Node

JavaScript

**What is NodeJS?**

* + JavaScript runtime built on Chrome’s V8 JavaScript engine
  + Let’s you run JavaScript on the server
  + Used to build powerful, fast & scalable web applications

**Major benefits of using NodeJS?**

* + **Fast**, built on google chromes v8 engine very fast in code execution
  + **Asynchronous**, never waits for API to return data thus making it asynchronous
  + **Scalable**, because of event mechanism which helps the server respond in a non-blocking way

**Why is NodeJS single threaded?**

* + Uses a **single threaded** model in order to support async

processing

* + **Async processing**, an application can perform better and is more scalable under web loads
  + Node use of **single-threaded** model approach rather than typical thread-based implementation

**Differences between Asynchronous & Non-blocking?**

* + **Asynchronous** means not synchronous using this we can make asynchronous HTTP request that do not wait for the server to respond. These functions continue to respond to the request for which it has already received the server response
  + **Non-blocking** functions are used in regards with I/O operations. They immediately respond with whatever data is available and keeps in case, any answers couldn’t be retrieved then the API returns immediately with an error

**What is package.json?**

* + In NodeJS the heart of the entire application is basically the **package.json** file.
  + It’s basically the **manifest file** that contains the metadata of the project where we define the properties of package

**What is Event-driven programming?**

* + **Event-driven** programming is a programming approach that heavily makes use of events for triggering various functions
  + **Events** can be anything like a mouse click, key press, etc. when an event occurs, a call back function is executed that is already registered with elements

**What is event loop in NodeJS and how does it work?**

* + **Event loop** handles all the asynchronous callbacks in an application
  + It’s is one of the most important aspects of NodeJS & the reason node has **non-blocking** I/O
  + Since node is an **event-driven** language you can easily attach a listener to an event then when the event occurs the callback will be executed by the specific listener
  + Functions like **setTimeout**, **http.get**, and **fs.readFile** are called, NodeJS executed the event loop and then proceeds with the further code without waiting for the output
  + Once entire operation is finish, node receives the output and then executes the **callback** function. This is why all the callbacks functions are placed in a queue in a loop. Once the response is received, they are executed one by one

**Explain REPL in the context of NodeJS?**

* + **REPL** is Node stands for **R**ead, **E**val, **P**rint and **l**oop.
  + It represents computer environment such as a window console or Unix/Linus shell where any commands can be entered and then the system can respond with an output
  + NodeJS comes bundled with a REPL environment by default

**What tasks should be done asynchronously using the event loop?**

* + I/O operations, Heavy computation, anything requiring blocking.

**What steps which “Control Flow” controls the function calls in NodeJS?**

* + Control the order of execution, Collect data, limit concurrency and call the next step in the program

**What is an error-first callback in NodeJS?**

* + **Error-first** callbacks in NodeJS are used to pass errors and data
  + **First parameter** you need to pass to these functions has to be an error object while the other parameters represent the associated data

**Explain the purpose of module.exports?**

* + **Module** is used to encapsulated all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file

**What do you understand by Reactor Pattern?**

* + **Reactor Pattern,** is basically a concept of non-blocking I/O operation
  + Patterns provides a handler that is associated with each I/O operation and as soon as an I/O request is generated it is submitted to a **demultiplexer**
  + **Demultiplexer** is a notification interface which is capable of handling concurrency in non-blocking I/O mode
  + Helps in collecting each and every request in the form of an **event queue**, resulting in the generation of the event queue
  + Simultaneously, we have our event loop which iterates the events present in the Event Queue

**What are the major security implementation?**

* + Major two are Authentications, Error handling

**What do you understand by callback hell?**

* + **Callback hell** is also known as the Pyramid of Doom, it’s a pattern caused by intensively nested callbacks which are unreadable
  + It is caused by improper implementation of the asynchronous logic

**Explain the concept of middleware in NodeJS?**

* + **Middleware** is function receives the Request and response objects
  + Application **request-response cycle** these functions have access to various request & response objects along with the next function of the cycle
  + Most commonly performed tasks by the **middleware** function are executed any type of code, update or modify the request and the response objects, finish the request-response cycle, invoke the next middleware in the stack

**Explain the working of the control flow function?**

* + **Control flow** function is basically the code that is executed between the asynchronous function calls
  + Order of flow

1. The order of execution must be controlled
2. Then, the required data need to be collected
3. Next, the concurrency must be limited
4. Once done, the next step of the program has to be invoked

**Two arguments that async.queue takes as input?**

* + The arguments that async.queue takes as inputs are Task function, concurrency value

**What are global objects in NodeJS?**

* + **Global** are the objects which are global in nature and are available in all the modules of the application that can be used objects directly in your applications
  + **Global objects** can be modules, functions, string, object, etc. some objects can be in the module scope instead of global scope

**Explain the usage of buffer class in NodeJS?**

* + **Buffer class** in NodeJS is used for storing raw data in a similar manner of an array integers
  + It corresponds to a raw memory allocation that is located outside the V8 heap
  + It is a **global class** that is easily accessible can be accessed in an application without importing a buffer module
  + Buffer class is used because pure JavaScript is not compatible with binary data

**How does NodeJS handle the child threads?**

* + NodeJS is a **single threaded** process & doesn’t expose the child threads or thread management methods
  + You can still make use of the child threads using spawn() for some specific asynch**ronous I/O** tasks which execute in the background & don’t usually executed any JS code or hinder with the main event loop in the application

**What are stream in NodeJS along with its various types?**

* + **Streams** in NodeJS are collection of data similar to arrays & strings, they are objects using which you can read data from a source or write data to a destination in a continuous manner
  + These streams are useful for reading and processing a large set of data
  + Four fundamental types **streams**, **readable**, **writeable**, **duplex** and **transform**.

**What is the use of NODE\_ENV?**

* + If project is in the production stage NodeJS promotes convention of making use **NODE\_ENV** variable to flag it.
  + This helps in taking better judgement during the development of the projects
  + when you set you **NODE\_ENV** to production, your application tends to perform 3 times faster

**Differentiate between readFile vs createReadStream?**

* + **readFile**, is fully buffered process which returns the response only when the complete file is push into the buffer & is read
  + It’s memory intensive process and in case of large files the processing can be very slow
  + **createReadStream**, is partially buffered which treats the entire process as an event series. The entire file is split into chunks which are then processed and sent back as a response one by one
  + Once done they are finally removed from the buffer unlike **readFile**, **createReadStream** is really effective for the processing of large files

**What are the various timing features of NodeJS?**

* + **setTimeout/clearTimeout**, used to schedule code execution after a designated amount of milliseconds
  + **setInterval/clearInterval,** used to execute a block of code multiple times
  + **setImmediate/clearImmediate,** usedto executed code at the end of the current event loop cycle
  + **process.nextTick,** usedto schedule a callback function that needs to be invoked in the next iteration of the event loop

**What’s the difference between NodeJS vs Ajax?**

* + Node is a server-side javaScript whereas Ajax uses a client-side technology, Ajax is mostly used for updating or modifying the webpage contents without having to refresh it
  + Other hand node is required to develop the server software that typically executed by the servers instead of the web browsers

**What do you understand about Event Emitter?**

* + **EventEmitter** is a NodeJS class that includes all the objects that capable of emitting events
  + Objects contain an **EventEmitter.on()** function through which more than one function can be attached to the named events that are emitted by the object
  + Whenever an **EventEmitter** object throws an event, all the attached functions to that specific event are invoked synchronously

**Why does Express ‘app’ & ‘server’ must be kept separate?**

* + Express ‘app’ & ‘server’ must be kept separate as by doing this you will be separating the API declaration from the network