Introduction to NodeJS

Unit#4

Marwadi University

Department of Information Technology

Advanced Web Programming (3161611)

Tejas Chauhan

Highlights

- Introduction
- Setup Node JS Environment
- Package Manager
- Features
- Console Object
- Concept of Callbacks

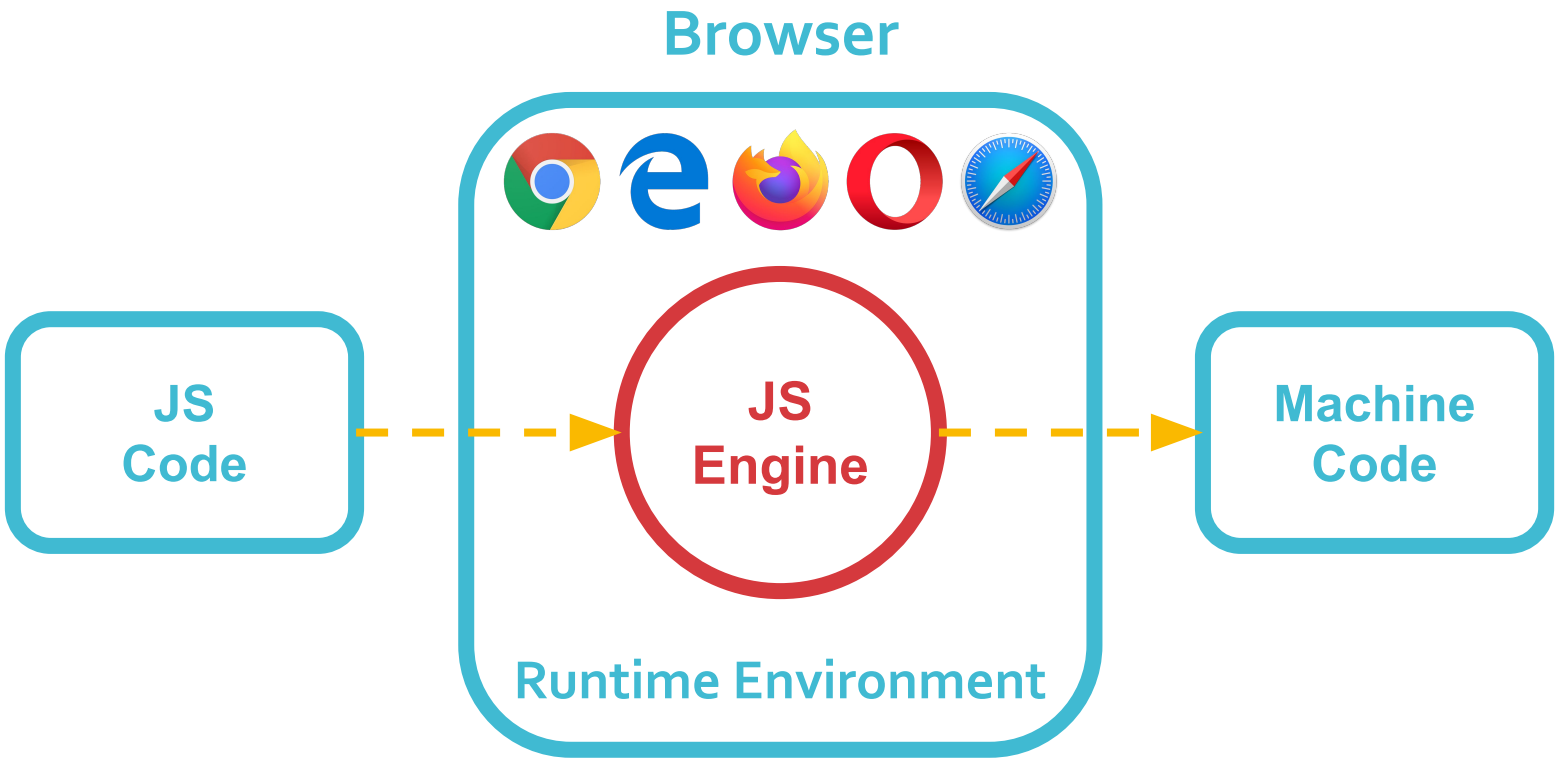
Introduction

- A runtime environment for executing JavaScript code
- Open-source
- Cross-platform
- We often use Node to build back-end services
  - API: Application Programming Interface
- Node is ideal for highly-scalable, data-intensive and real-time apps.

NodeJS

- Easy to get started
- Great for prototyping and agile development
- Javascript everywhere
- Cleaner and more consistent codebase
- Large ecosystem of open-source libs

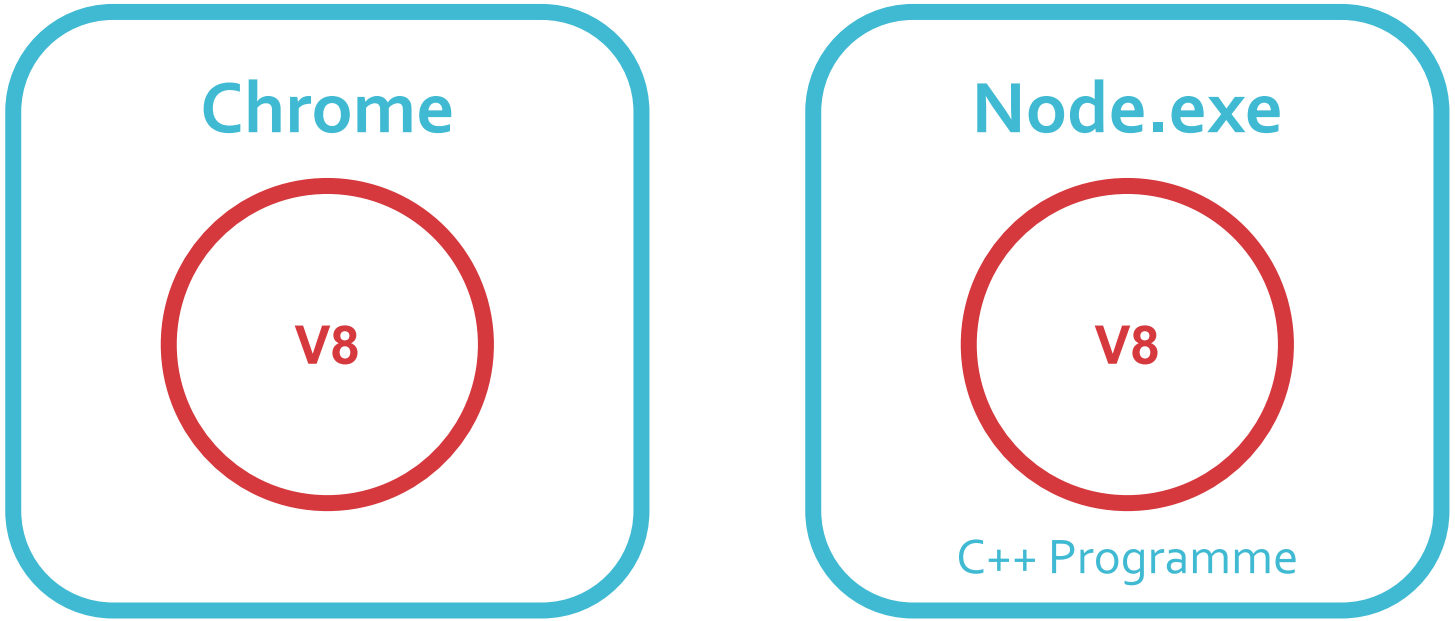
Browser Architecture



- Edge: Chakra, Firefox: SpiderMonkey, Chrome: V8

NodeJS Architecture

- Ryan Dahl (2009)
  - Node.js: A runtime environment for executing JavaScript code.



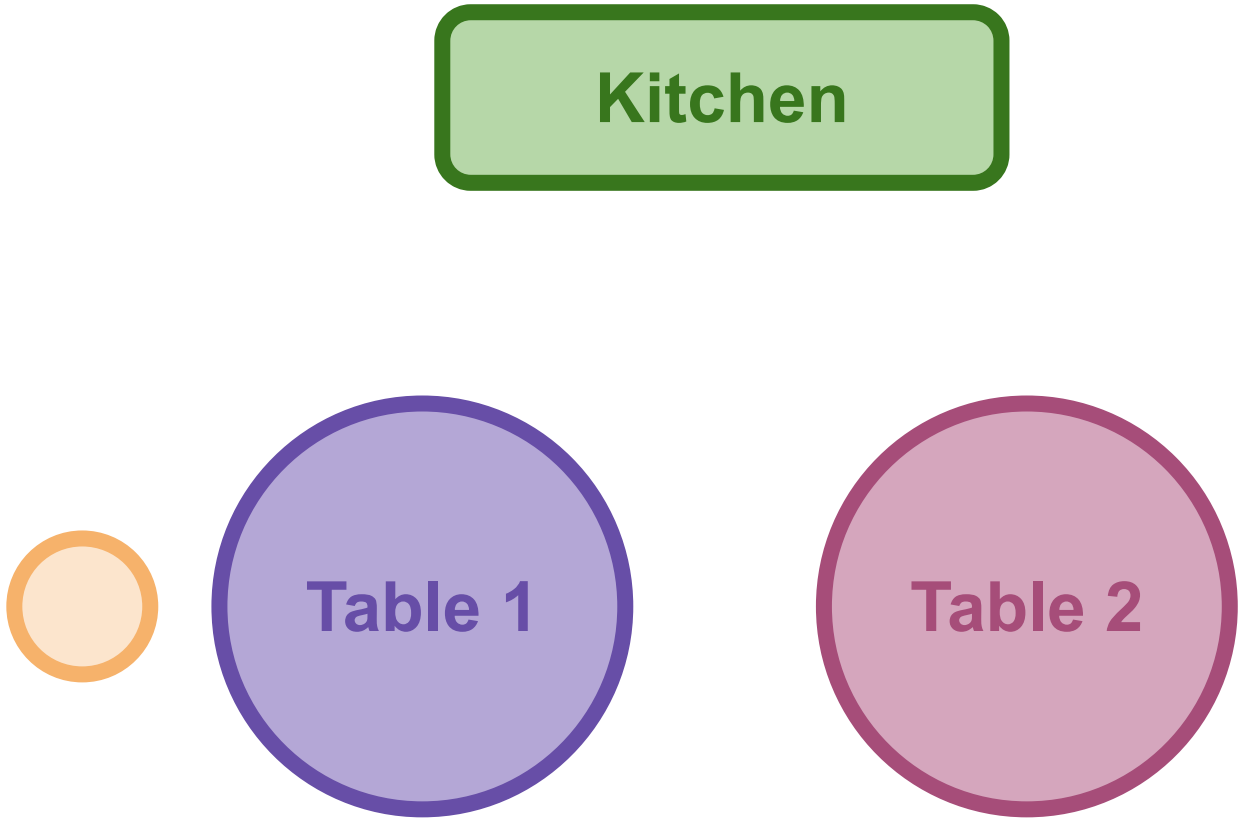
- So, we can work with file system, network, database etc.

NodeJS

- Node.js is not a programming language!
- Node.js is not a framework!
- Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine to execute JavaScript code.
- Node is ideal: Highly-scalable, data-intensive and real-time apps

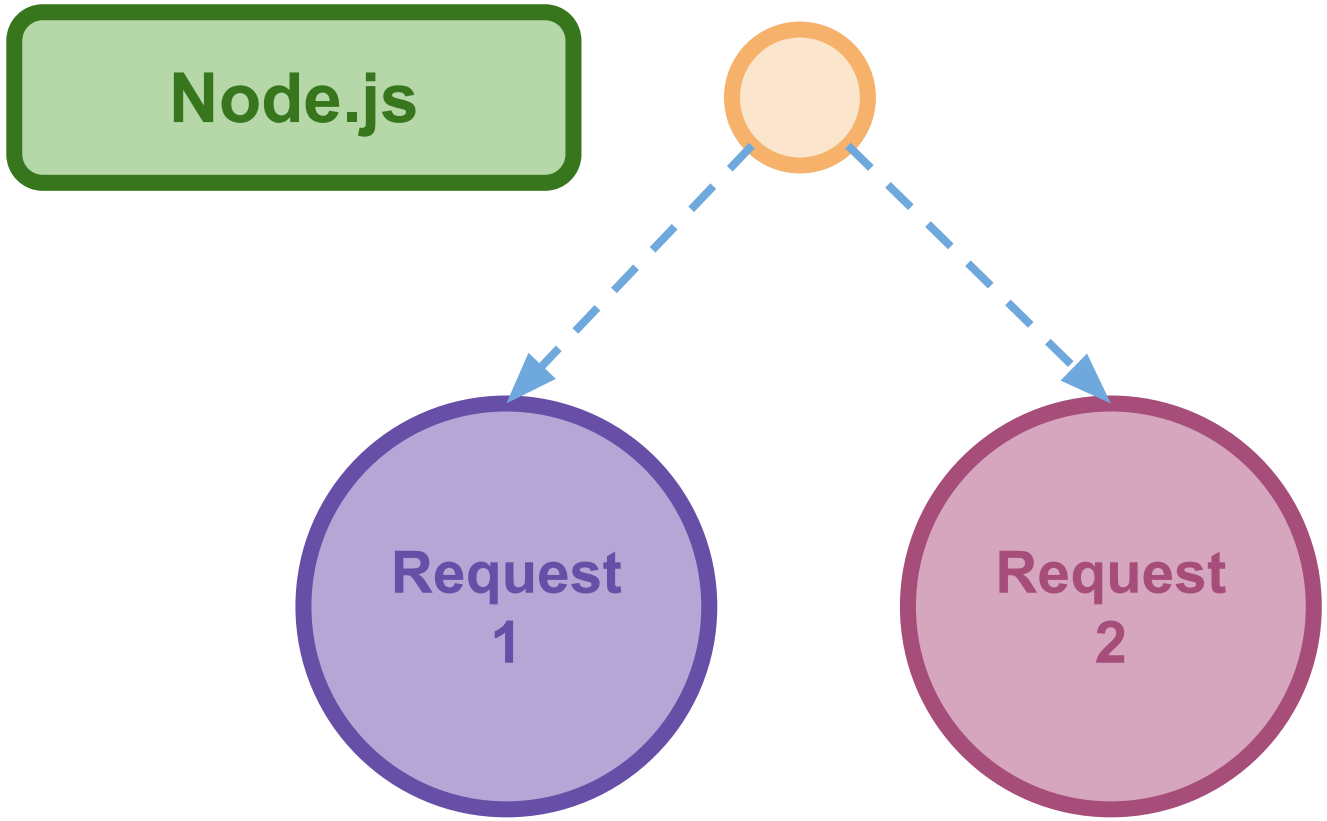
How it Works?

- Non-blocking Asynchronous

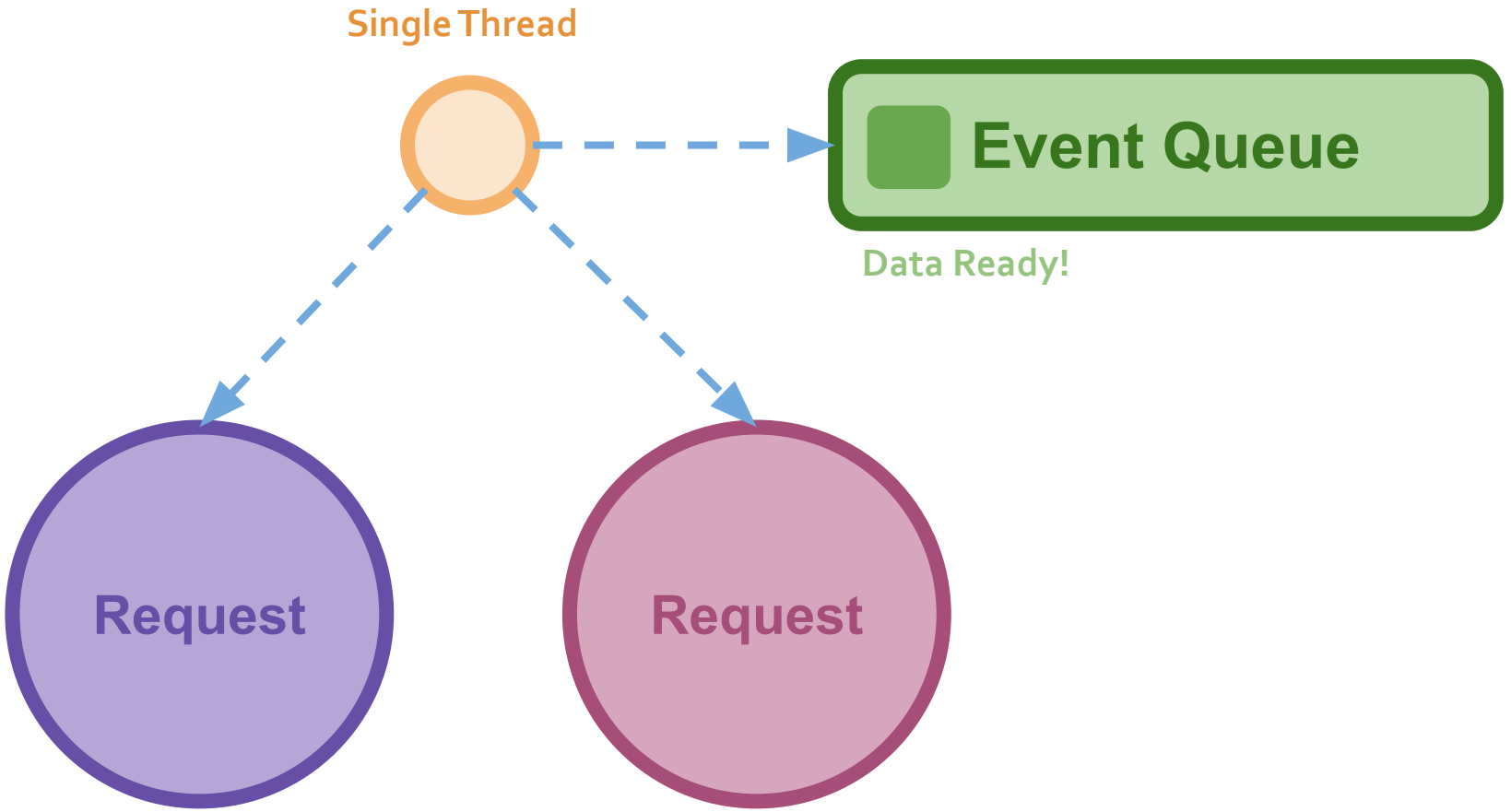


How it Works?

- Non-blocking, Asynchronous, Single Threaded



How it Works?



Features

- Extremely fast:
  - Built on Google Chrome's V8 JavaScript Engine
  - Its library is very fast in code execution.
- I/O is Asynchronous and Event Driven:
  - All APIs of Node.js library are asynchronous i.e. non-blocking.
  - So 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.
- Single threaded:
  - It follows Single threaded model with event looping.

Features

- No buffering:
  - It cuts down the overall processing time while uploading audio and video files.
  - The applications never buffer any data.
  - These applications simply output the data in chunks.
- Highly Scalable:
  - It is highly scalable because event mechanism helps the server to respond in a non-blocking way.
- Open source:
  - Open source community support: Provided many excellent modules to add additional capabilities to Node.js applications.
- License:
  - It is released under the MIT license.

Setup NodeJS Environment

- Let's install Node and write some code!
  - Nodejs.org
- Windows: Download file and follow instructions on installer.
- Linux:
  - sudo apt install nodejs
  - sudo apt install npm
- Check: command/terminal:
  - node -v or node –version
  - npm -v or npm –version

Package Manager

- Two main functionalities
- Online repositories
  - For node.js packages/modules which are searchable on https://www.npmjs.com/
- Command line utility
  - To install Node.js packages: npm
  - Do version management and dependency management of Node.js packages

Package Manager

- Installing Modules using npm
- Syntax:

```
npm install <Module Name>
```

- Example to install famous Node.js web framework called express

```
npm install express
```

Node.js REPL

- Read:
  - It reads user's input; parse the input into JavaScript data-structure and stores in memory.
- Eval:
  - It takes and evaluates the data structure.
- Print:
  - It prints the result.
- Loop:
  - It loops the above command until user press ctrl-c twice.

Node.js REPL

- Computer environment / shell
  - You can enter the commands
  - System responds with an output in an interactive mode
- REPL starts by running "node" command on the command prompt / terminal.
- Can also execute various mathematical operations.

Console Object

- The console module provides a simple debugging console that is similar to the JavaScript console mechanism provided by web browsers.
- The module exports two specific components:
  - A Console class with methods such as console.log(), console.error() and console.warn() that can be used to write to any Node.js stream.
  - A global console instance configured to write to process.stdout and process.stderr. The global console can be used without calling require('console').

```
console.log('hello world');
console.log('hello %s', 'world');
```


Console Object

- Console class methods:
  - console.assert(value[, ...message]): writes a message if value is falsy or omitted.
  - console.clear(): will clear only the output in the current terminal viewport for the Node.js binary.
  - console.count([label]): Maintains an internal counter specific to label and outputs to stdout the number of times console.count() has been called with the given label.
  - console.countReset([label]): Resets the internal counter specific to label.
  - console.error([data][, ...args]): Prints to stderr with newline.

Console Object

- Console class methods:
  - console.table(tabularData[, properties]): Try to construct a table with the columns and rows of tabularData and log it.
  - console.time([label]): Starts a timer that can be used to compute the duration of an operation.
  - console.timeEnd([label]): Stops a timer that was previously started by calling console.time() and prints the result to stdout.
  - console.timeLog([label][, ...data]): For a timer that was previously started by calling console.time(), prints the elapsed time and other data arguments to stdout.
  - And few more.....

Callbacks

- In a synchronous program, you would write something along the lines of:

```
function processData () {
  var data = fetchData ();
  data += 1;
  return data;
}
```

- This works just fine and is very typical in other development environments.
- However, if fetchData takes a long time to load the data (maybe it is streaming it off the drive or the internet), then this causes the whole program to 'block' - otherwise known as sitting still and waiting - until it loads the data.

Callbacks

- Node.js, being an asynchronous platform, doesn't wait around for things like file I/O to finish.
- Node.js uses callbacks.
- A callback is a function called at the completion of a given task; this prevents any blocking, and allows other code to be run in the meantime.

Callbacks

- The Node.js way to deal with the previous example would look a bit more like this:

```
function processData (callback) {
  fetchData(function (err, data) {
    if (err) {
      console.log("An error has
occurred. Abort everything!");
      return callback(err);
    }
    data += 1;
    callback(data);
  });
}
```


Callbacks

- At first glance, it may look unnecessarily complicated, but callbacks are the foundation of Node.js.
- Callbacks give you an interface with which to say, "and when you're done doing that, do all this."
- This allows you to have as many IO operations as your OS can handle happening at the same time.
- For example, in a web server with hundreds or thousands of pending requests with multiple blocking queries, performing the blocking queries asynchronously gives you the ability to be able to continue working and not just sit still and wait until the blocking operations come back.
- This is a major improvement.

Callbacks

- The typical convention with asynchronous functions (which almost all of your functions should be):

```
function asyncOperation ( a, b, c, callback ) {
  // ... lots of hard work ...
  if ( /* an error occurs */ ) {
    return callback(new Error("An error has
occurred"));
  }
  // ... more work ...
  callback(null, d, e, f);
}

asyncOperation ( params.., function ( err,
returnValues.. ) {
  //This code gets run after the async
operation gets run
});
```


Callbacks

- The general idea is that the callback is the last parameter.
- The callback gets called after the function is done with all of its operations.
- Traditionally, the first parameter of the callback is the error value. If the function hits an error, then they typically call the callback with the first parameter being an Error object.
- If it cleanly exits, then they will call the callback with the first parameter being null and the rest being the return value(s).

Review

- Introduction
- Setup Node JS Environment
- Package Manager
- Features
- Console Object
- Concept of Callbacks

Thank You.