Node.js: Introduction

Computer Science and Engineering ■ The Ohio State University

Lecture JS1

Presented by: Ruksana Kabealo, Nick Sarkauskas, Jake Taylor, Patrick Travis, Caleb Woy, & Michael Zhan

A History of Node.js

- Created by Ryan Dahl in 2009
- Frustrated trying to update progress meter on a web page
- Created Node.js since web developers already knew JS
- Package manager, NPM, was introduced in 2010

Node vs PHP - Similarities

- Used on the backend to serve:
  - Static content
  - Dynamic web pages
  - Requests for data
- Used to run web sockets
- Can be run on Linux, MacOS, and Windows
- Open Source

Node vs PHP - Differences

- Node is a runtime environment and PHP is a scripting language
  - Node allows the use of JS beyond in browser
  - PHP needs to be interpreted by a web server
- JS can be used to develop desktop applications
  - E.g. Skype, Visual Studio Code, Slack
- Node is Asynchronous, Reactive and Non-blocking

Why Use Node?

- Using JavaScript across the stack
    - Allows for maximal reuse of developer resources
- Allows for thousands of concurrent connections on a single thread
    - Allows for less memory utilization
- Excels at real-time apps that don’t require intensive computing power

Node.js - Javascript on the Server

- Javascript code normally runs in the browser in what's called a **Javascript Engine**
- Many Javascript engines [1]:
  a. V8 - By Google for Chrome
  b. SpiderMonkey - By Mozilla for Firefox
  c. Chakra - By Microsoft for Edge & I.E.
- Node.js uses the V8 Javascript Engine.
- Modified for running on a server, not the browser

Node.js: Capabilities

- Almost Anything! Think of it like Ruby.
- One user claimed to have run with 1 million concurrent connections on a single server [2]
- Ruby Equivalents:
  - Rails -> Express.js
  - Shoes! -> Node-Qt
  - Mechanize/Nokogiri -> **Request.js/Cheerio.js**

Node.js: Web Scraping Example

Computer Science and Engineering ■ The Ohio State University

- Request.js/Cheerio.js
- Scrape Wikipedia for list of presidents' birthdays
- Pray to the Demo Gods

Do we get to mess with him now?
No. We wait until the worst possible moment.
CI/CD Demo
Rosdhub 2018

The Demo Gods

Node.js: Modules

- Modules - any file or folder which can be loaded using require()
  - Files: any .js file
  - Folders: any folder which contains an index.js file or a package.json file containing a “main” field

- Benefits of modules: (1) SPOC and (2) organization
- Some default modules: url, fs, assert

Node.js: Modules

- Versatile! Modules can hold functions, constants, etc…

- Full documentation of node module capabilities in the node.js api: https://nodejs.org/api/modules.html#modules_modules

- You can create your own modules! Two main parts:
  - Code must require the module
  - Module must export necessary information

Node.js: Modules Example - Using Functions

app.js:

```
function factorial (n) {
    var result = 1;
    for(var i = 1; i <= n; i++){ result *= i; }
    return result;
};

var n = 4;
console.log(`The factorial of ${n} is ${factorial(n)}`); // note the ``
```

Node.js: Modules Example - Using Modules

factorial.js:

```
exports.factorial = function (n) {
    var result = 1;

    for(var i = 1; i <= n; i++){ result *= i; }

    return result;
};

exports.msg = “Pascal was rad!!”;
```

Node.js: Modules Example - Using Modules

app.js:

```
var number = require('./factorial.js');                // require factorial module

var n = 4;

console.log(`The factorial of ${n} is ${number.factorial(n)}`);

console.log(`${number.msg}`);                // an additional secret message
```

Node.js: Package Management

- (NPM) Manages packages/modules. Comes with Node.js.

- Package - a directory containing:
    - one or more modules
    - a package.json file with metadata about the package

- Package vs. module: all modules are packages but not all packages are modules! (though many can be used as modules anyways)

Using NPM

- Install a package called “upper-case”
  - `npm install upper-case`
- Using a package inside `node`
  - `var uc = require('upper-case');`
  - `var x = uc(‘hello’);`
  - `console.log(x); //=> x = ‘HELLO’`

Reliable loading and Pathing

- Loading packages
  - By default upon install, package is placed in node_modules folder
  - Node checks:
    - core modules
    - node_modules
    - require.paths array
- Use module.paths string within package.json to specify locations of sub-modules

Using NPM for local hosting

- To install this package:
  - `npm install local-web-server`
- This package works with npm from the command line
- Navigate to the directory you want to serve
- Enter:
  - `ws`
  - `ws --https`

Most popular Node.js packages

- Express - Node.js server framework
- Body-parser - parses incoming bodies in middleware before handlers
- Async - utility module which provides powerful functions for working with asynchronous JavaScript

Asynchronous Node.js

- Designed so that I/O is asynchronous
    - Non-blocking, I/O distributed to other threads for faster execution
- Different ways to deal with asynchrony
    - Callbacks
    - Promises
    - Generators
    - Async /await

Summary

- Server-side JS thanks to the V8 Javascript Engine
- Open source offers perks
  - Many packages/modules means flexibility
- Asynchronous
  - Requires methods to deal with
  - Allows for faster feedback
- Slides, questions, demo code, mini-syllabus all available at: https://github.com/rkabealo/JS1-Tech-Team-Resources