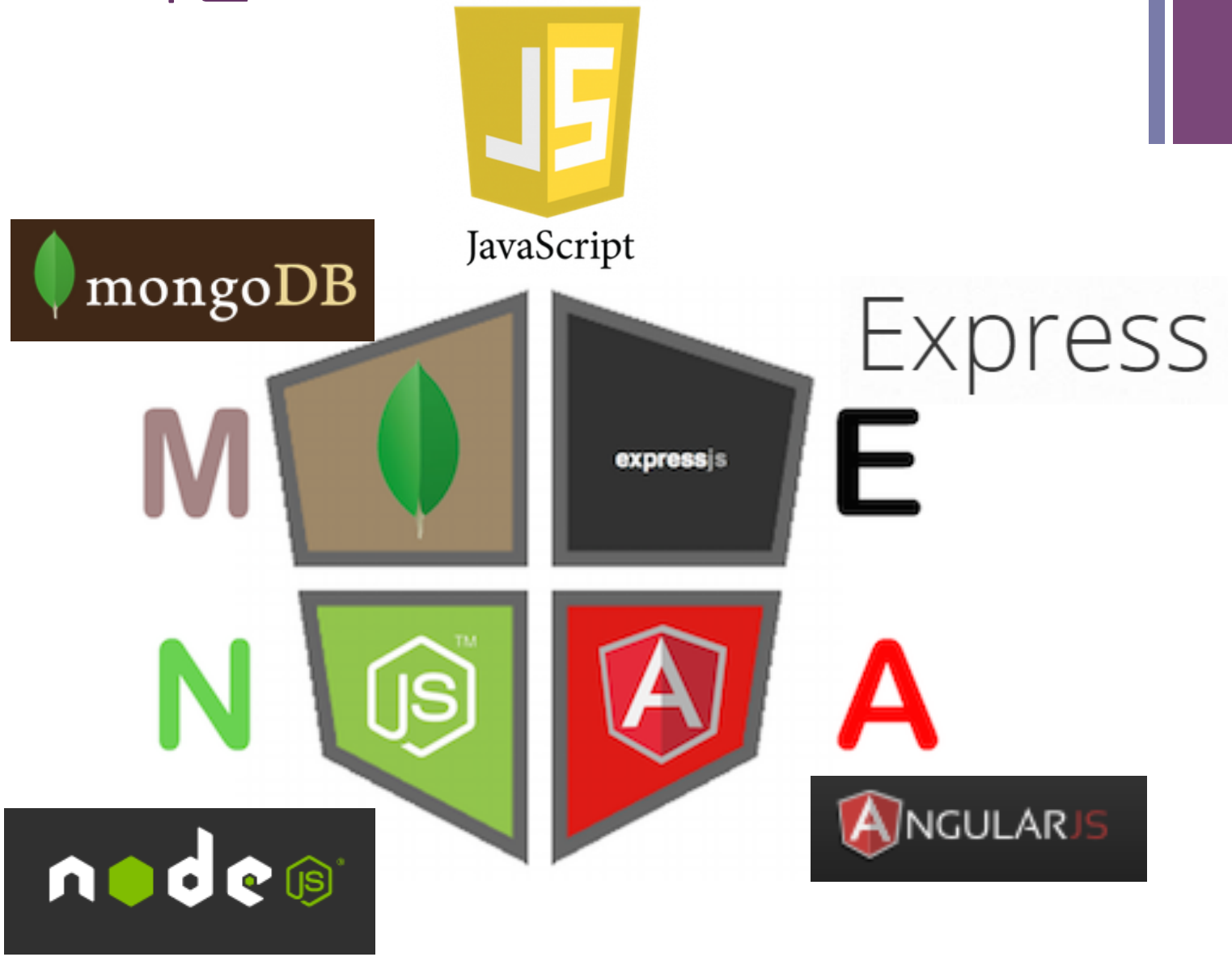
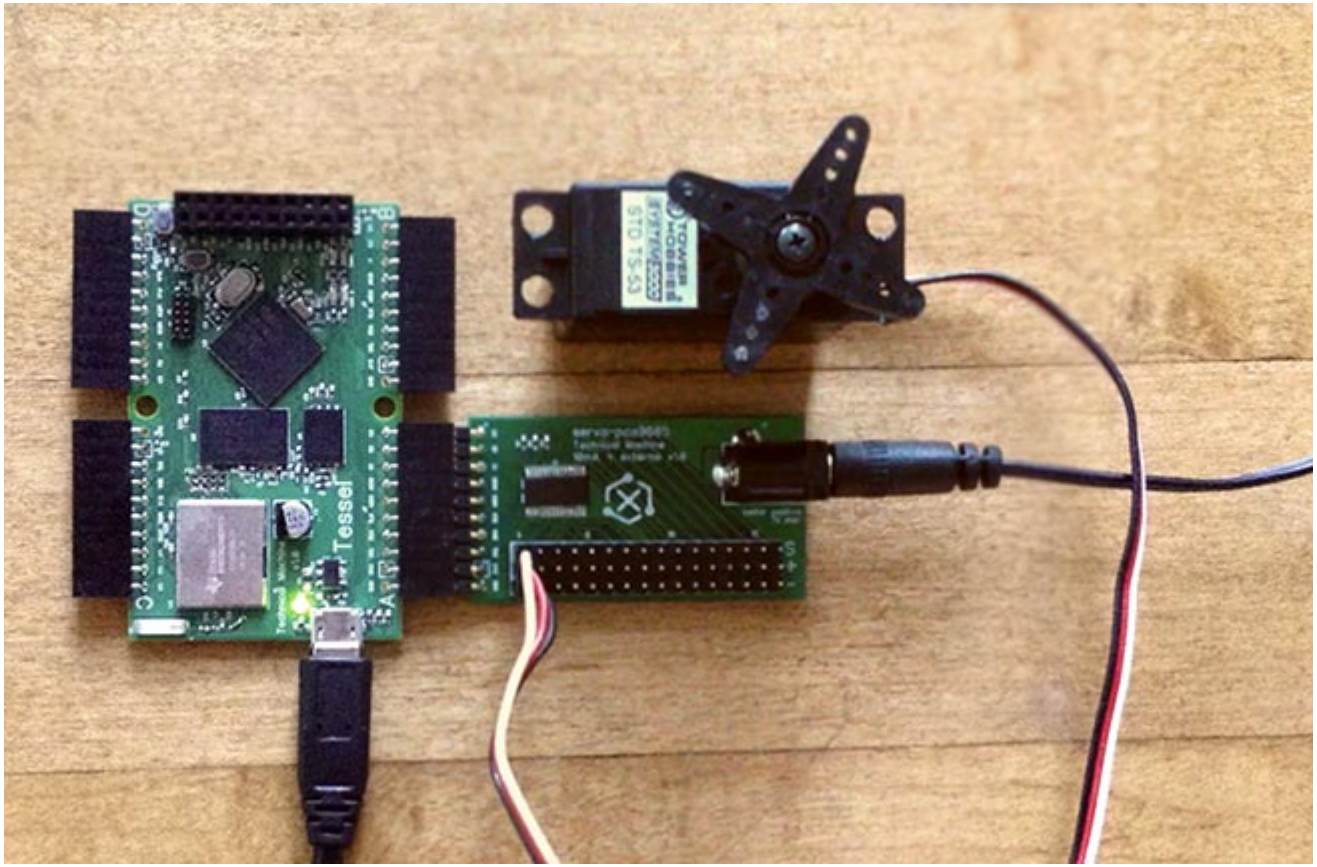
MEAN 스택을 사용한 IoT 개발

박재호(jrogue@gmail.com)

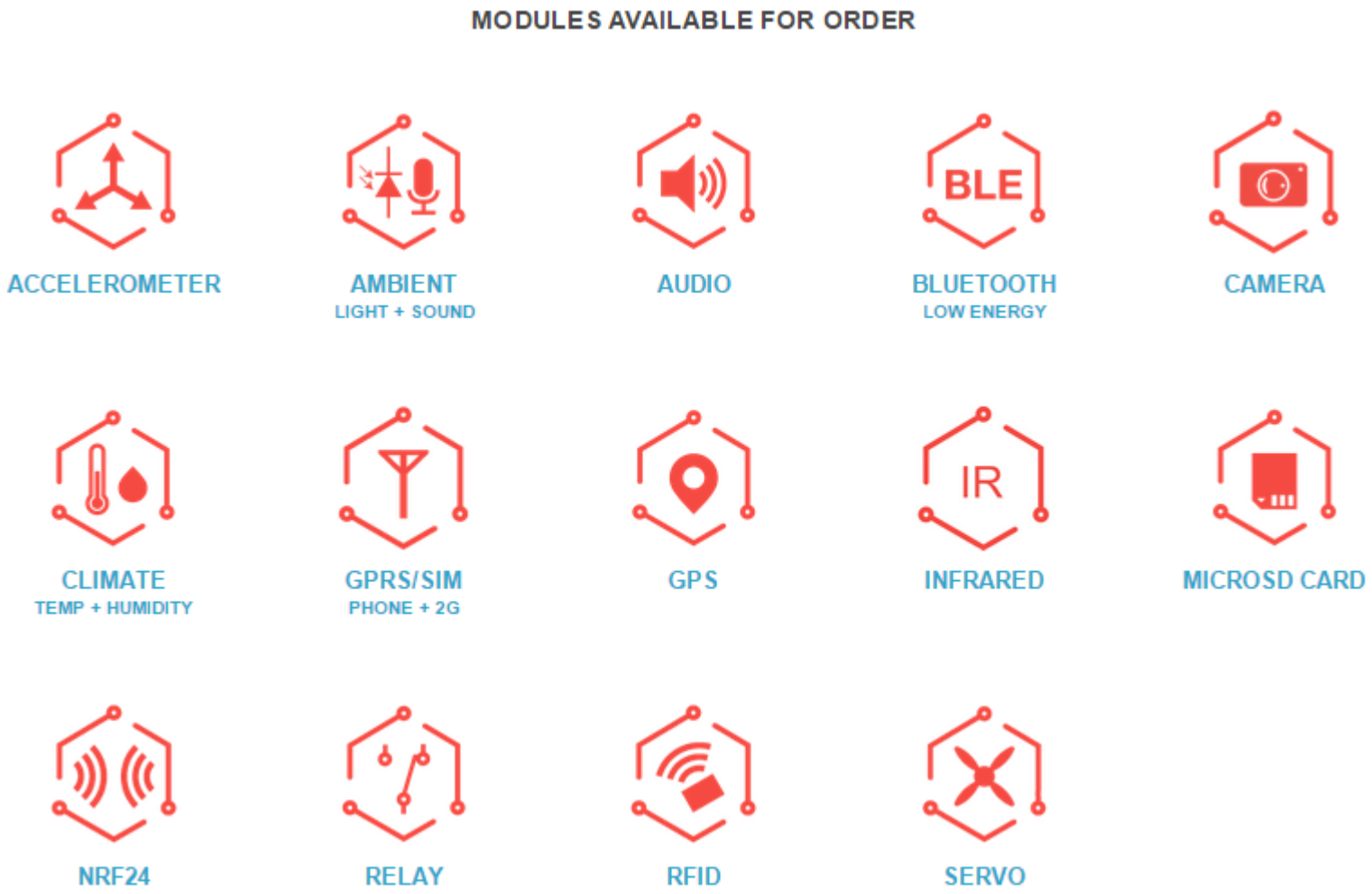
MEAN이란?


JavaScript
mongoDB
Express
M
E
N
A
expressjs
node
ANGULARJS

TESSEL



TESSEL 모듈 유형


MODULES AVAILABLE FOR ORDER
ACCELEROMETER
AMBIENT
LIGHT + SOUND
AUDIO
BLE
BLUETOOTH
LOW ENERGY
CAMERA
CLIMATE
TEMP + HUMIDITY
GPRS/SIM
PHONE + 2G
GPS
IR
INFRARED
MICROSD CARD
NRF24
RELAY
RFID
SERVO

TESSEL 스펙

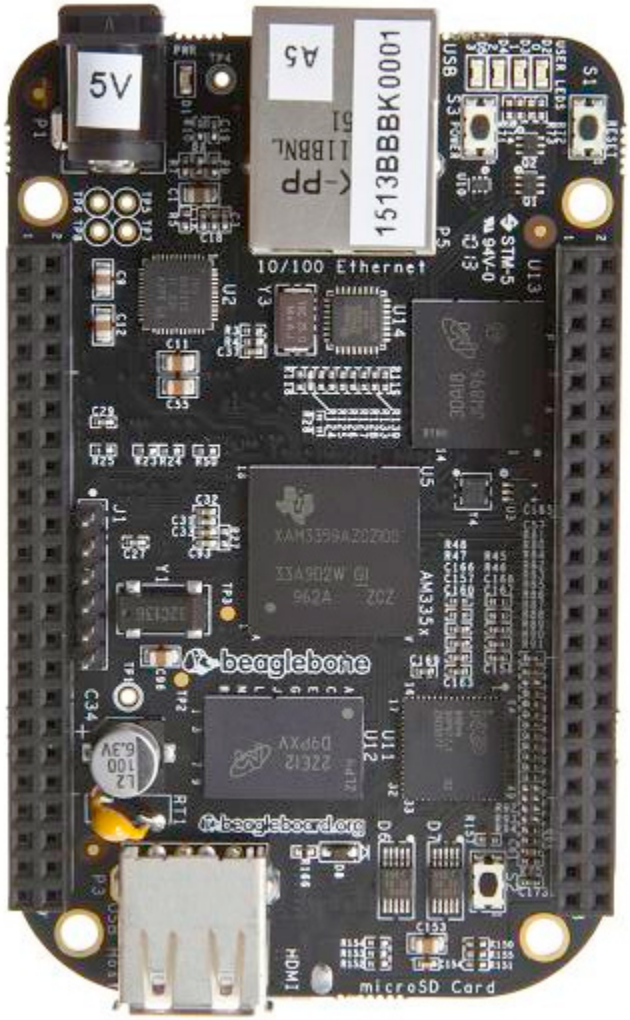
THE NITTY

- Programmable via JavaScript
- 14 different hardware modules for added capabilities
- Compatible with 10,000's of Node.js packages on NPM
- Deploy over USB or remotely by WiFi

THE GRITTY

- 180mhz ARM Cortex-M3 LPC1830
- 32mb SDRAM
- 32mb Flash
- TI CC3000 WiFi radio
- 20-pin GPIO bank for general prototyping
- Micro USB or battery power

Beagle Bone Black



Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

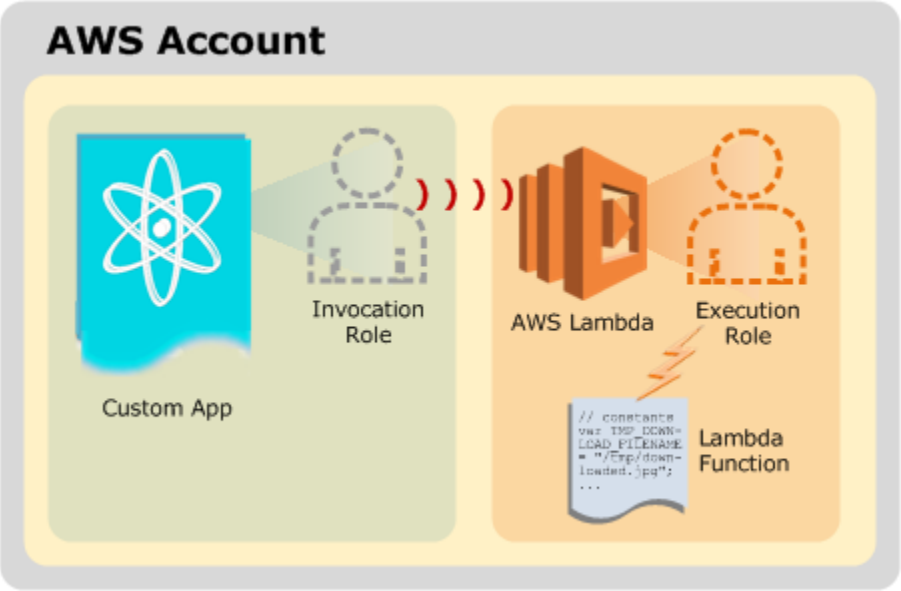
bonescript

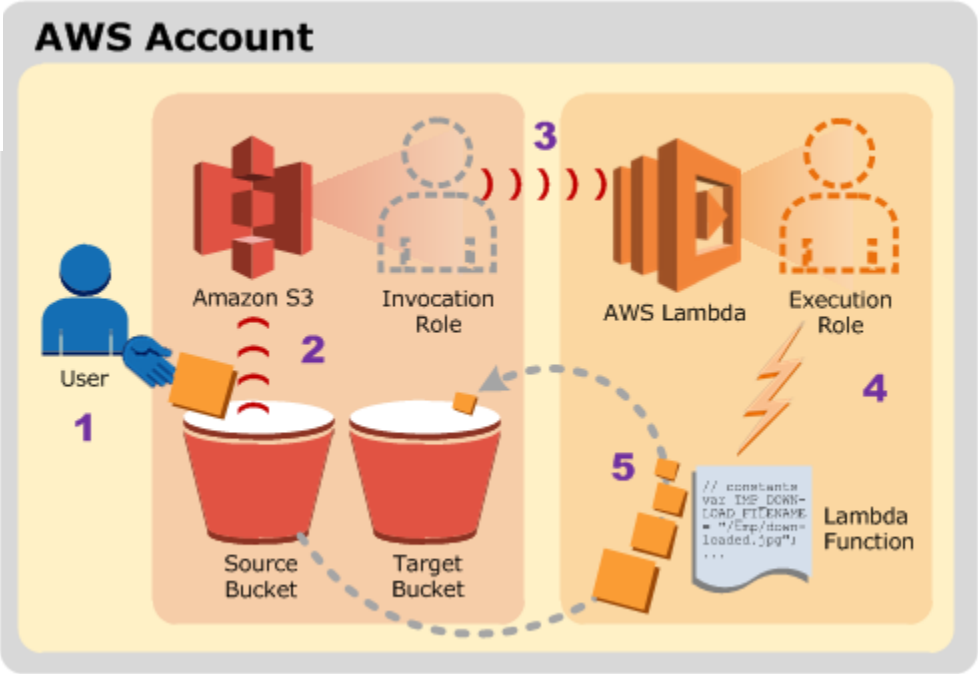
- Beagle Bone에서 아두이노 스타일의 프로그램이 가능하게 만들어 주는 자바스크립트 라이브러리

```
var b = require('bonescript');

b.pinMode('P8_12', b.INPUT);
b.pinMode('P8_13', b.OUTPUT);

setInterval(copyInputToOutput, 100);

function copyInputToOutput() {
    b.digitalRead('P8_12', writeToOutput);
    function writeToOutput(x) {
        b.digitalWrite('P8_13', x.value);
    }
}
```

AWS Lambda(1)

AWS Lambda(2)


AWS Account
Custom App
Invocation Role
AWS Lambda
Execution Role
Lambda Function


AWS Account
User
1
Amazon S3
2
Invocation Role
3
AWS Lambda
Execution Role
4
5
Source Bucket
Target Bucket
Lambda Function

AWS Lambda(3)

1

```
console.log('Loading event');
exports.handler = function(event, context) {
  console.log("value1 = " + event.key1);
  console.log("value2 = " + event.key2);
  console.log("value3 = " + event.key3);
  context.done(null, "Hello World");  // SUCCESS with message
}
```

2

```
$ aws lambda upload-function \
   --region us-east-1 \
   --function-name helloworld \
   --function-zip file-path/helloworld.zip \
   --role IAM-role-ARN \
   --mode event \
   --handler helloworld.handler \
   --runtime nodejs \
   --debug
```

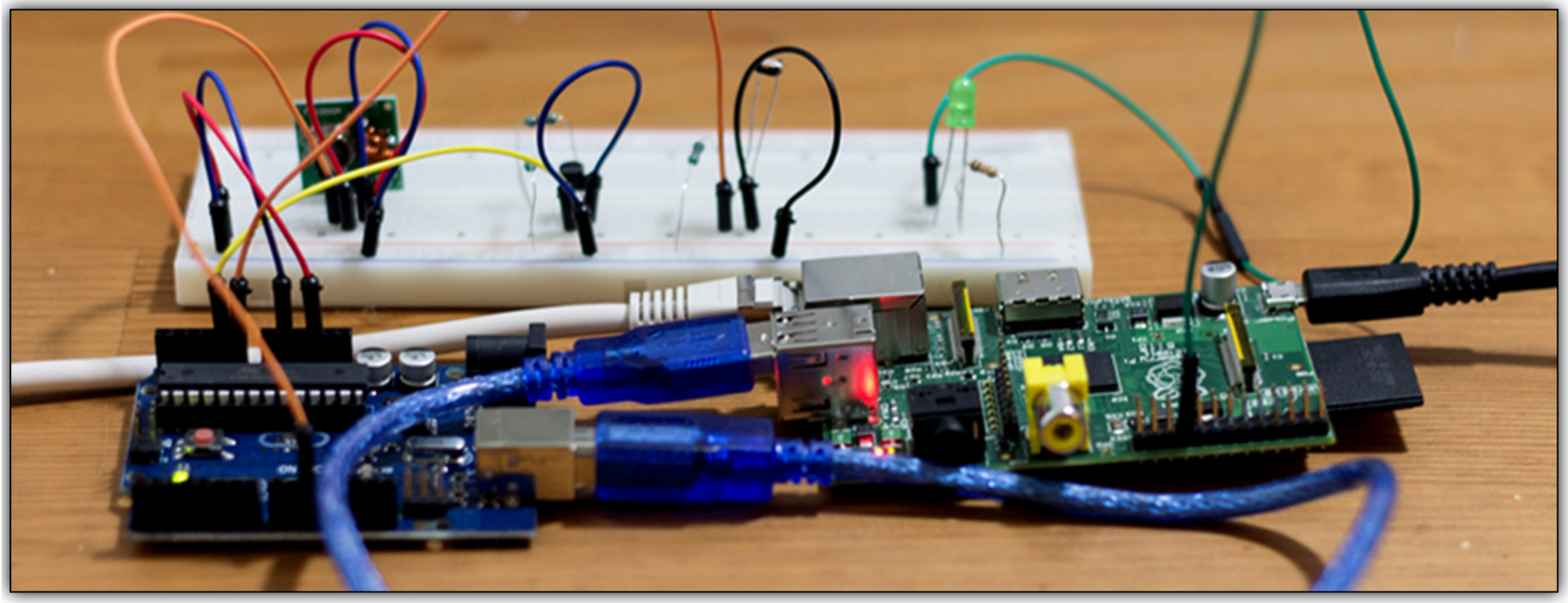
3

```
{
   "key1":"value1",
   "key2":"value2",
   "key3":"value3"
}
```

```
$ aws lambda  invoke-async \
 --function-name helloworld \
 --region us-east-1 \
 --invoke-args inputfile.txt \
 --debug
```

heimcontrol.js(1)

Raspberry PI and Arduino using Node.js

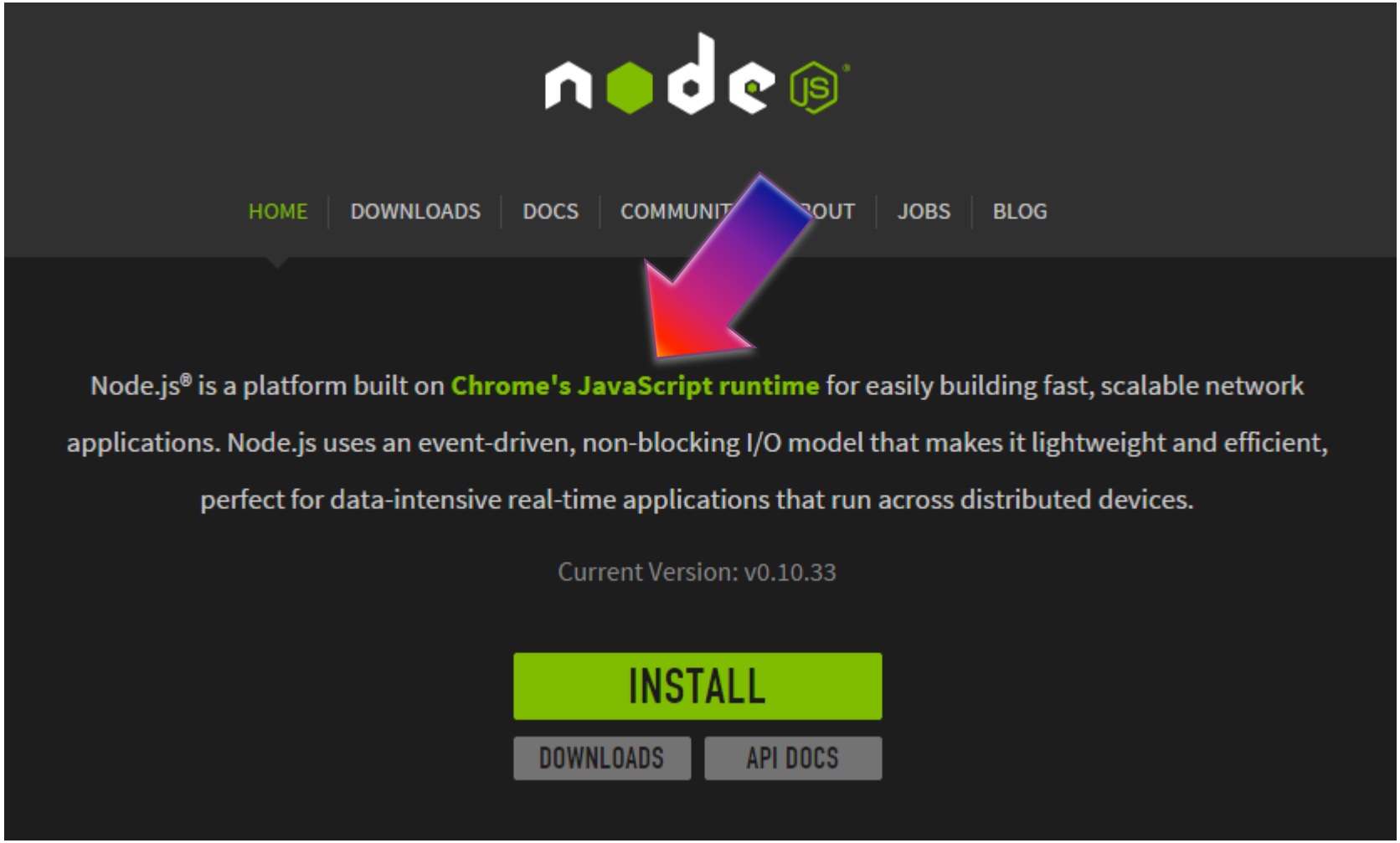


heimcontrol.js(2)

HTML5/CSS3
Jade(for templating)

Socket.io(for Websockets)
Requirejs

Node.js인 이유는?(1)



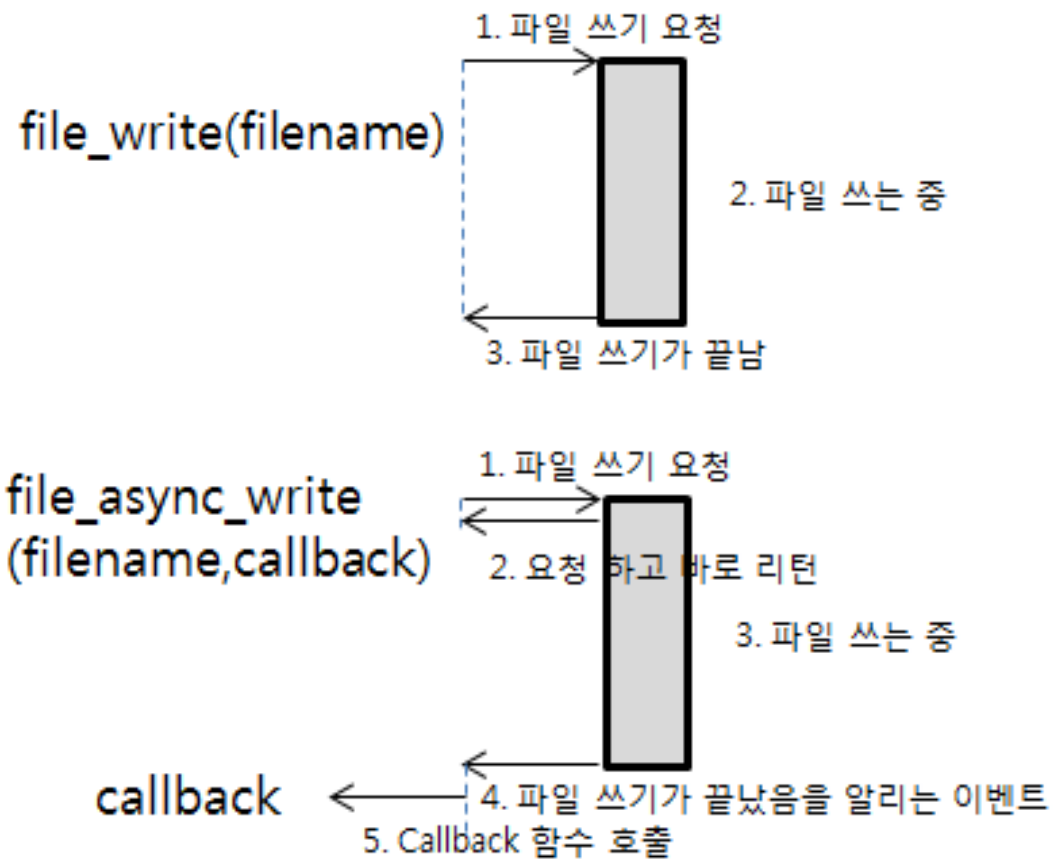
Node.js인 이유는?(2)

- 개발 시간과 성능의 조화
  - 아무리 성능이 좋더라도 개발 시간이 오래 걸릴 경우 생산성이 떨어진다
    - 주의: 1행을 작성하는 시간은 어셈블리나 SQL이나 비슷하다
  - 스크립트 언어의 경우 빨리 개발은 가능하지만 성능이 떨어진다
  - 자바스크립트를 사용한 Node.js는 개발과 성능 양쪽 모두 뛰어나다
- 스타트업 vs 기업
  - 스타트업은 빨리 변화하는 생태계에서 살아남기 위해 필연적으로 오픈 소스에 매진한다
  - 대기업은 유지보수와 안정성, 그리고 성능에 관심이 많다
  - Node.js는 양쪽 모두를 만족시킨다(오픈 소스 기반, 자바와 같은 강력한 생태계)

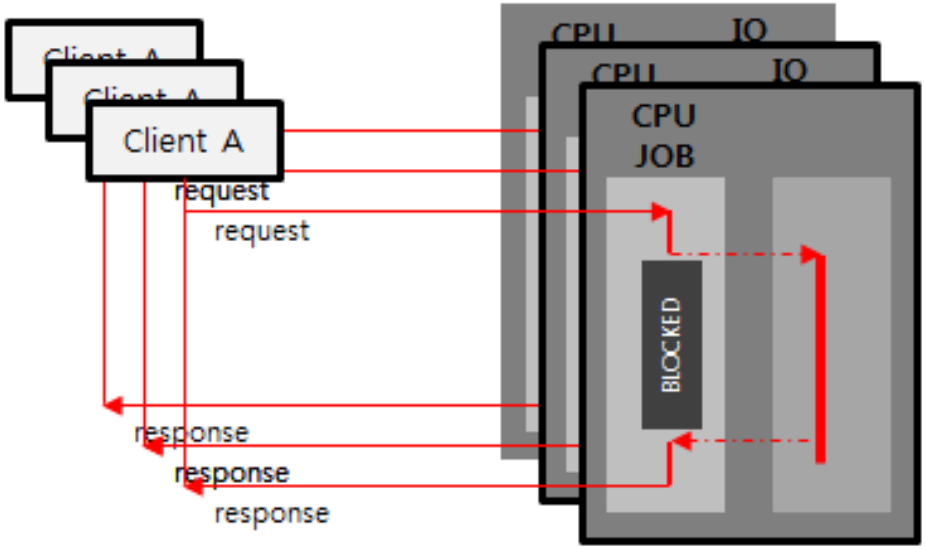
Node.js인 이유는?(3)

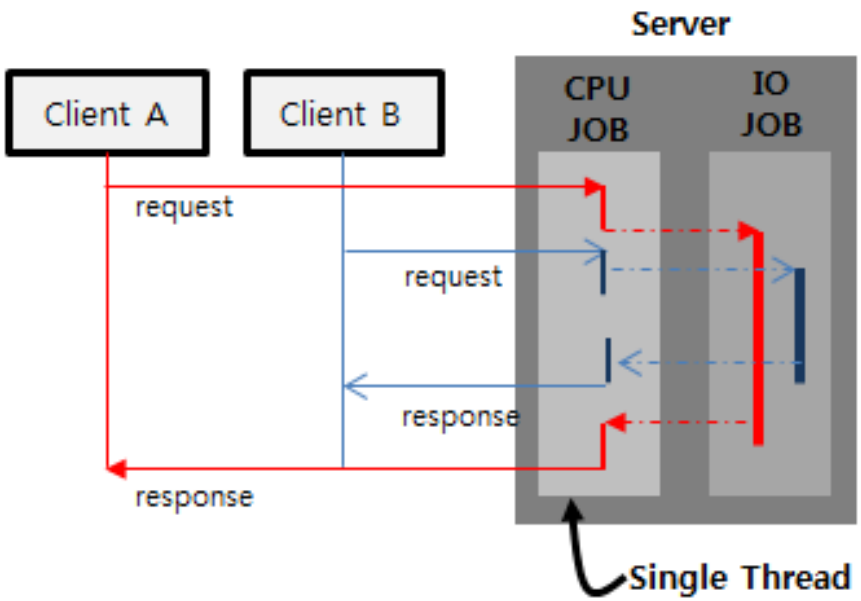
- 뛰어난 성능
  - 월마트는 블랙 프라이데이에 Node.js로 전제 모바일 사이트 오픈
    - 초당 두 배 요청을 받아들이고 응답시간을 35%(200ms) 줄임
    - 서버 CPU 사용량이 2%, 200만 명 온라인 접속 방어
  - 이벤트 아키텍처
    - 스레드가 아닌 단일 프로세스
    - 대신 동기식 차단이 아닌 비동기식 I/O
    - 웹 브라우저 특성상 이미 자바스크립트는 콜백을 사용하는 이벤트 구동 방식을 지원(비동기식 코드에 대한 표현력이 강함)

동기식 입출력 vs 비동기식 입출력

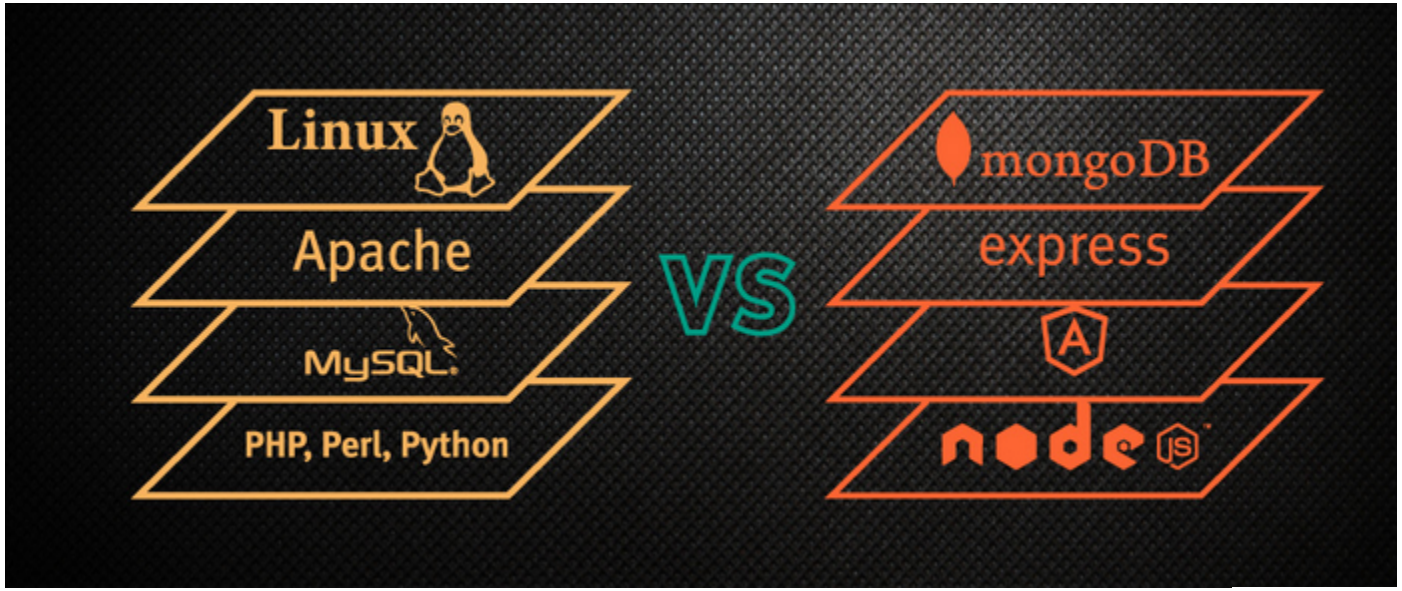


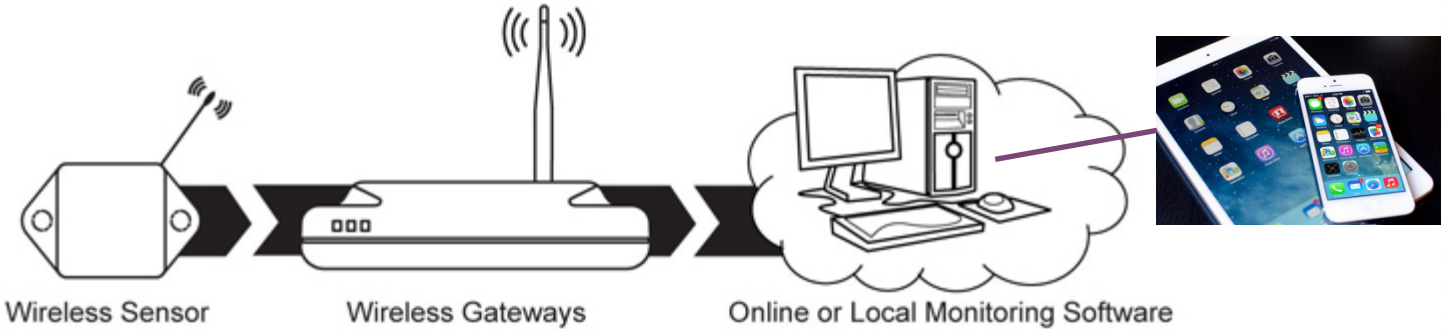
멀티 스레드 vs 단일 스레드





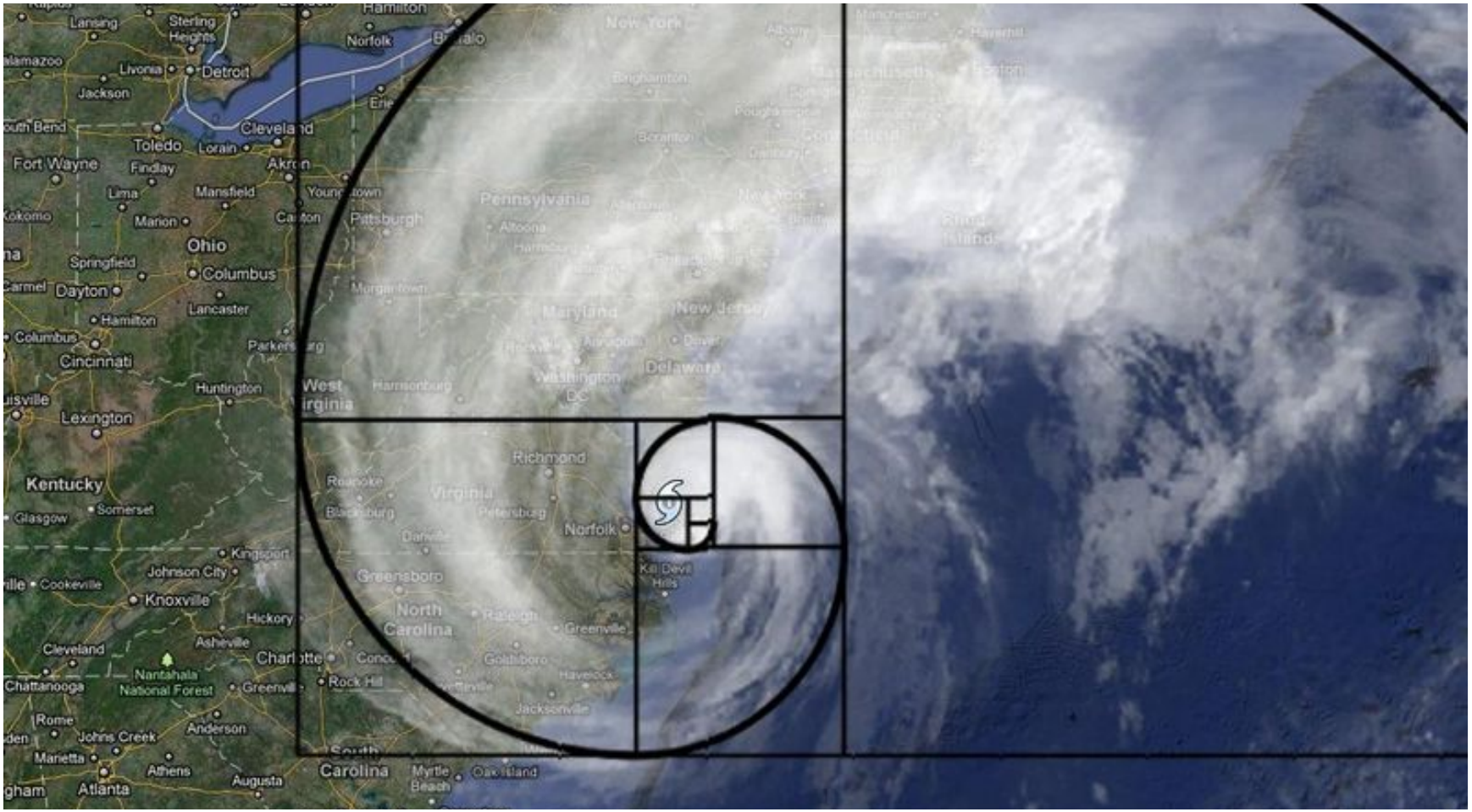
LAMP vs MEAN



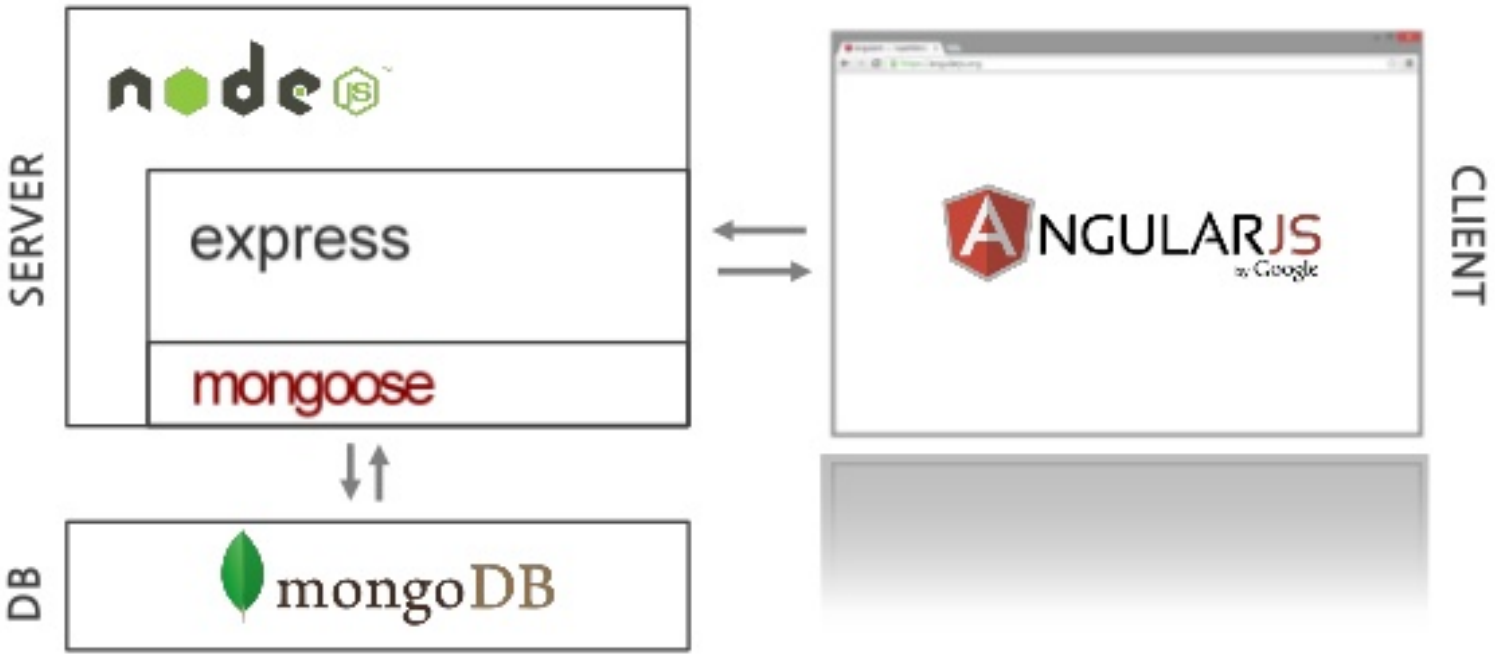


JavaScript

자기 조직화


Detroit
Toledo
Cleveland
Akron
Pittsburgh
Ohio
Columbus
Dayton
Cincinnati
Lexington
Kentucky
Pennsylvania
Maryland
Delaware
Richmond
Virginia
Norfolk
Greensboro
North Carolina
Knoxville
Charlotte
Atlanta
Augusta
Myrtle Beach

MEAN stack 아키텍처



선행조건

- MEAN 기본 스택 준비
  - Node.js 설치: http://nodejs.org/
  - MongoDB 설치: http://www.mongodb.org/
  - 자바스크립트 편집기는 brackets 추천: http://brackets.io/

- 예제 소스 코드 저장소 위치
  - Node.js용 node-collector: https://bitbucket.org/jrogue/node-collector
  - AngularJS용 angular-lamp-app: https://bitbucket.org/jrogue/angular-lamp-app

무엇을 만들어볼까?

lamp 관리 시스템

기본 아키텍처

첫 Node.js 프로그램: / server_plain_vanilla.js

```
/*jslint node: true */
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World₩n');
}).listen(3000, '127.0.0.1');
console.log('Server running at http://127.0.0.1:3000/');
```

Node.js 프로그램을 위한 패키지 설정: /package.json

```
{
  "name": "node-collector",
  "private": true,
  "version": "0.1.0",
  "description": "A starter project for Node.js",
  "license": "MIT",
  "devDependencies": {
    "express": "4.x",
    "body-parser": "1.4.x“,
    "mongoose": "^3.8.20"
  }
}
```

의존성 설치와 목록

```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>npm install

C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>npm list
node-collector@0.1.0 C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector
├─┬ body-parser@1.4.3
│ ├── bytes@1.0.0
│ ├── depd@0.3.0
│ ├── iconv-lite@0.4.3
│ ├── media-typer@0.2.0
│ ├── qs@0.6.6
│ ├── raw-body@1.2.2
│ └─┬ type-is@1.3.1
│   └── mime-types@1.0.0
├─┬ express@4.10.4
│ ├─┬ accepts@1.1.3
│ │ ├─┬ mime-types@2.0.3
│ │ │ └── mime-db@1.2.0
│ │ └── negotiator@0.4.9
│ ├── content-disposition@0.5.0
│ ├── cookie@0.1.2
│ ├── cookie-signature@1.0.5
```

Node.js를 사용한 첫 RESTful 프로그램: /server.js

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

var app = express();

app.get('/lamps', function(req, res) {
   res.send([{name: 'lamp1'}, {name: ‘lamp2'}]);
});
app.get('/lamps/:id', function(req, res) {
   res.send({id:req.params.id, name: "lamp" + req.params.id, status: "lamp
ready"});
});

app.listen(3000);
console.log('Listening on port 3000...');
```

실행과 테스트 방법

명령 프롬프트 - node server

```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>node server
Listening on port 3000...
```

localhost:3000/lamps

[{"name":"lamp1"},{"name":"lamp2"}]

localhost:3000/lamps/1

{"id":"1","name":"lamp1","description":"lamp ready"}

Node.js 모듈 사용: /routes/lamps.js

```
exports.findAll = function (req, res) {
    res.send([{name: 'lamp1'}, {name: 'lamp2'}, {name: 'lamp3'}]);
};

exports.findById = function (req, res) {
    res.send({id: req.params.id, name: "lamp" + req.params.id, status: "lamp
ready"});
};
```

```
var express = require('express'),
    bodyParser = require('body-parser'),
    lamps = require('./routes/lamps.js');

var app = express();

app.get('/lamps', lamps.findAll);
app.get('/lamps/:id', lamps.findById);

app.listen(3000);
console.log('Listening on port 3000...');
```

몽고DB 연동을 위한 준비: /db.js

```
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/lamps', function () {
    console.log('mongodb connected')
});
module.exports = mongoose;
```

몽고DB 모델 생성: /models/lamp.js

```
var db = require('../db');

var Lamp = db.model('Lamp', {
    lampname:     { type: String, required: true },
    status:       { type: String, required: true },
    date:  { type: Date, required: true, default: Date.now }
});
module.exports = Lamp;
```

Node.js 모듈 내용 변경(1): /routes/lamps.js

```
var Lamp = require('../models/lamp');

exports.addLamp = function (req, res, next) {
    var lamp = new Lamp({
        lampname: req.body.lampname,
        status: req.body.status
    });
    lamp.save(function (err, lamp) {
        if (err) { return next(err); }
        res.status(201).json(lamp);
    });
};
```

Node.js 모듈 내용 변경(2): /routes/lamps.js

```
exports.updateLamp = function (req, res, next) {
    Lamp.findOne({ _id: req.params.id }, function (err, lamp) {
        if (err) { return next(err); }
        lamp.status = req.body.status;
        lamp.save(function (err, lamp) {
            if (err) { return next(err); }
            res.status(201).json(lamp);
        });
    });
};

exports.deleteLamp = function (req, res, next) {
    Lamp.remove({ _id: req.params.id }, function (err, lamp) {
        if (err) { return next(err); }
        res.status(201).json(lamp);
    });
};
```

Node.js 모듈 내용 변경(3): /routes/lamps.js

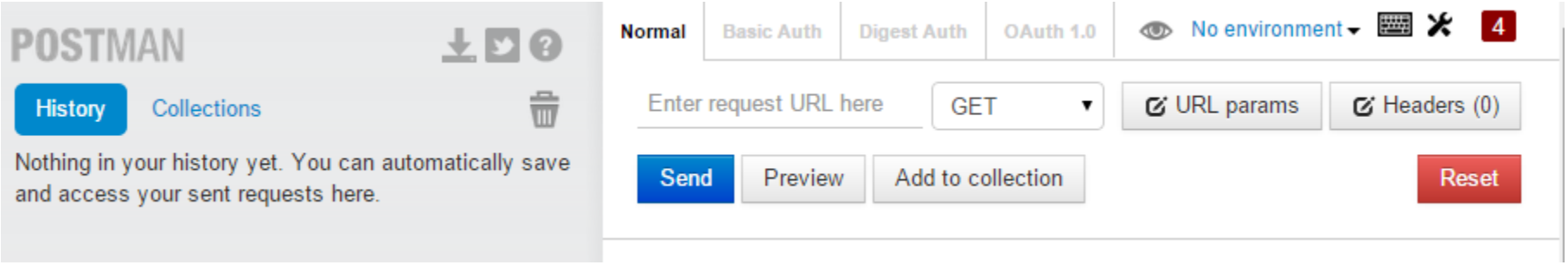
```
exports.findAll = function (req, res, next) {
    Lamp.find(function (err, lamps) {
        if (err) { return next(err); }
        res.status(201).json(lamps);
    });
};

exports.findById = function (req, res, next) {
    Lamp.findOne({ _id: req.params.id }, function (err, lamp)
{
        if (err) { return next(err); }
        res.status(201).json(lamp);
    });
};
```

CRUD 전체 정의를 위한 변경: / server.js

```
var express = require('express'),
    bodyParser = require('body-parser'),
    lamps = require('./routes/lamps.js');

var app = express();

app.use(bodyParser.json());

app.post('/lamps', lamps.addLamp);
app.get('/lamps', lamps.findAll);
app.get('/lamps/:id', lamps.findById);
app.put('/lamps/:id', lamps.updateLamp);
app.delete('/lamps/:id', lamps.deleteLamp);

app.listen(3000);
console.log('Listening on port 3000...');
```

POSTMAN vs curl

크롬 확장: REST 클라이언트



http://curl.haxx.se/ (홈페이지)
http://www.confusedbycode.com/curl/ (윈도우 32비트/64비트)

서버 구동 후 생성 테스트(curl 사용)

```
C:₩Program Files₩MongoDB 2.6 Standard₩bin>curl -v -H "Content-Type: application/
json" -XPOST --data "{₩"lampname₩":₩"lamp_j1₩",₩"status₩":₩"initializing₩"}" loc
alhost:3000/lamps
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> POST /lamps HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
> Content-Type: application/json
> Content-Length: 46
>
* upload completely sent off: 46 out of 46 bytes
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 121
< Date: Sat, 06 Dec 2014 08:02:10 GMT
< Connection: keep-alive
<
{"__v":0,"lampname":"lamp_j1","status":"initializing","_id":"5482b80291fdff40203
dafd5","date":"2014-12-06T08:02:10.141Z"}* Connection #0 to host localhost left
intact
```

몽고DB 셸을 사용한 확인

```
C:₩Program Files₩MongoDB 2.6 Standard₩bin>mongo lamps
MongoDB shell version: 2.6.5
connecting to: lamps
> db.lamps.find();
{ "_id" : ObjectId("5482b80291fdff40203dafd5"), "lampname" : "lamp_j1", "status"
 : "initializing", "date" : ISODate("2014-12-06T08:02:10.141Z"), "__v" : 0 }
>
```

(전체) 읽기 테스트

```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>curl -v -XGET localhost:3000/l
amps
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /lamps HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
>
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 123
< ETag: W/"7b-ed28d7ac"
< Date: Sat, 06 Dec 2014 09:46:10 GMT
< Connection: keep-alive
<
[{"_id":"5482b80291fdff40203dafd5","lampname":"lamp_j1","status":"initializing",
"__v":0,"date":"2014-12-06T08:02:10.141Z"}]* Connection #0 to host localhost lef
t intact
```

변경 테스트

```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>curl -v -H "Content-Type: appl
ication/json" -XPUT --data "{₩"lampname₩":₩"lamp_j1₩",₩"status₩":₩"blinking₩"}"
localhost:3000/lamps/5482b80291fdff40203dafd5
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> PUT /lamps/5482b80291fdff40203dafd5 HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
> Content-Type: application/json
> Content-Length: 42
>
* upload completely sent off: 42 out of 42 bytes
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 117
< Date: Sat, 06 Dec 2014 10:07:20 GMT
< Connection: keep-alive
<
{"_id":"5482b80291fdff40203dafd5","lampname":"lamp_j1","status":"blinking","__v"
:0,"date":"2014-12-06T08:02:10.141Z"}* Connection #0 to host localhost left inta
ct
```

(변경 내역 반영 검증을 위한) 읽기 테스트

```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>curl -v -XGET localhost:3000/l
amps/5482b80291fdff40203dafd5
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /lamps/5482b80291fdff40203dafd5 HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
>
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 117
< ETag: W/"75-79b7b248"
< Date: Sat, 06 Dec 2014 10:07:30 GMT
< Connection: keep-alive
<
{"_id":"5482b80291fdff40203dafd5","lampname":"lamp_j1","status":"blinking","__v"
:0,"date":"2014-12-06T08:02:10.141Z"}* Connection #0 to host localhost left inta
ct
```

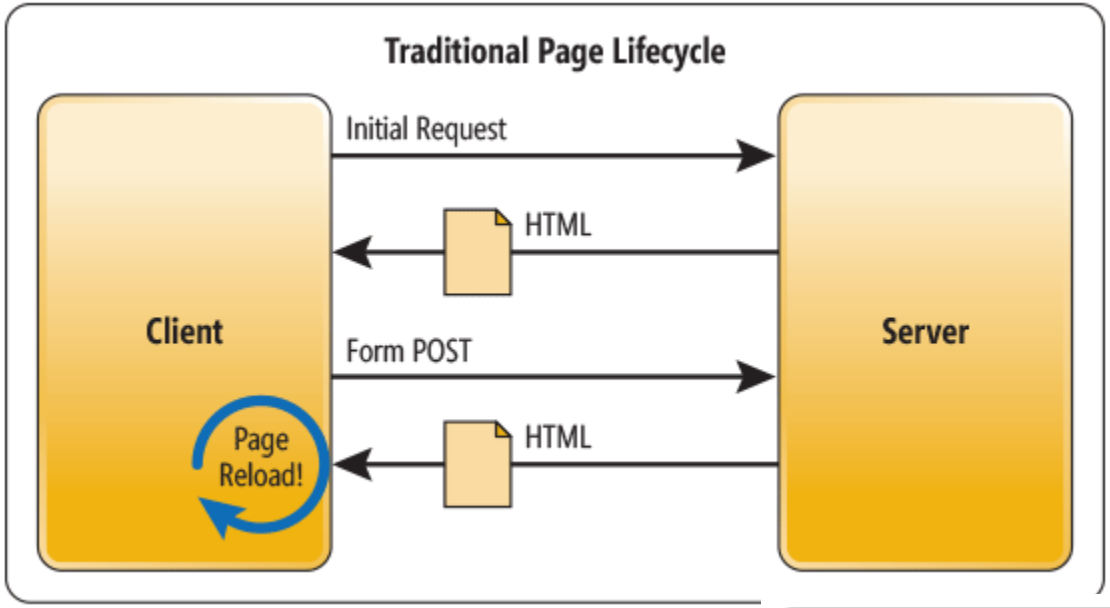
삭제 테스트

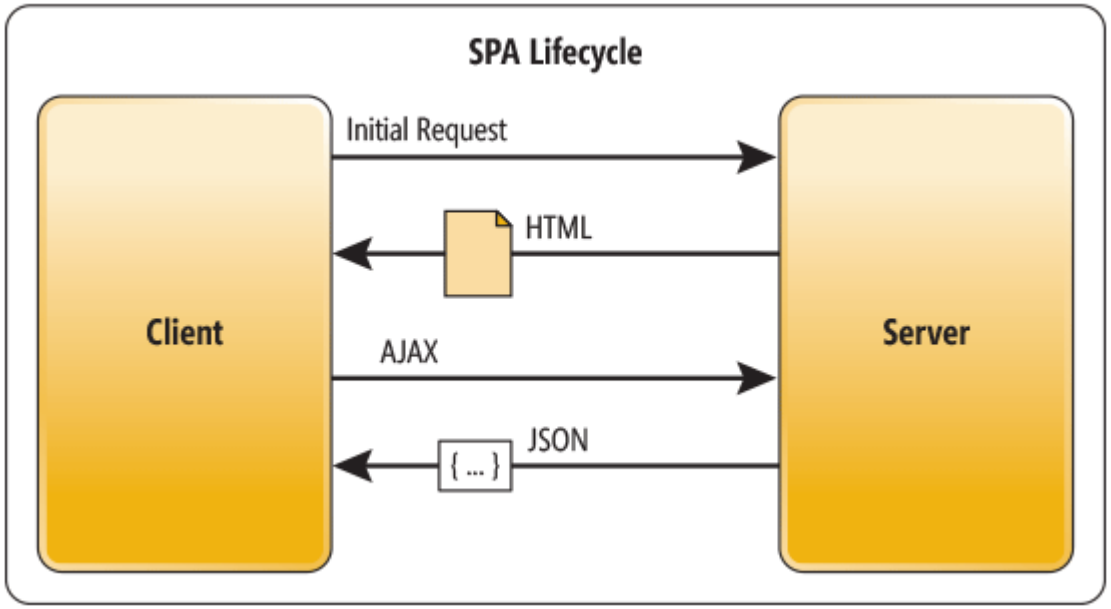
```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>curl -v -XDELETE localhost:300
0/lamps/5482b80291fdff40203dafd5
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> DELETE /lamps/5482b80291fdff40203dafd5 HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
>
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 1
< Date: Sat, 06 Dec 2014 10:12:11 GMT
< Connection: keep-alive
<
1* Connection #0 to host localhost left intact
```

최종 확인

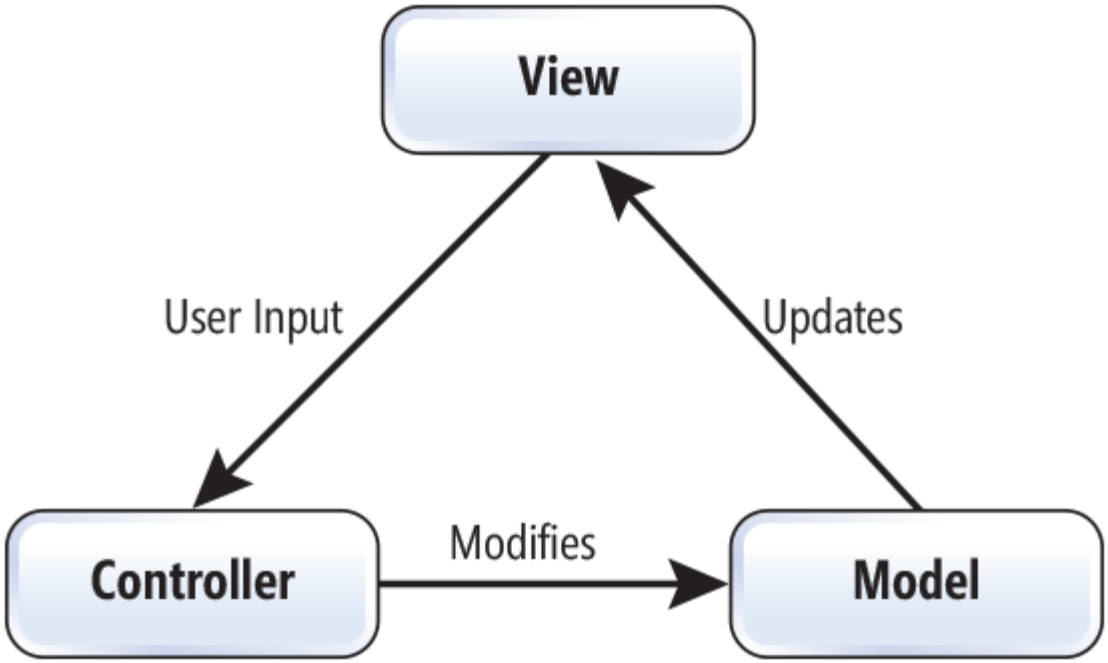
```
C:₩Users₩jrogue₩Documents₩2014KELP₩node-collector>curl -v -XGET localhost:3000/l
amps
* Hostname was NOT found in DNS cache
*   Trying ::1...
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /lamps HTTP/1.1
> User-Agent: curl/7.39.0
> Host: localhost:3000
> Accept: */*
>
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 2
< ETag: W/"2-d4cbb29"
< Date: Sat, 06 Dec 2014 10:13:32 GMT
< Connection: keep-alive
<
[]* Connection #0 to host localhost left intact
```

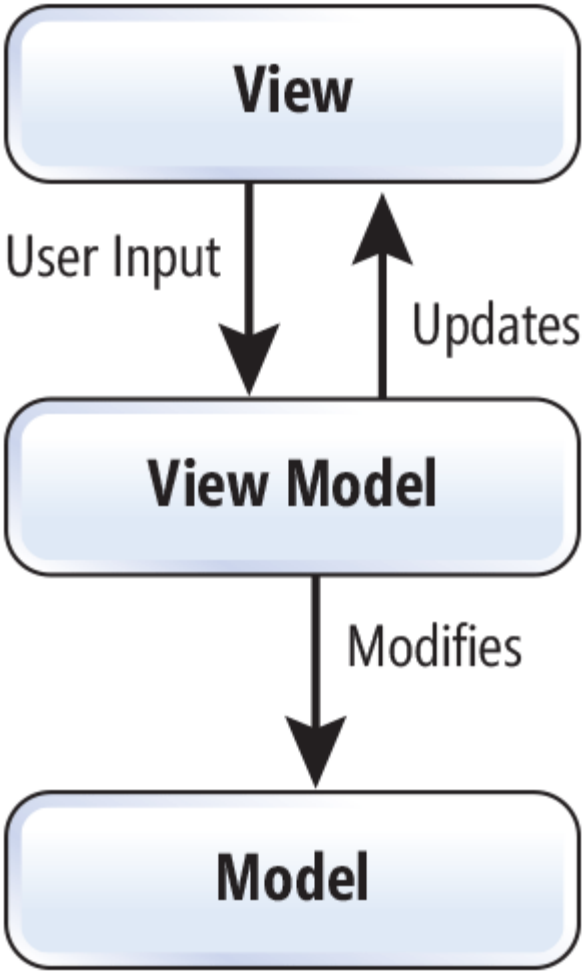
Single Page Application



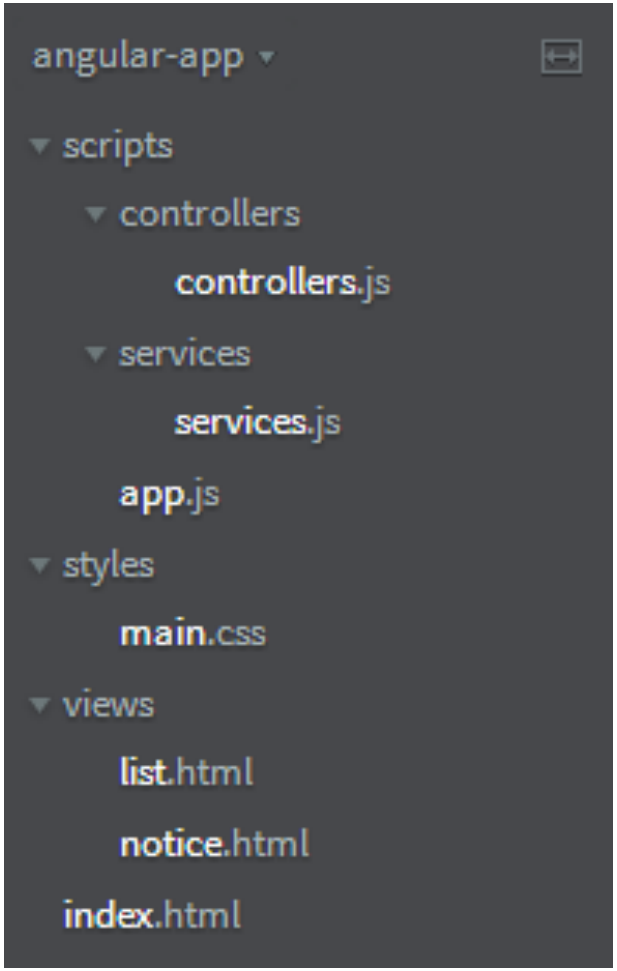


MVC vs MVVM


View
User Input
Updates
Controller
Modifies
Model


View
User Input
Updates
View Model
Modifies
Model

AngularJS 샘플 앱의 디렉터리 구조



/index.html

```
<!DOCTYPE html>
<html ng-app="LampApp">
<head>
  <link href="styles/main.css" rel="stylesheet">
  <link rel="stylesheet" ref="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">
</head>
<body>
  <ul class="menu">
    <li><a href="#/notice">notice</a></li>
    <li><a href="#/list">list</a></li>
  </ul>

  <div class="container" ng-view="">></div>

  <script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.5/angular.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.5/angular-resource.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.5/angular-route.min.js"></script>
  <script src="scripts/app.js"></script>
  <script src="scripts/services/services.js"></script>
  <script src="scripts/controllers/controllers.js"></script>
</body>
</html>
```

views/list.html

```
<div class="span6">
  <table class="table table-striped table-condensed">
    <thead>
      <tr>
        <th style="min-width: 300px;">Lamp ID</th>
        <th style="min-width: 120px;">Name</th>
        <th style="min-width: 120px;">Status</th>
        <th style="min-width: 120px;">Action</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="lamp in lamps">
        <td>{{ lamp._id }}</td>
        <td>{{ lamp.lampname }}</td>
        <td>{{ lamp.status }}</td>
        <td><input type="radio" value="on"> on <input type="radio" value="off"> off</td>
      </tr>
    </tbody>
  </table>
</div>
```

views/notice.html

```
<div>
    <p>환영합니다.</p>
    <p>{{ message }}</p>
</div>
```

scripts/services/services.js

```
'use strict';

var services = angular.module('lampApp.services', ['ngResource']);

var baseUrl = 'http://localhost₩₩:3000';

services.factory('NoticeFactory', function ($resource) {
    return $resource(baseUrl + '/notice', {}, {
        query: { method: 'GET', params: {} }
    });
});

services.factory('LampsFactory', function ($resource) {
    return $resource(baseUrl + '/lamps', {}, {
        query: { method: 'GET', isArray: true }
    });
});

services.factory('LampFactory', function ($resource) {
    return $resource(baseUrl + '/lamps/:id', {}, {
        show: { method: 'GET' }
    });
});
```

scripts/controller/controller.js

```
'use strict';

var app = angular.module('lampApp.controllers', []);

// Clear browser cache (in development mode)
//
// http://stackoverflow.com/questions/14718826/angularjs-disable-partial-caching-on-dev-machine
app.run(function ($rootScope, $templateCache) {
    $rootScope.$on('$viewContentLoaded', function () {
        $templateCache.removeAll();
    });
});

app.controller('NoticeCtrl', ['$scope', 'NoticeFactory', function ($scope, NoticeFactory) {
    $scope.message = '간단한 AngularJS 테스트 프로그램입니다. Lamp 상태를 보여줍니다. 아직 Lamp 조작 기능은 구현되어 있지 않습니다.';
}]);

app.controller('ListCtrl', ['$scope', 'LampsFactory', function ($scope, LampsFactory) {
    $scope.lamps = LampsFactory.query();
}]);
```

scripts/app.js

```
"use strict";

// Declare app level module which depends on views, and components
angular.module('LampApp', [
    'ngResource',
    'ngRoute',
    'lampApp.services',
    'lampApp.controllers'
]).
config(['$routeProvider', function ($routeProvider, $httpProvider) {
    $routeProvider.when('/notice', { templateUrl: 'views/notice.html', controller: 'NoticeCtrl'});
    $routeProvider.when('/list', { templateUrl: 'views/list.html', controller: 'ListCtrl'});
    $routeProvider.otherwise({redirectTo: '/notice'});
}]);
```

HTTP 서버 사용하기

- npm install http-server -g
- cd angular-app
- http-server

또는 Brackets에서

앱 결과 화면(index.html)

[notice | list]

환영합니다.

간단한 AngularJS 테스트 프로그램입니다. Lamp 상태를 보여줍니다. 아직 Lamp 조작 기능은 구현되어 있지 않습니다.

앱 결과 화면(index.html에서 list)

[notice | list]

Lamp ID	Name	Status	Action
5483db1c0ea3770c2077a652	lamp_j1	ready	on off
5483db2e0ea3770c2077a653	lamp_j2	initializing	on off
5483def6274b3a94299d5eae	lamp_j3	offline	on off

도전 과제

- socket.io를 사용해 polling하지 않고서 추가/변경/삭제되는 Lamp 목록을 보여준다.
- hard coded된 notice 문구 대신, 특정 Lamp에 문제가 생겼을 경우 서버에서 경고를 받아 출력한다.
- Action을 추가해, Lamp를 켜고/끄게 만든다.
- bower를 사용해 AngularJS 개발 과정에서 복잡한 프론트엔드 의존성을 자동으로 관리하게 만든다.

실제 사례

Thing 클라우드

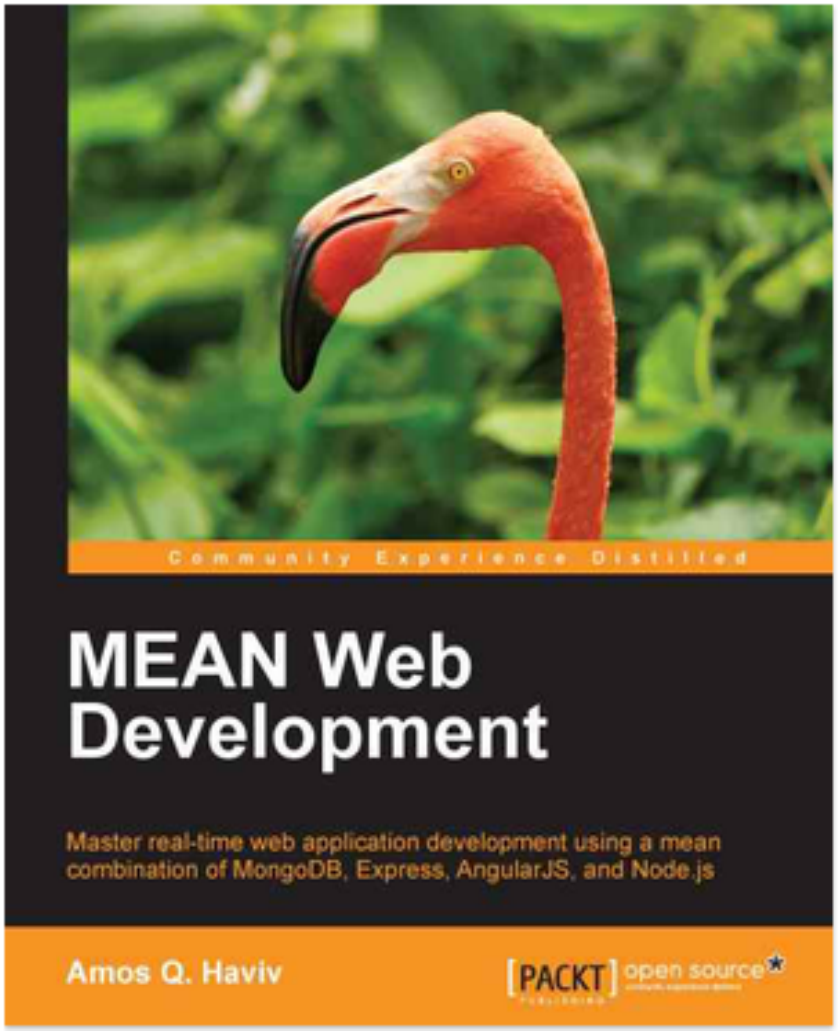
daliworks
a remarkable way

참고 자료

- http://docs.aws.amazon.com/lambda/latest/dg/walkthrough-custom-events.html
- http://ni-c.github.io/heimcontrol.js/
- http://coenraets.org/blog/2012/10/creating-a-rest-api-using-node-js-express-and-mongodb/
- https://blog.codecentric.de/en/2013/03/home-automation-with-angularjs-and-node-js-on-a-raspberry-pi/
- http://mean.io/
- http://bcho.tistory.com/881

참고 서적(번역 진행 중)


Write Modern Web Apps with the MEAN Stack
Mongo, Express, AngularJS, and Node.js
DEVELOP AND DESIGN
Jeff Dickey


Community Experience Distilled
MEAN Web Development
Master real-time web application development using a mean combination of MongoDB, Express, AngularJS, and Node.js
Amos Q. Haviv
[PACKT] open source


Q
&
A