Chapter 19.5: node.js

CS 80: Internet Programming

Instructor: Mark Edmonds

Overview

- Node (node.js) enables server-side 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

```
1 node hello_world.js
```

Example: `http_server.js`

```
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`

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

function start() {
  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.");
}

```

```
15 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`

```
1 function route(pathname) {
2   console.log("About to route a request for " + pathname);
3 }
4
5 exports.route = route;
```

Example: `http_server_router.js`

```
1 var http = require("http"); // require the node HTTP module
2 var url = require("url");
3
```

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

Example: `index.js`

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