Logical Operators

AND

Or

NOT

1. Types: && (AND), || (OR), ! (NOT)
2. AND (&&): All conditions must be true for the result to be true.
3. OR (||): Only one condition must be true for the result to be true.
4. NOT (!): Inverts the Boolean value of a condition.
5. Lower Priority than Math and Comparison operators

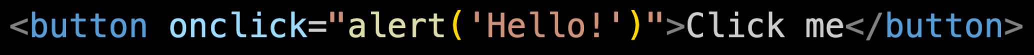
Logical Operators

```
console.log("And Operator");
console.log(true && true);
console.log(true && false);
console.log(false && true);
console.log(false && false);

console.log("Or Operator");
console.log(true || true);
console.log(true || false);
console.log(false || true);
console.log(false || false);

console.log("Not Operator");
console.log(!true);
console.log(!false);
console.log(!!true);
```

```
let num = 5;
if (num > 0 && num % 2 === 0) {
  console.log("Positive and Even number");
} else if (num > 0) {
  console.log("Positive and Odd number");
} else if (num < 0) {
  console.log("Negative number");
} else {
  console.log("Number is Zero");
}
```

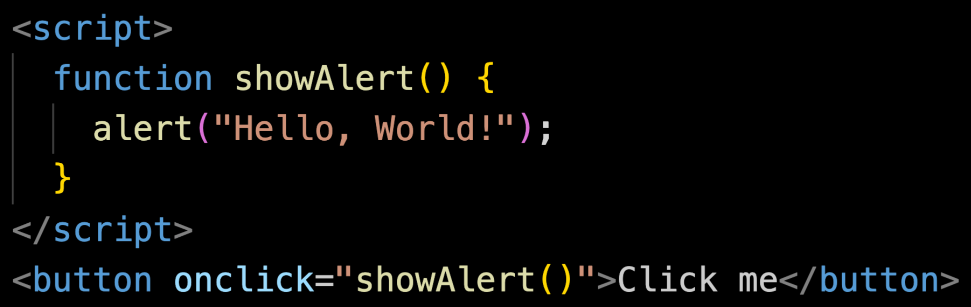
Including Script

1. Inline JavaScript:
   - Place JavaScript code directly within the HTML element using the onclick, onload, or other event attributes.



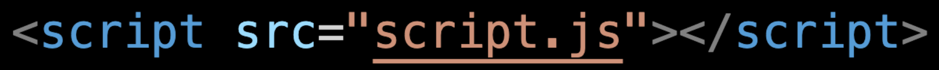
```
<button onclick="alert('Hello!')">Click me</button>
```

2. Internal JavaScript:
   - Embed JavaScript code within a <script> tag in the HTML document's <head> or <body>



```
<script>
  function showAlert() {
    alert("Hello, World!");
  }
</script>
<button onclick="showAlert()">Click me</button>
```

3. External JavaScript:
   - Link to an external JavaScript file using the <script> tag with the src attribute.



```
<script src="script.js"></script>
```

Template Literals

1. **Syntax:** Enclosed by backticks (`) instead of single or double quotes.
2. **Multi-line Strings:** Allows creating multi-line strings without the need for escape characters.
3. **String Interpolation:** Embed expressions within a string using ${expression} syntax.
4. **Expression Evaluation:** Supports embedding any valid JavaScript expression, including arithmetic operations, function calls, and ternary operations.

```javascript
// Syntax
let greeting = `Hello, World!`;

// Multi-line Strings
let multiLine = `This is a
multi-line string.`;

// String Interpolation
let firstName = 'John';
let hello = `Hello, ${firstName}!`;

// Expression Evaluation
let a = 5;
let b = 10;
let result = `The sum of a and b is ${a + b}.`;
```

Revisiting Undefined & Null

1. undefined:

- A variable that has been declared but has not yet been assigned a value.
- Represents the absence of a value in a variable.
- Also returned when trying to access an object property that does not exist.

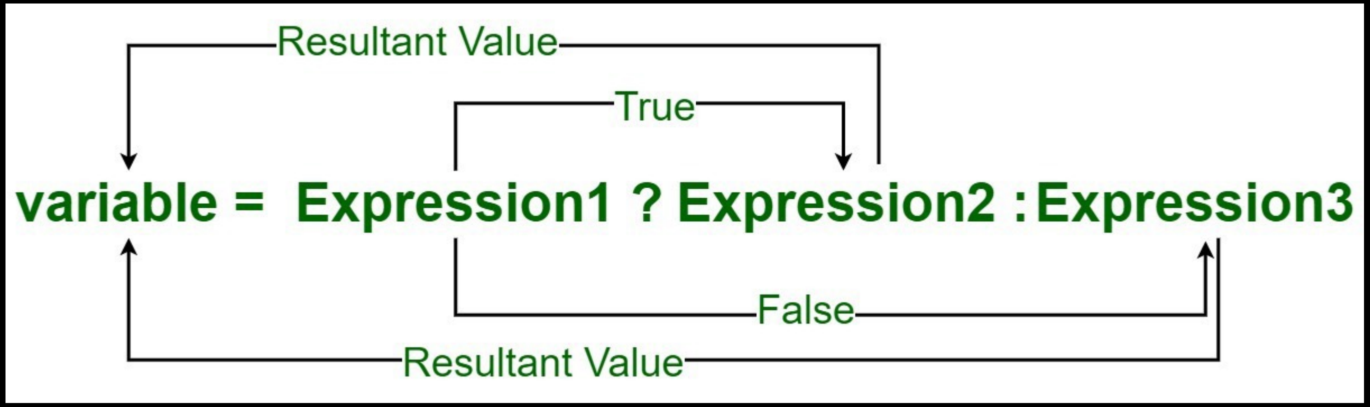
```
let a;
console.log(a); // Output: undefined
let obj = {};
console.log(obj.property); // Output: undefined
```

2. null:

- Represents the intentional absence of any object value.
- Often used to explicitly indicate that a variable should be empty.
- Both undefined and null are loosely equal (==) but not strictly equal (===).

```
let b = null;
console.log(b); // Output: null
console.log(undefined == null); // Output: true
console.log(undefined === null); // Output: false
```

If alternates



1. Ternary Operator: condition ? trueValue : falseValue

   Quick one-line if-else.

2. Guard Operator: value || defaultValue

   Use when a fallback value is needed.

3. Default Operator: value ?? fallbackValue

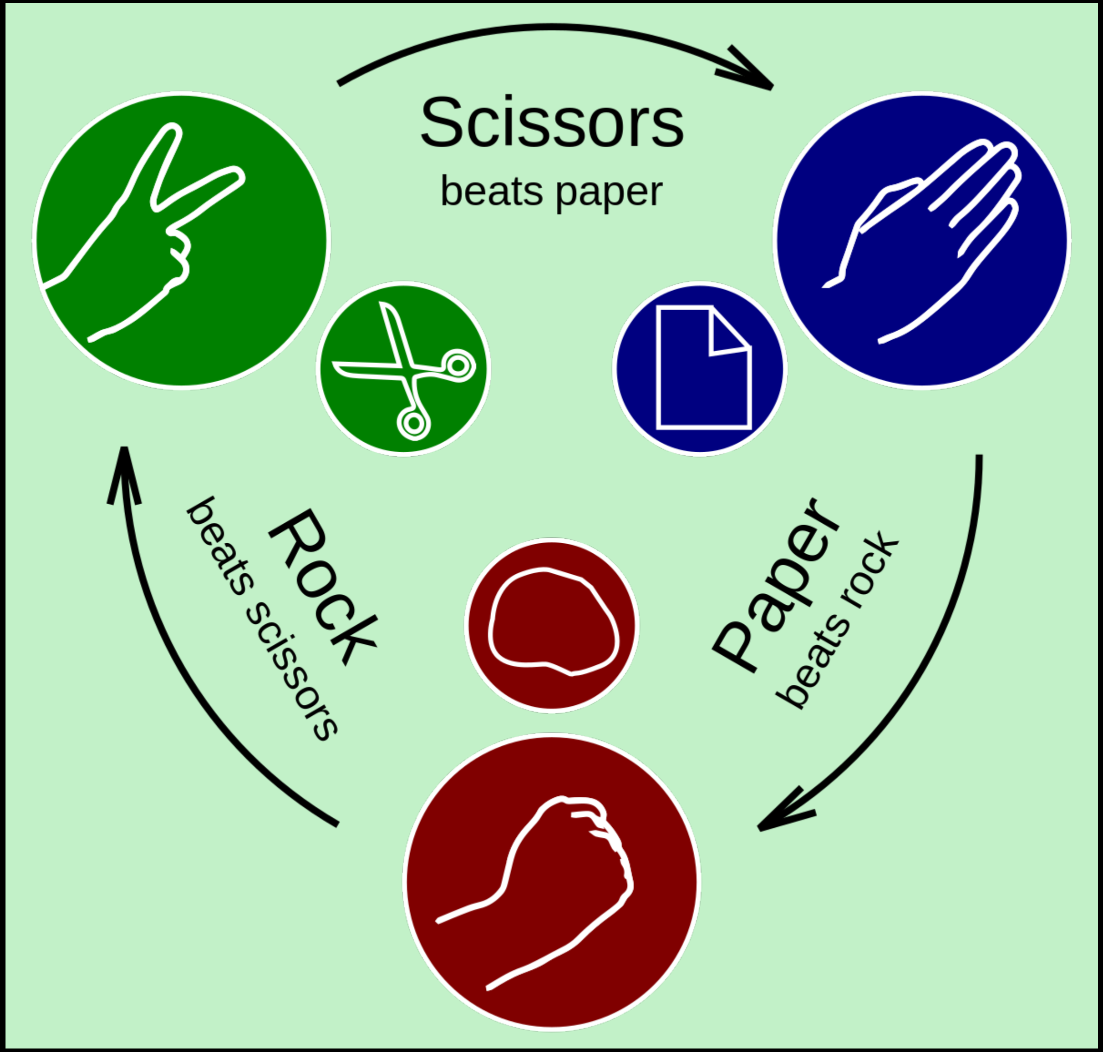
   Use when you want to consider only null and undefined as falsy.

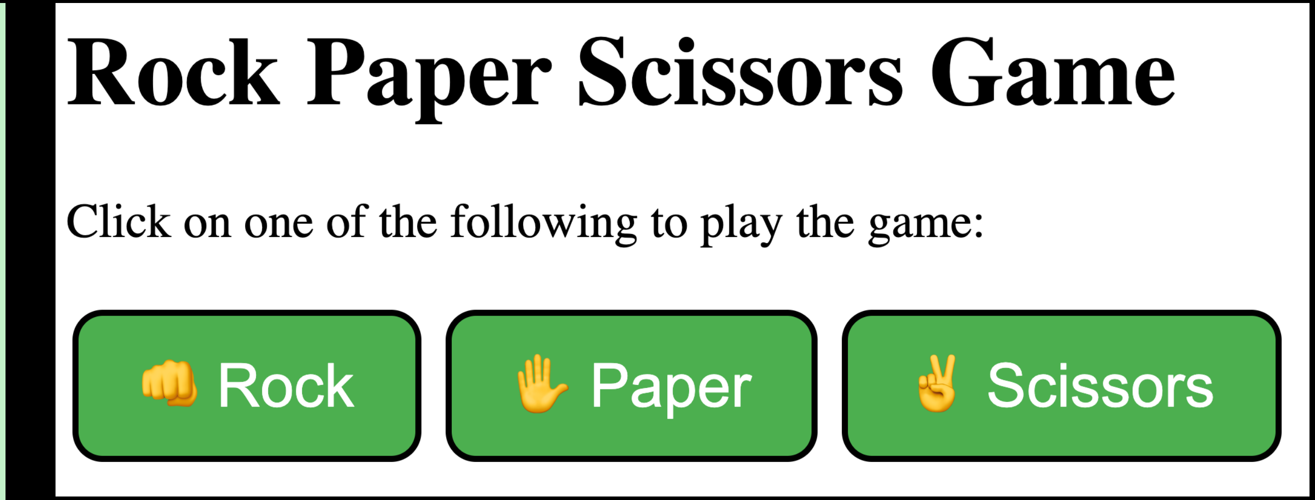
4. Simplifies conditional logic.
5. Use wisely to maintain readability.

If alternates

```javascript
// Ternary Operator:
let age = 20;
let canVote = (age >= 18) ? 'Yes' : 'No';
console.log(canVote); // Output: Yes

// Guard Operator:
let username = '';
let defaultUsername = 'Guest';
let displayName = username || defaultUsername;
console.log(displayName); // Output: Guest

// Default Operator:
let userAge = null;
let defaultAge = 18;
let ageToDisplay = userAge ?? defaultAge;
console.log(ageToDisplay); // Output: 18
```

Project Rock-Paper-Scissor Game





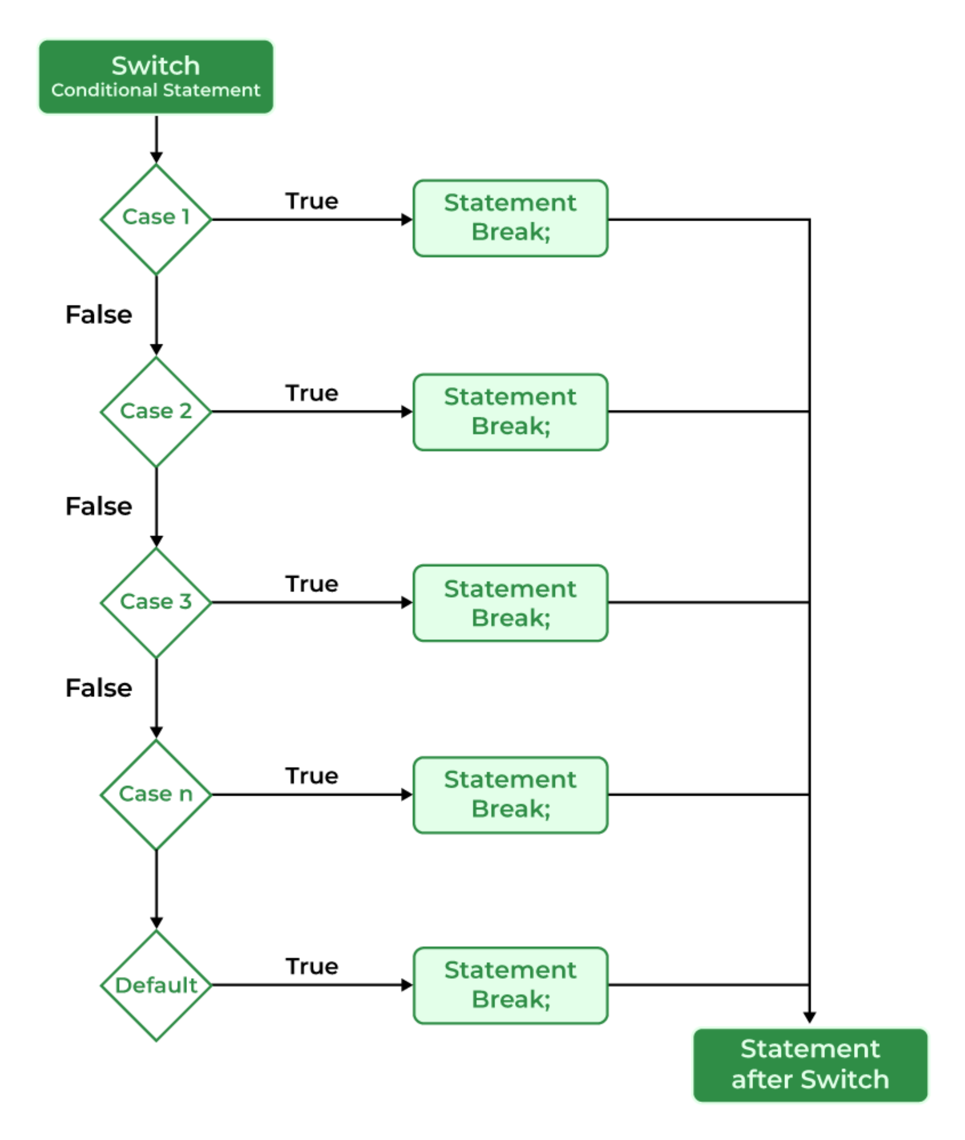
Rock Paper Scissors Game

Click on one of the following to play the game:

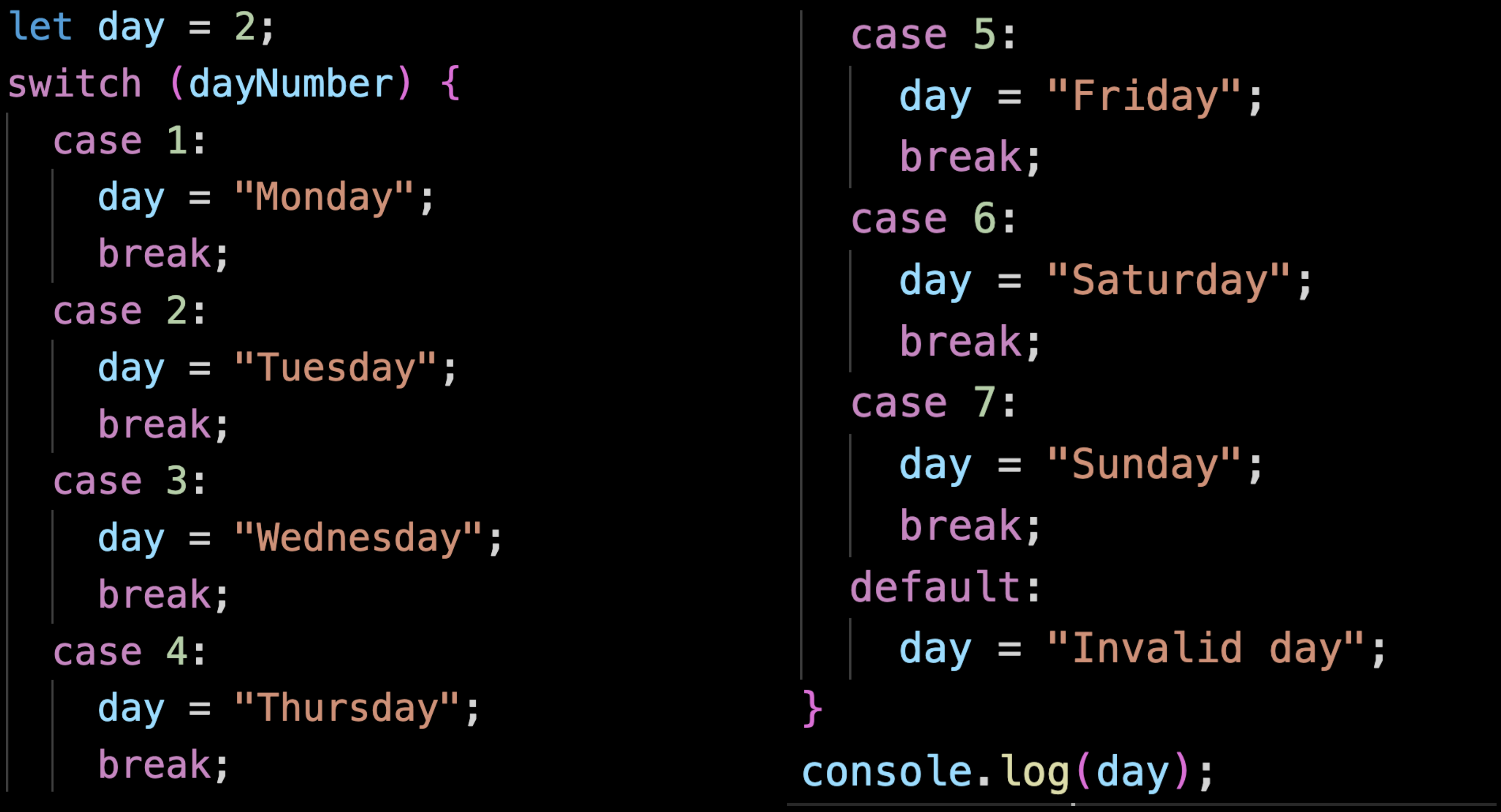
- Use Math.random to do computer move.
- Use logical operators to compute the result

Switch

1. Multi-way Branching: switch provides a cleaner method for multi-way branching than multiple if-else statements when testing the same expression.
2. Case Labels: Represents individual branches. Execution jumps to the matching case label.
3. Break Statement: Typically used to exit the switch block after a case is executed to prevent "fall through" to subsequent cases.
4. Default Case: Optional. Executes if no case matches. Placed at the end of the switch block.
5. Enhances Readability: For certain types of conditional logic, switch can make the code more readable compared to nested if-else statements.
6. The switch statement compares the expression's value strictly (===) with the values of the case clauses.



Switch



```
let day = 2;
switch (dayNumber) {
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
    break;
  case 7:
    day = "Sunday";
    break;
  default:
    day = "Invalid day";
}
console.log(day);
```

Practice Exercise

Decision Control

1. If-Else Statement: Write code that checks if a number is positive, negative, or zero. Use an ` if-else ` statement for this purpose.
2. Nested If Statement: Write code to determine the ticket price for a movie. The function should consider the following:
   - If the viewer is under 13, the ticket is free.
   - If the viewer is between 13 and 60, check if it's a weekend. If yes, the ticket price is Rs 500; otherwise, it’s Rs 300.
   - If the viewer is over 60, the ticket price is Rs 250.
3. If-Else If Ladder: Write code to determine grades based on marks:
   - Above 90 is 'A’.
   - 80 to 89 is 'B’.
   - 70 to 79 is 'C’.
   - 60 to 69 is 'D’.
   - Below 60 is 'F’.
4. Ternary Operator: Use the ternary operator in JavaScript to assign a value to a variable named ` status ` . The value should be "Adult" if the age is 18 or above, and "Minor" otherwise.

Practice Exercise

Decision Control

5. Switch Case: Write a JavaScript switch case statement that evaluates the variable `day` and returns a specific greeting:
   - "Happy Monday!" for Monday.
   - "Terrific Tuesday!" for Tuesday.
   - "Wonderful Wednesday!" for Wednesday.
   - "Thriving Thursday!" for Thursday.
   - "Fun Friday!" for Friday.
   - "Super Saturday!" for Saturday.
   - "Serene Sunday!" for Sunday.
6. Comparison and Logical Operators: Write a JavaScript expression using logical operators that checks if a variable `age` is either below 13 or above 65. Also, use a comparison operator to determine if another variable `income` is greater than or equal to 50000.
7. Guard Operator: Demonstrate the use of the guard operator to handle null or undefined values. Write codewhere a variable `userInput` is checked. If `userInput` is null or undefined, assign "No input provided" to another variable `output`.