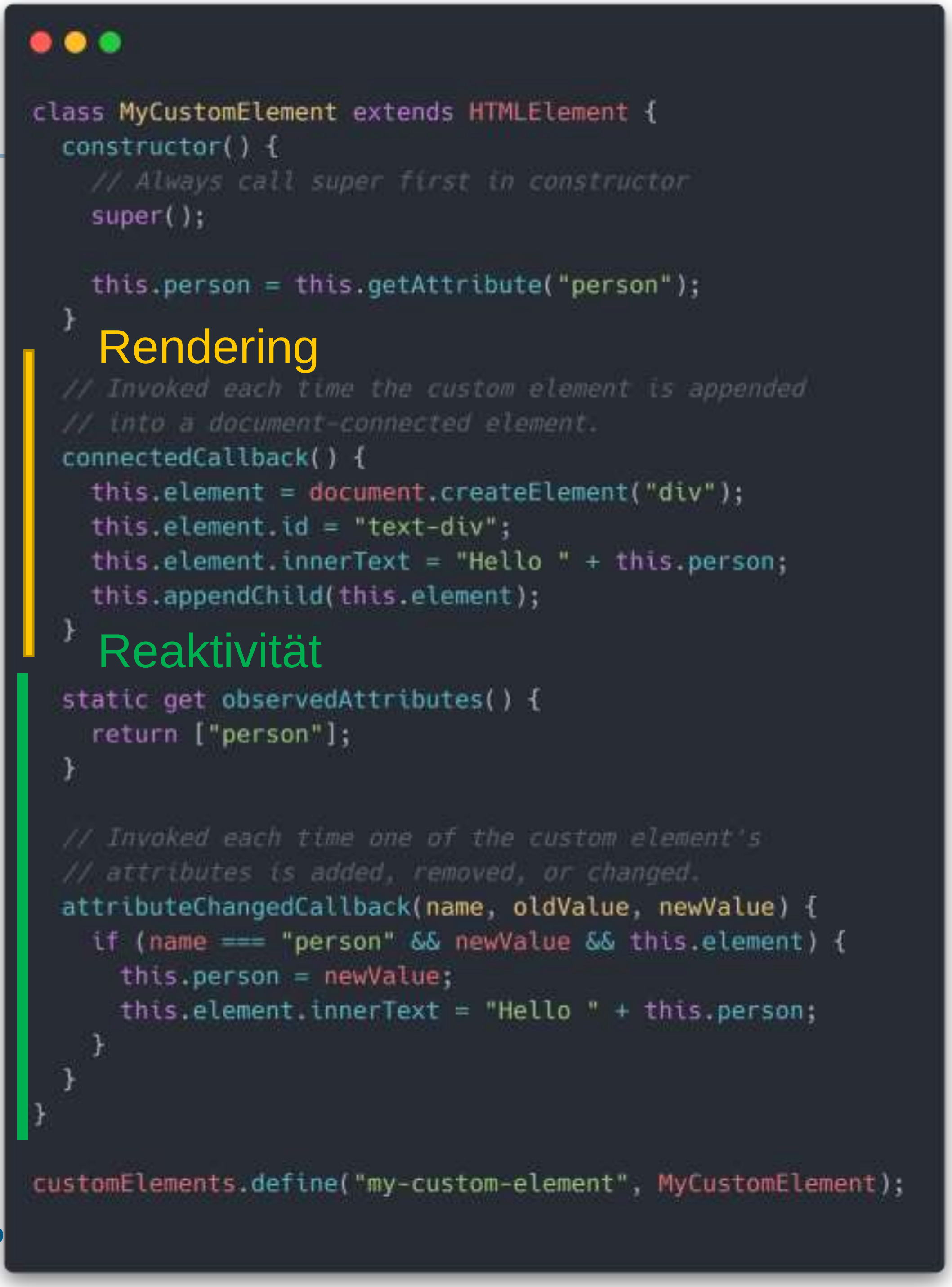
Custom Elements

- Eigene HTML-Elemente
- Definition als Klasse



```
Hello Peter
Hello Sandra
```

```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
··▼<body> == $0
    <h1>Custom Elements</h1>
   ▶<my-custom-element person="Peter">…</my-custom-element>
   ▼<my-custom-element person="Sandra">
      <div id="text-div">Hello Sandra</div>
    </my-custom-element>
```

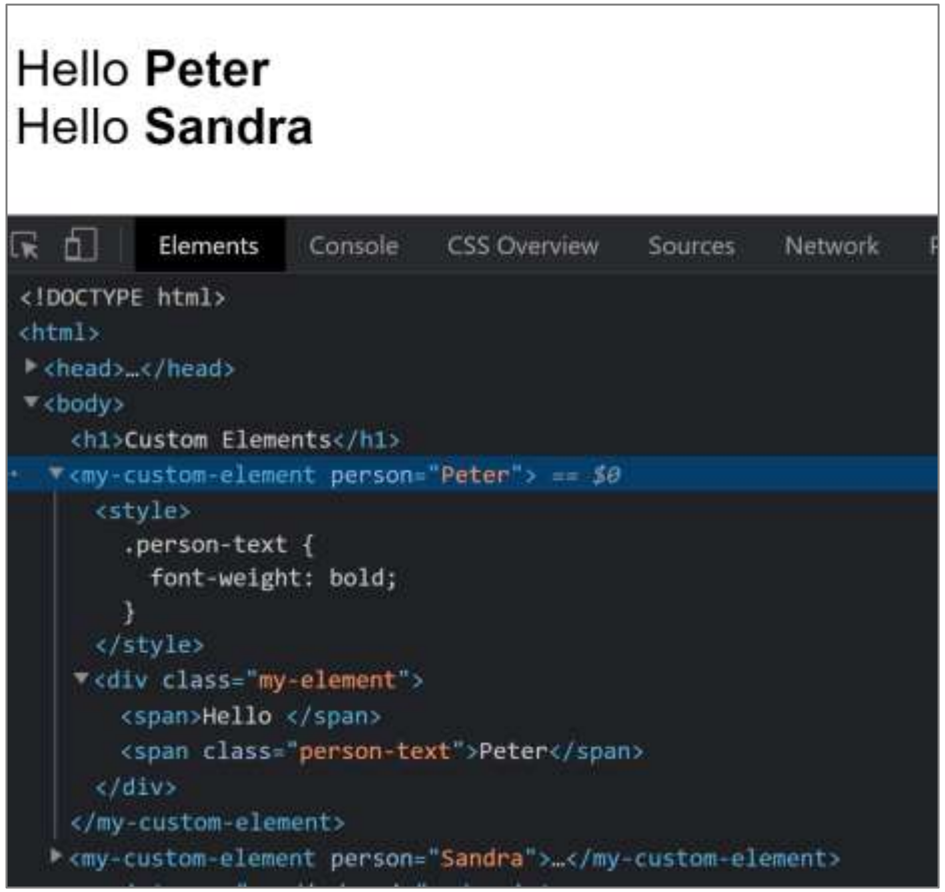


```
class MyCustomElement extends HTMLElement {
  constructor() {
    // Always call super first in constructor
    super();

    this.person = this.getAttribute("person");
  }

  // Invoked each time the custom element is appended
  // into a document-connected element.
  connectedCallback() {
    this.element = document.createElement("div");
    this.element.id = "text-div";
    this.element.innerText = "Hello " + this.person;
    this.appendChild(this.element);
  }

  static get observedAttributes() {
    return ["person"];
  }

  // Invoked each time one of the custom element's
  // attributes is added, removed, or changed.
  attributeChangedCallback(name, oldValue, newValue) {
    if (name === "person" && newValue && this.element) {
      this.person = newValue;
      this.element.innerText = "Hello " + this.person;
    }
  }
}

customElements.define("my-custom-element", MyCustomElement);
```

Wie funktionieren Web Components?

Template

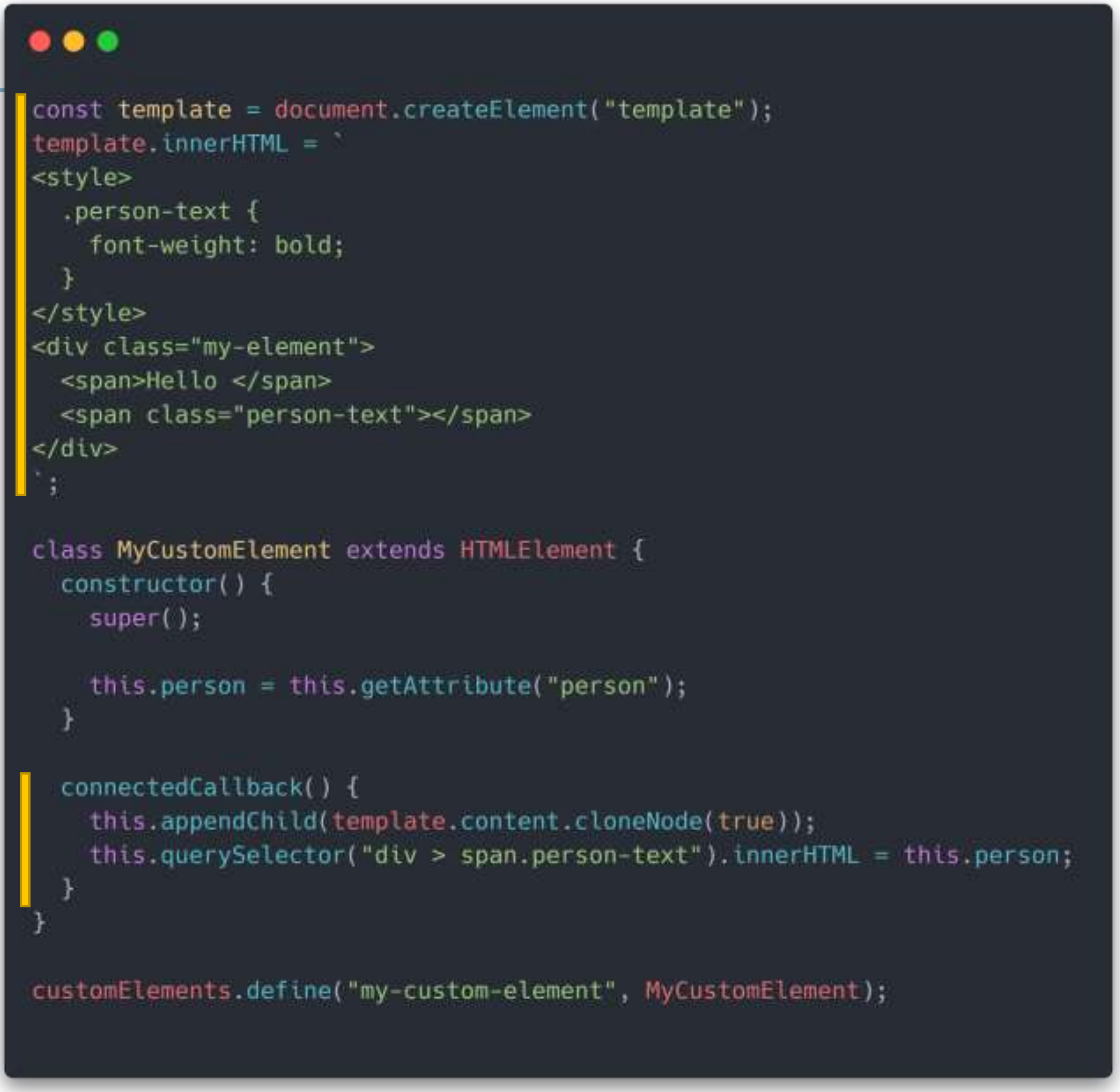
- Gruppierung einer HTML-Struktur



Hello **Peter**
Hello **Sandra**

Elements | Console | CSS Overview | Sources | Network

```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body>
    <h1>Custom Elements</h1>
  ▼<my-custom-element person="Peter"> == $0
      <style>
        .person-text {
          font-weight: bold;
        }
      </style>
    ▼<div class="my-element">
        <span>Hello </span>
        <span class="person-text">Peter</span>
      </div>
    </my-custom-element>
  ▶<my-custom-element person="Sandra">…</my-custom-element>
```



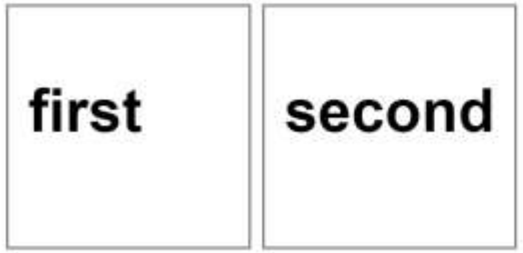
```
const template = document.createElement("template");
template.innerHTML = `
<style>
  .person-text {
    font-weight: bold;
  }
</style>
<div class="my-element">
  <span>Hello </span>
  <span class="person-text"></span>
</div>
`;

class MyCustomElement extends HTMLElement {
  constructor() {
    super();

    this.person = this.getAttribute("person");
  }

  connectedCallback() {
    this.appendChild(template.content.cloneNode(true));
    this.querySelector("div > span.person-text").innerHTML = this.person;
  }
}

customElements.define("my-custom-element", MyCustomElement);
```

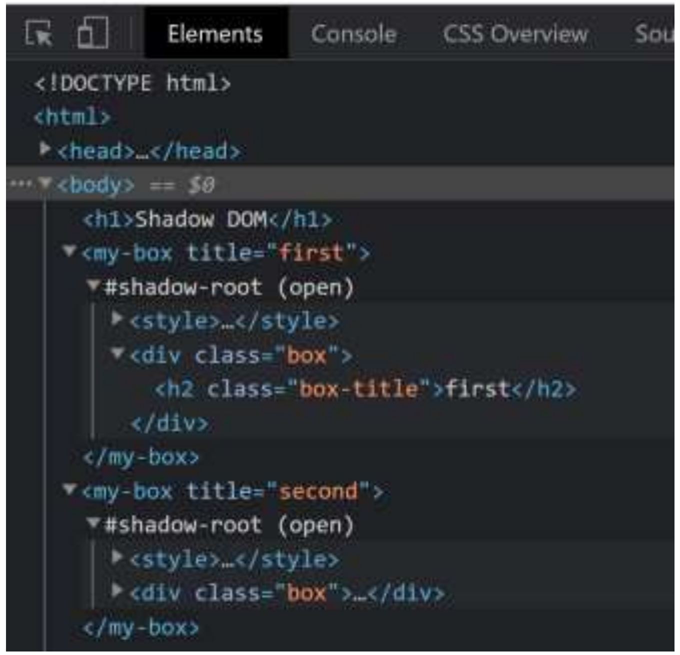
HTML Template (MDN)

Wie funktionieren Web Components?

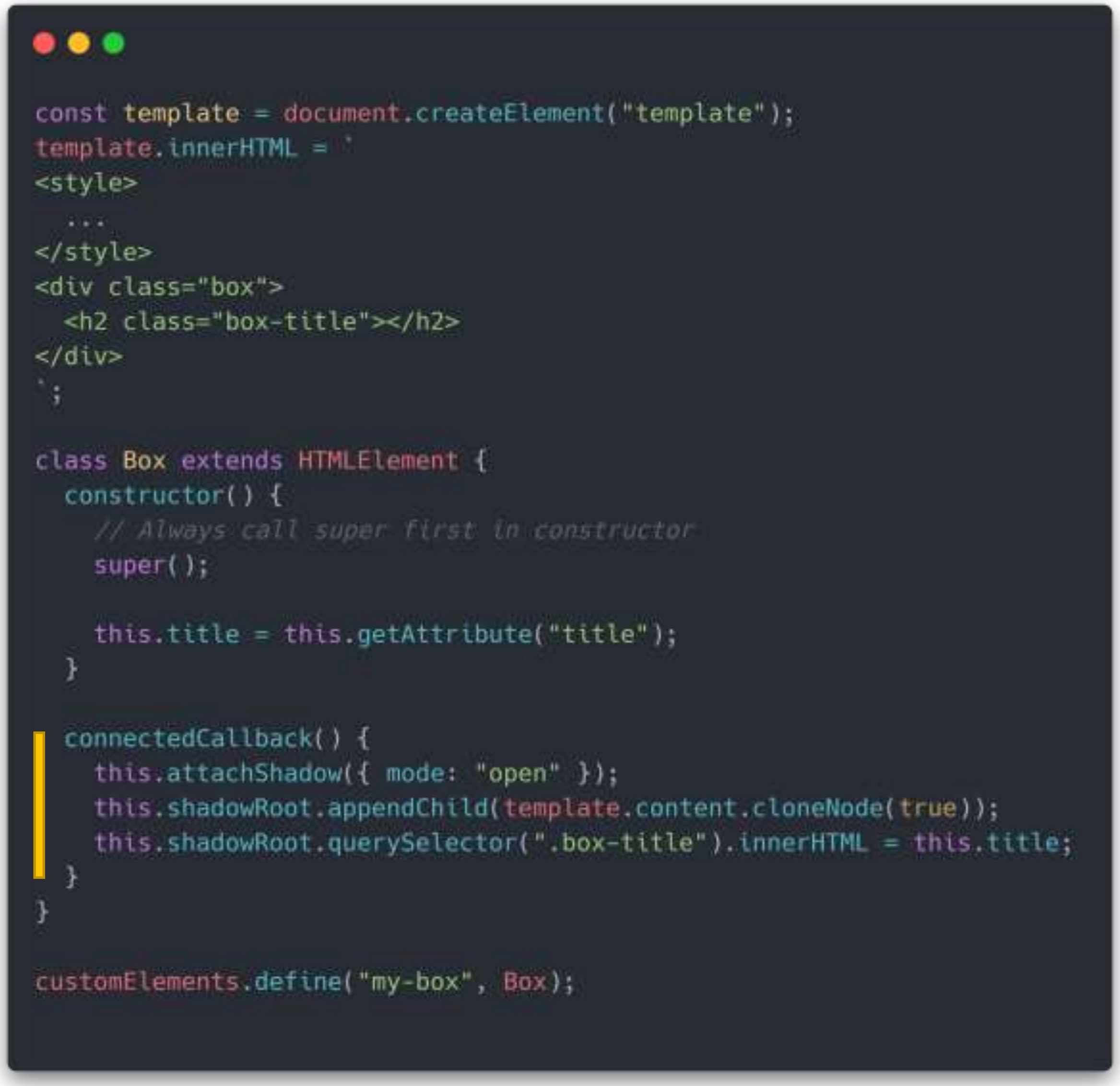
Shadow DOM

- Kapselung / Separierung des Inhalt zum restlichen DOM
- Eigenes DOM -> Shadow DOM
- Mode open / close
  close -> kein JS Zugriff von Außen
- Styles von Außen
  - CSS-Variablen
  - CSS-Shadow-Parts





```
<!DOCTYPE html>
<html>
 ▸<head>…</head>
··· ▾<body> == $0
    <h1>Shadow DOM</h1>
   ▾<my-box title="first">
     ▾#shadow-root (open)
       ▸<style>…</style>
       ▾<div class="box">
          <h2 class="box-title">first</h2>
        </div>
    </my-box>
   ▾<my-box title="second">
     ▾#shadow-root (open)
       ▸<style>…</style>
       ▸<div class="box">…</div>
    </my-box>
```



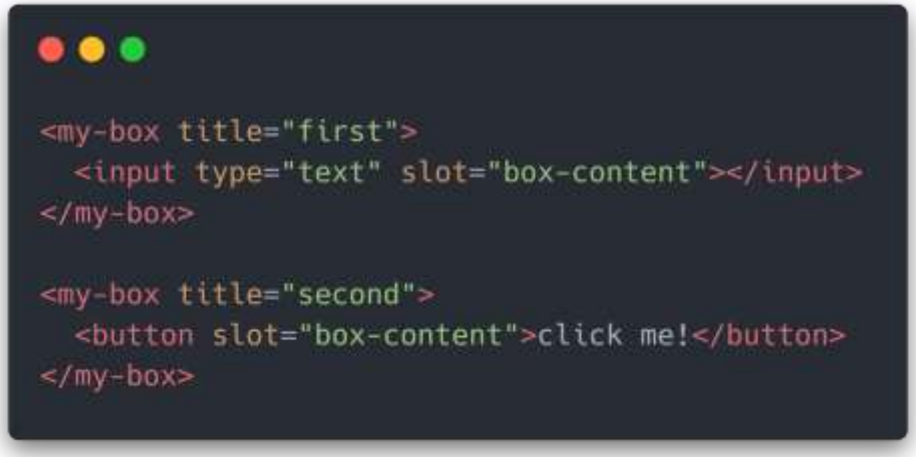
```
const template = document.createElement("template");
template.innerHTML = `
<style>
  ...
</style>
<div class="box">
  <h2 class="box-title"></h2>
</div>
`;

class Box extends HTMLElement {
  constructor() {
    // Always call super first in constructor
    super();

    this.title = this.getAttribute("title");
  }

  connectedCallback() {
    this.attachShadow({ mode: "open" });
    this.shadowRoot.appendChild(template.content.cloneNode(true));
    this.shadowRoot.querySelector(".box-title").innerHTML = this.title;
  }
}

customElements.define("my-box", Box);
```

Shadow DOM (MDN)

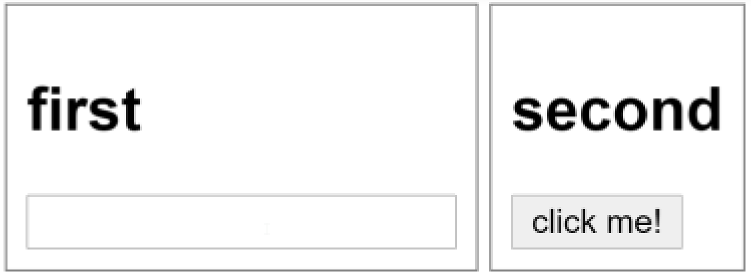
Wie funktionieren Web Components?

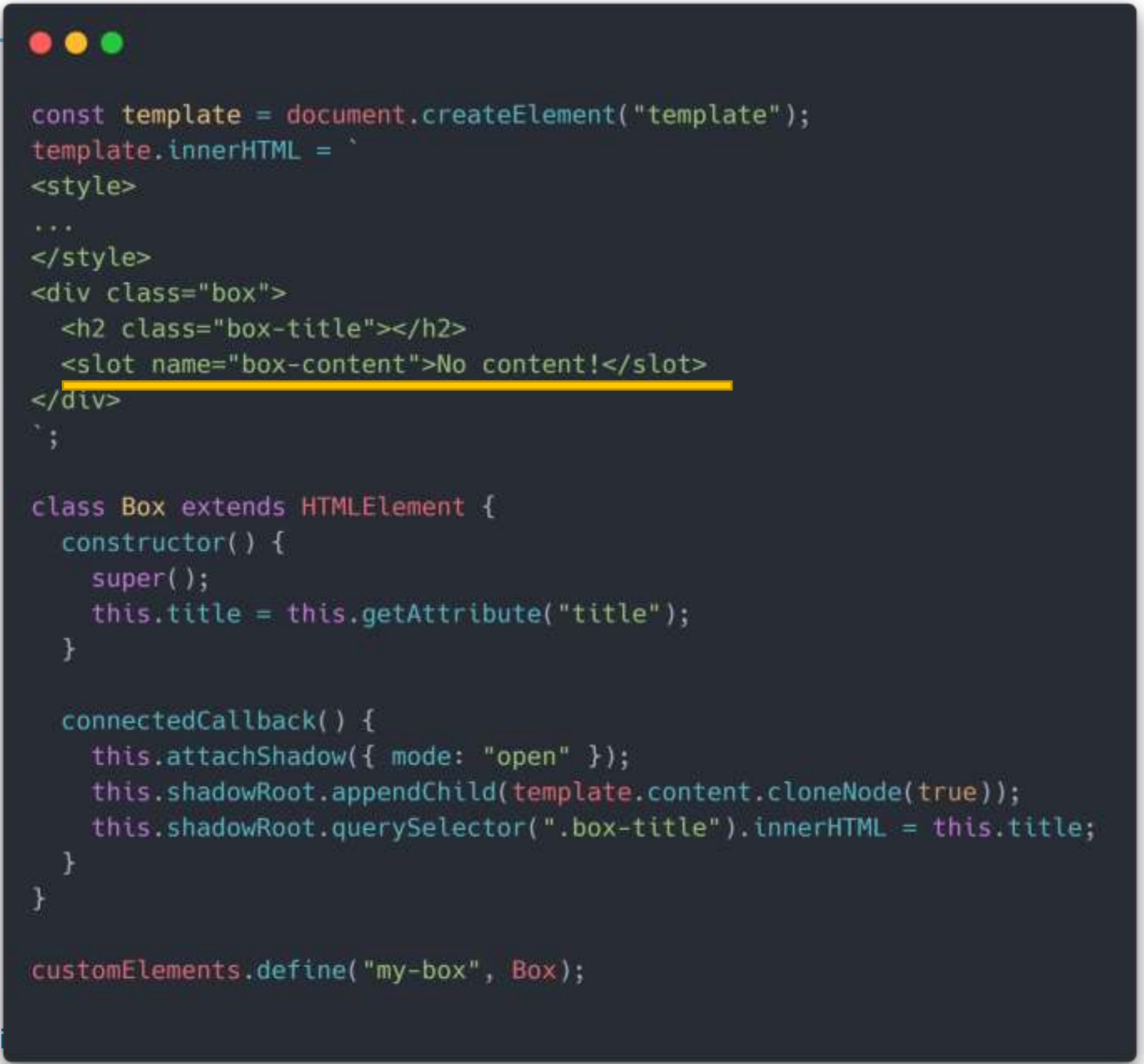
Slots

- Platzhalter im Template
- Von außen können beliebige HTML Element reingegeben werden



```
<my-box title="first">
  <input type="text" slot="box-content"></input>
</my-box>

<my-box title="second">
  <button slot="box-content">click me!</button>
</my-box>
```





```
const template = document.createElement("template");
template.innerHTML = `
<style>
...
</style>
<div class="box">
  <h2 class="box-title"></h2>
  <slot name="box-content">No content!</slot>
</div>
`;

class Box extends HTMLElement {
  constructor() {
    super();
    this.title = this.getAttribute("title");
  }

  connectedCallback() {
    this.attachShadow({ mode: "open" });
    this.shadowRoot.appendChild(template.content.cloneNode(true));
    this.shadowRoot.querySelector(".box-title").innerHTML = this.title;
  }
}

customElements.define("my-box", Box);
```

MT AG

Slot (MDN)

MT AG

Web Components in APEX

Web Components in APEX

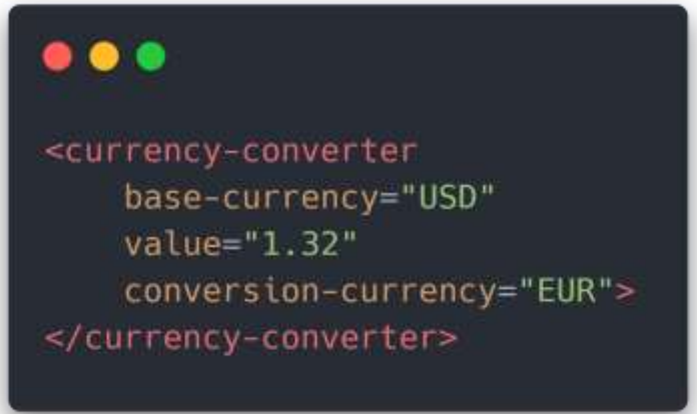
Wo kann man Web Components in APEX verwenden

→ Überall da, wo man HTML einfügen kann

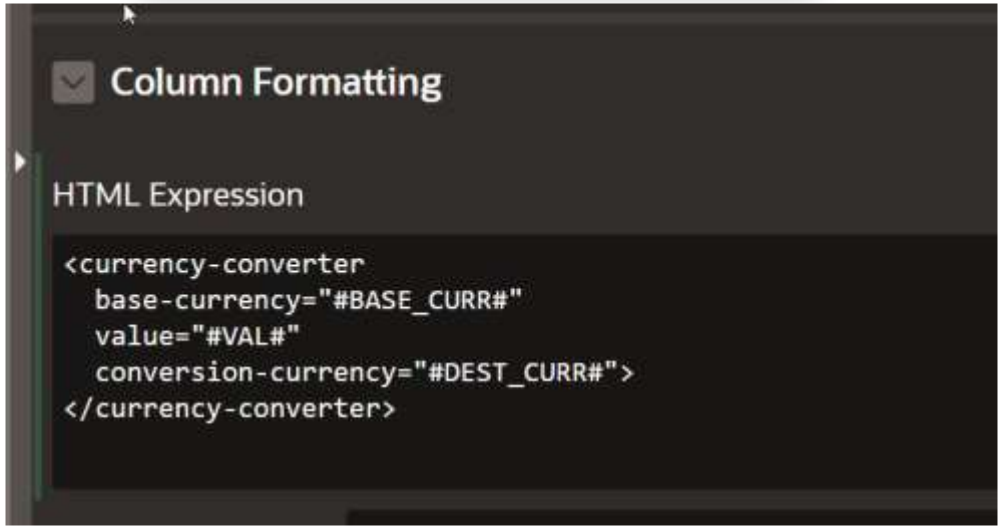
- Regionen
- Report Spalten
- Interactive Grid Detail View
- Templates
- htp.p
- Plug-Ins
- …

Web Components in APEX

Beispiel: Report Spalte



```
<currency-converter
    base-currency="USD"
    value="1.32"
    conversion-currency="EUR">
</currency-converter>
```

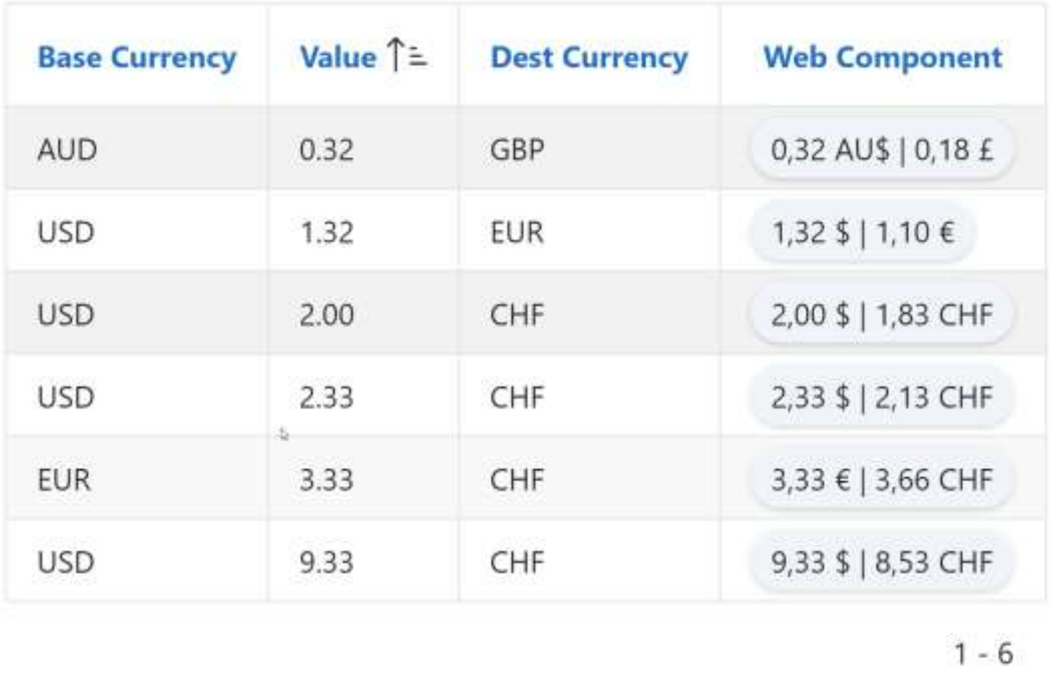


Column Formatting

HTML Expression

```
<currency-converter
  base-currency="#BASE_CURR#"
  value="#VAL#"
  conversion-currency="#DEST_CURR#">
</currency-converter>
```

JS-Datei mit Definition muss importiert sein!



Base Currency	Value	Dest Currency	Web Component
AUD	0.32	GBP	0,32 AU$ | 0,18 £
USD	1.32	EUR	1,32 $ | 1,10 €
USD	2.00	CHF	2,00 $ | 1,83 CHF
USD	2.33	CHF	2,33 $ | 2,13 CHF
EUR	3.33	CHF	3,33 € | 3,66 CHF
USD	9.33	CHF	9,33 $ | 8,53 CHF

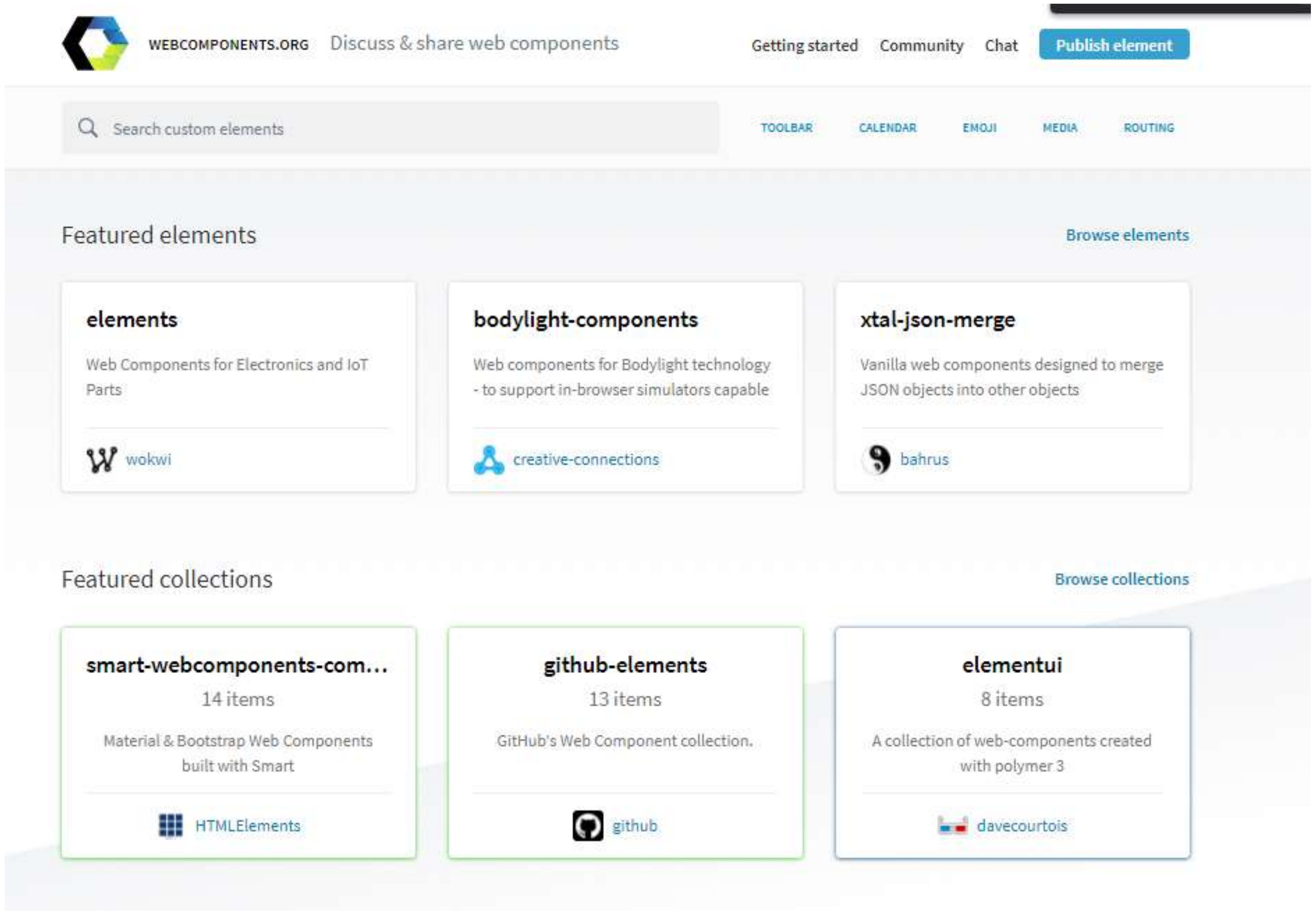
1 - 6

MT AG

Tipps und Tricks

Tipps und Tricks

Bestehende Komponenten verwenden - webcomponents.org



WEBCOMPONENTS.ORG Discuss & share web components

Getting started | Community | Chat | Publish element

Search custom elements

TOOLBAR CALENDAR EMOJI MEDIA ROUTING

Featured elements — Browse elements

elements
Web Components for Electronics and IoT Parts
wokwi

bodylight-components
Web components for Bodylight technology - to support in-browser simulators capable
creative-connections

xtal-json-merge
Vanilla web components designed to merge JSON objects into other objects
bahrus

Featured collections — Browse collections

smart-webcomponents-com...
14 items
Material & Bootstrap Web Components built with Smart
HTMLElements

github-elements
13 items
GitHub's Web Component collection.
github

elementui
8 items
A collection of web-components created with polymer 3
davecourtois

Tipps und Tricks

Lit

- Projekt von Google
- Framework für Web Components
- Macht die Syntax deutlich leichter
- Weit verbreitet

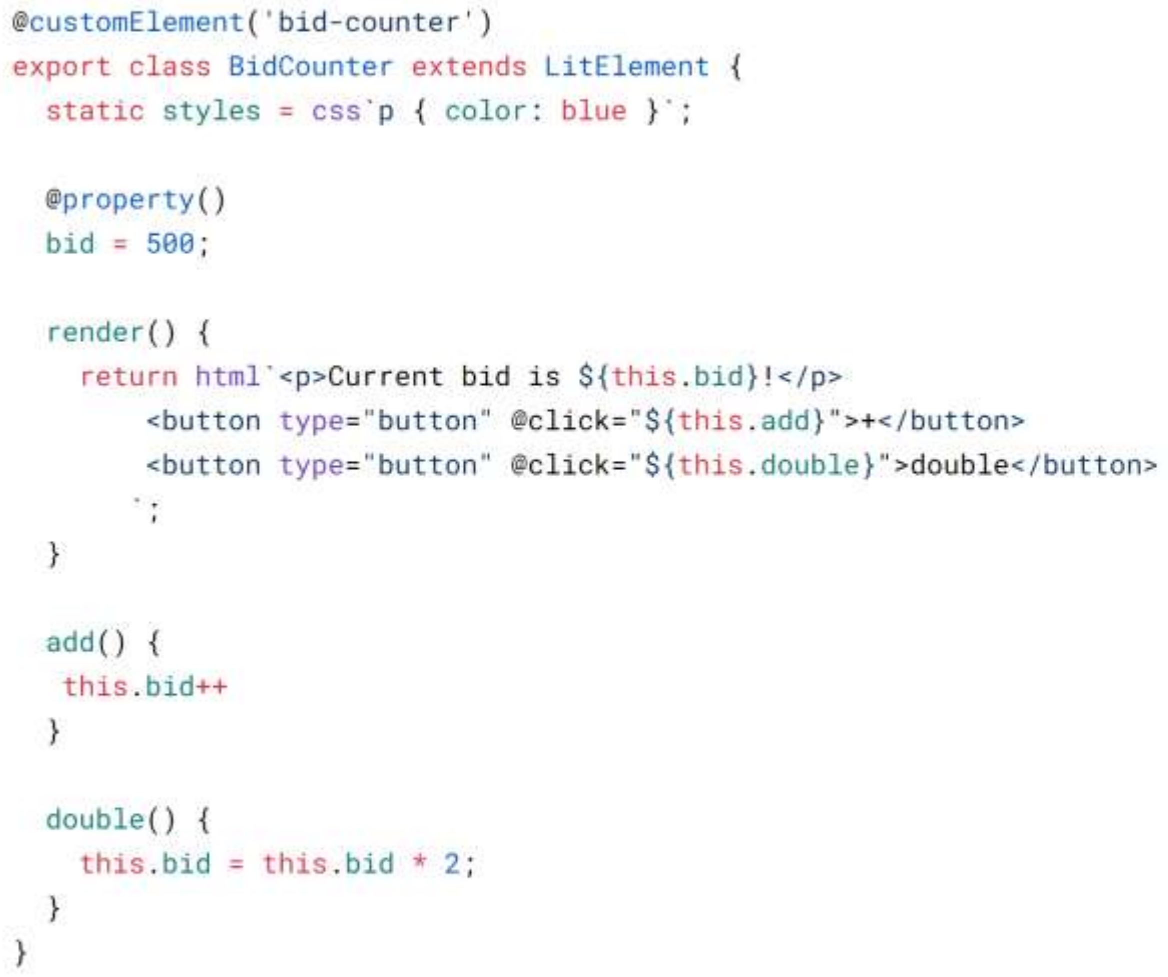
- Z. B. bei Chrome Devtools eingesetzt

[Best Practices (Google)]

Adobe	Spectrum Web Components
Alaska Airlines	Auro Design System
Cisco	Momentum UI
Google	Many projects
Home Assistant	Frontend UI
IBM	Carbon Web Components
ING	Lion Web Components
Microsoft	PWA Starter
Red Hat	One Platform Components
SAP	UI5 Web Components
Shoelace	Components
vaadin}>	Fusion Framework
vmware	Clarity Core
Wired Elements	Wired Elements

Tipps und Tricks

Lit Beispiel



```
@customElement('bid-counter')
export class BidCounter extends LitElement {
  static styles = css`p { color: blue }`;

  @property()
  bid = 500;

  render() {
    return html`<p>Current bid is ${this.bid}!</p>
        <button type="button" @click="${this.add}">+</button>
        <button type="button" @click="${this.double}">double</button>
       `;
  }

  add() {
   this.bid++
  }

  double() {
    this.bid = this.bid * 2;
  }
}
```

Lit Playground

MT AG

demo

Tipps und Tricks

Entwicklungsumgebung einrichten

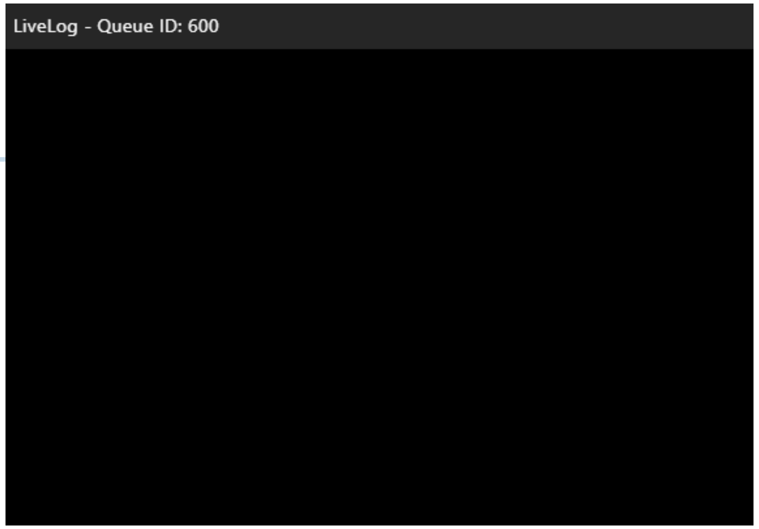
- https://github.com/phartenfeller/webcomponents-template
- https://github.com/phartenfeller/lit-webcomponent-template

- Vorraussetzung: Node.js installiert
- git clone
- npm install
- npm start (entwickeln)
- npm build (Bundles erzeugen)

MT AG

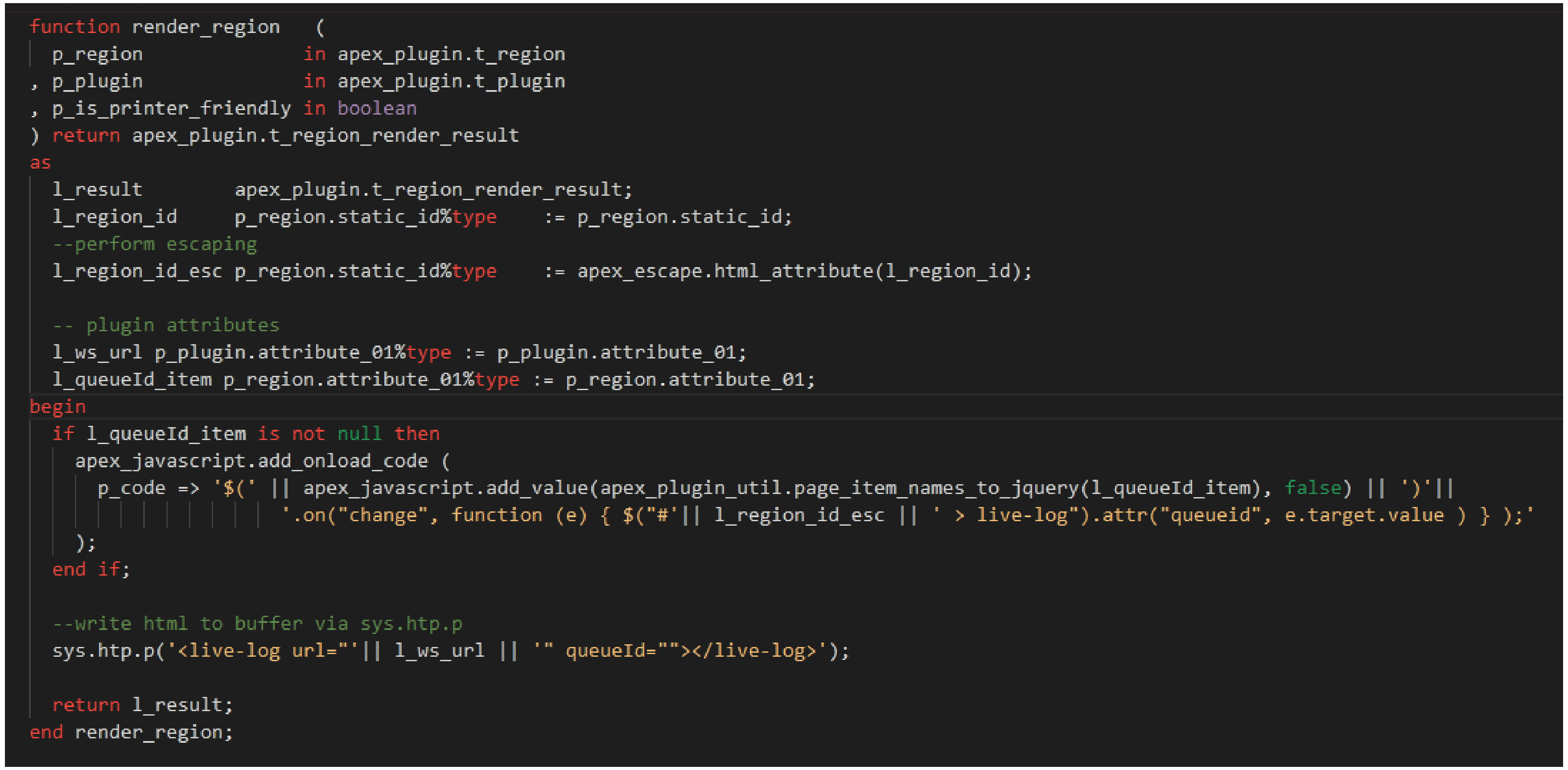
Web Components und APEX Plug-Ins

Web Components und APEX Plug-Ins



- Leichtere Nutzung
- Besserer Überblick
- Wenig Aufwand (je nachdem wie gut man sich mit Plugins auskennt)
- Abfragen einbindbar

→ APEX typische LowCode API



```
function render_region   (
  p_region               in apex_plugin.t_region
, p_plugin               in apex_plugin.t_plugin
, p_is_printer_friendly in boolean
) return apex_plugin.t_region_render_result
as
  l_result        apex_plugin.t_region_render_result;
  l_region_id     p_region.static_id%type    := p_region.static_id;
  --perform escaping
  l_region_id_esc p_region.static_id%type    := apex_escape.html_attribute(l_region_id);

  -- plugin attributes
  l_ws_url p_plugin.attribute_01%type := p_plugin.attribute_01;
  l_queueId_item p_region.attribute_01%type := p_region.attribute_01;
begin
  if l_queueId_item is not null then
    apex_javascript.add_onload_code (
      p_code => '$(' || apex_javascript.add_value(apex_plugin_util.page_item_names_to_jquery(l_queueId_item), false) || ')'||
                '.on("change", function (e) { $("#'|| l_region_id_esc || ' > live-log").attr("queueid", e.target.value ) } );'
    );
  end if;

  --write html to buffer via sys.htp.p
  sys.htp.p('<live-log url="'|| l_ws_url || '" queueId=""></live-log>');

  return l_result;
end render_region;
```

MT AG

demo

MT AG

Fazit

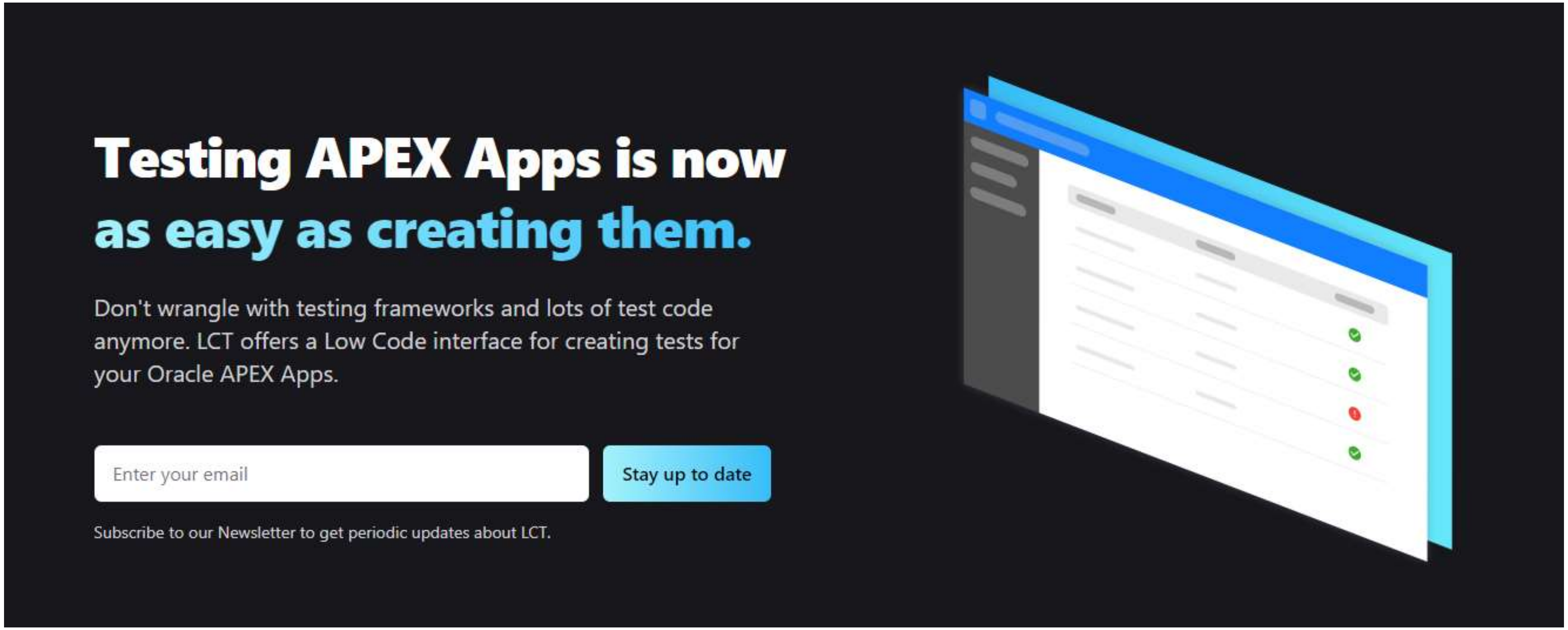
Fazit

- HTML-Standard, der immer relevanter wird
- Nutzbar in allen Webanwendungen (nicht nur APEX)
- Große Auswahl bestehender Komponenten auf webcomponents.org
- Komfortable Entwicklung in einer lokalen Umgebung
- Simple Einbindung in APEX (JS importieren + HTML-Tag)
- Mit Lit elegante Syntax
- Kapselung erhöht Wartbarkeit (sofern die Entwickler sich mit Web Components auskennen)

Empfehlung: Als APEX Plug-In bündeln für noch leichtere Einbindung und um die Übersicht zu wahren

In eigener Sache

APEX Anwendungen testen ohne eigenen Code zu schreiben



Testing APEX Apps is now as easy as creating them.

Don't wrangle with testing frameworks and lots of test code anymore. LCT offers a Low Code interface for creating tests for your Oracle APEX Apps.

Subscribe to our Newsletter to get periodic updates about LCT.

Newsletter - lct.software

Vielen Dank für Ihre Aufmerksamkeit!

Blog: https://hartenfeller.dev/blog/

@phartenfeller