FARAH ALHARASATANY

PROMISES VS. OBSERVABLES

https://farah-profile.vercel.app/

1 Definition

Promise

It represents a **single** future value that will resolve or reject **once**.

Observable

It represents a **stream of values** that can be emitted **over time**.

2

Nature

Promise

Single event (resolved or rejected).

Observable

Multiple events (can emit multiple values).

3 Lazy Execution

Promise

It's **not lazy** and executed **immediately upon creation.**

Observable

It's **lazy** and **only** executed **when subscribed to**.

4

Cancellation

Promise

Cannot be cancelled **once started**.

Observable

Can be cancelled by **unsubscribing** from it.

5

Composition

Promise

Can be composed using:

- .then()
- .catch()
- .finally()

Observable

Can be composed using powerful operators like:

- map.
- filter.
- merge.
- concat.

6

Error Handling

Promise

Errors are handled using:

- .catch().
- try...catch.

Observable

Errors are handled using:

- The error callback in the subscription.
- The operators like catchError.

7

Memory Management

Promise

No mechanism to stop or clean up once resolved.

Observable

Subscribers **can explicitly unsubscribe** to prevent memory leaks.

9

Execution Context

Promise

Always **asynchronous**.

Observable

Can be **synchronous** or **asynchronous**.

10

API Support

Promise	Built into JavaScript (Promise is a **native object**).
Observable	Requires a library like **RxJS**.

11

Use Case

Promise

Making a single HTTP request or reading a file.

Observable

Handling real-time data like WebSocket streams, event listeners, or multi-value data streams.

12

Example

Promise

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Observable

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

constructor(private http: HttpClient) {}

getData(): Observable<any> {
  return this.http.get('https://api.example.com/data');
}

// Usage
this.getData().subscribe({
  next: data => console.log(data),
  error: error => console.error('Error:', error),
});
```

PROMISE

- deal with **one asynchronous** event at a time.
- Emit a single value at a time.
- Are not lazy: execute immediately after creation.
- Are not cancellable.
- Don’t provide any operations.
- Push errors to the child promises.

OBSERVABLE

- handle a **sequence of asynchronous events** over a period of time.
- Emit multiple values over a period of time.
- Are lazy: they’re not executed until we subscribe to them using the subscribe() method.
- Have subscriptions that are cancellable using the unsubscribe() method, which stops the listener from receiving further values.
- Provide the map for forEach, filter, reduce, retry, and retryWhen operators.
- Deliver errors to the subscribers.