**Lab 02**

Danielle Hooven

JavaScript for Programmers – WEBD146

Jeffrey Lam

February 3, 2021

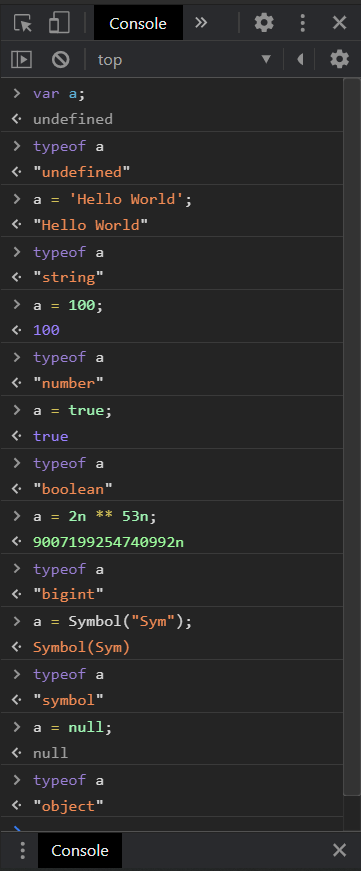
**Lab 02**

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Go to your console, create a variable, and assign each of the different primitives to the

same variable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/



// DYNAMIC VARIABLES

*var* a;

typeof a // returns: "undefined"

a = 'Hello World';

typeof a // returns: string

a = 100;

typeof a // returns: "number"

a = true;

typeof a // returns: "boolean"

a = 2*n* \*\* 53*n*;

typeof a // returns: "bigint:

a = Symbol("Sym");

typeof a // returns: "symbol"

a = null;

typeof a // returns: "object"

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. Next, create a simple program that has a function and a code block, and define variables

in each of the global, function, and block contexts.

3. Then, write console log entries verifying the scope behavior in each case.

Additional Learning Source Credit:

<https://www.codecademy.com/learn/introduction-to-javascript/modules/learn-javascript-scope>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// GLOBAL - a value/function in the global scope can be used anywhere in the entire program.

*var* x = 'global';

// FUNCTION - one visible within the function

*function* scopeLog() {

*let* x = 'local';

    // CODE BLOCK - only visible within a {...} code block.

    if (x === 'local') {

*let* x = 'code block';

        console.log("I'm a " + x + " variable."); // verify code block scope behavior

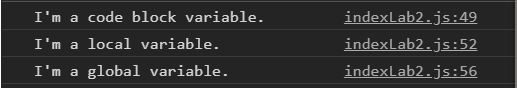
    };

    console.log("I'm a " + x + " variable."); // verify local scope behavior

};

scopeLog(); // call scopeLog function

console.log("I'm a " + x + " variable."); // verify global scope behavior



/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. Observe that:

    a. Concatenation makes everything string.

    b. 1\*”2” converts to number

    c. Operators – Operators in JavaScript are functions which leads to some interesting situations.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// Concatentation makes everything a string.

*var* mystring = "1";

*var* mynumber = 2;

console.log(mystring + mynumber); // returns: "12" - “string”

// 1\*"2" converts to a number.

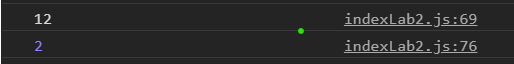
*var* a = 1;

*var* b = "2";

console.log(a \* b); // returns: 2 - number

// Operators are functions - interesting situations

// see 5 - 8



/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. Execute 1<2<3.

6. Now execute 3<2<1.

7. Now execute 1<true<3.

8. Now execute 1<false<3.

Additional Learning Source Credit:

<https://www.codingame.com/playgrounds/6349/javascript-why-does-3-true-4-and-other-quirks>

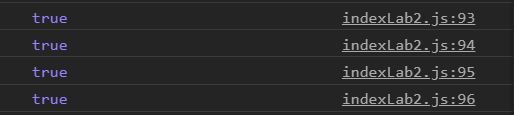
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

console.log(1<2<3); // returns: true

console.log(3<2<1); // returns: true

console.log(1<true<3); // returns: true

console.log(1<false<3); // returns: true



/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9. In a Word document, in 1-2 paragraphs, explain what happened and why.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

For #1, a JavaScript variable value can be changed dynamically. Dynamic typing means variables are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types.

For #2 and #3, a global scope variable can be used anywhere in the entire program, a function scope variable is only visible within the function, and a code block variable is only visible within the curly racket {...} code block. To console.log the values of each of these variables, the log must be written within their respective locations.

For #4, when the plus '+' operator is placed between two operands and the first is a string, it will convert the other number to a string and concatenate them, the result is a string data type. In the second scenario, the order of operands begins with a number multiplied '\*' a string and the result is a number because the first operand is evaluated first.

For #5 - #8, I think I've figured it out. Order of operations is important. JavaScript evaluated the first operand before anything else. In equations, boolean data types are converted into a number - true = 1 and false = 0.

This means:

(1<2<3)

(true<3)

(1<3)

true

(3<2<1)

(false<1)

(0<1)

true

(1<true<3)

(1<1<3)

(false<3)

(0<3)

true

(1<false<3)

(1<0<3)

(false<3)

(0<3)

true