**What is NodeJS**

Node JS is a JavaScript runtime environment. But what is that, one might ask. By run-time environment, the infrastructure to build and run software applications is meant. To build applications in JavaScript, in this case. Let’s see what are the Node JS definition versions.

Node as a “Javascript runtime built on Chrome V8 engine”. Node.js is an open-source and cross-platform environment to execute code. it is a development platform aimed at building server-side applications. And also Node is a platform with its own web server for better control. That is certainly enough to grasp the main idea.

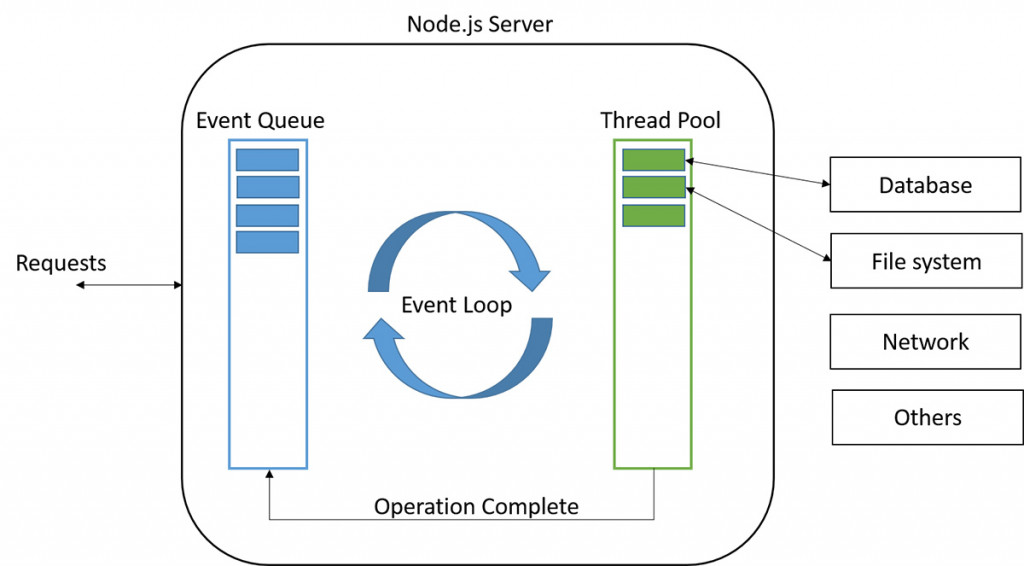
A brief summary would be as goes:

* Node JS is a server framework, and is free
* It runs on Windows, Linux, Mac OS, etc.
* Node utilizes JavaScript on the server

How does Node JS work? Taking a simple task of opening a file on a server, the sequence would be:

* A task goes to the file system
* The system is ready for next requests
* When a file is opened and read, the system sends the content to the client

In other words, with Node you do not have to wait and can go on with next tasks. This is one of the reasons it is so efficient.



**Why we Choose NodeJS as our backend**

The one of the major reasons we use Nosejs as our backend is that this JavaScript language doesn’t block I/O – meaning input/output communication method. Another benefit is single-threaded event loop, that is responsible for abstracting I/O from external requests. Speaking plainly, this means that Node initiates the event loop at the start, processes the input, and begins the order of operations. And also it has below advantages also

* Google JavaScript engine. Translation: fast and scalable web apps in a result.
* For server-side applications. Meaning, Node is an event-driven model of programming, where the flow is determined by certain events (user actions, messages, etc.).
* Easier and scalable. That is, to make apps like Uber or Trello and scaling out on multi-CPU servers.
* Per-process and across servers. Translation: Node can scale on individual process basis spreading out the load across multi-core servers.

This all seems a bit tough, we realize. So behold a summary of coherent benefits to using Node.js.

main reasons to use Node.js

* Good for beginner developers, JavaScript is simple to learn, rich framework (Angular, Node, Backbone, Ember)
* It is fast, due to Google innovative technologies and the event loop
* Ability to keep data in native JSON (object notation) format in your database
* Multiple modules (NPM, Grunt, etc.) and supportive community
* Good to create real-time apps, such as chats and games
* Single free codebase
* Good for data streaming, thus for audio and video files, as example
* Sponsored by Linux Foundation, as well as PayPal, Joylent, Microsoft, Walmart
* Wide range of hosting options
* JS is the longest running language, 99% of developers know some of it

**Why we Choose AngularJS for Frontend**

Angular uses HTML to define the app’s user interface. HTML is a declarative language which is more intuitive and less convoluted than defining the interface procedurally in JavaScript. HTML is also less brittle to reorganize than an interface written in JavaScript, meaning things are less likely to break. Plus you can bring in many more UI developers when the view is written in HTML.

HTML is also used to determine the execution of the app. Special attributes in the HTML determine which controllers to use for each element. These attributes determine “what” gets loaded, but not “how”. This declarative approach greatly simplifies app development in a sort of WYSIWYG (what you see is what you get) way. Rather than spending time on how the program flows and what should get loaded first, you simply define what you want and Angular will take care of the dependencies.

All the points up till now mean that you get to write less code. You don’t have to write your own MVC pipeline. The view is defined using HTML, which is more concise. Data models are simpler to write without getters/setters. Data-binding means you don’t have to put data into the view manually. Since directives are separate from app code, they can be written by another team in parallel with minimal integration issues. Filters allow you to manipulate data on the view level without changing your controllers. Yes, this is sort of a summary bullet point, but writing less code is a one of the main reason.

**Connection Between Frontend and Backend**

The Connection between Front end and backend mainly established using AJAX requests. Backend is developed using Mainly NodeJS and Front end developed by using AngularJS , JavaScript and HTML.



**Defining HOST**

Since this is an application that runs on peer to peer technology we have to connect individual peers to other peers. So give right Host number and connect with the all other peers by following start topology has to done so carefully

**Updating the Host in each Peer**

**Inside the Record Vote Function**

function RecordeVote(CName)

{

// 3002 / 3007 / 3008

$.ajax({url: "http://localhost:3005/spawnBrew/"+CName, success: function(result)

{

alert("Your Input Recorded ..!");

},

error: function(result)

{

alert("Your Input Recorded ..!");

}

});

}

**Inside the Display Block chain function**

<div class="col-md-12 align-self-center">

<h1 class="text-themecolor">Graduate Student Election - 2018 Hampton University</h1>

<h4 class="text-themecolor"><a class="asynch\_link" href="http://localhost:3005/spawnBrew/DisplayVotes"> Click Here to Display BlockChain</a></h4>

</div>