

Error handling - Asynchronous code

```
try {  
  setTimeout(function() {  
    random;  
  }, 1000)  
} catch (error) {  
  console.log('error', error)  
}
```

Since, setTimeout will take sometime this code would have already executed.

@codewithsimran

So in asynchronous code when something fails in this case, we don't even get an error

```
Promise.resolve('silent fail')  
  .then(res => {  
    throw new Error('fail')  
    return res  
  })  
  .then(res => {  
    ;  
  })
```

Even though we are getting throwing an error, we don't get any error!

Because it silently fails! Not good!

★★ We should always have a .catch with promises, otherwise we won't know what might happen.

★★ Even if you have nested promises you should write a catch statement for each of those.

```
Promise.resolve('fail')  
  .then(res => {
```

@codewithsimran

```
    Promise.resolve().then(() => {  
      throw new Error('Big fail')  
    }).catch(error => { ... }  
  })
```

```
  .then(  
    )
```

→ for the inner promise

```
  .catch(err => {
```

```
  })
```

→ for the outer promise

If we don't put a .catch for the inner promise, it'll still be handled by the outer catch (But in some environments) you'll get a warning.

Error handling with async await

```
const errorFunc = async function() {  
  try {  
    await Promise.reject('fail')  
  } catch(err) {  
    console.log(err)  
  }  
}
```

```
errorFunc();
```

@codewithSimran

Since async await let's you write code in a synchronous fashion (await line by line) we can use a try catch block to handle errors.