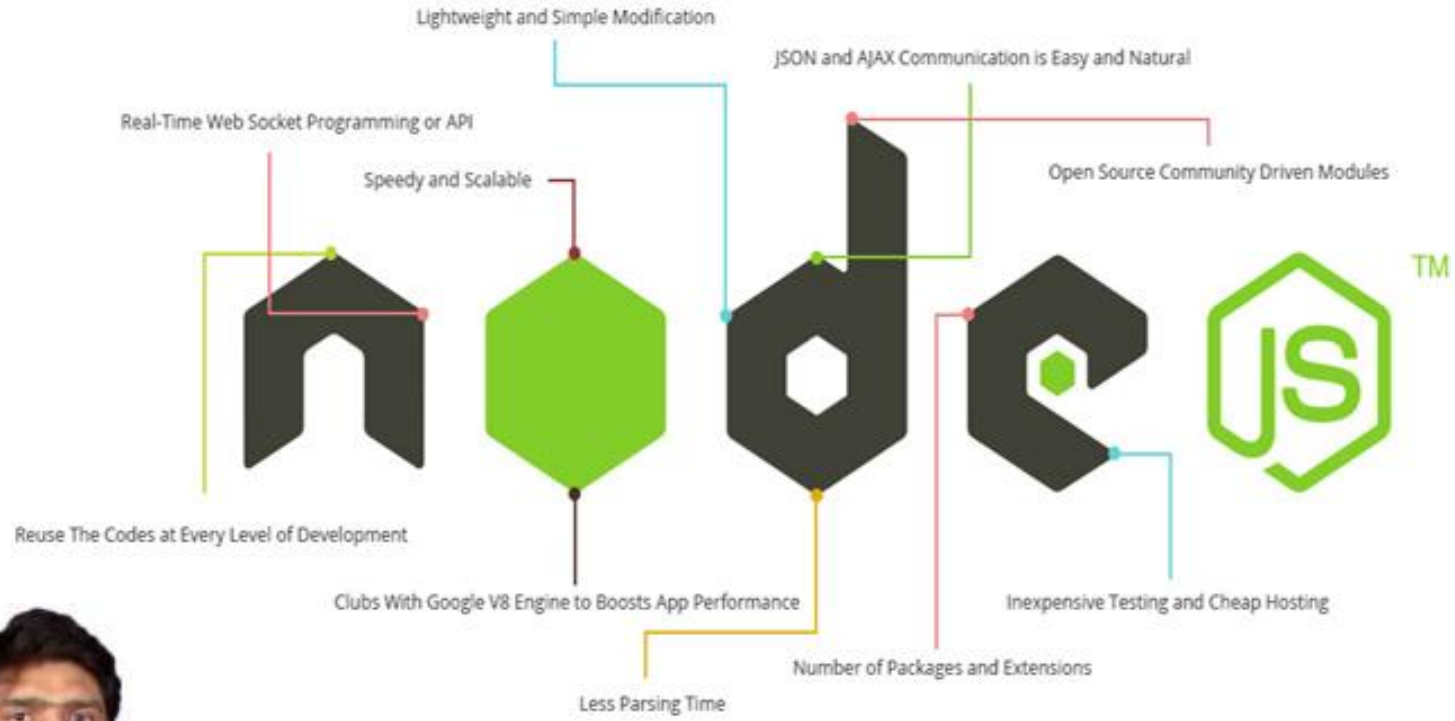


Syed Awase Khirni

RESEARCHER | ENTREPRENEUR | TECHNOLOGY COACH

@sak008 | sak@sycliq.com/sak@territorialprescience.com | +91. 9035433124

Syed Awase earned his PhD from University of Zurich in GIS, supported by EU V Framework Scholarship from SPIRIT Project (www.geo-spirit.org). He currently provides consulting services through his startup www.territorialprescience.com and www.sycliq.com . He empowers the ecosystem by sharing his technical skills worldwide, since 2008.

Terms of Use

- You shall not circulate these slides without written permission from **Territorial Prescience Research I Pvt ltd.**
- If you use any material, graphics or code or notes from these slides, you shall seek written permission from TPRI and acknowledge the author Dr. Syed Awase Khirni
- If you have not received this material, post-training session, you shall destroy it immediately and not use it for unauthorized usage of the material. If any of the material, that has been shared is further used for any unauthorized training by the recipient, he shall be liable to be prosecuted for the damages. Any supporting material that has been provided by the author, shall not be use directly or indirectly without permission.
- If this material, has been shared to any organization prior to the training and the organization does not award the contract to TPRI, it should not use the training material internally. If by any chance, the organization is using this training material without written permission, the organization is liable to pay for the damages to TPRI and is subjected to legal action, jurisdiction being Bangalore. It shall also pay for any expenses, legal, recovery and all applicable damages and costs incurred by TPRI.
- TPRI has right to claim damages ranging from USD 50000 to USD 10,0000 dollars as damages, for unauthorized usage.
- Any organization, which does not intend to go ahead with training or does not agree with the terms and conditions, should destroy the material from its network immediately. The burden of proof lies on the client, with whom this material has been shared.
- Recovery of the damages and all expenses incurred including legal fees will be born by the client organization/candidate/party, which has violated these terms and conditions.
- Only candidates who have attended the training session in person from Dr. Syed Awase Khirni, TPRI are entitled to hold this training material. They cannot further circulate it, or use it or morph it, or change it to provide trainings. This training material cannot be used by any other candidates other than the registered individuals for the class room based session.
- TPRI reserves all the rights to this material and code plays and right to modify them as and when it deems fit.
- If you agree with the terms and conditions, please go ahead with using the training material. Else please close and destroy the slide and inform TPRI immediately.

Slide Version Updates

Last Updated	Version	Release Date	Updated by	Code Plays Done @
	V 0.5		Syed Awase	ITC
	V 0.7	2015	Syed Awase	Harman
	V 0.8	2016	Syed Awase	JCI
	V 0.9	2016	Syed Awase	ITC
	V1.0	2017	Syed Awase	Harman

NodeJS: Introduction

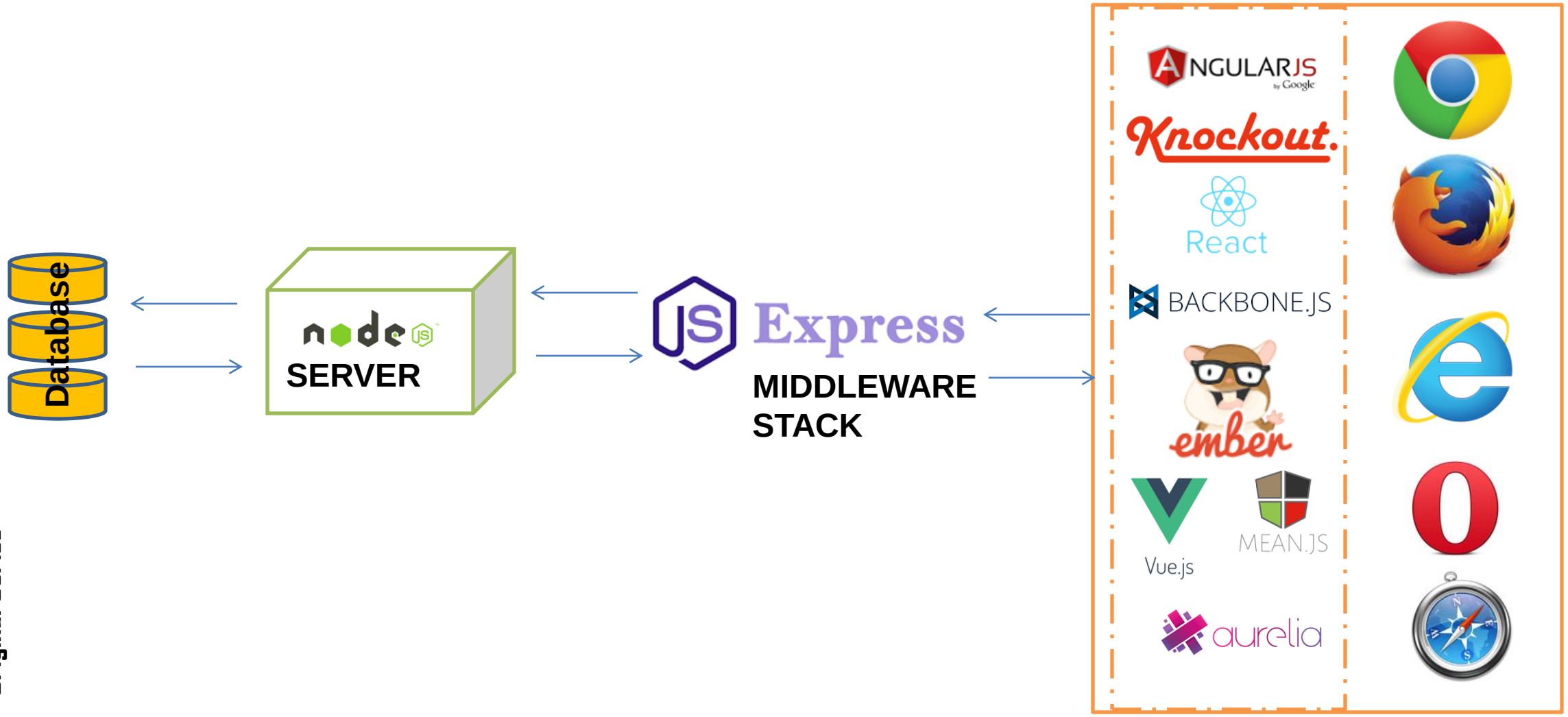
- V8 is an opensource JavaScript engine developed by Google. Written in C++ and powers Google Chrome Browser. V8 converts javascript code directly to Microprocessor instructions and uses hidden classes.
- Node.js runs on V8 Engine and created by Ryan Dahl in 2009 and maintained by Joyent.
- Similar to EventMachine in Ruby or Python's Twisted or Perl's AnyEvent Framework.
- The Reactor Pattern: **an event driven architecture**
- It is platform for building scalable, high performance networked web applications.
- Used by **Netflix, paypal and Uber**

Why Nodejs?

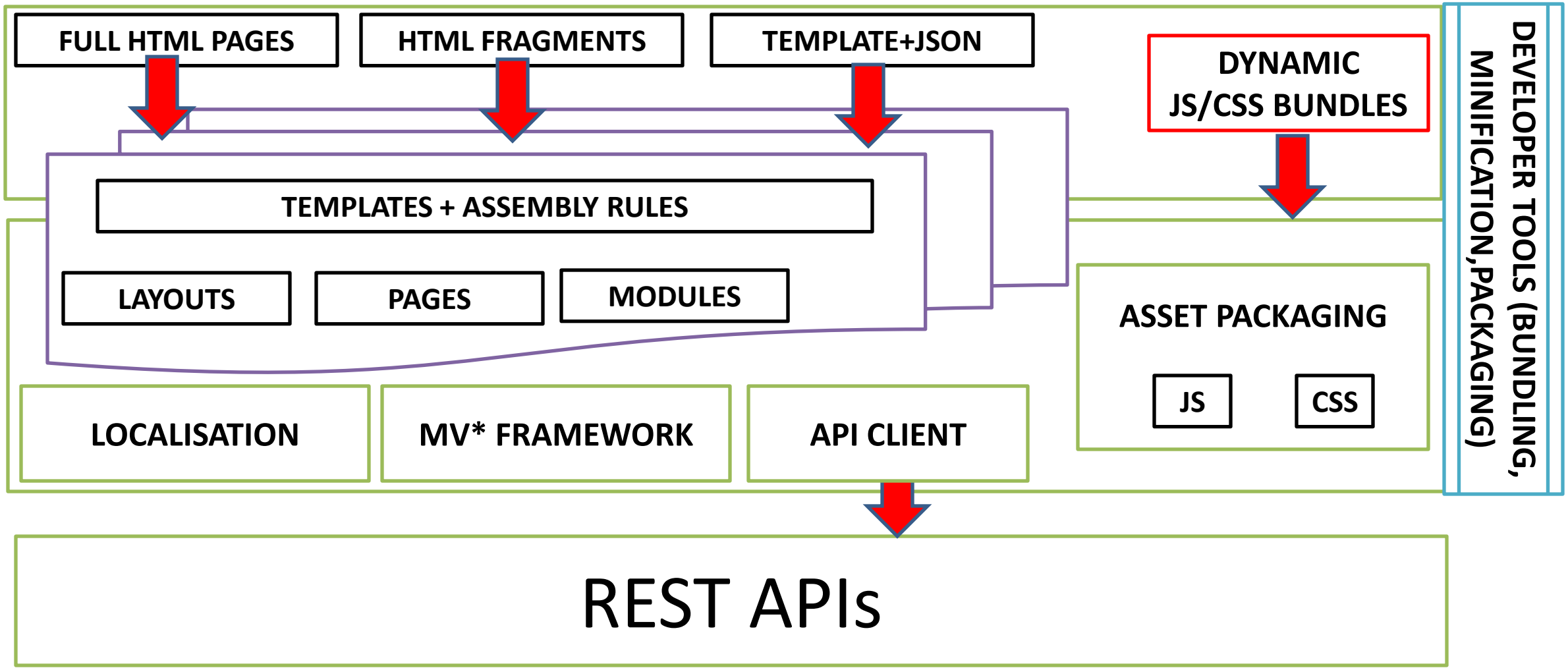
- Vast open-source community with 70k+ re-usable modules
- Reduce development time by 50% or more
- Reuse existing UX tools
- Low latency –mobile
- Event-driven and fast by design:50x faster
- Non-blocking, high throughput: helps scale 20x
- 3 million developers download every release of Node.js

Users of Node.js

ISOMETRIC JAVASCRIPT APPLICATION ARCHITECTURE



NODE.JS ARCHITECTURE (UI LAYER)



Introduction

- A server-side javascript framework for building high performance network applications which are well optimized for high concurrent environments.
- Uses an event-driven, non-blocking I/O model, which makes it light weight. It makes use of **event-loops via JavaScript's callback functionality to implement the non-blocking IO**

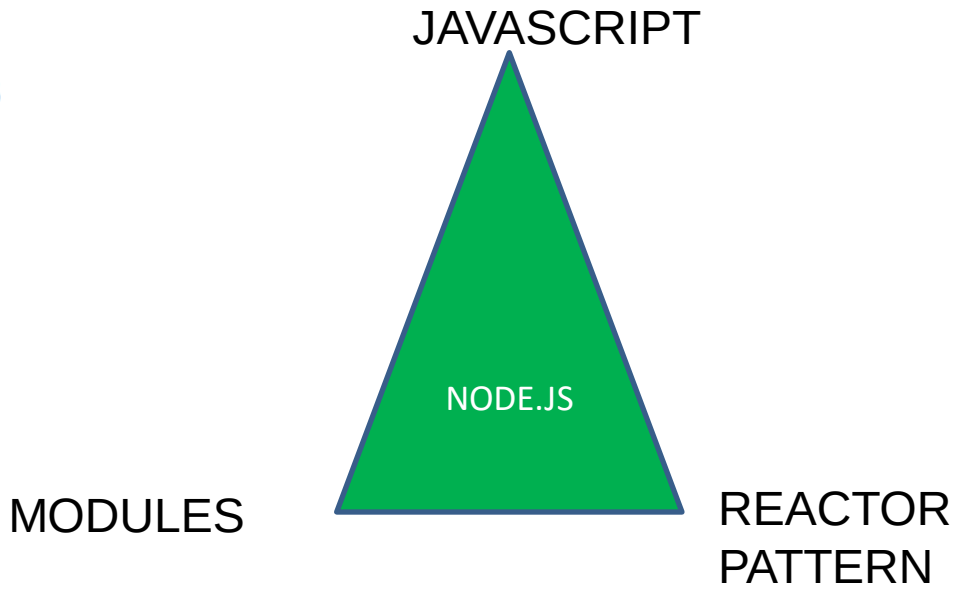
Installing NodeJS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

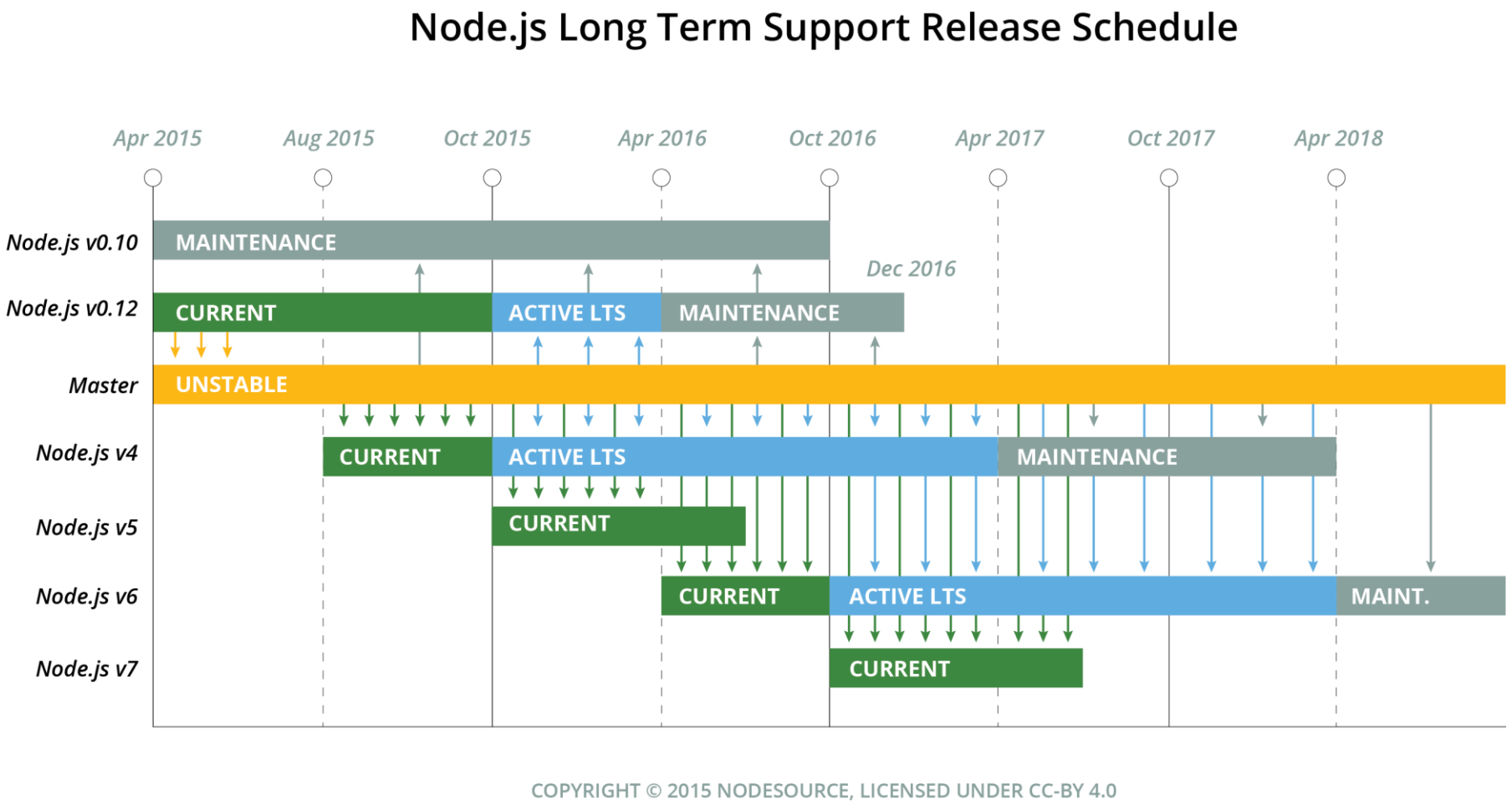
Download for Windows (x64)

v6.9.1 LTS	v7.0.0 Current
Recommended For Most Users	Latest Features
Other Downloads | Changelog | API Docs	Other Downloads | Changelog | API Docs

Or have a look at the LTS schedule.



Node.js Long Term Support Release Schedule



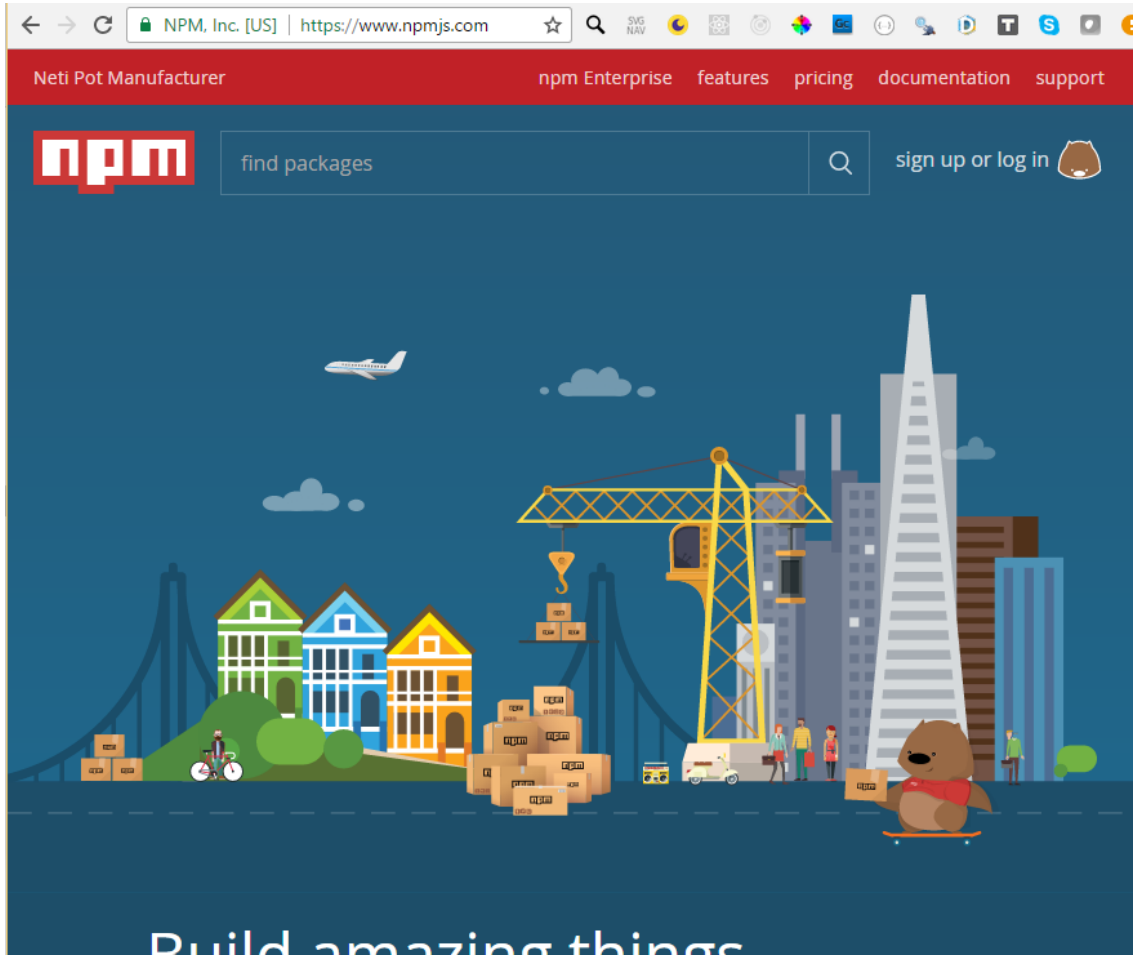
Original Series

What can I build with Node.JS?

- Allows you to build scalable network applications using JavaScript on the server-side
- Node.JS is not a web framework
- Node.JS is **not Multi-threaded**, it is a single threaded server.
- Data Streaming servers
- APIs for mobile apps
- Apps that require heavy I/O
- Websocket Server like a chat server
- FTP Server
- Ad Server
- Any Real-Time Data Apps
- MicroServices/WebAPI
- HTTPServer
- TCP Server

NODE PACKAGE MANAGER (NPM) npmjs.org

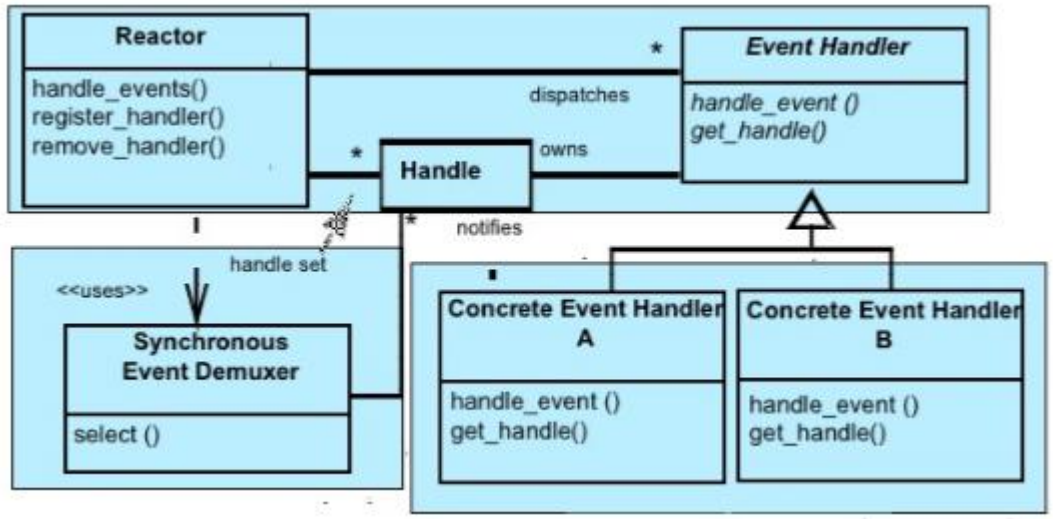
- Express
- Jade –HTML template system
- Socket.io – to create real-time apps
- Nodemon – to monitor Node.js and push change automatically
- CoffeeScript



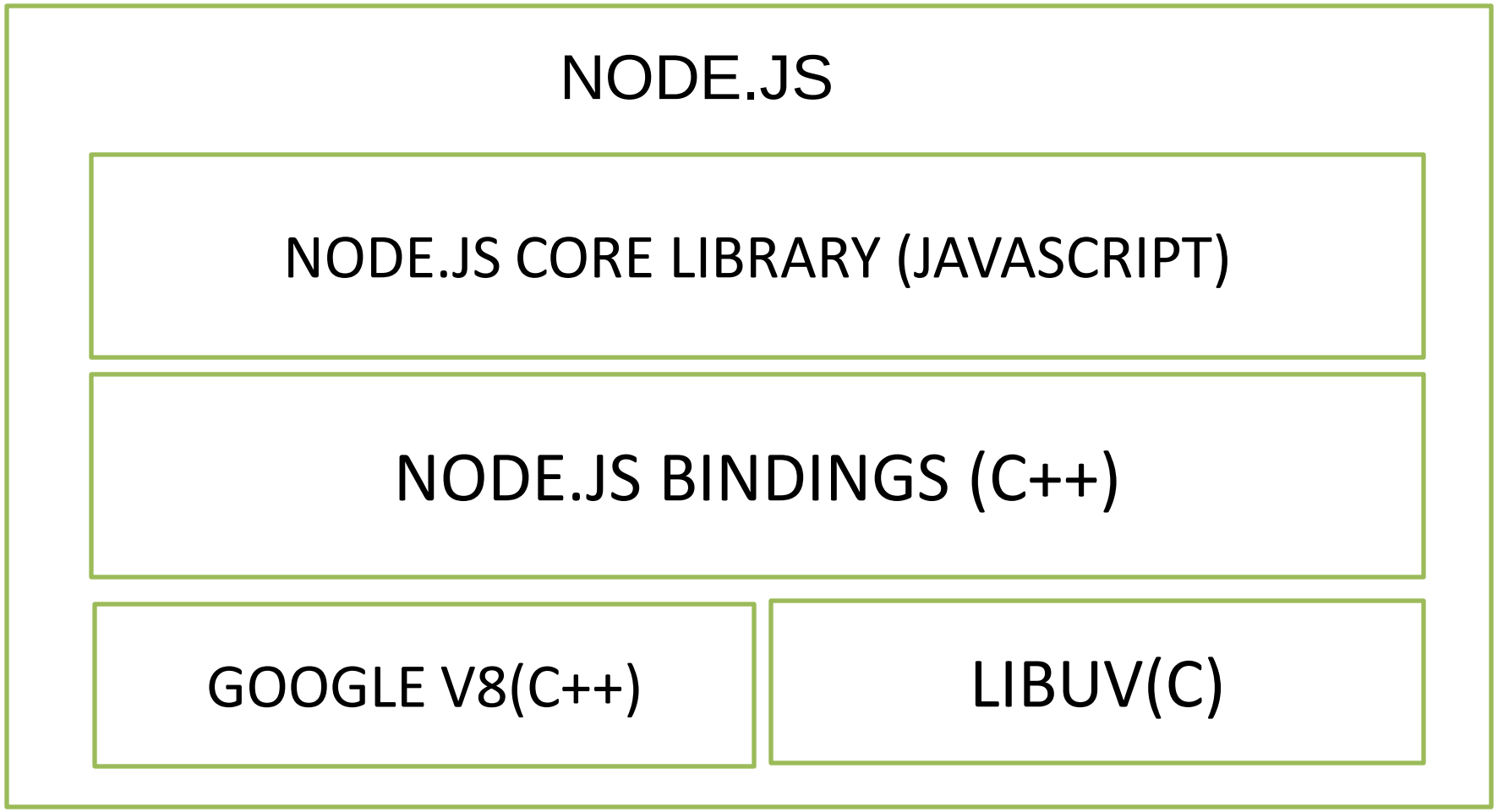
Reactor Pattern

- The reactor model follows a purely event driven system where the entire system can be implemented as a **single-threaded process with a series of event generators and event handlers.**
- **An event loop that continues to run which takes all of the generated events, sends them to all registered event handles and then starts over again**
- **Written by Doug Schmidt in 1995 for writing scalable servers. An alternative to thread-per-connection model**

Allows event-driven applications to demultiplex and dispatch service requests that are delivered to an application from one or more clients.

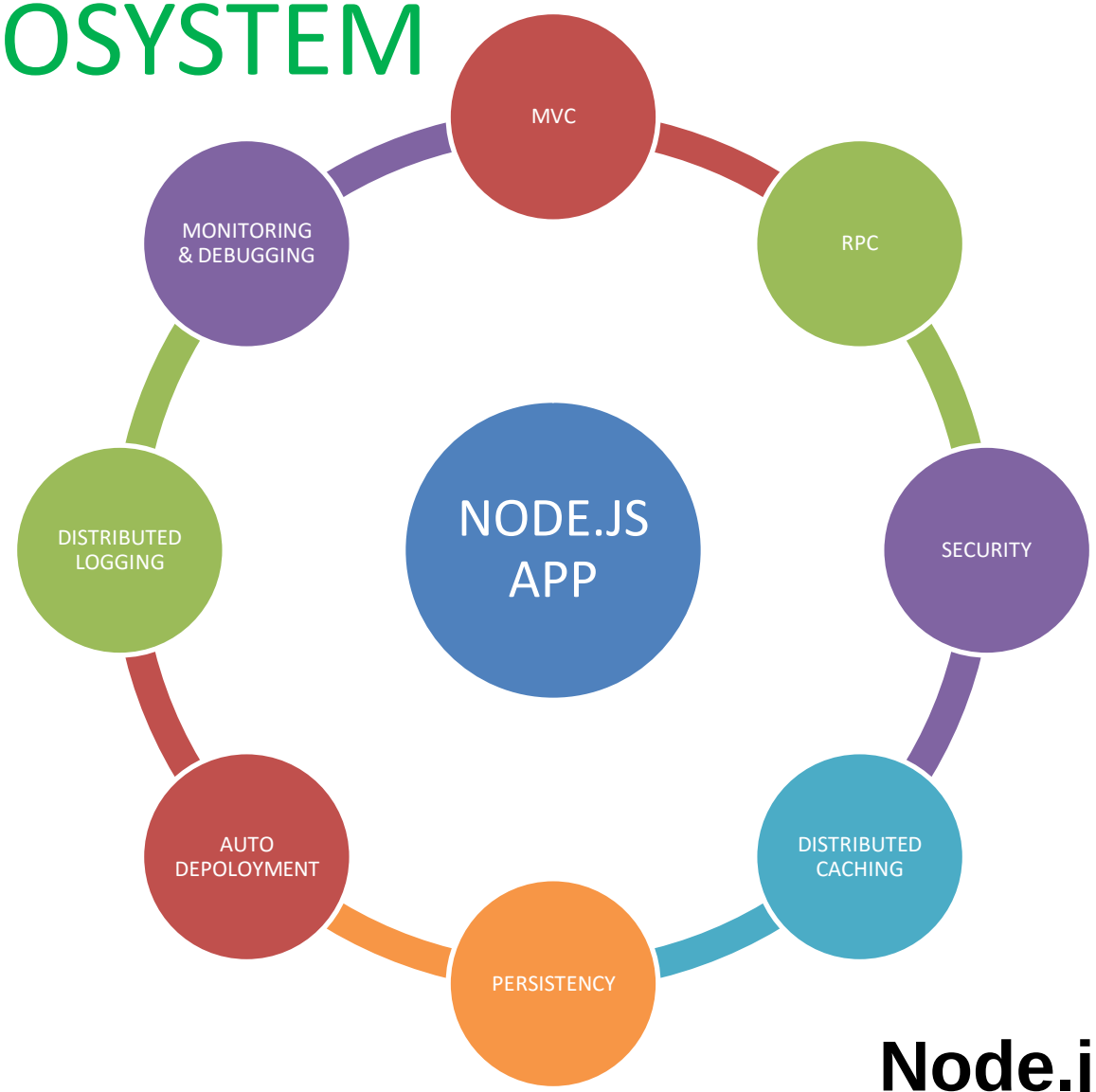


NODE.JS ARCHITECTURE



https://nodejs.org/api/index.html

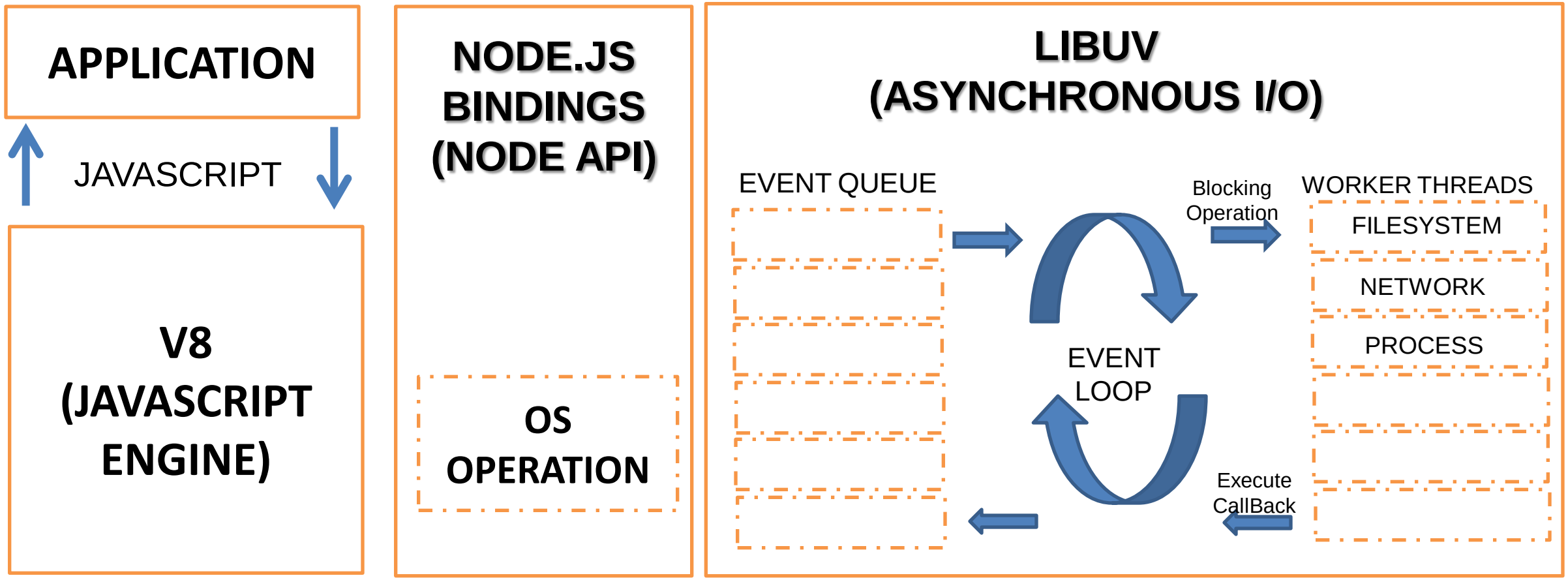
NODE.JS ECOSYSTEM



Node.js is written in C++

Node.JS

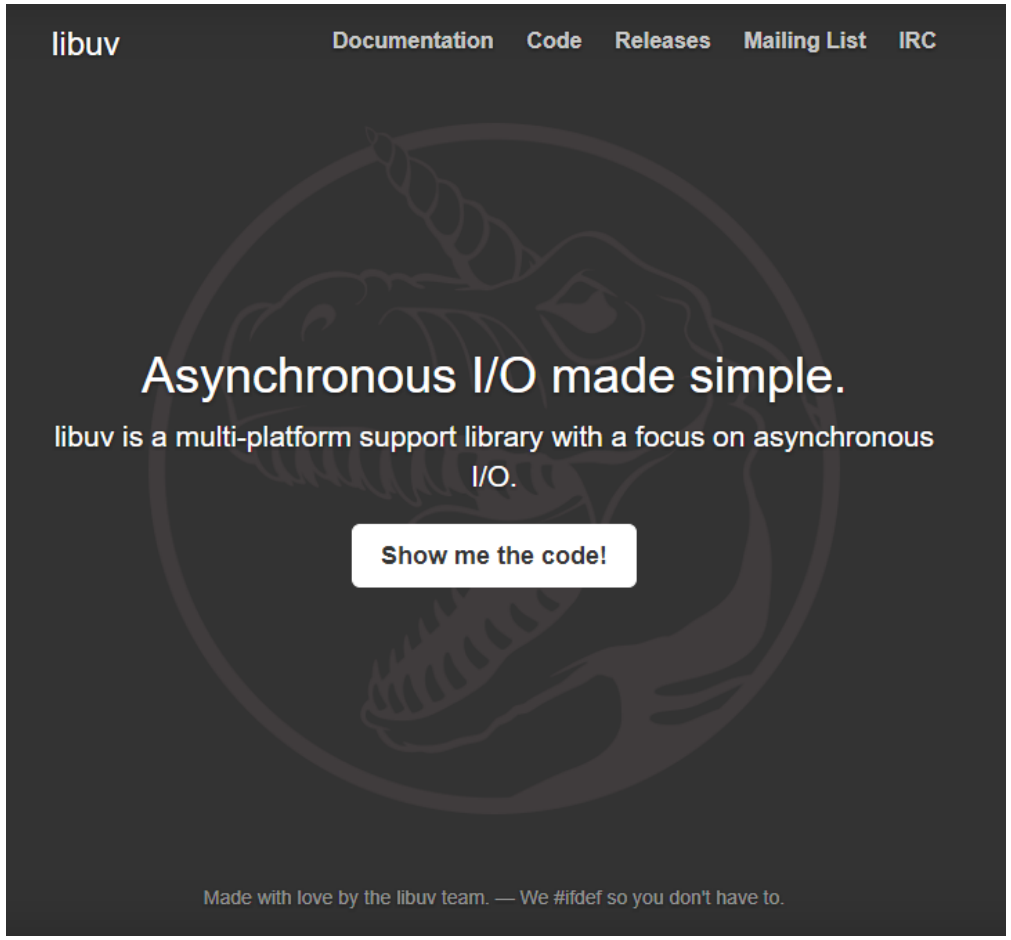
Visit: http://luajit.org/



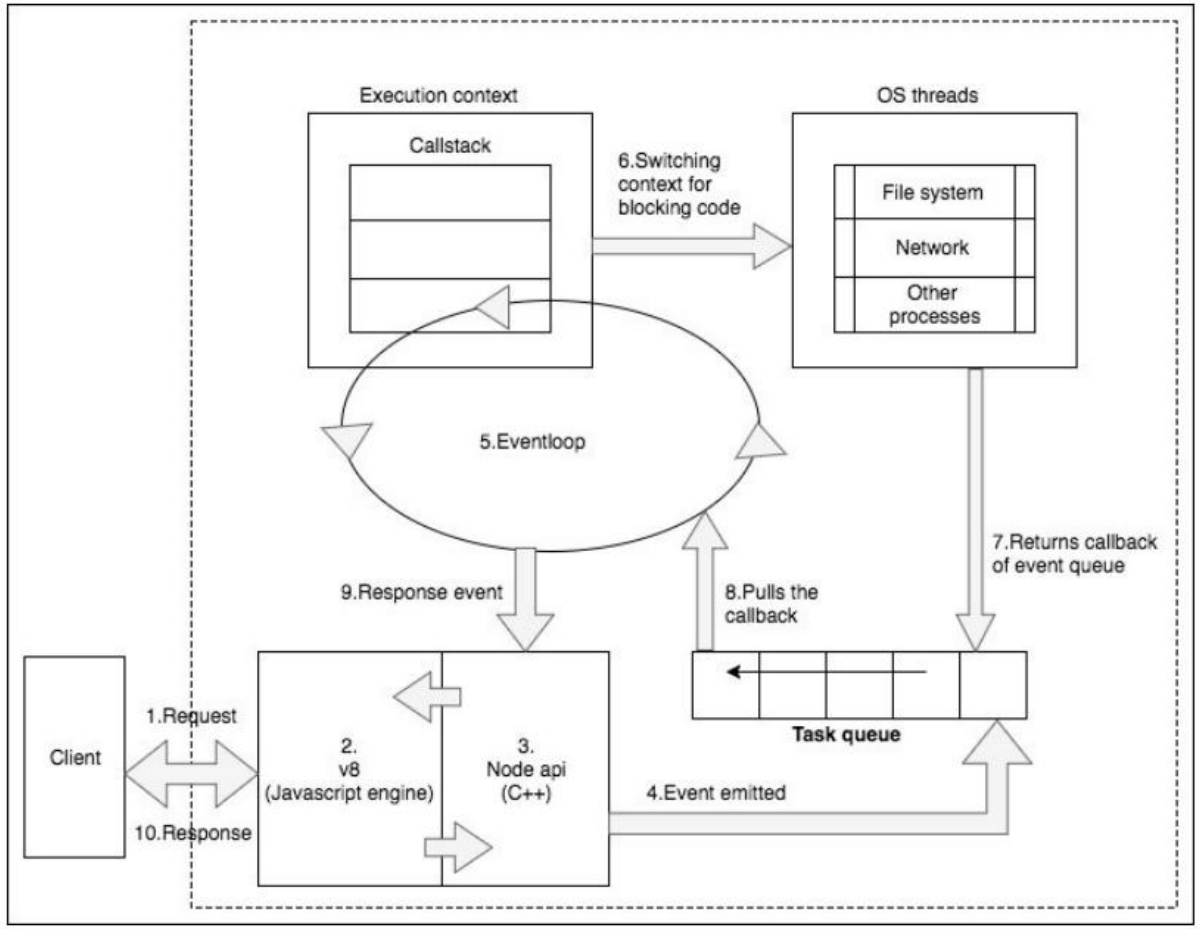
libuv is a multi-platform C library that provides support for asynchronous I/O based on event loops. It supports epoll(4) , kqueue(2) , Windows IOCP, and Solaris event ports. It is primarily designed for use in Node.js but it is also used by other software projects.

System Events	Custom Events
C++ Core LIBUV	JavaScript core Event Emitter

http://libuv.org/

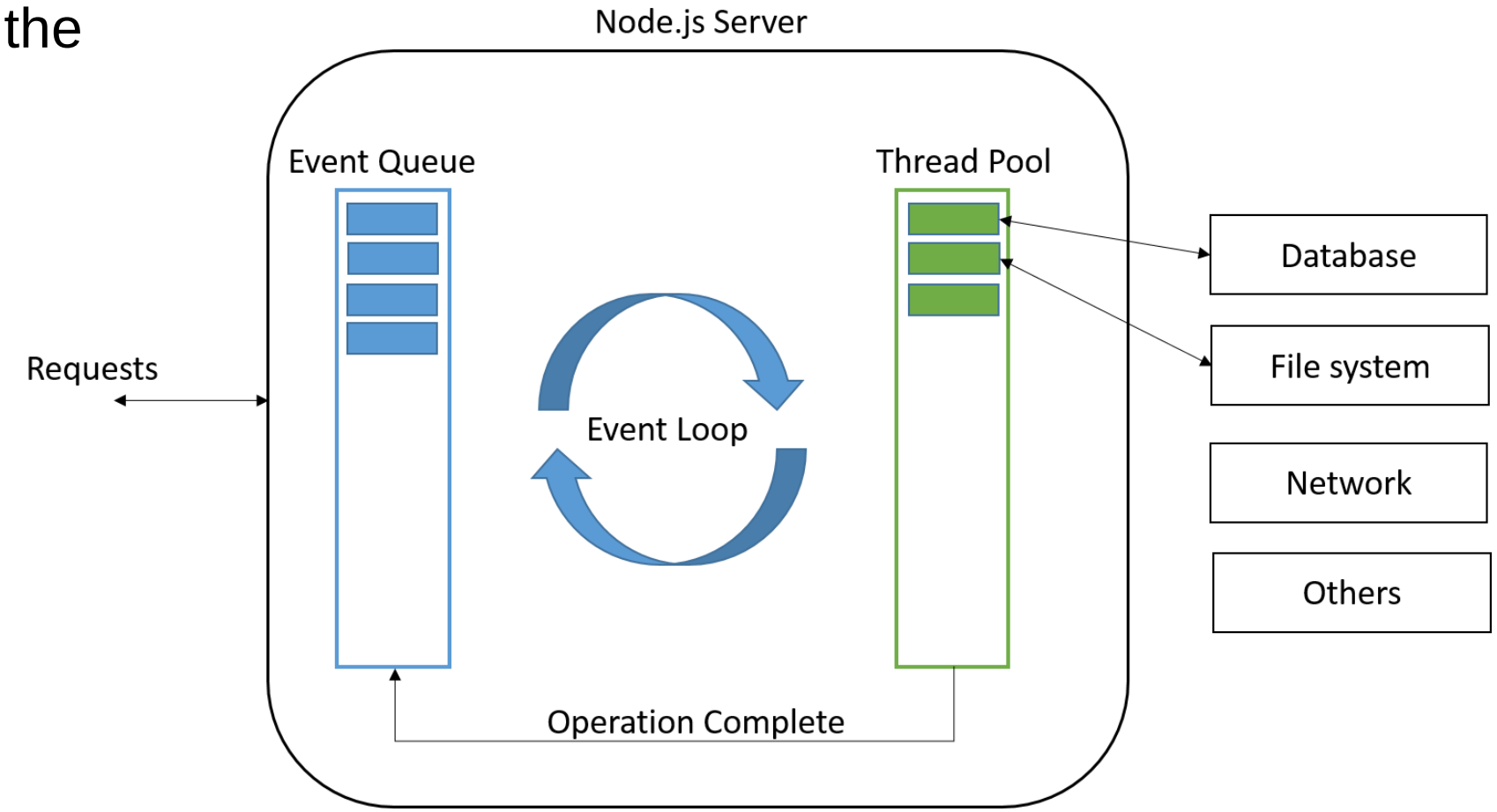


NODEJS ARCHITECTURE

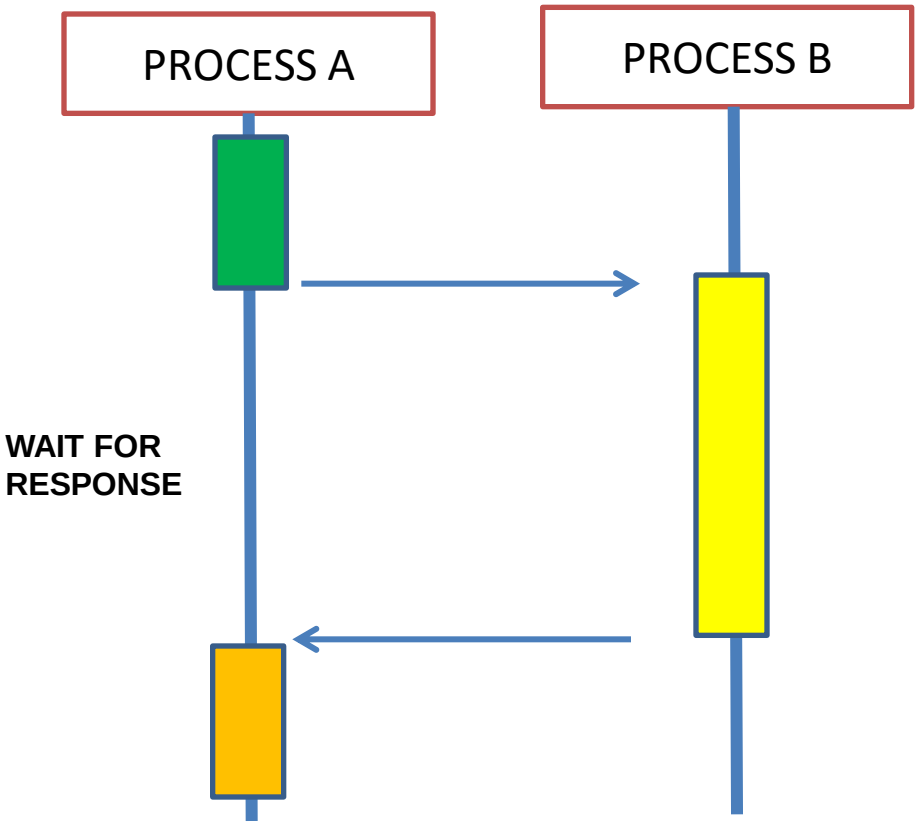


Node.js Event Loop

- Event-loops are at the core of event-driven programming

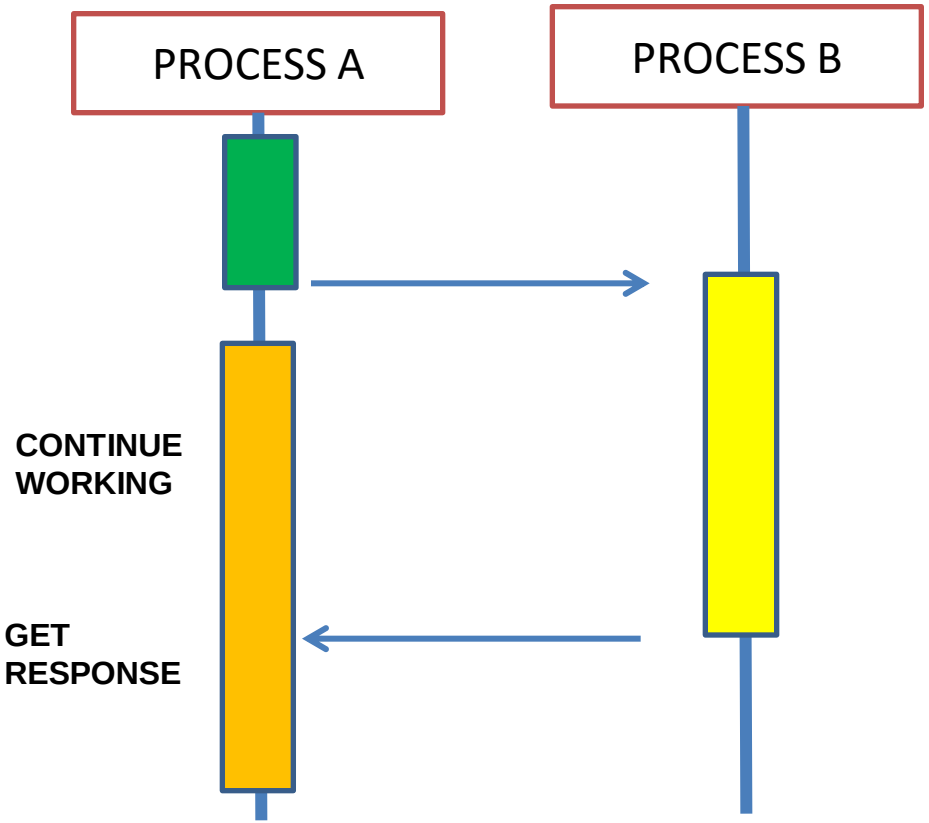


SYNCHRONOUS



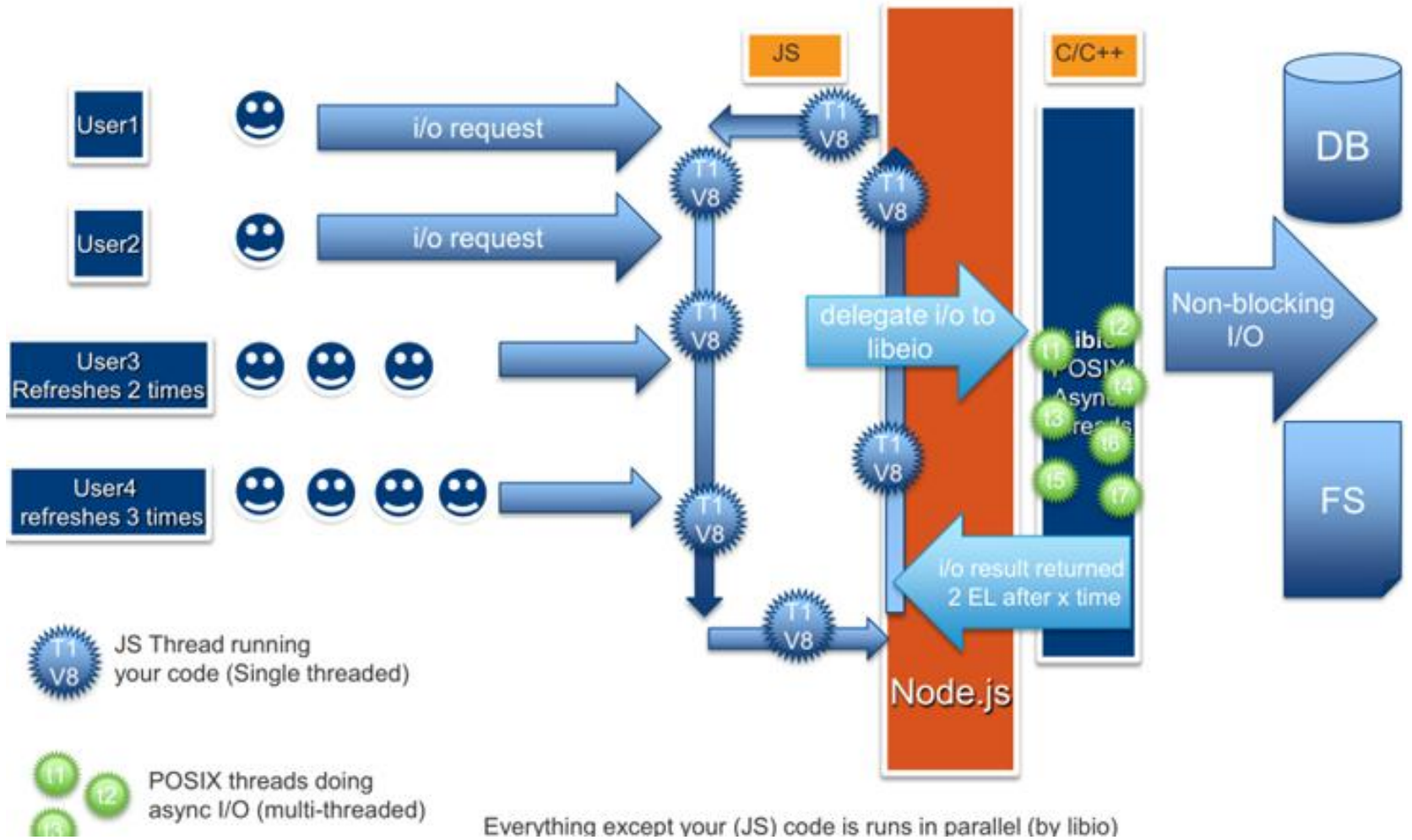
JavaScript is designed to be synchronous

ASYNCHRONOUS



NodeJS is asynchronous

Non-Blocking and Event I/O in Node.JS



Blocking I/O

```
var a = db.query('SELECT * from huge_table');
console.log('result a:', a);

console.log('Doing something else');
```

Non-Blocking I/O

```
db.query('SELECT * from huge_table', function(res) {
    console.log('result a:', res);
});

console.log('Doing something else');
```

When to use NodeJS

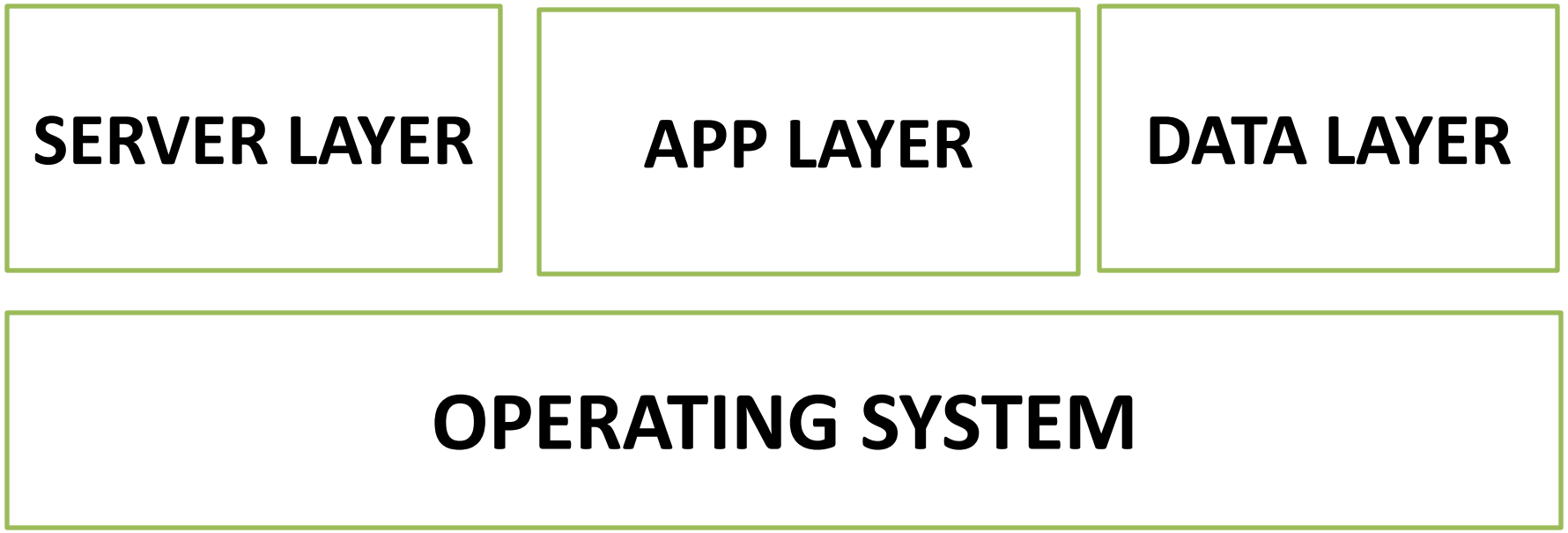
- It runs JS, same language can be used for server and client
- Single-threaded event-driven system which is fast and simple unlike Java or Ruby on Rails Frameworks, when handling multiple requests at once.
- Suitable for agile development and rapid production iterations
- Strong community
- Ability to handle thousands of concurrent connections with minimal overhead on a single process.
- Creating streaming based real-time services, web chat applications, static file servers

When NOT to use NodeJS

- Runs on JS, which **has no compile-time type checking**. For large, complex safety critical systems, or projects including collaboration between different organizations a language which encourages contractual interfaces and provides static type checking may save debugging time.
- Nested callback hell
- Open-source and raw npm packages

NodeJS STACK

We build our **own stack ground up in nodejs**



NODEJS PERFORMANCE STORY

Black Friday Sale 2013: Case Study : 200,000 K Users Online with just 1% CPU Utilization.

Doubles the number of requests per-second and reduced response time by 35% or 200 milliseconds

Re-implementation in Node.js => page load time decreased by 50%

Moved from Rails for their mobile traffic, reducing the number of servers from 30 to 3 and increased the performance by upto 20x faster.

Source: http://www.nearform.com/nodecrunch/node-js-becoming-go-technology-enterprise/

Top NPM Packages

- Browserify
- Grunt-cli
- Bower
- Gulp
- Grunt
- Express
- Npm
- Cordova
- forever

https://www.npmjs.com/browse/star

- Most dependent-upon packages
  - Lodash
  - Request
  - Async
  - Underscore
  - Express
  - Bluebird
  - Chalk
  - Commander
  - Debug

NodeJS REPL (Read Evaluate Print Line)

```
PS E:\CT\NodeJS\CodePlay\nodejs> node
> function MinionHello(){console.log("Hello! Minion1 is here !!!");}
undefined
> MinionHello();
Hello! Minion1 is here !!!
undefined
> parseInt(Math.random()*213123)
195841
> .save syedawase_repl.js
Session saved to:syedawase_repl.js
> .exit
```

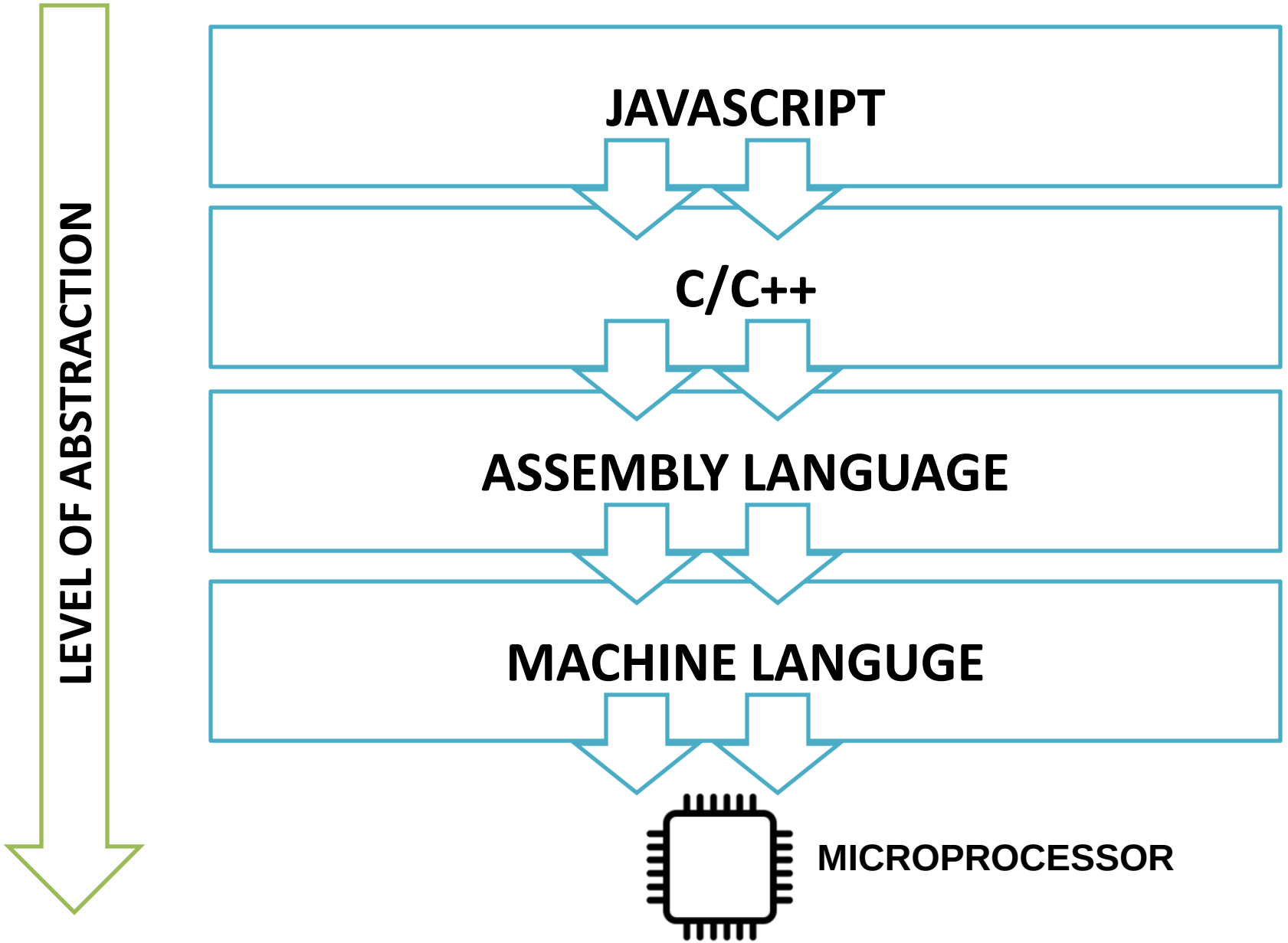
NodeJS REPL

```
PS E:\CT\NodeJS\CodePlay\nodejs> node
> fs.readFile("pega.txt","utf8",function(error,content){
... console.log(content);
... });
undefined
> pega 2 broad streams
1. business architect (system architect, senior system architect, lead system architect)
2. system architect (system architect, senior system architect, lead system architect)
3. UI specialist
4. Decision Specialist
https://pdn.pega.com/community/pega-product-support

Learning Paths
 1. Certified System Architect
 2. Certified Senior System Architect
 3. Certified Lead System Architect
 4. Certified DCO Architect
 5 Certified UI Specialist
 6. Certified Customer Service System Architect
 7. Certified Pega Marketing Consultant
 8. Certified Senior Pega Marketing Consultant
 9. Certified Pega Decisioning Consultant


>
>
> .exit
```

NodeJS REPL

Http Server

```
PS E:\CT\NodeJS\CodePlay\nodejs> node
> .load ex2basichttpserver.js
> var http= require('http');
undefined
> http.createServer(function(req,res){
...     res.writeHead(200,{
.....         'Content-Type':'text/plain'
.....     });
...     res.end("This is my Basic Http Server with Node.js: SAK");
...     console.log("Please check the port 9333 in your browser");
...
... }).listen(9333,"127.0.0.1");
Server {
  domain:
   Domain {
     domain: null,
     _events: { error: [Function] },
     _eventsCount: 1,
     _maxListeners: undefined,
     members: [] },
  _events:
   { request: [Function],
     connection: [Function: connectionListener] },
  _eventsCount: 2,
  _maxListeners: undefined,
  _connections: 0,
  _handle: null,
  _usingSlaves: false,
  _slaves: [],
  unref: false
```

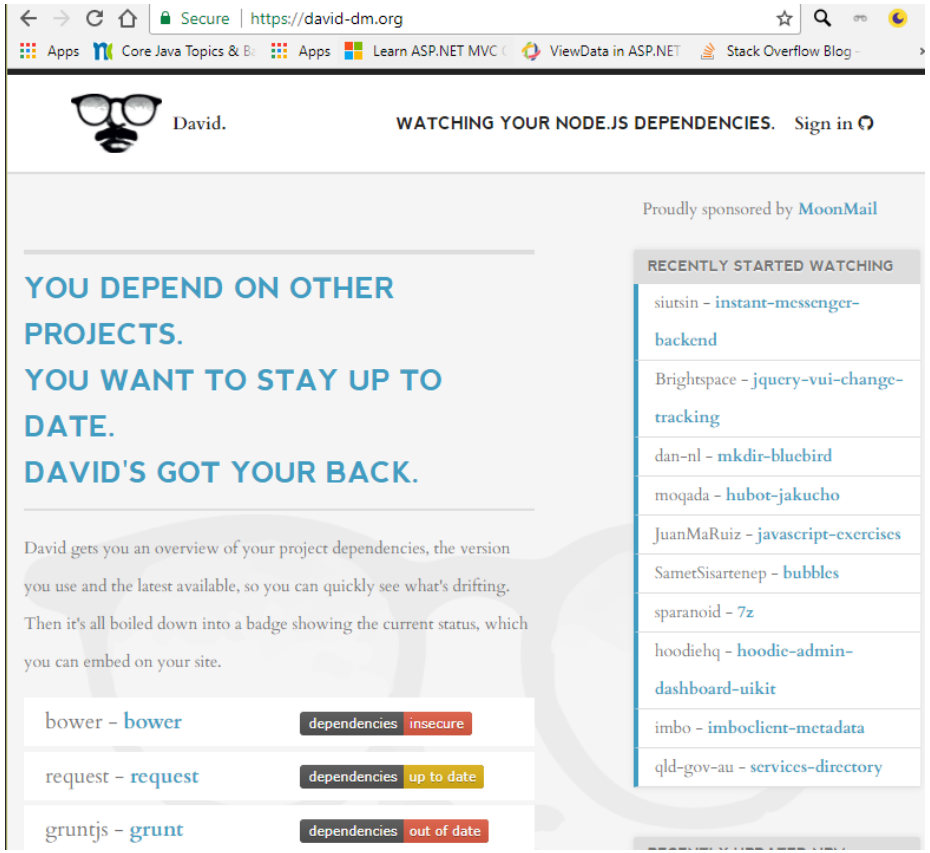


SYED AWASE

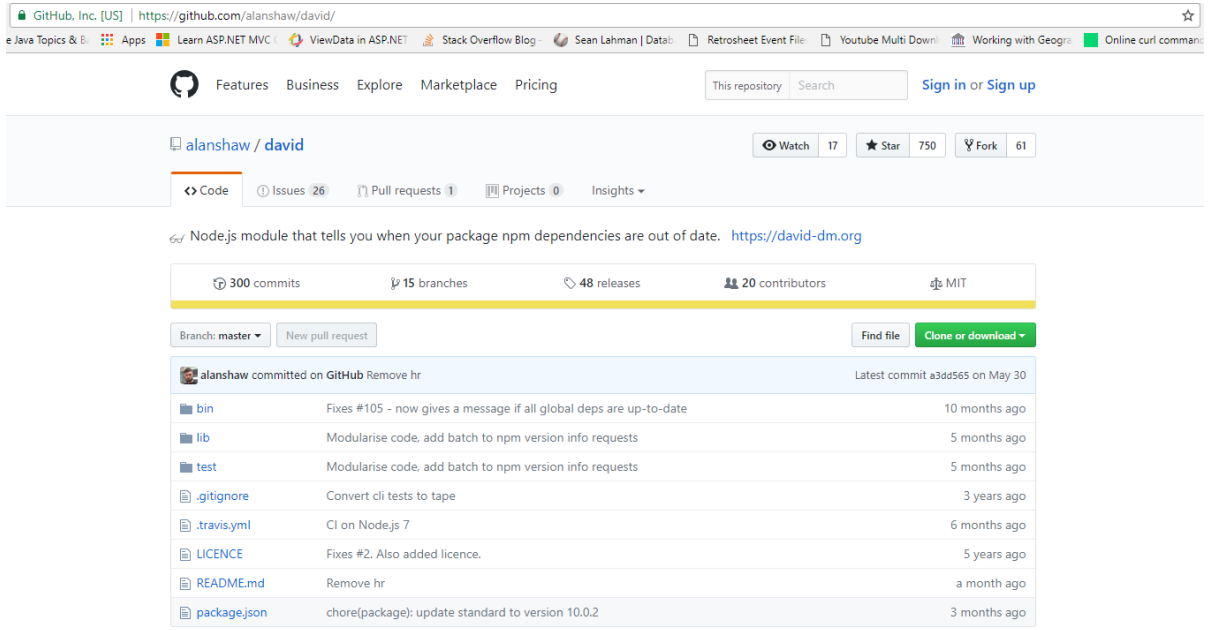
NPM PLAYBOOK

Watching NODE.JS DEPENDENCIES

https://david-dm.org/



https://github.com/alanshaw/david/



Npm install –g david

NPM CLI

CLI Command	Description
npm-access	Set access level on published packages
	npm access public [<package>] npm access restricted [<package>] npm access grant <read-only|read-write> <scope:team> [<package>] npm access revoke <scope:team> [<package>] npm access ls-packages [<user>|<scope>|<scope:team>] npm access ls-collaborators [<package> [<user>]] npm access edit [<package>]
npm-adduser	Add a registry user account
	npm adduser [--registry=url] [--scope=@orgname] [--always-auth] [--auth-type=legacy] aliases: login, add-user
	npm adduser --registry=http://private-registry.example.com --always-auth
Npm bin	Display npm bin folder
	npm bin [-g|--global]

NPM CLI

CLI Command	Description
Npm-bugs	Bugs for a package in a web browser
	npm bugs [<pkgname>] aliases: issues
Npm-build	Build a package
	npm build [<package-folder>]
	npm run-script build
Npm-bundle	Removed in 1.0
Npm-cache	Manipulates packages cache
	Npm cache add <tarball file> Npm cache add <folder> Npm cache add <tarball url> Npm cache add <name>@<version> Npm cache clean [<path>] npm cache rm Npm cache verify

NPM CLI

CLI Command	Description
Npm-config	Manage the npm configuration files
	Npm config set <key><value>[-g|--global] Npm config get <key> Npm config delete <key> Npm config list Npm config edit Npm get <key> Npm set <key> <value> [-g|global]
Npm-dedupe	Reduce duplication
	Npm dedupe Npm ddp
Npm-deprecate	Deprecate a version of a package
	npm deprecate <pkg>[@<version>] <message>

NPM CLI

CLI Command	Description
Npm-dist-tag	Modify package distribution tags
	Npm dist-tag add <pkg>@<version> [<tag>] Npm dist-tag rm <pkg> <tag> Npm dist-tag ls [<pkg>]
Npm-docs	Docs for a package in a web browser
	Npm docs [<pkgname>] Npm home .
Npm-doctor	Check your environments
	Npm doctor
Npm-edit	Edit an installed package
	Npm edit <pkg>[@<version>]
Npm-explore	Browse an installed package
	Npm explore <pkg>

NPM CLI

CLI Command	Description
Npm-init	Interactively create a package.json file
	Npm init Npm init-y
Npm-install	Install a package
	npm install (with no args, in package dir) npm install [<@scope>/]<name> npm install [<@scope>/]<name>@<tag> npm install [<@scope>/]<name>@<version> npm install [<@scope>/]<name>@<version range> npm install <git-host>:<git-user>/<repo-name> npm install <git repo url> npm install <tarball file> npm install <tarball url> npm install <folder>

NPM CLI

CLI Command	Description
Npm-link	Symlink a package folder
	npm link (in package dir) npm link [<@scope>/]<pkg>[@<version>]
Npm-logout	Logout of the registry
	npm logout [--registry=<url>] [--scope=<@scope>]
Npm-ls	List installed packages
	npm ls [[<@scope>/]<pkg> ...]
Npm-outdated	Check for outdated packages
	npm outdated [[<@scope>/]<pkg> ...]
Npm-owner	Manage package owners
	npm owner add <user> [<@scope>/]<pkg> npm owner rm <user> [<@scope>/]<pkg> npm owner ls [<@scope>/]<pkg>

NPM CLI

CLI Command	Description
Npm-pack	Create a tarball from a package
	npm pack [[<@scope>/]<pkg>...]
Npm-ping	Ping npm registry
	npm ping [--registry <registry>]
Npm-prune	Remove extraneous packages
	npm prune [[<@scope>/]<pkg>...] [--production]
Npm-publish	Publish a package
	npm publish [<tarball>|<folder>] [--tag <tag>] [--access <public|restricted>]
Npm-rebuild	Rebuild a package
	npm rebuild [[<@scope>/<name>]...]

NPM CLI

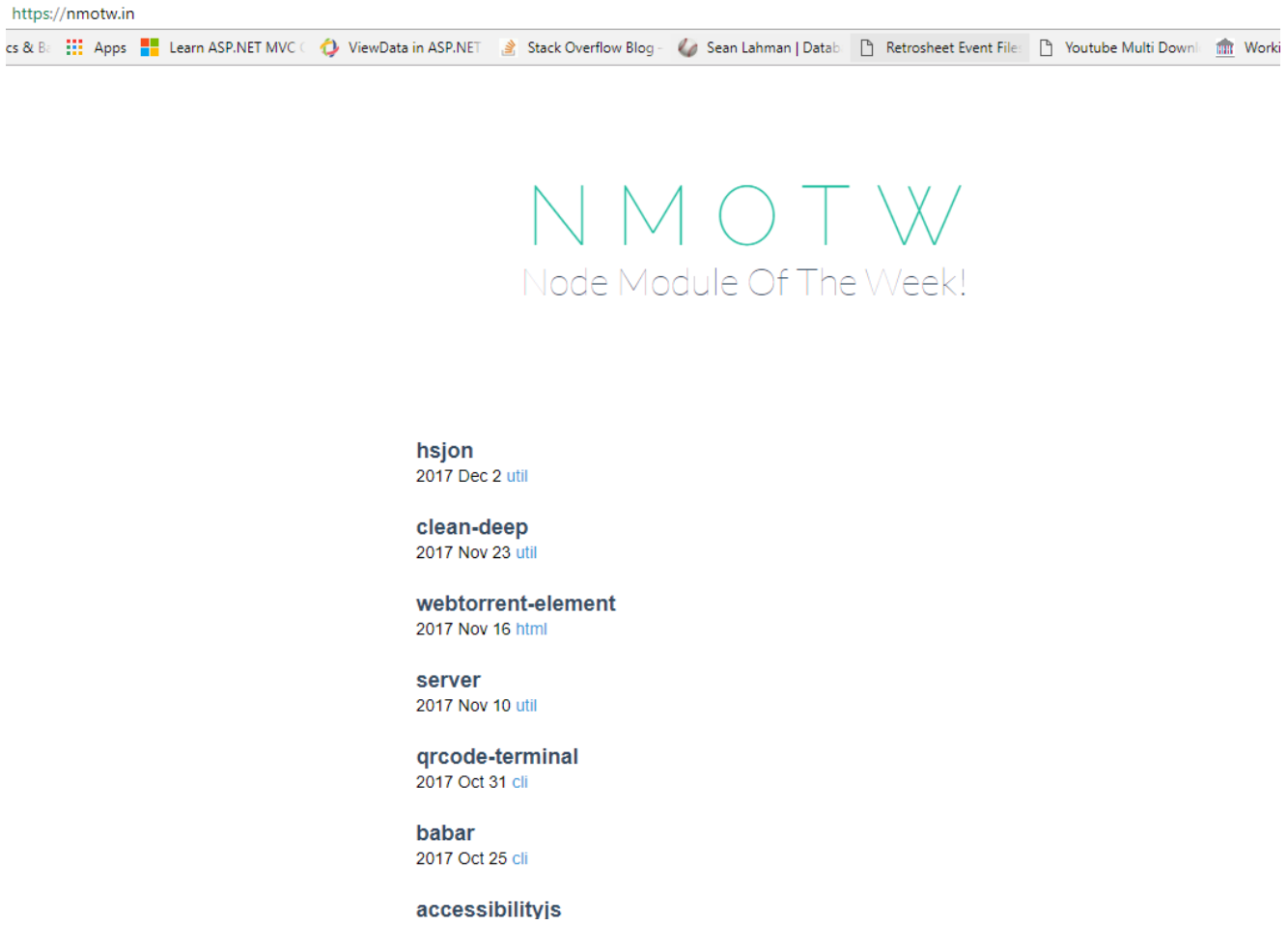
CLI Command	Description
Npm-repo	Open package repository page in the browser
	Npm repo [<pkg>]
Npm-restart	Restart a package
	Npm restart [--args]
Npm-root	Display npm root
	npm root [-g]
Npm-shrinkwrap	Lock down dependency versions for publication
	npm shrinkwrap
Npm-star	Mark your favorite packages
	npm star [<pkg>...] npm unstar [<pkg>...]

NPM CLI

CLI Command	Description
Npm-uninstall	Remove a package
	pm uninstall [<@scope>/]<pkg>[@<version>]... [-S|--save|-D|--save-dev|-O|--save-optional]
Npm unpublish	Remove a package from the registry
	npm unpublish [<@scope>/]<pkg>[@<version>]
Npm-update	Update a package
	npm update [-g] [<pkg>...]
Npm-version	Bump a package version
	npm version [<newversion> | major | minor | patch | premajor | preminor | prepatch | prerelease | from-git]
Npm-view	View registry info
	npm version [<newversion> | major | minor | patch | premajor | preminor | prepatch | prerelease | from-git]

Node module of the week

https://nmotw.in/



SYED AWASE

CREATING NODE MODULES

Writing your own Modules


```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * Writing your own module in Nodejs
 */
module.exports ={
    sayHello:function(){
        return "Hello: Syed Awase! How are you doing?";
    },
    doTask:function(){
        return "What task you want me to automate or do for you?";
    }
}
```

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * consuming the module
 */
var HelloModule=require('./ex3A-module.js');

var speakout = HelloModule.sayHello();
console.log(speakout);

var executeTask=HelloModule.doTask();
console.log(executeTask);
```

Built-in NodeJS Modules

http fs path os cluster

NODEJS MODULES

Module	Description
Async	Provides straight-forward,powerful functions for working with asynchronous javascript
Browserify	Used to serve up a bundle by analyzing recursively all the require() calls in your application to the **browser**
Csv	Used for csv generation, parsing, transformation and serialization for Node.js
Debug	Debugging utility modelled after node's core debugging technique
Express	A fast un-opinionated, minimalist web framework. Provides small, roboust tooling for HTTP servers, making it a great solution for single page applciations and public HTTP APIs
Forever	A simple CLI tool for ensuring that a given node script runs continuously forever
Grunt	A javascript task runner that facilitates creating new projects and makes performing repetitive but necessary tasks such as linting, unit testing, concatenating and minifying files
Gulp	Stream build system that helps you automate painful or time consuming tasks in your development workflow.

NODEJS MODULES

Module	Description
Hapi	Stream build system that helps you automate painful or time consuming tasks in the development workflow
Http-server	A simple zero-configuration command-line http server, powerful for production usage.
Inquirer	A collection of common interactive command line user interfaces
Jshint	Static analysis tool to detect errors and potential problems in javascript code and to enforce your team's coding conventions
KOA	A web app framework, which is an expressive http middle ware for node.js to make web applciations and APIs more enjoyable to write.
Lodash	Modern javascript utility library delivering modularity , performance and extras
Moment	Library for parsing, validating, manipulating and formatting dates
Mongoose	MongoDB object modeling tool designed to work in an asynchronous environment
MongoDB	High-level API on top of mongodb-core.
Nodemon	Simple monitor script for use during development to watch files in the directory.

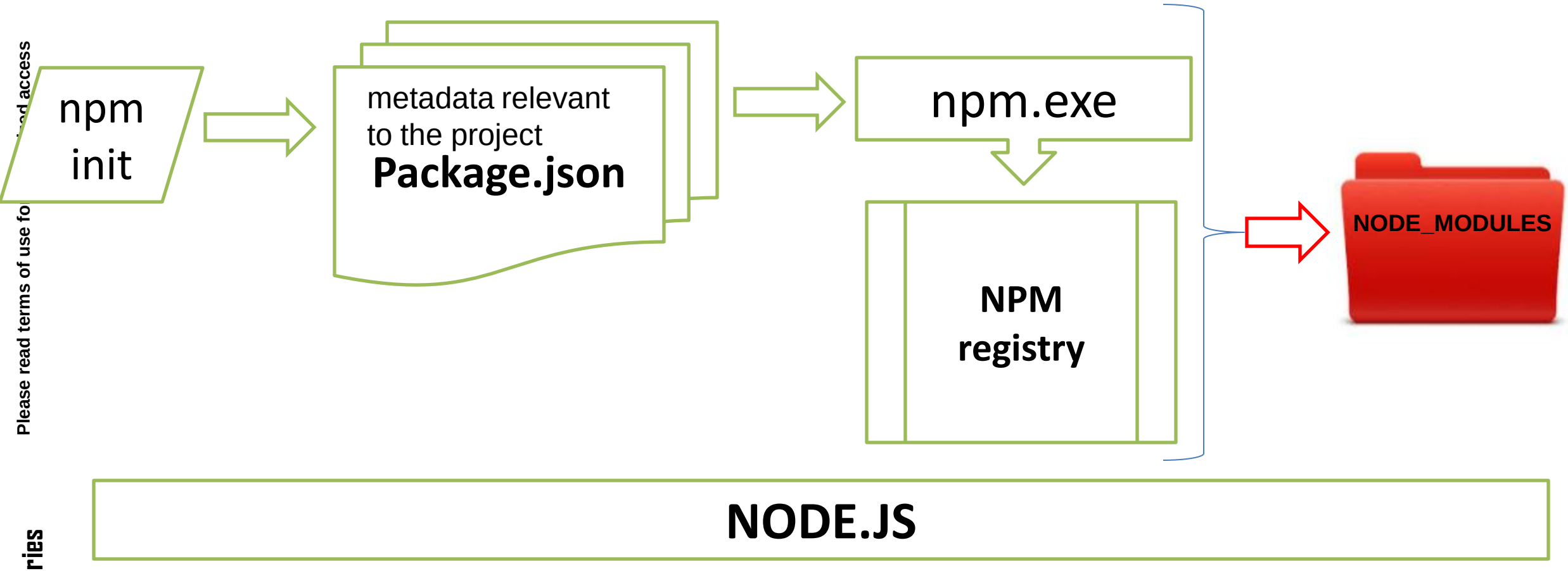
NODEJS MODULES

Module	Description
Nodemailer	Enables email-sending from a node.js applications
Optimist	Nodejs library for option parsing with an argv hash
PhantomJS	Headless webkit with JS API. Has fast and native support for web standards, DOM handling, CSS selector, JSON, Canvas, and SVG
Passport	Authentication middleware for node.js.
Q	Promises library. A promise is an object that represents the return value or the thrown excepton that the function may eventually provide.
Request	A simplified Http request client make it possible to make http calls. It supports https and follows redirects by default.
Socket.io	Realtime framework server
Sails	API driven framework for buildign real-time apps using MVC conventions.
Through	Enables stream construction. It is easy way to create a stream that is both readable and writeable.

NODEJS MODULES

Module	Description
Underscore	Utility library for javascript that provides support for usual functional suspects (each, map, reduce, filter) without extending any core javascript objects
Validator	A node js module for library of string validators and sanitizers
Winston	A multi-transport async loggin library for node.js
WS	A simple to use, blazing fast and thoroughly tested websocket client, server and console for node.js
Xml2js	A simple XML to javascript object converter
Yo	A CLI tool for running yeaoman generators
Zmq	A high performance asynchronous messaging library, aimed at use in distributed or concurrent applications.

Node Package Manager



Creating NODEJS Package

Hextobinconvertor.js

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * Exports package : binary to hexadecimal convertor
 */

exports.bin =function(input){
    return input.toString(2);
};

exports.hex = function(input){
    return input.toString(16);
};
```

Demo.js

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * Consuming the custom package created
 */
var hextobin = require('./bintohexconvertor.js');
var num = 100;
var mybin = hextobin.bin(num);
console.log("binary" + mybin);

var myhex = hextobin.hex(num);
console.log("binary" + myhex);
```

Publishing NODEJS PACKAGES

```
//to add a user account to npmjs
npm adduser

//login to your account
npm login

// credentials stored on the client
npm config ls

// publish to the npmjs repository
npm publish
```

Minimal Node.js Framework

- They provide the most basic of functions and API and minimalistic Node.js functionality
  - Express.js : lightweight and easy-to-use framework to develop pure JS or hybrid web or mobile applications of any scale.
  - KOA: provides server and server related functionalities
  - Total.js: enterprise large scale web applications.
  - Sails.js: service driven architecture and is equipped with data-driven API sets. Most useful for multi-player games, chat apps, real-time dashboard applications, enterprise-level node.js apps.

Full-stack Node.js Framework

- They are prepackaged with all the necessary application scaffolding, complete template engines, web sockets and persistence libraries to accelerate the building of real-time scalable web and mobile applications.
  - Meteor
  - Mean.io

SYED AWASE

BUILT-IN NODE MODULES: UTIL

SYED AWASE

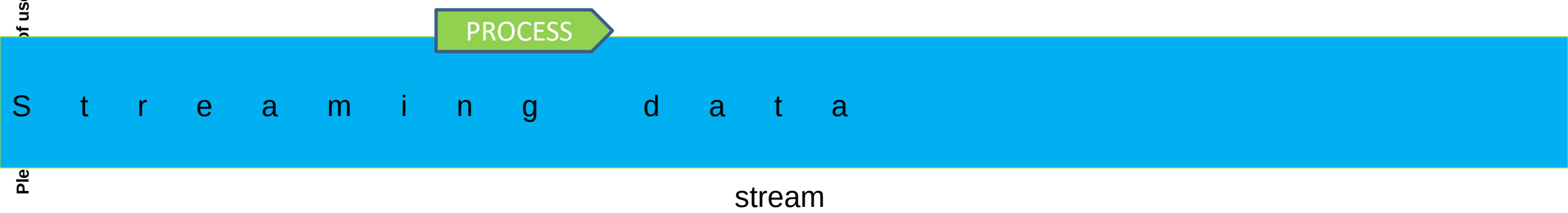
BUILT-IN NODE MODULES: STREAMS AND BUFFERS FILE SYSTEMS

Buffer

- A temporary holding spot for data being moved from one place to another.
- The buffer is of limited size.

Stream

- A sequence of data made available over time.
- Pieces of data that eventually combine into a whole.



Buffer

- Creating a buffer and printing it.

```
var buf = new Buffer(100);
var moviebuff =new Buffer(['PIKU','Yellow River','PK','Batman','Superman']);
var cars= new Buffer([10,20,30,40,50]);
console.log(buf);
console.log(moviebuff);
console.log(cars);
console.log(cars.toString());
console.log(moviebuff.toString());
```

- Writing to a buffer

```
//writing to a buffer
//buf.write(string[,offset],[,length],[,encoding]);
//the index of the buffer to start writing at.
// the length/ number of bytes to write.
// encoding what is the encoding
var myStringStream = buf.write("Hello you fool i love you");
console.log(myStringStream);
console.log(myStringStream.toString());
var stmt = new Buffer("We will we will rock you");
var jsonstmt=stmt.toJSON(stmt);
console.log(jsonstmt);
```

Buffer

- Concatenating two buffers

```
// concatenating buffer
var valueOne = new Buffer("PointOne");
var valueTwo = new Buffer("PointTwo");
var valueThree = new Buffer("PointThree");
var valueFour = Buffer.concat([valueOne,valueTwo,valueThree]);
console.log("We shall wakeup from the dream:"+ valueFour.toString());
```

- Comparing buffers

```
//comparing buffers
var result = valueOne.compare(valueTwo);
console.log("comparing two buffers:"+result);
```

- Copying buffers

```
//copying buffers
var mySteak = new Buffer("I love barbeque");
var mySalad=new Buffer('I love salad with smoked salmon fish');
//copying buffer mySteak to mySalad.
mySteak.copy(mySalad);
console.log("buffer (mSalad) content is:"+ mySalad.toString());
console.log("buffer(mySteak contentis:"+mySteak.toString());
```

- slicing buffers

```
//slicing buffer
var AmazonBuffer = new Buffer("Lets get it going");
var butterSlice = AmazonBuffer.slice(0,15);
console.log("butterSlice is:"+ butterSlice.toString());
```

Buffer

- Computing buffer length and isBuffer check

```
//buffer length
//it return the size of the buffer
var bugger = new Buffer("this bugger is bigger than burger");
//length of the buffer
console.log("buffer length:"+ bugger.length);
//checking whether it is a valid encoding
var ubcity=new Buffer('UBCity','utf-8');
var acity = new Buffer('Amsterdam','ascii');
//base64 encoding
var aceofbase = new Buffer('All that she wants ', 'base64');
var ternary=new Buffer("terminator",'binary');
console.log("buffer check:"+Buffer.isBuffer(ubcity));
```

Streams

- Streams are objects that let you read data from a source or write data to a destination in continuous fashion.
- Type of streams
  1. Readable-stream which is used for read operation
  2. Writable-Stream which is used for write operation.
  3. Duplex: Stream which can be used for both read and write operation
  4. Transform- a type of duplex stream when the output is computed based on the input

- Each type of stream is an EventEmitter instance and throws several events at different instance of times.

Stream

Read

```
var fs = require('fs');
var guardianData='';
//create a readable stream
var readerStream = fs.createReadStream('./2014guardiannew.txt');
//set the encoding to the utf-8 format
readerStream.setEncoding('UTF8');
//handle stream events --> data, end and error
readerStream.on('data',function(chunk){
    guardianData+=chunk;
});
//print data on completion of stream read
readerStream.on('end', function(){
    console.log(guardianData);
});
readerStream.on('error', function(err){
    console.log(err.stack);
});
console.log("Program Ended");
console.log(guardianData);
```

Write

```
var fs =require('fs');
var stenoData = 'Hello, Syed you fool i love you, come enjoy the joy
ride';
//creating a writable stream
var writerStream =fs.createWriteStream('./syedOutput.txt');
//write the data to stream with encoding to be UTF8
writerStream.write(stenoData,'UTF8');
//mark the end of the file
writerStream.end();
//HandleStream events => finish and error
writerStream.on('finish',function(){
    console.log('Write completed');
});
writerStream.on('error',function(err){
    console.log(err.stack);
});
console.log("Program has Ended");
```

Stream

Piping the streams

- A mechanism where the output of one stream is sent as input to another stream.
- Used to get data from one stream and to pass the output of that stream to another stream.

Write

```
var fs =require('fs');
var stenoData = 'Hello, Syed you fool i love you, come enjoy the joy
ride';
//creating a writable stream
var writerStream =fs.createWriteStream('./syedOutput.txt');
//write the data to stream with encoding to be UTF8
writerStream.write(stenoData,'UTF8');
//mark the end of the file
writerStream.end();
//HandleStream events => finish and error
writerStream.on('finish',function(){
    console.log('Write completed');
});
writerStream.on('error',function(err){
    console.log(err.stack);
});
console.log("Program has Ended");
```

Stream

- Asynchronous and Synchronous reading of files.

```
var fs =require('fs');
//Aysnchronous read
fs.readFile('eins.txt', function(err,data){
    if(err){
        return console.error(err);
    }
    console.log("Asynchronous read:"+data.toString());
});
//Synchronous reading
var data = fs.readFileSync('eins.txt');
console.log("Synchronous read:"+data.toString());

//asynchronous file -opening
console.log("I am going to open the file now");
fs.open('eins.txt', 'r+', function(err,fs){
    if(err){
        return console.error(err);
    }
    console.log("file open successfully");
});
```

Stream

File Stats

Method	Description
Stats.isFile()	Return true if file type is of simple file
Stats.isDirectory()	Returns true if directory
Stats.isBlockDevice()	Returns true if file type of a block device
Stats.isCharacterDevice()	Returns true if type of a character device
Stats.isSymbolicLink()	Returns true if file type of symbolic link
Stats.isFIFO()	Returns true if file type of a FIFO
Stats.isSocket()	Returns true if file type of socket

```
//extracting file information using stat
fs.stat('./eins.txt',function(err,stats){
    if(err){
        return console.error(err);
    }
    console.log("file statistics");
    console.log(stats);
    console.log("is it a file?:"
    +stats.isFile());
    console.log("isDirectory?"
    +stats.isDirectory());
    //checking for block device
    console.log("isSocket"
    +stats.isBlockDevice());
    //checking for character device
    console.log("isCharacterDevice"+
    stats.isCharacterDevice());
    //checking for FIFO
    console.log('isFIFO'+stats.isFIFO());
});
```

Stream

Writing to a file

```
var fs=require('fs');
console.log("Going to write into existing file");
fs.writeFile('myfile.txt', 'simplelearning', function(err){
    if(err){
        return console.error(err);
    }
    console.log("Data written successfully!");
    fs.readFile('myfile.txt', function(err,mydata){
        if(err){
            return console.error(err);
        }
        console.log("Asynchronous read:"+data.toString());
    });
});
```

Buffered file read

```
var fs = require('fs');
var mybugger = new Buffer(2048);
console.log("opening the file:khulja sim sim");
fs.open('./simsim.txt','r+',function(err,fd){
    if(err){
        return console.error(err);
    }
    console.log("file opened successfully!");
    fs.read(fd, mybugger,0,mybugger.length, 0, function(err,bytes){
        if(err){
            console.log(err);
        }
        console.log(bytes+"bytes read");
        if(bytes>0){
            console.log(mybugger.slice(0,bytes).toString());
        }

    });
    // Close the opened file.
    fs.close(fd, function (err) {
        if (err) {
            console.log(err);
        }
        console.log("File closed successfully.");
    });
});
```

Stream

File delete

- Fs.unlink(path,callback)
  - To delete a file
    - Path -> the file name including path
    - Callback -> call back function no arguments other than a possible exception are given to the completion callback.

```
var fs=require('fs');
console.log("Going to delete an existing
file");
fs.unlink('./one.txt', function(err){
    if(err){
        return console.error(err);
    }
    console.log("file deleted
    successfully!");
});
```

Stream

Create a directory

```
var fs=require('fs');

console.log("Going to create directory
/sak");
fs.mkdir('./sak', function(err){
    if(err){
        return console.error(err);
    }
    console.log("Directory successfully
    created");
})
```

Read a Directory

```
//reading a directory
var fs = require('fs');
console.log("start reading the directory");
fs.readdir("./",function(err,files){
    if(err){
        return console.error(err);
    }
    files.forEach(function(file){
        console.log(file);
    });
});
```

Stream

Remove a Directory

```
var fs = require('fs');
console.log("deleting directories");
fs.rmdir('./sak', function(err){
    if(err){
        return console.error(err);
    }
    console.log("read directory");
    fs.readdir('./', function(err,files){
        if(err){
            return console.error(err);
        }
        files.forEach(function(file){
            console.log(file);
        });
    });
});
```

SYED AWASE

BUILT-IN NODE MODULES: PATH

Path

Path Module

- Path module is used for handling and transforming file paths.

```
var path=require('path');
//Normalizing path
console.log('normalization:'+path.normalize
('/usr/home//sak/awasedocs/..'));
//joining path s
console.log('joining path:'+path.join('/syedawase','azeez',
'ameese/docs'));
//resolving
console.log('resolve:'+path.resolve('syedawase.js'));
//extension name
console.log('extension name:'+path.extname('syedawase.js'));
```

SYED AWASE

BUILT-IN NODE MODULES: DOMAIN

DOMAIN

- It is used to intercept unhandled error. These unhandled error can be intercepted using internal binding or external binding.
  - Internal binding- Error emitter is executing its code within the run method of a domain.
  - External binding – Error emitter is added explicitly to a domain using its add method.

```
var EventEmitter =require('events').EventEmitter;
var domain=require('domain');
var emitter1=new EventEmitter();
//create a domain
var sycliq=domain.create();
sycliq.on('error',function(err){
    console.log("domain1 handled this error("+err.message+")");
});

sycliq.add(emitter1);

emitter1.on('error',function(err){
    console.log("handling error(" + err.message + ")");
})
emitter1.emit('error', new Error('to be handled by listener'));
emitter1.removeAllListeners('error');
emitter1.emit('error', new Error('to be handled by domain1'));
```

SYED AWASE

BUILT-IN NODE MODULES: DOMAIN

DOMAIN

- It is used to intercept unhandled error. These unhandled error can be intercepted using internal binding or external binding.
  - Internal binding- Error emitter is executing its code within the run method of a domain.
  - External binding – Error emitter is added explicitly to a domain using its add method.

```
var EventEmitter =require('events').EventEmitter;
var domain=require('domain');
var emitter1=new EventEmitter();
//create a domain
var sycliq=domain.create();
sycliq.on('error',function(err){
    console.log("domain1 handled this error("+err.message+")");
});

sycliq.add(emitter1);

emitter1.on('error',function(err){
    console.log("handling error(" + err.message + ")");
})
emitter1.emit('error', new Error('to be handled by listener'));
emitter1.removeAllListeners('error');
emitter1.emit('error', new Error('to be handled by domain1'));
```

SYED AWASE

BUILT-IN NODE MODULES: URL

URL Module

- URL module splits up a web address into readable parts.

```
var url=require('url');
var myaddress
='http://sycliq.com:8080/cropanalytics?
year=2015&month=june';
var myquery = url.parse(myaddress,true);
//to query the host information
console.log(myquery.host);
//query the pathname
console.log(myquery.pathname);
//query parameter search
console.log(myquery.search);
//returns an object
var mydata = myquery.query;
console.log(mydata.month);
```

File Server

```
var http=require('http');
var url=require('url');
var fs = require('fs');
//create a server
http.createServer(function(req,res){
    var myquery = url.parse(req.url,true);
    var fileName= "."+myquery.pathname;
    fs.readFile(fileName, function(err,data){
        if(err){
            res.writeHead(404,{'Content-Type':'text/html'});
            return res.end("404 Not found");
        }
        res.writeHead(200,{'Content-Type':'text/html'});
        res.write(data);
        return res.end();
    });
}).listen(9191);
```

SYED AWASE

BUILT-IN NODE MODULES: NET

NET MODULE

- It is used to create both servers and clients.
- It provides an asynchronous network wrapper.

Basic Server

```
//creating a server using net
var net = require('net');
var server = net.createServer(function(connection){
    console.log('client connected');

    connnection.on('end',function(){
        console.log('client disconnected');
    });

    connection.write('Hello, syed you fool, i love you');
    connection.pipe(connection);
});

//listening to the server port
server.listen('9191', function(){
    console.log("You are listening at 9191");
});
```

NET MODULE

Client

```
var net = require('net');
var client = net.connect({port:8080},
function(){
    console.log('you are now connected to
    the server');

});

client.on('data', function(data){
    console.log(data.toString());
    client.end();
});

client.on('end', function(){
    console.log('disconnected from server');
});
```

Basic TCP Server with Node.JS

```
var net = require('net');
net.createServer(function(socket){
    socket.write("Echo Server Syed Awase Listening???\r\n");
    socket.pipe(socket);
}).listen(6333, "127.0.0.1");
```

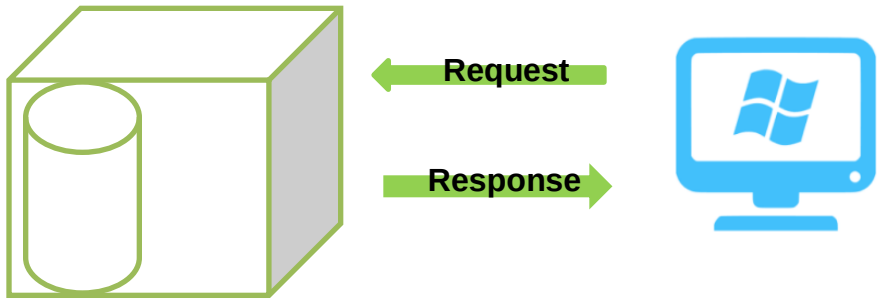
SYED AWASE

BUILT-IN NODE MODULES: HTTP

Basic HTTP Server with Node.JS

- Node.js helps us create a basic http server, using http package
- Using http.createServer, we can create a http server
- It has a function call with request and response object.

```
var http= require('http');
http.createServer(function(req,res){
    res.writeHead(200,{
        'Content-Type':'text/plain'
    });
    res.end("This is my Basic Http Server with Node.js: SAK");
    console.log("Please check the port 9333 in your browser");

}).listen(9333,"127.0.0.1");
```



Http Server with MIME Types

```
//http access
var http=require('http');
//filesystem
var fs = require('fs');
var path = require('path');
var host = '127.0.0.1';
var port="8888";
//define the mime type to serve
var mimes = {
    '.html':'text/html',
    '.css':'text/css',
    '.js':'text/javascript',
    'gif':'image/gif',
    '.jpg':'image/jpeg',
    '.png':'image/png'
}
//creating a server
var server = http.createServer(function(req,res){
    var filepath =(req.url==='/')?('./index.html'):('.'+req.url);
    //extracting mimes
    var ctype = mimes[path.extname(filepath)];
    //check if the file exists
    fs.exists(filepath, function(file_exists){
        if(file_exists){
            //reading and serving the file
            fs.readFile(filepath, function(error,content){
                if(error){
                    res.writeHead(500);
                    res.end();
                }else{
                    res.writeHead(200,{'Content-Type':ctype});
                    res.end(content,'utf-8');
                    console.log('Please open the url http://localhost:8888');
                }
            });
        }else{
            res.writeHead(404);
            res.end('Page not found error: the current file requested is not available');
        }
    })
}).listen(port,host);
```

NodeJS REPL

Http Server using Streams

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * //using streams for efficient rendering of data
 */
var http= require('http');
var host='127.0.0.1';
var port = "8000";
//filesystem
var fs = require('fs');
var path = require('path');

//defining mime types to serve
var mimes ={
    ".html":"text/html",
    ".css":"text/css",
    ".js":"text/javascript",
    ".gif":"image/gif",
    ".jpg":"image/jpeg",
    ".png":"image/png"
};
var server = http.createServer(function(req,res){
    var filepath = (req.url ==='/')? ('./index.html'):('.'+req.url);
    //extracting mimes
    var ctype = mimes[path.extname(filepath)];
    //check if the file exists
    fs.exists(filepath, function(file_exists){
        if(file_exists){
            res.writeHead(200,{'Content-Type':ctype});
            var streamFile=fs.createReadStream(filepath).pipe(res);
            streamFile.on('error',function(){
                res.writeHead(500);
                res.end();

            });
        }else{
            res.writeHead(404);
            res.end("Page Not Found Error: The Current File requested is not available on Server");
        }
    })

}).listen(port,host);
```

SYED AWASE

NPM PACKAGE: QR-IMAGE

QR-Image

- QR Code generator for nodejs
- Used to generate image in png, svg, eps and pdf formats.
- Supports numeric and alphanumeric modes
- Supports UTF-8 encoding format

- npm install qr-image –save

Creating an QR-Image

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * npm package:qr-image demo
 */
var qrImage = require('qr-image');
var fs = require('fs');

qrImage
    .image("http://www.territorialprescience.com",{type:'jpeg', size:50})
    .pipe(fs.createWriteStream("TRPI.jpeg"));

    qrImage
    .image("I love NPM!",{type:'svg'})
    .pipe(require('fs').createWriteStream('msg.svg'));
```

SYED AWASE

NPM PACKAGE: QR-IMAGE

QR-Image

- QR Code generator for nodejs
- Used to generate image in png, svg, eps and pdf formats.
- Supports numeric and alphanumeric modes
- Supports UTF-8 encoding format

- npm install qr-image –save

Creating an QR-Image

```
/**
 * © COPYRIGHT 2014 Syed Awase Khirni
 * TPRI-SYCLIQ sak@territorialprescience.com/sak@sycliq.com
 * +91.9035433124
 * npm package:qr-image demo
 */
var qrImage = require('qr-image');
var fs = require('fs');

qrImage
    .image("http://www.territorialprescience.com",{type:'jpeg', size:50})
    .pipe(fs.createWriteStream("TRPI.jpeg"));

    qrImage
    .image("I love NPM!",{type:'svg'})
    .pipe(require('fs').createWriteStream('msg.svg'));
```

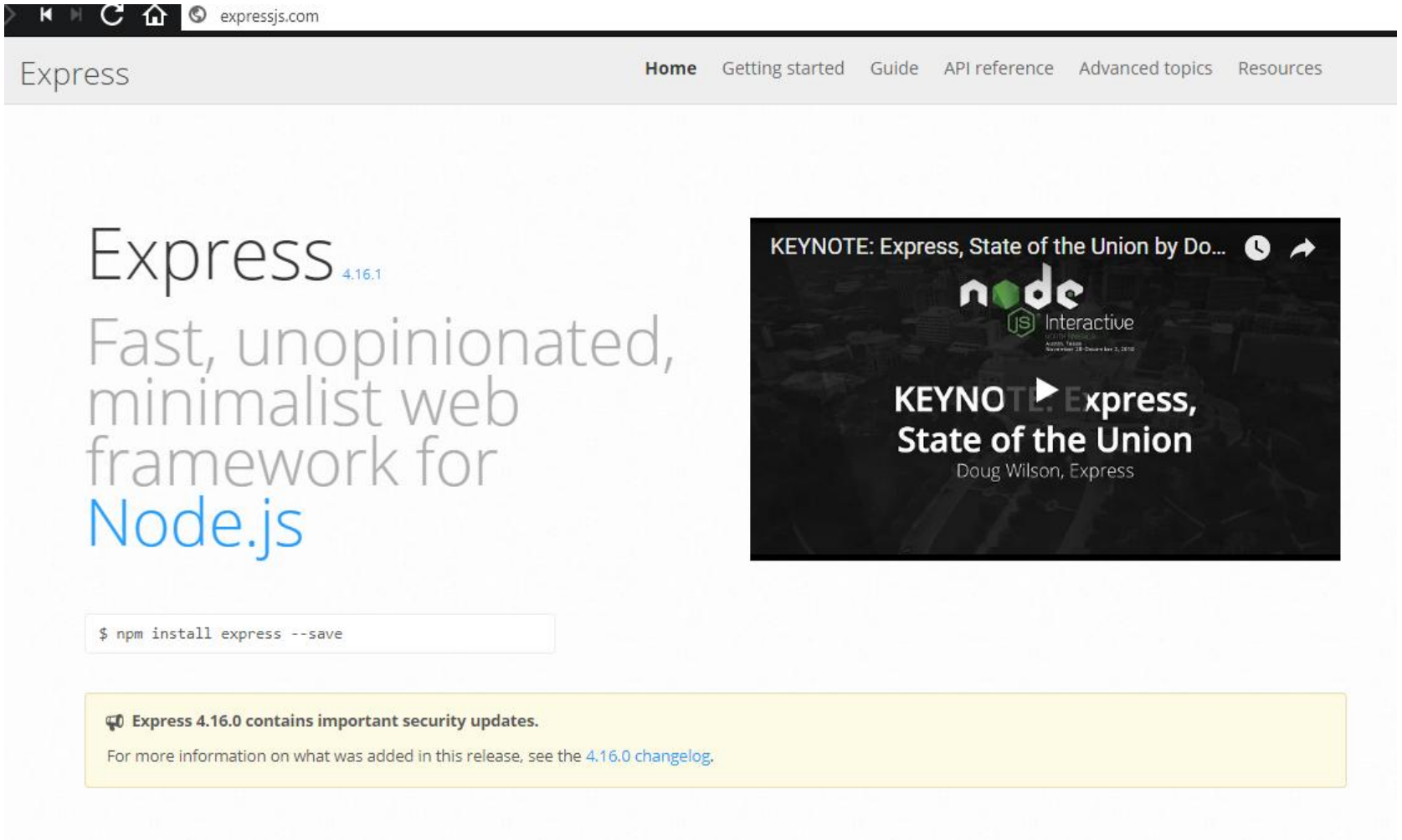
Fast, unopinionated, minimalist web framework for node.js

SYED AWASE

NODE WITH EXPRESS: APPLICATION DEVELOPMENT

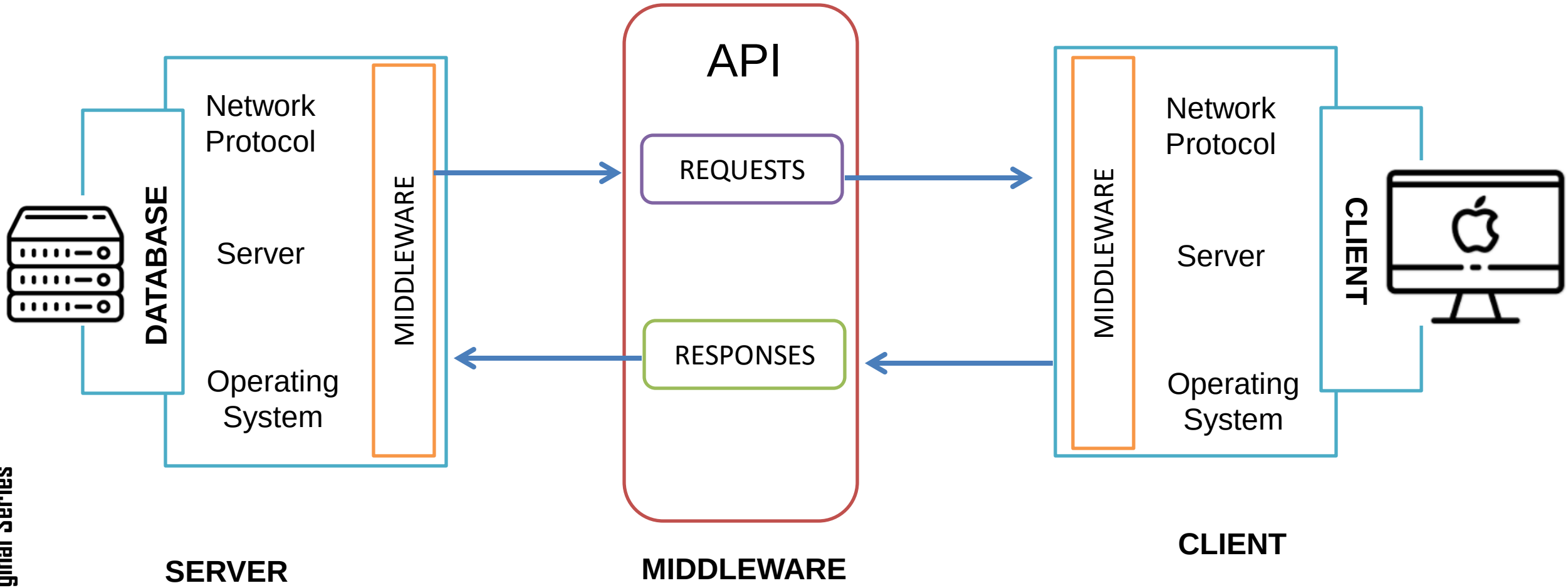
ExpressJS

http://expressjs.com



Used by

Myspace, country, yummly, mozilla persona and geeklist



Original Series

Express

1. Powerful routed API
2. Huge Community
3. Simple and Fast API
4. Detailed Documentation
5. Support for third party plugins

A layer built on top of Node.js that helps manage a server and our routes.

1. Labour intensive tasks and scaffolding
2. Provides no universal way of organizing things

KOA

1. Lightweight framework
2. ES6 generators
3. Easy learning curve

1. Unstable and lack of community support

Sails.js

1. Many automated generators
2. Requires no additional routing
3. Transparent support for socket.io
4. Quick Rest API generation

1. Waterline ORM is rather limited
2. Poor documentation

Express Installation

Global installation of express

- npm install –g express

Project specific installation of express

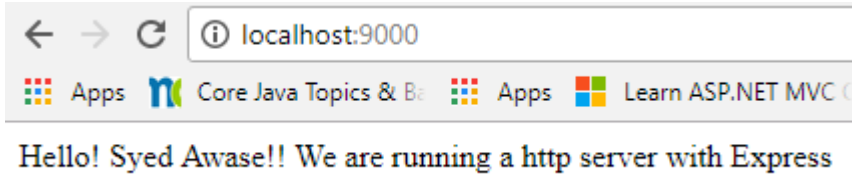
- Npm install –save express

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo> npm install -g express
+ express@4.16.2
updated 1 package in 14.229s
```

Basic http server with express demo

1. npm init –y
2. Npm install –save express
3. Node basichttpserver.js

```
//basic http server using express
var express = require('express');
//instantiation
var app = express();
app.get('/',function(request,response){
    response.send("Hello! Syed Awase!! We are running a http server with Express");
});

//listening port
app.listen(9000, function(){
    console.log("Express Http Server running at 9000 port");
});
```



Express with basic routes demo

1. npm init –y
2. Npm install –save express
3. Node basicroutes.js

```
//import express package
var express = require('express');
var app = express();
//root route request
app.get('/', function(request,response){
    response.send("<h1>SycliQ Http Root Route using Express</h1>");
});
// routing to cropdata
app.get('/cropdata',function(request,response){
response.send("<h1>Crop Data Set Page</h1>");
});
//error redirection
app.use(function(request,response,next){
    response.status(404).send("sorry, the route requested does not exist");
});
//start the server in the port 9000!
app.listen(9000, function(){
    console.log("listenning at port 9000: http://localhost:9000");
});
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\02-basicroutes> node .\basicroutes.js
listenning at port 9000: http://localhost:9000
```

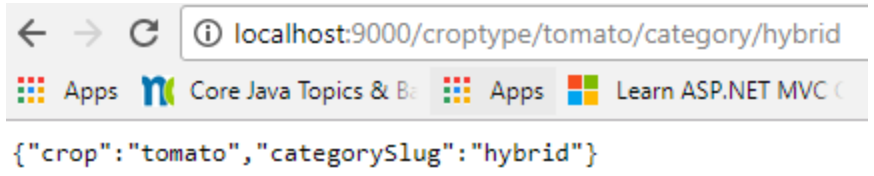
Express with route handlers

1. npm init –y
2. Npm install –save express
3. Node basicroutes.js

```
// install express
var express = require('express');
//create an instance of express
var app = express();
app.get('/', function (request, response) {
    response.send('<h1>SycliQ HTTP SERVER FOR IOT</h1>');
});
// add a routes that answers to all the request types -CRUD
app.route('/commodity')
    .get(function (request, response) {
        response.send('<h1>Commodity:GET OPERATION</h1>');
    })
    .post(function (request, response) {
        response.send('<h1>Commodity:POST OPERATION</h1>');
    })
    .put(function (request, response) {
        response.send('<h1>Commodity:PUT OPERATION</h1>');
    });
// for alternative routes, redirect to
app.get('*', function (request, response) {
    response.send('<h1>The ROUTE YOUR REQUESTED DOES NOT EXIST</h1>');
});
//use regular express
app.get(/com/, function (request, response) {
    response.send('/com/');
});
app.use(function (request, response, next) {
    response.status(404).send("Sorry, You are tresspassing! kindly go to home!!");
});
app.listen(9000, function () {
    console.log('Example app listening on port 9000.');
});
```

Express with route handlers with parameters

1. npm init –y
2. Npm install –save express
3. Node basicroutes.js




```
//import express
var express = require('express');
//instantiate
var app = express();
//route get
app.get('/', function (request, response) {
    response.send('<h1>SycliQ Express HTTP Server</h1>');
});

// route with parameters
app.get('/croptype/:crop/category/:categorySlug', function (request, response) {
    console.log(request.params);
    var cropname = request.params.crop;
    var category = request.params.categorySlug;
    response.send(request.params);
});

app.use(function (request, response, next) {
    response.status(404).send("Sorry! You are tresspassing!");
});

app.listen(9000, function () {
    console.log('Example app listening on port 9000.');
});
```

Type of responses

Method	Description
res.download()	Prompt a file to be downloaded
res.end	End the response process
res.json()	Send a JSON response
res.jsonp()	Send a JSON response with JSONP support
res.redirect()	Redirect a request
res.render()	Render a view template
res.send()	Send a response of various types
res.sendFile()	Send a file as an octet stream
res.sendStatus()	Set the response status code and send its string representation as the response body.
res.cookie()	Modify response cookies
res.clearCookies()	Modify response cookies
res.set()	Set response headers

Request

Method	Description
req.app	Holds a reference to the instance of express application that is using the middleware
req.baseUrl	URL path on which a router instance was mounted
req.body	Contains key-value pairs of data submitted in the request body, default is undefined.
req.cookies	This property is an object that contains cookies sent by the request
req.fresh	Indicates whether the request if “fresh”
req.stale	Opposite of req.fresh
req.hostname	Contains the hostname derived from the HOST HTTP HEADER
req.ip	Contains the remote IP address of the request
req.path	Contains the path part of the request URL
req.method	Contains a string corresponding to the HTTP method of the request:GET,PUT,POST …
req.originalUrl	It retains the original request URL
req.params	An object containing properties mapped to the named route parameters.

Request

Method	Description
req.protocol	Contains the request protocol string:either http or https
req.query	An object containing a property for each query string parameter in the route.
req.route	Contains the currently-matched route, a string.
req.secure	A boolean property that is true if a TLS(HTTPS) connection is established
req.signedCookies	The property contains signed cookies sent by the request.
req.subdomains	An array of subdomains in the domain name of the request
req.xhr	A Boolean property that is true if the request's X-Requested-With header field is "XMLHttpRequest", indicating that the request was issued by a client library such as jQuery.
qeq.accepts(types)	Checks if the specified content types are acceptable.

BASIC CRUD WITH MONGODB, MONGOOSE, BODY PARSER AND LODASH

EXPRESS APPLICATION WITH MONGODB

BASIC CRUD APP WITH MONGODB

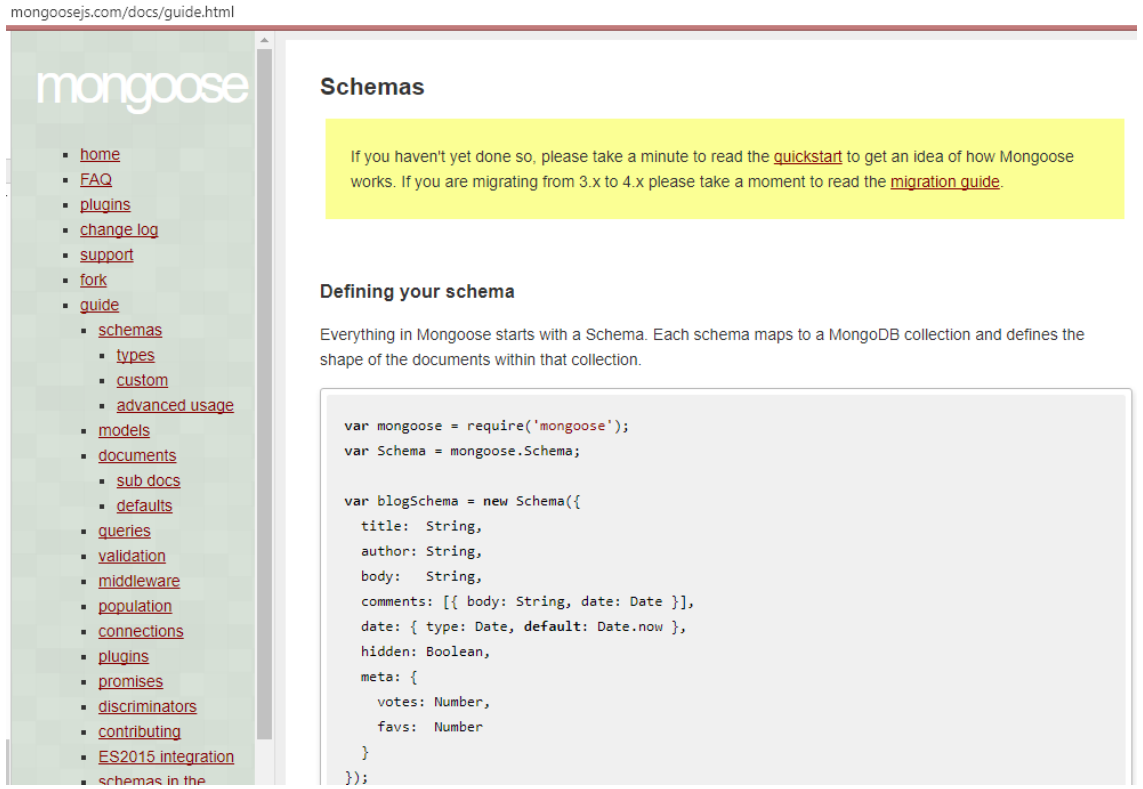
Step1: Install dependent packages

- npm install express
- npm install body-parser
- npm install mongoose
- npm install lodash

1. npm init –y
2. Npm install –save express
3. Npm install –save body-parser
4. Npm install –save mongoose
5. Npm install –save lodash

Mongoose

http://mongoosejs.com/docs/guide.html



Mongoose Schema Types

- String
- Number
- Date
- Buffer
- Mixed
- Objectid
- Array

http://mongoosejs.com/docs/schematypes.html

ORM

Name	Description
Mongoose	Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment
Waterline	An ORM extracted from the Express-based Sails web framework. It provides a uniform API for accessing numerous different databases, including Redis, mySQL, LDAP, MongoDB and Postgres
Bookshelf	Features both promise-based and traditional callback interfaces, providing transaction support, eager/nested-eager relation loading, polymorphic associations, and support for one-to-one, one-to-many and many-to-many relations. Works with Postgresql, MySQL and SQLite
Objection	Supports Postgresql, MySQL and SQLite
Sequelize	A promise based ORM for Node.js and io.js. It supports PostgreSQL,MySQL, MariaDB, SQLite and MSSQL and features solid transaction support, relations, read replication and more.

Popular packages to use with expressjs

Name	Description
Body-parser	Request payload
Compression	Gzip
Connect-timeout	Set request timeout
Cookie-parser	Cookies
Cookie-session	Session via cookies store
Csurf	CSRF
Errorhandler	Errorhandler
Forever	
Nodemon	
pm2	
Qs	Analogous to query

Name	Description
Express-session	
Method-override	
Morgan	Server logs
Response-time	
Serve-index	
Serve-static	Static content
vhost	
Cookies	
Keygrip	
Connect-multiparty	
Connect-busboy	

Popular packages to use with expressjs

Name	Description
Express-validator	
Less	
Passport	Authentication library
Helmet	Security headers
Connect-cors	CORS
Connect-redis	
Nconf	
Winston	
nunjucks	
Node-bunyan	
dotenv	

Name	Description
backpack	
Multer	

BASIC CRUD APP WITH MONGODB

Step2: create index.js

- Create index.js for supporting CORS(cross origin resource sharing) support.
- add server code to render the API
- add mongoose connection instance code.

```
//express application
var express = require('express');
//create an instance
var app = express();
// allow cross origin resource sharing
app.use(function(request,response,next){
    response.header("Access-Control-Allow-Origin","*");
    response.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    next();
});
//bodyparser package is used for parsing the rest client to post data
var bodyParser = require('body-parser');
//database connectivity with mongodb
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/Sycliq/Customers');
//body parser package
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
    extended:true
}));
var customers = require('./customer_routes.js')(app);
//get request operations
app.get('/',function(request,response){
    response.json({message:"SycliQ Geospatial Analytics Express API"});
});
var server = app.listen(9555,function(){
    console.log('Server running at http://localhost:9555');
});
var db=mongoose.connection;
db.on('error',console.error.bind(console,'connection error:'));
db.once('open',function(){
    // it is connected to the mongodb database
    console.log("Successfully connected to MongoDB");
});
```

BASIC CRUD APP WITH MONGODB

Step 3: create customer_model.js

- Create customer_model.js file and export the model using mongoose.model

```
//customer model
var mongoose = require('mongoose');
var customerSchema=mongoose.Schema({
    customerId:String,
    customerName:String,
    customerType:String,
    customerLocation:String,
    customerEmail:String
});
module.exports=mongoose.model('Customer',customerSchema);
```

BASIC CRUD APP WITH MONGODB

Step 4: create customer_routes.js

- Create customer_routes.js mapping the CRUD operations.
- **CREATE A NEW CUSTOMER**

```
//utility library required for processing
var lodash=require('lodash');
var Customer =require('./customer_model.js');
module.exports=function(app){
    /**Create */
    app.post('/addcustomer',function(request,response){

        var newCustomer = new Customer(request.body);
        newCustomer.save(function(err){
            if(err){
                response.json({
                    info:"Error::Customer Creation",error:err
                });
            };
            response.json({
                info:"Customer created Successfully"
            });
        });
    });
}
```

BASIC CRUD APP WITH MONGODB

Step 5: create customer_routes.js

- Create customer_routes.js mapping the CRUD operations.
- **READ ALL CUSTOMERS**

```
/* Read ALL CUSTOMERS*/
app.get('/getallcustomers', function (request, response) {
    Customer.find(function (err, customers) {
        if (err) {
            response.json({ info: 'error during find Customer', error: err });
        };

        response.json({ Customers: customers });
    });
});
```

BASIC CRUD APP WITH MONGODB

Step 6: create customer_routes.js

- Create customer_routes.js mapping the CRUD operations.
- **READ CUSTOMER BY ID**

```
/* GET CUSTOMER BY ID*/
app.get('/customer/:id', function (request, response) {
    Customer.findById(request.params.id, function (err, customer) {
        if (err) {
            response.json({ info: 'error during find customer', error: err });
        };
        if (customer) {

            setTimeout(function () {
                response.json({ info: 'customer found successfully', data: customer });
            }, 10000);
        } else {
            response.json({ info: 'customer not found' });
        }
    });
});
```

BASIC CRUD APP WITH MONGODB

Step 7: create customer_routes.js

- Create customer_routes.js mapping the CRUD operations.
- **UPDATE CUSTOMER BY ID**

```
/* Update */
app.put('/customer/:id', function (request, response) {
    Customer.findById(request.params.id, function (err, customer) {
        if (err) {
            response.json({ info: 'error during find customer', error: err });
        };
        if (customer) {
            _.merge(customer, request.body);
            customer.save(function (err) {
                if (err) {
                    response.json({ info: 'error during customer update', error: err });
                };
                response.json({ info: 'customer updated successfully' });
            });
        } else {
            response.json({ info: 'customer not found' });
        }

    });
});
```

BASIC CRUD APP WITH MONGODB

Step 8: create customer_routes.js

- Create customer_routes.js mapping the CRUD operations
- **UPDATE CUSTOMER BY ID**

```
/* Update */
app.put('/customer/:id', function (request, response) {
    Customer.findById(request.params.id, function (err, customer) {
        if (err) {
            response.json({ info: 'error during find customer', error: err });
        };
        if (customer) {
            _.merge(customer, request.body);
            customer.save(function (err) {
                if (err) {
                    response.json({ info: 'error during customer update', error: err });
                };
                response.json({ info: 'customer updated successfully' });
            });
        } else {
            response.json({ info: 'customer not found' });
        }

    });
});
```

BASIC CRUD APP WITH MONGODB

Step 9: create customer_routes.js

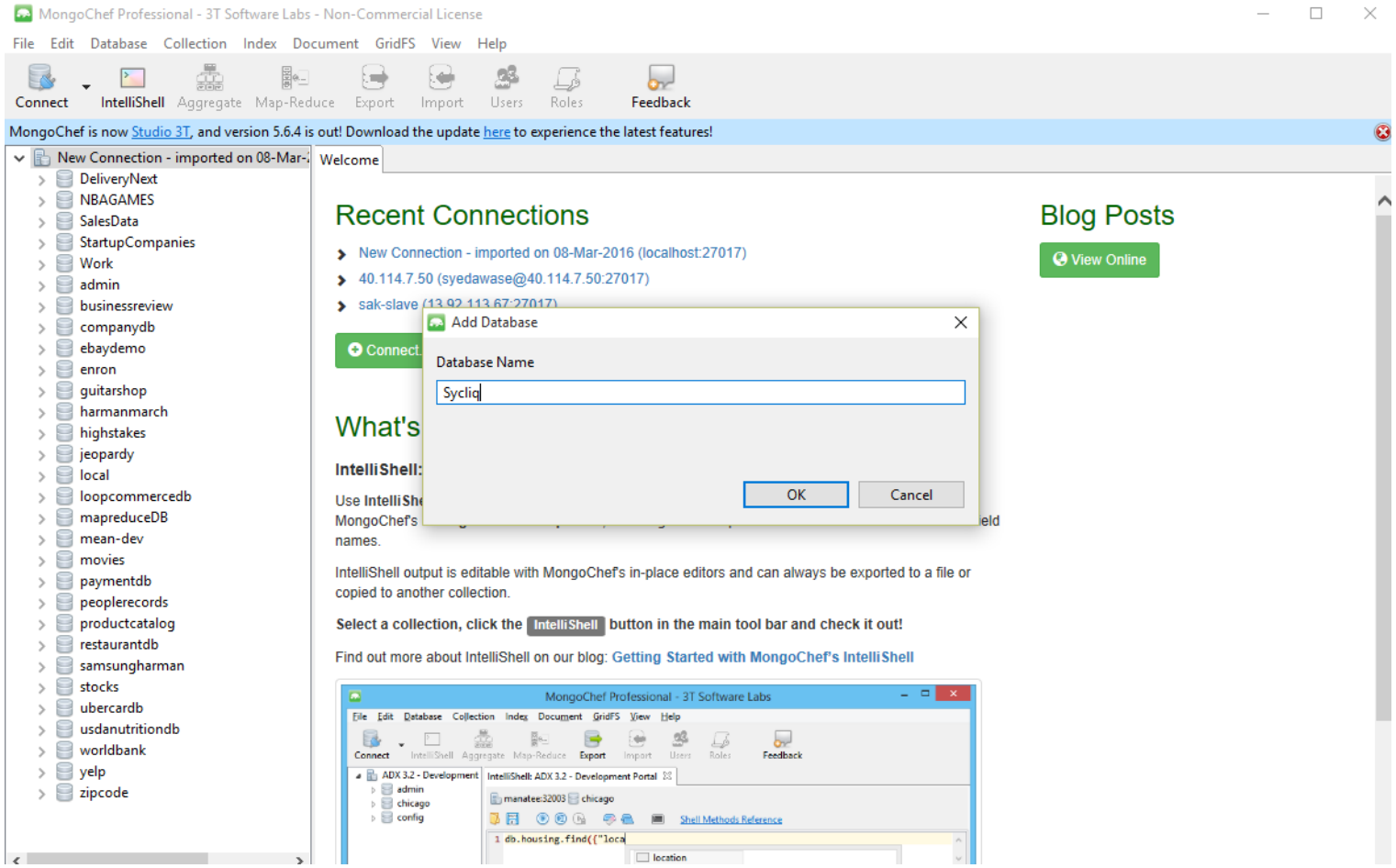
- Create customer_routes.js mapping the CRUD operations.
- **DELETE CUSTOMER BY ID**

```
/* Delete */
app.delete('/customer/:id', function (request, response) {
    Customer.findByIdAndRemove(request.params.id, function (err) {
        if (err) {
            response.json({ info: 'error during remove customer', error: err });
        };
        response.json({ info: 'customer removed successfully' });
    });
});
```

BASIC CRUD APP WITH MONGODB

Step 10: create mongodb database

- Create a mongodb database named **sycliq**

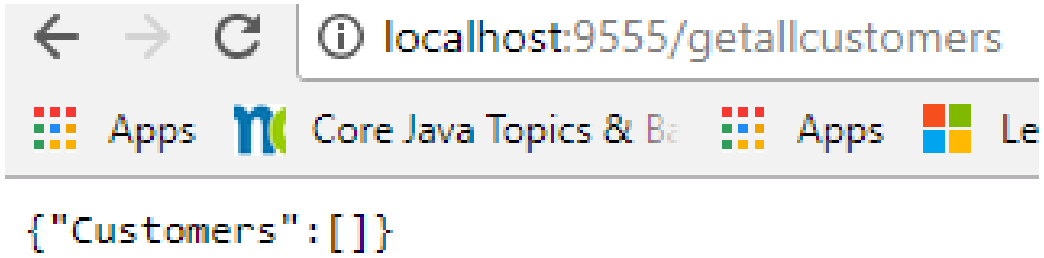


BASIC CRUD APP WITH MONGODB

Step 11: run your application.

- Run your application using **node index.js**

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\06-basic-apidemo> node index.js
(node:2648) DeprecationWarning: `open()` is deprecated in mongoose >= 4.11.0, use `ope
ongoosejs.com/docs/connections.html#use-mongo-client
Server running at http://localhost:9555
Successfully connected to MongoDB
```

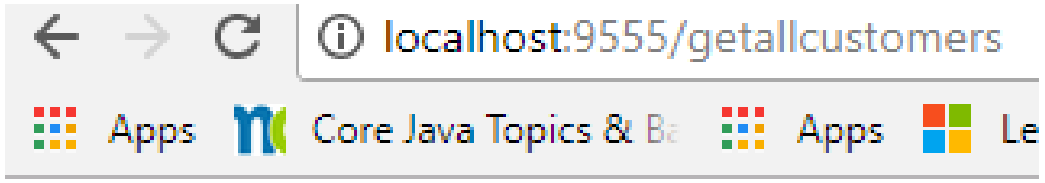


BASIC CRUD APP WITH MONGODB

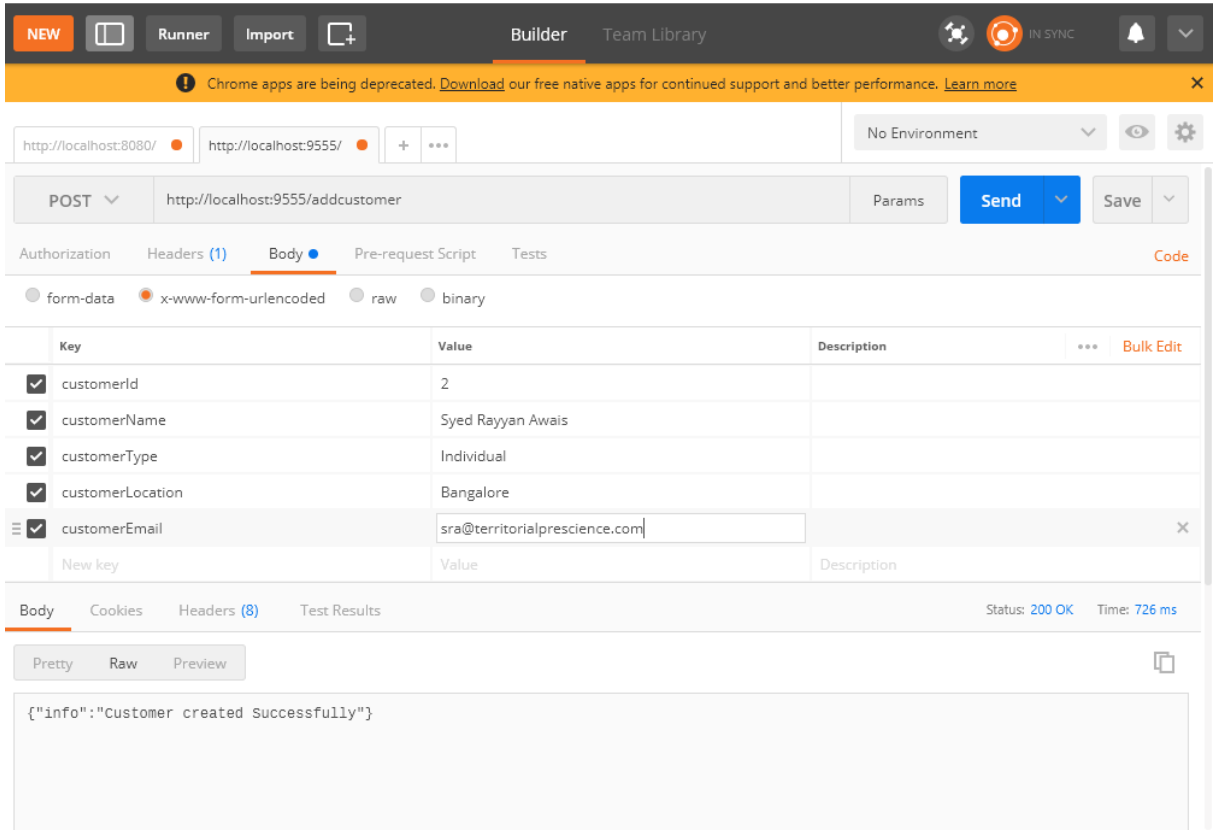
Step 12: post data to using postman

- Run your application using **node index.js**
- **Open "postman" advanced restclient**

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\06-basic-apidemo> node index.js
(node:2648) DeprecationWarning: `open()` is deprecated in mongoose >= 4.11.0, use `ope
ongoosejs.com/docs/connections.html#use-mongo-client
Server running at http://localhost:9555
Successfully connected to MongoDB
```




```
{"Customers":[]}
```

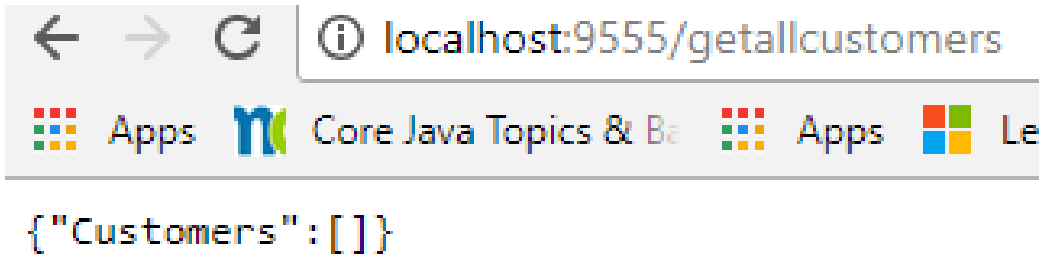


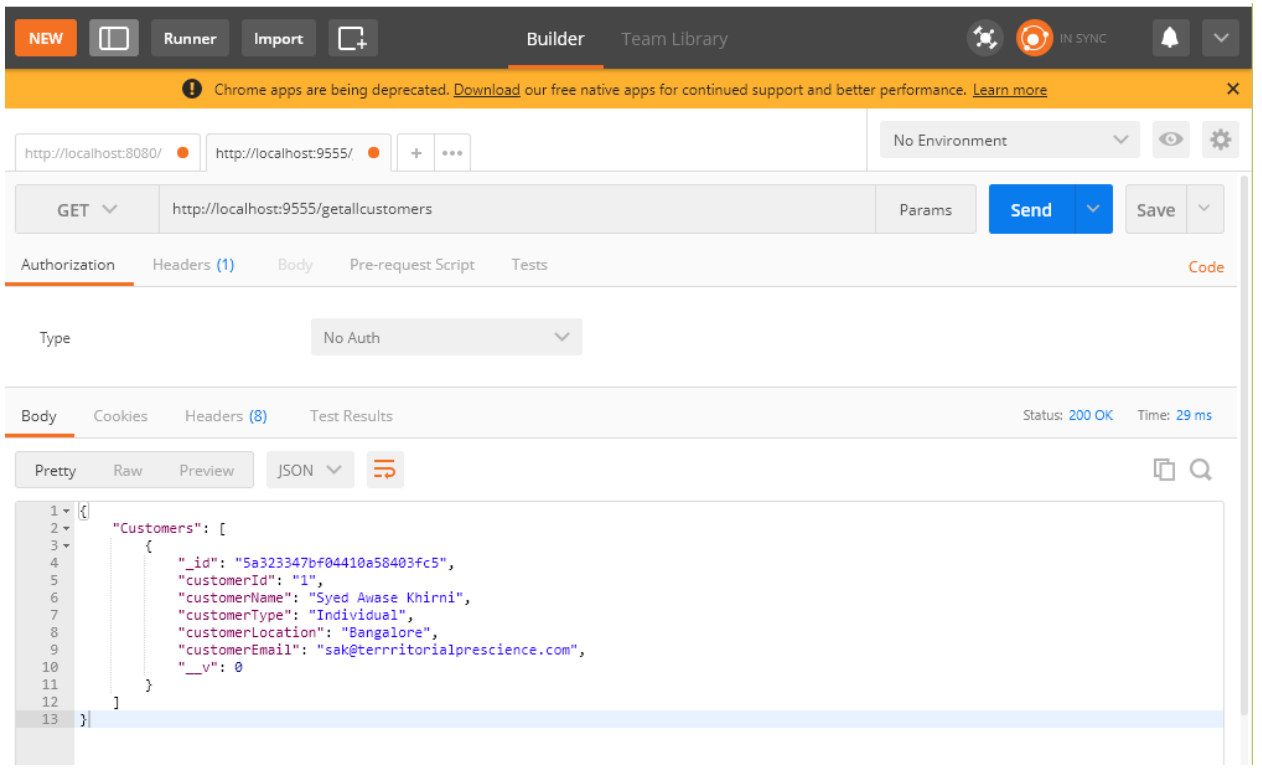
BASIC CRUD APP WITH MONGODB

Step 13: post data to using postman

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\06-basic-apidemo> node index.js
(node:2648) DeprecationWarning: `open()` is deprecated in mongoose >= 4.11.0, use `ope
ongoosejs.com/docs/connections.html#use-mongo-client
Server running at http://localhost:9555
Successfully connected to MongoDB
```

- Run your application using **node index.js**
- **Now let us list all customers**





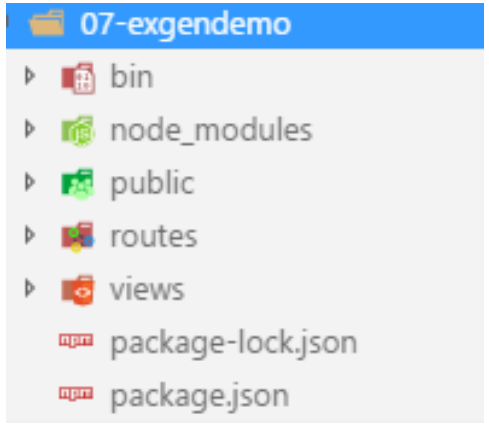
SCAFFOLDING APPLICATION USING EXPRESS

EXPRESS SCAFFOLDING APPLICATION

EXPRESS SCAFFOLD

Step 1:

•Scaffolding your application using express.




```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\07-exgendemo> npm install --save express
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\07-exgendemo> express
destination is not empty, continue? [y/N] y

   create : .
   create : ./package.json
   create : ./app.js
   create : ./public/javascripts
   create : ./public/images
   create : ./public
   create : ./routes
   create : ./routes/index.js
   create : ./routes/users.js
   create : ./views
   create : ./views/index.jade
   create : ./views/layout.jade
   create : ./views/error.jade
   create : ./public/stylesheets
   create : ./public/stylesheets/style.css
   create : ./bin
   create : ./bin/www

   install dependencies:
     > cd . && npm install

   run the app:
     > SET DEBUG=07-exgendemo:* & npm start
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\07-exgendemo> npm install --save
up to date in 0.912s
```

EXPRESS SCAFFOLD

Step 2:

- It has installed all the package dependencies required for building REST API

```
{
  "name": "07-exgendemo",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.13.2",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "~4.13.1",
    "jade": "~1.11.0",
    "morgan": "~1.6.1",
    "serve-favicon": "~2.3.0"
  }
}
```

EXPRESS SCAFFOLD

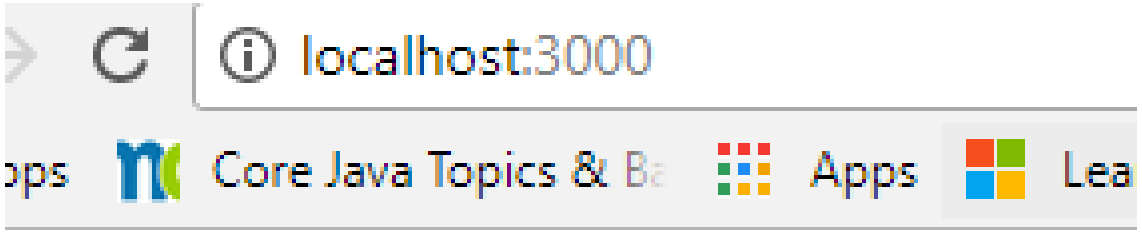
Step 3:

- It has installed all the package dependencies required for building REST API

```
{
  "name": "07-exgendemo",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.13.2",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "~4.13.1",
    "jade": "~1.11.0",
    "morgan": "~1.6.1",
    "serve-favicon": "~2.3.0"
  }
}
```

EXPRESS SCAFFOLD

Step 4:



•Now run your application using **npm start**

EXPRESS SCAFFOLD

Step 5:

- Install mysql drivers and CORS for the express scaffolded project.

```
npm install mysql --save
```

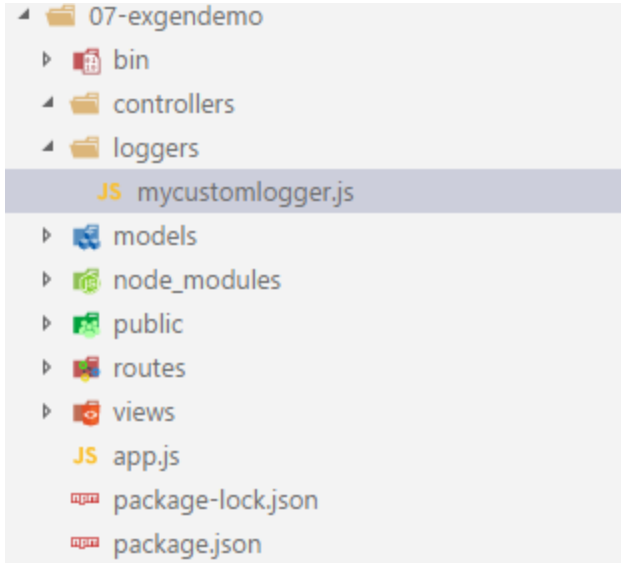
```
npm install cors --save
```

Middleware

- app.use function adds middleware to the application.
- **Express.static('foldername')** is used to serve static middleware file from the "foldername"
- Middleware functions are executed **sequentially that access request and response object.**
- It performs actions such as validation, authentication, data parsing.

```
var express = require('express');
var app =express();

//static folder to render static files
app.use(express.static('public'));
app.listen(9555);
```

Creating a custom logger




```
//mycustomerlogger module
module.exports = function(request,response,next){
    var start =+new Date();
    var stream = process.stdout;
    var url=request.url;
    var method = request.method;

    response.on('finish',function(){
        var duration = +new Date() -start;
        var message = method +'to'+url+'\ntook'+duration+'ms\n\n';
        stream.write(message);
    });
    next();
}
```

Consuming the logger module:
var logger = require('./mycustomlogger.js'); app.use(logger)

Product and Customer Model

Store Model

```
//store model
var mongoose = require('mongoose');
var storeSchema =mongoose.Schema({
    storeName:String,
    storeId:mongooose.Schema.ObjectId,
    storeLocation:String,
    productsInStore:[{
        type:Schema.Types.ObjectId,
        ref:'Product'
    }]
});

module.exports = mongoose.model('Store',storeSchema);
```

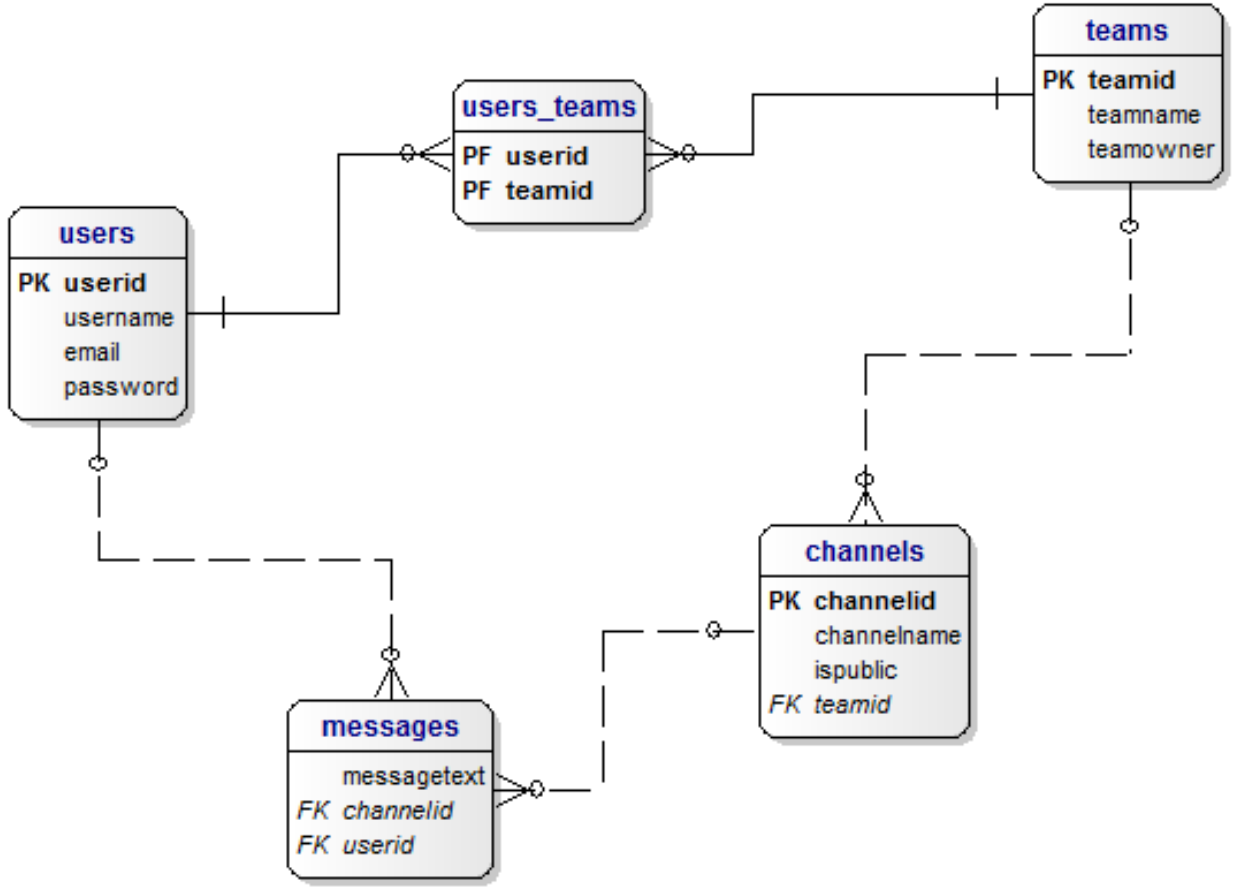
Product Model

```
//Product Model
var product = require('product');
var productSchema=mongoose.Schema({
    productId: mongooose.Schema.ObjectId,
    productName:String,
    productDesc:String,
    productRating:String,
    productPrice:number,
    storeforProduct:[{
        type:Schema.Types.ObjectId,
        ref:'Store'
    }]
});

module.exports = mongoose.model('Product',
productSchema);
```

SLACK APPLICATION WITH POSTGRESQL + SEQUELIZE

EXPRESS WITH SEQUELIZE +POSTGRES

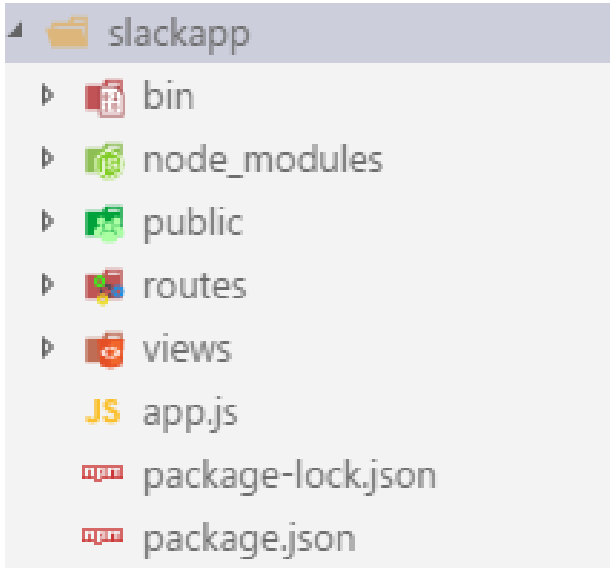
SLACK ERD



SLACK APP WITH POSTGRES +SEQUELIZE

Step 1: scaffold your express application.

- Create an express scaffold for slackapp




```
npm install -g express-generator
```

```
Express slackapp
```

```
cd slackapp
```

```
npm install –save
```

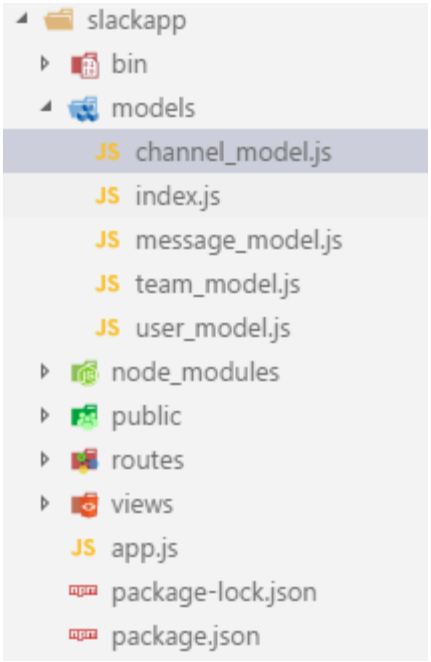
```
npm start
```

```
SET DEBUG=slackapp:* & npm start
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 2: create models folder

•Create models folders with the following files.



- mkdir models
- Touch index.js
- Touch user_model.js
- Touch message_model.js
- Touch team_model.js
- Touch channel_model.js
- npm install –g sequelize
- npm install –g pg

SLACK APP WITH POSTGRES +SEQUELIZE

Step 3: models/index.js

- Create models folders with the following files.

```
//http://docs.sequelizejs.com/
var Sequelize = require('sequelize');
//connect to the postgres database
var dbSequelize = new Sequelize('slackdb', 'postgres', 'postgres', {
    host: 'localhost',
    dialect: 'postgres'
});
//register the models
var slackModels ={
    User:dbSequelize.import('./user_model'),
    Message:dbSequelize.import('./message_model'),
    Team:dbSequelize.import('./team_model'),
    Channel:dbSequelize.import('./channel_model')
};
//iterate through the object keys and associate
Object.keys(slackModels).forEach((modelName)=>{
    if('associate'in slackModels[modelName]){
        slackModels[modelName].associate(slackModels);
    }
});
slackModels.dbSequelize=dbSequelize;
slackModels.Sequelize=Sequelize;
module.exports =slackModels;
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 4: models/user_model

```
//User Model
module.exports = function (dbSequelize, DataTypes) {
    var User = dbSequelize.define("user", {
        username: {
            type: DataTypes.STRING,
            unique: true
        },
        email: {
            type: DataTypes.STRING,
            unique: true
        },
        password: {
            type: DataTypes.STRING
        }
    });

    User.associate = function (slackModels) {
        User.belongsToMany(slackModels.Team, {
            through:'member',
            foreignKey:'userid'
        });
    }
    return User;
};
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 5: models/team_model

```
//team_model.js
module.exports = function (dbSequelize, DataTypes) {
    var Team = dbSequelize.define("team", {
        teamname: {
            type: DataTypes.STRING,
            unique: true
        }
    });

    Team.associate = function (slackModels) {
        Team.belongsToMany(slackModels.User, {
            through: 'member',
            foreignKey: 'teamid'
        });
        Team.belongsTo(slackModels.User,{
            foreignKey:'userid'
        });
    };
    return Team;
};
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 6: models/channel_model

```
//channel_model.js
module.exports = function (dbSequelize, DataTypes) {
    var Channel = dbSequelize.define("channel", {
        channelname: {
            type: DataTypes.STRING,
            public:DataTypes.BOOLEAN
        }
    });

    Channel.associate = function (slackModels) {
        //one to many
        Channel.belongsTo(slackModels.Channel, {
            foreignKey: 'teamid'
        });
        Channel.belongsTo(slackModels.User, {
            foreignKey: 'userid'
        });
    };
    return Channel;
};
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 7: models/message_model

```
//message_model.js
module.exports = function (dbSequelize, DataTypes) {
    var Message = dbSequelize.define("message", {
        msgtext: {
            type: DataTypes.STRING
        }
    });

    Message.associate  = function (slackModels) {
        //one to many
        Message.belongsTo(slackModels.Channel, {
           foreignKey: 'channelid'
        });
        Message.belongsTo(slackModels.User, {
            foreignKey: 'userid'
        });
    };
    return Message;
};
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 8: register the models in bin/www and invoke

- Run you application using **npm start**

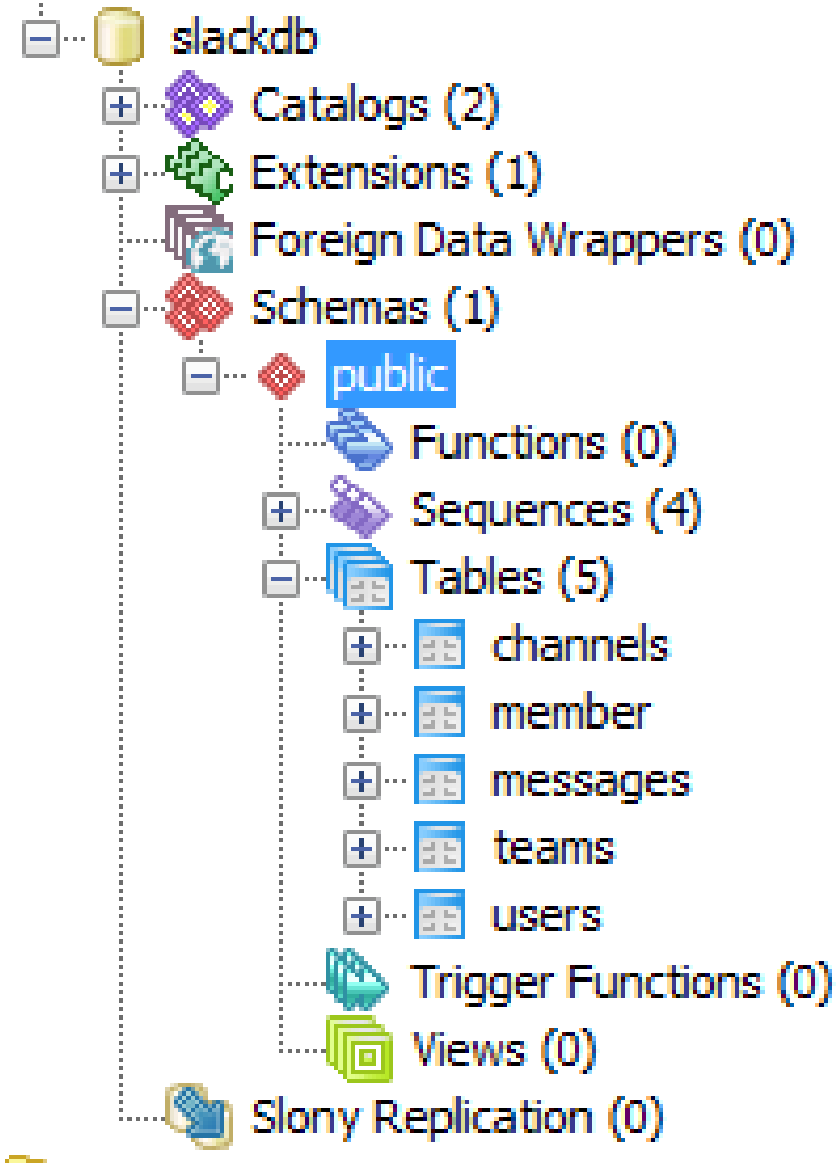
```
//reference to the directory of models
var slackModels = require('../models/');

//forcefully dropping and recreating database
slackModels.dbSequelize.sync({ force:true}).then(function(){
//slackModels.dbSequelize.sync().then(function(){
  server.listen(port);
  server.on('error', onError);
  server.on('listening', onListening);

});
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\10-express-sequelize\slackapp> npm start

> slackapp@0.0.0 start E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\10-express-sequelize\slackapp
> node ./bin/www

sequelize deprecated String based operators are now deprecated. Please use Symbol based operators for better security, read
ual/tutorial/querying.html#operators node_modules\sequelize\lib\sequelize.js:236:13
Executing (default): DROP TABLE IF EXISTS "member" CASCADE;
Executing (default): DROP TABLE IF EXISTS "teams" CASCADE;
Executing (default): DROP TABLE IF EXISTS "messages" CASCADE;
Executing (default): DROP TABLE IF EXISTS "channels" CASCADE;
Executing (default): DROP TABLE IF EXISTS "users" CASCADE;
Executing (default): DROP TABLE IF EXISTS "users" CASCADE;
Executing (default): CREATE TABLE IF NOT EXISTS "users" ("id"   SERIAL , "username" VARCHAR(255) UNIQUE, "email" VARCHAR(25
atedAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, UNIQUE ("username"), UNIQUE ("ema
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, a
ray_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix,
 AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'users' GROUP BY i.relname, ix.indexr
x.indkey ORDER BY i.relname;
Executing (default): DROP TABLE IF EXISTS "channels" CASCADE;
Executing (default): CREATE TABLE IF NOT EXISTS "channels" ("id"   SERIAL , "channelname" VARCHAR(255), "createdAt" TIMESTA
TIMESTAMP WITH TIME ZONE NOT NULL, "teamid" INTEGER REFERENCES "channels" ("id") ON DELETE SET NULL ON UPDATE CASCADE, "use
N DELETE SET NULL ON UPDATE CASCADE, PRIMARY KEY ("id"));
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 9: verify the database in postgresql

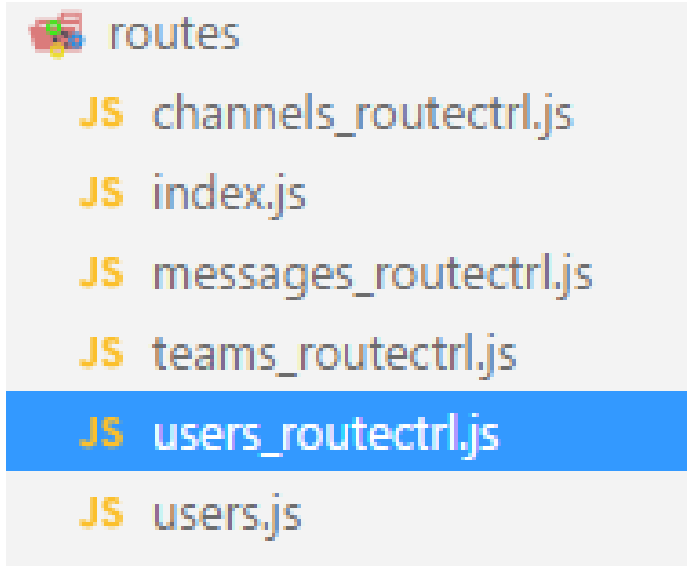
• open pgadmin-iii and check database for the tables created.



SLACK APP WITH POSTGRES +SEQUELIZE

Step 10: create routes for all the models

- register the routes in app.js




```
var index = require('./routes/index');
var users = require('./routes/users_routectrl');
var teams = require('./routes/teams_routectrl');
var messages = require('./routes/messages_routectrl');
var channels = require('./routes/channels_routectrl');
```

```
app.use('/', index);
app.use('/users', users);
app.use('/messages',messages);
app.use('/teams',teams);
app.use('/channels',channels);
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 11: users_routectrl.js

- create the routes for users: CRUD operations.

```
var express = require('express');
var router = express.Router();
var Sequelize = require('sequelize');
var dbSequelize = new Sequelize('slackdb', 'postgres', 'postgres', {
    host: 'localhost',
    dialect: 'postgres'
});
var userModel = dbSequelize.import('../models/user_model');

/* GET ALL users listing. */
router.get('/allusers', function (req, res, next) {
    //res.send('<h1>All Users</h1>');
    var code = 500;
    var message = "SycliQ SLACK Server Error";
    var result = "";
    var page = req.query.page || 1;
    var limit = req.query.limit || 10;
    var offset = (page - 1) * limit;

    userModel.findAndCountAll({
        offset: +offset,
        limit: +limit
    }).then(function (result) {
        code = 200;
        message = 'OK';
        res.json({
            code: code,
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 12: teams_routectrl.js

- create the routes for teams: CRUD operations.

```
var express = require('express');
var router = express.Router();
var Sequelize = require('sequelize');
var dbSequelize = new Sequelize('slackdb', 'postgres', 'postgres', {
    host: 'localhost',
    dialect: 'postgres'
});
var teamModel = dbSequelize.import('../models/team_model');


/* GET ALL teams listing. */
router.get('/allteams', function (req, res, next) {

    var code = 500;
    var message = "SycliQ SLACK Server Error";
    var result = "";
    var page = req.query.page || 1;
    var limit = req.query.limit || 10;
    var offset = (page - 1) * limit;

    teamModel.findAndCountAll({
        offset: +offset,
        limit: +limit
    }).then(function (result) {
        code = 200;
        message = 'OK';
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 13: messages_routectrl.js

- create the routes for messages: CRUD operations.

```
var express = require('express');
var router = express.Router();
var Sequelize = require('sequelize');
var dbSequelize = new Sequelize('slackdb', 'postgres', 'postgres', {
    host: 'localhost',
    dialect: 'postgres'
});
var messageModel = dbSequelize.import('../models/message_model');


/* GET ALL teams listing. */
router.get('/allmessages', function (req, res, next) {

    var code = 500;
    var message = "SycliQ SLACK Server Error";
    var result = "";
    var page = req.query.page || 1;
    var limit = req.query.limit || 10;
    var offset = (page - 1) * limit;

    messageModel.findAndCountAll({
        offset: +offset,
        limit: +limit
    }).then(function (result) {
        code = 200;
        message = 'OK';
        res.json({
            code: code,
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 14: channels_routectrl.js

- create the routes for channels: CRUD operations.

```
var express = require('express');
var router = express.Router();
var Sequelize = require('sequelize');
var dbSequelize = new Sequelize('slackdb', 'postgres', 'postgres', {
    host: 'localhost',
    dialect: 'postgres'
});
var messageModel = dbSequelize.import('../models/message_model');


/* GET ALL teams listing. */
router.get('/allmessages', function (req, res, next) {

    var code = 500;
    var message = "SycliQ SLACK Server Error";
    var result = "";
    var page = req.query.page || 1;
    var limit = req.query.limit || 10;
    var offset = (page - 1) * limit;

    messageModel.findAndCountAll({
        offset: +offset,
        limit: +limit
    }).then(function (result) {
        code = 200;
        message = 'OK';
        res.json({
            code: code,
```

SLACK APP WITH POSTGRES +SEQUELIZE

Step 15: route url access

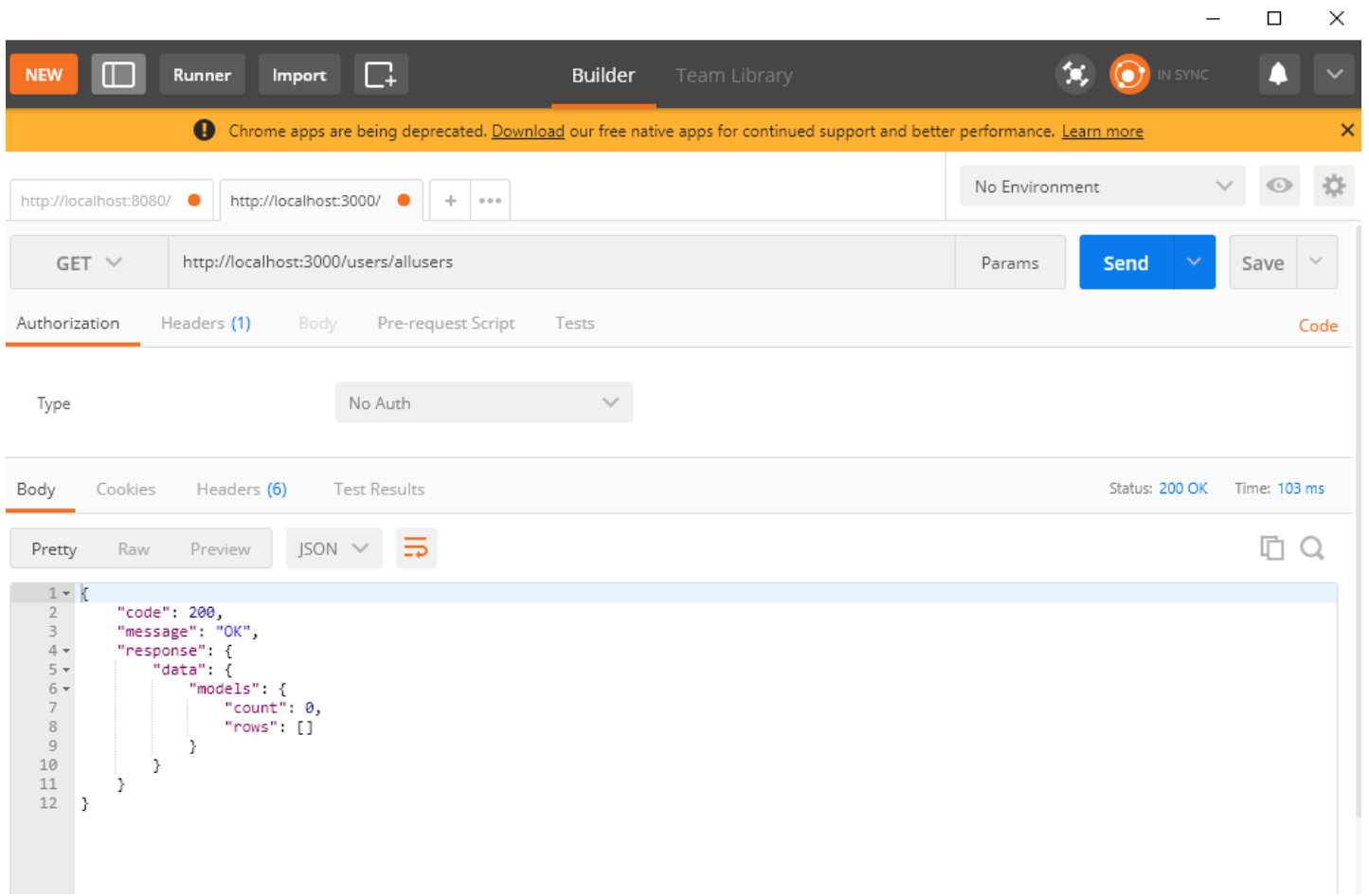
- Npm start

http://localhost:3000/	index
http://localhost:3000/users/getallusers	List all users
http://localhost:3000/users/createuser	Create user
http://localhost:3000/users/update/:userid	Update user
http://localhost:3000/users/delete/:userid	Delete user
http://localhost:3000/teams/getallteams	List all teams
http://localhost:3000/teams/createteam	Create team
http://localhost:3000/teams/update/:teamid	Update team
http://localhost:3000/teams/delete/:teamid	Delete team
http://localhost:3000/channels/getallchannels	List all channels
http://localhost:3000/channels/createchannel	Create channel
http://localhost:3000/channels/update/:channelid	Update channel
http://localhost:3000/channels/delete/:channelid	Delete channel

SLACK APP WITH POSTGRES +SEQUELIZE

Step 16: route url access

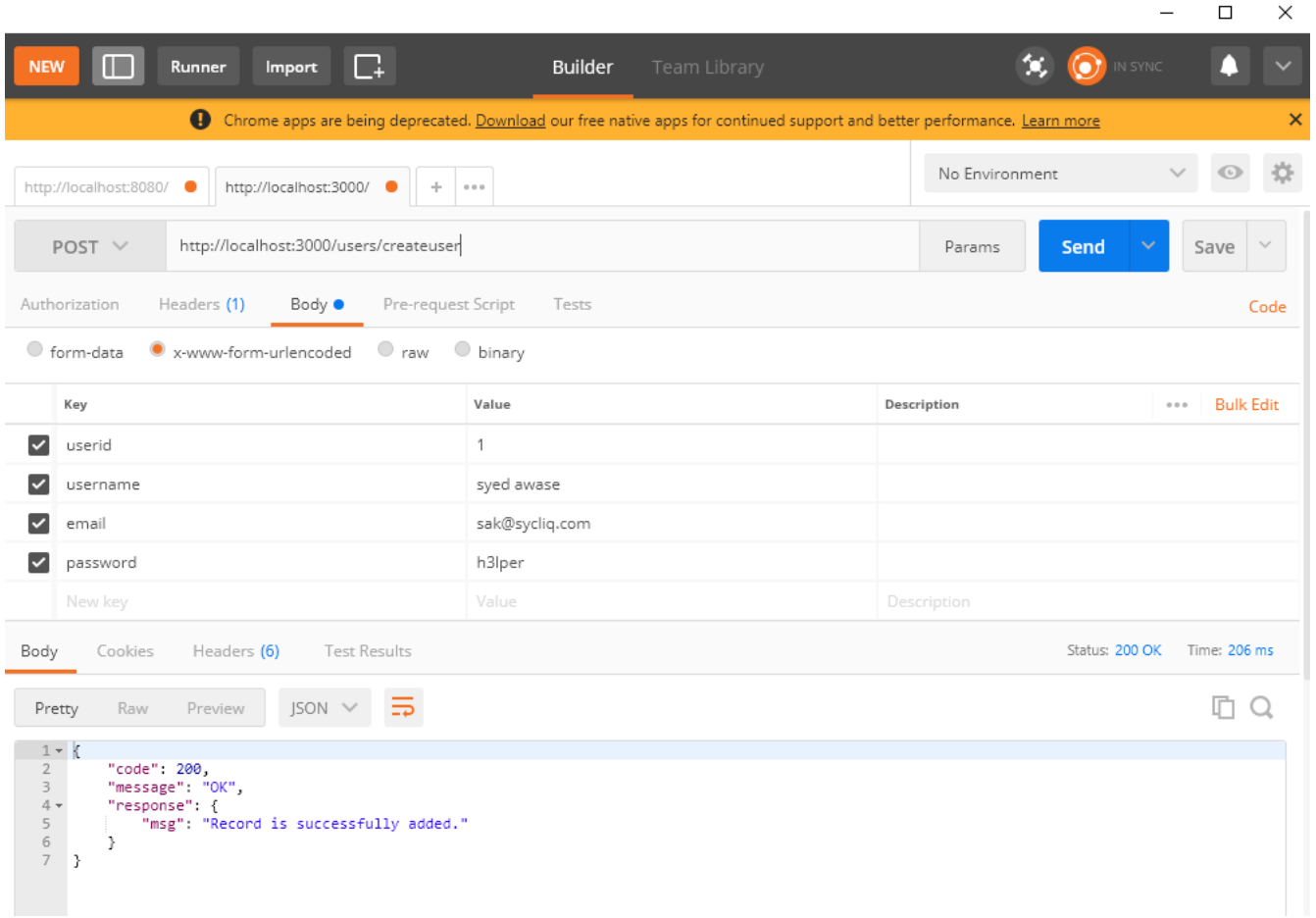
- Npm start



SLACK APP WITH POSTGRES +SEQUELIZE

Step 17: route url access

- Create user



https://github.com/cdimascio/generator-express-no-stress

EXPRESS GENERATOR WITH NO STRESS

Express Generator with No Stress

Step 1:

- installing generator-express-no-stress

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\09-express-nostress> npm install -g yo generator-express-no-stress
[ ..................] - fetchMetadata: sill resolveWithNewModule gh-got@5.0.0 checking installable statusssle statusble status
Mode                 LastWriteTime         Length Na
me
----                 -------------         ------ --
--
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsde
mo\09-express-nostress> npm install -g yo generator
-express-no-stress
[ ..................] | fetchMetadata: sill resolveW
ithNewModule is-plain-obj@1.1.0 checking installabl
e statustatusble status[ ..................] | fetch
C:\Users\SyedAwase\AppData\Roaming\npm\yo-complete
-> C:\Users\SyedAwase\AppData\Roaming\npm\node_modu
les\yo\lib\completion\index.js
C:\Users\SyedAwase\AppData\Roaming\npm\yo -> C:\Use
rs\SyedAwase\AppData\Roaming\npm\node_modules\yo\li
b\cli.js

> spawn-sync@1.0.15 postinstall C:\Users\SyedAwase\
AppData\Roaming\npm\node_modules\generator-express-
no-stress\node_modules\spawn-sync
> node postinstall


> yo@2.0.0 postinstall C:\Users\SyedAwase\AppData\R
oaming\npm\node_modules\yo
> yodoctor


Yeoman Doctor
Running sanity checks on your system

√ Global configuration file is valid
√ NODE_PATH matches the npm root
√ Node.js version
√ No .bowerrc file in home directory
√ No .yo-rc.json file in home directory
√ npm version
```

Step 3:

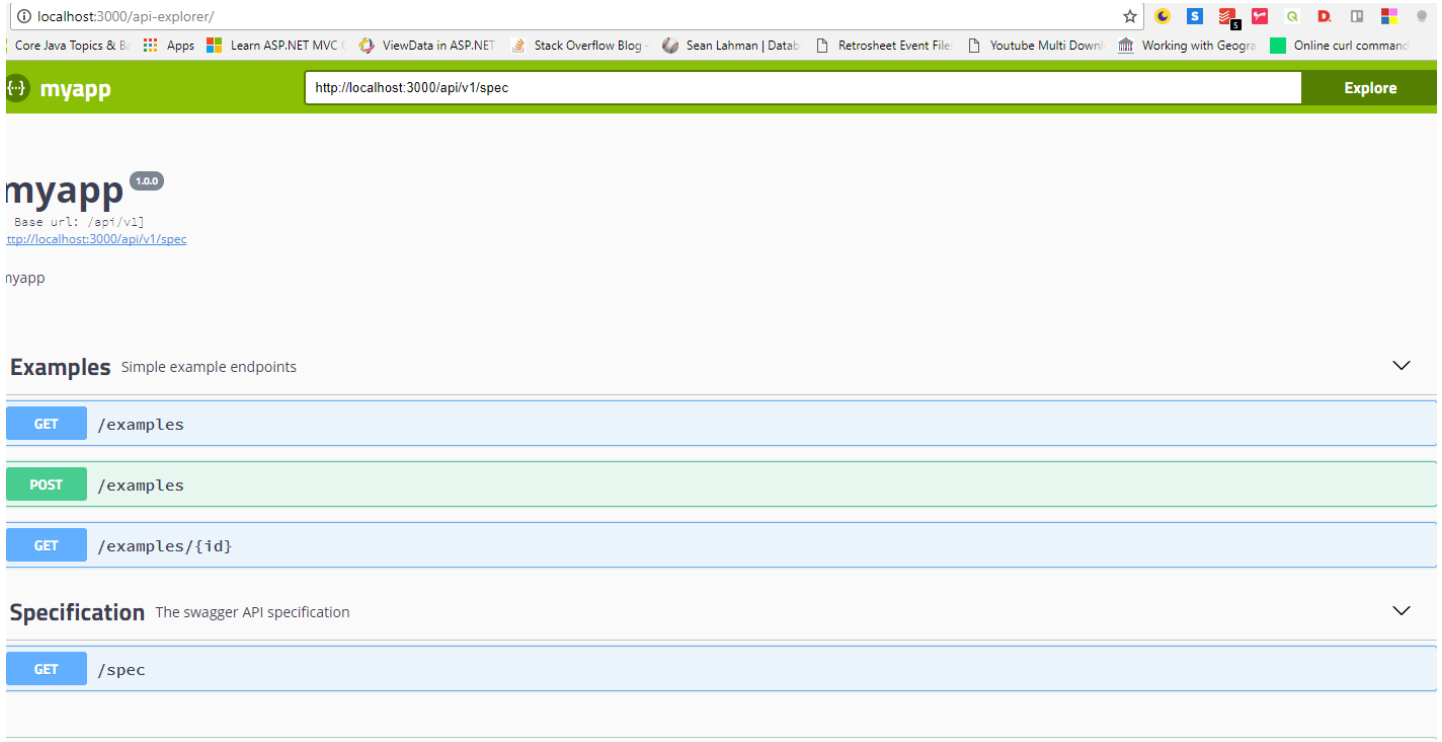
- scaffolding express

```
mo\09-express-nostress> yo express-no-stress myapp
? App description [My cool app] myapp
? API Root [/api/v1]
? Version [1.0.0]
   create myapp\backpack.config.js
   create myapp\build.js
   create myapp\package.json
   create myapp\public\api-explorer\favicon-16x16.png
   create myapp\public\api-explorer\favicon-32x32.png
   create myapp\public\api-explorer\index.html
   create myapp\public\api-explorer\oauth2-redirect.html
   create myapp\public\api-explorer\swagger-ui-bundle.js
   create myapp\public\api-explorer\swagger-ui-bundle.js.map
   create myapp\public\api-explorer\swagger-ui-standalone-preset.js
   create myapp\public\api-explorer\swagger-ui-standalone-preset.js.map
   create myapp\public\api-explorer\swagger-ui.css
   create myapp\public\api-explorer\swagger-ui.css.map
   create myapp\public\api-explorer\swagger-ui.js
   create myapp\public\api-explorer\swagger-ui.js.map
   create myapp\public\index.html
   create myapp\README.md
   create myapp\server\api\controllers\examples\controller.js
   create myapp\server\api\controllers\examples\router.js
   create myapp\server\api\services\examples.db.service.js
   create myapp\server\api\services\examples.service.js
   create myapp\server\common\env.js
   create myapp\server\common\logger.js
```

Step 3:

- Running your application
- Cd myapp
- Npm run dev
- Running in production mode
- Npm run compile
- Npm start

http://localhost:3000/api-explorer/



SYED AWASE

EXPRESS WITH ES6

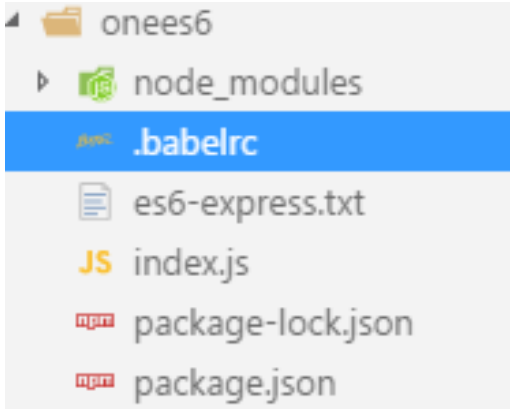
SYED AWASE

EXPRESS WITH ES6:EXAMPLE ONE

Express ES6 DEMO

Step 1:

- Scaffold your application to create an es6 express application
- Create **".babelrc"**




```
npm init -y
```

```
npm install –save express
```

```
npm install –save babel-cli babel-preset-es2015
```

.babelrc

```
{
    "presets": ["es2015"]
}
```

Express ES6 DEMO

Step 2:

- create index.js
- Add “start script” to package.json
-

```
"scripts": {
  "start":"babel-node index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

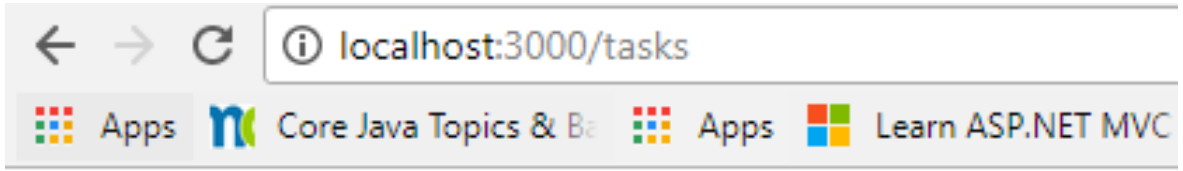
```
import express from 'express';
const PORT=3000;
const app=express();
/**get operations */
app.get("/",(request,response)=> response.json({
    status:"MyTask API"
}));
//formatting the results
app.set('json spaces',4);
app.get('/tasks',(request,response)=>{
    response.json({
        tasks:[
            {taskTitle:"Write Node.js Modules"},
            {taskTitle:"import the node.js modules"}
        ]
    });
});

app.listen(PORT,()=>console.log(`myTask API-Port ${PORT}`));
```

Express ES6 DEMO

Step 3:

- start your application
- npm start

http://localhost:3000/tasks




```
{
    "tasks": [
        {
            "taskTitle": "Write Node.js Modules"
        },
        {
            "taskTitle": "import the node.js modules"
        }
    ]
}
```

SYED AWASE

EXPRESS WITH ES6:TWO-PRODUCTLISTING

Express ES6 ProductListing

Step 1:

- Scaffold your application to create an es6 express application
- npm install –save consign
- Consign makes applications easier to develop with logical file separation and automatic script loading.
- Create **“.babelrc”**

npm init -y

npm install –save express

npm install –save babel-cli babel-preset-es2015

npm install --save sequelize sqlite3

Npm install –save consign

.babelrc

```
{
    "presets": ["es2015"]
}
```

Npm package: Consign

- It makes applications easier to develop with logical file seperation and automatic script loading.
- It is used to autoload models, routes, schemas, configs, controllers, object maps, etc.
- It conforms to Model-View-Router (MVR) pattern

- npm install consign --save

Building API with Express+Consign+ES6

- **Step 1**: Create Project manifest file package.json
  - Npm install –save
  - Create .babelrc

```
{
  "presets":["es2015"]
}
```

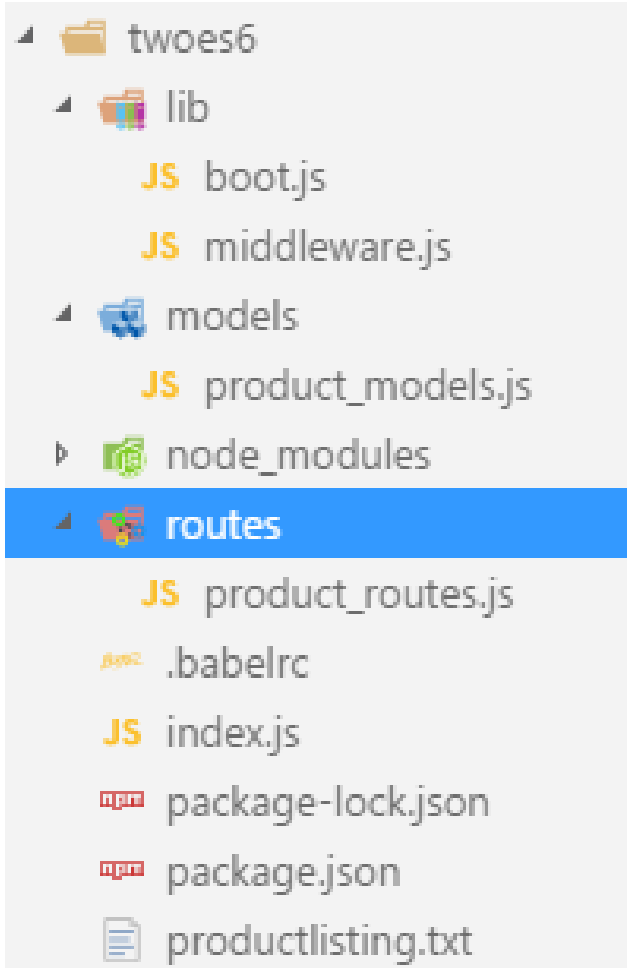
```
{
  "name": "productapi",
  "version": "1.0.0",
  "description": "webapi demo with es6,consign and express",
  "main": "index.js",
  "scripts": {
    "start": "babel-node index.js"
  },
  "keywords": ["express", "consign",
  "es6"],
  "author": "Syed Awase Khirni",
  "license": "ISC",
  "dependencies": {
    "babel-cli": "^6.5.1",
    "babel-preset-es2015": "^6.5.0",
    "consign": "^0.1.6",
    "express": "^4.13.4"
  }
}
```

Building API with Express+Consign+ES6

- **Step 2**: folder structure

Add start script to package.json

```
"scripts": {
  "start": "babel-node index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```



Building API with Express+Consign+ES6

- **Step 3**: create *models/product_models.js*

```
module.exports = app => {
    return {
        findAll: (params, callback) => {
            return callback([
                {
                    prodId: 21314,
                    proName: "LG Airconditioner",
                    prodPrice: 26000.00
                },
                {
                    prodId: 123442332,
                    proName: "Daikin Airconditioner",
                    prodPrice: 221313.00
                }
            ]);
        }
    };
};
```

Building API with Express+Consign+ES6

- **Step 4**: create *routes/product_routes.js*

```
module.exports=app=>{
    const Products=app.models.product_models;
    app.get("/products",(req,res)=>{
        Products.findAll({},(products)=>{
            res.json({products:products});
        });
    });
};
```

- **Step 5**: create *lib/middleware.js*

```
module.exports = app=>{
     app.set("port",3000);
     app.set("json spaces",4);
 };
```

Building API with Express+Consign+ES6

- **Step 6**: create *lib/boot.js*

```
module.exports = app => {
    app.listen(app.get("port"), () => {
        console.log(`Product API - Port $
        {app.get("port")}`);
    });
};
```

- **Step 7**: create *index.js*

```
import express from "express";
import consign from "consign";
const app = express();
consign()
    .include("models")
    .then("lib/middleware.js")
    .then("routes")
    .then("lib/boot.js")
    .into(app);
```

- **Step 8**: npm start
- **Step 9**: http://localhost:3000/products

ADDITIONAL PACKAGES TO BE USED WITH EXPRESS

PACKAGES TO USE WITH EXPRESS

NCONF

https://github.com/indexzero/nconf

- A configuration library for nodejs
- Used to create multiple levels of config that override each other, providing smart defaults
- Hierarchical node.js configuration with files, environment variables, command-line arguments and atomic object merging.
- Developed by faltiron

- Installation
- Npm install nconf --save

Nconf.argv(options)	Loads process.argv using yargs.
Nconf.env(options)	Loads process.env into the hierarchy
Nconf.file(options)	Loads the configuration data at options.file into the hierarchy
Nconf.defaults(options)	Loads the data in options.store into the hierarchy
Nconf.overrides(options)	Loads the data in options.store into the hierarchy

WINSTON

https://github.com/winstonjs/winston

- Winston is a multi-transport logging library.

- Installation
  - npm init -y
  - npm install –save winston

```
'use strict';
var winston = require('winston');
winston.level='debug';
winston.info('Sycliq-debugging');
winston.debug('Debugging--SycliQ Platform');
```

WINSTON LOGGIN LEVELS

https://github.com/winstonjs/winston#logging-levels

- Logging levels in winston conform to the severity ordering.
- Severity of all levels is assumed to be numerically ascending from most important to least important.
- Each level is given a specific integer priority. The higher the priority the more important the message is considered to be.

Winston logging levels	Npm logging levels
Emerg: 0	Error:0
Alert: 1	Warn:1
Crit: 2	Info: 2
Error:3	Verbose:3
Warning: 4	Debug:4
Notice:5	Silly:5
Info:6	
Debug:7	

WINSTON

Colorize winston console log output

```
'use strict';
var winston = require('winston');
var logger = new (winston.Logger)({
    transports:[
        //colorize the output to the console
        new (winston.transports.Console)({
            colorize:true
        })
    ]
});
logger.level='debug';
logger.info('SycliQ Platform');
logger.debug("SycliQ Platform Debugging");
```

Adding timestamp to the log entries.

```
'use strict';
var winston=require('winston');
var tsFormat = function(){
    new Date().toLocaleTimeString();
}

var logger = new(winston.Logger)({
    transports:[
        //colorize the output to the console
        new (winston.transports.Console)({
            timestamp:tsFormat,
            colorize:true,
        })
    ]
});

logger.level="debug";
logger.info("SycliQ Platform");
logger.debug("SycliQ Platform Debugging");
```

WINSTON: Logging to an external file

```
'use strict';
var winston=require('winston');
var fs = require('fs');
var env=process.env.NODE_ENV||'development';
var logDir='SycliQlog';
//Create the log directory if it does not exist
if(!fs.existsSync(logDir)){
    fs.mkdirSync(logDir);
}
var tsFormat=function(){
    new Date().toLocaleTimeString();
}
var logger=new(winston.Logger)({
    transports:[
        //colorize the output to the console
        new (winston.transports.Console)({
            timestamp:tsFormat,
            colorize:true,
            level:'info'
        }),
        new (winston.transports.File)({
            filename:'${logDir}/results.log',
            timestamp:tsFormat,
            level:env==='development'? 'debug':'info'
        })
    ]
});
logger.info("SycliQ Platform");
logger.warn("Sycliq Warning Message");
logger.debug("SycliQ debugging info");
```

Daily log file


```
'use strict';
const winston = require('winston');
const fs = require('fs');
const env = process.env.NODE_ENV || 'development';
const logDir = 'log';
// Create the log directory if it does not exist
if (!fs.existsSync(logDir)) {
    fs.mkdirSync(logDir);
}
const tsFormat = () => (new Date()).toLocaleTimeString();
const logger = new (winston.Logger)({
    transports: [
        // colorize the output to the console
        new (winston.transports.Console)({
            timestamp: tsFormat,
            colorize: true,
            level: 'info'
        }),
        new (require('winston-daily-rotate-file'))({
            filename: `${logDir}/-results.log`,
            timestamp: tsFormat,
            datePattern: 'yyyy-MM-dd',
            prepend: true,
            level: env === 'development' ? 'verbose' : 'info'
        })
    ]
});
logger.debug('SycliQ Platform');
logger.verbose('SycliQ Platform Info');
logger.info('SycliQ Platform-GeoAnalytics ');
logger.warn('SycliQ Platform Warning Message');
logger.error('SycliQ Platform Error');
```

Send Email:Postmark https://postmarkapp.com/pricing

https://postmarkapp.com/developer

- Postmark helps deliver and track transactional emails for web applications.
- Maximum email size is 10mb
- Only support for specific mail attachment types.

```
#!/usr/bin/env node

// Paste your API key here...
var POSTMARK_KEY = "00000000-0xxx-0xx0-x000-xx00x00xx00x";

var postmark = require("postmark")(POSTMARK_KEY);

postmark.send({
    "From": "outgoing@server.com",
    "To": "user@domain.com",
    "Subject": "This is a subject",
    "TextBody": "This is a message body"
}, function (err, to) {
    if (err) {
        console.log(err);
        return;
    }
    console.log("Email sent to: %s", to);
});
```

nsp:check for vulnerable node.js modules

- Npm install –g snyk
- Snyk test

```
npm install -g nsp
```

```
nsp audit-package
nsp audit-shrinkwrap
npm install -g david
npm outdated
david -g // getting global dependencies
```

EXPRESSJS + WEBSOCKET USING SOCKET.IO

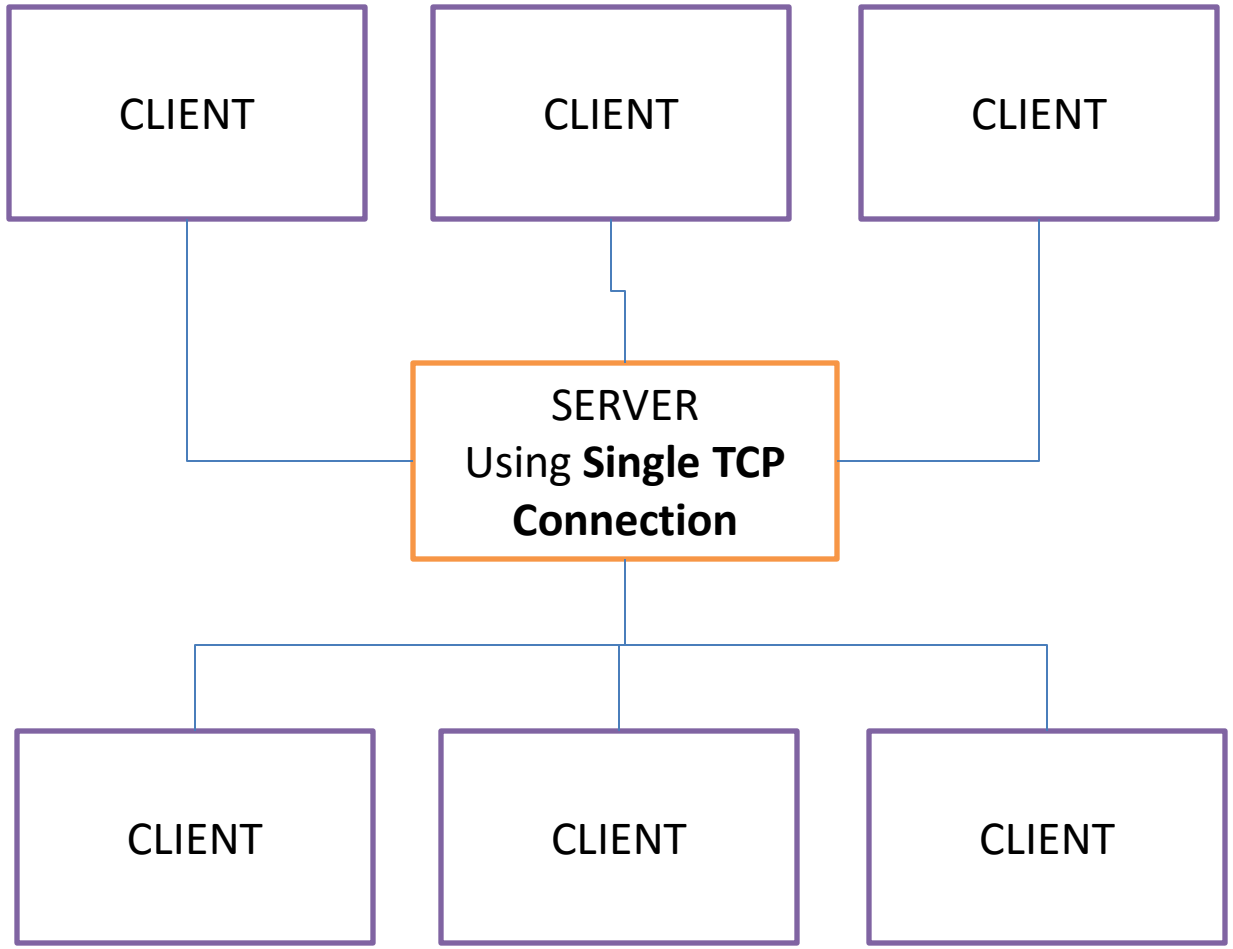
NODE.JS WEBSOCKET AND SOCKET.IO

web socket

WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.

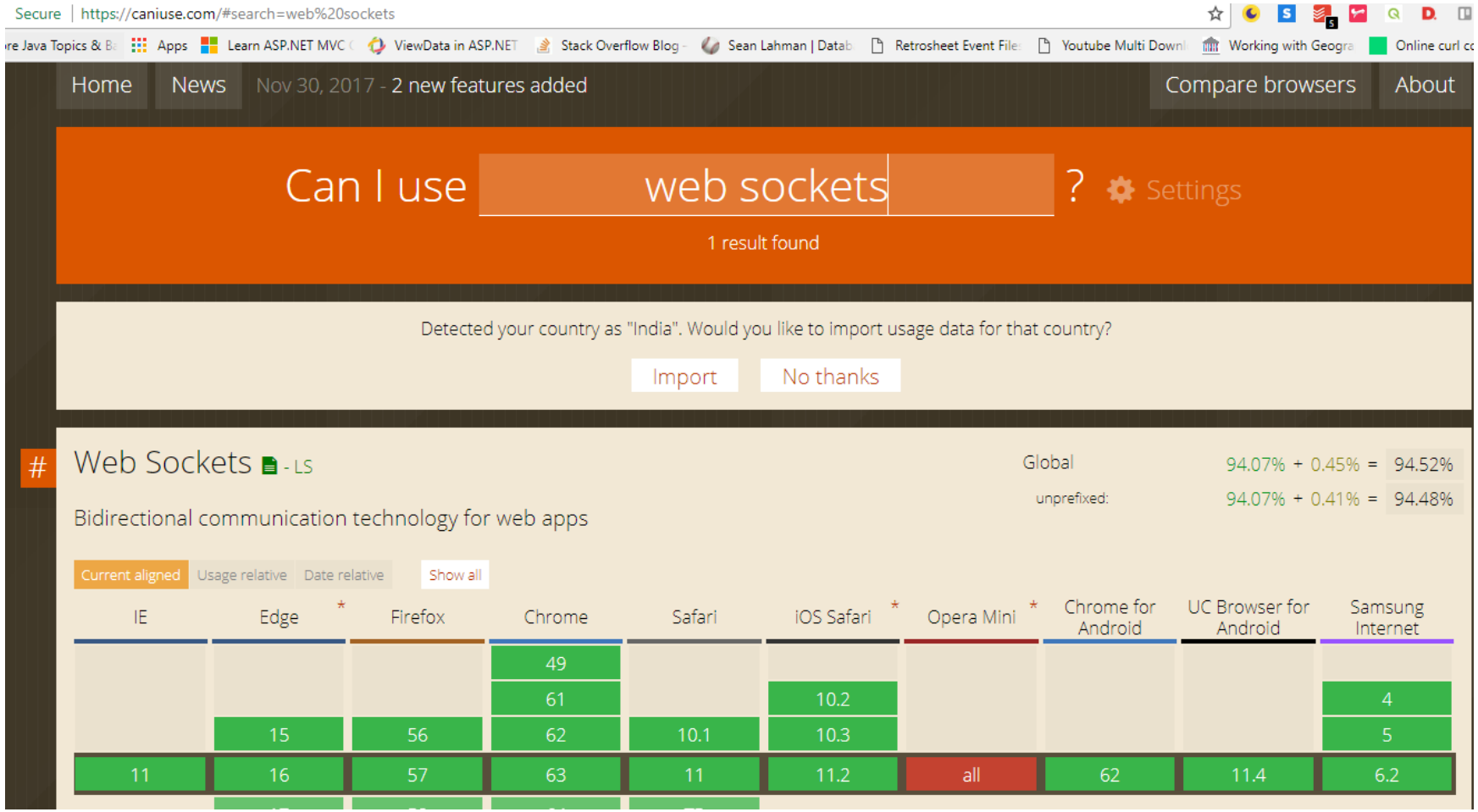
WEBSOCKETS

- Web sockets are communication between a client (browser) and server
- Bidirectional (data flows both ways)
- Allows real-time data flow.
- Excellent cross browser and cross platform support.
- Low-latency
- Ability to send audio,video and image as binary stream



Web sockets browser support

https://caniuse.com/#search=web%20sockets



Websocket use

- Used to create multiplayer browser games
- Collaborative code editing
- Live streaming of data
- Online drawing canvas applications or image manipulation software
- Multi-channel chat room application.

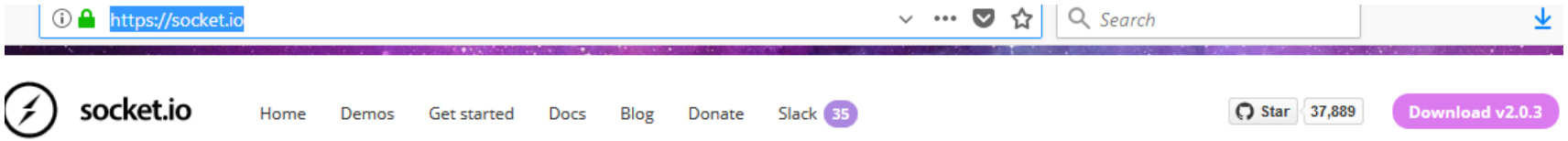
Socket.io

- Socket.io enables realtime bidirectional event-based communication.
- Real-time analytics: push data to clients that gets represented as real-time counters, charts or logs
- Instant messaging and chat
- Binary streaming
- Document collaboration

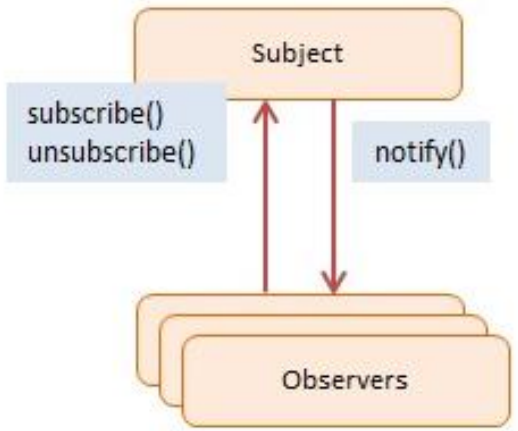
Websocket users

Socket.io

https://socket.io/



PUB-SUB PATTERN



Socket.io Installation

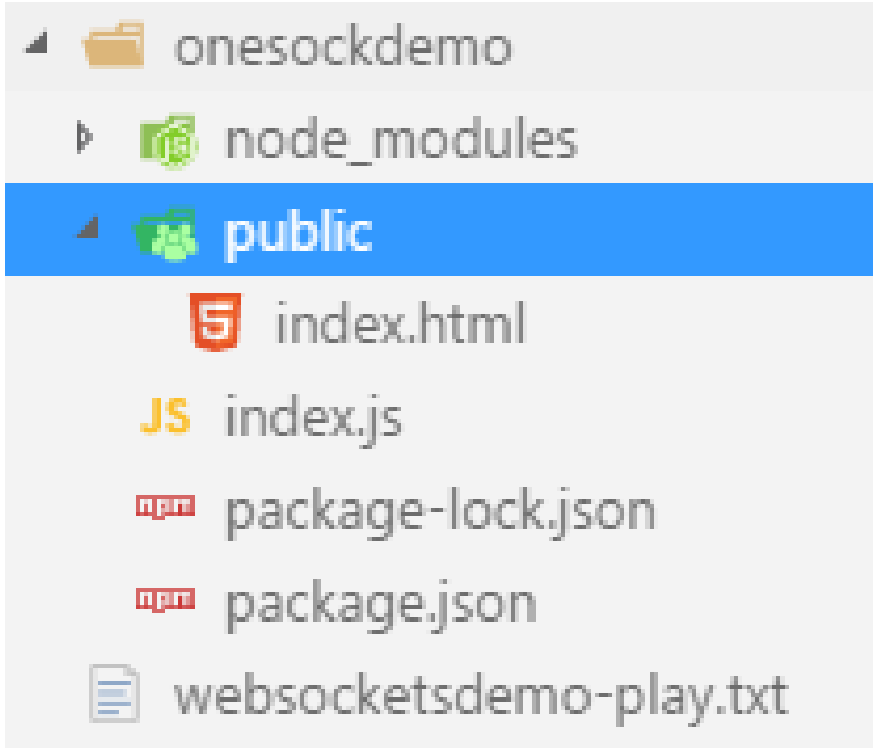

```
npm init -y
```

```
npm install –save  express path socket.io
```

```
mkdir public
```

```
touch public/index.html
```

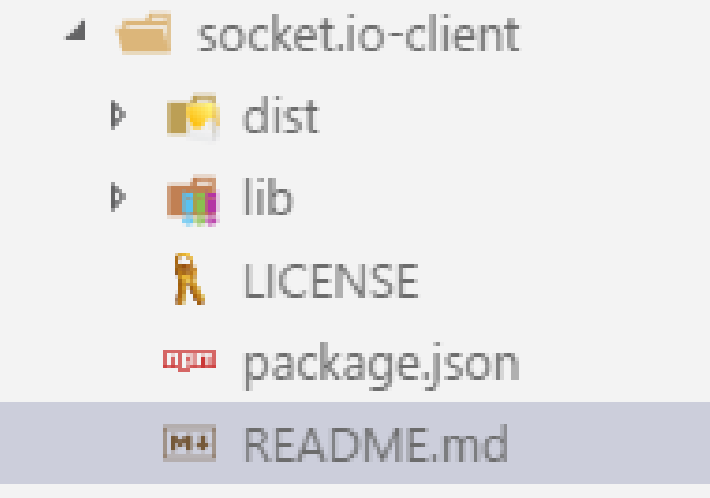
```
touch  index.js
```



Basic Socket Demo

Step 1:

- navigate to **node_modules/socket.io-client/README.md**



(http://travis-ci.org/socketio/socket.io-client)
[![Dependency Status](https://david-dm.org/socketio/socket.io-client.svg)]
(https://david-dm.org/socketio/socket.io-client)
[![devDependency Status](https://david-dm.org/socketio/socket.io-client/dev-status.svg)]
(https://david-dm.org/socketio/socket.io-client#info=devDependencies)
![NPM version](https://badge.fury.io/js/socket.io-client.svg)
![Downloads](http://img.shields.io/npm/dm/socket.io-client.svg?style=flat)
[](http://slack.socket.io)

[![Sauce Test Status](https://saucelabs.com/browser-matrix/socket.svg)](https://saucelabs.com/u/socket)

How to use

A standalone build of `socket.io-client` is exposed automatically by the
socket.io server as `/socket.io/socket.io.js`. Alternatively you can
serve the file `socket.io.js` found in the `dist` folder.

```html
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io('http://localhost');
  socket.on('connect', function(){});
  socket.on('event', function(data){});
  socket.on('disconnect', function(){});
</script>
```

Basic Socket Demo

Step 2:

- create client side code for socket.io => public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

    <script src="/socket.io/socket.io.js"></script>
</body>
</html>
```

Basic Socket Demo

Step 3:

•Create index.js server for socket.io communication, by instantiating express

```
//index.js for Socket Application demo
var express = require('express');
var path = require('path');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var port =8080;

app.use(express.static(path.join(__dirname,"public")));

server.listen(port,function(){
    console.log('SycliQ Customer Chat Server'+port);
});
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\12-websocketsdemo\onesockdemo> node index.js
SycliQ Customer Chat Server8080
```

Basic Socket Demo

Step 4:

•Create connection for socket.io in index.js

```
//index.js for Socket Application demo
var express = require('express');
var path = require('path');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var port =8080;

app.use(express.static(path.join(__dirname,"public")));

//connections using socket.io
io.on('connection', function(socket){
    console.log("new connection made to SycliQ Chat Server");
});

server.listen(port,function(){
    console.log('SycliQ Customer Chat Server'+port);
});
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\12-websocketsdemo\onesockdemo> node index.js
SycliQ Customer Chat Server8080
```

Basic Socket Demo

Step 5:

•Establish a handshake between chat server and client, using index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

    <script type="text/javascript" src="/socket.io/socket.io.js"></script>

    <script type="text/javascript">
        var socket=io("http://localhost:8080");
    </script>
</body>
</html>
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\12-websocketsdemo\onesockdemo> node index.js
SycliQ Customer Chat Server8080
new connection made to SycliQ Chat Server
new connection made to SycliQ Chat Server
new connection made to SycliQ Chat Server
```

Basic Socket Demo

Step 6:

- Write methods to send messages from server to client and back.
- **Emit()**
- **On()**

```
//index.js for Socket Application demo
var express = require('express');
var path = require('path');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var port = 8080;
app.use(express.static(path.join(__dirname, "public")));
//connections using socket.io
io.on('connection', function (socket) {
    console.log("new connection made to SycliQ Chat Server");
    //emit method to send message from the socket server
    socket.emit("msg-from-socketServer", {
        messagefromSrvr: "You are connected to SycliQ Chat Server"
    });
    socket.on("msg-from-clientSide", function (msgfromSycliQClient) {
        console.log(msgfromSycliQClient);
    });
});
server.listen(port, function () {
    console.log('SycliQ Customer Chat Server' + port);
});
```

Basic Socket Demo

Step 7:

- Now add socket.on and socket.emit methods to the client to receive message from the server and send message back to the server from the client.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>SycliQ Chat Client </h1>
    <script type="text/javascript" src="/socket.io/socket.io.js"></script>
    <script type="text/javascript">
        var socket = io("http://localhost:8080");

        socket.on('msg-from-socketServer', function (evt) {
            document.body.appendChild(document.createTextNode(evt.messagefromSrvr));
            socket.emit('msg-from-clientSide', {
                messagefromSrvr: "Return calling from client"
            });
        });
    </script>
</body>
</html>
```

```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan\expressjsdemo\12-websocketsdemo\onesockdemo> node index.js
SycliQ Customer Chat Server8080
new connection made to SycliQ Chat Server
{ messagefromSrvr: 'Return calling from client' }
```

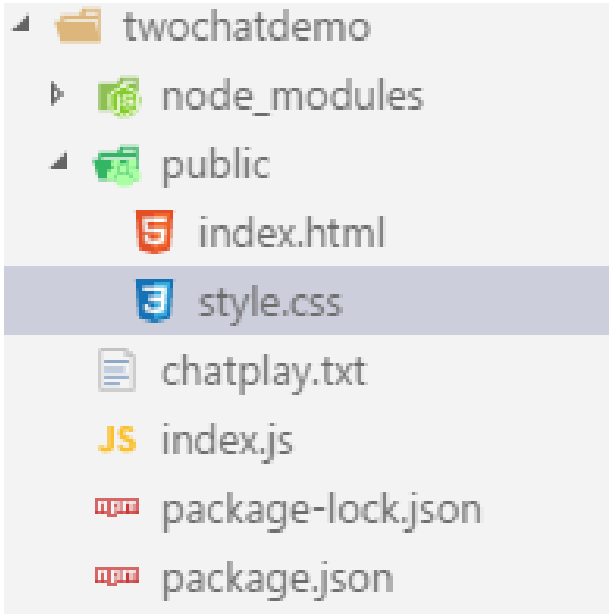
WEB CHAT APPLICATION WITH SOCKET.IO

SOCKETS.IO: EXAMPLE TWO

WEB CHAT APPLICATION

Step 1:

•Scaffold and install the dependencies required for building a web chat.




```
npm init -y
npm install –save  express path socket.io nodemon
mkdir public
touch public/index.html
touch  index.js
Touch public/style.css
```

WEB CHAT APPLICATION

Step 2:

- Scaffold and install the dependencies required for building a web chat.
- Write your index.js

```
//index.js
var express = require('express');
var socket=require('socket.io');
//instance of express
var app = express();
var server = app.listen(9000,function(){
    console.log("You are listening to SycliQ Chat Server @port 9000");
});
//rendering static files
app.use(express.static('public'));
//setting up sockets
var io = socket(server);
io.on('connection',(socket)=>{
    console.log("connection made", socket.id);
    //Handling chat events
    socket.on('chat',function(data){
        io.sockets.emit('chat',data);
    });
    //handling keyboard events
    socket.on('keyboarding',function(data){
        socket.broadcast.emit('typing something',data);
    });
});
```

WEB CHAT APPLICATION

Step 4:

- Write your clientchat.js

```
//clientside chat
var socket = io.connect('http://localhost:9000');
// Query DOM
var message = document.getElementById('message'),
    handle = document.getElementById('handle'),
    btn = document.getElementById('send'),
    output = document.getElementById('output'),
    feedback = document.getElementById('feedback');
// Emit events
btn.addEventListener('click', function () {
    socket.emit('chat', {
        message: message.value,
        handle: handle.value
    });
    message.value = "";
});
//keyboarding events
message.addEventListener('keypress', function () {
    socket.emit('keyboarding', handle.value);
});
// Listen for events
socket.on('chat', function (data) {
    feedback.innerHTML = '';
    output.innerHTML += '<p><strong>' + data.handle + ': </strong>' + data.message + '</p>';
});
socket.on('keyboarding', function (data) {
    feedback.innerHTML = '<p><em>' + data + ' is typing a message...</em></p>';
});
```

WEB CHAT APPLICATION

Step 5:

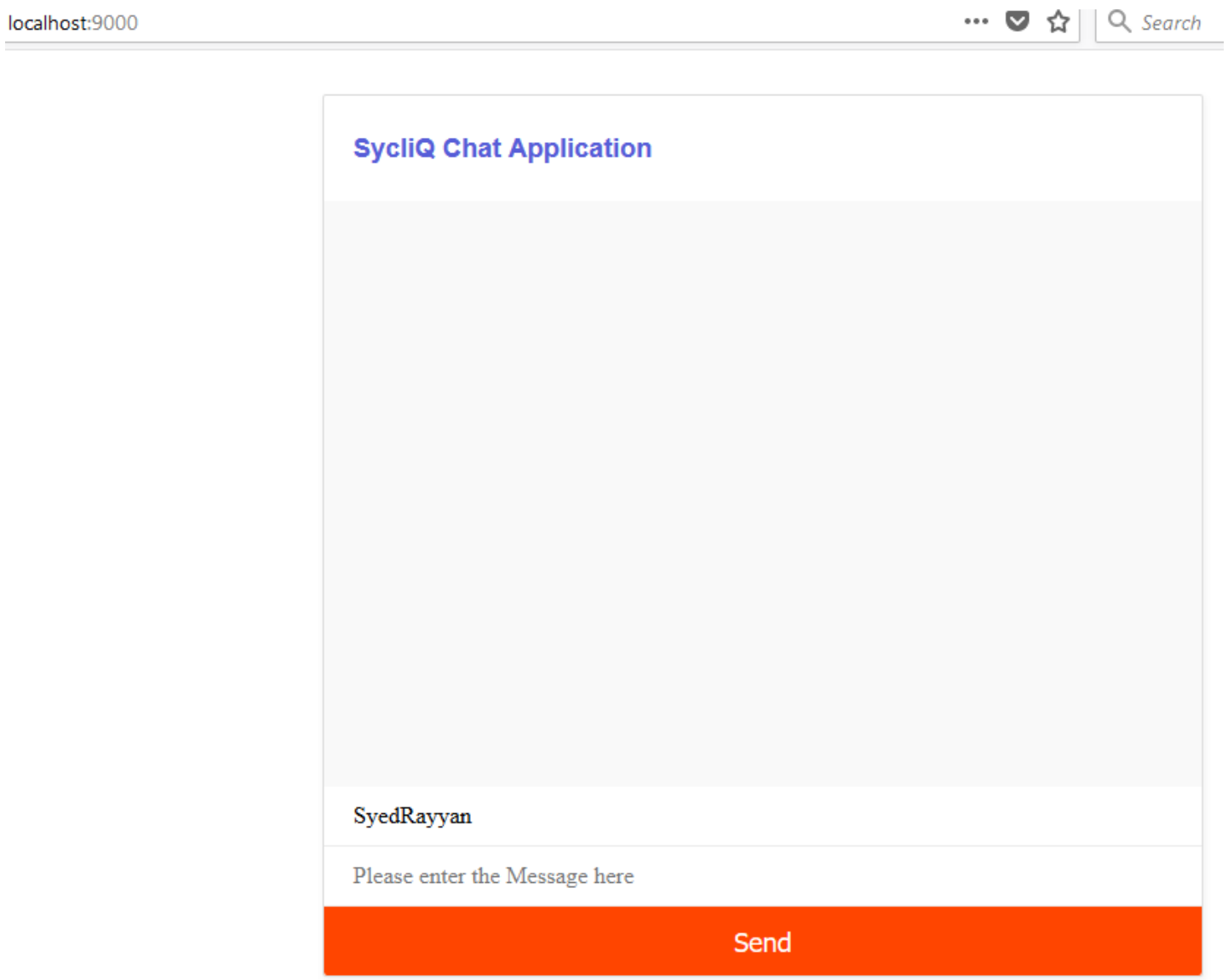
- Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link href="./style.css" rel="stylesheet" />
    <script type="text/javascript" src="/socket.io/socket.io.js"></script>
</head>
<body>
    <div id="sycliq-chat">
        <h2>SycliQ Chat Application</h2>
        <div id="chat-window">
            <div id="output"></div>
            <div id="feedback"></div>
        </div>
        <input id="handle" type="text" placeholder="Chat Handle Name" />
        <input id="message" type="text" placeholder="Please enter the Message here" />
        <button id="send">Send</button>
    </div>
    <script src="./clientchat.js"></script>

</body>
</html>
```

WEB CHAT APPLICATION

Step 6:

- Run your application using **nodemon index.js**
- **Open http://localhost:9000 in multiple browsers and start using chat.**
- **On making any changes nodemon automatically restarts your application.**



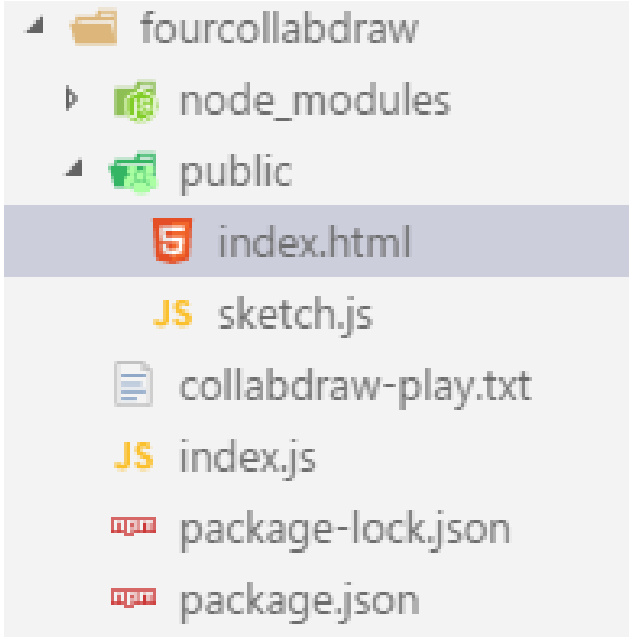
COLLABORATIVE DRAWING APPLICATION USING NODEJS,EXPRESS,SOCKET.IO,NODEMON, P5

SOCKETS.IO: EXAMPLE THREE

COLLABORATIVE DRAWING APP

Step 1:

•Scaffold and install the dependencies required for building a web chat.




```
npm init -y
```

```
npm install –save  express path socket.io nodemon
```

```
mkdir public
```

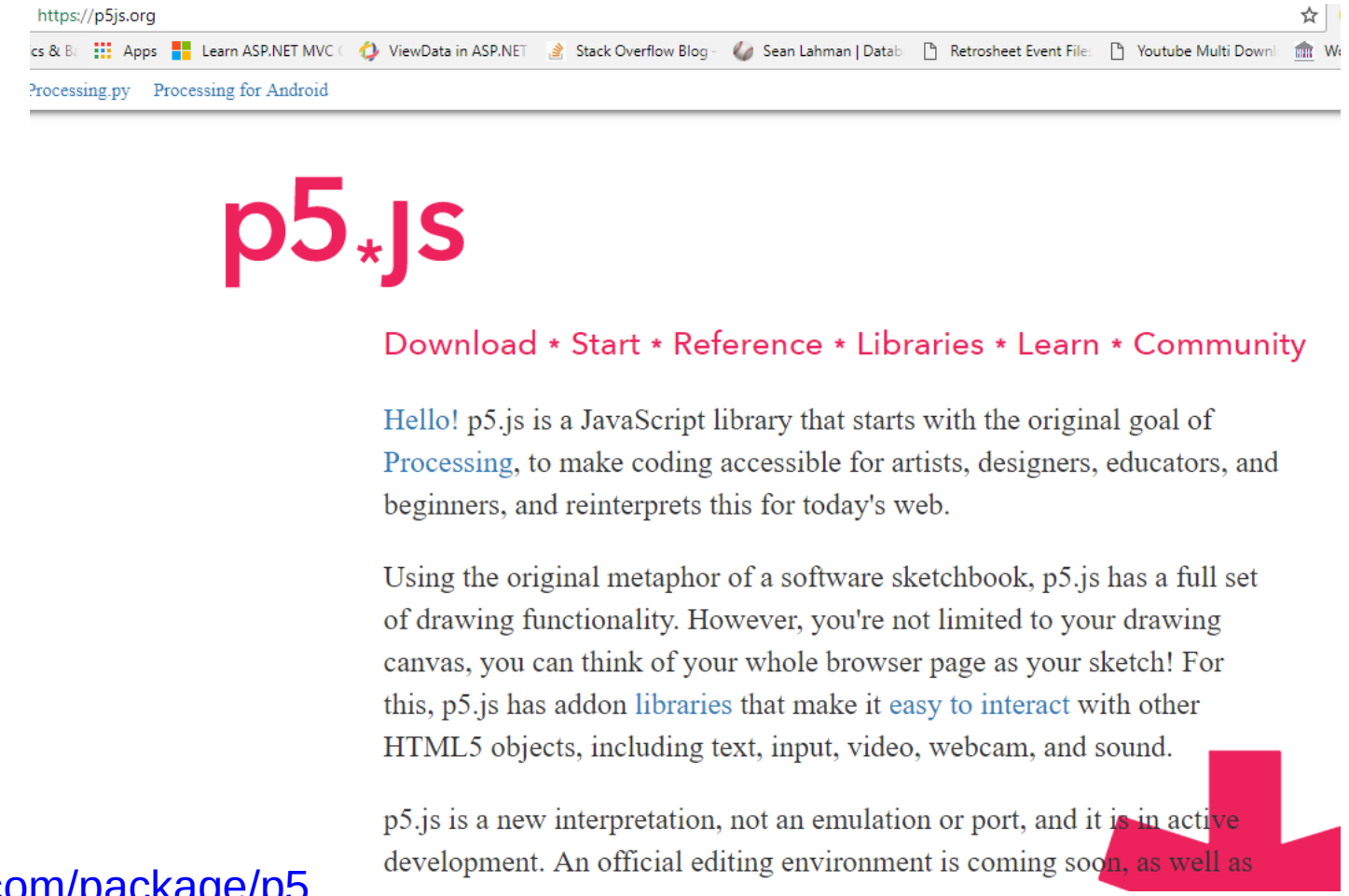
```
touch public/index.html
```

```
touch  index.js
```

```
Touch public/style.css
```

P5.js interactive library for drawing

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.16/p5.js"></script>
```



https://p5js.org/

https://www.npmjs.com/package/p5

COLLABORATIVE DRAWING APP

Step 2:

•Create index.html and add the **cdn for p5.js library for interactive drawing.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Collaborative Drawing Application</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.16/p5.js"></script>
    <script src="./sketch.js"></script>
</head>
<body>
    <h1>Collaborative Drawing Application</h1>

</body>
</html>
```

COLLABORATIVE DRAWING APP

Step 3:

•Create sketch.js file basic drawing methods.

```
//sketch.js used by p5 to draw
function setup() {

}
function draw() {
    ellipse(50, 50, 80, 80);
}
```

COLLABORATIVE DRAWING APP

Step 4:

- Instantiate, express, socket.io to create websocket server using **index.js**

```
//index.js for Collaborative Drawing Application
var express = require('express');
var path = require('path');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var port = 8080;
app.use(express.static(path.join(__dirname, "public")));

server.listen(port, function () {
    console.log('SycliQ Customer Chat Server' + port);
});
```

COLLABORATIVE DRAWING APP

http://localhost:8080/

Step 5:

- Let's run our basic scaffolded application to check whether all the components are rendering properly.
- **Node index.js**

COLLABORATIVE DRAWING APP

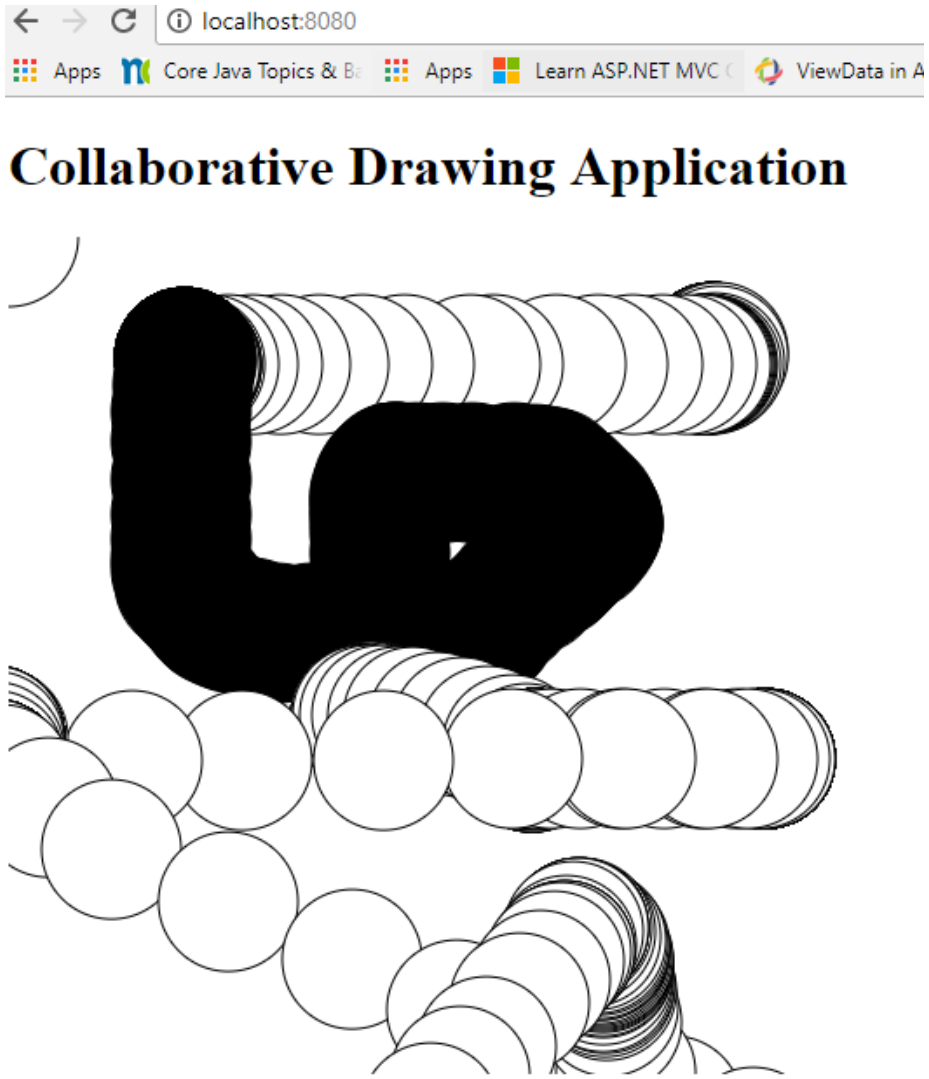
Step 6:

- **lets add some more code in the sketch.js and check if p5 is rendering based on mouse interactions.**

```
//sketch.js used by p5 to draw
function setup() {
    createCanvas(640, 480);
}

function draw() {
    if (mouseIsPressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```

COLLABORATIVE DRAWING APP

http://localhost:8080/



Step 7:

- Let's run our basic scaffolded application to check whether all the components are rendering properly.
- **Node index.js**
- **On mouseover, we can see the circles drawing on the canvas.**

COLLABORATIVE DRAWING APP

Step 8:

- Add socket.io client libraries to **index.html as highlighted in the figure**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Collaborative Drawing Application</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.16/p5.js"></script>
    <script src="./sketch.js"></script>
    <!--socket-io-client-->
    <script type="text/javascript" src="/socket.io/socket.io.js"></script>
</head>
<body>
    <h1>Collaborative Drawing Application</h1>

</body>
</html>
```

COLLABORATIVE DRAWING APP

http://localhost:8080/

Step 9:

- update the sketch.js by adding the socket connection method.

```
//sketch.js used by p5 to draw
function setup() {
    createCanvas(640, 480);
    socket=io.connect('http://localhost:8080');
}
function draw() {
    if (mouseIsPressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```

COLLABORATIVE DRAWING APP

http://localhost:8080/

Step 10:

• lets add mousedrag method and emit it to the socket server.

```
//sketch.js used by p5 to draw
function setup() {
    createCanvas(640, 480);
    socket = io.connect('http://localhost:8080');
}

function mouseDragged() {
    console.log(mouseX + "," + mouseY);
    var data = {
        x: mouseX,
        y: mouseY
    }
    socket.emit('mouse', data);
    if (mouseIsPressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}

function draw() {

}
```

COLLABORATIVE DRAWING APP

http://localhost:8080/

Step 11:

- lets add methods to accepts data on to the server and broadcast the message to all the clients.

```
//index.js for Collaborative Drawing Application
var express = require('express');
var path = require('path');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);
var port = 8080;
app.use(express.static(path.join(__dirname, "public")));
//connection
io.on('connection',function(socket){
    console.log("Collaborative Drawing Socket Server is Running!!"+socket.id);
    socket.on('mouse',function(data){
        console.log(data);
        //lets broadcast to all the clients
        socket.broadcast.emit('mouse',data);
    })
});
server.listen(port, function () {
    console.log('SycliQ Customer Chat Server' + port);
});
```

COLLABORATIVE DRAWING APP

Step 12:

- update the client to render new data.
- **refreshDrawing function to render the new data.**

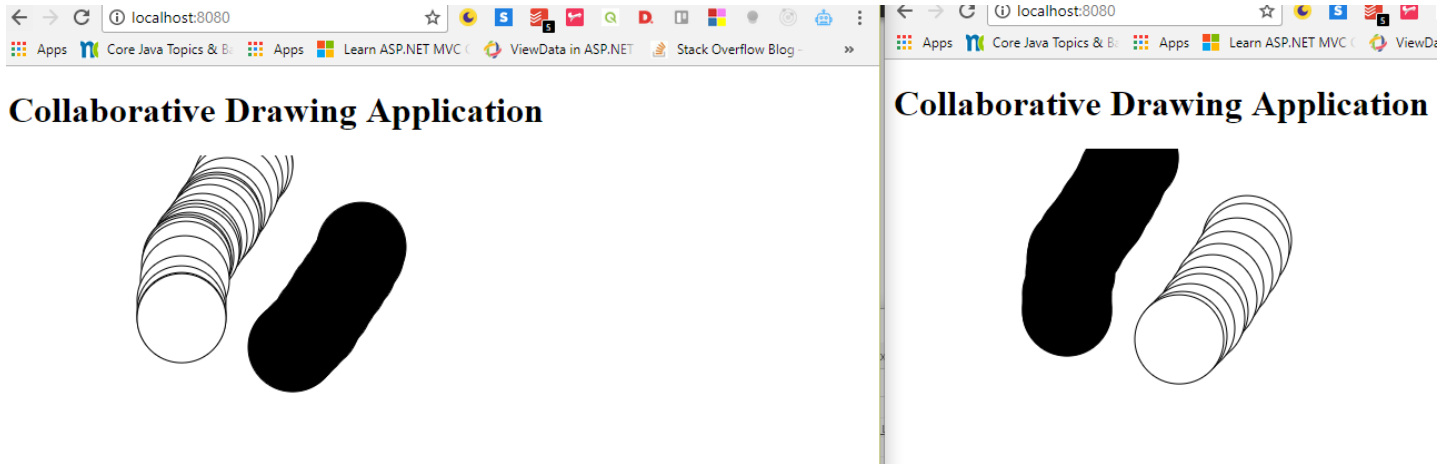
```
//sketch.js used by p5 to draw
function setup() {
    createCanvas(640, 480);
    socket = io.connect('http://localhost:8080');
    socket.on('mouse', refreshDrawing);
}
function refreshDrawing(data){
    if (mouseIsPressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(data.x, data.y, 80, 80);
}
function mouseDragged() {
    console.log(mouseX + "," + mouseY);
    var data = {
        x: mouseX,
        y: mouseY
    }
    socket.emit('mouse', data);
    if (mouseIsPressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}

function draw() {}
```

COLLABORATIVE DRAWING APP

Step 13:

- run your application with **nodemon index.js**
- **Observer that when we draw on one client, the same is updated on the second client.**

http://localhost:8080/



SCRAPING WITH NODEJS +CHERIO.JS

NODE.JS WEB SCRAPING

SYED AWASE KHIRNI

YARN:BASIC SCRAPPING EXAMPLE

Cheerio.io

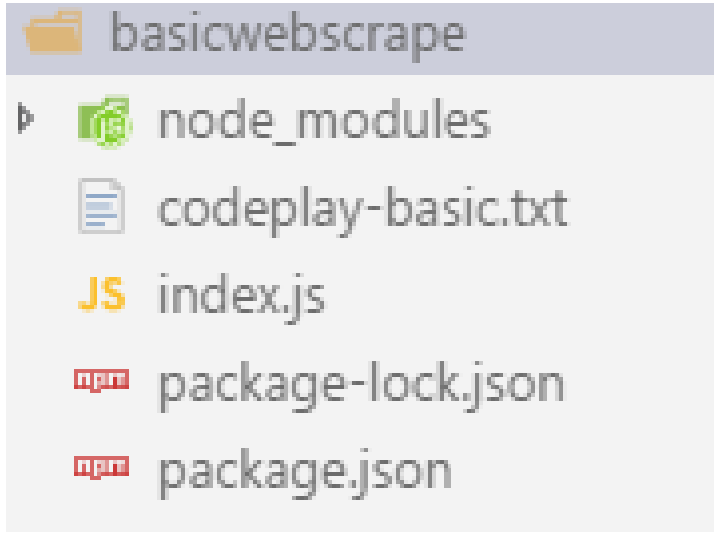
A fast flexible and lean implementation of jQuery designed specifically for the server.

https://github.com/cheeriojs/cheerio

BASIC SCRAPPING APPLICATION

Step 1:

•Scaffold and install the dependencies required for building a scrapping application.




```
npm init -y
```

```
npm install –save  express cheerio
path handlebars  request
```

```
touch  index.js
```

BASIC SCRAPPING APPLICATION

Step 2:

•Create express server

index.js

```
//index.js
var express = require('express');
var path=require('path');
var request=require('request');
var cheerio = require('cheerio');
var fs = require('fs');
var app=express();
var port = 8080;


app.listen(port,function(){
    console.log("Sycliq Scraper Running @8080");
});
```

BASIC SCRAPPING APPLICATION

Step 3:

- Create a request function to extract data from imdb and indeed website as shown in the figure.

```
//index.js
var express = require('express');
var path=require('path');
var request=require('request');
var cheerio = require('cheerio');
var fs = require('fs');
var app=express();
var port = 8080;
var url ="http://www.imdb.com/title/tt2527336/?ref_=inth_ov_tt";
request(url,function(err,response, body){
    var $=cheerio.load(body);
    var originalTitle = $('.originalTitle');
    var originalTitleText =originalTitle.text();
    console.log(originalTitleText);
});
var indeedUrl ="https://www.indeed.co.in/jobs?q=gis+jobs&l=Bengaluru%2C+Karnataka";
request(indeedUrl,function(err,response,body){
    var $=cheerio.load(body);
    $('.company').filter(function(){
        var companyName=$(this);
        companyNameText = companyName.text();
        console.log(companyNameText);
    });
})
app.listen(port,function(){
    console.log("Sycliq Scraper Running @8080");
});
```

BASIC SCRAPPING APPLICATION

Step 4:

- Create a request function to extract data from indeed and store it as an object.

```
request(indeedUrl,function(error,response,body){
    var $=cheerio.load(body);
    var companyName=$('.company');
    var cnText =companyName.text();
    var jobTitle = $('.jobTitle font');
    var jtText = jobTitle.text();
    var location=$('.location');
    var lText = location.text();
    var summary = $('#job_summary p');
    var sText = summary.text();
    var job={
        companyName:cnText,
        jobTitle:jtText,
        location:lText,
        summary:sText
    };
    console.log(job);
});
app.listen(port,function(){
    console.log("Sycliq Scraper Running @8080");
});
```

SYED AWASE KHIRNI

YARN:PACKAGE MANAGER

Yarn is designed to be a faster, **more reliable and more secure** npm client that retains access to the vast **npm** (Node Package Manager) registry

JavaScript Package managers

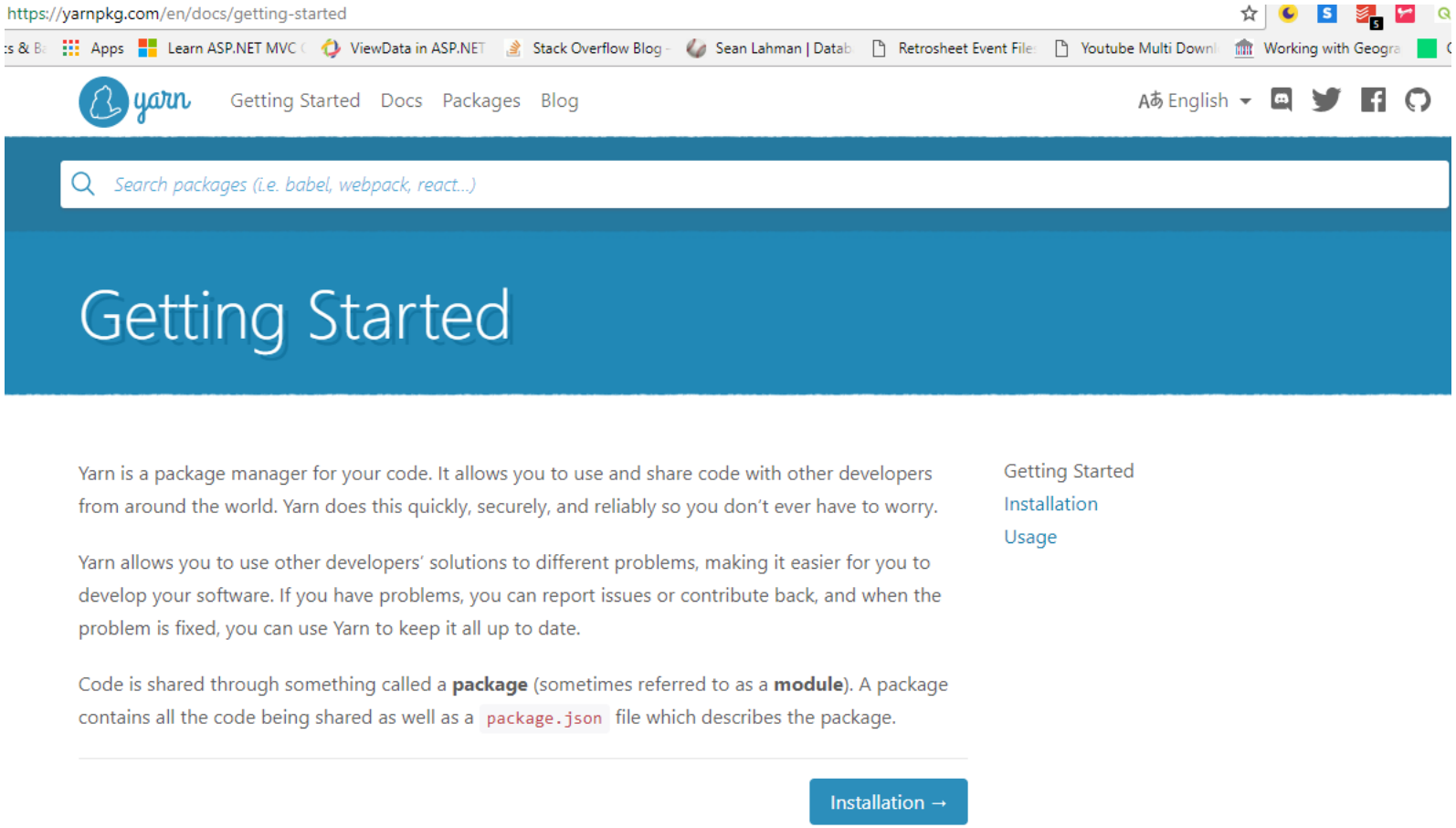
BOWER

duo

Yarn

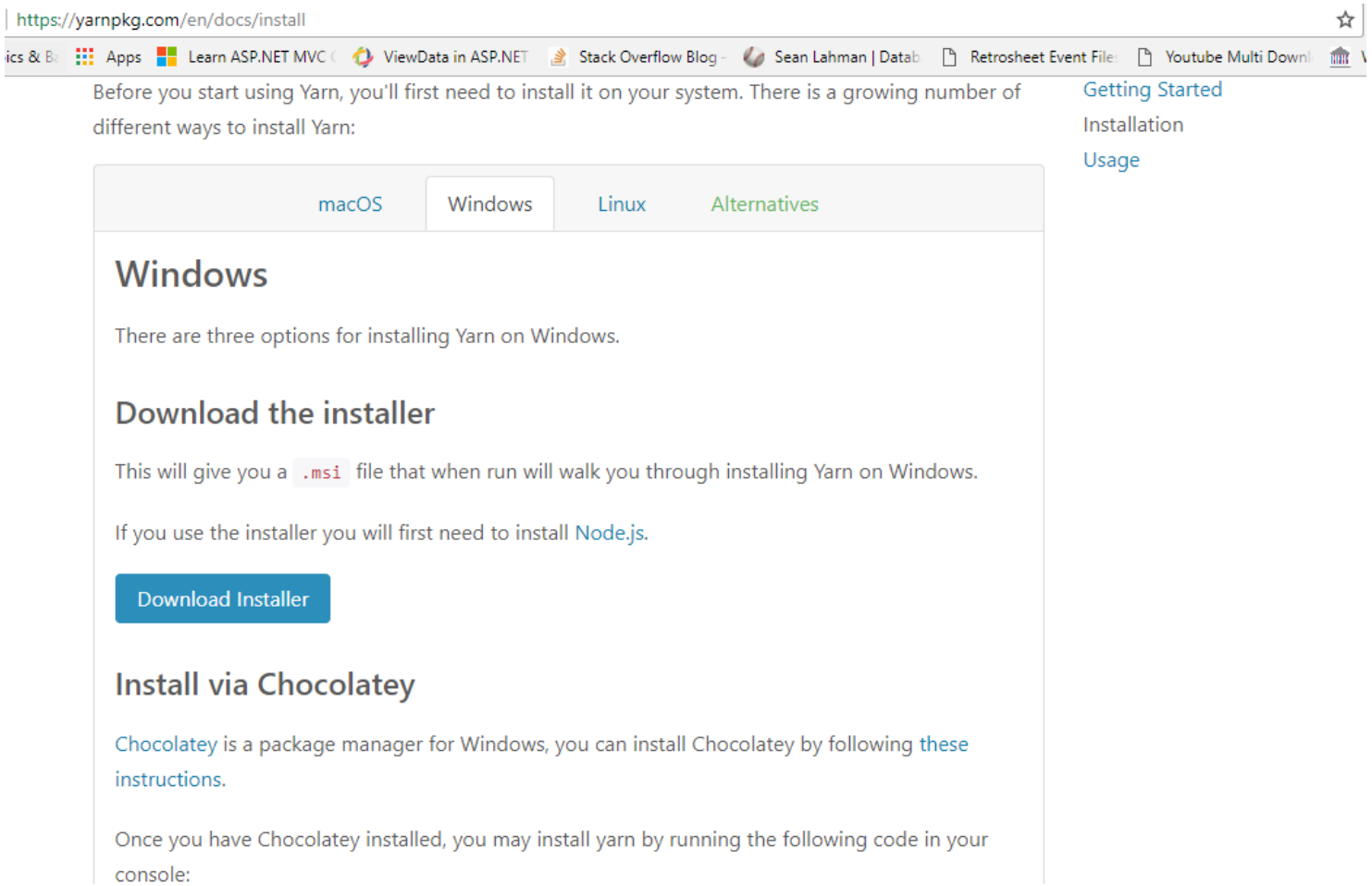
https://yarnpkg.com/en/docs/getting-started

- Developed by Facebook.



Installing Yarn

https://yarnpkg.com/en/docs/install



NPM

- NPM was designed around the idea of Semantic Versioning (semver).
- Predictable dependency tree (if desired) was only possible through **npm shrinkwrap, which is NOT DEFAULT behaviour.**

YARN

- Yarn was announced in Oct 2016, by Facebook.
- It's main intial goal was to address npm installations not being deterministic due to semver related behaviour.
- Every yarn install generates a yarn.lock which is similar to npm-shrinkwrap.json, which is default behaviour.
- Parallel operations needed to enhance the speed.
- Yarn does not need to have an internet connection to install dependencies that are already cached locally.

NPM		YARN
• Npm install	===	• Yarn
• Npm install express --save	===	• Yarn add express
• Npm uninstall express --save	===	• Yarn remove express
• Npm install express –save-dev	===	• Yarn add express --dev
• Npm update --save	===	• Yarn upgrade
• Npm install express@latest --save	===	• Yarn add express
• Npm install express --global	===	• Yarn global add express

NPM		YARN
• Npm init	===	• Yarn init
• Npm link	===	• Yarn link
• Npm outdated	===	• Yarn outdated
• Npm publish	===	• Yarn publish
• Npm run	===	• Yarn run
• Npm cache clean	===	• Yarn cache clean
• Npm login	===	• Yarn login

NPM		YARN
• Npm install --production	===	• Yarn --production
• Npm xmas		• Yarn licenses ls
• Npm visnup		• Yarn licenses generate-disclaimer
		• Yarn why express
		• Yarn upgrade-interactive
• Npm view	===	• Yarn info
• Npm list --depth	===	• Yarn list –depth=0

NPM		YARN
	===	• Yarn config list
	===	
	===	

Yarn config list


```
PS E:\CT\NodeJS\Dec2017NodejsDanskePlan> yarn config list
yarn config v1.3.2
info yarn config
{ 'version-tag-prefix': 'v',
  'version-git-tag': true,
  'version-git-sign': false,
  'version-git-message': 'v%s',
  'init-version': '1.0.0',
  'init-license': 'MIT',
  'save-prefix': '^',
  'ignore-scripts': false,
  'ignore-optional': false,
  registry: 'https://registry.yarnpkg.com',
  'strict-ssl': true,
  'user-agent': 'yarn/1.3.2 npm/? node/v8.9.1 win32 x64' }
info npm config
{ '//registry.npmjs.org/:_authToken': '68bccda6-4ba0-4182-811f-1db80b5169b5',
  python: 'C:\\Python27\\python.exe',
  'C:\Python27\python.exe': '' }
Done in 0.05s.
```

Yarn Commands

Command	Description
Yarn add	Adds a package to use in your current package
Yarn init	Initializes the development of a package
Yarn install	Installs all dependencies defined in a package.json file
Yarn publish	Publishes a package to a package manager
Yarn remove	Removes an unused package from the current package
Yarn<script>[<args>]	Run used defined scripts
Yarn <command> [<args>]	
Yarn create	Creates new projects
Yarn config set	Yarn config set <key> <value>
Yarn config delete <key>	

Yarn Commands

Command	Description
Yarn cache list [-pattern]	Stores every package in a global cache in your user directory on the file system.
Yarn cache dir	Prints out the path where global cache is currently stored.
Yarn cache clean <module_name>	Clears global cache.
Yarn config set cache-folder <path>	
Yarn check –integrity	Verifies the version and package dependencies
Yarn upgrade	Upgrade packages to their latest version based on the specified range
Yarn upgrade-interactive	Similar to upgrade but in interactive mode
Yarn import	Assists in migration of projects currently relying on npm-shinkwrap.json
Yarn info <package>	Information about a package.
Yarn init	Interactively creates or updates a package.json file

Yarn Commands

Command	Description
Yarn install –check-files	Verifies already installed files
Yarn install –flat	Installs all the dependencies, but only allows one version for each package
Yarn install –force	Forcefull refetch
Yarn licenses list	Lists all the licenses of installed packages
Yarn outdated	Checks for outdated package dependencies
Yarn owner	Defines owner and permissions
Yarn pack	Yarn pack –filename creates a compressed gzip archive of package dependencies.
yarn publish	Publishes a package to the npm registry
Yarn team	Manage teams in organizations and change team memberships.
Yarn test	Runs the test script defined by the package.

Yarn Commands

Command	Description
Yarn unlink	Remove a symlinked packaged created with yarn link
Yarn upgrade	Upgrades packages to their latest version based on the specified range.
Yarn why	Shows information about why a package is installed.
Yarn versions	Updates the package version
Yarn version –new-version <version>	Creates a new version specified by <version>
Yarn version –no-git-tag-version	Creates a new verison without creating a git tag
Yarn bin	Will print the folder where yarn will install executables files for your packages.

Advantages of Yarn

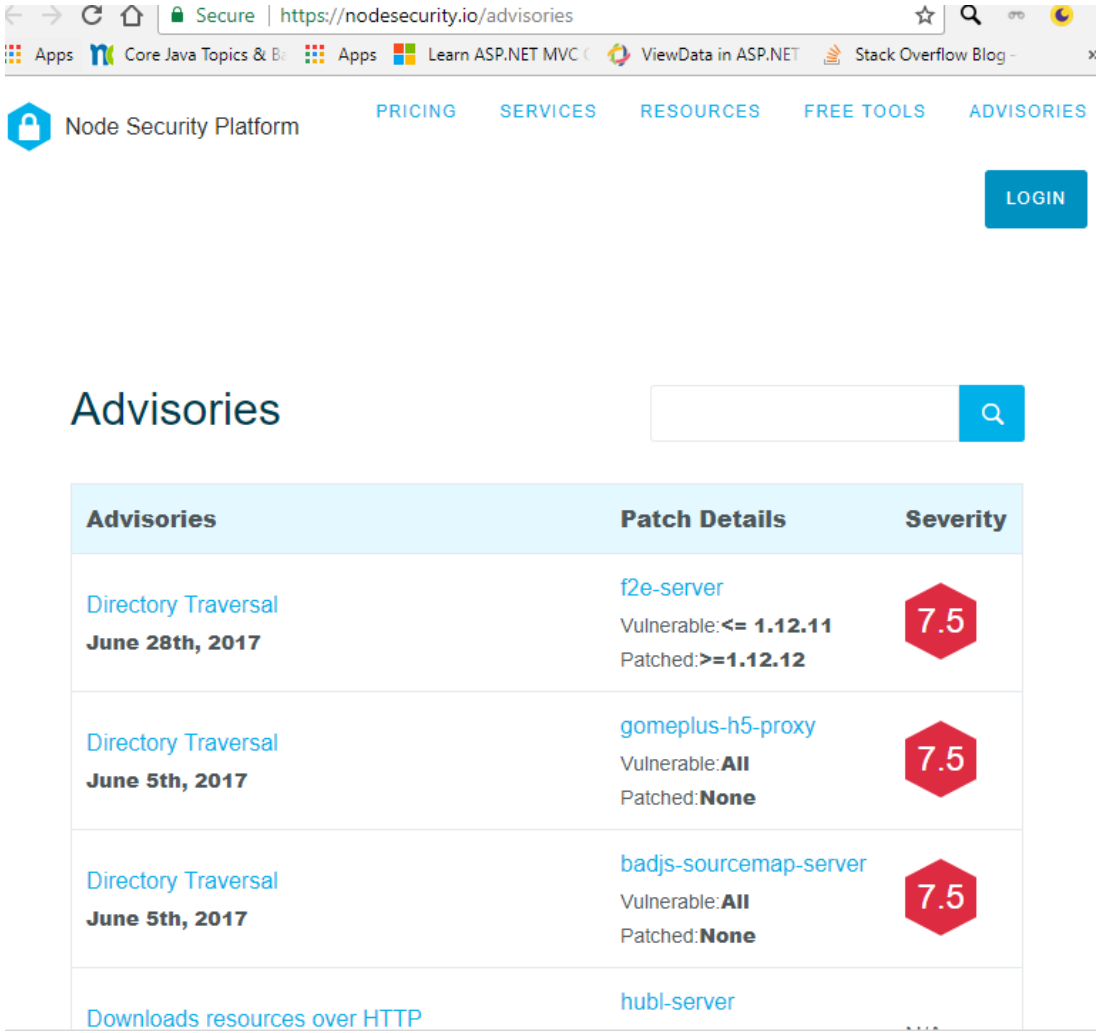
- Security: uses checksum to verify integrity of every installed package before its code is executed. (NPM does not do this by default)
- Offline Mode: if it is in the cache, it would load it immediately.
- Deterministic: dependencies installed in the same way across all machines.
- Network Performance: efficiently queues up requests and avoids request waterfalls in order to maximize network utilization.
- Network resilience: a single request failing wont cause an install to fail. Requests are retired upon failure.
- Flat mode: resolve mismatching versions of dependencies to a single version to avoid creating duplicates.
- Parallel installation: npm executes tasks sequentially, while yarn executes tasks in parallel

SYED AWASE

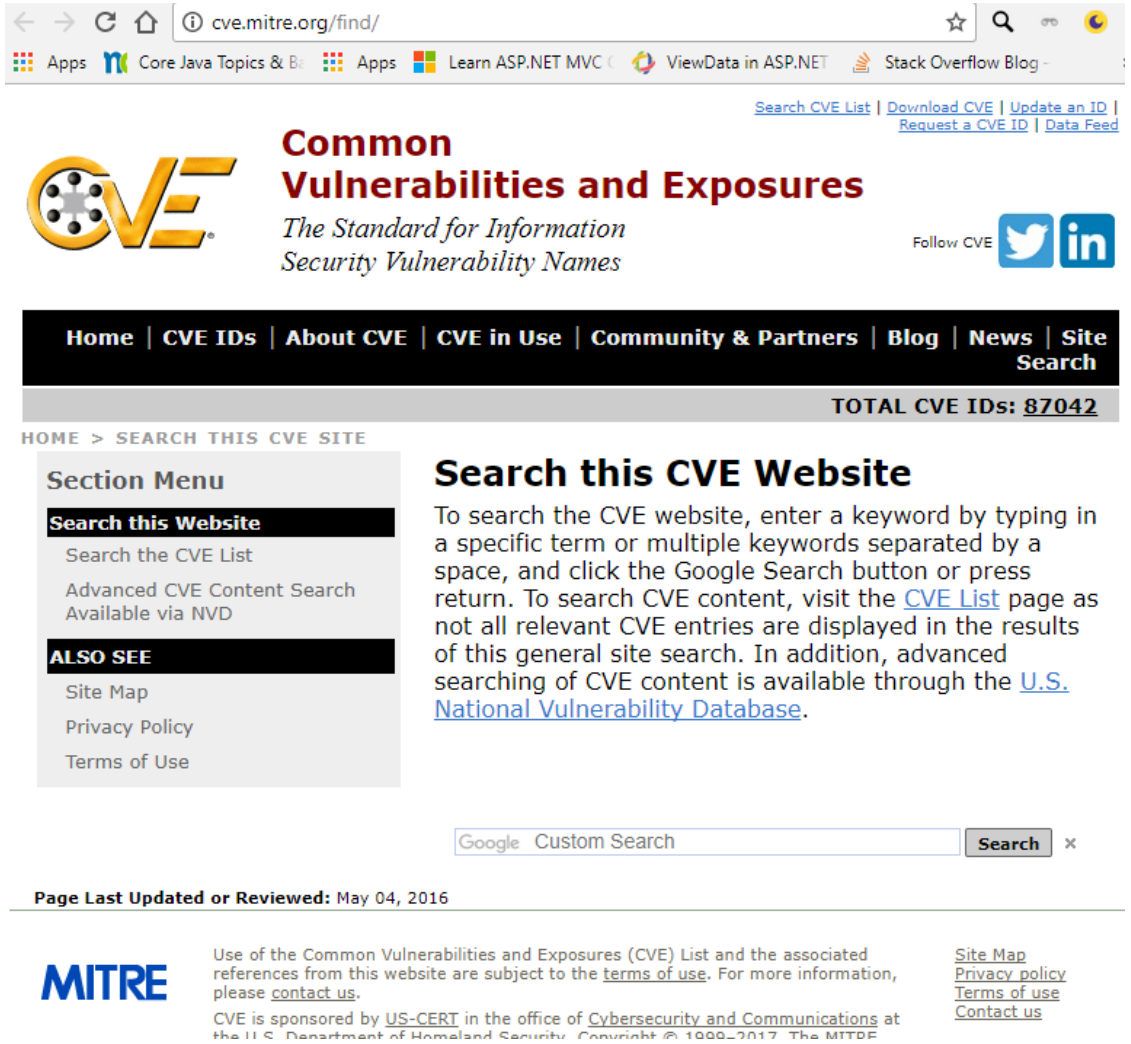
NODEJS SECRUITY

Node Security Platform

Source:https://nodesecurity.io/advisories



Common Vulnerabilities and Exposures



Source: http://cve.mitre.org/find/

NPMJS: SECURITY PACKAGES

crypto-js	helmet	Safe-buffer	kerberos
snyk	xss	Password-generator	cryptiles
Koa-helmet	nsp	hpp	Koa-csrf
	Xss-filters	dompurify	Owasp-password-strength

NPMJS: SECURITY PACKAGES

passmarked	ssri	Helmet-csp	credential
Eslint-plugin-security	Md5-o-matic	csprng	hsts
nocache	redefine	Helmet-crossdomain	frameguard
	jailed	Connect-roles	Evil-scan

Original Series

NPMJS: SECURITY PACKAGES

Csp-parse	Api-core	Don’t-sniff-mimetype	Firebase-
x10	clerobee	authy	targaryen
Easy-pbkdf2	Content-security-policy-builder	Vault-cipher	authorized
	Redact-url	Node-wsfederation	Secure-filters

Original Series

NPMJS: SECURITY PACKAGES

Stormpath-config	arpscan	Virgil-crypto	recaptcha2
Osprey	Express-jwt-permissions	Referrer-policy	fusker
saslmechanisms	Sasl-plain	Sql-injection	Nodesecurity-npm-utils
	hpkp	X-xss-protection	Oauth-wrap

Original Series

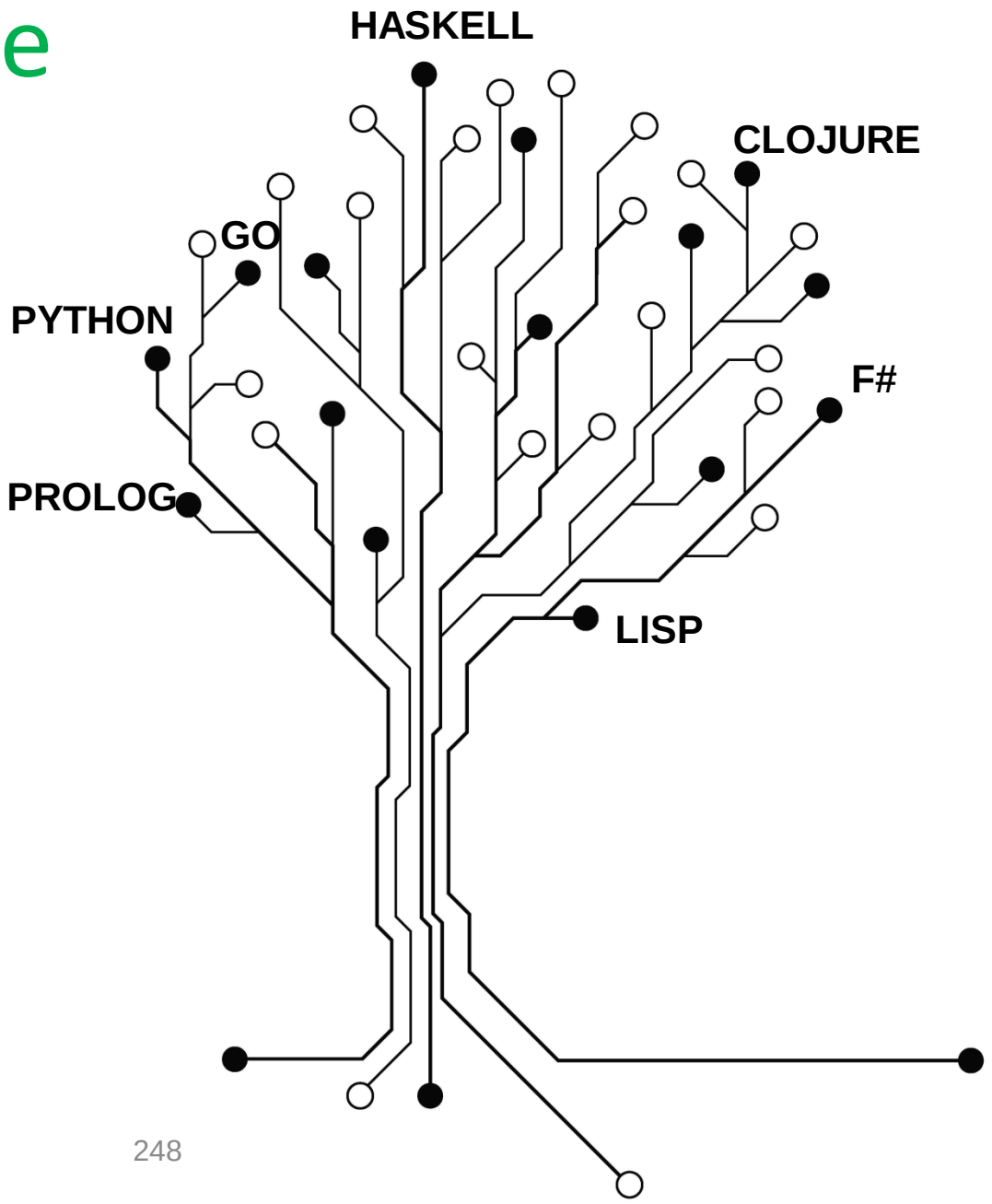
Security HTTP Headers

- Strict-Transport-Security
- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy

TPRI-SYCLIQ PROGRAMS OVERVIEW

EMPOWERING YOU

Artificial Intelligence

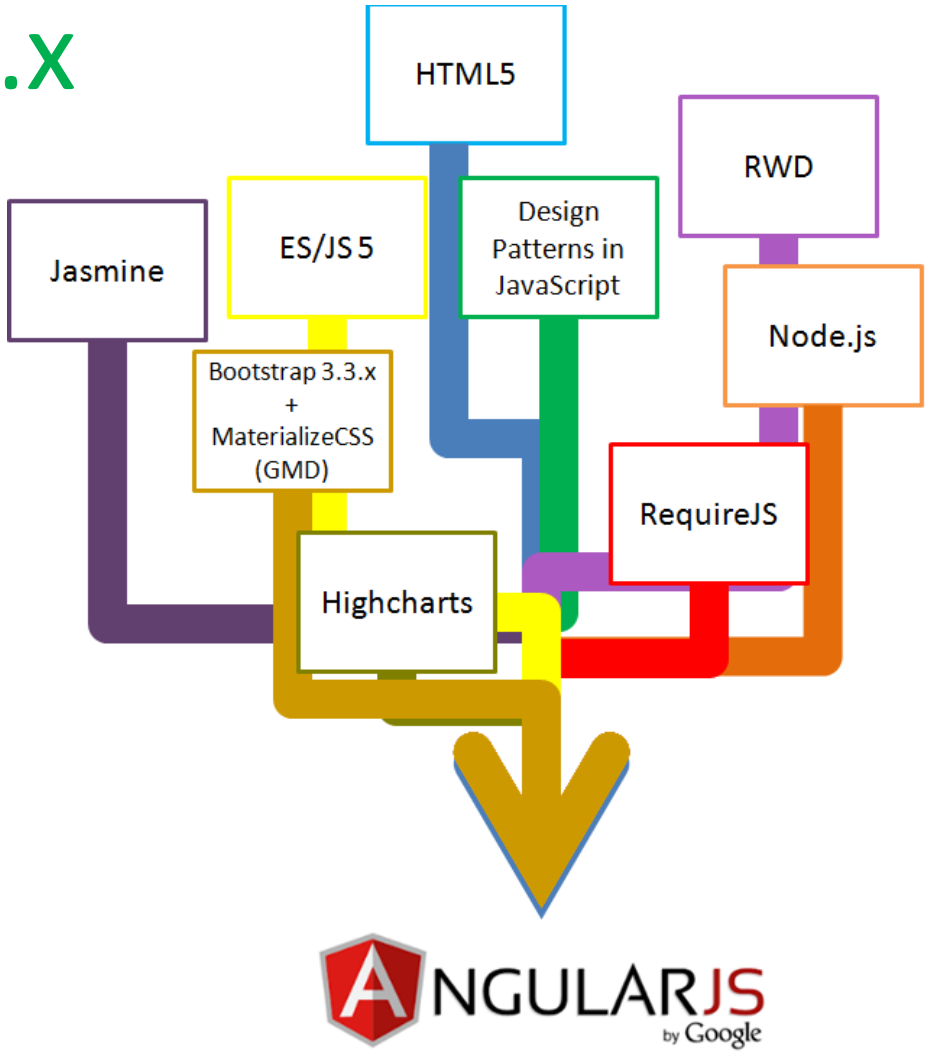


- We also train on AI Stack
- Reach out to us sak@sycliq.com or sak@territorialprescience.com
- www.territorialprescience.com
- www.sycliq.com
- +91.9035433124

Angular 4.x/Angular JS 1.5.x

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124



Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

NOW OFFERING!

JAVASCRIPT FRAMEWORKS
CODE DRIVEN CAPACITY BUILDING PROGRAM

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124

Original Series

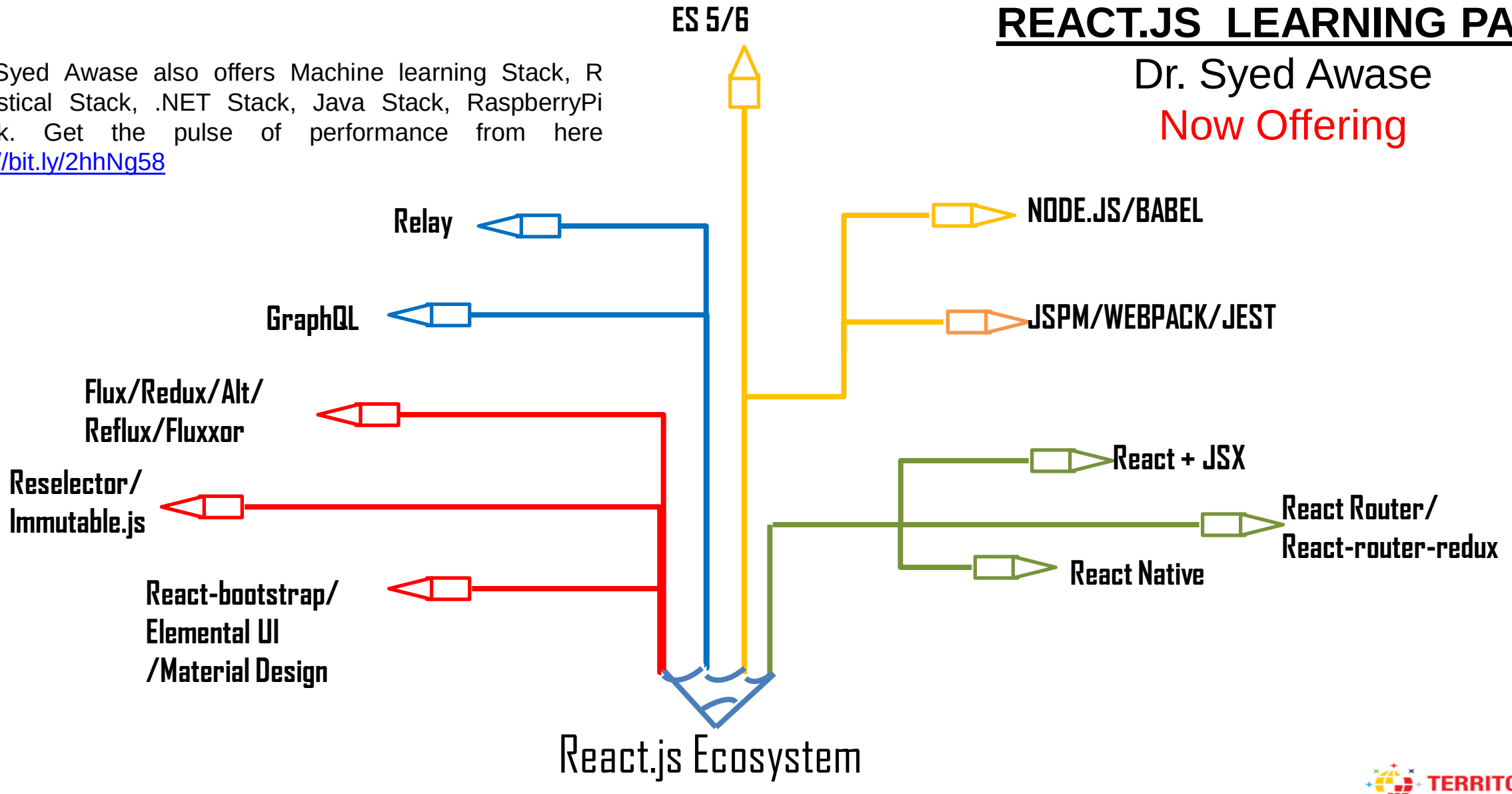
Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

REACT.JS LEARNING PATH

Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

Dr. Syed Awase

Now Offering

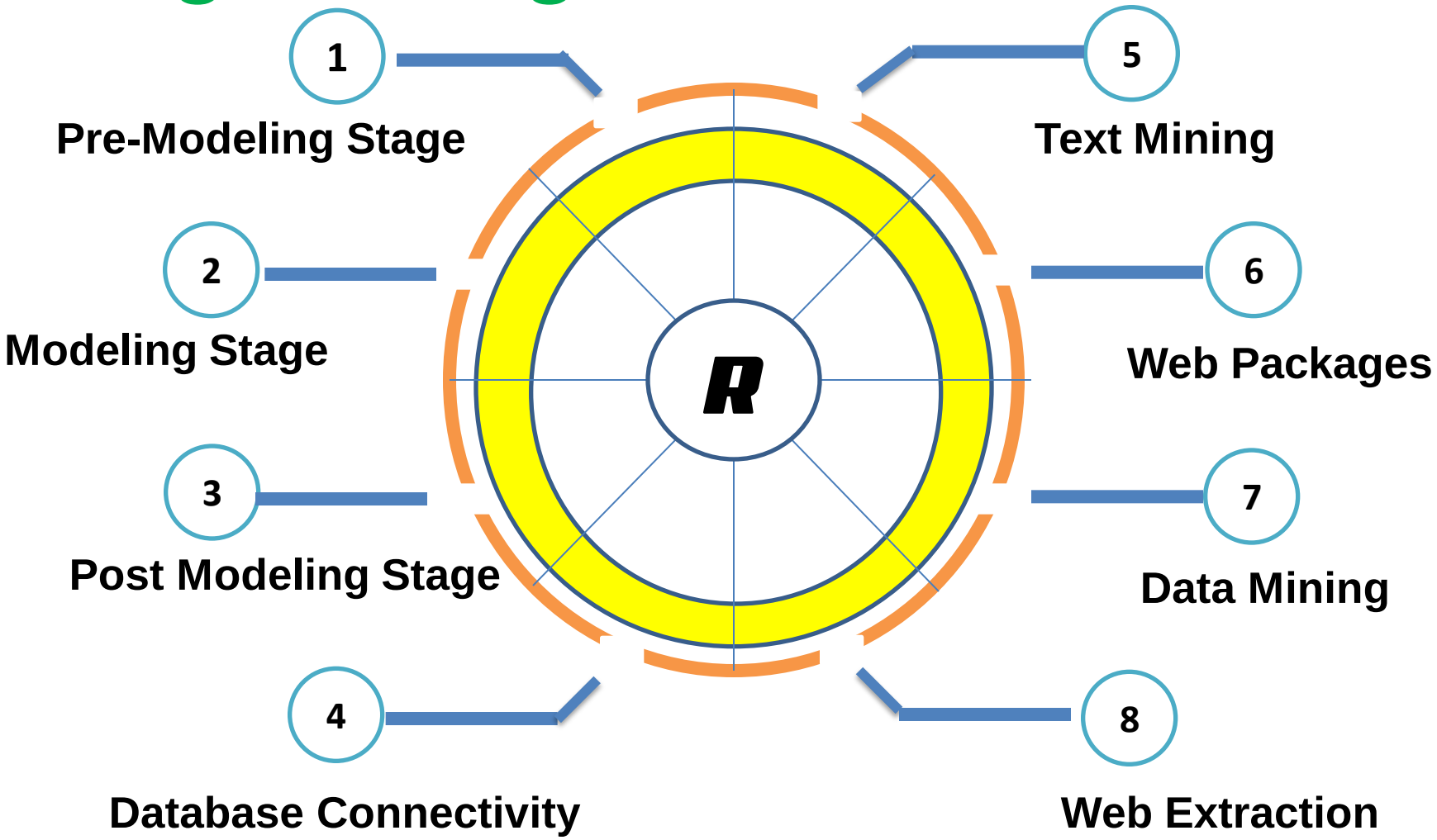


Original Series

R-Statistical Programming

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/ +91.9035433124



Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

Thank You

We also provide Code Driven Open House Trainings : sak@territorialprescience.com or sak@sycliq.com

Java Technologies

- Core Java
- Hibernate
- Spring Framework
- Play Framework
- Hadoop
- Groovy & Grails

Microsoft Technologies

- C# Core
- Entity Framework
- MVC 5/6
- Web Api
- OWIN/KATANA
- WCF
- WPF

Python

- Python
- Django
- Flask
- Numpy
- Scipy
- Machine Learning

Data Science

- R Statistical Programming
- Julia

SQL and NoSQL

- Oracle
- PostgreSQL
- MSSQL
- MongoDB
- Neo4j
- Redis
- Firebase
- Apache Cassandra

Client-Side Frameworks

- Angular JS 1.5.x
- Angular 2.4.x
- React JS
- KnockOut JS
- VueJS
- Backbone JS
- EMBER JS
- Hapi JS
- METEORJS
- MEANJS
- Coffeescript
- Dart

Others

- LISP
- CLOJURE
- RUST
- GO
- Raspberry PI
- Coming Soon
  - PHP
  - RoboticOS