**Assignment 1: sumAsync Function**

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

javascript

```
function sumAsync(a, b, callback) {
  setTimeout(() => {
    const sum = a + b;
    callback(sum);
  }, 1000); // 1 second delay
}
// Example usage:
sumAsync(3, 5, (result) => {
  console.log("Sum:", result); // Sum: 8 after 1 second
});
```

**Assignment 2: getData Function with Promise**

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

```
function getData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("Data fetched successfully.");
    }, 2000); // 2 seconds delay
  });
}

// Example usage:
getData().then((message) => {
  console.log(message); // "Data fetched successfully." after 2 seconds
});
```

**Assignment 3: fetchData Function using Fetch API**

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

```
async function fetchData(url) {
  const response = await fetch(url);
  const data = await response.json();
  return data;
```

```
}

// Example usage:
fetchData('https://jsonplaceholder.typicode.com/todos/1')
  .then((data) => console.log(data));
```

**Assignment 4: Another fetchData using Fetch API**

```
async function fetchData(url) {
  try {
    const response = await fetch(url);
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error fetching data:', error);
  }
}

// Example usage:
fetchData('https://jsonplaceholder.typicode.com/todos/1')
  .then((data) => console.log(data));
```

**Assignment 5: multiplyWithCallback Function**

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.

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

// Example usage:
multiplyWithCallback([1, 2, 3, 4], (result) => {
  console.log(result); // [2, 4, 6, 8]
});
```

**Assignment 6: fetchUserDataAndPosts using Promise Chaining**

Create a function fetchUserDataAndPosts that takes a user ID and fetches the user details and their posts using separate API calls.

```
function fetchUserDataAndPosts(userId) {
  const userUrl = `https://jsonplaceholder.typicode.com/users/${userId}`;
  const postsUrl = `https://jsonplaceholder.typicode.com/posts?userId=${userId}`;
    return fetch(userUrl)
    .then(response => response.json())
    .then(user => {
      return fetch(postsUrl)
        .then(response => response.json())
        .then(posts => {
          return { user, posts };
        });
    });
}

// Example usage:
fetchUserDataAndPosts(1).then(data => console.log(data));
```

**Assignment 7: fetchMultipleData with Promise.all()**

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

javascript

Copy code

```
function fetchMultipleData(urls) {
  const promises = urls.map((url) => fetch(url).then((response) => response.json()));
  return Promise.all(promises);
}
// Example usage:
const urls = [
  'https://jsonplaceholder.typicode.com/todos/1',
  'https://jsonplaceholder.typicode.com/todos/2',
  'https://jsonplaceholder.typicode.com/todos/3'
];
```

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

**Assignment 8: racePromises using Promise.race()**

Create a function racePromises that takes an array of promises and returns the result of the first promise that resolves or rejects.

```
function racePromises(promises) {
  return Promise.race(promises);
}
// Example usage:
const promise1 = new Promise((resolve) => setTimeout(() => resolve('First!'), 1000));
const promise2 = new Promise((resolve) => setTimeout(() => resolve('Second!'), 2000));

racePromises([promise1, promise2]).then((result) => console.log(result)); // "First!" after 1 second
```