Exercise on Function Scope, Block Scope, Constants

This exercise will test your knowledge on the scope and sequence of function execution. You will have to keep track of the function's path using console.log().

Exercise 1:

Check the following code snippet riddle:

Determine the values logged to the console before you execute it.

```
'use strict';

var guessMe1 = 1;
let guessMe2 = 2;

{

    try {
        console.log( guessMe1, guessMe2 ); // (A)
    } catch(err) {
        console.log("Error");
    }

    let guessMe2 = 3;
    console.log( guessMe1, guessMe2 ); // (B)
}

console.log( guessMe1, guessMe2 ); // (C)

const print_func = () => {

    console.log( guessMe1 ); // (D)
    var guessMe1 = 5;
    let guessMe2 = 6;
    console.log( guessMe1, guessMe2 ); // (E)
};


console.log( guessMe1, guessMe2 ); // (F)
```

Explanation:

The output in the console will be as follows:

The output in the console will be as follows:

```
guessMe1 and/or guessMe2 not defined
1 3
1 2
1 2
```

Let’s examine the six console logs one by one.

(A): Here, `guessMe1` holds the value original value of `1` as it has not been defined again in the block. The error is thrown because of `guessMe2`. Why? This is because we have defined it with the `let` keyword in line 14 of the block:

```
let guessMe2 = 3;
```

If we remove the `try` method and simply print line 9:

```
console.log( guessMe1, guessMe2 );
```

an error will still tell you that `guessMe2` is uninitialized. Hence `Error` was the first output.

(B): This `console.log()` will work because now

`guessMe1 == 1` and `guessMe2 == 3`.

However, keep in mind that these values are only true for this particular block.

(C): This will print the original values of both variables as we have moved outside the scope of the block above.

`guessMe1 == 1` and `guessMe2 == 3`.

(D): The `print_func` function is never called in the program, hence its contents never run.

(E): Just as for (D), the `print_func` function is never called in the program, hence its contents never run.

(F): The `console.log` in this line is the same as that in (C). The values of both variables have not been altered anywhere in the global scope. Hence, the output is the same.

Exercise 2:

Modify the code such that all six console logs print out their values exactly once, and the printed values are the following:

```
1 3
1 3
1 2
5
5 6
1 2
```

You are not allowed to touch the console logs, just the rest of the code.

```
//Edit this code
'use strict';

var guessMe1 = 1;
let guessMe2 = 2;

{

    try {
        console.log( guessMe1, guessMe2 ); // (A)
    } catch(err){console.log("Error");}

    let guessMe2 = 3;
    console.log( guessMe1, guessMe2 ); // (B)
}

console.log( guessMe1, guessMe2 ); // (C)

const print_func = () => {

    console.log( guessMe1 ); // (D)
    var guessMe1 = 5;
    let guessMe2 = 6;
    console.log( guessMe1, guessMe2 ); // (E)
};


console.log( guessMe1, guessMe2 ); // (F)
```

Explanation:

The code in the solution is pretty self-explanatory. However, this is not the only solution.

The basic idea is to clearly define `guessMe1` and `guessMe2` before each `console.log()` call.