7.4 JS Conditional Operators

armed statue of liberty - answered by Samantha Zoe Santos

- Condition statements are used to control the flow of a program based on whether a given condition is `true` or `false`.
- Condition statements allow you to execute different blocks of code depending on the evaluation of these conditions.

Keyword(s)	Description	Sample code	Console output

`if`	• Used to conditionally execute a block of code when a specified condition evaluates to `true`.	`let x = 15;`
		
		`if (amount >= 12) {`
		`  console.log("That is at least a dozen!");`
		`}`
			`That is at least a dozen!`

`else`	• Used to specify an alternative block of code to execute when the condition associated with the preceding `if` statement evaluates to false.	`let x = 5;`
		
		`if (amount >= 12) {`
		`  console.log("That is at least a dozen!");`
		`} else {`
		`  console.log("That is not a`
			`That is not a dozen!`

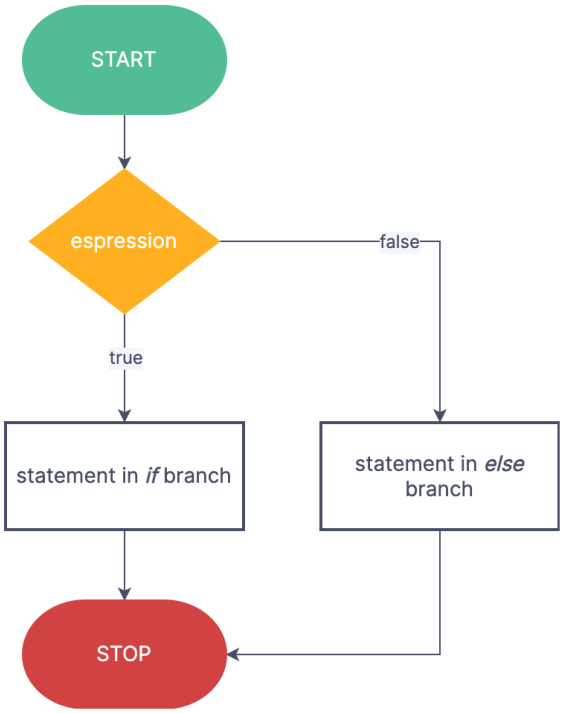
		`dozen!");` `}`	

`else if`	• Used to introduce an additional conditional branch in a series of `if`-`else` statements.	`let x = -2;`
		
		`if (amount >= 12) {`
		`  console.log("That is at least a dozen!");`
		`} else if (amount < 0) {`
		`  console.log("Negative numbers can not be dozens!");`
		`} else {`
		`  console.log("That is not a dozen!");`
		`}`
			`Negative numbers can not be dozens!`

`switch`	• Used to evaluate a single expression against multiple possible cases and execute code blocks associated with the matching case.	`let fruit = "banana";`
		`let message;`
		
		`switch (fruit) {`
		`  case "apple":`
		`  message = "You have an apple.";`
		`  break;`
		`  case "banana":`
		`  message = "You have a banana.";`
		`  break;`
		`  case "orange":`
		`  message = "You have an orange.";`
		`  break;`
		`  default:`
		`  message = "You have something else.";`
		`}`
		
		`console.log(message);`
			`You have a banana.`

`case`	• Used to define individual conditions or values to be compared against the switch expression.		
`default`	• Used to specify a fallback or default action or value to be taken when no other specific condition or case is met		
`break`	• Used within loops and `switch` statements to exit the loop or switch block prematurely.		

Conditional statements



- Use the `if` statement to specify a block of JavaScript code to be executed if a condition is true.

```
if (hour < 18) {
  greeting = "Good day";
}
```

- Use the `else` statement to specify a block of code to be executed if the condition is false.

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

- Use the `else if` statement to specify a new condition if the first condition is false.

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Conditional ternary operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

```
variablename = (condition) ? value1:value2
```

- For example:

```
let voteable = (age < 18) ? "Too young":"Old enough";
```

- If the variable `age` is a value below 18, the value of the variable voteable will be "Too young", otherwise the value of voteable will be "Old enough".

Writing conditions

- **Comparison operators** are used in logical statements to determine equality or difference between variables or values.
- Given that `x = 5`, the table below explains the comparison operators.

Operator	Description	Comparing	Result
`==`	equal to	`x == 8`	`false`
		`x == 5`	`true`
		`x == "5"`	`true`
`===`	equal value and equal type	`x === 5`	`true`
		`x === "5"`	`false`
`!=`	not equal	`x != 8`	`true`
`!==`	not equal value or not equal type	`x !== 5`	`false`
		`x !== "5"`	`true`
		`x !== 8`	`true`
`>`	greater than	`x > 8`	`false`
`<`	less than	`x < 8`	`true`
`>=`	greater than or equal to	`x >= 8`	`false`
`<=`	less than or equal to	`x <= 8`	`true`

- Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age < 18) text = "Too young to buy alcohol";
```

Logical operators

- **Logical operators** are used to determine the logic between variables or values.
- Given that `x = 6` and `y = 3`, the table below explains the logical operators:

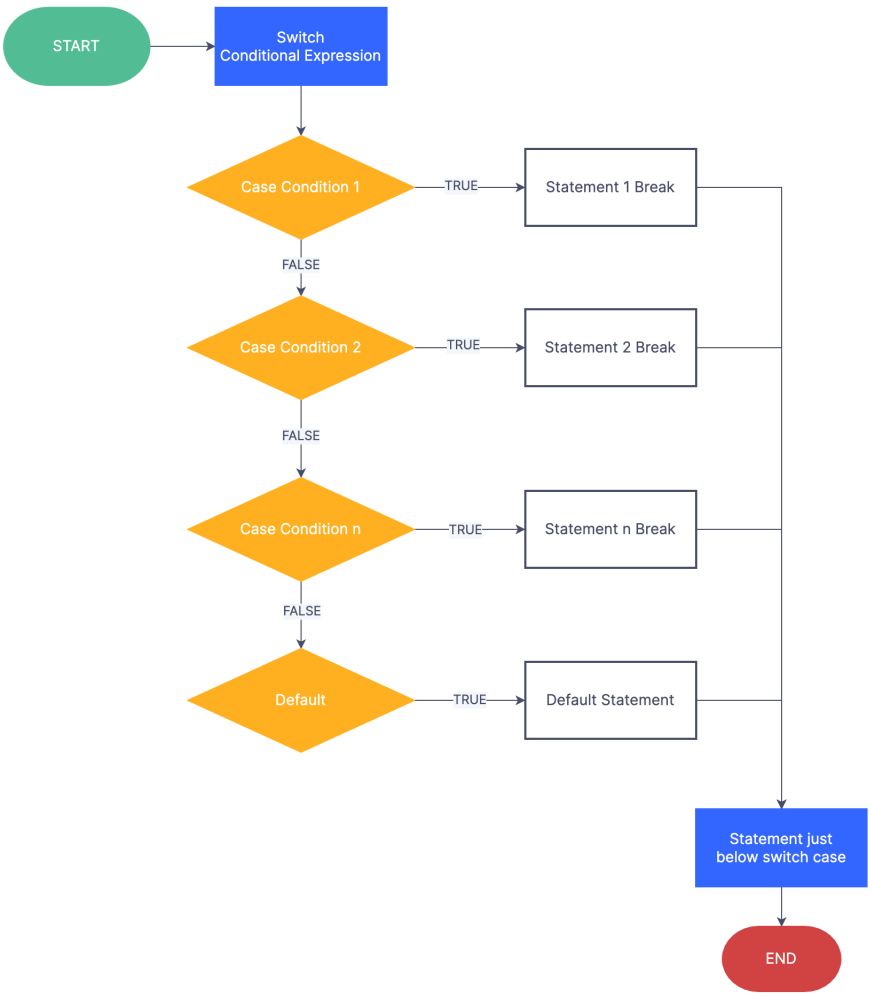
Operator	Description	Example	Result
`&&`	and	`x < 10 && y > 1`	`true`
`||`	or	`x == 5 || y == 5`	`false`
`!`	not	`!x == y)`	`true`

AND		
F	F	F
T	F	F
F	T	F
T	T	T

OR		
F	F	F
T	F	T
F	T	T
T	T	T

NOT	
F	T
T	F

The switch statement



- The `switch` statement is used to perform different actions based on different conditions.
- Use the `switch` statement to select one of many code blocks to be executed.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

- The `switch` expression is evaluated once.
- The value of the expression is compared with the values of each `case`.
  - If there is a match, the associated block of code is executed.
  - If there is no match, the `default` code block is executed.
- Switch cases use strict comparison (`===`).
  - The values must be of the same type to match.
  - A strict comparison can only be `true` if the operands are of the same type.
- When JavaScript reaches a `break` keyword, it breaks out of the `switch` block.
  - This will stop the execution inside the `switch` block.
  - It is not necessary to break the last `case` in a `switch` block. The block breaks (ends) there anyway.
- The `default` keyword specifies the code to run if there is no `case` match.
  - The `default` case does not have to be the last case in a `switch` block.
  - If `default` is not the last `case` in the `switch` block, remember to end the `default` case with a break.
- If multiple cases matches a `case` value, the first `case` is selected.
  - If no matching cases are found, the program continues to the `default` label.
  - If no `default` label is found, the program continues to the statement(s) after the `switch`.

Additional Material

- **Learn more**
  - W3Schools