JAVA SCRIPT

1) What will be the output of this code?

```
console.log(x);
 var x=5;
```

<u>o/p</u>: undefined

<u>Explanation</u>: As we know that interpreter will execute code line by line but before execution JavaScript engine scans the code and stores the memory to every variable, but it considers default value as undefined rather than user defined value that's why the output will be undefined.

2) What will be the output of this code?

```
console.log(x);
var x;
```

<u>o/p</u>: undefined

<u>Explanation</u>: As we know that interpreter will execute code line by line but before execution the JavaScript engine scans the code and stores the memory to every variable and takes default value as undefined that's why the output will be undefined.

3) What will be the output of this code?

```
console.log(a);
a=10;
var a;
```

<u>o/p</u>: undefined

<u>Explanation:</u> In JavaScript, variables declared using var are hoisted. This means the declaration (var a;) is moved to the top during execution, considers value as undefined as default but the assignment happens later that's why the output is undefined.

4) What will be the output of this code?

```
console.log(a);
```

<u>o/p:</u> Reference Error: a is not defined

<u>Explanation:</u> Since variable a was not declared at all, JavaScript will throw a ReferenceError.

5) What will be the output of this code?

```
console.log(a);
  var a=10;
  console.log(a);
  a=20;
  console.log(a);
```

o/p: undefined

```
10
20
```

Explanation:

step1: In JavaScript, variables declared using var are hoisted. This means the declaration (var a;) is moved to the top during execution, considers value as undefined as default but the assignment happens later that's why the output is undefined.

Step2: Now a has been initialized with 10 and then console.log(a) gets executed so the output will be 10.

Step3: Here reassigning for the variable is done with a value of 20 and then console.log(a) gets executed so the output will be 10.

6) What will be the output of this code?

```
console.log(f);
 var f=100;
 var f;
 console.log(f);
```

o/p: undefined

```
100
```

Explanation:

Step1: In JS, variables declared with var are hoisted. This means the declaration of f is moved to the top, but not its initialization. So, the output will be undefined for first line.

Step2: Here the initialization is done so the variable f holds a value of 100.

Step3: Here it re-declares the variable, but the value is already stored the value remains same and unchanged.

Step4: console.log(f) will display 100.

7) What will be the output of this code?

```
console.log(g);
```

```
  var g=g+1;
 console.log(g);
```

<u>o/p:</u> undefined

```
NaN
```

<u>Explanation:</u>

Step1: First console.log(g); output will be undefined because the declaration of g is moved to the top, but not its initialization with the help of hosting.

Step2: var g = g + 1; attempts to add 1 to g, but g is undefined.

Step3: The second console.log(g); output is NaN because value is not assigned for g.

8) What will be the output of this code?

```
  var h;
 console.log(h);
 h=50;
 console.log(h);
```

<u>o/p:</u> undefined

```
50
```

<u>Explanation:</u> At first variable is declared but value is not assigned to variable and hence when we give console.log(h); it will give the default value of the variable that is undefined and then the value is assigned to the variable h here if we give console.log(h) then the output will be the value 50.

9) What will be the output of this code?

```
console.log(i);
i=10;
var i=40;
console.log(i);
```

<u>o/p</u>: undefined

```
40
```

<u>Explanation:</u> In first step console.log(g); output will be undefined because the declaration of i is moved to the top of the scope, but not its initialization with the help of hosting.

In the second step value 10 is assigned to the variable.

In the third step reassigning for the variable is done with the value of 40 and hence in the last step console.log(i); it will display the output as 40.