

WEB COMPONENTS

Angular

1

ANGULAR



Web Componets



ANGULAR

*“Los Componentes Web son una ‘suite’ de diferentes tecnologías que permiten crear **elementos personalizados reutilizables** — con su funcionalidad encapsulada apartada del resto del código — y utilizarlos en las aplicaciones web.”*

Fuente: developer.mozilla.org



ANGULAR

- Los Web Components **no dependen de ningún *framework***. Son tecnologías nativas del navegador para crear componentes web reutilizables y encapsulados.
- En Angular los Web Components se llaman ***Angular Elements***.
- Tecnologías:
 - Elementos personalizados (*Custom elements*)
 - Permiten crear etiquetas HTML propias con comportamiento definido por el desarrollador.
 - Shadow DOM
 - Un DOM oculto y aislado dentro de un elemento para encapsular estructura y estilos.
 - Plantillas HTML (*HTML Templates*)
 - Fragmentos de HTML que no se renderizan hasta que se clonan o activan manualmente.
 - Modulos ES (*ES modules*)
 - Archivos JavaScript que permiten importar y exportar código de forma organizada y nativa en el navegador.



ANGULAR

- **Acercas del *shadow DOM*:**
 - **Aislamiento de estilos:** Los estilos de Shadow DOM no se filtran y los estilos externos no afectan al componente.
 - **Aislamiento del DOM:** El DOM interno del componente está oculto; `document.querySelector` no puede acceder a los elementos internos.
 - **Rendimiento:** Shadow DOM añade una sobrecarga de renderizado mínima, insignificante para componentes complejos.



ANGULAR

■ Ejemplo JS:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web Component</title>
  <script>
    class GameboyImage extends HTMLElement {
      connectedCallback() {
        const noImageURL = 'https://.../noImage.png'
        const imageURL = this.getAttribute('imageURL') || noImageURL;
        this.innerHTML = `<img src=${imageURL} width='100px' height='100px'>`;
      }
    }
    customElements.define('gb-image', GameboyImage);
  </script>
</head>
<body>
  <gb-image imageURL="https://.../game_cover.png"></gb-image>
</body>
</html>
```



ANGULAR

Exportar componentes Angular como webcomponents



ANGULAR

■ Instalación:

- `npm install @angular/elements`
- `npm install @angular/platform-browser-dynamic`



ANGULAR

■ Creación:

- Crear un componente normal.
- Los argumentos de entrada se definen como `@input` en el componente:

```
export class MovieCard {  
  public movie : any;  
  private movies : any;  
  private httpService = inject(HttpService);  
  @Input() inputTitle : string = "";  
  
  ngOnChanges() {  
    console.log("Título a buscar:" + this.inputTitle);  
    this.httpService.getMoviesData().subscribe((data) => {  
      this.movies = data;  
      console.log(this.movies);  
      this.movie = this.movies.find((movie: any) =>  
movie.title.toLowerCase().includes(this.inputTitle.toLowerCase()));  
    })  
  }  
}
```



ANGULAR

■ Creación:

- Los argumentos de entrada desde el punto de vista de un componente 'normal':

```
<app-movie-card [inputTitle]="movieTitle"></app-movie-card>
```

- Los argumentos de entrada desde el punto de vista de un componente 'web':

```
<movie-card-element input-title="superman"></game-card-element>
```

Atención al cambio de nombre
automático



ANGULAR

! IMPORTANT: Avoid using the component's selector as the custom element tag name. This can lead to unexpected behavior, due to Angular creating two component instances for a single DOM element: One regular Angular component and a second one using the custom element.

■ Modificación main.ts:

```
import { Type } from '@angular/core';
import { createCustomElement } from '@angular/elements';
import { appConfig } from '../app/app.config';
import { Ps3FinderComponent } from '../app/components/ps3-finder-
component/ps3-finder-component';
import { createApplication } from '@angular/platform-browser';

async function initializeAndDefineWebComponent (
  component: Type<any>,
  tag: string
) {
  const appRef = await createApplication(appConfig);
```



ANGULAR

! IMPORTANT: Avoid using the component's selector as the custom element tag name. This can lead to unexpected behavior, due to Angular creating two component instances for a single DOM element: One regular Angular component and a second one using the custom element.

■ Modificación main.ts:

```
const customElement = createCustomElement(component, {
  injector: appRef.injector
});

if (!customElements.get(tag)) {
  customElements.define(tag, customElement);
  console.log(`Web Component '${tag}' registrado.`);
}
}

initializeAndDefineWebComponent(Ps3FinderComponent, 'ps3-finder-
web-component')
  .catch(err => console.error('Error al inicializar la aplicación
de Angular para Web Component:', err));
```



ANGULAR

- Hacer el build:
 - `ng build --configuration production --output-hashing none`
- Importar y usar:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="polyfills.9dd7ed950d755d20.js"></script>
  <script src="main.04c1512bc951463c.js"></script>
</head>
<body>
  <movie-card-element input-title="batman"></movie-card-element>
</body>
</html>
```



ANGULAR

Uso de librerías de webcomponents



ANGULAR

- Algunas librerías de web componentes:
 - Vaadin: <https://vaadin.com/>
 - Shoelance: <https://shoelace.style/>
 - Ionic Core: <https://www.npmjs.com/package/@ionic/core>
 - FAST: <https://fast.design/>
 - Material Web Components de Google (¿discontinuado?): <https://m3.material.io/develop/web>
 - Stencil: <https://stenciljs.com/>



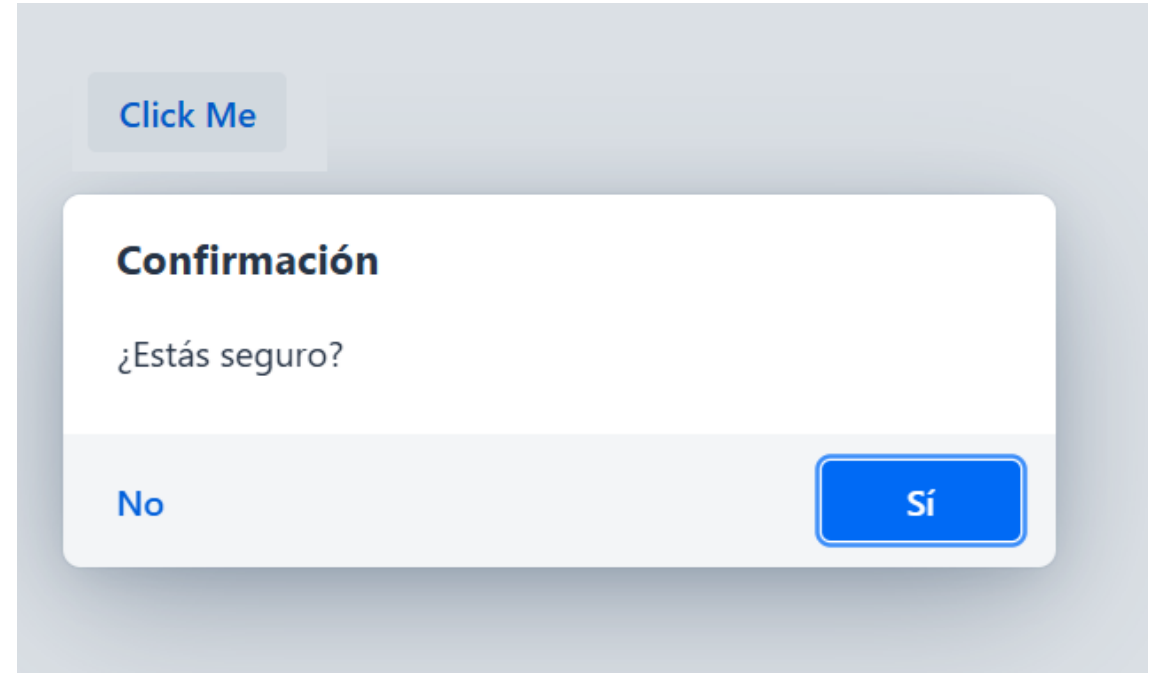
ANGULAR

- Algunas librerías de Angular (no Web Components):
 - Angular Material.
 - NgRx.
 - Nebular.
 - PrimeNG.
 - ngx-charts
 - Angular Universal



ANGULAR

- Ejemplo: VAADIN.
 - Instalar los componentes:
 - `npm install @vaadin/button`
 - `npm install @vaadin/confirm-dialog`





ANGULAR

■ Ejemplo: VAADIN.

CUSTOM_ELEMENTS_SCHEMA →
Es un esquema (conjunto de reglas)
que permite incluir librerías de
terceros (custom elements).

```
import { Component, CUSTOM_ELEMENTS_SCHEMA }
from '@angular/core';
import '@vaadin/button';
import '@vaadin/confirm-dialog';

@Component({
  selector: 'app-vaadin-component',
  imports: [],
  templateUrl: './vaadin-component.html',
  styleUrls: ['./vaadin-component.css'],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
})
export class VaadinComponent {
  public opened = false;

  abrirDialogo() {
    this.opened = true;
  }
  onConfirm() {
    this.opened = false;
    console.log("OK");
  }
  onCancel() {
    this.opened = false;
    console.log("CANCEL");
  }
}
```



ANGULAR

■ Ejemplo: VAADIN.

```
<vaadin-button (click)="abrirDialogo()">
  Click Me
</vaadin-button>
<vaadin-confirm-dialog header="Confirmación" message="¿Estás seguro?"
confirm-text="Sí" cancel-button-visible
  cancel-text="No" (confirm)="onConfirm()" (cancel)="onCancel()"
[opened]="opened" #dialog>
</vaadin-confirm-dialog>
```



ANGULAR

Integración con ~~¿Polymer?~~ Lit



ANGULAR

- Polymer está discontinuado.
- Lit → <https://lit.dev/>
- Es una librería ligera para crear Web Components usando JavaScript o TypeScript.
- Se enfoca en hacer que los componentes personalizados sean reactivos, rápidos y fáciles de mantener.
- Lit no es un framework completo como Angular o React; es solo una herramienta para construir Web Components que luego puedes usar en cualquier proyecto web, incluyendo Angular, React o incluso HTML puro.



ANGULAR

Tecnología

Qué es

Angular Elements

Permite **convertir componentes Angular en Web Components** que luego se pueden usar en cualquier proyecto web.

Lit

Librería ligera para **crear Web Components desde cero**, sin depender de Angular.



ANGULAR

- **Enlaces de interés:**

- **Web Components:**

- https://developer.mozilla.org/es/docs/Web/API/Web_components
 - <https://lenguajejs.com/webcomponents/componentes/que-son-webcomponents/>

- **Angular Elements:**

- <https://angular.dev/guide/elements>

- **Lit:**

- <https://lit.dev/>

- **How to integrate Web Components using Lit in Angular**

- <https://www.thisdot.co/blog/how-to-integrate-web-components-using-lit-in-angular>