1. Getting started

What is Node

Node.js is an open source and cross platform runtime environment to execute JavaScript outside the browser. Node is ideal for building highly scalable, data-intensive and real-time apps.

Benefits of Node:

- Great for prototyping and agile development
- Superfast and highly scalable
- Uses JavaScript everywhere
- Cleaner and more consistent codebase
- Large ecosystem of open-source libraries

Node architecture

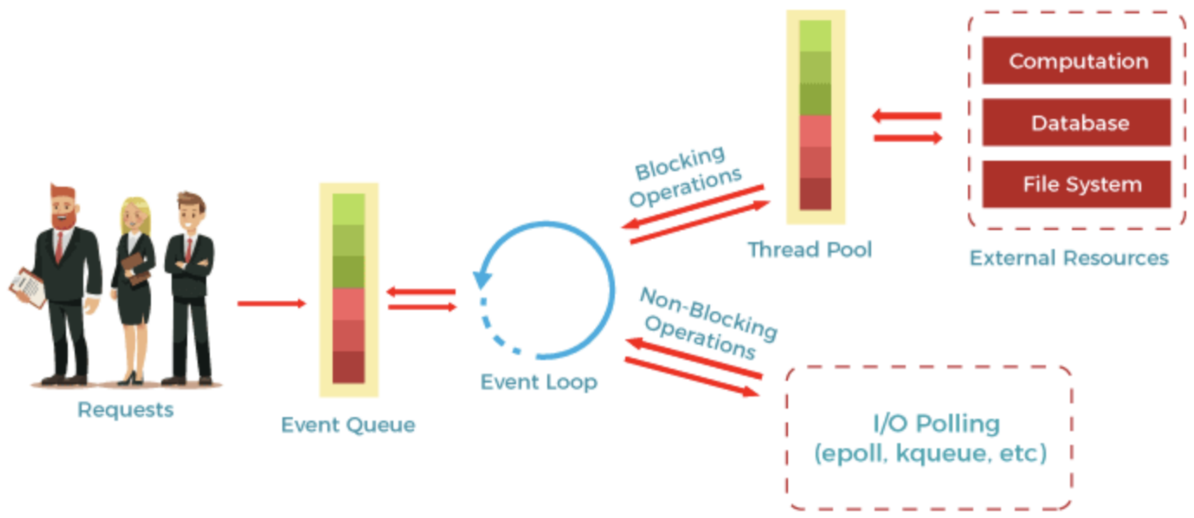
Before Node, JavaScript was used to build applications that run inside of a browser. Every browser has something called a JavaScript Engine. For example, Edge uses Chakra, Firefox uses Spidermonkey and Chrome uses V8.
In 2009 Ryan Dahl created Node to execute JS outside of a browser, he took V8 engine and embedded it inside a C++ program and called it Node. So, Node is a runtime environment to execute JS code outside of a browser, but it also contains certain objects that provide the environment for JS code, but these objects are not the same that we have in a browser.

Node is a program that includes a V8 engine and some additional modules that give us capabilities not available inside browsers.

How Node works

Node has a non-blocking or asynchronous architecture, a single thread is used to handle multiple requests. When a request arrives, that single thread is used to handle that request. Node has something called Event Queue, Node is continuously monitoring this Event Queue in the background, when it finds an event in the queue it will take it out and process it. This architecture makes Node ideal for building I/O intensive apps. It is possible to serve more clients without the need to include more hardware. In contrast, Node should not be used for CPU intensive apps, like video encoding or an image manipulation service.

The workflow of a Node.js web server typically looks like the following diagram.



1. According to the above diagram, the clients send requests to the web server to interact with the web application. These requests can be non-blocking or blocking and used for querying the data, deleting data, or updating the data.
2. The web server receives the incoming requests and adds those to the Event Queue.
3. After this step, the requests are passed one by one through the Event Loop. It checks if the requests are simple enough not to require any external resources.
4. The event loop then processes the simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients.
5. A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing external resources, such as computation, database, file system, etc.
6. Once the task is completed, the response is sent to the Event Loop that sends that response back to the client.

Installing Node.js

In a browser we can open the following link:

https://nodejs.org

You should download the latest stable version.
In a terminal we can run `node -version` and check that Node is installed and what version we are using.

Your first Node program

To create and execute a program with Node, first you need to create a JS file and add some code to it, for example:

```
function sayHello(name) {
      console.log('Hello ' + name);
}

sayHello('Seraffina');
```

Then, back in the terminal to execute the code just run `node` and then pass the file as an argument:

```
first-app $node app.js
hello Serafina
```