|  |  |
| --- | --- |
| A picture of a winding road and trees  node.js | Misra, Shikhin |

Contents

[Introduction 2](#_Toc144240774)

[ECMAScript 3](#_Toc144240775)

[Chrome's V8 Engine 5](#_Toc144240776)

[JavaScript Runtime 8](#_Toc144240777)

[What is Node.js 9](#_Toc144240778)

[Browser vs Node.js 10](#_Toc144240779)

[Hello World! (outside the browser) 11](#_Toc144240780)

[Modules 12](#_Toc144240781)

[Local Modules 12](#_Toc144240782)

[Module Exports 14](#_Toc144240783)

[Module Scope 16](#_Toc144240784)

[Module Wrapper 17](#_Toc144240785)

[Module Caching 18](#_Toc144240786)

[Import Export Patterns 21](#_Toc144240787)

[Module.Exports vs Exports 22](#_Toc144240788)

[ES Modules 22](#_Toc144240789)

[Importing JSON and Watch Mode 22](#_Toc144240790)

[Built-in Modules 22](#_Toc144240791)

[Path Module 22](#_Toc144240792)

[Callback Pattern 22](#_Toc144240793)

[Events Module 22](#_Toc144240794)

[Extending from EventEmitter 22](#_Toc144240795)

[Character Sets and Encoding 23](#_Toc144240796)

[Streams and Buffers 23](#_Toc144240797)

[Asynchronous JavaScript 23](#_Toc144240798)

[fs Module 23](#_Toc144240799)

[fs Promise Module 23](#_Toc144240800)

[Streams 23](#_Toc144240801)

[Pipes 23](#_Toc144240802)

[HTTP Module 23](#_Toc144240803)

[Creating a Node Server 23](#_Toc144240804)

[JSON Response 24](#_Toc144240805)

[HTML Response 24](#_Toc144240806)

[HTML Template 24](#_Toc144240807)

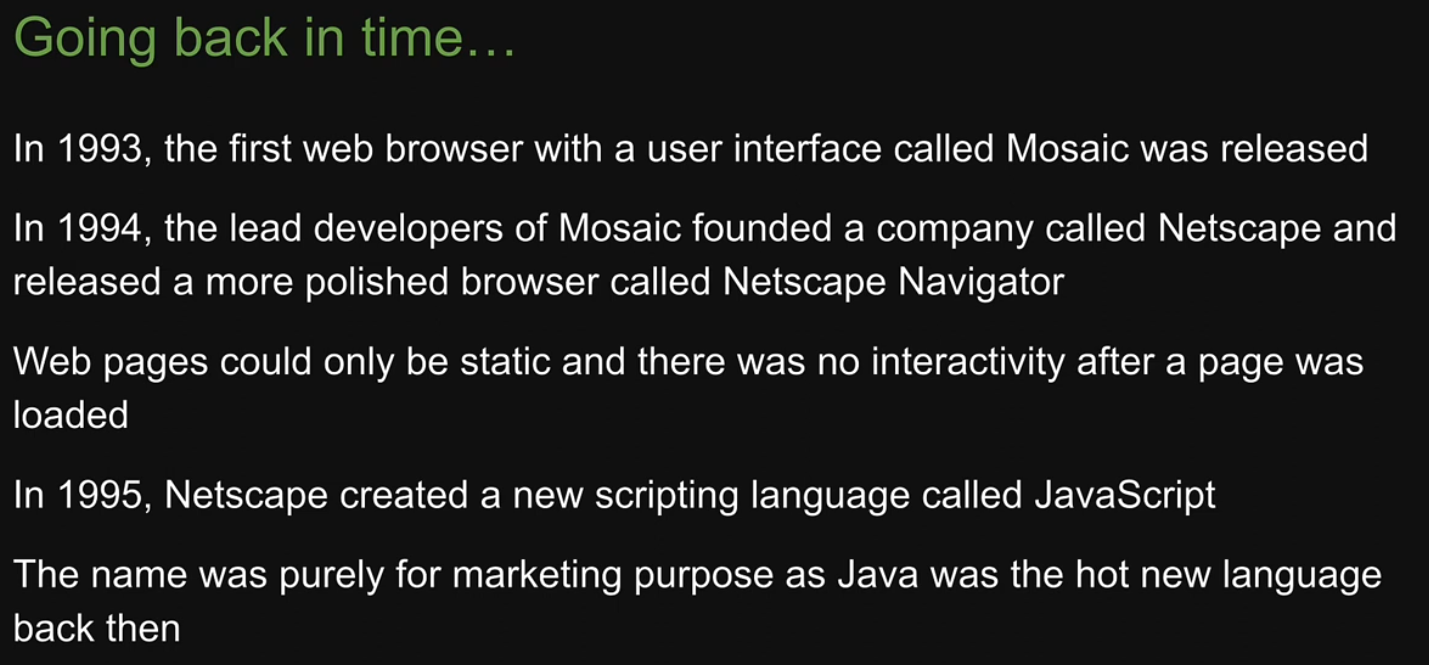
[HTTP Routing 24](#_Toc144240808)

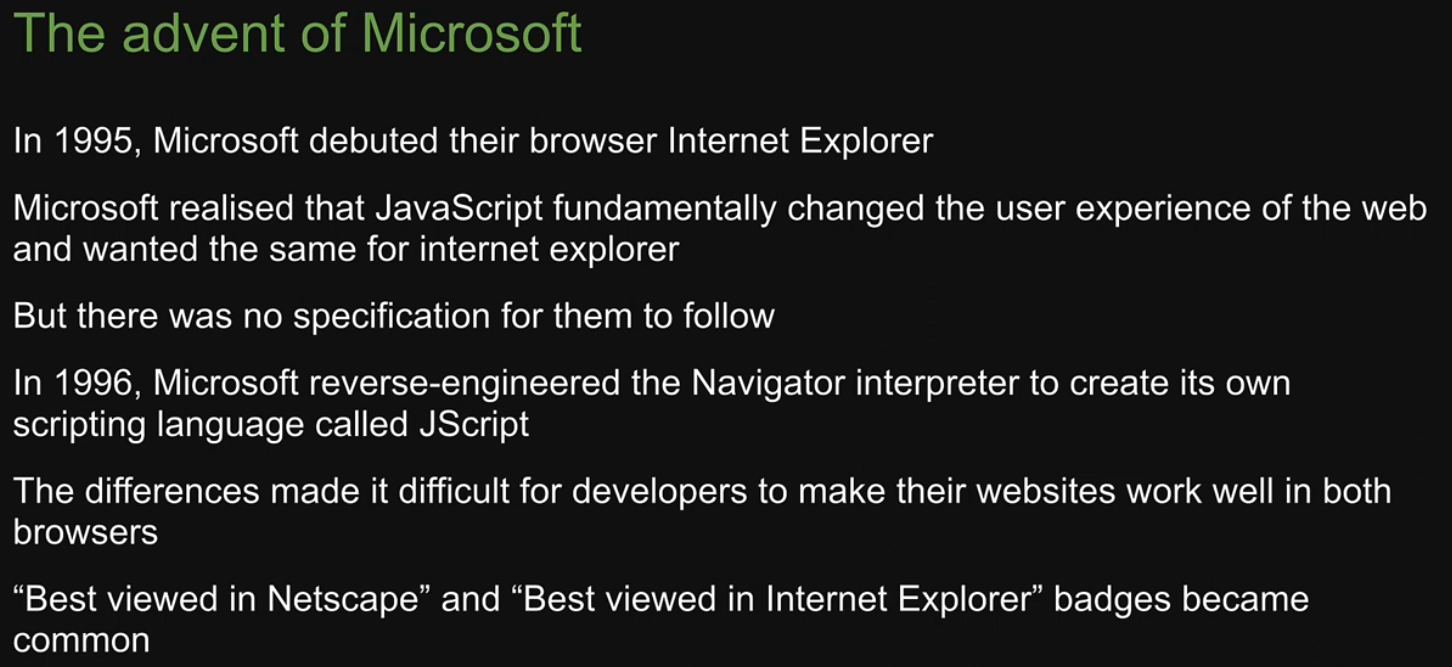
[Web Framework 24](#_Toc144240809)

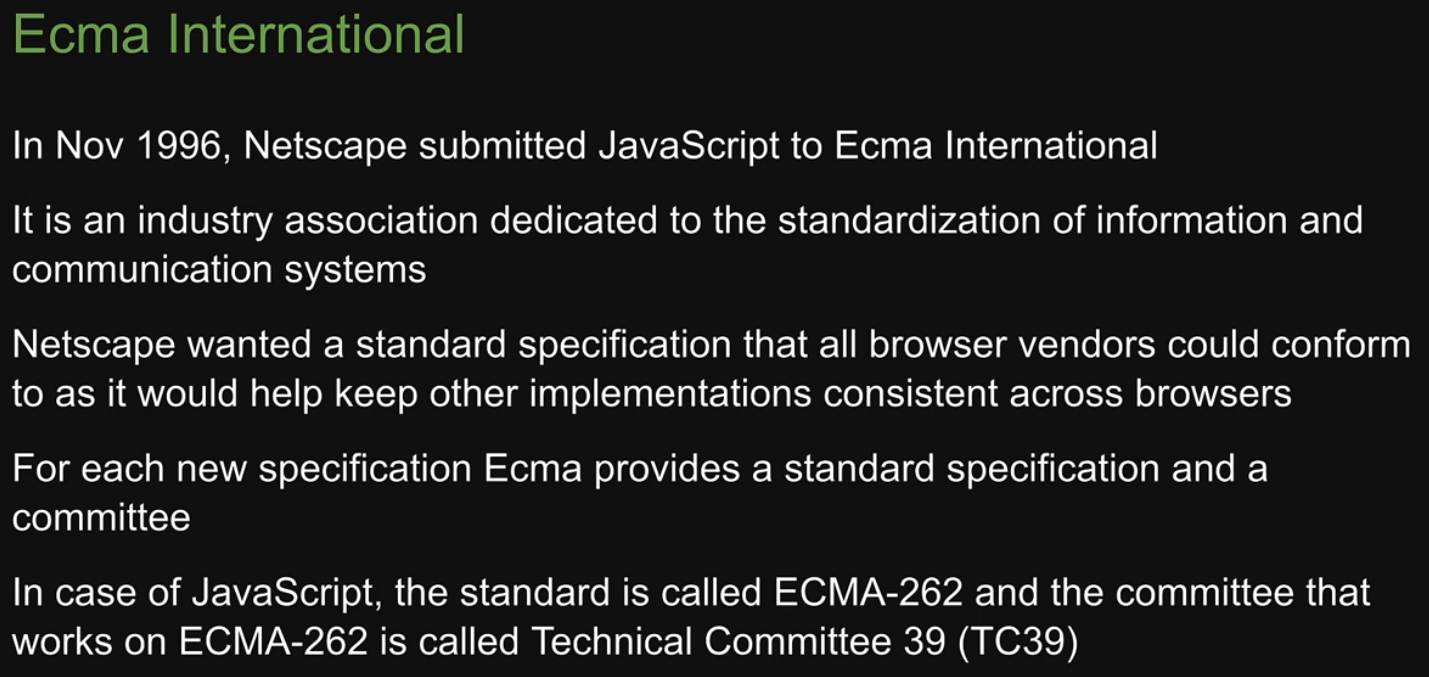
# Introduction

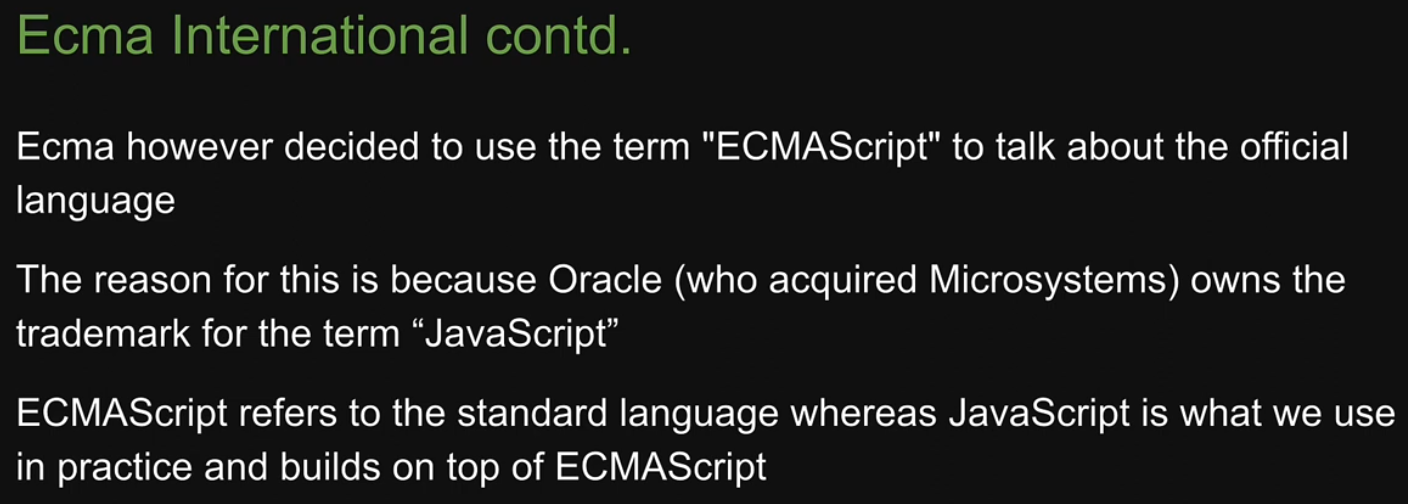


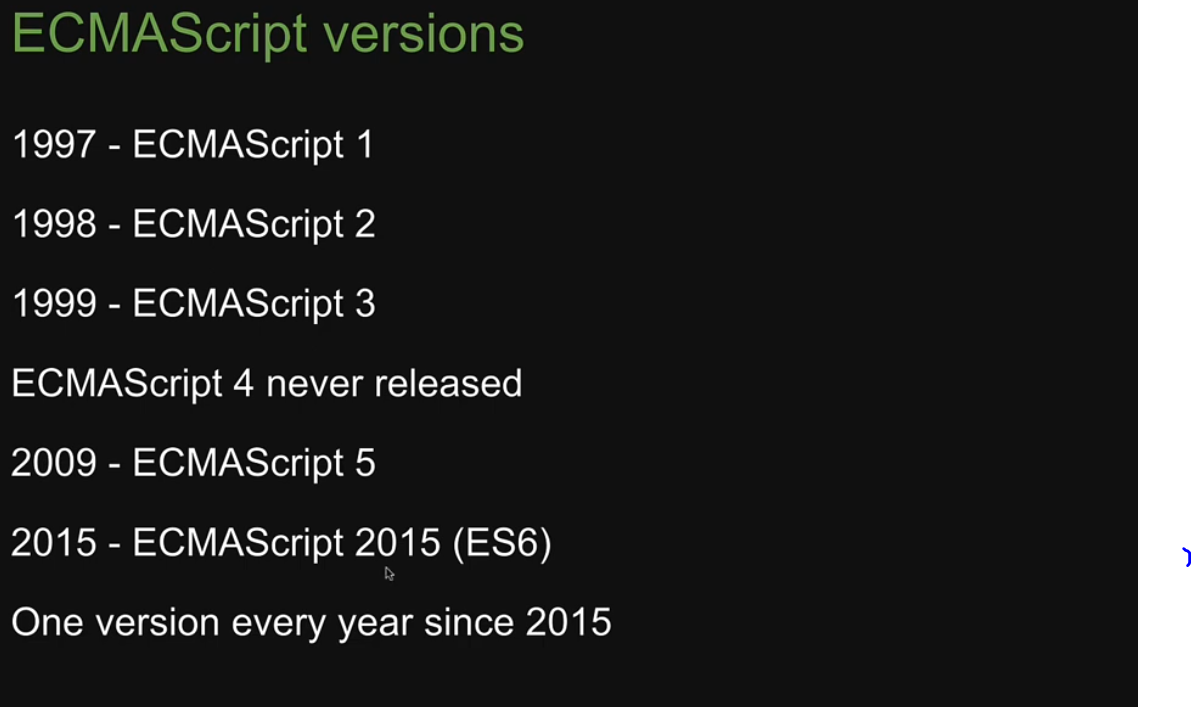
# ECMAScript

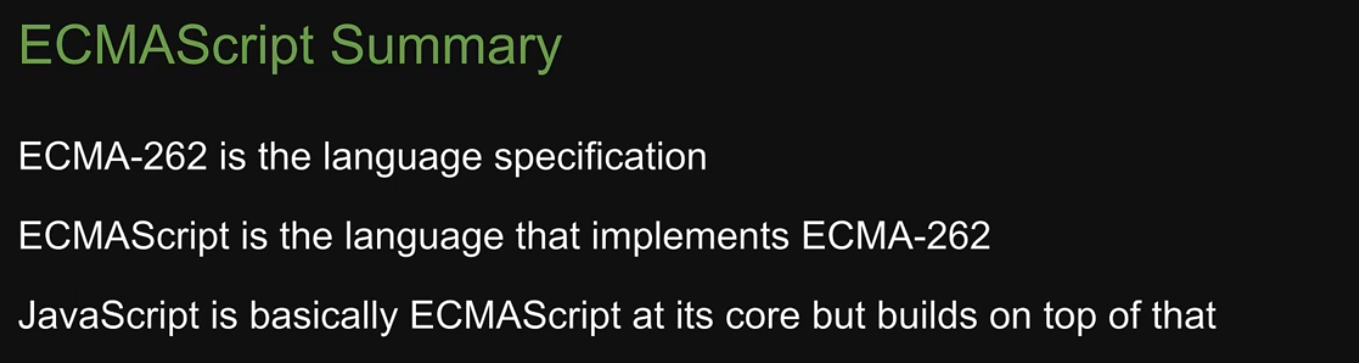




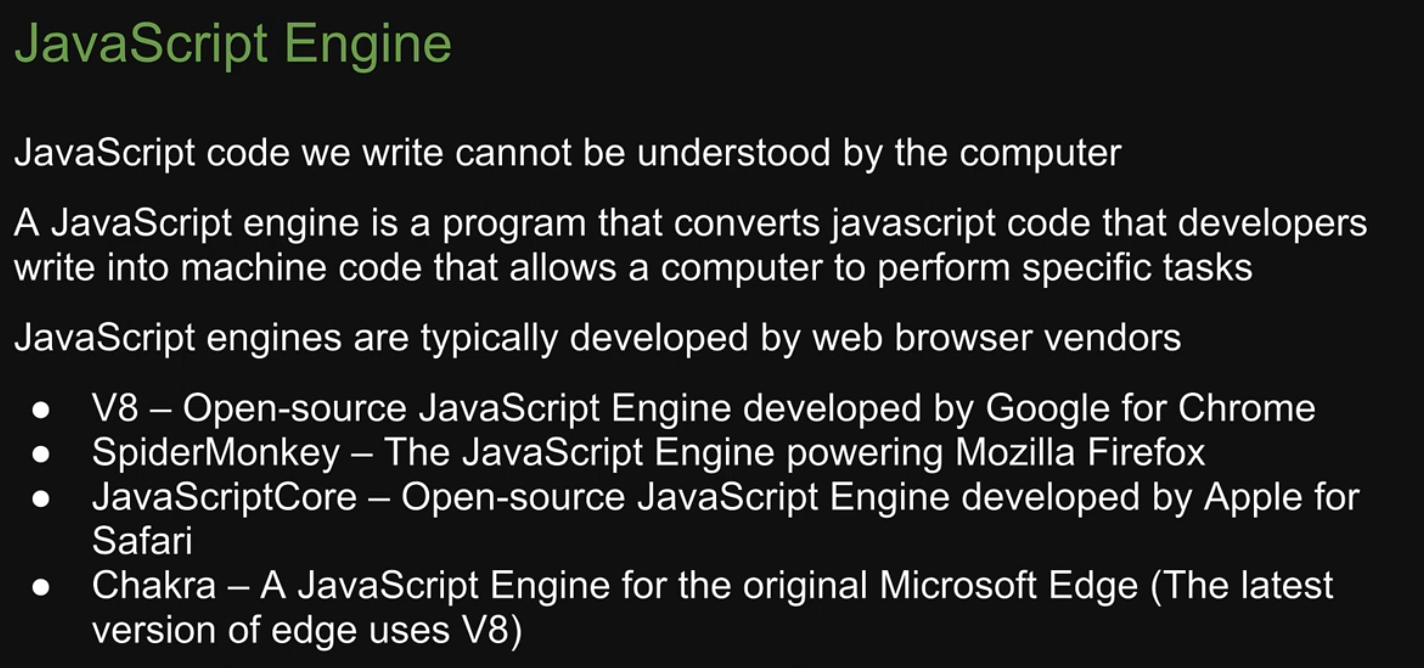


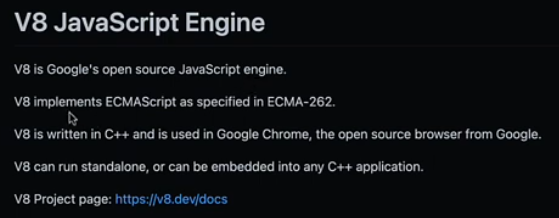


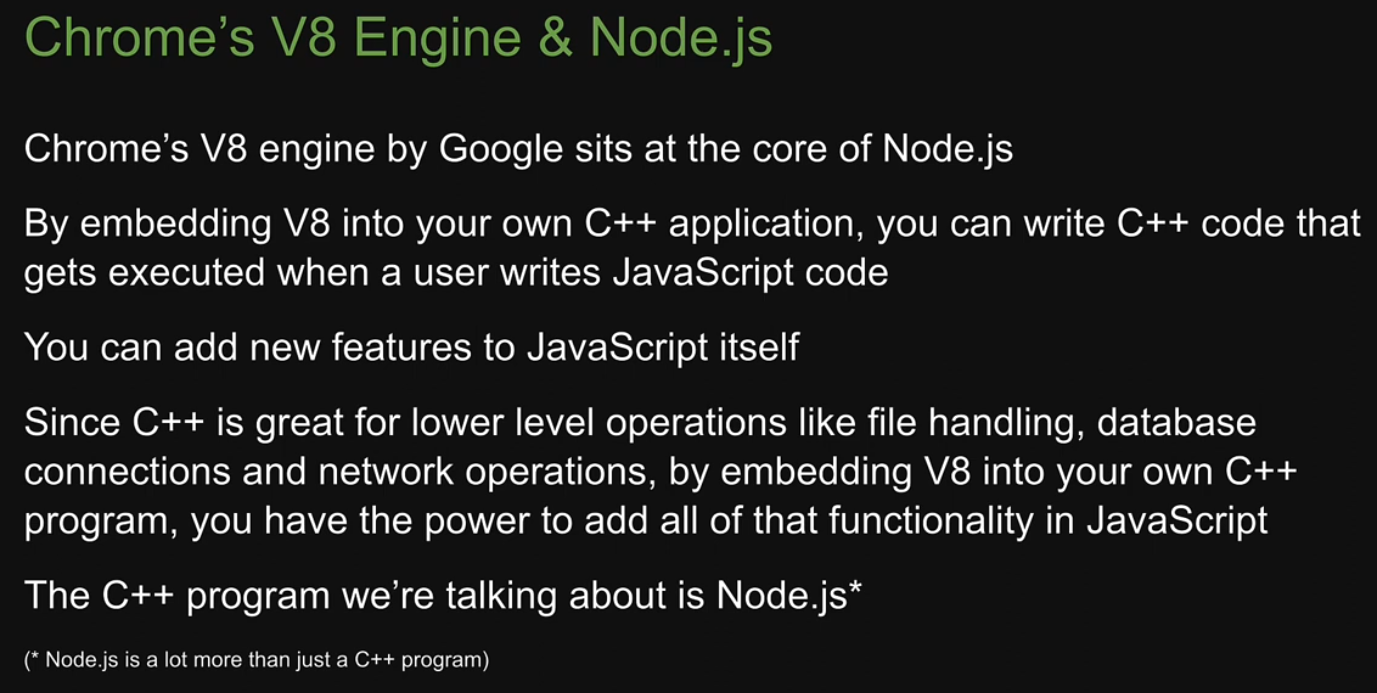


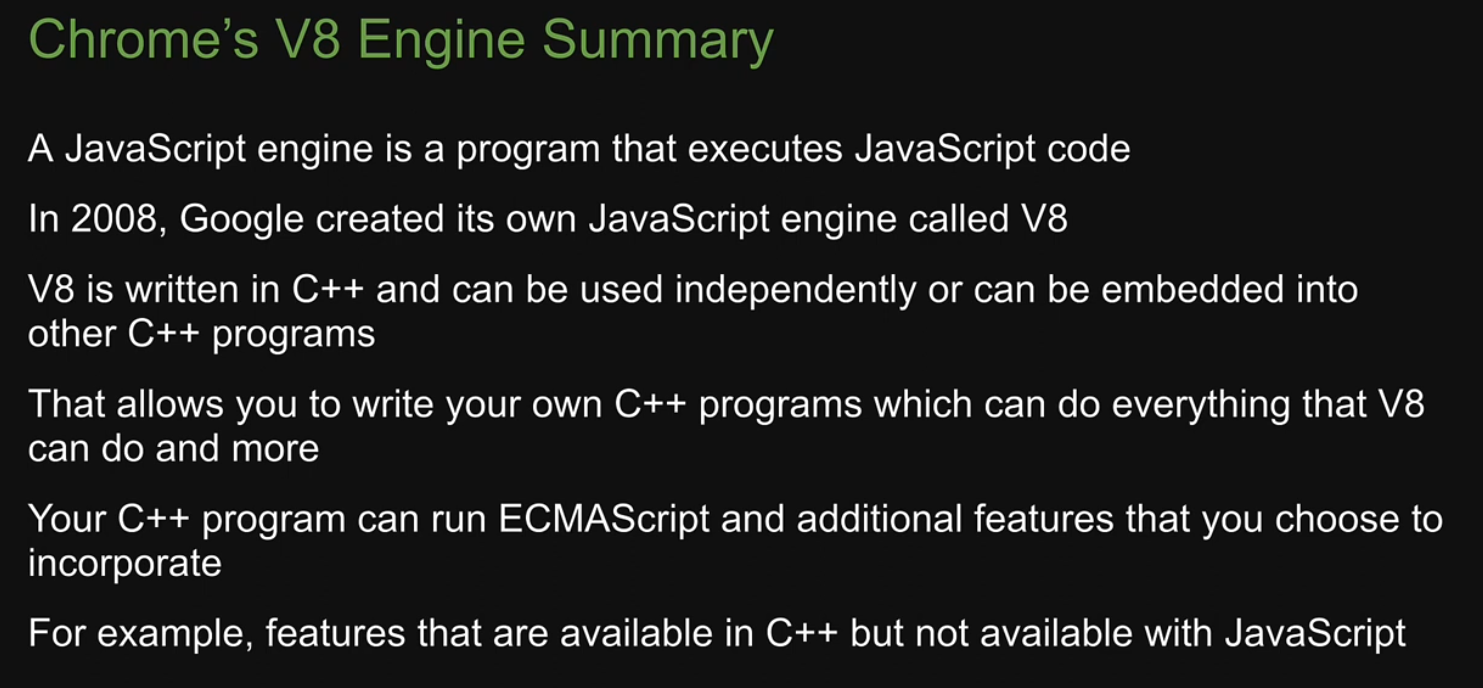


# Chrome's V8 Engine

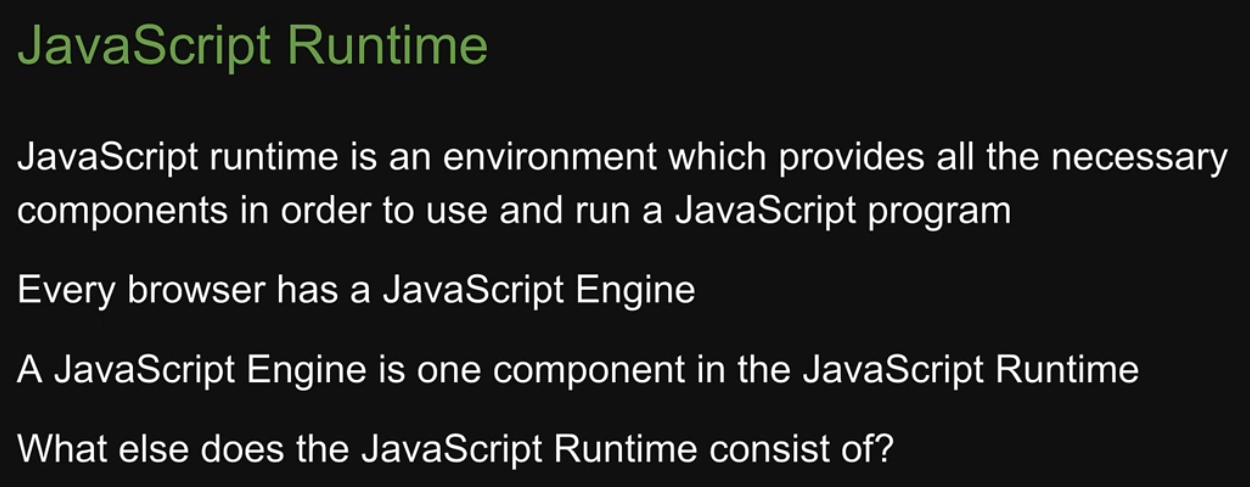


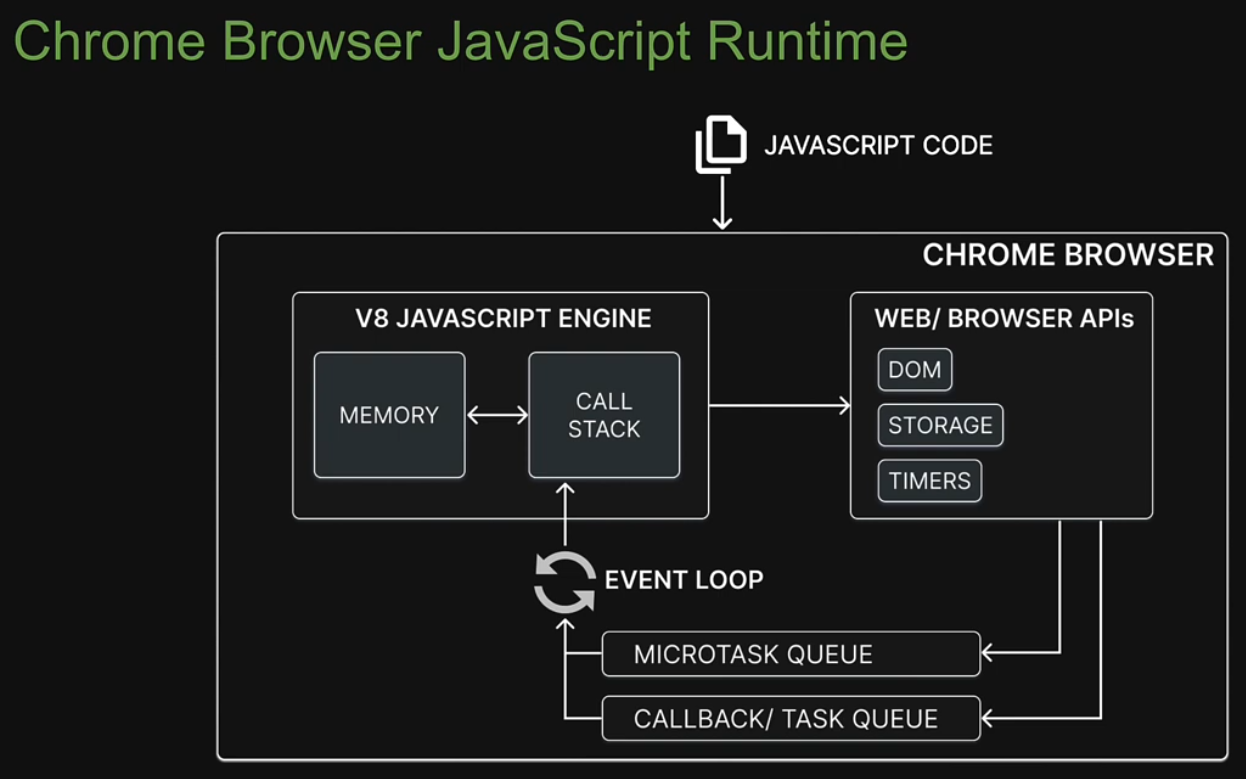






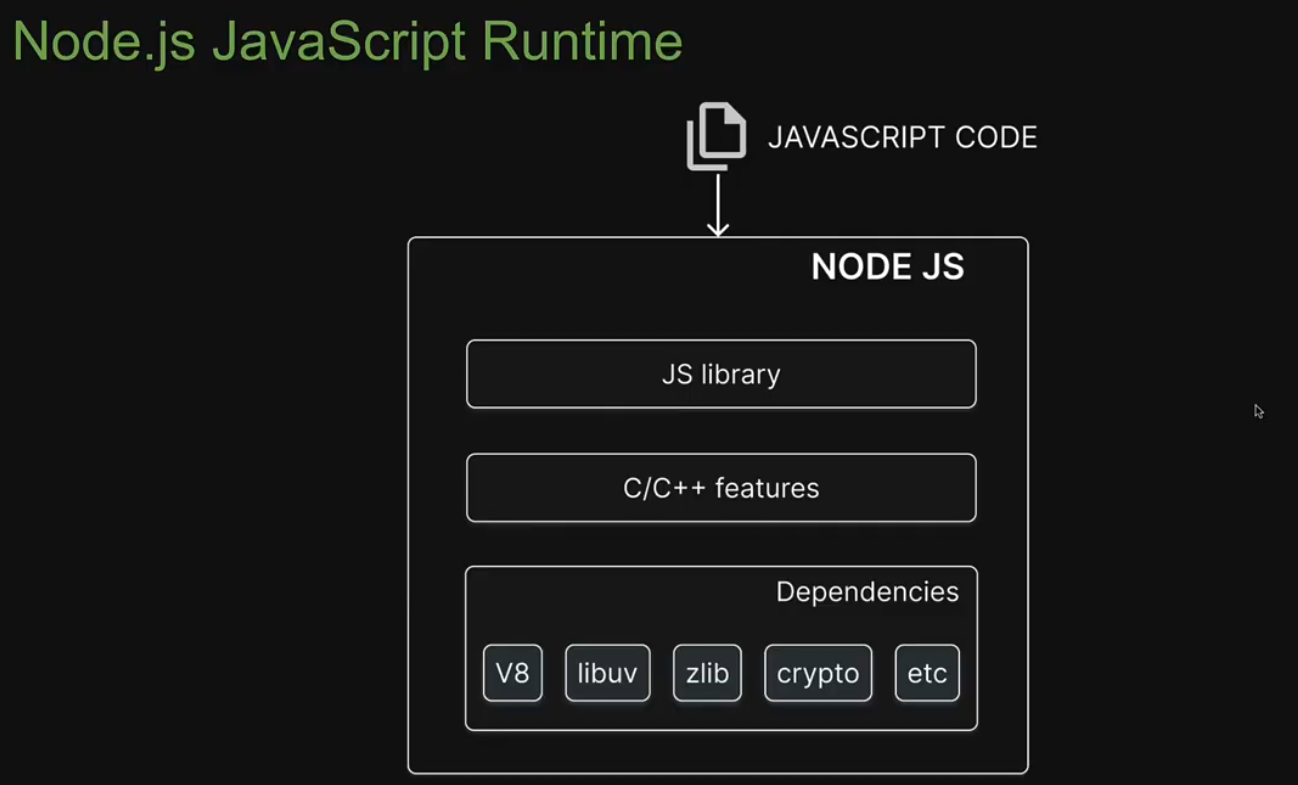
# JavaScript Runtime

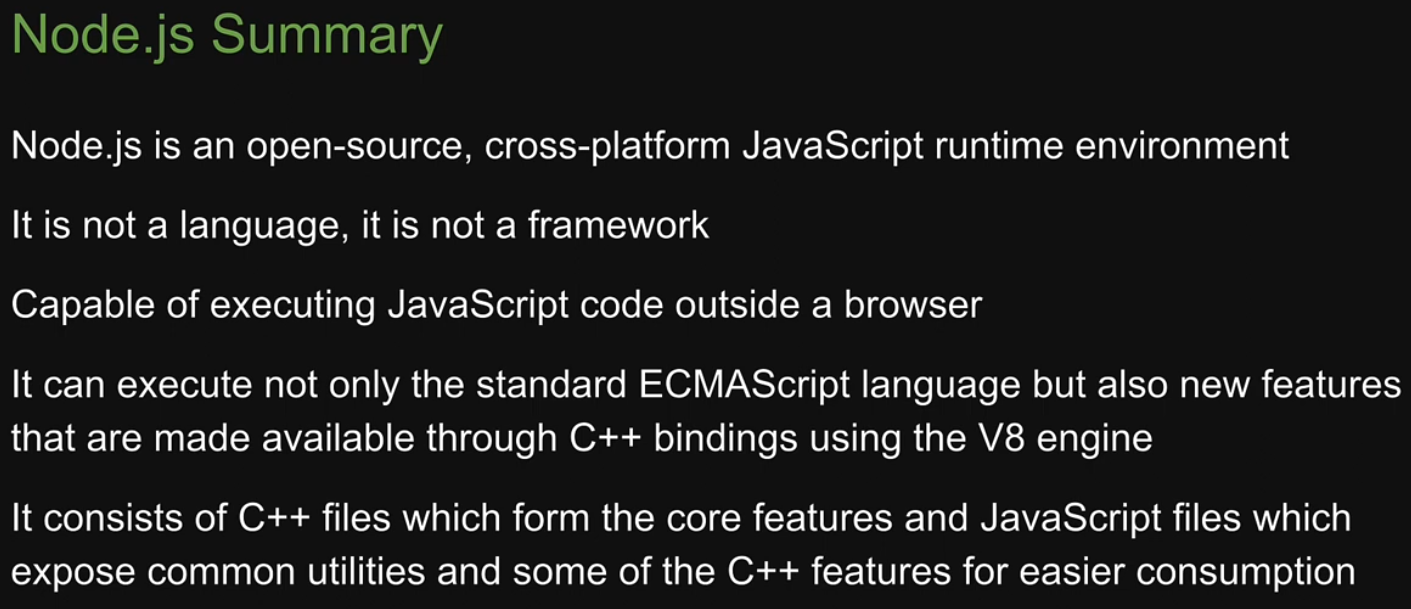




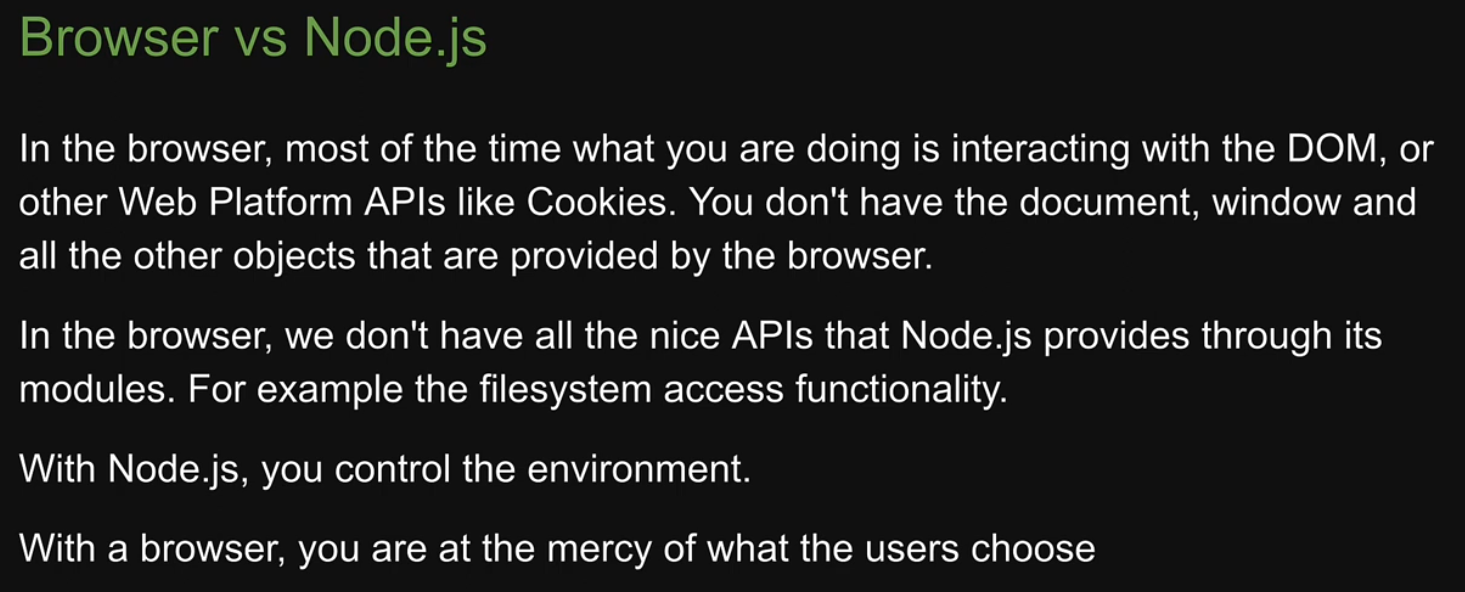
# What is Node.js





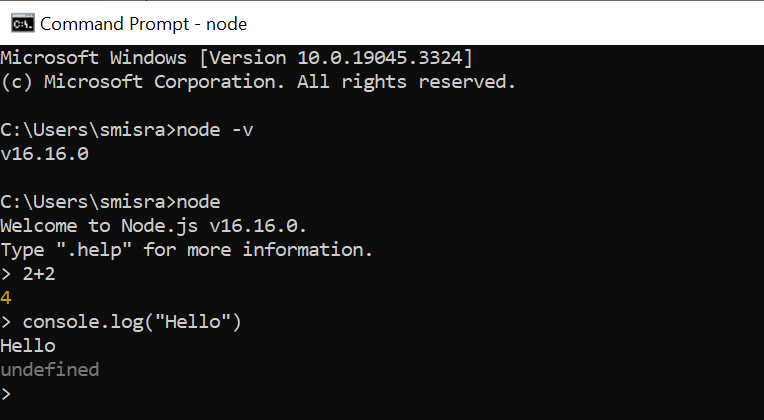


# Browser vs Node.js

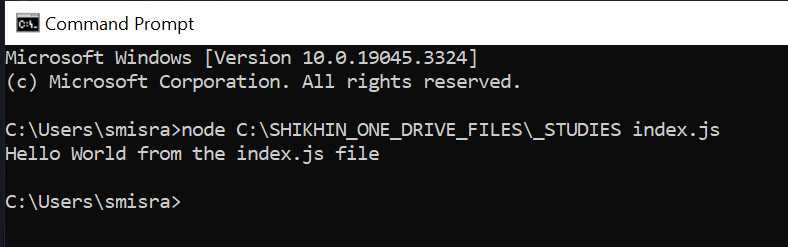


# Hello World! (outside the browser)

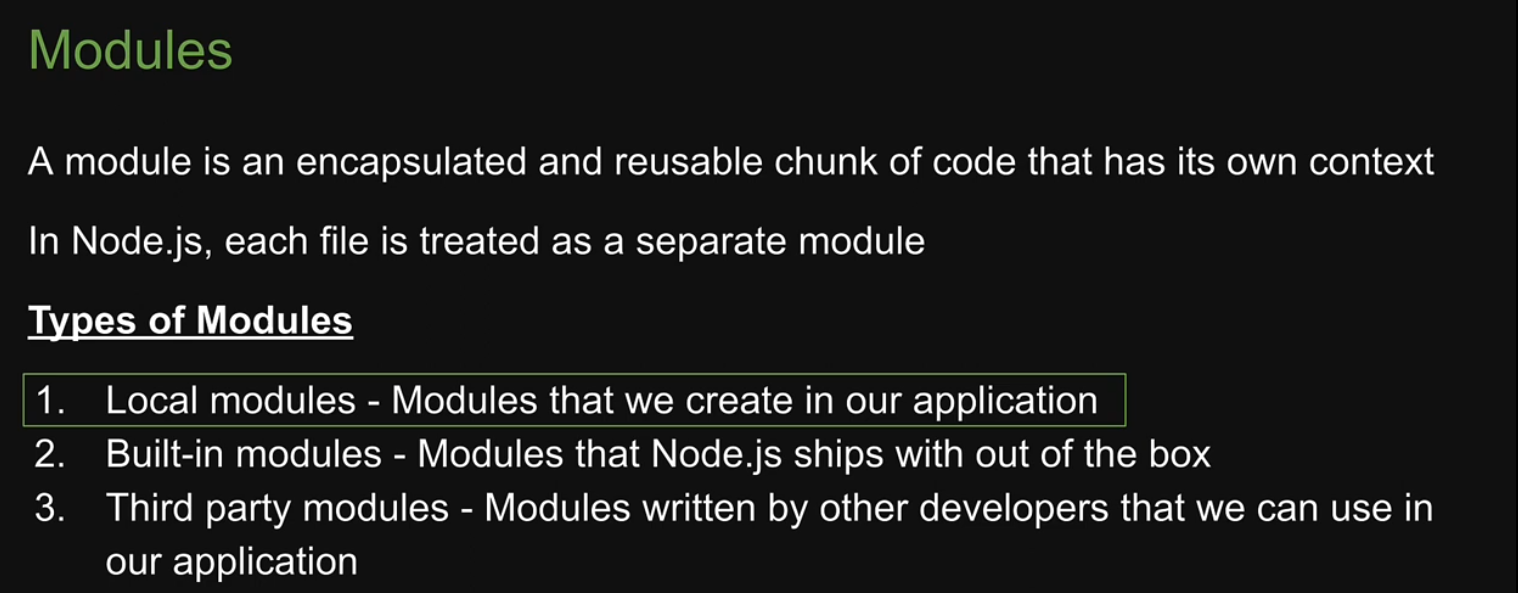
Below we are executing javascript in the form of REPL. Note that we are executing JavaScript in node from CMD rather than from browser.



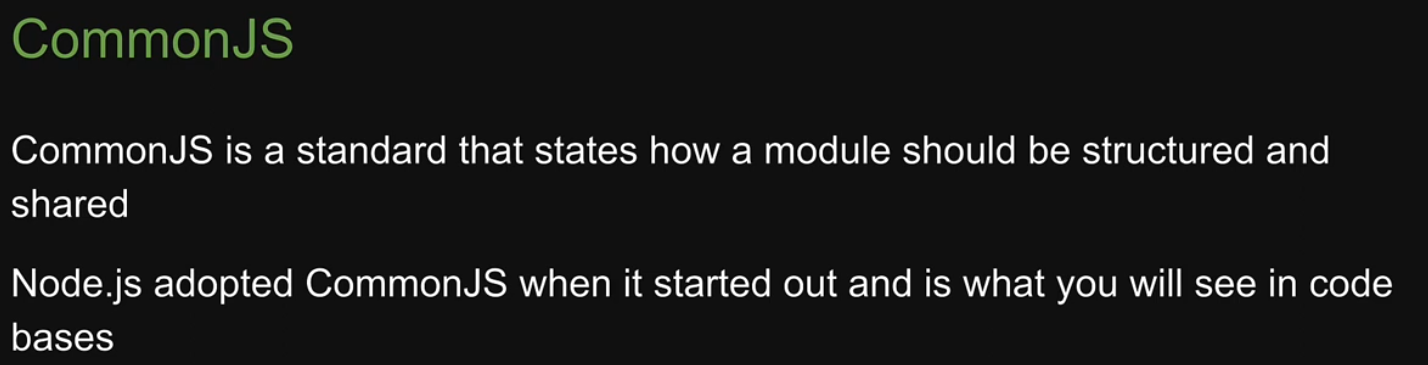
Below we are running JavaScript from a file using node. Note that we are again running JavaScript without browser.



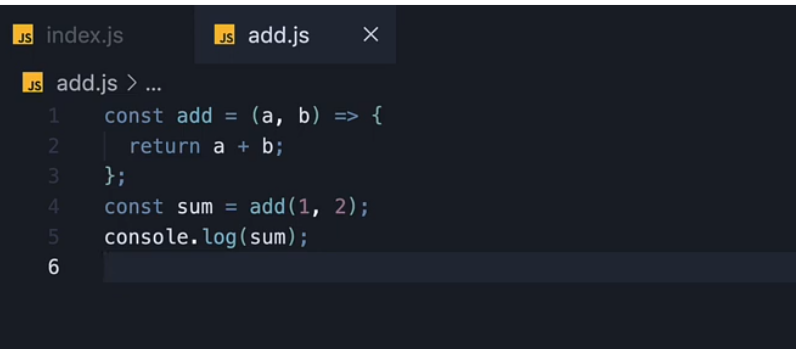
# Modules



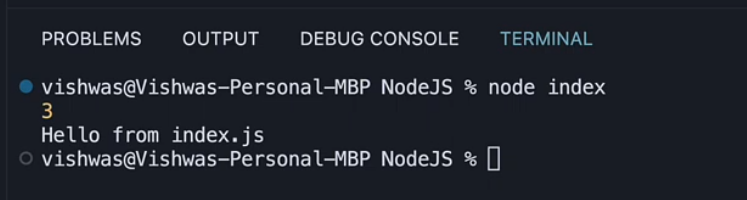
# Local Modules



We use “**require**” module to club modules together.







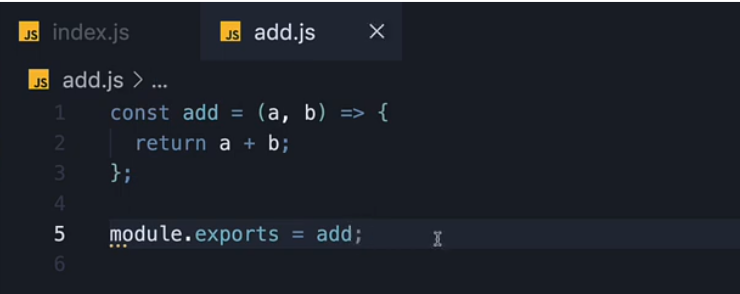
We can omit the .js extension in the “require” function. Node will infer itself the extension.



# Module Exports

Module exports are the actual way of exposing the functionalities coded in a particular module for external modules. In the last section we used the require function to include the complete file rather than using the add functionality in the file.

Here instead of adding the numbers we will expose the add function by using module.exports

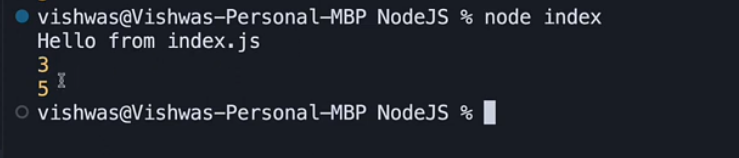


What is special about this object? The value of this object is what the “require” function returns in index.js.

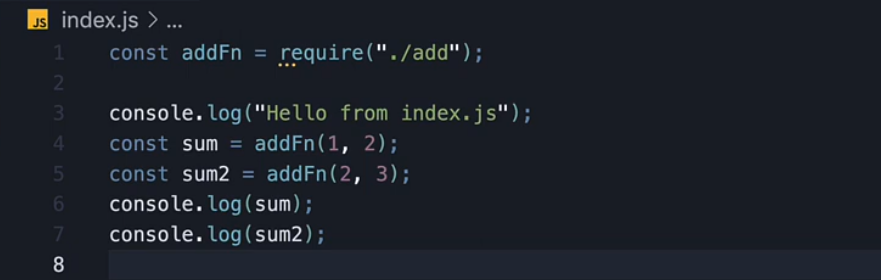
We can store this value in a variable in index.js.



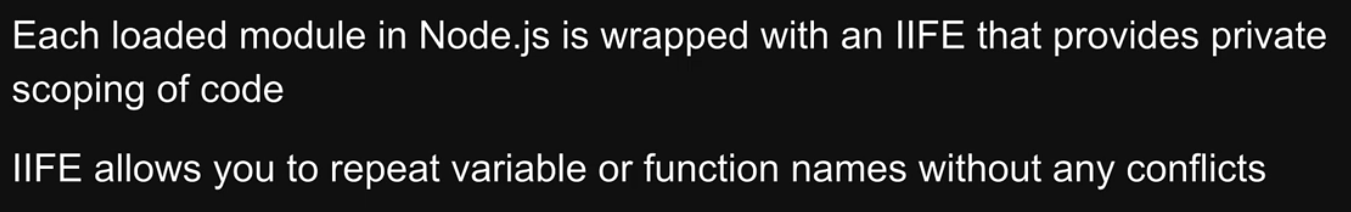
Now we can re use the add function as many times as we want.

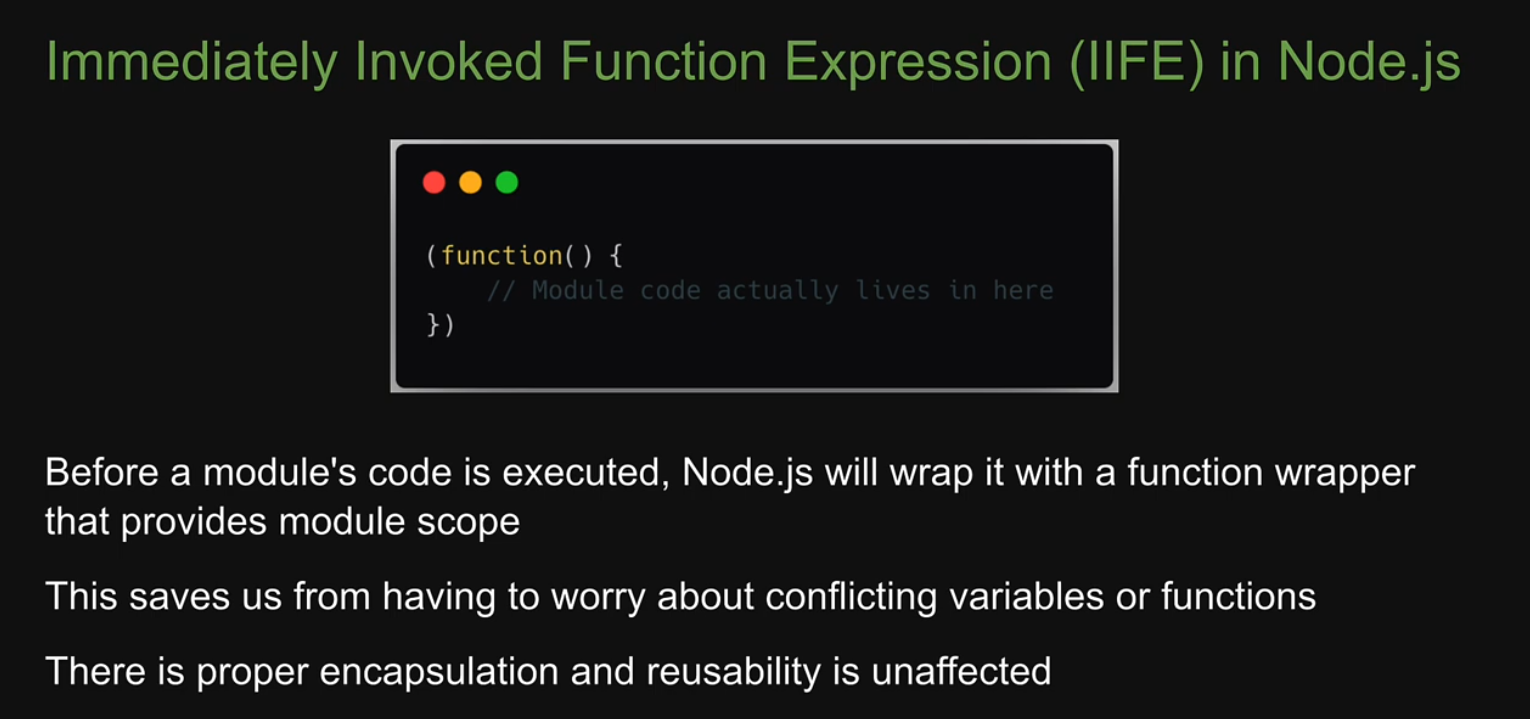


Note that the name of the const variable can be anything. We have changed the name below. The return value doesn’t have to be same as exported value.

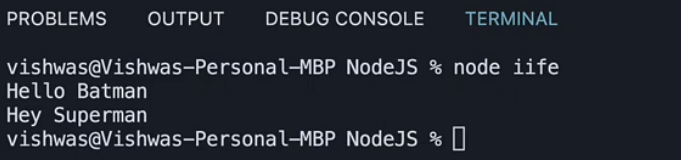


# Module Scope

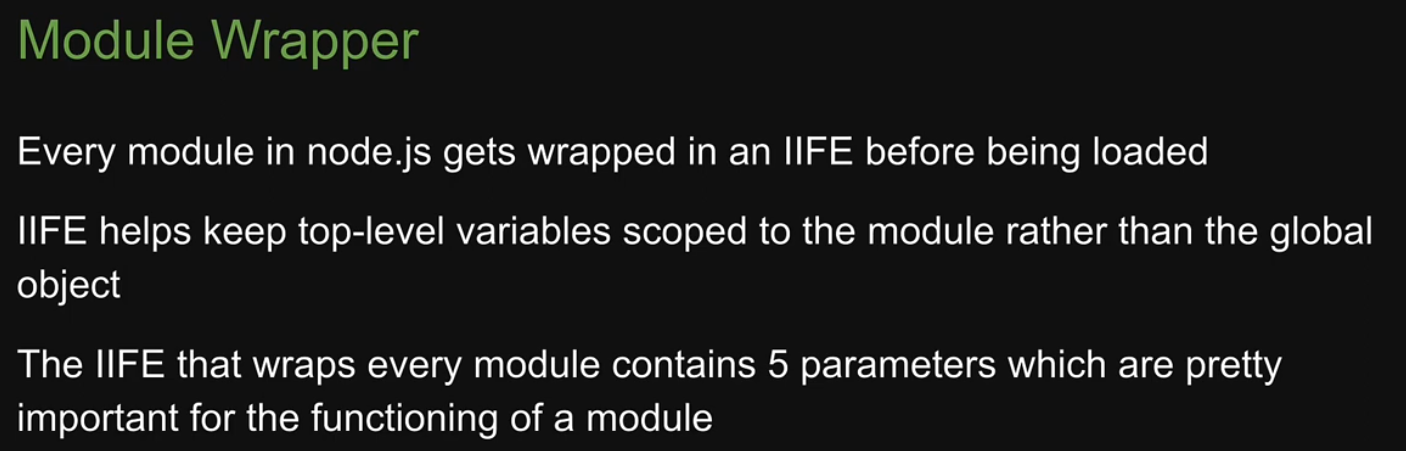




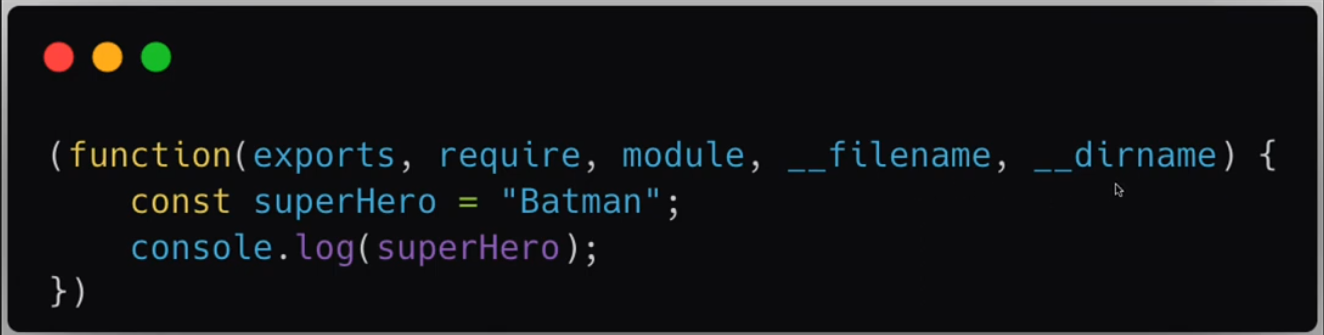




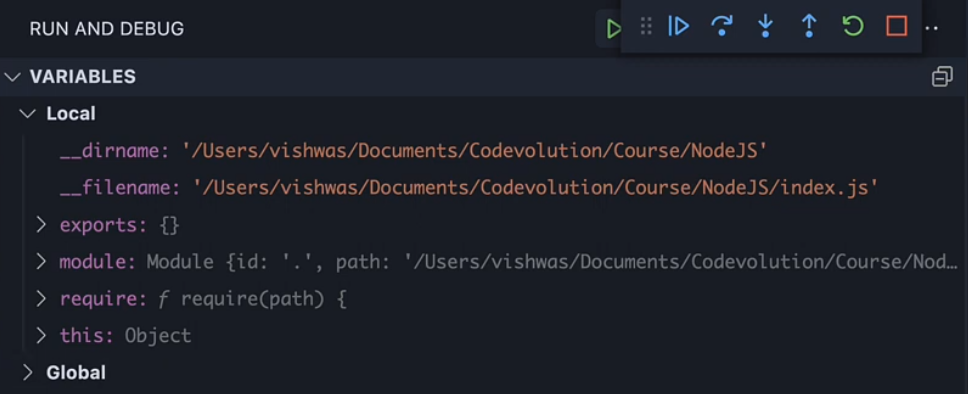
# Module Wrapper



Keep in mind that before a module’s code is executed, node.js will wrap it with a function that contains 5 parameters. This is how we get access to “require” and “module.exports” within a file. They are not magical keywords. They are injected during the execution by node.js.

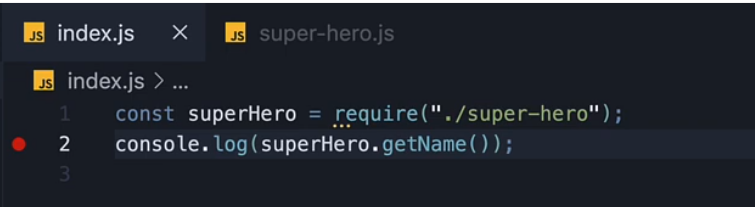


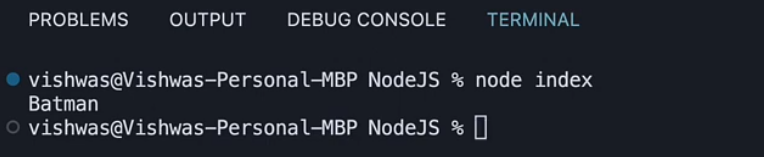
Just to demonstrate the parameters, we have run the program in debug mode



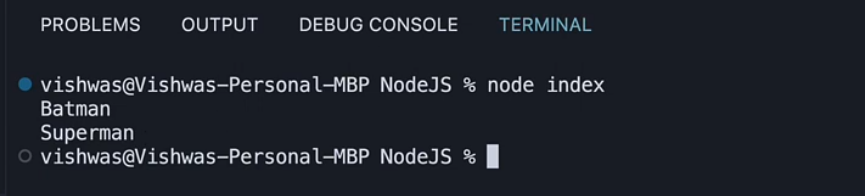
# Module Caching











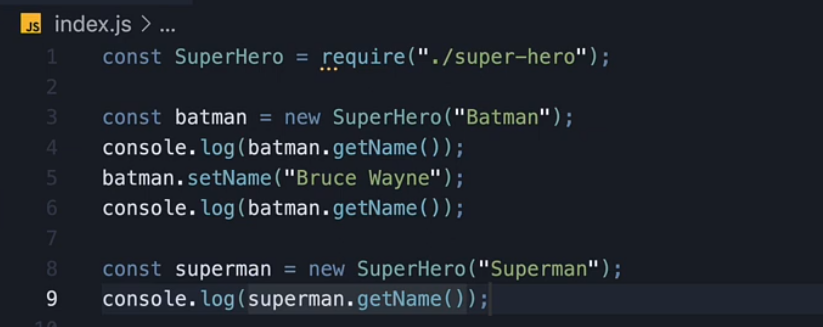
Now we create a new instance of super-hero class. But the getName() returns the same name that we had set earlier i.e. Superman.

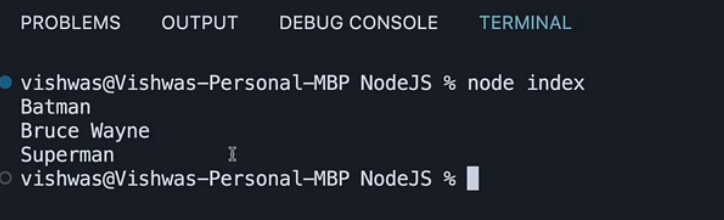
This is because of the concept of module caching.



Now, how do we deal with the situation where we need new instances of the class. This is done by exporting the class instead of the instance.

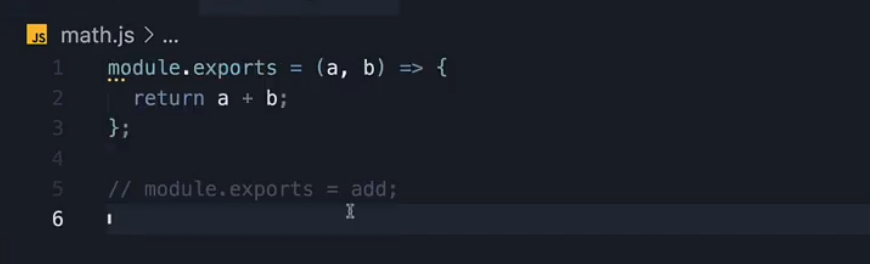






# Import Export Patterns

Directly using module.exports instead of a variable.



# Module.Exports vs Exports

# ES Modules

# Importing JSON and Watch Mode

# Built-in Modules

# Path Module

# Callback Pattern

# Events Module

# Extending from EventEmitter

# Character Sets and Encoding

# Streams and Buffers

# Asynchronous JavaScript

# fs Module

# fs Promise Module

# Streams

# Pipes

# HTTP Module

# Creating a Node Server

# JSON Response

# HTML Response

# HTML Template

# HTTP Routing

# Web Framework