9-Elements

Elements for Web Components

```
# 1. Componentes independientes del framework
# 2. Desarrollo y despliegue con Angular
# 3. Consumo en HTML
```

1. Componentes independientes del framework

1.1 Origen y potencial

> Hay lácteos que aguantan más que algunos frameworks.

Los **Web Components** son independientes de los *frameworks*.

Usos posibles

- Librerías de diseño multiplataforma
- Migración paulatina de aplicaciones legacy
- Integración dinámica en grandes soluciones CMS
- Mejoras funcionales en aplicaciones server side

1.2 Estándares y tecnología

Bajo el término Web Components se esconden diversas tecnologías

- Shadow DOM
- HTML templates
- ~~HTML imports~~
- **Custom elements.**

El estándar:

> Los **Custom Web Elements** son etiquetas HTML encapsuladas reutilizables para usar en páginas y aplicaciones web. Sólo requieren HTML y JavaScript.

La tecnología:

> Angular Elements empaqueta tus componentes como **Custom Web Elements**.

2. Desarrollo y despliegue con Angular

Un componente común de Angular

libs\currency\src\lib\converter\converter.component.html

```
<form>
  <label>Amount to convert: </label>
  <input name="amount"
         [(ngModel)]="amount"
         type="number"
         (change)="convert()" />
  <label>Converted amount: </label>
  <input name="convertedAmount"
         [(ngModel)]="convertedAmount"
         type="number"
         readonly />
</form>
```

libs\currency\src\lib\converter\converter.component.ts

```
@Component({
  selector: 'angular-boss-converter',
  templateUrl: './converter.component.html',
  styleUrls: ['./converter.component.css']
})
export class ConverterComponent implements OnInit {
  @Input() factor = 1.1;
  @Input() amount = 0;
  @Output() converted = new EventEmitter<number>();
  convertedAmount = 0;
  constructor() {}
  ngOnInit() {
    this.convert();
  }
  convert() {
    this.convertedAmount = this.amount * this.factor;
    this.converted.next(this.convertedAmount);
  }
}
```

El componente sigue siendo Angular

apps\warehouse\src\app\app.module.ts

```
import { CurrencyModule } from '@angular-boss/currency';

@NgModule({
  imports: [
```

```
    CurrencyModule
  ],
})
export class AppModule {}
```

apps\warehouse\src\app\app.component.html

```
<angular-boss-converter amount="100"
                        factor="1.5"></angular-boss-converter>
```

2.1 Exponer los componentes

Necesitamos un proyecto para exportar el componente.

```
generate @nrwl/angular:application external-currency
```

Incorporamos las herramientas de Angular Elements

```
add @angular/elements
```

Exportamos el componente y eliminamos todo lo demás

apps\external-currency\src\app\app.module.ts

```
import { ConverterComponent, CurrencyModule } from '@angular-boss/currency';
import { Injector, NgModule } from '@angular/core';
import { createCustomElement } from '@angular/elements';
import { BrowserModule } from '@angular/platform-browser';
import 'zone.js';
@NgModule({
  imports: [BrowserModule, CurrencyModule],
  entryComponents: [ConverterComponent]
})
export class AppModule {
  constructor(private injector: Injector) {}
  ngDoBootstrap() {
    const el = createCustomElement(ConverterComponent, {
      injector: this.injector
    });
    customElements.define('external-currency-converter', el);
  }
}
```

> Atención a la importación de zone.js

Es necesaria para poder usar la detección de cambios en aplicaciones no angular

> Atención a ngDoBootstrap

Es necesario porque el proyecto ya no dispone de su propio componente de bootstrap

2.2 Compilación y despliegue

Toca aplicar la magia de `@angular/elements` y utilidades como `ngx-build-plus` podemos compilarlo como un Web Component.

Agregar herramientas de ayuda

```
ng add ngx-build-plus --project external-currency
```

Polyfills

Si queremos garantizar la compatibilidad en todos los navegadores

```
ng g ngx-build-plus:wc-polyfill --project external-currency
```

> ojo que algunas versiones se equivoca en la generación de los assets y hay que moverlos a mano

---

Compilación

Nombres legibles

```
"outputHashing": "none",
```

Generación

```
ng build --prod --single-bundle --project external-currency
```

3. Consumo en HTML

`apps\vanilla\index.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Vanilla Currency</title>
    <base href="/apps/vanilla/" />
    <meta name="viewport"
          content="width=device-width, initial-scale=1" />
    <link rel="icon"
          type="image/x-icon"
          href="favicon.ico" />
  </head>
  <body>
    <h2>Convert money:</h2>
    <external-currency-converter amount="15"></external-currency-converter>
  </body>
</html>
```

3.1 Copiar

```
dist\apps\external-currency

apps\vanilla\
```

3.2 Importar

```
    <script src="main-es2015.js"
            type="module"></script>
    <script src="main-es5.js"
            nomodule
            defer></script>
```

> una web HTML pura, mostrando un componente creado en Angular

> **Blog de apoyo:** Elementos Angular para los Web Components

> > By Alberto Basalo

> Next:

Angular.Builders

Angular Blueprint

Academia Binaria