CSE312 - Web Dev

This is a brand new course

About the Course

- We will build full-stack web applications and explore how they work
- We'll start by using existing [back end] frameworks and servers
- Remove frameworks later and build an app from scratch
- Focus on adding features to our apps and understanding how they work

About the Course

- We will only spend one day (today) on front end specific topics

- We will not cover front end frameworks (Vue, Angular, React, etc)

  - You are encouraged to study these on your own

- Very little focus on UX

Development

- Use any IDE or text editor you prefer
- We will use multiple languages throughout the course
  - Javascript
  - Python
  - Scala? Too soon?
  - Your choice of language after lecture 7

Roadmap for Today

- HTML
  - The foundation of any web page
- CSS
  - Adding style to the page
- JavaScript
  - Making the page dynamic
- Bootstrap?
  - Library to make the page look better with little effort

HTML

HTML

Hyper **T**ext **M**arkup **L**anguage

Hyper Text: Text that can contain links to other resources

Markup Language: Special markers add information to the text that is not displayed. In HTML we use tags that tell the browser how to display the text

HTML is not a programming language

HTML

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
</body>
</html>
```

Save this in a file with a .html extension and open it in a web browser to see the web page below

HTML - Elements

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
</body>
</html>
```

HTML uses angle brackets to define elements

Each element has an open tag <h1> and close tag </h1>

Everything between the open and close tag is the content of that element

HTML - Elements

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
</body>
</html>
```

head - Content that does not appear on the page

title - The text that appears on the browser tab

body - Content that appears on the page

h1 - Header 1 (can use h1 - h6 where h1 is the largest header)

p - Paragraph of text

div - A division. Typically used as a featureless container

HTML - Elements

```
<br/>
```

Some elements are self-closing

```
<hr/>
```

The open and close tags are contained in one tag

br - Line break

hr - horizontal rule (line)

HTML - Attributes

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
</body>
</html>
```

Elements can contain properties which are defined in the open tag of the element

These Attributes are key-value pairs

We have an empty division with a property named id with a value of “myDiv”

HTML - Comments

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <!--<p>My content</p>-->
    <div id="myDiv"></div>
</body>
</html>
```

HTML only supports block comments

<!-- Starts a comment

--> ends a comment

HTML - Unicode

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>&#9820;   Web Page</h1>
    <!--<p>My content</p>-->
    <div id="myDiv"></div>
</body>
</html>
```

Insert any unicode character with the syntax &#<unicode_value>;

♜ == ♜

Some common character have names

→ == →

Extra white space is ignored in HTML. Add extra spaces with   (non-breaking space)

HTML

And that's HTML.

We'll use many more elements and attributes as they come up

I'll assume that you can pick up HTML very quickly and understand new elements with little effort

I recommend W3 Schools to explore all the elements/attributes available to you

https://www.w3schools.com/html/default.asp

CSS

CSS

Used to add style to HTML

Keep CSS in a separate file

HTML should only be concerned with raw content and basic organization

CSS is concerned with how that content is styles

CSS

Note: You will not be graded on the quality of your CSS in this class

We'll cover the absolute basics so you are aware of CSS

You are encouraged to explore it further if you want to make your sites look good

CSS

```css
body {
    background: aqua;
}

p {
    color: #ff0000;
    font-size: 50px;
}
```

First Web Page

My content

We "import" our CSS file from the head of our HTML file using the link element

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
</body>
</html>
```

CSS

```
body {
    background: aqua;
}

p {
    color: #ff0000;
    font-size: 50px;
}
```

CSS allows us to edit to look of each element

Hear we change the background color to "aqua" a names color

We change the text of each paragraph to red using its RGB value and the font size to 50 pixels

Many more properties that can be set

CSS

```css
p {
    color: #ff0000;
    font-size: 50px;
}
```

But this sets every paragraph on our page to same style

What if we want more flexibility?

```html
<p>I want this to be red</p>
<p>I want this to be green</p>
```

CSS - Classes

```css
p {
    font-size: 50px;
}

p.red {
    color: #ff0000;
}

p.green {
    color: #00ff00;
}
```

Add classes to each element to determine which style(s) should be applied

Here, all paragraphs will have large text and the specific classes will be the assigned colors based on their class

```html
<p class="red">I want this to be red</p>
<p class="green">I want this to be green</p>
```

CSS

```css
p {
    font-size: 50px;
}

p.red {
    color: #ff0000;
}

.green {
    color: #00ff00;
}

#myDiv {
    border-style: double;
}
```

We can apply styles to different subsets of elements

.<class_name> will apply the style to all elements on that class regardless of elements type

#<element_id> will apply the style to the element with that id

id's must be unique

```html
<p class="red">I want this to be red</p>
<p class="green">I want this to be green</p>
<div class="green" id="myDiv"></div>
```

CSS - Multiple Classes

```css
.green {
    color: #00ff00;
}

.wide {
    width: 100%;
}
```

Elements can have multiple classes to apply combinations of styles

Multiple classes are separated by spaces

```html
<div class="green wide" id="myDiv">Div Content</div>
```

CSS

There are many more features of CSS that we won't cover

Again, I recommend W3 Schools for more

https://www.w3schools.com/css/default.asp

JavaScript

JavaScript

We will not thoroughly cover JavaScript syntax

If you haven't used JavaScript before, you are expected to learn the basic syntax and concepts on your own (or stop by office hours)

Topics such as variables, conditionals, loops, functions, and data structures won't be explicitly covered in lecture

As always, I recommend W3 Schools for a good tutorial/reference
https://www.w3schools.com/js/default.asp

Front End JavaScript

We've seen the basic building blocks for web pages in HTML and CSS

This allows us to build what I call "poster sites" - The equivalent of making a poster that people can only view

In this course we want to make web apps that allow our users to interact with our content

To start this interaction, we need JavaScript (Specifically, ECMAScript)

Front End JavaScript

HTML and CSS are not programming languages

JavaScript is a programming language

Javascript allows us to add code to our page

When a user visits your site their browser will:

1. Download your JavaScript code
2. Run your code on their machine

What are the security implications of this?

Front End JavaScript - Security

There are restrictions to what JavaScript can do in the browser:

Cannot read/write files
Cannot access other tabs or windows
Cannot access data (ex. cookies, local storage) from other sites

Vulnerabilities are still exposed/patched

With that, let's see our first script

Front End JavaScript

```
var myDiv = document.getElementById("myDiv");

myDiv.innerHTML = "Content added from JavaScript";
```

Here we call the document.getElementById method which returns the elements with the provided id

The element is an object with a key “innerHTML” whose value is the content (between the open and close tags) of the element

We’ll save this code in a file named “myCode.js”

Front End JavaScript

```
var myDiv = document.getElementById("myDiv");

myDiv.innerHTML = "Content added from JavaScript";
```

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>CSE312 - First Page</title>
</head>

<body>
    <h1>First Web Page</h1>
    <p>My content</p>
    <div id="myDiv"></div>
    <script src="script.js"></script>
</body>
</html>
```

We “import” our javascript code by adding a script element at the bottom of the body element with a src (source) attribute containing our JavaScript filename

This runs our script once the body is loaded

Front End JavaScript

The script runs when the body loads and sets the content of “myDiv” resulting in this page (CSS removed)

JavaScript Libraries

Let's use JavaScript libraries to do cool things without writing code from scratch

JavaScript Libraries

```
<script src="https://momentjs.com/downloads/moment.js"></script>
```

Download external libraries using the script tag the same way we downloaded our own code

Makes an HTTP request to download the library

We can add the script element in the head since we don't need any HTML elements rendered before downloading the library

JavaScript Libraries

```
<script src="https://momentjs.com/downloads/moment.js"></script>
```

When the page loads there will be a second request and the library is downloaded

The library is simply a JavaScript file that others have written and shared with us for free. We call this open-source since everyone has access to the source code

source: https://github.com/moment/moment/

JavaScript Libraries

```
...

<script src="https://momentjs.com/downloads/moment.js">

</script>

...
```

We literally download their code and run it in our browser

```
//! moment.js

;(function (global, factory) {
    typeof exports === 'object' && typeof module !== 'undefined' ? module.exports = factory() :
    typeof define === 'function' && define.amd ? define(factory) :
    global.moment = factory()
}(this, (function () { 'use strict';

    var hookCallback;

    function hooks () {
        return hookCallback.apply(null, arguments);
    }

    // This is done to register the method called with moment()
    // without creating circular dependencies.
    function setHookCallback (callback) {
        hookCallback = callback;
    }

    function isArray(input) {
        return input instanceof Array || Object.prototype.toString.call(input) === '[object Array]';
    }

    function isObject(input) {
        // IE8 will treat undefined and null as object if it wasn't for
        // input != null
```

JavaScript Libraries

Great!.. so what did we just download?

JavaScript Libraries

```
<p id="datetime"></p>
<script src="script.js"></script>
```

```
var elem = document.getElementById("datetime");
elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>")
```

The library defines a function named moment() which returns the current time as an object that contains several methods

Here we get the current time and call the format function to choose how it's displayed using date/time placeholders (ex. MM for a 2 digit month)

JavaScript Libraries

```
<p id="datetime"></p>
<script src="script.js"></script>
```

```
var elem = document.getElementById("datetime");
elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>")
```

Displays the time in the <p> element with id "datetime" in the format

2018-10-18
10:23:57 pm

JS

JavaScript Libraries

This is ok, but it'd be more useful to show the current time instead of the time our code ran

To do this, we'll first wrap our previous code in a function

Then we'll call the built-in "setInterval" function with our function as an argument which will call our function at fixed intervals

The second argument is a Number representing a time in milliseconds

This will call displayTime once per second to give us a live clock

```
function displayTime(){
    var elem = document.getElementById("datetime");
    elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>");
}

setInterval(displayTime, 1000);
```

JS

JavaScript Libraries

Reminder: You are learning concepts, not memorizing specific examples

The concept is using JavaScript libraries to do cool things without writing code from scratch

The moment library was an example of this concept

JavaScript - Events

We saw a few examples of running a JS script to change the page, but what about reacting to the user? That's what we really want after all

Let's explore browser events and write code that can react to them

See here for a list of events - https://www.w3schools.com/js/js_events_examples.asp

Front End JavaScript

```
<div class="green wide" id="myDiv" onmouseenter="makeBlue(this)"
onmouseleave="makeGreen(this)"></div>
```

```
function makeBlue(elem) {
    elem.style.setProperty("color", "#000077")
}
```

```
function makeGreen(elem) {
    elem.style.setProperty("color", "#007700")
}
```

We add a few more attributes to our div to run JavaScript functions on the mouse enter and mouse leave events

The value of these attributes is any valid JavaScript

To keep our project organized, we'll call functions that are defined in a separate file

Front End JavaScript

```
<div class="green wide" id="myDiv" onmouseenter="makeBlue(this)"
onmouseleave="makeGreen(this)"></div>
```

```
function makeBlue(elem) {
    elem.style.setProperty("color", "#000077")
}

function makeGreen(elem) {
    elem.style.setProperty("color", "#007700")
}
```

We pass the element itself as an argument to the functions

In the function, we change one of the style properties of the element

We can run any valid JavaScript here so what you can do is only limited by what you can conceive and code

Activity

Write a web page using HTML, CSS, and Javascript

To meet the requirements you must read documentation (W3 Schools recommended) which will be a theme throughout the course. Your page must contain:

- An HTML file with with at least one list and one table
- An external CSS file that adds style to at least 3 of your HTML elements and uses at least one class
- An external JavaScript file that reacts to at least one event that can be triggered by the user