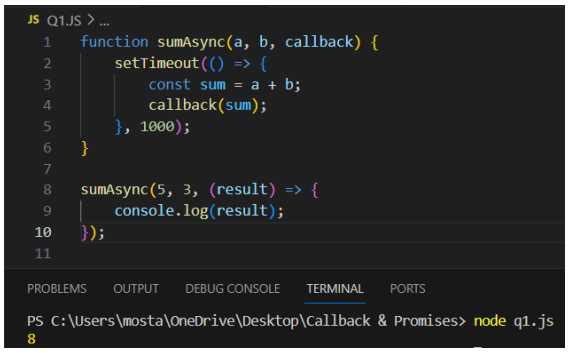
Assignment-5 Solution
Callback & Promises

Assignment 1

Write a function sumAsync that takes two numbers as arguments and uses a callback to return their sum after a delay of 1 second.

Solution: It prints 8 after 1 seconds.



Assignment 2

Create a function getData that returns a Promise. The Promise should resolve after 2 seconds with a message "Data fetched successfully."

Solution:

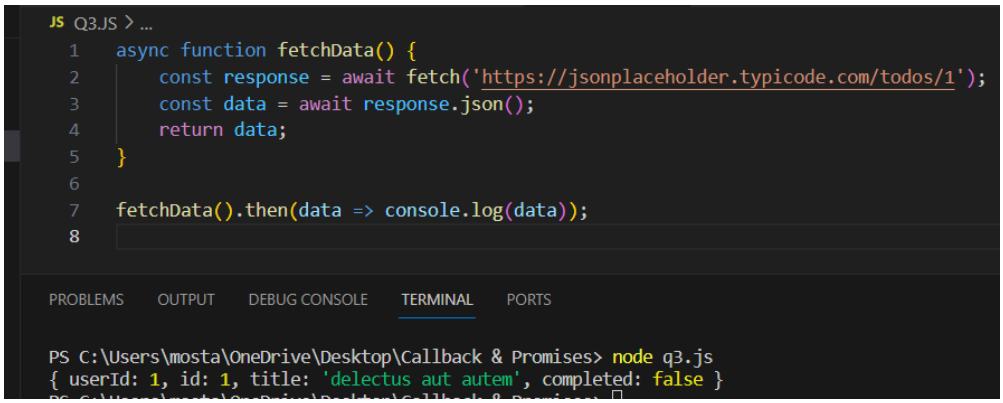
Assignment 3

Write an asynchronous function fetchData that uses the Fetch API to retrieve data from a given URL and returns the parsed JSON response.

API to be used

https://jsonplaceholder.typicode.com/todos/1

Solution:



Assignment 4

Write an asynchronous function fetchData that uses the Fetch API to retrieve data from a given URL (https://jsonplaceholder.typicode.com/todos/1) and returns the parsed JSON response.

Solution:

```
JS Q4.JS > fetchData
async function fetchData() {
    const url = 'https://jsonplaceholder.typicode.com/todos/1';
    try {
        const response = await fetch(url);
        if (!response.ok) {
            throw new Error('Network response was not ok ' + response.statusText);
        }
        const data = await response.json();
        return data;
    } catch (error) {
        console.error('There has been a problem with your fetch operation: ', error);
    }
}

fetchData().then(data => console.log(data));

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises> node q4.js
{ userId: 1, id: 1, title: 'delectus aut autem', completed: false }
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises>
```

Assignment 5

Implement a function multiplyWithCallback that takes an array of numbers and a callback function. The function should multiply each element of the array by 2 and pass the result to the callback.

Solution:

```
JS Q5.JS > ...
function multiplyWithCallback(numbers, callback) {
    const result = numbers.map(number => number * 2);
    callback(result);
}

multiplyWithCallback([1, 2, 3], (result) => {
    console.log(result);
});

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises> node Q5.JS
[ 2, 4, 6 ]
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises>
```

Assignment 6

Create a function fetchUserDataAndPosts that takes a user ID and fetches the user details and their posts using separate API calls. Use promise chaining to ensure the posts are retrieved only after the user details are fetched. Return an object with user details and posts.

API to be used

For user: https://jsonplaceholder.typicode.com/users/${userId}

For post: https://jsonplaceholder.typicode.com/posts?userId=${userId}

Solution:

```
JS Q6.JS > ...
function fetchUserDataAndPosts(userId) {
    const userUrl = `https://jsonplaceholder.typicode.com/users/${userId}`;
    const postsUrl = `https://jsonplaceholder.typicode.com/posts?userId=${userId}`;

    return new Promise((resolve, reject) => {
        fetch(userUrl)
            .then(response => response.json())
            .then(user => {
                fetch(postsUrl)
                    .then(response => response.json())
                    .then(posts => {
                        resolve({
                            user,
                            posts
                        });
                    })
                    .catch(error => reject(error));
            })
            .catch(error => reject(error));
    });
}

// Example usage:
fetchUserDataAndPosts(1)
    .then(data => console.log(data))
    .catch(error => console.error('Error:', error));
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises> node q6.js
{
  user: {
    id: 1,
    name: 'Leanne Graham',
    username: 'Bret',
    email: 'Sincere@april.biz',
    address: {
      street: 'Kulas Light',
      suite: 'Apt. 556',
      city: 'Gwenborough',
      zipcode: '92998-3874',
      geo: [Object]
    },
    phone: '1-770-736-8031 x56442',
    website: 'hildegard.org',
    company: {
      name: 'Romaguera-Crona',
      catchPhrase: 'Multi-layered client-server neural-net',
      bs: 'harness real-time e-markets'
    }
  },
  posts: [
    {
      userId: 1,
      id: 1,
      title: 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit',
      body: 'quia et suscipit\n' +
        'suscipit recusandae consequuntur expedita et cum\n' +
        'reprehenderit molestiae ut ut quas totam\n' +
        'nostrum rerum est autem sunt rem eveniet architecto'
    },
```

```
    {
      userId: 1,
      id: 2,
      title: 'qui est esse',
      body: 'est rerum tempore vitae\n' +
        'sequi sint nihil reprehenderit dolor beatae ea dolores neque\n' +
        'fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\n' +
        'qui aperiam non debitis possimus qui neque nisi nulla'
    },
    {
      userId: 1,
      id: 3,
      title: 'ea molestias quasi exercitationem repellat qui ipsa sit aut',
      body: 'et iusto sed quo iure\n' +
        'voluptatem occaecati omnis eligendi aut ad\n' +
        'voluptatem doloribus vel accusantium quis pariatur\n' +
        'molestiae porro eius odio et labore et velit aut'
    },
    {
      userId: 1,
      id: 4,
      title: 'eum et est occaecati',
      body: 'ullam et saepe reiciendis voluptatem adipisci\n' +
        'sit amet autem assumenda provident rerum culpa\n' +
        'quis hic commodi nesciunt rem tenetur doloremque ipsam iure\n' +
        'quis sunt voluptatem rerum illo velit'
    },
    {
      userId: 1,
      id: 5,
      title: 'nesciunt quas odio',
      body: 'repudiandae veniam quaerat sunt sed\n' +
        'alias aut fugiat sit autem sed est\n' +
        'voluptatem omnis possimus esse voluptatibus quis\n' +
        'est aut tenetur dolor neque'
    },
```

```
    {
      userId: 1,
      id: 6,
      title: 'dolorem eum magni eos aperiam quia',
      body: 'ut aspernatur corporis harum nihil quis provident sequi\n' +
        'mollitia nobis aliquid molestiae\n' +
        'perspiciatis et ea nemo ab reprehenderit accusantium quas\n' +
        'voluptate dolores velit et doloremque molestiae'
    },
    {
      userId: 1,
      id: 7,
      title: 'magnam facilis autem',
      body: 'dolore placeat quibusdam ea quo vitae\n' +
        'magni quis enim qui quis quo nemo aut saepe\n' +
        'quidem repellat excepturi ut quia\n' +
        'sunt ut sequi eos ea sed quas'
    },
    {
      userId: 1,
      id: 8,
      title: 'dolorem dolore est ipsam',
      body: 'dignissimos aperiam dolorem qui eum\n' +
        'facilis quibusdam animi sint suscipit qui sint possimus cum\n' +
        'quaerat magni maiores excepturi\n' +
        'ipsam ut commodi dolor voluptatum modi aut vitae'
    },
```

Assignment 7

Write a function fetchMultipleData that takes an array of URLs and uses Promise.all() to fetch data from all the URLs concurrently. Return an array of responses.

API to be used

Change todo id for each API call

https://jsonplaceholder.typicode.com/todos/1

Solution:

JS Q7.JS > ...

```js
async function fetchMultipleData(urls) {
    try {
        const responses = await Promise.all(urls.map(url => fetch(url)));
        const data = await Promise.all(responses.map(response => response.json()));
        return data;
    } catch (error) {
        console.error("Error:", error);
    }
}

const urls = [
    'https://jsonplaceholder.typicode.com/todos/1',
    'https://jsonplaceholder.typicode.com/todos/2',
    // Add more URLs as needed
];

fetchMultipleData(urls).then(data => console.log(data));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises> node Q7.JS
[
  { userId: 1, id: 1, title: 'delectus aut autem', completed: false },
  {
    userId: 1,
    id: 2,
    title: 'quis ut nam facilis et officia qui',
    completed: false
  }
]
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises>
```

Assignment 8

Create a function racePromises that takes an array of promises and returns the result of the first promise that resolves or rejects. Use Promise.race() to implement this.

Solution:

JS Q8.JS > ...

```js
function racePromises(promises) {
    return Promise.race(promises);
}

let p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("p1"), 1000);
});

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => reject("p2"), 500);
});

let p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("p3"), 1500);
});

racePromises([p1, p2, p3])
  .then((value) => console.log("Resolved:", value))
  .catch((reason) => console.log("Rejected:", reason));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises> node q8.js
Rejected: p2
PS C:\Users\mosta\OneDrive\Desktop\Callback & Promises>
```