

UNIVERSIDAD DE LAS FUERZAS ARMADAS-SEDE
SANTO DOMINGO

May 31

DEPARTAMENTO DE CIENCIAS DE LA
COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA
INFORMACIÓN

Personalizar Web Components

Fecha:
14/06/2022

INTEGRANTES:
Borrero Jorge
Chabla Ariel
Contreras Jessie
Gómez Jenniffer

WebComponents



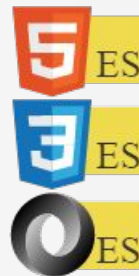
Custom Elements



Templates



Shadow DOM



Modules



CSS Scopes

Tabla de Contenido

Introducción

01

Sistema de
Objetivos

02

Desarrollo

03

Conclusiones

04

Recomendaciones

05

Bibliografía

06

Objetivo General

Investigar y comprender cómo se aplicará la personalización de web components.

Objetivos Específicos

- Indagar acerca de las definición que describen de los web components
- Conocer y analizar acerca de sintaxis y funcionalidad de los web component
- Crear ejemplos explicativos del uso de web components el cual será comprendido con los conceptos investigados.

Introducción

IONIC brinda un conjunto de plataformas de desarrollo móvil y web, cuenta con una biblioteca de interfaz de usuario con más de 100 componentes, entre ellos están los componentes web y modelos estándar para el desarrollo web.

¿Cuál es el origen de los Web Components?

Los web components surgieron como una propuesta por parte de Google a la W3C, prácticamente a la vez que apareció el framework AngularJS. En paralelo la W3C trabajaba en el concepto de los web components que “bebían” mucho del concepto utilizado en AngularJS basado en directivas.

Todo esto nacía con una serie de ideas para poder crear nuevos elementos encapsulados y ajenos a cualquier librería. Estas ideas eran:

- Mantener las etiquetas semánticas y declarativas
- Estándar recomendado por la W3C sin riesgo de acoplarse a una tecnología en concreto
- Alto rendimiento sin necesidad de sacrificar tiempo de ejecución
- Disponible en todos los sitios sin riesgo a su adopción

Estándares y tecnología



Bajo el término Web Components se esconden diversas tecnologías. No todas ellas están al mismo nivel de aceptación, e incluso alguna no ha visto la luz, pero este es un esbozo de lo que tenemos:

- **Shadow DOM:** Manipulación de un árbol en memoria antes de aplicar sus cambios al verdadero.
- **HTML templates:** Fragmentos de HTML que no se utilizan en la carga de la página, pero que se pueden instanciar más adelante.
- **ES Modules:** Inclusión de documentos JS en forma de módulos de manera estándar y ágil.
- **Custom elements:** son etiquetas HTML con funciones encapsuladas, reutilizables y listas para usar en páginas y aplicaciones web.

Las partes que componen los web components



Custom elements

Etiquetas HTML que encapsulan contenido HTML, incluyendo directrices CSS y scripts.



ES modules

Los ES modules son módulos que exportan objetos.



Shadow DOM

Permite añadir árboles DOM ocultos a un árbol de documentos.



HTML templates

Las llamadas HTML templates son de plantillas de archivos HTML

Prerrequisitos



El conocimiento práctico de TypeScript y Angular es necesario ya que estamos usando Angular para crear nuestro componente web.



También necesita tener Node.js y NPM instalados en su sistema junto con Angular CLI. Simplemente puede ejecutar `npm install -g @angular/cli` para instalar la CLI.



Deberías tener la última versión de Angular-CLI. Revisa la versión node.js: `node -v`, npm: `npm -v`.



Presentación

EJEMPLO 1º – CUSTOM ELEMENTS

```
1  import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
2  import { TestBed, waitForAsync } from '@angular/core/testing';
3
4  import { AppComponent } from './app.component';
5
6  describe('AppComponent', () => {
7
8      beforeEach(waitForAsync(() => {
9
10         TestBed.configureTestingModule({
11             declarations: [AppComponent],
12             schemas: [CUSTOM_ELEMENTS_SCHEMA],
13         }).compileComponents();
14     }));
15
```

```
1  import { NgModule } from '@angular/core';
2  import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
3
4  const routes: Routes = [
5
6    {
7      path: 'tab5',
8      loadChildren: () => import('./tab5/tab5.module').then( m => m.Tab5PageModule)
9    },
10
11  ];
12  @NgModule({
13    imports: [
14      RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
15    ],
16    exports: [RouterModule]
17  })
18  export class AppRoutingModule {}
```

```
<ion-content>
  <ion-item>
    <ion-label position="stacked">Calificacion Detallada</ion-label>
    <ion-input></ion-input>
  </ion-item>

  <ion-list>
    <ion-item>
      <ion-select
        placeholder="Seleccionar Alumno"
        (ionChange)="handleChange($event)"
        (ionCancel)="pushLog('ionCancel fired')"
        (ionDismiss)="pushLog('ionDismiss fired')"
      >
        <ion-select-option value="p1">Jorge Borrero</ion-select-option>
        <ion-select-option value="p2">Nicole Andrade</ion-select-option>
        <ion-select-option value="p3">Germania Andi</ion-select-option>
      </ion-select>
    </ion-item>
  </ion-list>
```

Datos Alumno

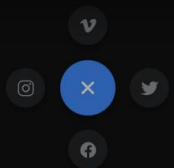
Calificación Detallada

Seleccionar Alumno ▲

Agregar y Compartir

- ☐ Jorge Borrero
- ☐ Nicole Andrade
- ☐ Germania Andi

CANCEL OK

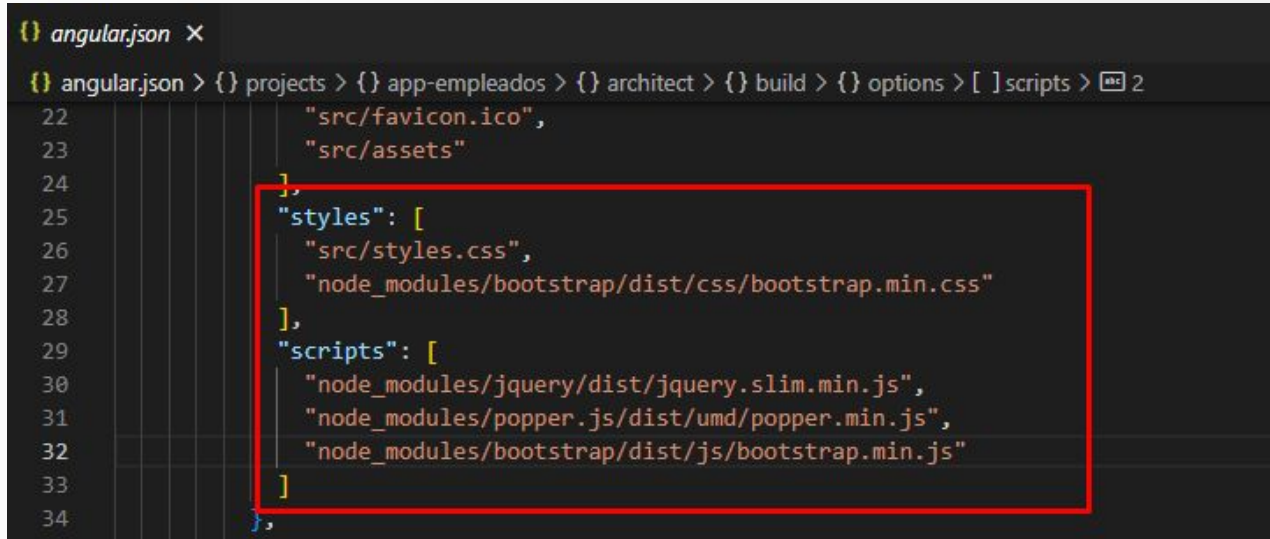


EJEMPLO 2º - HTML TEMPLATE

```
tsconfig.json X
tsconfig.json > {} compilerOptions
1  /* To learn more about this file see: https://angular.io/config/tsconfig. */
2  {
3    "compileOnSave": false,
4    "compilerOptions": {
5      "strictPropertyInitialization": false,
6      "baseUrl": ".",
7      "outDir": "dist/out-tsc",
```

Tendremos que deshabilitar la opción de `strictPropertyInitialization` en el fichero `tsconfig.json`, asignándole el valor `false`. Hacemos esto porque, a partir de Angular 12, el modo estricto está habilitado por defecto, lo que quiere decir que TypeScript protestará si intentamos declarar una propiedad de clase sin asignarle un valor en el constructor (una práctica muy común en Angular).

Se añaden los estilos y script del bootstrap tanto para el css como para el formato Json



```
{ } angular.json x
{ } angular.json > { } projects > { } app-empleados > { } architect > { } build > { } options > [ ] scripts > 2
22     "src/favicon.ico",
23     "src/assets"
24   ],
25   "styles": [
26     "src/styles.css",
27     "node_modules/bootstrap/dist/css/bootstrap.min.css"
28   ],
29   "scripts": [
30     "node_modules/jquery/dist/jquery.slim.min.js",
31     "node_modules/popper.js/dist/umd/popper.min.js",
32     "node_modules/bootstrap/dist/js/bootstrap.min.js"
33   ]
34 }
```


ngModel.- Realiza un seguimiento del valor vinculado a esta directiva

```
<form class="form-group row">
  <div class="form-group col-md-3">
    <label for="nombre">Nombre:</label>
    <input type="text" name="nombre" id="nombre" placeholder="Nombre" class="form-control" [(ngModel)]="cuad<
  </div>
  <div class="form-group col-md-3">
    <label for="apellido">Apellido:</label>
    <input type="text" name="apellido" id="apellido" placeholder="Apellido" class="form-control" [(ngModel)]<
  </div>
  <div class="form-group col-md-3">
    <label for="cargo">Cargo:</label>
    <input type="text" name="cargo" id="cargo" placeholder="Cargo" class="form-control" [(ngModel)]="cuadroC<
  </div>
  <div class="form-group col-md-3">
    <label for="salario">Salario:</label>
    <input type="text" name="salario" id="salario" placeholder="Salario" class="form-control" [(ngModel)]="c<
  </div>
```

La directiva ngFor es la que se encarga de presentar una lista de elementos en pantalla de una forma sencilla combinando el concepto de bucle y plantilla.

```
<app-empleado-hijo-c *ngFor="let empleado of empleados; let i=index"
[empleadoDeLista]="empleado" [Indice]="i"></app-empleado-hijo-c>
```

```
src > app > empleado-hijo-c > TS empleado-hijo-c.component.ts > ...
1  import { Component, Input, OnInit } from '@angular/core';
2  import { Empleado } from '../empleado.model';
3
4  @Component({
5    selector: 'app-empleado-hijo-c',
6    templateUrl: './empleado-hijo-c.component.html',
7    styleUrls: ['./empleado-hijo-c.component.css']
8  })
9  export class EmpleadoHijoCComponent implements OnInit {
10
11    @Input() empleadoDeLista: Empleado;
12    @Input() indice: number;
13
14    constructor() { }
15
16    ngOnInit(): void {
17    }
18
19  }
```

app.component.html TS empleado-hijo-c.component.ts empleado-hijo-c.component.html X

src > app > empleado-hijo-c > empleado-hijo-c.component.html > div.container > div.row > div.col

Go to component

```
1 <div class="container">
2   <div class="row">
3     <div class="col">
4
5       {{Indice+1}}: {{empleadoDeLista.nombre}} {{empleadoDeLista.apellido}}
6       {{empleadoDeLista.cargo}} {{empleadoDeLista.salarario}}
7     </div>
8   </div>
9 </div>
10
```

empleado-hijo-c.component.html

TS app.component.ts X

src > app > TS app.component.ts > AppComponent > agregarEmpleado

```
9 })
10 export class AppComponent {
11   titulo = 'Listado de Empleados';
12
13   empleados:Empleado[]=[
14
15     new Empleado("Juan", "Diaz", "Presidente", 7500),
16     new Empleado("Camila", "Fernandez", "Secretaria", 700),
17     new Empleado("Pedro", "Lorenti", "Vendedor", 500),
18     new Empleado("Lourdes", "Quiñonez", "Vendedor", 500),
19
20   ];
21
22   agregarEmpleado(){
23     let miEmpleado=new Empleado(this.cuadroNombre, this.cuadroApellido, this.cuadroCargo, this.cuadroSalarario);
24     this.empleados.push(miEmpleado);
25   }
26
27   cuadroNombre:string="";
28   cuadroApellido:string="";
29   cuadroCargo:string="";
30   cuadroSalarario:number=0;
```

Listado de Empleados

Nombre:

Apellido:

Cargo:

Salario:

Agregar

- 1: Juan Diaz Presidente 7500
- 2: Camila Fernandez Secretaria 700
- 3: Pedro Lorenti Vendedor 500
- 4: Lourdes Quiñonez Vendedor 500

← → ↻ ⓘ localhost:4200

Dimensions: Surface Pro 7 ▾ 912 x 1368 75% ▾ No throttling ▾

Listado de Empleados

Nombre: Apellido: Cargo: Salario:

Agregar

app-empleado-hijo-c 912 x 24

1: Juan Diaz Presidente 7500
2: Camila Fernandez Secretaria 700
3: Pedro Lorenti Vendedor 500

DevTools is now available in Spanish! Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance

```
<html lang="en">
  <head>...</head>
  <body>
    <app-root _ngghost-oal-c49 ng-version="14.0.4">
      <div _ngcontent-oal-c49 class="container">...</div>
      <br _ngcontent-oal-c49>
      <br _ngcontent-oal-c49>
      <br _ngcontent-oal-c49>
      <app-empleado-hijo-c _ngcontent-oal-c49 _ngghost-oal-c48 ng-reflect-empleado-de-lista="[object Object]" ng-reflect--index="0">
        <div _ngcontent-oal-c48 class="container">
```

EJEMPLO 3º – CUSTOM ELEMENTS

Custom_Elements_Schema define un esquema que permite que un NgModule contengan lo siguiente:

- Crear un módulo que importe sus componentes web y lo envuelve en un componente angular real.
- Permite agregar a cada componente en el que esté utilizando la etiqueta HTML personalizada.
- Permite cualquier etiqueta de HTML en la plantilla Html para ese componente.

```
import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { TestBed, waitForAsync } from '@angular/core/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

  beforeEach(waitForAsync(() => {

    TestBed.configureTestingModule({
      declarations: [AppComponent],
      schemas: [CUSTOM_ELEMENTS_SCHEMA],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  });
  // TODO: add more tests!

});
```


Extrae los datos de la dirección indicada, especificando la ruta en donde se encuentran especificados los comandos.

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
  },
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'update',
    loadChildren: () => import('./update/update.module').then( m => m.UpdatePageModule)
  },
  {
    path: 'resultado',
    loadChildren: () => import('./resultado/resultado.module').then( m => m.ResultadoPageModule)
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```



REGISTRO



REGISTRO CUSTOM

Crear un menú agregando un direccionamiento con routerLink que permite aplicar un elemento en una plantilla, convirtiendo ese elemento en un enlace para iniciar la navegación a una ruta.

Puede abrir uno o más componentes enrutados de más ubicación en la página.

```
<div id="container">

  <ion-grid>
    <ion-row>
      <ion-col>
        <ion-icon name="laptop-outline" style="font-size: 90px;"></ion-icon>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-button color="success" [routerLink]="['/update']">Registro</ion-button>
      </ion-col>
    </ion-row>
  </ion-grid>

  <ion-grid>
    <ion-row>
      <ion-col>
        <ion-icon name="reader-outline" style="font-size: 90px;"></ion-icon>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-button color="success" [routerLink]="['/resultado']">Registro Custom</ion-button>
      </ion-col>
    </ion-row>
  </ion-grid>

</div>
/ion-content>
```


TS resultado.module.ts U X

Prueba > src > app > resultado > TS resultado.module.ts > ResultadosPageModule

```
5 import { IonicModule } from '@ionic/angular';
6
7 import { ResultadosPageRoutingModule } from './resultado-routing.module';
8
9 import { ResultadosPage } from './resultado.page';
10 import { ResultadosInfoComponent } from '../resultadoinfo/resultadoinfo.component';
11 import { ComponentesModule } from '../componentes/componentes.module';
12 import { RouterModule } from '@angular/router';
13
14 @NgModule({
15   imports: [
16     ComponentesModule,
17     CommonModule,
18     FormsModule,
19     IonicModule,
20     RouterModule.forChild([
21       {
22         path:'',
23         component: ResultadosPage
24       }
25     ])
26   ],
27   declarations: [ResultadosPage]
28 })
29 export class ResultadosPageModule {}
30
```

TS componentes.module.ts U X

Prueba > src > app > componentes > TS componentes.module.ts > ComponentesModule

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { ResultadosInfoComponent } from '../resultadoinfo/resultadoinfo.component';
4 import { IonicModule } from '@ionic/angular';
5
6
7
8 @NgModule({
9   declarations: [
10     ResultadosInfoComponent
11   ],
12   imports: [
13     CommonModule,
14     IonicModule,
15   ],
16   exports:[
17     ResultadosInfoComponent
18   ]
19 })
20 export class ComponentesModule { }
21
```

<> resultado.page.html U X

Prueba > src > app > resultado > <> resulta

1

2 <app-resultadoinfo>

3

4 </app-resultadoinfo>

<> resultadoinfo.component.html U X

Prueba > src > app > resultadoinfo > <> resultadoinfo.component.html > ion-content > ion-list > ion-item > ion-label

```
2   <ion-header>
3     <ion-toolbar>
4       <ion-title>Registro de Datos</ion-title>
5     </ion-toolbar>
6   </ion-header>
7
8   <ion-content [fullscreen]="true">
9     <ion-item color="primary">
10       <ion-label>Datos del Clientes</ion-label>
11     </ion-item>
12     <ion-list>
13       <ion-item>
14         <ion-label position="stacked">Nombre: </ion-label>
15         <ion-input></ion-input>
16       </ion-item>
17       <ion-item>
18         <ion-label position="stacked">Apellido: </ion-label>
19         <ion-input></ion-input>
20       </ion-item>
21       <ion-item>
22         <ion-label position="stacked">Cédula: </ion-label>
23         <ion-input></ion-input>
24       </ion-item>
25       <ion-item>
26         <ion-label position="stacked">Dirección: </ion-label>
27         <ion-input></ion-input>
28       </ion-item>
29
30     <ion-list>
31       <ion-radio-group value="F">
32         <ion-list-header>
33           <ion-label>Sexo</ion-label>
34         </ion-list-header>
35
36         <ion-item>
37           <ion-label>Femenino</ion-label>
38           <ion-radio slot="start" value="F"></ion-radio>
39         </ion-item>
40     </ion-list>
```

Registro de Datos

Datos del Clientes

Nombre:
Jennifer

Apellido:
Gómez

Cédula:

Dirección:

Sexo

☒ Femenino

☐ Masculino

Quiere recibir notificaciones de nuevas actualizaciones



ENVIAR

```

src > app > component > empresa > TS empresa.component.ts > Empresa
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-empresa',
5    templateUrl: './empresa.component.html',
6    styleUrls: ['./empresa.component.scss'],
7  })
8
9  export class Empresa implements OnInit {
10
11    @Input('progressValue') progressValue;
12    data: any = { myToggle: true };
13    constructor() { }
14
15    ngOnInit() {}
16
17  }
18

```

OnInit permite definir su propio método de inicialización a cualquier componente.

```

<ion-list>
  <ion-item>
    <ion-label>Ver la designación de empleado a la Nueva Sucursal</ion-label>
    <ion-toggle (click)="isClicked(data.myToggle)" [(ngModel)]="data.myToggle" color="primary">
    </ion-toggle>
  </ion-item>
</ion-list>

<div *ngIf="data.myToggle; then sucursal else empleado"></div>

<ng-template #sucursal>
  <ion-card>
    <ion-card-header>
      Información de Sucursal
    </ion-card-header>

    <ion-list lines="none">
      <ion-item>
        Nombre: Importadora Luz de América
      </ion-item>

      <ion-item>
        Ubicación: Kilómetro 24 Vía Quevedo
      </ion-item>
      <ion-item>

```

```

<ion-header>
  <ion-toolbar>
    <ion-title class="ion-text-center">Asignación de Empeados</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <app-empresa></app-empresa>
</ion-content>

```



Conclusiones

The slide features a decorative graphic on the right side. It includes two hexagons, one smaller than the other, both with a blue-to-red gradient. To their right is a large, thick, stylized arrow pointing towards the right, also with a blue-to-red gradient. The background is white with a blue-to-red gradient on the right side.

Los componentes web personalizados en iónico permiten agregar componentes web y devolverlos en componente angular real, que pueden ser utilizados en las plantillas de HTML.

Gracias a la implementación de componentes web se pudo reutilizar código de grandes y pequeños fragmentos estéticos del proyecto.

Recomendaciones



Verificar la sintaxis de cada etiqueta.



Añadir schema a cada componente, para que agregan los componentes en la plantilla.



Content

Referencias

Angular. (s. f.). Recuperado 12 de julio de 2022, de <https://angular.io/>



Gracias