ANGULAR

CLASE #1

Es un framework para el desarrollo de aplicaciones web. Está pensado para dividir un proyecto en componentes. Y ser reutilizado en proyectos medianos y grandes.

Sus principales competidores son Vue (Proyecto iniciado por Evan Vue en 2014) y React (Proyecto iniciado por Facebook en 2013).

Angular tiene su salida al mercado en 2016 pero tiene una versión previa no compatible y solo mantenida para proyectos antiguos llamada Angular Js (2010), el proyecto de angular es propiedad de la empresa Google.

**NOTA: PARA DESARROLLAR EN FORMA EFECTIVA UNA APLICACIÓN EN ANGULAR DEBEMOS INSTALAR AL MENOS DOS HERRAMIENTAS BÁSICAS:**

* + - **Node.js**
    - **Angular CLI (Comand line interface)**

Instalación node.js:

Lo primero que debemos instalar será node.js esto debido a que gran cantidad de programas para el desarrollo en angular están implementadas en node y luego reiniciamos el pc.

Comandos:

* Window + R : Nos abre una ventana de ejecución de sistema en la cual procederemos a abrir un terminal escribiendo la palabra CMD
* Node -v : Comprobar versión node

Instalación Angular CLI:

Para instalar este software lo hacemos desde la misma línea de comandos de node.js (Por eso lo instalamos primero), y escribimos el siguiente comando:

* cmd: npm i -g @angular/cli

Creación de un proyecto y prueba de funcionamiento:

Para crear un proyecto vamos a utilizar la aplicación Angular CLI que acabamos de instalar.

Desde la línea de comando del node.js ejecutamos el siguiente comando.

* ng new nombre\_proyecto

Cuando ejecutamos el comando para crear nuestro proyecto, nuestro terminal nos va a hacer la pregunta de querer compartir detalles y cambios de forma anónima con Google en la cual debemos seleccionar sí o no con la letra Y o N según la decisión del desarrollador, posteriormente presionamos enter. Posteriormente nuestra terminal nos pedirá la opción de instalar angular routing y debemos escoger si con la letra Y o no con la letra N. Después de configurar las rutas, nuestro terminal nos pedirá seleccionar un formato de estilos que ya vienen preestablecidos para proyectos angular las cuales son:

* CSS.
* SCSS.
* Sass.
* Less.

Del listado que nos arroja la terminal debemos desplazarnos con las flechas del teclado y seleccionar nuestras hojas de estilo de preferencia y presionar enter.

Levantar servicios:

Se refiere a generar nuestro servidor de forma local con node js. Para ello deberemos acceder a la carpeta raíz de nuestro proyecto y ejecutar el comando:

\* ng serve -o

Para detener el servidor que hemos levantado de nuestra aplicación deberemos presionar las teclas ctrl + c, el terminal nos preguntara si deseamos detener el servidor.

En la carpeta raíz de nuestro proyecto ejecutar el comando:

* code . : este comando nos abre el proyecto en visual studio code.

Componentes:

En la carpeta de nuestro proyecto existe una subcarpeta llamada (SRC), y dentro de esta otra carpeta llamada (APP) y dentro de esta subcarpeta un archivo llamado (app.component.html).

El archivo con extensión .HTML es la pagina inicial o principal de nuestro proyecto angular el cual tiene una estructura base de etiqueta HTML y estilos ya predestinado el cual podemos modificar a nuestro modo.

En angular se programa utilizando el lenguaje TypeScript y la lógica se trabaja con los archivos (component.ts).

El concepto de asignación de variables se llama interpolación.

Otro archivo que se asocia a componente es (app.component.css) donde se almacena todos los estilos que se van a aplicar a nuestros componentes.

Otro archivo que crean los componentes se llama (app.component.spec.ts) este archivo tiene por objetivo definir Código de testing para medir el correcto funcionamiento de nuestros componentes.

Otro archivo fundamental que nos crea Angular CLI es (app.module.ts) en la misma carpeta donde se encuentran los 4 archivos del componente.

Interpolación en los archivos HTML de Angular

Una de las características principales de Angular es separar la vista del modelo de datos. En el modelo de datos tenemos las variables y en la vista implementamos como se muestran los datos.

Para acceder a las propiedades dentro del template del HTML se utiliza {{Nombre\_Variable}}.

Cuando se trata de vectores la primera forma que podemos acceder a ellos es mediante un subíndice {{Nombre Variable [0]}}.

En principio podemos decir que si los datos son siempre los mismos no tiene sentido definir propiedades en la clase y sustituimos luego en el HTML, posteriormente veremos que las propiedades las vamos a cargar mediante una petición a un servidor web, en esas circunstancias veremos la potencia que tiene modificar las propiedades y luego de forma inmediata se modifica la vista.

TRABAJO:

* Crear proyecto en angular con el nombre 'Proyecto001'
* Tomar capturas del paso a paso
* Realizar las 4 operaciones básicas de la matemática

CLASE #2

Captura de eventos:

Otra actividad muy común es la captura de eventos. La presión de un botón, la presión de una tecla, el desplazamiento del cursor. Son eventos que podemos capturar.

El evento más común que podemos encontrar en cualquier aplicación es la presión de un botón.

Directiva ngModel:

Esta directiva nos permite tener una clase unidireccional entre una propiedad de nuestra clase con el valor que se muestra, un control de formulario HTML de tipo input, textarea, select, etc.

**Nota: Cuando utilizamos la directiva ngModel debemos importar la clase “FormsModule” en el archivo app.component y especificar en la propiedad imports.**

CLASE #3

Componentes:

Creación:

Hasta ahora siempre hemos desarrollado toda la lógica en el componente que se crea por defecto al crear un proyecto en Angular. La realidad es que en proyectos de mediano y gran tamaño no podemos disponer toda la lógica en un único componente. Los componentes son una de las características fundamentales de Angular. Ayudan a extender las características básicas de las etiquetas HTML y encapsular código, estos componentes los debemos crear desde la línea de comandos de Node js, es decir, desde la terminal.

Para crear un componente debemos ejecutar en la terminal el comando “ng generate component “nombre“.

**Nota: Los componentes por buenas prácticas deben estar en minúsculas y no usar CamelCase.**

**Página: php.net**

CLASE #4

Componentes:

Pasar datos de componente padre al componente hijo:

Vimos en el concepto anterior que una aplicación con Angular está conformada por componentes. Siempre hay un componente padre y este puede tener uno o más componentes hijos, a su vez los componentes hijos pueden tener componentes hijos de ellos y así sucesivamente.

En este concepto veremos una técnica para pasar datos del componente padre al componente hijo. Podemos pasar datos de un componente en el momento en el que definimos una etiqueta de esta:

**<app-dado valor = ”3”></app-dado>**

En la clase DadoComponent podemos definir la sintaxis para definir una propiedad que llega como parámetro del componente padre:

**@Input() valor: string = “”;**

No debemos asignar ningún dato a la propiedad “valor” ya que se cargará cuando creemos un objeto de esta clase.

Para definir el decorador @Input() debemos importar la clase “Input”

Componentes:

Disparo de eventos del componente hijo al componente padre:

Ahora veremos cómo podemos capturar un evento en el

componente padre que eviten el componente hijo:

**<app-cronometro [inicio]=”15” (multiplo10)=”actualizar($event)”></app-cronometro>**

En este componente tenemos una propiedad llamada inicio que le enviamos un dato y capturamos un evento llamado “multiplo10” que emite el componente “AppCronometro”.

CLASE #5

Componentes:

Llamar a métodos del componente hijo desde el template del padre:

Otra forma de comunicarnos del componente padre al componente hijo es la posibilidad de llamar a métodos del componente hijo definiendo una variable en el template HTML donde declaramos el componente hijo.

Para llamar a los métodos el componente debe definir una variable en el template HTML:

**<app-dato #dado1></app-dado>**

**<button (click)=“dado1.tirar()”></button>**

Para definir una variable local le antecedemos el carácter “#” al nombre. Luego podemos llamar a métodos indicando el nombre de la variable y el método a llamar.

Componente:

Enlace de propiedades (Property Binding):

Hemos visto cómo enviar datos simples mediante propiedades (Property Binding). Ahora veremos en profundidad el concepto (Enlace de propiedades) estas son muy utilizadas si tenemos que enviar una estructura de datos compleja a un componente hijo vimos que si teníamos que enviar un dato de tipo de string a una propiedad de un componente lo indicamos mediante el valor a enviar:

**<app-dado valor = “3”></app-dado>**

En algunas situaciones podemos recuperar el dato del modelo mediante interpolación, pero el componente hijo lo recibe siempre como string:

**<app-dado valor = “{{valor}}”></app-dado>**

El problema se presenta si queremos enviar una estructura de datos más compleja como puede ser un arreglo, en este caso el concepto de Property Binding (Enlace de propiedades) nos resuelve el problema.

Si tenemos el componente “app-listadoarticulos” y tenemos que enviar el vector artículos debemos colocar el nombre de la propiedad entre “[]”.

TRABAJO:

* Crear una calculadora científica la cual me permita realizar todas las operaciones de la misma, prohibido utilizar métodos o funciones de js, diseñar la calculadora para un niño de 8 años.

CLASE #6

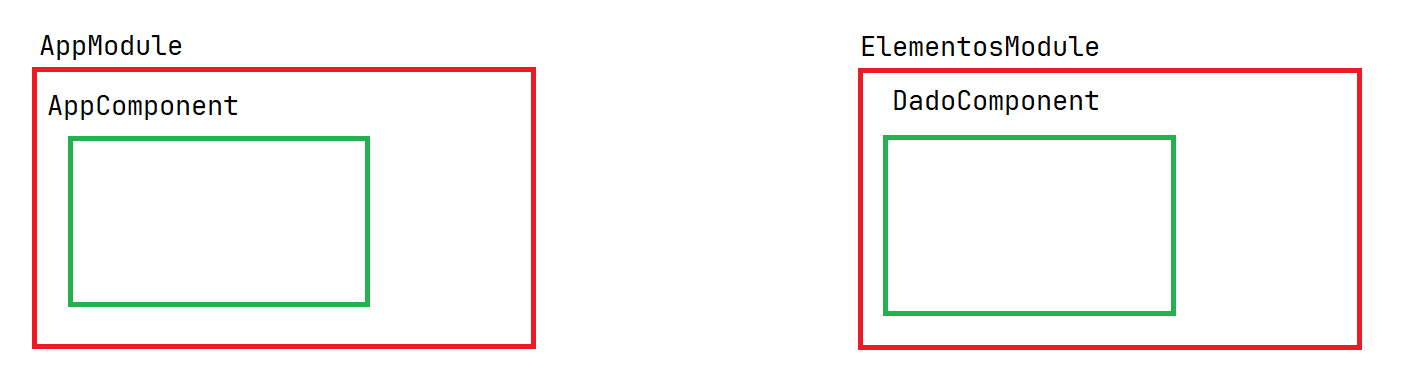
Módulo:

Creación y consumo:

Los módulos nos permiten organizar una aplicación compleja en Angular.

Cuando creamos una aplicación con Angular CLI se crea en forma automática el módulo “app.module” en el archivo “app.modulo.ts”.

Un módulo tiene por objetivo agrupar un conjunto de componentes relacionados entre sí.



Para crear un módulo se debe de abrir una terminal de comandos y ejecutar el siguiente comando:

* ng generate module elementos

Al ejecutar este comando se crea una carpeta llamada “elementos” dentro de la carpeta “app” y además un archivo llamado “elementos.module.ts”.

**Nota: se deberá exportar el componente creado y se deberá declarar a su vez el componente creado en el archivo o módulo base.**

Posteriormente, abrimos y modificamos el módulo que lo va a importar para poder consumirlo

Petición de un archivo JSON a un servidor:

Cuando tenemos que hacer peticiones de archivos JSON a un servidor en Angular disponemos de una clase llamada **httpClient** que nos facilita esta actividad.

Para hacer uso de la clase **httpClient** debemos importar el módulo “HttpClientModule”

**API: https://scratchya.com.ar/vue/datos.php**

Como primer paso, importamos el módulo “HttpClientModule” en el archivo “app.module.ts”. Deberemos importar el módulo en nuestro archivo “app.module.ts” y posteriormente importarlo en los “imports” del módulo.

Posteriormente a ello, toda la lógica de lectura de datos la implementaremos en el componente que por defecto ha creado Angular CLI.

**Nota: Como ejemplo la lógica se da implementada en este ejemplo en el archivo por defecto de Angular, pero, el uso de este módulo debe hacerse en cada uno de los componentes que lo requieran.**

Después de importar el módulo “HttpClient” deberemos importar nuevamente en el componente donde lo vayamos a usar.

A su vez, en Angular podemos definir una propiedad en los parámetros del constructor que se inyectan cuando se crea el componente:

**constructor (private http: HttpClient) { }**

Luego, la propiedad “http” que es de la clase “HttpClient” nos servirá para hacer la petición al servidor.

Como lógica podemos crear en el método “OnInit” el cual se ejecuta una vez que el template del componente este creado procederemos a recuperar del servidor los datos llamando al método **GET** de la propiedad “http”.

A partir del objeto que retorna el método **GET** llamamos al método “subscribe” y le pasamos una función anónima. Recibe como parámetros los datos recuperados del servidor

TRABAJO:

* Modificar el problema anterior, URL: rickandmortyapi.com/api/character, del api rickandmorty se deberá visualizar en pantalla los datos de cada personaje.