JavaScript Promises

Back to JS page

Table of Contents

- JavaScript Promises
  - Creating a Promise
    - Example 1
  - The then() Method
    - Example 2
  - The catch() Method
    - Example 3
  - The finally() Method
    - Example 4
  - Promise Chaining
    - Example 5
  - Document
  - Reference

Why Promises?

Promises provide a cleaner way to handle asynchronous operations compared to callbacks. A Promise is an object representing the eventual completion or failure of an asynchronous operation.

Promise states:

- **pending**: Initial state, neither fulfilled nor rejected
- **fulfilled**: Operation completed successfully
- **rejected**: Operation failed

Creating a Promise

A Promise is created using the `new Promise()` constructor which takes a function with `resolve` and `reject` parameters:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Promises</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Creating a Promise</h4>
<p id="demo"></p>

<script>
const myPromise = new Promise(function(resolve, reject) {
  let x = 0;
  if (x === 0) {
    resolve("OK");
  } else {
    reject("Error");
  }
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = "Promise resolved: " + value;
});
</script>

</body>
</html>
```

Example 1

Result View Example

The then() Method

The `then()` method takes two arguments: a callback for success and another for failure. It is the primary way to handle resolved promises:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Promises</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The then() Method</h4>
<p id="demo"></p>

<script>
function fetchData() {
  return new Promise(function(resolve) {
    setTimeout(function() {
      resolve("Data loaded successfully!");
    }, 2000);
  });
}

fetchData().then(function(value) {
  document.getElementById("demo").innerHTML = value;
});

document.getElementById("demo").innerHTML = "Loading...";
</script>

</body>
</html>
```

Example 2

Result View Example

The catch() Method

The `catch()` method provides a callback for when a promise is rejected:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Promises</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The catch() Method</h4>
<p id="demo"></p>

<script>
const myPromise = new Promise(function(resolve, reject) {
  let success = false;
  if (success) {
    resolve("Operation succeeded");
  } else {
    reject("Operation failed");
  }
});

myPromise
  .then(function(value) {
    document.getElementById("demo").innerHTML = value;
  })
  .catch(function(error) {
    document.getElementById("demo").innerHTML = "Error: " + error;
  });
</script>

</body>
</html>
```

Example 3

Result View Example

The finally() Method

The `finally()` method executes regardless of whether the promise was fulfilled or rejected:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Promises</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The finally() Method</h4>
<p id="demo"></p>

<script>
const myPromise = new Promise(function(resolve) {
  setTimeout(function() {
    resolve("Promise completed");
  }, 1500);
});

myPromise
  .then(function(value) {
    document.getElementById("demo").innerHTML = value + "<br>";
  })
  .finally(function() {
    document.getElementById("demo").innerHTML += "Finally: cleanup done!";
  });
</script>

</body>
</html>
```

Example 4

Result View Example

Promise Chaining

Promises can be chained to handle sequential asynchronous operations:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Promises</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Promise Chaining</h4>
<p id="demo"></p>

<script>
function step1() {
  return new Promise(function(resolve) {
    setTimeout(function() {
      document.getElementById("demo").innerHTML += "Step 1 complete<br>";
      resolve(10);
    }, 1000);
  });
}

function step2(number) {
  return new Promise(function(resolve) {
    setTimeout(function() {
      let result = number * 2;
      document.getElementById("demo").innerHTML += "Step 2: " + number + " * 2 = " + result + "<
      resolve(result);
    }, 1000);
  });
}

function step3(number) {
  return new Promise(function(resolve) {
    setTimeout(function() {
      let result = number + 5;
      document.getElementById("demo").innerHTML += "Step 3: " + number + " + 5 = " + result + "<
      resolve(result);
    }, 1000);
  });
}

// Chain the promises
step1()
  .then(function(result1) { return step2(result1); })
  .then(function(result2) { return step3(result2); })
  .then(function(result3) {
    document.getElementById("demo").innerHTML += "Final result: " + result3;
  });
</script>
```

```
</body>
</html>
```

Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Promises