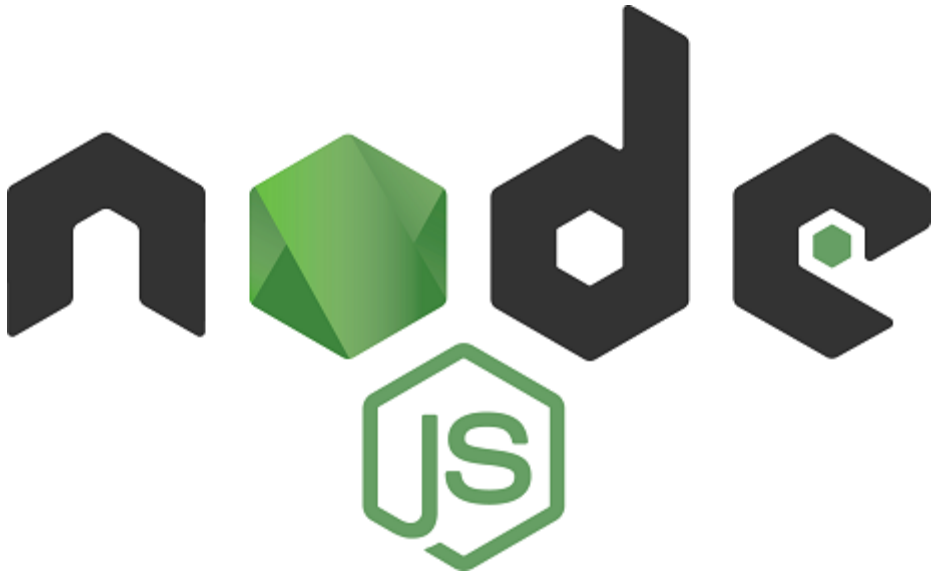

node
JS

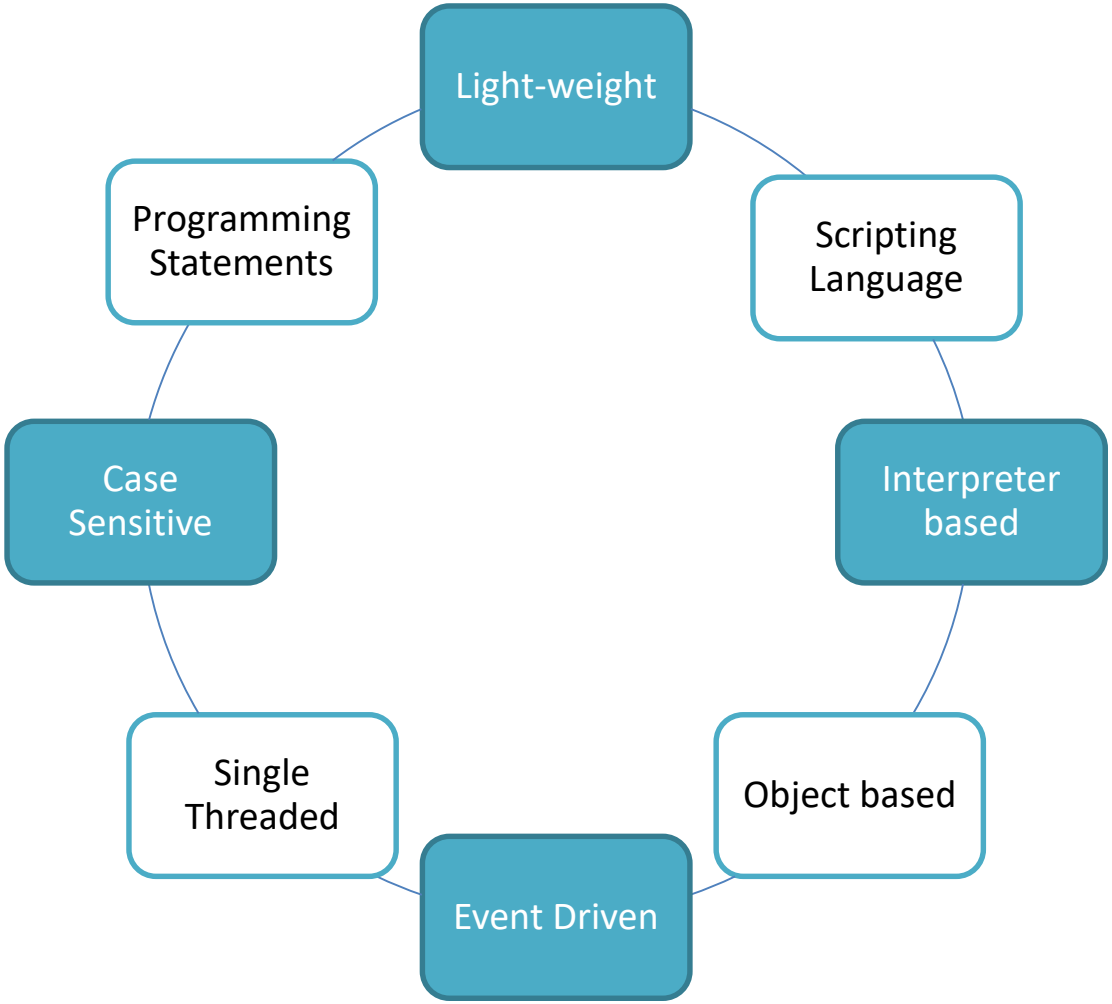
Training Agenda

- NodeJS Overview
- Modules System
- File System
- Buffers & Streams
- Event System
- Express
- Testing
- Data Persistence
- View Engines

Prerequisites:

- ✓ Basic knowledge of HTML, CSS & JavaScript
- ✓ Interest to Learn

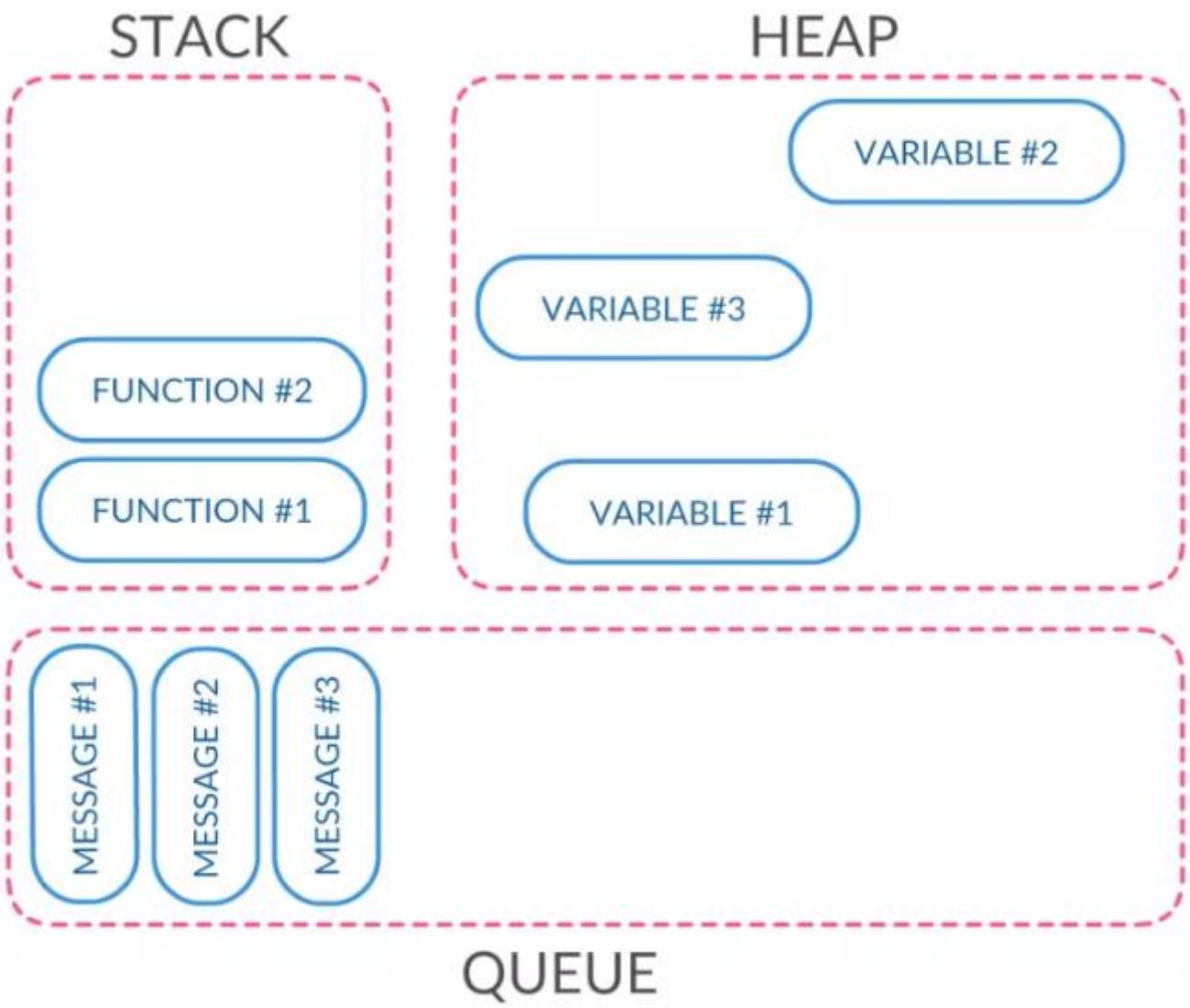
JavaScript Overview



Heap, Stack & Queue

- Stack :
  - The *stack* is where the runtime stores the functions that will be called. It has a LIFO structure: the last instruction added to the stack is the first one executed.
- Heap :
  - The *heap* is where the runtime saves all the variables and constants. It is an unstructured piece of memory.
- Queue :
  - Runtime saves messages in a queue. They are processed when the stack is empty. The queue is a FIFO data structure: items are processed in chronological order, taking out the first item pushed.

Event Loop


STACK
HEAP
VARIABLE #2
VARIABLE #3
FUNCTION #2
FUNCTION #1
VARIABLE #1
MESSAGE #1
MESSAGE #2
MESSAGE #3
QUEUE

NodeJS

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

-nodejs.org

NodeJS : Features

- Extremely fast
- I/O is Asynchronous
- Event Driven
- Single threaded
- Highly Scalable
- No buffering
- Open source

NodeJS Process Model

Node.js runs in a single process and the application code runs in a single thread.

All the user requests to web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request.

An event loop is constantly watching for the events to be raised for an asynchronous job and executing callback function when the job completes.

Internally, Node.js uses *libev* for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O.

Module System

Use of imports/exports for importing and exporting the modules in application.

Module System Features	Each file is its own module.
	Each file has access to the current module definition using the module variable.
	The export of the current module is determined by the module.exports variable.
	To import a module, use the globally available require function.

Require Function

The Node.js require function is the main way of importing a module into the current file.

The require function blocks further code execution until the module has been loaded.

There is no accidental global scope.

you can choose to call require() based on some condition and therefore load the module only if you need it.

After the first time a require call is made to a particular file, the module.exports is cached.

Module allows you to share in-memory objects between modules.

Treats module as an object factory.

Module System(Cntd…)

Assume you require('something').

- If something is a core module, return it.
- If something is a relative path (starts with ‘./’ , ‘../’) return that file OR folder.
- If not, look for node_modules/filename or node_modules/foldername each level up until you find a file OR folder that matches something.

When matching a file OR folder:,

- If it matched a file name, return it.
- If it matched a folder name and it has package.json with main, return that file.
- If it matched a folder name and it has an index file, return it.

Important Globals

Console	the console plays an important part in quickly showing what is happening in your application when you need to debug it.
Timers	setTimeout only executes the callback function once after the specified duration. But setInterval calls the callback repeatedly after every passing of the specified duration.
__filename and __dirname	These variables are available in each file and give you the full path to the file and directory for the current module.
Process	Use the process object to access the command line arguments. Used to put the callback into the next cycle of the Node.js event loop.
Buffer	Native and fast support to handle binary data.
Global	The variable global is our handle to the global namespace in Node

Node Package Manager (NPM)

npm install	Install the packages/ dependencies
npm uninstall	Uninstall the packages/dependencies
npm config get/set	Gest /set the npm eco-system
npm update	Update the project dependencies
npm ls	List down the dependencies
npm search	Search the listed package on npm registry
npm init	Generates package.json file in local project directory
npm outdated	List down the outdated package

Event System

EventEmitter is a class designed to make it easy to emit events (no surprise there) and subscribe to raised events.

Subscribing : *built-in support* for multiple subscribers is one of the advantages of using events.

Unsubscribing : EventEmitter has a removeListener function that takes an event name followed by a function object to remove from the listening queue.

EventEmitter provides a function `**once**` that calls the registered listener only once.

EventEmitter has a member function, **listeners**, that takes an event name and returns all the listeners subscribed to that event.

Memory Leak : A common cause of memory leak is forgetting to unsubscribe for the event.

Creating your own EventEmitter (Custom Events)

Buffers & Streams

Streams play an important role in creating performant web applications.

Improvement in user experience and better utilization of server resources is the main motivation behind steams.

All of the stream classes inherit from a base abstract Stream class which in turn inherits from EventEmitter.

All the streams support a pipe operation that can be done using the pipe member function.

Streams (Cntd…)

Readable	A readable stream is one that you can read data from but not write to.
	A good example of this is process. stdin, which can be used to stream data from the standard input.
Writable	A writable stream is one that you can write to but not read from.
	A good example is process.stdout, which can be used to stream data to the standard output.
Duplex	A duplex stream is one that you can both read from and write to.
	A good example of this is the network socket. You can write data to the network socket as well as read data from it.
Transform	A transform stream is a special case of a duplex stream where the output of the stream is in some way computed from the input.
	A good example of these is encryption and compression streams.

Express

Web application framework for Node apps.

To create an express app, make a call require('express')

Express can accept middleware using the 'use' function which can be registered with http.createServer.

Express Request / Response objects are derived from standard NodeJS http Request / Response

Request object can handle URL : Route Parameter & Querystrings

Express Router is used to mount middlewares and access all REST APIs

Cookies

- A *cookie* is some data sent from the web server and stored in the user's web browser.
- Cookies provide a great foundation for creating user sessions.
- Cookie-parser populates the parsed cookies into the 'req.cookies' object

```
npm install cookie-parser
```

- Prevent cookie modification by user using signing.
- Cookie-session used to maintain the session of user.

```
npm install cookie-session
```

Data Persistence

- Why NoSQL ?
  - Scalability
  - Ease of Development
- NoSQL servers can be placed into four broad categories:
  - Document databases (for example, MongoDB)
  - Key-value databases (for example, Redis)
  - Column-family databases (for example, Cassandra)
  - Graph databases (for example, Neo4J)

MongoDB

- A MongoDB deployment consists of multiple databases.
- Each database can contain multiple collections.
  - A *collection* is simply a name that you give to a *collection of documents*.
- Each collection can contain multiple documents.
  - A document is effectively a JSON document

```
npm install mongodb
```

CRUD Operation

- db.collection.insertOne()
- db.collection.insertMany()
- db.collection.find()
- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()
- db.collection.deleteOne()
- db.collection.deleteMany()

Mongoose (ODM)

- MongoDB works with what are effectively simple JSON documents. This means that the business logic that operates on the documents (functions/methods) must live elsewhere.
- Using an Object Document Mapper (ODM) we can map these simple documents into full-form JavaScript objects (with data + methods for validation and other business logic).

```
npm install mongoose
```

View Engines

Jade	uses whitespace and indentation as a part of the syntax.
Handlebar	developers *can't write* a lot of JavaScript logic inside the templates.
EJS	Follows JavaScript-ish syntax. Embed JavaScript code in template. Commonly used.
Vash	Vash is a template view engine that uses Razor Syntax.Look familiar to people who have experience in ASP.Net MVC.

Testing

- Node.js comes with assert as a core module, provide you with simple logic checks to throw errors based on invalid situations.
- Mocha is a feature-rich JavaScript test framework running on node.js and the browser, making asynchronous testing simple and fun.

```
npm install -g mocha
```

Mocha API :

Functions	Description
before	executes the registered callback before it runs the first test
after	executes the callback after it runs the last test
beforeEach	executes the callback before executing *each* test
afterEach	executes the callback before executing *each* test

Testing (Cntd…)

- Chai is a library that provides you with additional assertion functions.

```
npm install chai
```

- Chai : good test follow the AAA pattern –
  - **Arrange** all the necessary preconditions and inputs.
  - **Act** using the object / method under test.
  - **Assert** that the expected results have occurred.
- Functions provided by chai mostly focus on stronger type checks (isDefined/isUndefined, isNull / isNotNull, isFunction, isNotFunction)
- Chai also provides BDD (behavior-driven development) style assertions.
  - These are easily chainable.
  - These read in better *English*, so the test code is closer to the **behavior** that you are trying to test.

Istanbul

- Istanbul instruments your ES5 and ES2015+ JavaScript code with line counters, so that you can track how well your unit-tests exercise your codebase.

- The *nyc* command-line-client for Istanbul works well with most JavaScript testing frameworks: tap, mocha, AVA, etc.

```
npm install nyc
```

Debugging

Console Object

Debugger statement

Node-inspect

References

Books :

- Beginning NodeJS by Basarat Ali Syed
- Node.JS Web Development by David Herron

Web :

- https://nodejs.org/en/docs
- http://www.youtube.com
- https://stackoverflow.com