ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING

Late Assignment I

Fundamental of Web Development

Prepared By: - Misganaw Meseret

Jan 2020

Table of contents

Java script:compiled or interpreted? 1

The history of “type of null” 3

Hoisting with let and const 4

Semicolons in JavaScript: To Use or Not to Use? 5

Statements and expressions in JavaScript 6

References 8

Java script:compiled or interpreted?

For me JavaScript is more of a compiled programming language than it is interpreted.

Compiler :- a program that converts instructions into a machine-code or lower-level form so that they can be read and executed by a computer.

Interpreter :- a program that can analyse and execute a program line by line.

Compiled programming language :-A compiled language is a programming language whose implementations are typically compilers (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code, where no pre-runtime translation takes place).

Translated programming language:-An interpreted language is a type of programming language for which there is an interpreter which runs over a virtual machine. The interpreter excuses the code line by line and convert it into low level machine language.

If JavaScript was entirely an interpreted language :

- it should not have thrown errors that are found after interpret able codes
- Hoisting would not be true
- Google’s v8 engine would not exist
- If JavaScript was entirely a compiled language:
- Spider monkey still exists and it interprets JavaScript

So JavaScript can be both compiled and interpreted it basically depends on its implementation in browsers.

The history of “type of null”

All values in JavaScript are either primitive or objects.In this sense, null is primitive and primitive values are immutable which means you cannot add properties to them.The “type of” allows us to distinguish between primitives and objects.

When we type type of null it will return object.which is incorrect as null is a primitive value.which cannot be fixed to not break existing code.

The first version of JavaScript used 32 bits which consisted of a small type tag(1-3) and the actual data of the value.

Object is assigned with “000”. in other case null(JSVAL_NULL) was the machine code NULL pointer or an object type tag plus a reference that is zero.

Steps perform by engine’s code of type of is

1. The engine checks whether the value is undefined(VOID)
2. The engine checks for the object tag which technically is also assigned for null.

That is why it says a type of null is an object.

Hoisting with let and const

Hoisting in JavaScript:- JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

During the compile phase, just microseconds before your code is executed, it is scanned for functions and variable declarations. All these functions and variable declarations are added to the memory So that they can be used even before they are actually declared in the source code.

Variables declared with let and const are hoisted. Where they differ from declarations in the hoisting process is in their initialization.

During the compilation phase, JavaScript variables declared with var and function are hoisted and automatically initialized to undefined.
Or they will only get initialized when assignment is evaluated during run time by a JavaScript engine which means they cannot be accessed before the engine evaluates its value which is called “Temporal Dead zone”,A time span between variable creation and its initialization where they can’t be accessed.

In Contrast, variables declared with let and const are hoisted but not initialized.

Initialization**:-** is the means of assigning an initial value to a variable.

Variables declared with the keyword let are block scoped and not function scoped.

It’s not an error to reference let and const variables in code above their declaration as long as that is not executed before their declaration.

Semicolons in JavaScript: To Use or Not to Use?

The reason why semicolons are sometimes optional in JavaScript is because of automatic semicolon insertion, or **ASI**.

There are 3 main points to be aware of when it comes to ASI.

1. A semicolon will be inserted when it comes across a line terminator or a '}' that is not grammatically correct. So, if parsing a new line of code right after the previous line of code still results in valid JavaScript, ASI will not be triggered.
2. If the program gets to the end of the input and there were no errors, but it's not a complete program, a semicolon will be added to the end. Which basically means a semicolon will be added at the end of the file if it's missing one.

3. There are certain places in the grammar where, if a line break appears, it terminates the statement unconditionally and it will add a semicolon. One example of this is return statements.

I personally advise to use semicolons for reasons like

- compression could cause your valid code to throw an error because those programs may rely on semicolons.
- it can be harder to debug without semicolons

Statements and expressions in JavaScript

JavaScript distinguishes expressions and statements.

An expression produces a value and can be written wherever a value is expected.Any unit of code that can be evaluated to a value is an expression. a statement performs an action. Loops and if statements are examples of statements. Creating a variable or a function,looping through an array of elements, evaluating code based on a specific condition are examples of JavaScript statements.

```
const y = getAnswer();
```

Above is an example of an expression.This call will return a value, i.e. this function call will resolve to a value.

Not everything in the code becomes a value. So, not everything in the code is an expression, although, most things are.seeing expressions helps you understand the process of computation.

You can’t put statements where expression are expected.For example, passing a const statement as function argument will produce an error. Or trying to assign the if statement to a variable.Thi just doesn’t make sense in the language, because only expressions are expected in these cases:

```
console.log(const x);   // error!
let b = if (x > 10) { return 100; };
```

References

intro to Programming: https://en.hexlet.io/courses/intro_to_programming/lessons/expressions/theory_unit

MDB web docs: https://developer.mozilla.org/en-US/docs/web/JavaScript/Reference/Operators/function

2ality-javascript and more: https://2ality.com/2013/10/typeof-null.html#:~:text=In%20JavaScript%2C%20typeof%20null%20is,it%20would%20break%20existing%20code.&text=The%20data%20is%20a%20reference%20to%20an%20object.