1. What is lexical structure?

When we look at or use a programming language, we need to understand the basic rules which determine how a program is written or used. This is referred to as the lexical structure or syntax of the language.

1. What is Unicode?

They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding could contain enough characters: for example, the European Union alone requires several different encodings to cover all its languages.

1. Explain all the keywords present in the JavaScript with examples.

**1. goto keyword**

Used to return execution control to a specific location. In general, the goto can be accomplished by the break and continue keywords.

**Example**

var no=0;

sposition

document.write(" something print here ");

no++;

if(no < 10) goto sposition;

Now the same code we rewrite with break and continue keywords as

var no=0;

sposition: while(true) {

document.write(" something print here ");

no++;

if(no < 10) continue sposition;

break;

}

#### ****2. in**** keyword

It is an operator returns true if the specified property is present in the specified object, else it returns false.

**Example**

var fruits={f1: "apple", f2: "banana", f3: "orange"};

// output as true expected

console.log('apple' in fruits);

#### 3. instanceof keyword

Returns true if the object is an instance of the class otherwise false

**Example**

var fruits=["apple", "banana", "orange"];

// Returns true

fruits instanceof Object;

// Returns true

fruits instanceof Array;

// Returns false

fruits instanceof String;

#### ****4.**** arguments****keyword****

Represents the list of parameters passed to the function when calling the function.

**Example**

const func = function(p1, p2, p3) {

const param = Array.from(arguments);

console.log(param) // [11, 22, 33]

}

func(11, 22, 33);

#### ****5.**** public****keyword****

It is an access modifier that can be used with attributes, classes, constructors and methods which make it accessible to other classes.

**Example**

public class Employee {

public String efn = "Joseph";

public String eln = "Doe";

}

class MainClass {

public static void main(String[] args) {

Employee obj = new Employee ();

System.out.println("Name= " + obj.efn + " " + obj.lname);

}

}

#### ****6.**** D****o**** keyword

Used to define a do-while loop.

**Example**

var a=1;

do {

document.write("loop is running for " + a + "times</p>");

a++;

}

while(a <= 10);

#### ****7.**** Function keyword

Used to define a function to execute a block of code.

**Example**

var func = function(){

return "Hello";

}

alert(func());

#### 8. class\* keyword

Used to define a class.

**Example**

public class Employee {

public String efn = "Joseph";

public String eln = "Doe";

}

#### ****9. return**** keyword

Used to return from the function or method with or without a value.

**Example**

var func = function(){

return "Hello";

}

#### ****10. Eval**** keyword

Used to evaluate a specified string. The eval use as a global function eval().

**Example**

function fun( ) {

var str1=2;

var str1=3;

var res = eval(new String(str1 + str2));

document.write(res);

}

fun();

1. What are shorthand operators, explain with a suitable example?

If you’re adding, subtracting, multiplying, dividing, or remaindering two values, there’s no need to type out the entire equation. Use these shorthands to save time and code:

## Adding Two Values

As an example, we’ll look at the addition of two variables. Here’s how you might write that code in JavaScript:

let x = 1;  
let y = 2;**x = x + y;**console.log(x);  
// 3

Hopefully the above code is pretty straight forward. We’ve declared our variables x, and y, and assigned them values of 1 and 2 respectively. We then set x equal to x + y. When we log out x, we get 3!

Now, here’s what the shorthand code to do this looks like:

let x = 1,   
 y = 2;**x += y;**console.log(x);  
// 3

Instead of using x = x + y, we can write x += y. Both of these code bits do the exact same thing — add x + y, then assign the resulting value to x.

1. What is “use Strict” in JavaScript?

The purpose of "use strict" is to indicate that the code should be executed in "strict mode". With strict mode, you can not, for example, use undeclared variables. All modern browsers support "use strict" except Internet Explorer 9 and lower: Directive. "use strict"