ES6 Async/Await, Promises, and Related Concepts: A Step-by-Step Guide

Created by	A Arun kumaar Krishna
Created time	@September 6, 2024 4:28 PM
Tags	Javascript es6

1. Promises

A Promise is an object representing the eventual completion or failure of an asynchronous operation.

```
const myPromise = new Promise((resolve, reject) => {
  // Asynchronous operation
  if (/* operation successful */) {
    resolve(result);
  } else {
    reject(error);
  }
});
```

2. Then, Catch, and Finally

Promises have methods to handle their resolution or rejection:

```
myPromise
  .then(result => {
    // Handle successful result
  })
  .catch(error => {
```

```
    // Handle error
  })
  .finally(() => {
    // Always executed
  });
```

3. Async/Await

Async/await is syntactic sugar for working with Promises:

```
async function myAsyncFunction() {
  try {
    const result = await myPromise;
    // Handle result
  } catch (error) {
    // Handle error
  }
}
```

4. Multiple Async/Await

You can use multiple await statements in an async function:

```
async function fetchMultipleData() {
  const post = await fetch('https://jsonplaceholder.typicode.co
  const comments = await fetch('https://jsonplaceholder.typicod
  return { post, comments };
}
```

5. Multiple Async/Await with try-catch and finally

You can use try-catch-finally with multiple await statements for better error handling:

```
async function fetchMultipleDataWithErrorHandling() {
  try {
    const post = await fetch('https://jsonplaceholder.typicode.c
    const comments = await fetch('https://jsonplaceholder.typic
    return { post, comments };
  } catch (error) {
    console.error('An error occurred:', error);
    throw error; // Re-throw the error if needed
  } finally {
    console.log('Fetch operation completed');
    // Cleanup code, if any
  }
}
```

6. Multiple Then

You can chain multiple then methods:

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => response.json())
  .then(post => fetch(`https://jsonplaceholder.typicode.com/comm
  .then(response => response.json())
  .then(comments => {
    // Handle post and comments
  })
  .catch(error => {
    // Handle any errors in the chain
  });
```

7. Promise.all

Promise.all allows you to run multiple Promises concurrently:

```
const postPromise = fetch('https://jsonplaceholder.typicode.com/
const commentsPromise = fetch('https://jsonplaceholder.typicode

Promise.all([postPromise, commentsPromise])
  .then(([post, comments]) => {
    // Handle both post and comments
  })
  .catch(error => {
    // If any promise rejects, this catch handles it
  });
```

8. Promise.allSettled

Promise.allSettled waits for all promises to settle, regardless of whether they fulfill or reject:

```
const promises = [
  fetch('https://jsonplaceholder.typicode.com/posts/1').then(res
  fetch('https://jsonplaceholder.typicode.com/nonexistent').then
];

Promise.allSettled(promises)
  .then(results => {
    results.forEach(result => {
      if (result.status === 'fulfilled') {
        console.log('Fulfilled:', result.value);
      } else {
        console.log('Rejected:', result.reason);
      }
    });
  });
```

This guide provides a step-by-step approach to understanding and using ES6 async/await, Promises, and related concepts in JavaScript.