Chapter 19.5: node.js

CS 80: Internet Programming

Instructor: Mark Edmonds

Overview

- Node (node.js) enables server-side JavaScript
- Not a programming language, but rather a server-side application that allows you to program server-side applications in JavaScript
- Much of what we did with PHP can be done with node.js
- Node has much more functionality than PHP; you can basically write whatever JavaScript program you want without needing to execute in a browser

Overview

- Node is designed to make it easy to write I/O-based programs that run of a server
- I/O-based programs include web servers, databases, etc.

Overview

- Node uses event-based asynchronous processing
- We will use same event-listener and callbacks we learned in JavaScript

Installing Node

- Node can be installed from nodejs.org
- Node is a command-line program, and you start node with by typing `node` at your terminal/command prompt

Hello, world!

- Save the following in `hello_world.js`

```
1 console.log("Hello, world!");
```

- Launch the program with

```
node hello_world.js
```

Example: `http_server.js`

```
// basic HTTP server with an export
var http = require("http"); // require the node HTTP module

function onRequest(request, response)
{
  console.log("Request received.");
  response.writeHead(200,
  {
    "Content-Type": "text/plain"
  }); // set HTTP response header
  response.write("Hello World"); // write content into HTTP request
  response.end(); // finishes the response
}

http.createServer(onRequest).listen(8888);

console.log("Server has started.");
```

Modules

- We wrote `var http = require("http");` in the HTTP server example
- http is a module that our node application requires
- But we also want to write our own models
- This is accomplished using `exports`

Example: `http_server_export.js`

```
// basic HTTP server with an export
var http = require("http"); // require the node HTTP module

function start()
{
  function onRequest(request, response)
```

```
7    {
8      console.log("Request received.");
9      response.writeHead(200,
10     {
11       "Content-Type": "text/plain"
12     }); // set HTTP response header
13     response.write("Hello World"); // write content into HTTP request
14     response.end(); // finishes the response
15   }
16
17   http.createServer(onRequest).listen(8888);
18   console.log("Server has started.");
19 }
20
21 exports.start = start;
```

Example: `index.js`

```
1 var server = require("./http_server_export");
2
3 server.start();
```

Modules

- Modules are a core component of node.js
- They allow you to modularize code
- This breaks our I/O-based application easier to manage and scalable
- Each module can be responsible for a specific kind of I/O

Routing

- So far, every HTTP resquest was handled the same way
- Routing allows us to specify which modules process certain HTTP requests
- We'll look at the URL and the data in the GET/POST parameters and make a decision about where this HTTP request should be routed.

Example: `router.js`

```
// create a router; this could also route to different applications (
    controllers) based on the file extension, or other information
// here, we just print the request we are routing
function route(pathname)
{
  console.log("About to route a request for " + pathname);
}

exports.route = route;
```

Example: `http_server_router.js`

```
var http = require("http"); // require the node HTTP module
var url = require("url");

function start(route)
{
  function onRequest(request, response)
  {
    var pathname = url.parse(request.url).pathname;
    console.log("Request for " + pathname + " received.");

    route(pathname);

    response.writeHead(200,
    {
      "Content-Type": "text/plain"
    }); // set HTTP response header
    response.write("Request for " + pathname + " received."); // write
        content into HTTP request
    response.end(); // finishes the response
  }

  http.createServer(onRequest).listen(8888);
  console.log("Server has started.");
}

exports.start = start
```

Example: `index_router.js`

```
var server = require("./http_server_router");
var router = require("./router");

server.start(router.route);
```

Fileserver

- But we actually want our webserver to *do* something.
- Let's write a basic webserver, capable of serving files to the client (similar to our Python SimpleHTTPServer we used earlier)

Example: `http_fileserver.js`

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

function start(route)
{
  function onRequest(request, response)
  {
    // look for file in current directory
    var filename = "." + url.parse(request.url).pathname;
    console.log("Request for " + filename + " received.");
    fs.readFile(filename, function(err, data)
    {
      // error getting the file, return a 404
      if (err)
      {
        response.writeHead(404,
        {
          'Content-Type': 'text/html'
        });
        return response.end("404 Not Found");
      }
      // successfully retrieved file
      response.writeHead(200,
      {
```

```
26          'Content-Type': 'text/html'
27        });
28        // write the file's data into the response
29        response.write(data);
30        return response.end();
31      });
32    }
33    http.createServer(onRequest).listen(8888);
34    console.log("Server has started.");
35  }
36
37  exports.start = start
```

Example: `index_fileserver.js`

```
1  var server = require("./http_fileserver");
2  var router = require("./router");
3
4  server.start(router.route);
```