



# Desarrollo de WebComponents con Angular, @angular/elements



Victor de Andrés · [Follow](#)

5 min read · Jan 13, 2019



1



Originally published at [victordeandres.es](https://victordeandres.es).

## Web Components



## @angular/elements

Hace ya un tiempo escribí un post introductorio a los webcomponents. Al final de aquel post indicaba varias librerías con las cuales podríamos diseñar nuestros propios webcomponents. Hoy, como continuación a aquel post vamos a añadir una nueva librería para realizar nuestros webcomponents. En esta ocasión vamos a utilizar una nueva funcionalidad disponible desde la versión 6 de angular, los `@angular/elements`.

Al final de este post podréis encontrar el enlace a todo el código que vamos a ver a continuación.

## Comencemos

Lógicamente para desarrollar nuestro primer webcomponent deberemos tener instalado como mínimo la versión de angular 6 en nuestro equipo, que ya dispone de la funcionalidad `@angular/elements` nativamente. Aunque yo voy a utilizar la versión 7 que ya se encuentra disponible para su descarga.

El primer paso será inicializar nuestro proyecto. Lo haremos con el comando `new` de `angular-cli`.

```
ng new WebComponentDemo --prefix webcomponentdemo
```

En esta ocasión he añadido el flag “prefix” para añadir “webcomponentdemo” a todos los componentes que creemos en este proyecto.

Una vez construido el esquema de nuestro proyecto debemos instalar las dependencias necesarias para crear nuestro primer webcomponent con angular. Es muy sencillo solo deberemos introducir el siguiente comando:

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

## Nuestro primer WebComponent.

Una vez instaladas todas las dependencias necesarias comenzaremos a desarrollar nuestro webcomponent.

En esta ocasión voy a realizar un componente sencillo. Un botón que irá sumando uno a nuestro contador cada vez que lo pulsemos.

Como es habitual en los proyectos en los cuales utilizamos angular usaremos el ng-cli para crear la estructura de nuestro componente.

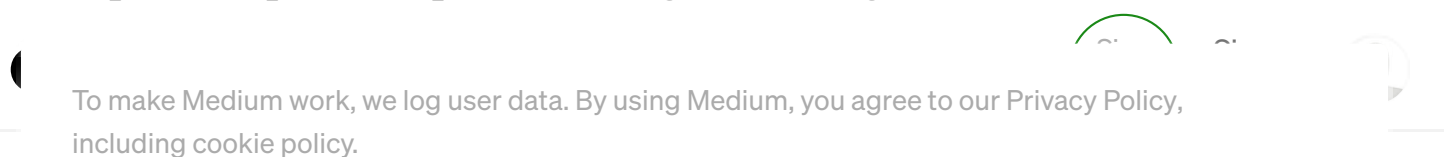
```
ng generate component incrButton --inline-style --inline-template  
--view-encapsulation Native
```

Veamos un poco más detalle los flags que hemos utilizado para crear

nuestro webcomponent.

Los dos primeros flags que hemos utilizado son ‘— inline — style’ e ‘— inline-template’. Con ellos hemos indicado que queremos que los estilos y la plantilla de nuestro componente se encuentren en un único fichero. Y no como es habitual los tres ficheros diferentes que encontramos para cada componente en los proyectos de angular. Un fichero para la lógica, otro con los estilos y un tercero con la plantilla de nuestro componente.

Y el último flag que hemos utilizado es ‘— view-encapsulation’ el cual nos permite especificar el tipo de encapsulación de nuestro componente. De esta manera facilitaremos la compatibilidad con distintos navegadores. Las opciones posibles para este flag son las siguientes:



- None: No utiliza el el Shadow DOM.
- Emulated: No utiliza el Shadow DOM pero emula su funcionamiento.
- Native Utiliza el Shadow DOM con soporte nativo del navegador.

Para este ejemplo he utilizado la opción ‘Native’, para utilizar el soporte nativo del navegador.

Una vez construido la base de nuestro webcomponent procederemos a añadir la lógica, plantilla y los estilos del mismo.

El código de nuestro webcomponent será el siguiente:

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

@Component({
  selector: 'webcomponentdemo-incr-button',
  template: ` <button class="incrButton"
(click)="handleClick()">      { {label } }</button> `,
  styles: [`
    .incrButton {
      background-color: #44c767;
      -moz-border-radius: 28px;
      -webkit-border-radius: 28px;
      border-radius: 28px;
      border: 1px solid #18ab29;
      display: inline-block;
      cursor: pointer;
      color: #ffffff;
      font-family: Arial;
      font-size: 14px;
      padding: 16px 31px;
      text-decoration: none;
      text-shadow: 0px 1px 0px #2f6627;
    }
    .incrButton:hover {
      background-color: #5cbf2a;
    }
  `],
  encapsulation: ViewEncapsulation.Native })

export class IncrButtonComponent {
  constructor() { }

  @Output() action = new EventEmitter<number>();
  @Input() label = 'Incrementa valor';

  private contadorClick = 0;
  public handleClick(): void {
    this.contadorClick++;
    this.action.emit(this.contadorClick);
  }
}
```

Como puedes ver en el código; hemos creado un botón que cada vez que interactuamos con él sumará uno a un contador que comienza en 0.

Además hemos creado dos interacciones en nuestro webcomponent. Una

de entrada o input, que nos permitirá personalizar el texto de nuestro botón, y una de salida o output que enviará el valor de nuestro contador.

## Registremos nuestro webComponent

Hasta el momento no hemos realizado nada distinto a lo que realizamos habitualmente al desarrollar un proyecto en angular. De hecho podríamos añadir nuestro componente en el fichero `app.component.html` y el componente funcionaría normalmente.

Para que nuestro componente pueda ser reutilizable y los podamos utilizar como componente embebido en cualquier otro proyecto, no solo en proyectos angular. Realizaremos las siguientes modificaciones.

Primero debemos agregar nuestro componente, en nuestro ejemplo 'IncrButtonComponent' al array de `entryComponents`, ya que el componente que hemos creado no es parte de ningún componente superior, queremos que nuestro componente sea parte del bootstrapping de la aplicación.

Y el siguiente paso es crear nuestro componente. En esta ocasión lo vamos a realizar en el constructor de nuestra función principal. Para ello debemos añadir "Injector" en la lista de nuestro `@angular/core` y "createCustomElement" desde `@angular/elements`.

Una vez realizados nuestros cambios, el fichero `app.components.ts` será:

```
import { BrowserModule } from '@angular/platform-browser';
```

```
import { NgModule, Injector } from '@angular/core';
import { createCustomElement } from '@angular/elements';
import { IncrButtonComponent } from './incr-button/incr-button.component';

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

export class AppModule {
  constructor( private injector: Injector ) {
    const el = createCustomElement(IncrButtonComponent, {
  injector });
    customElements.define('wc-incr-button', el); }
  ngDoBootstrap() { }
}
```

## Construcción | Empaquetado Web Component

El siguiente paso, una vez construido nuestro webcomponent, es la construcción o empaquetado de nuestro webcomponent. Como estamos trabajando en Angular para hacer la construcción de nuestro componente utilizaremos la instrucción:

```
ng build --prod --output-hashing=none
```

Pero utilizar directamente este comando sin ningún flag más nos generará un output de cuatro ficheros (runtime.js, scripts.js, polyfills.js y main.js), lo cual dificultará la distribución de nuestro web component.

Para construir o empaquetar nuestro webcomponent en un único fichero podemos ejecutar el siguiente comando:



```
ng build --prod --output-hashing=none && cat
dist/WebComponentDemo/ {runtime,polyfills,scripts,main }.js >
wcincrbutton.js.gz
```

Este comando realizará la construcción de nuestro webcomponent y el resultado lo unirá en un único fichero comprimido.

## Demo

Hasta aquí la construcción de nuestro webComponent con Angular. Ahora realizaremos la prueba definitiva. Utilizaremos nuestro webComponent en una página web sencilla.

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Angular Elements</title>
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <script type="text/javascript" src="element.js"></script>
</head>
<body>
  <p style="font-size: 18px">
    <span>Valor Actual:</span> <span id="currentValue">0</span>
  </p>
  <webcomponentdemo-custom-button label="Suma++">
  </webcomponentdemo-custom-button>
  <script>
    var button = document.querySelector('webcomponentdemo-
custom-button');
    button.addEventListener('action', function(event) {
      document.getElementById("currentValue").innerHTML =
event.detail;
    });
  </script>
</body>
</html>
```

En esta sencilla página web cada vez que pulsemos el botón con el texto “Suma ++”, se sumará 1 al valor actual.

Tal como indiqué al principio de este post a continuación dejo el link a este proyecto en [Github](#) donde podrás ver todo el código de este proyecto. Así como este link con la [demo](#) funcional de este ejemplo.

Angular

Web Components

Web App Development

Development

JavaScript


**Written by Victor de Andrés**

10 Followers

Follow

**More from Victor de Andrés**



 Victor de Andrés

## Tutorial implementación de notificaciones push en Angular

Tutorial paso a paso para la implantación de notificaciones push en Angular

5 min read · May 19, 2019

 20  1



 Victor de Andrés


## Tutorial. Crea tu propio servidor notificaciones push con NodeJS.

Tutorial paso a paso para la creación de un servidor notificaciones push en NodeJS.

4 min read · Jan 10, 2020

 40 



 Victor de Andrés

## JavaScript—ES6. Array VS Set

Javascript—ES6. Diferencia entre Array y Set

4 min read · May 11, 2019

 34 



 Victor de Andrés

## Crea tu propio “routing” en Svelte.

Desarrolla tu propio sistema de rutas en Svelte.

5 min read · Feb 13, 2020

 70 



[See all from Victor de Andrés](#)

## Recommended from Medium



Sinan Öztürk

### ViewProviders in Angular

First of all we can only use viewProviders key in @Component metadata.

3 min read · Mar 14, 2024



Minko Gechev in Angular Blog

### Introducing Angular v17

Last month marked the 13th anniversary of Angular's red shield. AngularJS was the...

17 min read · Nov 8, 2023



4.7K

49



## Lists

## Stories to Help You Grow as a Software Developer

19 stories · 930 saves

## General Coding Knowledge

20 stories · 1055 saves

## Growth Marketing

11 stories · 86 saves

## Modern Marketing

102 stories · 509 saves


---

 Gurinderpal Singh Narang

## Template querying in Angular using @ViewChild and...

When we want to get a reference of an element from a template in an Angular...

3 min read · Nov 3, 2023

 20  リン (linh) in Goalist Blog

## Storybook & Stencil comparisons

I. What are they


3 min read · Nov 17, 2023

 1  cory lewis

## Angular 17 Component Library wired for Web Components!

I found that there are quite a few of resources out on the web for creating an...

7 min read · Dec 18, 2023

 147  5 Murat Karagözgil

## Angular 18—New Features(Beta)

A Detailed Look at the New Features Coming in Angular 18

2 min read · Mar 12, 2024

 240  13

See more recommendations

---

[Help](#) [Status](#) [About](#) [Careers](#) [Blog](#) [Privacy](#) [Terms](#) [Text to speech](#) [Teams](#)