Lecture 05
The Web
(HTML, CGIs, & CSS)
Abbreviations

aka   Also Known As
CSS   Cascading Style Sheets
CWS   Course Web Site

Reading

Learning Web Design (2/e, library reserve)

Extracts from HTML & XHTML (5/e) — The Definitive Guide, Chapter 8 on Cascading Style Sheets also available online from the University at http://safari.ora.com > MY BOOKSHELF – Book 27

Western Civilization’s tutorial on CSS properties (for reference)

Sitepoint’s CSS Introduction and Documentation (for reference)
http://reference.sitepoint.com/css

The “Cascading Style Sheet Properties Quick Reference” (for reference)

CSS Pocket Reference (3/e), by Eric Meyer, O’Reilly & Associates (book store)

Optional background reading

Web Style Guide, by Lynch & Horton, Yale Univ Press, 0-300-07675-4 (Library Reserves)
Web Style Guide (2/e), by Lynch & Horton, Yale Univ Press, 0-300-09682-8

Optional reference text
Please read and high-light, before lab:

- Assignment 5
- This week’s slides

There are hyper-text commented source files for many of the web pages used in this lecture

- Handouts > Commented HTML on the CWS

Major topics today

- read and recall pearl
- the client-server paradigm
- putting HTML in context
- relative vs absolute URLs
- tables as a layout tool
- forms
- styles in HTML (esp cascading style sheets)

Today’s lecture assumes an elementary understanding of HTML tags & attributes (eg from CS 100)

- if you lack it, see Learning Web Design, or https://www.codecademy.com/courses/web-beginner-en-HZA3b/0/1
Road Map

New applications for this week

• PageSpinner
  a Mac-only HTML editor

• TextWrangler
  a text editor

• EditiX
  a cross-platform XML editor

• StyleMaster
  a cross-platform CSS editor

This week’s lecture builds on the preceding weeks’ material

• tables
• styles
• graphics
The Read ‘n Recall Pearl

Dialogs and Menus
- Menus
- Dialogs
- Toolbars & Palettes

Documentation
- The user manual
- Online help
- Online tutorials
Assumptions

You have an understanding of:

- Tables
- Styles
- Indirection
Things to Think About

• How does the manipulation of data objects differ from other applications?
• Is there more than one way to manipulate a data object?
Client-Server File Sharing

**EG the AppleShare file Server on Oscar**

- your lab Mac is the “client:” slower, cheaper, smaller disks
- student.cs is the “server machine:” faster, more expensive, bigger disks
- AppleShare is the “server application,” running on the server machine
- “share points” are folders on the server that are made available over the network
- “network disks”
  - “mounting” a share point (use the Finder’s Go > Connect to Server… menu item)
    creates a “network disk” on your client machine
  - an icon appears on the desktop, just as for a local disk
  - use a network disk just like a local disk, although it’s a bit slower
- “network folders”
  - these are subfolders (aka subdirectories) of a network disk
  - unlike the other terms on this slide, “network folder” is a CS200-invented term

As distinct from “peer-to-peer” file sharing
The Web (like file-sharing) has a client-server model

Many of the machines on the Internet are “web servers”
• any machine (“client”) on the web can ask them for data
• actually, they’re asking a particular application running on that machine for data (which is identified by a “port”)

The client uses a “browser” to request & display web pages
• eg Firefox, Safari, Chrome, Explorer, Opera, ...
• browsers decide how to render the HTML, based on
  HTML tags found in the document
  what kind of display is available
  user preferences
• browsers are often (but not always) consistent in how they do this

For security
• a web server can only return files in the “server subtree”
• sometimes that’s rooted in the folder holding the server app
• usually this “web root folder” can be set when the web server is started

The default Mac OS X web server application (Apache) is at /usr/sbin/httpd; the default web document root folder is at /Library/WebServer/Documents/.
Data Returned By Web Servers

The files returned can be

• web pages
• pictures
• other stuff...

A “web page” is a TEXT file containing

• text to be displayed (text “elements”)
• “tags”
  
  eg <html> and <p> that control presentation of the text—they’re really styles
  “links” containing “URLS” (eg <a href="• • •"> and <img src="• • •">)

  to other web pages
  to graphics, for display on the page
  to sounds, to be played when the page is viewed
  etc—add post-install “plugins” to handle new file types
  (/Library/Internet Plug-Ins/ or ~/Library/Internet Plug-Ins/ on Macs)

• URL = Uniform Resource Locator (eg “www.student.cs.uwaterloo.ca/~cs200/index.html”)
  (aka URI = Uniform Resource Indicator—yuck)
Formatting Tags in WordPerfect

What you see here are “property tags” but they could equally well be (named) style tags.

Strip out all the formatting codes to get a “text” or “ascii” file.

---

<table>
<thead>
<tr>
<th>Styles</th>
<th>Table</th>
<th>List</th>
<th>Merge</th>
<th>BookMark</th>
<th>HTML</th>
</tr>
</thead>
</table>

This document provides a set of macros to author your own HTML home pages in addition to the built-in HTML features in WP 3.5. WordPerfect provides a WYSIWYG editing environment along with a good set of tools for text manipulation. HTML has been and still is an obstacle and time consuming process in producing a home page, and WordPerfect’s HTML WYSIWYG capabilities makes it easier and faster to get the job done.

**To Install the HTML Macros included with this Readme:**

1. With this Read Me document open, choose **Preferences** from the **Edit** menu, then click the **Librarian** icon.

2. Choose **Macros** from the **Resource** pop-up menu at the top of the dialog box.

The list boxes will change to show macros. The list on the left shows the macros currently installed in your private library [Library (USA) if you are using U.S. English software]. The list box on the right shows the macros contained in the document.
A Toy Web Page

The HTML for this web page

```
<HTML><HEAD><TITLE>Jen's Fake Home Page</TITLE></HEAD><BODY bgcolor="#FFFFFF"><IMG SRC="star.gif"><IMG SRC="JenBanner.gif"><IMG SRC="star.gif"><CENTER><H2>Welcome to my Web page</H2></CENTER><IMG SRC="Exclamation.gif" ALIGN=left HSPACE=6><P><STRONG>Warning!</STRONG> This is not my <EM>real</EM> home page. It's just a little something I made up for the occasion. But just in case you're interested, I'll tell you a bit about me.</P><HR><H2>Places I've Lived</H2><UL><LI>Akron, OH</LI><LI>Hudson, OH</LI><LI>South Bend, IN</LI><LI>Boston, MA</LI></UL>(Adapted from "Designing for the Web - Getting Started in a New Medium" by Jennifer Niederst and Edie Freedman.)</BODY></HTML>
```

Yuck!

Recall that browsers ignore

- multiple blanks
- carriage returns
- blank lines

Use these to make your HTML more readable!

- You’ll lose marks in CS 200 if you don’t
- You’ll make your life difficult if you don’t
<HTML>
  <HEAD>
    <TITLE>Jen's Fake Home Page</TITLE>
  </HEAD>

  <BODY bgcolor="#FFFFFF">
    <IMG SRC="star.gif"><IMG SRC="JenBanner.gif"><IMG SRC="star.gif">
    <CENTER><H2>Welcome to my Web page</H2></CENTER>
    <IMG SRC="Exclamation.gif" ALIGN=left HSPACE=6>
    <P>
      <STRONG>Warning!</STRONG> This is not my <EM>real</EM> home page.
      It's just a little something I made up for the occasion.
      But just in case you're interested, I'll tell you a bit about me.
    </P>
    <HR>
    <H2>Places I've Lived</H2>
    <UL>
      <LI>Akron, OH</LI>
      <LI>Hudson, OH</LI>
      <LI>South Bend, IN</LI>
      <LI>Boston, MA</LI>
    </UL>
    <P style="font-size:80%">
      (Adapted from "Designing for the Web - Getting Started in a New Medium"
      by Jennifer Niederst and Edie Freedman.)
    </P>
  </BODY>
</HTML>
**Discussion Points**

**HTML is stored in “[ASCII] text files”**

**HTML tags as (named) styles**
- whose definitions are supplied by the browser
- same web page, different browser, generally similar (but not identical) appearance...

**Always use closing tags (eg </P>, </LI>)**
- otherwise the browser must guess their location
- be prepared for XHTML, XML & CSS

<HEAD> ... </HEAD>
- <HEAD> does not mean "header"
- <HEAD>... </HEAD> contain information about the page
- eg <meta name=description value="a paragraph">
- eg <meta name=author value="Bugs Bunny">
- eg <title>... </title>
  - ... shows up in most browsers’ title bar
  - browsers use ... to label bookmarks
  - index engines give words in ... extra weight

<BODY>... </BODY> contain info displayed in the page

---

Camino 1.6
If you view this page in any of the common browsers, you will find that they share two major deficiencies:
- paragraphs are separated by too much white space (typically an entire blank line),
- and list items are separated by too little white space (typically none beyond the current line-to-line leading).

Firefox 3.0
If you view this page in any of the common browsers, you will find that they share two major deficiencies:
- paragraphs are separated by too much white space (typically an entire blank line),
- and list items are separated by too little white space (typically none beyond the current line-to-line leading).

Omniweb 5.7
If you view this page in any of the common browsers, you will find that they share two major deficiencies:
- paragraphs are separated by too much white space (typically an entire blank line),
- and list items are separated by too little white space (typically none beyond the current line-to-line leading).

Opera 9.5
If you view this page in any of the common browsers, you will find that they share two major deficiencies:
- paragraphs are separated by too much white space (typically an entire blank line),
- and list items are separated by too little white space (typically none beyond the current line-to-line leading).
Browsers also *ignore* tags they don’t recognize

- eg tags you misspell
- this is actually a feature
  - so newly invented tags don’t screw up old browsers
  - so IE-specific tags don’t screw up Netscape, & vice-versa
  - etc
- but it makes debugging HTML harder
- therefore ... when a tag doesn’t seem to have any effect
  - suspect misspelling
- Validation - see the assignment for details
  - That’s what the `<!DOCTYPE ...>` magic incantation is for (see next slide)

**Upper case vs lower case**

- who cares?
  - HTML is case-insensitive: `<TITLE>...</TITLE>` works fine
  - XHTML requires that tags be lower case
  - XML is case-sensitive
- suggestion: use lower case
### A Simple HTML Table

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
    <HEAD>
        <TITLE>Mark Report</TITLE>
    </HEAD>

    <BODY>
        <H2>Top CS200 Marks</H2>
        <TABLE BORDER=1 ALIGN="center">
            <TR ALIGN="center">
                <TH>ID Number</TH>
                <TH>Final Grade</TH>
            </TR>
            <TR ALIGN="center">
                <TD>94010203</TD>
                <TD>81%</TD>
            </TR>
            <TR ALIGN="center">
                <TD>98102030</TD>
                <TD>75%</TD>
            </TR>
            <TR ALIGN="center">
                <TD>96000123</TD>
                <TD>67%</TD>
            </TR>
        </TABLE>

    </BODY>
</HTML>

For a list of valid doctypes, see http://www.w3.org/QA/2002/04/valid-dtd-list.html
## HTML Table Tags

- `<table> . . . </table>`
  - surround the entire table

- `<tr> . . . </tr>`
  - surround a table row

- `<td> . . . </td>`
  - surround a table (cell) definition

**By default**

- a table and its cells are as wide as they need to be

**For details, see**

- HTML The Definitive Guide (library reserve)
- PageSpinner Help

### Tables as a (deprecated) all-purpose layout tool in HTML

**(Hmm. Are tables a useful layout tool in word processors?)**

**HTML Tables can be nested**

- and HTML cells can be “merged” horizontally (colspan=”n”)
- and/or vertically (rowspan=”n”)

No L-shaped regions, however
For `<a href="http://jcbserver.cs.uwaterloo.ca/cs200/search/search.shtml"> search </a>`,
(equivalent to `<a href="http://jcbserver.cs.uwaterloo.ca:80/cs200/search/search.shtml"> search </a>)

- http is the “protocol”
- jcbServer is the server’s “local name”
- cs.uwaterloo.ca is the “domain” in which the server is located
- jcbServer.cs.uwaterloo.ca is the server’s “host domain name”
- 80 is the “port” on which jcbServer’s web server application is listening
- /cs200/search/search.html is the “absolute path” from the server’s web root folder to the file
Another Example URL

Another example:

---

**THE FOURTH PART OF GAUL**

A novel of the Veneti Gaul revolt against Caesar and the epic voyage of its survivors to the New World

*by*

[John C Beatty](fragments/bio/biography.html)

---

**Synopsis**

*In 58 B. C. Rome was the superpower of the Mediterranean world, and in that year Julius Caesar took up the governorship of the Roman Province in southern France or Gaul, as it was then called. The Roman Senate expected Caesar to govern the province, extract a reasonable amount of revenue, and guard the frontier against incursion by the many Gaulish tribes to the north.*

---

`<a href="fragments/bio/biography.html">John C Beatty</a>`

**fragments/bio/biography.html**

- is a “relative path” to the target file
- starting in the folder containing the page holding the link
- (note the LACK of an initial “/” and host domain name)

**<IMG SRC= ... > works the same way**
Relative vs Absolute Paths (1)

The web root folder

- a web server can only return files in the “server subtree”
- sometimes that’s rooted in the folder holding the server app
- sometimes this “web root folder” can be set elsewhere
Relative vs Absolute Paths (2)

A link in CS100.html to CS200.html could be written as

- `<a href="/2nd%20Year/cs200/cs200.html">CS200</a>`
- Note the initial slash
- This is an “absolute path”
- The host domain name is implicit
  
  \[\text{ie the same as the referencing web page}\]

Or as

- `<a href="../../2nd%20Year/cs200/cs200.html">CS200</a>`
- Note the initial “../../”
- This is a “relative path”
  
  \[\text{../ means “go up one level to the parent folder”}\]
  \[\text{../../ means “go up two levels,” etc}\]

\[\text{note: for security reasons, the web server application will prevent the path from rising above the web root folder!}\]

- Note that cs200/cs200.html is also a relative path

Or using an explicit host domain name
AND an absolute path

- `<a href="http://jcbServer.uwaterloo.ca/2nd%20Year/cs200/cs200.html">CS200</a>`
When to use relative vs absolute paths

**Absolute paths**
- always start at the web root folder
- are necessary between machines
- if a host domain name is present, the path is necessarily absolute

**Relative paths**
- start at the document containing the link
- make it MUCH easier to move web pages around as a group
  
  eg if all the cs200 pages use relative URLs amongst themselves then I can move the cs200 subtree somewhere else without breaking the links between the cs200 pages
- but if a file in the group links to a file on the same machine not in the group it must use an absolute file path (or the group can’t be moved w/o breaking that link)

**IMPORTANT**
- you MUST write "%20" instead of a space in URLs
- and look out for trailing blanks

**So within a web site...**
- use a relative path when the two files are more likely to be *moved together* (eg a page & an image in it)
- use an absolute path when the two files are more likely to be *moved separately*
Forms

An HTML form is a web page with "[interface] widgets" for supplying data

- text edit boxes
- check boxes
- radio buttons
- pop-up menus

![Example form](image-url)
Forms and CGIs

Web Servers can’t know in advance
• what data will be sent to them from forms
• what should be done with it

So there’s a convention (the “Common Gateway Interface”)
• for identifying the particular application
• to which form data should be sent for processing

Actually, the CGI scheme is more general than this
a web server can run any application and return its output

When the submit button is pressed
• the data is sent to a web server
• the web server forwards the data to a “cgi” (a separate program)
• the cgi processes the data & returns a web page to the server
• the server passes that response on to the browser

Plug-ins
The CS 200 request marks form (simplified)

To retrieve your marks to date in CS 200, please enter your last name (case doesn't matter) and student id in the boxes shown below.

Then click on the <STRONG>Fetch Marks</STRONG> button - ordinarily it shouldn't take more than thirty seconds or so for your marks to come back. (Please be patient - I'm only a Mac Ilfx!)

If you find a discrepancy, please notify the course tutor as soon as possible.

Note the use of a “hidden parameter” that the user never sees so forms for different courses can use the same cgi.
What gets sent to the server (GET)

GET ../ReportMarks.cgi?course=cs200&surname=Daly&idnumber=00000000 HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.06 (Macintosh; U; PPC, Nav)
Host: jcbServer.cs.uwaterloo.ca
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8

The rules for this stuff are part of the “http protocol.”

What comes back

Your CS 200 Marks are...

ID Number | 00000000
Last Name  | Daly
Assignment 1 Mark | 98

./ReportMarks.cgi locates the program (a “cgi”) to which the server forwards the form’s data

Notice

that the URL from which a web page came always appears in the location bar
that the forms data is encoded in the URL
how that URL appears in what’s sent to the server
why the path to the cgi had better not contain a question mark!
POST Actions

The request marks form could have said

```
<FORM ACTION="http://../ReportMarks.cgi" METHOD="POST">
```

In which case the forms data would be transferred a bit differently

```
POST ../ReportMarks.cgi HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.06 (Macintosh; U; PPC, Nav)
Host: 192.168.1.100
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png,
        */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
CONTENT-LENGTH: 43
course=cs200&surname=Daly&idnumber=00000000
```

The details of this difference are not important to us

But you do need to know that

- POST is necessary for large quantities of data — say > 256 characters
- POST’ed form data does not appear in the URL and therefore cannot be bookmarked
<HTML>
  <HEAD>
    <TITLE>Request Your Marks in CS 200</TITLE>
  </HEAD>
  <BODY>
    <P>
      To retrieve your marks to date in CS 200, please enter your last name
      (case doesn't matter) and student id in the boxes shown below.
    </P>
    <P>
      Then click on the <STRONG>Search</STRONG> button - ordinarily it shouldn't
      take more than thirty seconds or so for your marks to come back. (Please be
      patient - I'm only a Mac IIfx!)
    </P>
    <P>
      If you find a discrepancy, please notify the course tutor as soon as possible.
    </P>
  </CENTER>
</HTML>
<FORM ACTION="http:../ReportMarks.cgi" METHOD="POST">
   <INPUT TYPE="hidden" NAME="course" VALUE="cs200">
   <TABLE>
      <TR>
         <TD><STRONG>Your Last Name:</STRONG></TD>
         <TD><INPUT TYPE="text" NAME="surname" SIZE="33"></TD>
      </TR>
      <TR>
         <TD><STRONG>Your ID Number:</STRONG></TD>
         <TD><INPUT TYPE="text" NAME="idnumber" SIZE="33"></TD>
      </TR>
      <TR>
         <TD COLSPAN="2"><STRONG>Which:</STRONG></TD>
         <TD><INPUT TYPE="checkbox" NAME="which" VALUE="assign"> Assignments</TD>
         <TD><INPUT TYPE="checkbox" NAME="which" VALUE="midterm"> Midterm</TD>
         <TD><INPUT TYPE="checkbox" NAME="which" VALUE="final"> Final</TD>
         <TD><INPUT TYPE="checkbox" NAME="which" VALUE="course"> Course Mark</TD>
      </TR>
      <TR ALIGN=left><INPUT TYPE="submit" VALUE="Fetch Marks"></TR>
   </TABLE>
</FORM>

**Styles in HTML**

Most tags are named styles: `<strong>`, `<em>`, `<p>`, `<ol>`, ...

- these are “logical tags”
- browsers decide how to render them

Although a few are not: `<b>`, `<i>`

- these are “physical tags”
- browsers have no choice

Generally speaking

- the browser, not the web author, controls layout
- browsers sometimes behave differently

But appearance is important!
Controlling HTML Layout

The response

- abusing tables and frames
- new tags and attributes
- proprietary tags and attributes
- postscript and pdf (Adobe’s “Portable Document Format”) via “plugins”

Wouldn’t it be nice ...

- if HTML had USER-DEFINED [named] styles?
- like those we find in word processors?
- that authors could use to control layout?
Cascading Style Sheets

Style definitions may appear at the beginning of a web page
There are five style definitions in this example

the first specifies the default font to be used
the second and third attach default “properties” (ie attributes) to <SPAN> and <P>
the last two define property bundles that can be applied to <P>

```html
<HTML>
  <HEAD>
    <STYLE TYPE="TEXT/CSS">
      BODY  { FONT-FAMILY: 'Myriad Pro', sans-serif }
      SPAN  { COLOR:       green                    }
      P     { TEXT-ALIGN:  center                   }
      P.A   { FONT-WEIGHT: bold                     }
      P.B   { FONT-STYLE:  italic                   }
    </STYLE>
  </HEAD>

  <BODY>
    <P>         This paragraph will be centered.            </P>
    <P         CLASS=A> This paragraph will be centered and bold.   </P>
    <P         CLASS=B> This paragraph will be centered and italic. </P>
    <P>         This word will be <SPAN>green</SPAN>.       </P>
  </BODY>
</HTML>
```
Or ... style definitions may be EXTERNAL to a web page

- so that multiple web pages can use them
- much more important for HTML than for word processing

An example style sheet file named simple.css

An example html file that uses simple.css

Notice that all four paragraphs are centered and in Myriad Pro

- The contents of <P class=A>...</P> and <P class=B>...</P> “inherit” the properties of <P>, so they don’t have to explicitly set them
- Inner elements inherit properties from outer elements containing them (when that makes sense) — eg the <P>...</P> inherit Myriad Pro from the <BODY>...</BODY> in which they appear
A Larger Example – Task B from the Winter 98 Lab Exam

Sample Lab Exam - Task B

This page is best displayed with Internet Explorer 4 or 5; Netscape 4 gets some of the leading wrong...

To use computers effectively it is important to select an appropriate tool for each task; more often than not this involves using several tools to solve a problem, passing data from one tool to the next as you work. This task is an example of such.

The problem. As a teacher, you often want to look at a bar chart (like the example on the right) showing the distribution of marks for a course -- how many people received marks between 70 and 75, how many between 75 and 80, etc.

The solution. You have a FileMaker table containing the final grades for each student, but databases aren't good at making graphs, and you often want to include such graphs in end-of-term reports for your department chair. So you need to transfer data between FileMaker, Excel, and Microsoft Word as you work. Because you and your colleagues do this often, you want to work out a convenient and efficient way of doing so.

Password-protected demonstration solutions may be found in the Demo Solutions subfolder of the 200exams disk on jcbServer. Copy them to your personal subfolder of 200exams before running them.

Demo Course Grades. When the Do It! button on the Choose layout is clicked, grades for the course currently selected by the Which Course popup are written into a file called Data in the same folder.

Demo Make Histograms. Click the Get Data button in any of the worksheets to open the Data file, copy its contents into the Grades worksheet, and then close it...
LabExam.css

Note the period that begins each style name
- such styles can be used in any tag (if they make sense...)

<DIV> is (effectively) a replacement for <P>
- with no default properties
- (some properties of built-in tags like <P> can’t be over-ridden) [still?]
- <div> and <p> are examples of “block level tags” or elements
  (ie they cause a line break)

<SPAN> is an “inline level tag” (aka an “inline element”)
- with no built-in properties
- does *not* cause a line break
- <strong>, <em>, <img> and <a> are other examples

“STYLE” is used to set CSS attributes for a particular tag only
- eg <div style="color:green">blah, blah, blah...</div>
- eg <p style="color:green">good stuff</p>
- eg <li style="color:green">clear desk</li>
- eg <span style="color:green">Good work!</span>

Block level tags generate automatic line breaks before & after.
Inline tags do not; one can follow another on the same line.
Sample Lab Exam - Task B

This page is best displayed with • • •

To use computers effectively it is important • • •

The problem.
As a teacher, you often want • • •

The solution.
You have a FileMaker table • • •
Password-protected demonstration solutions

Demo Course Grades.
When the Do It! button on the Choose layout

Demo Make Histograms.
Click the Get Data button in any

The charts in By 5 and By 10 can

You are to duplicate the behaviour

(The relative weight of important pieces}

Password-protected demonstration solutions may be found in the Demo Solutions subfolder of the 200exams disk on JobServer. Copy them to your personal subfolder of 200exams before running them.

Demo Course Grades. When the Do It! button on the Choose layout is clicked, grades for the course currently selected by the Which Course popup are written into a file called Data in the same folder.

Demo Make Histograms. Click the Get Data button in any of the worksheets to open the Data file, copy its contents into the Grades worksheet, and then close it.
Another Example, from “Eric Meyer on CSS”

http://jcbserver.uwaterloo.ca/cs200/ericMeyer/ericMeyer.html
body {
    background: rgb(153,102,51);
    color:       black;
    font-family: 'Myriad Pro', sans-serif;
}

div{  
    background: rgb(251,233,198);
    color:       rgb(122,74,26);
    margin:      0 2em;
}
p{
    margin:      0;
    padding:     0.25em 1em 0.25em 1em;
    text-indent: 1.25em;
    line-height: 120%;
}
h1, h2{
    margin:      0;
    padding:     0.25em 0.5em 0.25em 0.5em;
}
div#p1{
    margin:      0 2em 0 10em;
}
div#p2{
    margin:      0 10em 0 2em;
}
div#menu{
    float:       right;
    width:       5em;
    padding:     0;
    margin:      0 -1.5em 0.25em 0.5em;
    border:      3px solid rgb(50,50,175);
    background:  white;
}
div#menu a{
    display:     block;
    text-align:  center;
    padding:     0.2em 0.5em 0.2em 0.5em;
}
div#footer{
    margin:      0 11em 0 2em;
    padding:     0.2em 0 0.5em 0;
    text-align:  center;
    font-style:  italic;
    color:       rgb(128,128,128);
}

CSS for <div ID="p2">...</div>
Other Selectors

What we've seen

- `p { ⋅ ⋅ ⋅ }` sets attributes for all `<p>` tags
- `p.name` sets attributes for all `<p class="name" ⋅ ⋅ >` tags
- `p#name` sets attributes for the `<p id="name" ⋅ ⋅ >` tag

There are a variety of useful selectors we haven’t discussed, including

- `h1, h2, h3 { ⋅ ⋅ ⋅ }` grouped selectors (same attributes for multiple tags)
- `p[title] { ⋅ ⋅ ⋅ }` attributes for all tags `<p title= ⋅ ⋅ >` (ie `<p>` tags having a title attribute)
- `p[title="important"] { ⋅ ⋅ ⋅ }` attributes for all tags `<p title="important" ⋅ ⋅ >`
- `h1 > strong { ⋅ ⋅ ⋅ }` attributes for `<strong>` appearing “immediately” within an `<h1>`
  eg `<h1>This is <strong>very</strong> important.</h1>`
- `h1 + p { ⋅ ⋅ ⋅ }` attributes for any `<p>` that immediately follows an `<h1>`
  eg `<h1>Section A</h1><p>For this para.</p>`<p>But not this one.</p>`
- `tr > td:first-child` attributes for `<td>` when it is the first child of a `<tr>`
  eg `<tr> <td>matches this</td> <td>but not this</td> </td>`

and various combinations of these.

Effectively, they do “pattern matching.”

See Chapter 2 of “CSS The Definitive Guide” if you’re curious.

(You’re not expected to memorize these for CS 200—this is useful “read and recall [someday] info”)
The HTML

```html
<html>
<head>
    <title>A Simple Eric Meyer • • •</title>
    <LINK REL="STYLESHEET" HREF="ericMeyer.css">
</head>
<body>

<div class="wrap" id="p1">
    <h1>much room to think</h1>
    <p>
        Sometimes I find myself • • •
    </p>
    <p>
        I suppose it's the taste • • •
    </p>
</div>

<div class="wrap" id="p2">
    <h2>a fun guy</h2>
    <div id="menu">
        <a href="test2.html">stories</a>
        <a href="test3.html">poems</a>
        <a href="test4.html" id="lastlink">art</a>
    </div>
    <p>
        Good evening, my name is Rootsy • • •
    </p>
</div>

<div class="wrap" id="footer">
</div>
</body>
</html>
```

Notice that attributes can come from more than one style definition

- eg from div.wrap {...} and div#p1 {...}, applied to <div class="wrap" id="p1">
- when that happens, they’re merged
- if there’s a conflict, “the more specific wins”
  
  eg p1 over wrap because only one tag can have id=p1
Discussion

The “class” attribute can be used by many tags, which share its meaning
The “id” attribute is supposed to uniquely identify a tag (i.e., only be used once)
Both specify a style to use on their content

Style sheets can come from
- the web page author
- the user, who can often specify a style sheet in the browser’s preferences
- the browser (i.e., built-in)
  — And this is their order of priority (from high to low) when a conflict arises for a particular attribute

Browsers are finicky about CSS syntax
- if your CSS seems to have no effect, check for syntax errors / use a CSS validator

The detailed rules for resolving conflicts are (considerably) more complicated;
- see Section 8.1.9 of “HTML & XHTML - The Definitive Guide,” 5/e, for a fuller explanation
- or Chapter 3, “Structure and the Cascade,” in Cascading Style Sheets - The Definitive Guide for details
  — but you shouldn’t need to

Note: there’s a lot more to CSS than we’ve had time to talk about
Safari’s > Show Inspector

Good evening. My name is Robby, and I'll be your host for this evening. It's been said that I should just stop with this whole Webbery thing and pixies so I try to keep the source visible, and I hope you like the taste, but we're all adults here, and we can handle the truth.

Universal Access: □ Never use font sizes smaller than 14
□ Press Tab to highlight each item on a webpage
Option→Tab highlights each item.

Style sheet: jcbDocs.css
Proxies: Change Settings...
□ Show Develop menu in menu bar

Stuff we eat. Take mushrooms, for example. They’re kind of because it's "nasty." They’re shaped like little umbrellas, but you most varieties are grown with liberal quantities of what I like which our mothers always warned would kill us. Heck, some mushrooms that so many of us like?

Fungi. I understand that grasshoppers can be very tasty, too, so weird stuff. But fungus? I don't know. Maybe the morel is that if you see what I mean. It could be that the mushroom-eaters aliens who are secretly planning to take over the world? Just wonder when I'll get my instructions from the mothership.
For More (Optional) Information on CSS

Western Civilization’s

“Complete CSS Guide” at

“Properties Introduction” (the Complete CSS Guide’s section on CSS attributes) at

HTML & XHTML — The Definitive Guide, 5/e, by Musciano & Kennedy, Chapter 9, “Cascading Style Sheets”

on reserve in the library (an earlier edition, without “XHTML” in the title

or at http://safari.ora.com > MY BOOKSHELF – Book 26 from a University computer

Chapter 8 “Cascading Style Sheets” & the appendix “Cascading Style Sheet Properties Quick Reference”

(the 6th edition was published in October of 2006)

Cascading Style Sheets — The Definitive Guide, by Eric Meyer


or the 2/e at http://www.safari.ora.com > MY BOOKSHELF – Book 9 from a University computer

Typetester: a neat web page (w/Javascript+CSS) for comparing various fonts side-by-side in your browser

typetester.maratz.com

XyleScope: a neat tool for dissecting the CSS in pages you encounter on the web ($20, Mac only)

www.culturedCode.com/xyle/


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi lacus felis, euismod at, pulvinar sit amet, dapibus eu, eros.
Common Sources of Confusion in the Lab

You can use Firefox or Safari’s File > Open File... menu item
• that’s effectively what PageSpinner’s Preview button / menu item does
• HOWEVER
  File > Open File... is not as fussy about paths as most web servers
  – *it won’t complain about spaces in URLs*
  – *it won’t complain about trailing blanks in file names*
• So you MUST also ask student.cs.uwaterloo.ca web server to access your web pages to be certain that the URLs in them work
  www.student.cs.uwaterloo.ca/YourUsername/root.html

Browsers “cache” (most) pages you have browsed on your local disk
• When you’ve changed the contents of a page and saved it to your network disk from PageSpinner, *option-click the Reload button* or  to ensure your browser REALLY gets the new version

And a word of advice ... use
• closing tags (eg </TD>, </P>)
• indentation
• blank lines

to structure your HTML
  — it makes debugging *much* easier
Finally...

Remember that <head> does NOT mean <header>

• <head> ... </head>
  
  enclose information ABOUT the web page
  
  that is not displayed IN the page

• <style> ... </style> illustrates this better than <title>...</title>

Most browsers have a “View > Source...” menu item
that will show you the HTML source for the page currently being displayed

Warning

• CSS is still not perfectly implemented by contemporary browsers,
  although the situation is much better now than it was a few years ago

• So use the latest release of whatever browser you like
  when experimenting with Cascading Style Sheets

• Also, StyleMaster has lots of info about browser quirks & bugs
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="ToyTable.css"?>
<doc>

<xmlTable>
  <row>
    <cell>X</cell>
    <cell>–</cell>
    <cell>O</cell>
  </row>
  <row>
    <cell>–</cell>
    <cell>–</cell>
    <cell>O</cell>
  </row>
  <row>
    <cell>O</cell>
    <cell>–</cell>
    <cell>X</cell>
  </row>
</xmlTable>

<example>
  This text is centered WITHIN the space occupied by the <example> ... </example> block.
</example>

</doc>
ToyTable.css

doc {
  font-family: "Comic Sans MS", sans-serif;
  font-size: 20pt;
}
xmlTable {
  display: table;
  color: black;
  background-color: yellow;
  margin-top: 1em;
  margin-left: auto;
  margin-right: auto;
}
row {
  display: table-row;
}
cell {
  display: table-cell;
  padding-right: 0.2em;
  padding-left: 0.2em;
  padding-top: 0.2em;
  padding-bottom: 0.2em;
  width: 1.5em;
  text-align: center;
}
example {
  display: block;
  padding: 0.5em;
  background-color: cyan;
  margin-left: 0%;
  margin-right: 50%;
  text-align: center;
}

Use LR margins of auto to center a block-level element within its parent.

Use text-align: center to center text within an element.
XML—w/o the <?xml-stylesheet ... ?>

<?xml version="1.0" encoding="UTF-8"?>

<doc>

  <xmlTable>
    <row>
      <cell>X</cell>
      <cell>–</cell>
      <cell>O</cell>
    </row>
    <row>
      <cell>–</cell>
      <cell>–</cell>
      <cell>O</cell>
    </row>
    <row>
      <cell>O</cell>
      <cell>–</cell>
      <cell>X</cell>
    </row>
  </xmlTable>

  <example>
    This text is centered WITHIN the space occupied by the <example> ... </example> block.
  </example>

</doc>
CSS + JavaScript To Change The Default Font

Example: changing the font locally

- local:  http://127.0.0.1/cs200/switchingFontsWithJavascript/JensHomePage.html
The Javascript for this Example

Notice that the CSS attribute “font-family” becomes “fontFamily” in Javascript because “–” is illegal in a variable name.

Think about Excel scripting as you read this JavaScript.
Example: switching style sheets dynamically

- local: http://127.0.0.1/cs200/switchingStyleSheets/JensHomePage.html

See JensHomePage.html and the javascript source file global.js in .../cs200/switchingStyleSheets/ for details if you’re curious.

(Note that the color of the “Warning!” paragraph is chosen randomly.)
CSS + JavaScript

Adding a little JavaScript to CSS (of the sort covered in CS 100)

- dynamically changing CSS attributes
- collapsing menus
- absolute positioning of layers (remember Photoshop & Illustrator?)
- and much, much more

Example:

- local: http://127.0.0.1/dynamicMenus.html
The CSS for this Example

<html>
<head>
<title>An Illustration of Dynamic Fonts &amp; Menus</title>

<style type="text/css">
<!--
 body     { font-family:      sans-serif; }
#menuOne { display:          none;       }
#menuTwo { display:          none;       }
#title   {
    font-size:        15pt;
    font-weight:      bold;
    margin-bottom:    15pt;
    text-align:       center;
}
#content  {
    position:         absolute;
    left:             2.5in;
    top:              0.5in;
    background-color: #FFFF00;
    padding:          25px;
}
-->

</style>

</head>
</html>
The Content for this Example

An Illustration of Dynamic Fonts & Menus

Larger Title

Smaller Title

UofW (+/-)

CS
CS 200
CS 230
CS 436
jcbServer
Math
Oscar
UofW

Other sites (+/-)

Daring Fireball
Mac Geekery
Ars Technica
Slashdot
Apple
Microsoft
Mozilla

Oops ... this browser doesn't implement the IFRAME tag.

Adapted from Section 6.13 of “The CSS Cookbook,” by Christopher Schmitt, O’Reilly & Associates.

See also the documentation for CSS attributes in the “CSS Pocket Reference.”
<script language="JavaScript">

var currentFontSize = 15;
function changeFontSize( delta ) {
    var titleElement  = document.getElementById("title").style;
    currentFontSize   = currentFontSize + delta;
    titleElement.fontSize = currentFontSize + "pt";
}

function toggleMenu( menuClicked ) {
    processOneMenu( "menuOne", menuClicked );
    processOneMenu( "menuTwo", menuClicked );
}

function processOneMenu( menuToCheck, menuClicked ) {
    var menuClickedStyles = document.getElementById(menuClicked).style;
    var menuToCheckStyles = document.getElementById(menuToCheck).style;
    if( menuClicked == menuToCheck ) {
        if( menuClickedStyles.display == "block" ) {
            menuClickedStyles.display = "none";
        } else {
            menuClickedStyles.display = "block";
        }
    } else {
        menuToCheckStyles.display = "none";
    }
}

function setContentTo( theURL ) {
    var theDiv = document.getElementById("content");
    theDiv.src = theURL;
}

</script>
An example of “exotic selectors” in use

```html
<html>
<head>
  <style>
    td#title { color: black; }
    tr > td:first-child { color: blue; text-align: center; }
    tr > td + td + td { color: red; }
    tr > td + td + td + td { color: black; }
  </style>
</head>
<body>
  <table align='center' border=0 cellpadding=0 cellspacing=0 width=707>
    <tr>
      <td colspan=5 align=center id="title">
        ASCII Character Codes
      </td>
    </tr>
    <tr>
      <td colspan=5 align=center>
        ASCII Character Codes
      </td>
    </tr>
    <tr>
      <td><a name=table>Bits</a></td>
      <td>Decimal Value</td>
      <td>Char</td>
      <td>Abbr</td>
      <td>Meaning</td>
    </tr>
    <tr>
      <td>32</td>
      <td>0010 0000</td>
      <td>!</td>
      <td></td>
      <td>space</td>
    </tr>
    <tr>
      <td>33</td>
      <td>0010 0001</td>
      <td>
        
      </td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td>34</td>
      <td>0010 0010</td>
      <td>
        
      </td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td>35</td>
      <td>0010 0011</td>
      <td>
        
      </td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td>36</td>
      <td>0010 0100</td>
      <td>$
        
      </td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td>37</td>
      <td>0010 0101</td>
      <td>%
        
      </td>
      <td></td>
      <td></td>
    </tr>
  </table>
</body>
</html>
```