**Top 60+ ES6 Interview Questions | 2023**

By [Great Learning Team](https://www.mygreatlearning.com/blog/author/greatlearning/) Updated on Nov 16, 2022 11889

Are you an aspiring Front end Developer? Then, ES6 is an important scripting language you would have come across. In this article, we have compiled a comprehensive list of ES6 interview questions and answers that will come in handy at the time of need. Once you are prepared with the questions we mentioned in our list, you will be ready to get into numerous front-end development job roles like junior Developer, [front-end developer](https://www.mygreatlearning.com/blog/front-end-developer-guide/), javascript developer and more.

Before getting started, let see what is the history of es6,

When JavaScript initially arrived in the programming world, the name was chosen for marketing considerations. At the time, Java was growing more and more popular worldwide. Later, ECMA (European Computer Manufacturers Association) received Javascript and its specification for standardisation.

Later, it was known as ECMAScript of ES. The first edition was published in June 1997. The sixth edition of the language, referred to as ES6, was later changed to ECMAScript 2015. Iterators, for/of loops, arrow functions, typed arrays, promises, reflection, and many other new features were included in this release.

Launched in June 2016, ES6. With the release of the eighth version in June 2017, new capabilities like concurrency and atomics as well as syntactic integration with promises were implemented.

Before continuing, let’s gain some understanding of ES6.

* About 3295 businesses employ ES6 in their IT stacks. These include Hepsiburada, StackShare, Slack, Discord, Glovo, KAVAK, Tech Stack, and HENNGE K.K.
* In the USA, a JavaScript developer who is familiar with ES6 can expect to make an average pay of $104,270 per year. And with just 1-2 years of expertise, one can earn about $111,018 annually.

**Frequently asked top 10 ES6 interview questions**

Great Learning has prepared a list of the top 10 ES6 interview questions that will help you during your interview

* [What is es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-es6)
* [How many parts are there in an es6 module?](https://www.mygreatlearning.com/blog/es6-interview-questions/#how-many-parts-are-there-in-an-es6-module)
* [What is a class in es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-a-class-in-es6)
* [What is the arrow function in es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-the-arrow-function-in-es6)
* [What is export default es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-export-default-es6)
* [Which keywords can be used to implement inheritance in es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#which-keywords-can-be-used-to-implement-inheritance-in-es6)
* [What is destructuring in es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-destructuring-in-es6)
* [What is Event propagation, capturing, bubbling?](https://www.mygreatlearning.com/blog/es6-interview-questions/#hat-is-event-propagation-capturing-bubbling)
* [What is the difference between es5 and es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#what-is-the-difference-between-es5-and-es6)
* [How to define a function in es6?](https://www.mygreatlearning.com/blog/es6-interview-questions/#ow-to-define-a-function-in-es6)
* [ES6 Interview Questions FAQs](https://www.mygreatlearning.com/blog/es6-interview-questions/#es6-interview-questions-faqs)

**1. What is es6?**

ES6 is Ecmascript 2015,6th new version of[Javascript](https://www.mygreatlearning.com/blog/javascript-tutorial/) programming language. ES6 is considered to be the major upgrade to javascript since ES5 was introduced in 2009. With the introduction of ES6, whole new features are added to the language like Classes, Modules, Object Manipulation etc. to name a few.

**2. How many parts are there in an es6 module?**

Modules were part of the Javascript community for a very long time but they are not built into the language, they were implemented by the libraries. With the introduction of ES6, modules were implemented in the language. Modules refer to the reusable pieces of code within our application or system. They are stored in one file. To use them, there are two parts to it, Default Export and Named Export. Similarly, we can import the modules to the current file.

**3. What is a class in es6?**

Classes in JavaScript are introduced from ES6 in 2015. In OOPS programming, a class is a template for creating objects, providing initial values to variables by providing them constructor and implementation functions or methods. For Eg:

Class Order {

constructor(id){

this.id=id;

}

getId(){

console.log('The Order Id of Your Order ${this.id}')

}

}

Order o1 =new Order('5gf123');

o1.getId();

Output: The Order Id of Your Order 5gf123

**4. What is the arrow function in es6?**

An arrow function expression can be used as an alternative to the old JavaScript function expression, but is limited and cannot be used all the time. Arrow functions are lexically bound to this keyword, unlike the normal functions which are dynamically binding.

Arrow functions cannot be used as a method and constructors.

For eg: const x= () =>{ console.log("Hello Guys !")};

**5. What is export default es6?**

The export statement is used to export functions, objects, and variables, etc. to other JavaScript modules with the help of the import statement.  
There are two types of ways to export  
1.) Named Export  
2.) Default Export: is when we want to export any single, variable, function or object, etc. For Eg: In Test.js file var x1=10; export default x1; In Com.js file we want to import the x import y1 from ‘./Test.js’ here y1 is importing x1 from Test.js

**6. Which keywords can be used to implement inheritance in es6?**

To Implement inheritance in ES6, extend keyword is used to implement inheritance.In the earlier versions of Javascript , there is no concept of classes but with the introduction of the ES6 , Pure Object Oriented features were added to the language .  
For Example to implement Inheritance in Javascript.

class Animal {

constructor(legs) {

this.legs = legs;

}

walk() {

console.log('Mamals can ' + this.legs + ' legs');

}

}

class Bird extends Animal {

constructor(legs) {

super(legs);

}

fly() {

console.log('flying');

}

}

let bird = new Bird(2);

bird.walk();

bird.fly();

**7. What is destructuring in es6?**

Destructuring is a great way to “destructure” or separate the values of arrays or objects into variables. Destructuring also works with complex functions that have many parameters, default values, etc.  
For Eg ;

let animal =['dog','cat','rabbit'];

let [d,c,r] = arr; //destructuring

console.log(d);

**8. What is Event propagation, capturing, bubbling?**

When an event occurs on a DOM, that event does not entirely occur  
on one element. In the Bubbling Phase, the event bubbles up or it goes to its parent,  
to its grandparents, to its grandparent’s parent until it reaches all the way to the window while in the Capturing  
Phase the event starts from the window down to the element that triggered the event or the event target.  
  
Event Propagation has three phases.

* **Capturing Phase**– the event starts from the window and then goes down to every element until it reaches the target element.
* **Target Phase –** the event has reached the target element.
* **Bubbling Phase –**the event bubbles up from the target element and then goes up every element until it reaches the window.

**9. What is the difference between es5 and es6?**

ES6 and ES5 are both Versions of JavaScript introduced by ECMAScript in 2015 while the later in 2009. Both ES5 and ES6 have some similar points but at the same time, there are differences also Some of the features are:

**Variables:** Let and const Till ES5, there was only one to define the variables by using the var keyword but with the advent of ES6, let and const keyword are also introduced in the language. var is function scope while the later are block Scope.

function(){

var x=5;

}

let a =10;

const m =16;

**Destructuring:** Destructuring is great way to “destructure” or separate the values of arrays or objects into variables .

For Eg:

let animal =['dog','cat','rabbit'];

let [d,c,r] = arr; //destructuring

console.log(d);

**Defining Objects**:

var boy = "Ram";

var marks = 90;

var class = 12;

var gen = "male";

// ES5 var object1 = { boy:boy,marks:marks,class:class,gen:gen };

// ES6 var object2 = { boy,marks,class,gen };

**String:** Interpolation Earlier in ES5, we used to concatenate the two strings with the help of a concatenation operator but with String Operator it becomes a lot handier to do it. For eg:

var x='The'+'order'+id;

let x +`The Order ${id}`;

**10. How to define a function in es6?**

<!--td {border: 1px solid #ccc;}br {mso-data-placement:same-cell;}-->

In ES6 , Arrow Functions are Introduced

// Arrow Function Break Down

// 1. Remove the word function and place arrow between the argument and opening body bracket

(c1) => {

return c1 + 100;

}

2. Remove the body brackets and word "return"

(y) => y + 100;

3. Remove the argument parentheses

a => a + 100;

**11. How to import css in es6?**

This program helps you to understand “how to import CSS in es6”. CSS files can be imported to JavaScript Files by using the import command as follows.

index.css File

.container {

padding: 20px;

}

Main.js

import a from './index.css'"

**12. What are the new features introduced in ES6?**

Some Of The Major New Features of the ES6 are

1. Variables :Let and const :Till ES5 , there was only one to define the variables by using the var keyword but with the advent of ES6 , let and const keyword are also intrtoduced in the

language.var is function scope while the later are block Scope

function(){

var x=5;

}

let a =10;

const m =16;

2.Destructuring

Destructuring is great way to “destructure” or seperate the values of arrays or objects into variables .

For Eg ;

let animal =['dog','cat','rabbit'];

let [d,c,r] = arr; //destructuring

console.log(d);

3.Defining Objects

var boy = ""Ram"";

var marks = 90;

var class = 12;

var gen = ""male"";

// ES5

var object1 = { boy:boy,marks:marks,class:class,gen:gen };

// ES6

var object2 = { boy,marks,class,gen };

4.String Interpolation

Earlier in ES5, we used to concatenate the two strings with the help of a concatenation operator but with String Operator it becomes a lot handier to do it

For eg:

var x='The'+'order'+id;

let x +`The Order ${id}`;

5.Arrow Functions

// Arrow Function Break Down

// 1. Remove the word ""function"" and place arrow between the argument and opening body bracket

(a) => {

return a + 100;

}

// 2. Remove the body brackets and word ""return"" -- the return is implied.

(a) => a + 100;

// 3. Remove the argument parentheses

a => a + 100;"

**13. Define let and const keywords.**

“Let and Const are introduced in ES6.Earlier , variables in JS are declared with var Keyword var was function scoped where let and const is block scope var can be redeclared and updated but let cannot be redecLared but can be updated as

let order = 50;

let order = 60;

// In the console error will be like:

Uncaught SyntaxError: Identifier 'points' has already been declared

But This Is Possible:

let order = 50;

order = 60;

Scope of Let

let x=10;

if(x>1){

let x=2;

}

on calling x we will get 10

Const are the variables which cannot be declared or updated .If once a value is declAred it cannot be updated

const key = 'fegf123';

// Then try to redeclare it:

key = 'dwed1234'

// I get the following error:

Uncaught TypeError: Assignment to constant variable."

**14. What is Functional Programming?**

Functional Programming is a type of programming in which we build our applications with functions using expressions that calculates a value without mutating or changing the arguments that are passed to it. JavaScript Array has a map, filter, and reduce methods which are the most famous functions in functional programming because they don’t mutate or change the array which makes these functions pure and which are features of a Functional Programming Language.

**Array.Map Method**: The map method creates a new array with the results of calling a provided callback function on every element in the array.

const words = ["Functional", "Procedural", "Object-Oriented"];

const wordsLength = words.map(word => word.length);

**Array.filter:** The filter method creates a new array with all elements that pass the test in the callback function.  
let numbers={1,2,3,4};

const evenNumbers = numbers.filter(x => x%2 );

**15. Give an example of an Arrow function in ES6? List down its advantages.**

An arrow function expression can be used as an alternative to the old javascript function expression, but is limited and cannot be used all the time. Arrow functions are lexically bound to this keyword, unlike the normal functions which are dynamically bound.  
Arrow functions cannot be used as a method and constructors

Arrow syntax binds this to the surrounding that means they are lexically scoped than normal functions which are dynamically scoped  
The syntax gives a return when there is no body block, resulting in simpler code in some cases  
Last but not least, => is shorter and simpler than function, although stylistic issues are often subjective Advantages of **Arrow functions are listed below:-**  
1.) Arrow Functions do not have a binding with this, i.e. arrow functions do not have their own this.  
To understand this in a more simple manner, let’s take an example:

let ty={

x :""1"",

random: () =>{

console.log(x); //do not have binding with this

},

random2(){

console.log(x); //have binding with this

}

};

ty.random();

ty.random2();

Output:

undefined

1

2.) Another feature is that regular functions have its arguments , but arrow functions do not have

for eg:

let y=function(){

console.log(argument);

}

new y=(1,2,3);

output:

Arguments {0:1,1:2,2:3}

let y=() =>{

console.log(argument);

}

new y(1,2,3);

Type Error y is not defined

3.) Arrow functions allow us to return where there is no implicit return

let y =() => 1;

**16. Discuss the spread operator in ES6 with an example.**

The spread operator is introduced in JavaScript ES6. It takes an array or iterable and expands it into individual elements.  
It is also used to make copies of JS objects as shown in the below example  
For Eg:

let a=[1,2,3]

let b=[4,5,6 ,...a]

console.log(b) // 4,5,6,1,2,3

Another Use of Spread Operator is that after using this it makes code concise and increases its code readability.

**17. What are the template literals in ES6?**

Template literals are introduced in ES6 to overcome the problem with string concatenation in our programs. Now we have to use ` character instead of ‘ or ” quotes, to produce a new string.

console.log(`Ram likes to eat mango`)

MultiLine

console.log(`Ram likes to eat mango\nHe is a good Boy`)

Expressions

const pi=3.14

console.log(`The radius of Circle is ${pi}\*2\*2`)

The ${} syntax is used to put an expression in it and it will produce the value.

**18. What do you understand by the Generator function?**

In ES6, a new kind of function is introduced which is different from the regular functions. Unlike regular functions, Generators functions can stop midway  
and then it can continue from where it was paused  
We can define Generatares functions as:-

function\* gen() {

console.log('1st');

yield 1;

console.log('2nd');

yield 2;

}

We can define generator function by adding \* after function and yield statement pauses the execution and returns the value  
  
we can call this function as  
let g= gen()  
console.log(g) // this would give generator object  
  
This Generator Object returns another object with two properties done and value. Generator is iterable

let g1=gen.next()

console.log(g1) //invoked 1st time

{ value: 1, done: false }

g1=gen.next()

console.log(g1) //invoked 2nd time

{ value: 2, done: false }

g1=gen.next() //{ value: undefined, done: true }

console.log(g1)

**19. Why are functions are called first-class objects?**

Functions in JavaScript are First-class Objects because they are treated as any other value in the language.

* They can be assigned to variables.
* They can be properties of an object which are called methods, they can be an item in the array.
* They can be passed as arguments to a function, and they can be returned as values of a function.

**20. Discuss the Rest parameter in ES6 with an example.**

Rest parameters are introduced in ES6 so that we can gather any number of arguments into an array.  
For Better Understanding, let’s understand with this an example

Rest parameters are introduced in ES6 so that we can gather any number of arguments into an array.

For Better Understanding, let's understand with this an example

Without Rest parameter

function add(a,b) {

let sum = 0;

sum=a+b;

return sum

}

With rest arguments, with this, we can now accept any number of arguments

function add(...a) {

let sum = 0;

for (let a1 of a) sum += a;

return sum

}

add(5) // returns 5

add(1,2) // returns 3

add(1, 2, 3, 4, 5) // returns 15

**21. What do you mean by IIFE?**

IIFE’s are Immediately-Invoked Function Expressions. They are the functions that are called immediately after being defined.  
We can make any function IIFE by wrapping it in parenthesis and following pair of parentheses at the end

(function() {

// Code

})()

With Arrow function

(() => {

// Code

})()

We use IIFE to Aliasing global variables, Create private variables and functions, and Asynchronous functions in loops.

**22. What are the default parameters?**

Default function parameters are used to initialize parameters with default values if no value or undefined is passed to the variable

function setOrder (id ,name=Demo){ //here we have passed demo

console.log(`The OrderId ${id} is of ${name} `)

}

**23. Discuss the for…of loop.**

for…of is used to iterate over the elements of an array. It does so nicely and shortly, without the need of additional variables .

For example:

const animal = ['cat', 'dog'];

for (const a of animal) {<br>console.log(animal);<br>}

Output:<br>// 'cat'<br>// 'dog'

**24. Discuss the for…in loop.**

for ..in loop is uesd to iterate over the properties of object,a key is assigned to the key variable. The loop continues for all object properties

const Employee = {

name: 'Ram',

EmpID: 71222,

age: 45

}

// using for...in

for ( let k in Employee ) {

// display the properties

console.log(`${k} => ${Employee[k]}`);

}

Output:

Ram 71222 45

**25. Define Map.**

The map is a feature introduced in ES6, is a collection of elements where each element is stored as a Key-value pair. It can hold both objects and primitive  
values. On iterating on the map object, it returns the key-value pairs.

We can create Map like  
var map = new Map([[1 , 2], [2 ,3 ] ,[4, 5]]);  
  
console.log(map);

Output would be like :

// 1 => 2

// 2 => 3

// 4 -> 5

Some of the Important Methods of Map are :  
**Map.prototype.set() –** It adds the key and value to the Map Object.  
Syntax:

|  |  |
| --- | --- |
| 1  2 | const map = new Map();  map.set('id', 'qwert'); |

**Map.prototype.has()-**It return a boolean value depending on whether the specified key is present or not

let map =new Map()

map.set('class','12th')

console.log(map.has('class')) //true

**Map.prototype.get()-** It returns the value of the corresponding key

let map = new Map();

map.set('name', 'Ram');

console.log(map.has('class')) //true

console.log(map.get('name')) //Ram

**Map.prototype.delete() - It deletes** both the key as well as a value from the map.

const map = new Map();

map.set('name', 'Ram');

console.log(map.delete('name'));

// expected result: true

// (true indicates successful removal)

**26. Define set.**

Sets are also introduced in ES6, stores a collection of Objects which contain unique values. In sets, no value can repeat itself. To create a new empty Set, the syntax is as follows:  
  
let set = new Set();  
The Set object provides the following useful methods:

* add(value) – appends a new element. It returns the Set object
* clear() – removes all elements from the Set object.
* delete(value) – deletes an element specified by the value.
* entries()– returns a new Iterator that contains an array

**27. What do you understand by Weakmap?**

In Es6 ,Weakmap is introduced which is a type of collection similar to map.In this the values are stored as key-value pairs where keys are weakly referenced .  
Weakmap object allows the keys of object only.  
For Eg:

const w=new WeakMap(); //Weakmap is created

const o={}; object1

const o1={}

const o3={};

The WeakMap methods are :  
**1. set()-** It set the given key to the weakmap  
eg;  
for the given above weakmap  
const w=new WeakMap(); //Weakmap is created  
const o={}; object1  
const o1={}  
const o3={};  
w.set(o,”cool”)  
w.set(o1,”Great”)  
w.set(o2,”Learning”)  
**2. has()-**It tells wheteher the given key is present in weakmap or not  
Eg:  
const w=new WeakMap(); //Weakmap is created  
const o={}; object1  
const o1={}  
const o3={};  
w.set(o,”cool”)  
w.set(o1,”Great”)  
w.set(o2,”Learning”)  
w.has(o1) //true  
**3. get()-**It returns the value of the given key  
Eg:  
const w=new WeakMap();  
const o={}; object1  
const o1={}  
const o3={};  
w.set(o,”cool”)  
w.set(o1,”Great”)  
w.set(o2,”Learning”)  
w.get(o) //cool is returned

**4. delete()-**It deletes the element from the weakmap  
const w=new WeakMap();  
const o={}; object1  
const o1={}  
const o3={};  
w.set(o,”cool”)  
w.set(o1,”Great”)  
w.set(o2,”Learning”)  
w.delete(o)

**28. Explain Promises in ES6.**

A promise is an object that returns a value which will be returned in future, but not now.Because of this feature, the promise is very well-suited for handling asynchronous operations.  
A JavaScript Promise object can be:  
Pending-(working), the result is undefined.  
Fulfilled- the result is a value.  
Rejected-the result is an error object.  
settled-fulfilled or rejected

to create a promise:

let completed = true;

let greatLearning = new Promise(function (resolve, reject) {

if (completed) {

resolve("GreatLearning course is completed.");

} else {

reject("Complete the Previous Assignments");

}

});

**29. What do you understand by Weakset?**

Weakset is a collection of objects, adapts the same properties of a set but does not contain duplicate values.  
the way to define the weakset are:-  
const w=new WeakSet();  
Important methods of weakset are:-

* add(value)-a new object is appended with the given value to the weakset.
* w.add(value)
* delete(value)-Deletes the value from the WeakSet collection.
* w.delete(value)
* has(value)-Returns true if the value is present in the WeakSet Collection, false otherwise.
* w.has(value)
* length()-Returns the length of weakSetObject
* w.length()

**30. What do you understand by Callback and Callback hell in JavaScript?**

Callback is a function that is passed an Argument to another function  
For Eg :  
  
Callback hell is also known as the Pyramid of Doom. Callback hell is nothing but calling another callback inside the callback. Here the callbacks nested are related to each other.  
For Example in the example below set timeout is being called inside another set timeout and so on…This is a small example of callback hell

let numbers = [1, 2, 4, 7, 3, 5, 6];

function isEvenNumber(x) {

return x % 2;

}

const evenNumbers = numbers.filter(isEvenNumber);

console.log(evenNumbers); // [ 2,4,6]

so using callback,

let evenNumbers = numbers.filter(function(x) {

return x % 2;

});

console.log(evenNumbers); // [ 2,4,6 ]

const getRol1No = () => {

setTimeout( () => {

console.log('API getting roll no');

let roll\_no = [1,2,3,4,5];

console.log(roll\_no);

setTimeout( (rollno) => {

const biodata = {

name : 'vinod', age : 26

}

console.log(` My roll no is ${rollno). My name is ${ biodata.name} and I am ${biodata.age} years old`);

setTimeout((name) =>{

biodata.gender

},2000,biodata.name)

},2000, roll\_no[1])

},2000):

}

**31. What do you understand by the term Hoisting in JavaScript?**

Hoisting in Javascript is done during compilation, it is the process to move the variables, and functions declarations to the top of the scope  
Please be careful that only declaration gets hoisted NOT the initialization. When you execute a code, the JavaScript engine V8 or spider monkey etc. creates the global execution context.  
The global execution context has two phases: creation and execution.  
During the creation phase, the JavaScript engine moves the variable and function declarations to the top of your code. This feature is known as hoisting in JavaScript.  
**There are broadly two main types of hoisting  
1. Variable Hoisting-**the JavaScript engine moves the variable declarations to the top of the script

console.log(x); // undefined

var x = 1;

In execution phase, this would look like something

var x

console.log(x);

x = 1;

**2. Function hoisting**

let x = 20,

y = 10;

let result = add(x,y);

console.log(result);

function add(a, b){

return a + b;

}

During the execution phase, it would look something like this function add(a, b){

return a + b;

}

let x = 20,

y = 10;

let result = add(x,y);

console.log(result);

**32. What is AJaX?**

AJAX stands for Asynchronous JavaScript and XML. It is a group of related technologies used to display data asynchronously.  
What this means is that we can send data to the server and get data from the server without reloading the web page.  
  
Technologies use for AJAX.

* HTML – web page structure
* CSS – the styling for the webpage
* JavaScript – the behaviour of the webpage and updates to the DOM
* XMLHttpRequest API – used to send and retrieve data from the server
* PHP, Python, Nodejs – Some Server-Side language

**33. List the new Array methods introduced in ES6?**

Some of the Array Methods Introduced in ES6 are:-  
1.concat()- this is used to merge two or more arrays together.  
eg:  
let a1=[1,2,3]  
let b1=[4,5,6]  
console.log(a1.concat(b1)); // output would be [1,2,3,4,5,6]  
2. every()-This function checks whether the elements satisfy the condition or not. If it satisfies the condition it returns true and if a condition is not satisfied, it returns false and does not check for remaining values.  
  
let a =[1,2,4];  
a.every(function(num){  
return num<10;  
})  
  
returns true  
  
3.filter()- This method creates a new array that pass the given criteria and condition.It does not execute on empty array and also does not change the original array  
let words = [‘aaa’, ‘abcd’, ‘a1a1’, ‘bcd123’, ‘ghjytr’, ‘fgfegf’];  
console.log(words.filter(function(str){  
return str.length>4;  
}))  
[“bcd123”, “ghjytr”, “fgfegf”]  
  
4.find()- It returns the values of element as soon as the given condition satisfies.If the condition is satisfied , it returns from the array,return the value and  
does not check for remaining values.It does not change the original array.  
let a = [3, 10, 18, 20];  
console.log(a.find(function(b){  
return b==18}))  
// returns 18  
  
5. forEach()-This method used to iterate through its elements of the Arrays.  
let b =[1,2,3,4];  
b.forEach(function(i){console.log(i)})  
  
6. Array.from() -The Array.from() method returns an Array from the given array , in case of string it return character of the string .  
It accepts three parameters as object ,mapFunction and this which is optional  
For eg ;  
let a=”I am a good Boy”;  
console.log(Array.from(a)) //It will print [“I”, ” “, “a”, “m”, ” “, “a”, ” “, “g”, “o”, “o”, “d”, ” “, “B”, “o”, “y”]  
  
  
7. indexOf()-This method is used to find the index of the first occurrence of the search element provided as the argument to the method.  
let b=[4,5,6,7]  
console.log(b.indexOf(4)) //returns 0

**34. What are the new String methods introduced in ES6?**

Some of the String Method Introduced in ES6 are:-  
1.startsWith – This method accepts two parameters one is queryString and another is position. This method returns either true if the string starts with the provided string  
from the specified position, otherwise false.  
let str = “GreatLearning”;  
  
console.log(str.startsWith(“Great”))  
  
2.endsWith()-This method accepts two parameters one is queryString and another is position. This method returns either true if the string ends with the provided  
string for a specified length, otherwise false.  
let str = “GreatLearning”;  
  
console.log(str.endsWith(“Learning”))  
  
3. includes-This method accepts two parameters one is queryString and another is the position. This method will true if the string is present  
let str = “GreatLearning”;  
  
console.log(str.includes(“ern”));  
  
4. repeat(count)-This method accepts a single argument which is an integer value that represents the number of times the string is to be repeated.  
let str = “GreatLearning”;  
console.log(str.repeat(2));

**35. Define Webpack.**

Webpack is a tool that compiles JavaScript modules, also known as a module bundler.  
It is used to generate a single file from a large file that runs in-app. Webpack can run Babel which transpires to ES5. It can also be used to transpile CoffeeScript to JavaScript,  
can also run a development web server.  
Webpack is not only used on the front end, it’s also useful in backend Node.js development as well.

Predecessors of webpack are as follows

- Grunt

- Broccoli

- Gulp

The web pack mode is introduced in web pack 4 which sets the environment on which the web pack works. It can be development or production defaults to production.

Development mode:

- builds very fast

- is less optimized than production

- does not remove comments

- provides more detailed error messages and suggestions

Production mode

- is slower to build, since it needs to generate a more optimized bundle.

- The resulting JavaScript file is smaller in size, as it removes many things that are not needed in production.

**36. Define Babel.**

Babel is a JavaScript transpiler that converts ES6JavaScript into old ES5 JavaScript that can run in any browser.  
For every major release, the syntactical sugar is being added to JavaScript, but that is not immediately supported by Browser so to overcome this babel is used

In

// ES2020 nullish coalescing

function Hello(val) {

return val ?? "Hello world";

}

Out

function Hello(val) {

return val != null ? val : "Hello world";

}

**37. Mention some popular features of ES6.**

* There are various features of ES6 that includes:
* Template literals
* Support for Map/Set and WeakMap/WeakSet
* Arrow Functions
* Upgraded regular expressions
* Enhanced object properties
* Support for variables, immutable variables, and constants
* Default Parameters
* Multi-line strings are supported

**38. What are the object-oriented features supported in ES6?**

* Some object-oriented features of ES6 are as follows:
* Class Creation
* Inheritance
* Class methods and static methods
* Constructors
* Getter and Setter creation

**39. Give a thorough comparison between ES5 and ES6.**

* Various features of ES5 and ES6 are similar. Some main differences between ES5 and ES6 are as:

|  |  |
| --- | --- |
| **ES5** | **ES6** |
| 1. It is the fifth edition of ECMAScript. | 1. It is the sixth edition of ECMAScript. |
| 2. ES5 supports several primitive datatypes such as Boolean, string, number, null, and undefined. | 2. This edition supports additional datatypes of JavaScript including symbols as a primitive datatype. |
| 3. It has a larger community base. | 3. As it is the later edition after ES5, the community base is not much large as ES5 has. |
| 4. It supports both functions and returns keywords to define a function. | 4. It supports arrow function to define a function. |
| 5. Variables can only be defined with the var keyword. | 5. It supports let and const keywords to define variables. |
| 6. It has a lower performance than ES6. | 6. It has some enhanced features that help in better performance. |

**40. What is the difference between let and const? What distinguishes both from var?**

* Some differences between let, const, and var are as follows:

|  |  |  |
| --- | --- | --- |
| **let** | **const** | **var** |
| let variable comes with block scope. | const variable also has block scope. | But the var variable has functional scope. |
| This variable can be updated easily, but it doesn’t allow redeclaration into the scope. | Const variable cannot be updated as it is a constant and it also doesn’t allow redeclaration into the scope. | This variable can be easily updated and re-declared within the scope. |
| To access this variable, you need to initialize it first. | const variable also needs initialization before accessing it. | It can be accessed without any initialization. |
| Declaration of let variable can be done without initialization. | Const variables cannot be declared without initializing them. | It doesn’t require initialization for declaration. |

**41. Name some array methods that were introduced in ES6.**

* Some useful array methods of ES6 are below:

|  |  |
| --- | --- |
| **Methods** | **Definition** |
| Array.fill | This method is used to fill particular elements of the array with fixed values. |
| Array.from() | It is used when we want to convert non-array and iterable values into true arrays. |
| Array.prototype.keys() | This method returns an object of the array iterator with the keys of the array. |
| Array.of() | This method is used to create a sample of a variable number of arguments rather than a type or a specific number of arguments. |
| Array.prototype.values() | It is used to provide values to each key. |
| Array.prototype.find() | The find method is used to find values from an array. We can also pass some criteria in this method. |

**42. Name some string functions introduced in ES6.**

|  |  |
| --- | --- |
| **Methods** | **Definition** |
| includes | This method returns true when the string contains a given argument. |
| repeat | This function is used to create a new string and return that string containing provided number of copies. |
| startswith | It is used to check if a string starts with given specific characters of a string. |
| endswith | This function is used to determine if a string ends with specific characters of a given string. |

**43. Compare the ES5 and ES6 codes for object initialization and parsing returned objects.**

* **Object Initialization in ES5:**

var

a1 = 4, a2 = 5, a3 = 6;

obj = {

a1 : x,

a2 : y,

a3 : z

};

* **Object Initialization in ES6:**

const

a1 = 4, a2 = 5, a3 = 6;

ob = {

a1

a2

a3

};

In ES5, we need to use variables with the same name to initialize object properties. But, in the case of ES6, we don’t need to repeat the process of using variables with objects’ names.

Now, let us see how you can parse returned objects in ES5 and ES6 respectively:

* **Parsing returned objects in ES5:**

var

ob = getObject(),

x = ob.x,

y = ob.y,

z = ob.z;

* **Parsing returned object in ES6:**

const (x, y, z} = getObject();

As we have seen in the above examples, we must need the returned object first in ES5 to extract values from it while ES6 destructuring made it easier that doesn’t require the use of an object as a variable.

**44. How do you use Destructuring Assignment to swap variables?**

* The swapping of variables using destructuring assignment can be done as follows:

var x = 4, y = 5;

[x, y] = [y, x]

console.log(x);

console.log(y);

//here the values of both the variables are swapped.

**45. What is the result of the spread operator array shown below?**

[…’macbook’]

Output: [‘m’, ’a’, ’c’, ’b’, ’o’, ’o’, ’k’]

* A string can be iterated, and the spread operator is used to transfer each character of an iterable string to one element. This results in each character of the string as an element of an array.

**46. What is the Prototype Design Pattern?**

* The prototype pattern is used to design new objects and it returns objects with the values that are copied from the prototype or object. It is also known as the Properties pattern. It is used when we initialize business objects with values keeping in mind the default settings of the database we are using. JavaScript is a prototypal language in which new objects and prototypes are created using a Prototype design pattern.

**47. What is a WeakMap in ES6? How is it different from a Map?**

* WeakMap is another data structure added in ES6 that enables you to save a collection of key-value pairs. As the name suggests, it has the same features as a Map. The main difference is that the keys used in WeakMap are not of a primitive data type but must be objects that can hold any value. Another difference is that the keys are held weakly in WeakMap as compared to Map. So when the keys are referenced in WeakMap by using an object, that object can be garbage collected. And when the object is not in use, the JavaScript Garbage Collector finds it and removes it from the memory. This way keys are weakly held in WeakMap than Map.

**48. What is the advantage of using the arrow syntax for a constructor method?**

* There are various benefits of using Arrow syntax but the main advantage is that the value of ‘this’ keyword is set when we create a function. Also, the value cannot be changed or updated. So when we create an object with the help of a constructor, **this** refers to that object. To better understand see the code below:

const Shape = function(animal) {

this.animal = animal;

this.animal1 = function() { console.log(this.animal);};

this.animal2 = () => { console.log(this.animal);};

};

const dog = new Pet(‘Dog’);

const puppy = new Pet(‘Puppy’);

dog.animal1();

puppy.animal2();

dog.animal1.call(puppy);

dog.animal2.call(puppy);

dog.animal1.apply(puppy);

dog.animal2.apply(puppy);

dog.animal1.bind(puppy);

dog.animal2.bind(puppy);

var animalFromList1 = dog.animal1;

sayNameFromList1;

var animalFromList2 = dog.animal2;

animalFromList2;

In the above example, it is demonstrated that **this** can be modified but the arrow function remains the same always.

**49. What is a Temporal Dead Zone?**

* Temporal Dead Zone is an entity that is temporary and hence cannot be used for any purpose. The variable hosting cannot be done using the let keyword in ES6 and therefore its declarations don’t increase to the top of the execution context. Whenever we reference a variable in the block without initializing it, ES6 throws a ReferenceError. Therefore, until initialization, the variable falls under the ‘temporal dead zone’.

Example:

console.log(x);

console.log(y);

var x = 6;

let y = 5;

Explanation: Here we are trying to show the value of x variable as output which is undefined before taking as output and hence it results in ReferencError.

**50. What is the difference between Set and WeakSet in ES6?**

* **Set:**Set() is used to define different kinds of iterable objects which are similar to arrays. These objects must hold different values. See the example below to better underset a Set():

setVariable.add(4); // here 4 is added into the set as a new value

setVariable.add(2); //here, 2 is already added to the set, so it should be like 1,2,3

setVariable.add(“HelloWorld”); //Now a string is added to the set: 1,2,3, ‘HelloWorld’

setVariable.has(“Monday”); //This returns false as it doesn’t contain Monday

setVariable.has(“HelloWorld”); //This condition returns Boolean values as true and false, in this case, it is true as it contains the string.

setVariable.delete(“HelloWorld”); //This statement deletes the string from the set

setVariable.has(“HelloWorld”); //Now it will return false as the string is deleted from the set

setVariable.size; // it returns the size of the set which is 3

setVariable.clear(); // this statement clears all the values of set

* **WeakSet():**WeakSet is very similar to a Set(), but the difference is that it can hold objects only. If there is no reference of variables in WeakSet, then the variables can be removed automatically.

See the example below:

var weakSetVariable = new WeakSet ([{x:1}]);

var objX = {o:1}

var objY = {o:2}

weakSetVariable.has(objX); //the condition returns false as it was not added to the WeakSet

weakSetVariable.add(objX); //it adds an object to the set

weakSetVariable.add(objY); //it adds another object objY to the set

weakSetVariable.has(objY); //Now the condition returns true as the object is in WeakSet

delete objX; //It deletes the first object from the set

weakSetVariable.has(objX); //The statement returns false i.e. the objX is deleted in last statement

weakSetVariable.delete(objY); //deletes the second object from the WeakSet

weakSetVariable.add(1); //it will throw an error as no primitive values can be added

**Difference**: As you can see that objects are weakly held in WeakSet as compared to Set where if there is no reference to any object, then the WeakSet will be taken by the garbage collector. The values are held strongly in the case of a Set() as the values still exist even if they are not referenced. Another important difference is that WeakSet() cannot hold primitive values whereas Set() can hold.

**51. What is Proxy in ES6?**

* Proxy is an API in Es6 that is used to implement meta programming in ES6. You can define the custom behaviour of objects for fundamental operations. Proxy objects are also capable to perform some operations on behalf of real objects.

**52. What is the difference between const and Object.freeze()?**

* These two methods are totally different from each other where const can be applied to bindings such as variables whereas object.freeze can be used for values of objects. Object.freeze is used to create immutable objects that don’t allow change of properties, however, const is used to create immutable bindings such that new values cannot be assigned to bindings once a value is assigned.

**53. Why does the following not work as an IIFE (Immediately Invoked Function Expressions)? What needs to be modified in order for it to be classified as an IIFE?**

**function x(){}();**

* In the given expression, the JavaScript parser interprets the function in two parts such that function f(){} and (). The first part of declaration is a function itself whereas, the other is to execute the function without describing its name. The statement will give Uncaught SyntaxError as the name of the function is not given for execution.

To fix this implementation, we need to add more brackets after function declaration such as **(function x(){})()** and **(function x(){}()).**The void operator can also be used in a way such that: void function x(){}();. But, the given expression will always remain undefined which doesn’t make it classified as an IIFE function.

**54. Explain Internationalization and Localization.**

* **Internationalization:**It is a process of developing and changing the specifications of products and services so that they can easily be adapted based on the languages and regions all over the world.
* **Localization:**This process is the adaption of internationalized software, products, or services.

In simple words, we can say that internationalization is a process of creating and transforming products and services to make it easier for local languages and cultures to adapt whereas localization is a process that implements one or more cultures.

**55. List the advantages of Arrow Function.**

There are several benefits of using Arrow functions such as:

* Reduced code size
* There’s no requirement of using the return statement. It is optional for a single-line function.
* Functional braces are also optional for single-line statements.
* Arrow functions bind the context lexically.

**56. Explain the de-structuring assignment in ES6.**

De-structuring is a process of extracting data from objects and arrays in a single variable. This method can also be used to extract smaller fragments from objects and arrays. To better understand, see the example below:

let message = [‘Hi’, ‘Ashu’];

let [a1, a2] = message;

console.log (a1, a2);

**Output:**

Hi Ashu

**57. What is meant by Map in ES6?**

A new collection type is introduced in ES6, called Map.  A Map can be used to hold key-value pairs of values of different types that can be used as a key or a value. The keys and values defined in a Map object can be primitive or objects.

**58. Explain for…of the loop with an example.**

The for…of loop is used to iterate vales of strings, arrays, etc. For example:

var colors = [‘Red’, ‘Black’, ‘White’, ‘Blue’];

for (let value of colors){

console.log(value);

}

**Output:**

Red

Black

White

Blue

**59. What is meant by WeakSet?**

A Weakset can be used to store objects in a collection weakly. As the name suggests, it holds the values of objects weakly as compared to Set. Therefore it cannot store the same values of the objects and hence cannot be iterated. Some methods that are included in WeakSet are add(value), delete(value), and has(value).

**60. What is meant by WeakMap?**

WeakMaps are very similar to Maps where the only difference is that the keys in WeakMaps should be objects only. WeakMap holds each element as a key-value pair where the keys are referenced weakly. Some methods that are used in WeakMap are – get(key), set(key), and has(key).

**61. Mention the 3 states of promises in ES6.**

The 3 states of promises in ES6 include the following:

* **Pending:** It shows that the result is still not computed. Pending is the initial state of all promises in ES6.
* **Rejected:** This state is used to represent failure that occurred while computing.
* **Fulfilled:** This state shows that the operation has been completed.

**62. What is meant by Callback and Callback hell?**

**Callback:**Callback is used when we work with events. It is used to handle a function when the execution of another function is completed. It is also possible to pass a function as an argument to another function using Callback.

**Callback hell:**It is useful when you are developing web applications. A web application requires a lot of code in its development which makes it tedious to handle the callback. At this moment, callback nesting is done using callback hell which resolves this problem.

**63. List some new array methods that were introduced in ES6.**

There are various arrays methods in ES6 that includes the following:

* Array.from()
* Array.of()
* Array.prototype.fill()
* Array.prototype.values()
* Array.prototype.keys()
* Array.prototype.find()

**64. Give an example spread operator in ES6.**

The spread operator is very useful to expand the iterable values of any array or string. It is represented with three dots (…) to get the list of parameters. The example of a spread operator is given below:

let arr1 = [1,2,3];

let arr2 = [0,…arr1,4,5,6,7,8,9];

console.log(arr2);

**Output:**

[

0,1,2,3,4,5,6,7,8,9

]

**65. What is meant by the generator function?**

The generator function is used to work with various iterators and functions. The generator function can be paused in between execution for one or more times and also allows it to be resumed whenever needed. To declare a generator function, you need to use an asterisk after the **function** keyword, i.e., **function\***.

**66. What are modules in JavaScript?**

A module is a piece of code that makes maintenance of code easier. It also increases the reusability of the code. The modules of JavaScript contain a piece of JavaScript code in different files. Each module gets executed once after it is loaded to the main function.

**67. When shouldn’t you use arrow functions?**

Several reasons incorporate for not using arrow functions such as:

* Object methods: Let us see an example below to better understand this concept:

var x = {

y:1,

func: () => {

this.y--;

}

}

In the above example, the value of y doesn’t get dropped when we call x.func. It is because ‘this’ is not bound to any specific value and can inherit the value from its parent scope.

Function hoisting or named functions: Arrow functions cannot be used whenever hoisting or the use of named functions is needed as they are anonymous functions.

this keyword: Arrow functions don’t have this keyword of their own and depend on the outer context. Therefore, we are not able to use this keyword or arguments in a function whenever required.

**68. Explain named export and default export in ES6.**

**Named Export:**This module is used to export multiple values such that the name of the imported module should be the same as that of the exported module.

**Default Export:**When you want to export default functions, classes, or an object, this module comes into view. It doesn’t require the same name of imports and hence the naming of imports is fully autonomous.