Both observables and promises help us work with asynchronous functionality in JavaScript. Promises deal with one asynchronous event at a time, while observables handle a sequence of asynchronous events over a period of time.

Let’s see the **difference between observable and promise**(observable vs promise)

|  |  |
| --- | --- |
| **Observables** | **Promises** |
| Emit multiple values over a period of time. | Emit a single value at a time. |
| Are lazy: they’re not executed until we subscribe to them using the subscribe() method. | Are not lazy: execute immediately after creation. |
| Have subscriptions that are cancellable using the unsubscribe() method, which stops the listener from receiving further values. | Are not cancellable. |
| Provide the map for forEach, filter, reduce, retry, and retryWhen operators. | Don’t provide any operations. |
| Deliver errors to the subscribers. | Push errors to the child promises. |

Now let’s see code snippets / examples of a few operations defined by observables and promises.

|  |  |  |
| --- | --- | --- |
| **Operations** | **Observables** | **Promises** |
| Creation | const obs = new Observable((observer) => {  observer.next(10);  }) ; | const promise = new Promise(() => {  resolve(10);  }); |
| Transform | Obs.pipe(map(value) => value \* 2); | promise.then((value) => value \* 2); |
| Subscribe | const sub = obs.subscribe((value) => {  console.log(value)  }); | promise.then((value) => {  console.log(value)  }); |
| Unsubscribe | sub.unsubscribe(); | Can’t unsubscribe |