

Answer sheet

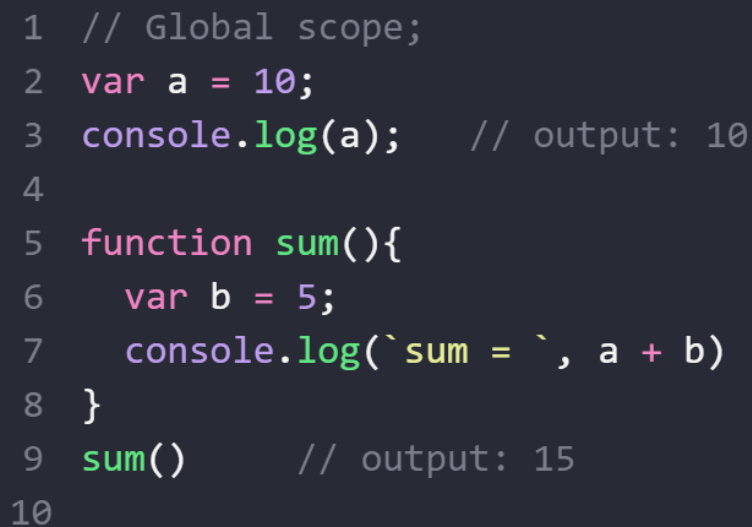
Q1: What is Scope in JavaScript? Explain with example.

Answer: জাভাস্ক্রিপ্টে Scope বলতে বুঝায় কোন ভেরিয়েবল বা ফাংশন কোন কোন জায়গা থেকে অ্যাক্সেস করা যাবে। Scope প্রধানত দুই প্রকার।

1. Global scope
2. local scope

Global scope:

গ্লোবালি কোন ভেরিয়েবল বা ফাংশন ডিক্লেয়ার করা হলে সেটা গ্লোবাল স্কোপে পড়ে। গ্লোবাল স্কোপে ডিফাইন করা কোনো ভ্যারিয়েবল বা ফাংশন যেকোনো জায়গা থেকে এক্সেস করা যায়। যেমন:



```
1 // Global scope;
2 var a = 10;
3 console.log(a); // output: 10
4
5 function sum(){
6     var b = 5;
7     console.log(`sum = `, a + b)
8 }
9 sum() // output: 15
10
```

এখানে `a` কে গ্লোবালি ডিফাইন করা হয়েছে। অর্থাৎ `a` গ্লোবাল স্কোপ। তাই `a` কে ফাংশনের ভিতরে এবং বাহিরে উভয় জায়গায় এক্সেস করা যাচ্ছে।

Local scope:

কোন একটি ভ্যারিবেল বা ফাংশন বা অবজেক্ট কে একটি ফাংশনের ভিতরে ডিফাইন করা হলে ঐ ভ্যারিয়েবল বা ফাংশন বা অবজেক্টটি লোকাল স্কোপ পায়। ফলে ফাংশনের ভিতরে ডিফেন্ড করা ভেরিয়েবল বা ফাংশন বা অবজেক্টটি ফাংশনের ভিতর যে কোন জায়গা থেকে access করা যায়। কিন্তু ফাংশানের বাইরে থেকে এক্সেস করা যায় না। যেমন:

```
1 // Local Scope
2 function localScope() {
3   var a = 10;
4   console.log("from inside the function: a is", a); // from inside the function: a is 10
5 }
6 localScope();
7 console.log("from outside the function: a is", a); // ReferenceError: a is not defined
```

এখানে লোকাল স্কোপ নামে ফাংশনের মধ্যে ডিফেন্ড করা ভেরিয়েবল a ফাংশনের ভিতর এক্সেস করা গেলেও ফাংশনের বাইরে এক্সেস করা যাচ্ছে না। ফাংশানের বাইরে ReferenceError দেখাচ্ছে।

Q2: What is temporal dead zone in JS? Explain with example

Answer: জাভাস্ক্রিপ্টে var কিওয়ার্ড দিয়ে লেখা কোন ভেরিয়েবল ডিক্লেয়ার করার আগে সেটা এক্সেস করা যায়। তবে সেক্ষেত্রে এর ভ্যালু হবে undefined. কারণ এক্সিকিউশন কন্ট্রোলারের ট্রিয়েশন ফেইজে প্রত্যেকটা ভেরিয়েবলের ভ্যালু undefined সেট করা হয়। আর var এর ভ্যালু undefined সেট করা হয় গ্লোবাল অবজেক্টে। তাই ভেরিয়েবল ডিফাইন করার আগে এক্সেস করতে চাইলে undefined আউটপুট আসে।

কিন্তু let বা const কিওয়ার্ড দিয়ে লেখা কোন ভেরিয়েবল ডিক্লেয়ার করার আগে সেটা এক্সেস করা যায় না। let/const দিয়ে কোন ভেরিয়েবল ডিক্লেয়ার করার সাথে সাথে ভ্যালু ইনিশিয়ালাইজ করতে হয়।

let/const এর ভ্যালু undefined সেট করা হয় Script নামে অন্য এক স্থানে। কেউ যদি let/const দিয়ে ভেরিয়েবল ডিক্লেয়ার করে ভ্যালু সেট না করে তাহলে ReferenceError শো করে। আর let / const দিয়ে কিওয়ার্ড দিয়ে লেখা কোন ভেরিয়েবল ডিক্লেয়ার করার আগে সেটা এক্সেস করতে চাওয়াকেই জাভাস্ক্রিপ্টে temporal dead zone বলে।

```

1 // Temporal dead zone
2 console.log(a); // undefined
3 var a = 10;
4 console.log(a); // 10
5
6 console.log(b); // ReferenceError: Cannot access 'b' before initialization
7 let b = 5;
8 console.log(b);
9

```

Q3: What is closure? Explain with example.

Answer: জাভাস্ক্রিপ্টে কোন প্যারেন্ট ফাংশনের ভেরিয়েবল বা অবজেক্ট তার চাইল্ড ফাংশন ব্যবহার করতে পারে। যদি চাইল্ড ফাংশনটাকে রিটার্নও করে দেওয়া হয় তাহলেও সে তার প্যারেন্টের লোকাল ভেরিয়েবলগুলোর এক্সেস নিতে পারে। আর এটাই হচ্ছে ক্লোজার (closure)। যেমন:

```

1 // Closure
2 function myFunc() {
3   let a = 10;
4   return function () {
5     let b = 5;
6     console.log(a - b);
7   };
8 }
9
10 myFunc(); // output: 5

```

Q4: What is 'use strict' and what is the benefits of using it?

Answer: জাভাস্ক্রিপ্টে use strict ব্যবহার করা হয় বিভিন্ন unpredictable behaviour রোধ করার জন্য। জাভাস্ক্রিপ্টে কখনো কখনো weird behave করে। যেমন:

```
1 // without "use strict";
2 name = "Nazmul";
3 console.log(name); // output: Nazmul
4
5 function printAge() {
6   console.log(this.age);
7 }
8 printAge(); // output: undefined
```

এখানে name ভ্যারিয়েবল ডিফাইন না করেই এক্সেস করা যাচ্ছে। আবার printAge ফাংশন this.age এর ভ্যালু undefined দেখাচ্ছে। অথচ age ডিফাইনই করা হয়নি।

এ ধরনের unpredictable behaviour অনেক ক্ষেত্রেই প্রবলেম করতে পারে। বিশেষ করে বড় এপ্লিকেশনে এ ধরনের অনেক অনাকাঙ্ক্ষিত আচরণ অনেক সমস্যার সৃষ্টি করতে পারে। এক্ষেত্রে use strict ব্যবহার করলে এই ধরনের unpredictable behaviour থেকে বাঁচা যায়।

```
1 // using "use strict"
2
3 "use strict";
4 name = "Nazmul";
5 console.log(name); // ReferenceError: name is not defined
6
7 function printAge() {
8   console.log(this.age);
9 }
10 printAge(); // TypeError: Cannot read properties of undefined (reading 'age')
```

Q5: What will be the following code output and why?

`{a: 1} == {a: 1}`

Answer: false.

দুটি অবজেক্ট কখনো কম্পেয়ার করা যায় না। দুইটা অবজেক্ট কমপেয়ার করলে সব সময় false রিটার্ন করে। যদিও অবজেক্ট গুলো একই হয়।

কারণ: আমরা জানি অবজেক্ট রেফারেন্স টাইপ। দুইটি অবজেক্ট দেখতে একই রকম হলেও তাদের রেফারেন্স ভিন্ন হয়। রেফারেন্স ভিন্ন হওয়ার কারণে দুইটি একই রকম অবজেক্ট == বা === দিয়ে কম্পেয়ার করলে আউটপুট সব সময় false আসে।

Solution of problem No.1

```
1 function fec(num) {
2   num = Number(num);
3   let sum = 1;
4   for (let i = 1; i <= num; i++) {
5     sum *= i;
6   }
7   return sum;
8 }
9 console.log(fec(4));           // output: 24
10 console.log(fec("8"));        // output: 40320
```

Solution of problem No.2

```
1 function QuestionsMarks(str) {
2   let res = false;
3   for (let i = 0; i < str.length; i++) {
4     for (let j = i + 1; j < str.length; j++) {
5       if (Number(str[i]) + Number(str[j]) === 10) {
6         if (str.slice(i + 1, j).split("?").length - 1 === 3) {
7           res = true;
8         } else {
9           res = false;
10        }
11      }
12    }
13  }
14  return res;
15 }
16 console.log(QuestionsMarks("arrb6???4xxbl5???eee5")); // true
17 console.log(QuestionsMarks("aa6?9")); // false
18 console.log(QuestionsMarks("acc?7??sss?3rr1?????5")); // true
```

Solution of problem No.3



```
1 function LongestWord(sen) {  
2   let arr = sen.match(/[a-z0-9]+/gi);  
3   let res = arr.sort((a, b) => b.length - a.length);  
4   return res[0];  
5 }  
6  
7 console.log(LongestWord("Hello world123 567")); // output: world123  
8 console.log(LongestWord("fun&!! time"));         // output: time  
9 console.log(LongestWord("I love dogs"));          // output: love
```