Lecture 7.1

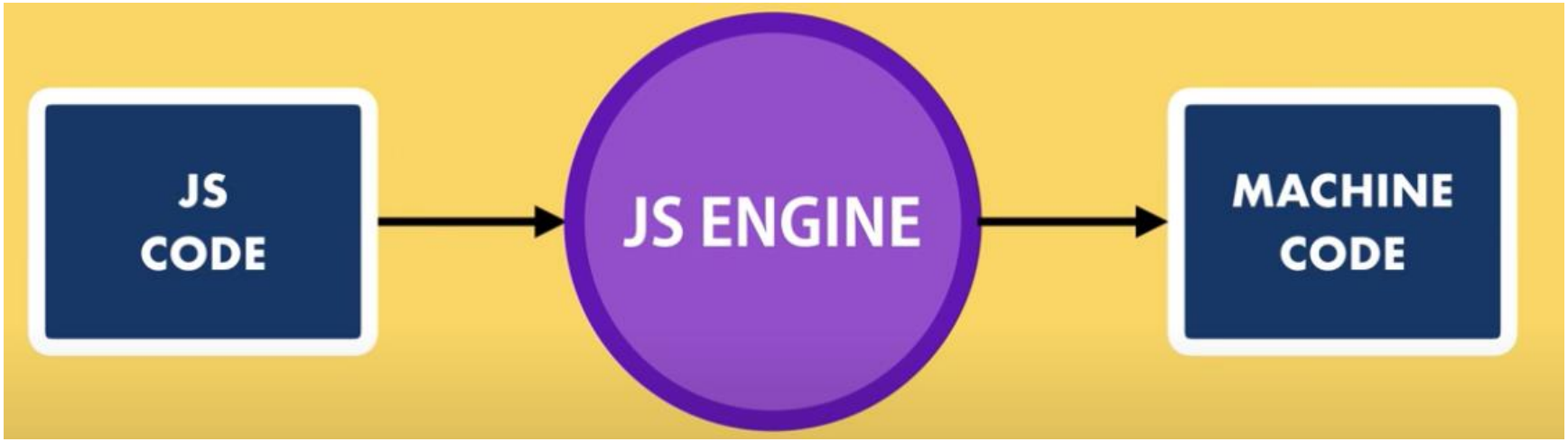
Introduction to Node.JS

Course Instructor
Engr. Madeha Mushtaq

Node JS

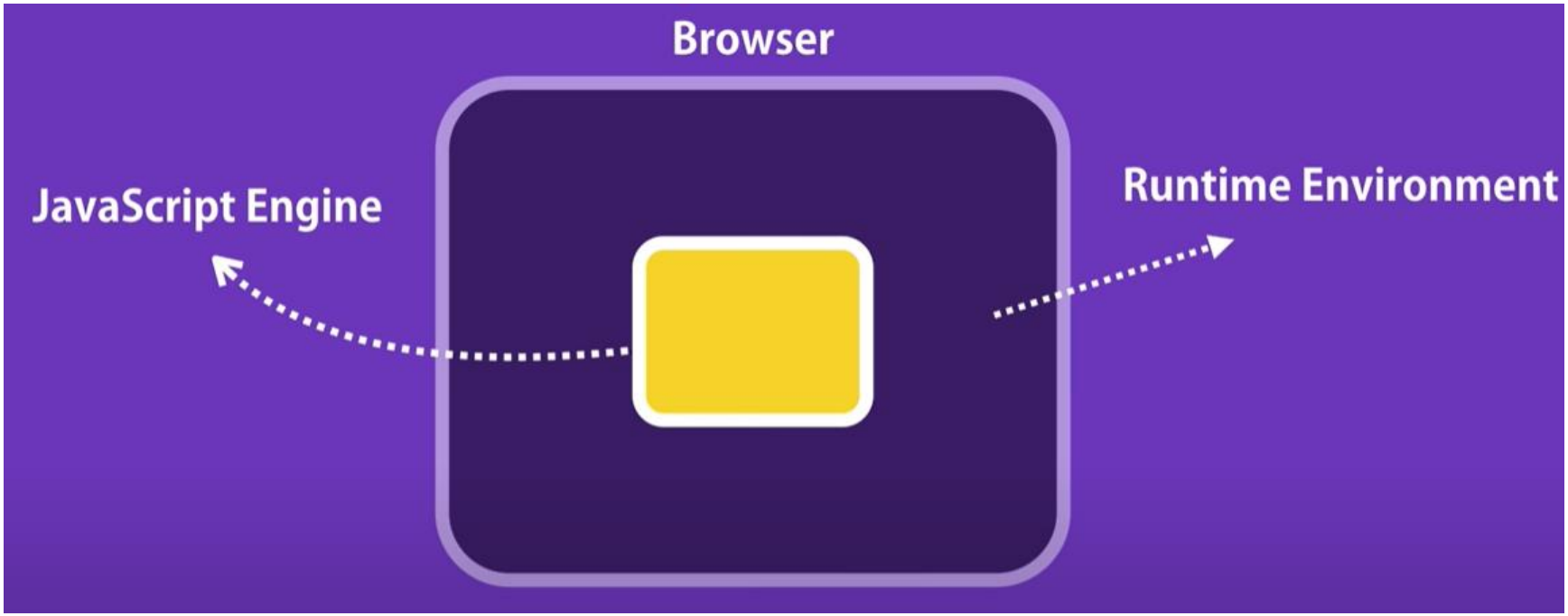
- Node.JS is an open source, cross platform runtime environment for executing Javascript code outside of a browser.
- Server-side solution for JS.
- Node is ideal for building highly scalable applications.
- Created by Ryan Dahl starting in 2009
- Node.js is a platform (is not a framework), Express is a framework for Node.js.

Node JS Architecture



Node JS Architecture

Node JS Architecture



Node JS Architecture

Why Node JS

- Great for prototyping and agile development
- Superfast and highly scalable
- JavaScript everywhere
- Cleaner and more consistent codebase
- Large ecosystem of open-source libs

Why Node JS

NODE APP

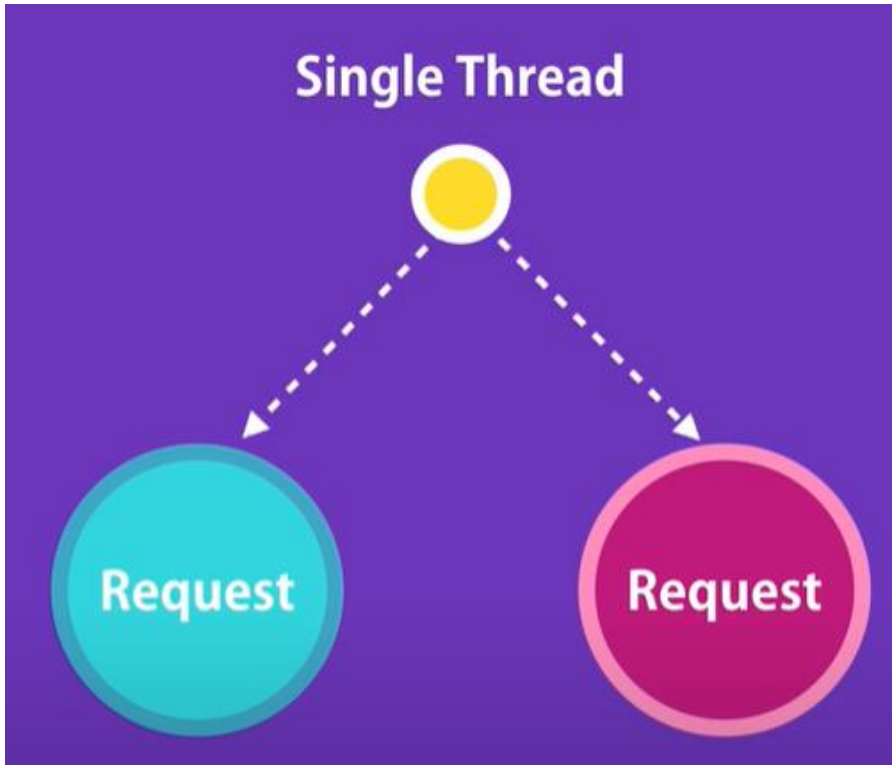
- **Built twice as fast** with fewer people
- **33%** fewer lines of code
- **40%** fewer files
- **2x** request/sec
- **35%** faster response time

Node JS

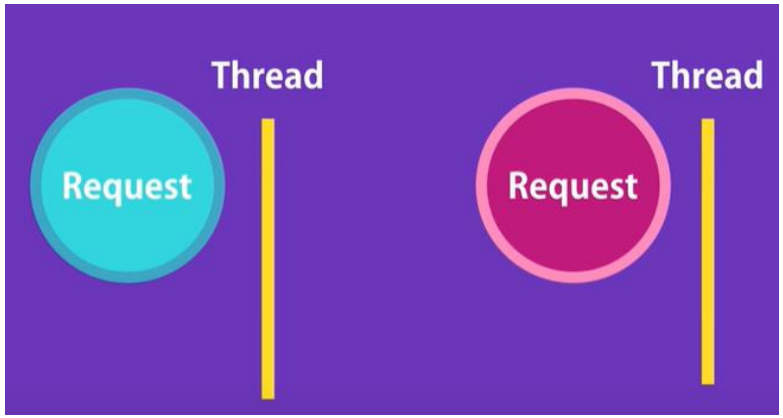
- Node JS provide an easy way to build scalable network applications.
- Node is a platform for writing JavaScript applications outside web browsers.
- There is no DOM built into Node, nor any other browser capability.
- Because of the nature of C, Node can perform amazingly well when dealing with networking and OS system calls.

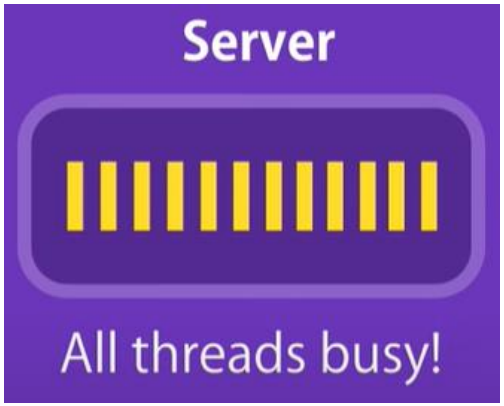
Asynchronous Architecture

- Node applications are highly scalable because of the Non blocking or Asynchronous nature of node.
- Node architecture is asynchronous.



Synchronous/Blocking Architecture



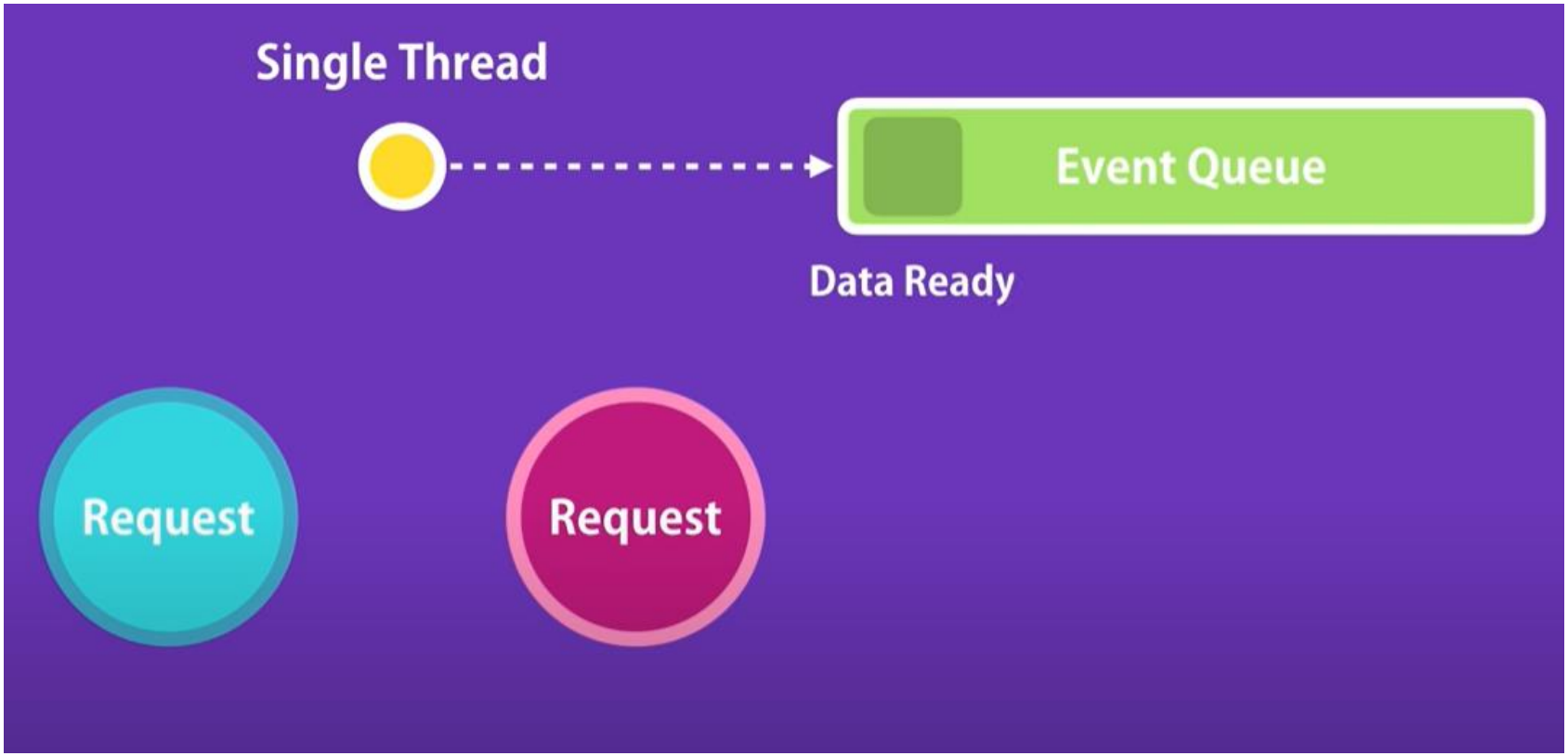


New Requests have to wait

Or add more Hardware to avoid waiting

Concurrency: Event Loop

- A single thread is used to handle multiple requests.

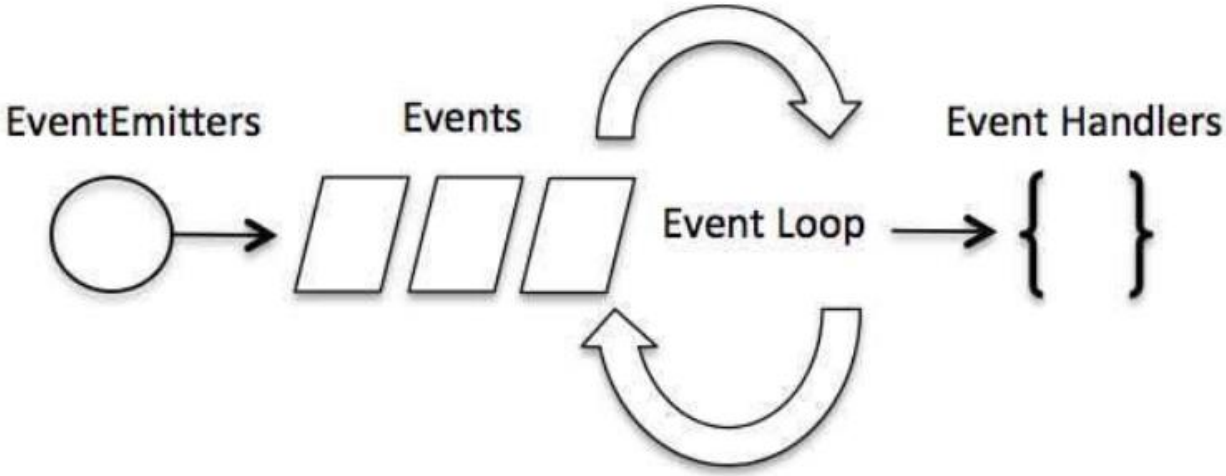


Concurrency: Event Loop

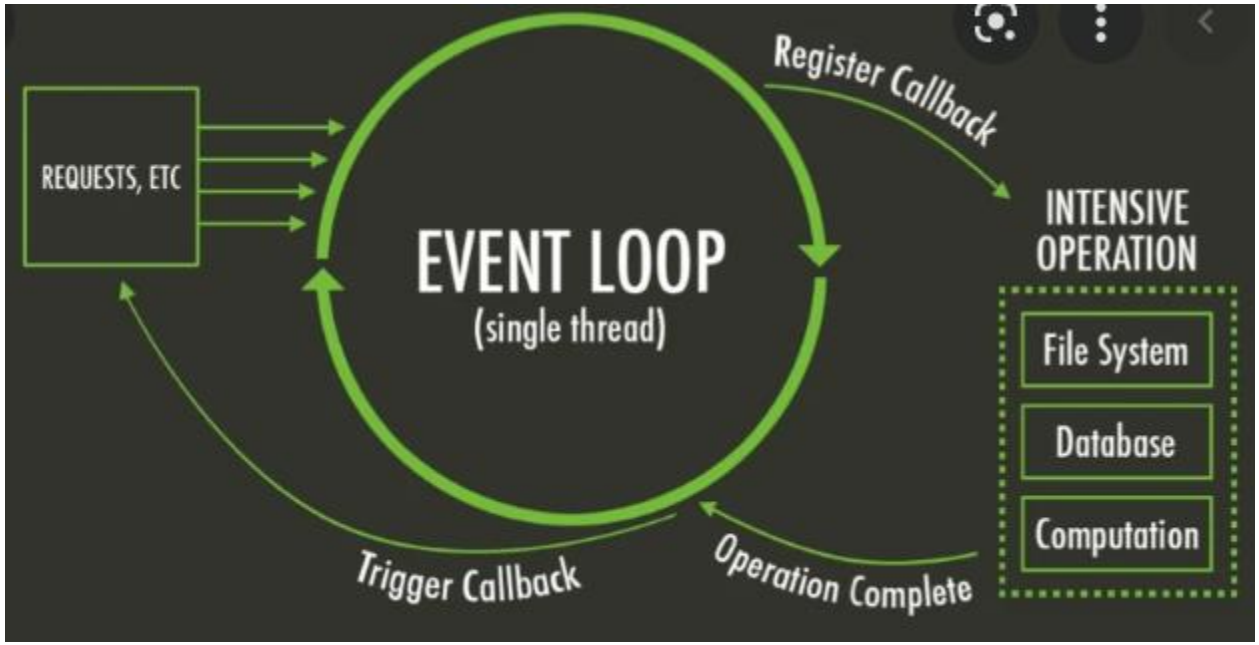
- Node uses an event loop with a stack
- Alleviates overhead of context switching.
- It process each request as events E.g., HTTP server doesn't wait for the I/O operation to complete while it can handle other request at the same time.
- To avoid blocking, it makes use of the event driven nature of JavaScript by attaching callbacks to I/O requests.

Concurrency: Event Loop

- Event Loops are the core of event driven programming, almost all the UI programs use event-loops to track the user events. For example clicks, Ajax.
- Event Loop means single threaded infinite cycle.



Concurrency: Event Loop



Things to Remember…

- Node JS is ideal for I/O intensive applications.
- Node provides high performance for real time applications.
- Don’t use Node JS for CPU intensive applications.

Node Installation

- Download Node JS from https://nodejs.org/en
- Install the latest stable version that is recommended for most users.

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download for Windows (x64)

18.16.0 LTS
Recommended For Most Users

20.2.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Other Downloads | Changelog | API Docs

For information about supported releases, see the release schedule.

Node Installation

- After installation you can check the version:

Command Prompt

```
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>node --version
v18.16.0
```

First Node JS Code

node1.js

```
function sayhello(name)
{
console.log('Hello '+ name);
}
sayhello('Madeha');
```

- To run the code, go to the directory where the file is stored using command prompt.

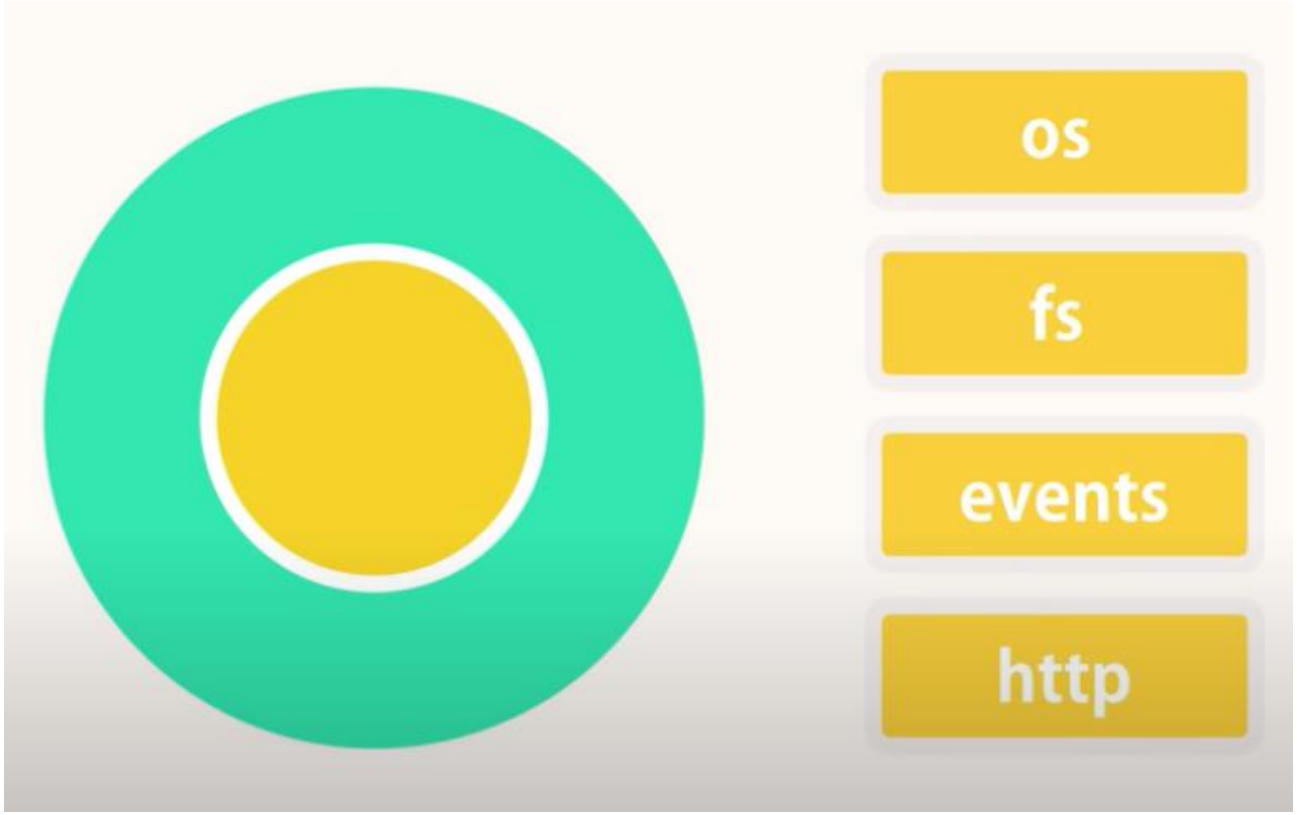
```
C:\Node>node node1.js
Hello Madeha

C:\Node>
```

Node Module System

- Node.js includes a core set of modules for many types of network and file I/O.
- Building blocks for I/O-based applications
  - HTTP, HTTPS, filesystem, UDP, and NET (TCP).
- The core is intentionally small, low-level, and uncomplicated.
- Third-party modules build upon these blocks to offer greater abstractions for common problems.

Node Module System

- These are some of the modules built into the core of Node.



Node Module System

- Node.js heavily relies on modules, creating a module is easy, just put your JavaScript code in a separate js file and include it in your code by using keyword require, like:
- var modulex = require(‘./modulex’);
- Libraries in Node.js are called packages and they can be installed by using NPM (Node Package Manager).

Node Module System

- NPM (Node Package Manager) comes bundled with Node.js installation.
- Each modules can be bundled under a single package.
- NPM is used to install, update, uninstall and configure Node JS Platform modules/packages very easily.
- npm install ‚package_name‘;

Node Module System

- A module in node.js can be exported, and it can be imported and used in other files.
- Exporting
  - module.exports{module to be exported}
- Importing
  - The "require" function helps in importing modules.
  - var varname = require(‘module name’);

Node Path Module

Path.js

```
1  const path = require('path');
2
3  var pathobj = path.parse(__filename);
4
5  console.log(pathobj);
```

```
C:\Node>node path
{
  root: 'C:\\',
  dir: 'C:\\Node',
  base: 'path.js',
  ext: '.js',
  name: 'path'
}

C:\Node>
```

Node OS Module

OS.js

```
const os = require('os');

var totalmemory = os.totalmem();

var freememory = os.freemem();

console.log('Total Memory: ' + totalmemory);

//Template String
console.log(`Free Memory: ${freememory}`);
```

```
C:\Node>node os
Total Memory: 8482717696
Free Memory: 2393616384

C:\Node>
```

Node FS Module

- **Synchronous readdir:**
- Fs.readdirsync() is used to synchronously read the contents of a given directory. It returns an array with all the file names or objects in the directory. It accepts two parameters, path and options.

FS.js

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

const files = fs.readdirSync('./');
console.log(files);

```

```
C:\Node>node FS
[ 'FS.js', 'Logger.js', 'node1.js', 'OS.js', 'Path.js' ]

C:\Node>
```

Node FS Module

- **Asynchronous readdir:**
- Fs.readdir() first parameter is the path and second is a callback function.
- The function has two parameters, first one is errors, second is the result.

```
fs.readdir('./', function (err, files){
    if(err) console.log('Error', err);
    else console.log('Result', files);
});
```

Node Events Module

- One of the core concept of node is the concept of Event.
- A lot of Node's core functionality is based upon this concept (concept of events).
- An event is basically a signal that indicates that something has happened in our application.
- Node has Events module to work with events.
- Eventemitter class is one of the core building blocks of node. And a lot of classes are based on this eventemitter class.

Node Events Module

event.js

```
1   const EventEmitter = require('events');
2   const emitter = new EventEmitter();
3
4   //Register a Listener
5   emitter.on('messageLogged',function(){
6       console.log('Listener called');
7   });
8
9   //Raise an event
10  emitter.emit('messageLogged');
```

```
C:\Node>node event
Listener called

C:\Node>
```

Node Events Module

- Event Arguments:

event_arg.js

```
const EventEmitter = require('events');
const emitter = new EventEmitter();

//Register a Listener
emitter.on('messageLogged',function(arg){
    console.log('Listener called', arg);
});

//Raise an event
emitter.emit('messageLogged', {id: 1, url: 'http://'});
```

```
C:\Node>node event_arg
Listener called { id: 1, url: 'http://' }

C:\Node>
```

```
//Register a Listener
emitter.on('messageLogged',(arg) =>{
    console.log('Listener called', arg);
});
```

Node Events Module

- Extending EventEmitter Class:

event_arg.js | Logger.js

```
1  const EventEmitter = require('events');
2
3  class Logger extends EventEmitter{
4      log(message){
5      console.log(message);
6
7      //Raise an event
8      this.emit('messageLogged', {id: 1, url: 'http://'});
9      }
10 }
11 module.exports = Logger;
```

event_extend_class.js | Logger.js

```
1  const EventEmitter = require('events');
2
3  const Logger = require('./Logger');
4  const logger = new Logger();
5
6  //Register a Listener
7  logger.on('messageLogged',(arg) =>{
8      console.log('Listener called', arg);
9  });
10
11 logger.log('message');
```

Node Http Module

http.js

```
1  const http = require('http');
2  const server = http.createServer((req, res) =>{
3      if(req.url === '/'){
4          res.write('Hello World');
5          res.end();
6      }
7
8      if(req.url === '/api/courses'){
9          res.write(JSON.stringify([1,2,3]));
10         res.end();
11     }
12 });
13 server.listen(3000);
14 console.log('Listening on port 3000...');
```

Node Http Module

- An HTTP server listening for requests at port 3000.

```
var http = require('http');
var fs = require('fs');
var url = require('url');
// Create a server
http.createServer( function (request, response) {
    // Parse the request containing file name
    var pathname = url.parse(request.url).pathname;
    // Print the name of the file for which request is made.
    console.log("Request for " + pathname + " received.");
    // Read the requested file content from file system
    fs.readFile(__dirname+pathname, function (err, data) {
        if (err) {
            console.log(err);
            // HTTP Status: 404 : NOT FOUND
            response.writeHead(404, {'Content-Type': 'text/html'});
        } else {
            //Page found
            // HTTP Status: 200 : OK
            response.writeHead(200, {'Content-Type': 'text/html'});
            // Write the content of the file to response body
            response.write(data.toString());
        }
        // Send the response body
        response.end();
    });
}).listen(3000);
// Console will print the message
console.log('Listening on port 3000...');
```

Node Http Module

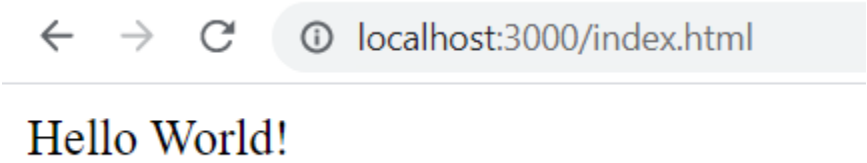
- Create an html file named index.htm in the same directory where you created the server.

index.html

```html
<html>
    <head>
        <title>Sample Page</title>
    </head>

    <body>
        Hello World!
    </body>
</html>
```

Node Http Module

- Make a request to Node.js server in any browser.



- Verify the Output at server end.

```
C:\Node>node http_server
Listening on port 3000...
Request for /index.html received.
```

- Error Message.

```
Request for / received.
[Error: ENOENT: no such file or directory, open ''] {
  errno: -4058,
  code: 'ENOENT',
  syscall: 'open',
  path: ''
}
```