**Important points:**

* Javascript is a scripting language that was introduced to make web pages alive and be interactive with the user.
* Javascript at its development phase was called Mocha
* ES is ECMAScript and ECMA is European Computer Manufacturers Association.
* **JSON.parse()** is used to convert the text into a JavaScript object.
* **JSON.stringify()** to convert it into a string.
* "use strict" defines that the JavaScript code should be executed in "strict mode".

Difference between ES5, ES6 and ES7.

**ES5:** New features with respective to previous:

(refered <https://www.greycampus.com/blog/programming/java-script-versions>)

* 'USE STRICT' directive:

eg:

'use strict'

x = 10; //leads to an error

* New methods in an array:

--> **isArray() :** This method checks if the object is an array or not and returns the result in true or false.

eg:

let arr = [1,2,3,4,5];

let a = 5;

console.log(Array.isArray(arr)); //returns true

console.log(Array.isArray(a)); //returns false

--> **forEach():** It executes the function for every element found in the array.

eg:

let arr = [1,2,3,4,5];

arr.forEach(function(element) {

console.log(element); //prints all the elements of the array

})

--> **map():** The map() method creates a new array by mapping every element of the array (on which the map is used).

eg:

let arr = [2,4,6,8,0];

let newarr = arr.map(function(element) {

return element \* element;

});

console.log(newarr);

// prints an array that has squares of the first array

// i.e. [ 4, 16, 36, 64, 0 ]

--> **filter():** It creates a new array that contains elements which got filtered by making array elements pass some condition.Eg:

var arr = ['hockey', 'cricket', 'basketball', 'running'];

const result = arr.filter(arr => arr.length > 6);

console.log( result)

// Output: [ 'cricket', 'basketball', 'running' ]

--> **reduce():** It applies a function to each element in an array and reduces the array to a single element.

--> **reduceRight():** It is quite same to reduce method, except for the traversal that happens from right to left.

--> **every():** if every element satisfies then returns true if at least one element doesn’t satisfy the condition then returns false.

const arr = [1, 2, 3, 4, 5];

let ret = arr.every(val => {

return val > 3

})

console.log(ret); // returns false

--> **some():** if at least one element that gets satisfied with the condition and if found returns true and if none satisfies then false.

--> **indexOf():** Returns the index of the first match in an array. If not found then returns -1.

--> **lastIndexOf():** Same as indexOf but checks and returns from the other end of an array.

* JSON Support:

--> **parse():** it parses a json string that is like an object.

eg:

let jsonString = '{"a" : 1,"b" : 2}';

let obj = JSON.parse(jsonString);

console.log(obj.b); // results is 2.

--> **stringify():** This method converts an object to a JSON string.

console.log(JSON.stringify({x:5, y:6}));

* New methods in a Date:

--> **now()**

--> **valueOf()**

* Getters and Setters:

The get method returns the value of a variable, and the set method sets the value of the variable.

* Property methods:

--> **Object.defineProperty():** This method lets the user define the property of an object and/or change its value.

**ES6:** New features:

(refered <https://www.w3schools.com/js/js_es6.asp>, <https://www.greycampus.com/blog/programming/java-script-versions>)

* JavaScript let

Declaring a variable using 'let' : allows to declare a variable with block scope, like the body of an if statement or for loop .

Eg:

<script>

var a = 5;

// Here a is 5

{

let a = 10;

// Here a is 10

}

// Here a is 5

**Advantages:**

- minimize the scope of your variables.

- can access it only after it was declared. Until then, the variable is considered to be in the 'Temporal Dead Zone'.

* JavaScript const

The const statement allow to declare a constant (a JavaScript variable with a constant value).

Eg:

var x = 10;

// Here x is 10

{

const x = 2;

// Here x is 2

}

// Here x is 10

* Arrow Functions

allows a short syntax for writing function expressions.

Eg:

// ES5

var x = function(x,y) {

return x \* y;

}

// ES6

var x = (x,y) => x\*y;

or

const x = (x,y) => {

return x \* y

};

* Classes

A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.

Eg:

class Car {

constructor(brand) {

this.carname = brand;

}

}

mycar = new Car("Ford"); //creating an object

--> Ability to create and and inherit classes

eg:

class MyComponent extends React.Components {

}

* Default Parameter Values

ES6 allows function parameters to have default values.

Eg:

function add(x, y = 5) {

// y is 5, if not passes or undefined

return x + y;

}

add(5); //will return 10,

add(5,10); // will return 15

* Array.find()

The find() method returns the value of the first array element that passes a test function.

Eg:

var numbers = [2, 4, 6, 8, 10];

var first = numbers.find(number);

function number(value) {

return value > 4;

} // it returns 6

when, var first = numbers.findIndex(number)

// it will return 2 (i.e. index of 6)

* New Number Properties

ES6 added the following properties to the Number object:

: **EPSILON**

: **MIN\_SAFE\_INTEGER:** constant represents the minimum safe integer in JavaScript (-(253 - 1)).

: **MAX\_SAFE\_INTEGER:** constant represents the maximum safe integer in JavaScript (253 - 1).

* New Number Methods

ES6 added 2 new methods to the Number object:

**Number.isInteger():** method returns true if the argument is an integer.

**Number.isSafeInteger():** method returns true if the argument is a safe integer.

* New Global Methods

-- **isFinite()** method returns false if the argument is Infinity or NaN.

-- **isNaN()** method returns true if the argument is NaN. Otherwise it returns false.

* For..Of

is an alternative for both for...in and forEach() and loops iterable data structures like Arrays, Maps, Sets, and strings.

Eg:

const arr = ['one', 'two', 'three'];

for(const a of arr) {

console.log(a);

}

// one two three (result)

* Rest Operator

Rest Operator is used to handle function parameters. It uses three dots as its syntax (i.e …).

eg:

function fun(a, ...b) {

console.log('a: ' +a+ 'b:' +b);

}

fun(1,2,3,4);

// Output: a: 1b:2,3,4

* Spread Operator

It is used to split the contents of an array.

* Destructuring:

It helps in unpacking values from an array or an object.

* Template Literals/ Strings:

It allows embedded expressions, which makes print statements easy.

* Promises

It is used to handle Asynchronous Programming in a more elegant way.

eg:

let myPromise = new Promise((resolve, reject) => {

let theDecider = true;

if(theDecider) {

resolve("success");

}

else {

reject("failure");

}

});

myPromise.then(res => {

console.log('res: ', res);

});

myPromise.catch(err => {

console.log('err: ', err);

});

* Other features:

Set, WeakSet, WeakMap, Generators, Symbols, Unicode, Modules, Proxies, Built-Ins, Binary and Octal, Reflect, Tail Call Optimization

ES7: New features

(refered <https://www.greycampus.com/blog/programming/java-script-versions>

* Exponentiation Operator

raises the first operand to the power of the second operand.

Eg: var x = 5;

var z = x \*\* 2; // result is 25

or console.log(3\*\*2); // returns 9

* Array includes

Returns true if an array includes a value, if not returns false.

eg : array.includes(myItem) // true or false

var animals = ['cat', 'rat', 'bat'];

console.log(animals.includes('cat')); // returns true

ES5 vs ES6:

<https://www.javatpoint.com/es5-vs-es6>

