# Promise Error Handling

**Summary**: in this tutorial, you will learn how to deal with error handling in promises.

Suppose that you have a function called `getUserById()` that returns a Promise:

```
function getUserById(id) {
    return new Promise((resolve, reject) => {
        resolve({
            id: id,
            username: 'admin'
        });
    });
}
```

## Normal error

First, change the `getUserById()` function to throw an error outside the promise:

```
function getUserById(id) {
    if (typeof id !== 'number' || id <= 0) {
        throw new Error('Invalid id argument');
    }

    return new Promise((resolve, reject) => {
        resolve({
            id: id,
            username: 'admin'
        });
    });
}
```

Second, handle the promise by using both `then()` and `catch()` methods:

```
getUserById('a')
    .then(user => console.log(user.username))
    .catch(err => console.log(err));
```

The code throws an error:

```
Uncaught Error: Invalid id argument
```

When you raise an exception outside the promise, you must catch it with `try/catch`:

```
try {
    getUserById('a')
        .then(user => console.log(user.username))
        .catch(err => console.log(`Caught by .catch ${err}`));
} catch (error) {
    console.log(`Caught by try/catch ${error}`);
}
```

Output:

```
Caught by try/catch Error: Invalid id argument
```

## Errors inside the Promises

We change the `getUserById()` function to throw an error inside the promise:

```
let authorized = false;

function getUserById(id) {
    return new Promise((resolve, reject) => {
        if (!authorized) {
            throw new Error('Unauthorized access to the user data');
        }

        resolve({
```

```
            id: id,
            username: 'admin'
        });
    });
}
```

And consume the promise:

```
try {
    getUserById(10)
        .then(user => console.log(user.username))
        .catch(err => console.log(`Caught by .catch ${err}`));
} catch (error) {
    console.log(`Caught by try/catch ${error}`);
}
```

Output:

```
Caught by .catch Error: Unauthorized access to the user data
```

If you throw an error inside the promise, the `catch()` method will catch it, not the try/catch.

If you chain promises, the catch() method will catch errors that occur in any promise. For example:

```
promise1
    .then(promise2)
    .then(promise3)
    .then(promise4)
    .catch(err => console.log(err));
```

In this example, if any error in the promise1, promise2, or promise4, the `catch()` method will handle it.

## Calling reject() function

Throwing an error has the same effect as calling the `reject()` as illustrated in the following example:

```
let authorized = false;

function getUserById(id) {
    return new Promise((resolve, reject) => {
        if (!authorized) {
            reject('Unauthorized access to the user data');
        }

        resolve({
            id: id,
            username: 'admin'
        });
    });
}

try {
    getUserById(10)
        .then(user => console.log(user.username))
        .catch(err => console.log(`Caught by .catch ${err}`));
} catch (error) {
    console.log(`Caught by try/catch ${error}`);
}
```

In this example, instead of throwing an error inside the promise, we called the `reject()` explicitly. The `catch()` method also handles the error in this case.

## Missing the catch() method

The following example does not provide the `catch()` method to handle the error inside the promise. It will cause a runtime error and terminate the program:

```
function getUserById(id) {
    return new Promise((resolve, reject) => {
        if (!authorized) {
```

```
            reject('Unauthorized access to the user data');
        }
        resolve({
            id: id,
            username: 'admin'
        });
    });
}

try {
    getUserById(10)
        .then(user => console.log(user.username));
    // the following code will not execute
    console.log('next');

} catch (error) {
    console.log(`Caught by try/catch ${error}`);
}
```

Output:

```
Uncaught (in promise) Unauthorized access to the user data
```

If the promise is resolved, you can omit the `catch()` method. In the future, a potential error may cause the program to stop unexpectedly.

## Summary

- Inside the promise, the `catch()` method will catch the error caused by the `throw` statement and `reject()`.
- If an error occurs and you don't have the `catch()` method, the JavaScript engine issues a runtime error and stops the program.