**ECMAScript-history**

ECMAScript is name of the official standard with JavaScript being the most well-known implementation of the standard. ActionScript (Macromedia) and JScript (Microsoft) are examples of other implementations.

ECMAScript 1, the first version of the JavaScript language standard, was released in June 1997. Exactly a year later, ECMAScript 2 was released, which contained only minor changes to keep in sync with a parallel ISO standard for JavaScript.

18 months later in December 1999, ECMAScript 3 was released, introducing a lot of the popular features of JavaScript that we now take for granted like regular expressions, try/catch exception handling, and formatting for numeric output.

ECMA Script5, the incremental upgrade to ES3, was released in December 2009, more than a decade after ES3. It is the version of JavaScript that is fully supported in all the browsers in use today except for Internet Explorer 8.

It added several enhancements to the standard library such as:

* JSON parsing/serialization support
* Array prototype methods (map, foreach)
* Methos for listing properties (object.keys).

There were also syntactic updates like:

* Allowances for dangling commas at the end of lists or object definitions
* Allowances for reserved words (new, for) as object properties1.

It introduced strict mode, which made JavaScript a cleaner language by forbidding some features, performing additional runtime checks, and throwing more exceptions.

**ECMA Script6**

The latest, 6th edition of the language, ECMAScript 2015 (or ES6) is probably the most significant update since the first version in 1997. The main goal of the latest release has been to provide a better support for creating larger applications and libraries. This means a more mature syntax, new shortcuts to make coding easier, and also new methods, keywords, data types, and many other enhancements.

**1.Let Keyword**

ES6 introduces the new let keyword that allows us to declare local variables in the scope of a block, such as a statement, an expression, or a(n inner) function.

**2.New const Keyword**

The new **const** keyword makes it possible to declare constants, also known as immutable variables, to which we cannot reassign new content later.

Immutable variables are not always fully immutable in ECMAScript 6 though, as if a constant holds an object, we can later change the value of its properties and methods.

**3. Arrow functions**

ECMAScript 6 facilitates how we write **anonymous functions**, as we can completely omit the function keyword. We only need to use the new syntax for **arrow functions**, named after the => arrow sign (fat arrow), that provides us with a great shortcut.

**4. New spread Operator**

The new spread operator is marked with 3 dots (…), and we can use it to sign the place of multiple expected items. One of the most common use cases of the spread operator is inserting the elements of an array into another array

**5. Default Values for Parameters & New Rest Parameters**

Good news, that in ECMAScript 6 we can add default values to the parameters of a function. This means that if we don’t pass in arguments later in the function call, the default parameters will be used. In ES5 the default values of parameters are always set to undefined, so the new possibility to set them to whatever we want is definitely a great enhancement of the language.

**6.New for ..of Statement**

With the help of the new for..of loop we can iterate over arrays or other iterable objects easily. Along with the new for..of statement, ECMAScript 6 introduces two new iterable objects too, map for key/value maps, and set for collections of unique values that can also be primitive values and object references. When we use the for..of statement, the code inside the block is executed for each element of the iterable object.

**7.Template Literals**

ECMAScript 6 provides us with a new alternative for string concatenation. **Template literals** allow us to easily create templates in which we can embed different values to any spot we want.

**8.Classes**

ES6 introduces JavaScript classes that are built upon the existing prototype-based inheritance. The new syntax makes it more straightforward to create objects, take leverage of inheritance, and reuse code. It will also make it easier for beginners arriving from other programming languages to understand how JavaScript works.

#### 9. Modules

The goal for ECMAScript 6 modules was to create a format that both users of CommonJS and of AMD are happy with:

* Similar to CommonJS, they have a compact syntax, a preference for single exports and support for cyclic dependencies.
* Similar to AMD, they have direct support for asynchronous loading and configurable module loading.

Being built into the language allows ES6 modules to go beyond CommonJS and AMD (details are explained later):

* Their syntax is even more compact than CommonJS’s.
* Their structure can be statically analyzed (for static checking, optimization, etc.).
* Their support for cyclic dependencies is better than CommonJS’s.

The ES6 module standard has two parts:

* Declarative syntax (for importing and exporting)
* Programmatic loader API: to configure how modules are loaded and to conditionally load modules.

**10. Loads of New Methods**

ECMAScript 6 introduces many new methods for the existing String Prototype, Array Object, Array Prototype, and Math Object. The new methods can significantly improve the way how we can manipulate these entities.