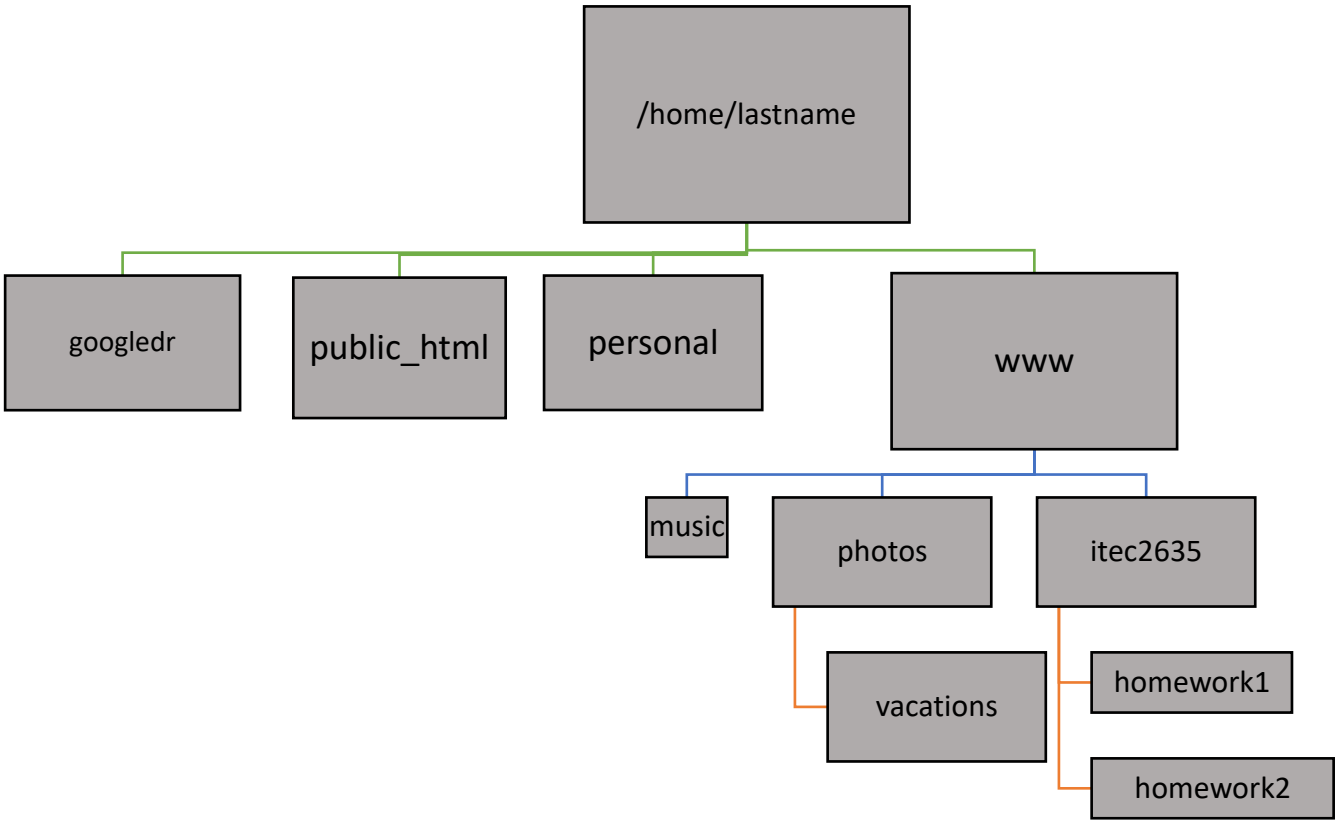
http://oldtown.glendon.yorku.ca/~yourlastname

Rstudio-server: https://oldtown.glendon.yorku.ca/ (click on "advanced", etc.…)
Log in information:
Username: yourlastname (always in **small** letters)
Password: your_student_number
If you have a double last name, use only the first one.
Your web pages are at: **http://oldtown.glendon.yorku.ca/~yourlastname**

Please do the exercises below to prepare for our next class

1.

a) Create the hierarchical organization for your webpage at Glendon which includes the following structure:



b) Protect your web page at the URL:

http://oldtown.glendon.yorku.ca/~yourlastname/itec2635/homework1

with the username “glendon01” and password “gl01”

2. Absolutely Essential:

   a. Google: “html introduction w3schools” and read, for example,
      - https://www.w3schools.com/html/html_intro.asp
      - https://www.w3schools.com/html/html_basic.asp

Do the exercise, click on “try it yourself”, etc…

   b. In summary: Html, head, body, title, h1-6, and comments tags and !DOCTYPE declaration; meta tag

3. Can you make your own web page that looks just like this one:
http://oldtown.glendon.yorku.ca/~titou/itec2915/lect1/dog.html [3]
(replace "titou" by "yourlastname”)

Hints:
- Web browser to visit [3] above; view “page source”; copy the html code.
- Rstudio-server login; Editor to create a new html file, paste the html code; save as dog.html inside /home/yourlastname/www/2915/lect1/

4. Can you make your own web page that looks just like this one:
http://oldtown.glendon.yorku.ca/~titou/imagine/ex1.html [4]
(replace "titou" by "yourlastname”)

Hints:
- Web browser to visit [4] above; view “page source”; copy the html code.
- Rstudio-server login; Editor to create a new html file, paste the html code; save as ex1.html inside /home/yourlastname/www/imagine/ex1.html

5. Can you make an account at https://trinket.io/?

Readings:

CSS vs HTML

HTML is a markup language that describes the *content* of a web page. When a web browser reads HTML, it turns the code into text (or other objects like images or videos) and renders them on the screen. For example, when a modern browser reads the code:

```
<h1>My Blog</h1>
<h4>July 25th: The Big Job Interview</h4>
<p>Today was the day! I put on my best suit and left the house at 7:30am…</p>
```

It can show it as plain text to the viewer.

My Blog

July 25th: The Big Job Interview

Today was the day! I put on my best suit and left the house at 7:30am…

However, if we wanted to go a step further, and display more styled content like this…

We would need to use “Cascading Style Sheets,” or CSS. Where HTML describes the *content* of a web page, CSS describes the *style* of that content, changing things like font, color, background, location, and more.

CSS: An Overview

CSS Notes

Basic Ideas

1. In much earlier days of the internet, you could style html elements by specifying attributes inside the tags. For example:

```
<p><font size="2" color="#1c87c9">Blue text</font></p>
<p><font size="3" color="red">Red text, font size increased.</font></p>
<p><font face="arial" color="#8ebf42">Green text, typeface changed.</font></p>
```

   However, now we use a different approach: Cascading Style Sheets (CSS). For example, now, HTML tags support a *style* attribute which is a string composed of property-value pairs, which are similar to the HTML attributes. For example:

```
<p style="font-size:16px; color:#1c87c9;">Blue text.</p>
<p style="font-size:125%; color:red;">Red text, font size increased.</p>
<p style="font-size:18px; color:#8ebf42; font-family:arial;">Green text, typeface changed.
```

   A number of tags and attributes have been deprecated:

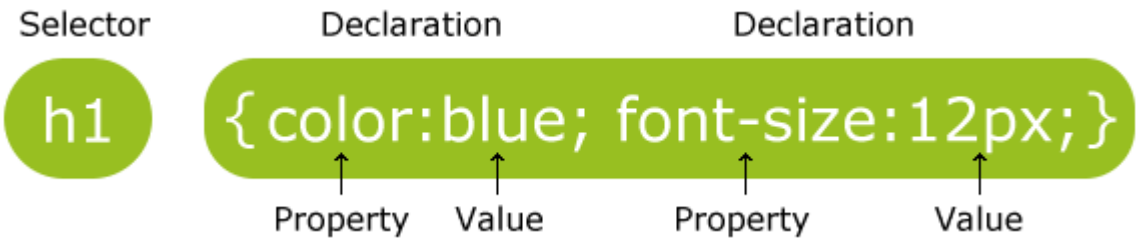
   https://www.w3docs.com/learn-html/deprecated-html-attributes.html
   https://www.w3docs.com/learn-html/deprecated-html-tags.html

2. CSS is the only way HTML should be styled. There are many more CSS properties compared to styling with HTML attributes. Probably most sites/apps you use have very sophisticated CSS styling.

3. There are three ways to use CSS. The example above is called *inline* (least used, and only sparingly). The other two was are *internal* and *external*. I would guess that most sites use external, with some internal. We will talk later about when to use each technique. Three Ways to Insert CSS

4. CSS Syntax (Source: http://www.w3schools.com/CSS/default.asp) for using the *internal* or *external* approach:

   A CSS rule set consists of a selector and a declaration block:



   - The selector points to the HTML element you want to style.
   - The declaration block contains one or more declarations separated by semicolons.
   - Each declaration includes a property name and a value, separated by a semicolon.

CSS is descriptive. To assign attributes to an element, we first name the *identifier* of the element type we want to assign a rule to. Then within brackets {} we give a set of property names and values.

Example:

```
p {
      color: red
      font-family: “Times New Roman”;
}
```

This code makes all <p> elements look just like this.

Generally speaking, CSS is written as:

```
/* Comments go between slashes and asterisks */
(Element Selector) {
      (First property Name) : (Some Value);
      (Second property Name) : (Some Value);
}
```

You can include as many declarations, or “property: value;” pairs as you want for that element type. Each declaration is made up by first listing the property name (the type of rule that it’s going to be), then a colon, then the value (the specific version of that rule that we’re going to apply), and then a semicolon.

So what sorts of things can be assigned styling rules through CSS? Every HTML element that we have seen so far can be used as a CSS selector, and can be styled using CSS. For example, we could have:

```
h3 {
      color: blue;
      background-color: yellow;
      font-size: 16px;
}

h6 {
      color: purple;
      background-color: white;
      font-size: 14px;
}
```

This would go through your document and change the text color, background color, font size, etc. of every h3 and h6 elements that it finds.

As we learn about CSS, it is important to remember that CSS statements usually *cascade*. When the browser reads your style sheets, it will attempt to apply their rules to every element that matches the identifiers that you have given, with more specific instructions superseding general ones.

For example, when we apply the following CSS:

```
    body {color: green};
       p {color: blue };
       i {color : red };
```

To the following HTML:

```
<body>
```

```
      <h3>This is an h3 element </3>
      <p>This is a p element, <i>but this segment is a nested i element</i></p>
      <ul>  <li>These are</li>
            <li>ul / li</li>
            <li>elements</li>     </ul>
</body>
```

We get:

<h3>This is an h3 element</h3>
This is a p element, but this segment is a nested i element
 These are
ul / li
elements

As we can see, the green color “cascaded” down to everything within the body tags, then the blue color was applied to everything within a p tag, and then the i tag took precedence over both. This allows us to create general rules for how our document is going to look, then break them with more specific rules for areas that need to stand out and look different.

Adding CSS to an HTML document (3 ways)

In order to use CSS, it first must be linked up to your HTML document, so that the browser can find your code and apply your declarations. There are four ways of doing this.

In Line Styling:

The first way of applying a CSS declaration is to put it directly inside an HTML element. HTML elements can have attributes, like “src” or “href.” Most HTML elements can also be given an attribute called “style,” which allows us to apply CSS rules directly to, and only to, that element. This is done like this:

```
<elementName style=”property1: value1; property2: value2; (etc)”> Content
</elementName>
```

Note, unlike other ways of applying CSS, these rules will only apply to the HTML element that contains that style declaration.

Use this method when there is a single HTML element that needs to behave differently from the rest of your document, or if you are not using full stylesheets for any reason.

Internal (Embedded) Style Sheets:

The second way of applying CSS is by using a <style> element in the head of your document. This is known as an internal style sheet, because the CSS code is located inside the HTML document. These rules will apply to the whole page. This looks something like this:

```
<!doctype html>
<html><head>
```

```
            <style>
                  body { font-family: lobster;
                        background-color: green; }
                       p { background-color: white; font-style: italics; }
            </style>
      </head>
<body> ….  </body></html>
```

This method is useful if you have a web site consisting of only one or two pages, where you won't need to apply the same CSS to multiple pages.

External (Linked) Style Sheets:

The last way of applying CSS is by putting a link to a CSS document into your HTML file. When the browser loads your page, it goes to that .CSS file, loads the stylesheet data, and applies them to your HTML.

This is done by first saving a document of CSS declarations as a .css file (in this case, we chose to name it mystyles.css), and then adding the following HTML to your document's head:

```
<head>
      <link rel="stylesheet" type="text/css" href="mystyles.css" media="screen" />
</head>
```

This method is the most complex, since it requires multiple files of code, but is much easier to apply when you have a large website with multiple web pages. This is because multiple pages can share the same CSS code, creating a similar look across pages.

ID’s and Classes

Sometimes, you will want to apply a CSS rule only to a specific chunk of text, or a small collection of items. When this is the case, we can use ids and classes.

ID's and classes are attributes that are added to HTML elements, which can then be used to target them with CSS. A class is used for a collection of items; an ID is typically only given to a single element. Classes can be given to multiple elements, and these elements do not have to be of the same type.

Classes and IDs are defined like this:

```
<p class=”MyClass” id=”FirstParagraph”> This is my text </p>
```

It is then possible to create CSS declarations that target either the class name or the ID name, instead of the element types. When being selected in CSS, IDs always have a “#” placed before them, while classes always have a “.” placed before them.
For example:

```
#FirstParagraph{
      font-family: lobster;
}

.MyClass {
      background-color: purple;
}
```

This will make the single element that has been given the “FirstParagraph” ID the font “lobster,” and will give ***every*** element that has been given the “MyClass” class a purple background.

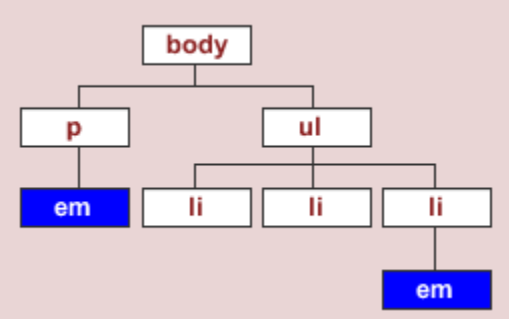
Remember that the # for IDs and the period for classes is used when you select them in CSS, ***not*** when you define them as attributes for your HTML elements.

1. **Example 1: Type selectors**

The most common and easy to understand selectors are type selectors. Type selectors will select any HTML element on a page that matches the selector, regardless of their position in the document tree. For example:

```
em {color: blue; }
```

This rule will select any element on the page and color it blue. As you can see from the document tree diagram below, all elements will be colored blue, regardless of their position in the document tree.



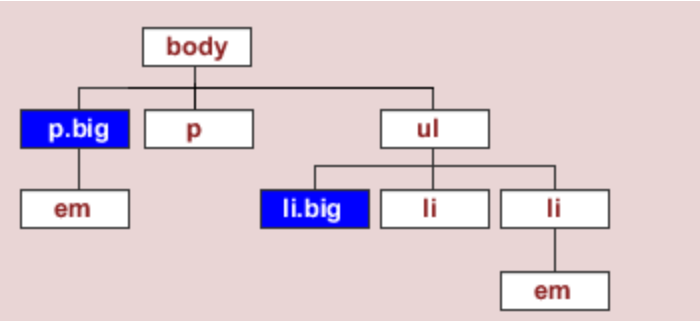
2. **Example 1: Class selectors**

While type selectors target **every** instance of an element, class selectors can be used to select any HTML element that has a class attribute, regardless of their position in the document tree.

For example, if you want to target the first paragraph and first list items on a page to make them stand out, you could mark up the page in the following way:

```
<body>
<p class="big">This is some <em>text</em></p>
<p>This is some text</p>
<ul>
<li class="big">List item</li>
<li>List item</li>
<li>List <em>item</em></li>
</ul>
</body>
```

The tree diagram would be:



The rule used could then be:

```
.big { font-size: 110%; font-weight: bold; }
```

3. Example 1: ID selectors

ID selectors are similar to class selectors. They can be used to select any HTML element that has an ID attribute, regardless of their position in the document tree. Examples of ID selectors:

```
#navigation { width: 12em; color: #333; }
div#navigation { width: 12em; color: #333; }
```

The major difference is that IDs can only be applied once per page, while classes can be used as many times on a page as needed.

ID selectors are well supported across standards-compliant browsers.

Online resources:

https://developer.mozilla.org/en-US/docs/Web/HTML

https://developer.mozilla.org/en-US/docs/Learn/CSS

https://w3schools.com - simple and clear, with interactive examples

https://openclassrooms.com/en/courses/5265446-build-your-first-web-pages-with-html-and-css

https://docs.trinket.io/getting-started-with-html#/html/a-basic-webpage

https://business.tutsplus.com/tutorials/make-page-layout-designs-microsoft-word--cms-34172

http://oldtown.glendon.yorku.ca/~titou/23/page01.html

https://sway.office.com/R0jC52cvv3pm6y5t?ref=Link