MODULE 2

SERVERSIDE JAVASCRIPTING WITH NODE.JS

What is Node.JS

> Allows you to build scalable network applications using JavaScript on the server-side.

It's fast because it's mostly C code

What Could You Build?

Like a chat server

> Websocket Server
> Fast File Upload Client
> Ad Server
> Any Real-Time Data Apps

What is NOT Node.JS

> A Web Framework
> For Beginners *It's very low level*
> Multi-threaded

You can think of it as a single threaded server

Blocking vs Non-Blocking Processing

> Blocking Code

Stop process until complete

```
var contents = fs.readFileSync('/etc/hosts');
console.log(contents);
console.log('Doing something else');
```

> Non-Blocking Code

```
fs.readFile ('/etc/hosts',
   function(err,contents){
      console.log(contents);
   }
);
console.log('Doing something else');
```

Callback Syntax

```
fs.readFile ('/etc/hosts',
   function(err,contents){
      console.log(contents);
   }
);
```

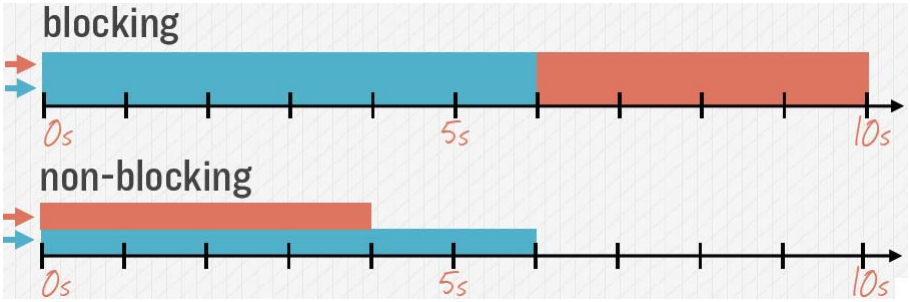
Same as

```
var callback= function(err,contents){
      console.log(contents);
}

fs.readFile ('/etc/hosts',  callback);
```

Blocking vs Non-Blocking

```
fs.readFile ('/etc/hosts',
   function(err,contents){
      console.log(contents);
   }
);
console.log('Doing something else');
```



Node.js : Hello Mars!

```
> hello.js
```

How we require modules

```
var http = require('http');
http.createServer( function(request, response) {
  response.writeHead(200);
```

Status code in header

```
  response.write("Hello Mars!");
```

Response body

```
  response.end();
```

Close the connection

```
}).listen(8080);
```

Listen for connections on this port

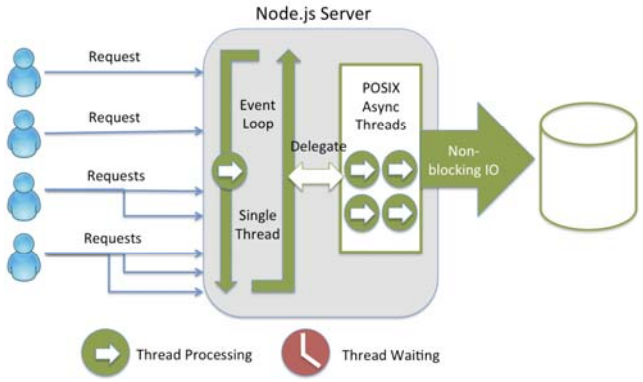
```
console.log('Listening on port 8080...');
```

Run the server

```
node hello.js
```

The Event Loop

```
var http = require('http');
http.createServer( function(request, response) {
  . . .
}).listen(8080);
console.log('Listening on port 8080...');
```



Creating the Node.js Application (1/3)

```
$ npm init
This utility will walk you through creating a
package.json file.
It only covers the most common items, and tries to guess
sane defaults.

See `npm help json` for definitive documentation on these
fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a
package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
```

Creating the Node.js Application (2/3)

```
name: (world)
version: (0.0.0)
description: World Countries Web App
entry point: (index.js)
test command:
git repository:
keywords:
author: Binnur Kurt
license: (ISC) BSD
About to write to c:\workspace\world\package.json:
```

Creating the Node.js Application (3/3)

```
{
  "name": "world",
  "version": "0.0.0",
  "description": “World Countries Web App",
  "main": "index.js",
  "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Binnur Kurt",
  "license": "BSD"
}


Is this ok? (yes) yes
```

Installing dependencies

```
npm install --save express mongoose nodemon
npm WARN package.json world@0.0.0 No repository field.
npm WARN package.json world@0.0.0 No README data
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/jade
npm http GET https://registry.npmjs.org/mongodb
npm http 304 https://registry.npmjs.org/jade
npm http 304 https://registry.npmjs.org/express
npm http 304 https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/accepts
npm http GET https://registry.npmjs.org/range-parser
npm http GET https://registry.npmjs.org/buffer-crc32
npm http GET https://registry.npmjs.org/methods
npm http GET https://registry.npmjs.org/parseurl
npm http GET https://registry.npmjs.org/proxy-addr
npm http GET https://registry.npmjs.org/cookie
. . .
```

Project Folders

> Create **public** and **server** folders