JavaScript Conditional Statements and Loops

Introduction to Conditional Statements

- **Conditional statements** allow you to perform different actions based on different conditions.
- The main types of conditional statements in JavaScript are:
  - `if` statement
  - `else` statement
  - `else if` statement
  - `switch` statement

The `if` Statement

- The `if` statement executes a block of code if a specified condition is true.
- **Syntax:**

```
if (condition) {
    // block of code to be executed if the condition is true
}
```

```
let age = 18;
if (age ⩾ 18) {
    console.log("You are an adult.");
}
```

The `else` Statement

- The `else` statement executes a block of code if the same condition is false.
- **Syntax:**

```
if (condition) {
    // block of code to be executed if the condition is true
} else {
    // block of code to be executed if the condition is false
}
```

The `else` Statement example

```
let age = 16;
if (age ⩾ 18) {
    console.log("You are an adult.");
} else {
    console.log("You are a minor.");
}
```

The `else if` Statement

- The `else if` statement specifies a new condition to test if the first condition is false.
- **Syntax:**

```
if (condition1) {
    // block of code to be executed if condition1 is true
} else if (condition2) {
    // block of code to be executed if condition2 is true
} else {
    // block of code to be executed if both conditions are false
}
```

The `else if` Statement example

```
let score = 85;
if (score ⩾ 90) {
    console.log("Grade: A");
} else if (score ⩾ 80) {
    console.log("Grade: B");
} else {
    console.log("Grade: C");
}
```

The `switch` Statement

- The `switch` statement is used to perform different actions based on different conditions.
- **Syntax:**

```
switch(expression) {
    case value1:
        // block of code to be executed if expression === value1
        break;
    case value2:
        // block of code to be executed if expression === value2
        break;
    default:
        // block of code to be executed if expression doesn't match any case
}
```

The `switch` Statement example

```
let day = 3;
switch(day) {
    case 1:
        console.log("Monday");
        break;
    case 2:
        console.log("Tuesday");
        break;
    case 3:
        console.log("Wednesday");
        break;
    default:
        console.log("Another day");
}
```

Introduction to Loops

- **Loops** are used to repeat a block of code a number of times.
- The main types of loops in JavaScript are:
  - `for` loop
  - `while` loop
  - `do ... while` loop

The `for` Loop

- The `for` loop repeats a block of code a specified number of times.
- **Syntax:**

```
for (initialization; condition; increment) {
    // block of code to be executed
}
```

```
for (let i = 0; i < 5; i++) {
    console.log("Iteration " + i);
}
```

The `while` Loop

- The `while` loop repeats a block of code as long as a specified condition is true.
- **Syntax:**

```
while (condition) {
    // block of code to be executed
}
```

```
let i = 0;
while (i < 5) {
    console.log("Iteration " + i);
    i++;
}
```

The `do ... while` Loop

- The `do ... while` loop repeats a block of code at least once, and then continues to repeat the loop as long as a specified condition is true.
- **Syntax:**

```
do {
    // block of code to be executed
} while (condition);
```

The `do ... while` Loop example

```
let i = 0;
do {
    console.log("Iteration " + i);
    i++;
} while (i < 5);
```

Practice Example 1

- **Example 1:**

```
let number = 7;
if (number % 2 === 0) {
    console.log("Even number");
} else {
    console.log("Odd number");
}
```

Practice Example 2

- **Example 2:**

```
for (let i = 1; i ⩽ 10; i++) {
    if (i % 2 === 0) {
        console.log(i + " is even");
    } else {
        console.log(i + " is odd");
    }
}
```

Practice Example 3

- **Example 3:**

```
let i = 1;
while (i <= 5) {
    console.log("Number " + i);
    i++;
}
```

Summary

- Conditional statements and loops are fundamental concepts in JavaScript.
- They allow you to control the flow of your program and perform repetitive tasks efficiently.
- Practice using these constructs to become proficient in JavaScript programming.