How to use Web Components

with any CSS toolkit

Jacob Rief
@jrief@fosstodon.org

Engineering Kiosk Alps

In this audience, who is a web developer?

...and did you ever encounter the problem of using a component for the web which did not fit your standard styling?

About myself

Full stack Django developer since 2012

Loves Python ...

... and JavaScript

Currently implementing the main Content Management System at the University of Innsbruck

Maintainer of many Open Source projects: https://github.com/jrief/

In this talk, I will present a solution for my latest project: https://django-formset.fly.dev/

Improve the usability of an input field

Example: Enter your phone number

```
<input type="text" name="phone_number" pattern="\+\d{3,16}" required>
```

What instead we want

An input field to enter a phone number with a auto-formatting and a immediate feedback, if we are in the correct country

Portable solutions?

Using a Web Component

Web Components have been added to the HTML standard in 2011

Supported by all browsers since mid 2010ths

```
<custom-phone-number>...</custom-phone-number>
```

```
<input type="text" is="custom-phone-number" name="phone_number"
pattern="\+\d{3,16}" required>
```

Connecting everything together

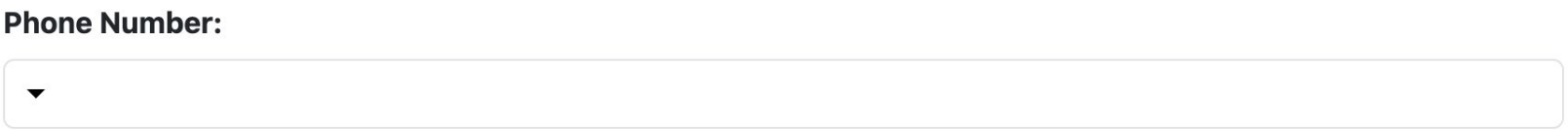
```
class PhoneNumberImplementation  {
    …  400 lines of code …
}

const PN = Symbol('PhoneNumberElement');

export class PhoneNumberElement extends HTMLInputElement {
   constructor() {
      super();
      this[PN] = new PhoneNumberImplementation (this);
   }
}

window.customElements.define('custom-phone-number', PhoneNumberElement, {extends: 'input'});
```

DOM Transformation




```
<label class="form-label">Phone Number:</label>
<input type="text" is="custom-phone-number" name="phone_number"
pattern="^\+\d{3,16}$" required>

<div role="textbox" aria-expanded="false" aria-haspopup="dialog">
    <div class="international-picker"></div>
    <div class="phone-number-edit" contenteditable="true"></div>
</div>
```

Use the new component

```
<html dir="ltr" lang="en">
  <head>
    …
    <link rel="stylesheet" href="https://some-cdn/…/bootstrap.css">
    <link rel="stylesheet" href="/path/to/bootstrap/my-component.css">
    …
    <script type="module" src="/path/to/js/my-component.js">
    …
  </head>
```

How to style the replacement?

Phone Number:

+43 512 50712345

```
<label class="form-label">Phone Number:</label>
<input type="text" is="custom-phone-number" name="phone_number"
pattern="^\+\d{3,16}$" required>

<div role="textbox" aria-expanded="false" aria-haspopup="dialog">
    <div class="international-picker"></div>
    <div class="phone-number-edit" contenteditable="true"></div>
</div>
```

Transferring the CSS classes?


```
<label class="form-label">Phone Number:</label>
<input type="text" is="custom-phone-number" name="phone_number"
pattern="^\+\d{3,16}$" required>

<div role="textbox" aria-expanded="false" aria-haspopup="dialog">
    <div class="international-picker"></div>
    <div class="phone-number-edit" contenteditable="true"></div>
</div>
```

Pilfering existing styles

```
<div role="textbox" aria-expanded="false" aria-haspopup="dialog">
    <div class="international-picker"></div>
    <div class="phone-number-edit" contenteditable="true"></div>
</div>
```

```
[is="custom-phone-number"] {
  --dummy-style: none;
}
[is="custom-phone-number"] + [role="textbox"] {
  display: flex;
  justify-content: flex-start;
}
[is="custom-phone-number"] + [role="textbox"].-focus {
  --dummy-style: none;
}
```

Pilfering existing styles

```
import styles from './PhoneNumber.css';
…
const declaredStyles = document.createElement('style');
declaredStyles.innerText = styles;
document.head.appendChild(declaredStyles);
```

```
const inputElement = // original HTML input element
…
for (let index=0; index<declaredStyles.sheet.cssRules.length; index++) {
   const cssRule = declaredStyles.sheet.cssRules.item(index);
   switch (cssRule.selectorText) {
      case '[is="custom-phone-number"] + [role="textbox"]':
         extraStyles = extractStyles(inputElement, [
            'background-color', 'border', 'border-radius', 'color',
            'outline', 'height', 'line-height', 'padding'
         ]);
         break;
      case '[is="custom-phone-number"] + [role="textbox"].-focus':
         extraStyles = extractStyles(inputElement, […]);
         break;
      …
   }
   extraRule = `${cssRule.selectorText}{${extraStyles}}`;
   declaredStyles.sheet.insertRule(extraRule);
}
```

`extractStyles` *is implemented using* `windows.`**`getComputedStyles`**`(…)`

Caveat 1: Pseudo classes

The `Window.getComputedStyle()` method returns an object containing the values of all CSS properties of an element, after applying active stylesheets and resolving any basic computation those values may contain.
[from mozilla web docs]

This only works for HTML elements and pseudo elements.

It does not work for pseudo styles, such as `:hover`, `:focus`, etc.

Caveat 2: Transitions

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.
[from mozilla web docs]

We can’t emulate transitions while initializing our components, in Bootstrap transitions can last ~50-250 milliseconds.

Solution is to temporarily deactivate them.

Caveat 3: Code duplication

If we insert a style element every time we instantiate a we component, we get a lot of duplicate styles.

Can be prevented, by checking if styling rule for

```
[is="custom-phone-number"] {…}
```

exists.

Demo time

On https://django-formset.fly.dev there are a few demos

Thanks for your attention

Questions?