

Module 1: Fall Semester Overview; Designing Web Apps; Review HTML & CSS

Edgardo Molina, PhD | Head Instructor

CUNY Tech Prep 2018-2019

Fall Semester Overview

Designing Web Applications

Review: HTML & CSS, Intro to Bootstrap

Fall Semester Overview

Weekly Class Meeting outline

Class Outline:

- Meet for 2.5 hours weekly (**Bring your laptop computer!**)
- A full stack topic lecture (~40-60 minutes)
- In-class coding / Student presentations (~90-110 minutes)

Before Class:

- Review lecture slides
- Read assigned materials

After Class:

- Complete in-class coding projects
- Ask questions if any material was not clear

In-class Projects

- Weekly in-class coding projects
- We will guide you through them
- Your responsibility to complete after class

Team Project

- Work in teams of 3
- You will be assigned a project mentor
- Provide mentor with **weekly** progress reports
- 2 **Presentations** during semester
- 2 **Code Reviews** during semester
- You will **demo** the project at the end of semester (Mid-December)

Any questions about the syllabus, the Fall semester, or anything else?

Designing Web Apps

First: What are your favorite web apps?

How would you build a clone of your favorite web app?

Before we begin, we need to *gather the requirements*

- What is the purpose of the application?
 - Does it deliver a service, product or content?
- Who will be the users of our application?
- What user interactions (UX) do we want to allow?
- Who can perform those interactions?
- What data will I need to store?

We need to determine:

- the *types* of users for our application
- the *actions* each user type is allowed in our application

Example: Ecommerce

Seller

- can request account
- can post products
- gets paid for purchases

Buyer

- can sign up for an account
- can browse all products
- can purchase products

Admin

- can approve seller accounts
- can remove sellers
- can remove products

Example: Newspaper

Reporter

- can create/edit articles

Editor

- can edit any article
- can publish an article
- can remove article

Reader

- can read published articles
- can comment on articles

Blogger

- needs an account
- can create blog entries

Reader

- can browse all blog entries
- can read a blog entry
- does not need an account

Next we should Mockup the screens and UX

List all of blog entry titles

My First Blog

My latest blog post is about React.JS

Jquery: learning some JavaScript

Tutorial: how to register a domain

Second post

First Blog post

Read a blog entry

My First Blog

My latest blog post is about React.JS

Today I learned about a new frontend JavaScript framework. It is called React and was developed at Facebook. Looks like a promising framework and it's Open Source!

Form to create a blog entry

My First Blog

New Blog Entry

Title:

Body:

Figure 1: Mockups for a Blog

What data do we care about?

What data do we care about?

- Our form tells us part of the story.
- We should at a minimum save the data our users entered
 - Is there more data we should be saving?

Class Exercise: Let's design the
models for our favorite app

Review: HTML & CSS; Intro to Bootstrap

HTML: HyperText Markup Language

HTML is a markup language used to describe the **semantic** structure of information in a web page *document*.

In other words, HTML tags give meaning to parts of the document. Tags are used to identify headings, paragraphs, lists, etc.

Think of a table-of-contents/outline for a textbook or long article.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CUNY Tech Prep</title>
</head>
<body>
  <!-- The HTML Body -->
</body>
</html>
```

...

<body>

<h1>CUNY Tech Prep**</h1>**

<p>The program for the brightest
New York computer scientists!**</p>**

<h2>Apply to the program**</h2>**

<p>

Apply ****here****.

</p>

</body>

...

Can you tell how that HTML page looks? What is its background color, text color, font, or size?

NO



HTML is NOT...

HTML is **NOT** for describing how things will *look* (color, font, placement, etc...).

Do **NOT** use tables to describe the *layout* of a web page

We use **CSS** to describe a web pages *styles*: look, layout, and animations.

Instead:

- Assign your HTML tags **classes** and **id's** for CSS use
- Use **<div>**'s and ****'s to group multiple content tags or select text for style purposes.

HTML documents are trees!

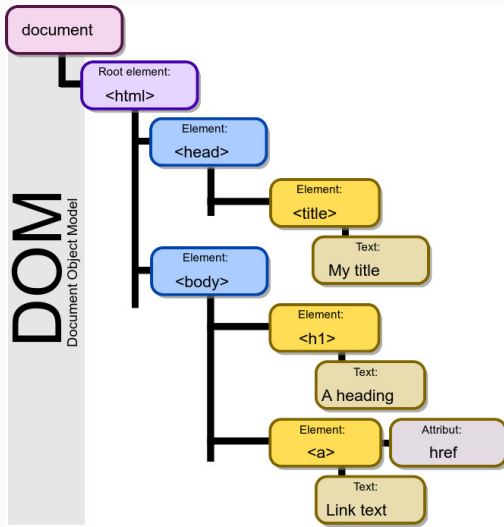


Figure 2: The DOM Tree

`<html>` is the root tag for the HTML document. Contains a *head* and *body*.

`<head>` contains the document metadata.

`<body>` contains the document contents.

All tags must be opened and closed such that the structure remains a tree.

CORRECT: `<p>hi</p>`

WRONG: `<p>hi</p>`

- `<h1>` - `<h6>` tags are for headings
- `<p>` tags are for paragraphs
- `` tags are for very important points
- `` tags are for emphasis
- `` and `` tags are for unordered and ordered lists
- `<table>` tags are for tabular data
- `<a>` tags are for links

- `<form>` for wrapping web based interactive forms
- `<input>` for short text inputs, check and radio boxes
- `<textarea>` for longer text inputs
- `<select>` for dropdown and list menus
- `<button>` for clickable buttons

Sometimes we want to select or group text and tags for style and JavaScript purposes. We do that with `<div>` and `` tags.

Block elements

- start on a new line
- take up the full width of the container it is in
- new content begins below it
- i.e. `<div>`, `<h1>`-`<h6>`, `<p>`

Inline elements

- do not start a new line
- only take up the width of the content
- new content begins immediately to the right of it
- i.e. ``, ``, ``, `<a>`

CSS: Cascading Style Sheets

CSS is a language for describing the presentation of HTML elements when rendered, printed, or used by screen readers.

CSS provides **selectors** for applying styles to HTML Tags, classes, and id's.

```
htmltagname { ... }
```

```
.classname { ... }
```

```
#idname { ... }
```

Apply styles on HTML tags

```
body {  
    left: 0;  
    margin: 0;  
    background-color: #eee;  
}
```

```
h2 {  
    border-bottom: 1px solid lightgray;  
}
```

Apply styles on id's

```
#banner {  
    height: 400px;  
    background-color: darkblue;  
    position: relative;  
}
```

Apply styles on classes

```
.row {  
  padding-bottom: 30px;  
}
```

// on an html tag within a class

```
.menu a {  
  color: #fff;  
  font-size: 15px;  
  text-decoration: none;  
  text-transform: uppercase;  
}
```


Bootstrap

“Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.”

<http://getbootstrap.com/>

Why use Bootstrap?

- Provides nicer/modern default styles
- Provides commonly used HTML components
 - Navigation Menus, Alerts, Panels, Dropdowns, etc...
- Provides a responsive **grid** system for layouts
- Easily extensible

The bootstrap grid system splits **containers** into **rows**. Each **row** can contain up to 12 **columns**.

```
<div class="container">  
  <div class="row">  
    ...  
  </div>  
  <div class="row">  
    ...  
  </div>  
</div>
```

Two types of containers:

- **container** is a fixed-width container. You can assign the width.
- **container-fluid** is a full-width container. Takes up entire width of the browser.
- containers cannot be nested.

Example: Stacked-to-horizontal

Using a single set of `.col-md-*` grid classes, you can create a basic grid system that starts out stacked on mobile devices and tablet devices (the extra small to small range) before becoming horizontal on desktop (medium) devices. Place grid columns in any `.row`.

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

Figure 3: Sample Grid

```
<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

Figure 4: Sample Grid Code

Example: Mobile and desktop

Don't want your columns to simply stack in smaller devices? Use the extra small and medium device grid classes by adding `.col-xs-*` `.col-md-*` to your columns. See the example below for a better idea of how it all works.

<code>.col-xs-12 .col-md-8</code>		<code>.col-xs-6 .col-md-4</code>
<code>.col-xs-6 .col-md-4</code>	<code>.col-xs-6 .col-md-4</code>	<code>.col-xs-6 .col-md-4</code>
<code>.col-xs-6</code>		<code>.col-xs-6</code>

Figure 5: Responsive Grid on Desktop

Example: Mobile and desktop

Don't want your columns to simply stack in smaller devices? Use the extra small and medium device grid classes by adding `.col-xs-*` `.col-md-*` to your columns. See the example below for a better idea of how it all works.

`.col-xs-12 .col-md-8`

`.col-xs-6 .col-md-4`

`.col-xs-6 .col-md-4`

`.col-xs-6 .col-md-4`

`.col-xs-6 .col-md-4`

`.col-xs-6`

`.col-xs-6`

Figure 6: Responsive Grid on Mobile

A Responsive Grid

```
<!-- Stack the columns on mobile by making one full-width and the other half-width -->
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

Figure 7: Responsive Grid code

How do we add Bootstrap to our HTML pages

- Download it and add as part of your project
- Use a CDN link

Links:

- [HTML Reference](#)
- [CSS Reference](#)
- [Block-vs-inline](#)
- [Block-vs-inline 2](#)
- [Bootstrap 4.0](#)
- [Bootstrap 4.0 Grid System](#)