آرگومان reject در promise يك نوع exception است و تابع settimeout قابليت هندل كردن آن را ندارد.ما ميتوانيم متد catch را بعد اتمام promise فراخواني كنيم و exception موجود را كه تابع settimeout توانايي هندل كردن آن را نداشت،هندل كنيم.

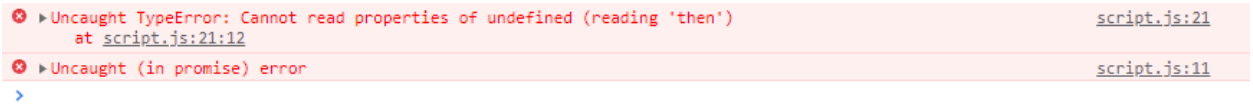
```
function timeoutPromiseReject(interval) {
  return new Promise((resolve, reject) => {
    setTimeout(function () {
      reject("error");
    }, interval);
  }).catch((e)=>{
    console.log(e)
  });
};
```

به اين صورت error موجود هندل ميشود و خروجي در كنسول براي اين قسمت از كد error است.

When you use the setTimeout, the promise constructor has already completed, synchronously, before you throw the exception... so it obviously cannot catch it after the fact. More precisely, exceptions are synchronous to their execution stack, and you've moved the exception to a separate execution stack by wrapping it in the timeout.

نتيجه نهايي

در ابتدا كد اين خطا رو مي دهد وقتي كه ما هنوز هيچ تغييري در آن نداديم



چون ما هيچ يك از promise ها را فراخواني نكرديم.
بايد از متود race استفاده كنيم كه به اين صورت است

The `Promise.race()` method returns a promise that fulfills or rejects as soon as one of the promises in an iterable fulfills or rejects, with the value or reason from that promise
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/race

اين متود باعث ميشود كه promise اي كه زودتر call ميشود اجرا شود و شرط اجرا شود

```
function timeTest() {
  let a = [timeoutPromiseResolve(5000),
  timeoutPromiseReject(2000),
  timeoutPromiseResolve(1000)];
  return Promise.race(a);
}
```

اين تغيير باعث ميشود فانكشن reject كه تايم كمتري دارد و زودتر call مي شود اجرا شود و وارد catch شود و خطاي كد برطرف ميشود