**CONTROL FLOW**

**Question 1: What is control flow in JavaScript? Explain how if-else statements work with an example.**

**Control Flow in JavaScript**

Control flow in JavaScript refers to the **order in which the code is executed**. Normally, code runs sequentially from top to bottom. However, using **control flow statements**, you can alter this sequence based on conditions, loops, or other logic.

**if-else Statements**

The if-else statement is a control flow structure that allows you to execute a block of code based on a condition. It evaluates a condition (a boolean expression) and:

* Executes the code inside the if block if the condition is true.
* Executes the code inside the else block if the condition is false.

**Syntax:**

if (condition) {

// Code to execute if condition is true

} else {

// Code to execute if condition is false

}

**Example:**

let age = 18;

if (age >= 18) {

console.log("You are eligible to vote.");

} else {

console.log("You are not eligible to vote.");

}

**Output:**

You are eligible to vote.

**Nested if-else**

You can also nest if-else statements for multiple conditions:

let marks = 85;

if (marks >= 90) {

console.log("Grade: A");

} else if (marks >= 75) {

console.log("Grade: B");

} else {

console.log("Grade: C");

}

**Output:**

Grade: B

**Key Points:**

1. The condition inside the if must evaluate to true or false.
2. Use comparison operators (>, <, ==, ===, etc.) in the condition to compare values.
3. The else block is optional. You can use only if if needed.

**Example with else if:**

let number = 0;

if (number > 0) {

console.log("Positive number");

} else if (number < 0) {

console.log("Negative number");

} else {

console.log("Zero");

}

**Output:**

Zero

**Question 2: Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?**

**switch Statement in JavaScript**

The switch statement is another control flow structure in JavaScript that allows you to execute one of many possible code blocks based on a specific value or expression. It is particularly useful when you have a **single variable or expression** that can have many possible outcomes, making it a cleaner alternative to multiple if-else if statements.

**Syntax of switch Statement:**

switch(expression) {

case value1:

// Code to execute if expression matches value1

break;

case value2:

// Code to execute if expression matches value2

break;

// Add more cases as needed

default:

// Code to execute if no case matches (optional)

}

* **expression**: The value being evaluated (could be a variable or an expression).
* **case valueX**: Each case represents a possible match for the expression. If a match is found, the corresponding code block is executed.
* **break**: Stops the execution of the switch statement once a case is matched. Without break, the code will continue executing the next cases (this is called "fall-through").
* **default**: An optional block that runs if none of the cases match. It's like the "else" block in an if-else statement.

**Example of switch Statement:**

let day = 3;

let dayName;

switch(day) {

case 1:

dayName = "Monday";

break;

case 2:

dayName = "Tuesday";

break;

case 3:

dayName = "Wednesday";

break;

case 4:

dayName = "Thursday";

break;

case 5:

dayName = "Friday";

break;

case 6:

dayName = "Saturday";

break;

case 7:

dayName = "Sunday";

break;

default:

dayName = "Invalid day";

}

console.log(dayName);

**Output:**

Wednesday

**Key Points:**

* switch compares the **expression** with each case value using strict equality (===).
* The break statement prevents **fall-through**; without it, once a case matches, the code will continue executing all subsequent cases until a break is encountered or the switch statement ends.
* The default block is optional but useful as a fallback when none of the case values match.

**When to Use switch Instead of if-else:**

1. **Multiple Conditions on the Same Variable/Expression**:  
   If you need to check a single variable or expression against many possible values, the switch statement is cleaner and more readable than multiple if-else if statements.
2. // Example using switch
3. switch(color) {
4. case 'red':
5. console.log('Red color selected');
6. break;
7. case 'green':
8. console.log('Green color selected');
9. break;
10. case 'blue':
11. console.log('Blue color selected');
12. break;
13. default:
14. console.log('Unknown color');
15. }
16. **When Conditions Are Constant Values**:  
    When the conditions you are checking are constant values (like numbers, strings, or even boolean values), switch is preferable, as it avoids repetitive and lengthy comparisons.
17. **Cleaner and More Readable Code**:  
    switch is more concise and structured, especially when you need to match a value against many cases. It reduces nesting and makes your code easier to follow.

**When Not to Use switch:**

* When you need to check complex conditions that involve comparisons or ranges (e.g., x > 10 && x < 20), if-else is better.
* When you need to perform checks on expressions rather than a fixed set of values.

**Example Where if-else is Preferable:**

let age = 18;

if (age >= 18 && age < 21) {

console.log("Young adult");

} else if (age >= 21 && age < 65) {

console.log("Adult");

} else {

console.log("Senior");

}

In summary:

* Use **switch** for comparing a single variable or expression with multiple possible values (especially when the values are constants).
* Use **if-else** for more complex conditions or when evaluating a variety of different expressions or logical comparisons.