JAVA SCRIPT

1.What will be the output of this code?

Console.log(x);

Var x = 5;

Answer: Undefind.

Reason: when you call console.log(x); before the assignment, x has been declared but not yet assigned a value, leading to undefined.

2 .What will be the output of this code?

console.log(a);

var a;

Answer: Undefind.

Reason: The var declaration is hoisted to the top of the scope, so a is declared but not initialized before the console.log(a); line.Since a is declared but not assigned a value at the time of the log, it defaults to undefined.

3 .What will be the output of this code?

console.log(b);

b=10;

var b;

Answer: Undefind.

Reason: The var declaration is hoisted, so the declaration of b (var b;) is moved to the top of the scope. However, the assignment (b = 10;) happens later in the code.When console.log(b); is executed, b has been declared but not yet assigned a value, so it logs undefined.

4 .What will be the output of this code?

Console.log(c);

Answer: c is not defind.

Reason: If you run the code console.log(c); without declaring or assigning a value to c beforehand, it will throw a ReferenceError, indicating that c is not defined.

5 .What will be the output of this code?

```
console.log(e);
var e=15;
console.log(e);
e=20;
 console.log(e);
```

Answer: undefined

15

20

Reason: First console.log(e);: At this point, the variable e has been declared with var but not initialized, so it is hoisted. The output will be undefined. var e=15;: Here, e is assigned the value 15.Second console.log(e);: Now, e has been initialized to 15, so this will log 15.e=20;: The value of e is updated to 20.Third console.log(e);: This logs the updated value, which is 20.

6 .What will be the output of this code?

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

Answer: undefined

100

Reason: First console.log(f);: The variable f is declared with var, but since it hasn't been assigned a value yet, it is hoisted and results in undefined.var f=100;: The variable f is then assigned the value 100.var f;: This line declares f again, but since it was already declared, this does not change its value. Second console.log(f);: Now, f holds the value 100, so this logs 100.

7 .What will be the output of this code?

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

Answer: undefined

NaN

Reason: First console.log(g);: The variable g is declared with var, but it hasn't been initialized, so it is hoisted and results in undefined. Thus, the output is undefined.var g = g + 1;: At this point, g is still undefined, so the expression g + 1 evaluates to undefined + 1, which results in NaN (Not-a-Number). The variable g is then assigned this NaN value. Second console.log(g);: Now, g holds the value NaN, so this logs NaN.

8 .What will be the output of this code?

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

Answer: undefined

50

Reason: The first console.log(h); shows undefined because h is declared but not initialized. After h is assigned the value 50, the second console.log(h); outputs 50.

9 .What will be the output of this code?

```
console.log(i);
i = 10;
var i= 5;
console.log(i);
```

Answer: undefined

5

Reason: In the first console.log(i);, i is hoisted but not initialized, so it outputs undefined. When i = 10; runs, it doesn't affect the hoisted variable yet. After var i = 5;, the value of i is updated to 5, which is logged in the second console.log(i); .