DAY 1 - BASICS
Why do we need HTML anyway?

Page source without HTML

Browser renders like this
Hyper-Text Markup Language (HTML) is not a programming language.
  - Not executable – nothing to install
  - Merely "rendered" by a browser

Markup languages consist of a set of tags used to describe the contents of a document.
  - Defines paragraphs, items in a list, table data, etc.

HTML documents are Web pages.

In all honesty, we use XHTML Transitional coding for our pages, but you usually don't need to worry about that.
## Document Outline
- Version of (X)HTML
- HTML document
- Page information
- Page contents

## Lists
- `<ol>`: Ordered list
- `<ul>`: Unordered list
- `<li>`: List item
- `<dl>`: Definition list
- `<dt>`: Definition term
- `<dd>`: Term description

## Forms
- `form`: Form
- `fieldset`: Collection of fields
- `legend`: Form legend
- `label`: Input label
- `input`: Form input
- `select`: Drop-down box
- `optgroup`: Group of options
- `option`: Drop-down options
- `textarea`: Large text input
- `button`: Button

## Objects
- `object`: Object
- `param`: Parameter

## Empty Elements
- `area`
- `img`
- `base`
- `input`
- `br`
- `link`
- `col`
- `meta`
- `hr`

## Core Attributes
- `class`
- `style`
- `id`
- `title`

**Note:** Core Attributes may not be used in `base`, `head`, `html`, `meta`, `param`, `script`, `style` or `title` elements.

## Language Attributes
- `dir`
- `lang`

**Note:** Language Attributes may not be used in `base`, `br`, `frame`, `frameset`, `hr`, `iframe`, `param` or `script` elements.

## Keyboard Attributes
- `accesskey`
- `tabindex`

## Window Events
- `onLoad`
- `onUnload`

## Form Events
- `onBlur`
- `onReset`
- `onChange`
- `onSelect`
- `onFocus`
- `onSubmit`

## Keyboard Events
- `onKeyDown`
- `onKeyUp`
- `onKeyPress`

## Mouse Events
- `onClick`
- `onMouseOut`
- `onDblClick`
- `onMouseOver`
- `onMouseDown`
- `onMouseUp`
- `onMouseMove`
HTML Basics

```
<html>
  <head>
    <title>Robby's Page</title>
  </head>
  <body>
    <p>This is a picture of my cats.</p>
    <img src="mycats.jpg" />
  </body>
</html>
```

Spacing doesn’t matter (much), but indentation helps show nesting.
HTML Basics

Markup Tags - Nesting

- Open, surround, close
  
  `<tag attribute="value"...>stuff</tag>`

  `<p>This is my very short paragraph.</p>`

  `<a href="http://www.olemiss.edu">The University of Mississippi</a>`

- Empty or self-closing
  
  `<br />`

  `<img src="mypic.jpg" />`
The body section contains the content of the document along with (just enough) markup code to identify or classify it.

The head information is meta-data: e.g., it is not normally displayed directly, but instead it affects or describes the appearance and performance of other elements in the document.
Separation of form from content:

- **Content** – Information and its contextual meaning
- **Form** – The layout and arrangement of information

(Remind me to show you Zen Garden later.)
Conceps

"Get used to disappointment."
– The Princess Bride

Obstacles to consistent page rendering include:

- Different browsers
- Different resolutions
- Different preferences
- Different needs
- Different devices
This is a picture of my cats.
Dreamweaver Basics

- Top menu: File, Edit, View...
- Filename tab(s)
- Views: Code, Split, and Design plus Live View
- Title
- Upload/Download options
- Visual Aids and Validation
Dreamweaver Basics

- Nesting / Tag selector
- Properties window with context-sensitive formatting options
Dreamweaver Basics

- CSS Styles
- Files
  - Site name
  - Local / Remote view
  - Upload/Download buttons
  - File list
Let's get to work! Step 1...

Build a page that looks something like this:

- Use Header 1.
- Create un-ordered list with some items.
- Set page title.
Step 2...

Change the page:

- Insert table with several rows, 3 columns, and header on top.
- Give column headings names.
- Copy the list items into individual rows.
- Fill out remainder of table.
- Select cells to color and center.

**These are my pets:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Species</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barack</td>
<td>cat</td>
<td>white and gray</td>
</tr>
<tr>
<td>Calvin</td>
<td>dog</td>
<td>black</td>
</tr>
<tr>
<td>Franklin</td>
<td>cat</td>
<td>yellow and white</td>
</tr>
<tr>
<td>Herbert</td>
<td>cat</td>
<td>brown and black</td>
</tr>
<tr>
<td>Lou</td>
<td>cat</td>
<td>tortoiseshell</td>
</tr>
<tr>
<td>Michelle</td>
<td>cat</td>
<td>black</td>
</tr>
<tr>
<td>Millard</td>
<td>cat</td>
<td>pale yellow</td>
</tr>
<tr>
<td>Nellie</td>
<td>dog</td>
<td>white</td>
</tr>
</tbody>
</table>
Step 3...

Change the page again:

- Merge header cells and relabel.
- Merge right column cells, remove centering, vertically align to top, and fill with dummy text.
- Add bottom row with footer content.
- Create links for each item in left column.
- Change cell colors and adjust last menu item.
Step 4

Change the page once more:

- Insert a photo and align it to the right.
- Notice how browser window size affects layout.
- Set table width to 500 pixels.
- Notice browser window size no longer matters.
Compare list to table

- Control over format
- Amount of code needed
- Flexibility concerns
The Problem – part 1

- We've been mixing the layout details with the content, violating the "separation of form and content" rule.

- If we can't do it like this, what other options are there?

- Stay tuned for the exciting conclusion!
The Problem – part 2

- When we last saw our code, we were embedding all the layout details within the content, thus violating the rule about separating the two.

- What we need is a type of code that describes the layout of the content, but isn't all mixed in with the content.

- CSS to the rescue!
Cascading Style Sheets

- HTML describes only the content of the document.
  - `<h1>`, `<p>`, `<li>` describe the type of content, and not specifically how they should appear.
  - The formatting of these elements by the browser is very limited.

- CSS was added to describe the layout of the HTML elements.
  - Styles are normally saved in external files. These allow you to change the appearance of an entire site just by editing one single file.

- Zen Garden
Cascading Style Sheets

Example: The link element

```html
<a href="http://www.google.com">www.google.com</a>
```

- Visited links are **purple**
- Unvisited links are **blue**
- Active links are **red**

The style for each of these is re-definable!
Cascading Style Sheets

- Adding a style definition for the link element changes its appearance.

```css
a {
  font-weight: bold;
  color: yellow;
  background: blue;
  text-decoration: none;
}
```
Cascading Style Sheets

- CSS can also adjust only a particular aspect of some elements.

```css
a {
  font-weight: bold;
  color: yellow;
  background-color: blue;
  text-decoration: none;
}

a:hover {
  color: black;
  background-color: red;
}
```
Cascading Style Sheets

“Cascading” refers to the precedence of definitions for a given element.

1. Browser default
2. External style sheet
3. Internal style section *(inside the <head>)*
4. Inline style *(inside the HTML element)*
There are four ways to impose a style on HTML elements:

1. By element/tag name…
   Make all paragraphs bold face.
   ```html
   p {font-weight: bold ;}
   ```

2. By class attribute…
   Make anything with class="column" green.
   ```html
   .column {color: green ;}
   ```

3. By ID attribute…
   Make the one element with id="header" all uppercase.
   ```html
   #header {text-transform: uppercase ;}
   ```

4. By style attribute…
   Italicize this very element.
   ```html
   <p style="font-style: italic ;">
Cascading Style Sheets

Identifiers may also be combined.

- `p#myid {font-weight: bold;}`
  The one paragraph with id="myid" will be bold.

- `p.myclass {color: yellow;}`
  Any paragraph with class="myclass" will be yellow.

- `#mycontent .orange {color: orange;}`
  Any element with class="orange" inside the one element with id="mycontent" will be orange.
Cascading Style Sheets

Multiple identifiers can use the same style.

- `p.first, p.second, p.third {font-weight: bold;}`
  Any paragraphs with class="first", "second" or "third" will be bold.

- `table, td {border: 1px solid black;}`
  Every table and every table data cell will have black solid border 1 pixel wide.
Dynamic & Reusable Content

- What content is generated automatically?
  - Calendar events
  - System-generated dates
  - Anything requiring programming

- Which parts of the page will appear on other pages? Create separate files for those parts and include them back into their locations. This allows you to make updates in only one place that impacts the entire site.

<!--#include virtual="myfile.htm"-->
Deconstructing the page

- Identify recurring parts
  - **Header**
  - **Navigation**
  - **Footer**
  - **Stylesheet**
  - **Javascript**

- Identify dynamic content
  - Calendar events listing
  - Date of last modification
1. Define your site:
   - Site / New Site
   - Select Advanced Tab
   - Local Info…
     - Site name: your name
     - Local root folder: …/Documents/yourname
   - Remote Info…
     - Access: FTP
     - FTP host: cedar.olemiss.edu
     - Host directory: working
     - Login: maildemo
     - Password: ********
     - Use Secure FTP (SFTP): Checked
Define new page name

2. Edit → Preferences → Code Format
   Line break type: LF (Unix)

3. View Remote Files

4. Download the index.html file from the server

5. View Local Files

6. Change index.html to yourname1.html

7. Upload new file name to server

8. View in browser at
   www.olemiss.edu/working/maildemo/yourname1.html
9. Change “by” name and save/upload the page again.
10. Consider the left menu hover action

CSS

#leftcol a:hover {
    background: #395494;
    color: #dedede;
}

HTML

<body id="pg1">
...
<div id="leftcol">
    <ul id="navigation">
        <li id="head1"><a class="toplevel" href="index.html" id="li1">First Button</a></li>
        <li id="head2"><a class="toplevel" href="index.html" id="li2">Second Button</a></li>
        <li id="head3"><a class="toplevel" href="index.html" id="li3">Third Button</a></li>
        <li id="head4"><a class="toplevel" href="index.html" id="li4">Fourth Button</a></li>
        <li id="head5"><a class="toplevel" href="index.html" id="li5">Fifth Button</a></li>
        <li id="head6"><a class="toplevel" href="index.html" id="li6">Sixth Button</a></li>
    </ul>
</div>
Customize the menu

11. Change the filenames to use your own filenames

- <li id="head1"><a class="toplevel" href="yourname1.html" id="li1">First Button</a></li>
- <li id="head2"><a class="toplevel" href="yourname2.html" id="li2">Second Button</a></li>
- <li id="head3"><a class="toplevel" href="yourname3.html" id="li3">Third Button</a></li>
- <li id="head4"><a class="toplevel" href="yourname4.html" id="li4">Fourth Button</a></li>
- <li id="head5"><a class="toplevel" href="yourname5.html" id="li5">Fifth Button</a></li>
- <li id="head6"><a class="toplevel" href="yourname6.html" id="li6">Sixth Button</a></li>
Disassemble the page

12. Put the Header, Navigation, and Footer in separate files
   - yourname-head.htm
   - yourname-menu.htm
   - yourname-foot.htm

13. Include them in the original file:
   <!--#include virtual="yourname-head.htm"-->

(Note that this is a Server Side Include which your browser can only handle when viewing your page on a server.)
#### Disassemble the page

14. Put the contents of the `<head>` section in separate file `yourname-meta.htm`.

```html
<!--#include virtual="yourname-meta.htm"-->
```
Reassemble more pages

15. Change <body id="pg2"> and page name.
16. Save/upload as *yourname2.html*
17. Change <body id="pg3"> and page name.
18. Save/upload as *yourname3.html*
19. Repeat for remaining buttons.
20. Once all pages are created and uploaded, browse your site and see how the pages relate to each other.
Reminders

- Avoid uppercase characters and spaces Web folder and file names.
  
  Spaces get converted to “%20” text, and browsers hate them.

- Keep content current.
  
  If people wanted to see old information, they’d look on a printed piece.
Homework – Take Inventory

- Read through everything on your current site.
- Make a list of all the pages.
  - Indicate which pages can be deleted.
  - Indicate which pages need to be updated and begin getting the updates.
  - Consider if any pages need to be added.

- Review your current navigation menu.
  - Do you think visitors could easily find everything they want using your menu?
  - Can you consider reducing the number of links and still achieve that goal?