CS 5142
Scripting Languages

10/14/2013

Server-Side Javascript

Packages (Revisited)

- HTML tag `<script src="mod.js">` and coding conventions
  - No separate JavaScript language feature
- Problem: Global namespace pollution
  - What about require from last class?
- Node adds a simple module loading system

Node Modules

foo.js:

```
var MY_CONSTANT = 100;
exports.getit = function () {
  return MY_CONSTANT;
};
```

Add a new property to the module.exports object

or foo.js:

```
function getMyConstant() {
  return MY_CONSTANT;
};
exports.getit = getMyConstant;
```

bar.js:

```
var myfoo = require('./foo.js');
console.log( 'My imported function ' + myfoo.getit());
```

require returns the exports object

Modules

Core modules
are in node/lib

```
assert.js
dns.js
fs.js
http.js
https.js
module.js
net.js
os.js
...
```

Install other modules with
Node Packaged Modules (npm)

```
# install the module ‘sqlite3’

$ npm install sqlite3
```

Node and Databases

```
var fs = require("fs");
var file = "test.db";
var exists = fs.existsSync(file);
var sqlite3 = require("sqlite3").verbose();
var db = new sqlite3.Database(file);

db.serialize(function() {
  if(!exists) {
    db.run("CREATE TABLE Stuff (thing TEXT)");
  }

  var stmt = db.prepare("INSERT INTO Stuff VALUES (?)");
  for (var i = 0; i < 10; i++) {
    stmt.run("index #" + i);
  }
  stmt.finalize();

  db.each("SELECT rowid AS id, thing FROM Stuff", function(err, row) {
    console.log(row.id + ": " + row.thing);
  });
});

db.close();
```

Revisiting HTTP Server

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

var app = http.createServer(function (req, res) {
  res.writeHead(200, {
      'Content-Type': 'text/plain'
  });
  res.end('Hello World\n');
});
app.listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Use built-in module ‘http’

Create a server, pass a function that listens for requests

Tell the server to listen on port 1337

HTTP Protocol

```
initial line
Header1: value1
Header1: value1
Header1: value1

body?
```

GET /path/to/file HTTP/1.0
POST /path/to/file HTTP/1.0
HTTP/1.0 200 OK
HTTP/1.0 404 Not Found

0 or more headers

CRLF (blank line)

Optional body

Request Handler

```
var app = http.createServer(function (req, res) {
  var ret = "";

  ret += "Request URL:" + req.url + "\n";
  ret += "Method:" + req.method + "\n";
  ret += "Headers:" + JSON.stringify(req.headers)+"\n";

  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end(ret);
});
```

HTTP Headers

Tell the server that all response headers and the ody have been sent.

Status code
e.g., 200 Success,
404 Not found

Serving Multiple Pages

```
var app = http.createServer(function (req, res) {
  if (req.url == "/") {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end("Home Page");
  } else if (req.url == "/about") {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end("About Page");
  } else {
    res.writeHead(404, {'Content-Type': 'text/plain'});
    res.end("Page not found!");
  }
```

Home Page

About Page

Looking for something else

Processing Stages

- Often want to perform multiple processing steps on the server
  - Example: log the request, look in a cache for content, serve content from disk...
- The connect module provides bundled middleware and processing pipeline for node

Connect

```
var connect = require("connect");
var http = require("http");
var app = connect();

app.use(function(request, response, next) {
  console.log("Request for: " + request.url);
  next();
});

app.use(function(request, response) {
  response.writeHead(200, { "Content-Type": "text/plain" });
  response.end("Connect is working!\n");
});
```

Multiple Pages with Connect

Home Page

```
app.use(function(request, response, next) {
  if (request.url == "/") {
    response.writeHead(200,
      { "Content-Type": "text/plain" });
    response.end("Home Page!\n");
  } else {
    next();
  }
});
```

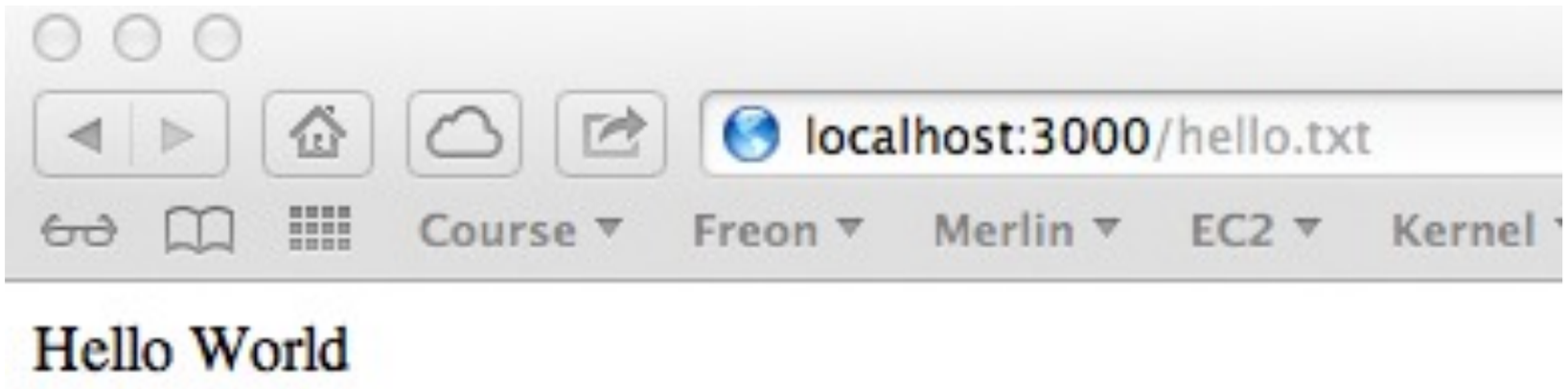
If not home page, go to
“use” function in the pipeline

Express for Web Apps

```
$ mkdir hello-world
$ cd hello-world
# create package.json file
$ cat package.json
{
  "name": "hello-world",
  "description": "hello world test app",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "express": "3.x"
  }
}

$ npm install
```

Express Hello World

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

app.get('/hello.txt', function(req, res){
  res.send('Hello World');
});

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



Multiple Pages or “Routing”

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

app.get('/', function(req, res){
  res.send('Home Page');
});

app.get('/about', function(req, res){
  res.send('About Page');
});

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

Using Templates

- Boilerplate HTML with tags that will insert variables or run program logic
- Replace tags with dynamic content
- Many examples: moustache, haml, dust.js
- We will look at Jade

Install Jade

```
$ mkdir hello-world
$ cd hello-world
# create package.json file
$ cat package.json
{
  "name": "hello-world",
  "description": "hello world test app",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "express": "3.x",
    "jade" : "0.27.2"
  }
}

$ npm install
```

Jade Templates

```
doctype 5
html(lang="en")
  head
    title= Jade
  body
    h1 Hello!
      p some stuff.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
  </head>
  <body>
    <h1>Hello!</h1>
      <p> some stuff. </p>
      </p>
    </div>
  </body>
</html>
```

Express + Jade

```
.
|____app.js
|____node_modules
|____views
  |____index.jade
  |____layout.jade
```

```
extends layout
block content
        p Hello <b>#{name}</b>
```

```
doctype 5
html
  head
    title= title
    block head
  body
```

Express + Jade

```
var express = require('express');
var app = express();
app.listen(3000);

app.get('/', function(req, res){
        res.render('index.jade', {
                        'title': 'Hello World',
                        'header': 'Hello World',
                        'name': 'Robert'
        })
})
```

Partial Templates

```
.
|____app.js
|____node_modules
|____views
  |____index.jade
  |____layout.jade
  |____partials
  | |____hello.jade
```

```
extends layout
block content
      each name in names
         .name
           include partials/hello
```

Partials in Action

```
var express = require('express');
var app = express();
app.listen(3000);
var names = ['Robert', 'Jim', 'Alice']
app.get('/', function(req, res){
        res.render('index.jade', {
                        'title': 'Hello World',
                        'header': 'Hello World',
                        'names': names
        })
})
```