JavaScript Essentials

Conditionals

Table of Contents

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

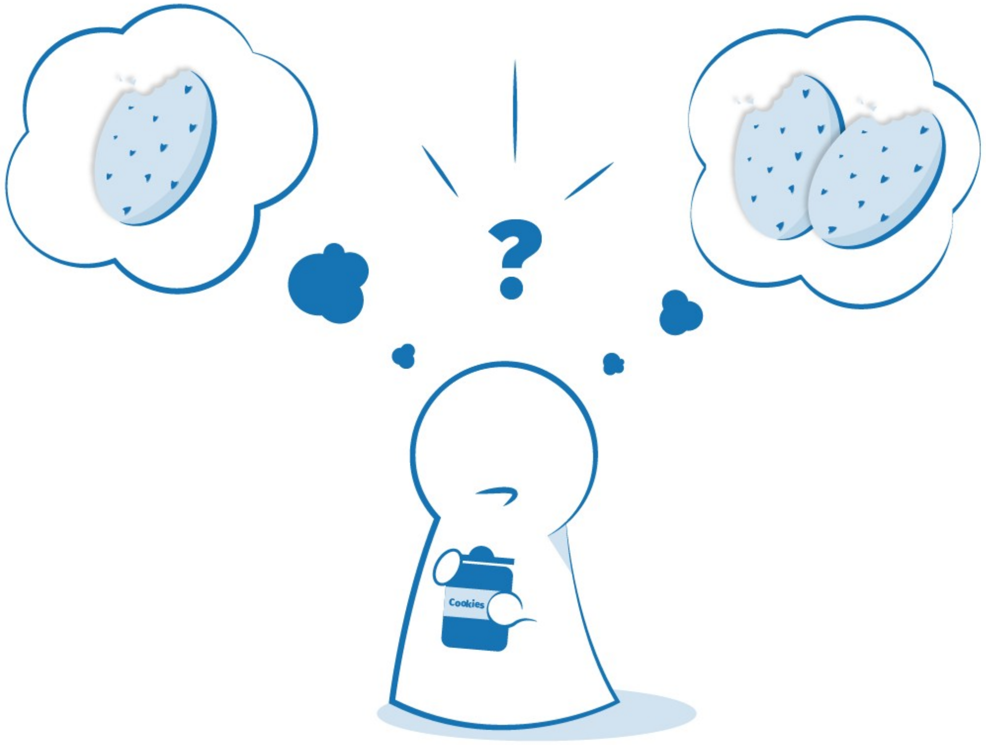
Section 1

Overview – What is a condition?

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?



Overview – Summary

- 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

if…else statement – Basic syntax

- 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:

```
if (condition) {
  code to run if condition is true
} else {
  run some other code instead
}
```

if…else statement – Basic syntax

- 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:

```
if (condition) {
  code to run if condition is true
}

run some other code
```

if…else statement – Basic syntax

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

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

- **Not recommended**

if…else statement – Basic syntax

- 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 { ... } — check out the following more involved example, which could be part of a simple weather forecast application

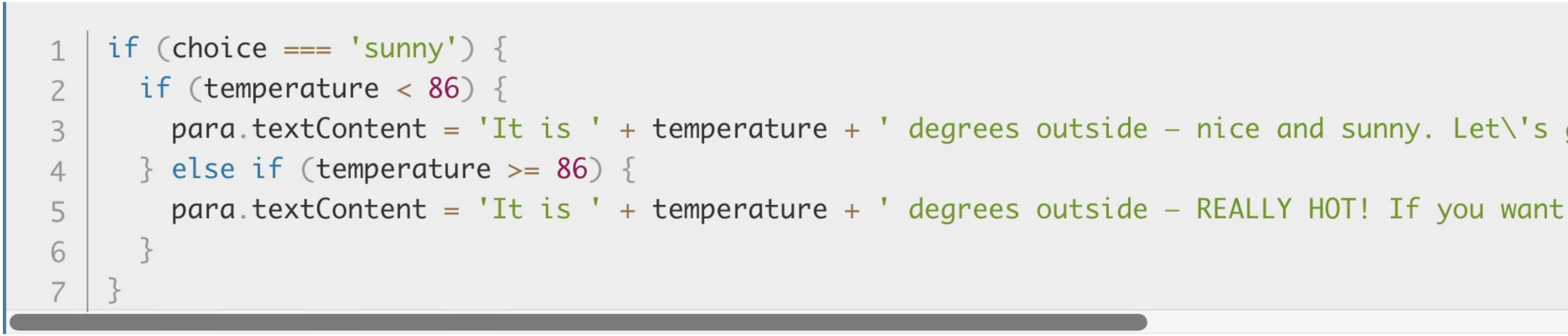
if…else statement – Tip on condition

- Comparison operators are used to test the conditions inside our conditional statements. We first looked at comparison operators back in our Basic math in JavaScript — numbers and operators article. Our choices are:

- === and !== — test if one value is identical to, or not identical to, another.
- < and > — test if one value is less than or greater than another.
- <= and >= — test if one value is less than or equal to, or greater than or equal to, another.

if…else statement – Nesting if…else

- It is perfectly OK to put one if...else statement inside another one — to nest them. For example, we could update our weather forecast application to show a further set of choices depending on what the temperature is:



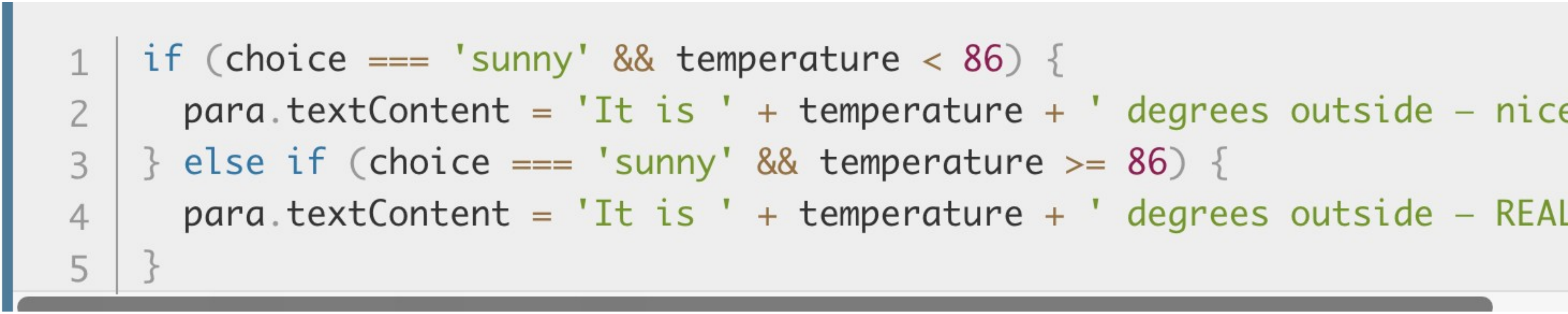
```
if (choice === 'sunny') {
  if (temperature < 86) {
    para.textContent = 'It is ' + temperature + ' degrees outside — nice and sunny. Let\'s
  } else if (temperature >= 86) {
    para.textContent = 'It is ' + temperature + ' degrees outside — REALLY HOT! If you want
  }
}
```

if…else statement – Logical operators

- 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:
- && — AND; allows you to chain together two or more expressions so that all of them have to individually evaluate to true for the whole expression to return true.
- || — OR; allows you to chain together two or more expressions so that one or more of them have to individually evaluate to true for the whole expression to return true.

if…else statement – Logical operators

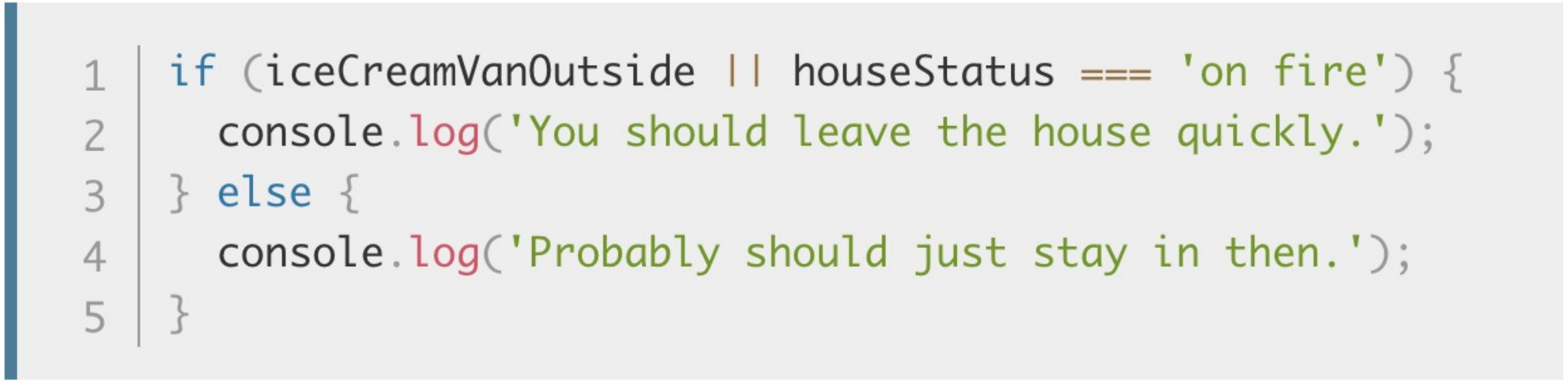
- To give you an AND example, the previous example snippet can be rewritten to this:



```
if (choice === 'sunny' && temperature < 86) {
  para.textContent = 'It is ' + temperature + ' degrees outside — nice
} else if (choice === 'sunny' && temperature >= 86) {
  para.textContent = 'It is ' + temperature + ' degrees outside — REAL
}
```

if…else statement – Logical operators

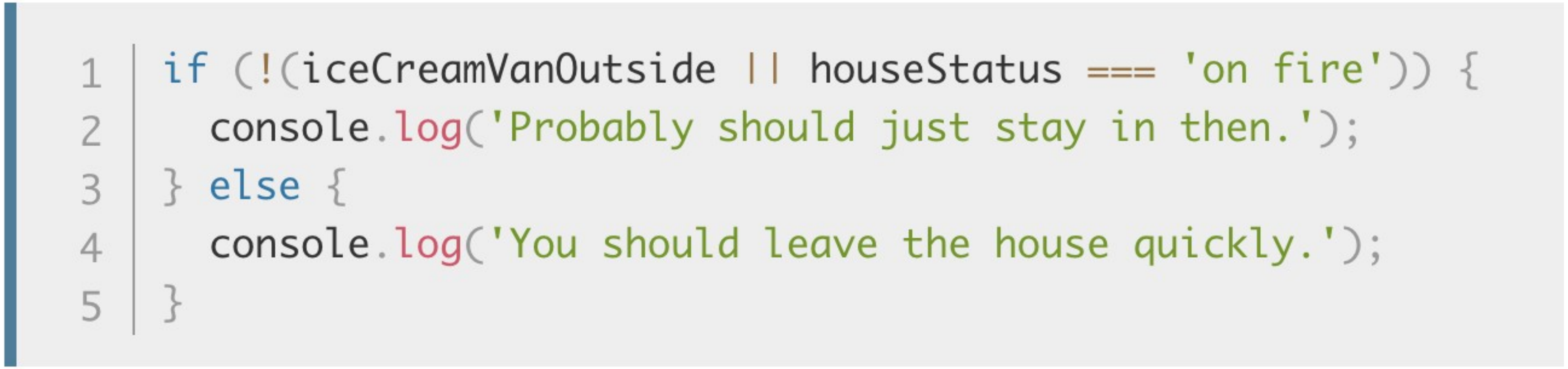
- Let's look at a quick OR example:



```
if (iceCreamVanOutside || houseStatus === 'on fire') {
  console.log('You should leave the house quickly.');
} else {
  console.log('Probably should just stay in then.');
}
```

if…else statement – Logical operators

- 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:



```
if (!(iceCreamVanOutside || houseStatus === 'on fire')) {
  console.log('Probably should just stay in then.');
} else {
  console.log('You should leave the house quickly.');
}
```

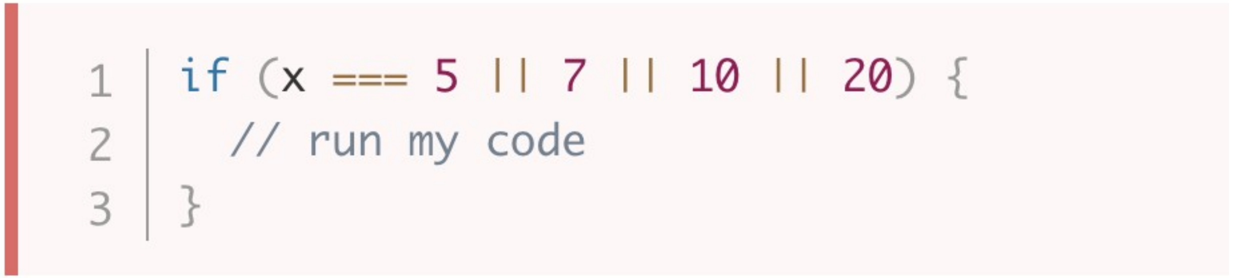
if…else statement – Logical operators

- You can combine as many logical statements together as you want, in whatever structure. The following example executes the code inside only if both OR statements return true, meaning that the overall AND statement will return true:

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

if…else statement – Logical operators

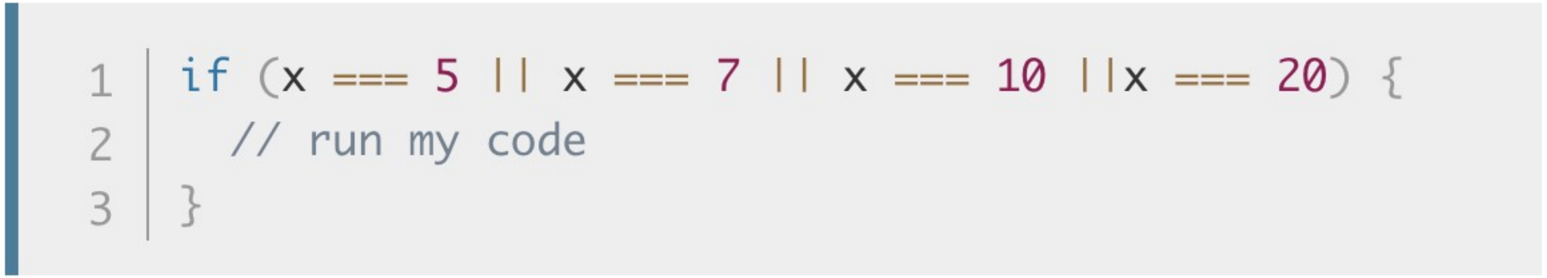
- A common mistake when using the logical OR operator in conditional statements is to try to state the variable whose value you are checking once, and then give a list of values it could be to return true, separated by || (OR) operators. For example:



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

if…else statement – Logical operators

- 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:



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

if…else - Summary

- if…else is probably the most common type of conditional statement in JavaScript
- Basic syntax for if…else is easy to remember
- 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 understand code
- 3 type of logical operators: AND, OR, NOT

Section 3

switch statement

switch statement – Problem with if…else

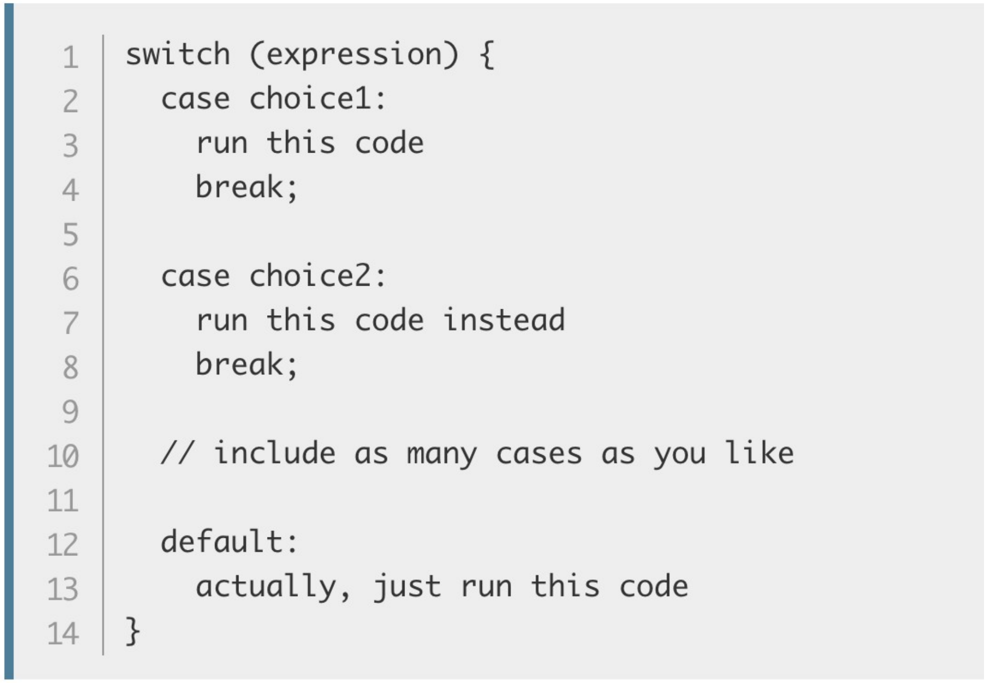
- if...else statements do the job of enabling conditional code well, but they are not without their downsides.
- They are mainly good for cases where you've got a couple of choices, and each one requires a reasonable amount of code to be run, and/or the conditions are complex (for example, multiple logical operators).
- For cases where you just want to set a variable to a certain choice of value or print out a particular statement depending on a condition, the syntax can be a bit cumbersome, especially if you've got a large number of choices.

switch statement – For the rescue

- In such a case, switch statements are your friend
- Take 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.

switch statement – For the rescue

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



```
switch (expression) {
  case choice1:
    run this code
    break;

  case choice2:
    run this code instead
    break;

  // include as many cases as you like

  default:
    actually, just run this code
}
```

switch statement - Practice Time

Practice switch statement

switch statement - Summary

- 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** keywork if you don't want your program to behave strangely

Section 4

Ternary operator

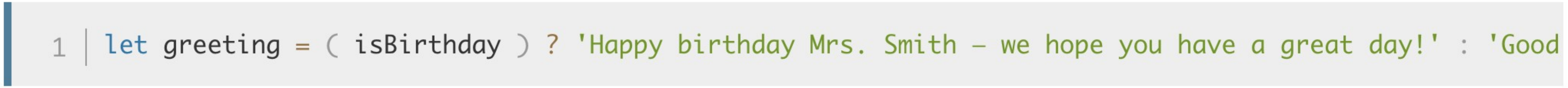
Ternary operator – Syntax

- 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 — this can be 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.
- The pseudocode looks like this:

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

Ternary operator – Example

- So let's look at a simple example:



```
let greeting = ( isBirthday ) ? 'Happy birthday Mrs. Smith — we hope you have a great day!' : 'Good
```

Ternary operator - Practice Time

Practice ternary operator

Ternary operator - Summary

- 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 an array, condition in if…else or switch, in template string and logical operators and much more

Thank you

Q&A