DESARROLLO DE APLICACIONES CON ANGULAR

Promises y Observables

ANGULAR

- Promises
  - Se utiliza para computación asíncrona.
  - En el constructor recibe como parámetro una función que a su vez recibe dos funciones: *resolve* y *reject*, que proporcionan las respuestas satisfactorias y erróneas (*resolve* y *reject* son nombres sugeridos).
  - La llamada a la promesa se gestiona mediante el encadenamiento de las funciones then, catch y finally.
  - Admiten un único flujo de datos.
  - No admiten cancelación.

ANGULAR

- Promises

```
const promise1 = new Promise((resolve, reject)=>{
    console.log("1. Inicio de Promise")
    setTimeout(()=>{
        console.log("3. Resolución de Promise");
        //resolve("Resuelto")
        reject("Error en Promise");
    },3000);
    console.log("2. Fin de Promise");
});
```

```
promise1.then((retorno)=>{
    console.log("4. Then:"+retorno);
}).catch((error)=>{
    console.log("4. Error:"+error);
}).finally(()=>{
    console.log("5. Finally");
})
```

ANGULAR

- Observables
  - Elementos asíncronos a los que es posible realizar una **suscripción**.
  - No es nativo JS=> Librería RxJS
  - Admiten reintentos.
  - Trabaja como un stream=>**el flujo de información puede ser infinito**.
  - Permite múltiples puntos de suscripción.
  - Genera un elemento de salida que se produce mediante el método **next**.
  - La suscripción se realiza mediante el método subscribe() que recibe la salida generada por next.
  - Se puede cancelar la suscripción con **unsubscribe**().

ANGULAR

- Función from de rxjs.
  - Convierte un array o una cadena de caracteres a Observable.
  - miObservable = from("Texto a procesar");

ANGULAR

- Funciones de los observables.
  - forEach→Permite recorrer el flujo
  - pipe→Permite aplicar operadores al flujo
  - subscribe→Permite suscribirse al flujo.

ANGULAR

- Función forEach.
  - Permite recorrer los elementos de un Observable.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
   this.miObservable.forEach(letra =>
     console.log("Letra:" + letra));
 }
```

ANGULAR

- Función pipe.
  - Permite aplicar operadores.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
   this.miObservable.pipe(
     filter(caracter => caracter!='o')
   ).forEach(c=>{
     console.log(c);
   });
 }
```

ANGULAR

- Función pipe.
  - Permite aplicar operadores.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
  this.miObservable.pipe(
    map(caracter => caracter.toUpperCase())
  ).forEach(c=>{
    console.log(c);
  });
}
```

ANGULAR

- Función pipe.
  - Permite aplicar operadores.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
  this.miObservable.pipe(
    map(caracter => this.convertir(caracter))
  ).forEach(c=>{
    console.log(c);
  });
 }

 convertir(letra:string){
  return letra.toUpperCase();
 }
```

ANGULAR

- Función pipe.
  - Permite aplicar operadores.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
   this.miObservable.pipe(
     filter(caracter => caracter!='o'),
     map(caracter => caracter.toUpperCase())
   ).forEach(c=>{
     console.log(c);
   });
 }
```

ANGULAR

- Función subscribe.

```
miObservable = from("Texto a procesar");

 ngOnInit():void {
   this.miObservable.subscribe(caracter => {
     console.log("Caracter:" + caracter);
   })
 }
```

ANGULAR

- Observables
  - Componentes:
    - Observable => Proporciona un stream de datos.
      - Métodos:
        - next(data) ➔ Proporciona el siguiente bloque de información del stream.
        - complete() ➔ Finaliza el stream.
    - Observer ➔ Utiliza los datos proporcionados por el Observable.
      - Se suscribe al Observable.
      - Métodos:
        - next(data) ➔ Recibe el siguiente bloque de datos.
        - error(error) ➔ Recibe el error.
        - complete() ➔ Finaliza el stream.

ANGULAR

- Ejemplo Observable:

```
import { Observable } from 'rxjs';

const observable = new Observable(subscriber => {
  subscriber.next(1);
  subscriber.next(2);
  subscriber.next(3);
  setTimeout(() => {
    subscriber.next(4);
    subscriber.complete();
  }, 1000);
});
```

ANGULAR

- Ejemplo Observer:

```
console.log('just before subscribe');
observable.subscribe({
  next(x) { console.log('got value ' + x); },
  error(err) { console.error('something wrong occurred: ' + err); },
  complete() { console.log('done'); }
});
console.log('just after subscribe');
```