***1.What is Ecmascript in Js?***

## *New Features in ES6*

* [The let keyword](https://www.w3schools.com/js/js_es6.asp#mark_let)
* [The const keyword](https://www.w3schools.com/js/js_es6.asp#mark_const)
* [Arrow Functions](https://www.w3schools.com/js/js_es6.asp#mark_arrow)
* [For/of](https://www.w3schools.com/js/js_es6.asp#mark_forof)
* [Map Objects](https://www.w3schools.com/js/js_es6.asp#mark_map)
* [Set Objects](https://www.w3schools.com/js/js_es6.asp#mark_set)
* [Classes](https://www.w3schools.com/js/js_es6.asp#mark_class)
* [Promises](https://www.w3schools.com/js/js_es6.asp#mark_promise)
* [Symbol](https://www.w3schools.com/js/js_es6.asp#mark_symbol)
* [Default Parameters](https://www.w3schools.com/js/js_es6.asp#mark_param)
* [Function Rest Parameter](https://www.w3schools.com/js/js_es6.asp#mark_rest)
* [String. includes()](https://www.w3schools.com/js/js_es6.asp#mark_includes)
* [String. startsWith()](https://www.w3schools.com/js/js_es6.asp#mark_startswith)
* [String. endsWith()](https://www.w3schools.com/js/js_es6.asp#mark_endswith)
* [Array. from()](https://www.w3schools.com/js/js_es6.asp#mark_array_from)
* [Array keys()](https://www.w3schools.com/js/js_es6.asp#mark_array_keys)
* [Array find()](https://www.w3schools.com/js/js_es6.asp#mark_array_find)
* [Array find Index()](https://www.w3schools.com/js/js_es6.asp#mark_array_findIndex)
* [New Math Methods](https://www.w3schools.com/js/js_es6.asp#mark_math_methods)
* [New Number Properties](https://www.w3schools.com/js/js_es6.asp#mark_number_properties)
* [New Number Methods](https://www.w3schools.com/js/js_es6.asp#mark_number_methods)
* [New Global Methods](https://www.w3schools.com/js/js_es6.asp#mark_global_methods)
* [Object entries](https://www.w3schools.com/js/js_es6.asp#mark_entries)
* [JavaScript Modules](https://www.w3schools.com/js/js_es6.asp#mark_modules)

# ***2. Difference between var, let and const keywords in JavaScript?***

**var keyword in JavaScript:** The *var* is the oldest keyword to declare a variable in JavaScript.

**Scope:**Global scoped or function scoped. The scope of the*var* keyword is the global or function scope. It means variables defined outside the function can be accessed globally, and variables defined inside a particular function can be accessed within the function.

[**let**](https://www.geeksforgeeks.org/javascript-let/)**keyword in JavaScript:** The *let*keyword is an improved version of the *var* keyword.

**Scope:**[**block scoped:**](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) The scope of a *let*variable is only block scoped. It can’t be accessible outside the particular block ({block}). Let’s see the below example.

[**const**](https://www.geeksforgeeks.org/javascript-const/)**keyword in JavaScript:** The *const* keyword has all the properties that are the same as the *let* keyword, except the user cannot update it.

**Scope:**[block scoped:](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) When users declare a *const* variable, they need to initialize it, otherwise, it returns an error. The user cannot update the *const*variable once it is declared.

## *ES5 Features*

* ["use strict"](https://www.w3schools.com/js/js_es5.asp#mark_use_strict)
* [String[*number*] access](https://www.w3schools.com/js/js_es5.asp#mark_string_property_access)
* [Multiline strings](https://www.w3schools.com/js/js_es5.asp#mark_string_multiline)
* [String. trim()](https://www.w3schools.com/js/js_es5.asp#mark_string_trim)
* [Array. isArray()](https://www.w3schools.com/js/js_es5.asp#mark_isarray)
* [Array forEach()](https://www.w3schools.com/js/js_es5.asp#mark_array_foreach)
* [Array map()](https://www.w3schools.com/js/js_es5.asp#mark_array_map)
* [Array filter()](https://www.w3schools.com/js/js_es5.asp#mark_array_filter)
* [Array reduce()](https://www.w3schools.com/js/js_es5.asp#mark_array_reduce)
* [Array reduceRight()](https://www.w3schools.com/js/js_es5.asp#mark_array_reduceright)
* [Array every()](https://www.w3schools.com/js/js_es5.asp#mark_array_every)
* [Array some()](https://www.w3schools.com/js/js_es5.asp#mark_array_some)
* [Array indexOf()](https://www.w3schools.com/js/js_es5.asp#mark_array_indexof)
* [Array lastIndexOf()](https://www.w3schools.com/js/js_es5.asp#mark_array_lastindexof)
* [JSON.parse()](https://www.w3schools.com/js/js_es5.asp#mark_json_parse)
* [JSON.stringify()](https://www.w3schools.com/js/js_es5.asp#mark_json_stringify)
* [Date.now()](https://www.w3schools.com/js/js_es5.asp#mark_date_now)
* [Date toISOString()](https://www.w3schools.com/js/js_es5.asp#mark_date_toiso)
* [Date toJSON()](https://www.w3schools.com/js/js_es5.asp#mark_date_tojson)
* [Property getters and setters](https://www.w3schools.com/js/js_es5.asp#mark_getter)
* [Reserved words as property names](https://www.w3schools.com/js/js_es5.asp#mark_reserved)
* [Object methods](https://www.w3schools.com/js/js_es5.asp#mark_object_methods)
* [Object defineProperty()](https://www.w3schools.com/js/js_es5.asp#mark_object_define_property)
* [Function bind()](https://www.w3schools.com/js/js_es5.asp#mark_function_bind)
* [Trailing commas](https://www.w3schools.com/js/js_es5.asp#mark_trailing_commas)

***3.What is Spread operator, Rest operator, and default parameter?***

In [Javascript](https://www.geeksforgeeks.org/javascript-tutorial/)both spread operator and rest parameter have the same syntax which is three dots **(…)**. Even though they have the same syntax they differ in functions.

[**Spread operator:**](https://www.geeksforgeeks.org/javascript-spread-operator/)The [spread](https://www.geeksforgeeks.org/javascript-spread-operator/) operator helps us expand an iterable such as an array where multiple arguments are needed, it also helps to expand the object expressions. In cases where we require all the elements of an iterable or object to help us achieve a task, we use a spread operator.

var\_name =[...iterable];

[**Rest operator:**](https://www.geeksforgeeks.org/javascript-rest-operator/)The [rest](https://www.geeksforgeeks.org/javascript-rest-operator/)parameter is converse to the spread operator. while spread operator expands elements of an iterable, rest operator compresses them. It collects several elements. In functions when we require to pass arguments but were not sure how many we have to pass, the rest parameter makes it easier.

**Note:**There must be only one rest operator in javascript functions.

**Syntax:**

function function\_name(...arguments) {

statements;

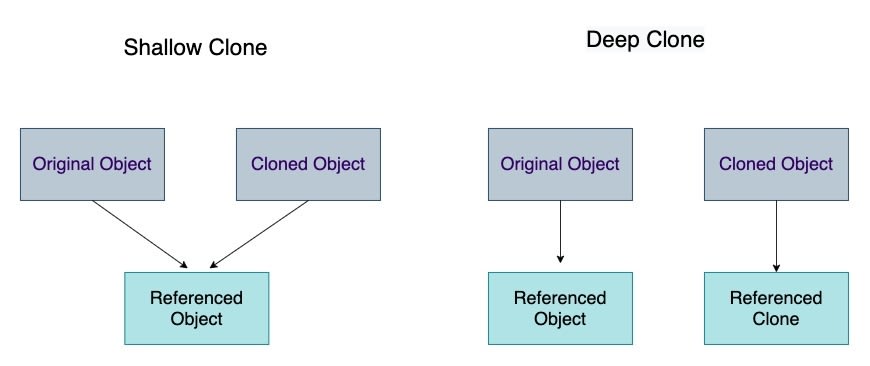
}

***4.What is Deep copy and Shallow copy in js?***

A deep copy means that all of the values of the new variable are copied and disconnected from the original variable. A shallow copy means that certain (sub-)values are still connected to the original variable.

**Shallow copy:**In the case of shallow copy when we copy the original object into the clone object then the clone object has the copy of the memory address of the original object. Means both points to the same memory address.

Both original object and cloned object internally point to the same referenced object. Since they point to the same memory address so if we changed the cloned object then changes would be reflected back to the original object because they point to the same memory address.



***What is the difference between a deep copy and a shallow copy***?

1. In Shallow copy, a copy of the original object is stored and only the reference address is finally copied.
2. In Deep copy, the copy of the original object and the repetitive copies both are stored.

**var** obj1 = {

        id: 1,

        company: "GFG"

};

**var** obj2 = {...obj1};

**var** obj2 = Object. assign({}, obj1);

    obj2.id = 2;

    console.log(obj1.id);

   console.log(obj2.id);

This is called shallow copy. A shallow copy means once we make changes in the clone object it will be reflected back to the original object as well.

But in the case of **deep copy,**changing the value of the cloned object will not reflect into the original object, because both are pointing to different reference objects. Because the original object has its own reference object and after cloning, the cloned object has its own referenced object. Both are different.

**DeepCopy:**

**var** obj1 = {

        id: 1,

        company: "GFG",

        details:

        {

            employee\_no: 10

        }

    };

**var** obj2 = JSON. parse(JSON. Stringify (obj1))

    obj2. details. employee\_no = 20;

    console. log(obj1.details.employee\_no);

    console. log(obj2.details.employee\_no);

In this case, we have a nested object and we are changing the employee\_no with the help of obj2. And because of it obj1 employee\_no also gets changed. So Object. assign doesn’t work perfectly in the case of nested object.

**JSON. stringify and JSON. parse:**It works perfectly in the case of nested objects. Unlike Object. assign operator it doesn’t change our original object when we make any changes in the clone object.

***5.What is promise, callback function, async await in jS?***

***Promise:***  A Promise is a JavaScript object that links producing code and consuming code

***Callback:*** A callback is a function passed as an argument to another function.

# **Asynchronous & Await :** Functions running in parallel with other functions are called asynchronous

A good example is JavaScript setTimeout()

***6.what is the difference between callback and promise in javascript?***

**A callback function is passed as an argument to another function whereas Promise is something that is achieved or completed in the future**. In JavaScript, a promise is an object and we use the promise constructor to initialize a promise.

7.***What is event bubbling and event capturing in js?***

When we clicked on the div with the child as its id, we should get the output as ‘child’ on our console. But unexpectedly, we are receiving a different output even we have not clicked on divs with parent and grandparent as their id. The concept of event bubbling comes into the picture. The child div lies inside the parent div as well as in the grandparent div. So, when the child div clicked, we indirectly clicked on both parent div and grandparent div. Thus, propagation is moving from inside to outside in the DOM or we can say events are getting bubble up.

Therefore, the process of propagating from the closest element to the farthest away element in the DOM (Document Object Modal) is called event bubbling.

8***. higher order function in javaScript?***

Basically, **a function which takes another function as an argument or returns a function** is known as a higher order function. Let's deep dive a bit to see both types of implementation, that is: Passing a function as an argument to another function. Returning a function from another function.

9.***Why we using Call, apply and bind methods?***

**Call () Method:**The call method is basically used to invoke the function with different**this** object.

**Syntax:**

object. objectMethod.call( objectInstance, arguments )

const obj1 = {

        firstName: "First\_name",

        lastName: "Last\_name"

    };

    const obj2 = {

        firstName: "Sachin",

        lastName: "Tendulkar"

    };

**function** printName() {

        console.log(**this**.firstName + " " + **this**.lastName);

    }

     printName.apply(obj2);

**Apply () Method:** Apply is very similar to the call function. The only difference is that in apply you can pass an array as an argument list

const obj1 = {

        firstName: "First\_name",

        lastName: "Last\_name"

    };

    const obj2 = {

        firstName: "Sachin",

        lastName: "Tendulkar"

    };

**function** printName(profession, country) {

        console.log(**this**.firstName + " "

            + **this**.lastName + " " +

            profession + " " + country);

    }

    printName.apply(obj2, ["Cricketer", "India"]);

**Bind () Method:** Bind method is similar to the call method