JavaScript For Web

Week 1, Lecture 5 - JavaScript Fundamentals: Logical Operators and Conditionals

Instructor: Jason Xu

Today's Overview

- Boolean variables, truthy, and falsy values
- Common used logical operators: `||` (OR), `&&` (AND) and `!` (NOT)
- The `if` statement
- The `else` statement
- THe `else if` statement
- Conditional operator `?`
- The conditional statement with logical operators
- Tutorial: Assigment 1, Exercise 3

Boolean variables

The boolean type has only two values: `true` and `false`.

```
let nameFieldChecked = true // yes, name field is checked
let ageFieldChecked = false // no, age field is not checked
```

What is **truthy** and **falsy** values?

In JavaScript, a **truthy** value is a value that is considered true when encountered in a Boolean context/operation. All values are truthy unless they are defined as **falsy**. That is, all values are truthy except:

- `false`
- `0`, `-0`, `0n` (The number/bigint zero)
- `""`, ``` `` ```, `''` (empty string)
- `null`, `undefined`, `NaN` (unknown, empty value)
- `document.all` (The only falsy build-in object in JavaScript)

Logical operators: `||` OR

The “OR” operator is represented with two vertical line symbols:

```
let isCCCStudent = true, isSFUStudent = false

let isBurnabyStudent = isCCCStudent || isSFUStudent
```

When `||` (OR) is used with boolean values, there are four possible logical combinations(at least one is `true`):

```
console.log( true  || true )   // true
console.log( false || true )   // true
console.log( true  || false )  // true
console.log( false || false )  // false
```

`||` (OR) finds the first truthy value

```
console.log( 1 || 0 ) // 1 (1 is truthy)
console.log( null || 1 ) // 1 (1 is the first truthy value)
console.log( null || 0 || 1 ) // 1 (the first truthy value)
console.log( undefined || null || 0 ) // 0 (all falsy, returns the last value)
```

Logical operators: `&&` AND

The AND operator is represented with two ampersands `&&`:

```
let isAssignmentComplete = true, isAttendedLectures = true

let passTheCourse = isAssignmentComplete && isAttendedLectures
```

When `&&` (AND) is used with boolean values, there are four possible logical combinations(all should be true):

```
console.log( true  && true )   // true
console.log( false && true )   // false
console.log( true  && false )  // false
console.log( false && false )  // false
```

Logical operators: `&&` AND continued

`&&` AND finds the first falsy value

- If the first operand is truthy, AND returns the second operand:

```
console.log( 1 && 0 ) // 0
console.log( 1 && 5 ) // 5
```

- If the first operand is falsy, AND returns it. The second operand is ignored

```
console.log( null && 5 ) // null
console.log( 0 && "no matter what" ) // 0
```

- When all values are truthy, the last value is returned

```
console.log( 1 && 2 && 3 ) // 3
```

Logical operators: `!` NOT

The boolean NOT operator is represented with an exclamation sign `!`

```
let isSomething = true

let isNotSomething = !isSomething
```

The operator accepts a single argument and does the following:

- Converts the operand to boolean type: `true` / `false`.
- Returns the inverse value.

```
console.log( !true ) // false
console.log( !0 ) // true
```

The precedence of NOT `!` is the **highest** of all logical operators, so it always executes first, then `&&` and `||`.

Convert a value to boolean type

A double NOT `!!` is sometimes used for converting a value to boolean type:

```
console.log( !!"non-empty string" ) // true
console.log( !!null ) // false
```

That is, the first NOT converts the value to boolean and returns the inverse, and the second NOT inverses it again. In the end, we have a plain value-to-boolean conversion.

There's a little more verbose way to do the same thing – a built-in `Boolean` function:

```
console.log( Boolean("non-empty string") ) // true
console.log( Boolean(null) ) // false
```

JavaScript Conditional

Sometimes, we need to perform different actions based on different conditions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements and operators:

- Use `if` to specify a block of code to be executed, if a specified condition is true.
- Use `else` to specify a block of code to be executed, if the same condition is false.
- Use `else if` to specify a new condition to test, if the first condition is false.
- Use `?` to specify a value to be assigned, if a specified condition is true.

The `if` Statement

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

The syntax is:

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

Note that `if` is in lowercase letters. Uppercase letters (`If` or `IF`) will generate a JavaScript error.

For example:

Let's print a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {
  console.log("Good day")
}
```

The `else` Statement

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

The syntax is:

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

For example:

If the hour is less than 18, print a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
  console.log("Good day")
} else {
  console.log("Good evening")
}
```

The `else if` statement

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

The syntax is:

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

For example:

If hour is less than 10:00, print a "Good morning" greeting, if not, but time is less than 20:00, print a "Good day" greeting, otherwise a "Good evening":

```
if (hour < 10) {
  console.log("Good morning")
} else if (hour < 20) {
  console.log("Good day")
} else {
  console.log("Good evening")
}
```

The Conditional Operator `?`

Let's go back to the previous senario: If the hour is less than 18, print a "Good day" greeting, otherwise "Good evening"

Sometime we assign a variable depending on a condition:

```
const hour = ...
let greetings

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

console.log(greetings)
```

The so-called “conditional” or “question mark” operator lets us do that in a shorter and simpler way.

```
let result = condition ? value1 : value2
```

The `condition` is evaluated: if it’s truthy then `value1` is assigned, otherwise – `value2`. Here is the shorter version of greetings:

```
const hour = ...
let greetings = (hour < 18) ? "Good day" : "Good evening"
console.log(greetings)
```

The Conditional Statement with Logical Operators

Most of the time, logical operators (`||`, `&&`, `!`) are used in conditional statements (`if`, `else`, `else if`) to perform different actions based on more complex conditions.

For example:

```
let hour = 9

if (hour < 10 || hour > 18) {
  console.log( 'The office is closed.' )
} else {
  console.log( 'The office is opened.')
}
```

OR

```
let hour = 9

if (hour > 10 && hour < 18) {
  console.log( 'The office is opened.' )
} else {
  console.log( 'The office is closed.')
}
```

The Conditional Statement with Logical Operators continued

We can even pass more conditions, for example:

```
let hour = 9, isWeekend = true

if (hour < 10 || hour > 18 || isWeekend ) {
  console.log( 'The office is closed.' )
} else {
  console.log( 'The office is opened.')
}
```

OR

```
let hour = 9

if (hour > 10 && hour < 18 && !isWeekend ) {
  console.log( 'The office is opened.' )
} else {
  console.log( 'The office is closed.')
}
```

Don’t replace `if` with `||` or `&&`

Sometimes, people use the AND `&&` operator as a "shorter way to write `if`". For example:

```
let x = 1
console.log((x > 0) && ( 'x is greater than zero!' ))
```

Although, the variant with `&&` appears shorter, `if` is more obvious and tends to be a little bit more readable. So we recommend using every construct for its purpose: use `if` if we want `if` and use `&&` if we want `AND`.

```
let x = 1

if(x > 0) {
  console.log('x is greater than zero!')
}
```

Summary

- Truthy and falsy value
- Logical Operators: `&&` AND, `||` OR, `!` NOT
- Conditional Statements/Operator: `if`, `else`, `else if`, `?`