COURSEWARE

The ECMAScript 6 Specification

Contents

Overview

ECMAScript 6 (ES6) is scripting-language specification that was incorporated into JavaScript, bringing with it a series of new features, methods and an additional operator, amongst other things.

In this section we will be covering the following features that have been added with ECMAScript 6:

- Keywords: `let` and `const`
- Arrow Functions
- The `class` Keyword
- Default Parameters
- The Exponentiation Operator
- New Global Methods
- New `Array` Methods
- New `Number` Attributes and Methods

Tutorial

Keywords: 'let' and 'const'

The `let` keyword should be considered a safer alternative to the `var` declaration.

A `let` variable respects its value within a specific scope, allowing for better garbage-collection.

▶ Example:

The `const` is a keyword applicable to variables, allowing the firm declaration of a value within its respective scope.

A `const` value cannot be modified once set - they are commonly allocated to values which follow a specific constant or standard, such as pi.

▶ Example:

Preventing hoisting

It is important to note that the `let` and `const` keywords prevent the natural hoisting behaviour of JavaScript.

▶ Example:

In the above example JavaScript knows all variables in the code due to hoisting.
Our variable `y` exists because it is grouped with declared variables before the code is executed (hoisted).

This means it has no value, but is recognised by the system as being a valid variable, despite being an uninitialised variable.

Our variable `y` only receives its value after its initialisation, as seen below (whereby it is assigned the value 5).

▶ Example:

In the above example, our `let y` only exists after its declaration, thus `y` does not exist - running this code results in an Uncaught Reference Error.

Arrow Functions

The new arrow function `=>` allows for a more concise implementation which removes the need for a `return` statement.

It is important to remember:

- Arrow functions do not have their own `this` value, as such you should avoid using them in this way within object/class methods.
- Like `let` and `const`, Arrow functions are not hoisted - remember to utilise them after they've been defined!
- Consider using `const` to define a value that stores an *arrow function* as function behaviour rarely changes after its declaration.

▶ ES6 Example:
▶ ES5 Example:

JavaScript Classes

ES6 allows JavaScript to create class structures that allow for the definition of more complex types, which can both include their own methods and be instantiated.

Declarations of a class always begin with the keyword `Class` followed by a name. A traditional class has a constructor, a function that defines operations to perform on the initialisation of the class. A constructor belongs to the class and functions within a class are often referred to as methods - *just like functions* - methods can accept parameter arguments.

▶ Example:

Default Parameter Values

But what do we mean when we 'Default' a parameter value, basically we can declare the reference value but give it an optional value should
the user wish to omit a parameter entry.

In cases where a method is called, you can instead define the method parameters to specify a default value. This means that you can substitute a missing parameter with a pre-determined value.

▶ Example:

The Exponentiation Operator

The '**' operator allows you to declare an exponent, this means we can now use 'n**x' rather than `Math.pow(n, x);`.

▶ ES6 Example:
▶ ES5 Example:

New Global Methods

As of ES6 there are two new globally accessible methods:

- `isFinite()`
- `isNaN()`

These methods are good methods to check types and their values in cases of mathematical application, this can further reinforce verification and validation techniques on front-end input fields.

▶ Example:

Array Methods

There are two new methods associated with the Array object:

Array.find(*function*)

The `Array.find()` method can take a function parameter to define its finding criteria - the first value within the array that meets that criteria will be returned.

▶ Example:

Array.findIndex(*function*)

The `Array.findIndex()` method is similar to `Array.find()`, instead of returning the value, it returns the index position.

Quick reminder: *Array index values begin from 0*

▶ Example

Number Attributes and Methods

As part of ECMAScript 5, the `Number` object has received some new constant variables and methods that relate to the floor and ceiling values of the `INTEGER` values.
These are particularly easy to observe and can be easily reviewed in JavaScript.

To quickly test this in Chrome you can enter these values into the browser console to see their respective values.
to see their respective values.

Try running the following attributes and methods of the 'Number' object and compare results.

▶ Example
Why might we use these values?

The argument here is we might not, however, constants and variables do offer fantastic points of reference for mathematically driven algorithms.
Perhaps certain processes might challenge the limit of a data type value; particularly if you wish to simply validate the data type.

Test Driven Development:

It's always a possibility that a method may change in the future so be aware that system reliance to constant values can change in syntax, don't forget to consider test driven development and ensure testing reviews these values!

Exercises

Here are some tasks you can try out to demonstrate your knowledge and become more familiar with the ECMAscript 6 features that are part of Javascript.

`let` and `const` Tasks

▶ Tasks

Arrow Functions and Exponential Operator

▶ Tasks

JavaScript Classes

▶ Task