## 1. What is ES6?

Es6 or ECMASCRIPT 2015 is sixth major release of ECMAScript language which comes with a lot of new features and syntax for writing web applications in Javascript. As currently, not all browsers support ES6, they support pre-versions of ES6. So to write web applications in ES6 that will support all Browsers we needed tools like Babel and Webpack.

## 2. Explain Destructuring Assignment in ES6?

Destructing assignment in another improvement in Es6. It allows us to extract data from array and objects into separate variables.

Example

let full\_name =['John','Deo'];

let [first\_name,last\_name]=full\_name;

console.log(first\_name,last\_name);

// outputs John Deo

Another example

let c=[100,200,330,400];

let [a,...b]=c;

console.log(a,b);

// outputs 100 [200, 330, 400]

## 3. What are template literals in Es6?

Template literals are the string with embedded code and variables inside. Template literal allows concatenation and interpolation in much more comprehensive and clear in comparison with prior versions of ECMAScript.

Let see an example of concatenating a string in JavaScript.

var a="Hello";

var b="John";

var c = a+ " " + b;

Console.log(c); //outputs Hello John;

In ES6 concatenation and interpolation is done by backtick “ in a single line. To interpolate a variable simply put in to {} braces forwarded by $ sign.>/p>

// In ES6

let a="Hello";

let b="John";

let c=`${a} ${b}`;

console.log(c); //outputs Hello John;

## 4. Explain Constants in Es6?

Constants also are known as immutable variables are a special type of variables whose content is not changed. In Es6 a constant is defined using const keyword. Constants in Es6 enable protection to overwrite a variable value, improve performance and helps programmers to write readable and cleaner code.

Example

In Es6

const WEBSITE\_URL = "http://www.abc.com";

WEBSITE\_URL="new url"; // generate an error;

console.log(WEBSITE\_URL);

In prior version of Es6

// and only in global context and not in a block scope

Object.defineProperty(typeof global === "object" ? global : window, "WEBSITE\_URL", {

value: "http://www.abc.com", enumerable: true,

writable: false,

configurable: false

});

console.log(WEBSITE\_URL);

Also, Read Best Node JS Interview Questions

5. What is Set in ES6?

Set is a collection of unique values. The values could be also primitives or object references.

Creating a Set in Javascript

let set = new Set();

set.add(1);

set.add('1');

set.add({ key: 'value' });

console.log(set); // Set {1, '1', Object {key: 'value'}}

## 6. Explain Generator function in ES6?

Generators are functions that can be exited and later re-entered. Their context (variable bindings) will be saved across re-entrances. A function keyword followed by an asterisk defines a generator function, which returns a Generator object.

A generator function returns an iterable object when it’s called. It is written using the new \* syntax as well as the new yield keyword introduced in ES6.

Each time yield is called, the yielded value becomes the next value in the sequence. Also, note that generators compute their yielded values on demand, which allows them to efficiently represent sequences that are expensive to compute, or even infinite sequences.

Generator Function Example.

function\* generator(i) {

yield i;

yield i + 10;

}

var gen = generator(10);

console.log(gen.next().value);

// expected output: 10

console.log(gen.next().value);

Further reading: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function\*

## 7. Explain WeakMap in ES6?

WeakMaps provide a way to extend objects from the outside without interfering with garbage collection. Whenever you want to extend an object but can't because it is sealed - or from an external source - a WeakMap can be applied. WeakMap was also introduced by ES6 in 2015

The WeakMap is same as Map where it is a collection of key/value pairs. But in WeakMap, the keys must be objects and the values can be arbitrary values. The object references in the keys are held weakly, meaning that they are a target of garbage collection (GC) if there is no other reference to the object anymore. The WeakMap API is the same as the Map API.

However, One difference to Map objects is that WeakMap keys are not enumerable. And there are no methods giving us a list of keys. If they were, the list would depend on the state of garbage collection, introducing non-determinism. If we want to have a list of keys, we should use a Map.

## 8. What is the use of let & const in JavaScript?

In modern javascript let & const are different ways of creating variables. Earlier in javascript, we use the var keyword for creating variables. let & const keyword is introduced in version ES6 with the vision of creating two different types of variables in javascript one is immutable and other is mutable.

const: It is used to create an immutable variable. Immutable variables are variables whose value is never changed in the complete life cycle of the program.

let: let is used to create a mutable variable. Mutable variables are normal variables like var that can be changed any number of time.

Previously, when we declare any variable using var, it was function scoped. Meaning the variable can be accessed within the function. This leads to wrap the code in a function whenever we need to create a new scope.

But let and const uses block scoping. This means the variable declared using these keywords only exist within the innermost block that surronds them.

If we declare a let variable inside a block like if condition, for loop, it can be accessed within the block.

Also we can not re-declare the same variable with the same scope.

const is immutable. It means the value must be given at the time of the declaration and it can not be re-assigned or changes. Although we cannot change the value of the const but we can mutate them.

## 9. List some new features of ES6

New Features in ES6.

Support for constants (also known as “immutable variables”)

Block-Scope support for both variables, constants, functions

Arrow Functions

Extended Parameter Handling

Template Literals

Extended Literals

Enhanced Regular Expression

Enhanced Object Properties

Destructuring Assignment

Modules, Classes, Iterators, Generators

Support for Map/Set & WeakMap/WeakSet

Promises, Meta-Programming ,Internationalization & Localization

## 10. What is Babel?

Babel is one of the most popular JavaScript transpilers and becomes the industry standard. It allows us to write ES6 code and convert it back in pre-Es6 JavaScript that browser supports.

For example look the below code snippet.

In ES6 (ECMASCRIPT 2015)

const PI = 3.141593;

PI > 3.0 ;

export{PI};

In ES5 after conversion

"use strict";

Object.defineProperty(exports, "\_\_esModule", {

value: true

});

var PI = 3.141593;

PI > 3.0;

exports.PI = PI;

## 11. List steps to install Babel?

Installation: In order to install Babel, you require node.js and NPM. Make sure Node.js is installed on your server.

To check node installed or not run below commands on your terminal.

node -v

npm -v

Installing Babel : We can install Babel CLI locally by running below command on terminal.

npm install --save-dev babel-cli

## 12. What is Webpack?

Webpack allows you to run an environment that hosts babel. Webpack is opensource javascript module bundler which takes modules with dependencies and generates static assets representing those modules.

## 13. List benefits of using Webpack?

Benefits of using Webpack.

It bundles your multiple modules and packs it into a single .js file.

It comes with integrated dev server. A small express app for local development. You simply include one Javascript tag pointed to the server, like localhost:8080/assets/bundle.js, and get live code updating and asset management for free.

## 14. What is Spread Operator in ES6?

Spread Operator provides a new way to manipulate array and objects in Es6.A Spread operator is represented by … followed by the variable name.

Example :

let a =[7,8,9];

let b=[1,2,3,...a,10];

console.log(b); // [1,2,3,7,8,9,10]

So spread operator spreads the contents of variable a and concatenates it in b.

Another Example

function print(...z){

console.log(z);

}

print(1,2,3,4);//[1,2,3,4]

15. How to create a Javascript class in ES6?

In Es6 you can create a class using the Class keyword.Below is sample javascript class in ES6.

class User{

constructor(name,age) {

this.name = name;

this.age = age;

}

getData() {

console.log(this.name + " is " + this.age + " years old !");

}

}

var user = new User("foo", 7);

s1.getData();

## Q4. What is Arrow function? What are all its uses? How it differs from normal function?

Arrow functions are a short-hand notation for writing functions in ES6. The arrow function definition consists of a parameter list ( ... ), followed by the =>marker and a function body. For single-argument functions, the parentheses may be omitted.

If the arrow function is implemented with “concise body” (without {}), it does not need an explicit return statement. Note the omitted { } after the =>.

Arrow functions behavior with this keyword varies from that of normal functions. Each function in JavaScript defines its own this context but arrow functions capture the this value of the nearest enclosing context.

There are four fundamental differences between arrow functions and function functions

They close over this, and do not have their own versions.

They can have a concise body (without { }) rather than a verbose one (but they can have a verbose body as well).

They cannot be used as constructors. E.g., you can’t use new with an arrow function. Hence arrow functions do not have a protoype property on them.

There is no generator syntax for arrow functions. E.g., there is no arrow equivalent to function \*foo() { ... }.

## Q5. What are all the new changes in Object literal notations in ES6?

ES6 allows declaring object literals by providing shorthand syntax for initializing properties from variables and defining function methods. It also enables the ability to have computed property keys in an object literal definition.

## Q7. What is Set?

Set objects are collections of unique values. Duplicate values are ignored, as the collection must have all unique values. The values can be primitive types or object references.

Also, NaN and undefined can also be stored in a Set. NaN is considered the same as NaN (even though NaN !== NaN).

## Q8. What is Class expression?

The Class expression is one way to define a class in ES6. Similar to function expressions, class expressions can be named or unnamed. If named, the name of the class is local to the class body only. JavaScript classes use prototype-based inheritance.

A class expression has a similar syntax to a class statement (declaration). However, with class expressions, you are able to omit the class name (“binding identifier”), which you can’t with class statements. Additionally, class expressions allow you to redefine/re-declare classes and don’t throw any type errors like class declaration. The constructor property is optional. And, typeof the classes generated using this keyword will always be “function”.

## Q10. What is WeakSet?

The WeakSet object lets you store weakly held objects in a collection.

WeakSet objects are collections of objects. An object in the WeakSet may occur only once; it is unique in the WeakSet's collection. The main differences to the Set object are:

Unlike Set, WeakSets are collections of objects only and not of arbitrary values of any type.

The WeakSet is weak: References to objects in the collection are held weakly. If there is no other reference to an object stored in the WeakSet, they can be garbage collected. That also means that there is no list of current objects stored in the collection. WeakSets are not enumerable.