What is NodeJS:

In software development, **Node.js** is an open-source, cross-platform runtime environment for developing server-side Web applications. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications.

* Open-source server side runtime environment. Not a language. It’s a wrapper around the code that you are goona write.

Node.js is a server side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009.

Node.js = Runtime Environment + JavaScript Library

Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

* **Asynchronous and Event Driven** All APIs of Node.js library are asynchronous that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.

## Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

* I/O bound Applications
* Data Streaming Applications
* Data Intensive Real time Applications (DIRT)
* JSON APIs based Applications
* Single Page Applications

## Where Not to Use Node.js?

It is not advisable to use Node.js for CPU intensive applications.

Async:

1. Send request
2. go on with other code
3. response come in any time on a callback

Sync:

1. Send request
2. Wait for response
3. go on with other code after response
4. function foo() {
5. var returnValue = bar();
6. console.log(returnValue);
7. }
8. function bar() {
9. return "bar";
10. }
11. In order to make the API "asynchronous" is to change it to use callbacks:
12. function foo() {
13. bar(function(returnValue) {
14. console.log(returnValue);
15. });
16. }
17. function bar(callback) {
18. callback("bar");
19. }

## What is Callback?

Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All APIs of Node are written is such a way that they supports callbacks.

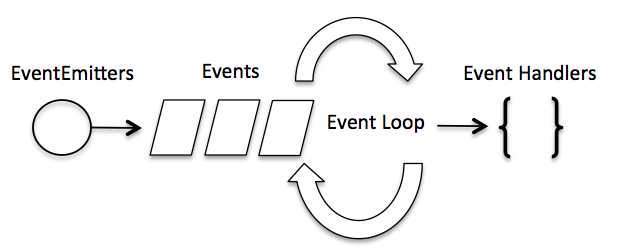
For example, a function to read a file may start reading file and return the control to execution environment immidiately so that next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process high number of request without waiting for any function to return result

**Event Loop :** Node js is a single threaded application but it support concurrency via concept of event and callbacks. As every API of Node js are asynchronous and being a single thread, it uses **async** function calls to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and whenever any task get completed, it fires the corresponding event which signals the event listener function to get executed.

## Event Driven Programming

Node.js uses events heavily and it is also one of the reasons why Node.js is pretty fast compared to other similar technologies. As soon as Node starts its server, it simply initiates its variables, delcares functions and then simply waits for event to occur.

In an event-driven application, there is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected.



# Node.js – Buffers :

Node provides Buffer class which provides instances to store raw data similar to an array of integers but corresponds to a raw memory allocation outside the V8 heap.

## What are Streams?

Streams are objects that let you read data from a source or write data to a destination in continous fashion. In Node.js, there are four types of streams.

* **Readable** - Stream which is used for read operation.
* **Writable** - Stream which is used for write operation.
* **Duplex** - Stream which can be used for both read and write operation.
* **Transform** - A type of duplex stream where the output is computed based on input.

Each type of Stream is an **EventEmitter** instance and throws several events at different instance of times. For example, some of the commonly used events are:

* **data** - This event is fired when there is data is available to read.
* **end** - This event is fired when there is no more data to read.
* **error** - This event is fired when there is any error receiving or writing data.
* **finish** - This event is fired when all data has been flushed to underlying system

This tutorial will give you understanding on commonly used operations on Streams.

## Synchronous vs Asynchronous

Every method in file system module have synchronous as well as asynchronous form. Asynchronous methods takes a last parameter as completion function callback and first parameter of the callback function is error. It is preferred to use asynchronous method instead of synchronous method as former never block the program execution where as the second one does.

## What is Web Server?

Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients.

## Express Overview

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates a rapid development of Node based Web applications. Following are some of the core features of Express framework:

* Allows to set up middlewares to respond to HTTP Requests.
* Defines a routing table which is used to perform different action based on HTTP Method and URL.
* Allows to dynamically render HTML Pages based on passing arguments to templates.

Express is a web development framework for NodeJS.

Bower:

Bower is a package manager for our web packages. Bower is used for front-end work.

* Npm install bower –g
* Bower init
* Bower install –save bootstrap
* Use git powershell: bower install font-awesome –save

What I learned:

Building web apps:

* Got express configured
* Simple routing
* Bootstrap templates.
* Bower

Gulp:

Task manager for web projects.

* Installed with npm
* package based.
* Code based configuration.
* Npm install gulp –save-dev

JSHint:

Code quality enforcement

* Detects potentials error.
* Enforces coding conventions.

JSCS: Code style.

* Enforces style conventions.
* Easily configurable.

npm install –-save-dev gulp-jshint gulp-jscs jshint-stylish

Wiredep:

npm install wiredep -–save-dev

Gulp:

Configuring Gulp.

Enforcing coding standard

Injectinf css and js references

Auto restart our application

Templating engines:

Allows exprss to build our HTML pages for us.