

# Building your first Node app with Connect & Express

# Node's HTTP module

- Very low level.
- No cookie handling or parsing.
- No session support built-in.
- No routing built-in.
- No static file serving.

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

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

```
var server = http.createServer(function (req, res) {
```

```
});
```

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

var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var result = { version: '1.0.0' };
  res.end(JSON.stringify(result));
});
```

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

var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var result = { version: '1.0.0' };
  res.end(JSON.stringify(result));
});
```

```
server.listen(3000, localhost);
```

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

var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var result = { version: '1.0.0' };
  res.end(JSON.stringify(result));
});

server.listen(3000, localhost);
```

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

```
var handler = function (req, res) {  
  res.writeHead(200, {'Content-Type': 'application/json'});  
  var result = { version: '1.0.0' };  
  res.end(JSON.stringify(result));  
};
```

```
var server = http.createServer(handler);
```

```
server.listen(3000, localhost);
```



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

var handler = function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var result = { version: '1.0.0' };
  res.end(JSON.stringify(result));
};

var server = http.createServer(handler);

server.listen(3000, localhost);
```

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

var dispatcher = function (req, res) {
  switch(req.url) {
    case '/version': version(req, res); break;
    case '/user': user(req, res); break;
    default: notFound(req, res); break;
  };
};

var server = http.createServer(dispatcher);

server.listen(3000, localhost);
```

# Why a framework?

- foundation for collaboration
  - required in large teams
- reuse common patterns
  - middleware, plugins, modules
- battle tested
  - sockets, timeouts, memory leaks, stability, debugging

connect

# Connect/Express Middleware

- Static File Server
- Body Decoder
- Cookie Decoder
- Session Provider
- Request Router
- Logs
- Sass/Less Compiler
- Virtual Host Router



You decide what blocks you want.

# .use() and next()

```
var connect = require('connect');
var s = connect();

s.use(function (req, res, next) {
  // Send to next handler if url not /about
  if (req.url !== '/about') return next();
  res.end('About page');
});

s.use(function (req, res, next) {
  res.end('Default page');
});

s.listen(8080, '127.0.0.1');
```

# Connect Router

```
s.use(require('connect-route')(function(app) {  
  app.get('/users', function(req, res, next) {  
    res.end('List users');  
  });  
  
  app.post('/users', function(req, res, next) {  
    res.end('Create a new user');  
  });  
  
  app.get('/user/:id', function(req, res, next) {  
    if (!is_admin(req)) return next(new Error('access denied'));  
    res.end('Edit user ' + req.params.id);  
  });  
}));  
  
s.use(function(err, req, res, next) {  
  res.end(err.message);  
});
```



# Full example with Connect

```
1 let connect = require('connect')
2   , app = connect()
3
4 // Http request logging
5 app.use(require('morgan')('combined'))
6
7 // favicon.ico support
8 app.use(require('serve-favicon')(__dirname + '/../static/favicon.ico'))
9
10 // Static file support
11 app.use(require('serve-static')(__dirname + '/static', {'index': false}))
12
13 // Cookie-based session support
14 app.use(require('cookie-session')({keys: ['secret1', 'secret2']})) // adds req.cookies
15
16 // Add req.body
17 app.use(require('body-parser').urlencoded())
18 app.use(require('body-parser').json())
19 app.use(require('body-parser').text())
20
21 app.use(require('connect-route')(require('./routes/users')))
22 app.use(require('connect-route')(require('./routes/index')))
23
24 // Error handler middleware
25 app.use((err, req, res, next) => {
26   res.end('Internal Server Error')
27 })
28
29 app.listen(8080, '127.0.0.1')
```

# Configurations

```
app.configure(function() {  
  // Global configurations  
  app.use(express.bodyParser()); // parse incoming data  
  app.use(express.cookieParser()); // parse cookies  
  app.use(express.session({ secret: 'your secret' }));  
});
```

```
app.configure('development', function() {  
  // Development configurations  
  // i.e. Error handler to print errors and show stack?  
});
```

```
app.configure('production', function() {  
  // Production configurations  
});
```

Controlled by:      `NODE_ENV=production node app.js`

express

# Express is easier

- Connect is easier than raw node.js, but still low level.
- ...but, it's good for simple RESTful API's
- ...or to make your own framework.
- Express is a simple framework inspired by Sinatra (Ruby).
- It's built on top of Connect.

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

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

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

```
var app = express();
```

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

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

```
var app = express();
```

```
app.configure(function () {  
  app.use(app.router);  
});
```

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

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

```
var app = express();
```

```
app.configure(function () {
```

```
  app.use(app.router);
```

```
});
```

```
app.get('/version', function (req, res) {
```

```
  res.writeHead(200, {'Content-Type': 'application/json'});
```

```
  var result = { version: '1.0.0' };
```

```
  res.end(JSON.stringify(result));
```

```
});
```



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

var app = express();

app.configure(function () {
  app.use(app.router);
});

app.get('/version', function (req, res) {
  res.writeHead(200, {'Content-Type': 'application/json'});
  var result = { version: '1.0.0' };
  res.end(JSON.stringify(result));
});

app.listen(3000);
```

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

var app = express();

app.configure(function () {
  app.use(app.router);
});

app.get('/version', function (req, res) {
  var result = { version: '1.0.0' };
  res.json(result);
});

app.listen(3000);
```

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

var app = express();

app.configure(function () {
  app.use(app.router);
});

app.get('/version', function (req, res) {
  var result = { version: '1.0.0' };
  res.json(result);
});

app.listen(3000);
```

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

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

```
var app = express();
```

```
app.configure(function () {  
  app.use(app.router);  
});
```

```
var version = function (req, res) {  
  var result = { version: '1.0.0' };  
  res.json(result);  
};
```

```
app.get('/version', version);
```

```
app.listen(3000);
```

```
var user = function (req, res) {  
  
    var item = { id: 1, name: 'steve' };  
  
    res.json(item);  
};  
  
app.get('/user', user);
```

```
var users = require('./users.json');
```

```
var user = function (req, res) {
```

```
    var item = users[req.query.id];
```

```
    res.json(item);
```

```
};
```

```
app.get('/user', user);
```

```
var users = require('./users.json');

var user = function (req, res) {

  var item = users[req.query.id];
  if (!item) return res.send(404, 'not found');
  res.json(item);
};

app.get('/user', user);
```

```
var users = require('./users.json');
```

```
var user = function (req, res) {
```

```
    var item = users[req.params.id];
```

```
    if (!item) return res.send(404, 'not found');
```

```
    res.json(item);
```

```
};
```

```
app.get('/user/:id', user);
```



```
var users = require('./users.json');
```

```
var user = function (req, res) {  
  if (!req.params.id) return res.json(users);  
  var item = users[req.params.id];  
  if (!item) return res.send(404, 'not found');  
  res.json(item);  
};
```

```
app.get('/user/:id?', user);
```

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

```
app.configure(function () {
```

```
    app.use(app.router);  
});
```

```
app.get('/user', user);
```

```
var express = require('express');  
var app = express();  
var bodyParser = require('body-parser')
```

```
app.configure(function () {  
  app.use(bodyParser.text());  
  app.use(app.router);  
});
```

```
app.get('/user', user);
```

```
var express = require('express');  
var app = express();  
var bodyParser = require('body-parser')
```

```
app.configure(function () {  
  app.use(bodyParser.text());  
  app.use(app.router);  
});
```

```
app.get('/user', user);
```

```
var register = function (req, res) {  
  // use req.body to create new user  
};  
app.post('/user', register);
```

```
var express = require('express');  
var app = express();  
var bodyParser = require('body-parser')
```

```
app.configure(function () {  
  app.use(bodyParser.text());  
  app.use(app.router);  
});
```

```
app.get('/user', user);
```

```
var register = function (req, res) {  
  // use req.body to create new user  
};  
app.post('/user', register);
```

```
app.configure(function () {  
  app.use(bodyParser());  
  app.use(app.router);  
});
```

```
var myMiddleware = function (req, res, next) {  
  res.setHeader('X-API-Version', '1.0.0');  
  return next();  
};
```

```
app.configure(function () {  
  app.use(bodyParser());  
  app.use(app.router);  
  app.use(myMiddleware);  
});
```

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

```
app.configure(function () {  
    app.use(app.router);  
});  
  
app.get('/user', user);
```



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

```
var validate = function (username, password, callback) {  
    var result = (username === 'steve' && password === '12345');  
    callback(null, result);  
};
```

```
app.configure(function () {  
    app.use(app.router);  
});
```

```
app.get('/user', user);
```

```
var express = require('express');
var app = express();
var basicAuth = require('basic-auth-connect');

var validate = function (username, password, callback) {
  var result = (username === 'steve' && password === '12345');
  callback(null, result);
};
var auth = basicAuth(validate);

app.configure(function () {

  app.use(app.router);
});

app.get('/user', user);
```

```
var express = require('express');
var app = express();
var basicAuth = require('basic-auth-connect');

var validate = function (username, password, callback) {
  var result = (username === 'steve' && password === '12345');
  callback(null, result);
};

var auth = basicAuth(validate);

app.configure(function () {
  app.use(auth);
  app.use(app.router);
});

app.get('/user', user);
```

```
var express = require('express');
var app = express();
var basicAuth = require('basic-auth-connect');

var validate = function (username, password, callback) {
  var result = (username === 'steve' && password === '12345');
  callback(null, result);
};

var auth = basicAuth(validate);

app.configure(function () {

  app.use(app.router);

});

app.get('/user', auth, user);
```

# Sanitizing Route Params

```
app.param('/:id', function(v) {  
  return parseInt(v, 10);  
});
```

```
app.get('/user/:id', function(req, res) {  
  res.send('user id: ' + req.params.id);  
});
```

# Route Param Functions

```
var integer = function(v) {  
  return parseInt(v, 10);  
};
```

```
app.param('/:id', integer);
```

```
app.get('/user/:id', function(req, res) {  
  res.send('user id: ' + req.params.id);  
});
```

# Redundant route code...

```
app.get('/user/:id', function(req, res) {  
  var id = req.params.id;  
  User.get(id, function(err, user) {  
    if (err) return next(err);  
    res.send('user ' + user.name);  
  });  
});
```

```
app.put('/user/:id', function(req, res) {  
  var id = req.params.userId;  
  User.get(id, function(err, user) {  
    if (err) return next(err);  
    user.update(req.body);  
  });  
});
```

Same logic every time there is :id

# Route Middleware

```
var loadUser = function(req, res, next) {  
  var id = req.params.id;  
  User.get(id, function(err, user) {  
    if (err) return next(err);  
    if (!user) return next(new Error('invalid userId'));  
    req.user = user;  
    next();  
  });  
});  
  
app.get('/user/:id', loadUser, function(req, res) {  
  res.send('user ' + req.user.name);  
});  
  
app.put('/user/:id', loadUser, function(req, res) {  
  req.user.update(req.body);  
});
```



# Stacking Route Middleware

```
app.put('/user/:id', isAdmin, loadUser, function(req, res) {  
  req.user.update(req.body);  
});
```

...cleaner than...

```
app.put('/user/:id', function(req, res, next) {  
  // Verify if the current user is an admin  
  isAdmin(req, res, function(err) {  
    if (err) return next(err);  
    loadUser(req, res, function(err, user) {  
      if (err) return next(err);  
      req.user.update(req.body);  
    });  
  });  
});
```

# Views

- "Built-in" support for :
  - Jade (Haml-like): <http://jade-lang.com/>
  - EJS (Embedded JS): <http://embeddedjs.com/>
- Support for partials.

# Rendering Views

```
app.get('/user/:id', loadUser, function(req, res) {  
  res.render('user_edit', {  
    locals: {  
      user: req.user,  
      title: 'Edit User ' + req.user.username  
    }  
  });  
});
```

# koa

```
1 // See Koa Wiki for supported middleware: https://github.com/koajs/koa/wiki
2 var koa = require('koa')
3   , app = koa()
4
5 // Error handler middleware
6 app.use(function *(next) {
7   try {
8     yield next
9   } catch (e) {
10    this.body = 'Internal Server Error: \n' + e.stack
11   }
12 })
13
14 // Http request logging
15 app.use(require('koa-logger'))
16 // favicon.ico support
17 app.use(require('koa-favicon')(__dirname + '/static/favicon.ico'))
18 // Static file support
19 app.use(require('koa-static')('./static'))
20 // Cookie-based session support
21 app.use(require('koa-session')(keys: ['secret1', 'secret2'])) // adds req.cookies
22 // Add this.req.body
23 app.use(require('koa-body-parser'))
24
25 app.use(require('koa-router')(app))
26 app.get('/home', function*(next) {
27   var user = yield User.findOne(this.params.id);
28   this.body = user;
29 })
30
31 app.listen(8080, '127.0.0.1')
```