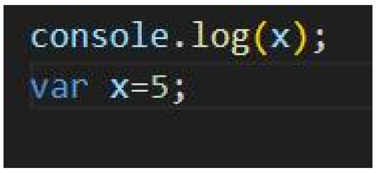
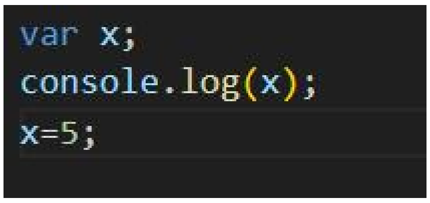
Code Output Explanations

1. **What will be the output of this code?**



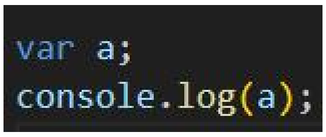
```
console.log(x);
var x=5;
```

   - **Output**: undefined
   - **Explanation**:   This happens because of hoisting. Here hoisting moves declaration (var x) to top of their scope. At top of the scope value is not assigned to x and x remains where it is. It looks like as:



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

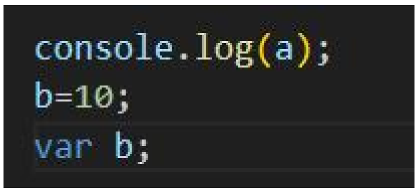
2. **What will be the output of this code?**



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

   - **Output**: undefined
   - **Explanation**: The var a; declaration is moved to the top of the scope. Since a doesn't have a value yet, so it's undefined.

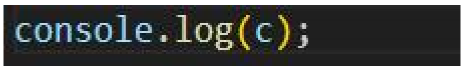
3. **What will be the output of this code?**



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

   - **Output**: undefined
   - **Explanation**: In this snippet both the declaration (var b) and the assignment (b = 10) exist, but only the declaration is hoisted to the top. The assignment remains in place. Since console.log(b) is executed before the assignment b = 10, the value of b is undefined.

4. **What will be the output of this code?**

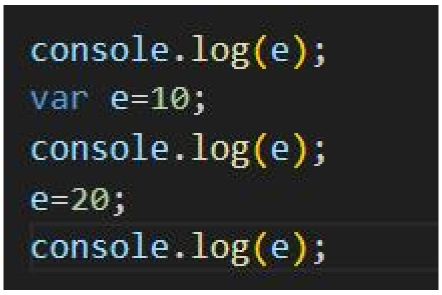


```
console.log(c);
```

   - **Output**: ReferenceError: c is not defined

- **Explanation**: c is accessed without declared or defined in the code**.**

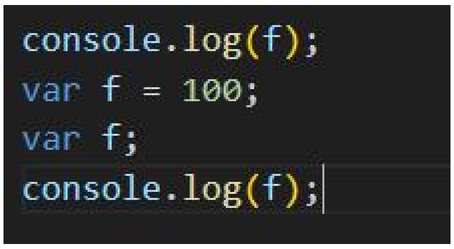
5. **What will be the output of this code?**



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

- **Output**:
  1. undefined
  2. 10
  3. 20
- **Explanation**:
  - The declaration var e is hoisted, but the assignment (e = 10) is not, So the first console.log(e) outputs undefined because e is declared but not initialized.
  - The second console.log(e) happens after e is initialized to 10, so it prints 10.
  - After the second one, e is reassigned to 20, so the final console.log(e) prints 20.

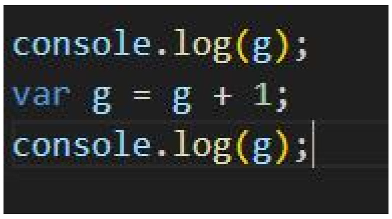
6. **What will be the output of this code?**



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

- **Output**:
  1. undefined
  2. 100
- **Explanation**:
  - The declaration var f is hoisted, but the assignment f = 100 is not. So, the first console.log(f) prints undefined because only the declaration is hoisted, and f hasn’t been assigned a value.
  - The second var f: var declarations are hoisted to the top, but redeclaring f doesn’t change anything. The assignment f = 100 has already happened, so the second console.log(f) prints 100.

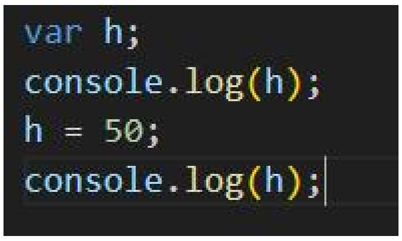
7. **What will be the output of this code?**



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

- **Output**:
    1. undefined
    2. NaN
- **Explanation**:
    - The declaration var g be hoisted, but not the initialization. So, the first console.log(g) outputs undefined because g is declared but not initialized.
    - In g = g + 1, since g is still undefined , undefined + 1 results in NaN (Not a Number). So, the second console.log(g) prints NaN.

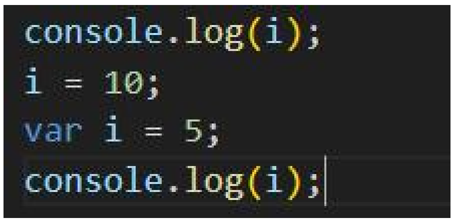
8. **What will be the output of this code?**



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

- **Output**:
    1. undefined
    2. 50
- **Explanation**:
    - The variable h is declared but not initialized, so the first console.log(h) prints undefined because h doesn't have a value.
    - After h is assigned the value 50, the second console.log(h) prints 50.

9. **What will be the output of this code?**



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

- **Output**:
    1. undefined

2. 5

- **Explanation**:
  - Due to hoisting, the declaration var i is moved to the top but the assignment (i = 5) stays where it is. So, the first console.log(i) outputs undefined because i is declared but not initialized.
  - The second console.log(i) prints 5 since i was assigned the value.