Branching and Conditionals

Week 3

Adib Sakhawat
IUT CSE ‘21

Conditional statements are fundamental in programming languages, allowing you to execute different code blocks based on certain conditions. In JavaScript, the primary conditional statements are `if`, `else if`, `else`, and the ternary operator (`? :`). Understanding how these work is crucial for controlling the flow of your programs.

The `if` Statement

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

Syntax

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

Example

```
let age = 18;

if (age >= 18) {
    console.log('You are eligible to vote.');
}
```

Explanation: In this example, the message "You are eligible to vote." will be printed to the console only if the `age` variable is greater than or equal to 18.

The `else` Statement

The `else` statement executes a block of code **if the same condition** is `false`. It must follow an `if` or `else if` statement.

Syntax

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

Example

```
let age = 16;

if (age >= 18) {
    console.log('You are eligible to vote.');
} else {
    console.log('You are not eligible to vote.');
}
```

Explanation: Since `age` is 16, which is less than 18, the else block executes, printing "You are not eligible to vote."

The `else if` Statement

The `else if` statement specifies a new condition to test if the previous condition(s) was `false`. You can have multiple `else if` statements following an `if`.

Syntax

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

Example

```javascript
let score = 85;

if (score >= 90) {
    console.log('Grade: A');
} else if (score >= 80) {
    console.log('Grade: B');
} else if (score >= 70) {
    console.log('Grade: C');
} else {
    console.log('Grade: F');
}
```

Explanation: The program checks each condition in order. Since `score` is 85, it meets the condition `score >= 80`, so "Grade: B" is printed.

The Ternary Operator

The ternary operator (`? :`) is a shorthand way of writing an `if-else` statement. It takes three operands: a condition, a result for `true`, and a result for `false`.

Syntax

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

Example

```javascript
let age = 20;
let canVote = (age >= 18) ? 'Yes, you can vote.' : 'No, you
cannot vote.';
console.log(canVote);
```

Explanation: If `age` is 18 or older, `canVote` is assigned "Yes, you can vote."; otherwise, it's assigned "No, you cannot vote."

Combining Conditional Statements

You can combine multiple conditions using logical operators like `&&` (AND), `||` (OR), and `!` (NOT).

Example with Logical Operators

```javascript
let isMember = true;
let age = 17;

if (isMember && age >= 18) {
    console.log('Access granted to member lounge.');
} else if (!isMember && age >= 18) {
    console.log('Please register to become a member.');
} else {
    console.log('Access denied.');
}
```

Explanation:

- If the user is a member and 18 or older, they get access.
- If they're not a member but 18 or older, they're prompted to register.
- Anyone under 18 is denied access.

Nested Conditional Statements

You can place conditional statements inside other conditional statements.

Example of Nested `if` Statements

```javascript
let userType = 'admin';
let isLoggedIn = true;

if (isLoggedIn) {
    if (userType === 'admin') {
        console.log('Welcome, Admin!');
    } else {
        console.log('Welcome, User!');
    }
} else {
    console.log('Please log in.');
}
```

Explanation: The outer `if` checks if the user is logged in. If so, the inner `if` checks the `userType`.

Practical Examples

Example 1: Checking Even or Odd Number

```
let number = 7;

if (number % 2 === 0) {
    console.log(number + ' is even.');
} else {
    console.log(number + ' is odd.');
}
```

Explanation: The `%` operator gives the remainder. If a number modulo 2 equals 0, it's even.

Example 2: Simple Calculator Using `if-else`

```
let operator = '+';
let num1 = 5;
let num2 = 3;
let result;

if (operator === '+') {
    result = num1 + num2;
} else if (operator === '-') {
    result = num1 - num2;
} else if (operator === '*') {
    result = num1 * num2;
} else if (operator === '/') {
    result = num1 / num2;
} else {
    console.log('Invalid operator');
}

console.log('Result:', result);
```

Explanation: Depending on the `operator`, the corresponding arithmetic operation is performed.

Best Practices

- **Use Braces `{}` Even for Single Statements:** Improves readability and reduces errors.

```
// Less readable
if (condition) console.log('Do something');

// More readable
if (condition) {
    console.log('Do something');
}
```

- **Avoid Deep Nesting:** Too many nested `if` statements can make code hard to read. Consider using logical operators or functions to simplify.
- **Use Strict Equality `===`:** Checks both value and type.

```
if (num === 10) {
    // Good practice
}
```

- **Consider the Ternary Operator for Simple Conditions:** But avoid using it for complex conditions as it can reduce readability.

Common Pitfalls

- **Assignment vs. Comparison:**

  Be careful not to use `=` (assignment) instead of `==` or `===` (comparison).

```javascript
if (x = 10) {
    // This assigns 10 to x and always evaluates to true
}

if (x === 10) {
    // Correctly checks if x equals 10
}
```

- **Falsy Values:**

  In JavaScript, values like `0`, `''` (empty string), `null`, `undefined`, and `NaN` are considered falsy.

  ```javascript
  let value = 0;

  if (value) {
      console.log('Value is truthy');
  } else {
      console.log('Value is falsy');
  }
  // Outputs: Value is falsy
  ```

- **Chaining Conditions Incorrectly:**

  Ensure that `else if` statements are properly placed and that conditions are mutually exclusive when necessary.

Summary

- `if` **Statement:** Executes code if a condition is true.
- `else` **Statement:** Executes code if the same condition is false.
- `else if` **Statement:** Checks another condition if previous ones are false.
- **Ternary Operator:** A shorthand for `if-else` statements.