**Curso de Angular Básico**

1. TypeScript

* Comando para ver la versión de TypeScript:

***tsc --version***

* Comando para transpilar código JavaScript:

***tsc appName.ts***

* Comando para crear el archivo de configuración JSON:

***tsc --init***

* Comando de observación de cambios de TypeScript:

***tsc -w***

***tsc --watch***

* Con let le asignamos espacio en memoria diferente a una variable según el scope, ejemplo:

En JS:

**var variable**

**var variable\_1**

En TS:

**let variable**

**let variable**

* Para excluir directorios en la transpilación de TS a JS, se configura el tsconfig.json escribiendo "exclude":[ directorioExcluido ] en la primera línea después de la llave.
* Estructura de una función auto invocada:

**(function(){**

**})();**

* Estructura de una función flecha auto invocada:

***(()=>{***

***})();***

* Para desestructurar un objeto en TS se utilizan las llaves normales, pero para desestructurar un arreglo se utilizan las llaves cuadradas.
* Las promesas siempre se declaran bajo dos argumentos, el ***resolve*** y el ***reject***.
* Clases:

***class NombreClase {***

***constructor(***

***){}***

***}***

Se pueden inicializar las variables de dos formas, una donde se referencia anteponiendo el ***this*** y otra donde se inicializan con ***public***.

2. Angular

* Comando para instalar Angular CLI:

***npm install -g @angular/cli***

* Comando para crear un proyecto Angular:

***ng new nombreApp***

* Comando para inicializar el servicio:

***ng serve***

* Comando para inicializar el servicio abriendo el proyecto:

***ng serve -o***

* Se conoce como interpolación cuando llamamos variables con la sintaxis de llave llave:  
    
  ***{{ variable }}***
* Explicación de los directorios:
  + ***e2e:*** Sirve para realizar pruebas end to end.
  + ***node\_modules:*** Sirve para ejecutar los servicios de NodeJS.
  + ***.editorconfig:*** Configuraciones del editor.
  + ***.gitignore:*** Configuración de git.
  + ***angular.json:*** Indica a Angular como es y cómo funciona la aplicación.
  + ***package-lock.json***: Indica a NodeJS como fue creado el package.json.
  + ***package.json:*** Indica la configuración del NodeJS.
  + ***README.md:*** Explica como funciona la aplicación.
  + ***tsconfig.js:*** Indica la configuración de trabajo de TypeScript.
  + ***tslint.json:*** Indica las reglas de codificación.
  + ***src:*** Directorio principal.
    - ***app:*** Directorio de la aplicación.
    - ***app.component.css:***  Archivo CSS que se aplica solo al HTML del directorio donde se encuentre.
    - ***app.component.html:*** Archivo HTML que se utiliza solo en el directorio donde se encuentre.
    - ***app.component.spec.ts:*** Archivo de pruebas automáticas.
    - ***app.module.ts:*** Archivo de los módulos del componente TS.
  + ***assets:*** Directorio para recursos estáticos.
    - ***.gitkeep:*** Archivo que hace que a pesar de que esté vacía la carpeta, se mantenga para cargarse en el repositorio.
  + ***environments:*** Directorio que almacena los archivos de las variables de entorno.
    - ***enviroment.prod.ts:*** Archivo que almacena las variables de entorno de producción.
    - ***enviroment.ts:*** Archivo que almacena las variables de entorno de desarrollo.
  + ***browserslist:*** Utilizado para mejorar el soporte del CSS.
  + ***karma.conf.js:*** Archivo de configuración de pruebas con karma.
  + ***main.ts:*** Archivo con el código de lanzamiento de la aplicación.
  + ***polyfills.ts:*** Archivo con las funciones que ayudan a la compatibilidad de versiones antiguas de navegadores web.
  + ***styles.css:*** Archivo de estilos globales de la aplicación.
  + ***tsconfig.app.json:*** Archivo con las configuraciones propias del TypeScript.
  + ***tsconfig.spec.json:*** Archivo con las configuraciones para realizar las pruebas.
  + ***tslint.json:*** Archivo para la configuración de las reglas de codificación.
* Creando un componente con Angular CLI:

***ng generate component directorio/nombreComponente***

***ng g c directorio/nombreComponente***

* Creando un componente con Angular CLI sin estilos:

***ng g c directorio/nombreComponente -is***

**(**is significa inline style**)**

* El elemento con ***\*ngIF*** se destruye o se crea según se necesite.
* Tener en cuenta que con las Directivas yo puedo colocar código como quisiera.
* Si se crea un componente por error, simplemente se borra del directorio y la importación del módulo.
* Para instalar Boostrap y que quede almacenada en los módulos, se ejecuta el comando:

***npm install bootstrap --save***

* Se puede modificar el archivo ***angular.json*** con la finalidad de definirle las librerías que deseemos que se configuren, si es el caso de librerías de ***css***, entonces se coloca en ***“styles”:[directorioLibreríaCSS],*** el directorio de la librería si se instala con el ***npm***, se encontraría en ***node\_modules***, si fuera un ***script*** o un librería entonces se colocaría en ***“scripts”:[directoioLibreríaScript]***
* Cuando especificamos las rutas es necesario que especifiquemos donde queremos que se renderice la aplicación, para eso se coloca en el HTML del app ***< router-outlet > </router-outlet >***.
* Para redirección a una ruta en especifica utilizo un arreglo de direcciones en el HTML:

***[routerLink]="['ruta']"***

(www.mipagina.com/ruta)

***[routerLink]="['ruta', 'ruta2', 'ruta3']"***

(www.mipagina.com/ruta/ruta2/ruta3)

* Para especificar si un enlace de un navegador está activo o no, debemos especificar en el elemento HTML esta propiedad con:

***routerLinkActive="active"***

* En el ***app.module.ts*** se colocan los servicios en la sección de ***providers:[servicio]***.
* Para utilizar el servicio se debe importar primero en el componente que se desea.
* Se puede especificar que el ***src*** se cargue con interpolación o que se cargue por medio de llaves cuadradas, así:

***[src]=”arreglo.valor”***

(Mejor de esta forma para indicarle que lo va a manejar Angular)

***src=”{{ arreglo.valor}}”***