Module 35-JavaScript Tricky Concepts

ইন্টারভিউ প্রশ্ন

এই মডিউল এ আমরা কয়েকটা জিনিস দেখবো। সেখান থেকে কয়েকটা জিনিস ইন্টারভিউতে প্রায়ই জিজ্ঞেস করে বসে।

১. জাভাস্ক্রিপ্ট কী কী ডাটা টাইপের ভেরিয়েবল আছে?

২. জাভাস্ক্রিপ্ট ফাংশন, বা array কি টাইপের জিনিস ?

৩. জাভাস্ক্রিপ্ট এ array যদি অবজেক্ট হয় তাহলে কিভাবে চেক করবে কোন একটা ভেরিয়েবল একটা array নাকি array না?

৪. এর মধ্যে undefined আর null এর মধ্যে ডিফারেন্স কি।

৫. double equal (==) আর triple equal (===) এই দুইটার মধ্যে ডিফারেন্স কি।

৬. বা implicit conversion কি জিনিস একটা কখন হয়।

৭. এছাড়াও জাভাস্ক্রিপ্ট এ কয়েক ধরনের স্কোপ আছে। এই স্কোপ গুলার মধ্যে ডিফারেন্স কি। কখন কোনটা হয়।

৮. ব্লক স্কোপ কি জিনিস? let, const কি টাইপের স্কোপ তৈরি করে?

৯. (এডভান্সড) Closure কি জিনিস? এইটা কিভাবে কাজ করে?

১০. Callback function কি জিনিস?

১১. (এডভান্সড) Hoisting কি জিনিস? (গুগলে সার্চ দিয়ে আরো ভালো করে শিখো)

১২. (এডভান্সড) কি ধরণের ভেরিয়েবল reference দিয়ে ফাংশনে পাঠানো হয় আর কোন ধরণের ভেরিয়েবল value হিসেবে পাঠানো হয়।

35-1 Module Overview, Dynamic data types

JavaScript Dynamic data types

```
let a=19;
console.log(typeof(a));
```

```
let b = 'Jamal Uddin Shahen Shah';
console.log(typeof(b));

let c = false;
console.log(typeof c);
```

35-2 Primitive data type and non-primitive data type

Primitive data types are number, string, boolean, NULL, Infinity and symbol. Non-primitive data types is the object.

7 data types in JavaScript.

JavaScript has different data types, primitive and non-primitive. There are **seven primitive data types, Number, String, Boolean, NULL, Undefined and Symbol** and one non-primitive data type 'object'. There are differences between NULL and undefined data types though both contain the same value.

```
/*
javascript total 7 data type:

1. number
2. string
3. boolean
4. undefined
5. null
6. object
7. symbol
*/

/*
Data types
primitive data types
1. number
2. string
3. boolean
4. undefined
5. null
7. symbol
non-primitive
6. object
*/
```

```
let a="hello";
let b=a;
console.log(a,b);
a="gelo";
console.log(a,b);

const x={job:"web devloper"};
const y=x;
console.log(x,y);
// x.job = "front end developer";
y.job = "front end developer";
console.log(x,y);
```

35-3 Different Truthy and Falsy values in JavaScript

```
/*
Falsy:
false
0
empty string
undefined
null
NaN
-------------------
Truthy:
true
any number (positive or negative )
any string including single whitespace, '0', 'false'
[]
{}
anything else that is not falsy will be truthy
*/
```

35-4 Null Vs Undefined, different ways you will get undefined

```
1. variable value not assigned
2. function but didn't write return
3. just wrote return but didn't return anything
4. parameter that isn't passed
5. property that doesn't exist in an object
6. accessing array element out of range
7. accessing deleted array element
```

8. explicitly set value to undefined

35-5 double equal (==) vs triple equal (===), implicit conversion

triple equals (===) will do the same comparison as double equals (including the special handling for NaN , -0 , and +0) but without type conversion; if the types differ, false is returned.

== in JavaScript is used for **comparing two** variables, but it ignores the datatype of variable. === is used for comparing two variables, but this operator also checks datatype and compares two values. Checks the equality of two operands without considering their type.

```
const first = "0";
const second = false;
if (first == second) {
  console.log("condition is true");
} else {
  console.log("condition is false");
}

//more comparison
// const a = { name: 'ali' };
// const b = { name: 'ali' };
const a = [];
const b = [];
if (a === b) {
  console.log("both are same");
} else {
  console.log("they are not same");
}
```

35-6 Block scope, global scope, Hoisting

যখন একটি ফাংশনের ভিতরে একটি ভেরিয়েবল তৈরি হয় তখন সেটাকে local scope বলে। এটি করার মাধ্যমে ভেরিয়েবলগুলকে শুধু সেই ফাওশনের মধ্যে access করা যায়'.

Global variables গুলাকে জাভাস্ক্রিপ্ট প্রোগ্রামের যেকোনো যায়গা থেকে অ্যাক্সেস করা যায়।

if,switch,conditions বা for এবং while loops এর ভিতরের area কে block scope বলে।

1. Global Scope

2. Local Scope
3. Block Scope

```
const favNum = 27;

function add(first, second) {
  // console.log(mood); // hoisting
  const result = first + second;

  if (result > 9) {
    let mood = "happy";
    mood = "cranky";
  }
  // console.log(mood);
  return result;
}
const sum = add(11, 35);
// console.log(mood);

for (let i = 0; i < 10; i++) {}
// console.log(i);
```

35-7 (advanced) Closure, encapsulation, private variable

What is closure with example?

In the above example, outer function Counter returns the reference of inner function IncreaseCounter(). IncreaseCounter increases the outer variable counter to one. ... As per the closure definition, if **inner function access the variables of outer function** then only it is called closure.

```
function stopWatch() {
  let counter = 0;
  return function () {
    counter++;
    return counter;
  };
}
let clock1 = stopWatch();
console.log(clock1());
console.log(clock1());
```

```
console.log(clock1());
```

35-8 (advanced) Callback function and pass different functions

What is a callback function in JavaScript?

A callback is **a function passed as an argument to another function**. This technique allows a function to call another function. A callback function can run after another function has finished.

```
function welcomeMessage(name, greetHandler) {
  greetHandler(name);
}
// const names = ['Tom Hanks', 'Tom Brady', 'Tom Cruise']
// const myObj = { name: 'Tom Chinku', age: 11 };

function greetMorning(name) {
  console.log("Good morning", name);
}
function greetEvening(name) {
  console.log("Good Evening", name);
}
function greetAfternoon(name) {
  console.log("Good afternoon", name);
}

welcomeMessage("Tom Hanks", greetMorning);
welcomeMessage("Sakib Hanks", greetAfternoon);
welcomeMessage("Bappa Raj", greetEvening);

function Name(name) {
  name();
}

function name(names) {
  console.log("Hamid Hosen");
}

Name(name);

function handleClick() {
  console.log("button is clicked");
```

```
}

document.getElementById("my-btn").addEventListener("click",
handleClick);

document.getElementById("btn").addEventListener("click", function () {
  console.log("buttn is clicked");
});
```

35-9 Module summary pass by reference pass by value