Lecture06: Basic Node.js

EGCI427

Reference: Apaichon Punopas, "NodeJS for Beginner", 28 January, 2018

Outline

- What is NodeJS ?
- Blocking I/O and Non Blocking I/O
- Node Package Manager
- Hello World Application
- List of NodeJS Library
- Express Library

What is NodeJS ?

- Open source , cross platform runtime environment for server side and networking application
- Written in JavaScript and Run on Linux, Mac, Windows, FreeBSD
- Provide an event driven architecture and a non blocking I/O that optimize and scalability
  - These technology uses for real time application.

What is NodeJS ?

- Use Google JavaScript V8 Engine to Execute Code
- Use by Groupon, SAP, LinkedIn, Microsoft, Yahoo, Walmart, Paypal

What is JavaScript ?

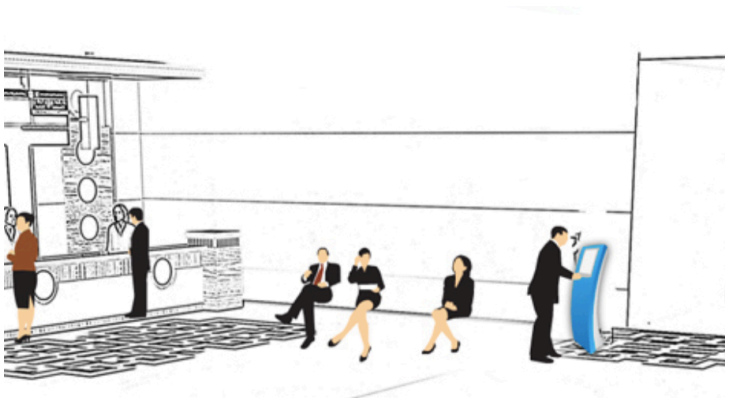
- Dynamic programming language
- Client side script to interact with user
- Most commonly used as a part of web browser (ES5)
- Better improvement in ES6

What is JavaScript ?

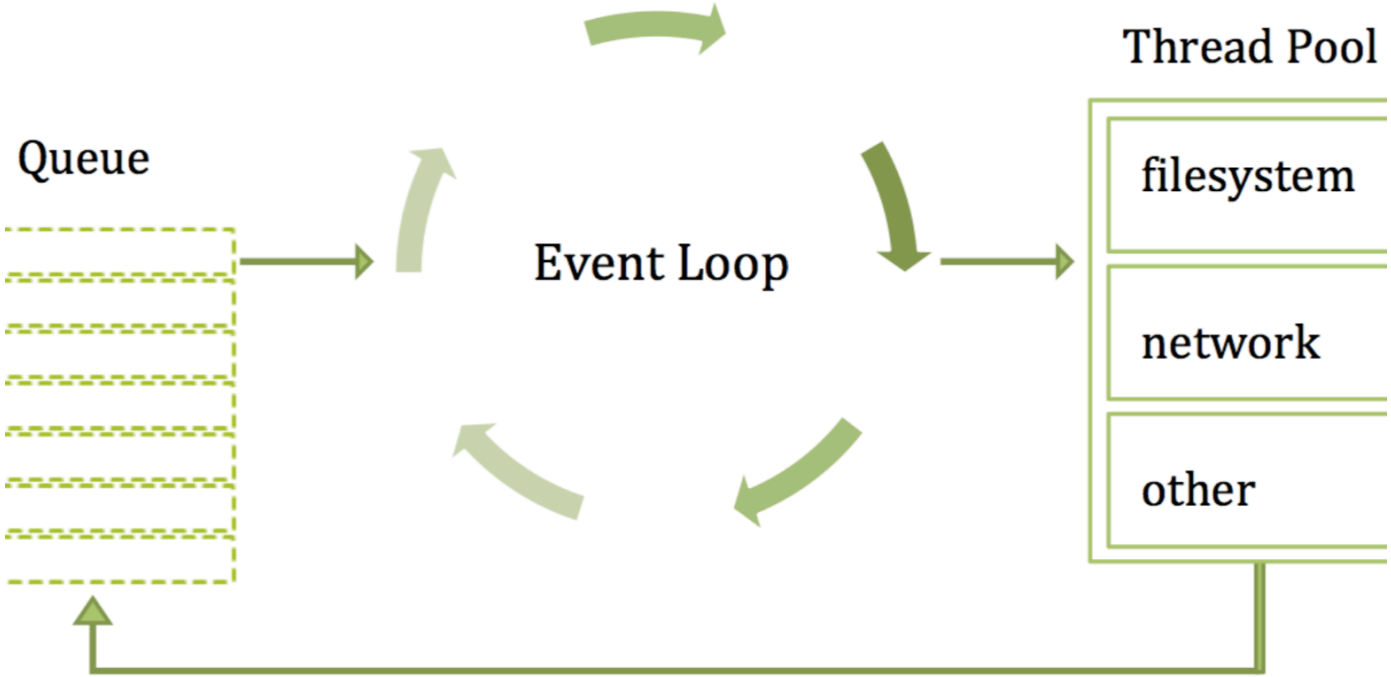
- Dynamic programming language
- Client side script to interact with user
- Most commonly used as a part of web browser (ES5)
- Better improvement in ES6

Blocking I/O & Non-Blocking I/O

- **Blocking** methods execute **synchronously** and **non-blocking** methods execute **asynchronously**.



NodeJS Event Loop



Javascript V8 Engine

- The V8 JavaScript Engine is an open source JavaScript engine developed by Google for the Google Chrome web browser.
- V8 compiles JavaScript to native machine code (IA-32,x86-64, ARM, or MIPS ISAs) before executing it.

NodeJS Pros. and Cons

Pros.

- Very light weight and fast.
- Easy to make to be real time
- Easy to configure.
- A lots of modules available for free
  - Ex. Node.js module for PayPal
- Work with NoSQL

NodeJS Pros. and Cons

Cons.

- Heavy CPU consuming algorithm/Job.
- Node.JS itself does not utilize all core of underlying system
  - Single threaded by default
  - Must write logic by your own to utilize multi core processor and make it multithreaded.

First App: Hello World

- Create example.js file
- Open Terminal then type node example.js

```
var http=require('http');
http.createServer(function(request, response)
{
  response.writeHead(200,{'Content-
   Type':'text/plain'});
  response.end('Hello World\n');
}).listen(8081);
console.log('Server running at
http://127.0.0.1:8124/');
```

Global Object

- *__filename* – return current file's name
- *__dirname* – return current directory.
- *module.exports*
  - Use for defining what a module exports and makes available through require()
- *setTimeout(cb, ms)*
  - Run callback cb after at least ms (milliseconds).
  - Timeout must be in the range of1-2,147,483,647 cannot span more than 24.8 days
- *clearTimeout(t)*
  - Stop a timer that was previously created withsetTimeout().

Global Object

Ex02

```
console.log(__filename);
console.log(__dirname);
var
t=setTimeout(helloWorld,3000);
clearTimeout(t);
```

Global Object

Ex02-1

- *setInterval(cb, ms)* – Run callback cb repeatedly every msmilliseconds.
- *clearInterval(t)* - Stop a timer that was previously created with setInterval().

```
var i =0;
var t Counter = setInterval(counter,2000);
function counter(){
  i++;
  console.log(i);
}
setTimeout(function(){
  clearInterval(Counter);
},10000);
```

Modules

Ex03

- External library and use by require()
- All libraries in nodemodules no need to refer path ‘./’ or ‘../’.

```
var circle = require('./circle.js');
console.log(circle.area(50));

// circle.js
var PI = Math.PI;
exports.area = function(r) {
  return PI*r*r;
};
```

Ex03--Practice

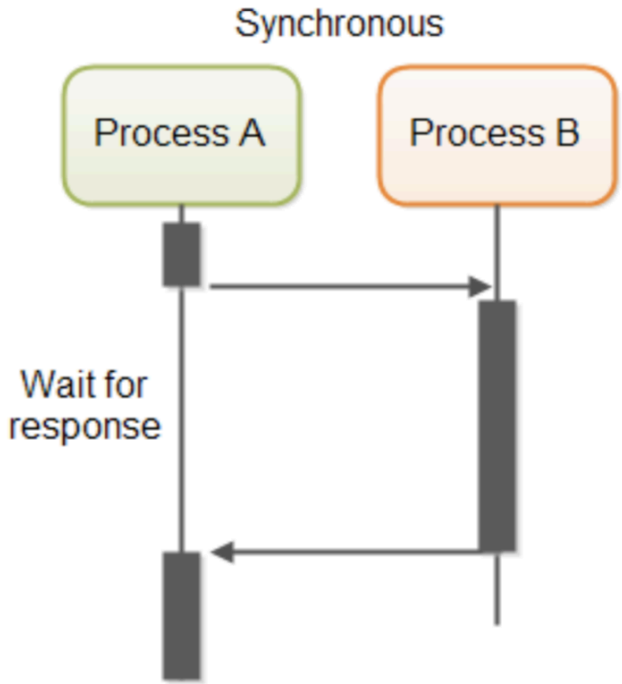
- Create app.js to call the module ‘method’ and show the result of add(1,2) and pi
- For ‘method’ module, there are a method call “add(a,b)” and variable pi = `Math.PI`

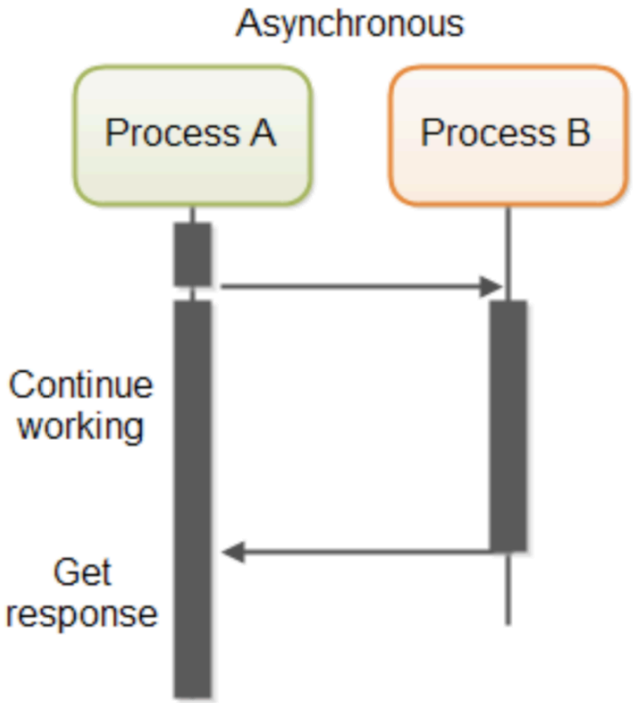
npm Node Package Manager

- npm is the package manager for javascript and more.
- https://www.npmjs.com/

```
npm install moduleName -option
```

Synchronous and Asynchronous





JavaScript Callback

Ex04

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

function readFile(str,callback){
 var lines=[];
 fs.readFile(str,function(err,data) {
  if(err)throw err;
  lines.push(data);
  console.log(data.toString());
 });
 callback(lines);
}
```

```
var a,b;
readFile('./20.txt',function(data){
  a=data;
});

readFile('./10.txt',function(data){
  b=data;
});
util.log("a: "+a+"\n");
util.log("b: "+b+"\n");
```

Core Modules

- Assertion
- Testing
- Buffer
- C/C++ Addons
- Child Processes
- Cluster
- Crypto
- Debugger
- DNS
- Domain
- Events
- File System
- HTTP/HTTPS
- Net
- OS
- Path
- Process
- Query
- Strings
- REPL
- Stream

Core Modules

- String
- Decoder
- Timers
- TLS/SSL
  TTY
- UDP/Datagram
- URL
- Utilities
- VM
- ZLIB

Assertion Testing

Ex05

- This module is used for writing unit tests for your applications, you can access it with require('assert ').

```
var assert = require('assert');
function add(a, b) {
  return a+b;
}
var expected = add(1,2);
assert( expected=== 4,'one plus two is three');
```

Buffer

Ex06

- Pure JavaScript is Unicode friendly but not nice to binary data.

```
buf = new Buffer(10);
buf.write("abcdefghj","ascii");
console.log(buf.toString('base64'));
buf = buf.slice(0,5);
console.log(buf.toString('utf8'));
```

Buffer

Ex06 -- More examples

```
var buf = newBuffer(10)
console.log(buf)

var buf = new Buffer([10,20,30])
console.log(buf)

var buf = new Buffer("Hello World")
console.log(buf.toString())

var buf = new Buffer(200)
len = buf.write("Hello World")
console.log(len)

var buf = new Buffer(200)
len = buf.write("Hello World")
console.log(buf.toString())
```

```
var buf = new Buffer(26)
for (let index = 0; index<26;index++) {
  buf[index] = index+97
}
console.log(buf)

var buf = new Buffer(26)
for (let index = 0; index<26;index++) {
  buf[index] = index+97
}
//console.log(buf.toString('utf8'))
console.log(buf.toString('ascii'))
```

Buffer
Json

Ex06 -- More examples

```
var buf = new Buffer('Hello World')
var json = buf.toJSON(buf)
console.log(json)

//-------------------------------------------------//

var buf1 = new Buffer('Hello World')
var buf2 = new Buffer('Good Afternoon')
var buf = Buffer.concat([buf1,buf2])

console.log(buf.toString())
console.log(buf1.toString() + buf2.toString())
```

Crypto

Ex07

- V8 comes with an extensive debugger which is accessible out-of-process via a simple TCP protocol.

```
var crypto=require('crypto');
var fs=require('fs');
var shasum = crypto.createHash('sha1');
var s = fs.ReadStream('file.txt');
s.on('data',function(d) {
  shasum.update(d);
});
s.on('end',function() {
  var d = shasum.digest('hex');
  console.log(d+' => file.txt');
});
```

Debugger

Ex08

- The crypto module offers a way of encapsulating secure credentials to be used as part of a secure HTTPS net or http connection

```
//node debug debugger.js
for(var i=0;i<10;i++){
  console.log(i);
}
```

DNS

Ex09

- This module contains functions that belong to two different categories:
  - Functions that use the underlying operating system facilities to perform name resolution.

```
var dns = require('dns');
dns.lookup('www.google.com', function
onLookup(err,addresses, family) {
  console.log('addresses:', addresses);
});
```

File System

Ex10

- File I/O is provided by simple wrappers around standard POSIX functions. To use this module do require('fs'). All the methods have asynchronous and synchronous forms.

```
var fs=require('fs');
fs.writeFile('message.txt','Hello Node',
function(err) {
 if(err) throw err;
console.log('It\'s saved!');});
```

File System

Ex10 -- More examples

- Read file
- Make new directory

```
var fs=require('fs')
var readMe = fs.readFileSync('./code.txt','utf8')
console.log(readMe)


fs.mkdir('Project', function(){
  fs.writeFileSync('./Project/Output.txt', readMe)
})
```

os

Ex13

- Few basic operating-system related utility functions

```
var os=require('os');
console.log(os.hostname());
console.log(os.type());
console.log(os.platform());
console.log(os.arch());
console.log(os.release());
console.log(os.uptime());
console.log(os.loadavg());
```

Readline

Ex14

- To use this module, do require('readline'). Readline allows reading of a stream (such as process.stdin) on a line-by-line basis.

```
var readline = require('readline');
var rl = readline.createInterface({
        input:process.stdin,
        output:process.stdout
});
rl.question("What do you think of node.js? (easy/difficult)
",function(answer) {
// TODO: Log the answer in a database
console.log("Thank you for your valuable feedback:", answer);
rl.close();});
```

Stream

Ex15

- A stream is an abstract interface implemented by various objects in Node.

```
var fs = require('fs');
var zlib = require('zlib');
var r = fs.createReadStream('file.txt');
var z = zlib.createGzip();
var w = fs.createWriteStream('file.txt.gz');
r.pipe(z).pipe(w);
```

Stream

Ex15 – more examples -- readStream

```
var fs=require('fs')
var data=''
var readStream=fs.createReadStream('code.txt')
readStream.setEncoding('utf8')
readStream.on('data',function(txt){
  data+=txt
})
readStream.on('end',function(){
  console.log(data)
})

//-------------------------------------------------//
readStream.on('error',function(err){
  console.log(err.stack)
})
```

Stream

Ex15 – more examples -- writeStream

```
var fs=require('fs')
var data='Hello World Modi quia maiores. Nihil sapiente et atque. Sint numquam molestiae iure nemo fugiat asperiores aspernatur ducimus. Sit rerum officiis reprehenderit hic nisi odit officia accusantium eos.'

var writerStream=fs.createWriteStream('output.txt')
writerStream.write(data,'utf8')
writerStream.end()
writerStream.on('finish',function(){
console.log("Output finish")
})
```

URL

Ex16

```
var url = require('url');
var result =
url.parse('http://user:pass@host.com:8080/p/a/t/h?
query=string#hash');
console.log(result);
```

Serving HTML

Ex17-1 : HTML

```
var http=require('http')
var fs=require('fs')
http.createServer(function(rew,res){
  res.writeHead(200,{'Content-Type':'text/html'})
  var myStream =
fs.createReadStream(__dirname+'/'+'index.html','utf8')
myStream.pipe(res)
}).listen(8081,'127.0.0.1')
```

Serving JSON Data

Ex17- 2 : JSON Data

```
var http=require('http')
var fs=require('fs')
var myUser={
  "name":"Reese",
  "job":"Coordinator",
  "age":"30"
}

http.createServer(function(rew,res){
  res.writeHead(200,{'Content-Type':'application/json'})
  res.end(JSON.stringify(myUser))
}).listen(8081,'127.0.0.1')
```

Routing

Ex18: Routing

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

http.createServer(function(req,res){
  if(req.url==='/home'||req.url==='/'){
    res.writeHead(200,{'Content-Type':'text/html'})
    var myStream = fs.createReadStream(__dirname+'/'+'index.html','utf8')
    myStream.pipe(res)

  }
  else if(req.url==='/facebook'||req.url==='/'){
    res.writeHead(200,{'Content-Type':'text/html'})
    var myStream = fs.createReadStream(__dirname+'/'+'facebook.html','utf8')
    myStream.pipe(res)
  }
```

Routing

Ex18: Routing (cont.)

```
  else if(req.url==='/youtube'||req.url==='/'){
    res.writeHead(200,{'Content-Type':'text/html'})
    var myStream = fs.createReadStream(__dirname+'/'+'youtube.html','utf8')
    myStream.pipe(res)
  }
  else{
    res.writeHead(404,{'Content-Type':'text/html'})
    var myStream = fs.createReadStream(__dirname+'/'+'notfound.html','utf8')
    myStream.pipe(res)
  }
}).listen(8081,'127.0.0.1')
```

Express: Hello Wolrd

Ex19-1: Express

```
//npm install express --save
var express=require('express')
var routing=express()
routing.get('/',function(req,res){
  res.send("<h1>Hello World</h1>")
})

routing.listen(8081)
```

Express: Route Params

Ex19-2: Express – Route Params

```
//npm install express --save
var express=require('express')
var routing=express()
routing.get('/profile/:name',function(req,res){
  res.send("<h1>Welcome "+req.params.name+"</h1>")
})

routing.listen(8081)
```

Express:
Practice

Ex19: Express -- Practice

- Read users.json file
- Show the information of one user by specifying the id in the url
  - http://localhost:8081/profile/1

```
id: 2
username: John
password: f56d6351aa71cff0debea014d13525e42036187a
fullname: John Doe
```

- Password must be encrypted

Express: Middleware

Ex20: Express -- Middleware

```
var express=require('express')
var app=express()
app.get('/',function(req,res){
  res.send("<h1>Hello User</h1>")
})

app.use('/user/profile/',function(req,res,next){
  console.log("Request: "+newDate(),req.method,req.url)
})

app.listen(8081)
```