

Web Components

with
Angular



Hello! I'm **Sherry List**

Azure Developer Technical Lead, Microsoft
Women Techmaker Lead

You can find me at [@SherrryLst](https://twitter.com/SherrryLst)



GDG CPH



Hi! I'm Ana Cidre

Developer Advocate, Ultimate Courses
Women Techmaker Lead

You can find me at [@AnaCidre_](#)





Web Components

A component that is **platform agnostic**

Their main goal is to **encapsulate** the code for the components into a nice, **reusable** package
for maximum **interoperability**.

Custom Elements Everywhere

Making sure frameworks and custom elements can be BFFs 🍺🍺

Check out: <https://custom-elements-everywhere.com/>

Related Issues



Yay! No open issues!

LIBRARY

Angular 7.2.0

SCORE

100%

BASIC TESTS

16/16

ADVANCED TESTS

14/14

Web components consist of three main technologies:

- HTML template
- Custom Elements
- Shadow DOM
- ~~HTML imports~~



The screenshot shows a development environment with the following components:

- PROJECT**: A sidebar with icons for file operations, search, file tree, and settings.
- INFO**: Project details: "Blank starter project for building ES6 apps." with 1 file and 1 dependency.
- FILES**: List of files: index.html (selected), index.js, package.json, style.css.
- DEPENDENCIES**: A section showing dependencies.
- index.html**: The selected file contains the following code:

```
1 <div class="main__container">
2   <h1>Simple button</h1>
3   <p>Here is a simple web component button</p>
4
5   <sa-button
6     text="Click here"
7     style="--background-color: navy; --color: aqua; --font-weight: 600; --padding:
8     12px;">
9     </sa-button>
10  </div>
```
- BROWSER PREVIEW**: Shows the rendered output of index.html. It features a large **Simple button** heading, a paragraph "Here is a simple web component button", and a blue button with the text "CLICK HERE".
- URL**: https://custom-element-button.stackblitz.io
- Console**: An empty console window.
- Page Footer**: Buttons for Twitter, GitHub, and StackBlitz, along with "Support Us" and "Feedback" links.

html

```
<sa-button  
  text="Click here"  
  style="--background-color: navy; --color: aqua; --font-weight: 600; --padding: 12px;">  
</sa-button>
```

js

```
const template = `

<style>
  .btn {
    font-weight: var(--font-weight, 200);
    background-color: var(--background-color, indianred);
    border: 0;
    border-radius: 5px;
    color: var(--color, #fff);
    padding: var(--padding, 10px);
    text-transform: uppercase;
  }
</style>

<button id="text" class="btn"></button>

`;
```

js

```
class SAButtonElement extends HTMLElement {  
  constructor() {  
    super();  
    const template = document.createElement("template");  
  
    // Shadow DOM  
    this.attachShadow({ "mode": "open" });  
    this.shadowRoot.appendChild(template.content.cloneNode(true));  
  }  
}
```

js

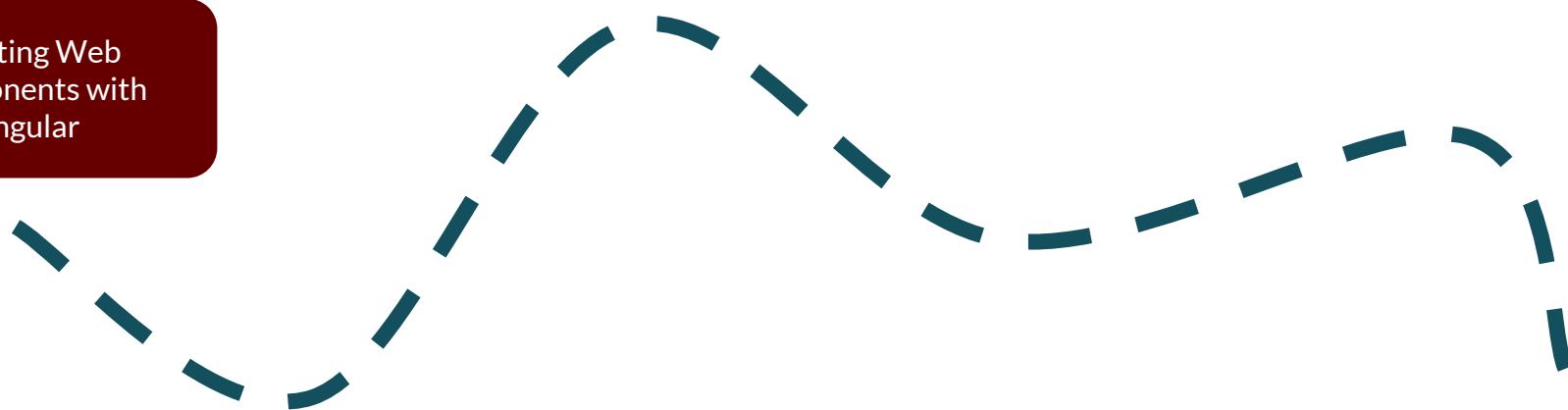
```
class SAButtonElement extends HTMLElement {  
  constructor() {  
    super();  
  
    const template = document.createElement("template");  
  
    // Shadow DOM  
  
    this.attachShadow({ "mode": "open" });  
  
    this.shadowRoot.appendChild(template.content.cloneNode(true));  
  }  
  
  ...  
}  
  
// Define custom element  
customElements.define("sa-button", SAButtonElement);
```

<http://bit.ly/angular-with-web-components-jfokus>

Web Components with Angular: The **What**, the **Why** and the **How**.

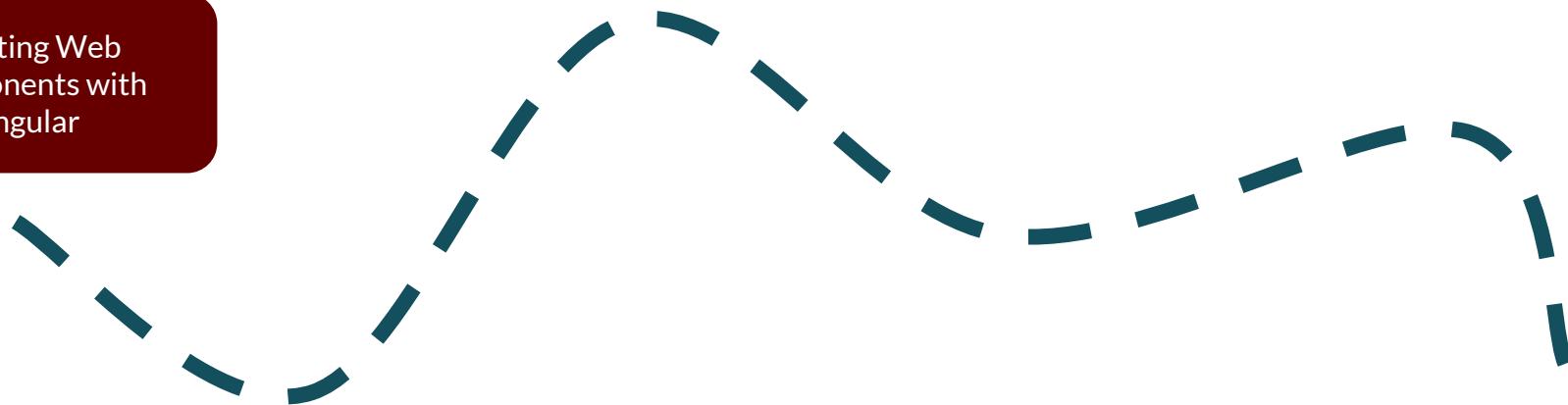


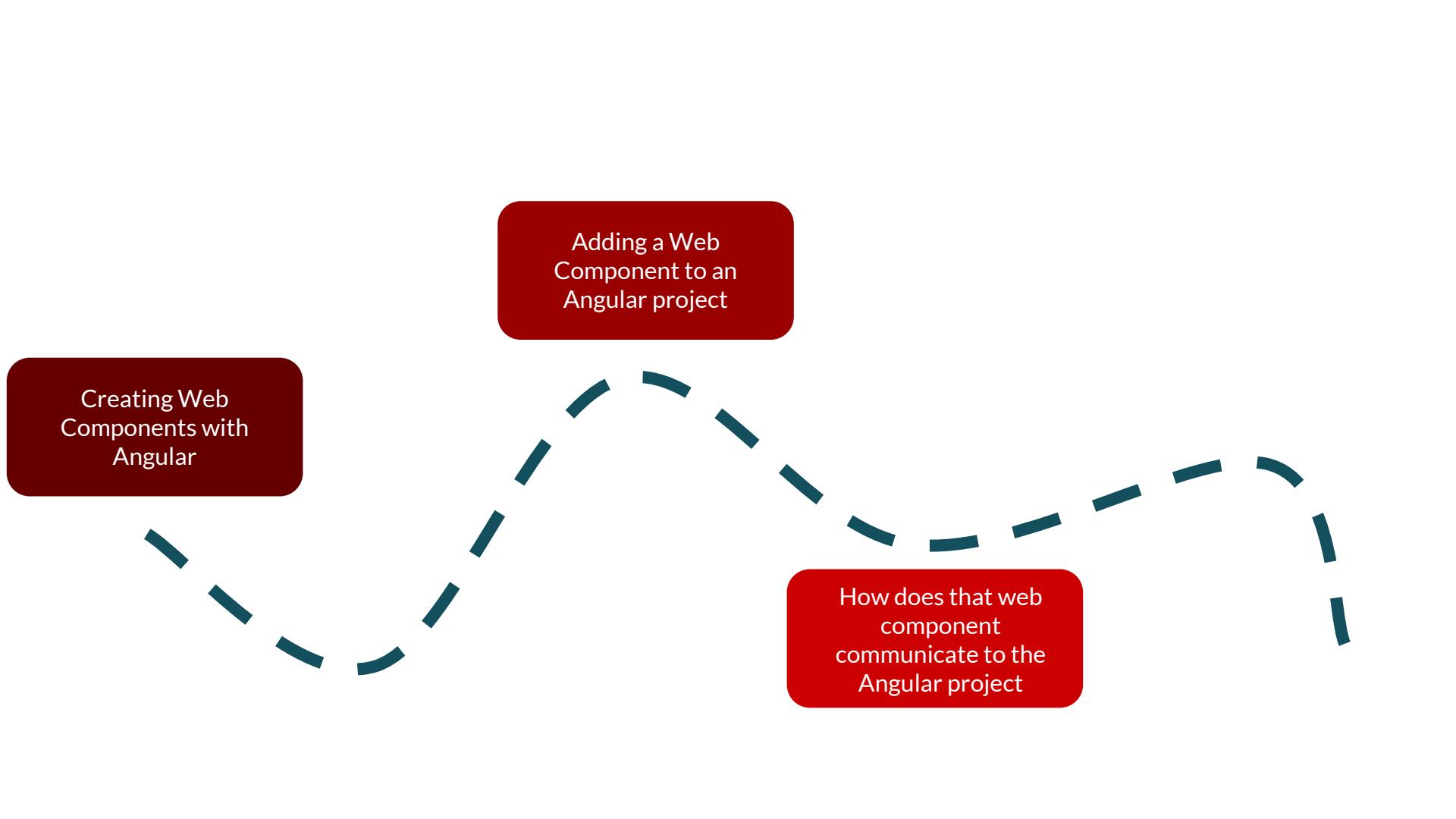
Creating Web
Components with
Angular



Adding a Web
Component to an
Angular project

Creating Web
Components with
Angular





Adding a Web Component to an Angular project

Creating Web Components with Angular

How does that web component communicate to the Angular project

Adding a Web Component to an Angular project

Creating Web Components with Angular

How does that web component communicate to the Angular project

A look into the future





Angular Elements

terminal

```
npm i -g @angular/cli
```

terminal

```
npm i -g @angular/cli
```

```
ng new sa-button --prefix sa --inline-template --style=scss
```

terminal

```
npm i -g @angular/cli
```

```
ng new sa-button --prefix sa --inline-template --style=scss
```

```
cd sa-button
```

terminal

```
npm i -g @angular/cli
```

```
ng new sa-button --prefix sa --inline-template --style=scss
```

```
cd sa-button
```

```
ng add @angular/elements
```

terminal

```
npm i -g @angular/cli
```

```
ng new sa-button --prefix sa --inline-template --style=scss
```

```
cd sa-button
```

```
ng add @angular/elements
```

```
npm install @webcomponents/custom-elements --save
```

polyfills.ts

```
import '@webcomponents/custom-elements/custom-elements.min';
```

tsconfig.json

```
{  
  "compileOnSave": false,  
  "compilerOptions": {  
    ...  
    "target": "es2015",  
    ....  
  }  
}
```

terminal

```
ng generate component button
```

button.component.ts

```
import { Component, OnInit } from '@angular/core';
```

```
@Component({  
  selector: 'sa-button',  
  template: ` ... `,  
  styleUrls: ['./button.component.scss'],  
})
```

```
export class ButtonComponent implements OnInit {  
  constructor() {}  
  ngOnInit() {}  
}
```

button.component.ts

```
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'sa-button',
  template: `<button id="text" class="btn">{{ text }}</button>`,
  styleUrls: ['./button.component.scss'],
})

export class ButtonComponent implements OnInit {
  @Input() text= 'Click me!';

  constructor() {}

  ngOnInit() {}

}
```

Web components consist of three main technologies:

- HTML template
- **Shadow DOM**
- Custom Elements



ViewEncapsulation.ShadowDom

This encapsulation mode uses the Shadow DOM to **scope styles only to this specific component.**

button.component.ts

```
import { Component, OnInit, ViewEncapsulation, Input } from '@angular/core';

@Component({
  selector: 'sa-button',
  template: ` ... `,
  styleUrls: ['./sa-button.scss'],
  encapsulation: ViewEncapsulation.ShadowDom
})
```

Web components consist of three main technologies:

- HTML template
- Shadow DOM
- Custom Elements



app.module.ts

```
import { NgModule, Injector } from '@angular/core';
import { createCustomElement } from '@angular/elements';

@NgModule({
  declarations: [ ButtonComponent ],
  imports: [ BrowserModule ],
  entryComponents: [ ButtonComponent ]
})
```

app.module.ts

```
@NgModule({  
  declarations: [ButtonComponent],  
  imports: [BrowserModule],  
  entryComponents: [ButtonComponent],  
})
```

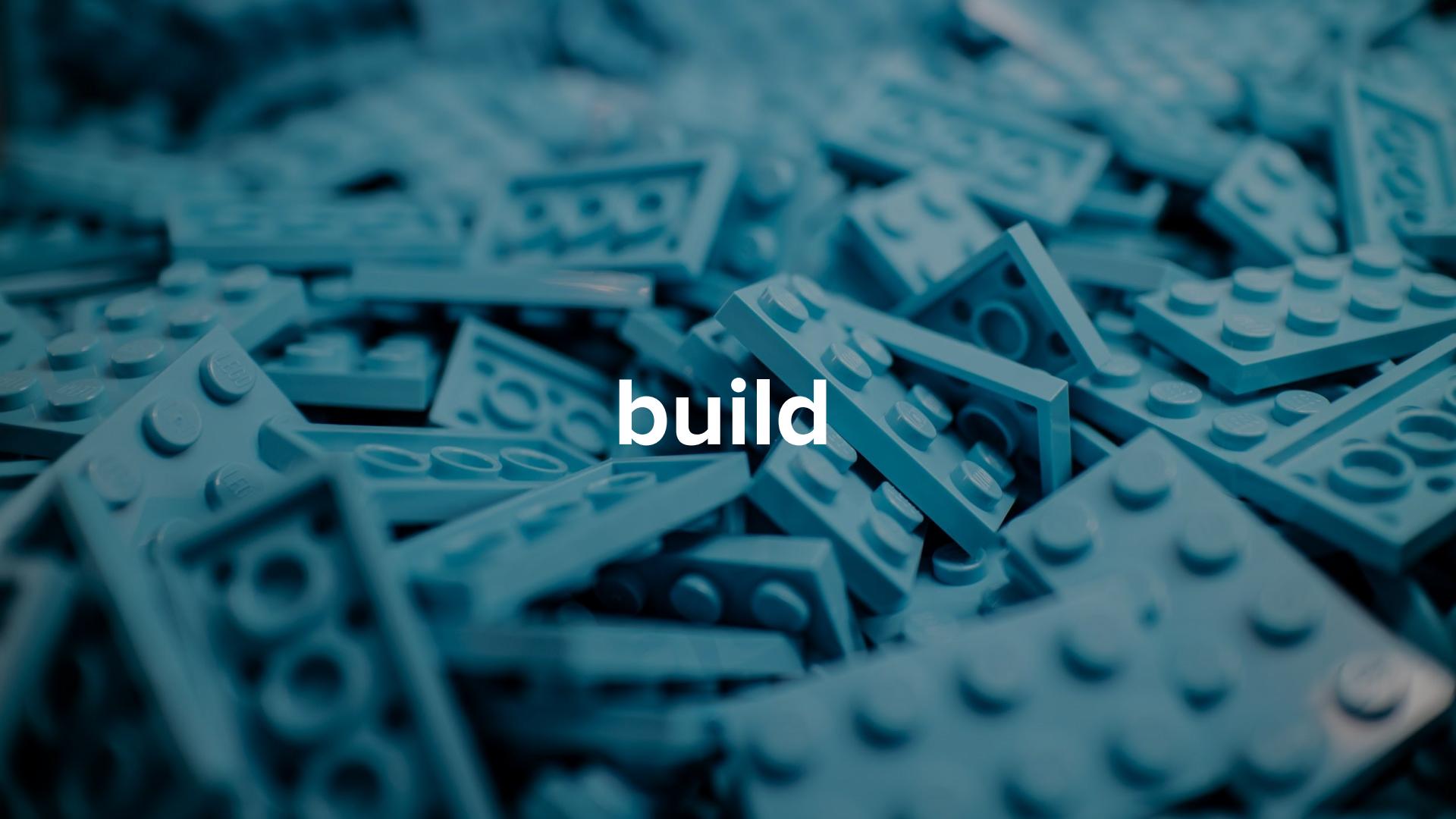
```
export class AppModule {  
  constructor(private injector: Injector) {  
    const saButton = createCustomElement(ButtonComponent, { injector });  
    customElements.define('sa-button', saButton);  
  }  
}
```

app.module.ts

```
@NgModule({  
    declarations: [ButtonComponent],  
    imports: [BrowserModule],  
    entryComponents: [ButtonComponent],  
})  
  
export class AppModule {  
    constructor(private injector: Injector) {  
        const saButton = createCustomElement(ButtonComponent, { injector });  
        customElements.define('sa-button', saButton);  
    }  
    ngDoBootstrap() {}  
}
```



Wait a min



build

It's not that easy!

- Huge bundle size
- No support for one bundle
- -> Eject
- Complicated!



ngx-build-plus



@ManfredSteyer

It's that easy!

- Extends Cli
- No Eject
- Build a single bundle
- No need to add Angular multiple times to your project
- It's simple!



How do we **implement** it?

terminal

```
ng add ngx-build-plus
```

angular.json

```
[...]  
"architect":{  
  "build":{  
    "builder": "ngx-build-plus:build",  
    [...]  
  }  
}  
[...]
```

webpack.extra.js

```
module.exports = {  
  externals: {  
    rxjs: 'rxjs',  
    '@angular/core': 'ng.core',  
    '@angular/common': 'ng.common',  
    '@angular/platform-browser': 'ng.platformBrowser',  
    '@angular/elements': 'ng.elements',  
  },  
};
```

terminal

```
ng build --prod  
  --extraWebpackConfig webpack.extra.js  
  --single-bundle true
```

Now our component is **ready!**



Ivy

The new backwards-compatible Angular renderer:

- Speed improvements
- Bundle size reduction
- Increasing flexibility

How do we **use** our web component?

terminal

```
npm install @webcomponents/custom-elements --save
```

polyfills.ts

```
import '@webcomponents/custom-elements/custom-elements.min';
```

app.module.ts

```
@NgModule({  
  [...],  
  schemas: [  
    CUSTOM_ELEMENTS_SCHEMA  
  ]  
})
```

```
export class AppModule {}
```

app.component.ts

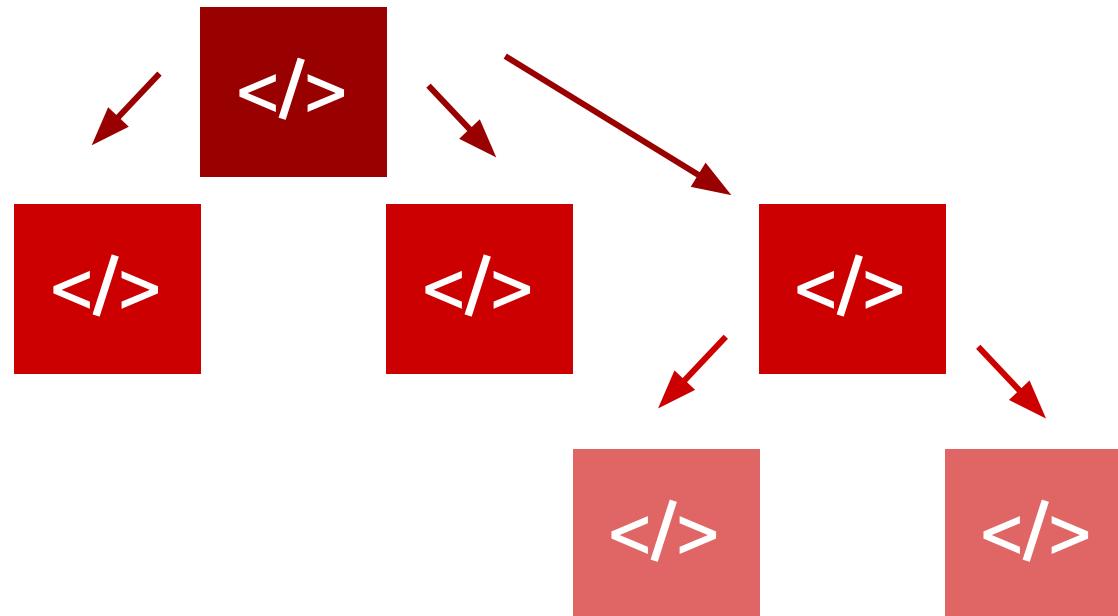
```
import { Component } from '@angular/core';
import * as saButton from '../webcomponents/sa-button/sa-button.js';

@Component({
  selector: 'sa-root',
  template: `
    <sa-button text="hi" ></sa-button>
  `,
  styleUrls: ['./app.component.css']
})

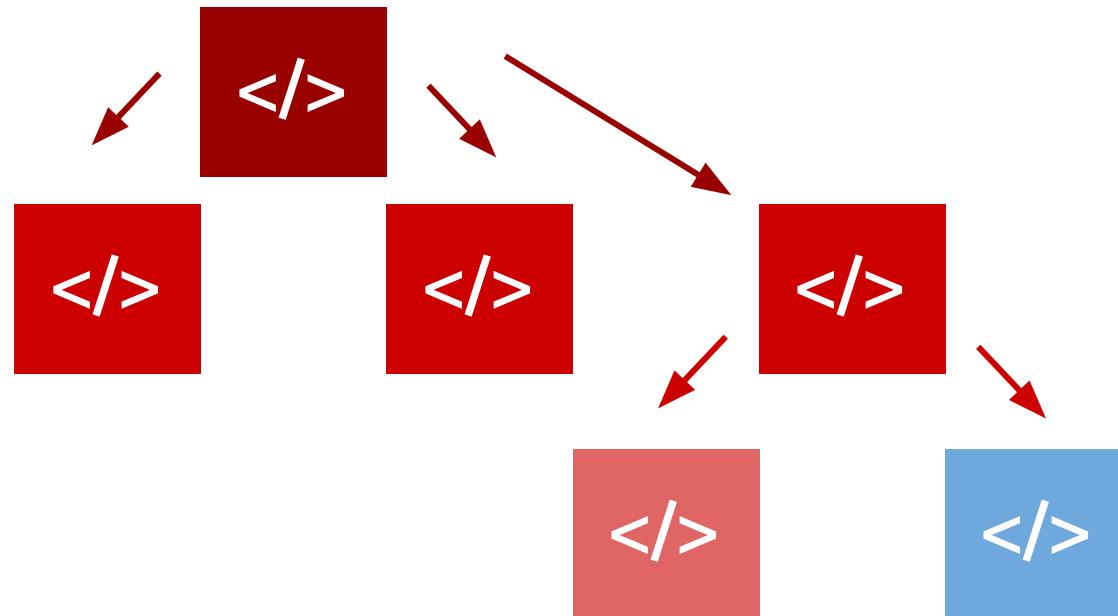
export class AppComponent {...}
```

How do we **communicate** with the WC?

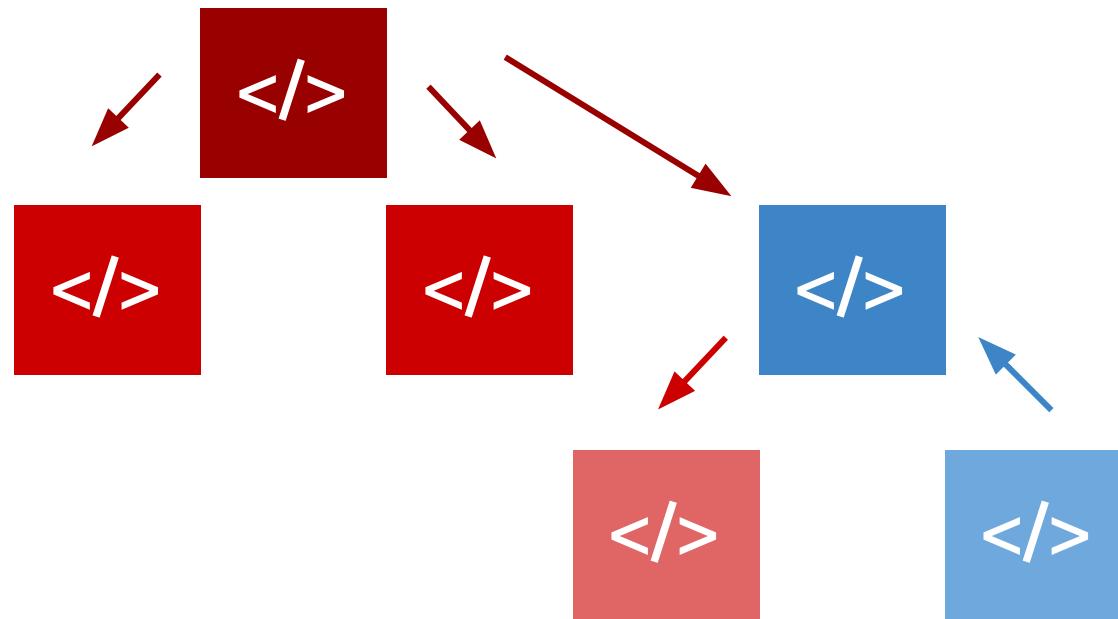
One way data flow



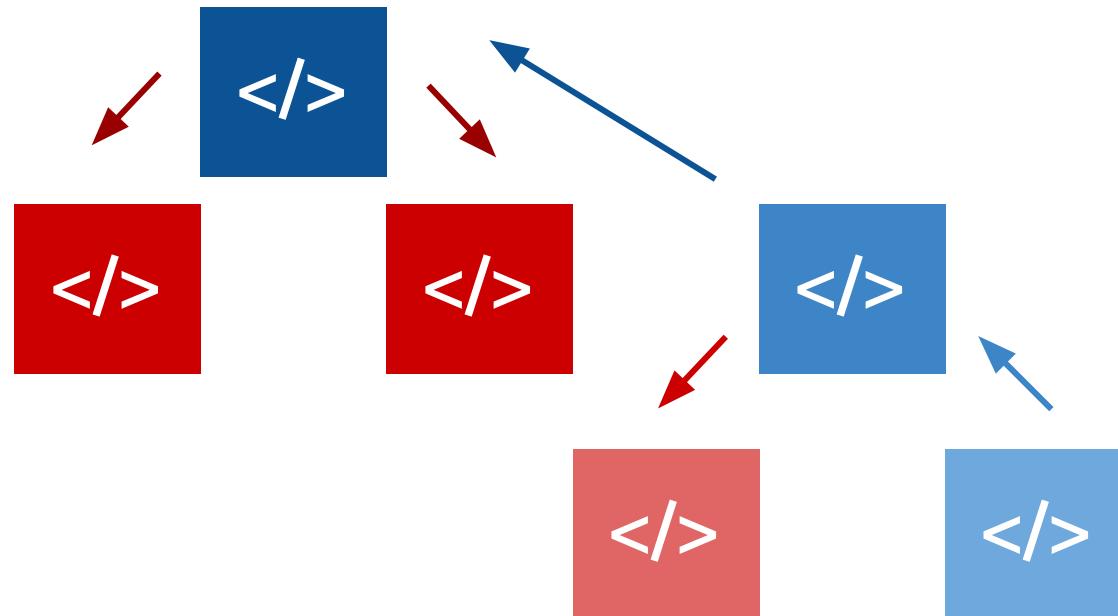
One way data flow



One way data flow



One way data flow





Sherry's Berries!!!



Strawberries

Fresh from our garden

[ADD TO BASKET](#)

Added: 2 Strawberries

[REMOVE](#)



Blackberries

Fresh from garden

[ADD TO BASKET](#)

Added: 0 Blackberries

[REMOVE](#)



Blueberries

Fresh from Sherry's garden!

[ADD TO BASKET](#)

Added: 0 Blueberries

[REMOVE](#)



Elements Console Sources > ▲ 1 : X

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    <!--bindings={-->
      "ng-reflect-ng-for-of": "[object Object],[object Object]"
    }-->
    <sa-root _ngcontent-c0 ng-version="7.0.4">
      <sa-dashboard _ngcontent-c0 _ngghost-c1>
        <sa-header _ngcontent-c1 _ngghost-c2>...</sa-header>
        <div _ngcontent-c1 class="berries">
          <!--bindings={-->
            "ng-reflect-ng-for-of": "[object Object],[object Object]"
          }-->
          <sa-card _ngcontent-c1 _ngghost-c3 ng-reflect-berry="[object Object]">
            
            <div _ngcontent-c3 class="shadow__info" id="shadow-info">...</div>
            <sa-counter _ngcontent-c3 _ngghost-c4 ng-reflect-berry="[object Object]">
              <div _ngcontent-c4 class="counter">
                ...<br/>
                <sa-button _ngcontent-c4 style="--padding:5px; --font-size: 14px;" text="Add to basket" ng-version="7.0.4"> == $0
                  <#shadow-root (open)>
                    <style>
                      .btn{font-weight:var(--font-weight,200);font-size:var(--font-size,24px);background-color:var(--background-color,#cd5c5c);border:0;border-radius:5px;color:fff;padding:var(--padding,10px);text-transform:uppercase}
                    </style>
                    <button class="btn" id="text">Add to basket</button>
                  </sa-button>
                  <p _ngcontent-c4>...</p>
                  <sa-button _ngcontent-c4 style="--padding:5px; --font-size: 14px;" text="Remove" ng-version="7.0.4"> == $0
                    <#shadow-root (open)>
                      <style>
                        .rmv{font-weight:var(--font-weight,200);font-size:var(--font-size,24px);background-color:var(--background-color,white);border:0;border-radius:5px;color:var(--color,cd5c5c);padding:var(--padding,10px);text-transform:uppercase}
                      </style>
                      <button class="rmv" id="text">Remove</button>
                    </sa-button>
                  </sa-counter>
                </div>
              </div>
            </sa-card>
          </div>
        </sa-dashboard>
      </sa-root>
    
```

Styles Event Listeners DOM Breakpoints Properties >

Filter :hover .cls +

element.style {

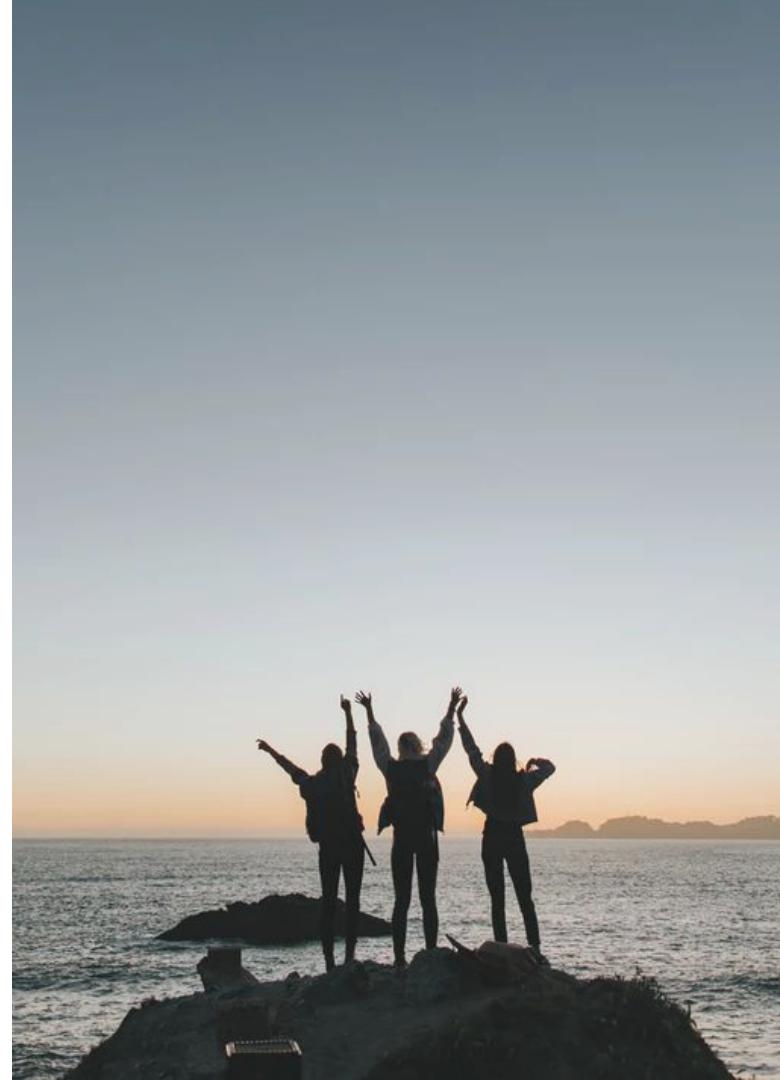
--padding: 5px;

margin -

border -

We are almost done!

- ✓ Create a web component with Angular
- ✓ Add a web component to an existing Angular project
- ✓ Use the web component in an Angular project



A photograph of a yellow van with a white roof rack driving away from the viewer on a paved road. The road is flanked by dry, scrubby vegetation. In the background, massive, layered red rock formations rise against a bright blue sky with wispy clouds. The text "Do Web Components rock?" is overlaid in the upper right quadrant.

**Do Web Components
rock?**

Web Components DO Rock!

- Maximum interoperability
- Support from Frameworks (Tooling)
- Many success stories
- Browsers are ready



Browser support



CHROME



OPERA



SAFARI



FIREFOX



EDGE



HTML TEMPLATES



STABLE



STABLE



STABLE



STABLE



STABLE



CUSTOM ELEMENTS



STABLE



STABLE



STABLE



STABLE

POLYFILL
DEVELOPING



SHADOW DOM



STABLE



STABLE



STABLE



STABLE

POLYFILL
DEVELOPING



ES MODULES



STABLE



STABLE



STABLE



STABLE



STABLE



What's next?

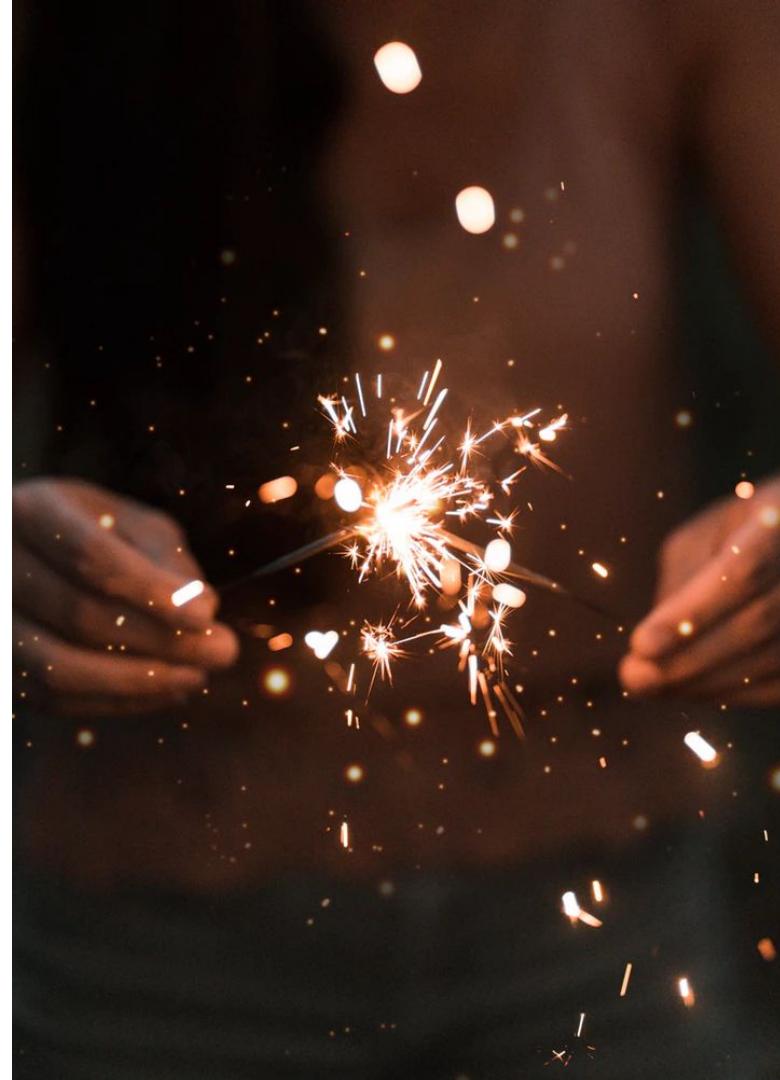
The background of the slide features a large, bright campfire on the left side, with many glowing embers and sparks rising into the dark sky. The rest of the slide is a solid black.

lit-html

Check out: <https://youtu.be/Io6JjgckHbg>

What is lit-html?

- <template> in JS with template literals
- Extremely fast (Partial update)
- Customizable and extensible
(Directives)



Simple button

Here is a simple web component button

N/A

```
<!doctype html>
<html class="">
  <head>...</head>
  <body>
    <div class="main__container">
      <h1>Simple button</h1>
      <p>Here is a simple web component button</p>
      <sa-button text style="--background-color: navy; --color: aqua; --font-weight: 600; --padding: 12px;" onclick="saFunction()">
        <#shadow-root (open)>
          <style>...</style>
          ...
          <button class="btn" id="n/a">n/a</button> == $0
        </sa-button>
      </div>
    </body>
  </html>
```

html body div.main__container sa-button #shadow-root button#n/a.btn

Styles Event Listeners DOM Breakpoints Properties Accessibility \$scope ADT Properties

Filter

:hov .cls +

element.style {

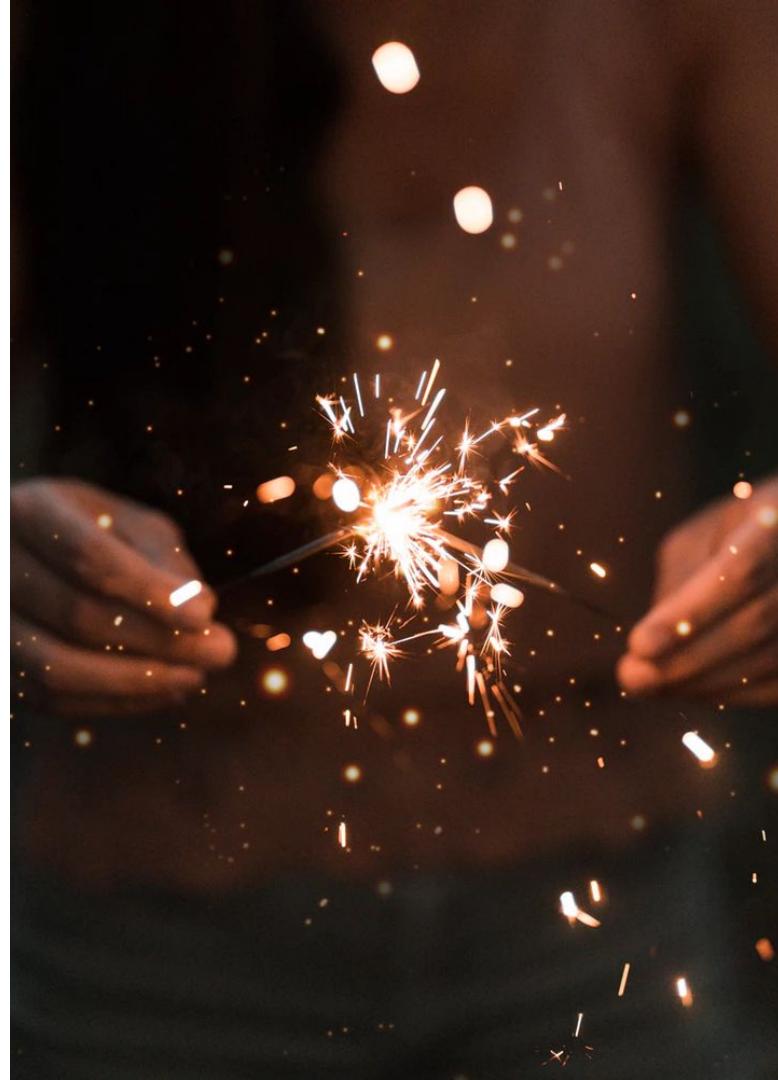
lit-element



Check out: <https://youtu.be/ypRdtjGooc>

What is lit-element?

- Base class for creating web components
- Uses lit-html to render into the element's Shadow DOM
- React to changes
- Extremely fast & light



Simple button

Here is a simple web component button

CLICK

The screenshot shows the Chrome DevTools Elements tab with the DOM tree open. The root node is the <body> element, which contains a <div> with the class "main__container". Inside this div is an <h1> element with the text "Simple button". Below it is a <p> element with the text "Here is a simple web component button". The next node is a <sa-button> element, which has an attribute "title" set to "Click". It also has a style attribute with values: --background-color: navy; --color: aqua; --font-weight: 600; --padding: 12px; and an onclick attribute set to "saFunction()". This button is rendered with the text "Click" inside it. The <sa-button> element is enclosed in a shadow root, indicated by the "#shadow-root (open)" label. The bottom of the screenshot shows the DevTools footer with tabs for html, body, Styles, Event Listeners, DOM Breakpoints, Properties, Accessibility, \$scope, and a filter bar.

```
<!DOCTYPE html>
<html class="">
  <head>...</head>
  ... <body> == $0
    <div class="main__container">
      <h1>Simple button</h1>
      <p>Here is a simple web component button</p>
      <sa-button title="Click" style="--background-color: navy; --color: aqua; --font-weight: 600; --padding: 12px;" onclick="saFunction()">
        #shadow-root (open)
          <!-->
          <style>...</style>
          <button id="text" class="btn">
            <!-->
            "Click"
            <!-->
          </button>
          <!-->
        </sa-button>
      </div>
    </body>
  </html>
```

html body

Styles Event Listeners DOM Breakpoints Properties Accessibility \$scope >

Filter :hov .cls +



```
class DemoElement extends LitElement {  
  
  @property()  
  filename;  
  
  render() {  
    return html`  
      <p>  
        ${fetchContent(this.filename,  
          (content) => html`content: ${content}`,  
          () => html`Loading...`,  
          () => html`Please enter a filename`,  
          (e) => html`Error: ${e.message}`  
        )}  
      </p>  
    `;  
  }  
}
```



#ChromeDevSummit

Check out: <https://youtu.be/ypRdtjGooc>

Theming



CSS Shadow Parts

::part()



Check out: <https://meowni.ca/posts/part-theme-explainer/>

::part

```
<x-foo>  
  #shadow-root  
  <div part="some-box"><span>...</span></div>  
  <input part="some-input">  
  <div>...</div> /* not styleable  
</x-foo>
```

::part

At a document which has <x-foo>:

```
x-foo::part(some-box) { ... }
```

```
x-foo::part(some-box):hover { ... }
```

```
x-foo::part(some-input)::placeholder { ... }
```

Angular ❤️ Web Components

<http://bit.ly/frontend-developer-love>

Thank you!



Sherry List
@SherrryLst



Ana Cidre
@AnaCidre_

Best practices

- <https://w3ctag.github.io/webcomponents-design-guidelines/>
- <https://github.com/webcomponents/gold-standard/wiki>
- <https://developers.google.com/web/fundamentals/web-components/best-practices>

Resources

- [https://www.softwarearchitekt.at/post/2018/07/13/angular-elements-part-i-a-dynamic
ic-dashboard-in-four-steps-with-web-components.aspx](https://www.softwarearchitekt.at/post/2018/07/13/angular-elements-part-i-a-dynamic-dashboard-in-four-steps-with-web-components.aspx)
- <https://meowni.ca/posts/part-theme-explainer/>
- [https://medium.com/google-developer-experts/are-web-components-a-thing-5a116
b1da7e4](https://medium.com/google-developer-experts/are-web-components-a-thing-5a116b1da7e4)
- <https://www.telerik.com/blogs/web-components-101-an-introduction-to-web-components>
- [https://www.softwarearchitekt.at/post/2019/01/27/building-angular-elements-with-the
cli.aspx](https://www.softwarearchitekt.at/post/2019/01/27/building-angular-elements-with-the-cli.aspx)