Node.js 勉強会　第１回

2013　Oct.　6th

Yumin Oliver Huang

はじめに

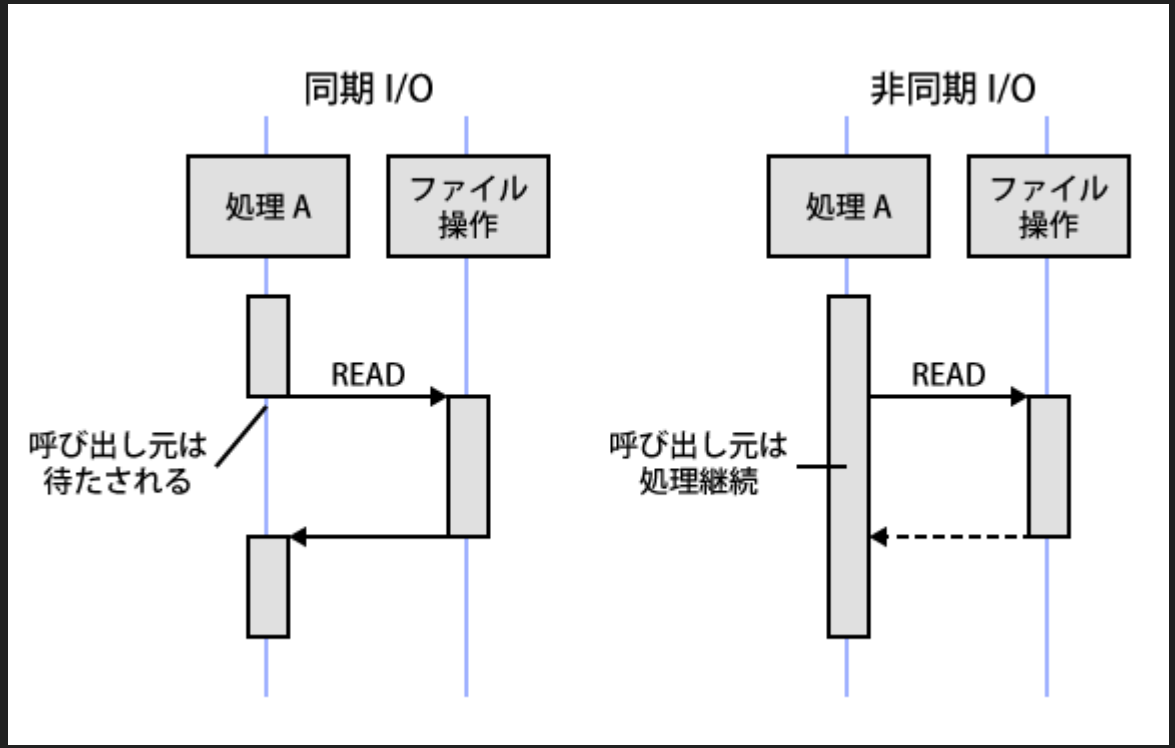
- サーバーサイドのJavaScriptで注目を浴びているNode（Node.js）を学んでいきます。
- 今回使用するVersionは最新の安定版を使っています。
- ほかの最新版(デベロッパー)などを使用した場合動かないことがあります。

使用バージョン

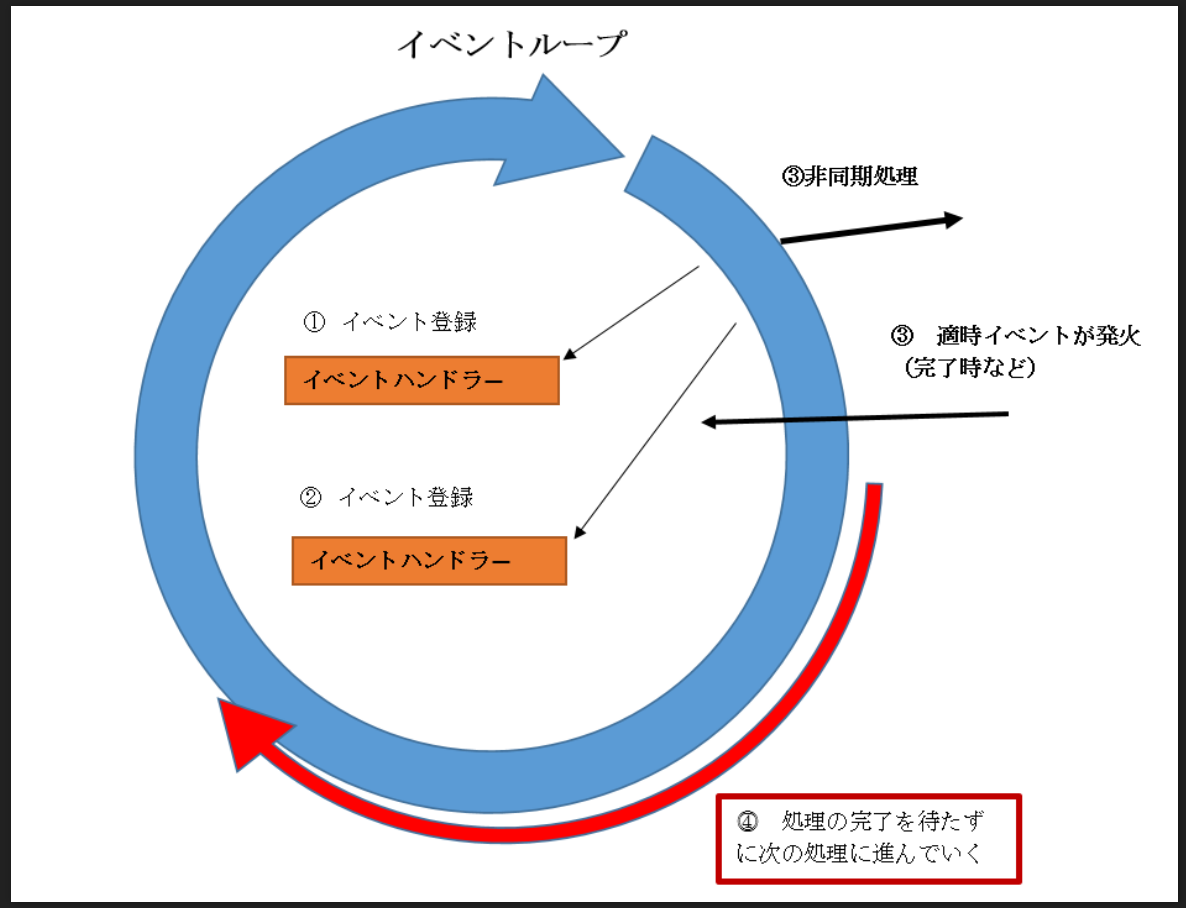
- OS : CentOS 6.4 64bit版
- Node : version 0.10.17
- Nave 0.4.3

Nodeとは

- 実行エンジンとしてGoogle製v8を使用
- 非同期I/oを主とする
  - シングルスレットイベントループモデル
- モジュールによる簡易な拡張



非同期とイベント駆動モデル



- 非同期っていうのは、簡単に言うと結果を待たずに先に進むことです。

Hello Node!!

```
var httpd = require('http').createServer(function(req, res) {
    res.setHeader('Content-Type', 'text/plain');
    res.write('Hello, Node!');
    res.end();
});
httpd.listen(8080);
```

```
$ node hello_node.js
```

Hello Node!!

それでは、ブラウザーで
http://XXX.XXX.XXX.XXX:8080 にアクセスしてみてください
＊XXX.XXX.XXX.XXXはLinuxのIPアドレスです。

Hello　Node!!

Hello Node!!

今回のNodeプログラムではcreateServer() コールバック関数を渡しています。サーバはリクエストを受け取ったら毎回その関数を実行しますがURLのパスに関する部分がありません。なので、挙動を見てみましょう。

以下にアクセスしてみてください。

http://XXX.XXX.XXX.XXX:8080

http://XXX.XXX.XXX.XXX:8080/aaa

http://XXX.XXX.XXX.XXX:8080/aaa/bbb/ccc.html

Hello Node 2!!

```
var  url  = require (‘url’);
var httpd = require(‘http’).createServer(function(req, res) {
        var pathname = url.parse(req.url).pathname;
        res.setHeader(‘Content-Type’, ‘text/plain’);
        if (pathname == ‘/hello’) {
                res.write(‘Hello, Node!’);
        } else if (pathname == ‘/bye’) {
                res.write(‘Goodbye, Node!’);
        } else {
                res.statusCode = 404;
                res.write(‘404: Not Found.’);
        }

        res.end();
});
httpd.listen(8080);

$ node hello_node.js
```

先ほど作成したhello_node.js 修正して、/helloにアクセスされた場合はHello, Node!と、/bye にアクセスされた場合はGoodbye,　Node!と表示するようにしてみましょう。

Package.Jason & Hello_Express

```
$ mkdir –p ~/node-work/project

$ cd ~/node-work/project
$ mkdir hello_express
$ cd hello_express
$ npm init
```

```
name: (hello_express)
version: (0.0.0)
description: hello_express
entry point: (index.js)
test command: mocha -R spec
git repository:
keywords:
author: oliver
license: (BSD-2-Clause)
About to write to
/home/node/Documents/node-
work/hello_express/package.json:
(略)
Is this ok? (yes):
```

Package.Jason

```
{
  "name": "hello_express",
  "version": "0.0.0",
  "description": "hello_express",
  "dependencies": {
        "express": "*"
   },
  “devDescription": {},
  "main": "index.js",
  "scripts": {
    "test": "mocha -R spec"
  },
  "author": "oliver",
  "license": "BSD-2-Clause"
}
```

package.jsonというファイルを作成しました。このファイルは json形式で、プロジェクトの情報や依存環境モジュールを管理します。

手入力で書いても良いのですが、今回はnpmの作成オプションを使って作成しました。

dependencies は本番環境用モジュール

devDependencies は開発用モジュール

とプロジェクトに

Express

$ vi package.json

```
"dependencies": {
    "express": "*"
  },
```

$ npm install

それではexpressを入れてみよう

Express

```
$ vi hello_express.js

var express = require('express')
      , app = express();

app.get('/', function(req, res) {
      res.send('Hello, Express!');
});

app.listen(8080);

$ node hello_express.js
```

起動できたら、ブラウザからアクセスをしてみましょう。

http://XXX.XXX.XXX.XXX:8080

Scaffold

```
$ cd ~/node-work/project
//-g グローバルにインストール
$ npm install express –g

//　スタイルシートとしてstylusを使用
$ express –c stylus hello_express2

// 依存モジュールのインストール
$ cd hello_express2 && npm install

$ node app.js
//npm start でもいける
```

それではexpressを入れてみよう

Scaffold MVC フレーム　ワーク

Scaffold で自動生成されたプロジェクトの構成

```
Root/                          ——  プロジェクトルート
 |-- app.js                    ——  ルーティングなどを行うメインファイル
 |-- package.json              ——  プロジェクト定義ファイル
 |-- node_modules              ——  プロジェクトで使用するモジュール
 |-- public                    ——  静的ファイルの格納
 |    |-- images               ——  画像ファイルの格納
 |    |-- javascripts          ——  JavaScriptファイルの格納
 |    |-- stylesheets          ——  スタイルシートの格納
 |         |-- style.styl      ——  Stylus形式のCSSファイル
 |
 |-- routes                    ——  Controllerを格納
 |    |-- index.js             ——  Indexアクセス時の処理
 |
 |-- views                     ——  Viewのテンプレートファイルの格納
      |-- index.jade           ——  indexページ用のテンプレート
      |--layout.jade           ——  ページ全体のテンプレート
```