

**ADDIS ABABA UNIVERSITY**

**ADDIS INSTITUTE OF TECHNOLOGY**

**CENTER OF INFORMATION TECHNOLOGY AND SCIENTIFIC COMPUTING**

**ASSIGNMENT 1 (New One)**

**Prepared By - Binyam Lemma**

**ID - ATR/1341/10**

**Department - IT**

**To - Mr. Fistum Alemu**

**January 24, 2021**

Contents

[Introduction 3](#_Toc62401276)

[What is JavaScript? 3](#_Toc62401277)

[Applications of JavaScript 3](#_Toc62401278)

[Why use JavaScript over other programming languages? 4](#_Toc62401279)

[Compiled language VS Interpreted language 5](#_Toc62401280)

[1. Is JavaScript Interpreted Language in its entirety? 5](#_Toc62401281)

[2. The history of “type of null”? 6](#_Toc62401282)

[3. Explain in detail why hoisting is different with let and const? 6](#_Toc62401283)

[4. Semicolons in JavaScript: To Use or Not to Use? 7](#_Toc62401284)

[What is ASI? 8](#_Toc62401285)

[The rules of JavaScript Automatic Semicolon Insertion 8](#_Toc62401286)

[Explicit Semicolons VS ASI 8](#_Toc62401287)

[5. Expression vs Statement in JavaScript? 9](#_Toc62401288)

[Reference 11](#_Toc62401289)

# 

# **Introduction**

## **What is JavaScript?**

JavaScript is the world’s most popular programming language.

JavaScript is a text-based lightweight, interpreted, or just-in-time (JIT), compiled with the first-class function programming language used on both the client-side and server-side that allows a user to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages. JavaScript gives web pages interactive elements that engage a user. Common examples of JavaScript that people might use every day include the search box on Amazon, or refreshing a Twitter feed.

## **Applications of JavaScript**

JavaScript is usually used for web-based applications and web browsers. However, JavaScript can also be applied beyond the Web in software, servers, and embedded hardware controls. Here are some basic things JavaScript is used for:

1. **Adding interactive behavior to web pages**

JavaScript allows users to interact with web pages. There are almost no limits to the things that can be done using JavaScript on a web page – these are just a few examples:

* Show or hide more information with the click of a button
* Change the color of a button when the mouse hovers over it
* Slide through a carousel of images on the homepage
* Zooming in or zooming out on an image
* Displaying a timer or count-down on a website
* Playing audio and video on a web page
* Displaying animations
* Using a drop-down hamburger menu

1. **Creating web and mobile apps**

Developers can use various JavaScript frameworks for developing and building web and mobile apps. JavaScript frameworks are collections of JavaScript code libraries that provide developers with pre-written code to use for routine programming features and tasks—literally a framework to build websites or web applications around.

Popular JavaScript front-end frameworks include React, React Native, Angular, and Vue.js. Many companies use Node.js, a JavaScript runtime environment built on Google Chrome’s JavaScript V8 engine. A few famous examples include PayPal, LinkedIn, Netflix, and Uber!

1. **Building web servers and developing server applications**

Beyond websites and apps, developers can also use JavaScript to build simple web servers and develop the back-end infrastructure using Node.js.

1. **Game development**

Of course, you can also use JavaScript to create browser games. These are a great way for beginning developers to practice their JavaScript skills.

## **Why use JavaScript over other programming languages?**

Aside from the unlimited possibilities, there are many reasons for web developers to use JavaScript over other programming languages:

* JavaScript is the only programming language native to the web browser
* JavaScript is the most popular language
* There’s a low threshold to get started
* It’s a fun language to learn

## **Compiled language VS Interpreted language**

Both Interpreter and Compiler are special programs that convert a program into machine language.

Compiled languages are the ones who comply with the human-understandable language (programming languages) to the machine-readable language before they are executed. However, interpreted languages are also human-readable languages (programming languages) and need a translation down to machine languages to get executed, but this translation is done at runtime. Therefore, there should be an interpreter in the environment, before running an interpreted language, compiled language applications can run directly once it compiled.

An interpreter runs and interprets the code line-by-line (making it slower than a compiler), checks for errors along with it, and stops when there is one. However, a compiler does not run the code. It directly compiles the whole code and is faster in checking for errors as compared to an interpreter.

# **Is JavaScript Interpreted Language in its entirety?**

Traditionally, JavaScript is an interpreted language, but this is not necessarily true at all times. For instance, JavaScript runs the V8 engine on Chrome, which compiles its native code internally. This compilation helps realize results on time. The same holds for Trace Monkey and Rhino.

The truth is that JavaScript has undergone significant evolution. This evolution has prompted the development of JIT compilers, which help optimize execution.

This is a common misconception now when someone says that JavaScript is an interpreted language then yes there is some truth to it but it depends on the implementation, you can implement a JavaScript engine that only compiles

In conclusion, JavaScript can be interpreted and/or compiled it depends on the implementation.

# **The history of “type of null”?**

From the MDN page about the behavior of the type of operator: null. // this stands since the beginning of JavaScript type of null === ‘object’. In JavaScript, “type of null” is 'object', which incorrectly suggests that null is an object. This is a bug and one that unfortunately cannot be fixed because it would break existing code.

Consequently, null had 0 as its type tag, and since 0 is for objects, then “type of” null fraudulently gives us “object”. For an in-depth explanation and In JavaScript, “type of” null is 'object', which incorrectly suggests that null is an object.

**Why it is not fixed?**

This bug is not fixed unfortunately because it would break existing code.

# **Explain in detail why hoisting is different with let and const?**

**What is Hoisting?**

Hoisting in terms of JavaScript means that a variable will be created in memory during the compile phase, and thus it can be used before it is declared.

**Difference**

Variables defined with let and const are hoisted to the top of the block, but not initialized.

This means the block of code is aware of the variable, but it cannot be used until it has been declared.

Variables declared with let and const are only initialized when their assignment (lexical binding) is evaluated during runtime by the JavaScript engine.

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

Using a let variable before it is declared will result in a ReferenceError.

Example - carName = "Volvo";  
 const carName;

This code will not run. The result will say” ReferenceError: Cannot access 'carName' before initialization.

Using a const variable before it is declared, is a syntax error, so the code will simply not run.

Example - carName = "Volvo";  
 const carName;

**Conclusion**

 let can be declared without being initialized but const must be initialized during declaration.

# **Semicolons in JavaScript: To Use or Not to Use?**

The reason semicolons are sometimes optional in JavaScript is automatic semicolon insertion or ASI.

## **What is ASI?**

ASI (Automatic Semicolon Insertion) is a feature that will automatically add semicolon during the parsing of the source code.

## **The rules of JavaScript Automatic Semicolon Insertion**

1. When the error token of a program from the left to right, a token that does not match the grammar rule has a semicolon inserted before it if either of the two following conditions is satisfied.
2. If the program is parsed until the end of the input and it is not yet a complete program (i.e. the wee no outright errors that would have caused an exception to throw before reaching the end of the tokens) a semicolon is appended.
3. If a line terminator should be encountered with a “restricted production”, ASI will add a semicolon before the line terminator is encountered.

## **Explicit Semicolons VS ASI**

* If a developer is going to write JS without optional semicolons, understanding what ASI does
* Minification, compression, etc. on otherwise valid JavaScript code could cause unforeseen errors if your code doesn't use semicolons
* Performance and file size is not significantly affected by semicolons or lack thereof

**Conclusion**

In conclusion, it is not mandatory to use a semicolon because of ASI but it’s better because to mind the structure and to get the developer into the habit of understanding and delimiting statements in your head, to have admittedly marginally a better understanding of how the code might parse.

# **Expression vs Statement in JavaScript?**

**Expressions**

Any unit of code that can be evaluated to a value is an expression. Since expressions produce values, they can appear anywhere in a program where JavaScript expects a value such as the arguments of a function invocation.

As per the MDN documentation, JavaScript has the following expression categories.

**Arithmetic Expression**

Arithmetic expressions evaluate to a numeric value.

**String Expressions**

String expressions are expressions that evaluate to a string.

**Logical Expressions**

Expressions that evaluate the Boolean value true or false are considered logical expressions. This set of expressions often involve the usage of logical operators && (AND), || (OR) and! (NOT).

**Primary Expressions**

Primary expressions refer to stand-alone expressions such as literal values, certain keywords, and variable values.

Also known as, values, left-hand-side expressions are those that can appear on the left side of an assignment expression.

**Assignment Expressions**

When expressions use the = operator to assign a value to a variable, it is called an assignment expression.

**Statements**

A statement is an instruction to perform a specific action. Such actions include creating a variable or a function, looping through an array of elements, evaluating code based on a specific condition, etc. JavaScript programs are a sequence of statements.

Statements in JavaScript can be classified into the following categories

**Declaration Statements**

Such type of statements creates variables and functions by using the var and function statements respectively.

**Expression Statements**

Wherever JavaScript expects a statement, you can also write an expression. Such statements are referred to as expression statements. However, the reverse does not hold. You cannot use a statement in the place of an expression.

**Conditional Statements**

Conditional statements execute statements based on the value of an expression. Examples of conditional statements include the if…else and switch statements.

**Loops and Jumps**

Looping statements includes the following statements: while, do/while, for, and for/in. Jump statements are used to make the JavaScript interpreter jump to a specific location within the program. Examples of jump statements include a break, continue, return and throw.

# **Reference**

Interpreted vs Compiled MDN - <https://medium.com/@almog4130/javascript-is-it-compiled-or-interpreted-9779278468fc>

Interpreted vs Compiled Yes for Dev.com - <https://yesfordev.com/is-javascript-compiled-or-interpreted/>

Tyenull - <https://neutrondev.com/mystery-history-typeof-null-javascript/>

Typenull - <https://bitsofco.de/javascript-typeof/>

Typenull - <https://www.xspdf.com/resolution/50420494.html>

Hoisting W3School - <https://www.w3schools.com/js/js_hoisting.asp#:~:text=Hoisting%20is%20JavaScript's%20default%20behavior%20of%20moving%20all%20declarations%20to,script%20or%20the%20current%20function>)

Hoisting JavaScript in Plain English - <https://www.medium.com/javascript-in-plain-english/how-hoisting-works-with-let-and-const-in-javascript-725616df7085s>

ASI Full Stack Academy **-** <https://www.youtube.com/watch?v=B4Skfqr7Dbs&ab_channel=FullstackAcademy>

ASI Flavio copes - <https://flaviocopes.com/javascript-automatic-semicolon-insertion/>

Semicolons Stack overflow - <https://stackoverflow.com/questions/537632/should-i-use-semicolons-in-javascript#_=_>

Semicolons DEV - <https://dev.to/adriennemiller/semicolons-in-javascript-to-use-or-not-to-use-2nli>

ASI Codeburst.io - <https://codeburst.io/ecmascript-automatic-semicolon-insertion-50f09091e377>

Statement vs Expression medium.com - <https://medium.com/launch-school/javascript-expressions-and-statements-4d32ac9c0e74>