

## Práctica 8

### TypeScript / Angular - PARTE 1

---

Para la próxima etapa de la asignatura se recomienda trabajar con Visual Studio Code.

#### TypeScript

Instale TypeScript (ver la sección *Tips de instalación y ejecución* en la teoría de Angular - Parte 1)

1. Genere los siguientes archivos en alguna carpeta o directorio:

##### **index.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
<title></title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1"
</head>
<body>
<h1 id="encabezado"></h1>
<script type="text/javascript" src="test.js"></script>
</body>
</html>
```

##### **test.ts**

```
let nombre: string = "Juan";
function saludo(nombre){
    return "Hola "+nombre;
}
document.getElementById("encabezado").innerHTML = saludo(nombre);
```

Luego ejecute el compilador de TypeScript desde la línea de comandos del sistema operativo:

**tsc test.ts**

2. Realice los siguientes incisos.
  - a) Genere el siguiente archivo:

##### **persona.ts**

```
class Persona{
    nombre: string;
}

let persona = new Persona();
```

```
persona.nombre = "Juan";
if (persona.nombre) {
    console.log(persona.nombre);
}
```

b) Luego ejecute el compilador de Typescript:

**tsc persona.ts**

- c) Establezca la propiedad **nombre** como **private** en la clase Persona
- d) Vuelva a compilar. ¿Generó errores?. ¿Generó el archivo persona.js?
- e) Pruebe compilar con distintos target (opción -t es3, ó -t es5 ó -t es6) y analice las diferencias en los archivos .js generados.

### 3. Getter & setters en Typescript

a) Genere el siguiente archivo:

**persona2.ts**

```
class Persona {
    private nombre: string;
    get nombre(): string {
        return this.nombre;
    }
    set nombre(nombre: string) {
        this.nombre = nombre;
    }
}

let persona = new Persona();
persona.nombre = "Saul Goodman";
if (persona.nombre) {
    console.log(persona.nombre);
}
```

b) Luego ejecute el compilador de Typescript:

**tsc persona2.ts**

## Angular

4. Angular utilizando *angular-cli* - <https://cli.angular.io/> (ver la sección *Tips de instalación y ejecución* en la teoría de Angular - Parte 1)

- a) Cree un proyecto Angular llamado *proyectoX* utilizando el comando:

```
ng new proyectoX
```

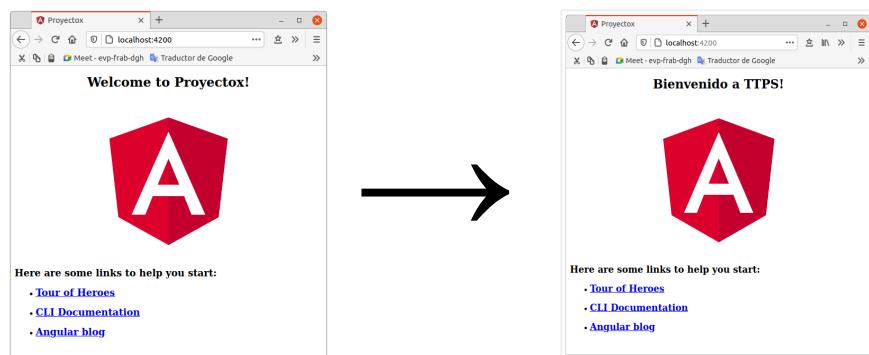
- b) Luego desde la carpeta del proyecto, construya la aplicación y arranque el web server

```
cd proyectoX/
ng serve
```

- c) Visualice el funcionamiento del proyecto desde un navegador en la URL:

<http://localhost:4200/>

- d) Analice la estructura del proyecto y luego cambie el texto “Welcome to ProyectoX” por “Bienvenido a TTPS”



5. Agregado de *Bootstrap* al proyecto

Es posible usar Bootstrap mediante CDN de librerías dentro de la etiqueta `<head></head>` o podemos instalarlo localmente mediante NPM con el comando:

```
npm install -save bootstrap
```

Una vez instalado, lo importamos en el archivo `angular.json` mediante la modificación del código:

*angular.json*

```
...
styles: [
    "node_modules/bootstrap/dist/css/bootstrap.min.css",
    "styles.css"
]
```

En el archivo **styles.css**, ubicado en la carpeta src, se pueden incluir todos los estilos CSS globales a toda la aplicación, sin tener que incluir una llamada al mismo en el index.html.

## 6. Creando modelo y componente

Se desea modelizar la registración de categorías. Para ello realice los siguiente pasos:

- a) Cree una clase modelo llamada **Categoría** de la siguiente manera:

```
cd src/app/
mkdir modelos
cd modelos
ng generate class categoria
```

- b) Escriba el contenido del archivo *src/app/modelos/categoria.model.ts*

```
export class Categoria {
    public nombre: string;
    public descripcion: string;

    constructor (nombre: string, descripcion: string){
        this.nombre = nombre;
        this.descripcion = descripcion;
    }
}
```

- c) Cree un componente Angular llamado *CategoriasComponent* utilizando el comando:

```
cd ..
ng g c categorias --spec false
```

- d) Escriba el contenido del archivo *src/app/categorias/categorias.component.ts*

```
import { Component, OnInit } from '@angular/core';
import { Categoria } from '../modelos/categoria';

@Component({
    selector: 'app-categorias',
    templateUrl: './categorias.component.html',
    styleUrls: ['./categorias.component.css']
})
export class CategoriasComponent implements OnInit {

    categorias: Categoria[] = [
        {'nombre':'Arte', 'descripcion':'Emprendimientos artísticos de cualquier tipo'},
        {'nombre':'Comida y bebidas', 'descripcion':'Emprendimientos gastronómicos y perfiles que comparten recetas'},
        {'nombre':'Ciencia y tecnología', 'descripcion':'Divulgación científica y emprendimientos tecnológicos'}];
    constructor() {}
```

```
    ngOnInit() {}  
}
```

- e) Escriba el contenido del archivo *src/app/categorias/categorias.component.html* para mostrar los datos de la categoría mediante interpolación

```
<h4 class="text-primary">Categorías</h4>  
<table class="table table-striped table-dark">  
  <thead>  
    <tr>  
      <th scope="col">Nombre</th>  
      <th scope="col">Descripción</th>  
    </tr>  
  </thead>  
  <tbody>  
    @for(categoría of categorías; track categoría.nombre) {  
      <tr>  
        <td>{{categoría.nombre}}</td>  
        <td>{{categoría.descripcion}}</td>  
      </tr>  
    }  
  </tbody>  
</table>
```

- f) Referencie el componente *Categorias* desde *src/app/app.component.html*

```
<div class="container" >  
  <h3 class="font-weight-bold text-center">Categorías</h3>  
  <app-categorias></app-categorias>  
</div>
```

Note que el tag utilizado es el selector declarado en el componente *CategoriasComponent*

- g) Visualice los cambios de la aplicación desde el navegador.

- h) A modo de prueba, modifique *src/app/categorias/categorias.component.html* para aplicar al nombre la categoría un pipe de formato, por ejemplo:

```
<td>{{categoría.nombre|uppercase}}</td>
```

- 7) Utilizando una capa de servicios con datos simulados (mock)

- a) Cree un directorio llamado servicios
- b) Cree una clase llamada *CategoriasServicio* mediante el comando:  
**ng g class servicios/categorias-servicio**
- c) Agregue a *CategoriasServicio* una variable estática llamada *categorias* inicializada con el arreglo de categorías del inciso d). Esta información estará disponible para toda la aplicación.

- d) Modifique *CategoriasComponent* para que inicialice su variable de instancia categorías a partir del arreglo obtenido de *CategoriasServicio*. Es aconsejable realizar este tipo de operaciones desde el método `ngOnInit()` del componente.

#### 8) Creando la registración de categorías

- Cree otro componente llamado *RegistrarCategoriaComponent* que permita agregar nuevos categorías a la aplicación. Para llevar a cabo esta tarea deberá agregar las nuevas categorías en el arreglo facilitado por *CategoriasServicio*.
- Cree en el template *RegistrarCategoriaComponent* un formulario para ingresar los datos de registración de categorías.
- Referencie el componente *RegistrarCategoriaComponent* utilizando su selector desde el template *CategoriasComponent*
- Verifique desde el navegador que aparece el formulario debajo del listado de categorías

Nombre	Descripción
Arte	Emprendimientos artísticos de cualquier tipo
Comida y bebidas	Emprendimientos gastronómicos y perfiles que comparten recetas
Ciencia y tecnología	Divulgación científica y emprendimientos tecnológicos

Nombre  Ocultar

Descripción

#### Utilizando la directiva `@if`

- e) Diseñe una solución para poder mostrar y ocultar el formulario de registración de categorías.

Nombre	Descripción
Arte	Emprendimientos artísticos de cualquier tipo
Comida y bebidas	Emprendimientos gastronómicos y perfiles que comparten recetas
Ciencia y tecnología	Divulgación científica y emprendimientos tecnológicos