INTRODUCTION TO JAVASCRIPT

DAY 3: SERVER SIDE JAVASCRIPT & BOTS

ANY QUESTIONS?

DAY TWO QUIZ

- What is AJAX?
- What is JSON?
- What is the difference between JSON and a js Object?
- What jQuery functions do we use to make AJAX requests?
- What routing pattern uses resource-based URIs for server operations?

QUIZ CONT'D

- What HTTP verb is used to read data from the server?
- What HTTP verb is used to write data to the server?
- What are the different ways to authenticate to an API?
- Which authentication method is most common for application-level operations?
- Which authentication method is most common for user-level operations?

QUIZ, CONT'D

```
let planet = {name: "Jupiter"}
```

- What is this code doing?
- How would we add Jupiter's moons to this object
- What would we need to do to send this object as an API response?
- What code would we need to write to find out how many moons this planet has?

TODAY'S AGENDA

- Server-side JavaScript with Node
- Routing and Responding to HTTP requests
- Webhooks
- Spark API
- Spark Bots

POSTMAN API REVIEW

- Turn off cache-control and postman headers
- GETs
- Auth
- POSTs

NEW TOOLS

- ngrok
- Node.js
- NPM

WHAT IS NODE.JS?

Node is an "asynchronous event-driven JavaScript runtime".

- Runtime - a place that executes code. Chrome Inspector, REPL.it, Node.js
- Event-driven - Implemented as an event loop, it just waits for instructions to execute.
- Asynchronous - Executes code in a way that doesn't block the client or force it to wait.
- Built with HTTP in mind. Networking is a first-class citizen. Streaming and latency are core design concerns.
- Full ES6 compliance!

NODE ECOSYSTEM

- Node Core - The Runtime
- Modules/Packages - External Libraries
- NPM - Node Package Manager (think sublime package manager)
- Task Runners (gulp/grunt) - deploy and utility

HELLO NODE

First thing's first. Let's launch Node in REPL mode and play with it. Drop into a terminal/console and type `node`

```
//do some js!
let i = 0;
i++;
function square(x) {
  return x*x
}
square(15)

.exit //leave the REPL
```

RUNNING SCRIPTS WITH NODE

```
//my_script.js
function hello() {
  console.log('Hello World')
}
hello()
```

```
//terminal
node my_script.js
```

Just a script runner. Nothing scary.

NODE DEBUGGER

- `node debug program.js`
- `n` -> next instruction
- `repl` -> enter repl to check values/change values
- `s/o` -> step in/out
- `c` -> continue until breakpoint
- `sb(5)` -> set breakpoint at line 5
- responds to `debugger` in source
- `help` -> you guessed it

CREATING A WEB SERVER

Enough talk. Node is a server-side js runtime, so let's create a server.

```
//server.js
var http = require("http");

http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Hello World");
  response.end();
}).listen(8888);
```

```
node server.js
```

ROUTING

Routing in a web application is a system of mapping a request to a URL with an action that handles that request and provides a response.

A web application will have multiple endpoints, so we need a way to capture the requested endpoint and make a decision.

API CONSIDERATIONS

- RESTful Routes and HTTP Verbs
- Resource Management (data, files, other APIs)
- Code Organization
- Productivity
- Performance

All of this is a lot when you're doing it by hand. Gotta be a better way!

NPM

NPM - the Node Package Manager, is used to install and manage dependencies and libraries in your projects.

- npmjs.com - find packages
- Create NPM-aware projects with `npm init`
- install packages with `npm install package`

EXPRESS

Flexible, minimal Node.js web application framework

```
npm init
npm install express --save
touch index.js
```

```
const express = require('express')
const app = express()

app.get('/', function(req, res) {
  res.send('Hello Express!')
})

app.listen(8888, function() {
  console.log("Hello, Seattle, I'm listening")
})
// console
node index.js
```

EXPRESS MIDDLEWARE

Middleware is a function or functions inserted into Express to act on the request/response objects and provide extra functionality.

BODY PARSER MIDDLEWARE

```
app.post('/', function(req,res) {
  console.log(req.body);
  res.end("SUCCESS")
})
```

```
const bodyParser = require('body-parser')

app.use(bodyParser.json());
```

```
npm i body-parser
```

DESIGN AN API WITH POSTMAN

Postman is great for testing APIs, but we can also use it to help us drive the design of our own server code!

CONSUMING APIS IN NODE

- Not that different from what we've already done!
- Be mindful of the Request/Response lifecycle.

INTERESTING NODE MODULES

- Nodemailer
- Cheerio.js
- CSV
- google "node module *task*"

NODE API LAB

Create an Express server application with a route of GET /api. In your route handler, consume an API (eg, the Star Wars API) and output the data from the third-party API to the browser.

WEBHOOKS

Webhooks are endpoints that allow one system to notify another of something that happened. You can think of it like an "event api" or a "web callback".

Semantic differences aside, webhooks exist to receive notifications of events, e.g. a message was posted in a Spark channel.

CHAT BOTS

A chat bot is essentially an API that responds to events in a chat program. It consists of two parts:

- Webhook Endpoint to receive event notifications.
- Server Code to process event/respond

EXPLORING THE CISCO SPARK API

- Create and interact with rooms, search for people, post and retrieve messages.
- Use bot to interact with *other* APIs to provide rich mashup experiences.
- https://developer.ciscospark.com/index.html

HOSTING BOTS

ngrok is a tool that lets us tunnel a public request to our private machine. Great for development and testing.

For a real bot, you'd host it somewhere publicly accessible.

Note: If you restart ngrok you have to update the endpoint url. not persistent.

CREATING A SIMPLE CHAT BOT

- Register Bot (https://developer.ciscospark.com/add-bot.html)
- Add Bot to Room
- `./ngrok http 8888`
- Get Room Id, Set Up Webhook
- Update Bot Template
- Run it!
- When updating, stop the bot, not ngrok.

BOT ACCOUNT LIMITATIONS

- Bots can only "hear" messages that mention them.
- Bots are limited in some API functions (can't get room messages)

We can use a separate API token, and Sparky, to get around this.

BEYOND JUST BOT CODE

You can combine your bot with other API code (consuming and serving) to really make interesting things happen.

PROJECT

Upgrade your Chat Bot to respond to more than one command, and to interact with a third-party API. You can use APIs from exercises we've already done, like Star Wars or Github, or explore something new!

QUESTIONS?

OTHER BOT PLATFORMS