NODE JS.

Chapter 1. INTRODUCTION

- What is Node.js?
- Features of Node.js
- Who Uses Node.js?
- Concepts
- Where to Use Node.js?
- Where Not to Use Node.js?
- History of Node.js

WHAT IS NODE JS?

- JavaScript runtime (usually Chrome V8, also Microsoft Chakra)
- Fast and scalable network applications
- Event-driven
- Non-blocking Input/Output
- Open-source
- Cross-platform
- JavaScript programming language

Features: Asynchronous and Event Driven

- No waiting for an API response
- Server continues to next line of code
- Event system triggers callback when response is ready
- Single threaded but handles more requests
- No buffering of any data.

Who Uses Node.js?

Walmart Netflix Uber Intel Joyent Microsoft PayPal Red Hat GoDaddy

Amazon NodeSource Cars.com CodeFresh Dynatrace Fidelity Google

Groupon nearForm npm RisingStack SAP Saucelabs Snyk

Sphinx Yahoo! Yld! Samsung

https://www.quora.com/What-companies-are-using-Node-js-in-production

https://nodejs.org/en/foundation/members

Node.js Foundation Members

RisingStack

SAUCELABS

Joyent®

GROUPON

Google

IBM

redhat.

PayPal

YAHOO!

intel®

Microsoft

GoDaddy™

npm

cars.com™

SAP®

YLD

snyk

Who Uses Node.js?

IBM Intel Joyent Microsoft PayPal Red Hat GoDaddy NodeSource

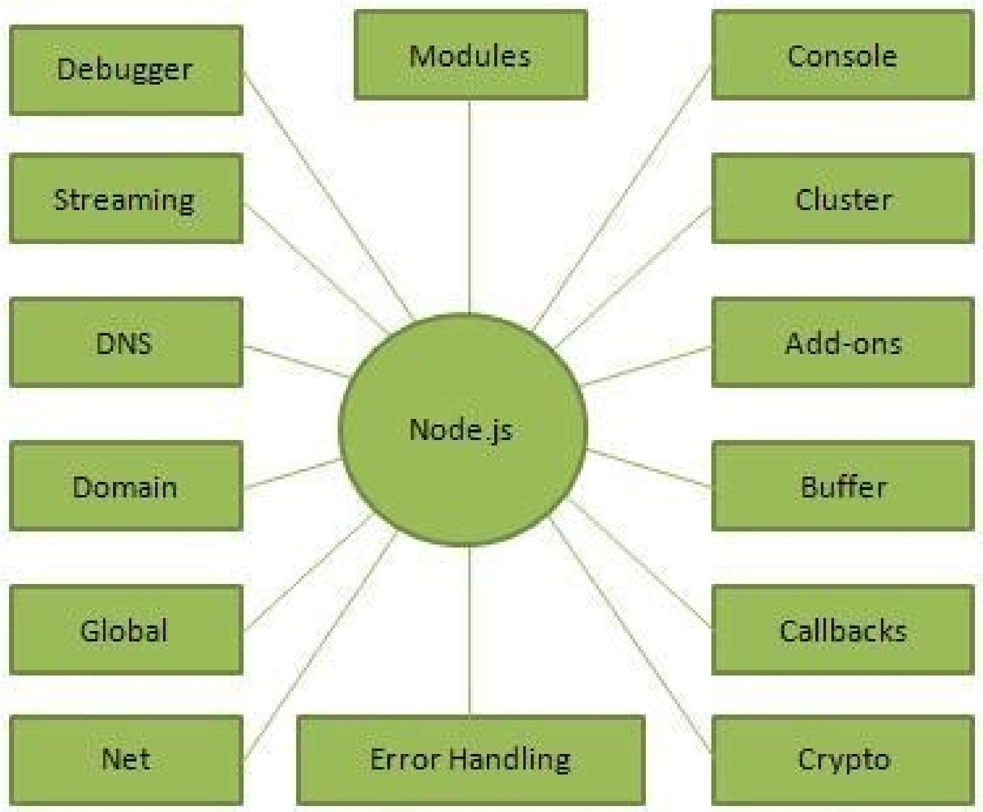
Cars.com CodeFresh Dynatrace Fidelity Google Groupon nearForm

npm RisingStack SAP Saucelabs Snyk Sphinx Yahoo! Yld! Samsung

https://www.quora.com/What-companies-are-using-Node-js-in-production

https://nodejs.org/en/foundation/members

CONCEPTS


Debugger
Modules
Console
Streaming
Cluster
DNS
Add-ons
Node.js
Domain
Buffer
Global
Callbacks
Net
Error Handling
Crypto

Where to Use Node.js?

- 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.

History of Node.js

- 2009 Ryan Dahl inspired to create Node.js while travelling, he presented his work at JSConf, early npm is previewed
- 2010 Express, Socket.io created
- 2011 LinkedIn, Uber adopt Node.js
- 2012 Dahl assigns Joyent governance of Node.js
- 2013 Ghost Blogging CMS created, MEAN stack (MongoDB, ExpresssJS, AngularJS, and Node.js) arrives, eBay, Walmart, PayPal adopt Node.js
- 2014 Joyent initiates Node.js Advisory board created. IO.js launched
- 2015 Joyent initiates Node.js Foundation launched, IO.js and Node.js merge. Apigee, RisingStack and Yahoo! Join the Node.js Foundation, first Node.JS Foundation Conference -Node Interactive- held. IBM acquires StrongLoop
- 2016 Google Cloud Platform joins Node.js Foundation. Samsung acquies Joyent. Node.js 6 becomes LTS

https://blog.risingstack.com/history-of-node-js
https://nodejs.org/static/documents/2016-survey-report.pdf

Chapter 2.
ENVIRONMENT SETUP

- Software Installation
  - Node.js
  - NPM
  - code editor (notepad++, Visual Studio Code, TextMate, IntelliJ)
- PATH Setting
  - non-Windows: Export PATH=$PATH:/usr/local/nodejs/bin
  - Windows: c:\program files\nodejs in Environment variables
- Verify Installation
  1. console.log("Hello, World!") in main.js
  2. node main.js
  3. Hello, World! Is displayed

Chapter 3.
FIRST APPLICATION: Web Server

- Step 1
  - Import module using **require**

- Step 2
  - Create the server using a function

- Step 3 Test Request and Response
  - Run the server
  - Request server page using web browser

Chapter 4.
REPL Terminal

- Read - Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval - Takes and evaluates the data structure.
- Print - Prints the result.
- Loop - Loops the above command until the user presses ctrl-c twice.

Getting to a REPL Terminal

- Type on terminal (command line)
  - node
- Use someone else‘s
  - https://repl.it/languages/nodejs

Using REPL

- Simple Expressions
- Variables
- Multiline Expression
  - (shift - enter)
- Underscore Variable
  - Use underscore (_) to get the last result
- REPL Commands
- Stopping REPL
  - (ctrl-c twice)

Chapter 5.
Node Package Manager: NPM

- Provides searchable repositories for node.js packages/modules hosted on

  https://www.npmjs.com

- Command line utility to
  - install node.js packages,
  - do version management
  - and dependency management of node.js packages

NPM version

- Check the version

```
npm --version
```

- Update the version

```
sudo npm install npm -g
```

- Advanced: NVM (node version manager)
  - https://github.com/creationix/nvm/blob/master/README.md

Working with Packages: Installing, Updating, Searching

- Installing Modules
  - npm install <Module Name>
- Uninstalling Modules
  - npm uninstall <Module Name>
- Updating Modules
  - npm update <Module Name>
- Searching Modules
  - npm search <Module Name>

Working with Packages: package.json

- name - name of the package
- version - version of the package
- description - description of the package
- homepage - homepage of the package
- author - author of the package
- contributors - name of the contributors to the package
- dependencies - list of dependencies. NPM automatically installs all the dependencies mentioned here in the node_module folder of the package.
- repository - repository type and URL of the package
- main - entry point of the package
- keywords - keywords

Chapter 6.
CALLBACK CONCEPT

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 the APIs of Node are written in such a way that they support callbacks.

Blocking Code Example

fs.readFileSync

This command blocks until it reads the file and then it proceeds to the rest of the program

In case a program needs to use any data to be processed, it should be kept within the same block to make it sequential execution.

Non-Blocking Code Example

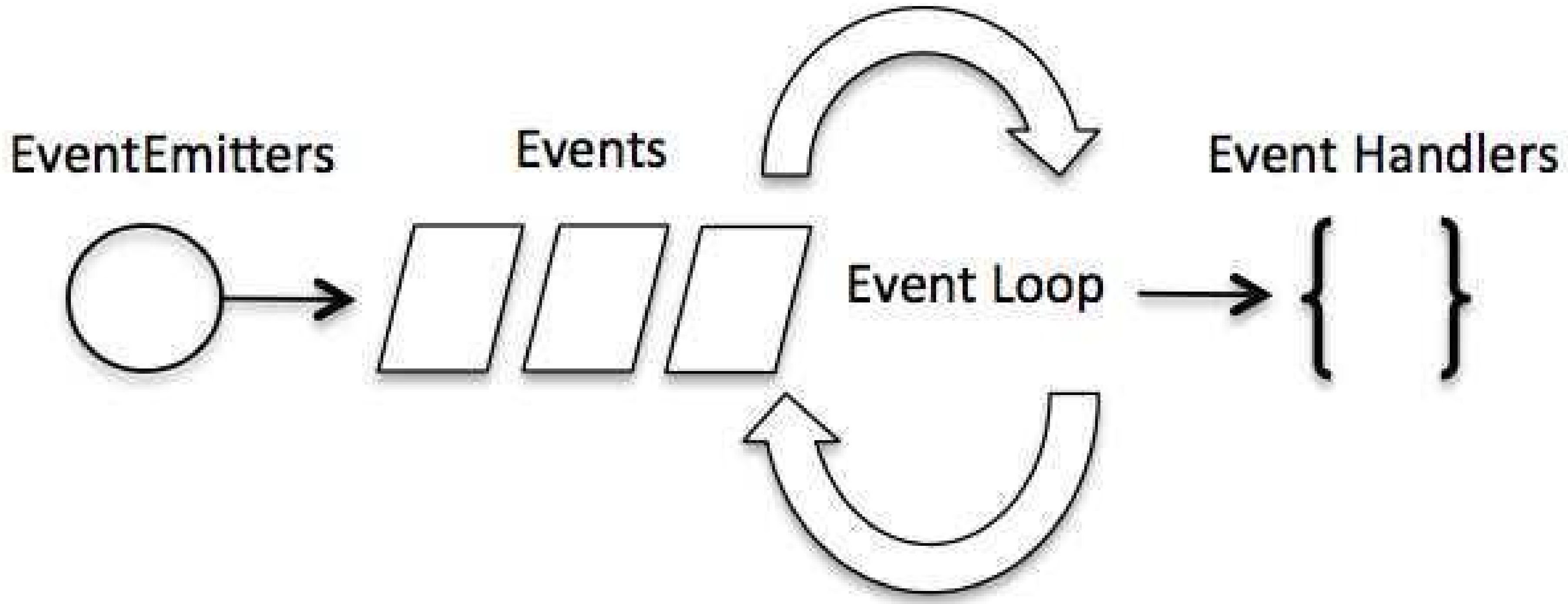
fs.readFile

This command does not wait for file reading to complete, it proceeds to print "Program Ended" and at the same time, the program without blocking continues reading the file

Chapter 7.
EVENT LOOP

Node.js is a single-threaded application, but it can support concurrency via the concept of event and callbacks. Every API of Node.js is asynchronous and being single-threaded, they use async function calls to maintain concurrency. Node uses the observer pattern. Node thread keeps an event loop and whenever a task gets completed, it fires the corresponding event which signals the event-listener function to execute.

Event-Driven Programming



Event ≠ Callback

- Events look like callbacks but they are different
- Callback functions are called when an asynchronous function returns its result
- Event handling functions listen for events. Whenever an event gets fired, its listener function starts executing

Callback mechanism

async functions accept a callback as the last parameter and a callback function accepts an error as the first parameter

```
fs.readFile('input.txt', function (err, data) {
  if (err){
    console.log(err.stack);
    return;
    }
    console.log(data.toString());
  });
```

Chapter 8.
EVENT EMITTER

- EventEmitter class is part of the events module
- EventEmitter class can be assigned to any object
- Any object can
  - trigger events using eventEmitter.emit(anEvent, dataOptional)
  - Assign one or more event handlers to a specific event eventEmitter.on(anEvent, eventHandler)
- EventEmitter can also emit an 'error' event

Chapter 9. BUFFERS

- Allows storing raw data such as binary from TCP stream or file system requests
- Buffer is a global class
- Storage outside of V8 heap

Create Buffer examples

- var buf = new Buffer(10);
- var buf = new Buffer([10, 20, 30, 40, 50]);
- var buf = new Buffer("Simply Easy Learning", "utf-8");

Write and Read Buffer examples

- Write
  - len = buf.write("Simply Easy Learning");
- Read
  - buf.toString('ascii');

Other Buffer examples

- JSON
  - var json = buf.toJSON(buf);
- Concatenate
  - var buffer3 = Buffer.concat([buffer1,buffer2]);
- Compare
  - var result = buffer1.compare(buffer2);
- Copy
  - buffer1.copy(buffer2);
- Slice
  - var buffer2 = buffer1.slice(0,9);

Chapter 10.
STREAMS

- Streams are objects that let you read data from a source or write data to a destination in continuous fashion.
- Streams are instances of EventMitter

Stream Types

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

Stream operations

- Read
- Write
- Pipe
- Chain

Common Streams

- 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 the data has been flushed to underlying system.

Chapter 11.
FILE SYSTEM

- Used to interact with the operating system
- Methods of the file system class can be either
  - Synchronous or
  - Asynchronous
- Command line utility to
  - install node.js packages,
  - do version management
  - and dependency management of node.js packages

File system methods

- fs.open(path, flags[, mode], callback)
  - open file
- fs.stat(path, callback)
  - get file information
- fs.writeFile(filename, data[, options], callback)
  - write file
- fs.read(fd, buffer, offset, length, position, callback)
  - read file
- fs.close(fd, callback)
  - close file

File system methods continued

- fs.ftruncate(fd, len, callback)
  - truncate
- fs.unlink(path, callback)
  - delete
- fs.mkdir(path[, mode], callback)
  - make directory
- fs.readdir(path, callback)
  - read directory
- fs.rmdir(path, callback)
  - remove directory

Chapter 12.
GLOBAL OBJECTS

- Global objects are global to the entire application
- Available in all modules and do not need to be defined
- Use these globals directly
- They include:
  - modules
  - functions
  - strings
  - Object

Global object examples

- __filename
- __dirname
- setTimeout(cb, ms)
- clearTimeout (t)
- setInterval(cb, ms)
- console
- process

Chapter 13.
UTILITY MODULES

There are several utility modules available in Node.js module library. These modules are very common and are frequently used while developing any Node-based application.

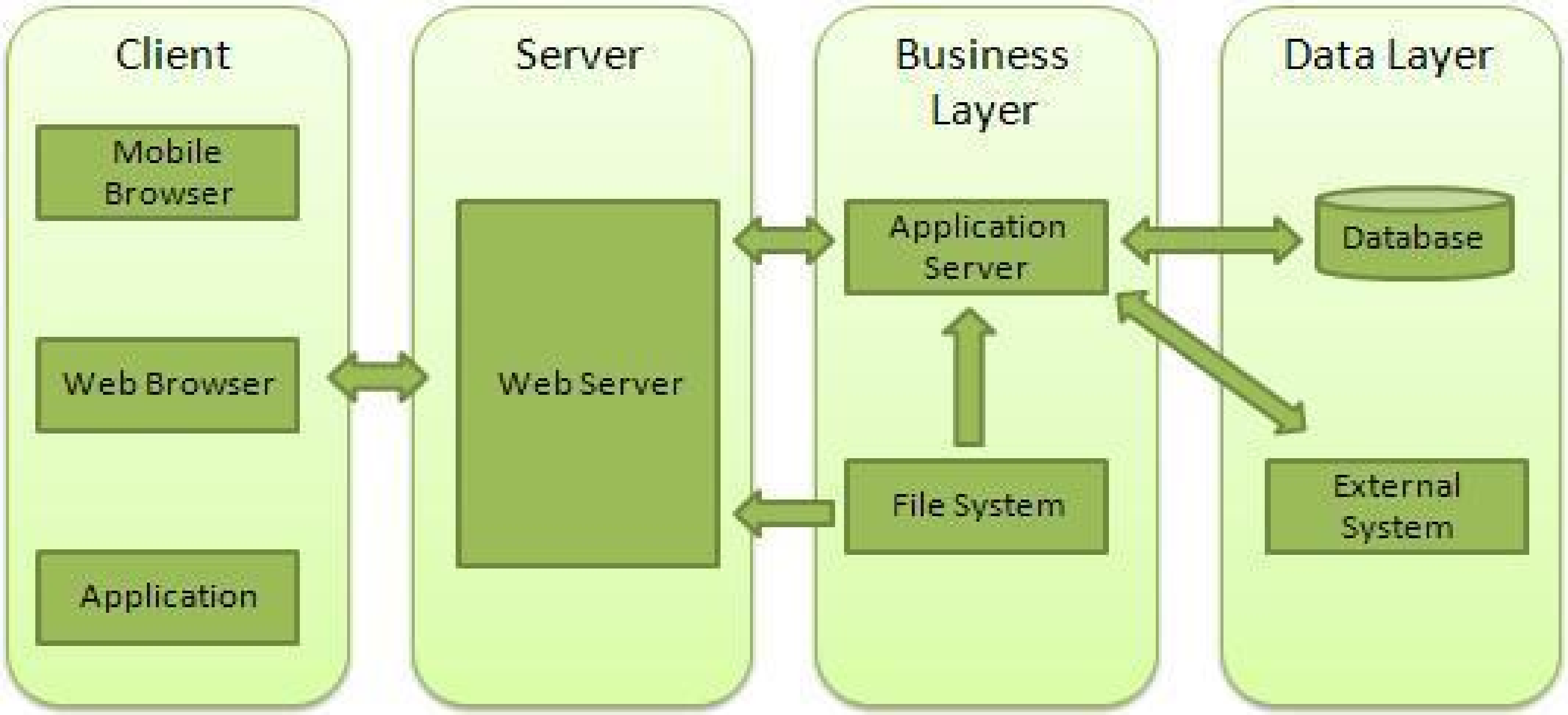
Module Name & Description

- OS Module
  - Provides basic operating-system related utility functions.
- Path Module
  - Provides utilities for handling and transforming file paths.
- Net Module
  - Provides both servers and clients as streams. Acts as a network wrapper.
- DNS Module
  - Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.
- Domain Module
  - Provides ways to handle multiple different I/O operations as a singlegroup

Chapter 14.
WEB MODULE

A 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. Web servers usually deliver html documents along with images, style sheets, and scripts.

Web Application Architecture



Web Application Architecture

- Client - This layer consists of web browsers, mobile browsers or applications which can make HTTP requests to the web server.
- Server - This layer has the Web server which can intercept the requests made by the clients and pass them the response.
- Business - This layer contains the application server which is utilized by the web server to do the required processing. This layer interacts with the data layer via the database or some external programs.
- Data - This layer contains the databases or any other source of data.

Creating a Web Server using Node

PAGE 118

Creating a Web client using Node

PAGE 120

Chapter 15.
EXPRESS FRAMEWORK

- Express is a minimal and flexible Node.js web application framework
- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.
- https://expressjs.com

Express Framework Overview

- Installation
- Request & Response
- Request Object
- Response Object
- Basic Routing
- Serving Static Files
- GET Method
- POST Method
- File Upload
- Cookies Management

Chapter 16.
RESTful API

- REST stands for REpresentational State Transfer. REST is a web standard based architecture that uses HTTP Protocol. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.
- A REST Server simply provides access to resources and a REST client accesses and modifies the resources using HTTP protocol. Here each resource is identified by URIs/ global IDs. REST uses various representation to represent a resource, for example, text, JSON, XML, but JSON is the most popular one.

HTTP Methods

- GET
  - This is used to provide a read-only access to a resource.
- PUT
  - This is used to create a new resource.
- DELETE
  - This is used to remove a resource.
- POST
  - This is used to update an existing resource or create a new resource.

Chapter 17.
SCALING AN APPLICATION

Node.js runs in a single-thread mode, but it uses an event-driven paradigm to handle concurrency. It also facilitates creation of child processes to leverage parallel processing on multi-core CPU based systems.

child_process module

- exec
  - child_process.exec method runs a command in a shell/console and buffers the output
- spawn
  - child_process.spawn launches a new process with a given command.
- fork
  - The child_process.fork method is a special case of the spawn() to create child processes.fork

cluster module

- A single instance of Node.js runs in a single thread. To take advantage of multi-core systems the user will sometimes want to launch a cluster of Node.js processes to handle the load
- cluster
  - The cluster module allows easy creation of child processes that all share server ports

Chapter 18. PACKAGING

THIS SECTION IS NOW OUT OF DATE

Chapter 16.
RESTful API

- Provides searchable repositories for node.js packages/modules hosted on

  https://www.npmjs.com
- Command line utility to
  - install node.js packages,
  - do version management
  - and dependency management of node.js packages