2025-JS-MOCK

Here’s a short JavaScript program demonstrating the use of assignment and arithmetic operators, along with debugging.

```
// Assignment Operator
let a = 10;
let b = 5;

// Arithmetic operations
let sumVal = a + b;  // Addition
let diffVal = a - b; // Subtraction
let prodVal = a * b; // Multiplication
let quotVal = a / b; // Division
let modVal = a % b;  // Modulus (remainder)
let expVal = a ** b; // Exponentiation (10^5)
let floorDiv = Math.floor(a / b); // Floor division (no direct operator in JS)

// Updating variables using assignment operators
a += 3;  // Equivalent to a = a + 3
b *= 2;  // Equivalent to b = b * 2

// Display results
console.log("Sum:", sumVal);
console.log("Difference:", diffVal);
console.log("Product:", prodVal);
console.log("Quotient:", quotVal);
console.log("Modulus:", modVal);
console.log("Exponentiation:", expVal);
console.log("Floor Division:", floorDiv);
console.log("Updated a:", a);
console.log("Updated b:", b);
```

✘

Points: 0/1

1. What will be the output of `let x = 15 / 4; console.log(Math.floor(x));`?

- Select - ✘

Correct answer: 3

✘

Points: 0/1

2. Which of the following is an example of an assignment operator in JavaScript?

- Select - ✘

Correct answer: +=

✘

Points: 0/1

3. What is the result of `console.log(7 % 3);`?

- Select - ✘

Correct answer: 1

✘

Points: 0/1

4. What is the correct operator for exponentiation in JavaScript?

- Select - ✘

Correct answer: **

✘

Points: 0/1

5. Which of the following is the best practice for declaring variables in modern JavaScript?

- ○ Using var for all variables
- ○ Using only const for all variables
- ○ Using let and const instead of var ✓
- ○ Declaring all variables globally

✘

Points: 0/1

6. What is the recommended way to compare values in JavaScript?

- ○ Using === ✓
- ○ Using ==
- ○ Using =
- ○ Using !=

✘

Points: 0/1

7. Why is it a best practice to use `try...catch` for error handling in JavaScript?

- ○ It helps prevent syntax errors
- ○ It ensures that the program doesn’t break unexpectedly ✓
- ○ It makes the code slower
- ○ It improves performance

✘

Points: 0/1

8. What is the best practice when working with asynchronous code in JavaScript?

- ○ Writing nested callbacks for better control
- ○ Using `setTimeout()` for all asynchronous operations
- ○ Using async/await instead of promise chaining where possible ✓
- ○ Ignoring asynchronous errors

✘

Points: 0/1

9. What is the main advantage of using external JavaScript files instead of internal scripts?

- ○ Internal scripts are faster than external scripts
- ○ External scripts must always be loaded at the start of the
- ○ External scripts cannot be reused
- ○ External scripts make the webpage load faster by reducing HTML size ✓

✘

Points: 0/1

10. How can an external JavaScript file be linked to an HTML document?

- ○ `<script src="script.js"></script>` ✓
- ○ `<script href="script.js"></script>`
- ○ `<script link="script.js"></script>`
- ○ `<js include="script.js"></js>`

✘

Points: 0/1

11. What is a potential drawback of using an internal script inside an HTML file?

- ○ Internal scripts increase HTML file size and reduce maintainability ✓
- ○ Internal scripts must be written inside a
  - ○ Internal scripts cannot use JavaScript functions
  - ○ Internal scripts are difficult to modify

✘

Points: 0/1

12. Which of the following best describes when to use an internal script?

- ○ When the script is small and specific to a single page ✓
- ○ When external files are not supported by the browser
- ○ Internal scripts should always be used over external scripts
- ○ When the script is large and used across multiple pages

✘

Points: 0/1

13. How can you ensure an external JavaScript file loads after the HTML content?

- ○ By using `<link>` instead of `<script>`
- ○ By placing `<script src="script.js"></script>` before the closing `</body>` tag ✓
- ○ By adding the `async` attribute to the `<script>` tag
- ○ By placing `<script src="script.js"></script>` inside the `<head>`

JavaScript has **seven** primitive data types: `string`, `number`, `boolean`, `null`, `undefined`, `bigint`, and `symbol`. Below is a **JavaScript program** that demonstrates the use of these data types.

```
// Declaring variables of primitive data types
let myString = "Hello, JavaScript!";  // String
let myNumber = 42;                    // Number
let myBoolean = true;                 // Boolean
let myNull = null;                    // Null
let myUndefined;                      // Undefined (not initialized)
let myBigInt = 1234567890123456789 0n;   // BigInt
let mySymbol = Symbol("unique");      // Symbol

// Displaying values in console
console.log("String:", myString);
console.log("Number:", myNumber);
console.log("Boolean:", myBoolean);
console.log("Null:", myNull);
console.log("Undefined:", myUndefined);
console.log("BigInt:", myBigInt);
console.log("Symbol:", mySymbol);
```

✘

Points: 0/1

14. What is the output of `typeof null` in JavaScript?

- Select - ✘

Correct answer: "object"

✘

Points: 0/1

15. Which primitive data type is used to store large integer values beyond the `Number` limit?

- Select - ✘

Correct answer: BigInt

✘

Points: 0/1

16. What will be the output of `console.log(typeof myUndefined);` if `myUndefined` is declared but not assigned a value?

- Select - ✘

Correct answer: "undefined"

✘

Points:
0/1

17. Which of the following correctly declares a Symbol in JavaScript?

- ○ `let sym = new Symbol("id");`
- ○ `let sym = Symbol("id");` ✓
- ○ `let sym = Symbol.new("id");`
- ○ `let sym = symbol("id");`

✘

Points:
0/1

18. What is the difference between `null` and `undefined` in JavaScript?

- ○ undefined is a number, while null is an object
- ○ null means "not declared," while undefined means "empty"
- ○ null is an empty value assigned by the user, while undefined is automatically assigned to uninitialized variables ✓
- ○ Both are the same and interchangeable

JavaScript arrays are used to store multiple values in a single variable. Below is a **JavaScript program** demonstrating different ways to declare and use arrays.

```
// Declaring arrays using different methods
let numbers = [10, 20, 30, 40, 50];  // Array of numbers
let fruits = ["Apple", "Banana", "Cherry"]; // Array of strings
let mixedArray = [1, "Hello", true, null]; // Mixed data types

// Accessing array elements
console.log("First element of numbers:", numbers[0]);
console.log("Second element of fruits:", fruits[1]);

// Modifying an array element
numbers[2] = 99;
console.log("Modified numbers array:", numbers);

// Adding an element to the array
fruits.push("Mango");
console.log("Fruits array after push:", fruits);

// Removing the last element
fruits.pop();
console.log("Fruits array after pop:", fruits);

// Finding length of an array
console.log("Length of mixedArray:", mixedArray.length);
```

✘

Points:
0/1

19. What will be the output of `console.log([1, 2, 3].length);` ?

- Select - ✘

Correct answer: 3

✘

Points:
0/1

20. How can you add an element to the end of a JavaScript array?

- ○ `array.push(element);` ✓
- ○ `array.append(element);`
- ○ `array.insert(element);`
- ○ `array.add(element);`

✘

Points: 0/1

21. What is the correct way to access the second element of an array `let arr = [5, 10, 15, 20];` ?

- Select - ✘

Correct answer: arr[1]

✘

Points: 0/1

22. What will happen if you access an array element that does not exist, like `console.log(arr[10]);` ?

- ○ It will return undefined ✓
- ○ It will throw an error
- ○ It will return 0
- ○ It will return null

✘

Points: 0/1

23. How do you remove the last element from an array in JavaScript?

- ○ `array.remove();`
- ○ `array.pop();` ✓
- ○ `array.shift();`
- ○ `array.delete();`

True/False Questions

✘

Points: 0/1

24. In JavaScript, the `const` keyword can be used to declare variables, but their values can be reassigned later.

- Select - ✘

Correct answer: False

✘

Points: 0/1

25. The `===` operator in JavaScript checks for both value and type equality.

- Select - ✘

Correct answer: True

✘

Points: 0/1

26. The `typeof null` in JavaScript returns `"null"`.

- Select - ✘

Correct answer: False

✘

Points: 0/1

27. Arrays in JavaScript can store different data types in the same array.

- Select - ✘

Correct answer: True

✘

Points: 0/1

28. The `push()` method in JavaScript removes the last element of an array.

- Select - ✘

Correct answer: False

Here’s a simple JavaScript program that demonstrates basic concepts like functions, loops, and conditionals.

```
function isPrime(num) {
    if (num < 2) return false;
    for (let i = 2; i <= Math.sqrt(num); i++) {
        if (num % i === 0) {
            return false;
        }
    }
    return true;
}

function printPrimes(limit) {
    console.log(`Prime numbers up to ${limit}:`);
    for (let i = 2; i <= limit; i++) {
        if (isPrime(i)) {
            console.log(i);
        }
    }
}

// Example usage
let limit = 20;
printPrimes(limit);
```

✘

Points: 0/1

29. What will the output be when `printPrimes(10);` is called?

- 1, 2, 3, 5, 7
- 2, 4, 6, 8, 10
- None of the options
- 2, 3, 5, 7 ✓

✘

Points: 0/1

30. What is the time complexity of the `isPrime(num)` function?

- Select - ✘

Correct answer: O(√n)

✘

Points: 0/1

31. What does `if (num < 2) return false;` check for?

- If the number is less than 2, it's not prime ✓
- If the number is greater than 2
- If the number is even
- None of the options

✘

Points: 0/1

32. What is the purpose of `Math.sqrt(num)` in the loop condition?

- ○ To check if the number is even
- ○ To reduce unnecessary iterations ✓
- ○ To find the square of the number
- ○ To double the number

✘

Points: 0/1

33. What keyword is used to define a function in JavaScript?

- Select - ✘

Correct answer: function

✘

Points: 0/1

34. What happens if you call `printPrimes(1);` ?

- ○ Throws an error
- ○ Prints 1
- ○ Prints 2
- ○ Prints "Prime numbers up to 1:" and nothing else ✓

Here's a **JavaScript program** that demonstrates the use of `Math.random()`, `Math.round()`, `Math.abs()`, `Math.floor()`, `Math.ceil()`, `Math.min()`, `Math.max()`, `Math.pow()`, and `Math.sqrt()`.

```
// Generate a random number between 1 and 100
let randomNum = Math.floor(Math.random() * 100) + 1;
console.log("Random Number:", randomNum);

// Rounding functions
console.log("Round(4.7):", Math.round(4.7));
console.log("Ceil(4.2):", Math.ceil(4.2));
console.log("Floor(4.9):", Math.floor(4.9));

// Absolute value
console.log("Abs(-10):", Math.abs(-10));

// Power and square root
console.log("2^3:", Math.pow(2, 3));
console.log("Sqrt(25):", Math.sqrt(25));

// Minimum and Maximum
console.log("Min(5, 10, 2):", Math.min(5, 10, 2));
console.log("Max(5, 10, 2):", Math.max(5, 10, 2));
```

✘

Points: 0/1

35. What does `Math.random()` return?

- ○ A whole number between 0 and 1
- ○ A random integer
- ○ A random decimal between 0 and 1 ✓
- ○ A random number between 1 and 100

✘

Points: 0/1

36. What is the output of `Math.floor(7.8)` ?

- Select - ✘

Correct answer: 7

✘

Points: 0/1

37. What will `Math.pow(3, 2)` return?

- Select - ✘

Correct answer: 9

✘

Points: 0/1

38. What will `Math.max(4, 8, 2, 9, 1)` return?

- Select - ✘

Correct answer: 9

✘

Points: 0/1

39. What will `Math.abs(-15)` return?

- Select - ✘

Correct answer: 15

✘

Points: 0/1

40. What is the difference between local and global scope in JavaScript?

- ○ Local variables are accessible everywhere, while global variables are limited to functions.
- ○ Global variables are declared inside functions, while local variables are declared outside functions.
- ○ Local variables are declared inside a function and are accessible only within that function, while global variables are accessible throughout the program. ✓
- ○ There is no difference between local and global scope in JavaScript.

✘

Points: 0/1

41. What happens when you redefine a `var` variable inside a function that was already declared globally?

- ○ It modifies the global variable.
- ○ It deletes the global variable.
- ○ It creates a new local variable inside the function without affecting the global variable. ✓
- ○ It causes an error.

✘

Points: 0/1

42. How are primitive data types (like numbers) passed to a function in JavaScript?

- ○ By both value and reference
- ○ They are not passed at all
- ○ By reference
- ○ By value ✓

✘

Points: 0/1

43. How are objects passed to a function in JavaScript?

- ○ By value
- ○ By both value and reference
- ○ By reference ✓
- ○ Objects cannot be passed to functions

✘

Points: 0/1

44. What is the return value of the following function?

```
function add(a, b) {
    return a + b;
}
console.log(add(5, 10));
```

- Select - ✘

Correct answer: 15

✘

Points: 0/1

45. What will be the output of the following code?

```
let x = 10, y = 20;
console.log(x != y && x < y);
```

- Select - ✘

Correct answer: true

✘

Points: 0/1

46. What will be the output of the following code?

```
let p = false, q = true;
console.log(!(p && q) == (p || !q));
```

- Select - ✘

Correct answer: true

✘

Points: 0/1

47. What is the output of the following switch statement?

```
let fruit = "Mango";
switch (fruit) {
    case "Apple":
        console.log("Apple selected");
        break;
    case "Mango":
        console.log("Mango selected");
    case "Banana":
        console.log("Banana selected");
        break;
    default:
        console.log("Unknown fruit");
}
```

- ○ Error
- ○ Mango selected and Banana selected ✓
- ○ Unknown fruit
- ○ Mango selected

✘

Points: 0/1

48. What will be the output of the following `for` loop?

```
for (let i = 0; i < 5; i++) {
    if (i == 3) {
        break;
    }
    console.log(i);
}
```

- ○ Error
- ○ 0 1 2 ✓
- ○ 0 1 2 3
- ○ 0 1 2 3 4

✘

Points: 0/1

49. What will be the output of the following `for-in` loop?

```
let obj = { a: 1, b: 2, c: 3 };
for (let key in obj) {
    console.log(key);
}
```

- ○ 1 2 3
- ○ a 1 b 2 c 3
- ○ Error
- ○ a b c ✓

✘

Points: 0/1

50. Which event fires when a webpage has completely loaded?

```
<body onload="alert('Page Loaded!')">
</body>
```

- Select - ✘

Correct answer: onload

✘

Points: 0/1

51. What does the `onfocus` event do?

- ○ Fires when an element loses focus
- ○ Fires when the mouse moves over an element
- ○ Fires when a key is pressed
- ○ Fires when an element gains focus ✓

✘

Points: 0/1

52. What will happen when you type in the following text input?

```
<input type="text" onkeydown="console.log('Key Pressed!')">
```

- ○ Nothing happens
- ○ Logs "Key Pressed!" when the input loses focus
- ○ Logs "Key Pressed!" when a key is released
- ○ Logs "Key Pressed!" when a key is pressed ✓

✘

Points: 0/1

53. What happens when the mouse moves over and then moves out of an element?

```
<div onmouseover="console.log('Mouse Entered!')" onmouseout="console.log('Mouse Left!')">
  Hover over me
</div>
```

- ○ Nothing happens
- ○ Logs "Mouse Entered!" when hovered and "Mouse Left!" when moved out ✓
- ○ Logs "Mouse Left!" only once
- ○ Logs "Mouse Entered!" only once

✘

Points: 0/1

54. What will be the output of the following JavaScript code?

```
document.write("<h1>Hello, World!</h1>");
```

- ○ Displays "document.write is not defined"
- ○ Prints "Hello, World!" in the console
- ○ Displays `"Hello, World!"` inside an `<h1>` tag on the webpage ✓
- ○ Does nothing

✘

Points: 0/1

55. What is the difference between `innerHTML` and `textContent`?

- ○ innerHTML only retrieves text, while textContent retrieves HTML
- ○ innerHTML is used for forms, while textContent is used for divs
- ○ textContent only retrieves text, while innerHTML retrieves both text and HTML ✓
- ○ There is no difference

✘

Points: 0/1

56. What will be the output of the following code?

```
<div id="demo">Hello <b>World</b>!</div>
<script>
   let element = document.getElementById("demo");
   console.log(element.textContent);
</script>
```

- ○ Error
- ○ "Hello World!" ✓
- ○ "Hello **World**!"
- ○ "**World**"

✘

Points: 0/2

57. What does `getElementById("demo")` return? (Select both the right answers)

```
let element = document.getElementById("demo");
```

- ☐ A NodeList of elements with the ID "demo"
- ☐ null if no element with ID "demo" exists ✓
- ☐ An array of elements with the ID "demo"
- ☐ The first element with the ID "demo" ✓

✘

Points:
0/1

58. What will `getElementsByTagName("p")` return?

let paragraphs = document.getElementsByTagName("p");
console.log(paragraphs.length);

- ○ A single element
- ○ null if no elements exist
- ○ An array of all elements
- ○ A live HTMLCollection of elements ✓

✘

Points:
0/1

59. What does the `setAttribute()` method do?

let element = document.getElementById("demo");
element.setAttribute("class", "newClass");

- ○ Does nothing if the attribute already exists
- ○ Adds an attribute only if it doesn't exist
- ○ Modifies or adds the specified attribute ✓
- ○ Removes an attribute from an element

✘

Points:
0/1

60. What will the following `createElement()` code do?

let newDiv = document.createElement("div");
newDiv.textContent = "Hello!";
document.body.appendChild(newDiv);

- ○ Logs `"Hello!"` to the console
- ○ Replaces the entire document with a `<div>` containing `"Hello!"`
- ○ Creates a new `<div>` with `"Hello!"` inside and adds it to the document ✓
- ○ Throws an error