

Instructions: Answer all questions. Write code where required and explain your answers clearly.

Part 1: Variables and Scope

1. Explain how var works in JavaScript. What is variable hoisting? Give a code example.

- a. **Var** => it's a keyword used to declare variables. Variables declared with var are either function-scoped or globally-scoped

i.

```
var x = 3
console.log(x)
```

ii.

```
> |
function print(){
  var x = 4
  console.log(x)
}
print()
```

iii.

```
4
```

iv.

```
>
```

- b. **Variable hoisting** => all declaration of variables move to the top of their containing scope (function scope or global scope). Only the declaration is hoisted not the assignment of a value.

i.

```
console.log(y)
var y = 4
```

ii.

```
>
function print(){
  console.log(x)
  var x = 4
}
print()
```

iii.

```
undefined
```

iv.

```
>
```

2. What is the scope of a variable declared with var inside a function? What about inside a block (e.g., an if statement)?

- a. Inside a function it is a **function scope**.
- b. Inside a block it is a **global scope**.

3. List all JavaScript primitive types in ES5. Give an example of each.

a. Number

```
> var x = 123
< undefined
> typeof(x)
i. < 'number'
```

b. String

```
> var y = "Ahmed"
< undefined
> typeof(y)
i. < 'string'
```

c. Boolean

```
> var z = true
< undefined
> typeof(z)
i. < 'boolean'
```

d. Undefined

```
> var b
< undefined
> typeof(b)
i. < 'undefined'
```

e. Null

```
> var a = null
< undefined
> typeof(a)
i. < 'object'
```

- ii. Null represents the absence of any object value

4. What is the difference between a primitive type and an object type? Give an example where this difference is important.

a. Primitive type

- i. Directly stored (the variable stores the assigned value)
- ii. Passed by value

b. Object type

- i. Stored by reference (the variable stores a reference to the object's location in the memory)
- ii. Passed by reference

c. Example:

```

> var x = 5, y=5
< undefined
> typeof(x)
< 'number'
> typeof(y)
< 'number'
> x==y
< true

```

i.

1. In the first case x and y stores the same value which is 5 so x==y returns true

```

> var x = new Number(5), y = new Number(5)
< undefined
> typeof(x)
< 'object'
> typeof(y)
< 'object'
> x==y
< false

```

ii.

1. While in the second case x and y stores different references so x==y returns false

5. Create a number, string, and boolean using both literal and constructor syntax. Show the difference in their types using typeof.

```

> var x=5,y=true,z='Hello'
< undefined
> typeof(x)
< 'number'
> typeof(y)
< 'boolean'
> typeof(z)
< 'string'
> var x = new Number(5), y = new Boolean(true), z = new String('Hello')
< undefined
> typeof(x)
< 'object'
> typeof(y)
< 'object'
> typeof(z)
< 'object'

```

a.

6. Why is it generally recommended to use literals instead of constructors for primitive types?

- Applying the KISS principle. If we don't need anything beyond a simple container of data, we go with a simple literal.
- If we don't have behavior associated with an object, we should use simple literal.

7. Given the following code, what will be the output? Explain why.

```
var x = 123.4567;  
console.log(x.toFixed(2));  
console.log(x.toPrecision(4));
```

- ```
123.46
```
- ```
123.5
```
- toFixed(2) returns 2 digits after the point and round the number
- toPrecision(4) returns exactly 4 digits from x and round the number

8. What is NaN? How can you check if a value is NaN? Give an example.

- NaN => not a number
- Check it using isNaN() or Number.isNaN()

```
> var x = 'a123'  
⏏ undefined
```

```
> parseInt(x)  
⏏ NaN
```

c.

```
> var x = ''  
⏏ undefined
```

```
> parseInt(x)
```

d.

```
⏏ NaN
```

9. What is the difference between parseInt, parseFloat, and Number? Give an example for each.

- parseInt
 - Parses a string and returns an integer

```
> parseInt("42")
```
 - ```
⏏ 42
```
- parseFloat
  - Parses a string and returns a floating-point number.

```
> parseFloat("4.1")
```
  - ```
⏏ 4.1
```

c. Number

- i. Converts a value to a number (integer or float).

```
> Number("42.9")
```

```
< 42.9
```

```
> Number("42")
```

```
< 42
```

ii.

10. What is the difference between implicit and explicit type casting? Give an example of each.

a. Implicit Type Casting

- i. Done automatically by the compiler/interpreter.

```
> 5 + "10"
```

```
< '510'
```

ii.

b. Explicit Type Casting

- i. Done manually

```
> Number("42.9")
```

```
< 42.9
```

```
> Number("42")
```

```
< 42
```

ii.

11. What will be the result and type of the following expressions? Explain your answer.

a. true + 5

- i. The result => 6
ii. True will implicitly be casted to 1

b. "10" - 2

- i. The result => 8
ii. "10" will implicitly be casted to 10

c. 12 - "1a"

- i. The result => NaN
ii. "1a" will be NaN so 12 - NaN = NaN

d. 5 / 0

- i. The result => Infinity
ii. In javascript dividing by 0 doesn't throw an error

e. 5 + undefined

- i. The result => NaN
ii. Undefined will be converted to NaN, so 5+NaN = NaN

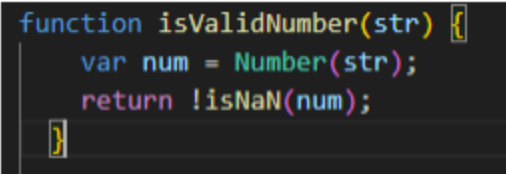
12. What will be logged to the console in the following code? Explain each step.

- a. `var a = "15.5";`
 - i. `a` is a string
- b. `var b = +a;`
 - i. `+` operator converts `a` to a number
- c. `console.log(b, typeof b);`
 - i. the value of `b` (15.5) and its type ("number").

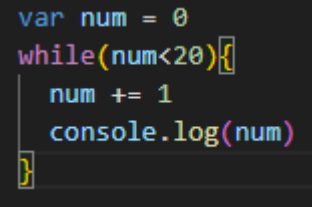
13. What will be the output of:

- a. `var result = 20 > true < 5 == 1;`
- b. `console.log(result);`
- c. Explain why.
 - i. The result is true
 - ii. `20 > true` will result true (true is converted to 1)
 - iii. `true < 5` will result true (true is converted to 1)
 - iv. `true == 1` will result true (true is converted to 1)

14. Write a function that takes a string and returns true if it can be converted to a valid number, and false otherwise.

a. A code snippet in a dark-themed editor showing a function `isValidNumber(str)` that returns `!isNaN(Number(str))`.

15. Write a program that prints all numbers from 1 to 20 using a while loop.

a. A code snippet in a dark-themed editor showing a `while` loop that increments `num` from 0 to 20 and logs each value.

16. Write a program that asks the user to enter numbers until they enter 0, using a **do...while** loop. After the loop ends, print the sum of all entered numbers (excluding 0).

a.

```
var sum = 0;
var num;

do {
  num = Number(prompt("Enter a number (0 to stop):"));
  if (num !== 0 && !isNaN(num)) {
    sum += num;
  }
} while (num !== 0);

console.log(sum);
```

17. Write a program that takes a number from 1 to 7 and prints the corresponding day of the week using a **switch** statement. Use a **for** loop to test your program with all numbers from 1 to 7.

a.

```
function DayOfWeek(num) {
  var day;
  switch (num) {
    case 1: day = "Saturday"; break;
    case 2: day = "Sunday"; break;
    case 3: day = "Monday"; break;
    case 4: day = "Tuesday"; break;
    case 5: day = "Wednesday"; break;
    case 6: day = "Thursday"; break;
    case 7: day = "Friday"; break;
    default: day = "Invalid number";
  }
  return day;
}

for (var i = 1; i <= 7; i++) {
  console.log(DayOfWeek(i));
}
```