# More about HTML

Digging in a little deeper

# Structural v. Semantic Markup

- *Structural* markup is using to encode information about the *structure* of a document. Examples: headings, sections, paragraphs, footers

- *Semantic* markup means using tags that *add meaning* to the content. Examples: citations, acronyms, quoted text, lists, emphasized text

Structural markup such as <h1> and <p> and  delineate major sections of the document and are also *block-level* elements that sit on their own line.

Semantic markup such as <cite> are *inline* elements that add extra information without altering the structure of a document.

R·I·T

# An Example

- Which tags are *structural*, and which are *semantic*?:

```
<html>
   <head>
     <title>Joke Page</title>
   </head>
   <body>
      <h1>My Favorite Joke</h1>
      <p>What did the fish say when he <em>bumped</em>
                                          his
      <strong>head</strong>?</p>
      <p><strong>Dam!</strong></p>
   </body>
</html>
```

# An Example

- Structure and semantics?
  - <p> – This is a paragraph of information
  - <h1> – This is a heading (in this case, heading level 1)
  -  – This information should be emphasized
  -  – Emphasize this information as strongly as possible
- Notice how each of these tags gives extra information about the text they enclose.

# So… What about Presentation?

- *Semantic* markup gives *meaning* to the various pieces of content on a web page.
- *Semantic* markup does *not* directly define how the content will appear
- Defining how a page will display in a view is done using *Styles*
- *Styles* "connect" to the semantic markup to define how an item will appear

# A Brief Style Example

```
<html>
   <head>
        <title>A Page with Style</title>
        <style type="text/css">
               p {color: blue;
                  font-family: Arial, san-serif;
                  font-size: 24pt;
               }
        </style>
   </head>
   <body>
        <h1>A style example</h1>
        <p>Because of the style definition, any text in a
   paragraph tag will display in blue, using an Arial font
   with sized at 24 points tall.</p>
        </body>
</html>
```

# More on Styles

- Styles give us extensive control over the *presentation* of our page's content
- Styles also allow us to keep the *presentation* of the information separate from the actual *content*
- We will discuss styles in detail as we continue in this course

# More Semantic Markup Tags
# Unordered Lists

- Lists allow us to present information in a structured and ordered method
- The most common list is an unordered list. This frequently appears like a traditional "bullet" list
- Unordered lists have two parts:
  - An outer set of tags that indicate that this content is a list of information (the <ul> and </ul> tags)
  - An inner set of tags that define *each* item in the list (the <li> and </li> tags)

# Unordered Lists

- For example:

```
…
<ul>
  <li>Apples</li>
  <li>Pears</li>
  <li>Cherries</li>
</ul>
…
```

- Would display as:

  - Apples
  - Pears
  - Cherries

# Ordered Lists

- Ordered lists are similar to unordered lists
- Ordered lists normally display using numbers instead of "bullet" characters
- For an ordered list, you use the <ol> and </ol> tags in place of the <ul> and </ul> tags
- Like the unordered list, you use the <li> and </li> tags to indicate an item in the list

# Ordered Lists

- For example:

```
…
<ol>
  <li>Apples</li>
  <li>Pears</li>
  <li>Cherries</li>
</ol>
…
```

- Would display as:

1. Apples
2. Pears
3. Cherries

# Formatting a List

- The default for an *unordered* list is a bullet list
- The default for an *ordered* list is a numbered list
- How the numbers or bullets appear – even *if* they appear – can all be controlled using styles
- Lists are a very powerful tool in your web design arsenal. Later in this course, we'll show you how to combine lists with CSS styles to create a tabbed navigation system

# HTML Comments

```
<!-- this is an HTML comment -->
```

- Note the double dashes and spaces
- Won't validate if not formatted just right
- Use them and your HTML whitespace for readability!

# More Structure…

- 
  - “Logical division”
  - Block-level
  - Used for breaking your content into discrete chunks, structurally. Very useful later on.
- 
  - Inline
  - Creates a section of content that spans some smaller piece of a larger chunk, like a <p>

# More structure…

- HTML5 has introduced a bunch of structural tags for further blocking of content
  - <main>
  - <article>
  - <aside>
  - <figure>
  - <caption>
  - <header>
  - <footer>
  - <nav>
- Use them as they make sense!

# Hypertext links

- So far, we've only been working with a single web page
- A web *site*, however, is a collection of web pages that relate to each other
- To "jump" from a page to another page requires the use of an *anchor* tag which will create a hypertext link. Here's an example that links to a web site on another server:

```
<a href="http://www.rit.edu">RIT Main Page</a>
```

- Note that if the page is on another server, we need to use the http:// *protocol*.
- The above URL to RIT's home page is an *absolute* URL

# Connecting with Other Web Pages

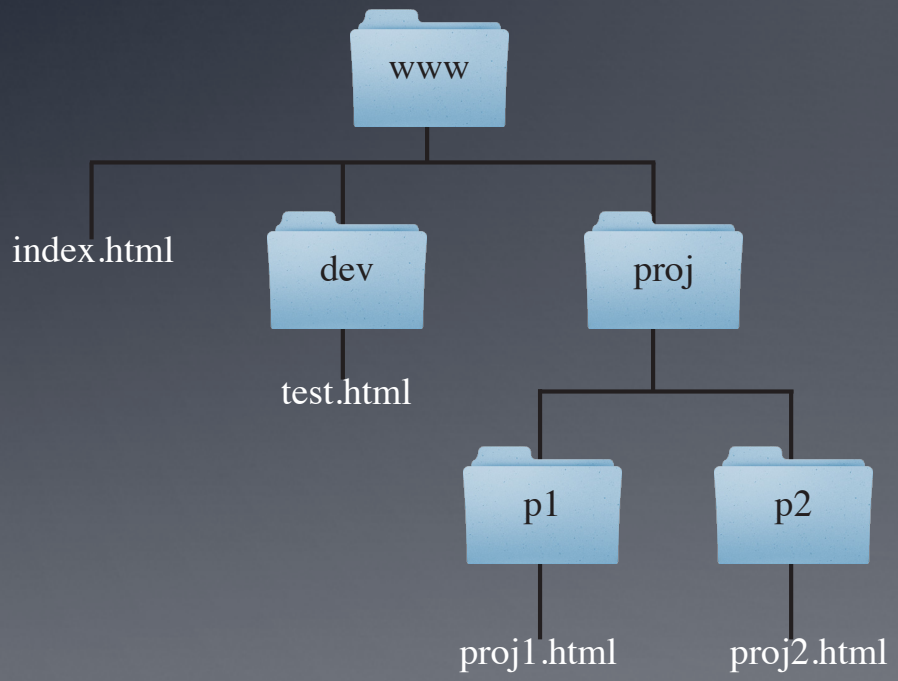
- The *anchor* tag is very easy to use if you are connecting with pages on *other* servers.
- Your web site, however, will most likely have many pages all residing on the same server in a *directory structure*
- Learning how to specify documents in directories on a Unix – or, really, *any* server – requires that you understand the idea of *relative addressing*
- Addresses to files are also called *paths*

# Relative addressing to a file in the *same folder*

- This first case is the simplest. If a file is in the same folder, we can link to it just by using the file name.
- Example: if we have a page named `proj1.html`, and we want to link it to a page named `bio.html` that is located in the same folder, we would use the following HTML:

```
<a href="bio.html">Bio Page</a>
```

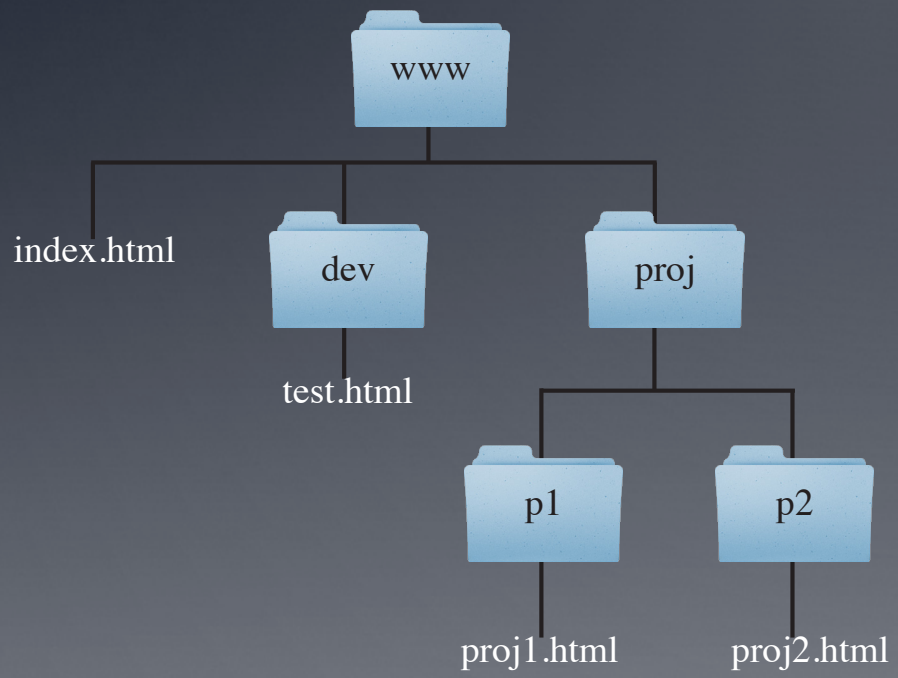
# Relative addressing to a file in a *sub-folder*




- For this next case, we want to link to a file that is located several folders "down"
- If we want to link `index.html` to `proj1.html` we will need to include the names of the folders in our path.
- See HTML below:

```
<a href="proj/p1/proj1.html">P1</a>
```

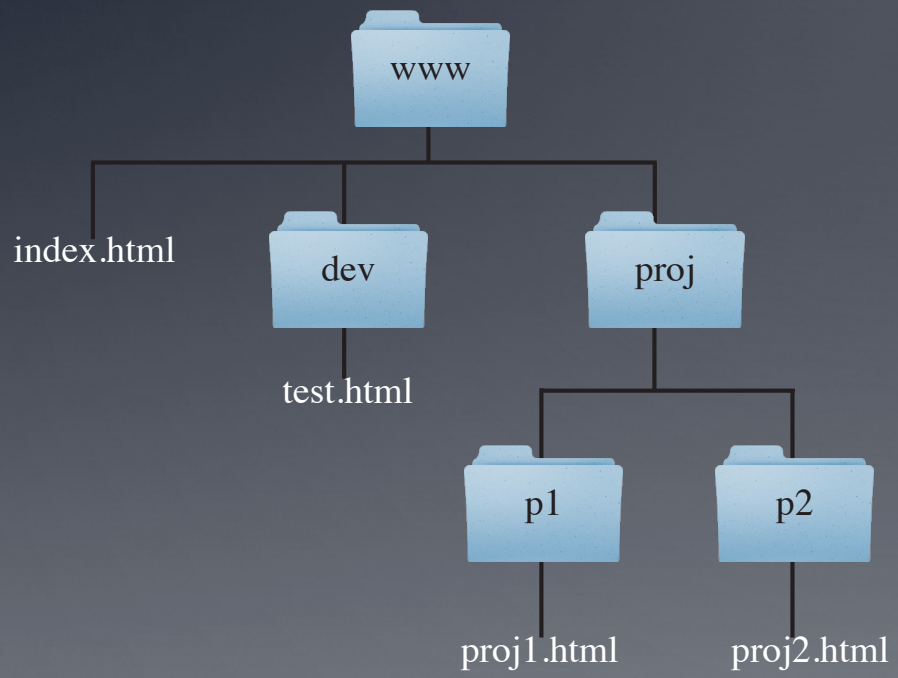
# Relative addressing to a file that is located "up" one or more levels




- For this next case, we want to link `proj1.html` to `index.html`, which is located two folders "up"
- To move "up" one level, we use "`..`" in our file path. To move up two levels, we use "`../../`"
- See HTML below:

```
<a href="../../index.html">P1</a>
```

# Relative addressing to a file that is located in a "sibling" directory




- For this next case, we want to link `proj1.html` to `test.html`, which is located two folders "up", and one folder "down"
- To move "up" the 2 levels, we use "`../../`" in our file path. To move back down, we use the name of the folder "`dev/`", followed by the name of the file we want to link to.
- See HTML below:

```
<a href="../../dev/test.html">P1</a>
```

R·I·T

# Some path notes

- Don't start a path with /
  - This goes to the *root* (usually WAY above your user account)
- Remember rules for filenames in your paths
  - No spaces
  - No special characters (?, !, *, etc.)
  - Avoid caps (case-sensitive)

# And now…

- Do the ICE!

R·I·T