4-Control Flow

Chapter 4: Control Flow

Control flow determines the order in which code is executed based on conditions and logic. In JavaScript, control flow is the backbone of creating interactive and dynamic applications.

4.1 Conditional Statements

Conditional statements allow your code to make decisions and execute different blocks of code depending on specific conditions.

4.1.1 `if` Statement

The `if` statement executes a block of code if a specified condition evaluates to `true`.

Syntax:

```
if (condition) {
  // Code to execute if the condition is true
}
```

Example:

```
let age = 18;

if (age >= 18) {
  console.log("You're allowed to vote!");
}
```

4.1.2 `if...else` Statement

The `else` statement runs if the `if` condition is false.

Syntax:

```
if (condition) {
```

```
  // Code to execute if the condition is true
} else {
  // Code to execute if the condition is false
}
```

Example:

```
let age = 16;

if (age >= 18) {
  console.log("You're allowed to vote!");
} else {
  console.log("You're too young to vote.");
}
```

4.1.3 `if...else if...else` Statement

Use this structure when you have multiple conditions to check.

Syntax:

```
if (condition1) {
  // Code to execute if condition1 is true
} else if (condition2) {
  // Code to execute if condition2 is true
} else {
  // Code to execute if none of the conditions are true
}
```

Example:

```
let score = 85;

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

4.1.4 Ternary Operator

The ternary operator is a shorthand for `if...else` statements.

Syntax:

```
condition ? expressionIfTrue : expressionIfFalse;
```

Example:

```
let age = 20;
let message = age >= 18 ? "You're an adult." : "You're a minor.";
console.log(message); // Output: "You're an adult."
```

4.2 Switch Statements

The `switch` statement is an alternative to multiple `if...else if` conditions, often cleaner for handling multiple specific cases.

Syntax:

```
switch (expression) {
  case value1:
    // Code to execute if expression === value1
    break;
  case value2:
    // Code to execute if expression === value2
    break;
  default:
    // Code to execute if no cases match
}
```

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("Unknown day");
  }
```

4.3 Loops

Loops allow you to execute a block of code multiple times, which is especially useful for repetitive tasks.

4.3.1 `for` Loop

The `for` loop is best for running a block of code a specific number of times.

Syntax:

```
for (initialization; condition; increment) {
  // Code to execute
}
```

Example:

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

4.3.2 `while` Loop

The `while` loop executes as long as the specified condition is true.

Syntax:

```
while (condition) {
  // Code to execute
}
```

Example:

```
let counter = 0;

while (counter < 5) {
  console.log("Counter:", counter);
  counter++;
}
```

4.3.3 `do...while` Loop

The `do...while` loop is similar to `while`, but it executes the block of code *at least once*, even if the condition is false.

Syntax:

```
do {
  // Code to execute
} while (condition);
```

Example:

```
let num = 5;

do {
  console.log("Number:", num);
  num++;
} while (num < 3); // Executes once even though the condition is false
```

4.3.4 `for...of` Loop

The `for...of` loop iterates over iterable objects like arrays.

Syntax:

```
for (element of iterable) {
  // Code to execute
}
```

Example:

```
let colors = ["red", "green", "blue"];

```

```
for (let color of colors) {
  console.log(color);
}
```

4.3.5 `for...in` Loop

The `for...in` loop iterates over the keys of an object.

Syntax:

```
for (key in object) {
  // Code to execute
}
```

Example:

```
let person = { name: "Rick", age: 30, profession: "Developer" };

for (let key in person) {
  console.log(key + ": " + person[key]);
}
```

4.4 Practical Examples

1. `**FizzBuzz` Program:**

```
for (let i = 1; i <= 20; i++) {
  if (i % 3 === 0 && i % 5 === 0) {
    console.log("FizzBuzz");
  } else if (i % 3 === 0) {
    console.log("Fizz");
  } else if (i % 5 === 0) {
    console.log("Buzz");
  } else {
    console.log(i);
  }
}
```

2. **Checking for Prime Numbers:**

```
function isPrime(num) {
```

```
  if (num <= 1) return false;

  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
  }
  return true;
}

console.log(isPrime(7)); // Output: true
```