Clase 1:

Los comandos "ng" son todos de angular

Comando : ng v (vemos la versión de angular y también de node)

Angular cli versión de consola y otra cosa es la versión de angular y nos tira los paquete de angular

Cuando hacemos "ng new" crea un prpyecto angular con toda una estructura de archivos

El archivo principal es el "package.json" el cual es el que maneja todos los datos del proyecto y de las dependencias del proyecto, todos los paquetes que usamos se guardan ahi , dependencias va a ser todo lo que vamos a necesitar al poner en producción y devdependencias solo cuando esta en desarrollo aunque hay paquetes innecesarios

Tanto dependencias como devdependencias estan guardados en la carpeta node\_modules por lo tanto cuando traslade mi proyecto y en github mo la sube ya que es muy pesado , cuando tenemos cualquier proyecto con un package simplemente con hacer :

"npm i " nos crea la carpeta node\_modules (la cual no se guarda dentro de github)

El objeto Script es importante "ng" es el nombre del script y el valor del objeto es un comando de consola

Hay dos maneras de ejecutar esos comandos, una es con "npm" ejemplo "npm start" y otra es "npm run comando" cuando ponemos npm y el nombre del Script hay muchos que los reconoce y si quiero crear mi propio comando lo corremos con "npm run" los que son nativos los hacemos sin run

“npm start “ es lo mismo que escribir “ng serve” ya que start le dijimos en el package que es lo mismo

Pero no exactamente lo mismo ya que al hacer npm start va a haber uno montón de archivos disponibles dentro de mi aplicación y recursos

Siempre para iniciar un proyecto voy a poner "npm start"

Nuestro proyecto en si esta en la carpeta "src" ahi:

Tenemos el index.html que tiene un componente customizado "app-root" el cual es todo nuestro proyecto de angular

El archivo main.ts es el que inicia el proyecto

Después tres carpetas :

\*enviroment

\*assets

\*app (codigo a utilizar) , dentro tiene 5 archivos

esos archivos es con lo que esta conformado un componente

En el archivo app.component.ts es la LOGICA del componente el cual importa

Tiene también una anotation que le dice lo que corresponde a la clase y es un componente y lo utilizamos y le decimos que el "selector" va a ser el NOMBRE DE MI ETIQUETA COMO LE LLAMAREMOS EN HTML

también esta la propiedad "template" que es la vista del componente es decir el html y le ponemos una dirección a la vez que el atributo "styleUrls" que le ponemos una dirección de un archivo .css

Este componente tenemos el app.modules.ts crea un modilo que se hace solo y importa un appComponent

El modulo no es mas que un anotation donde tenemos declaraciones donde agregamos rodos los componentes para que puedan ser usados en declarations lo ponemos

en providers son servicios o clases

en bootstrap nos dice como ppr cual componente va a arrancar

Cuando quiero crear un componente ejecutamos el comando :

ng generate

que lo que hace es generar un componente o cualquiee cosa de angular

Ejemplo ponga : "ng generate component" para un componente o también modules,service

Luego le pongo el nombre de carpeta y el nombre del componente en este caso me aparecera la carpeta dentro de src/app/Components , el codigo quedaría así :

“ng generate component Components/saludar”

“npm start” va a levantar un servidor para que podamos ver en el navegador nuestra pagina ponemos “ localhost:4200” y ahí lo vemos nuestro servidor en la pagina

En appcomponent.html es donde comienza todo y lo que hacemos es incluir app-saludar dentro de ese html y ya podemos ver el contenido de saludar.component.html

Se agrega solo en app.module.ts y no lo tenemos que hacer a mano SOLO SI ES UN COMPONENTE PERO SI ES OTRA COSA DEBEMOS AGREGARLO A MANO

Hay una regla para escribir componentes:

-Debemos tener en mente que debe ser reutilizable

-que dentro del componente NO HAYA LOGICA es decir que la hagamos en otro lado

Dos tipos de componentes:

-tontos : tienen cero lógica

-contenedor : es el encargado de buscar los datos y pasarselo al tonto

En saludar.component.ts dentro de la clase creamos un atributo y le pasamos un valor , para "bindear"(pasar datos entre ts y html bidireccional)

Para leer ese atributo desde el html utilizamos "**interpolacion**" que es con **doble llave** y ponemos el nombre de la variable de typescript pero no podemos escribir código dentro de esta interpolacion.

\*En este caso este bindeo es desde typescript hasta html (one way)

Desde el ts podemos crear una variable "src" por ejemplo que sería la dirección de una imagen y tengo varias maneras de pasarle a html , un es interpolando (no la utilizamos) y la otra es con [ ] los corchetes ejemplo [src] = src es conveniente utilizar corchetes cuando le pasamos A LAS PROPIEDADES en este caso a "src" de la etiqueta "img" tambien a la propiedad width o height podemos modificarlo

\*En este caso este bindeo es desde typescript hasta html (one way)

El ultimo bindeo es de HTML A TS :

Lo podemos hacer a través de un BOTON o de in EVENTO.

En el caso de un botón para manejar un evento le sacamos el "on" y ponemos solo el nombre por ekemplo "onclick" pasa a "click" y le pasamos una función manejadora que creamos en ts .

También necesitamos indicar que trabajamos con un evento Y UTILIZAMOS LOS PARENTESIS "( )"

En el nombre de la PROPIEDAD .

\*En este caso este bindeo es desde typescript hasta html (one way)

Todo lo nativo ,angular los tiene pero los encapsula ejemplo "$event" o algun atributo .

TWO WAY DATA BIND

Hay listeners que si cambiamos algp en ts se refresca en html y alrevez tambien

Dentro de un input de tipo texto le indivamos que sea de "doble via" y para eso colocamos la sentencia [(ngModel)] ="variable " es decir que lo que ponga dentro del input se va a guardar en la variable "variable en typescript

Tira error , lo que hacemos es importar el modulo “FormsModule” en el módulo principal (app.module.ts), para solucionarlo

\*Hay un error dentro de un "form" ya que debemos agregarle un "**name**" a la propiedad de cada input SI O SI EN LOS FORM

\*Es ideal crear objetos para guardar los atributos dentro de este y poder acceder de manera rapida desde typescript

Directiva "\*ngIf" que va para el form y lleva un "asterisco" adelante y lo que hacemos es ponerle cualquier condición por ejemplo una bandera que cuando envié el botón dentro de la manejadora diga "true" etc

En el div hacemos otro ngif y validamos que si "enviado" es false que no se vea el form

Podemos hacer un else \*ngif=mostrar==true:else div2

Y utilizamos un ng-template para mostrar