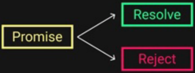
ReactOverflow

JavaScript JS PROMISE Explained

Promise

is a JavaScript object that allows you to run non-blocking (aka async) code. It produces a value when the async operation completes successfully or produces an error if it doesn't complete.



To learn how Promise work let's look at this example

```javascript
const dataPromise = new Promise((resolve, reject) => {
  setTimeout(() => resolve('Then this'), 1000);
});

dataPromise.then(data => {
  console.log(data);
});

dataPromise.catch(error => console.log(error));

console.log('This runs first');

// *Output*
// This runs first
// *after a second*
// Then this
```

When we create a new Promise it will run the given callback parallel to the rest of the code in the script.

The callback will gets two functions as arguments: resolve and reject, when the callback succeed in it's task then it call resolve else call reject

We then can run specific code depending whether the promise succeed or not using `.then` and `.catch`

The most common use case of Promise is fetching data from a REST API

```
fetch('http://example.com/movies.json')
  // Handle success
  .then(response => response.json())
  // Handle Error
  .catch(error => console.log(error));
```

The fetch API from JavaScript return a Promise because a HTTP request can take a lot of time and then we can call functions depending on weather the request succeed or failed

Here is a simple Cheatsheet for Promise

```
const promise = new Promise((resolve, reject) => {
  // Time consuming Async (non-blocking) code
  // On Success (data is optional result)
  resolve(data)
  // On fail (error is optional reason)
  reject(error)
});

promise.then((data) => {
  // On success
});

promise.catch((error) => {
  // On fail (for error handling)
});

promise.finally(() => {
  // Runs for both resolve and reject
})
```

The return value of `.then`, `.catch` and `.finally` also a returns Promise instance so you can chain them

```
new Promise(function(resolve, reject) {

  setTimeout(() => resolve(1), 1000); // (*)

}).then(function(result) { // (**)

  alert(result); // 1
  return result * 2;

}).then(function(result) { // (***)

  alert(result); // 2
  return result * 2;

}).then(function(result) {

  alert(result); // 4
  return result * 2;

});
```

.then can also return a newly created Promise instance

```
new Promise(function(resolve, reject) {

  setTimeout(() => resolve(1), 1000);

}).then(function(result) {

  alert(result); // 1

  return new Promise((resolve, reject) => { // (*)
    setTimeout(() => resolve(result * 2), 1000);
  });

}).then(function(result) { // (**)

  alert(result); // 2

  return new Promise((resolve, reject) => {
    setTimeout(() => resolve(result * 2), 1000);
  });

}).then(function(result) {

  alert(result); // 4

});
```

Did you learn something new?

Let me know in the comments