
INFO 654 – Information Technologies

Week 4 – HTML

Week 4 Agenda

- I. Qs: Lab/Quiz?, Current Events?, other Qs?
- II. Web page rendering
- III. HTML:
purpose/process
- IV. HTML
elements, tags, tag attributes
- V. Your coding environment
- VI. hands-on!

<break>

- I. Markup validation
- II. Qs and debugging
- III. Upcoming: web project
- IV. Work time on HTML and brainstorming your web projects

Network & Databases

Lab/Quiz

- **Due by the start of class next week**
SEPT 27
 - Full description doc posted on the LMS
- Open book/open class slides (as you would also look up information from trusted sources in a real-world environment)
- Submit as a PDF or DOC as in instructions
- *(also remember we start 10 mins early next week with Tk!)*

Current Events

Presentation

- **On the date you signed up for**
 - Full description doc posted on the LMS
- “Current” could be from anytime in 2022 or tell me in advance if you have something that might be applicable but is not in current year
- Presentation is in-class, at beginning of class
- If you have any visuals, be able to access them from the front-of-room workstation (online or on USB)
- No addt'l materials due

Looking Ahead...

Today we will begin to talk about the first of three coding languages commonly used for website creation and how they work together to generate web sites. Between today and the next few weeks we'll look at:

HTML

Hypertext Markup Language

to structure content on a web page and manage the entire page rendering process

CSS

Cascading Style Sheets

to shape how content appears on the page, for styling and design

JS

JavaScript

to create dynamic actions and interactivity between the site and the viewer

And Looking Behind, Where We Were Last Week

What is the World Wide Web?

How are web pages generated?

What happens when you type a URL into your browser...?

Requesting a Web Page

- Your computer is the client computer and the computer with the Web page is the server (Web server)
- When you click a Web link, your computer gets the page for you...beginning the client/server interaction
- Servers do not store Web pages in the form seen on our screens
- The **pages are stored as a *description of how they should appear on the screen***
 - Web pages are written in HyperText Markup Language (HTML) (which “calls in” other code and resources)
- The **browser receives the description/source file and creates the Web page image** that is described

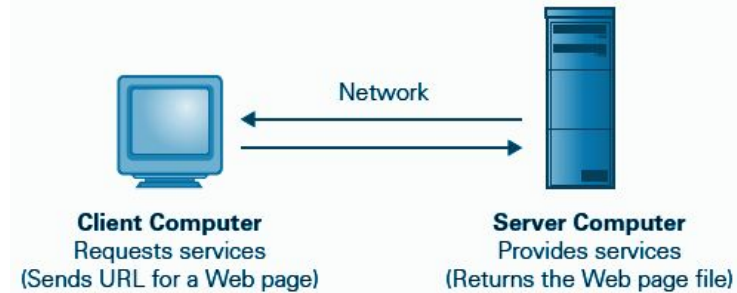


Figure 3.2 The basic client/server interaction, as illustrated by the browser (client) requesting Web pages provided by the Web server.

JULIA EVANS
@b0rk



When You Open a Web Page....

When you open a web page made of these 3 coding languages, what typically happens is:

the **HTML** tells the browser about your page (metadata) and in what order to call on the other two code aspects; It also provides text content and links to resources and media.

the **CSS** focuses on styling information -- the HTML will tell the browser which CSS instructions for design styles to follow for which content pieces, then the CSS will provide the details on styling to the browser for the content piece.

the **JavaScript** has functions and processes written out in code form that the browser can understand so that the browser can run through a variety of processes and render various things for the viewer to see; the HTML will tell the browser when to call upon specific JavaScript functions/actions and how to handle interactivity.

When You Open a Web Page, imagine.....

It could be like
(after the URL-to-IP has been resolved)...:

The browser says, “hey, hi there **index.html**,
*what should I show this viewer here looking at
me who gave your host server a call?*”

HTML says, “hi there! I have some HTML5
incoming for ya. And here’s a few things about
my page you should know right from the start
before you start rendering stuff. And then you
should begin by showing this stuff here. And my
friend CSS here will tell you what it should really
look like. And me and my friend JavaScript are
going to talk together a lot back and forth to tell
you the different things you should show. CSS
will chime in to remind you how to keep it
looking right.”

When You Open a Web Page, imagine.....

It could be like...:

A PERFORMANCE

You are coding a visual show!

(well, like a very heavily scripted and technically spec'd performance)

HTML + CSS + JavaScript = interactive website

roles (and allegorical roles in the “performance”):

HTML **structure, content** *Master of Ceremonies, Writer, Choreographer*

CSS **style, visuals, layout** *Art Director, Designer, Sets & Costumes*

JavaScript **action, dynamics** *Dancer, Acrobat, Aerialist, Improv*

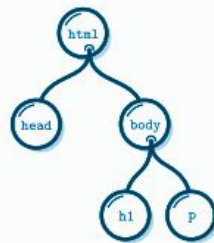
Working together to generate the website, to create the “performance” for the viewer.

Generating Web Sites via HTML, CSS, and JS

HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript are the languages that run the web. They're very closely related, but they're also designed for very specific tasks.

- HTML is for adding meaning to raw content by marking it up.
- CSS is for formatting that marked up content.
- JavaScript is for making that content and formatting interactive.

Think of HTML as the abstract text and images behind a web page, CSS as the visuals that actually gets displayed, and JavaScript as the behaviors that can manipulate both HTML and CSS.



HTML



CSS



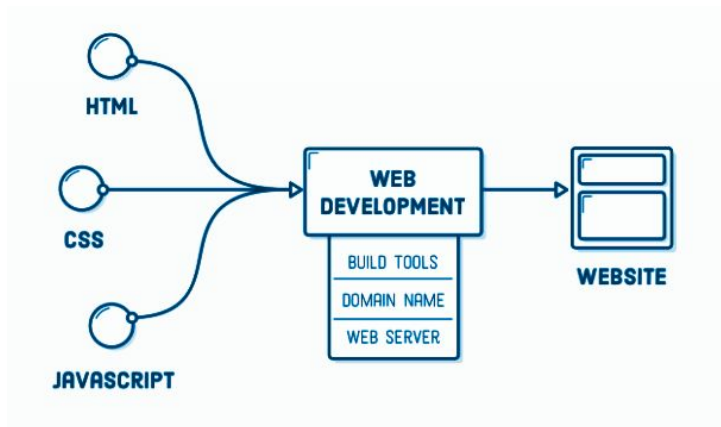
JAVASCRIPT

Bigger Picture: Web Development

Keep in mind, mastering HTML, CSS, and JavaScript are only a prerequisite for becoming a professional web developer, should your career path go more in that direction. There are a bunch of other practical skills that you need to run a website:

- Organizing HTML into reusable templates
- Web server management or service
- Moving files from your local computer to your web server
- Reverting to a previous version when you screw something up
- Pointing a domain name at your server
- Potential server-side scripting needs
- and more...

Dealing with these complexities involves setting up various “environments” to organize your files and handle the building/deploying of your website.



HTML

.html

HTML role

what it does: **structure content**

informal allegory:

Master of Ceremonies, Conductor, Choreographer

HTML shapes the process of generating the web page on a browser

HTML syntax/punctuation to be alert to:

tag structures: `< > </>` vs. ``

tag attribute structures: `<div id="myIdentifier"> </div>`

HTML: HyperText Markup Language

“HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content.”

-- MDN

“HTML contains your page content and describes its meaning.”

-- Visual QuickStart Guide HTML and CSS

“HTML defines the content of every web page on the Internet. By “marking up” your raw content with HTML tags, you’re able to tell web browsers how you want different parts of your content to be displayed. “

-- Interneting Is Hard

Includes:

- *Content (text)*
- *References (to files, resources, etc.)*
- *Markup (to organize and extend content, etc.)*
- *Information for the browser*

each of these are written in and provided to the browser in the format of text

HTML: HyperText Markup Language

- **A markup language for structuring content:**
HTML consists of a series of **elements**, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way or fall a certain way in the hierarchy/tree of content. The enclosing **tags** can make a word or image hyperlink to somewhere else, can emphasize words, can structure phrases as a list, and so on.
- Markup's purpose **is also to facilitate navigation to different resources and content across networked users/computers**. Links are a fundamental aspect of the Web.

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>Interactive Data Visualization Tutorial</title>
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro">
  <link rel="stylesheet" href="./style.css">
</head>
<body>
  <h1>Hello World!</h1>
  <h3>Tutorial Assignment: 1_2_basic_html </h3>
</body>
</html>
```


HTML: HyperText Markup Language

Anatomy of an HTML element



Elements and tags are *not* the same things. Tags begin or end an element in source code, whereas elements are part of the DOM, the document model for displaying the page in the browser.

(we'll talk about the DOM more in two weeks)

Tree Structure (or a nesting of cabinets, shelves, and boxes...)

HTML elements in tree structure -- also conceptualized as “parents-and-children” (i.e.: “parent element of”, “child element of...”) or could be thought of as nesting cabinets/boxes

- When one element contains another
 - (such as a list item `` within an unordered list ``)
- An example of the nested structure that is core to writing HTML
- Important to place closing tags for “parent” elements accordingly -- OUTSIDE the end closing tag of any “child” elements related to it.

(Note: This is one example of the importance of syntax in coding — things like punctuation, capitalization, and placement are key to communicate via code, and each code language has different syntax expectations. It is very common for code to “break” and not run as expected, simply due to something like a missing semicolon or misplaced end tag. Be sensitive to this and check your code as you go.)

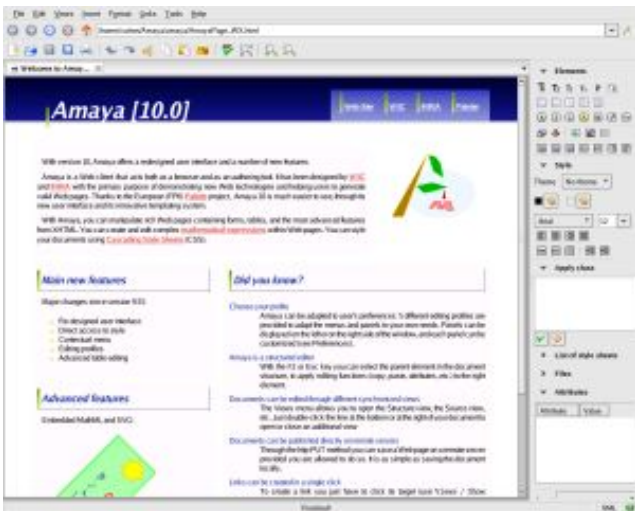


How do we create and work
with web page files?

INTRODUCTION TO WEB PAGES

Why Code HTML by Hand?

- The only way to learn is by **doing**
- **WSIWYG** (??) editors...
 - Often generate unreadable code
 - Ties you down to that particular editor
 - Cannot help you manipulate backend databases
 - Little help when it comes to Javascript
- Hand coding HTML allows you to have *finer-grained control* of your page's look
- **HTML** is demonstrative of other *important* concepts:
 - **Structured documents**
 - **Markup**
 - **Metadata**



HTML Elements

Each individual markup code is referred to as an *element* or *tag* with a specific purpose

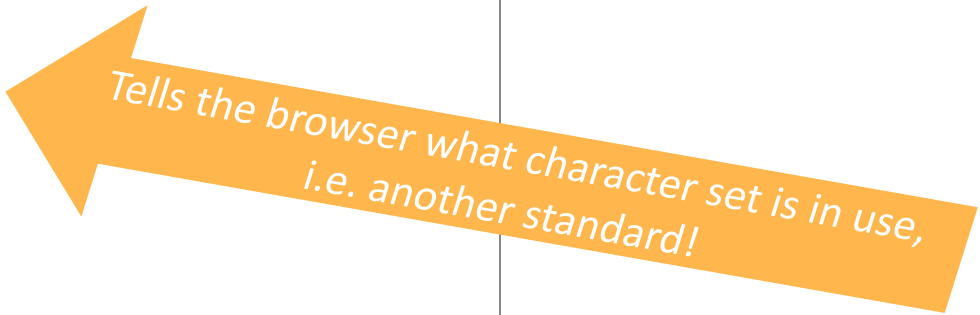
- The W3C defines the standards (<https://www.w3.org/standards/>) for building and rendering web pages, include HTML and CSS

HTML tags that are required for every Web page:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
  </body>
</html>
```



Tells browser our HTML version



Tells the browser what character set is in use,
i.e. another standard!

HTML 5

- The `<!DOCTYPE>` declaration is not an HTML tag; it is **an instruction to the web browser** about what version of HTML the page is written in
`<!DOCTYPE html>` tells the browser we're using HTML5
- HTML5
 - Latest HTML standard
 - Wide adoption on the web today
 - Integration of multimedia
- You may encounter sites using previous versions of HTML, but that is not best practice for new work

```
<!DOCTYPE html>
<html>
...
</html>
```

Required Tags

```
<!DOCTYPE html>
```

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

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>My cool page</title>
```

```
  </head>
```

```
  <body>
```

```
    The main content of the page goes here
```

```
  </body>
```

```
</html>
```

The head section contains the beginning material like the title and other information that applies to the whole page such as metadata

The body section contains the content of the page

Setting Up Your Coding Environment

- *Hint – you're already setup!*

- All you need is:

- A browser (Firefox, Chrome, etc.)

- A text editor

- (operating systems come with text editors installed – TextEdit can be found on the Mac; Notepad comes with Windows; other options have free tiers or are shareware, such as BBEdit, SublimeText, Atom, and VS Code)*

- You may wish to open your favorite text editor and favorite browser side-by-side

Text Editors

Notepad/Textedit could get the job done but there are much better choices (either free or with extended trial periods):

- **BEdit (Mac, trial/free version)**
 - <https://www.barebones.com/products/bbedit/>
- Sublime Text (Windows, Mac, Linux – free ongoing evaluation/donation request)
 - <https://www.sublimetext.com/>
- Notepad++ (Windows - free)
 - <http://notepad-plus-plus.org/>
- Brackets (Windows, Mac, Linux - free)
 - <http://brackets.io/?lang=en>
- Atom (Windows, Mac, Linux - free)
 - <https://atom.io/>
- Visual Studio Code (Windows, Mac, Linux – free)
 - <https://code.visualstudio.com/>

BBEdit

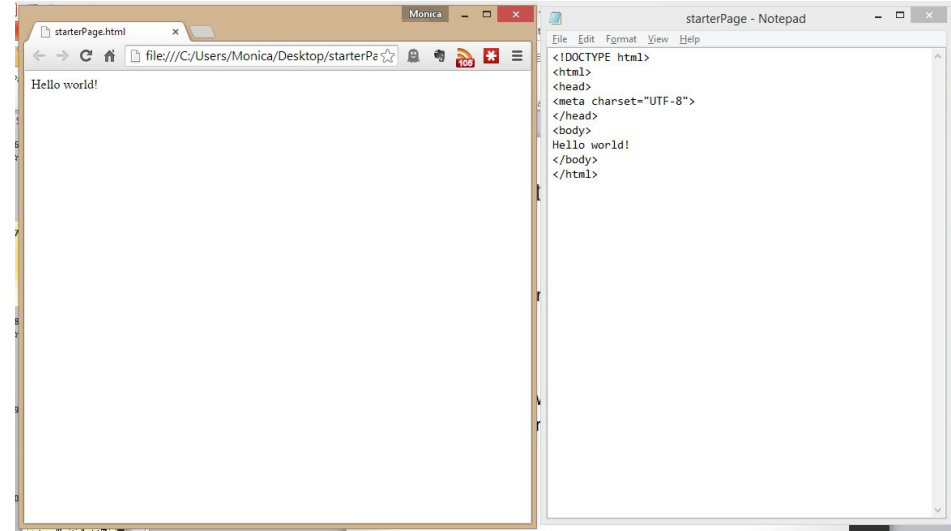
Open BBEdit on your lab computer. We will create a new basic HTML file :

1. File -> New text document
2. Type in your html basic page code
(*sample at right*)
 - a. Note you can choose “light” or “dark” modes under Preferences → Text Colors → Color Scheme
3. File -> Save As
 - a. Save the file as: index.html
 - b. Save it in a folder where you can easily organize class demo code files
4. Open the file with the browser of your choice (Firefox or Chrome)
 - a. A built-in function for this can be found under the “Markup” menu

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
  </head>
  <body>
  </body>
</html>
```

Compose and Check!

- A productive way to work is to keep two windows open:
 - your text editor
 - your browser
- After writing a few HTML tags, **save** the file
 - **Refresh** the browser to see your changes
- Repeat!



As HTML is written, files must be opened in two applications:

1. *the text editor, to **make** changes*
2. *the browser, to **see** the changes*

(refresh/reload the page to see changes)

Get Ready, Get Set

Hello World!

- Set up your coding environment for work in the Pratt lab
- Open BBEdit
 - adjust any settings you like (such as light vs dark color modes)
 - today we'll all use BBEdit but in future you can choose your favorite
- Open Chrome or Firefox
- You may want to set them up narrow, side by side
- Add some text within your `<body></body>` tags and save the file and refresh the browser!



<HTML> TAGS

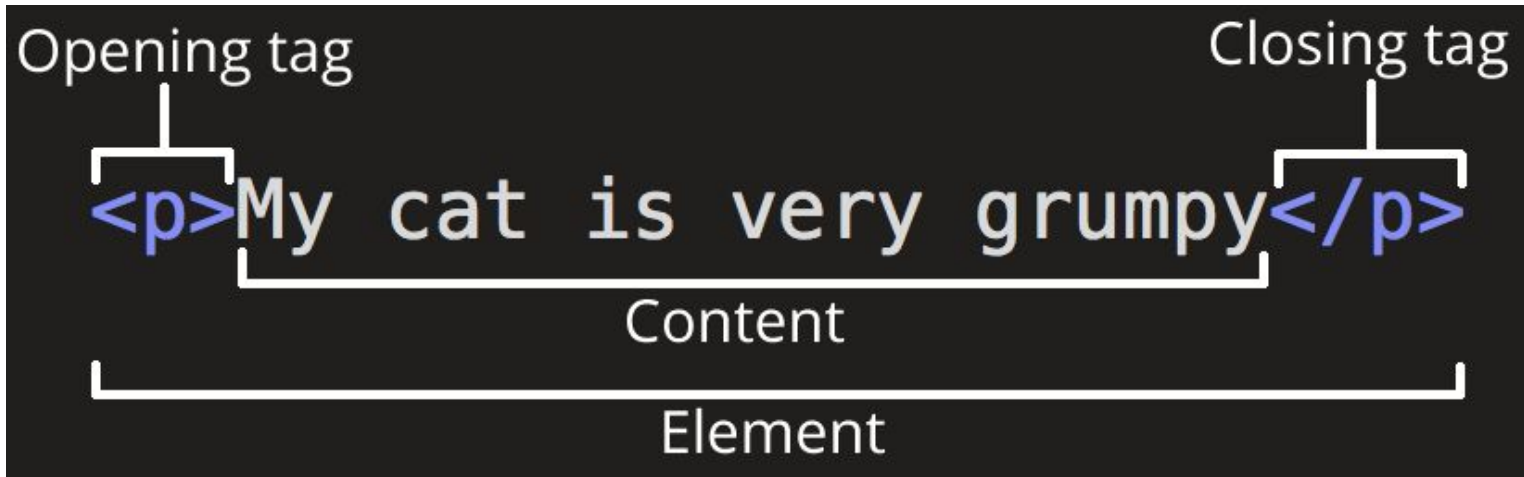
You will use/see these most frequently!

BASIC HTML TAGS

Hands On: HTML Tag Fundamentals

Examples of coding elements and tags we will go over in this session:

```
<!DOCTYPE html>  
<html >  
<head>  
<title>  
<body>  
<h1>  
<h2>  
<p>  
<table><tr><td>
```



"HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way." -- MDN

Tags

Tags are a fundamental concept in HTML. You use them to delineate and set up the different elements of your web page. They are a **primary structuring mechanism** for web development.

The vast majority of tags have a syntax that involves a “start” (or opening) and a “stop” (or closure or ending). Like this:

`<body>` = *“hey browser, here is where the HTML is telling you some content to flow”*
`</body>` = *“this is where that stuff ends”*

As you review code examples, look for the `< >` and the `</ >` to find where different aspects of what is being told to the browser to do/display begin and end.

The exceptions are the small number of “self-closing” tags, such as `` and `<input>`

Adding Title and Metadata Tags

- Most of what is in the <head> section of your code is like whispering background/meta information to the browser about your page (except for <title>)
- Title is displayed to users in the top bar of the display.
 - Insert the <title>...</title> tag within the <head>...</head> tags.
- Meta tag specifies information about the web document
 - Provides information for search engine indexing

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My great page</title>
<meta name="description"
content="A first web page.">
<meta name="keywords"
content="HTML, CSS, webpage">
</head>
<body>
</body>
</html>
```

Adding Content to the Body

- The body section contains the content of the page
 - ALL content that is meant for the user/viewer should be placed within the `<body>...</body>` tag
 - it may be “called in” to the body area via scripts and resource links

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My great page</title>
<meta name="description"
content="A first web page.">
<meta name="keywords"
content="HTML, CSS, webpage">
</head>
<body>
Some great page content
would go here!
</body>
</html>
```

Sections and Paragraphs

- The heading tag creates headings at different levels
 - `<h1>..</h1>` is the largest, top-level heading `<h6>` is the lowest
- The `<p>` tag is the paragraph element
 - Groups sentences and sections of text together.

```
<body>
<p>A paragraph of text. Some more text.</p>
<h2>Heading Level 2</h2>
<h3>Heading Level 3</h3>
<h4>Heading Level 4</h4>
<h5>Heading Level 5</h5>
<p>Another wonderful paragraph.</p>
<h6>Heading Level 6</h6>
</body>
```

Making Links: HTML anchor <a> tag

- The *anchor* element specifies a hyperlink reference (href) to a file/page
 - The text between the <a> and is displayed on the web page as the link text
 - The *href* attribute indicates the file name or URL, i.e. where you want the link to go
 - (bonus: do you remember the attribute that makes it load into a new page/new tab...?)

```
<h1>Some cool links:</h1>
```

```
<a href="https://www.pratt.edu">Visit Pratt</a>
```

Absolute vs. Relative Links

- **Absolute link**

- Link to *other* web sites

- **Relative link**

- Link to pages on *your own site/server* using the *filepath*
- They are **relative** to the folder your file is within, e.g. (*example at right*)
- (*bonus: remember what the “../” indicates...?*)

```
<a  
href="https://www.pratt.edu">Vi  
sit Pratt!</a>
```

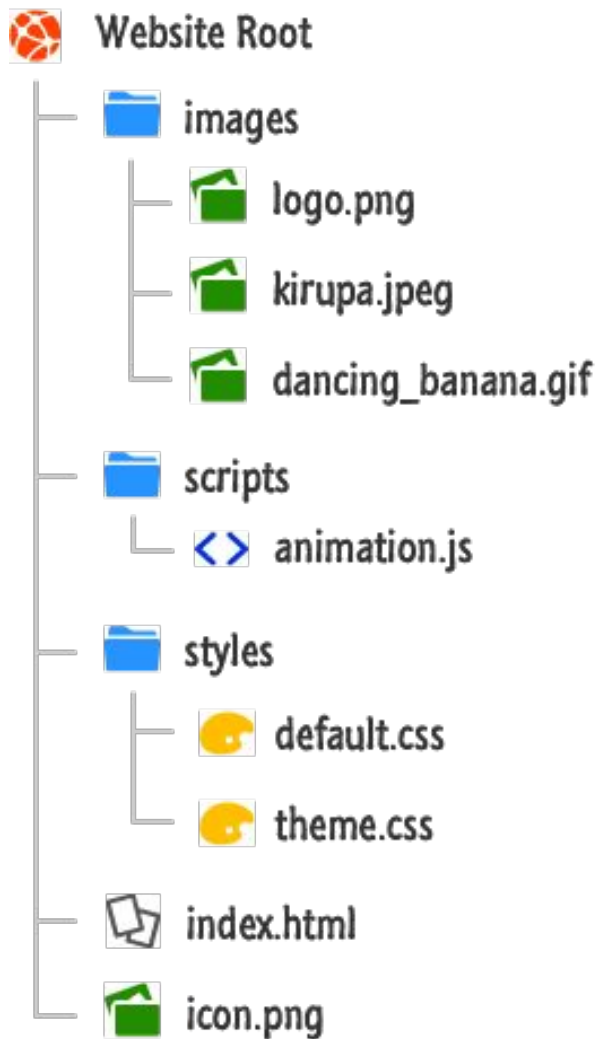
```
<a href="about.html">About  
Us</a>
```

```
<a href="friends/tom.jpg">Tom's  
Picture</a>
```

```
<a  
href=" ../mice/jerry.jpg">Jerry's  
Picture</a>
```

Working With Filepaths

- Your browser uses the file path to figure out where to go and grab a resource from
 - E.g. another webpage, an image, or CSS file, etc.
- File paths are *relative* to where you currently are in the file system
- More examples at:
https://www.kirupa.com/html5/all_about_file_paths.htm



Email Links



- An email link can be created using *mailto:* in the href attribute
 - When clicked, this will automatically launch the default mail program configured for the browser
 - This will vary based on how the user's computer is set up!

```
<a href="mailto:me@hotmail.com">Email Me!</a>
```

Images

- An image tag specifies a file that contains an image
 - `src` is the abbreviation for source, i.e. the location of the image file expressed as a filename
- Filename uses the same rules for absolute and relative pathnames as anchor tags
 - the file name needs to use the correct file extension
 - `.gif`, `.png`, and `.jpg` are frequently used extensions
- *alt* value specifies an alternative form for the image, usually a textual description
 - **This is extremely important for accessibility**, e.g. screen readers

```

```


Linked Images

- You can *nest* the tag within <a> tag to create a linked image
 - you are wrapping the entire content of the with the anchor link
 - Clicking on the image opens the link

```
<a href="https://www.barcelona.com/">  
  
</a>
```

Unordered Lists

- One type of list is the unordered list
 - Unordered list tags `` and `` surround the items of the list
 - The items are enclosed in list item tags, `` and ``

```
<ul>  
<li>An item in my  
list</li>  
<li>Another thing  
in my list</li>  
</ul>
```

Ordered Lists

- Another list is the ordered list
 - It uses the tags `` and ``
 - The items are enclosed in list item tags, `` and ``
- You can make different types of ordered lists:
http://www.w3schools.com/tags/att_ol_type.asp

```
<ol>
<li>An item in
my list</li>
<li>Another
thing in my
list</li>
</ol>
```

Nested Lists

- You can also have a list within a list, i.e. *nested* lists
 - Make a sublist within an item's `..` tag in the main list
- Notice that sublists can use a different bullet symbols

```
<ul>  
<li>Choice A</li>  
<li>Choice B  
  <ul>  
    <li>Sub 1</li>  
    <li>Sub 2</li>  
  </ul>  
</li>  
</ul>
```

Row 1, column 1	Row 1, column 2	Row 1, column 3
Row 2, column 1	Row 2, column 2	Row 2, column 3

Presenting tabular data

TABLES

```
<table>
  <tr>
    <td>Row 1, column 1</td>
    <td>Row 1, column 2</td>
    <td>Row 1, column 3</td>
  </tr>
  <tr>
    <td>Row 2, column 1</td>
    <td>Row 2, column 2</td>
    <td>Row 2, column 3</td>
  </tr>
</table>
```

Basic Table Tags

- The **table** is enclosed in `<table>` and `</table>`
- Each **row** is enclosed in `<tr>` and `</tr>`
 - **Cells** are surrounded by table data tags, `<td>` and `</td>`
 - A **heading** row may be created by using `<th>` and `</th>` instead of the `<td>` tag

```
<table>
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
</tr>
<tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
</tr>
<tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
</tr>
</table>
```

Simple Table Example

1	2	3
4	5	6
7	8	9

```
<table border="1">
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
```

The border attribute is deprecated; a better CSS approach next week!

More Complex Tables

- Table cells can *span across* more than one column or row.
- The attributes COLSPAN ("how many across") and ROWSPAN ("how many down") indicate how many columns or rows a cell should take up.

Colspan	

	Rowspan

Colspan Example

Merged	
Third Cell	Fourth Cell

```
<table>
<tr>
  <td
    colspan="2">Merged</td>
</tr>
<tr>
  <td>Third Cell</td>
  <td>Fourth Cell</td>
</tr>
</table>
```

Rowspan Example

First Cell	Merged
Third Cell	

```
<table>
<tr>
<td>First Cell</td>
<td
rowspan="2">Merged</td>
</tr>
<tr>
<td>Third Cell</td>
</tr>
</table>
```

1	2	3
4	5	6
7	8	9

1	2	
3	4	5
6	7	

Table Exercises!

	2	3
1		5
6	4	7

	2	
1	5	
4		3

```

<table border="1">
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>

```

1	2	3
4	5	6
7	8	9

1	2	
3	4	5
6	7	

1	2	3
	4	5
6		7

	2	
1	5	3
4		

```

<table border="1">
<tr>
<td rowspan="2">1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td rowspan="2">4</td>
<td>5</td>
<td>6</td><td>7</td>
</tr>
</table>

```

```

<table border="1">
<tr>
<td colspan="2">1</td><td>2</td>
</tr>
<tr>
<td>3</td><td>4</td><td>5</td>
</tr>
<tr>
<td>6</td><td colspan="2">7</td>
</tr>
</table>

```

```

<table border="1">
<tr>
<td rowspan="2">1</td>
<td colspan="2">2</td>
</tr>
<tr>
<td>5</td><td rowspan="2">3</td>
</tr>
<tr>
<td colspan="2">4</td>
</tr>
</table>

```

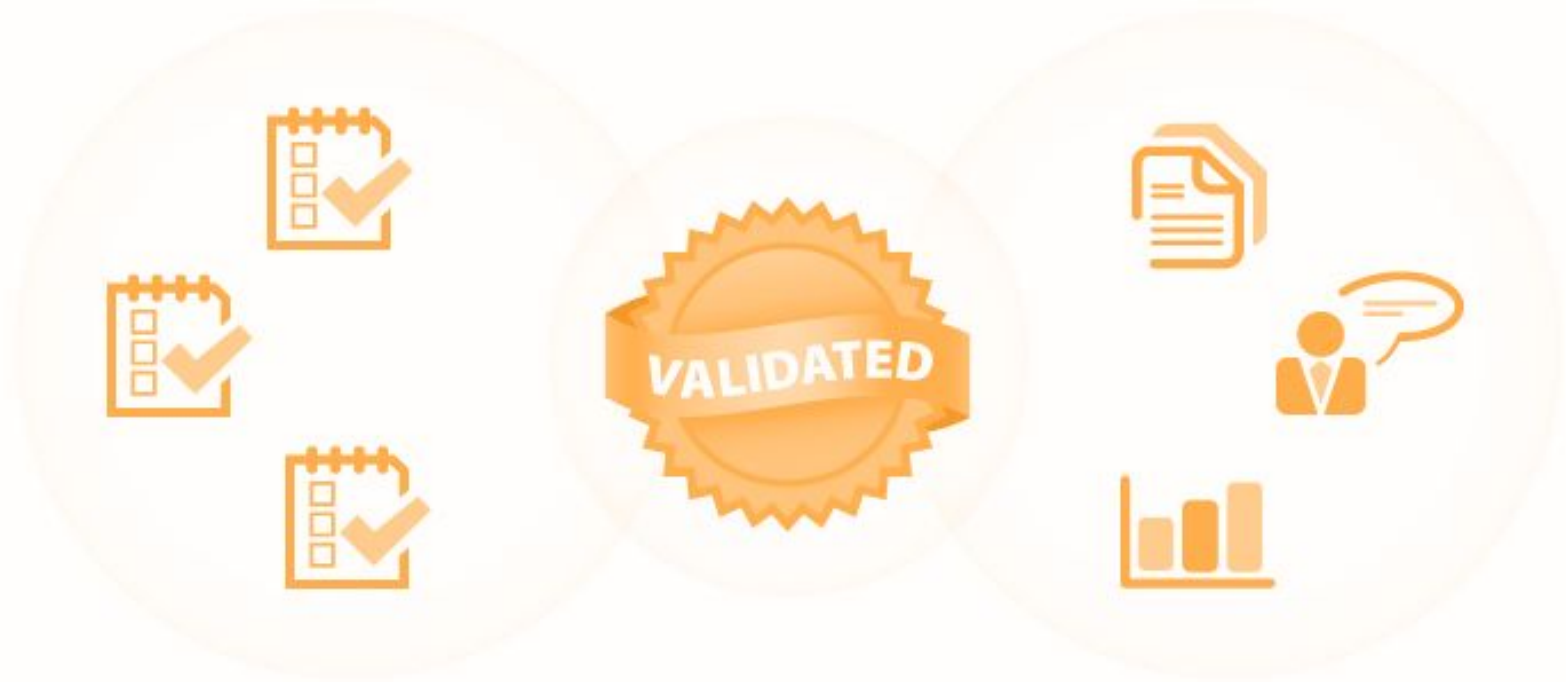
<break!>

Activity: More Practice

Let's live code together and create a new page.

Once again, we'll incorporate some kitten pix.....

Get your coding environment ready with a new, empty file saved as index.html in a separate folder from where you have done any small coding work



Checking your work!

MARKUP VALIDATION

Markup Validation Service

- This service checks to make sure your HTML is correct and identifies any errors.
 - You'll be validating your files for the web assignment
- Within your <head> tag, make sure you have this line:

```
<meta charset="UTF-8">
```

- This code specifies that the character encoding for the Web page will be UTF-8, or Unicode Translation Format for bytes
- Now, go to: <https://validator.nu/#textarea>
 - Copy-paste or Choose the "File Upload" option, upload your file and hit validate!

Upcoming: Creating A Site via Multiple Files

A website is simply an organized collection of HTML pages that are linked to each other through some form of navigational links

The *index.html* page is displayed by default if no other page is specified so is commonly used as the homepage

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8"> </head>
<body>
Welcome to my website!
<a href="resume.html">See my
resume</a>
</body>
</html>
```

resume.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8"> </head>
<body>
My Resume
<a href="index.html">Back to
home</a>
</body>
</html>
```

Upcoming: HTML Calling in Resources (that “master of ceremonies” role...!)

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <title>My cool page</title>
```

```
  </head>
```

```
  <body>
```

```
    The main content of the page goes here
```

```
  </body>
```

```
</html>
```

we will add more to the <head> section to call in resources like our CSS style file, font library APIs, and code libraries

in the appropriate places, near the end of the <body>, we will add JavaScript script file calls

Best Practices - Code Separation

Though the three coding languages, HTML, CSS, and JavaScript, can be configured to work together from within files, the best practice development strategy is to store each in separate files with the appropriate file extension on a website server to generate the web pages, e.g.:

index.html
style.css
script.js

In later sessions you'll see how the CSS and JavaScript are triggered via the HTML to join in the web page creation process.

Note that other resources needed by the website, such as image files and audio files, would typically be stored in a subfolder within the same folder directory environment with a name such as "assets." We'll touch on file organization more in later weeks of the course.

Tips: Debugging Strategies

Throughout this course's work, if you don't see the expected results....

→ Check your code “anatomy”!

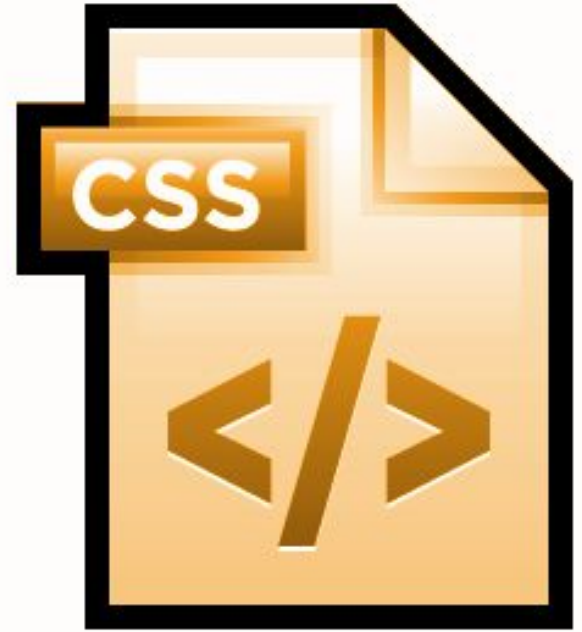
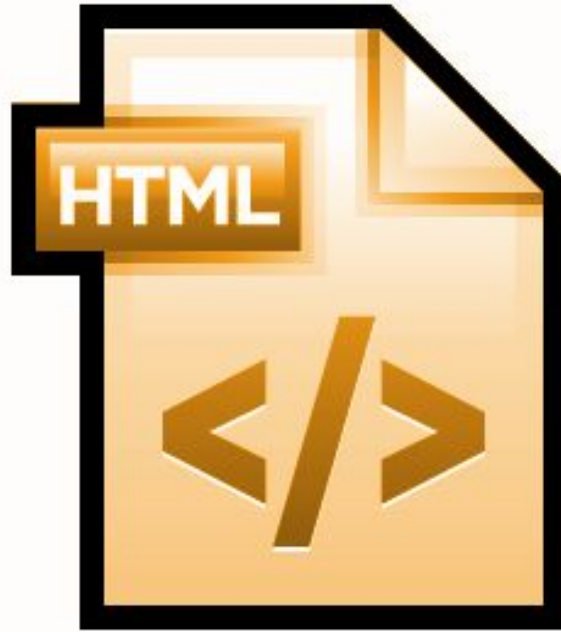
are you missing any start or end tags?

is every single necessary punctuation mark in place?

→ Check your code “physiology”!

is everything in the right section and in the right order?

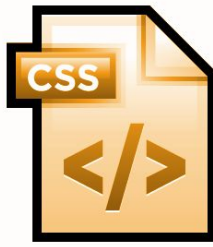
do you have all of the elements at the top/in the head that are required by the later items in order for the code to run?



Assignment description

WEB FUNDAMENTALS PROJECT

Web Fundamentals Project



- This assignment will give you more experience in:
 - Working with HTML & CSS
 - Making your personal/professional site or another site focus of your choice (see examples in full description on Canvas)
 - Host on:
 - Pratt's mysite web-space (exists this semester only)
 - GitHub using GitHub Pages (exists as long as you want!)
 - we'll go over hosting options next week w/Tk -- **be here 10 mins early**
- Due Nov 1 via Canvas (the link to your live site and your code files) by the start of class
 - Full description posted on Canvas

Web Fundamentals Project: Topic Options

“The specific content that you include is up to you, but you may include:

- work experience,
- areas of expertise or interest,
- personal information,
- career goals,
- hobbies,
- pet profiles,
- “biography” of an object,
- information technologies memory diary,
- online multimedia short story,

etc. “

examples:

- *a simple professional profile/resume site*
- *a **prototype** professional profile /resume site, for gathering content and designs you plan to launch on a CMS*
- *a site done as a project reflection tracking the process of working on a final project for another class*
- *a site focusing on your non-degree creative work*
- *a site highlighting any of the personal interests at left, or others!*

More Places to Practice

- **Codecademy web track** (free portions)
 - <https://www.codecademy.com/tracks/web>
 - Many other tutorials on various web and programming topics
- **LinkedIn Learning** (formerly Lynda.com) – tech training & tutorials (free use through Pratt libraries)
 - see link on Canvas or via Pratt's library site (use Pratt credentials to access account creation page)
 - "HTML Essential Training" and "CSS Fundamentals" are applicable to our work
- **W3schools HTML5 tutorial**
 - <http://www.w3schools.com/html/>
 - Short, concise pages on each tag/technique with examples
 - Approach with a critical eye as some code samples are older and not to current accessibility standards
- **Mozilla Development Network (MDN)**
 - https://developer.mozilla.org/en-US/docs/Web/HTML#beginners_tutorials

coming up!

(the website dance continues...)

Coming Up...



- Next week: Cascading Style Sheets (CSS)
 - aka, “meet the Art Director!”
 - Adding some style to our HTML
 - Looking at hosting options for web projects
 - freeCodeCamp tutorial homework will continue into CSS
 - **TECH TUTOR Tk joins us 10-15 minutes BEFORE start time -- please come early!**
- Get started on the Web Basics Project:
 1. Spend time thinking about **the topic/subject** of your site
 2. Start brainstorming potential ***designs and structures*** for your site (“design ideation”), preferably using visual tools such as paper-and-pen or a digital mockup tool--e.g. start sketching out a *sitemap* and content you'd like on each *page(s)*
 3. Create your web development environment at home!
Install a code editor to work with on your primary/home computer, e.g. VS Code or Sublime Text or Atom or BBEdit

have a great week!
