EXAMEN TEORICO ANGULAR

1.- ¿Cómo mostrarías en un párrafo de un componente el valor de una propiedad de su clase llamada “edad”?

Con la Interpolación.

** La edad es de: {{ edad }} **

2.- Un componente padre muestra en su vista un hijo “cabecera” y otro hijo “item” tantas veces como elemnetos haya en un array de su clase asociada llamada “items”.¿Cómo serñia el codigo de esa vista?

```
<div>
<app-cabecera></app-cabecera>
<div *ngFor="let item of items">
        <app-item [data]="item"></app-item>
</div>
</div>
```

3.- Explica de forma detenida como se produce el arranque de una aplicación Angular.

Una vez que haces el ng serve -o , se inicializa el AppModule, en este es donde esta definido todos los componentes, servicios… Carga el componente raíz, que angular lo va a encontrar dentro del AppModule. Lo encuentra y ejecuta una serie de acciones. Despues de ejecutarse eso examina el html del componente y lo muestra en el navegador.

4.- Si en la vista de un componente aparece el siguiente código <hijo>Texto</hijo>, ¿qué características de Angular estoy usando?¿Cuál es su utilidad?

Estas utilizando la característica de componentes hijos.

La utilidad de los componentes hijos es dividir la interfaz de usuario en partes más pequeñas y reutilizables, lo que facilita la construcción, mantenimiento y escalabilidad de la aplicación.

5.-Escribe el código del controlador de un componente hijo que envíe a su padre , en forma de Output, el numero de veces que se ha pulsado un botón. Escribe la linea de código que necesita el padre para acceder a esos datos.

```
export class Hijo {
  contadorClicks: number = 0;
  @Output() contadorActualizado = new EventEmitter<number>();

  incrementarContador() {
    this.contadorClicks++;
    this.contadorActualizado.emit(this.contadorClicks);
  }
}
```

La línea de código que necesita el padre es:

```
<app-child (contadorActualizado)="actualizarContador($event)"></app-child>
```

6.-Crea una directiva en Angular llamada “MiDirectiva” que cambie el color de fondo del elemento al que se le aplica, utilizando un color personalizado proporcionado como parametro, y proporciona un ejemplo de uso en un componente.

```
export class MiDirectivaDirective implements OnInit {
  @Input('miDirectiva') colorPersonalizado: string;
  constructor(private el: ElementRef, private renderer: Renderer2) { }
  ngOnInit() {
    this.renderer.setStyle(this.el.nativeElement, 'background-color', this.colorPersonalizado);
  }
}
```

Ejemplo de uso en un componente:

```
<div miDirectiva="lightblue">
        Este es un ejemplo de componente que utiliza la directiva MiDirectiva.
</div>
```

7.- Implementa un formulario model-driven en Angular con campos “Nombre” (cadena de texto requerida) y “Edad” (numero mayor a 0).

```
export class Formulario implements OnInit{
        formulario: FormGroup;
        constructor(private fb: FormBuilder) {}
        ngOnInit(): void {
                this.formulario = this.fb.group({
                        nombre: [‘’, Validators.required],
                        edad: [null, Validators.min(1)]
                });
        }
}
```

8.- Crea un servicio “Contador” con un método que devuelva un observable donde recogeremos el numero de veces que se ha llamado a otro método del servicio llamado “contar”. Pon un ejemplo de uso de este servicio.

```
export class ContadorService {
  private contador: number = 0;
  private contadorSubject = new Subject<number>();

  constructor() { }

  contar(): void {
    this.contador++;
    this.contadorSubject.next(this.contador);
  }

  obtenerContador(): Observable<number> {
    return this.contadorSubject.asObservable();
  }
}
```

Ejemplo de uso:

```
export class EjemploComponenteComponent implements OnInit {
  contadorActual: number;

  constructor(private contadorService: ContadorService) { }

  ngOnInit(): void {
    this.contadorService.obtenerContador().subscribe(contador => {
      this.contadorActual = contador;
    });
  }
  incrementarContador() {
    this.contadorService.contar();
  }
}
```

9.- En Angular, ¿cómo se capturan y utilizan los parámetros de una ruta en un componente?

En primer lugar defines las rutas en el app-routing.ts :

```
const routes: Routes = [
        { path: 'producto/:id', component: DetalleProductoComponent }
];
```

Y luego se utilizan asi en un componente:

```
import { ActivatedRoute } from '@angular/router';
export class DetalleProductoComponent implements OnInit {
        productId: string;
        constructor(private route: ActivatedRoute) { }
        ngOnInit(): void {
                this.route.params.subscribe(params => {
                        this.productId = params['id'];
                });
        }
}
```

REPASO CÓDIGOS

INPUTS:

Componente Hijo:

TS:

```
@Input() nombre!: string
```

HTML:

```
<p>Hola soy {{nombre}}</p>
```

Componente Padre:

```
<app-hijo [nombre]=”nombre”></app-hijo>
```

OUTPUTS:

Componente Hijo:

TS:

```
@Output() dinero: any = new EventEmitter<any>()
mostrarDinero(){
    this.dinero.emit(24)
}
```

Componente Padre:

TS:

```
public dinero : number = 0

recogerDinero(dinero:number){
    this.dinero = dinero
}
```

HTML:

```
<app-hijo (dinero)=”recogerDinero($event)”></app-hijo>
```

PASAR DATOS POR PARÁMETROS

app-routing:

```
{path: ‘paginaDetalle/:nombre’, component PaginaDetalle}
```

formulario.ts:

```
enviarDatos(){
    this.router.navigate([‘paginaDetalle’,this.formulario.value[‘nombre’]])
```

formulario.html

```
<form [formGroup]="formulario" (ngSubmit)="enviarDatos()">
```

FORMULARIO REACTIVO

appModule:

En imports:

ReactiveFormsModule

formulario.ts:

```
export class FormularioComponent implements OnInit{
    formulario!: FormGroup;
    constructor(private fb: FormBuilder, private router: Router) {}

 ngOnInit(): void {
    this.formulario = this.fb.group({
    nombre: ['',Validators.required],
    });
}
```

html:

```
<form [formGroup]="formulario" (ngSubmit)="enviarDatos()">
<div>
   <label for="nombre">Nombre: </label>
   <input type="text" id="nombre" formControlName="nombre">
<div *ngIf="formulario.get('nombre')?.invalid && formulario.get('nombre')?.touched">
      <div *ngIf="formulario.get('nombre')?.errors">El nombre es obligatorio</div>
</div>
```

TUBERIAS

ts:

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
name: 'tuberiaMayusculas'
})
export class TuberiaMayusculasPipe implements PipeTransform {
transform(mensaje: String): String {
return mensaje.toUpperCase();
}
}
```

html:

```
{{mensaje | tuberiaMayusculas}}
```

GUARDS

ts:

```
import { inject } from '@angular/core';
import { CanActivateFn, Router } from '@angular/router';
import { LoginService } from '../servicios/servicioLogin/login.service';

export const loginGuardGuard: CanActivateFn = (route, state) => {
  if(inject(LoginService).estaLogueado()){
    return true;
  } else {
   inject(Router).navigate(["/login"]);
    return false;
  }
};
```

app-routing:

```
{path: 'formulario', component: FormularioComponent, canActivate: [loginGuardGuard]},
```

DIRECTIVAS

ts:

```
import { Directive, ElementRef, HostListener, Input, Renderer2 } from '@angular/core';
@Directive({
  selector: '[appDirectivaColor]'
})
export class DirectivaColorDirective {
  @Input('textoColor') color: any;

  constructor(private el: ElementRef, private renderer: Renderer2) { }

  @HostListener('mouseenter') ratonEncima() {
    this.renderer.setStyle(this.el.nativeElement, 'color', `${this.color}`);
  }

  @HostListener('mouseleave') ratonFuera() {
    this.renderer.removeStyle(this.el.nativeElement, 'color');
  }
}
```

html:

```
<h2 appDirectivaColor [textoColor]="{color: 'red'}">{{nombre}}</h2>
```

OBSERVABLES

descripcionService:

```
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ServicioDescripcionService {

  private descripcion$: BehaviorSubject<string> = new BehaviorSubject('')
  constructor() { }

  enviarDescripcion(descripcion: string) {
    let descrip = descripcion
    this.descripcion$.next(descrip)
  }

  getDescripcion$(): Observable<string> {
    return this.descripcion$.asObservable()
  }
}
```

hijo.ts:

```
enviarDescripcion(descripcion: string) {
    this.descripcionService.enviarDescripcion(descripcion)
  }
```

hijo.html:

```
<input type="text" #inputDescripcion>
<button (click)="enviarDescripcion(inputDescripcion.value)">Agregar Descripcion</button>
```

padre.ts:

```
public descripcion!: string

constructor(private descripcionService: ServicioDescripcionService){}

ngOnInit(): void {
    this.descripcionService.getDescripcion$().subscribe(descripcion => {
        this.descripcion = descripcion
    }
```

padre.html

```
    <p>La descripcion es: {{descripcion}}</p>
```