

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING

**JavaScript and its unusual behaviors**

**Prepared By:** - Eyob Alemu

February 2020

Table of Contents

[Introduction 2](#_Toc62068390)

[Is JavaScript interpreted in its entirety? 3](#_Toc62068391)

[The history of “typeof null” 5](#_Toc62068392)

[Resources 8](#_Toc62068393)

# Introduction

JavaScript a really powerful language and it can be used in many different areas. But for all its powerfulness and usefulness, some the languages behaviors are a little bit odd. This document tries to see some of those odd behaviors and reason about how they are the way they are.

The behaviors of JavaScript seen in this document are:

* Whether or not JavaScript is interpreted or compiled language
* Why is the type of null an object when it should be primitive data type
* Hoisting with var vs let and const
* The usage of semicolons
* Expressions and statements in JavaScript

The reasoning in this document is based on different sources mentioned in the references page. But when there are personal opinions of the writer, it will be mentioned right along the assumption that it is the writer’s opinion.

The document uses “Courier New” font for code snippets.

# Is JavaScript interpreted in its entirety?

Every program is a set of instructions, this instructions can be written in any high level programming language. But these high level programming languages are mainly designed to be human readable which makes them hard for machines to understand as they are. Therefore, in order for machines to understand them, these programs need to be converted to machine readable format also known as machine code. To do this there are two mainly used methods interpretation and compilation.

Compilation is the process of converting an entire source code to the target machine code and executing. Compiled languages will have a build step where the compiler takes the source code and converts it to the respective machine code. Since the entire source code is going to be converted to machine code once, compiled languages are fast at execution time. But modifying the source code results in recompilation.

In interpretation the source code is read and executed by a program call the interpreter. This interpreter reads the source code line by line and executes each command one at a time. Due to this line by line execution, interpreted languages are relatively slower. But when editing the source code there won’t be any need for recompilation or rebuild steps since the interpreter is reading line by line.

According to the MDN documentation, “JavaScript is a lightweight, interpreted, or just-in-time compiled language.” If interpretation and compilation are as described above, then what is just-in-time compilation?

Just-in-time compilation (JIT) is a method used to improve performance of interpreted languages. While interpreted program is being executed the JIT compiler determines the most frequently used code and compiles it to machine code. This machine code is an optimized code for the target CPU architecture then every time that same block of code is needed, this already compiled machine code will be used. This way JIT helps to avoid the inefficiency caused by recompiling a code block over and over again.

A JavaScript engine is a program that executes a JavaScript code and it can be implemented as a plain old interpreter or as JIT compiler.

Some of the engines that implement plain old interpretation are,

* Futhark : ECMAScript engine for Opera browser versions 9.5 – 10.10
* Jscript : engine used in Internet Explorer for version up to IE9
* Rhino : one of several engines from Mozilla using the Java platform
* V4 (QJSEngine) : Qt’s newer ECMAScript engine

Some of the engines that use JIT are,

* Carakan: engine used by Opera browser in versions 10.5 – 15
* V8: an open source engine used in Google Chrome, Node.js, Deno and V8.NET
* Chakra (Jscript9): and engine used in IE after IE9 also used in Microsoft Edge
* SpiderMonkey: the engine used in Mozilla Firefox and other Gecko applications

The above mentioned engines are just an example and there are lots of other engines implemented either as an interpreter or JIT. What all this shows is that classifying the JavaScript language is more or less dependent on the implementation. It is the writer’s opinion that JavaScript can be both compiled and interpreted language depending on the implementation used by the engine. Therefore, it is hard to say that JavaScript is just an interpreted language specially with the introduction of JIT.