**Introduction to CS321 Technologies**

In classic software development people often speak of the MVC of Model-View-Controller tiers of the software. The Controller, not surprisingly, controls what happens; the View is what is presented to the user, and the Model is the underlying structure of the data. The idea behind an MVC approach is to separate these three parts and de-couple them so as to make the whole software more robust. For example, you can change the View (change the way data presented from a table of numbers to a graph) without changing the Controller or the Model.

Obviously, from the course title, in this course we will focus on programming for the web. Here ' the web' means everything from simple web pages and apps to more complex sites such as an intranet (web site specific to one company.)

In design for the web, you will sometimes see people write about the MVC approach, but it is more common to talk about **Structure, Presentation, and Behavior**.

Current ideal practice for web site design identifies three facets of a web page, designed with three different technologies (Here you can think of the three technologies as three languages):

* **Structure** (and content) **– implemented in HTML5 (aka the Model)**
* **Presentation – implemented with CSS (aka the View)**
* **Behavior** (or Function or Action**) – implemented with JavaScript, and its jQuery library (aka the Controller)**

In practice, there is some cross-over among these three parts of a web page, (and some libraries handle more than one facet) but this is still a good way to approach site design. So, here is a slightly more extensive explanation:

* **Structure** (and content) **– implemented in HTML5**This is like the outline for a book or a paper – you identify the major chunks in your book (Title page, Table of Contents, Chapter1, etc.) and you write the content.

On a web page, the structural elements include paragraphs, images, tables, lists.
You can think of this as the skeleton for your page; everything else hangs on it.

These structural elements are all named in **tags**. For example, a paragraph begins with <p> and ends with </p>. Everything between those two tags is the content of the paragraph. If your page includes the HTML5 **<p>OK, I get it.</p>** then it will produce a paragraph with one sentence in it:

 OK, I get it.

I will sometimes write ‘HTML’ for ‘HTML5’, but everything in this course will be HTML5.

Defining a clear structure for the pages on your site is a very important for accessibility: screen readers (for vision impaired users) depend on the structure your tags provide.

* **Presentation – implemented with CSS**Presentation refers to how your page looks.: which paragraphs are in which font, which words are in bold-face, where are images placed on the page, etc.

Once upon a time (5 years ago) it was considered good practice to separate the presentation and structure, but now it has become imperative.

 Why? Because there are so many hand-held devices (smart phones, tablets, etc.) and they and desk-top monitors have such different screen sizes that we need to be able to have different presentations for different ranges of screen sizes. The mobile web has changed the way we think about site and page design.

When we design a page and think about presentation we also need to think about accessibility. How does your page appear to a person with poor vision? (Such a user will typically have over-ridden your font sizes so that all words are enlarged.) What about a user who is color-blind?

Those of you who are interested in graphic design will be particularly intrigued with CSS. We will learn enough CSS to get you started, and for you to learn more intricacies on your own. Of course, there are courses in the Comm Department which will help you become an accomplished graphic designer.

**Do:** Look at [www.csszengarden.com](http://www.csszengarden.com) All the pages there have the same HTML content, but they have very different presentations. (The Safari book site at myacm.acm.org also has a book about this site.) This site demonstrates the power of presentation.
* **Behavior – implemented with JavaScript, and its jQuery library**

So far (aside from links) our pages are pretty, but static. They sit there and do nothing.
If we want to get any input from the user (What city do you want the weather for? Where do you want to fly?) we need to have a language which can understand the user’s responses and capture them (that is, it needs to be able to store the user’s answers in variables), and then take some action based on those answers.

On the web, the language is JavaScript (which I will often refer to as JS). Over 80% of all web sites also use a very powerful library of JavaScript functions, known as jQuery.

 In 2015 there was a massive extension of JavaScript known formally as ECMAScript2015 or ES6. ("Classic" JavaScript was ES5.) ES6 was followed fairly quickly by ES7, ES8, etc. I will refer to all of these as ES6 or ES6+ or ESNext.

**Is there more? Of course! There are:**

* **Connections- and the back end (writing servers in JS with Node.js)**Very often how a web site behaves is determined by programs on the server.
The user (client) makes a request of the server. The program on the server may gather some data, or look things up in a database, etc. and then the response is sent back to the user. This dialog between the user and the server may continue for several steps.
In terms of the MVC approach, part of the Controller is on the server or back end.
We will work with Node.js to design such server functionality.
* **Being responsive/ Going mobile**As indicated above, today people work on devices with a huge range of sizes. Responsive web design is about ensuring that our sites work well on all these sizes.
* **The full stack**This refers to the ability to work with all the parts of web site development. We will work with the MEAN stack – where MEAN stands for MongoDB, Express, Angular and Node.js --- it is understood that the MEAN stack includes HTML5, CSS and JavaScript/ESNext.
* **Interactions:**Of course our 3 technologies – HTML5, CSS3 and JavaScript – all work together. For example, even a static web page uses HTML and CSS.

Beyond that, when part of a webpage changes without reloading the page (e.g., in google maps when you slide part of the map, or when you expand or contract a menu) then you are using AJAX. AJAX is a way to use HTML5 (to design the content of the basic page), CSS (to determine what is shown) and JavaScript (to see respond to the user’s cursor movements or clicks.) Often you are also communicating with the web server to retrieve information that you didn’t need when you displayed the original page.

Other times, you will fill out a form on a page, and the web server will respond with a new completely different page. So, we will need to communicate with servers and we will also sometimes need to have the server design and send an appropriate new page. We use **Node.js** on the server, but there are other possible choices. (PHP, python, and Java using the Apache web server are common.)
* **Which brings us to HTML5 (and XHTML)**:
XHTML is very much like HTML, except you may define new tags or new structural elements. So, instead of being limited to a few tags like <p> for paragraph, you might define a new tag <xray> or <chemical>.

XHTML is used for describing data, and the new tags have meaning attached to them – so we speak of the **semantic web**. (Semantics refers to meaning; syntax refers to grammar – e.g., those annoying semi-colons at the end of a line.)

When you use an RSS feed you are reading XHTML data.
During the course we will explain how HTML and XHTML are related.

At one point it looked as though XHTML would be the next-generation of HTML after HTML4 (also called HTML), but HTML5, with its support for embedding videos and other fun stuff is much more popular than XHTML. We will use HTML5, but you will be able to write XHTML also with a little extra work. Today, XHTML is used mainly in certain industries and applications; while most web pages are written for HTML5, there are still about a quarter of all pages in XHTML, so you may be called upon to maintain XHTML.

* **Going mobile**Luke Wroblewski refers to the “mobile moment” for a type of application – which is when more than half the interactions are on mobile devices. We have already passed the mobile moment for many common applications, including on-line banking, amazon.com orders, and searching. So, as we design web sites we need to think carefully about how those sites appear on mobile devices. The book “Mobile First” will help us in this effort.

**Will I learn everything there is to know about web development?**Absolutely not! First, there are a number of different frameworks (libraries) that people use to make web development easier. We will work with jQuery, but some people prefer React for the same purpose. Similarly, there are many alternatives to Express and Angular, not to mention different approaches to managing CSS.

What you will learn is the foundational pieces for web development and how to pick up a new technology as the need occurs.

*Throughout this course we will use only professional level resources.* In the last project you will be required to learn three new technologies on your own. You will see that, as the semester progresses, you will depend less and less on my notes, and more and more on these professional level resources.

**An aside about file extensions**

**HTML** files (and XHTML files) have either **.htm** or **.html** extensions. It doesn’t matter which, but if the file has been saved with one of them then any reference to the file must use the same extension. I suggest **.html**, which is much more common. When you reference a file you must have the correct file extension – that is, .htm and .html are not interchangeable.

**CSS** files have **.css** extensions and **JavaScript** files have .**js** extensions.

Image files which are made up of flat broad areas of color have the .**gif** extension.
Photographic files have either a .**jpg** or .**jpeg** extension. As with html files, the two extensions are not interchangeable.