8
JavaScript Introduction

Why are we studying JavaScript?

- Most widely deployed programming language in history.
- There is high demand for JavaScript developers.
- The language is very relevant in today's technology landscape.
- Uses many of the same programming paradigms we have already covered in Perl and Python - objects, hashes, functions, references, closures, etc.
- JavaScript, along with html and css, form the basis for the open web standards.
- Easy to get started, only need a web browser and a text editor.
- But not just for the web anymore - many JS-based IoT and robotics projects now such as Arduino, Tessel, and NodeBots.

History of JavaScript

- Created by Brendan Eich of Netscape fame.
- Mid 1990's - web pages were boring, no dynamic html, but did have Java applets and later, flash.
- Originally developed as a glue language for the web - adding dynamic functionality to static html.
- The Browser Wars of the late 1990's and early 2000's.
- Fast forward to 2005 - XMLHttpRequest, Gmail, and Ajax.
- The name given to JavaScript is often a source of confusion - it is not related to Java at all, even though both have C-like syntax structures (but so do other languages - C++, C#, Objective-C, etc).

JavaScript and the ECMA standards

- Ecma International - Standards organization of European origins, standardizes a lot of technology including programming languages, similar to ANSI/ISO.
- ECMAScript is basically the JavaScript standard - defines the minimum level of functionality every interpreter must implement to run the language.
- Microsoft and Adobe had their own implementations called JScript and ActionScript, the the idea was to standardize the language specification.
- Several versions since 1995, current one is called ES6.
- Source of much confusion for those new to the language.
- A somewhat humorous take on the language by one of its earliest proponents - http://www.crockford.com/javascript/javascript.html

JavaScript’s built-in types

- JavaScript is a loosely typed language, not un-typed.
- Numbers - only one number type (basically a double-precision float)
- Strings - single and double-quote means the same thing - no differences
- Booleans - true/false, the Boolean(value) function returns truthy or falsy
- null - a value that isn’t anything
- undefined - default value for uninitialized variables, missing object members
- NaN - not a number - the result of an erroneous arithmetic operation
- Objects - Everything else is an object!

JavaScript Scope

- JavaScript's Global Namespace - The default. If you declare a variable in a script that is not inside of some other scope it is in the global namespace.
- Name collisions - using the same name that was already used elsewhere.
- Local scope - variables declared inside of a function are in local scope. Now the same names can be used in different functions to refer to different things.
- There is no block-level scope in JavaScript, only function scope.
- Scope also affects object lifetime - global variables hang around for as long as the app is running, local variables die when the function they are bound to goes out of scope.
- Immediately Invoked Function Expression (IIFE).

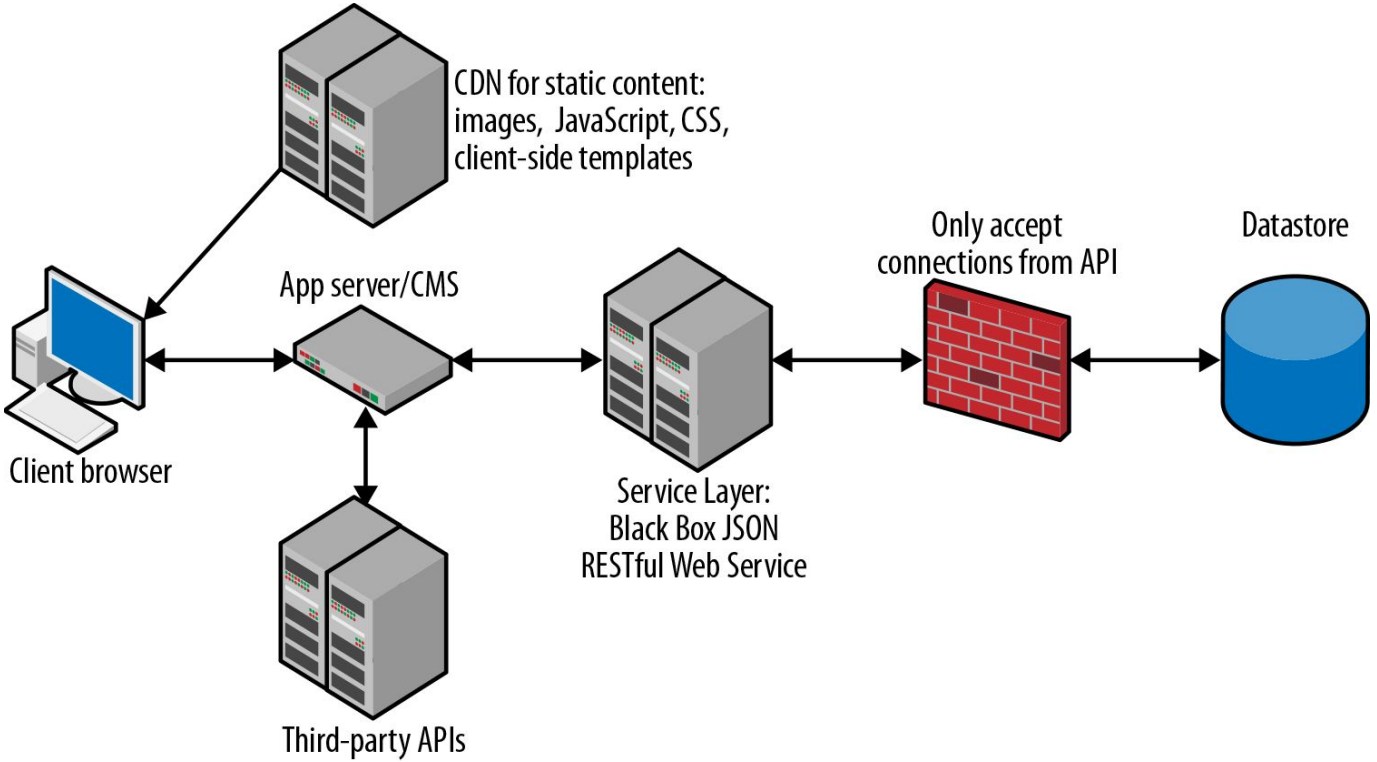
JavaScript Prototypes and Inheritance

- JavaScript does not have classes like Perl and Python.
- Instead JavaScript objects inherit directly from other objects.
- JavaScript prototypes - The parent object from which a child object inherited is called that object’s prototype.
- Object has a “pointer” to it’s parent object.
- When JavaScript encounters a situation where an object does not have a referenced member, it will go up the inheritance chain to look for it.
- Essentially in JavaScript, a “class” is a function object that serves as a constructor, along with an attached prototype object.

Web 101 refresher

- WWW - A way to link documents using internet technologies.
- Client-Server nature of the web - request/response model.
- The 3 legged stool of the web - HTTP, HTML, Web Browser.
- The great thing about web apps is that no matter what language, framework or operating system you use to develop them, all web applications work with HTML, CSS, and JavaScript over HTTP.
- How we got to this point is a convoluted mess but it is what we have to work with, and really not as terrible as it seems.

Web Applications



Web Applications

- HTML, CSS, and JavaScript - these are what the web browser understands and can work with.
- But there are many ways for programmers to get the browser what it needs.
- CSS Preprocessors - higher level language that “compiles” to CSS.
- Languages like CoffeeScript, Elm, TypeScript, Fable, and Kotlin also “compile” or more correctly transpile to JavaScript.
- Some frameworks like JSP, ASP.NET generate all the HTML, CSS, and JavaScript on the server and then send the whole canned package down to the browser.
- Other frameworks like Angular make use of Ajax calls and html templates.

JavaScript and the web browser

- The DOM - Document Object Model - a data structure (tree) consisting of the guts of a web page. DOM manipulation, DOM traversal.
- The Window Object - The container object in which your JS code lives inside the browser, window is the name of the global object in the browser.
- Events - Hooks that you can use to customize your application's behavior.
- jQuery - JavaScript helper library for doing many common tasks efficiently.
- Angular - Revolutionary client side JS framework released in 2013.
- React/Redux - Latest client side JS framework that is all the rage now.
- Polyfills - Extra code needed to fill-in for a missing browser feature.
- Transpiling - Converting to a version of JS that the browser understands.

JavaScript on the web server

- Node.js is a scalable, asynchronous, event driven, non-blocking JavaScript runtime for building network applications.
- The “N” in the MEAN stack - MongoDB, Express, Angular, and Node.
- Acts as the web server part of an application.
- Node runs on Google’s V8 JavaScript engine.
- Event driven and asynchronous - these translate into high performance and high scalability.
- CommonJS - effort to make JavaScript break free of the browser by adding APIs that make it easier to build server apps, command line tools, desktop GUI apps, etc.

NPM

- Node Package Manager - the default package manager for node.js.
- Comes with node and is installed at the same time.
- Serves similar purpose to other package managers such as Pear (PHP), CPAN (Perl), Bundler/RubyGems (Ruby), and NuGet (.NET).
- That purpose is to manage packages of code and their dependencies and versions.
- NPM has a CLI (command line interface) that interacts with a remote code repository to download packages.
- You specify packages for your project in a packages.json file.
- Yarn is an up and coming alternative to NPM created by Facebook.

JSON

- JSON - JavaScript Object Notation, ok to pronounce it like the name “Jason”.

- Easy to learn because it is the same exact format used to define objects in JavaScript code.
- JSON is pervasive in technology stacks like MEAN because the data that forms the content of your application exists in the database (Mongo) in the exact same format as it does in the front-end of your app in the browser (Angular). This means there is zero friction when it comes to translating the stored form of your data to its displayed form (no entities, normalization, etc).
- Came about as a lighter weight data format compared to XML.

Security Concerns

- XSS - Cross-site Scripting - a type of attack that exploits the same origin policy to escalate permissions of malicious code in a web browser.

- CSRF/XSRF - Cross-site Request Forgery - a type of attack that exploits the fact that websites only authenticate requests came from a user's browser, not from the user herself. Can be mitigated by using authentication tokens.

- CORS - Cross-origin Resource Sharing - a type of vulnerability that can exist by relaxing the rules of the same origin policy to allow XHR (Ajax) requests to domains or ports other than the page from which they originated.

Some other JavaScript odds and ends

- Minification - The process of “compressing” a script file by removing all whitespace and minimizing the length of variables with the goal of reducing the size in bytes of the source code text file to a minimum so it can be downloaded faster, making the site load faster so users don’t abandon it.
- Linting - The process of static code analysis to improve code quality, maintainability and even catch bugs due to improper coding techniques.
- CDNs - Content Delivery Networks - network of proxy servers that serve up cached static content (images, common open source code like Angular, Bootstrap). Goal is improved performance and reduced load on the application host.

Term Project

- For our term project we will build a web application based on node.js
- We will attempt to use Heroku to host our app.
- More details to follow next class period.
- You will have approx 1 month to complete it and it will be worth 20% of your final grade.
- I am open to any student ideas of what kind of app you want build! But we all still have to build the same app (so I can make a fair grading rubric).

What is Heroku?

- Cloud-based application host - Can host node, java, ruby on rails, php, etc.
- Easy to use because all you have to do to publish your app after setup is run “git push heroku master”.
- Suggest you spend some time here to familiarize yourself with the basics - https://www.heroku.com/nodejs
- Please sign up for a free heroku account, if you do not already have one.
- Also download and install the Heroku CLI from https://devcenter.heroku.com/articles/heroku-cli

Online Textbook

- We will use the material from this online book as our textbook for JavaScript.
- http://chimera.labs.oreilly.com/books/1234000000262
- Read Chapters 1 and 2 for next time.
- We will try to cover the material in the book through chapter 5.
- Another great resource is the You Don't Know JS series - https://github.com/getify/You-Dont-Know-JS
- You can download the whole book in PDF from https://www.gitbook.com/book/maximdenisov/you-don-t-know-js/details

Additional Resources

- JavaScript Wikipedia page - https://en.wikipedia.org/wiki/JavaScript
- Mozilla's JS Guide - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- Understanding Scope in JavaScript - https://scotch.io/tutorials/understanding-scope-in-javascript
- Understanding Prototypes in JavaScript - http://yehudakatz.com/2011/08/12/understanding-prototypes-in-javascript/
- Douglas Crockford's videos - https://www.youtube.com/watch?v=v2ifWcnQs6M