

CS 200

Lecture 05

The Web

(HTML, CGIs, & CSS)

Miscellaneous Notes

Abbreviations

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

Reading

The Non-Designer's Design Book, by Robin Williams:

- Chapters 1 – 6

Optional Background 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)

http://www.westciv.com/style_master/academy/css_tutorial/introduction/css_intro.html

Sitepoint's CSS Introduction and Documentation (for reference)

<http://reference.sitepoint.com/css>

Administrivia

Please read and high-light, before lab:

- Assignment 5: Due Monday June 15 at 11:59 pm
- 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
<https://www.student.cs.uwaterloo.ca/~cs200/#handouts>

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 (especially cascading style sheets – CSS)

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

Road Map

New applications for this week

- BBEdit
 - a text editor
- EditiX
 - a cross-platform XML editor
- StyleMaster
 - a cross-platform CSS editor
- You can use any text editor you'd like, though some are nicer than others for various reasons

This week's lecture builds on the preceding weeks' material:

- tables
- styles
- graphics



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 a local Mac Server

- 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

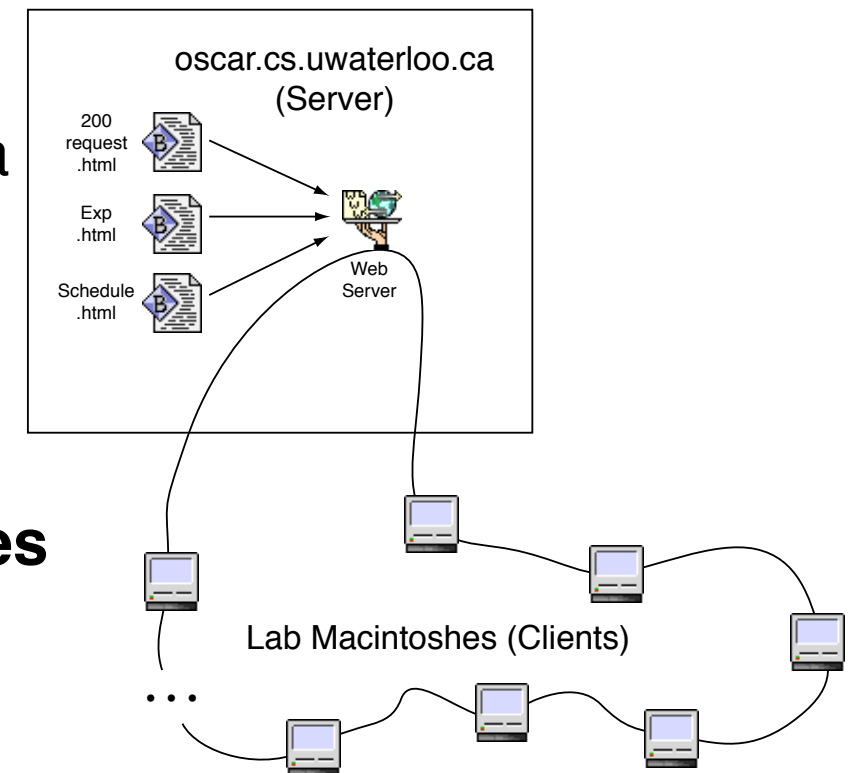
The Web (like file-sharing) has a client-server model

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, ...
- 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

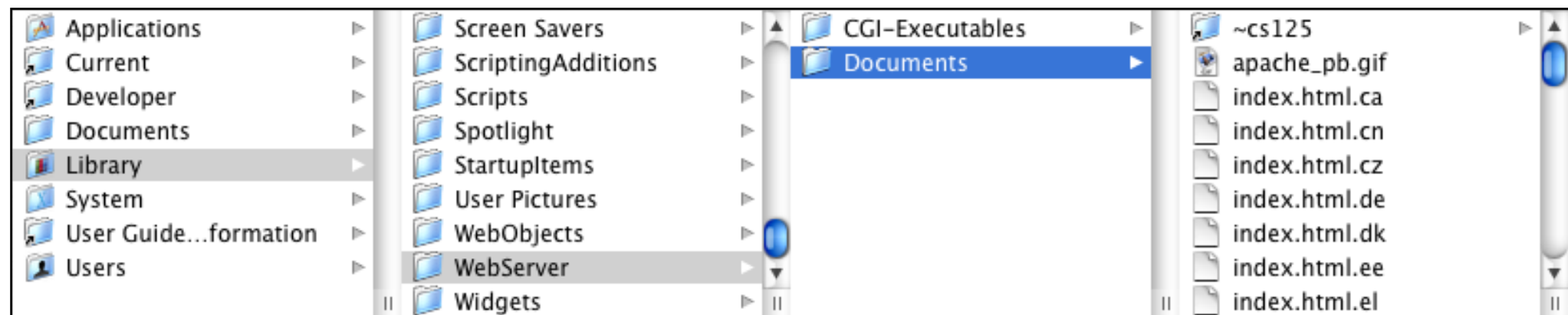


The default Mac OS X web server application (Apache) is at `/usr/sbin/httpd`; the default web document root folder is at `/Library/WebServer/Documents/`.

The Client-Server Model Continued

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



Data Returned By Web Servers

The files returned can be

- web pages
- pictures
- other stuff...

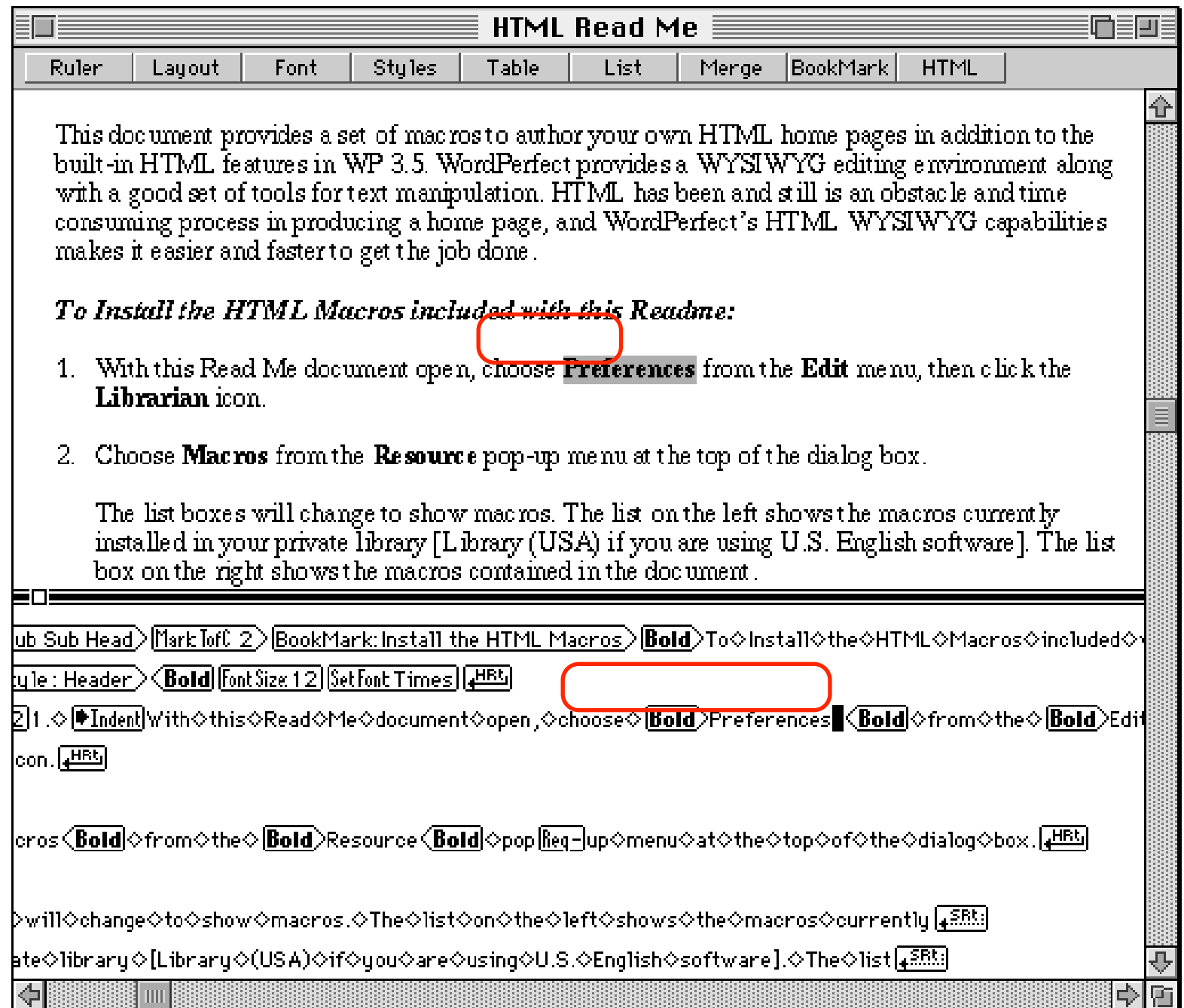
A “web page” is a TEXT file containing

- 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 and)
 - *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”)

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



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>Sourth 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>
```

Yuck!



The HTML for our Toy Web Page, Readably Formatted

<HTML>

<HEAD>

<TITLE>Jen's Fake Home Page</TITLE>

</HEAD>

<BODY bgcolor="#FFFFFF">

<CENTER><H2>Welcome to my Web page</H2></CENTER>

<P>

Warning! This is not my real 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>

Akron, OH

Hudson, OH

South Bend, IN

Boston, MA

<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>



HTML Formatting

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
- It will be very difficult for anyone else to read and use your code if you don't

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>`, ``)

- 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

More Discussion Points

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 Chrome, & 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)
https://www.w3schools.com/tags/tag_doctype.asp

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>
```

ID Number	Final Grade
94010203	81%
98102030	75%
96000123	67%

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

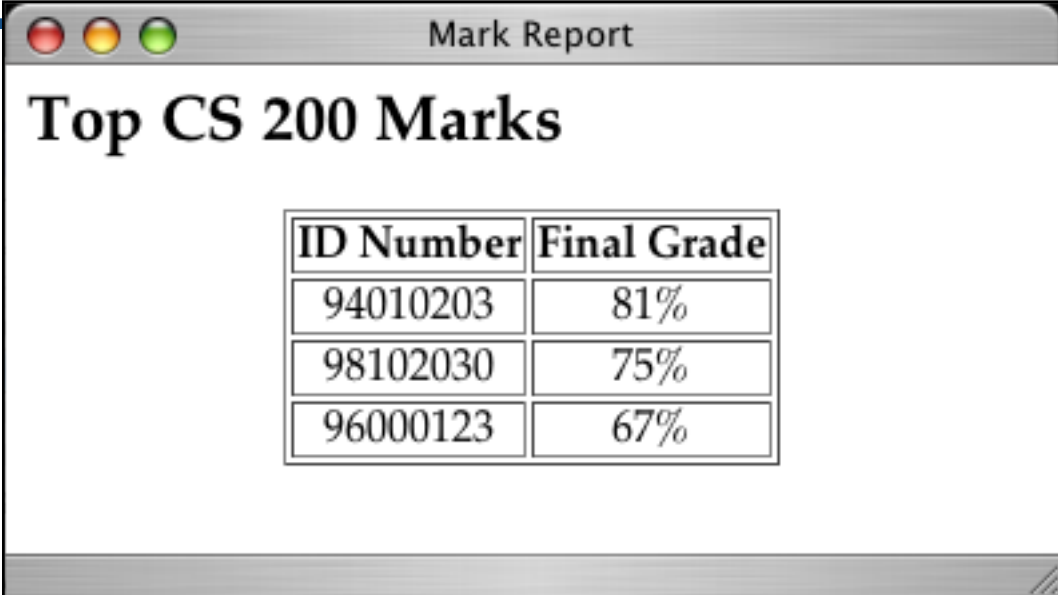
- HTML The Definitive Guide (library reserve)
- PageSpinner Help

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



ID Number	Final Grade
94010203	81%
98102030	75%
96000123	67%

Tables for Layout

Mark Twain

Samuel Langhorne Clemens (November 30, 1835 - April 21, 1910), better known by his pen name Mark Twain, was an American humorist, satirist, novelist, writer, and lecturer.

Although Twain was confounded by financial and business affairs, his humor and wit were keen, and he enjoyed immense public popularity. At his peak, he was probably the most popular American celebrity of his time. In 1907, crowds at the Jamestown Exposition thronged just to get a glimpse of him. He had dozens of famous friends, including William Dean Howells, Booker T. Washington, Nikola Tesla, Helen Keller, and Henry Huttleston Rogers. Fellow American author William Faulkner is credited with writing that Twain was "the first truly American writer, and all of us since are his heirs." Twain died in 1910 and is buried in Elmira, New York.

X	O	X
	X	
X		O

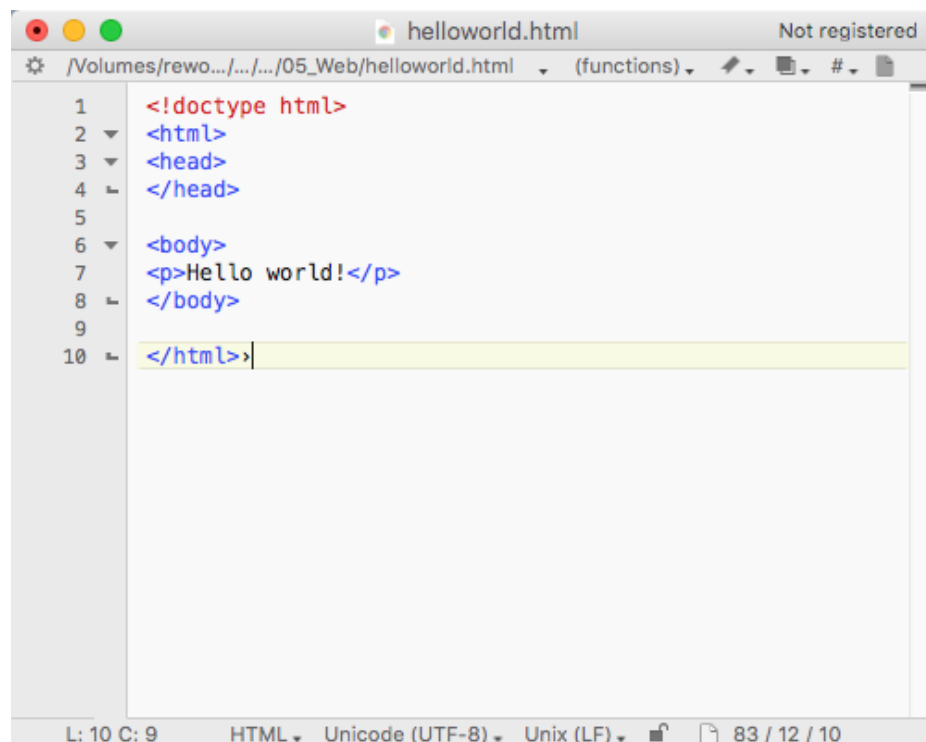
Pen Names

Clemens maintained that his primary pen name, "Mark Twain," came from his years working on Mississippi riverboats, where two fathoms (12 feet, approximately 3.7 meters) or "safe water" was measured on the sounding line. The riverboatman's cry was "mark twain" or, more fully, "by the mark twain" ("twain" is an archaic term for two). "By the mark twain" meant "according to the mark [on the line], [the depth is] two fathoms". Clemens provides a footnote to Chapter 8 ("Perplexing Lessons") of *Life on the Mississippi* where he explains "mark twain" as "two fathoms" and "Mark three is three fathoms". The name may also have come from his wilder days in the West, where he would buy two drinks and tell the bartender to "mark twain" on his tab.

From the online encyclopedia Wikipedia, at http://en.wikipedia.org/wiki/Mark_twain.

Previewing your Webpage

- Save your work in the text editor
- Open the same file up in a web browser
- Every time you save the text file, refresh the webpage
- For example...



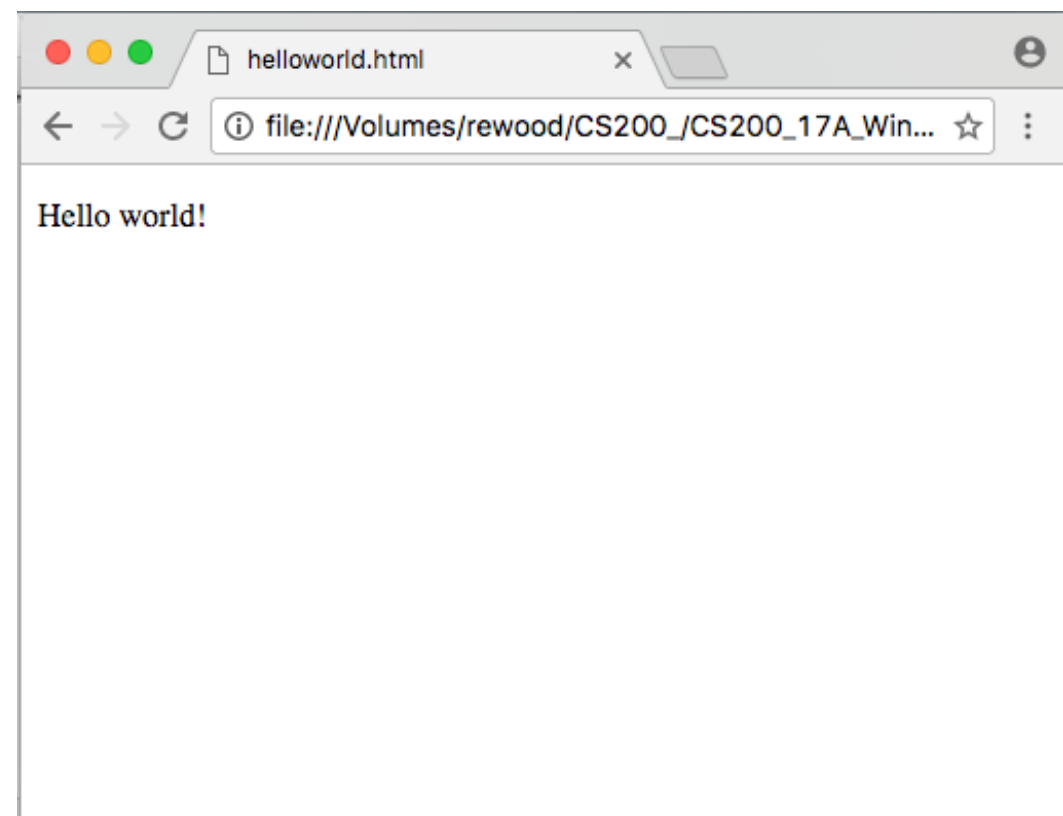
The screenshot shows the BBEdit text editor with a file named 'helloworld.html' open. The code is as follows:

```
1 <!doctype html>
2 <html>
3 <head>
4 </head>
5
6 <body>
7 <p>Hello world!</p>
8 </body>
9
10 </html>
```

The status bar at the bottom indicates the file is in HTML format, using UTF-8 encoding, with Unix line endings. The cursor is at line 10, column 9.

BBEdit

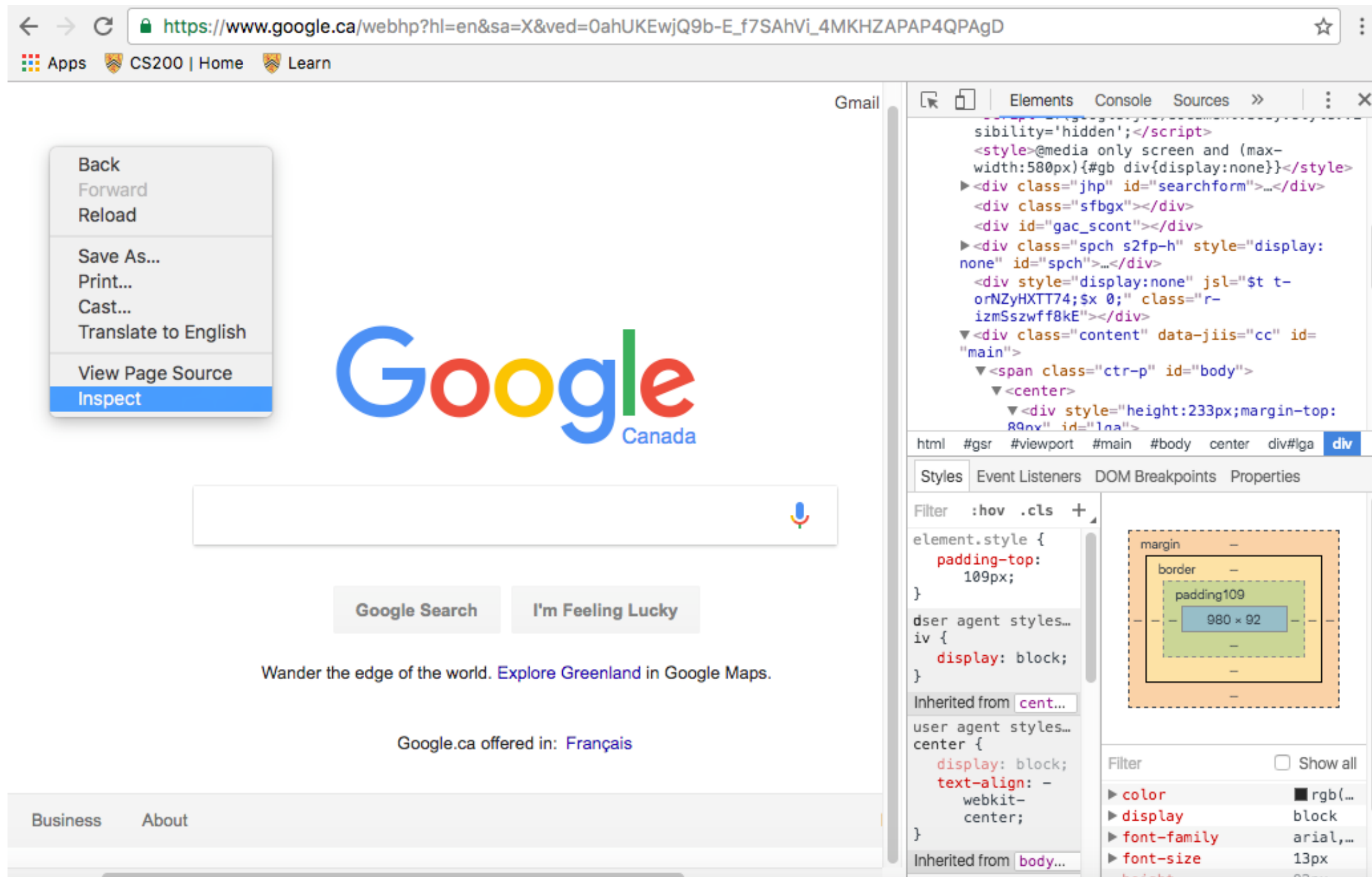
- Some text editors or in browser text editors allow you to preview your page within the app
- It's recommended to also preview it in common web browsers to make sure it works where others will see it



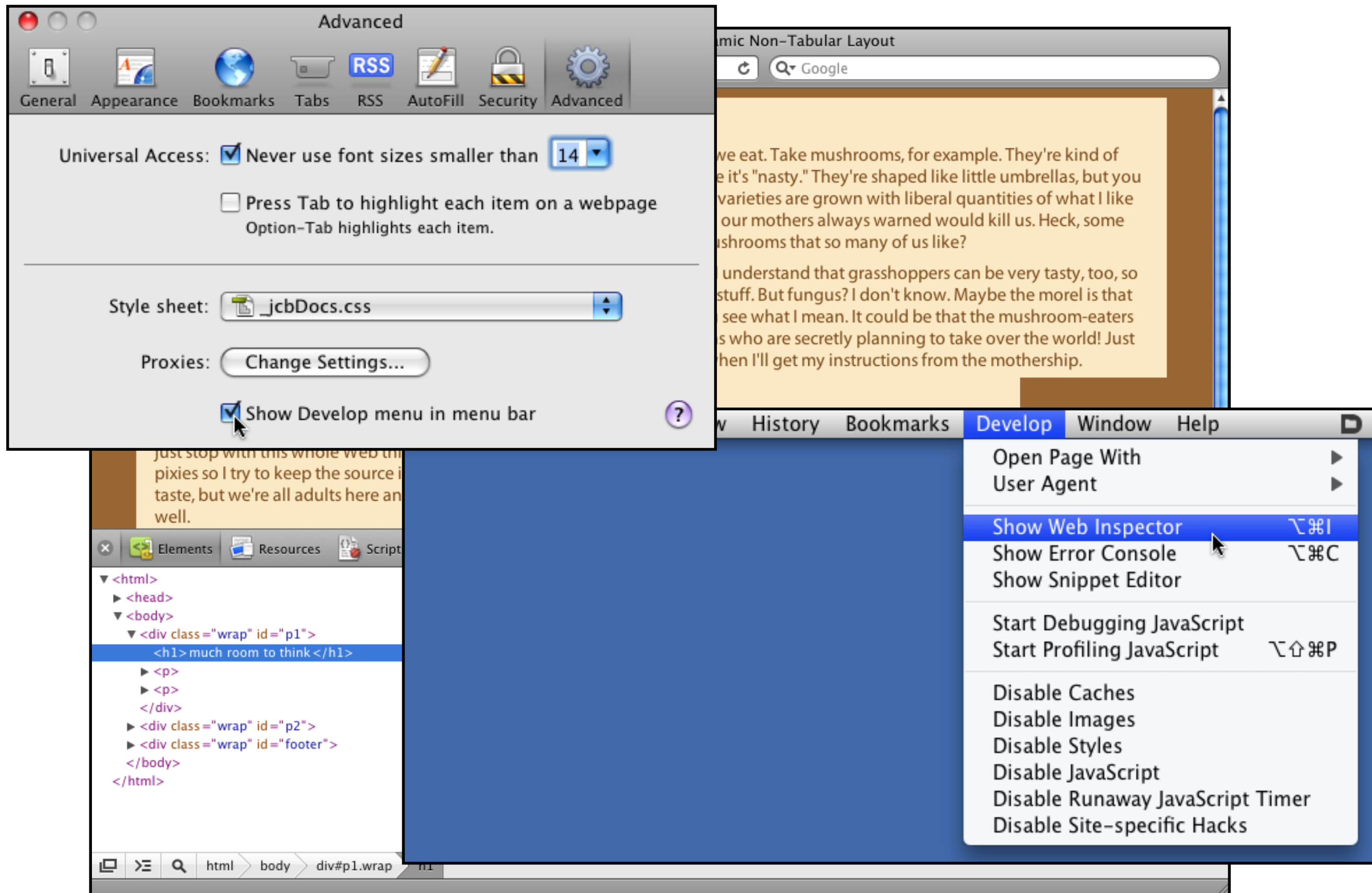
Chrome

Chrome's Inspect

Most browsers allow you to view the page source, which will help you with debugging your own webpages.



Safari's > Show Inspector



URLs — Uniform Resource Locators

For ` search `,
(equivalent to ` search `)

- `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

[An anchor with another protocol: ` . . . `]



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

JOHN CARRIE BEATTY

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



Gaul in 56 BC

[Editorial Note](#)

[Prologue](#)

[Purchase from Amazon](#)

[Purchase from Barnes & Noble](#)

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.

`John C Beatty`

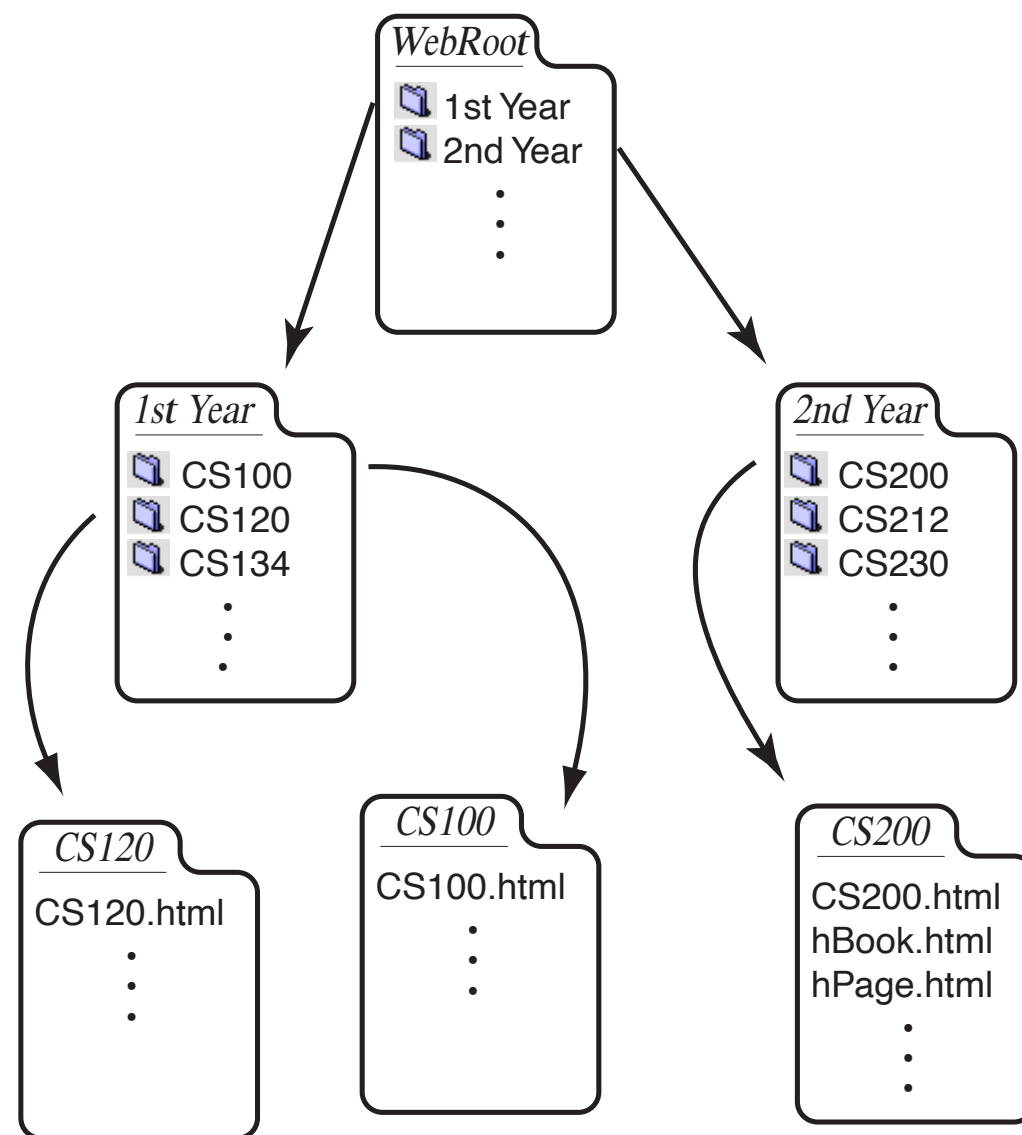
`fragments/bio/biography.html`

- 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)
- `` 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)

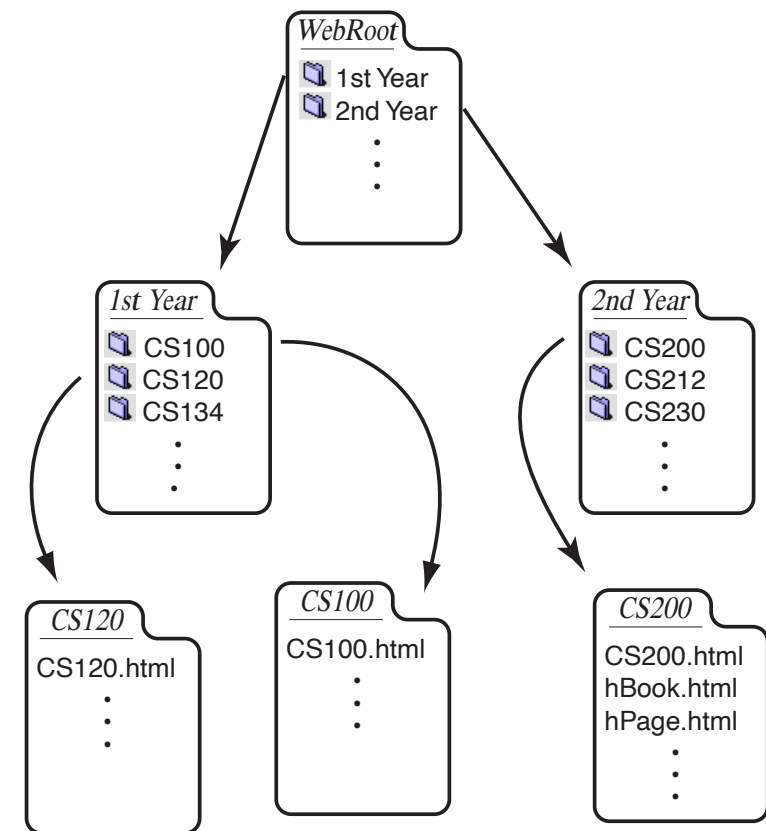
A link in CS100.html to CS200.html could be written as

- `CS200`
- Note the initial slash
- This is an “absolute path”
- The host domain name is implicit
 - ie the same as the referencing web page

Or as

- `CS200`
- Note the initial “../../”
- This is a “relative path”
 - `../` means “go up one level to the parent folder”
 - `../../` means “go up two levels,” etc
 - **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



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 (like to another server) 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

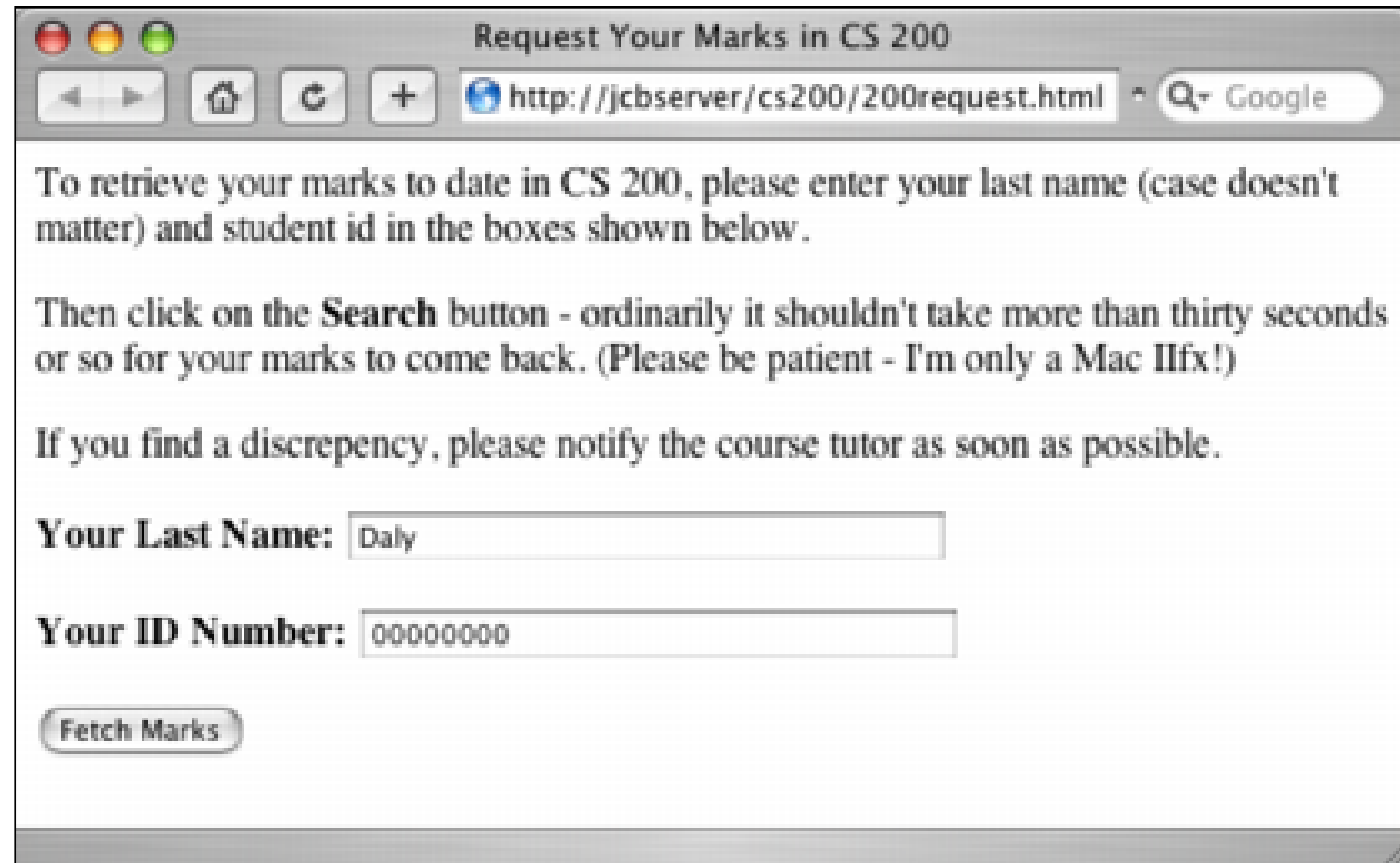
So within a web site...

- 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



The screenshot shows a web browser window with the title "Request Your Marks in CS 200". The address bar displays the URL "http://jcbserver/cs200/200request.html" and a search bar with the text "Google". The main content area contains the following text:

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 **Search** 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 IIx!)

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

Below the text, there are two text input fields:

Your Last Name:

Your ID Number:

At the bottom of the form, there is a button labeled "Fetch Marks".

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

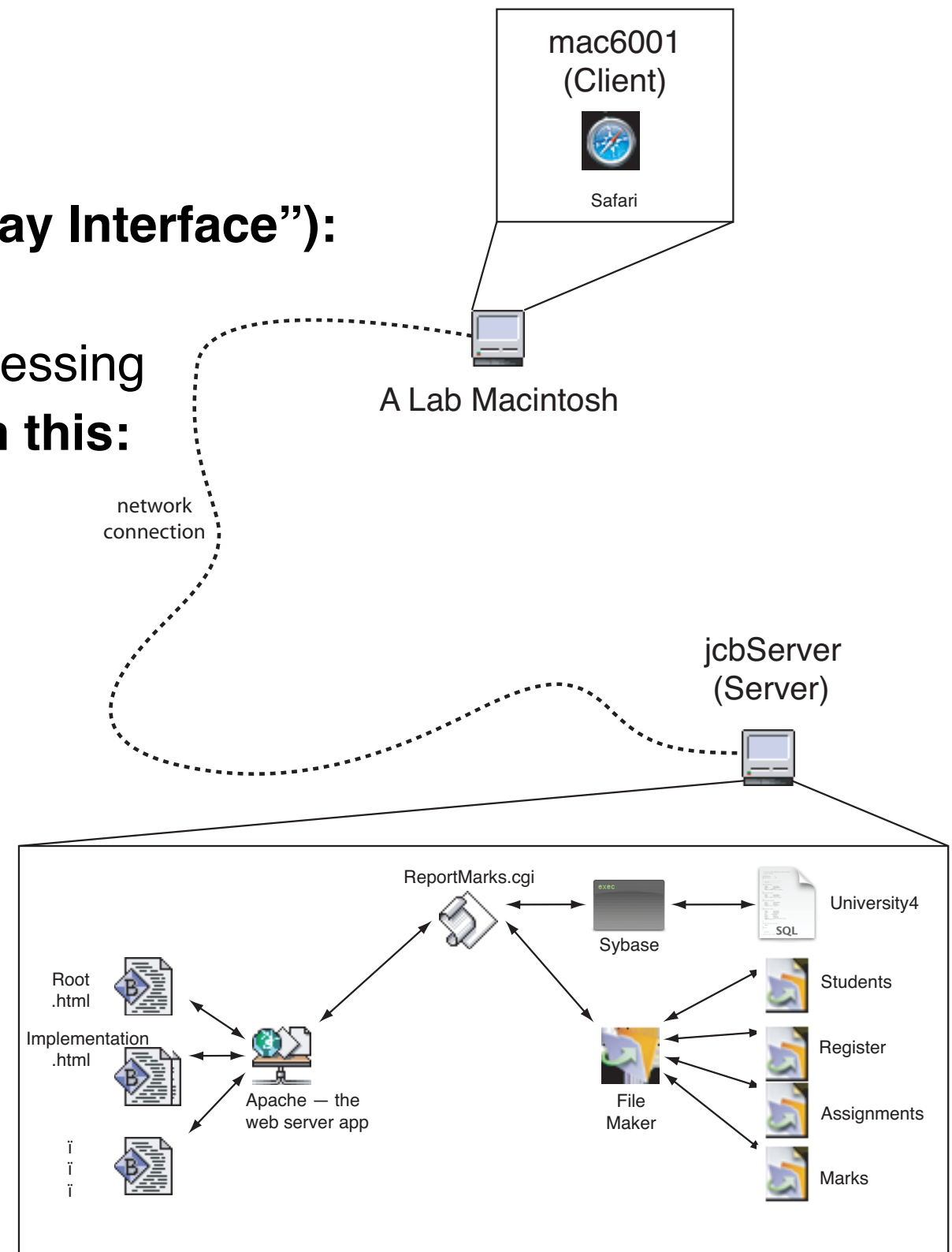
Actually, the CGI scheme is more general than this:

- 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)

T

```
<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>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 IIx!)</P>
    <P>If you find a discrepancy, please notify the course tutor as soon as
    possible.</P>
```

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

```
  <P>
```

```
    <STRONG>Your Last Name: </STRONG>
```

```
    <INPUT TYPE="text" NAME="surname" SIZE="33">
```

```
  </P>
```

```
  <P>
```

```
    <STRONG>Your ID Number: </STRONG>
```

```
    <INPUT TYPE="text" NAME="idnumber" SIZE="33">
```

```
  </P>
```

```
  <P> <INPUT TYPE="submit" VALUE="Fetch Marks"> </P>
```

```
  <P> <INPUT TYPE="hidden" NAME="course" VALUE="cs200"> </P>
</FORM>
```

```
</BODY>
</HTML>
```

The screenshot shows a web browser window titled "Request Your Marks in CS 200". The address bar shows the URL "http://jcbserver/cs200/200request.html". The page content includes instructions: "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." followed by "Then click on the **Search** 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 IIx!)" and "If you find a discrepancy, please notify the course tutor as soon as possible." Below the text are two input fields: "Your Last Name:" with the value "Daly" and "Your ID Number:" with the value "00000000". At the bottom is a button labeled "Fetch Marks".

Note the use of a “hidden parameter” that the user never sees so that forms for different courses can use the same cgi.

What Gets Sent to the Server (GET)

What comes back

GET ../ReportMarks.cgi?course=cs200&surname=Daly&idnumber=00000000 HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.06 (Macintosh; U; PPC, Nav)

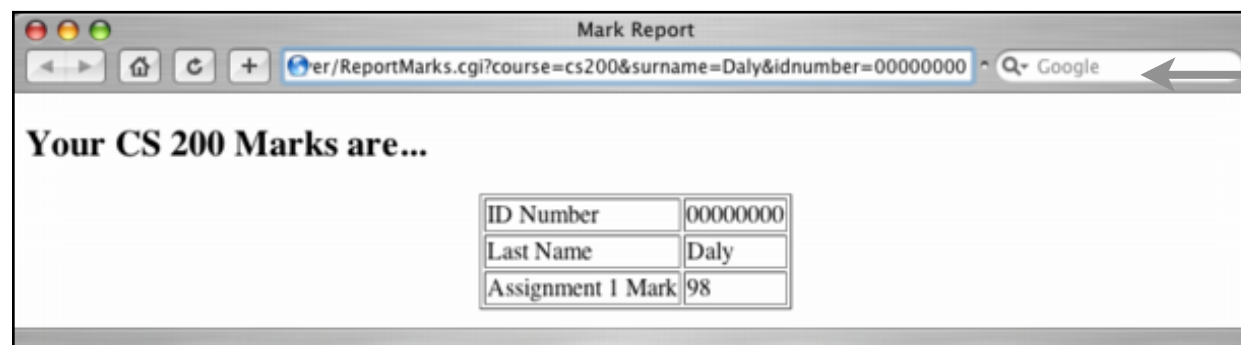
Host: jcbServer.cs.uwaterloo.ca

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."

../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!

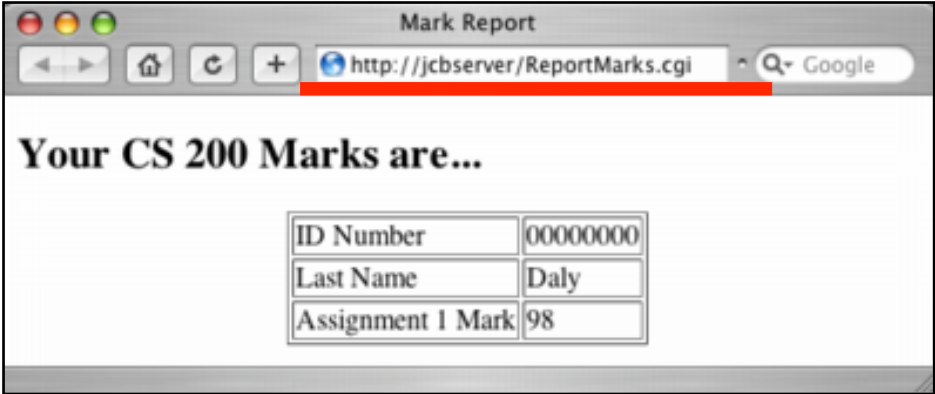
POST Actions

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=000000000
```



What is going on when a page loads?

GET	POST
Requests data from a specific resource.	Submits data to be processed by a specific resource.
Data is submitted as part of the URL	Data is submitted in the request body
Less secure but faster	More secure but slower
Can be cached by browser	Not Cached by Browser
Length Limited by URL size	MaxLength determined by server

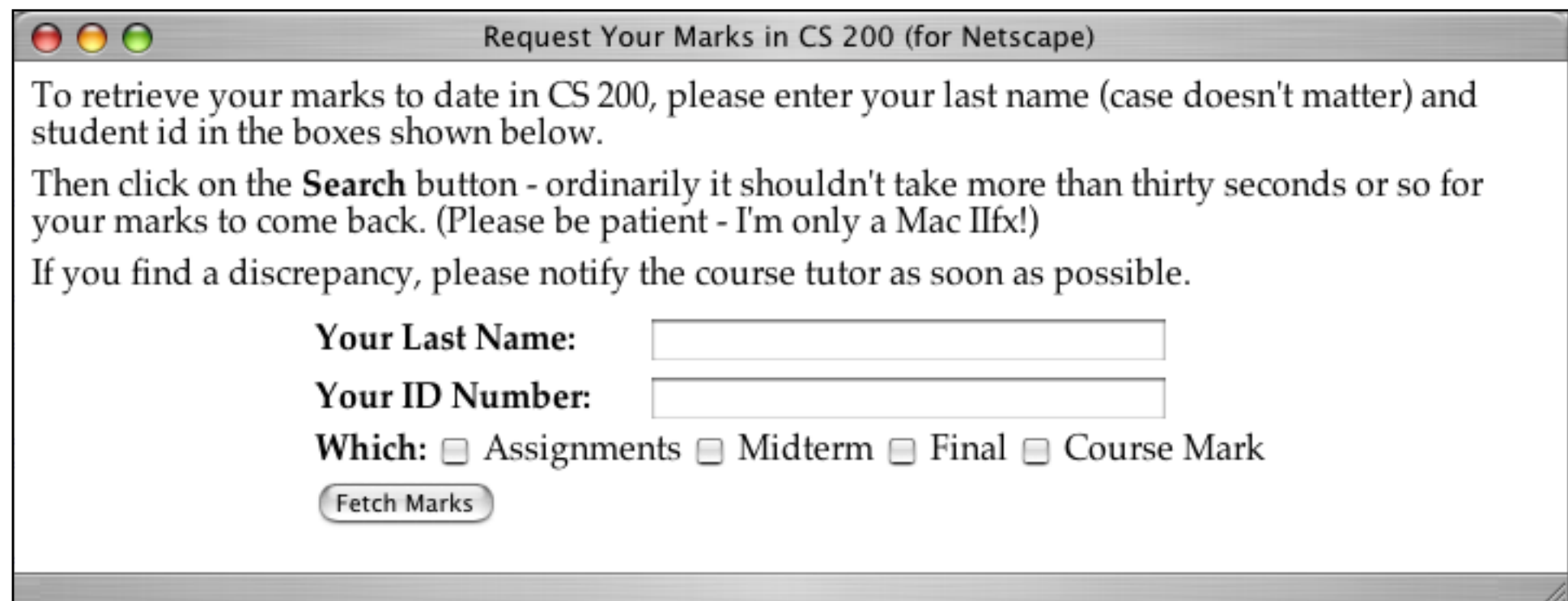
The details of the difference between POST and GET are not important to us

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*

Laying Out a Form as a Table – The Form

```
<HTML>                                <!-- The beginning of the web page - the FORM and TABLE are on the next page. -->
  <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>
```



Request Your Marks in CS 200 (for Netscape)

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 **Search** 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!)

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

Your Last Name:

Your ID Number:

Which: ☐ Assignments ☐ Midterm ☐ Final ☐ Course Mark


```
<FORM ACTION="http://ReportMarks.cgi" METHOD="POST">
  <INPUT TYPE="hidden" NAME="course" VALUE="cs200">
```

<xxx>	form tags
<yyy>	table tags

```
  <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>
```

```
        <INPUT TYPE="checkbox" NAME="which" VALUE="assign"> Assignments
```

```
        <INPUT TYPE="checkbox" NAME="which" VALUE="midterm"> Midterm
```

```
        <INPUT TYPE="checkbox" NAME="which" VALUE="final"> Final
```

```
        <INPUT TYPE="checkbox" NAME="which" VALUE="course"> Course Mark
```

```
      </TD>
```

```
    </TR>
```

```
    <TR>
```

```
      <TD ALIGN=left><INPUT TYPE="submit" VALUE="Fetch Marks"></TD>
```

```
    </TR>
```

```
  </TABLE>
```

```
</FORM>
```

```
</CENTER>
```

```
</BODY>
```

```
</HTML>
```

Styles in HTML

Most tags are named styles: , , <p>, , ...

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

Although a few are not: , <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
- 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

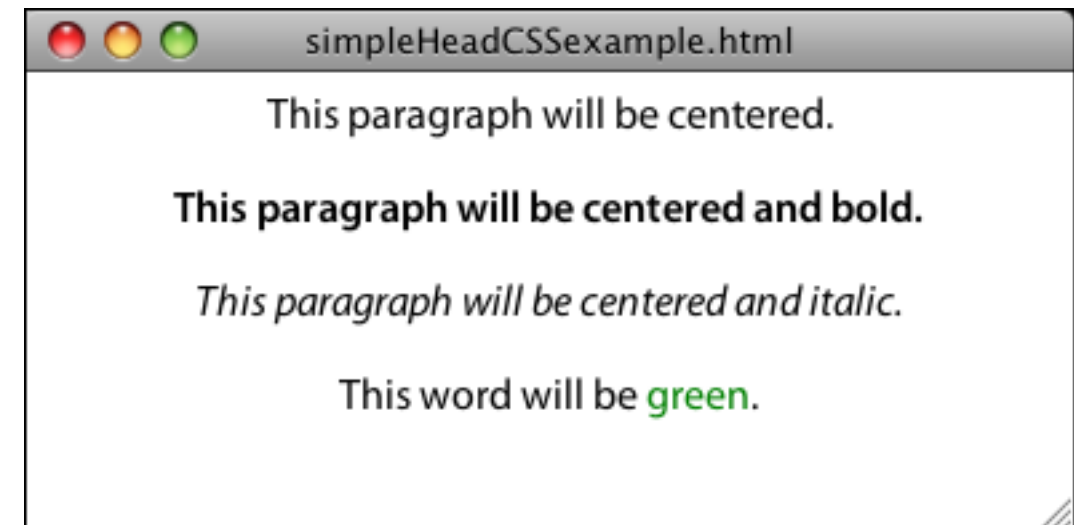
Style definitions may appear at the beginning of a web page

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 and <P>
- the last two define property bundles that can be applied to <P>, by using classes

These are called “selectors”

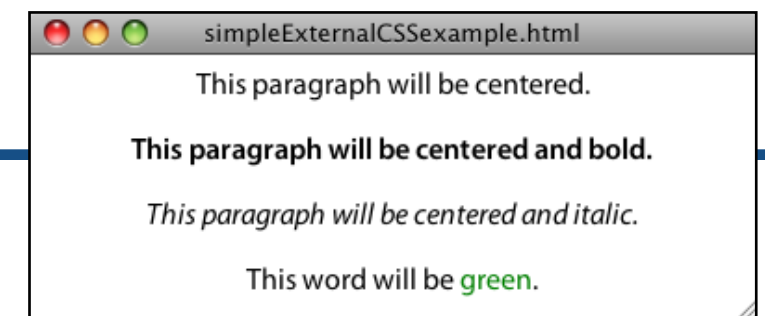
```
<HTML>
<HEAD>
  <STYLE TYPE="TEXT/CSS">
    BODY { FONT-FAMILY: 'Myriad Pro', sans-serif }
    SPAN { COLOR: green }
    P { TEXT-ALIGN: center }
    P.A { FONT-WEIGHT: bold }
    P.B { FONT-STYLE: italic }
  </STYLE>
</HEAD>
<BODY>
  <P> This paragraph will be centered. </P>
  <P CLASS=A> This paragraph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



External CSS files

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



```
BODY { FONT-FAMILY: 'Myriad Pro', sans-serif }
SPAN { COLOR: green }
P { TEXT-ALIGN: center }
P.A { FONT-WEIGHT: bold }
P.B { FONT-STYLE: italic }
```

- An example html file that uses simple.css

```
<HTML>
  <HEAD>
    <LINK REL="stylesheet" TYPE="text/css" HREF="simple.css">
  </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>
```

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

(Many browsers now let users supply a “default” file sheet.)

A Larger Example – Task B from the Winter 98 Lab Exam

T

Lab Exam – Task A

http://jcbserver.uwaterloo.ca/cs200/SampleExams/LabExam/TaskI

Google

University of Waterloo

CS 200

math.uwaterloo.ca

computer science

descriptions administrivia lectures resources project exams surveys marks

Use Menus

Introduction

Preliminaries

Task A

Task B

Task C

• cs home

• 200 home

• search

▼ people

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.

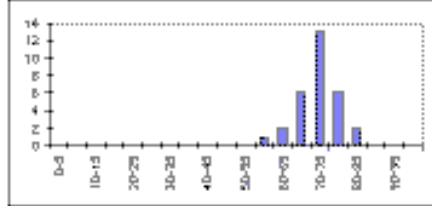
The problem. As a teacher, you often want to look at a bar chart (like the example on the right) showing the distribution of marks for a course – how many people received marks between 70 and 75, how many between 75 and 80, etc.

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



Mark Range	Number of People
0-5	0
5-10	0
10-15	0
15-20	0
20-25	0
25-30	0
30-35	0
35-40	0
40-45	0
45-50	0
50-55	1
55-60	2
60-65	4
65-70	6
70-75	14
75-80	6
80-85	2
85-90	0
90-95	0

LabExam.css

Note the period that begins each style name

- these are called *classes*
- 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)

 is an “*inline level tag*” (aka an “*inline element*”)

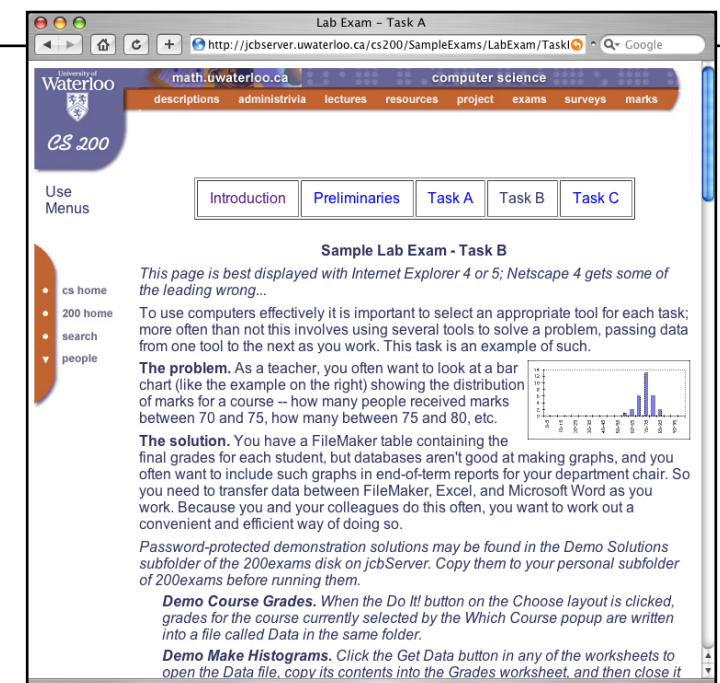
- with no built-in properties
- does *not* cause a line break
- , , and <a> are other examples

“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
- eg. Good work!

Block level tags generate automatic line breaks before & after. Inline tags do not; one can follow another on the same line.

```
.Block {
    font-family: "Palatino, serif";
    line-height: 15px;
    margin-top: 7px;
}
.Item1 {
    font-family: "Palatino, serif";
    line-height: 15px;
    margin-top: 5px;
    margin-left: 25px;
}
.Item2 {
    font-family: "Palatino, serif";
    margin-top: 5px;
}
.Footnote {
    font-family: "Palatino, serif";
    font-size: 10pt;
    line-height: 13px;
    margin-top: 5px;
}
.Emphasize {
    font-style: italic;
}
.Shout {
    font-weight: bold;
}
```



TaskB.html — the “body” part of the web page (1)

T

```
<HTML>
```

```
<HEAD>
```

```
<LINK REL="STYLE SHEET" HREF="LabExam.css">
```

```
<TITLE>Lab Exam - Task B</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<DIV CLASS="Block" style="text-align:center; font-weight:bold">
```

Sample Lab Exam - Task B

```
</DIV>
```

```
<DIV CLASS="Block" style="font-style: italic">
```

This page is best displayed with • • •

```
</DIV>
```

```
<DIV CLASS="Block">
```

To use computers effectively it is important

• • •

```
</DIV>
```

```
<IMG SRC="TaskB1small.gif" align=right>
```

```
<DIV CLASS="Block">
```

```
<SPAN class="Shout">The problem.</SPAN>
```

As a teacher, you often want • • •

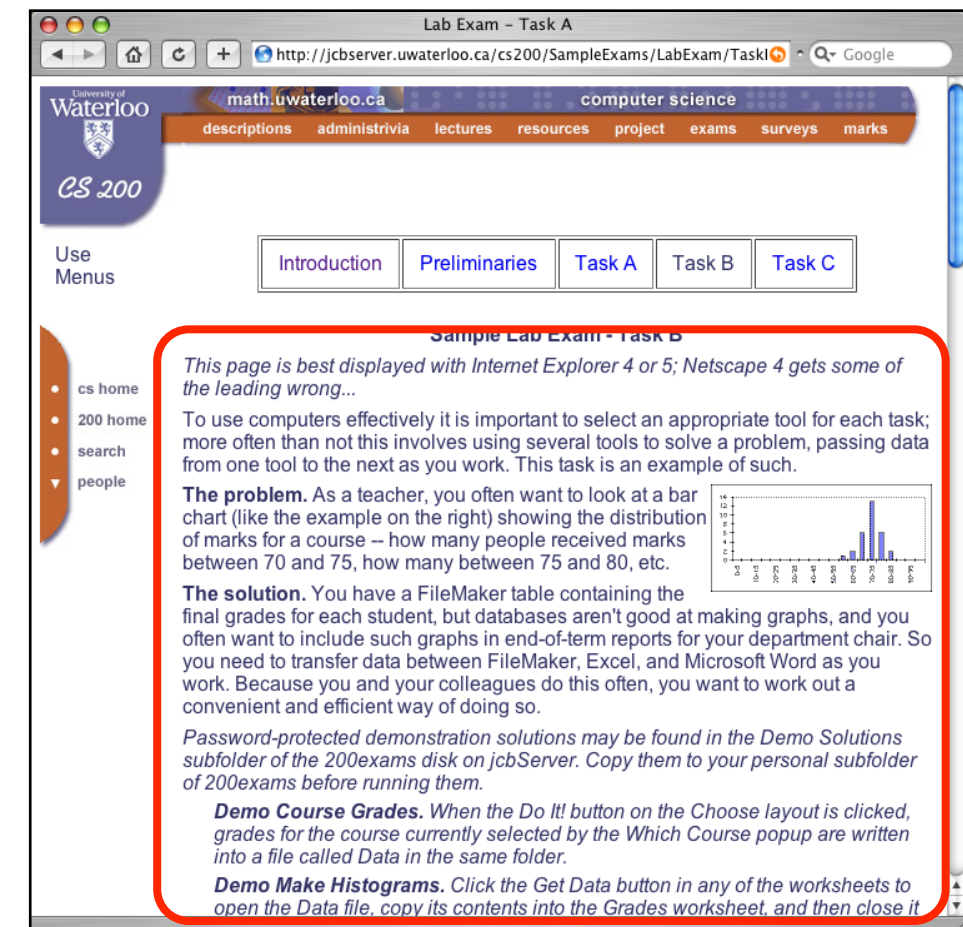
```
</DIV>
```

```
<DIV CLASS="Block">
```

```
<SPAN class="Shout">The solution.</SPAN>
```

You have a FileMaker table • • •

```
</DIV>
```



TaskB.html — the “body” part of the web page (2)

```
<I>
  <DIV CLASS="Block">
    Password-protected demonstration solutions • • •
  </DIV>

  <DIV CLASS="Item1">
    <SPAN class="Shout">Demo Course Grades.</SPAN>
    When the Do It! button on the Choose layout • • •
  </DIV>

  <DIV CLASS="Item1">
    <SPAN class="Shout">Demo Make Histograms.</SPAN>
    Click the Get Data button in any • • •
  </DIV>

  <DIV CLASS="Item1">
    The charts in By 5 and By 10 can • • •
  </DIV>

</I>

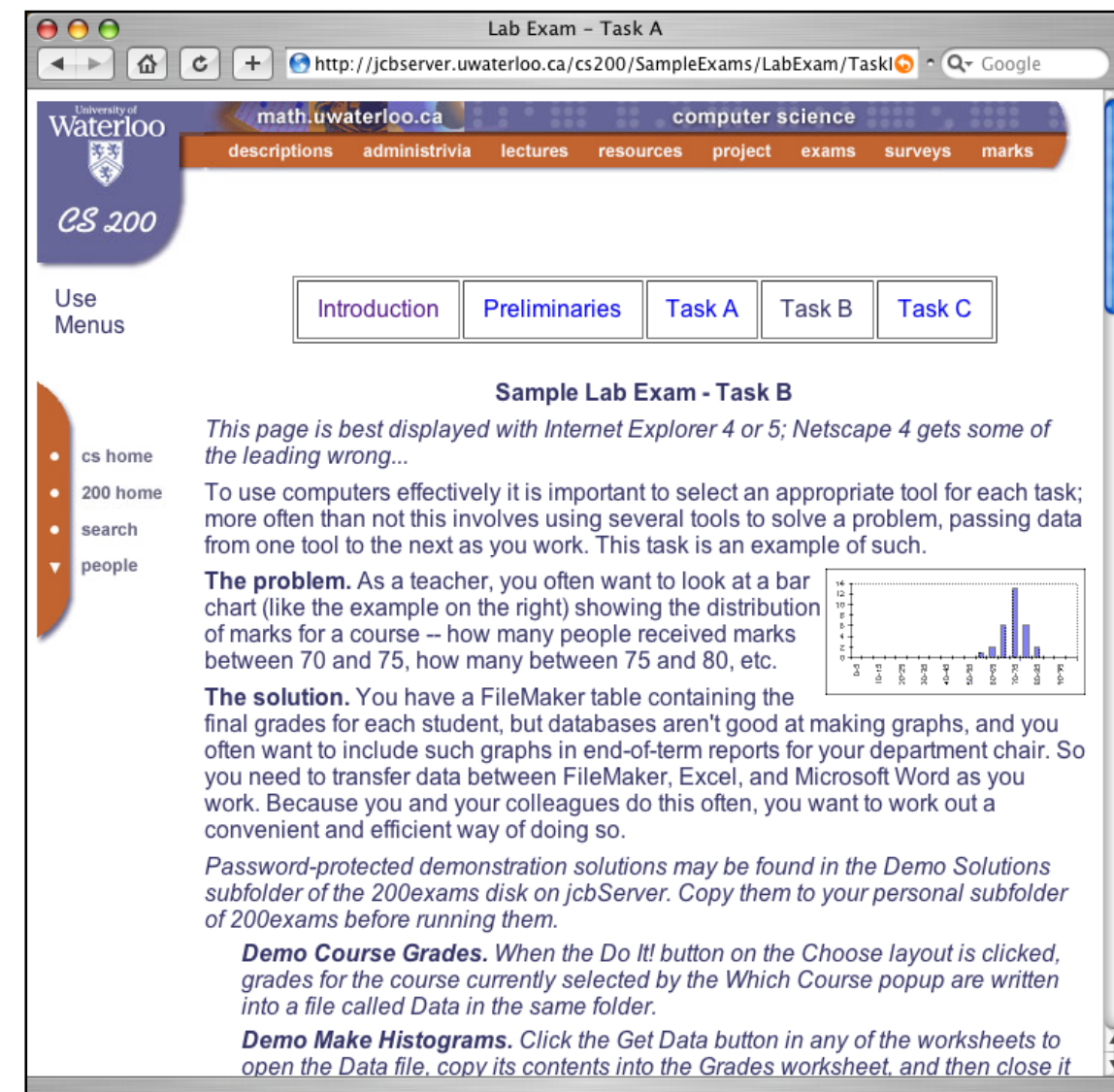
<DIV CLASS="Block">
  You are to duplicate the behaviour • • •

</DIV>

<DIV CLASS="Block">
  (The relative weight of important pieces • • •

</DIV>

• • •
```



Lab Exam - Task A

http://jcbserver.uwaterloo.ca/cs200/SampleExams/LabExam/TaskB

University of Waterloo

math.uwaterloo.ca

computer science

descriptions administrivia lectures resources project exams surveys marks

CS 200

Use Menus

Introduction Preliminaries Task A Task B Task C

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.

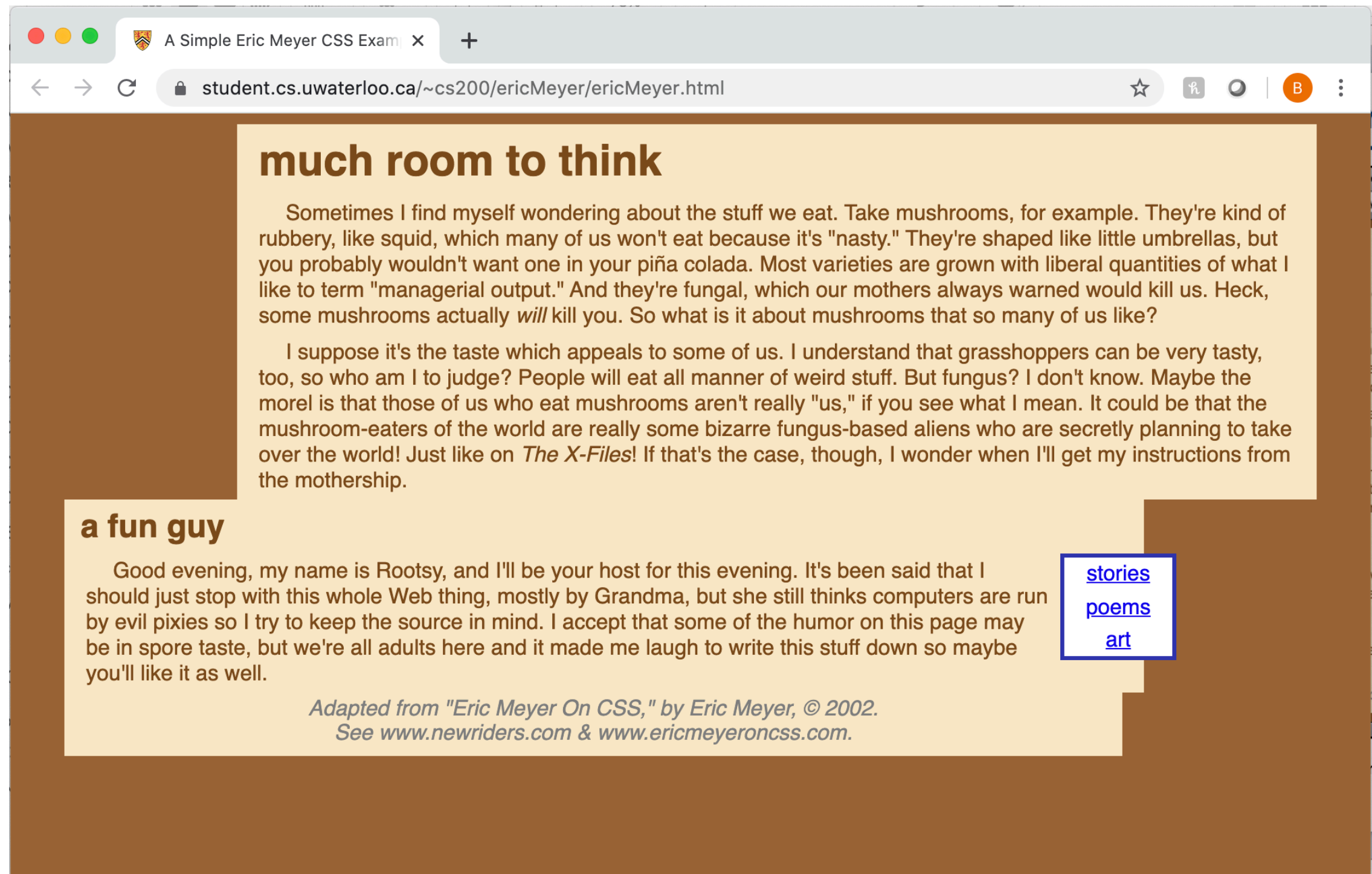
The solution. You have a FileMaker table containing the final grades for each student, but databases aren't good at making graphs, and you often want to include such graphs in end-of-term reports for your department chair. So you need to transfer data between FileMaker, Excel, and Microsoft Word as you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

Another Example, from “Eric Meyer on CSS”



<https://student.cs.uwaterloo.ca/~cs200/ericMeyer/ericMeyer.html>

The Style Sheet

$0 \leq r, g, b \leq 255$

```
body {  
  background: rgb(153,102,51);  
  color: black;  
  font-family: 'Myriad Pro', sans-serif;  
}
```

a colour name

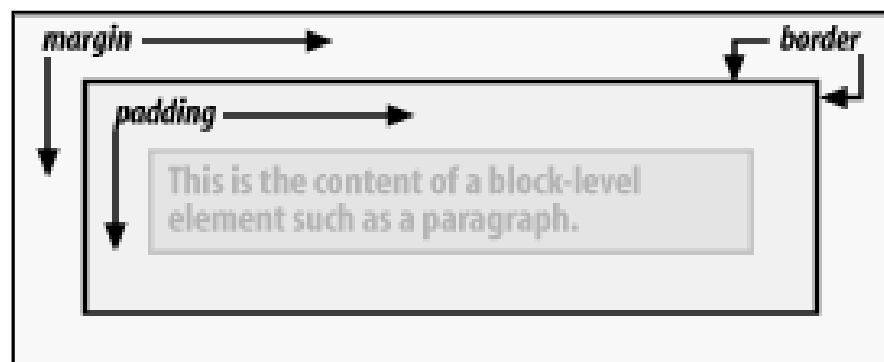
```
div.wrap {  
  background: rgb(251,233,198);  
  color: rgb(122,74,26);  
  margin: 0 2em;
```

top, right, bottom, left

```
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;  
}
```



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

```
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;
```

top, right, bottom, left

```
}  
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);  
}
```



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 only** `<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 `` appearing “immediately” *within* an `<h1>`

eg `<h1>This is very 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>`

`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> </tr>`

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>
<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">
  Adapted from "Eric Meyer On CSS,"
  by Eric Meyer, &copy; 2002.
  <br>
  See www.newriders.com &amp;
  www.ericmeyeroncss.com.
</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

Classes vs. IDs

Classes

- can be used any number of times
- defined like:

```
.red {  
  color: red;  
}
```

- the . indicates that it's a class
- this can now be applied to other styles
 - eg. `<p class="red">` will have all the attributes from `<p>` and all the attributes from `.red`
- If you want a class that can ONLY be applied to the `<p>` tag (and not to any other tags), defined it like:

```
p.red {  
  color: red;  
}
```

- You can still use `<p class="red">` with the same results as before, but `<div class="red">` has no meaning

IDs

- should only be used once (for use with JavaScript – you don't need to know why)
- defined like:

```
#red {  
  color: red;  
}
```

- this can now be applied to other styles
 - eg. `<p id="red">` will have all the attributes from `<p>` and all the attributes from `#red`

So what should you use?

- there are reasons to use both classes and IDs
- for CS 200: classes

Discussion

The “class” attribute can be used by many tags, which share its meaning
The “id” attribute is supposed to uniquely identify a tag (ie 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 (ie. 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

For More (Optional) Information on CSS

Western Civilization's

- “*Complete CSS Guide*” at http://www.westciv.com/style_master/academy/css_tutorial/introduction/css_intro.html
- “**Properties Introduction**” (the Complete CSS Guide’s section on CSS attributes) at http://www.westciv.com/style_master/academy/css_tutorial/properties/index.html

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

- 3rd edition, © 2006, O’Reilly & Associates, 0-596-52733-0.
- 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/

The “Lorem ipsum... generator” at <http://www.lipsum.com/>

Common Sources of Confusion in the Lab

You can use Firefox, Chrome or Safari's File > Open File... menu item

- 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 your text editor, **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