

JavaScript Essentials

Conditionals



Table of Contents

1. Overview – What is condition ?
2. if...else statement
3. switch statements
4. ternary operator
5. Q&A

Lesson Objectives

- Understand the needs to make decisions and carry out actions accordingly depending on different inputs, scenario
- Understand how conditional statements work in JavaScript
- Able to use if...else, switch, ternary operator with ease to make decisions

Section 1

Overview – What is a condition?

- In any programming language, the code needs to make decisions and carry out actions accordingly depending on different inputs.
- **For example:** in a game, if the player's number of lives is 0, then it's game over.
- In a weather app, if it is being looked at in the morning, show a sunrise graphic; show stars and a moon if it is nighttime.

Overview – What is a condition?



- Conditional plays a big role in any Programming language
- It helps to make decisions and carry out actions accordingly depending on different inputs
- JavaScript provide such conditionals or control flow mechanism

Section 2

if...else statement

- Let's look at by far the most common type of conditional statement you'll use in JavaScript — the humble if...else statement.
- Basic if...else syntax looks like the following in pseudocode:

```
1 | if (condition) {  
2 |     code to run if condition is true  
3 | } else {  
4 |     run some other code instead  
5 | }
```

- You should note that you don't have to include the else and the second curly brace block — the following is also perfectly legal code:

```
1 | if (condition) {  
2 |     code to run if condition is true  
3 | }  
4 |  
5 | run some other code
```

- As a final point, you may sometimes see if...else statements written without the curly braces, in the following shorthand style:

```
1 | if (condition) code to run if condition is true  
2 | else run some other code instead
```

- **Not recommended**

- Example 1: The parent might say "Hey sweetheart! If you help me by going and doing the shopping, I'll give you some extra allowance so you can afford that toy you wanted."

- The last example provided us with **two** choices, or outcomes — but what if we want **more than two**?
- There is a way to chain on extra choices/outcomes to your if...else — using **else if**. Each extra choice requires an additional block to put in between `if() { ... }` and `else { ... }`
- Example 2: Checking weather

- Comparison operators are used to test the conditions inside our conditional statements
 1. `==` and `!=` : test if one value is identical to, or not identical to, another.
 2. `<` and `>` : test if one value is less than or greater than another.
 3. `<=` and `>=` : test if one value is less than or equal to, or greater than or equal to, another.

- It is perfectly OK to put one if...else statement inside another one — to nest them.

```
if (choice == 'sunny') {  
    if (temperature < 37) {  
        para.textContent =  
            'It is nice and sunny outside t  
    } else if (temperature ≥ 37) {  
        para.textContent = 'It is SUPER H  
    }  
} else if (choice == 'rainy') {
```

- If you want to test multiple conditions without writing nested if...else statements, logical operators can help you. When used in conditions, the first two do the following:
 1. && — AND
 2. || — OR

- To give you an **AND** example, the previous example snippet can be rewritten to this:

```
if (choice == 'sunny' && temperature < 37) {  
    para.textContent =  
        'It is nice and sunny outside today. Wear shorts!'  
} else if (choice == 'sunny' && temperature ≥ 37) {  
    para.textContent = 'It is SUPER HOT. Dont go out.';  
} else if (choice == 'rainy') {
```

- Let's look at a quick **OR** example:

```
1  if (iceCreamVanOutside || houseStatus === 'on fire') {  
2      console.log('You should leave the house quickly.');
```

```
3  } else {  
4      console.log('Probably should just stay in then.');
```

```
5  }
```

- The last type of logical operator, **NOT**, expressed by the **!** operator, can be used to **negate** an expression. Let's combine it with **OR** in the above example:

```
1  if (!(iceCreamVanOutside || houseStatus === 'on fire')) {  
2      console.log('Probably should just stay in then.');
```

```
3  } else {  
4      console.log('You should leave the house quickly.');
```

```
5  }
```

- You can combine as many logical statements together as you want, in whatever structure

```
1 | if ((x === 5 || y > 3 || z <= 10) && (loggedIn || userName === 'Steve')) {  
2 |     // run the code  
3 | }
```

- Common mistake (wrong syntax):

```
1 | if (x === 5 || 7 || 10 || 20) {  
2 |     // run my code  
3 | }
```

- This is logically not what we want!
- To make this work you've got to specify a complete test either side of each **OR** operator:

```
1 | if (x == 5 || x == 7 || x == 10 || x == 20) {  
2 |     // run my code  
3 | }
```

- **if...else** is probably the most common type of conditional statement in JavaScript
- Basic syntax: **if ...else**
- You can have as much **else...if** as you need
- You can nest as much **if...else** as you want (**Not recommended**)
- Make use of Logical operators to strike for cleaner and easy to understandab code
- 3 type of logical operators: **AND, OR, NOT**

Section 3

switch statement

- if...else statements do the job of enabling conditional code well, but they are not without their downsides.

```
var field;  
// check field  
if (field === 'id') {  
    // code  
} else if (field === 'name') {  
    // code  
} else if (field === 'age') {  
    // code  
} else if (field === 'address') {  
    // code  
} else if (field === 'clazz') {  
    // code  
} // more else if
```

- In such a case, switch statements are your friend
- It takes a single expression/value as an input,
- Look through a number of choices until they find one that matches that value
- Executing the corresponding code that goes along with it.

- Here's some more pseudocode, to give you an idea:

```
1  switch (expression) {  
2      case choice1:  
3          run this code  
4          break;  
5  
6      case choice2:  
7          run this code instead  
8          break;  
9  
10     // include as many cases as you like  
11  
12     default:  
13         actually, just run this code  
14 }
```

- Convert previous example to switch:

```
var field;  
// check field  
switch (field) {  
    case 'id': {  
        // code  
        break;  
    }  
    case 'name': {  
        // code  
        break;  
    }  
    case 'age': {  
        // code  
        break;  
    }  
    default: {  
        // code  
    }  
}
```

Practice 1: switch statement

- When you get a large number of choices think about **switch**
- Syntax for switch is: `switch (value) { case: }`
- **default** case is optional but it's recommended to include one
- Do not forget the **break** keyword if you don't want your program to behave strangely

Section 4

Ternary operator

- The ternary or conditional operator is a small bit of syntax that tests a condition and returns one value/expression if it is true, and another if it is false
- The pseudocode looks like this:

```
1 | ( condition ) ? run this code : run this code instead
```


Ternary operator – Example

- So let's look at a simple example:

```
var isLoggedIn = false;  
  
var greeting = isLoggedIn ? 'Hello user, you are logged in' : 'Hello stranger';  
greeting;  
"Hello stranger"
```

Practice 2: ternary operator

- Useful in some situations, and can take up a lot less code than an if...else block if you simply have two choices that are chosen between via a true/false condition.
- Is an expression not a statement so it can be anywhere that expects an expression, for example: on the right side of = operator, an item in an array, condition in if...else or switch, in template string and logical operators and much more

Thank you

Q&A

