Module 1: Introduction to JavaScript

1. JavaScript Basics

What is JavaScript?

- JavaScript is a versatile, high-level programming language used for web development to add interactivity, control, and functionality to websites.
- JavaScript runs in the browser (client-side) and can also be used server-side with Node.js.

Setting up the Environment

- Use the in-browser console (open DevTools with `F12` or `Ctrl+Shift+I`) or setup XAMPP for a local environment.
- Basic setup of a JavaScript file linked to HTML using `<script src="script.js"></script>`.

Variables and Data Types

- **Data Types**: `string`, `number`, `boolean`, `null`, `undefined`, `symbol`, `object` (arrays and functions included).

Variable Declaration: Introduce `let`, `const`, and `var` with differences.

```
let age = 25;  // block-scoped
const name = "Alice";  // constant, block-scoped
var city = "Paris";  // function-scoped
```

Operators and Expressions

- Arithmetic (`+`, `-`, `*`, `/`, `%`)
- Assignment (`=`, `+=`, `-=`), Comparison (`==`, `===`, `!=`, `!==`, `>`, `<`)
- Logical (`&&`, `||`, `!`)

---

2. Basic Functions and Scope

Function Syntax

Function Declaration:

```
function greet(name) {
    return `Hello, ${name}!`;
}
```

Function Expression:

```
const greet = function(name) {
    return `Hello, ${name}!`;
};
```

Arrow Functions:

```
const greet = (name) => `Hello, ${name}!`;
```

Scope and Closures

- **Scope**: Local vs. Global Scope
- **Closure**: Functions remember variables from their original scope even after execution completes.

Exercises

1. Declare variables for name, age, and favorite color, and log them in a sentence.
2. Create a function that adds two numbers and returns the result.
3. Write a function that returns a personalized greeting based on a given name.

Module 2: Working with Data Structures

1. Arrays

Basic Array Operations

- **Create an Array**: `let fruits = ['apple', 'banana', 'cherry'];`
- **Methods**: `push()`, `pop()`, `shift()`, `unshift()`, `indexOf()`, `splice()`

Higher-Order Methods

`map`: Applies a function to each element and returns a new array.

```
let numbers = [1, 2, 3];
let doubled = numbers.map(num => num * 2);  // [2, 4, 6]
```

`filter`: Filters elements based on a condition.

```
let even = numbers.filter(num => num % 2 === 0);  // [2]
```

`reduce`: Reduces the array to a single value

```
let sum = numbers.reduce((acc, num) => acc + num, 0);  // 6
```

2. Objects

Creating and Manipulating Objects

Basic syntax:

```
const person = { name: "Alice", age: 25, city: "Paris" };
```

Object Destructuring and Spread Operator

```
const { name, age } = person;
const newPerson = { ...person, city: "London" };
```

Exercises

1. Create an array of numbers, and use `.map()` to return a new array with each number squared.
2. Write an object representing a car with properties for brand, model, and year.
3. Destructure the `car` object to retrieve the brand and year.

Module 3: Control Structures and Logic

1. Conditional Statements

`if`, `else`, `switch` Statements

```
let color = "red";
if (color === "blue") {
    console.log("Blue selected");
} else if (color === "red") {
    console.log("Red selected");
} else {
    console.log("Color not recognized");
}
```

```
switch (color) {
    case "blue":
        console.log("Blue selected");
        break;
    case "red":
        console.log("Red selected");
        break;
    default:
        console.log("Color not recognized");
}
```

Module 4: Advanced JavaScript Concepts

1. Error Handling

`try...catch` Blocks

```
try {
    let result = riskyOperation();
} catch (error) {
    console.error("An error occurred:", error);
}
```

2. Promises and Async Programming

Promises:

```
let promise = new Promise((resolve, reject) => {
    let success = true;
    if (success) resolve("Operation succeeded");
    else reject("Operation failed");
});
```

Async/Await:

```
async function fetchData() {
    try {
        let data = await fetch(url);
        let result = await data.json();
    } catch (error) {
        console.error(error);
```

```
    }
}
```

Exercises

1. Create a function that simulates a successful or failed API call using Promises.
2. Write a function using `async/await` to fetch data from a sample API.

Module 5: The DOM and Events

1. Document Object Model (DOM)

Accessing Elements

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

Manipulating Elements

```
button.textContent = "Click Me!";
```

2. Event Handling

Event Listeners

```
button.addEventListener("click", () => console.log("Button clicked"));
```

Module 6: ES6+ Features

1. Arrow Functions and Lexical `this`

- Arrow functions maintain `this` from the context they are created in, unlike regular functions.

2. Destructuring, Template Literals, Spread and Rest Operators

```
let fruits = ["apple", "banana"];
let newFruits = [...fruits, "cherry"];
```

Module 7: Working with APIs

1. Fetch API

Basic Usage

```
fetch("https://api.example.com/data")
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error(error));
```

Using `async/await`

```
async function getData() {
    let response = await fetch("https://api.example.com/data");
    let data = await response.json();
}
```