WDD 330 Syllabus

Description

This course is designed to give students the skills required to create web applications using HTML, CSS , and JavaScript. It is intended to help the student learn to do this without the aid of third-party frameworks or libraries. Because of this, the course focuses on how to solve larger, ill-structured business problems by designing and creating web applications.

Learning Model Architecture

Prepare:

Each week you will work directly with your team. You help your team become successful if you come prepared for these team meetings. This preparation includes thinking about the technologies you are exploring, completing the assignments given during the previous team meeting, and doing the necessary research to plan the next steps for your team. Just as in life, if you do more than fulfill the minimum requirements assigned to you, you will be more successful in your team and in the class. Magnify your professional calling.

Teach One Another:

Team meetings and a collaborative environment have been established to enable each of you to draw from the strengths of others so that your weaknesses can become strengths. This requires effort on the part of both the knower and the learner so that 'both [may be] edified' (see Doctrine and Covenants 50:22).

Ponder/Prove

Pondering is integral to success in both this course and life. You should be pondering and reflecting not only on what you are learning but how you are learning it. Self-understanding is vital to professional life. It is also fundamental to our eternal exaltation.

Proving is also integral to success. You should prove what you think you know through experimentation. Just finding an example of some principle

on the web or from your team is insufficient for successful learning. You must be playing with and generating your own examples to understand how technologies work and what they do. This way you become a blessing to yourself and your current and future team members.

Standards

Be responsible for your own education. It is important that you prepare yourself each week to contribute actively in learning with your fellow students. Be respectful of each other's time and be prompt to any meetings that may be scheduled.

NODE and NPM

Node is a JavaScript runtime that has gained a lot of popularity in the last few years. There are many helpful tools and utilities that can be run using it. NPM is the Node Package Manager. It gets installed automatically with Node and is used to add and remove utilities.

Lesson 1: Software Installation

This semester we will be using several tools as we learn more about HTML and CSS. Below is the list of things you will need to install and setup. Even though we will not be using many of these for several weeks I highly recommend getting them all installed and verified as working this week. You will have a much better semester if you do.

NODE and NPM

Node is a Javascript runtime that has gained a lot of popularity in the last few years. There are many helpful tools and utilities that can be run using it. NPM is the Node Package Manager. It gets installed automatically with Node and is used to add and remove utilities. (For more detailed instructions

visit: http://blog.teamtreehouse.com/install-node-js-npm-windows)

BEFORE following the instructions below make sure that you do not already have Node installed. Open up a terminal (Mac) or CMD prompt/powershell (Windows) window and type `node -v`. If it gives you a version and NOT an error you already have Node.

Windows Users:

1. **Download the Windows installer from the Node.js® web site. You will want to grab the Recommended version (LTS) instead of the latest version!**
2. **Run the installer (the .msi file you downloaded in the previous step.)**
3. **Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).**
4. **Restart your computer. You won't be able to run Node.js® until you restart your computer.**
5. **To verify that it worked type the following in the command prompt: `node -v`. If you get the version number back you are good to go.**

Mac Users:

For the mac we are going to install the Node Version Manager (nvm) tool first, and then we will use that to install Node. There are a couple of advantages to doing it this way...the biggest of which is that it makes it really easy to switch versions of Node when needed (happens more than you might think as a developer) Before we can install that however, we need the Apple developer tools. You can get those in one of two ways:

1. **If you think you will ever want to build native MacOs or IOS apps download and install XCode. Apple's XCode development software is used to build Mac and iOS apps, but it also includes the tools you need to compile software for use on your Mac. XCode is free and you can find it in the Apple App Store but it is quite large! After you install it, open it at least once to accept the Terms. The next step won't work unless you do.**

2. ***Recommended for most! If you don't want to download and install the full XCode (It's quite large), you can install just the developer tools by following these instructions.**

Installing Node and NPM with nvm is pretty straightforward.

1. **To install nvm, open the Terminal app and type:**
   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
   ```
2. **Once nvm is done installing you can grab the latest copy of the Long Term Support (LTS) version by typing the following:**
   ```
   nvm install --lts
   ```
3. **To verify that it worked type the following in the Terminal: `node -v`. If you get the version number back you are good to go.**

Since macOS 10.15, the default shell is zsh and nvm will look for .zshrc to update, none is installed by default. If you get an error installing above, create this file by typing the following in your terminal:

```
touch ~/.zshrc
```

SASS

We will be learning to use the SASS CSS Preprocessor this semester. SASS must be compiled before sending it to the browser. We will be using a Node.js Sass compiler. Becasue of this make sure to complete the Node installation instructions above before continuing.

- **Open your Terminal or Command Prompt. On the Mac the Terminal.app comes installed by default. It's located in your Utilities folder. On Windows, run `cmd`.**
- **Install Sass. In your open terminal window type: `npm install -g sass` This will install Sass and any dependencies for you.**
- **Double-check. You should now have Sass installed, but it never hurts to double-check. In your terminal application you can type: sass --version**
  It should return something like: `1.14.1 compiled with dart2js 2.0.0`. Congratulations! You've successfully installed Sass.

Sass actually has two versions of it's syntax. The original syntax is called just Sass and uses a `.sass` extention on your files. The other syntax which has become more popular and which we will be using is the SCSS syntax with a `.scss` file extention. The SCSS syntax is still considered Sass...and the compiler we just installed will work for either. The file extension will tell it what it needs to know.

StyleLint

CSS Linters point out problems with your CSS code. They do basic syntax checking as well as applying a set of rules to the code that look for problematic patterns or signs of inefficiency. The rules are all configurable, so you can easily write your own or omit ones you don't want.

- **Open your command line (Terminal on the mac, Run->cmd on Windows)**
- **Enter the following:**

```
npm install -g stylelint stylelint-config-standard
```

Analyze-css

Like CSS Lint, analyze-css inspects your CSS for complexity and performance.

- **Open your command line (Terminal on the mac, Run->cmd on Windows)**
- **Enter the following:**

```
npm install -g analyze-css
```

UnCSS

UnCSS looks at your HTML and CSS files and removes unused CSS! This is great especially if you choose to use 3rd party libraries like Bootstrap.

- **Open your command line (Terminal on the mac, Run->cmd on Windows).**
- **Enter the following:**

```
npm install -g uncss
```

MANAGING THE MODERN FRONTEND WORKFLOW

Development workflow has become quite complicated for Web development. Let's take a medium-sized project for example. It could have dozens of JavaScript files, several CSS files, 3rd party libraries, and who knows how many icons, fonts, images, etc. involved. It might be using a CSS preprocessor like SASS or it could be transpiling the JavaScript to make sure that new features will work in older browsers. It's no wonder that developers have produced tools to help manage it. This activity will introduce a simple implementation of some of these tools. The tools fall into three categories:

Package managers: These keep track of all of the external dependencies for our app. This includes development tools and libraries we might be using. It not only knows which packages to download, but it tracks versions as well. We are using **npm** for our package manager.

Bundlers: Bundlers handle the compiling, transpiling, concatenating, minifying, and moving around of assets in our project. We are using **Snowpack** as our bundler. Other common bundlers are Parcel and Webpack.

Task managers: These keep track of what needs to be done and when. There will generally be scripts defined in the task manager for each phase of development. Our project is fairly simple so we are just using **npm** again for task manager. Other common managers are **Grunt** or **Gulp**.

We will be using an NPM/Node-based workflow for this course this semester. It is important that you understand how these tools work together.

PREPARATION

Do not treat these activities as a task to check off in as little time as possible. Instead, always treat them as opportunities to learn.

Some activities and discussions will be designed to move beyond the readings...give examples and use cases, talk about potential problems, etc. Because of this, it is extremely important that you ask for clarification on

parts of the readings that didn't make sense. Your instructor is happy to talk about them...but will assume everything is good unless you ask.

Tips

- Read for understanding, not completion.
- Don't skip the code examples and exercises! Make sure to spend time reviewing these.
- Take, good notes of questions that arise in your mind as you read.
- If there's a chapter in the book that's not assigned, it is not because those topics are not important...but often because you are expected to already understand those concepts from a previous course. You may want to read some of those unassigned chapters if the topics look unfamiliar to you.
- There is not a specific list of exercises given because not everyone needs the same thing.

QUESTION

What should I do if none of my fellow mate is responding to my email or message sending to them ?