

Task 1: Basic JavaScript Review

Objective:

Write simple programs using JavaScript ES6 features such as arrow functions, promises, and async/await. Ensure the code runs without errors and uses ES6 syntax.

Prerequisites:

- Basic understanding of JavaScript.
- Node.js installation.

Concepts:

1. Arrow Functions:

- Arrow functions provide a concise syntax for writing function expressions.
- They do not have their own `this` context.

Example:

```
// arrowFunctionsExample.js
// Example of an arrow function to square a number
const square = (n) => n * n;
console.log(`Square: ${square(4)}`); // Output: Square: 16

// Example of an arrow function to check if a number is even
const isEven = (num) => num % 2 === 0;
console.log(isEven(5)); // Output: false
```

2. Promises:

- Promises represent the eventual completion (or failure) of an asynchronous operation and its resulting value.
- They provide a method to handle asynchronous operations in a more readable way.

Example:

```
// promisesExample.js
// Example of a promise that simulates a successful database query
const queryDatabase = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('Query successful');
  }, 2000);
});

queryDatabase
  .then(message => {
    console.log(message); // Output after 2 seconds: Query successful
  })
  .catch(error => {
    console.error('Error:', error);
  });
```

3. Async/Await:

- async functions enable the use of `await` to pause the execution until the promise is resolved.
- They make asynchronous code look and behave more like synchronous code.

Example:

```
// asyncAwaitExample.js
// Example of async/await to fetch user data
const fetchUserData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve({ name: 'Alice', age: 30 });
    }, 1500);
  });
};

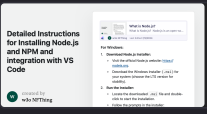
const getUserData = async () => {
  try {
    const data = await fetchUserData();
    console.log('User Data:', data); // Output after 1.5 seconds: User Data:
    { name: 'Alice', age: 30 }
  } catch (error) {
    console.error('Error:', error);
  }
};

getUserData();
```

Setup:

1. Install Node.js:

Ensure Node.js is installed on your machine. You can access the instructions here:

-  **Detailed Instructions for Installing Node.js and NPM and integration with...**
For Windows: — Download Node.js Installer: — Visit the official Node.js website: <http://nodejs.org/en/download/>



w3o NThing

Last Edited 7/3/2024

Tasks:

1. Arrow Functions:

- **Task:**

Create a file named `arrowFunctions.js` and write your own code that:

- Defines an arrow function named `calculateArea` that takes the radius of a circle as a parameter and returns the area of the circle. Use the formula ($\pi \times r^2$). Log the result of calling this function with a sample radius.

- Defines an arrow function named `convertToUpperCase` with a single string parameter that returns the string in uppercase. Log the result of calling this function.

- **Outcome:**

Ensure the function calculates the area correctly and converts strings to uppercase correctly.

2. Promises:

- **Task:**

Create a file named `promises.js` and write your own code that:

- Creates a promise named `fetchWeatherData` that simulates fetching weather data (use `setTimeout` to delay the resolution by 3 seconds) and resolves with a weather report object containing temperature and condition.
- Uses `.then()` to log the weather report when the promise is resolved.
- Uses `.catch()` to log an error message in case of rejection.

- **Outcome:**

Ensure the promise resolves correctly with a simulated weather report.

3. Async/Await:

- **Task:**

Create a file named `asyncAwait.js` and write your own code that:

- Defines an `async` function named `fetchUserDataAsync` that uses `await` to call a promise `fetchUserData` which resolves with a user profile object (name, age, and email).
- Logs the user profile inside a `try...catch` block to handle both success and error cases.

- **Outcome:**

Ensure the `async` function retrieves the user profile data correctly and handles errors properly.

Instructions:


Perform the following tasks:


1. Write the required code in separate files (`arrowFunctions.js`, `promises.js`, `asyncAwait.js`).
2. Run each file using Node.js to ensure the code executes without errors and demonstrates the use of ES6 syntax.

Resources:

-  **Arrow function expressions - JavaScript | MDN**
An arrow function expression is a compact alternative to a traditional function expression,...
 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions
-  **Using promises - JavaScript | MDN**
A Promise is an object representing the eventual completion or failure of an asynchronous...
 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

•

 **How to use promises - Learn web development | MDN**
Promises are the foundation of asynchronous programming in modern JavaScript. A promi...

 https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await

GitHub Instructions

1. Open in Visual Studio Code:

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.

If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. Complete the Task:

- Write your solution in the respective files (`arrowFunctions.js` , `promises.js` , `asyncAwait.js`).

3. Run and Test Your Code:

- Run your code to ensure it works correctly. Use the following command for each file:

```
node <filename>.js
```

4. Commit Your Changes:

- Commit your changes with a meaningful message:

```
git commit -m "Completed Basic JavaScript Review task"
```

5. Push Your Changes to Your Forked Repository:

- Push your changes to your forked repository:

```
git push origin main
```

6. Create a Pull Request:

- Go to your forked repository on GitHub.
- Click on the "Pull Requests" tab.
- Click the "New Pull Request" button.
- Ensure the base repository is the original template repository and the base branch is main.
- Ensure the head repository is your forked repository and the compare branch is main.
- Click "Create Pull Request".
- Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Fork the repository on GitHub

# Clone the forked repository
git clone https://github.com/your-github-username/repository-name.git
cd repository-name

# Complete the task by writing and running the code in the specified files

# Add, commit, and push your changes
git commit -m "Completed Basic JavaScript Review task"
git push origin main

# Create a pull request on GitHub
```