

express

High performance, high class web development for **Node.js**

ExpressJS

Web development with ExpressJS

Telerik Software Academy
Learning & Development
<http://academy.telerik.com>

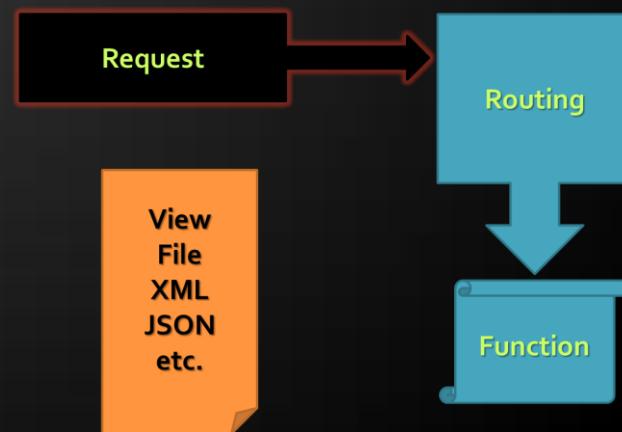
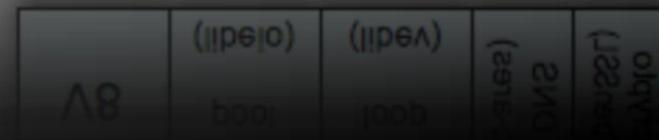
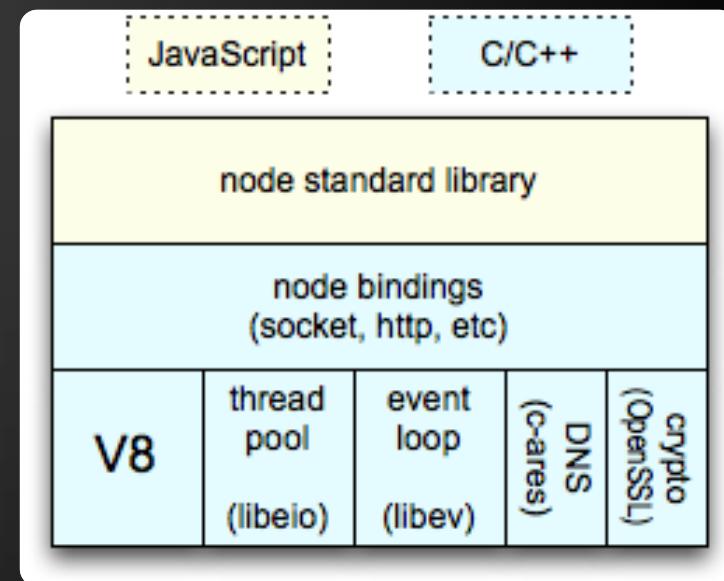


Table of Contents

1. Middleware
2. ExpressJS
3. Views and layout
4. Working with Data
5. Common and Advanced Scenarios



- ◆ Event-Driven, Asynchronous IO, Server-Side JavaScript library in C
- ◆ Open Source
- ◆ Available on
 - ◆ Windows
 - ◆ Service
 - ◆ Under IIS (iisnode)
 - ◆ *nix systems
- ◆ As a service
 - ◆ Azure
 - ◆ Heroku



- ◆ Basic server implementation

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

http.createServer(function(req, res) {
    res.writeHead(200, {
        'Content-Type': 'text/plain'
    }); //return success header

    res.write('My server is running! ^_^'); //response
    res.end(); //finish processing current request
}).listen(1234);
```

- ◆ Connect is a middleware framework for node
 - ◆ Built on top of node's Http Server
 - ◆ <http://www.senchalabs.org/connect/>

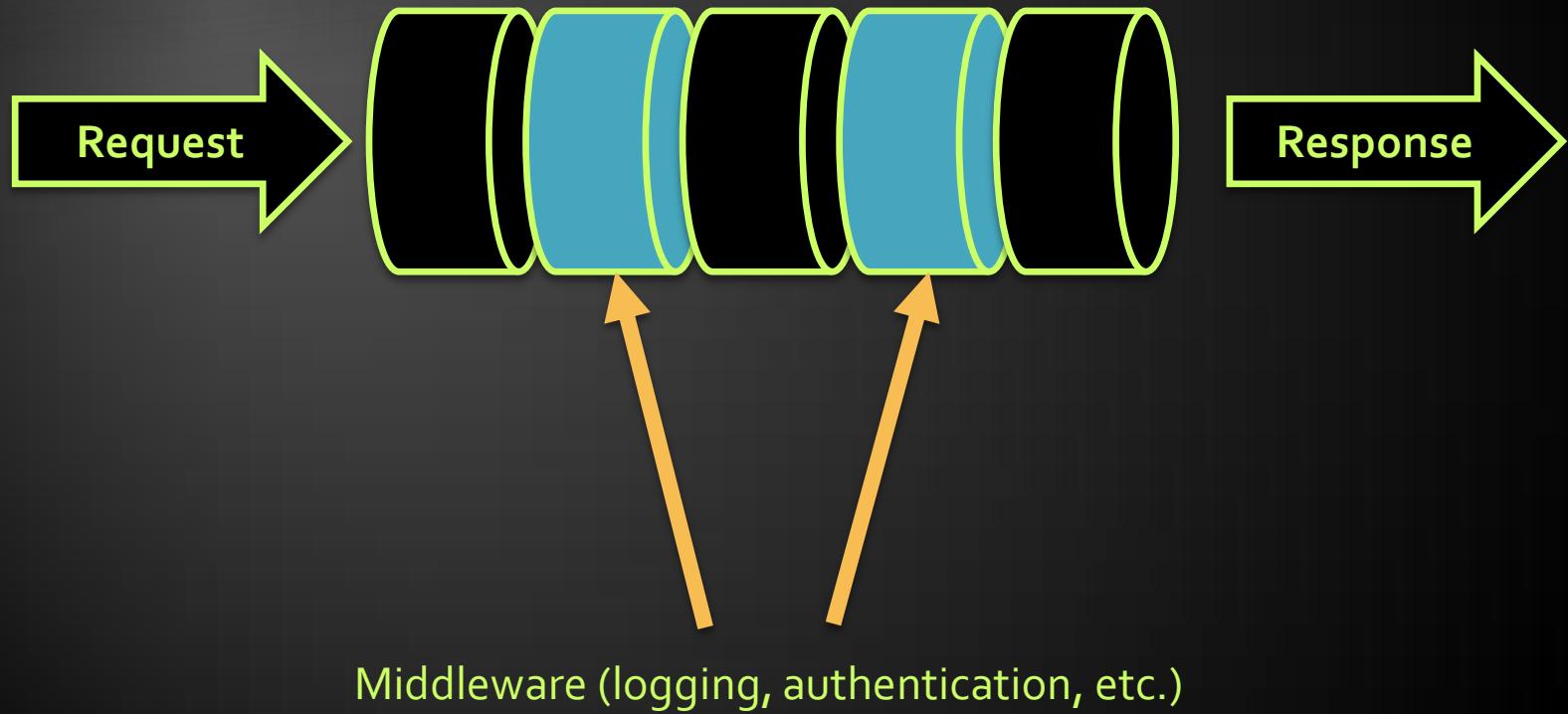
```
$ npm install connect
```

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

var app = connect()
  .use(connect.logger('dev'))
  .use(connect.static('public'))
  .use(function(req, res){
    res.end('hello world\n');
  })

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

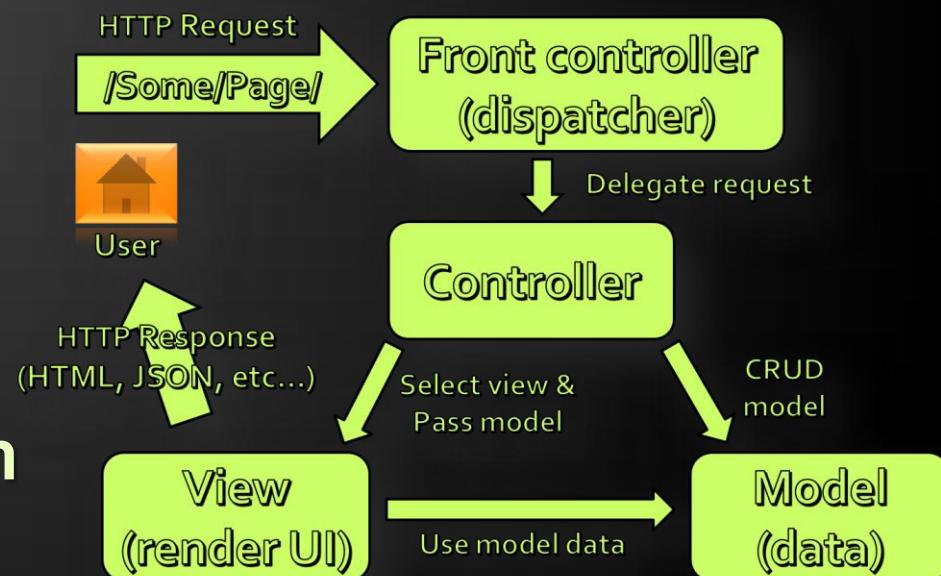
◆ Request Processing Pipeline



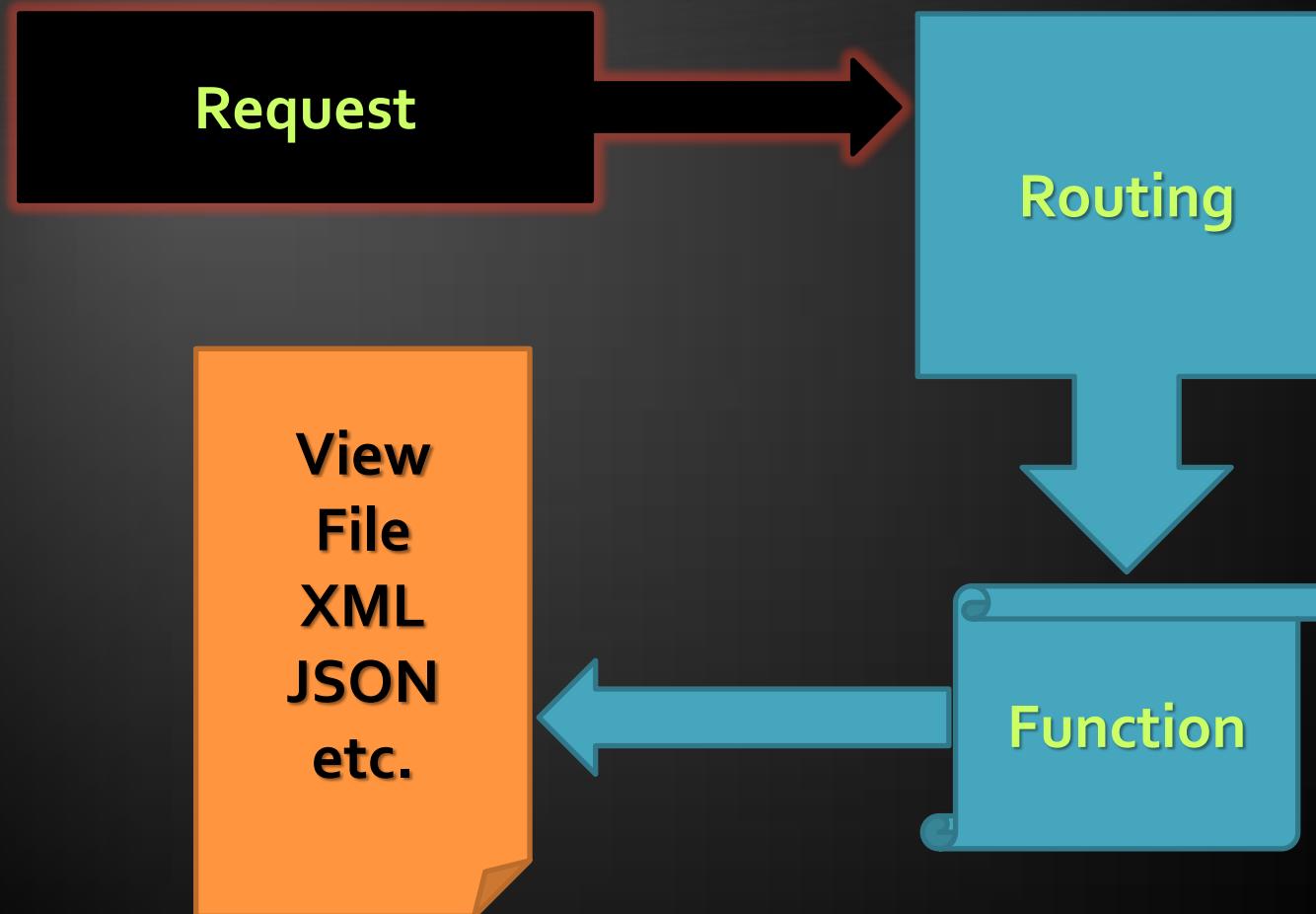
◆ Custom middleware function for connect

```
var connect = require('connect'),  
    util = require('util');  
  
var interceptorFunction = function(request, response, next) {  
    console.log(util.format('Request for %s with method %s',  
        request.url, request.method));  
    next();  
};  
  
var app = connect()  
    // .use('/log', interceptorFunction)  
    .use(interceptorFunction)  
    .use(function onRequest(request, response) {  
        response.end('Hello from Connect!');  
    }).listen(3001);
```

- ◆ Has middleware built-in
- ◆ Adds functionality to the normal server
 - ◆ Request / Response enhancements
 - ◆ Routing
 - ◆ View Support
 - ◆ HTML Helpers
 - ◆ Content Negotiation



Basic Architecture



First Express App

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

var app = express();

app.get('/', function (request, response) {
    response.send('Welcome to Express!');
});

app.get('/customer/:id', function (req, res) {
    res.send('Customer requested is ' + req.params['id']);
});

app.listen(3000);
```

express

High performance, high class web development for **Node.js**

Demo: Creating Express Applications

- Simple ExpressJS application and "nodemon"
- Create routes and require() them
- Pass parameters
- Configure middleware

- ◆ User Interface
- ◆ Based on Templates
- ◆ Support for multiple View Engines
 - ◆ Jade, EJS, JSHTML, ...
- ◆ Default is Jade
 - ◆ <http://jade-lang.com>

```
app.get('/', function (req, res) {  
  res.render('index');  
});
```

Views in ExpressJS – Example

```
var express = require('express'),  
    path = require('path');  
var app = express();  
app.configure(function () {  
    app.set('views', __dirname + '/views');  
    app.set('view engine', 'jade');  
    app.use(express.static(path.join(__dirname, 'public')));  
});  
app.get('/', function (req, res) {  
    res.render('empty');  
});  
app.listen(3000);
```

```
doctype  
html(lang="en")  
head  
    title Welcome to this empty page  
body
```



Demo: Views in ExpressJS

- Show simple views in ExpressJS
- Jade syntax examples
- Layouts and blocks
- Stylus

- ◆ Pass data to the views

```
res.render('index', { title: 'Customer List' });
```

- ◆ Read data from the views (bodyParser)

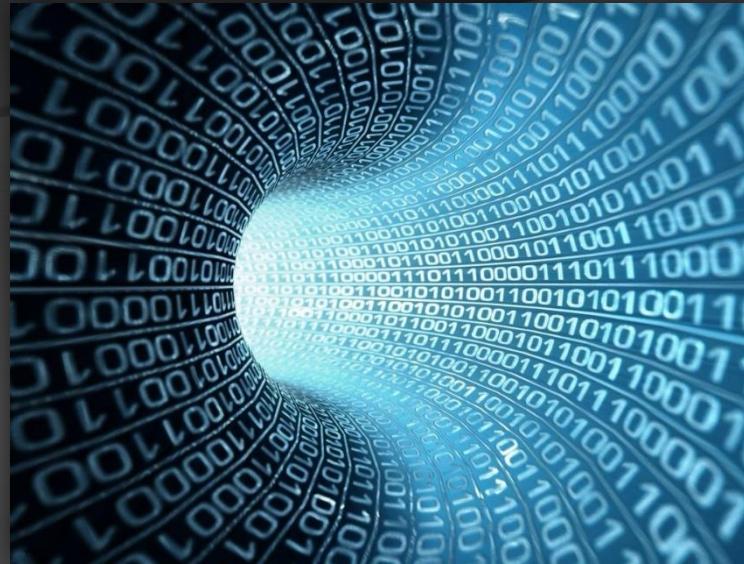
```
res.render('index', { title: 'Customer List' });
```

- ◆ Read and send files

```
var filePath = req.files.picture.path;  
// ...  
res.download(filePath);  
res.sendfile(filePath);
```

- ◆ Data for all views

```
app.locals.clock = { datetime: new Date().toUTCString()};
```



Demo: Working with data

- Pass data to views (`customer.index`)
- Submit data from views (`customer.create`)
- Content negotiation (`customer.details`)
- Upload files (`customer.create`)
- Helpers (`app.locals`)



Demo: Advanced Scenarios

- Cookies
- Sessions
- Custom middleware
- Authentication and authorization

- ◆ Express.js Samples
 - ◆ <https://github.com/visionmedia/express>
- ◆ Database Support
 - ◆ MS SQL
 - ◆ CouchDB
 - ◆ PostgreSQL
 - ◆ Redis
- ◆ Socket.io
 - ◆ Real-time support

- ◆ Search on [npm.org](https://www.npm.org) before you re-invent the wheel
- ◆ Express is Lightweight Framework and Fast to work with
- ◆ Testing is not optional
 - ◆ Mocha
- ◆ JavaScript can get messy

Questions?

1. Create a web site (with normal design) using Express with private and public parts – CRUD operations over a model by your choice
 - At least 6 pages
 - Use Express
 - Use File upload
 - Use Jade
 - Use Stylus
 - Use Passport module for authentication
 - Use good application architecture