WORKSHEET 1: BUILDING A WEBSITE WITH HTML AND CSS

Getting Started

1) Create a directory (folder) in a convenient place on your computer and call it XML-workshop.

2) Download the following files from the location on the board and save them to your XML-workshop folder.

template.html
style.css
info.txt
The CaseFiles directory and all of its contents
wellformed-quiz.pdf

3) Download and install the Atom text editor. For Windows, go to atom.io. For others, installation instructions can be found here: https://atom.io/docs/v0.194.0/getting-started-installing-atom

Writing the html

The Template

Open template.html with Atom.

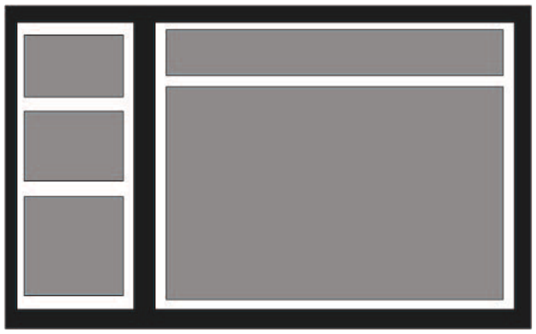
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="utf-8" />
5  </head>
6  <body>
7  </body>
8  </html>
9
```

What we have here is a basic template for an html page. The basic features are here for a well-formed XML document. Lines 2 and 8 are the tags for the root element, <html>, which will contain all other elements, and all elements are nested and closed properly. (Lines 1 and 4 are giving the browser helpful information, but these are topics for another time.)

Additionally, this is valid html, as every html must have a <head> element and a <body> element, in that order, and these must be the only children of the html root element. Lines 3 and 5 bound the head of the html document. (Note that this not the header you might see at the top of a webpage, but rather a section to declare metadata, links, and other information about the webpage.) Lines 6 and 7 are the body. What we put here will show up on the page in your browser, and this is where we will put most of our content.

Adding Div Elements

The final website will have five regions, grouped into two vertical regions, as shown in the following illustration.



Each of these blocks is bounded by a <div> element, as follows:

```
<body>
     <div></div>
     <div></div>
     <div></div>
     <div></div>
     <div></div>
</body>
```

Hint: Be sure to copy the syntax exactly (e.g. don't forget the forward slash in the closing tags.)

Next, we will collect this five divs into two groups, one for each vertical column of the final layout, again using div tags. Often, html pages will have many layers of div elements.

```
<body>
     <div>
          <div></div>
          <div></div>
          <div></div>
     </div>
     <div>
          <div></div>
          <div></div>
     </div>
</body>
```

Exercise: Working with the participants around you, draw a tree of your html document.

Labeling our divs with id attributes

We are going to distinguish each of the div elements with its own unique id attribute, which we give a value based on the position on the page.

```
<body>
     <div id="leftPanel">
          <div id="title"></div>
          <div id="menu"></div>
          <div id="info"></div>
     </div>
     <div id="main">
          <div id="header"></div>
          <div id="container"></div>
     </div>
```

```
</body>
```

We might think of these like labels on a box of in the archive. If we asked someone to retrieve such a box, we would need to provide a call number or some other ID. The computer works in a similar way.

N.b. in valid html, @id values can be just about anything, but they must be unique; no two elements may have the same one.

Adding Text

Now we will add some text to the website.

So far, nothing we have done will show up on the site, and we'll have a blank browser. But once we add some text as the *content* of div elements, we will have something to show on the page.

```
<body>
     <div id="leftPanel">
          <div id="title">TITLE</div>
          <div id="menu">MENU OF OTHER PAGES</div>
          <div id="info">INFORMATION</div>
     </div>
     <div id="main">
          <div id="header">HEAD MENU</div>
          <div id="container">CONTENT</div>
     </div>
</body>
```

The text we've added here is just for place-holding. Later, we'll replace it with better stuff.

Peer review: With your neighbors, check the syntax of these div elements and their attributes. Do your files match? *Hint: use the color coding of the text editor to help find errors.*

Save your page (windows shortcut: ctrl+S)
View your page by opening it in a web browser (Firefox is recommended – I can’t guarantee that the page will look the same in other browsers.)

Q: How are the sections of text arranged on the page, compared with what you expected? Why? What is missing?

[template2.html]

A Title for the Page

Fill in the text of the div with the id="title", deleting the place-holder text, as follows:

```
<div id="title">Pirates of the Old Bailey ...an exercise in
     html</div>
```

Save and view your page. *Hint: In order to view changes, you'll need to reload the browser after each save.*

So far, our text all looks basically the same. Now we will use an <h1> element to identify this text as a kind of header, and the browser will notice that this text is different than the rest.

Enclose the title in <h1> tags.

```
     <div id="title"><h1>Pirates of the Old Bailey ...an exercise in
          html</h1></div>
```

Save and view your page. What changed?

Replace the <h1> with <p> (paragraph) tags, then save and view the page again. What changed? How is it different now than what we had before adding the <h1> tags?

Let's adjust our title once more. Enclose the phrase "Pirates of the Old Bailey" with <h1> tags, and "…an exercise in html" with <p> tags.

Peer review: Does your html match with your neighbors?

Save and view your page.

While we are on the subject of titles, we are going to add <title> element to the html <head>, like this:

```
     <head>
          <meta charset="utf-8" />
          <title>Pirates from the Archives</title>
     </head>
```

Save and view your page. Recall that the html <head> is for information *about* the page, rather than page content. Where does this title appear? Where else do you think it might appear? *Hint: Do a Google search for Old Bailey Online. What does it have for <title>?*

Now that we've seen the difference, let's change our title to match our heading:

```
     <title>Pirates of the Old Bailey ...an exercise in html</title>
```

<u>Adding some more text</u>

In our info section, we'll add a couple of paragraphs about the site. In the div with the id="info", replace the place-holder text with the following two paragraphs, which can be copy-pasted from the info.txt file.

```
     This website was created by {insert your name here} for the
          workshop "An Introduction to XML for Historical and
          Literary Research," a two-part series offered at D-Lab,
          April 19 and 26, 2016.
```

```
Materials from the Old Bailey Online and its partners are used
     for educational purposes only. All other parts of this site
     may be shared freely.
```

Nest each of these paragraphs in a <p> element. Now, inside div@id="info", you should have two <p> elements, each with text content.

Save and view your page. *Hint: you might have gotten a strange line break in your html file, in the middle of the paragraph of text. Save and view your page anyway. What happened to the paragraph?*

Adding Hyperlinks

Now that we have some text, we have something to hyperlink. Let's add a link to the mention of the workshop name in the paragraph text, using the <a> (anchor) element:

```
<a>"An Introduction to XML for Historical and Literary
     Research,"</a>
```

If you look at the page now, you won't see any change. Firefox is ignoring the <a> tags, because the address of the link hasn't been declared. Let's tell Firefox where you want the link to go. For this, we use the href attribute, and give it the value of the destination URL (Universal Resource Locator—a web address or location on your computer where a file can be found). Here is the attribute-value pair that you want to use:

```
href="http://dlab.berkeley.edu/training/introduction-xml-
     historical-and-literary-research-2-part-series"
```

You can copy-paste the URL from the file info.txt (but not from the pdf!), but if you are feeling extra patient, go ahead type it out by hand. Reading URLs is a good skill to have!

Save and view your page.

Voila! a link. But if you are like me, you might be annoyed that the underlining from the link extends to the punctuation. With your neighbors, see if you can fix it so that the comma and quotes are not underlined.

Next, using what you've just learned, let's add a second link—this time to the mention of the Old Bailey Online. The URL is in info.txt.

Finally, let's add an ampersand to the dates: April 19 & 26, 2016. It turns out this is tricky, because & is a *reserved character* in html. Just like the less-than symbol:

```
<
```

which indicates the start of a tag, when the browser encounters an ampersand, it interprets it not as plain text but as a part of the encoding (called an entity). In this case

```
&
```

indicates that the following characters are code for a special character. For example, in order to render a less-than sign, we use the special code

```
      &lt;
```

All such codes begin with an ampersand and end with a semi-colon. To render an ampersand, the code is

```
      &amp;
```

Go ahead and insert that entity into your date, so it looks like this:

```
      April 19 &amp; 26, 2016
```

Save and view your page.

Creating Lists

There are two types of lists in html, ordered lists <ol></ol> and unordered lists <ul></ul>. In each case, the items of a list are further encoded with <li></li> or ‘list item’ tags. First, we’ll make an ordered list with five items, which should look like this:

```
      <ol>
            <li>Documents</li>
            <li>People</li>
            <li>Maps</li>
            <li>Timeline</li>
            <li>Index</li>
      </ol>
```

Add this list to the site as a child of div#menu, replacing the place-holder text "MENU OF OTHER PAGES." *Hint: div#menu is shorthand for the div element with the attribute id="menu".*

Save and view your page.

Next, change your ol tags to ul and check the difference. (We'll leave the list unordered.)

Now, let's enclose each of these five items in anchor tags, but not yet specifying the URL. The first should look like this:

```
      <li><a href="">Documents</a></li>
```

We don't yet have a URL for these pages, so we'll have to leave the href attribute without a value. But take a look at the page. How does this differ from what happened earlier, when our <a> element had no attribute?

A Note on Best Practices: Nesting and Display

Now might be a good time to think about displaying the nesting of your page. In xml and html, the general rule is this:

When an element is **contained by** another element and the two are written on separate lines, the child should be offset to the right of its parent.

For example, see the above markup for our list, where the list item <li> is contained by the unordered list <ol> element, and so we've offset the <li> to the right.

Note: In xml and its applications, this display does not matter to the computer at all. But in certain languages (e.g. python), it does matter. And, in general, it is a great practice because it helps you visualize the structure of the file and it helps with troubleshooting when there is a problem.

Use the Tab key to replicate the "nesting" of elements seen in the following model:

```
<div>
     <ol>
          <li> </li>
          <li> </li>
          <li> </li>
          <li> </li>
          <li> </li>
     </ol>
</div>
```

Hint: In many text editors, Tab will move a line to the right, and Shift+Tab will move a line back to the left. (In Atom, the line might have to be highlighted first, depending on the position of the cursor.)
D:\academics\Course and Workshop Materials\Workshops\XML\Pirates of the Old Bailey\pirates.html
Now extend that logic to the whole document, using Tab and Shift+Tab to arrange the elements so that the way they nest is implied by the visual display of the html document in the text editor.

Peer Review: Check with your neighbors. Are their documents nested like yours?

[template3.html]

More Lists!

Our page is going to have two more lists. And this time, one is going to be nested inside another. In div#header (the div with the id="header"), we will add the following nested lists. But first, take a moment to look over this mass of code, and draw it out as a tree, treating the first <ul> as the root element.

```
<ul>
     <li><a href="pirates.html">Home</a></li>
     <li>Case List
          <ul>
               <li><a href="">Richard Coyle</a></li>
               <li><a href="">Edward Johnson, et al.</a></li>
```

```
                <li><a href="">John Mackarel</a></li>
                <li><a href="">George White and William
            Burwood</a></li>
                <li><a href="">Levi Thomas and William
            M'Clive</a></li>
                <li><a href="">James Jennings Smith, et
            al.</a></li>
                <li><a href="">Isaac Cooper, et al.</a></li>
            </ul>
        </li>
        <li><a href="">About</a></li>
    </ul>
```

Once you feel comfortable with this as a tree, go ahead and enter it all inside div#header.

Adding @class attributes

In addition to @id attributes, we can label our html elements with @class attributes. Recall that an @id must be unique—no two elements may have the same one. With @class, we can give multiple elements the same label, so the computer knows to treat them as the same kind. If @id is like a call number, unique to a given work, we might think of @class as a category like ‘books’ ‘journal articles’ or ‘dvds’ in a library catalog.

We are going to label each list by giving a @class to its <ul> element.
1) Under div#menu, give <ul> the attribute class="nav-menu".
2) Under div#header, give the first ul the attribute class="top-menu".
3) For the embedded list, give its <ul> the attribute class="sub-menu".

Note: Here, it so happens that each @class has a unique value, so we very well might have used @id instead, and it would amount to the same thing. However, in a later example, we'll use @class attributes that repeat and therefore could not be exchanged for @id.

Save and view your page.

[template4.html]

The Main Content: Subsections

The main section of the page is going to contain eight sections, seven of which follow the same pattern. First, we will create eight <div> elements, as children of div#container, replacing the place-holder text. Each one should have the attribute class="entry", and look like this:

```
    <div class="entry"></div>
```

Write eight of these in your code, one after the other.

Next, we will give all but the last of these <div> elements two children, an <a> and a <div>, like this:

```
<div class="entry">
     <a class="post-thumbnail" href=""></a>
     <div class="entry-text"></div>
</div>
```

Leave the eighth one empty. You should now have seven identical sections under #container. Check with your neighbor.

Save your page.

<u>Adding images</u>

To add an image to an html page, we use the self-closing <img/> element, and we point to its URL with the attribute @src (source). In this case, our URLs are on a relative path in our own filesystem, and the information can be found in info.txt. (Be sure that the CaseFiles directory is stored in the same place as your html file, or the reference won't work.) Here is what the first case will look like:

```
<img src="CaseFiles/coyle-piracy1.gif" />
```

For this page, we will embed the image in the hyperlink in our div.entry like this:
Hint: div.entry is shorthand for the div element with the attribute class="entry". Dot for @class, hashtag for @id.

```
<div class="entry">
     <a class="post-thumbnail" href="">
          <img src="CaseFiles/coyle-piracy1.gif" />
     </a>
     <div class="entry-text"></div>
</div>
```

Following this pattern, add an image for each of the seven cases, using the URLs provided in info.txt

Next, we'll add some content to our div.entry-text, giving it an <h2> header and a <p> paragraph, all with @class attributes. Here is how it will look, with the content from info.txt:

```
<div class="entry">
     <a class="post-thumbnail" href="">
          <img src="CaseFiles/coyle-piracy1.gif" />
     </a>
     <div class="entry-text">
          <h2 class="entry-title">Richard Coyle</h2>
          <p class="entry-content">24th February 1737</p>
     </div>
</div>
```

Save and view your page.

[template5.html]

Following that same pattern, fill in the template for the remaining six cases with the information in info.txt. A couple of cases have an extra line of information, for which you can use a second <p> element.

Finally, to the eighth div.entry (currently empty), add the value "empty" to the @class attribute, separated by a blank space, so it looks like this:

```
	<div class="entry empty"></div>
```

Save and view your page. Check your page and code with your neighbors. Do you see seven images, with headers and descriptions?

And that's the end of our content, but wow! … that's one ugly page.

Now for the Magic!

Now we'll connect our CSS stylesheet to the webpage, which will *dramatically* alter its appearance.

In the <head> of the html document, add the following line:

```
	<link rel="stylesheet" type="text/css" href="style.css" />
```

Hint: Just like the images, this is a relative URL, so the file must be in the same directory as the html file for this link to work.

Save and view your page.

Question**:** Can you identify the changes that CSS made to the page? There are a lot. Compare your answers with your neighbors.

Next week, we'll get into the basics of CSS, and we'll take a look at two important XML applications: TEI and KML.