Node.js
“JavaScript has certain characteristics that make it very different than other dynamic languages, namely that it has no concept of threads. Its model of concurrency is completely based around events.” - Ryan Dahl
Event in the DOM

Dom Element Events

→ Click
→ Hover
→ MouseOver
→ ...

$("element").on("click", function(){ ... });
var http = require('http');

var server = http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
Event Loop

Node.js Processing Model

Clients send HTTP requests to Node.js server...

Event loop returns result to client

Responses are sent to main thread via callback

Event loop is woken up by OS, passes request + response objects as JavaScript closures to worker functions with callbacks

Non-blocking Worker (internal C++ threadpool)

Long-running jobs run on worker threads...
var http = require('http');

var server = http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
Module

```javascript
var http = require('http');  // http.js
var fs = require('fs');       // fs.js
```

How does `require` return the libraries?

How the libraries are structured?
Module

→ Use `require` to use other modules in your code
→ Use `exports` in your module to expose it publicly

```javascript
var other = require('other_module');
module.exports = function() {
  console.log(other.doSomething());
}
```
Module

How can you turn this code into a module?

```javascript
var http = require('http');
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```
```javascript
var http = require('http');

function start() {
  var server = http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('Hello World');
  });
  server.listen(8080);
}

exports.start = start;
```
Using the module

```javascript
var server = require("./server");
server.start();
```
Module

- Ordinary Javascript files following CommonJS module spec

- Run on their own scope so that they do not conflict with other modules

- Node facilitate module interoperability using some globals
Module

Module in file,

var var= require(‘./fileName’); → same directory

var var= require(‘../fileName’); → parent directory

var var= require(‘/dir/fileName’); → absolute directory

Module as a package,
should located in
/Home/mashiyat/my_app/node_module
Node Package Manager (npm)

→ npm is the official package manager for Node.js http://npmjs.org

→ Search for node module by,
  $ npm search packageName

→ If you know the module name you can install by, $ npm install moduleName
Node Package Manager (npm)

```json
{
    "author": "I.M. Awesome <awesome@example.com>",
    "name": "tmod",
    "description": "Test Module",
    "version": "0.0.1",
    "repository": {
        "url": ""
    },
    "engines": {
        "node": ">0.4.1"
    },
    "dependencies": {},
    "devDependencies": {}
}
```

package.json

CSC309
Node Version Manager (NVM)

Node is constantly evolving!
get nvm to manage your versions!