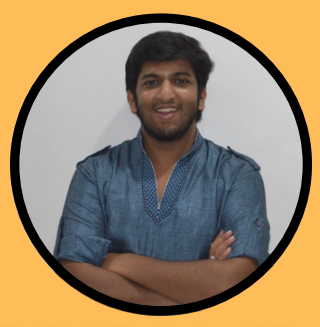
JS Tutorial

JAVASCRIPT TRICKY INTERVIEW QUESTIONS PART - 1



Nikhil Vallore

Mentor | Software Engineer

Swipe >>

1. Variable Scope :

```
for (var i = 0; i < 10; i++)  {
  setTimeout(() => console.log(i), 0);
}
```

What is the output of it?

0 1 2 3 4 5 6 7 8 9
Is it? swipe to find out...

@nikhilvallore

Swipe >>

Output: 10 10 10 10 10 10 10 10 10 10

why?
The problem is that the variable i is declared with var, which has function scope instead of block scope. This means that the reference to the i variable will not change, even though it is being updated in the for loop. As a result, all 10 setTimeout invokes will have access to the same i variable, which will have a value of 10 when the loop is finished

@nikhilvallore

Swipe >>

2. Variable hoisting :

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

var b = 'b';

function a() {
console.log('a');
}

var c = function c() {
console.log('c');
}
```

What is the output of it?

@nikhilvallore

Swipe >>

Output: function a() undefined undefined

When you use the "function" keyword to create a function in JavaScript, it will be moved to the top of the scope and be available to use from the beginning of that scope. This is called hoisting. If you try to use the function before it is defined, it will still print "function a()"

On the other hand, if you use the "var" keyword to declare a variable, it will be created and assigned the value of "undefined" until it is actually initialized with a value. For example, if you declare a variable "b" with "var b", it will be available throughout the scope but will have a value of "undefined" until you assign it a value like "b = 'b'"

@nikhilvallore

Swipe >>

3. Coercion :

```
const result1 = "2" + 1;
const result2 = 2 + true;
const result3 = "2" - 1;
const result4 = "x" * 1;

console.log(result1, result2, result3, result4)
```

What is the output of it?

@nikhilvallore

Output: 21 3 1 NaN

JavaScript automatically converts data types when using math operations, except when using the "+" operator. If a string is present, it will treat the operation as string concatenation instead of a math operation.

For example, in the code "2 + 1", the result is "3" (a number), but in the code "2 + '1'", the result is "21" (a string).

In the code "true + 2", the boolean value of "true" is converted to a number (1) and the result is "3".

However, if the conversion is not possible, like in the code "'x' + 2", JavaScript will return "NaN" (Not a Number).

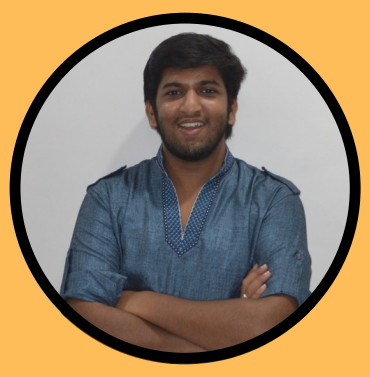
@nikhilvallore

Found it interesting?

Follow ME on Linked

for more such content

Until Next Time Keep Learning and Keep Sharing!!



Nikhil Vallore