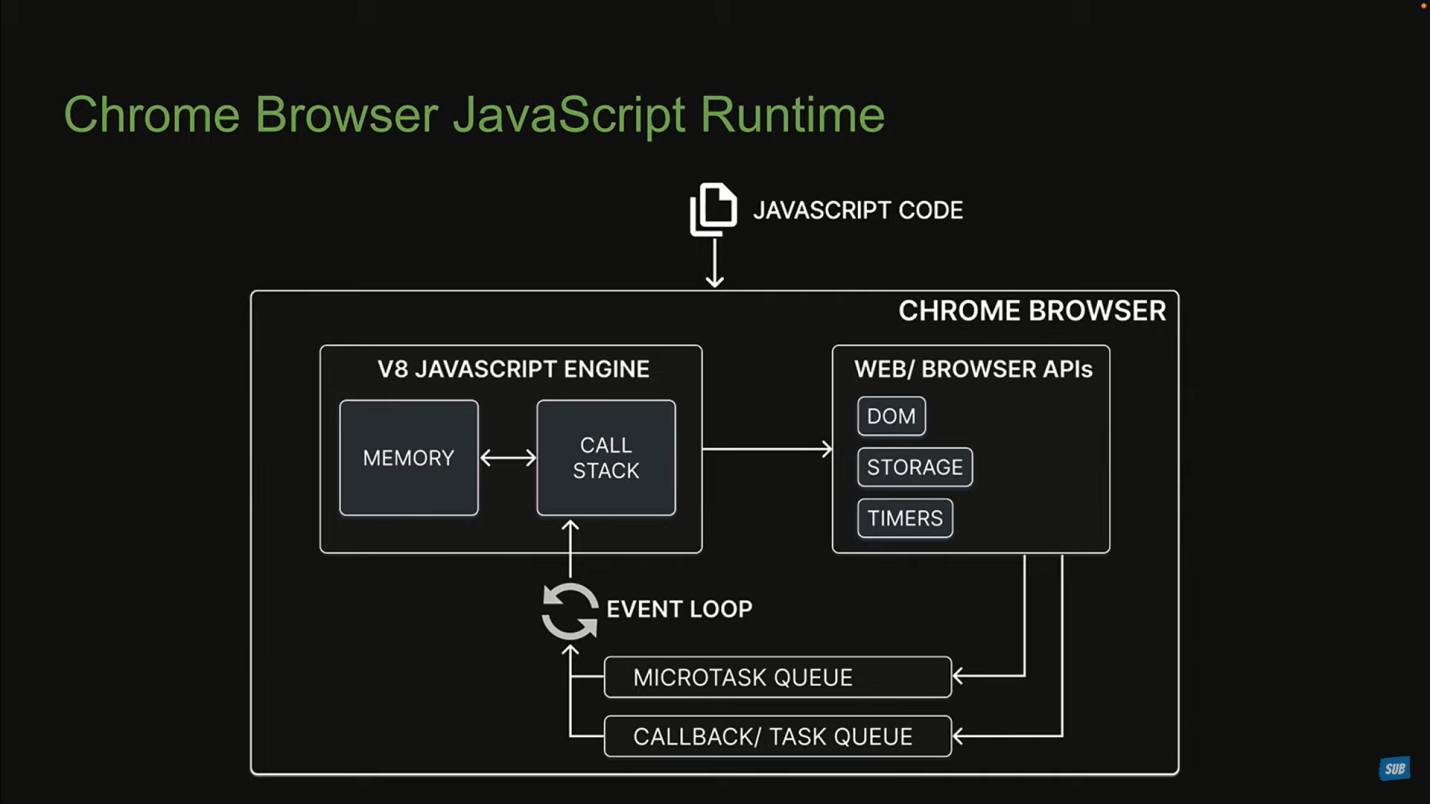
**Node JS**

JavaScript Engine.  
It Converts JS code into Machine code

**JS Run Time**  
It is the environment in which JS program runs.  
Chrome Browser Run Time includes

1. V8 Engine (consists of heap memory, Call Stack)  
2. Web/Browser API’s (DOM, Storage, Timers (setTimeOut, setInterval, Promises)). They are provided by browser.  
3. Queues (Micro Task Queues, Callback Queues, etc.)  
4. Event Loop



**Executing JS with Node**

1. Using Node REPL (Read, Evaluate, Print, Loop). Just type Node in cmd prompt and then perform appropriate actions. (Less Likely to be used.)  
2. Executing code in JS file in cmd prompt. Type node <filename>.js. (More Likely to be used)

**Module**

Module is re-useable chunk of code. In Node JS, every file is a module.  
Types Of Modules: -

1. Local Modules: Modules we create in our application.  
2. Build In Modules: Modules that Node.js ships with out of the box.  
3. Third party Modules: Modules written by other developers that we can use in our application.

**Local Modules**

To Load another module we need ‘require()’ function.  
We can use module.exports to return a module. This returned module can be captured using require() keyword.

**Module Wrapper**

Under the hood, NodeJS does not run our code directly, it wraps the entire code inside a function before execution. This function is termed as Module Wrapper Function or simply Module Wrapper.

Module Wrapper has 5 arguments:

1. \_\_dirname: - Specifies path to folder name of current module.  
2. \_\_filename: - Specifies path of file name of current module.  
3. module: - provides reference to current module.  
4. require: - used to import a module via path.  
5. Exports: -

In any Module we get access to these global variables, but they are specific to the Module in which they are used.

If we export using ‘module.exports’ then we should import using require function  
If we export using ‘export default’ then we should import using ‘import’ keyword

**Asychronous JavaScript**

JavaScript is synchronous, blocking, single threaded language. This nature is however not beneficial for writing apps.  
We want non-blocking asynchronous behaviour which is made possible by browser for frontend and Node.js for backend.

**Build In Modules**

Sometimes referred as Core Modules

1. Path  
2. Events  
3. Fs  
4. Stream  
5. http

**Path Module**

Used while we work for file & directory paths.  
Importing path module.  
const path = require(‘node:path’); // prefixing it with node to indicate that it is built in module.

Or we can write const path = require(‘path’);  
Path Module has about 14 different properties and method exposed. Some of them are as follows:

1. path.basename(\_\_filename or \_\_dirname)  
2. path.extname(\_\_filename or \_\_dirname)  
3. path.parse(\_\_filename)  
4. path.isAbsolute(\_\_filename)  
5. path.join(“folder1”, “folder2”, “index.html” ) --> gives folder1/folder2/index.html  
6. path.resolve(“folder1”, “folder2”, “index.html”) --> will give absolute/complete path