JavaScript Promises Explained with Examples

In JavaScript, a **Promise** is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value. Promises are very useful for handling operations like API requests or file reads, where you may not know when the data will be available.

A **Promise** has three states:

1. **Pending**: The initial state. The operation is not yet completed.
2. **Fulfilled**: The operation has completed successfully, and a value is available.
3. **Rejected**: The operation has failed, and a reason (typically an error) is available.

Creating a Promise

You create a new promise using the `Promise` constructor, which takes a function with two parameters: `resolve` (called when the task is successful) and `reject` (called when the task fails).

Here's an example:

```
const myPromise = new Promise((resolve, reject) => {
  let success = true;

  if (success) {
        resolve("Task completed successfully!");
  } else {
        reject("Something went wrong...");
  }
});
```

Using `.then()` and `.catch()`

To handle the result of a promise, you use `.then()` for a successful outcome and `.catch()` for errors.

Example:

```
myPromise

  .then((message) => {

        console.log(message); // Output: "Task completed successfully!"

  })

  .catch((error) => {

        console.error(error); // Output if failed: "Something went wrong..."

  });
```

Real-World Example: Fetching Data from an API

Promises are often used to handle HTTP requests. Let's see how we can use a promise to fetch data using the `fetch` API.

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then((response) => {
            if (!response.ok) {
            throw new Error("Network response was not ok");
            }
            return response.json();
  })
  .then((data) => {
            console.log(data);
  })
  .catch((error) => {
            console.error("There has been a problem with your fetch operation:", error);
  });
```

In this example:

- The `fetch()` function returns a promise that resolves with the `Response` object representing the response to the request.
- The `.then()` chain is used to handle the successful resolution of the request.
- If there is an error (e.g., the server cannot be reached), the `.catch()` function handles it.

Example of `Promise.all`

If you need to wait for multiple promises to complete, you can use `Promise.all()`, which takes an array of promises and resolves once all of them are fulfilled (or rejects if any one of them fails).

```
const promise1 = Promise.resolve(10);

const promise2 = Promise.resolve(20);

const promise3 = new Promise((resolve) => setTimeout(resolve, 1000, 30));

Promise.all([promise1, promise2, promise3])

  .then((values) => {

      console.log(values); // Output: [10, 20, 30]

  })

  .catch((error) => {

      console.error("One of the promises failed:", error);

  });
```

Summary

- Promises are a way to handle asynchronous operations.
- A promise is in one of three states: pending, fulfilled, or rejected.
- You use `.then()` to handle a fulfilled promise and `.catch()` to handle errors.
- `Promise.all()` is useful when you want to wait for multiple promises to complete.

Promises are an essential part of modern JavaScript, making it easier to work with asynchronous code in a readable and organized way.

Create and Run a Pure JavaScript Project

Creating a simple JavaScript project from scratch is a great way to get hands-on experience with coding in JavaScript. Here's a step-by-step guide on how to set up a basic project, write some JavaScript code, and run it.

Step 1: Set Up Your Project Folder

1. **Create a New Folder**: On your computer, create a new folder for your project. You can name it something like `my-js-project`.
2. **Open the Folder in a Code Editor**: Open the folder using a code editor like Visual Studio Code (VS Code). VS Code is popular for JavaScript development, but you can use any editor of your choice.

Step 2: Create HTML and JavaScript Files

1. **Create an HTML File**: Inside your project folder, create a new file named `index.html`. This file will be the entry point for your web project.
2. **Write the Basic HTML Structure**: Add the following HTML code to your `index.html` file:

```html
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>My JavaScript Project</title>
</head>
<body>
	<h1>Hello, JavaScript!</h1>
	<script src="script.js"></script>
</body>
</html>
```

1. **Create a JavaScript File**: Create a new file named `script.js` in the same project folder. This file will contain your JavaScript code.

Step 3: Write JavaScript Code

1. **Open `script.js`**: Write some JavaScript code to test if everything is working properly. For example:

```
console.log("Hello, World!");


document.addEventListener("DOMContentLoaded", () => {
        alert("Welcome to My JavaScript Project!");
});
```

This code will log "Hello, World!" in the browser's console and display an alert when the page is loaded.

Step 4: Open and Run the Project

1. **Open the HTML File in a Browser**: Right-click on `index.html` in your code editor or file manager and select "Open with" followed by your web browser (e.g., Chrome, Firefox).
2. **Check the Output**:
   - You should see the page with the heading "Hello, JavaScript!".
   - An alert box should pop up with the message "Welcome to My JavaScript Project!".
   - Open the browser console (right-click and select "Inspect" > "Console" or press `F12`) to see the "Hello, World!" message.

Step 5: Make Changes and Experiment

1. **Modify `script.js`**: Add more JavaScript code to practice different functionalities.
    - For example, add a button to the HTML and use JavaScript to handle a click event:

```
<button id="myButton">Click Me!</button>
```

Then, in `script.js`, add the code to handle the button click:

```
const button = document.getElementById("myButton");

button.addEventListener("click", () => {

    console.log("Button clicked!");

    alert("You clicked the button!");

});
```