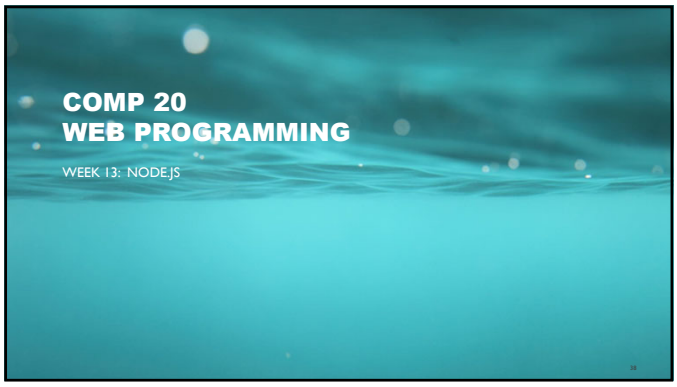

38

ANNOUNCEMENTS

Office Hours

Sunday 11am
Mon 7pm
Tues 3:30pm
Thurs 8pm

Plus: 8pm on Wednesday with Lisa

April 20

There WILL BE class on April 20 since we extended the spring break by one day

Assignments

Extra credit usability assignment due next Sunday

Node.js/MongoDb assignment due 4/23

Last Quiz

April 24-26

Topics: MongoDB, NoSQL, NodeJS, Heroku

39

FINAL PROJECT TIMELINE

- ~~Week 12 (4/5): finalize concept, set up dev environment, establish roles, project architected~~
- Week 13 (4/12): coding, "trial and error" testing
- Week 14 (4/19): MVP working / Lightening talks
- Week 15 (4/26): testing and enhancements

40

WORST AND … BEST

- http://libertyvan.com/
- http://cognosis.co.uk/

41

NODE.JS – WHAT IS IT?

- Free open source server environment
- Built to serve client requests
- Underlying language is JavaScript – which you already know.
- Asynchronous!
- Typically, a Node.JS file will consist of tasks that will be executed when someone accesses a certain port on a server

42

42

USE NODE.JS TO …

- Collect form data from user
- Handle client requests
- Interact with a database
- Interact with files on the server

43

43

NODE.JS VS PHP

- The big picture:
  - PHP handles one request at a time
  - Node.JS can handle multiple requests
- Node.JS is
  - Faster
  - Memory efficient
- Its not about picking one over the other – each has strengths for different purposes.

44

44

NODE.JS OVERVIEW

- Think of Node as a runtime engine to deploy server-side code
- Node.JS runs a script as a single thread but creates an environment that has the benefits of a multi-threaded system using callbacks
- This is especially significant when there is a resource that you may need to wait on.

45

45

OVERVIEW - CONTINUED

- Can run locally or deploy online via Git/Heroku
- Download at nodejs.org
- Run at the commandf line or load a .js file
- "Modules" allow for extended functionality
- Use npm (Node Package Manager) to manage/install modules

46

46

INSTALLING NODE.JS LOCALLY

- Download Node.js and npm (node package manager)
  - https://nodejs.org/en/download/
- Open a "command" window
- Use the command "node" to enter the node command interface
- The local Node environment is known as REPL : Read-Evaluate-Print-Loop
  - Useful for checking Node.js code

47

47

HELLO WORLD - SIMPLE

- Open a command window
- Type "node" to get into the Node.JS environment
- Enter the command:
  - console.log('Hello World");

- Other things to try:
  - .help
  - .load
  - .exit
- Use TAB key to see possible JavaScript commands

48

48

TRY IT

- Open a command line prompt and start the node terminal
- Try the following:
  - Display "Hi there" using console.log
  - Set a variable "x" to 30 and display x
  - Create a loop to display 10 asterisks on the screen

49

49

HELLO WORLD – USING A NODE.JS SERVER

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

- *In this case, activate the server by going to the URL: http://localhost:8080/*

50

50

BREAKING DOWN HELLO.JS

- `var http = require('http');`
- *http* is a pre-installed module
- You can install new modules or create your own
- There are a wealth of possible modules to be installed
- Example: install the email module
  - npm install nodemailer

51

51

THE HTTP MODULE

- The HTTP module helps you to create a server and then return a response when the server is accessed
- .**createServer**
  - Creates the server on the specified port

```
http.createServer(function (req, res) {
  //code for response goes here
}).listen(8080);
```

- **req** is request object coming in to the server
- **res** is the response object going out to the client

52

52

RESPONDING TO THE CLIENT

- `res.writeHead(200, {'Content-Type': 'text/html'});`
  - Writes the HTTP headers
- `res.write('Hello Strange World!');`
  - Writes to the page
- `res.end();`
  - Done creating page

53

53

TRY IT

- Create a server and display your name and the town you currently live in when the web server event is triggered

54

54

TRY IT: USING AN EXTERNAL JS FILE

- Copy the code from the previous slide into a file
- Name it tryit.js
- Open a command window and navigate to the relevant folder
- Run: "node tryit.js"
- Ctrl+C to stop

55

55

NODE.JS MODULES

- There are three types of modules – based on their location
  1. Core Modules
  2. Local Modules
  3. Third Party Modules

56

56

CORE MODULES

- To use a core module, you need to *require* it
  - `var m = require('the_module');`
- Common core modules
  - http creates Node.js http server.
  - url parses a url
  - querystring parses a query string
  - path deal with file paths
  - fs file I/O
  - util useful functions

57

57

HANDLING AN HTTP REQUEST

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  if (req.url == "/")
     res.write("This is the home page");
  else if (req.url == "/about")
     res.write ("This is the about page");
   else
     res.write ("Unknown page request");
  res.end();
}).listen(8080);
```

58

58

GETTING POST OR GET DATA SENT FROM THE CLIENT

- **For GET data use the URL module**
  - Gets the query string out of the url
  - var url = require('url');
- .parse() parses a url into parts
- .host protocol and server name/domain
- .pathname path within the url
- .search query string (including '?')
- .query returns query string as an object
- **For POST data, use the querystring module**

59

59

EXAMPLE: GET THE QUERY STRING

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var qobj = adr.parse(req.url, true).query;
  res.end(qobj);
}).listen(8080);
```

60

60

GET AN ELEMENT FROM THE QUERY STRING

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var qobj = url.parse(req.url, true).query;
  var txt = qobj.x;   // assume x is querystring parameter
  res.end(txt);
}).listen(8080);
```

61

61

TRY IT

- Create an html page with a form that asks for the user's name
- The form action should be set to your localhost server
- Read the name from the query string and display on the page

62

62

WORKING WITH FILES USING THE *FS* MODULE

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  file="hello.html";
  fs.readFile(file, function(err, txt) {
    res.writeHead(200, {'Content-Type':
'text/html'});
    res.write(txt);
    res.end();
  });
}).listen(8080);
```

63

63

LOCAL MODULES: GROW YOUR OWN USING *EXPORTS*

- Example: create a module to get the date/time

```
exports.dtModule = function () {
  return Date();
};
```

64

64

ADD THE MODULE TO YOUR SCRIPT

- *require* the module in the script
  - `var dt = require('./datemodule.js');`
- Use the module in the event function
  - `res.write("Current date/time: " + dt.dtModule());`

65

65

A MODULE CAN BE AN OBJECT

```
exports.myobj = {
  a: 1,
  b : function() { return 10;}
};
```

66

66

```
var http = require('http');
var obj = require('./myobj.js');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("a is " + obj.myobj.a + "<br>");
  res.write("b is " + obj.myobj.b() + "<br>");

  res.end();
}).listen(8080);
```

67

67

TRY IT

- Create a module to be serve the price of one of several items.
- Possible items are hotdogs ($3.50), fries ($2.00) and soda ($1.50)
- Create a .js file to load the module and read the value of a parameter called "item" from the query string and then display the price of that item

68

68