Recap + Additional Research Topics

CS-554 – WEB PROGRAMMING

What did we actually achieve here?

Lectures 1– 3: More Efficient Development

There many build tools we can use to make our life easier. There are a number of uses for these tools:

- Turning precompiled languages into proper languages, such as SASS to CSS
- Writing new age code to be compiled to supported code
- Optimizing assets to improve user experience

Lecture 4-5: Components. Everywhere.

The component design pattern allows us to focus on reusability and individualizing components for specific use cases.

We can use container components to handle states, and all things related to the data of the actual component.

We can then use stateless functional components to only handle view related logic.

Lecture 10: We can have an app, without a server

We can develop single page applications in order to have all the logic that used to require a webserver, without the webserver.

Instead, we can have a single page app for highly responsive user experiences, as well as for totally splitting up view code and server code.

Your single page applications will simply need to access an API.

Lecture 6: Speed is of the essence

Speed is one of the most important factors in a web application.

We can use technologies, such as Redis, to really aid in this.

- Redis is super fast
- Caching makes your app appaear to be super fast

Lecture 7: Concurrent Development

We can run many, many processes at once that all interact together through some form of inter process communication.

Our web server should never perform any hard logic, but instead should only act as an access point to our data, and to inform our workers as to what they need to do next.

Lecture 8: Sockets are fast, but REST is clean

There are many ways to communicate data back and forth between servers and clients, and web sockets allow for extremely quick communication.

Unfortunately, web sockets are essentially structure less, so you have to implement error handling and such from the ground up.

Sometimes, REST is useful for its high degree of structure. It’s perfectly normal and acceptable to have a REST API and use sockets, as well.

Lecture 9: You Don’t Need Node

Even if your web server is in one language, such as Node, you don’t need to be confined to Node.

Interprocess communication allows for you to start developing your applications in any number of languages, with various different technologies.

Lecture 11: “Why do we trust the internet with anything”?

There are a grotesque number of security vulnerabilities on the internet, and it’s really amazing that we actually use it for any sensitive information.

Security is a sliding scale. We can make things more secure, but never 100% secure.

Lecture 12-13: Web Development, Without the Web

We can use each lesson from web development to create other applications, that are not necessarily web based.

HTML, CSS, and JS allow us to create robust user interface based applications.

Many technologies allow us to use versions of browsers that have hooks into native functionality, to create desktop or mobile applications that are developed in browser application technologies.

What’s the rest?

What else should we learn?

You should learn every technology under the sun that interests you even a little bit.

There are **thousands** of web development technologies that we can use, in various different programming languages.

Content Management Systems

Many times, you will end up using some Content Management System as a solution to your problems.

A Content Management System is a software that will allow you to create applications based on using the application and placing a database into a particular state, rather than hard coding data.

Many web apps you build are technically content management systems.

Common Content Manage Systems:

- Wordpress
- Media Wiki
- Moodle
- Ghost

Other Languages

There are many languages you will need in Web Development. Most languages have some form of web application capabilities.

Learning multiple languages is an extreme benefit to a programmer.

Frameworks

Frameworks help in the development of applications. They provide sets of opinions to follow, many tools to use, and a tested architecture to follow.

Learning a few frameworks helps you learn the gist of any framework. Frameworks help you develop large applications, faster, and without rewriting everything from the ground up each time.

PHP

PHP is a scripting language that is used primarily for web applications. Many common web application platforms are built in PHP.

One of the most common web technologies, **Wordpress**, is written in PHP. You can make an entire job from developing for **Wordpress**.

Laravel is a PHP web framework with a strong first party eco-system and features that are meant to be intuitive for developers.

Ruby

Ruby is another programming that many web developers use, due to its eloquent syntax and high amount of third part created plugins (known as gems) that allow for focusing on gluing your application together.

Ruby has a very popular framework, known as **Ruby on Rails**, that allow for rapid application development, app scaffolding, and automatic database migrations.

C#

C# is a programming language developed by Microsoft that is used commonly in industrial settings. It is effectively replacing Java due to its syntactical differences and relatively eloquent language, as well as high performance without a Virtual Machine.

In **C#**, there is a popular and commonly used framework known as **ASP.Net** that allows for rapid application development for applications deployed on IIS, which works on Windows machines by utilizing the .Net Framework.

Lately, Microsoft has been developing a cross-platform environment to write C# in, utilizing native system features in many languages, known as **.Net Core**; this can run on OSX, Ubuntu, and Windows, and is proving to be a fundamental paradigm shift.