

Nodejs & intro MEAN

Rajeev Gupta

rgupta.mtech@gmail.com

Java trainer & consultant

What is MEAN?

MEAN STACK



Mongo DB
(database system)

Express

Express
(back-end web
framework)



Angular.js
(front-end
framework)



Node.js
(back-end runtime
environment)

What is MEAN?

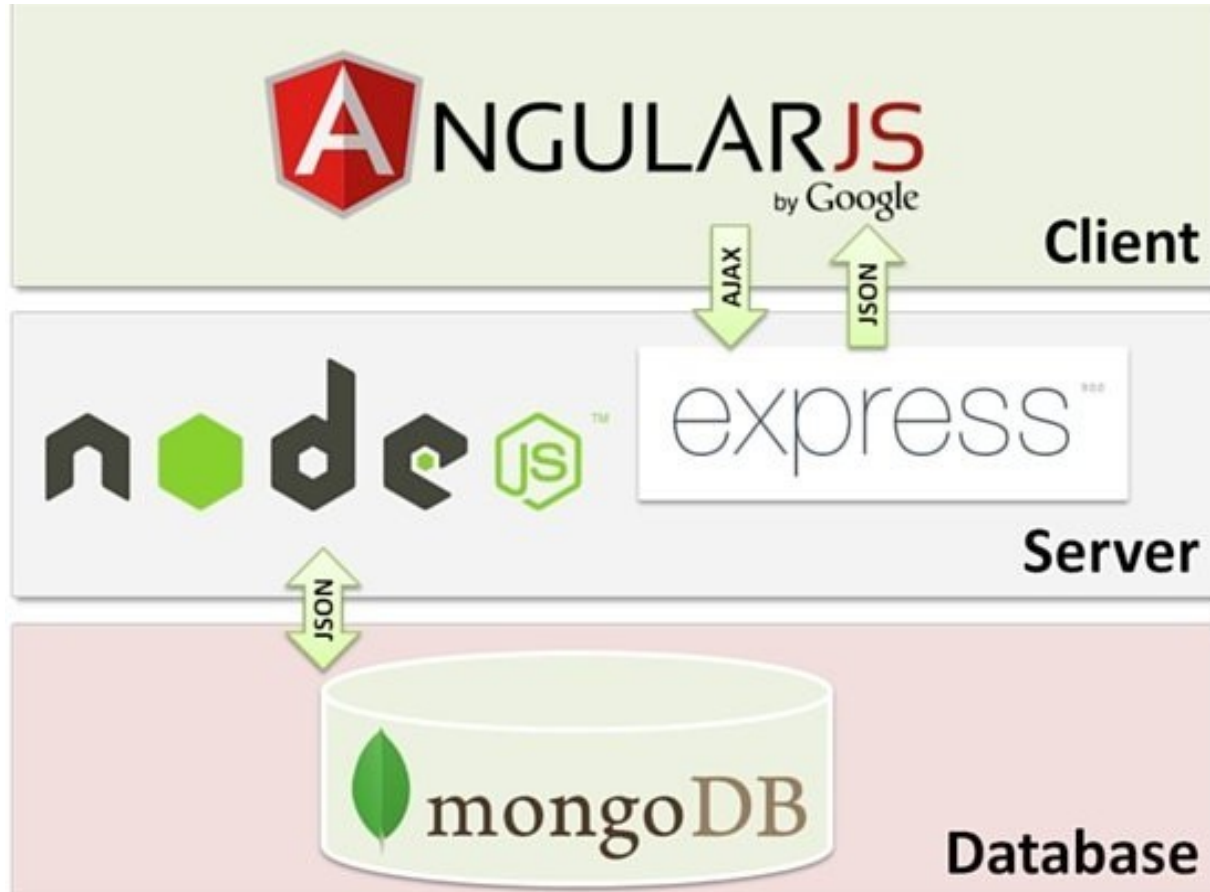
M = MongoDB --> DATA solution

E = Express JS (for use with nodeJS) -->
Routing and ease of transactions (above simple NodeJS) , session management, server side

A = AngularJS (for use with nodeJS) --> Presentation layer ease (above simple NodeJS) THIS IS CLIENT SIDE MVC

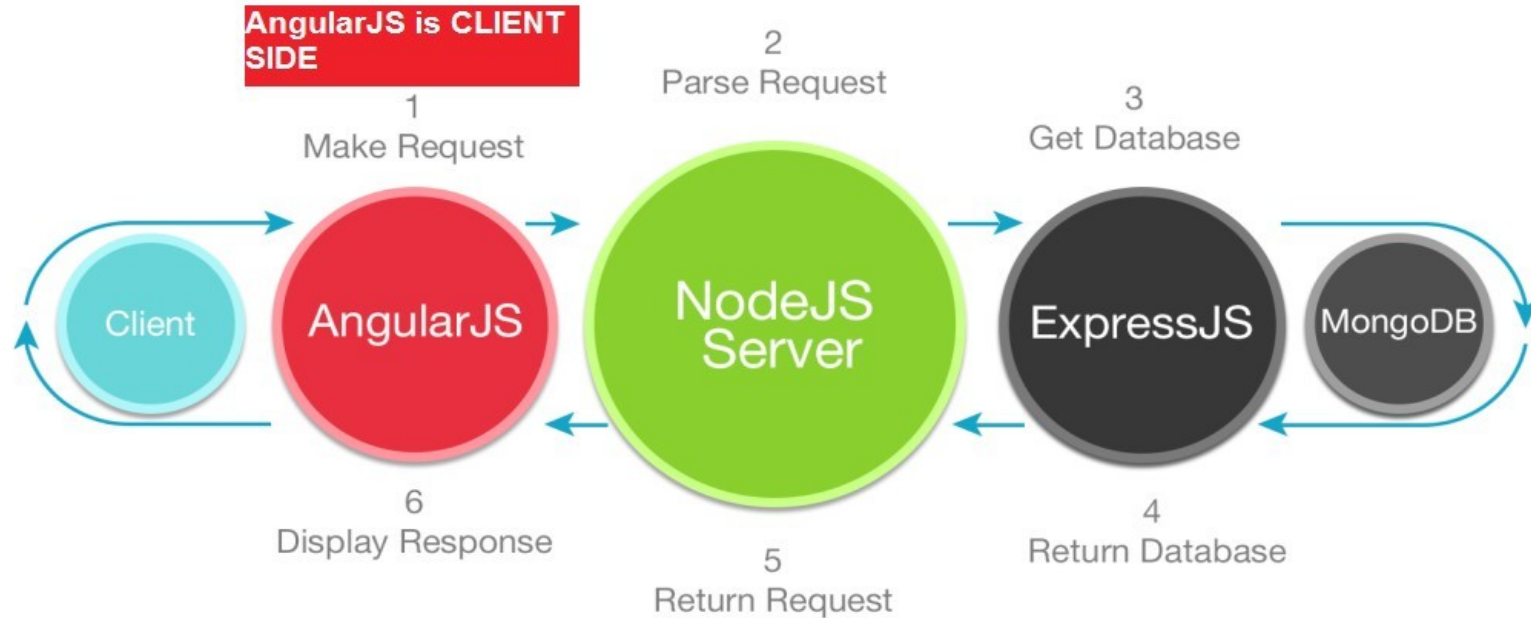
N = NodeJS -->
Platform for javascript frameworks Express and Angular built on top of nodejs used for serverside programs, server side

MEAN application flow



User Request Processing MEAN

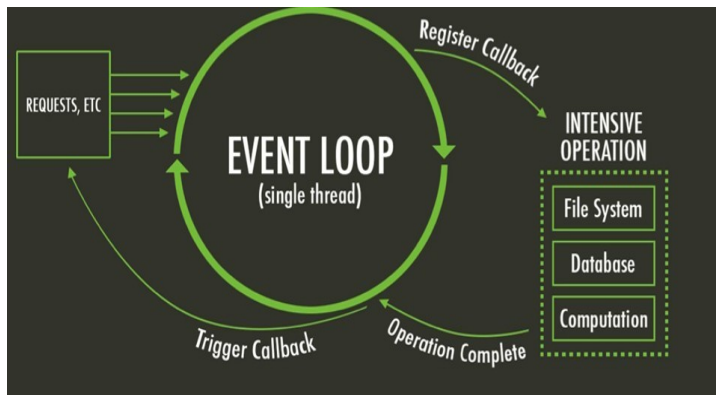
Here we can see how a user/client requests is processed and response returned



What does Angular give us

What is a single page application?

Angular used to create Single Page Application (SPA)



SPA = Think of Yahoo! or Google Mail as examples.

runs inside single page load, just updates part of the page as you need it.

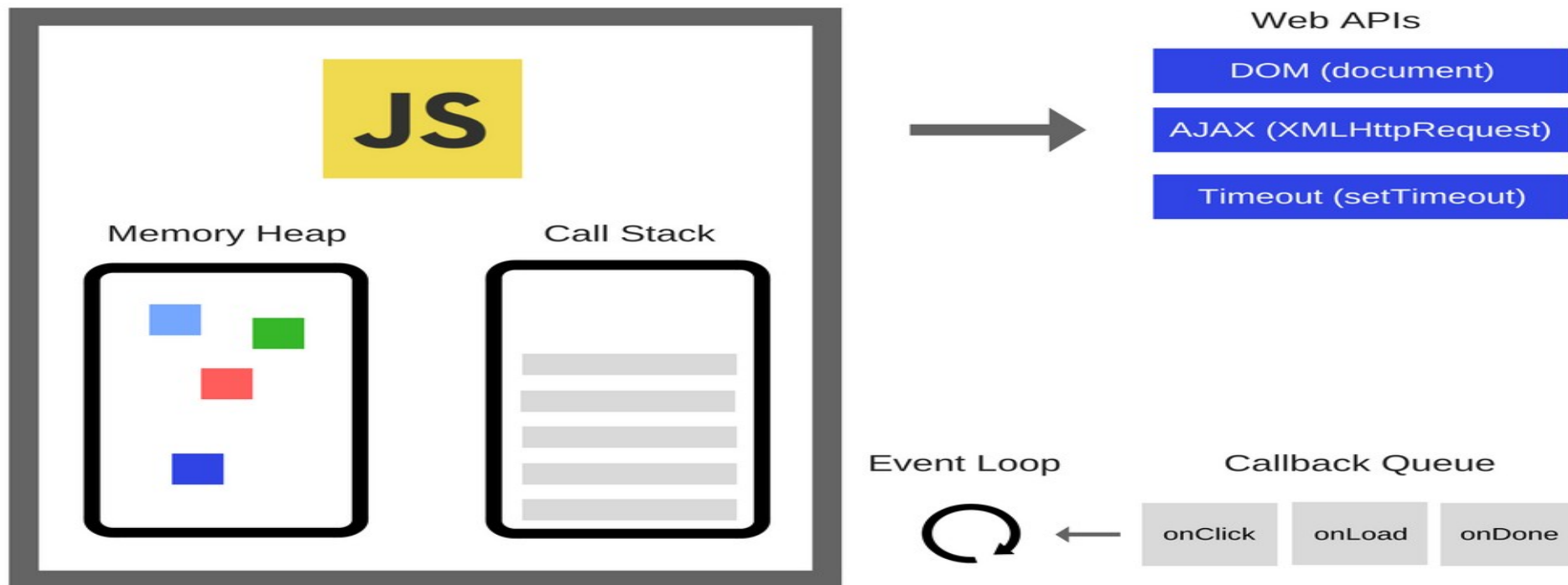
Positives: once you load it, it has fast response, moves some of computation to client side (in Angular)

How JavaScript work in Browser

What you are JavaScript?

I have a call stack, an event loop, a callback queue
and other apis and stuff

How JavaScript work in Browser



How JavaScript work in Browser

Concurrency & the Event Loop

One thing at a time,
except not really.

blocking

What happens when things are slow?

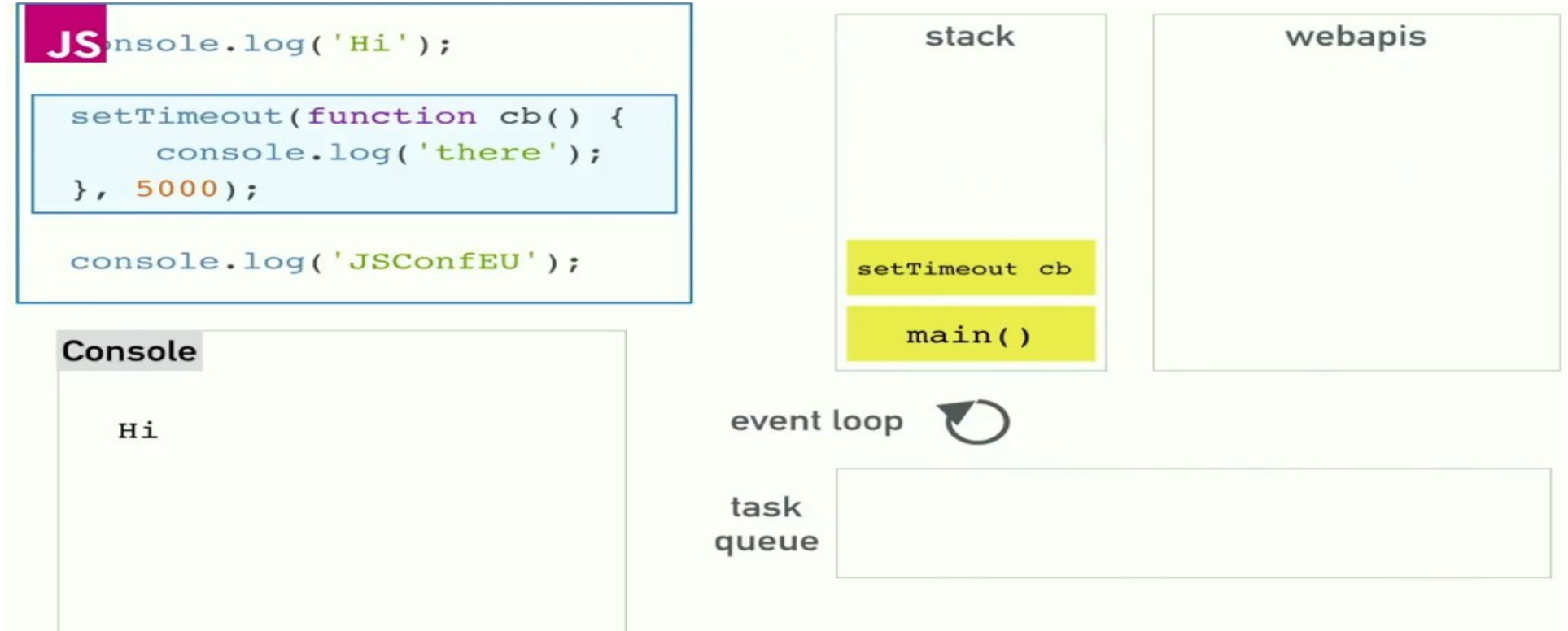
the solution? asynchronous callbacks

Here's a function,
Call me maybe?

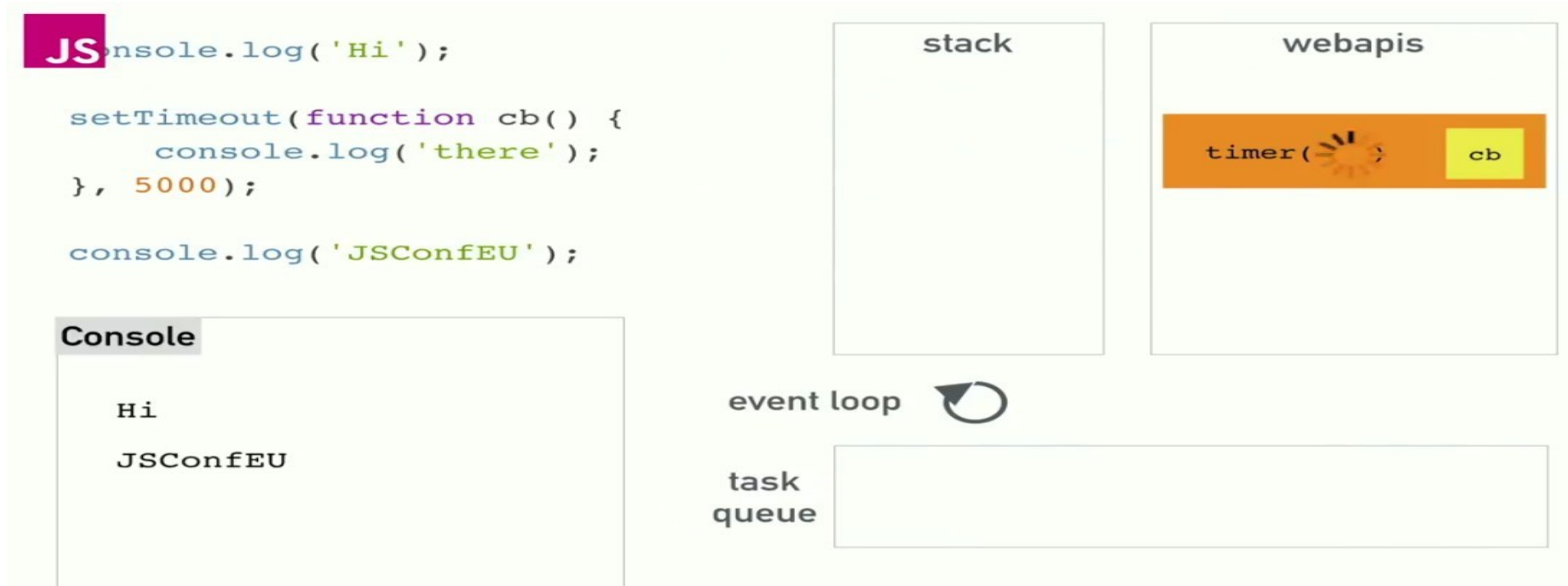
How JavaScript work in Browser

Javascript run in single thread but browser have Web API that can do other these, uses threads (in node we have c++ api)

How JavaScript work in Browser

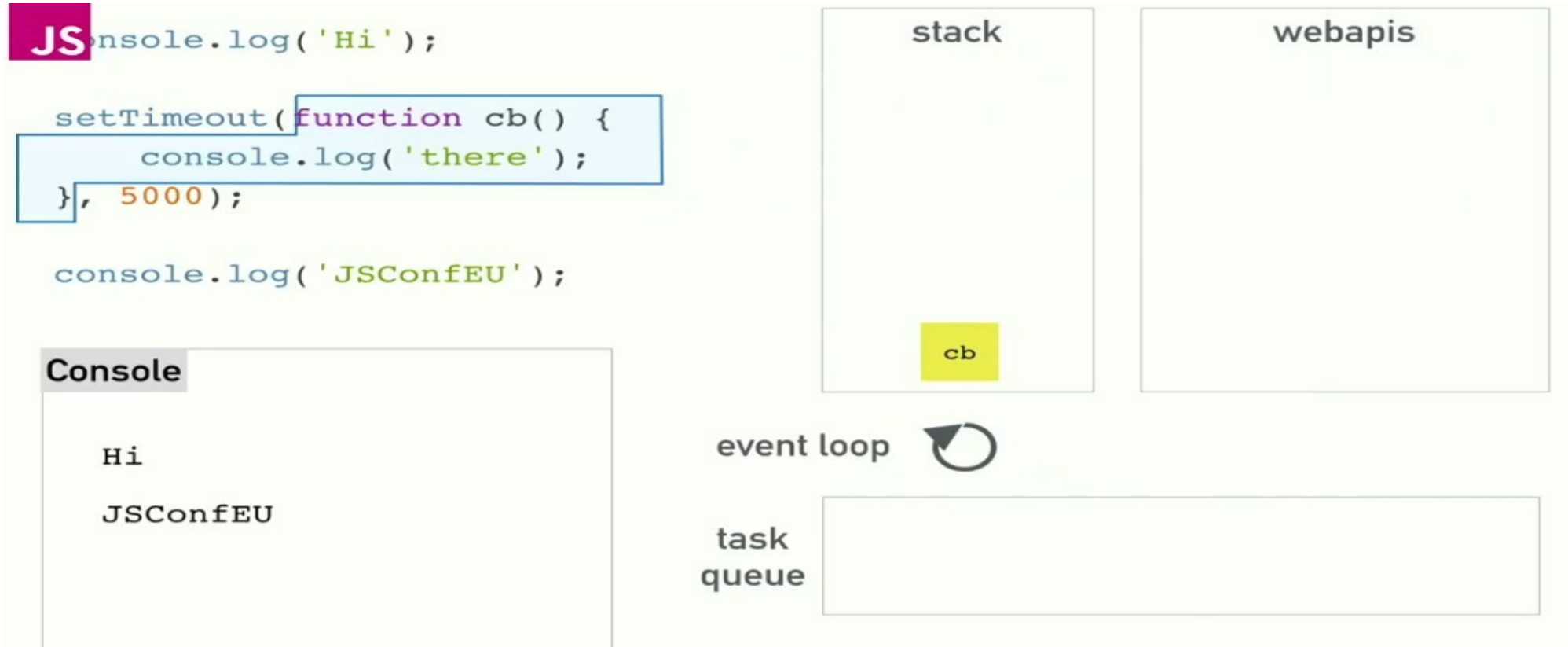


How JavaScript work in Browser

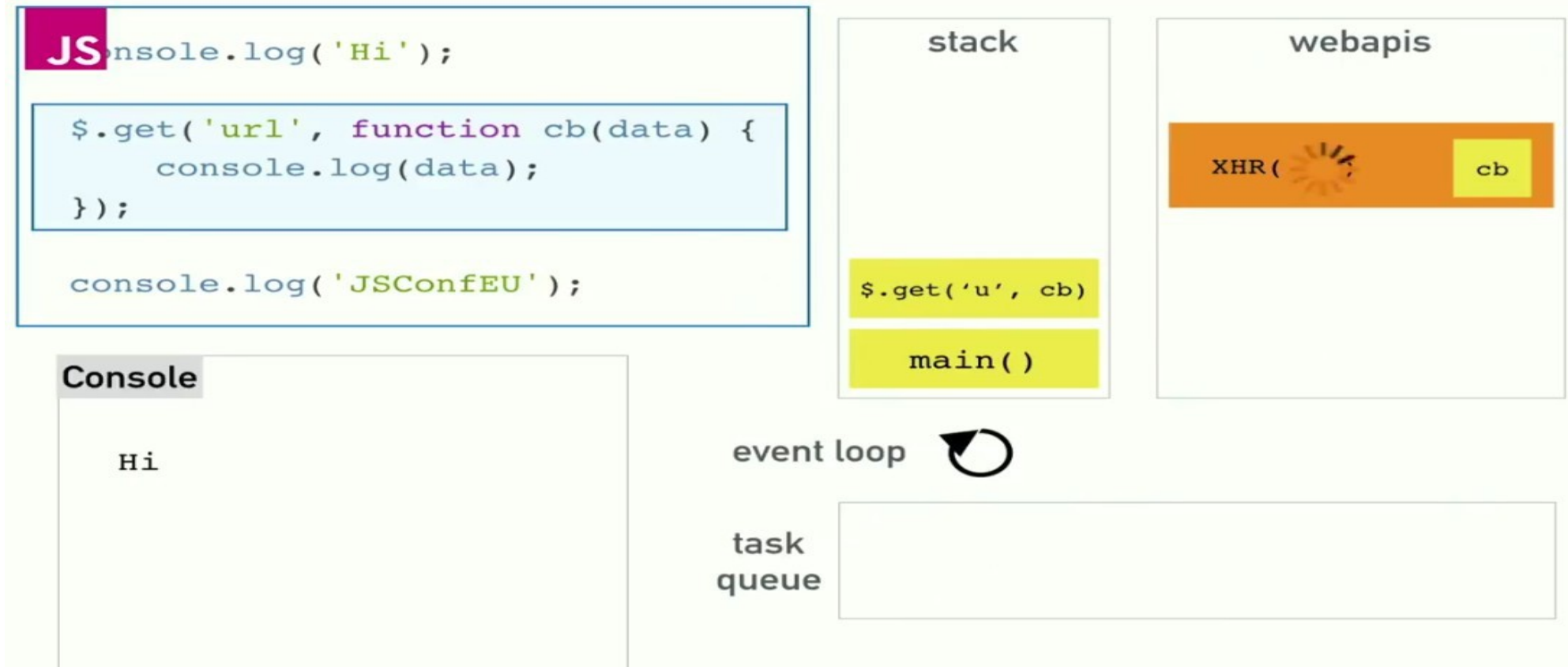


When webapis done there job they push call back function into the task queue, event loop have simple job once stack is empty, push that call back on the top of stack

How JavaScript work in Browser



How JavaScript work in Browser



What is node.js ?

Node.js is an open-source, cross-platform runtime environment used for the development of server-side web applications. Node.js applications are written in JavaScript and can be run on a wide variety of operating systems.

Node.js is based on an event-driven architecture and a non-blocking Input/Output API that is designed to optimize an application's throughput and scalability for real-time web applications.

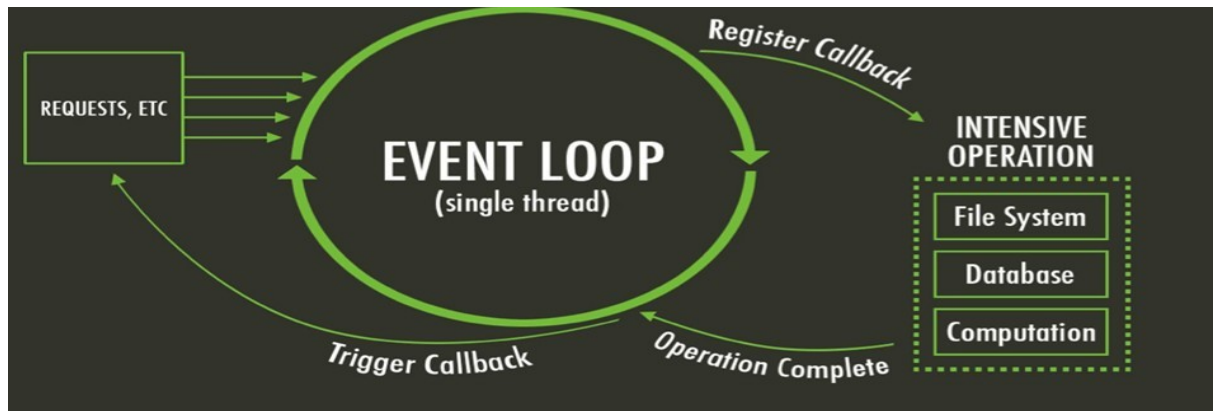
- Created in 2009
- Evented I/O for JavaScript
- Server Side JavaScript
- Runs on Google's V8 JavaScript Engine

Why Use Node.js ?

- Node's goal is to provide an easy way to build scalable network program
- Asynchronous event-driven IO helps concurrent request handling
 - This feature basically means that if a request is received by Node for some Input/Output operation, it will execute the operation in the background and continue with processing other requests.
- Node uses the V8 JavaScript Runtime engine
- The Node.js library uses JavaScript
- Active and vibrant community for the Node.js framework

What is unique about Node.js?

- 1. JavaScript on server-side thus making communication between client and server will happen in same language
- 2. Servers normally thread based but Node.JS is “Event” based. Node.JS serves each request in a Evented loop that can handle simultaneous requests



When to go for Node?

When to Use Node.js

Node.js is best for usage in streaming or event-based real-time applications like Chat applications

Game servers – Fast and high-performance servers that need to process thousands of requests at a time, then this is an ideal framework.

Good for collaborative environment – This is good for environments which manage documents. In a document management environment, you will have multiple people who post their documents and do constant changes by checking out and checking in documents. So Node.js is good for these environments because the event loop in Node.js can be triggered whenever documents are changed in a document managed environment.

Advertisement servers – Again here you could have thousands of request to pull advertisements from the central server and Node.js can be an ideal framework to handle this.

Streaming servers – Another ideal scenario to use Node is for multimedia streaming servers wherein clients have request's to pull different multimedia contents from this server

When not to go for Node?

The only scenario where it should not be used is where there are long processing times, which is required by the application.

Node is structured to be single-threaded. If an application is required to carry out some long-running calculations in the background, it won't be able to process any other requests. As discussed above, Node.js is used best where processing needs less dedicated CPU time. r

Threads VS Event-driven


Threads	Asynchronous Event-driven
Lock application / request with listener-workers threads	only one thread, which repeatedly fetches an event
Using incoming-request model	Using queue and then processes it
multithreaded server might block the request which might involve multiple events	manually saves state and then goes on to process the next event
Using context switching	no contention and no context switches
Using multithreading environments where listener and workers threads are used frequently to take an incoming-request lock	Using asynchronous I/O facilities (callbacks, not poll/select or O_NONBLOCK) environments


Installation

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Mature and Dependable

Stable
Latest Features

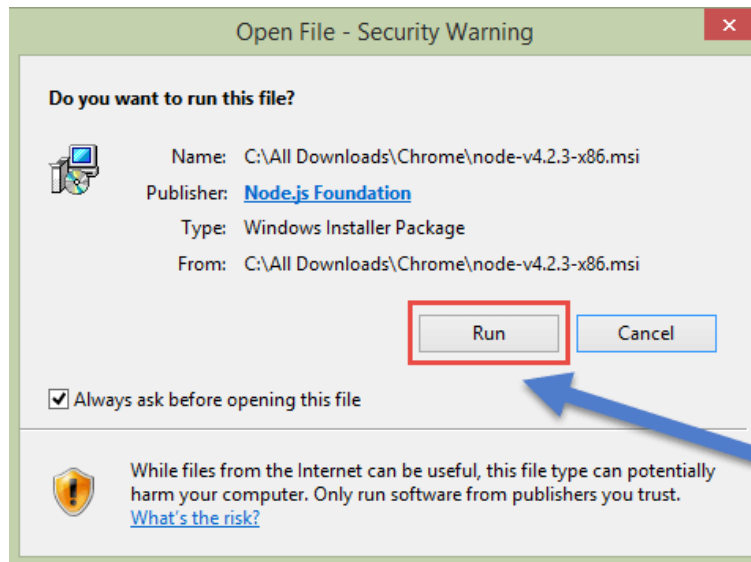

Windows Installer
node-v4.2.3-x86.msi


Macintosh Installer
node-v4.2.3.pkg

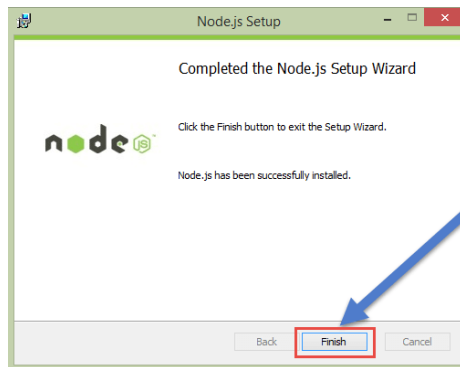

Source Code
node-v4.2.3.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	64-bit	
Mac OS X Binaries (.tar.gz)	64-bit	

Download the 32-bit installer



Click the Run button



Click the Finish button to complete the installation

Node.js VS Apache

- It's faster
- It can handle tons of concurrent requests

Platform	Number of request per second
PHP (via Apache)	3187,27
Static (via Apache)	2966,51
Node.js	5569,30

Hello world Nodejs

- Printing loop

```
for(var i=0;i<=5; i++){  
    console.log('hello to nodejs');  
}
```

- Using modules

```
var PI = Math.PI;  
  
exports.area = function (r) {  
    return PI * r * r;  
};  
  
exports.circumference = function (r) {  
    return 2 * PI * r;  
};
```

```
var circle = require('./circle.js');  
var area = circle.area(4);  
console.log( 'The area of a circle of radius 4 is '+ area);
```

What are modules in Node.js?

modules in Node js are a way of encapsulating code in a separate logical unit

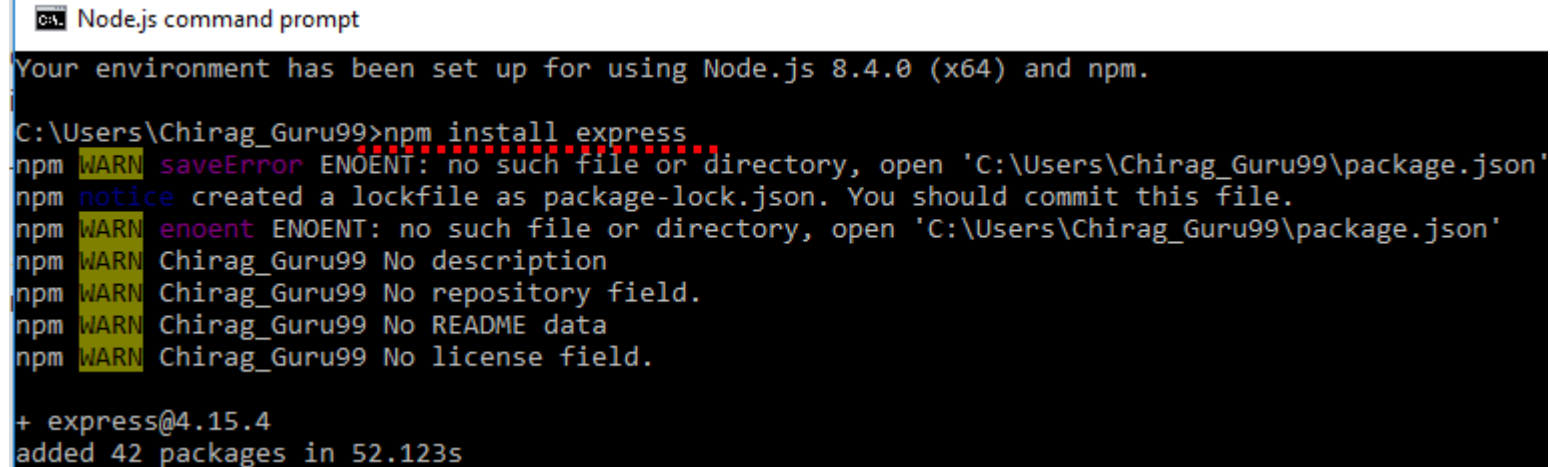
1. **Express framework** – Express is a minimal and flexible Node js web application framework that provides a robust set of features for the web and **mobile** applications.
2. **Socket.io** - Socket.IO enables real-time bidirectional event-based communication. This module is good for creation of chatting based applications.
3. **Jade** - Jade is a high-performance template engine and implemented with **JavaScript** for node and browsers.
4. **MongoDB** - The **MongoDB** Node.js driver is the officially supported node.js driver for MongoDB.
5. **Restify** - restify is a lightweight framework, similar to express for building REST APIs
6. **Bluebird** - Bluebird is a fully-featured promise library with a focus on innovative features and performance

How to install modules in Node.js?

In order to use modules in a Node.js application, they first need to be installed using the Node package manager.

The below command line shows how a module "express" can be installed.

npm install express



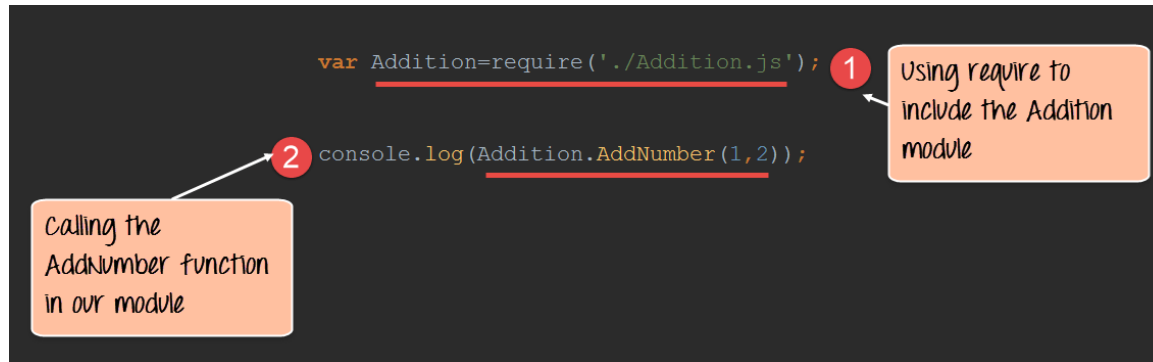
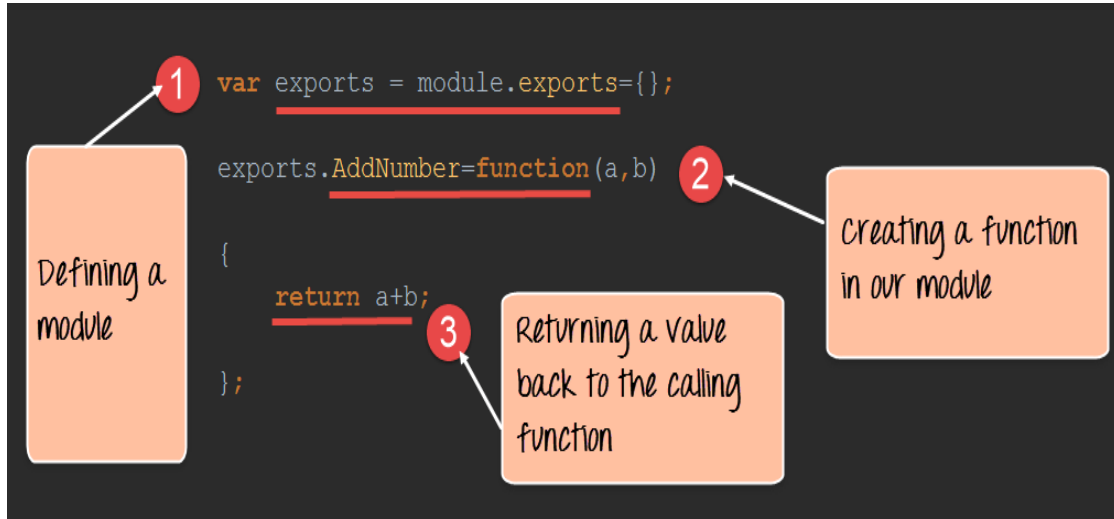
```
Node.js command prompt
Your environment has been set up for using Node.js 8.4.0 (x64) and npm.

C:\Users\Chirag_Guru99>npm install express
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Chirag_Guru99\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Chirag_Guru99\package.json'
npm WARN Chirag_Guru99 No description
npm WARN Chirag_Guru99 No repository field.
npm WARN Chirag_Guru99 No README data
npm WARN Chirag_Guru99 No license field.

+ express@4.15.4
added 42 packages in 52.123s
```

How to create Custom modules?

modules in Node js are a way of encapsulating code in a separate logical unit



Reading/ Writing data on file

- Writing

```
var fs = require('fs');  
var data = 'Hello World!';  
  
fs.writeFile('test.txt', data, function (err) {  
  if (err)  
    return console.log(err);  
  
  console.log('Hello World > test.txt');  
});  
console.log('testing');
```

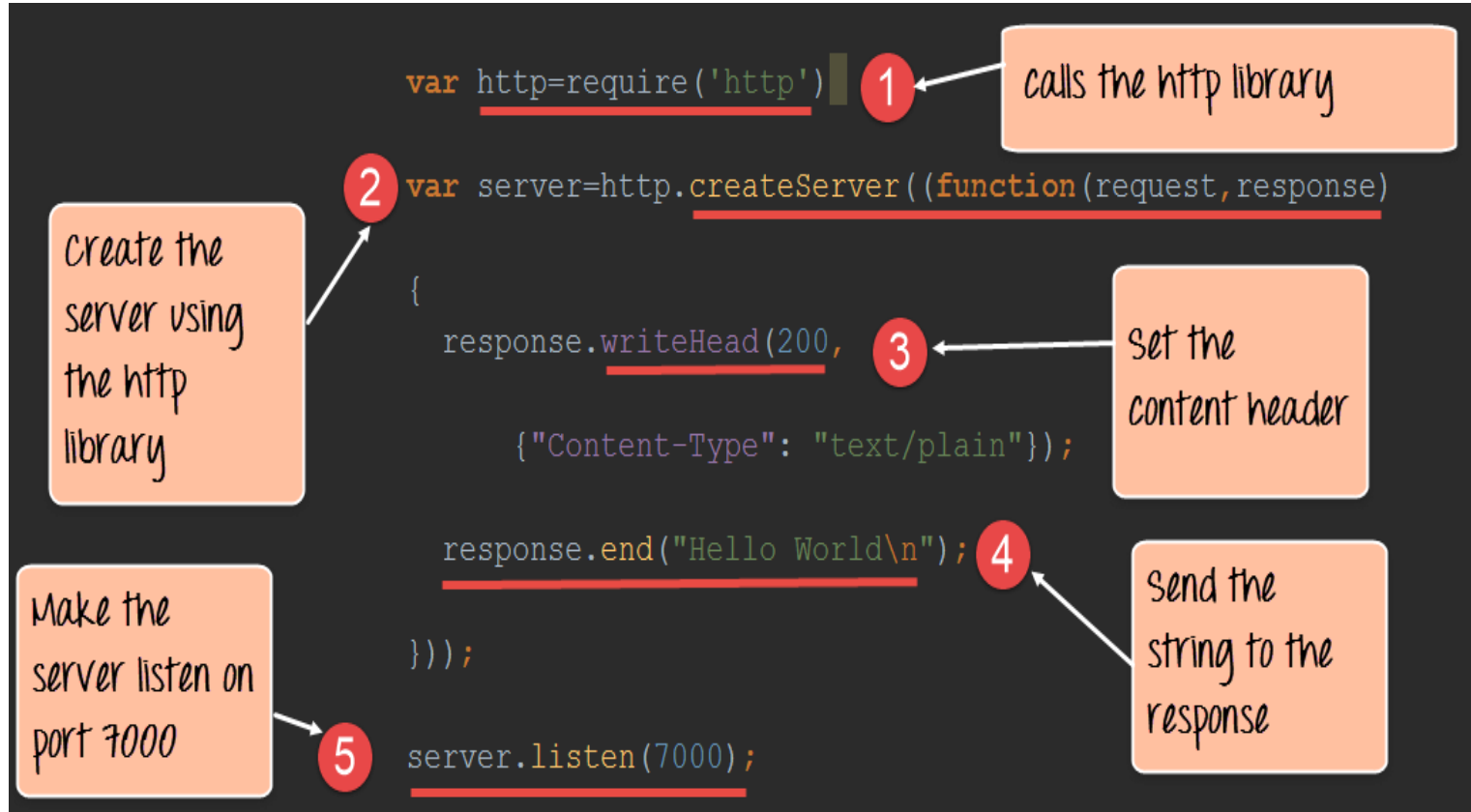
- Reading

```
var fs = require('fs');  
fs.readFile('test.txt', 'utf8', function (err,data) {  
  if (err) {  
    return console.log(err);  
  }  
  console.log(data);  
});
```

Create HTTP Web Server in Node.js

```
1 var http = require("http");
2
3 var srvr = http.createServer(function (request, response) {
4     //console.log(request);
5     response.writeHead(200, {'Content-Type': 'text/plain'});
6     response.end('Hello World\n');
7
8 });
9
10 srvr.listen(8081);
11
12 console.log('Server running at http://127.0.0.1:8081/');
```

Create HTTP Web Server in Node.js



Connect with mysql

```
//npm install mysql

var mysql      = require('mysql');|
var connection = mysql.createConnection({
  host      : 'localhost',
  user      : 'root',
  password  : 'root',
  database  : 'test'
});

connection.query('SELECT * from student', function(err, rows, fields) {
  if (!err){
    console.log('The solution is: ', rows);
    console.log('length:' + rows.length);

    for(var i=0;i<rows.length;i++){
      var row = rows[i];
      console.log(row.id + "," + row.name + "," + row.city + "," + row.age)
    }
  }
  else
    console.log('Error while performing Query.');
```

}); //end of query
connection.end();

File package.json

- Project information
- Name
- Version
- Dependencies
- Licence
- Main file
- Etc...

```
"name": "node-js-getting-started", "version": "0.2.5",
"description": "A sample Node.js app using Express 4", "engines":
{
  "node": "5.9.1"
},
"main": "index.js", "scripts": {
  "start": "node index.js"
},
"dependencies": {
  "body-parser": "^1.16.1",
  "cookie-parser": "^1.4.3",
  "cool-ascii-faces": "1.3.4",
  "ejs": "2.4.1",
  "express": "^4.13.3",
  "express-session": "^1.15.1",
  "mongodb": "^2.2.24",
  "multer": "^1.3.0",
  "pg": "4.x",
  "pug": "^2.0.0-beta11"
},
"repository": { "type": "git",
"url": "https://github.com/heroku/node-js-getting-started"
},
"keywords": [
  "node",
  "heroku", "express"
],
"license": "MIT"
```

Node.js Modules..... MANY

- <https://npmjs.org/>
- # of modules = 1,21,943
- Install a module.....inside your project directory
- \$npm install <module name>
- Using module..... Inside your javascript code
- `var http = require('http');`
- `var fs = require('fs');`
- `var express = require('express');`

npm is the package manager for **javascript**.



1,21,943
total packages



4,08,79,678
downloads in the last day



22,86,36,931
downloads in the last week



82,36,52,154
downloads in the last month

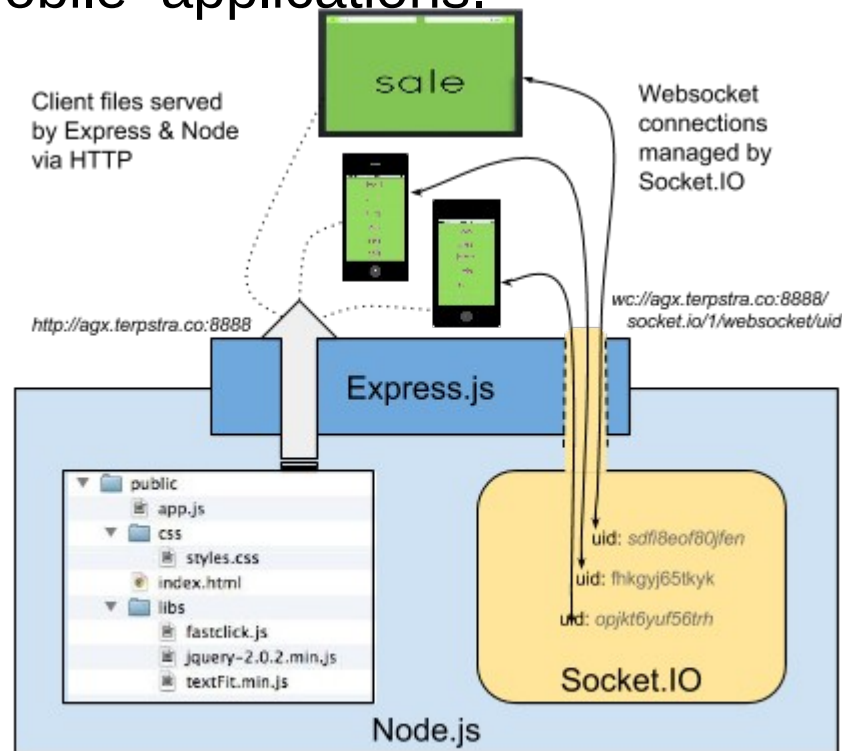
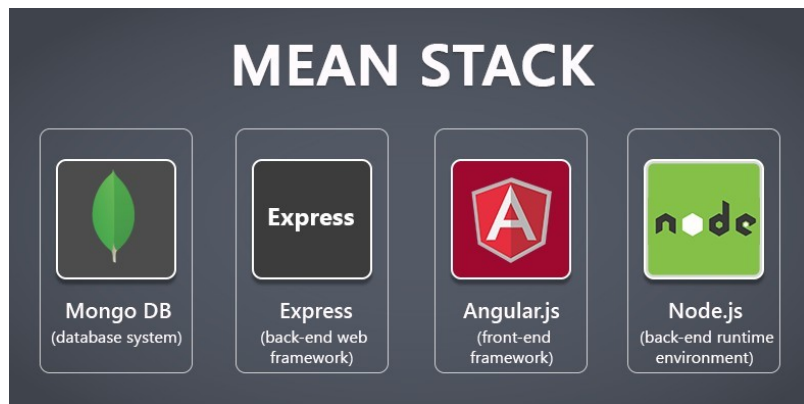
Hello World Example

- Create file index.js with the following code:
- `http.createServer(function (request, response) {`
- `// Send the HTTP header`
- `// HTTP Status: 200 : OK`
- `// Content Type: text/plain response.writeHead(200, {'Content-Type':`
- `'text/plain'});`
- `// Send the response body as "Hello World" response.end('Hello World\n'); }).listen(8081);`
- `// Console will print the message`
- `console.log('Server running at http://127.0.0.1:8081/');`

`$ node app.js`

Express

- minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.



Express

- Express gives ease of functionality
- Routing
- Delivery of Static Files
- “Middleware” – some ease in development (functionality)
- Form Processing
- Simple forms of Authentication
- Cookies and Session Manipulation

A lot of this you can do in NodeJS but, you may write more code to do it than if you use the framework Express.

Hello world

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

This says requires module express

```
var app = express()
```

Calls function express to initialize object app

```
app.get('/', function (req, res) {
```

```
  res.send('Hello World!')
```

```
})
```

```
app.listen(3000, function () {
```

```
  console.log('Example app
```

```
  listening on port 3000!')
```

```
})
```

App object has various methods like get that responds to HTTP get request. This code will be call the function specified when a GET for the URI / is invoked

Sets up the HTTP server for listening port 3000

Callback Function

A callback function is a function (It can be any function Anonymous Function, Arrow Function) passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

```
function show(){
    console.log("I am show Function");
}
function geeky(callback){
    callback();
}
geeky(show);
```

```
function show(a){
    console.log("I am show Function" + a);
}
function geeky(callback){
    var a = 101;
    callback(a);
}
geeky(show);
```

```
function show(a){  
    console.log("I am show Function" + a);  
}  
function geeky(callback){  
    var a = 101;  
    callback(a);  
}  
geeky(show);
```

```
function geeky(callback) {  
    var a = 101;  
    callback(a);  
}  
geeky(function(a) {  
    console.log("I am show Function " + a);  
});
```

```
function geeky(callback) {  
    var a = 101;  
    callback(a);  
}  
geeky(function show(a) {  
    console.log("I am show Function " + a);  
});
```

```
function geeky(callback) {  
    var a = 101;  
    callback(a);  
}  
geeky(a => console.log("I am show Function " + a));
```

```
function show(){  
    console.log("I am show Function");  
}  
function geeky(callback){  
    callback();  
}  
  
geeky(show);  
  
console.log("End");
```

Synchronous - It waits for each operation to complete, after that it executes the next operation.

```
setTimeout(function show(){  
    console.log("I am show Function");  
}, 5000);  
  
console.log("End");
```

Asynchronous - It never waits for each operation to complete, rather it executes all operations in the first GO only.