Start Here - Overview

Course Overview

Organization

This course is divided into three main sections which are largely addressed sequentially:

**Layout and Styling**

The first portion of the class focus on the static layout and styling of a web page. For some this may be review if you have done web publishing in the past. There is quite a bit of information to take in here but the problems to solve are not that intricate.

**Client Side Interaction**

The second portion of the class focuses on JavaScript and making interactive web pages in the browser. Things like forms that will display an error message if a password is too short or creating drop down menus are things that will be covered in this portion of the class.

**Server Side Interaction**

In this portion of the class we look at using a very simple database to store data between website visits. The technologies we will be using this term are Node.JS and MySQL. In addition we look at how we can track a user and data from page to page which is a critical first step in designing more complex systems like shopping carts for an on-line shopping web site.

Student Expectations

Prior Knowledge

Before starting work in this class students should know the fundamentals of object oriented programming. The following is a list of topic that you should be very comfortable with:

**Basic data types**

[This list (Links to an external site.)](http://msdn.microsoft.com/en-us/library/cc953fe1.aspx) of data types should be familiar or at least understandable after reading the type description.

**Common object types**

You should know what a string is. You should know if there is a difference between a string and an array. You should know the difference between a string in C and a string object in C++

**Complex data types**

You should know what an array is, what a struct is and what, if any, differences there are between them.

**Flow control primitives**

You must be very comfortable with for, while, if/else, switch and do/while. You should know how they work, you should know what i is equal to at the completion of a loop if the condition says for(i=0, i<5, i++), is it 5 or is it 6?

**Scope**

No matter where I declare int foo in your code, you should be able to figure out if any other arbitrary spot in your code can access that variable. You should know what public and private functions are and what happens (and what it means) when variables go out of scope.

**Problem solving**

If a language does not provide you a tool to do something, but you need that tool to make progress, you should be comfortable making it yourself. You should be comfortable making helper functions, even when the requirements don't call for it.

**Organization and documentation**

You should know that all of your source code probably does not belong in a single file and that every public function should have comments.

**Things not on this list**

This is not a comprehensive list, but it is a good start. If you don't know any of these topics, consider reviewing them before the class begins.

Code Quality

Code quality will be discussed on a section by section basis. However the general rule is that all code should by syntactically valid, consistently formatted in a readable way and produce zero errors, warnings or notices unless OKed before submission.