JavaScript Assignment – Lecture 1

Part 1: Variables and Scope

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

- Declare a variable in a function scope, and could be globally scoped if it's declared outside the function.
- Hoisting: moving the declaration of variable to the top of the scope, but assigning it as undefined variable.

```
  Live reload enabled.                                    (index):38
> console.log(x); var x=5;
  undefined                                               VM251:1
<· undefined
> console.log(x);
  5                                                       VM310:1
<· undefined
>
```

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

- The scope is function-scope.
- Global-scope.

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

1. Number
2. String
3. Bool
4. Undefined
5. Null

```
  Live reload enabled.                                    (index):38
> var num = 123;console.log(num);
  123                                                     VM929:1
<· undefined
> var str = "Hello";console.log(str);
  Hello                                                   VM933:1
<· undefined
> var bool_val = true;console.log(bool_val);
  true                                                    VM937:1
<· undefined
> var undef_val;console.log(undef_val);
  undefined                                               VM941:1
<· undefined
> var null_val = null;console.log(null_val);
  null                                                    VM945:1
<· undefined
>
```

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

Primitive:

- Immutable
- Stores the value; so, it can be copied and preserve its data regardless the copy made.

Object Type:

- Mutable
- Store the reference; so, the copy affects the original.
- Can be used explicitly with methods.

```
> x=5; y=5;
<· 5
> x==y
<· true
> x=new Number(6); y=new Number(6);
<· ▶ Number {6}
> x==y
<· false
>
```

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

```
> x=5;
<· 5
> obj = new Number(5);
<· ▶ Number {5}
> typeof obj;
<· 'object'
> typeof x;
<· 'number'
```

```
  Live reload enabled.                    (index):38
> x='a'; obj= new String ('a');
<· ▶ String {'a'}
> typeof x
<· 'string'
> typeof obj
<· 'object'
>
```

```
> x = true;  obj  = new Boolean(true);
<· ▶ Boolean {true}
> typeof x;
<· 'boolean'
> typeof obj
<· 'object'
>
```

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

1. Simplicity
2. Readability
3. Performance
4. Objects Are Always Truthy

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

var x = 123.4567;

console.log(x.toFixed(2));

allow only 2 dig after the decimal point, not rounding. Casting to string.

console.log(x.toPrecision(4));

allow 4 dig in general, rounding after the point. Casting to string.

```
> var x = 123.4567;
  console.log(x.toFixed(2));
  console.log(x.toPrecision(4));
123.46                                  VM2259:2
123.5                                   VM2259:3
```

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

"Not a Number": its type is Number.

```
ƒ (number)
> isNaN(0/0)
<· true
```

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

ParseInt: convert string into Integer.

ParseFloat: convert string into float.

Number: convert string or bool (strictly) into number.

```
> parseInt("5")
<· 5
> parseFloat("5.5")
<· 5.5
> Number(true)
<· 1
```

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

Implicit: done automatically:

+ : concatenate

- : cast to number & subtract

```
Live reload enabled.                    (index):38
> x="6"
<· '6'
> x-1
<· 5
> x="6"
<· '6'
> x+"amr"
<· '6amr'
>
```

Explicit: done by creating object of the intended class.

```
Live reload enabled.                    (index):38
> x="6"
<· '6'
> y=Number(x);
<· 6
```

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

- **true + 5 :** as True equals 1, where False equals 0.

```
> true+5
<· 6
>
```

- **"10" – 2:** implicit casting to number, in case of (-): cast str into number

```
> "10"-2
<· 8
```

- **12 - "1a" :** implicit casting to Number, founding “a” result in not valid number and a NaN value.

```
> 12 - "1a"
<· NaN
```

- **5 / 0:** Number; infinity as it’s pos / 0.

```
> 5/0
<· Infinity
```

- **5 + undefined :** casting undefined to number results in NaN.

```
> 5 + undefined
<· NaN
>
```

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

var a = "15.5";

var b = +a;

console.log(b, typeof b);

it's **Unary plus** which converts string into number.

```
> var a = "15.5";
    var b = +a;
    console.log(b, typeof b);

  15.5 'number'
```

13. What will be the output of:

var result = 20 > true < 5 == 1;

console.log(result);

Explain why.

True.

It goes like this:

- 20>true → 20>1 → true
- True<5 → 1<5 → True
- True ==1 → 1==1 → True.

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

```
JS script.js > couldbeconverted
1  function couldbeconverted(str){
2      x=Number(str)
3      return !Number.isNaN(x);
4  }
```

```
> couldbeconverted("5")
<· true
>
```

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

```
18
19
20
<· 20
> i=1
while (i <= 20) {
  console.log(i);
  i++;
}
```

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).

```js
sum = 0;
var num;

do {
  num = parseInt(prompt("Enter a number"));
  if (num !== 0) {
    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.

```js
function day(num) {
    switch (num) {
      case 1:
        return "Saturday";
      case 2:
        return "Sunday";
      case 3:
        return "Monday";
      case 4:
        return "Tuesday";
      case 5:
        return "Wednesday";
      case 6:
        return "Thursday";
      case 7:
        return "Friday";
    }
  }

  for (let i = 1; i <= 7; i++) {
    console.log(i, day(i));
  }
```

```
1 'Saturday'        script.js:21
2 'Sunday'          script.js:21
3 'Monday'          script.js:21
4 'Tuesday'         script.js:21
5 'Wednesday'       script.js:21
6 'Thursday'        script.js:21
7 'Friday'          script.js:21
Live reload enabled.  (index):38
> day(1)
<· 'Saturday'
```