***Javascript Toolkit:***

**1.What is scope in JavaScript?**

In JavaScript, each function gets its own *scope*. Scope is basically a collection of variables as well as the rules for how those variables are accessed by name. Only code inside that function can access that function's scoped variables.

A variable name has to be unique within the same scope. A scope can be nested inside another scope. If one scope is nested inside another, code inside the innermost scope can access variables from either scope.

2. **What is the difference between == and ====?**

== is the abstract equality operator while ==== is the strict equality operator. The == operator will compare for equality after doing any necessary type conversions. The === operator will not do type conversion, so if two values are not the same type === will simply return false

**3.Explain the concept of hoisting in JavaScript.**

Hoisting is a JavaScript mechanism where variables, function declarations and classes are moved to the top of their scope before code execution. Remember that JavaScript only hoists declarations, not initialisation. Let's take a simple example of variable hoisting,

**4.Explain the event loop in JavaScript.**

The event loop is a process that continuously monitors both the call stack and the event queue and checks whether or not the call stack is empty. If the call stack is empty and there are pending events in the event queue, the event loop dequeues the event from the event queue and pushes it to the call stack. The call stack executes the event, and any additional events generated during the execution are added to the end of the event queue.

### **5. What do you mean by NULL in Javascript?**

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number, and no array object.

### **6. What is event bubbling?**

JavaScript allows DOM elements to be nested inside each other. In such a case, if the handler of the child is clicked, the handler of the parent will also work as if it were clicked too.

**7.What is the main difference between localStorage and sessionStorage?**

LocalStorage is the same as SessionStorage but it persists the data even when the browser is closed and reopened(i.e it has no expiration time) whereas in sessionStorage data gets cleared when the page session ends.

**8. What is a callback function?**

A callback function is a function passed into another function as an argument. This function is invoked inside the outer function to complete an action. Let's take a simple example of how to use callback function

**function callbackFunction(name) {**

**console.log("Hello " + name);**

**}**

**function outerFunction(callback) {**

**let name = prompt("Please enter your name.");**

**callback(name);**

**}**

**outerFunction(callbackFunction);**

### **9. Explain Hoisting in javascript.**

Hoisting is the default behavior of javascript where all the variable and function declarations are moved on top. This means that irrespective of where the variables and functions are declared, they are moved on top of the scope. The scope can be both local and global.

**10.What is typeof operator**

You can use the JavaScript typeof operator to find the type of a JavaScript variable. It returns the type of a variable or an expression.

**11.How to make Asynchronous JavaScript?**

3 way to make JavaScript asynchronous, we can to use

i. Callback functions (setTImeout, etc)

ii. Promises (fetch)

iii. Async/ await

**12.Why is JavaScript treated as Single threaded?**

JavaScript is a single-threaded language. Because the language specification does not allow the programmer to write code so that the interpreter can run parts of it in parallel in multiple threads or processes. Whereas languages like java, go, C++ can make multi-threaded and multi-process programs.

**13. What is v8 Engine**

V8 is Google’s open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others. It implements ECMAScript and WebAssembly, and runs on Windows 7 or later, macOS 10.12+, and Linux systems that use x64, IA-32, ARM, or MIPS processors. V8 can run standalone, or can be embedded into any C++ application.

**14.What do you mean by Synchronous?**

Synchronous means the code runs in a particular sequence of instructions given in the program. Each instruction waits for the previous instruction to complete its execution.

**15.What do you mean by asynchronous?**

Asynchronous is a non-blocking architecture, so the execution of one task isn't dependent on another. Tasks can run simultaneously

**16.what is the difference setInterval and clearInterval?**

→The setTimeout() method is used to call a function after a certain period of time. setTimeout() is cancelled by clearTimeout() method.

→The setInterval() Javascript method is used to call a function repeatedly at a specified interval of time.setInterval() is cancelled by clearInterval() method.

**17.what is javaScript Heap?**

Heap(Or memory heap) is the memory location where objects are stored when we define variables. i.e, This is the place where all the memory allocations and de-allocation take place. Both heap and call-stack are two containers of JS runtime. Whenever runtime comes across variables and function declarations in the code it stores them in the Heap.

**18.what is javaScript Stack?**

**19. Tell the difference Between Primitive and Non-Primitive Data types in javaScript?**

**→**Primitive data types: The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

There are 7 types of primitive data types.

String,number,boolean,null,undefined,bigint,symbol

**→**Non-primitive data types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

There are 2 types of primitive data types.

Object, Array

**20. What are the Truthy and Falsy Values? give me some examples.**

truthy and falsy values are concepts used to determine the truthiness or falsiness of a value in programming languages. In most programming languages, including JavaScript, truthy values are those that evaluate to true in a boolean context, while falsy values evaluate to false.

In javascript, the following values are considered falsy: false,0,””,null,undefined,NaN

In javascript, the following values are considered truthy: true,Numbers,Strings,Objects,Arrays,Functions

**Here Some Examples:**

**Boolean(false); // false**

**Boolean(0); // false**

**Boolean(""); // false**

**Boolean(null); // false**

**Boolean(undefined); // false**

**Boolean(NaN); // false**

**Boolean(true); // true**

**Boolean(42); // true**

**Boolean("Hello"); // true**

**Boolean({}); // true**

**Boolean([]); // true**

**Boolean(function(){});// true**

**21. What is the difference between null and undefined? (important)**

null is an intentional absence of any object value that you can assign explicitly, while undefined indicates the absence of a value and is automatically assigned by JavaScript. null is of type object, whereas undefined is of type undefined.

**22. What is scope in javaScript?**

**23. Define Block scope and global scope?**

**24.Explain closure in JavaScript?**

**25.What is Regular Expression?**

**26.Tell me something about JS engine v8 internal mechanism?**

***27. What paradigm is Javascript***

*JavaScript is a multi-paradigm language, supporting imperative/procedural programming, Object-Oriented Programming and functional programming. JavaScript supports Object-Oriented Programming with prototypical inheritance.*

***ES6 Toolkit:***

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

**→**Maps and Sets

→Iterators

→Symbol

→Generators

→Arrow function

→Block-scoped variables

→Template Literals

→Classes

→Modules

→Destructuring

→Default parameters

→Rest parameters

→Spread operator

**3.Explain the difference between var, let, and const.**

→Var is function-scoped and can be redeclared and reassigned within its scope

→let is block-scoped and allows you to reassign its value within its scope but cannot be redeclared.

→const is also block-scoped, but it is a constant variable that cannot be reassigned or redeclared once a value is assigned to it

**4.How does arrow function syntax differ from regular function syntax?**

→Arrow functions don't bind their own this, arguments, super, or new.target. Instead, they inherit them from the surrounding scope.

→Arrow functions can't be used as constructors with the new keyword.

→Arrow functions don't have their own arguments object.

Regular function syntax, on the other hand, has its own this context, arguments object, and can be used as constructors.

**5.What is a template literal in ES6? Provide an example.**

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.

**Example:**

**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.**

**6.How can you create a class in ES6? Compare it with prototype-based inheritance.**

In ES6, you can create a class using the class keyword. It provides a more intuitive and concise syntax for working with objects and inheritance compared to the traditional prototype-based approach.

**Here's an example of creating a class in ES6:**

**class Animal {**

**constructor(name) {**

**this.name = name;**

**}**

**speak() {**

**console.log(`${this.name} makes a sound.`);**

**}**

**}**

**const dog = new Animal("Dog");**

**dog.speak();**

**// Output: Dog makes a sound.**

**7.What is the purpose of the spread operator (...) in ES6? Give 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**

**8.Explain the concept of destructuring assignment in ES6. Provide an example.**

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 :**

**const numbers = [1, 2, 3];**

**const [a, b, c] = numbers; //destructuring**

**console.log(a, b, c);**

**// Output: 1 2 3**

**9.What is the significance of the "default" keyword in ES6 modules?**

In ES6 modules, the default keyword is used to specify a default export for a module. It allows you to export a single value or function that will be the main/default export when importing the module.

For example, consider a module named hello.js that exports several functions. We can have one function designated as the default export:

**// hello.js**

**export default function add(a, b) {**

**return a**

**10.How does ES6 support promises for asynchronous programming? Provide an example.**

**11.What is the purpose of the "async" and "await" keywords in ES6? Give an example.**

The purpose of the "async" and "await" keywords in ES6 is to simplify the syntax for working with promises and make asynchronous code look more like synchronous code. The "async" keyword is used to define an asynchronous function, and the "await" keyword is used to pause the execution of the function until a promise is resolved or rejected. Here's an example:

**// Asynchronous function using async/await**

**async function getData() {**

**try {**

**const response = await fetch('https://api.example.com/data');**

**const data = await response.json();**

**console.log(data);**

**} catch (error) {**

**console.log('Error:', error);**

**}**

**}**

getData();

In this example, the getData function is defined as an asynchronous function using the "async" keyword. Inside the function, we use the "await" keyword to pause the execution until the promise returned by fetch resolves with a response. We then use another "await" to pause the execution until the promise returned by response.json() resolves with the parsed JSON data. If any promise rejects, the code inside the catch block is executed.

**12.How can you handle errors in asynchronous code using ES6 features?**

In ES6, you can handle errors in asynchronous code using features like promises and the try...catch statement with async/await. Here's an example:

**async function fetchData() {**

**try {**

**const response = await fetch('https://api.example.com/data');**

**if (!response.ok) {**

**throw new Error('Request failed with status code ' + response.status);**

**}**

**const data = await response.json();**

**console.log(data);**

**} catch (error) {**

**console.log('Error:', error);**

**}**

**}**

fetchData();

In this example, after fetching the data from the API, we check if the response status is not OK (e.g., 200). If it's not, we manually throw an error using the throw statement. The error is then caught by the catch block, and the corresponding error message is logged.

**13.What are generators in ES6? How do they differ from regular functions?**

**14.Explain the concept of modules in ES6. How do you import and export modules?**

Modules in ES6 are a way to organize and reuse JavaScript code. They provide a mechanism for encapsulating code into separate files, making it easier to manage dependencies and promote code reusability. Modules have their own scope and can export specific values or functionalities to be used by other modules. Here's an example of importing and exporting modules:

**// math.js - exporting a function**

**export function square(x) {**

**return x \* x;**

**}**

**// main.js - importing the function**

**import { square } from**

**15.Describe the concept of iterators and iterables in ES6.**

In ES6, iterators and iterables provide a standardized way to iterate over collections of values in JavaScript. An iterable is an object that implements the Symbol.iterator method, returning an iterator object with a next() method.

**16.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**

**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**

## **17. 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

**18. 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.

**Example:**

**Const x = () => { console.log(‘Hello Arrow Function’)};**