Week 2: JavaScript Conditions, Functions, Scope, and Loops

Lesson 1: Conditions (if/else statements)

1.1 What are Conditions?

- Conditions allow JavaScript to make decisions in the code.
- Example: "If it's raining, take an umbrella. Otherwise, wear t shirts."

1.2 If-Else Statements

```
Syntax:
if (condition) {
   // Code runs if condition is true
} else {
   // Code runs if condition is false
}
```

Example:

```
let age = 20;
if (age >= 18) {
   console.log("You are allowed to vote.");
} else {
   console.log("You are too young to vote.");
}
```

1.3 Else If (Chained Conditions)

Used for multiple conditions**:**

```
let score = 85;
if (score >= 90) {
   console.log("Grade: A");
} else if (score >= 75) {
   console.log("Grade: B");
} else {
   console.log("Grade: C");
}
```

1.4 Logical Operators in Conditions

- ❖ && (AND): Both conditions must be true.
- ❖ || (OR): At least one condition must be true.
- ❖ ! (NOT): Inverts a boolean value.

Example:

```
let hasTicket = true;
let isAdult = false;
if (hasTicket && isAdult) {
   console.log("You can enter the movie.");
} else {
   console.log("You cannot enter the movie.");
}
```

1.5 Exercises Using Conditions (if/else statements)

- Write a program that checks a user's bank balance and prints messages based on the balance.
- Modify the program to add VIP status if the balance is above 50,000.

```
// Define the user's bank balance
let userBalance = 60000;  // Modify this value to test with different balances
// Check the balance and print messages based on the balance
if (userBalance >= 50000) {
   console.log("Your balance is $" + userBalance + ". You are a VIP customer.");
} else if (userBalance >= 10000) {
   console.log("Your balance is $" + userBalance + ". You are a valued customer.");
} else if (userBalance >= 0) {
   console.log("Your balance is $" + userBalance + ". You have sufficient funds.");
} else {
   console.log("Your balance is negative. Please contact your bank.");
}
```

Lesson 2: Functions

2.1 What is a Function?

- ❖ A function is a reusable block of code.
- ❖ Helps avoid repeating code.

2.2 Creating a Function

```
function greet() {
  console.log("Hello, world!");
}
greet();
```

2.3 Function Parameters and Arguments

```
function greetUser(name) {
  console.log("Hello, " + name + "!");
}
greetUser("John");
```

2.4 Return Values

```
function add(a, b) {
  return a + b;
}
let sum = add(5, 10);
console.log(sum);
```

2.5 Exercises Using Functions

- Write a program that checks a user's bank balance and prints messages based on the balance.
- Modify the program to add VIP status if the balance is above 50,000.

```
// Function to check bank balance and print messages
function checkBankBalance(balance) {
    if (balance >= 50000) {
        console.log("Your balance is $" + balance + ". You are a VIP customer.");
    } else if (balance >= 10000) {
        console.log("Your balance is $" + balance + ". You are a valued customer.");
    } else if (balance >= 0) {
        console.log("Your balance is $" + balance + ". You have sufficient funds.");
    } else {
        console.log("Your balance is negative. Please contact your bank.");
    }
}

// Example usage
let userBalance = 60000;  // Modify this value to test with different balances
checkBankBalance(userBalance);
```

2.6 Exercises

- ❖ Write a function that takes two numbers and returns their multiplication.
- ❖ Create a function that converts Celsius to Fahrenheit.

Lesson 3: Scope (Global vs Local Variables)

3.1 What is Scope?

Determines where variables can be accessed.

- **Global Scope**: Accessible anywhere.
- **Local Scope**: Accessible only inside a function.

3.2 Example of Scope

```
let globalVar = "I am global";
function exampleFunction() {
  let localVar = "I am local";
  console.log(globalVar); // Accessible
  console.log(localVar);  // Accessible
}
exampleFunction();
console.log(localVar); // Error: localVar is not defined
```

3.3 Static Scope

- Inner functions can access outer function variables.

```
function outer() {
  let outerVar = "Outer";
  function inner() {
    console.log(outerVar);
  }
  inner();
}
outer();
```

3.4 Exercises

- Write a function where a variable is declared globally and accessed in different functions.
- Modify a function to use a locally declared variable and see the scope error.

Lesson 4: Loops (For Loop)

4.1 What is a Loop?

- ❖ Loops repeat a block of code multiple times.
- ❖ Example: Counting from 1 to 10.

4.2 For Loop Syntax

```
for (let i = 1; i <= 5; i++) {
  console.log("Iteration number: " + i);
}
```

4.3 Example of a For Loop

```
let fruits = ["Apple", "Banana", "Cherry"];
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

4.4 Exercises

- ❖ Create a for loop that prints numbers from 1 to 100.
- ❖ Modify the loop to print only even numbers.
- ❖ Write a function that prints a message multiple times using a loop.

Project: Order Processing System

Part 1: Basic Order System

1. Create a folder order-system.
2. Inside, create index.html and order.js.
3. In order.js, write a script to log console.log("Order system ready");.
4. Declare variables:
    - customerName: Stores the customer's name.
    - item: Stores the name of the item.
    - quantity: Stores the quantity of items ordered.
    - price: Stores the price per item.

5. Log a summary message:

```
console.log(`Order summary: ${customerName} ordered ${quantity} ${item}s. Total price: ${totalPrice}.`)
```

Part 2: Advanced Order System

1. Add a variable isExpressDelivery. If true, add a $5 delivery fee.
2. Calculate the total cost using a function.
3. Modify the order summary to include the delivery fee if applicable.
4. Log the final order message.

Final Project: Students Build Their Own Order System

Now, let students modify the project to handle different types of items (e.g., coffee, books, etc.). Encourage them to add new features like tax calculation or discount coupons.