JavaScript Quiz

Q1: What is the output of following code snippet ?

```
const func = _ => {
  let x = y = 0;
  ++x;
  return y;
}

func();
console.log(typeof x, typeof y);
```

A. undefined number
B. Error
C. int int
D. number number

Ans - (A) `let x = y = 0;` declares a local variable x and a global variable y since we are directly assigning to y. It is equivalent to writing

```
let x = 0;
global.y = 0;
```

So `typeof x` gives `undefined` as x is not defined in this scope.
And `typeof y` gives number

Q2: What is the output of following code snippet?

```
const fruits = ['Apple', 'Orange', 'Mango'];
fruits.length = 0;
console.log(fruits[1]);
```

A. undefined
B. Error (Programmer cannot set length property);
C. Error (Index out of bounds as fruits array is empty)
D. Orange

Ans - (A) When user sets the length of array, JavaScript automatically resizes the array.
So `fruits.length = 0` deletes all entries of the fruits array
And when we try to access element at index out of array boundary, JavaScript doesn't throw error. Rather it returns undefined.

Q3: What is the output of following code snippet ?

```
const arr = [];
for(var i = 0; i < 4; ++i);{
  arr.push(i + 1);
}
console.log(arr);
```

A. []
B. [1, 2, 3 ,4]
C. Error
D. [5]

Ans - (D) If you answered `[1, 2, 3, 4]`, you probably missed a `;` after the `for` loop. This creates an empty loop which runs 4 times. After loop termination, value of `i` is `4`. Since we have declared i using `var` keyword and not `let`, this means we can access `i` even outside loop. Next we have a block which pushes 5 to `arr`. If you answered `Error` because `arr` is declared `const`, then you are wrong because here we are not reassigning `arr`. Modifying current object/array is fine even when declared `const`.

Q4: What is the output of following code snippet ?

```
const getArr = (item) => {
  return
    [item];
}

console.log(getArr(3));
```

A. Error
B. undefined
C. [3]
D. null

Ans - (B) Following code snippets are equivalent

```
  return
    [item];
```

```
  return;
    [item];
```

JavaScript automatically inserts `;` when user skips it.

Q5: What is the output of following code snippet (Ignoring newline character added by console.log)?

```
let i;
for(i = 0; i < 3; ++i){
    const myFunc = () => {
        console.log(i);
    }

    setTimeout(myFunc, 100);
}
```

A. 1 2 3
B. 3 2 1
C. 3 3 3
D. 1 1 1

Ans (C): Here we have closure named myFunc which captures variable i lexically. `setTimeout` functions waits for 100ms before executing the closure. These closures will be executed after loop terminates and value of i becomes 3. Hence all of them prints 3.
To learn more about closures and scopes you can look at this.

Q6: What is the output of following code snippet ?

```
let f = 0;
for(let i = 0; i < 10; ++i){
  f += 0.1;
}

if(f === 1){
  console.log(1);
}
else if(f == 1){
  console.log(2);
}
else if(f < 1){
  console.log(3);
}
else {
  console.log(4);
}
```

A. 1
B. 2
C. 3
D. 4

Ans - (C): This is not related to JavaScript but precision of floating point numbers in general. Computers store numbers in binary format and if you try to convert 0.1 to binary it requires infinite digits to store. But we have only 64. So computer stores an approximation. And when we accumulate this error 10 times, it grows and we actually get something around 0.99999.. which is less than 1.

Q7: What is the output of following code snippet ?

```
console.log(x);
console.log(y);
var x = 5;
let y = 6;
```

A. undefined undefined
B. Error on line 1
C. Error on line 2
D. 5 6

Ans (C): Here the difference between `let` and `var` keywords comes into picture. Due to variable hoisting all variable and function declartions float to the top of your program. So when the first line runs. It prints undefined as variable `x` is defined but not assigned a value. But only variables declared using `var` keywords can be hoisted. This causes second line to throw an error as we are trying to access y before declaring it.

Q8: Which of the following statement(s) will NOT cause error ?

A.

```
let . = 1;
console.log([1, 2, 3][.]);
```

B.

```
var ಠ_ಠ = 5;
console.log(ಠ_ಠ + ಠ_ಠ);
```

C.

```
let export = 5;
++export;
```

D.

```
console.log(2 / 0);
```

Ans - (B, D)

A. `.` cannot be used as identifier in JavaScript.
B. We can use unicode charaecters as identifiers in JavaScript.
C. export is a keyword and cannot be used as an identifier
D. Division by 0 return Infinity in JavaScript.

Q9: Which of the following evaluates to true ?

A. `0 == "0"`
B. `0 == []`
C. `"0" == []`
D. `undefined == null`

Ans - (A, B, D) This is actually a very famous JavaScript meme. Check this.

Q10: Why does JavaScript have Java in its name ?
A. JavaScript is a stripped-down version of Java
B. They both originated on the island of Java
C. Coincidence
D. None of the above

Ans - (D): Read This on Quora.