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

Edgardo Molina, PhD | Head Instructor

CUNY Tech Prep 2018-2019

# Outline

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

# Projects

In-class Projects

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

# Projects

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)

# Questions

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?

# Designing Apps

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?

# User types and User stories

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

# Example: Blog

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


Read a blog entry


Form to create a blog entry


**Figure 1:** Mockups for a Blog

# Data Modeling

What data do we care about?

# Data Modeling

## What data do we care about?

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

# Review: HTML & CSS; Intro to Bootstrap

# HTML: HyperText Markup Language

# HTML

**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.

# Example HTML

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

# Example HTML Body

```
...
<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 <a href="/application">here</a>.
    </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 `class`es and `id`'s for CSS use
- Use `<div>`'s and `<span>`'s to group multiple content tags or select text for style purposes.

# HTML documents are trees!

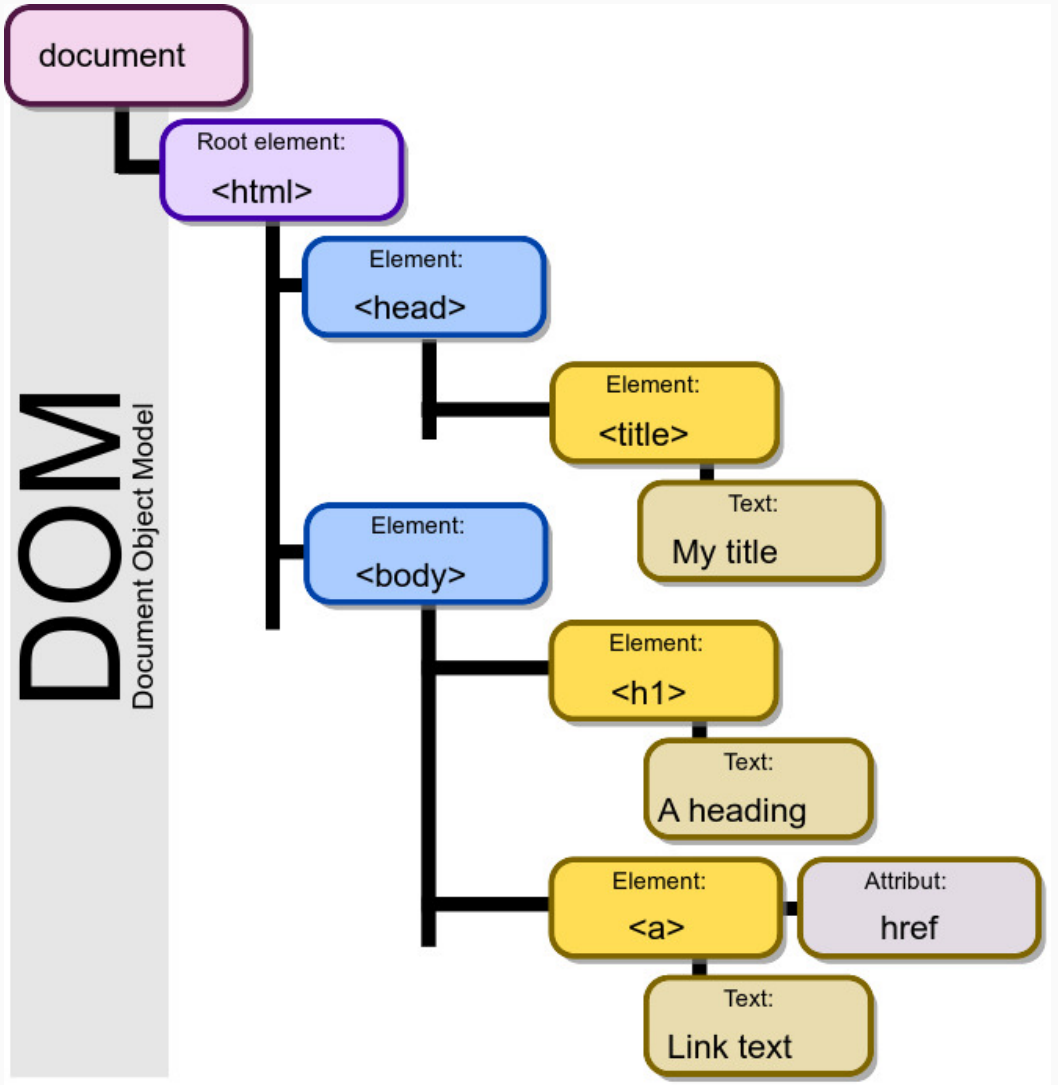


**Figure 2:** The DOM Tree

# HTML documents

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

`<head>` contains the document metadata.

`<body>` contains the document contents.

# HTML documents

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

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

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

# Common HTML Tags for content

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

# Common HTML Form tags for user input

- `<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

# divs and spans

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

# Block vs Inline elements

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. `<span>`, `<strong>`, `<em>`, `<a>`

# CSS: Cascading Style Sheets

# CSS

**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.

# Selectors

```
htmltagname { ... }

.classname { ... }

#idname { ... }
```

# CSS Examples

Apply styles on HTML tags

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

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

# CSS Examples

Apply styles on `id`'s

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

# CSS Examples

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

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

http://getbootstrap.com/

# Why use Bootsrap?

- 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

# Grid System

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>
```

# Grid System

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 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

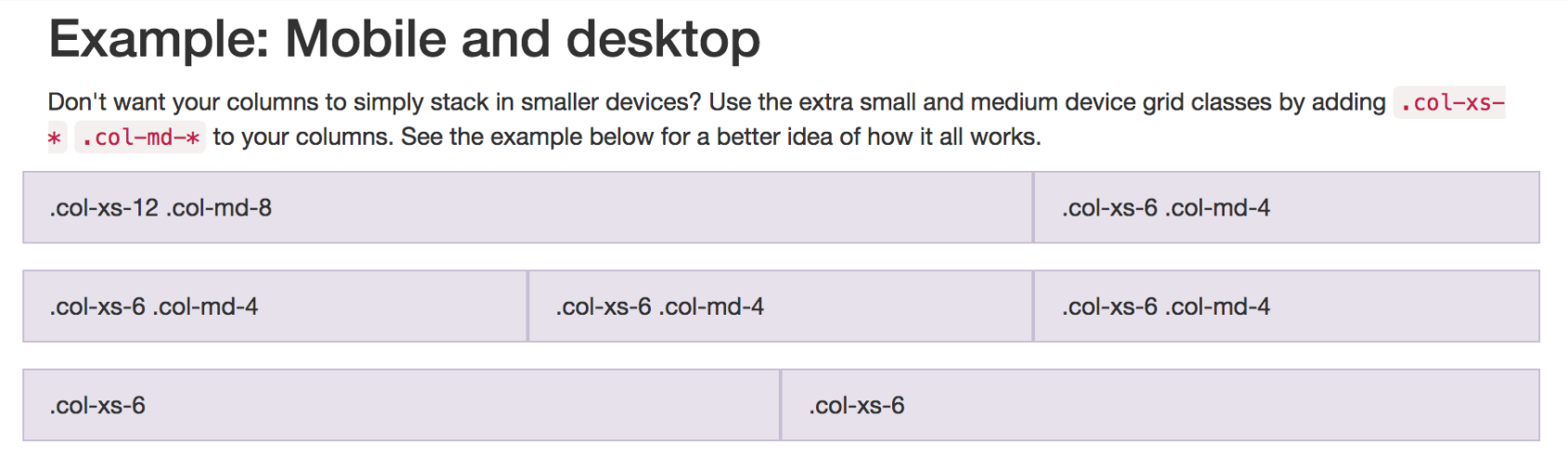


**Figure 5:** Responsive Grid on Desktop

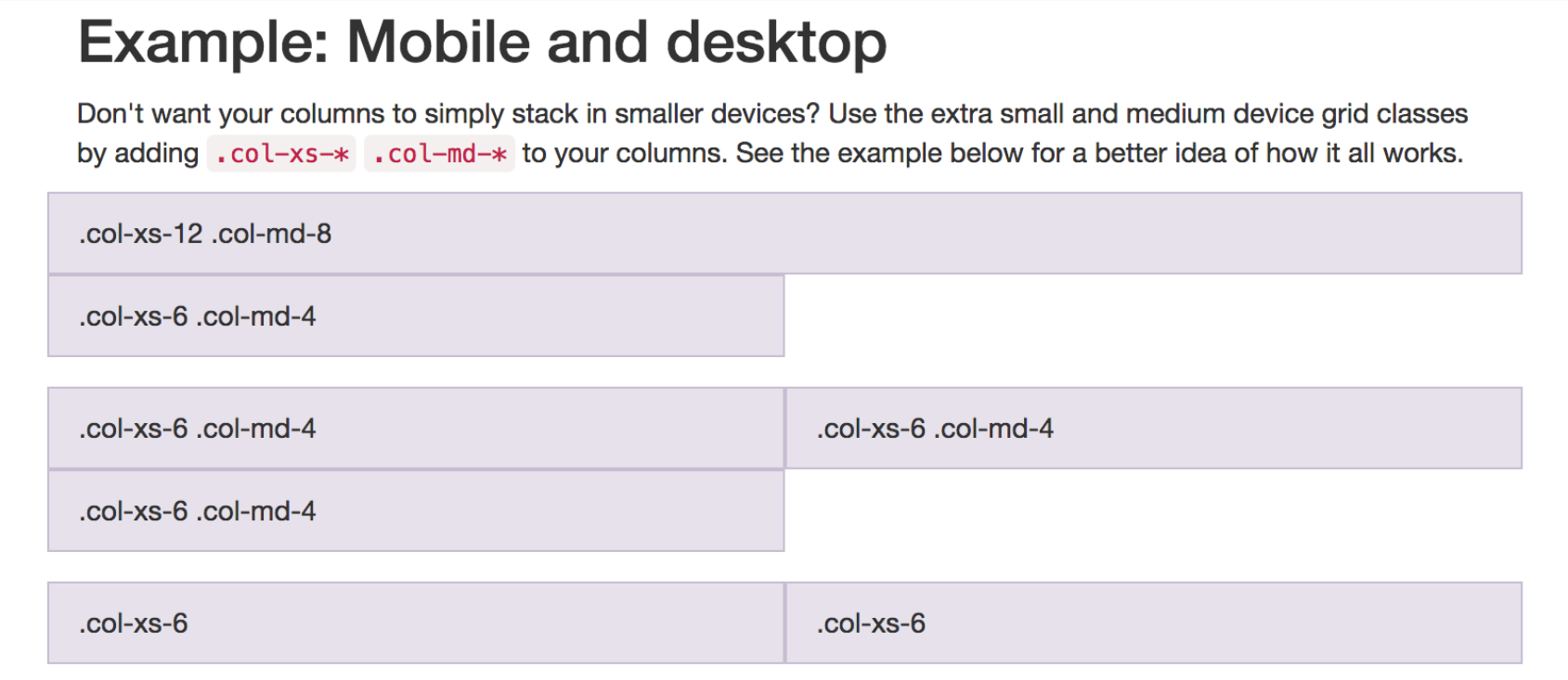


**Figure 6:** Responsive Grid on Mobile

# 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

# HTML/CSS & Bootstrap Resources

Links:

- HTML Reference
- CSS Reference
- Block-vs-inline
- Block-vs-inline 2
- Bootstrap 4.0
- Bootstrap 4.0 Grid System