**COMP 205 Playbook**

**Web Page Development**

**Fall 2023**

Janis Michael

Applied Science & Computing

St Lawrence College

544-5400 ext 1925

JMichael@sl.on.ca

**Every time you create a new web page:**

1. Run Visual Code and click File Open.
2. Find your COMP205 folder.
3. Double click MyTemplate.html.
4. Do not modify this file.
5. From the File menu, pick Save As.
6. Be careful to find the appropriate folder. For example, if you are working on the hyperlinks topic, save to the hyperlinks folder.
7. Type in a meaningful filename with an html extension.
8. Click Save.
9. Fill in the title element. Save.
10. You can now code your web page.

# Your First Mission, should you choose to accept it

1. Log in to Blackboard. Find the Playbook link in the Course Information content area and open it in Word. You will download it later.
2. Read and complete steps on page 2 of the playbook.
3. Read and complete steps on page 6 of the playbook.
4. In the course outline content area on Blackboard, download the course outline and learning plan to your COMP 205 folder.
5. In the Course Information content area:
6. Download the COMP 205 Playbook. Place the file in your COMP 205 folder.
7. Download and read the delivery schedule. Place the file in your COMP 205 folder.
8. Read the textbook, student success and windows skills items.
9. Print and coil bind the playbook (encouraged, not required). It is a black and white document. You can do this wherever you like. The Student Association (copycentre@sl.on.ca) will print and coil bind the book for under $20. Email them if you are interested, attaching the playbook. I do not suggest using a campus printer as that will cost you more money in the end.
10. Decide on a backup plan. This step is crucial to your success in both school and work. Damaged data is not an excuse for late or missed coursework. The college provides OneDrive space for this purpose. It can be accessed through slc.me. You are strongly encouraged to back up all your files to this drive.
11. Friendly reminders
    1. Bring the playbook and a charged laptop to every class this semester. Mic and web cam are required for alternate delivery classes.
    2. Your college email account is required for all communications with staff and faculty. Include your student number. Check it daily.

# Setting Up Your Laptop

1. Download Visual Code: <https://code.visualstudio.com/download>
2. Run Visual Code.
3. As a programmer, you need to know complete path and filenames, including file extensions. By default, Windows has file extensions hidden. You may also need to complete the steps at your lab computer. To display file extensions:

|  |  |  |
| --- | --- | --- |
| Windows 7 | Windows 8 | Windows 10, 11 |
| Start, Computer, Organize | Run File Explorer | Run File Explorer |
| Folder & Search Options | Click the View menu | Click the View menu |
| Click the View tab | Click the Options icon then the View tab | Turn on ‘Filename extensions’. |
| Turn off checkbox *Hide extensions for known file types*. Click Apply then OK | Turn off checkbox *Hide extensions for known file types.* Click Apply then OK |  |

# Table of Contents

Getting Started ………………………………………........................ 5

Headings and Lists ……….…………………………........................ 13

Tables ……………………………………………………….………… 21

Hyperlinks ……………………………………………………..…....... 27

Images, Sound and Video …………………………………………… 31

CSS Styles …………………………………………………………… 39

Grouping Elements ………………………………………………….. 53

Layouts ……………………………………………………………….. 59

Forms …………………………………………………………………. 67

Javascript …………………………………………………………….. 73

In Class Website ……………………………………………………… 79

References …………………………………………………………… 85

Grades ….…..……………………………………………...…………. 91

**Getting Started**

If you think education is expensive, try ignorance.

(Derek Bok)

# Set Up

1. Create folder CPA Program in the Documents folder.
2. Inside the new folder, create folder First Year. Inside it, create folder COMP205.
3. Inside COMP205, create the following folders **(no spaces)**.
   1. GettingStarted
   2. HeadingsAndLists
   3. Tables
   4. Hyperlinks
   5. ImagesSoundVideo
   6. CssStyles
   7. GroupingElements
   8. Layouts
   9. Forms
   10. JavaScript
   11. YourLastName**Then**YourFirstNameInClassWebsiteV1
   12. YourLastName**Then**YourFirstNameInClassWebsiteV2
   13. Quizzes
   14. TechnicalInterview
4. Check that your Outlook time zone setting is correct. It is different than your Windows time zone setting. Open your email through slc.me. Click on the Settings Gear Icon. Click View all Outlook Settings, General>Language and time. The current time zone should be UTC-5 Eastern Time (US & Canada). Save.

# Backups

* Keeping an extra copy of your files is extremely important. Computers break down just like anything else. A virus or hacker can wipe out all your data. Then there’s the human error piece: spilled drinks, lost or dropped equipment…
* Backups are your responsibility as a student and as an employee. You need to manage your data and keep it safe. Be sure you have a system set up for your home computer. It is very stressful when you lose work right before an important deadline.
* Note that lost or damaged work is not an excuse for late assignments at school or at placement. You must have backups in place.

# Documents Folder

* On the taskbar, click the File folder icon (called File Explorer or Windows Explorer).
* The folder you really care about is the Documents folder. It is very important that you save all your files in the Documents folder. This is required for the files to be backed up.
* The college provides you with a one drive account. This is a place in the cloud (a remote computer) where you can store backup copies of your files. If your computer or a file on your computer is lost or damaged, you can get a copy from the one drive account (assuming you have set it up properly).
* Once one drive is set up, all files and folders in Documents will be backed up automatically to one drive. You must be logged in to slc.me for the back up to happen. **The only way to keep your files safe is to put them in the Documents folder and be logged in to your slc account while you are working at the computer**.

# Downloads Folder

The Downloads folder contains all items downloaded from a browser. It is typically not backed up. If you want to keep something you downloaded, move it into a relevant folder in Documents.

# Emailing Faculty and Staff

Your emails are a reflection of your work. There need to be professionally written and free of spelling and grammar errors. Tips:

1. Check your email every day.
2. Include your student number on all emails to faculty and staff.
3. All emails must be through your slc email account. We cannot accept communication from outside email addresses.
4. Proofread your email before sending.
5. Do not use Reply All unless necessary. Be aware that reply all is the default.
6. Do not begin an email with Hey.
7. You never know who your email will be forwarded to so be cognizant of what you are saying. That is, be professional.

# Simplest of Simple Web Page

Editing a web page is easy. The file itself is a regular text file. You can use any text editor for your web page. You cannot use a software package that has formatting (for example Word, WordPad).

1. Run the Visual Code editor and type in text:
   1. From the File menu, pick New Text File.
   2. Write two sentences on the music you like. Don’t worry about the content, just type something quickly.
   3. Add two blank lines.
   4. Write two sentences on the movies or TV shows you like. Don’t worry about the content, just type something quickly.
2. To save the file:
   1. Click the Save icon.
   2. Type in the name of the file: firstone.html It must have extension html
3. Press ctrl-o.
4. Find the html file. Right click it, pick Open With then pick Google Chrome.
5. Your web page should be displayed in Chrome.
6. Making changes:
   1. Make a change to the file in Visual Code. Notice the filled dot beside the filename at the top of the window. This indicates a file that has not been saved. Press ctrl-s to save. The dot disappears.
   2. Remember that it is very important that you save changes before loading or reloading a web page.
   3. Return to the browser and press F5.
7. In the above exercise, you kept both Visual Code and the browser open while working. You don’t have to have them both open but it is easier. It is more efficient to keep them both open because you can window back and forth.
8. You can use Alt-Tab to switch to the previous program. You can also hold the Alt key down and hit tab to cycle through all running programs.

# Naming Your Files

Avoid spaces and even underscores in the filename. Search engines work more effectively with hyphens than underscores. Use the appropriate extension: html.

# Adding html to firstone.html

1. Look at your web page. It isn’t organized. Let’s add some structure to it.
2. Open the playbook in Word. Paste in the following code. The code looks pretty strange right now. It will make more sense in the near future.
   1. Add the following 3 lines to the top of your web page:

<h1>My First One</h1>

<h2>Music</h2>

<p>

* 1. Place the code below after your paragraph on music and before your paragraph on movies.

</p>

<h2>Movies/TV Shows</h2>

<p>

* 1. After your paragraph on movies/TV Shows, add: </p>

1. Save it. Run it.

# Where Does This Web Page Live?

In this course, we will be doing relatively simple web pages. For this reason, all the work can sit on your local computer: you can edit it there, save it there and run it there. As you get in to more complicated web pages, you will need a server. The server not only offers you a lot more functionality but it also means that other people can access your website.

# Way Back When

The internet began as a text based means for passing information between networked computers. Someone came up with the idea of hyperlinks to move from one content page to another. Many programmers came up with mechanisms to do this but the one that stuck was Hypertext Markup Language (HTML). Hypertext is any item or object that can be linked to another object. The text is *hyper* in the sense that it links to other pages. The interlinked pages became known as the World Wide Web. Eventually, programmers started coding browsers that support text and graphics. The industry exploded fairly rapidly to where we are today.

# What is in a URL?

<http://www.stlawrencecollege.ca/pay-for-college/bursaries-and-scholarships/>

<http://www.payrolldept.fakecompany.ca/~BBunny01/paystub.html>

|  |  |
| --- | --- |
| URL | Universal resource locator: a web address. |
| Protocol | http (hypertext transfer protocol) is the standard for communicating on the web. |
| Host name | www. Typically people start their web servers with www. Users have come to expect it. It isn’t a requirement, but it is very common. |
| Domain name | Examples: .com .org .in .br .ca .gc.ca  .cn .com.lb .com.ph .gr .nz .za |
| Subdomain | This refers to all text between the host name and the domain name. |
| Page name  (optional) | Filename at the end of the url. Common extensions are .html .php. |
| Username  (optional) | A tilde in a url can mean a username. In the above example, the user is logged in as BBunny01 |

# Acronyms

Html: hypertext markup language. Css: cascading style sheets.

# What are Elements and Tags?

Elements are the strange looking items with angled brackets < >. They are programming code, commands. They don’t show on the web page but they do tell the web page what the text means. In other words, the Table element tells the web page that the text inside the element is part of a table with rows and columns. Most html elements have an opening tag and a closing tag. For example, tag *<p>* starts a paragraph and tag *</p>* ends the paragraph.

<h1>My Heading</h1> is an element.

<h1> is an open tag.

</h1> is a close tag.

# The Paragraph Element

<p>This paragraph is exciting. I 'm sure all would agree.</p>

The terminology is as follows:

The paragraph **element** consists of an open paragraph **tag**, **content** then a closing paragraph **tag**.

Html tags should appear in lower case. Older html standards often suggested tags appear in upper case. This is not the current standard.

# Special Characters

Using your first one page, add the following line in the middle of a paragraph:

I like <some artist> very much.

Save and run. Notice the artist does not display. The < and > characters are confusing the browser. Fix the problem by changing the less than sign to *&lt;* and the greater than sign to *&gt;* Note the ampersand before and the semicolon after. Save and run. Add and test the code below:

&copy; Your name here

For a list of special characters, see <http://dev.w3.org/html5/html-author/charref>

# The doctype Element

<!doctype html>

This should be at the top of every html file. It tells the browser you are using html version 5. You will see variations on this command. They represent earlier versions of html. Add this element to your firstone.html file. Save it then refresh the browser.

# The Comment Element

Comments are used in programming languages to help humans. They can be placed anywhere in the document. To include a comment in your html code:

<!--The web browser ignores this. I can put whatever I want in here -->

Add your name in a comment on line 2 of your firstone.html file. The first line should always be the doctype element. Save it then refresh the browser.

# The html Element

<html lang = "en">

all your html code

</html>

This element tells the browser that everything between the start html tag (*<html>*) and the end html tag (*</html>*) is html code. This element also specifies the written language used (English). Screen readers use the lang attribute to determine pronunciation. There is only one html element in an html document. It is required on every page.

Place the open tag right after the comment. The closing tag should be the last line of the file. Save it then refresh the browser.

# The head, title and meta Elements

Add the code below after the open html tag in firstone.html file.

<head>

<meta charset = "UTF-8">

<title>Awesome title</title>

</head>

The head element appears near the top of your code. It provides information about the page. Its contents are not displayed in the browser but are important for successful display of your pages.

The title element is housed inside the head tag. It describes the title of the page. Most likely, the title will appear in the title bar at the top of your browser window. Indent all code between the open head tag and the end head tag.

There is only one title tag in an html document. It is required. This tag can be very useful. How exactly the title element is used is both browser dependent and client dependent. Possible uses:

* Search engines use it to provide a link to search results and to give the user an idea of the purpose of the webpage.
* Browsers use it to identify the page for bookmarks.

The meta tag is placed inside the head tag. It specifies the character set used. All English web pages use the UTF-8 character set. This allows your browser to display a large variety of characters including those from other languages. For example, (© ή ج ☻). This tag does not have an end tag. It is *self-closing.*

# The body Element

The body tag encompasses all data that will be displayed in the browser. There is only one body tag in an html document.

Add this element to your firstone.html file. Place the open body tag after the closing head tag. Place the closing body tag right before the closing html tag.

# A Template html Document

Once you are sure your firstone.html page is correct, use it to create a template html file. Steps:

1. Open firstone.html in Visual Code.
2. Save the file to your COMP205 folder as MyTemplate.html.
3. Delete all text between the <body> tag and the </body> tag.
4. Delete all text between the <title> tag and the </title> tag.
5. Make sure the file contains the following elements: doctype, comment with your name, html, head, meta, title, body.
6. Run the file in Chrome. The page should be blank.

**The template is a very important file. You will use it everyday in this class. Be sure you have it completed correctly before our next class.**

# The Footer Element

Use a footer element at the bottom of your web pages to provide a consistent look across the bottom of every page in the site. For example:

<footer>&copy My Great Company</footer>

Not all of our pages will have a footer element. For this reason, the footer element is not part of your template.

# One Drive Verification

Make sure your folders are in the correct place and backed up to one drive. If this is not done properly, your data will not be backed up and you could lose all your coursework.

To verify your work is backed up to the cloud (one drive):

1. Go to slc.me and sign in.
2. Click the 9 dot waffle in top left and pick OneDrive.
3. You should be able to see the folders you just created. Double click the semester 1 folder and verify all folders all there. Do the same with the semester 2 folder.

# At the End of Every Class

Check your one drive account shows the day’s files. Steps can be found under the One Drive Verification heading in this document. You are strongly encouraged to do this anytime you are working at a computer.

# Taking Flight

Your goal is to create a web page containing flight information. To save typing and ensure accuracy, start with your html template:

1. Load MyTemplate.html in Visual Code.
2. From the File menu, pick Save As. Save the file to the COMP205\GettingStarted folder, calling it Flights.html.
3. Change the title element.
4. Make sure the code you write is inside the body element: between <body> and </body>.
5. Code a main heading of Flights (h1).
6. Include 3 countries of your choice (h2). For example, an h2 heading that reads Canada.
7. Add 4 flights for each country (h3), for example Toronto to Victoria.
8. Add a footer element with your name and the date.
9. Test it.

**Headings and Lists**

Education is not the filling of a pail, but the lighting of a fire.

(William Yeats)

# Set Up

1. Log in to Blackboard.
2. Click the Headings and Lists group on the left hand side.
3. Download and unzip the HeadingsAndLists file, storing the files in COMP205\HeadingsAndLists.

# Heading Tags

The heading tags work with css (coming later) to display your text in various fonts. The heading tag is not at all related to the head element so try not to confuse the two.

In headings1.html, there is a hierarchy to the heading elements. For example fox and deer are mammals. When designing your code:

* There should be only one h1 tag.
* Design headings based on a logical hierarchy or based on categories.
* Do not select your heading elements based on the look of the heading. In other words, don’t use an h2 heading because you like its font size. Instead, choose based on category (ex reptiles).
* Do not indent after a heading element.
* Use css (coming later) to define the look of your headings.

# In Class Exercise

Your supervisor has provided you with the content of a web page (shown below). Your first task is to determine an appropriate hierarchy. Beside each heading, write the heading used (h1, h2, h3, h4). You are not allowed to add extra words or move text.

Vehicles

Cars

Civic

Cruze

Corvette

Trucks

Ford truck

Half ton Ford 150 truck

Three quarter ton Ford 250 truck

Dodge Ram truck

Boats

Ocean liner

1000 feet long

Row boat

15 feet long

Wooden oars

# A Quick Look at Basic CSS

So far in the course, we have been working with the html programming language. The css programming language works with html to tailor the look of your web pages.

The following code is placed after the open title tag:

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

File nature.css needs to be in the same folder as nature.html.

# Samples of Lists

|  |  |  |
| --- | --- | --- |
| Unordered List | Ordered List | Definition List |
| * Canada   + Calgary   + Charlottetown   + Kingston * India * China * Egypt * Italy * Mexico * South Africa | 1. Log in to your account. 2. Email your prof with your:    1. name and program.    2. interests.    3. experience in:       1. programming.       2. databases. 3. Chill. | COMP 205  Html  Css  Javascript  COMP 1111  Access  Databases  COMP 1100  C++ Prog |

# Characteristics of Lists

A list is comprised of multiple items. Each item begins on a new line and is indented from the left margin. Items in an unordered list are preceded by a bullet. Items in an ordered list are preceded by a number or letter. Definition lists do not have bullets or numbers. Use css to change the appearance of your list.

# Unordered Lists

This list starts and ends with tags: <ul> and </ul>. Each item in the list is surrounded by <li> and </li> tags. For debugging purposes, it is important that you indent after <ul>.

# Ordered Lists

This list starts and ends with tags: <ol> and </ol>. Each item in the list is surrounded by <li> and </li> tags. For debugging purposes, it is important that you indent after <ol>.

# Definition Lists

This list is useful when you want to provide a list of items but do not want bullets or numbers. To produce a definition list, 3 elements are used. This list starts and ends with tags: <dl> and </dl>. When trying to remember how they work, think about the acronyms: Definition List (dl), Definition Terms (dt) and Definition Descriptions (dd).

# ListExamples.html

<ol>

<li>iPhone</li>

<li>Android</li>

<li>Blackberry</li>

</ol>

<ul>

<li>Canada</li>

<li>India</li>

<li>China</li>

<li>Egypt</li>

<li>Italy</li>

<li>Mexico</li>

<li>South Africa</li>

</ul>

<dl>

<dt>COMP 205</dt>

<dd>Html</dd>

<dd>Css</dd>

<dd>Javascript</dd>

<dt>COMP 1111</dt>

<dd>Access</dd>

<dd>Databases</dd>

<dt>COMP 1100</dt>

<dd>C++</dd>

<dd>Programming</dd>

</dl>

# List Exercise 1

Write code to produce the three lists shown below:

1. Sleep
2. Eat
3. Study
4. Workout
5. Have fun

* Leopard
* Panther
* Lion
* Tiger

Monkeys

Macaque

Baboon

Langur

Reptiles

Crocodile

Python

# Nested Lists

Lists are powerful but can be very tricky. The correct placement of the closing tags is crucial. A few tips are shown below.

1. Create the outer list first and test fully.
2. Add the inner list:
   1. The bullet before the nested list should not have an end li tag. That is, remove the </li>.
   2. Code the inner list.
   3. Add the </li> below the inner list.
3. Repeat the previous step until done, testing fully after each new addition.

Nested Unordered List

<ul>

<li>Canada <!-- notice there is no /li here -->

<ul>

<li>Calgary</li>

<li>Charlottetown</li>

<li>Kingston</li>

</ul>

</li> <!-- notice this ends the Canada li -->

<li>India</li>

<li>China</li>

<li>Egypt</li>

<li>Italy</li>

<li>Mexico</li>

<li>South Africa</li>

</ul>

Nested Ordered List

<ol>

<li>Log in to your account.</li>

<li>Email your prof with your: <!-- notice there is no /li here -->

<ol>

<li>name and program.</li>

<li>interests.</li>

<li>experience in:

<ol>

<li>programming.</li>

<li>databases.</li>

</ol>

</li> <!-- notice this ends the experience li -->

</ol>

</li> <!-- notice this ends the email li -->

<li>Chill.</li>

</ol>

# Indentation

Right now, indentation might seem like a lot of work. I’m sure you can imagine that your html files are going to get very long. You will spend time coding of course but you will spend a much greater amount of time debugging and modifying. These tasks become extremely difficult without indentation and organization.

# Output Exercise

Working on your own, draw the output from following code snippet. Remember to include indentation in your output. No computers allowed. If you have a pencil, use it.

<ol>

<li>COMP 205 Test

<ol>

<li>Review in class work</li>

<li>Look at asst marking sheet</li>

<li>Prepare study sheet</li>

<li>Ace test</li>

</ol>

</li>

<li>Have fun

<ul>

<li>Friends Time

<ul>

<li>Dinner</li>

<li>Sports</li>

<li>Game</li>

</ul>

</li>

<li>Play music</li>

</ul>

</li>

<li>Call family</li>

</ol>

# In Class Website

Your client has provided you with the layout design. This is the layout you are to use for your in class website. Review the layout design then complete the steps below. The layout design can be found in the *In Class Website* section of this document.

Home Page

1. Load your html template file (MyTemplate.html) in Visual Code.
2. From the File menu, pick Save As. Save it to the COMP205\YourLastNameThenYourFirstNameInClassWebsite**V1**folder, calling it index.html. This will be the home page for your website.
3. Add relevant text for the title tag.
4. Add h1 and h2 headings.
5. Don’t worry about the hyperlinks for now.
6. Add multiple paragraphs. The content is to be 200 to 400 words of content related to your topic. This page is the user’s introduction to your website, company and/or products.
7. Include your footer text below the list.
8. Test your code before moving to the next point.
9. Somewhere below the headings and above the footer, add an ordered list with 4 (or more) points. Test it.
10. Add a nested unordered list inside the third point. Include two or more items in the list. The content of your lists should relate to your website topic.

Finally

Work through *Checklist for in class website (at end of lists topic)*. It can be found in the in class website section of this playbook.

**Tables**

Learning is what most adults will do

for a living in the 21st century.

(Perelman)

# Friendly Reminder

If looking for help, look through the playbook (including the References section) and provided examples.

# The Table Element

The table tag is very useful and verbose. In some ways it is intuitive to use but can be difficult to debug. Indentation is key. Your html code will define the structure and content of the table. With css code, you can customize the look of the table: the table layout options are extensive.

|  |  |  |
| --- | --- | --- |
| **Opening Tag** | **Closing Tag** | **Description** |
| <table> | </table> | Start and end the table. |
| <tr> | </tr> | tr: table row.  One complete table row. |
| <th> | </th> | th: table heading. Optional.  One piece of data earmarked as a heading (has unique formatting) |
| <td> | </td> | td: table data.  One piece of data in the table (one cell). Four td elements means four columns in the table. |

# BasicTable1.html

|  |  |  |
| --- | --- | --- |
| **Hero** | **Power** | **Nemesis** |
| The HTMLator | Standards compliance | Sloppy Code Boy |
| Captain CSS | Super-layout | Lord Deprecated |
| Browser Woman | Mega-Compatibility | Ugly Code Monster |

<table border = "1">

<tr>

<th>Hero</th>

<th>Power</th>

<th>Nemesis</th>

</tr>

<tr>

<td>The HTMLator</td>

<td>Standards compliance</td>

For now, we are using border = "1". We need border = "1" so that the table will be displayed with lines. Later, we will substitute border = "1" with css code.

<td>Sloppy Code Boy</td>

</tr>

<tr>

<td>Captain CSS</td>

<td>Super-layout</td>

<td>King Deprecated</td>

</tr>

<tr>

<td>Browser Woman</td>

<td>Mega-Compatibility</td>

<td>Ugly Code Monster</td>

</tr>

</table>

# BasicTable2.html

<table border = "1">

<tr>

|  |  |  |
| --- | --- | --- |
| **Breakfast** | cereal | bagel |
| **Lunch** | sandwich | salad |
| **Dinner** | chicken | steak |

<th>Breakfast</th>

<td>cereal</td>

<td>bagel</td>

</tr>

<tr>

<th>Lunch</th>

<td>sandwich</td>

<td>salad</td>

</tr>

<tr>

<th>Dinner</th>

<td>chicken</td>

<td>steak</td>

</tr>

</table>

# Table Building Steps

1. Plan it. It will make coding easier. Make a rough sketch on paper or in Word.
2. Create the table element and headings. Test.
3. Build one sample row with correct number of columns. Add one letter in each cell. Test.
4. Copy the working row for as many rows as you need. Test.
5. Add the data for the table. Test.

Try It Out

*Step 1:* Plan it. It will make coding easier. Make a rough sketch on paper.

|  |  |
| --- | --- |
| **CPA** | **CST** |
| 3 years | 2 years |
| Programming | Networking |
| Java | Python |
| 28 weeks WP | 14 weeks WP |

*Step 2:* Create table element and row 1 headings. **Test**.

<table border = "1">

<tr>

<th>CPA</th>

<th>CST</th>

</tr>

</table>

*Step 3:* Build one sample row with correct number of columns. Add one letter in each cell. **Test**.

<tr>

<td>a</td>

<td>a</td>

</tr>

*Step 4:* Copy the working row for as many rows as you need. **Test**.

*Step 5:* Add the data for the table. **Test**.

*Step 6*: Go back to the html file and do the click test.

# BudgetTable.html

|  |  |  |
| --- | --- | --- |
| **Student Monthly Budget** | | |
|  | **School Year** | **Parent’s House** |
| Rent | 500 | 500 |
| Food | 200 | 0 |
| Internet | 50 | 0 |
| Entertainment | 40 | 25 |
| Summer is $265 cheaper per month. | | |

<table border = "1">

<tr>

<th colspan = "3">Student Monthly Budget</th>

</tr>

<tr>

<th> </th>

<th>School Year</th>

<th>Parent’s House</th>

</tr>

<tr>

<td>Rent</td>

<td>500</td>

<td>500</td>

</tr>

<tr>

<td>Food</td>

<td>200</td>

<td>0</td>

</tr>

<tr>

<td>Internet</td>

<td>50</td>

<td>0</td>

</tr>

<tr>

<td>Entertainment</td>

<td>40</td>

<td>25</td>

</tr>

<tr>

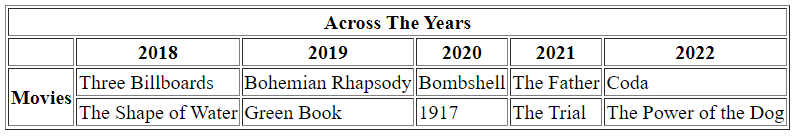
<td colspan = "3">Summer is $265 cheaper/month.</td>

</tr>

</table>

# Tables Exercise 1

Using the Table Building Steps two pages back, create the following table:



Hint: create the 10 movie names first and check it in a browser. Next, add in the Across The Years row and check in browser. Finally, add the rowspan.

# In Class Website

Goal: design and create the Tables page for your in class website. Remember to use the layout design provided by your client.

Copy your index page to a file called TablePage.html. Delete the paragraphs and lists. Change the title element. Using the Table Building Steps two pages back, code a table. The table should have a minimum of 5 rows and 4 columns **with relevant data inserted** in the cells. Include at least one colspan and one rowspan. The first row should be bolded headings.

**Hyperlinks**

A man’s errors are his portals of discovery.

(James Joyce)

# Design Decisions

To the user, a link should be recognizable as a link. When a person sees underlined text, they assume it is a hyperlink. For this reason, only use underlining for links.

# The anchor Element

<a href = "http://www.stlawrencecollege.ca" title ="slc" >

Awesome College opens in current tab</a>

<a target =\_blank href = <http://www.stlawrencecollege.ca>

title ="slc" >Awesome College opens in new tab</a>

The a means anchor, href is hypertext reference. Enclose the web address in double quotes. When html reads the code, it will underline the text Awesome College. The url must be valid, the text inside the html code can say anything and the link will still work. The title attribute indicates the text displayed when the user hovers over the link.

A link can be placed inside a paragraph.

<p>A hyperlink can be placed inside a paragraph:

<a href = "http://www.stlawrencecollege.ca" title ="slc" >

St Lawrence College</a>. It appears in the flow of the text. </p>

Media links:

<p><a href="PrideFlag.jpg" title ="pride" >Flag</a></p>

<p><a href="gain.pdf" title ="advice" >PDF file displays</a></p>

<p><a href="gain.pptx" title ="advice">

PowerPoint file downloads</a></p>

Email link:

<p>Feel free to contact me at any time. Email Janis:

<a href="mailto:JMichael@sl.on.ca">JMichael@sl.on.ca</a>.</p>

# Internal and External Links

When coding an external link, you include the full web address. The term for this is **absolute reference**. URL www.yoursite.com/index.html is an absolute reference because it contains the full web address (the text you would type in to a browser’s address bar). Absolute references are used to refer to a site somewhere else on the internet.

To refer to a web page on your site from within your site, you would use a **relative reference**.

When building your website, you will want to set up links to web pages that are within your own website. For example, you want a link to the About page from your Home page. To set up an internal link, make sure all website files are in the same folder. Next, create an anchor containing **only the file name and extension**. For example, your Home page would contain:

<a href = "About.html">About Us</a>

Note that the web address ([www.yoursite.com/](http://www.yoursite.com/)) is missing. The above line is a relative reference. The reference is relative to the current location. That is, the browser will look for file About.html in the same folder that holds the Home page.

# Links Exercise 1

Many web pages display links within a list. Using your template, create a new web page. Save the new file to the appropriate folder. Write code to produce the unordered list below:

* [COMP 205 Text](http://www.aharrisbooks.net/haio) Good reference
* [W3Schools](http://www.w3schools.com/) Web programming help

You will need the urls listed below:

[www.aharrisbooks.net/haio](http://www.aharrisbooks.net/haio)

[www.w3schools.com](http://www.w3schools.com)

Try adding the text below between the *a* and the *href* of the COMP 205 anchor element.

*target=****\_****blank*

# In Class Website

The goal now is to implement the hyperlinks in your website. Remember to use your client’s layout design.

The layout design shows links to four pages.

* Work on the Home page first. Create four links: Home, Table, Images, Form. When the link is clinked, the new page should open in the same tab. For now, the Images and Form page links do not work and the links will not be visually appealing.
* Once you have done the Home page, copy and paste the links to the Table page. The same four links should appear on every page of the site.
* Using your college email, add an email link to the footer of the Home page. Once you have tested it, copy the email link to the Table page.

Work through *Checklist for in class website (at end of Hyperlinks topic)*. It can be found in the in class website section of this playbook. Make sure your site contains all the items on the checklist before moving on. If something is missing, add it to the site.

**Images, Sound and Video**

Learning is a treasure that will follow its owner everywhere.

(Chinese proverb)

# The img Element

The img element is used to display an image.

<img src = "puppy.jpg" height="480" width="640" alt="cute pup"/>

The above code embeds an image in to the web page. Notice the tag ends with /> instead of </img>.

The src attribute specifies the name of img file. If no path is supplied (as above), the browser will look for the file in the folder that holds the html file. The height and width indicate the size of the image on the page. Use this feature to decide the size of the image relative to the other items on the page and to define the aspect ratio.

The alt attribute is required. It specifies the text displayed when the image cannot display. It is used by screen readers and meets accessibility guidelines. It is also used by the Google images search engine. If the user has turned off images or if the browser cannot find the file, the *cute canine* text is displayed.

Watch Yourself

There is a tendency to use the height and width attribute to set the size of the image. Avoid this quick fix. Think of this scenario. You have a large image but you want it to appear smaller on the web page. You decide to short cut it and use the height and width property to make a large image appear smaller. When the page is loaded, the browser loads the large image file and then scales it smaller. The page will take longer to load. Users will get annoyed and they may or may not wait. If you want to resize the actual image, use software designed for this purpose.

The height and width attributes are optional but recommended. If you omit them, the browser will calculate the size. However, while it is calculating, the browser won’t load the rest of the page. It is preferable for users to see other page elements over a blank page.

# Tips For Working with Images

* Use software to resize your images so the file size is smaller.
* Use the height and width attributes of the img element to specify the size of the image on the page. Be sure to maintain the correct aspect ratio.

# Compressing Images

There are a variety of tools you can use to compress your images. To compress your file using Paint:

1. Load the file in paint.Click the resize icon. Resize by percent. Try 50%.
2. Save the file with a different name. Look at the file size in windows explorer. If the file is too large, you can reduce by a further 50%. You will need to make sure the image is still a high enough quality to display well on the web page.

# Image Formats

Your choice of image format (png, jpg, svg, gif…) will vary based on the requirements of the image and the web page. Considerations include: file size, file compression without data loss, colour quality and copyright. There are advantages and disadvantages to each image format. Some formats take up more space than others.

# Terminology

Given:

<img src = "puppy.jpg" height="480" width="640" alt="cute pup"/>

The above code uses the img *element*. The src *attribute* is set to *value* "puppy.jpg". The height *attribute* is set to *value* "480".

# Absolute and Relative Paths

When we discussed hyperlinks, we talked about absolute and relative links.

<a href = "http://www.sl.on.ca">sl</a>

is an absolute link (also called absolute reference). It is a full url that refers to an external website.

A relative link (or relative reference) uses the current server location. For example:

<a href = "About.html">About Us</a>

is a relative link. Since no path is specified, the browsers looks in the current folder for the html file.

A Special Folder For Images

A completed website consists of many different files in a variety of file types. Style dictates that images be placed in their own folder. Later, you will create folder imgs inside your website folder. The src attribute of the img element now becomes:

src = "./imgs/puppy.jpg"

Here’s where things can get a little confusing. Paths and folders under Windows often use backslash (\). When writing code, use the forward slash (/). Both slashes work in a browser.

An Illustration

A synonym for folder is directory. The following directory structure exists on the C drive.

C:/

SLC

Courses

COMP205

Tables

Links

Asst1

MySite

COMP1100

COMP1111

In the table below, assume the current directory is **C:/SLC/Courses/COMP205**

|  |  |
| --- | --- |
| Relative Reference | Directory Path |
| . is the current directory | C:/SLC/Courses/COMP205 |
| .. is the parent directory | C:/SLC/Courses |
| ../.. is the parent of the parent | C:/SLC/ |
| / is the root directory | C:/ |

When coding images to display on a website, pay attention to both the placement of the file (which folder) and the reference to the file in your code. In the table below, assume the current directory is **C:/SLC/Courses/COMP205/MySite** That is, your html files are in the MySite directory.

|  |  |  |
| --- | --- | --- |
| Code | Where the browser will look for the image | Directory Path |
| src = "puppy.jpg" | Current folder | C:/SLC/Courses/COMP205/MySite |
| src = "./imgs/puppy.jpg" | The imgs folder inside the current folder | C:/SLC/Courses/COMP205/MySite/imgs |
| src =  "../Tables/puppy.jpg" | Up to the COMP205 folder then down in to the Tables folder | C:/SLC/Courses/COMP205/Tables |
| src = "/puppy.jpg" | Root folder | C:/ |

Relative References Exercise

Your usb drive has a directory structure of:

E:/

Sports

Karate

Soccer

Indoor

Outdoor

Swimming

Assume the current directory is **E:/Sports/Karate**. Write the directory path (without the filename) for each of the following:

**/**

**..**

**.**

**../..**

Assume the current directory is **E:/Sports/Karate**. Write the directory path (without the filename) that will be used by the browser to find the image.

src = "kata.jpg"

src = "./imgs/kata.jpg"

src = "../kata.jpg"

src = "../../kata.jpg"

src = "../Swimming/kata.jpg"

src = "/kata.jpg"

# Sound and Video

There are a number of approaches you can take with audio and video files. A few are included here. If you would like to learn more, look at linked in learning and google. Before using any files, make sure you have a legal right to do so. If you are unsure, get permission. Depending on your current browser settings, you may have to enable active content. Steps:

1. Click the Gear icon in the top right of the IE.
2. Click Internet Options then the Advanced tab.
3. Under the Security heading, turn on checkbox *Allow Active content to run in files on My Computer.* Click Apply then click OK.

# Audio

<audio controls = "controls">

<source src = "SevenYearsOld.m4a" type = "audio/mpeg">

<source src = "SevenYearsOld.ogg" type = "audio/ogg">

Your browser does not support HTML5 Audio

</audio>

The audio element specifies where on the page the sound controls will appear. The controls attribute tells the browser to display a panel to control the audio (play, pause, volume, playback). The audio file is stated in the source element(s). Any included text will be used by browsers that do not support this element.

See provided file AudioVideoExample.html

# Video

Video Formats

Different browsers support different formats. Formats include: MP3, MP4, Ogg and wav. An extension of mov indicates an MP4 format.

Video formats are particularly challenging. There are two main issues: file format and the coding mechanism within the file. You can use tools like VLC and FFmpeg to try to convert formats.

Because of compatibility issues and the fact that video files are very large, programmers often use a video service like YouTube.

See provided file AudioVideoExample.html

# In Class Website

Remember your client’s design layout.

Compressing Images

1. Create a folder called imgs inside your in class website folder.
2. Our next step will be to add images to the website. There will be 25 – 50 words below each image talking about the specific image.
3. Earlier in the course, you downloaded images for your in class website. Move the images to the new imgs folder. There should be 8 – 12 images (no more, no less).
4. Find small logos for Facebook and Twitter and save them to the new images folder. These two hyperlinks will appear on every page.
5. Use a tool to compress and resize each image.
6. Run Windows Explorer and find the images. The compressed images should be much smaller. Delete the original large image files.

Adding Images to the Website

Add a new page to your in class website. Call it ImagesPage.html.

1. The first goal is to add a Facebook link and a Twitter link. To do this:
   1. Code the page so that the downloaded Facebook logo is a hyperlink that opens in a new tab. Tips:
      1. Write a line of code to create an external link to the facebook home page.
      2. Write a line of code to create an image. Set the height and width to 20.
      3. Combine the two lines in to one.
      4. If you get a border that you don’t like, set the border attribute of the img element to "0".
   2. Do the same for the Twitter link.
   3. Test fully.
2. When you are sure the Images page is done correctly, add the facebook and twitter link to the other pages. Test fully.
3. The next goal is to add images to your Images page. You will create a figure element for each image. Reference: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figure>. To do this:
   1. Move the cursor below the hyperlinks.
   2. Open a figure tag.
   3. Place image element here. Set the height of the image to 250 and width to 300 (or similar values).
   4. Code a figcaption element. The caption text should be less than 8 words.
   5. Close the figure tag.
   6. Below the figure element, add 25 – 50 words about the image.
   7. Test fully.
   8. Repeat for the remaining seven images.
4. Next, try changing one of the filenames to be invalid and run it. This will test the alt attribute. Fix the filename to the correct spelling.

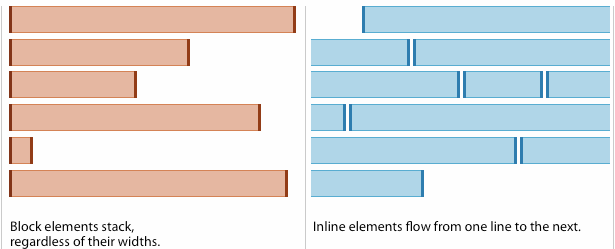
# Inline vs Block Level Elements

HTML elements are categorized as block level elements or as inline elements. With block elements (h1, p, ul, table), a new line appears before the open tag and after the closing tag. Inline elements (td, a, img, audio, video) are displayed without starting a new line. They do not interrupt the flow of the text. See BlockInlineExample.html.

Visual Example 1



Visual Example 2

**

**CSS Styles**

Learning without thought is labour lost.

(Confucius)

# Why Bother With Css?

Css will take boring looking web pages and make them visually appealing. Css 3 is very powerful and has a lot of sleek features.

Our goal is to separate meaning and presentation.HTML elements describe what the content is, not how it looks. Css controls your content’s appearance.

# What is Css?

Css stands for cascading style sheet. Use cascading style sheets to fine tune the display of your web pages: fonts, colours, backgrounds, margins, spacing etc. Css provides a consistent look and feel. Also, when there is a design change (new company logo), you implement the change in one location and all site pages are immediately effected.

A style sheet is a set of rules describing how the elements will appear on the page. The cascade defines the order of precedence for the styles. For example, if a css file states a blue font for paragraphs then that colour overrides the default colour of the browser. More on this concept later.

A style rule is a formatting instruction that works in conjunction with an element. For example, the style rule can be that a font colour is green. The rule can be applied to a paragraph tag, a heading tag or any tag that uses font colour.

# The Old Way

Earlier versions of html used a variety of attributes to make the page visually appealing. These attributes (bold, italics etc) were set in the html code. This is old school thinking. Today’s approach is leave all the work to css.

Use the playbook and in class work as your resource for help. If you do use google, be very careful. The internet has been around a long time and dated pages are not removed. Many online references teach older methods that will not work across different browsers and devices. All submitted coursework must use the methodologies taught in this course. This mirrors the workforce where your organization defines web programming standards for all programmers to follow.

# How Does It Work?

Css is a different programming language. It works directly with html but is its own language. The syntax is quite different. With css you will define the look of your web pages so that every web page follows a common theme. For example:

* All elements on the page have the same margin settings.
* All h1 headings are the same font, size and colour.
* All buttons on the site are the same shape, size and colour.
* The background images have a specific border or no border.
* All lists have the same bullet and spacing.
* Elements displayed on a mobile device will be vertical, on a desktop horizontal and on a tablet a combination of the two.

There are advantages to this method:

* Your site has a professional look and feel.
* The code is written once and applied to all chosen elements. Reusability is key.
* Because all css code is placed together, debugging is easier. You know where to look when you have a problem.
* Your site works well on multiple platforms.

# Adding CSS to Your Web Page

The html file contains the content and the structure of the page. The css code provides the look and feel of the page: colours, fonts, margins, spacing and much more. The css code is placed in a separate file with the extension css. A link to the css page is placed inside the head element, after the title element.

<link rel ="stylesheet" type = "text/css" href = "CSSExample1.css" />

The above statement is a mouthful for sure but its meaning is simple: use file CSSExample1.css to style this page.

The link element is used to define a relationship with an external resource. The rel attribute specifies the relationship between the current document and the linked document (stylesheet, aka css, is the most common usage). The type attribute tells the browser that the included file is written in the css programming language. The purpose of the href attribute is to indicate the name of the css layout file. With this code, the css file needs to be in the same folder as the html file.

See CSSExample1.html.

# Modifying Our Template

It is common practice to store your css file in its own folder. The folder is created inside your project folder then referenced in the html code.

1. Load your template. Add the css link to the head element of your template:

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

1. Save your template.

# Sports

Folder Setup

1. Create a folder called SportsExample inside folder COMP205\CssStyles.
2. Create folder css **inside the new SportsExample folder**.

Sports.html (structure and content)

1. Load your template.
2. From the File menu, pick Save As. Find the SportsExample folder and call the file sports.html.
3. Change the filename in link element to sports.css.
4. Fill in text for the Title element.
5. Add an h1 tag after the open body tag.
6. Add a paragraph element and copy and paste some text from the internet. Copy the url, you will need to reference it.
7. Below the close paragraph tag, code a hyperlink to the page you referenced.
8. Remember to include *http://* on all assessments.

Sports.css (appearance)

The purpose of this file is to define the look of your web page.

1. Start a blank document. Don’t use the html template.
2. Create a style (look) for the body element. Type in the code below. The syntax is case sensitive and finicky.

body

{

background-color: black;

color: orangered;

}

1. Save it in COMP205\CssStyles\SportsExample\css with the name sports.css. The two filenames do not have to have the same name but can be.
2. Run it. When html sees the body tag in the code, it looks to the css file to see if there are any style rules for the body tag. In other words, the browser sees a body tag in the html and runs to the css to decide how to display it. The above css code says all elements (all data) that appear inside the body tag should be in a orangered font and should have a background-color of black.
3. Style the text inside the h1 element:

h1

{

color:lightblue;

}

1. Style the paragraph element:

p

{

color:yellow;

font-size:large;

font-weight:bold;

}

1. Set up a style for the anchor tag so it displays in lime.

# Colours

Browsers will do their best to render a requested colour. Colours can be specified using name, rgb value or hex value.

References:

<https://www.w3schools.com/colors/colors_names.asp>

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

<https://www.webfx.com/web-design/color-picker/>

<https://color.adobe.com/explore/>. Type a colour in the textbox to see palettes in that colour theme.

Colour Names

Code example: color:green;

RGB

Code example: color:rgb(51, 153, 204);

Notice there is no space after rgb. Each dot (pixel) on the screen is made up of 3 tiny beams of light: red, green, blue. They work like stage lights. If you turn on only red, you get red light. If you turn on red and green, you get yellow light. If you turn on all 3, you get white. Lights are adjusted by level of brightness from 0 (off) to 255 (all the way on). Red is rgb(255,0,0), yellow is rgb (255, 255, 0). Display provided file colourchooser.html in a browser. You do not need to understand the code. The goal is to learn how colours are interpreted.

Hex

Code example: color:#4C8ED5;

A hex colour requires 6 digits and begins with a #. Within the 6 digits are 3 pairs of hexadecimal numbers representing red, green, blue. In the above example, the red value is 4C, the green is 8E and the blue is D5. D5 in hex converts to 213 in base 10. Hex codes are more precise which is important if accuracy is valued, for example an, art history website. Hex codes are more portable and predictable than colour names.

# Fonts

Fonts in css are handled by setting a font family. For example:

font-family: "Comic Sans MS", "Arial", "serif";

Double quotes are required if the font name contains spaces.

Fonts can vary with device, browser and operating system. For this reason, it is a good idea to write css code that lists multiple fonts. Whether or not a font will render properly on a device is dependent on the fonts installed on the device. For example, fonts with MS as part of the name are Microsoft fonts. They may not be installed on a Mac and Unix machine.

Choose a variety of fonts in the hope that some will be recognized by the browser. Once you have chosen your font, add one of the generic fonts as the last in the list. **Generic fonts are: serif, sans-serif, cursive, monospace.** Note that these can look quite different on different devices: laptop, ipad, phone.

When choosing fonts, remember to consider your target audience. Choose fonts that are appropriate for your clients. For example, Comic Sans is great if your audience is children, but not if your client is an investment company.

# Font Sizes Are Tricky

Setting a font size is not as simple as saying 12 point for text and 18 point for headings. What happens on a phone? What happens if the user zooms in? Measurements for font sizes can be categorized in to absolute measurements (px, pt, in, cm) and relative measurements. Relative measurements work well across devices, absolute measurements do not. The best approach is to use one of the relative measurement units shown below:

1. This list of named sizes is shown below:

xx-small x-small small medium large x-large xx-large

medium is the default size of the paragraph tag. If named sizes work for your application, use them. It is the simplest.

1. Named size *smaller* is not the same as named size *small*. *Smaller* means make the font smaller than the parent element.
2. Named size *larger* is not the same as named size *large*. *Larger* means make the font larger than the parent element.
3. Em is popular with web designers. It is relative to the normal size. 1em is the width of the letter M in the current font. 0.5em is half and 2em is double.
4. Percentage. The font size is set to a percentage of the default size of ordinary text. Use 50% to half the size and 200% to double it.

Best Strategies For Font Size

1. Define an overall size for the page. Use named sizes, ems or percentages to define the default size in the body tag. All tags will use this size. You can override the size by setting a font size within individual elements.
2. Do not change sizes without a good reason.
3. Modify specific elements. If you want to tweak a specific element a little bit, change its font size using one of the three relative measurements.

# More on Powerful *ems*

Ems are very useful. Ems can be specified with decimal points not just integers, allowing for more flexibility. Later in the course, you will see how useful this feature is for spacing elements. Also, ems give you the power to create scalable style sheets. A scalable style sheet is the best approach to achieve a properly rendered the web page. This applies to the following scenarios:

* various devices.
* when a page is resized or zoomed.

# More on Absolute and Relative Fonts

Absolute measurements are great for print but bad for screens. If you use absolute measurement on a screen, the user will not be able to change the font on some browsers. A web developer should set up font sizes in relative terms. This allows the user to determine a base font in their devices’ browser and your code’s relative fonts match it. There is a school of thought that says always use em for font sizes. At a minimum, it is considered good style to use the same measurement tool across the web site. For example, use only named sizes or use only ems.

# Embedded Fonts

You can add custom fonts to your site by placing the font file on your server and using the url keyword. For example:

@font-face {  
    font-family: myFavFont;

src: url(sansation\_light.woff);  
}

@font-face {  
    font-family: myOtherFont;

src: url(gabriola.ttf);  
}

h1 { font-family:MyFavFont, myOtherFont, serif;

# Additional Styles

font-style: italic

font-weight: bold

text-decoration: line-through (also, underline, overline)

text-align: center (also left and right)

# Borders

To define a border, use the border attribute. For example:

border-color:green;

border-style:dotted;

border-width:0.3em;

Handy keywords: solid, double, dotted, dashed, thin, medium, thick. Ems are recommended over pixels. Remember 1em is the width of the letter M in the current font.

You can shortcut three lines in to one. Note the order: width style colour.

border:0.3em dotted green;

To ensure proper rendering on all browsers and devices, define all three attributes.

# Table Styles

To get the look, you can style the whole table element, the tr element, the th element or the td element. Depending on the border statement used, you may or may not see lines inside the table. If you are not getting the lines you want, try:

table, tr, th, td

{

border-style:solid;

border-collapse: collapse;

}

To set the width of the table *within its parent*, use:

width: 80%;

# List Styles

The list-style-type attribute allows you to customize the look of your ordered and unordered lists. Sample values for this attribute are: disc (default), circle, square, upper-alpha, lower-alpha, lower-roman.

# Styling Hyperlinks

Hyperlinks have states based on user actions such as hover or visited. They are managed in css. There are only required if a change to the default styling is desired. No changes are made to the html.

|  |  |  |
| --- | --- | --- |
| State | Css | Description |
| Default | Code the **a** style. | With no css, the link will display as blue underlined text. This style is often used to style all links in all states the same way. For example, you want the same font for all links, regardless of their state. |
| Unvisited | Code the **a:link** style. This style applies to unvisited links. | With no css, the link will display as blue underlined text. |
| Visited | Code the **a:visited** style. | Without css, this is shown as a purple underlined style. |
| Hover | Code the **a:hover** style. | This state is enabled when the mouse hovers over the button. |
| Active | Code the **a:active** style. | This state occurs at the time the link is clicked. |

Best Practices

1. The order is important.
   1. a must be defined first as the others are based on it.
   2. a:hover must come after a:link and a:visited
   3. a:active must come after a:hover
2. Make sure the link looks like a link. That is, underlined or look like a button.
3. Test visited links. This is a little tricky because visiting a page once makes it ‘visited’. If you want to clear the setting for testing purposes, delete your browser history.
4. Don’t change font family or font size in a hover state. Changing fonts will have a poor visual effect and can interfere with your page layout.

# Vet Clinic Exercise (Animals.html)

It is important you complete this exercise or you will not have the material you need to do the in class website. The html file has been provided on Blackboard.

Main Body Style

Code the body style with colours and fonts.

1. Select a background colour and a font colour: <http://html-color-codes.info/color-names/>. Put some thought in to complementary colours.
2. Choose an appropriate font size for the body.
3. Select a minimum of three fonts: <http://www.w3schools.com/cssref/css_websafe_fonts.asp>

Heading Styles

1. Choose an appropriate font size for the h1 and h2 styles.
2. Code a hex value for h1: <http://www.colorpicker.com/>.
3. Code an RGB value for h2: <http://www.colorpicker.com/>.
4. Try out two of the styles listed under the Additional Styles on the previous page.

Combining Styles

Your next goal is to pick one colour that would work for all text in the paragraph, h3 and footer elements. Code one style to set up these elements. Hint: element names can be separated with commas.

List Styles

Style the unordered list to display squares for bullets. Style the ordered list to display the numbering as I, II, II etc.

Table Styles

Code the table to use a unique font. The table headings and table rows should have their own unique background colours. Set the table width to 60%.

It should have a dashed border in a unique colour. Set a border width.

# Trip Exercise

Tips and Set Up

Test after each change. Css is case sensitive. References for this exercise:

<http://w3schools.sinsixx.com/css/css_colornames.asp.htm>

<http://www.w3schools.com/cssref/pr_list-style-type.asp>

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

Create a folder called Trip with folders css and imgs. Choose two countries outside North America that you would like to visit or have visited.

Main Page

1. Create Trip.html based on your template file.
2. Add an h1 element.
3. Using a web tool, create 3 buttons: Home, FirstCountry and SecondCountry.
4. Usiong the three buttons, code three hyperlinks going across the top of the page after the h1 element.
5. Add an h2 element followed by a paragraph talking about the trip.
6. Add an h3 element that reads Important Dates.
7. Code the table shown below. Fill in data for all the cells.

|  |  |  |  |
| --- | --- | --- | --- |
| **Trip Details** | | | |
| Leaves From |  |  |  |
| Date |  |  |  |
| Cost |  |  |  |

1. The open table tag should read *<table>*. That is, it should not have border code.
2. Add an h3 element that reads Before Leaving.
3. Display the ordered list shown below:
   * + 1. Get passport.
   1. Get photos.
   2. Drive to Ottawa.
      * 1. Exchange money.
        2. Shop.
        3. Pack.
4. Create the following css styles. Each style should be unique. Remember the references at the top of this document.

|  |  |
| --- | --- |
| Element | Style |
| body | Choose font name and size, font colour, background colour. |
| h1, h2 | Choose a unique font size for each heading. |
| h3 | Italic. |
| p | Unique font and colour. |
| ol | Numbering displays as A. B. C. etc |
| table | Unique background colour. Font colour.  border-collapse:collapse; border:0.4em solid red; (any colour) |
| th and td | border:solid red; (same as table border colour) |
| a | margin-right:1em; More on this in our next topic. |
| a:hover | opacity:0.5; |

1. **Test fully, making sure all styles are rendering properly in the browser.**

CountryName1.html

1. Add an h1 element.
2. Include the 3 links from the main page.
3. Set up an h2 element that reads Cities.
4. Include an unordered list of cities to visit. Include 5 cities.
5. Add a nested unordered list to one of the cities. Inside this list, type three tourist attractions to visit in that city.
6. Set the bullet for the unordered lists to be a square.

list-style-type:square;

1. Add an image from your country. The image should appear below the lists.

CountryName2.html

1. Add an h1 element.
2. Include the 3 links from the main page.
3. Set up an h2 element the reads All About <YourCountry>.
4. Code a paragraph element and include information about the country.

# In Class Website

Design a colour theme for your in class web site. References are provided below:

<https://www.w3schools.com/colors/colors_names.asp>

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

<https://www.webfx.com/web-design/color-picker/>

<https://color.adobe.com/explore/>. Type a colour in the textbox to see palettes in that colour theme.

Create a css folder inside your web site folder. Create a css file in the new folder. Add your name in a comment at the top: /\* your name here \*/. In your html code, link to the css file. You will have to place this link in all html files on your site.

Create the following css styles. Each style should be unique.

1. Choose font name, font size, font colour and background colour for the body.
2. Ordered list numbering should display as A, B, C etc.
3. Define a style to set the unordered list bullet to a square.
4. Set width of table to 80% with a margin of auto. Set border collapse to collapse.
5. Write css code to define unique colours for the 3 hyperlink states. Set an opacity of 0.5 for hover.

# Alternative Approaches

Internal Style Sheet

With this approach, the style rules are declared in the <head> element of the page. These rules are referenced later in the body element. See InternalMethod.html.

External Style Sheet

The external css file approach covered earlier is much preferred. Use it for all web pages in this course.

# Order of Precedence

Here’s where it can get a little confusing. The body style says the text is green and the p style says the text is red. How does the browser decide? The text will be green when no other style overrides it. See precedence example.

This example illustrates the cascading portion of cascading style sheets. The green body style cascades down to all styles that do not have their own setting.

To think about it in a different way: cascading means that there is a hierarchy. For example, my html page has a body element. My css code sets the background colour for the body element to be yellow. From here, there are two scenarios:

1. My html has a paragraph element with no css code for background colour. Because the paragraph element is within the body, its background colour is yellow.
2. My html has a paragraph element with css code for background colour. The code sets the background colour of the paragraph element to black. The paragraph’s background colour shows as black.

If classes have conflicting rules, the latest class in the list overrides the earlier values.

# Getting the Lingo Down

A style sheet provides presentation rules.

A style sheet is a list of selectors (places in the page that you want to modify). For example:

h1 { **h1 is the selector.**

color:red; **There are 3 rules. Each rule contains**

font-size:xx-large; **an attribute and a value.**

font-weight:bold; **Color is an attribute. Red is a value.**

}

# The Chrome Developer Tool

Chrome provides a developer tool for debugging. It can be very handy because css does not provide error messages. To display the tool, press F12 (click 3 vertical dots in top right, More Tools, Developer Tools). The html and css code are displayed in a window.

Examining Page Layout

1. Run the precedence example in Chrome.
2. Press F12 or click 3 vertical dots in top right, More Tools, Developer Tools.
3. In the html code, click on the open body tag. The body highlights on the web page. If the highlight disappears, you can get it back by hovering over the open body tag.
4. The css code shows the settings for the body. Notice body occurs twice. The first one is our body styles, the second is default settings. Sizes display in pixels.
5. In the html code, click on the open paragraph tag. You may have to drag down to resize the small window. The paragraph highlights on the web page. If the highlight disappears, you can get it back by hovering over the open paragraph tag.
6. The css code shows the settings for the paragraph. The first one is our paragraph styles, the second is default settings.
7. While still examining the paragraph, take a look at the body styles that are displayed below the paragraph styles. Colour green is crossed out because the red paragraph colour has a higher precedence than the green body colour.

Examining Various Devices

1. With the precedence example and the developer window loaded, click the Toggle Device Toolbar icon. It is in the F12 window, to the left of the word Elements.
2. Click the icon again.
3. The word Dimensions appears in the centre of the browser window. Click the drop down to the right of it. This allows you to see the page on different devices.
4. Click responsive.
5. Click the drop down again and try out different devices.
6. Find the Rotate icon (beside the word Custom) and click it.
7. Try out various percents.

In Class Website

1. Open the main page of your in-class website.
2. Play around with the developer tool. Click on the various elements and examine the styles and the style precedence.

This tool can help you when your website doesn’t look the way you think it should. Also, it can save debugging time in future web development courses.

**Grouping Elements**

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

(Martin Fowler)

# The class Attribute

So far, we wrote styles that apply to all occurrences of an element. For example:

p

{

color: blue;

}

Changes the font colour of all paragraphs to blue. However, there may be times you want to change the display of one of the paragraphs or set presentation rules for just one section of a web page.

Steps:

1. Add the class attribute to your html.

<p class = "breadjoke">

A programmer's …

</p>

1. Create a class style.

.breadjoke

{

background-color:maroon;

}

Multiple elements can share the same class.

# Combining Classes

One element can use more than one class.

# The id Attribute

The id attribute works in a similar way to a class, though not used as often. The id attribute is also used with javascript.

Steps:

1. Add an id attribute to each html element you want to modify.

<p id = "quote">some text</p>

1. Make a style in css. Use a pound sign followed by the element’s id. The # indicates that you are not referring to an html element. Code the style.

#quote {

font-size: medium;

font-style: italic;

text-align:center;

}

Note that *quote* is case sensitive.

# Id Example

IdExample.html

<p>Here’s a regular paragraph about hiking</p>

<p id = "swimming">This paragraph text is inside the

swimming id.</p>

This text is in the body.

IdExample.css

body {

background-color:purple;

color:limegreen;

}

p {

background-color:white;

color:red;

}

#swimming {

background-color:lightyellow;

color:black;

font-size: medium;

font-style: italic;

text-align:center;

}

The id attribute can be applied to any element. Add:

<footer id = "contact">This footer text is inside the contact id

</footer>

#contact {

color:white;

}

The id attribute should always be unique. Standards dictate that you not repeat the same ID on one page. Use id when you have a style you want to apply to only one element on the page.

# 

# The Div (division) Element

This block level element is used a great deal in web pages. It is often used for grouping content. Examples:

1. A series of paragraphs appear one after the other on a web page. They can be blocked in a div element and styled as a whole.
2. On a retail website, the client wants the store hours and location to appear in a box with margins, whitespace and blue Arial 12 point font. To do this, define a div element around the hours and location text then write css code to set up the look of the content.

See provided example SportsDiv.

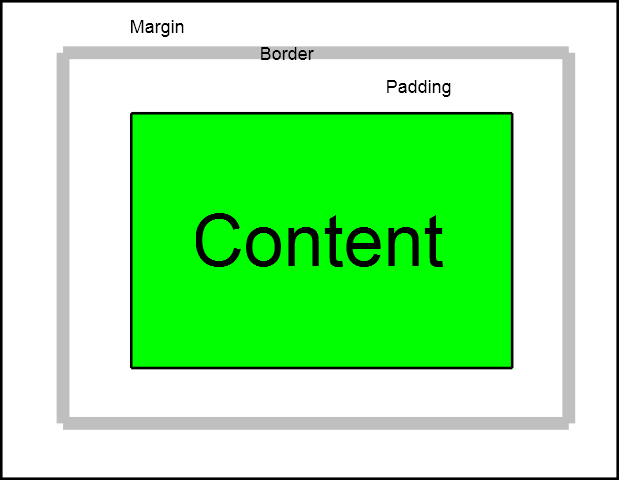
# The Box Model

The box model is a description of how the browser thinks when it lays out your pages. Understanding it is the key to getting the layout you want. A block level element describes its own space on the screen. Examples include: h1, p, ul, table, div. By default, a block level element takes up 100% of the width of the parent and has a new line before and after it. The implication is that any element appearing after a block level element is displayed on the following line.

Inline elements do not interrupt the flow of the page. They are displayed without starting a new line. They are only allowed within the context of a block level element. Examples: span, td, a, img. Inline elements appear immediately to the right of the previous element. If there isn’t enough room, the element appears left justified on the next line.

Margins, Borders and Padding

Elements have *layers of space* around them. See the image below. These layers can be used to add space around your elements.



Reference: Harris html and css

See Margins Padding Example.

# The Width Attribute

The width attribute is very handy. It is used to modify the width of a block level element.

# The Span Element

This inline element is used to apply special formatting to an element that is inside a block element. Span is much less common than div but is useful. It should only be used when you need to style an inline element within a block element. For example, highlight a specific word in a paragraph. See example DivSpan.

# Div Exercise

Review provided screen shot image. Next, load DivExercise html and css in Visual Code. Run it in Chrome and examine how the buttons are coded and styled. Your goal is to complete the html and css code to produce the web page shown in the screen shot image. Declare 3 div elements, naming them headings, navigation and maintext. Write the corresponding css code. Feel free to choose your own colours. Hint: the height and padding attributes can be set using ems. For the main text: try 3.75em for the height and 0.3em for the padding.

# In Class Website

1. Today you will begin the final version of the in class website**. Before making any changes, it is important that you make a copy of the website and work with the copy**. Copy the files in your version 1 folder to your version 2 folder.
2. Be sure that you are now working with the version 2 folder.
3. At this point, the h1 and h2 elements are no longer needed. You may keep them or remove them but they should be on all pages or no pages.
4. Working with the Home page:
   1. Using a div element, create a class that encompasses the six hyperlinks. Name the class **navflexcontainer**. This name will make sense in our next topic. Code the class with a bottom margin of 2em.
   2. Using a div element, create a class that encompasses the content below the navigation and above the date and contact information. Name the class **maincontainer**. Style it with margins, padding and background colour. If desired, set a min-width of 20em (or em value of your choice).
   3. Using a div element, create a class to encompass the contact information and date information (coming later). Name the class **footercontainer**. Style it with font colour, background colour, margin, text align and padding.
5. Once the Home page is working fully, modify the Table page to include navflexcontainer, maincontainer and footercontainer.
6. Modify the Image page to include navflexcontainer and footercontainer. Do not add the maincontainer div to the Image page.
7. Choose one word (or a few words) on your Home page to highlight using a span element. It can be company name, topic name or any other. It should appear in italics with a noticeably defined font colour or background colour. The word needs to appear 4 or more times on the page. If it does not, add it.
8. Run the developer tool. Click on the various elements and examine the styles and the style precedence. Try resizing the webpage/code window to test the current responsiveness of your website. Try out the icon *Toggle device toolbar*.
9. Work through *Checklist for in class website (at end of Grouping Elements topic)*. It can be found in the in class website section of this playbook. Make sure your site contains all the items on the checklist. If something is missing, add it to the site.

**Layouts**

Your time is limited.

Don’t waste it living someone else’s life.

(Steve Jobs)

# Friendly Reminder

If looking for help, look through the playbook (including the References section) and provided examples.

# Responsive Web Design

A method of designing websites to ensure a good visual experience for all devices and browsers. The website is ‘responding’ to the user’s behaviour.

# Centering

Centering.html

<div class = "mainBody">

<p>This paragraph is inside a div class. The div element has

a double border and a width of 75%. The text inside the

box is centred because of the text-align attribute.

The box with the border is centred on the page because of

the margin-left and margin-right attributes.</p>

<p>Here is a second paragraph. Filler text. Filler…</p> </div>

Centering.css

.mainBody {

border: 1em double black;

width: 75%;

margin-left:auto;

margin-right:auto;

text-align:center;

}

Setting the margins to auto is used to centre block level elements. Consider these points from the Harris html and css:

* You can adjust the width of a block. For example, set the main div of the page to 75% of the page body width.
* Use *text-align: center* to centre text inside a div.
* Centre an element by setting margin-left and margin-right to auto. This centres the element within its parent. It does not centre text.
* Background colours help you see what’s happening. Add them even if you don’t need them. You can remove them later. This is an excellent tool for debugging confusing layout behaviours.
* When you assign a percentage, you are asking for the percent inside the parent element.

# Parent Child Elements

See Parent Child example.

# WildCats Exercise

Note that the green section doesn’t quite pass the responsive test. You will need to allocate for that.

WildCats.html

<div class = "top">

Wild Cats

</div>

<div class = "main">

<p>The African Savanna ecosystem is a tropical grass… </p>

<img class = "leopard" src = "leopard.jpg" height ="100"

width="120" alt = "leopard" />

<img class = "panther" src = "panther.jpg" height ="100"

width="120" alt = "panther" />

</div>

WildCat.css

.top {

background-color:lightpink;

border: 0.2em dashed navy;

font-size:3em; /\* xx-large isn't big enough \*/  
 /\* hint: add 6 lines of code \*/

# }

# .main {

# background-color:lightgreen;

# border:0.2em solid black;

# /\* hint: add 5 lines of code \*/

# }

# .leopard {

# /\* hint: add 2 lines of code \*/

# }

# .panther {

# /\* hint: add 1 line of code \*/

# }

# Types of Layouts

There are many methods for building responsive websites. The methodologies have evolved over the years and are constantly improving. In work placement, we will see a wide variety of methods based on when they were designed and the knowledge of the coders.

The Table Layout is an older methodology based on defining your web page as a table and coding tables within tables. It is difficult to code and even more difficult to debug. The biggest limitation of this approach is that it does not fair well when the page is resized. It does not work across different screen sizes.

The Float Layout is based on dividing and organizing your web page into columns. This method is still in production in many websites today.

More powerful than the float layout is the Flexible Box Layout (flexbox). It provides a way to layout, align and distribute space on a responsive web page. This is the method we will use in this course.

Other approaches are Media Queries (Murach chapter 8) and Grid layout (Murach chapter 10). They are beyond the scope of this course.

# Flex Layouts

Flex layouts are used to produce responsive websites. Flex is all about getting the layout you want at varying screen sizes. The Flex Box Layout Model allows you to lay out your pages and choose the alignment and spacing of your elements. The model allows you to define what happens when the web page renders on various devices. We are specifying how the page is laid out and what happens when the window is resized. When using flex, you can still use box model properties: margin, padding, border, width and height. The width defaults to 100% of the parent element. The height is determined by the content of the item (unless the parent element has been set to a fixed height). Flexbox makes it easier to change a layout from horizontal to vertical as the screen gets smaller.

Flex layouts work in an interesting way. The goal is to leave the html layout as simple as possible. From there, the programmer writes css code to provide hints on how the page should be displayed and how the elements interact with each other. Notice css code provides hints, not rules. Understanding the concepts is the key to learning this material.

For example, a page has 10 elements that are displayed across the page from left to right. Flex allows you to decide the layout on different devices. This includes:

* Are all elements the same width or do the widths vary?
* What happens with the space between the elements?
* Does all the white space appear on the right hand side? On the left hand side? Or divided equally between the elements?
* What happens to the 10 elements and their size as the page is resized?

We will use flex layouts for all pages of our in class website. Flex will help us with:

* The six links on all pages: four links on the left hand side and two links on the right hand side. All links the same height, even if the content of each item varies.
* The Images page. When maximized, there are three images across. As the user resizes, it changes to two images across then one. Flex also specifies the space between the images and the number of images displayed at the bottom of the page.

The css code makes suggestions and the browser does the work. The advantage of this approach is that the browser decides based on the screen size and shape. For example, it will display differently on a phone than on a desktop.

Flex can be used with any element. It is often used with a div element.

Page content is placed in a *flex container*. *Flex items* are created inside the flex container.

The main idea behind the flex layout is to give the container the ability to alter its items’ width, height and order to best fill the available space. A flex container expands items to fill available free space or shrinks them to prevent overflow.

Flex layouts can be simple or highly complex. We are covering the basics.

Flex Example 1 (html and css)

*Example A*

The html code creates a flex container with 7 flex items. The css code defines:

Flex container

* + display:flex
    - * This tells the browser that the div is a flex container. The flex items in the container will be placed beside each other.
  + flex-direction:row
    - * Flow is by row, not column. Items are drawn left to right.
  + Flex-wrap:wrap
    - * As the window is resized, wrap the items to the next line. Similar to word wrap in Microsoft Word.

Flex item contains background colour and margin right.

**Notice that each flex item is the width of its contents.** By default, the width of a flex item is width of the content in the item. This means a flex item with a five letter word is wider than a flex item with a two letter word.

*Example B*

In the example, the html and css code are the same as Example A, except for one additional code line. The css for the flex items includes the code *flex:1*. This means that each flex item is the same width and the space is filled. Maximize the window to see the effect.

Now, slowly resize the window smaller. There comes a point where having all items the same size is not feasible. At this point, the browser then distributes the flex items as it deems appropriate for the device.

Flex Example 2 (html and css)

The html code creates a flex container with 7 flex items. Notice the additional class on the yellow marten flex item.

The css code for the flex container contains:

* + display:flex
  + flex-direction:row
  + flex-wrap:wrap
  + padding, margins and background colour.

The css for flex item contains:

* + background colour, border and margin bottom.
  + flex:1. Each flex item is the same width.

Notice the bigbox class has a flex of 2.75. The browser does its best to give the item 2.75 times the width of the other items. When there is extra space, the bigbox item gets the extra space. When there isn't enough space, the browser tries to give the bigbox item additional space. The browser is deciding how best to render your design.

Flex Example 3 (html and css)

In this example, the code *flex:1 and flex:2.75* have been removed. As a result, by default the width of each flex item is the width of the data in the flex item.

The code justify-content:center; appears in the flex container. justify-content specifies how the flex items are positioned within the container. justify-content:**flex-start** draws the items at the start of the flex container with white space on the right. **flex-end** does the opposite.

justify-content:center centres the block of items within the container with white space on the left and right sides. justify-content also determines how the page will look when the window is resized and the elements are split across two rows. Remember that justify content is referring to where all flex items, as a block, are drawn. It is not referring to the justification of each individual flex item.

If flex items are not perfectly spaced, you likely have a right or left margin on the flex item. If this is the case then the interior items have space on both sides but the first and last ones only have space on one side. Remember spacing is left right only, not top to bottom.

# Tips

* Borders and background colours can be very helpful for both learning and debugging.
* If you are surprised you are not seeing a change after editing the css, it may be because there is no extra space horizontally.
* Play with the three provided examples.
* Play with the chrome developer tool (F12). This will help you understand how the browser is thinking.

# Assessing Responsiveness

Be sure to check that all webpages are responsive. Slowly resize the window smaller and larger. Watch for poor behaviour: text (or any element) overlapping an element; text (or any element) bleeding out of borders/colours.

# Examples

See Gradients example and Borders and Backgrounds example.

# In Class Website

If you like, you can change the style for ol and ul in your **V2** copy. Do not change the V1 copy.

Navigation

In our last topic, you created a div class titled navflexcontainer. The goal now is to turn the navigation menu in to a flex layout. Navflexcontainer will have multiple flex items.

1. Start with the Home page.
2. Create a flex item, named **navflexitem**, for each of the six hyperlinks.
3. Create a flex layout for the navigation. Add padding, background colour and font settings to the navflexitem style. Do not add flex:1 to this style. Try adding a text shadow (optional):

text-shadow:0.2em 0.2em 0 rgba(0,0,0,0.2);

1. We want to create blank space between the Form link and the social media links. After the Form flex item, create a new flexitem div. The contents of the new div should be only a hard space. Type *&nbsp;* between the open tag and the close tag.
2. To create wide space between the Form link and the social media links, define the new extra space flex item to be three times wider than the other items.
3. To create a smaller space for the social media flex items, define each one to be 0.1 in relation to the other items.
4. Test fully to ensure your site is responsive. Resize the browser window slowly to examine how the page looks at various stages.

Once you are sure the above is working fully, modify the navigation on the other pages to mirror the Home page. Test fully.

The Images Page

Code the figure element style with the following:

margin-left:auto;

margin-right:auto;

Your goal is to create a flex layout for the Images content. Review the Images page in Visual Code and in Chrome. Notice the main content can be broken down in to multiple sets of the following:

figure, img, figcaption, /figure, the text below the image.

We will call each set a *content box*. Each content box displays as a rectangle with a background-color of your choosing. There is space around each content box. Each content box has the same background colour. Each content box displays and wraps as one unit.

When the window is maximized, there should be three or more images displayed across from left to right. As the window is resized smaller, the content boxes should wrap.

Flex container **imagespagemainflexcontainer** begins below the navigation and ends right before the date and contact information. Each flex item (**imagespagemainflexitem**) contains one content box.

Create some space between and above each content box. Add some space inside each content box.

If the text inside the content box is not displaying properly, add a max-width attribute to the image flex item style. Use ems. The actual number will depend on your other settings. Try 15em and modify as needed.

Remember to add justify content to the flex container. You can choose the value for the justify content attribute.

Add the line below to imagespagemainflexitem:

border-radius:0.75em;

Run it. Try it with 1em and 0.5em. If you like the look, you can leave in border radius. If you’re not a fan, you can delete it.

Developer Tool

The developer tool can really help you. Css and the browser have a mind of their own, they draw the page based on your instructions, the resolution and what it thinks is the best approach. The result is sometimes your page does not look as you expect. There are no error messages, which makes it tricky.

Run the developer tool (F12). Click on the various div elements and examine the styles and the style precedence. Try resizing the webpage/code window to test the current responsiveness of your website. Try out the icon *Toggle device toolbar* (Ctrl-Shift-M). Next, click on navflexitem socialmedia in the html. Examine the styles.

**Forms**

Whether you want to uncover the secrets of the universe, or you want to pursue a career in the 21st century, basic computer programming is an essential skill to learn.

(Stephen Hawking)

# Form Elements

Forms add interactivity to the page. Users can type in data and the data can be sent to a server for processing. There are many form elements available. Once you have mastered a few of them, learning new ones is much easier.

# The Form Element

<form action = "SomePage.html">form elements here</form>

This element is a container for all input elements in the form. You can think of it like a box around your data: a box with or without lines. It is required if you want to save user data. This element will be important later:

* The form action statement specifies the filename of the next web page to be displayed. For example, when the user clicks Submit, html looks at the filename in the form action statement and loads that page.
* Use CSS to style the form element as desired. For example, assign a background colour, set a width or centre the form.

# Labels

<label for = "username">Enter user name</label>

A label is a small piece of text beside an input element (data entry control). You can style the labels in css. All input elements must have an associated label element with a for attribute that matches the id attribute in the input element.

# Input Boxes

An input box is limited to one line of text. Each input box needs an associated label with a for attribute.

<input type = "text" id = "address" value = "Easy Street" />

<input type="date" id = "apptDate">

<input type="time" id = "apptTime">

<input type="color" id = "colourChoice">

<input type="month" id = "apptMonth">

<input type = "number" id = "age" />

<input type = "password" id = "pwd" />

Attributes

The value attribute is optional. It sets a default value.

<input type = "text" id = "city" value = "Kingston"/>

The required attribute specifies than an input field must be filled out before submitting the form.

<input type = "text" id = "city" required/>

The size attribute determines the visible width of the input box. Officially, it is the number of characters that are displayed in the input box. Realistically it depends on the font.

The maxlength attribute specifies the maximum number of characters allowed in the input box.

The id attribute can be used by JavaScript. It creates an identifier (variable name) for the field. In the example below, Firstname is a variable that will store the first name typed in by the user.

<input type = "text" id = "Firstname" />

Firstname is called a form variable. A form with 10 input elements will have 10 unique id values.

# FieldSets

The fieldset element is used to define various parts of a form. Html surrounds the elements inside the fieldset in a lined box. Note that the line colour may be the same colour as your background. If the line does not show up, code css to change colour(s). Standards dictate a legend be placed inside the fieldset.

<fieldset>

<legend>Log In</legend>

html elements here

</fieldset>

# Text Areas

Use this element to create an input box with multiple lines.

<textarea id="answer" rows="10" cols="25" maxlength="255"></textarea>

The 10 rows refers to how many rows are displayed at one time. Any text (including spaces) between the open and close tags will be displayed in the textarea. It can be used for a default value. If there is no default value, make sure there are no spaces between the tags. If you would like to remove the user’s ability to resize the textarea, set the resize attribute of the textarea to none.

To improve the look of the label and textarea, set the vertical-align attribute of the textarea to either top or middle.

# Listboxes

<label for = "venue">Venue:</label>

<select id = "venue" size = "5">

<option value = "KRock">KRock Centre</option>

<option value = "Rogers">Rogers Centre</option>

<option value = "CanadianTire">Canadian Tire Centre</option>

<option value = "Bell">Bell Centre</option>

<option value = "MTS">MTS Centre</option>

</select>

Listboxes provide the user with a list of options to choose from. Users like them because it saves typing. Programmers appreciate them because it enforces data integrity. The user cannot make a typo and cause an adverse reaction in calculations or reporting. Listboxes also preserve precious web page real estate because they take up very little space.

The select tag marks the beginning and the end of the list. The text between the start and end option tags is displayed to the user. There must be an option tag for every item in the list. The value attribute will be used by the programmer. The user will not see this value. If the user clicks *KRock Centre* then *KRock* (the text in the value attribute) is stored in the venue form variable. Since the user can only select one value, you need only one form variable ("venue").

You can get a different look using the size attribute. Try adding a size of “5” to the select tag.

To improve the look of the label and listbox, set the vertical-align attribute of the listbox to either top or middle.

# Checkboxes

<label for = "Pass">Back stage pass? </label>

<input type = "checkbox" id = "Pass" checked />

A check box can be on or off. If you have two checkboxes, they can both be on, both be off or one of each. Use a checkbox when you would like the user to answer a yes or no question.

# Radio Buttons

<input type = "radio" name = "price" id = "price" value = "300"/>Blue

<input type = "radio" name = "price" id = "price" value = "400"/>Red

<input type = "radio" name = "price" id = "price" value = "500"/>Yellow

Use radio buttons to provide the user with multiple options to choose from. Radio buttons are mutually exclusive. The user can select only one. Typically, a set of radio buttons is placed in a fieldset. This approach is not required but visually helps the user understand which radio buttons belong together. The browser also needs to know which radio buttons belong together. To do this, it uses the name attribute. The name attribute is the same for all radio buttons in the group.

# Buttons

Buttons perform an action when clicked. For example, save entered data then display a different web page.

<button>Save Data</button>

This element does not provide a link. Clicking the button will do nothing. To make it act like a button, you need to either place the button inside a form or write code to specify the action (JavaScript, php). The text inside the element can be replaced with an img element or any content.

# Linking Labels and Input Elements

To associate a label with an input element, code the for attribute of the label element to match the id attribute of the input element. This link is required for all input elements.

It is particularly useful for checkboxes. Run the following in Chrome.:

<label>Fun?</label>

<input type = "checkbox" id = "FunCheckBox"/>

When you click on Fun, nothing happens (poor UX). Change the first line to:

<label for = "FunCheckBox">Fun?</label>

In Chrome, click on Fun. The label is linked to the checkbox.

# Accessibility Requirements

1. Every input element needs an associated label with a for attribute. The label can be before the input element or beside it. The label’s for attribute matches its id attribute. If there is a name attribute, it should also match.
2. Group related items in a fieldset. Each fieldset needs a legend.
3. Do not use the placeholder attribute (grey text inside an input box). If you must use it, a linked label is still required.

# Design Ideas

In terms of formatting, you have a number of options:

* Place a label and its corresponding input element in a class. Style the pair.
* Place multiple input elements in a class and style them.
* Place a fieldset in a class and style it.
* Place the entire form in a class and style it.

# Styling Form Elements

Input Elements

input

{

styles here

}

To style specific input elements, include the type values:

/\* no spaces allowed in line below \*/

input[type="text"],input[type="password"]

{

styles here

}

Hover and Focus

You can change the display of an element when the user hovers over it. To maintain the responsiveness of the page, do not change the height or width of the element.

button:hover

{

opacity: 0.5;

}

You can change the display of an element when the element receives focus. The code below turns the background yellow when the user tabs in or clicks into the text area.

textarea:focus

{

background-color:yellow;

}

# In Class Website

Verify your in class web site has a fully implemented flex layout that is responsive on all pages. Your goal now is to create a new page with input elements. Think of an appropriate form for your website topic. It can be a shopping basket, a contact us page or a form that matches your topic. Create a new webpage following your client’s design plan. Add the following elements inside the maincontainer div:

* Form element with styled border.
* Text input box.
* Text area.
* List box.
* Checkboxes (at least 2).
* Radio buttons (at least 2).
* Date element.
* Time element.
* Try out some of the input element attributes.
* Add a style of your choosing for the labels.
* Set a unique font colour for the data entered in the input boxes.
* Make sure the forms page works properly when resized.
* Optional: add additional styles.

**To earn the marks, your code must be tailored to your topic. It cannot be the same code or content as the Playbook.**

**Javascript**

Java is to Javascript what car is to carpet.

(Chris Heilmann)

# JavaScript Overview

JavaScript is its own language and quite extensive in functionality. We are just scratching the surface.

JavaScript is a client side scripting language. Client side code runs in the browser and does not require interaction with a web server. Scripting code is executed line by line while the browser reads it. That is, it is not compiled before executed. JavaScript:

* makes the page interactive and is built into most web browsers.
* is often used to validate data entry or create a navigation menu. For example, a submenu pops up when the mouse hovers over it.
* has nothing to do with Java.
* allows comments with // or /\* \*/
* ignores white space.
* is case sensitive. This will require a high level of attention to detail. getElementById is not the same as getElementByID or GetElementById.

# Placement of the Code

There are multiple options for where you can place the JavaScript code:

1. In the body of the page between the script tags.
2. In a separate text file with the js extension. To link the javascript file to the html page, place the code below right before the close body tag.

<script type = text/javascript src = somefile.js> </script>

1. You may also see code that places javascript code inside html code. An example of this is using the onclick event of a button. This approach should not be used as it is deprecated.

# Welcome Example

See provided code.

# Event Listeners

Javascript allows the programmer to change the page at run time. It uses event listeners. Events are actions. Examples include click, change, mouse over, focus, resize, load. The programmer can use the event information to execute code. Examples:

* The user clicks a button and a video starts playing.
* The user picks Canada from a listbox and the list of provinces is displayed.
* The current date is displayed when the page loads.
* The user clicks into a textbox and its background colour changes.

# Highlander1 Example

Goal

Display Highlander when the user clicks the *Which Car* button.

Steps Overview

1. Html
   1. Create a placeholder element with a unique id.
   2. Create a button element with a unique id.
2. Javascript
   1. Code an event listener that will act when the button is clicked.
   2. Code an event handler to specify what should happen when the button is clicked.

Html Steps

Add an element to specify where the word Highlander should appear.

<p id = "carPlaceholder"></p>

Add a button, assigning a unique id attribute. The carButton id will be used by javascript.

<button id = "carButton">Which car?</button>

Javascript Steps

*Event Listener*

Write Javascript code to tell the browser to display Highlander when the button is clicked. An event listener is created. It listens (waits) for the button to be clicked and then it acts. Example:

document.getElementById("carButton").addEventListener("click",

myFunction);

The first part of the code:

document.getElementById("carButton")

seems tricky but it really is just the *Which Car* button. document refers to the web page. On the web page, find the one element that has the id carButton.

The full statement:

document.getElementById("carButton").addEventListener("click",

myFunction);

tells the browser when the user clicks the *Which Car* button, it should jump to myFunction and execute the code there.

*Event Handler*

Code a function to define what happens when the button is clicked.

function myFunction()

{

var carname = "Highlander";

// find the element whose id is carPlaceholder. Make the word

// Highlander the element's content. That is,

// put Highlander in the paragraph. The innerHTML

// property refers to the content between the open tag

// <p id = "carPlaceholder"> and the close tag </p>

// innerHTML can be text, html code or both.

document.getElementById("carPlaceholder").innerHTML =

carname;

}

# Highlander2 Example

See provided code.

# Highlander3 Example

See provided code.

# Javascript Exercise

Create a new web page. Add an h1 element. Add the following:

<label>Done School?</label>

<footer>Have a great day</footer>

Write code to add a checkbox to the page. When the checkbox is clicked:

* the word Congrats should appear on the page.
* the colour of all text on the page should change.
* the background colour of the page should change.

# Highlander4 Example

See provided code.

# Highlander5 Example

See provided code.

# In Class Website

Current Date

Add the current date to the footers. The date should appear in the format:

mmmmm dd, yyyy

For example, *December 10, 2023*. See provided file getMonthName.js.

To have the date displayed automatically when the page loads add an id to the open body tag then code the javascript. If you would like a hint, start with the highlander5 date example. The event listener is a little different than our previous ones (click, change), in that brackets have to be added after the function name:

document.getElementById("loadingBody").addEventListener("load", showDate());

Jump To Top Button

Add a Jump To Top button at the bottom of the Images page. The javascript code to execute it is shown below:

window.scrollTo({top: 0, behavior: 'smooth'});

Change Table

The goal now is to give the user the ability to change the look of the table. To do this in javascript, you must have a border on your table. Verify that you have a table border defined in your css code. Add a button to the Table page. The button should read Change Table. Add an id to the open table tag. Write javascript code to style the table as follows:

* font colour green (keyword color).
* border width 0.3em (keyword borderWidth).
* border style: dotted (keyword borderStyle).
* border colour: yellow (keyword borderColor).

Note that you are only changing the outside border. Once you have it working, you can change the above values, making sure the change is noticeable. Try out various border styles: dashed, groove, ridge, hidden, solid, double.

See debugging tips on next page…

Debugging Tips

1. Both the event listener for the jump button and the event listener for the change table button need an if statement:

if (reference to html element)

create event listener

Without the if statement, the code fails. This is because the jump button is on the images page, but not other pages. When you load the tables page, the javascript will fail because it can’t find the jump button. If javascript fails, the remaining code is ignored. The same will happen with loading the images page, given the change table button is not on the images page.

1. If your js code is not working, check to see if you have a colour defined in a th or td style. If you do, comment it out and try the js code again. Setting a colour in the table style should not conflict with the js code.

**In Class Website**

Learn how to be happy with what you have

while you pursue all that you want.

(Jim Rohn)

**Page Layout for In Class Website Pages**

The main heading will appear on every page. Think of it as a banner at the top of every page on the site.

A second heading appears here. It will be slightly smaller than the main heading.

Your site will have the six hyperlinks shown below. You may add extra pages or links if you like. The headings can be above or below the hyperlinks, your choice.

Home Table Images Form <the space between the links will be created in a later topic >  

This part of the page varies for each page.

Write your footer here. Include text you want on the footer of every page. Include the current date. Include a link to your college email address. It should display in the following format:

Your first and last name **|** Your college email address | Current Date

# Instructions

Instructions for building the website can be found in each topic in this playbook. To perform a search, use the softcopy of this document (on Blackboard).

# Checklist for In Class Website (at end of Lists topic)

# index.html

1. Load your html file in IE. You should have:
   1. The tab of the page should have a relevant title that is unique to this page.
   2. An h1 and an h2 heading.
   3. Steps described in the Headings and Lists section of this document under heading *In Class Website.*
   4. A footer that will be common to every page.
2. Load your html file in Visual Code. You should have the following elements:

* doctype
* your name in a comment
* html lang
* head
* meta
* title
* body
* h1
* h2
* p (for paragraph)
* footer

# Checklist for In Class Website (at end of Hyperlinks topic)

Elements for Every html Page

* doctype
* your name in a comment
* html lang
* head
* meta
* title
* body
* h1
* h2
* four hyperlinks
* footer with email link

Additional Elements

* Home page
  + - Steps described in the Headings and Lists section of this document under heading *In Class Website.*
* Table page
  + - A 5x4 (or larger) table with data relating to your website. Includes table headings. Includes at least one rowspan and colspan.

# Checklist for In Class Website (at end of Grouping Elements topic)

Elements for Every html Page

* doctype
* your name in a comment
* html lang
* head
* meta
* title
* link rel
* body
* six hyperlinks
* footer with email link

Additional Elements

* Home page
  + - Steps described in the Headings and Lists section of this document under heading *In Class Website.*
* Table page
  + - A 5x4 (or larger) table with data relating to your website. Includes table headings. Includes at least one rowspan and colspan. Table is not full width of the page and has unique border width and colour.
* Colour scheme as described in CSS Styles section of this document.
* Steps described in the Grouping Elements section of this document under heading *In Class Website Exercise*

# Final Checklist for In Class Website

Elements for Every html Page

* doctype
* your name in a comment
* html lang
* head
* meta
* title
* link rel
* body
* six hyperlinks
* footer with current date and contact us email link

Additional Elements

* Home page
  + - Steps described in the Headings and Lists section of this document under heading *In Class Website.*
* Table page
  + - A 5x4 (or larger) table with data relating to your website. Includes table headings. Includes at least one rowspan and colspan. Table is not full width of the page and has unique border width and colour.
* Colour scheme as described in CSS Styles section of this document.
* Steps described in the Grouping Elements section of this document under heading *In Class Website*
* Steps described in the Layouts section of this document under heading *In Class Website*
* Steps described in the Forms section of this document under heading *In Class Website*
* Steps described in the Javascript section of this document under heading *In Class Website*
* Web site is responsive. All pages work properly when resized and zoomed.

References

I’ve missed more than 9,000 shots in my career.

I’ve lost almost 300 games.

26 times, I’ve been trusted to take the game winning shot

and missed.

I’ve failed over and over again in my life.

And that is why I succeed.

(Michael Jordan)

# Websites

[www.w3schools.com](http://www.w3schools.com)

<http://www.w3schools.com/cssref/pr_list-style-type.asp>

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

<http://www.colorpicker.com/>

<http://html-color-codes.info/color-names/>

<https://color.adobe.com/explore/newest/>

# COMP 205 Html Elements

**Basic Html**

|  |  |
| --- | --- |
| Tag | Description |
| [<!doctype>](http://www.w3schools.com/tags/tag_doctype.asp) | Defines the document type |
| [<html>](http://www.w3schools.com/tags/tag_html.asp) | Defines an HTML document |
| [<title>](http://www.w3schools.com/tags/tag_title.asp) | Defines a title for the document |
| [<body>](http://www.w3schools.com/tags/tag_body.asp) | Defines the document's body |
| [<h1> to <h6>](http://www.w3schools.com/tags/tag_hn.asp) | Defines HTML headings |
| [<p>](http://www.w3schools.com/tags/tag_p.asp) | Defines a paragraph |
| [<!--...-->](http://www.w3schools.com/tags/tag_comment.asp) | Defines a comment |

**Formatting**

|  |  |
| --- | --- |
| Tag | Description |
| [<code>](http://www.w3schools.com/tags/tag_code.asp) | Defines a piece of computer code |
| [<sub>](http://www.w3schools.com/tags/tag_sub.asp) | Defines subscripted text |
| [<sup>](http://www.w3schools.com/tags/tag_sup.asp) | Defines superscripted text |

**Forms and Input**

|  |  |
| --- | --- |
| Tag | Description |
| [<form>](http://www.w3schools.com/tags/tag_form.asp) | Defines an HTML form for user input |
| [<input>](http://www.w3schools.com/tags/tag_input.asp) | Defines an input control |
| [<textarea>](http://www.w3schools.com/tags/tag_textarea.asp) | Defines a multiline input control (text area) |
| [<button>](http://www.w3schools.com/tags/tag_button.asp) | Defines a clickable button |
| [<select>](http://www.w3schools.com/tags/tag_select.asp) | Defines a drop-down list |
| [<option>](http://www.w3schools.com/tags/tag_option.asp) | Defines an option in a drop-down list |
| [<label>](http://www.w3schools.com/tags/tag_label.asp) | Defines a label for an <input> element |
| [<fieldset>](http://www.w3schools.com/tags/tag_fieldset.asp) | Groups related elements in a form |
| [<legend>](http://www.w3schools.com/tags/tag_legend.asp) | Defines a caption for a <fieldset> element |

**Images / Audio Video**

|  |  |
| --- | --- |
| Tag | Description |
| [<img>](http://www.w3schools.com/tags/tag_img.asp) | Defines an image |
| [<audio>](http://www.w3schools.com/tags/tag_audio.asp) | Defines sound content |
| [<source>](http://www.w3schools.com/tags/tag_source.asp) | Defines multiple media resources for media elements (<video> and <audio>) |
| [<video>](http://www.w3schools.com/tags/tag_video.asp) | Defines a video or movie |

**Links**

|  |  |
| --- | --- |
| Tag | Description |
| [<a>](http://www.w3schools.com/tags/tag_a.asp) | Defines a hyperlink |

**Lists**

|  |  |
| --- | --- |
| Tag | Description |
| [<ul>](http://www.w3schools.com/tags/tag_ul.asp) | Defines an unordered list |
| [<ol>](http://www.w3schools.com/tags/tag_ol.asp) | Defines an ordered list |
| [<li>](http://www.w3schools.com/tags/tag_li.asp) | Defines a list item |
| [<dl>](http://www.w3schools.com/tags/tag_dl.asp) | Defines a description list |
| [<dt>](http://www.w3schools.com/tags/tag_dt.asp) | Defines a term/name in a description list |
| [<dd>](http://www.w3schools.com/tags/tag_dd.asp) | Defines a description of a term/name in a description list |

**Tables**

|  |  |
| --- | --- |
| Tag | Description |
| [<table>](http://www.w3schools.com/tags/tag_table.asp) | Defines a table |
| [<caption>](http://www.w3schools.com/tags/tag_caption.asp) | Defines a table caption |
| [<th>](http://www.w3schools.com/tags/tag_th.asp) | Defines a header cell in a table |
| [<tr>](http://www.w3schools.com/tags/tag_tr.asp) | Defines a row in a table |
| [<td>](http://www.w3schools.com/tags/tag_td.asp) | Defines a cell in a table |

**Styles and Semantics**

|  |  |
| --- | --- |
| Tag | Description |
| [<style>](http://www.w3schools.com/tags/tag_style.asp) | Defines style information for a document |
| [<div>](http://www.w3schools.com/tags/tag_div.asp) | Defines a section in a document |
| [<span>](http://www.w3schools.com/tags/tag_span.asp) | Defines a section in a document |
| [<header>](http://www.w3schools.com/tags/tag_header.asp) | Defines a header for a document or section |
| [<footer>](http://www.w3schools.com/tags/tag_footer.asp) | Defines a footer for a document or section |
| [<main>](http://www.w3schools.com/tags/tag_main.asp) | Specifies the main content of a document |

**Meta Info**

|  |  |
| --- | --- |
| Tag | Description |
| [<head>](http://www.w3schools.com/tags/tag_head.asp) | Defines information about the document |
| [<meta>](http://www.w3schools.com/tags/tag_meta.asp) | Defines metadata about an HTML document |

# COMP 205 CSS Selectors

**Color Properties**

|  |  |
| --- | --- |
| Property | Description |
| [color](http://www.w3schools.com/cssref/pr_text_color.asp) | Sets the color of text |
| [opacity](http://www.w3schools.com/cssref/css3_pr_opacity.asp) | Sets the opacity level for an element |

**Background and Border Properties**

|  |  |
| --- | --- |
| Property | Description |
| [background-color](http://www.w3schools.com/cssref/pr_background-color.asp) | Specifies the background color of an element |
| [background-image](http://www.w3schools.com/cssref/pr_background-image.asp) | Specifies one or more background images for an element |
| [border](http://www.w3schools.com/cssref/pr_border.asp) | Sets all the border properties in one declaration |
| [border-bottom](http://www.w3schools.com/cssref/pr_border-bottom.asp) | Sets all the bottom border properties in one declaration |
| [border-bottom-color](http://www.w3schools.com/cssref/pr_border-bottom_color.asp) | Sets the color of the bottom border |
| [border-bottom-style](http://www.w3schools.com/cssref/pr_border-bottom_style.asp) | Sets the style of the bottom border |
| [border-bottom-width](http://www.w3schools.com/cssref/pr_border-bottom_width.asp) | Sets the width of the bottom border |
| [border-color](http://www.w3schools.com/cssref/pr_border-color.asp) | Sets the color of the four borders |
| [border-left](http://www.w3schools.com/cssref/pr_border-left.asp) | Sets all the left border properties in one declaration |
| [border-left-color](http://www.w3schools.com/cssref/pr_border-left_color.asp) | Sets the color of the left border |
| [border-left-style](http://www.w3schools.com/cssref/pr_border-left_style.asp) | Sets the style of the left border |
| [border-left-width](http://www.w3schools.com/cssref/pr_border-left_width.asp) | Sets the width of the left border |
| [border-right](http://www.w3schools.com/cssref/pr_border-right.asp) | Sets all the right border properties in one declaration |
| [border-right-color](http://www.w3schools.com/cssref/pr_border-right_color.asp) | Sets the color of the right border |
| [border-right-style](http://www.w3schools.com/cssref/pr_border-right_style.asp) | Sets the style of the right border |
| [border-right-width](http://www.w3schools.com/cssref/pr_border-right_width.asp) | Sets the width of the right border |
| [border-style](http://www.w3schools.com/cssref/pr_border-style.asp) | Sets the style of the four borders |
| [border-top](http://www.w3schools.com/cssref/pr_border-top.asp) | Sets all the top border properties in one declaration |
| [border-top-color](http://www.w3schools.com/cssref/pr_border-top_color.asp) | Sets the color of the top border |
| [border-top-style](http://www.w3schools.com/cssref/pr_border-top_style.asp) | Sets the style of the top border |
| [border-top-width](http://www.w3schools.com/cssref/pr_border-top_width.asp) | Sets the width of the top border |
| [border-width](http://www.w3schools.com/cssref/pr_border-width.asp) | Sets the width of the four borders |

**Basic Box Properties**

|  |  |
| --- | --- |
| Property | Description |
| [clear](http://www.w3schools.com/cssref/pr_class_clear.asp) | Specifies which sides of an element where other floating elements are not allowed |
| [float](http://www.w3schools.com/cssref/pr_class_float.asp) | Specifies whether or not a box should float |
| [height](http://www.w3schools.com/cssref/pr_dim_height.asp) | Sets the height of an element |
| [left](http://www.w3schools.com/cssref/pr_pos_left.asp) | Specifies the left position of a positioned element |
| [margin](http://www.w3schools.com/cssref/pr_margin.asp) | Sets all the margin properties in one declaration |
| [margin-bottom](http://www.w3schools.com/cssref/pr_margin-bottom.asp) | Sets the bottom margin of an element |
| [margin-left](http://www.w3schools.com/cssref/pr_margin-left.asp) | Sets the left margin of an element |
| [margin-right](http://www.w3schools.com/cssref/pr_margin-right.asp) | Sets the right margin of an element |
| [margin-top](http://www.w3schools.com/cssref/pr_margin-top.asp) | Sets the top margin of an element |
| [max-height](http://www.w3schools.com/cssref/pr_dim_max-height.asp) | Sets the maximum height of an element |
| [max-width](http://www.w3schools.com/cssref/pr_dim_max-width.asp) | Sets the maximum width of an element |
| [min-height](http://www.w3schools.com/cssref/pr_dim_min-height.asp) | Sets the minimum height of an element |
| [min-width](http://www.w3schools.com/cssref/pr_dim_min-width.asp) | Sets the minimum width of an element |
| [padding](http://www.w3schools.com/cssref/pr_padding.asp) | Sets all the padding properties in one declaration |
| [padding-bottom](http://www.w3schools.com/cssref/pr_padding-bottom.asp) | Sets the bottom padding of an element |
| [padding-left](http://www.w3schools.com/cssref/pr_padding-left.asp) | Sets the left padding of an element |
| [padding-right](http://www.w3schools.com/cssref/pr_padding-right.asp) | Sets the right padding of an element |
| [padding-top](http://www.w3schools.com/cssref/pr_padding-top.asp) | Sets the top padding of an element |
| [right](http://www.w3schools.com/cssref/pr_pos_right.asp) | Specifies the right position of a positioned element |
| [top](http://www.w3schools.com/cssref/pr_pos_top.asp) | Specifies the top position of a positioned element |
| [width](http://www.w3schools.com/cssref/pr_dim_width.asp) | Sets the width of an element |
| [vertical-align](http://www.w3schools.com/cssref/pr_pos_vertical-align.asp) | Sets the vertical alignment of an element |

**Text Properties**

|  |  |
| --- | --- |
| Property | Description |
| [text-align](http://www.w3schools.com/cssref/pr_text_text-align.asp) | Specifies the horizontal alignment of text |
| [text-indent](http://www.w3schools.com/cssref/pr_text_text-indent.asp) | Specifies the indentation of the first line in a text-block |
| [text-justify](http://www.w3schools.com/cssref/css3_pr_text-justify.asp) | Specifies the justification method used when text-align is "justify" |
| [word-wrap](http://www.w3schools.com/cssref/css3_pr_word-wrap.asp) | Allows long, unbreakable words to be broken and wrap to the next line |

**Text Decoration Properties**

|  |  |
| --- | --- |
| Property | Description |
| [text-decoration](http://www.w3schools.com/cssref/pr_text_text-decoration.asp) | Specifies the decoration added to text |
| [text-decoration-color](http://www.w3schools.com/cssref/css3_pr_text-decoration-color.asp) | Specifies the color of the text-decoration |
| [text-decoration-line](http://www.w3schools.com/cssref/css3_pr_text-decoration-line.asp) | Specifies the type of line in a text-decoration |
| [text-decoration-style](http://www.w3schools.com/cssref/css3_pr_text-decoration-style.asp) | Specifies the style of the line in a text decoration |
| [text-shadow](http://www.w3schools.com/cssref/css3_pr_text-shadow.asp) | Adds shadow to text |

**Font Properties**

|  |  |
| --- | --- |
| Property | Description |
| [font](http://www.w3schools.com/cssref/pr_font_font.asp) | Sets all the font properties in one declaration |
| [font-family](http://www.w3schools.com/cssref/pr_font_font-family.asp) | Specifies the font family for text |
| [font-size](http://www.w3schools.com/cssref/pr_font_font-size.asp) | Specifies the font size of text |
| [font-style](http://www.w3schools.com/cssref/pr_font_font-style.asp) | Specifies the font style for text |
| [font-weight](http://www.w3schools.com/cssref/pr_font_weight.asp) | Specifies the weight of a font |

**Table Properties**

|  |  |
| --- | --- |
| Property | Description |
| [border-collapse](http://www.w3schools.com/cssref/pr_border-collapse.asp) | Specifies whether or not table borders should be collapsed |

**Lists and Counters Properties**

|  |  |
| --- | --- |
| Property | Description |
| [list-style](http://www.w3schools.com/cssref/pr_list-style.asp) | Sets all the properties for a list in one declaration |
| [list-style-image](http://www.w3schools.com/cssref/pr_list-style-image.asp) | Specifies an image as the list-item marker |
| [list-style-type](http://www.w3schools.com/cssref/pr_list-style-type.asp) | Specifies the type of list-item marker |

Reference: w3schools.com

Grades

The future belongs to those who believe

in the beauty of their dreams.

(Eleanor Roosevelt)

# My Grades in COMP 205

Quiz 1 (12%):

Quiz 2 (12%):

In class website version 1 (26%):

Technical interview (20%):

In class website version 2 (30%):

Total (out of 100):

**Grades Breakdown:**

Percent Letter Grade

90 - 100 A+

85 - 89 A

80 - 84 A-

77 - 79 B+

73 - 76 B

70 - 72 B-

67 - 69 C+

63 - 66 C

60 - 62 C-

57 - 59 D+

53 - 56 D

50 - 52 D-

0 - 49 F

You have the power to define your future.

© St Lawrence College

Kingston, Ontario

2023 - 2024

Time is not measured by the passing of years but by

what one does,

what one feels

and what one achieves.

(Jawaharlal Nehru)