Node Introduction

Hello world and a bit more…

Prerequisites

- **Basic Javascript**
- **Basic HTML**

How we teach - SQ3R

- **Survey**
- **Ask question**
- **Code**
- **Compare**
- **Review**

Our goals

- What is node?
- What are the benefits of using node?
- Write our first node program
- Debugging our app
- Learn to use the REPL for faster development
- Learn about the different parts of node

What is node? - EX

- **Node is a JavaScript runtime**

What is JavaScript

- Programming language
- Dynamic language
- Interpreted. no compiler
- Single threaded*
- Began as the scripting language of the web
- At first only Browsers could run JavaScript now we can use JavaScript to write server applications, mobile apps, desktop apps

JavaScript History

- First graphical web browser Mosaic was released in 1993
- In 1994 Netscape browser was released and took the majority of the bowser market
- Netscape wanted the web pages to be more dynamic with the ability to run a programming language
- JS prototype was released in May 1995
- In 1996 JavaScript was given to ECMAScript to define the specification of the language.
- The latest versions:
  - ES6 - 2015, ES7 - 2016, ES8 - 2017, ES9 - 2018

What is Node - EX

- JavaScript runtime - can run js files
- Let’s see this in action with a small EX
- Install node
- create a js file that prints hello world to the console
- run that script using node

What is Node - EX

- Event driven
- this means it will run our script and that script can subscribe to events
- Let’s see this in action with a small EX
- Create a js script which use setTimeout to register a callback function that will run after a certain time has passed

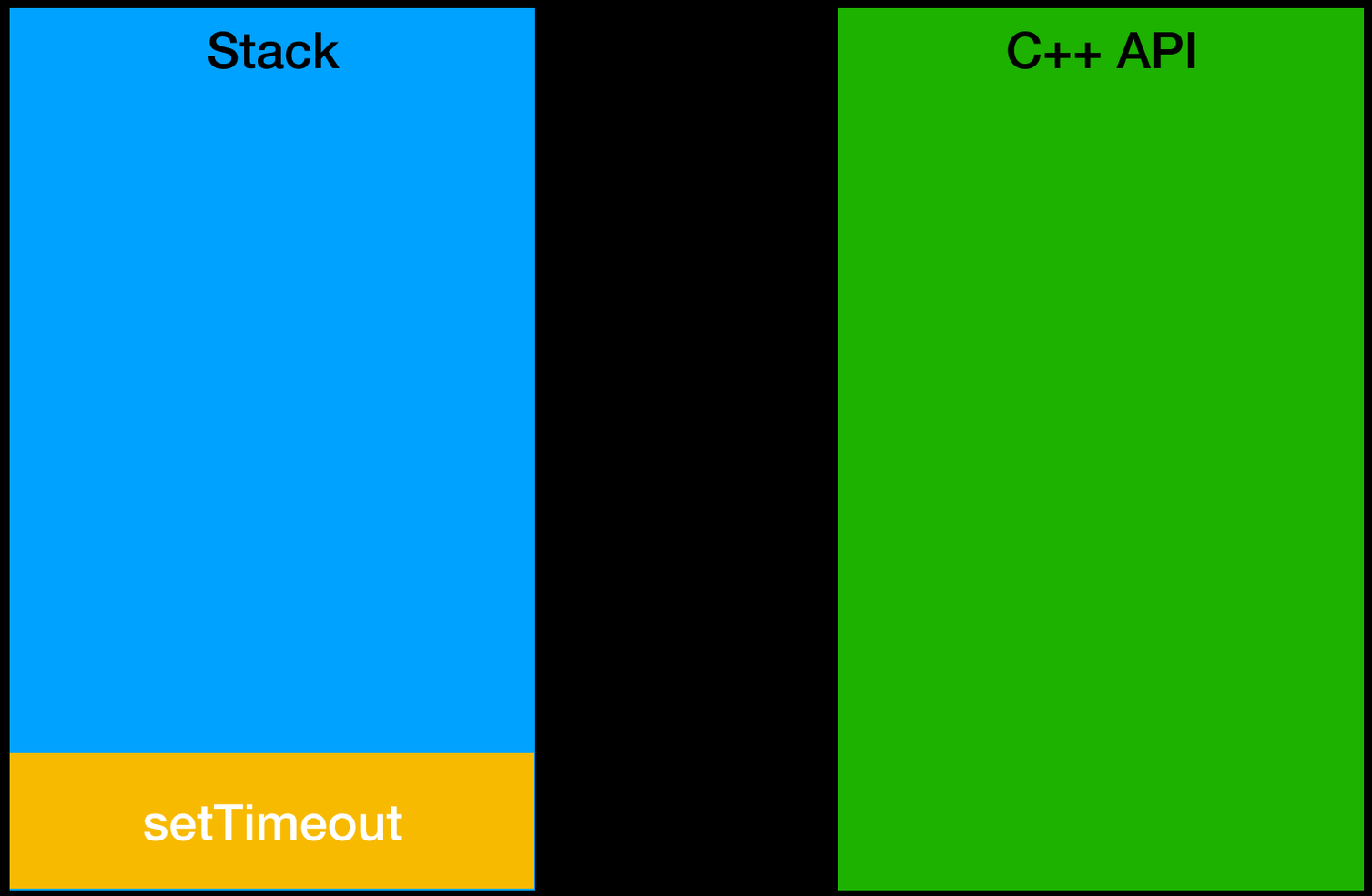
What is Node - Event loop

- How does node activate our callback with setTimeout after a certain time passed. Did node block everything and waited?
- We can place a console.log after the timer to verify that node did not block
- The code that is currently running is placed on the Stack
- JS is single threaded so we have a single Stack
- For async stuff node uses C++ API's
- after the async stuff return it will placed the callback function in a queue called the event loop

What is Node - Event loop

Our code:

setTimeout(exec, 2000)



What is Node - Stack

- We have a single stack
- Code we run is in a frame along with all the variables values
- The event loop can not push stuff to the stack if the stack is not empty

What is Node - C++ API's

- node is a combination of JS v8 engine combined with C++ API's
- the c++ API's use multithreading
- by default node has a thread pool of 4 threads (can be increased if needed)
- The C++ will utilise those threads
- The C++ will also use kernel API's which also run concurently
- So although JS is a single threaded it does not mean we are not using multi threading it just mean we have a single Stack

What is Node - Event Loop

- Simplifying the event loop - you can look at the event loop as a queue
- that queue will init when we start the node process
- the queue will run forever
- the event loop can push stuff on the stack if the stack is empty

Benefits of using node

- Let’s go over some of the benefits of using node as our server technology

Benefits of using node - performance

- https://medium.com/@mihaigeorge.c/web-rest-api-benchmark-on-a-real-life-application-ebb743a5d7a3
- Node performance is really good especially with tasks like database, networking and file system

Benefits of using node - Easy learning curve

- Node is really minimalistic and simple to learn
- The use of js means an even easier larning curve for frontend web developers

Benefits of using node - Cross Platform

- **Node will run on all the popular OS**
- **no need for a server running windows**

Benefits of using node - Single Thread

- We only have one stack trace to deal with which means no multi threading programming
- Still the C++ section is utilizing the thread pool which by default consists of 4 thread
- This means that we only have to deal with single thread while node will take care of using concurrency

Benefits of using node - Microservices

- Very easy to publish packages
- Very easy to use community packages
- Very large community developing packages

Debugging - EX

- Let’s try and place a breakpoint in our previous js script
- On the IDE of your choice try to place a breakpoint and run your the script with the debugger.

Node REPL

- We saw that we can use node runtime to run js scripts with the command:
  - > node <filename>
- We can also activate the runtime by typing
  - > node
- You can then write js commands and see the result of running those commands
- Each command you type is Read then Evaluated then the result is Printed and this process is Looped while the repl is activated
- Let’s open the REPL and examine how to use this tool

Node REPL -EX - Hello world

- Activate the REPL
- type hello world to the console
- notice that hitting the tab key will suggest how to complete the command
- you can type multiple commands by typing:
  - .editor
  - after finishing type ctrl + D fo run the commands
- You can save what you type by typing
  - .save <path-and-filename>
  - .load <path-and-filename>

Node REPL -TIP - Debug Console REPL

- You can activate the REPL while pausing on a breakpoint
- All the variables will be available in your REPL
- you can write your code and see the result right away which is really comfortable
- for multiple lines hit Shift + Enter

Summary

- This lesson is the entry point for understanding basic concepts in node
- We started writing simple programs and ran tham with node along with placing breakpoints and understanding how we can use the REPL to help us with our code writing.