**Creating Web Pages**

A *web page* is a combination of text and formatting directives that adhere to a standard called HTML (Hypertext Markup Language) – see <http://www.w3.org/TR/html4/> for details. A *web site* is a collection of inter-linked web pages. To create publically available web pages you need:

1. An editor program optimized for editing HTML. Any plain text editor will do.
2. An account on a web server
3. A means to copy files onto or create/edit files on that web server

**HTML Overview**

An HTML file consists of *text* and *tags*. Tags specify how to format and layout text and other media. Tags are enclosed in angular brackets, have a name and optionally one or more parameters (called attributes) with values enclosed in single or double-quotes. Tags usually have an “end tag” with the same name prefaced by a slash / or otherwise are defined via < … />:

**<tagname attrib1=”option1” attrib2=”option2”>text</tagname>** or

**<tagname param=”option” />**

Tags can be properly nested. Tags and attributes are spelled with lowercase letters, all attributes should have values, and all attribute values should be quoted.

**Examples:**

<b> **is a tag to begin formatting text in bold,** </b> ends that formatting directive

<b><i>***is properly nested***</i></b> but <b><i>is not</b></i>

This is a tag with its own, build-in end tag: <br />

This is <a href=”text.html”>a tag with a parameter/attribute</a>

Here is a list of most common formatting tags:

**Structure tags:**

<html></html> defines beginning and end of the document. Mandatory

<head></head> defines the header section of a document to contain property info for the entire document, such as the document title

<body></body> defines the body, or content, portion of a document

<title></title> defines the document title

<meta …/> defines meta-information

<link …/> defines a link to a supporting file such as a CSS style sheet

<div></div> defines text as a block-level group

<span></span> defines text as an inline group

<!DOCTYPE …/> defines the document type (mandatory for XHTML files)

**Character Formatting**

<b></b> formats as bold

<i></i> formats as italics

<tt></tt> formats as type writer (mono-spaced)

<font></font> defines font face and size

**Paragraph Formatting**

<p></p> Defines a paragraph

<br /> A line break

<hr /> A horizontal ruler (line)

<blockquote> indents a block

</blockquote>

<pre></pre> Pre-formatted text

<h1></h1> headline level 1 (2, 3, …. 6)

**Lists**

<ul></ul> unsorted list with bullets

<ol></ol> ordered list (numbered)

<li></li> a list item for either list type

**Links (anchors) and images**

<a></a> an anchor or link

<img …/> in image

**Tables**

<table></table> defines a table

<th></th> defines a header (title) for the table

<tr></tr> defines a table row

<td></td> defines a table column

**Programming**

<script></script> Defines a function in a programming language

<applet></applet> Defines a Java applet

<embed></embed> Defines an embedded object

**Exercise:**

* Identify how <http://pirate.shu.edu/~wachsmut/Teaching/CSAS2213/index.html> is laid out.
* Create a simple page with a title, a main header with some paragraph, a sub header with a list of links, and an image. Add a centered table to your document

**Linking**

One detail that is important to understand – as it is *the* defining property of the Web - is how to link to other information. Information on the web is described via a “**Uniform Resource Locator**”, or URL. More recently URL has been superseded by the more general URI (Uniform Resource Identifier) but we’ll stick with URL. A URL has 4 parts (there is actually a 5th part which we’ll ignore):

protocol://hostname:port/directory/subdir/filename.ext

Some parts can be missing and will be deduced from the context:

href=”http://www.mathcs.org/analysis/reals/index.html”

a **fully qualified** (complete) **URL** using the default port for http (port 80)

href=”/analysis/reals/index.html

an **absolute URL** (no protocol or server name, starts with a slash): the protocol , server name, and port are inherited from the document containing the link and the directories start from the root level of the web server

href=”index.html”

a **relative URL** (no protocol, no server name, and does not start with a slash): protocol, server name, port, and directories are inherited from the document containing the link (could include a directory part but without leading slash, where .. would mean “up one level”)

URL’s show up most commonly as option to the **href** parameter of the **a** tag.

**Tag Options**

We just saw the **href** parameter for the anchor tag a. Actually, like many other tags, it has a number of additional, optional parameters to tailor its behavior and appearance.

The anchor tag is <a> and closes with </a>. Some parameters are:

href specifies a target URL

target specifies a named window where the target URL will be displayed

title specifies a title that shows up like a 'tool tip' when you hover over the link

onclick event handler for clicking the mouse on the link

The body tag is <body> and closes with </body>. Some parameters are:

background specify URL for tiled background image

text sets the foreground color for text (for visual browsers).

link sets the color of text marking unvisited hypertext links

vlink sets the color of text marking visited hypertext links

alink sets the color of text marking hypertext links when selected by the user

[bgcolor](http://www.w3.org/TR/html4/present/graphics.html#adef-bgcolor) sets background color

[onload](http://www.w3.org/TR/html4/interact/scripts.html#adef-onload) event handler when page is done loading

**About Colors**

Colors are most flexibly described via their red, green, and blue (RGB) value, where each value is an 8-bit integer, sometimes written in hexadecimal form. Some colors have ordinary names, but RGB values are very commonly used. The 8-bit color values are usually given as hexadecimal numbers.

**Exercise:**

Hexadecimal numbers are base-16, using the 16 symbols 0, 1, 2, …, 9, A, B, C, D, E, and F. 16 symbols use 4 bits, so an 8-bit hex value has two ‘digits’. Convert:

* A16 to decimal:
* 1216 to decimal:
* B216 to decimal:
* FF16 to decimal:
* 200 to hex:
* 128 to hex:

A typical RGB color would now be:

* #00ff00, which is:
* #ff0000, which is:
* #ffffff, which is:

**HTML versus XHTML**

XHTML is a stricter version of HTML, i.e. HTML + more structure. For our purposes we will only use XHTML even when we say HTML. In fact, some of the rules we described before were actually XHTML rules, but since XHTML is compatible with HTML you might as well start learning XHTML right from the start. In HTML, for example, the <br> tag would be fine, but in XHTML it is not: it’s missing an end tag and should be written as <br />. XHTML makes a document “well-formed” and much easier to interpret by browsers, which is especially important for browsers on mobile device with access to much fewer resources (memory, speed).

There is also a standard called XML, which is a generalized version of XHTML with arbitrary tags:

* XML is designed to describe and structure data
* HTML is designed to describe how data should be formatted
* XHTML is designed to structure data with tags that have a fixed meaning

An HTML document *should* start with a DOCTYPE declaration describing the standard to which the document adheres. An XHTML document *must* have a DOCTYPE declaration; in fact a minimum XHTML document must have the following structure:

<!DOCTYPE Doctype goes here>  
<html>  
<head>  
…

<title>Title goes here</title>  
</head>  
<body>  
…

</body>  
</html>

Currently the most common DOCTYPE declaration is (should be in one line):

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

|  |  |  |
| --- | --- | --- |
| *Valid HTML segment*  Some text  <ul>  <li>John Doe  <li>Jane Smith  </ul> | *XHTML equivalent*  <p>Some text</p>  <ul>  <li>John Doe</li>  <li>Jane Smith</li>  </ul> | *XML snippet*  <xml>  <address>  <name>John Doe</name>  <email>doe@forrest.com<email>  </address>  </xml> |

To check if a page you created is written in proper HTM, you can run it through the W3C validation service available at <http://validator.w3.org/>

**About Web Page Creation**

The basic HTML tags are simple enough and there really are not that many. If you watch their order so that they are properly nested and follow a few ground rules they are easy enough to use. The problems are:

1. many tags have many options
2. it is confusing to determine which tags to use
3. it is difficult to create visually stunning pages

The *first problem* is usually solved by using an HTML editor that lets you *select* options for a particular tag and provides hints as to their meaning.

The *second problem* is usually simplified by advanced planning:

* Be clear about the purpose of your page; what are you trying to accomplish
* Where do you want to direct attention to; how do you want to structure your page
* How would you like your page to look overall; color scheme, navigation, header, footer, logo – be consistent

The *third problem* is easiest to solve: *don’t try* to create visually stunning pages. You are not a graphic designer, you have only limited time, and your strength is structured thinking, not color choices. Instead of *visually stunning* create a *functional* page that is logically organized and has useful content. Form is important, no doubt, but functionality is more important. Think of Google: it has no glitz at all, only functionality.

**Style Sheets**

HTML has evolved into a mixed-up language: it defines logical structure-defining elements like lists, headlines, etc. together with pure formatting directives like <font>, <b>, and <i>. It is advantageous to separate those two:

* want to define logical structures for our content
* want to separately define formatting options for the structures
* want to be able to apply one set of formatting rules to multiple web pages for uniformity

That’s what (X)HTML + CSS (**Cascading Style Sheets**) accomplishes. Style sheets define how a text structure should look, whereas (X)HTML defines where the structures start and end and how they are related. Styles sheets are cascading, because you can attach more than one style sheet to one (X)HTML document and they will all cascade into one style. Styles can be defined in three different ways

* *external*: the styles are define in a separate file, usually with .css extension
* *internal*: the styles are defined inside the HTML file, usually in the header section
* *inline*: the style is defined inside an existing tag

The most useful one is an external style sheet, so we will stick with that for the most part.

Each style in a style sheet has three parts, a *selector*, one or more *properties* with one or more *values* each*,* following this syntax:

selector {

property1: value1 [value2 ...];

property2: value1 [value2 ...];

...

}

To attach a style sheet to an external XHTML document you use the <link> tag in an HTML file. To attach a style sheet called style.css you would add the following to the *head* section of an XHTML document:

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

Here is the framework for an XHTML – CSS document pair (the CSS style sheet is called style.css):

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Title goes here</title>

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

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

</head>

<body>

<p>Content goes here</p>

</body>

</html>

While you can create style sheets with any text editor, it is convenient to use a more powerful editor specializing in web page creation – for example Expression Web – check with the Help Desk to get Expression Web installed!

**Exercise:**

Create a new HTML document as well as a new style sheet and save the empty files to the same directory. Attach your new style sheet to your web page. Create a level-1 headline, a list, and a paragraph of text.

Define the following style, save everything, then open/reload your HTML document:

h1 {

border: 2px black solid;

}

**More about style sheets:**

Style sheets define three types of styles:

#id ID’s are frequently used to define large structures in an HTML document. Each id can be used only once in each HTML document

.class Classes are styles that can be reused and applied to different elements via a class parameter, such as <p class=”name”> … </p>. The class=”name” parameter can be added to any HTML element. Multiple classes can be applied to one element by separating them with spaces, as in <p class=”name1 name2”>

element Elements are used to redefine how existing HTML elements are to be formatted

Common style properties are fonts, alignment, borders, margins, paddings, and locations. Here is a list of some common CSS properties (taken from <http://www.htmldog.com/guides/>) but there are additional ones:

**Text and Fonts**

* [font](http://www.htmldog.com/reference/cssproperties/font/), [font-family](http://www.htmldog.com/reference/cssproperties/font-family/), [font-size](http://www.htmldog.com/reference/cssproperties/font-size/), [font-weight](http://www.htmldog.com/reference/cssproperties/font-weight/), [font-style](http://www.htmldog.com/reference/cssproperties/font-style/), [font-variant](http://www.htmldog.com/reference/cssproperties/font-variant/)
* [text-align](http://www.htmldog.com/reference/cssproperties/text-align/), [text-decoration](http://www.htmldog.com/reference/cssproperties/text-decoration/), [text-indent](http://www.htmldog.com/reference/cssproperties/text-indent/), [text-transform](http://www.htmldog.com/reference/cssproperties/text-transform/)

**Colors and Backgrounds**

* [color](http://www.htmldog.com/reference/cssproperties/color/)
* [background](http://www.htmldog.com/reference/cssproperties/background/), [background-color](http://www.htmldog.com/reference/cssproperties/background-color/), [background-image](http://www.htmldog.com/reference/cssproperties/background-image/), [background-repeat](http://www.htmldog.com/reference/cssproperties/background-repeat/), [background-position](http://www.htmldog.com/reference/cssproperties/background-position/), background-attachment

**The Box Model - dimensions, padding, margin and borders**

* [padding, padding-top, padding-right, padding-bottom, padding-left](http://www.htmldog.com/reference/cssproperties/padding/)
* [border, border-top, border-right, border-bottom, border-left](http://www.htmldog.com/reference/cssproperties/border/)
* [border-style, border-top-style, border-right-style, border-bottom-style, border-left-style](http://www.htmldog.com/reference/cssproperties/border-style/)
* [border-color, border-top-color, border-right-color, border-bottom-color, border-left-color](http://www.htmldog.com/reference/cssproperties/border-color/)
* [border-width, border-top-width, border-right-width, border-bottom-width, border-left-width](http://www.htmldog.com/reference/cssproperties/border-width/)
* [margin, margin-top, margin-right, margin-bottom, margin-left](http://www.htmldog.com/reference/cssproperties/margin/)
* [width](http://www.htmldog.com/reference/cssproperties/width/), [min-width](http://www.htmldog.com/reference/cssproperties/min-width/), max-width, [height](http://www.htmldog.com/reference/cssproperties/height/), [min-height](http://www.htmldog.com/reference/cssproperties/min-height/), [max-height](http://www.htmldog.com/reference/cssproperties/max-height/)

**Positioning and Display**

* [float](http://www.htmldog.com/reference/cssproperties/float/), [display](http://www.htmldog.com/reference/cssproperties/display/), [position](http://www.htmldog.com/reference/cssproperties/position/)

**Lists**

* [list-style](http://www.htmldog.com/reference/cssproperties/list-style/), [list-style-type](http://www.htmldog.com/reference/cssproperties/list-style-type/), [list-style-image](http://www.htmldog.com/reference/cssproperties/list-style-image/), [list-style-position](http://www.htmldog.com/reference/cssproperties/list-style-position/)

**Lengths, Sizes, and Percentages**

Many properties refer to sizes, measured in a variety of units such as:

* **em** (such as font-size: 2em) is the unit for the calculated size of a font. So "2em", for example, is two times the current default font size
* **px** (such as font-size: 12px) is the unit for pixels
* **pt** (such as font-size: 12pt) is the unit for points
* **%** (such as font-size: 80%) is the unit for percentages

When a value is zero, you do not need to state a unit. The most useful units for font-sizes are 'em' or '%', whereas 'px' should be used for border sizes.

**Colors**

Colors are specified as name, an rgb (red/green/blue) value or a hex code. The following colors are all the same:

* rgb(255,0,0)
* rgb(100%,0%,0%)
* #ff0000
* red

Not all colors go well together, but there are several sites where you can pick appealing, matching colors. Try for example <http://www.colorjack.com/>

**Exercise:**

Change the header style to grey, large, a nice font, more margins, and borders only at the bottom and right side. Also define a class style to center text and apply it to a paragraph of text as well as the header.

We have seen selectors that redefine the HTML tag whose name they carry, and classes that can be applied multiple times. When are id selectors used? A common use for style sheets is to define overall segments for a page, such as a header, a footer, a navigational region, etc. That’s where id selectors come in. Create the following styles:

#container {

min-width: 600px;

}

#menu {

width: 200px;

float: left;

}

#content {

margin-left: 200px;

}

Save it and create a new HTML file. Attach the above style sheet, switch to code view and enter the following HTML tags (only the ones inside the body tag should be necessary):

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Welcome</title>

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

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

</head>

<body>

<div id="container">

<div id="menu">

<h1>The Menu</h1>

<ul>

<li><a href="">Menu 1</a></li>

<li><a href="">Menu 2</a></li>

</ul>

</div>

<div id="content">

<h1>Welcome</h1>

Here is the page content

</div>

</div>

</body>

</html>

Note that there is one XHTML violation –find it (hint: no text should be without tags).

Small changes can have big unexpected effects: In the above style sheet, change the “container” style to the following, then check the effect:

#container {

width: 900px;

margin: 0 auto;

}

HTML-element styles can be redefined inside different id sections so that they only apply to the HTML tag inside the id section: add the following styles to your style sheet, save it, then reload your HTML document:

#menu h1 {

font-size: 10px;

color: #000080;

}

#content h1 {

font-size: 18px;

color: white;

background-color: #505050;

}

That means that a header-1 tag inside a #menu id will be formatted one way, while the same header-1 tag inside the #content id will be formatted another way.

Now let’s be slick and define pseudo-buttons for our menu. Add the following styles without changing your HTML document:

#menu ul {

list-style: none;

display: block;

}

#menu li {

display: block;

}

The display property defines whether to display text in a line (display: inline) or stacked (display: block), while the list-style defines the style of the list (in this case no bullets).

Not much has changed, but the bullets have disappeared from the list items. Now we want to change colors so that the links change color when the mouse hovers over it. We will redefine the a tag (inside a menu id) as well as the “pseudo” class hover of the a selector. Add the following:

#menu a {

color: #ffffff;

background-color: #000080;

text-decoration: none;

}

#menu a:hover {

color: #000000;

background-color: #4040FF;

text-decoration: none;

}

CSS pseudo-classes are used to add special effects to some selectors. The most commonly used pseudo-classes are:

|  |
| --- |
| a:link {color:#FF0000} /\* unvisited link \*/  a:visited {color:#00FF00} /\* visited link \*/  a:hover {color:#FF00FF} /\* mouse over link \*/  a:active {color:#0000FF} /\* selected link \*/ |

Note: a:hover MUST come *after* a:link and a:visited in the CSS definition in order to be effective. For additional pseudo-classes, check <http://www.w3schools.com/CSS/css_pseudo_classes.asp>

View your document again and hover the mouse over the various menus – should be almost perfect. Now modify the styles so they say this:

#menu ul {

list-style-type: none;

display: block;

margin: 0;

padding: 0;

width: 100%;

}

#menu li {

display: block;

margin: 0;

padding: 0;

width: 100%;

}

#menu a {

color: #ffffff;

background-color: #000080;

text-decoration: none;

display: block;

}

#menu a:hover {

color: #000000;

background-color: #4040FF;

text-decoration: none;

display: block;

}

That should look nice, now – except for the colors, of course. Fix the colors, add margins, borders, and more “buttons” and you’re on your way to a useful web page.

**Exercise:**

Create a class style named newsbox that allows you to post ‘headline’ news in a small box floating on the right side of your content. Add a few links to try it out.

**Cascading Style Sheets**

Styles can be cascaded, i.e. multiple style sheets can be attached to one document. If there are conflicting styles, the “closest” one counts, otherwise styles are merged. For example, we could add a ‘base’ style sheet to our above document to ensure the basic dimensions of the sections we defined have well-defined values:

* create a second style sheet called “defaults.css”
* add the following styles:

#container {

margin: 0;

padding: 0;

border: 0;

}

#menu {

margin: 0;

padding: 0;

border: 0;

}

#content {

margin: 0;

padding: 0;

border: 0;

}

#footer {

margin: 0;

padding: 0;

border: 0;

}

* save this style sheet in the same directory as the other files
* add the following line to your HTML document to attach this sheet; add the line directly before the one attaching the original sheet:

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

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

**Review of HTML and CSS Use**

We have learned how to use multiple cascading style sheets to fine-tune the exact formatting and layout of a web page. To review, I’ll go through creating a half-way decently looking web page step by step.

Our web page(s) should have a header, a footer, a menu, and a content area:

|  |  |
| --- | --- |
| Header | |
| menu | content |
| Footer | |

We embed these section in an overall container section to center everything inside a browser. We define two style sheets: the first to provide base values, the second to define the actual layout.

Style sheet: **base.css**

/\* Base styles to provide default values \*/

#container {

margin: 0;

padding: 0;

border: 0;

}

#header {

margin: 0;

padding: 0;

border: 0;

}

#menu {

margin: 0;

padding: 0;

border: 0;

}

#content {

margin: 0;

padding: 0;

border: 0;

}

#footer {

margin: 0;

padding: 0;

border: 0;

}

Style sheet: **styles.css**

/\* Defining the overall sections \*/

#container {

width: 900px;

margin: 0 auto;

}

#header {

border-bottom: 1px gray solid;

}

#menu {

float: left;

width: 150px;

}

#content {

margin-left: 160px;

padding-left: 10px;

border-left: 1px gray solid;

}

#footer {

border-top: 1px gray solid;

}

Our sections now include borders to visually separate them. Next

* create a new HTML document named index.html
* Attach the base.css style sheet and the styles.css style sheet via <link …/> tags
* Add the id tags for the various sections via div tags

Your index.html file should now look like this:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

<title>Untitled 1</title>

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

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

</head>

<body>

<div id="container">

<div id="header">

<h1>Header</h1>

</div>

<div id="menu">

<h1>Menu</h1>

</div>

<div id="content">

<h1>Content</h1>

<p>This is some content.</p>

</div>

<div id="footer">

<p>footer</p>

</div>

</div>

</body>

</html>

Next we’ll add a menu to the menu section as a list of links. Right now the links can point to any URL, we will later change them to the real links. The code inside menu changes, everything else remains the same:

</div>

<div id="menu">

<h1>Menu</h1>

**<ul>**

**<li><a href="index.html">Menu 1</a></li>**

**<li><a href="index.html">Menu item 2</a></li>**

**</ul>**

</div>

<div id="content">

Now we redefine the tags inside the menu section to change the look of the header-1, and – more importantly - remove the bullets from the list and ‘buttonize’ the links. Since we have not yet made a color selection, we’ll use some grayish/blackish color scheme.

Add the following to the end of the styles.css sheet:

/\* Defining the tags in the menu section \*/

#menu ul {

list-style: none;

display: block;

margin: 0;

padding: 5px;

}

#menu a {

display: block;

text-decoration: none;

background-color: black;

color: white;

}

#menu a:hover {

display: block;

text-decoration: none;

background-color: gray;

color: black;

}

Now we define the overall colors in ‘layers’:

* We add the body tag to the beginning of our style.css. That will define the background color of the entire page, including all margins. We choose white.
* We leave colors of the container alone, which means the body colors will be the default
* Then we define the background color of the header as black with white text
* We leave colors of the content alone, which means the container colors will be the default, which in this case are the same as the body colors
* We choose gray as background color for the menu

body {

background-color: white;

color: black;

}

The other section id's get attributes accordingly. This will give our web page a little depressing looking – perhaps fit for the pages of a funeral home - but it’ll get the idea across. The rest is up to you and your imagination.

**Homework:**

Create your own web page design, using two style sheets and an index.html file. Create a class style named newsbox that allows you to post ‘headline’ news in a small box floating on the right side of your content. Add some text and a few links to try it out.