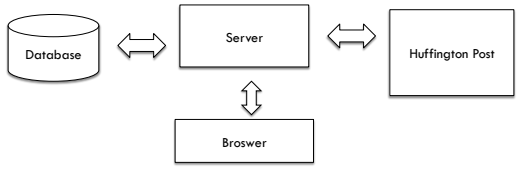
ECS 189
WEB PROGRAMMING

5/5

Announcements

- Today:
  - Server and node.js
- Remainder of class organized around larger project
- Do it in several steps
- This week: first server, static Web pages



- We write server and browser code
- Database can be running on same machine as server, but the interface to it is something like an API call

Server

- The HTML, CSS an Javascript that run on the browser are usually downloaded from a server, over the internet.
- A typical Web page generates queries that are sent to the browser, similar to the API calls we made in the the Weather App.
- So the server has to generate JSON responses and send them to the browser. These are called AJAX queries (Asynchronous JavaScript and XML…but we'll use JSON instead of XML).

Our server

- We're using a cloud server from a company called Digital Ocean
- Our server has the elegant name:
  138.68.25.50
- Getting it a real name would have cost us more money….soon this name will be very familiar to you.

Node.js

- Our server is a Unix machine, like most (but not all) servers
- Our server code will be written using node.js.
- Node.js is a way to run Javascript programs from the Unix command line:

```
node index.js
```

…runs the Javascript program in the file index.js.

Node.js

- Node.js runs on several OS's
- It uses V8, Google's Javascript compiler (the compiling is going on under the hood, you never see it, unlike C which you have to compile yourself)

Life before Node.js

- The classic Web browser runs on what was called the LAMP stack:

  Linux, Apache (Web server), MongoDB (database), PHP (scripting language).
- Node.js kind of replaces Apache+PHP. A server still needs an OS and, usually, a database.

Server modules in node.js

- Node.js also includes a set of Javascript modules that help us deal with problems like:
  - serving Web pages,
  - responding to AJAX queries,
  - querying APIs
  - and interacting with a database.

Modules

- A module is a file containing Javascript code.
- Objects, data and functions that programs in other files can see are labeled external.
- Modules provide another level of encapsulation and data hiding (in addition to functions and objects).
- They are something like C or C++ libraries.
- Node.js has modules, browsers do not! (even though they can use imported scripts such as JQuery or Angular).

Ports

- Many processes on the server are connecting to other machines over the internet
- To direct incoming traffic to the right process, each process uses a unique port number
- At the operating system/TCP level, a message comes in off the internet, and the system uses the port number to create an interrupt for the appropriate process
- We will each have our own permanent port number so we don't interfere with each other

Server code at a lower level

- Mostly hidden by node.js
- A Web server gets http requests and produces http responses.
- Http is a protocol for sending and receiving messages over the internet.
- Http requests and responses have:

  Header

  Body (sometimes)

HTTP request (browser->server)

GET /simple.html HTTP/1.1

Host: 45.55.29.158:8081

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:37.0) Gecko/ 20100101 Firefox/37.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive

Cache-Control: max-age=0

Body is empty.

HTTP response (server->browser)

HTTP/1.1 200 OK

Content-Type: text/html

Date: Thu, 30 Apr 2015 15:55:44 GMT

Connection: keep-alive

Transfer-Encoding: chunked

Body contains html file.

Some popular response codes

- 200 - OK "here's what you wanted"
- 301 – Moved Permanently "look over there"
- 304 – Not Modified "same as last time you asked so I am not sending the body again"
- 404 – Not Found "what the heck?"

Accessing the server

ssh 138.68.25.50

- You should be able to login using your Kerberos account credentials
- To get your port number, run:

  get-my-cs189h-port

…and it should type the port number you should use.

Simple Web server

From Eloquent Javascript, Chapter 20

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

- Brings in the http module.
- To keep the namespaces of modules distinct, all objects and functions from http have to be prefaced by "http."
- This is the same as object syntax

Handler function

function handler (request, response) {…

- All node.js servers use a handler function, which is a new kind of event handler – for incoming requests to the server.
- A node.js handler function takes two object arguments
- The request object contains information about the http request.
- We use the response object to build our response.

Typical handler structure

```
var url = request.url;
```

- Get whatever data we need out of request object

Fill in the response header

```
response.writeHead(200, {"Content-Type": "text/html"});
```

- Builds an http response
- Head contains return code 200 (“Here’s what you wanted”)

Fill in the response body

```
response.write("<h1>Hello!</h1>");
response.write("<p>You asked for <code>" + url +
"</code></p>");
```

- The response object might contain HTML, Javascript, CSS or JSON, depending on what was requested
- In this case, we are constructing some HTML and putting it in the body

Sending the response

```
response.end();
```

- Calling response.end() tells node.js that we have finished filling in the response object, and it is OK to send the response back to the browser.
- Remember: “ending is sending” for these http responses.

createServer

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

- Calling function createServer from the http module
- The function createServer creates a server object
- It takes the handler function as input
- The handler function will be called when the server gets an http request
- It’s like a callback function!

listen

```
server.listen(8082);
```

- This starts the server and tells node.js, Unix and TCP that requests to port 8082 should go to my server
- I cannot emphasize too much that your server should listen to YOUR PORT NUMBER, not mine

Running and using the server

- On the server (Digital Ocean), run the simple server program:

  node simple.js

- From any browser, anywhere, request the URL
  http://138.68.25.50/anyPageNameYouLike
- Should get response:

  Hello! You asked for anyPageNameYouLike

Summary

- Typical overall handler structure
  1. Make a handler function
     a) In it, get data out of request object
     b) Then construct response header
     c) Then construct response body
     d) Call response.end() when response is completed
  2. Create a server object using the handler
  3. Start it listening to YOUR PORT