## Joel Turbi

## Dr. Zavala

## CS350: Programming Language Paradigms Homework Assignment 4

**var, let and const – what's the difference?**

Go to the following for this assignment: <https://dev.to/sarah_chima/var-let-and-const--whats-the-difference-69e>

As you read each section, complete the table and answer the questions below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Scope  (global, function/locally, block) | Initial value when hoisted | Can be redeclared (yes or no) | Can be updated (yes or no) |
| **var** | Function / Global | Hoisted to the top of its scope and initialized with a value of  “**Undefined**”. | Yes | Yes |
| **const** | Block | Hoisted to the top of its scope,  but it is not initialized. | No | No |
| **let** | Block | Hoisted to the top of its scope, but unlike “**var**” it is not initialized. If used before declaration it returns a “**Reference Error**”. | No | Yes |

# VAR

1. What is the output of the following program?

* The output of the following program would be:
  + **Undefined**
  + **Undefined**

1. Why is that? Explain whether scope, hoisting, redeclaring and updating have anything to do with it and how.

* This is due to the declaration and initialization of both variables (a & b) at the end of the code. Therefore, hoisting both variables to the top of its scope and forcing it to be initialized as undefined.

function test(){

console.log(a); console.log(b);

}

test();

var a=1, b=2;

# CONST

1. What is the output of the following program?

* The output of the following program would be:
  + **Referenced Error**

1. Why is that? Explain whether scope, hoisting, redeclaring and updating have anything to do with it and how.

* This is due to both variables (a & b) being used before its declaration. Const variables just like let are also hoisted to the top of its scope, but not initialized. Hence, why we get a reference error.

function test(){

console.log(a); console.log(b);

}

test();

const a=1, b=2

# LET

1. What is the output of the following program?

* The output of the following program would be:
  + **Reference Error**

1. Why is that? Explain whether scope, hoisting, redeclaring and updating have anything to do with it and how.

* This is due to both variables (a & b) being used before its declaration. Const variables just like let are also hoisted to the top of its scope, but not initialized. Hence, why we get a reference error.

function test(){

console.log(a); console.log(b);

}

test();

let a=1, b=2;

**VAR, CONST, and LET**

1. What is the output of the following program?

* The output of the following program would be:
  + 100
  + 2

1. What is the output of the program after you uncomment the line “console.log(c);”?

* The output of the program after uncommenting the line “console.log(c);” would be:
  + 100
  + 2
  + Error Reference – c is not defined.

1. Why is that? Explain whether scope, hoisting, redeclaring and updating have anything to do with it and how.

* This is due to the variable **c** not being declared within the scope of where it is being called. Also, the value of the variable **b** remains to be 2, given that const variables can neither be updated nor redeclared and let variables are block scoped.

const a=1, b=2; function test(){

if (true){

var a=100; let b=300; const c=300;

}

console.log(a); console.log(b);

//console.log(c);

}

test();