COSC 2P89 — Lab 1 — HTML

Our first lab is a two-parter. We'll be creating a very basic (non-interactive, and non-styled) website, and then hosting it on sandcastle. The actual contents aren't terribly important, though you might find it easier to fill in more placeholders if you choose a topic anyway. Note that you'll probably have an easier time next week if you make more material this week.

It is expected that you're already familiar with the lecture material, so this lab will only cover *how*; not why.

Requirements:

Each lab is recorded as being out of **4**. Going through this lab step-by-step will cover everything you need for the *basic* requirements listed below, which will earn you **3** out of **4**. To earn the full 4 out of 4, you must also work out the final *completion* task.

Basic Requirements (3 out of 4):

You must create a site, hosted on sandcastle, that covers the following requirements:

- Minimum of 3 separate pages (one of which will be your 'home', or starting page)
  - Each page must have its own *title*
- Navigation between pages
  - Appearance isn't terribly important, but make sure it's structurally logical (e.g. put the same nav-bar into each page, at the same spot)
- A `header` and a `footer`
  - It doesn't matter what you put in them. Fake copyright info would make sense for the footer, and you can use the header for your nav-bar, if you like
- A table, somewhere
  - Doesn't matter if it has a border or not, or what's on it
- At least four *block* tags used, somewhere
- At least three *inline* tags used, somewhere
- At least one *image* that can be *clicked* to another page
  - One additional (non-clickable) image. If it's on the larger side, it'll make the *final requirement* easier
- Host your page so it starts at the address:
  - `https://www.cosc.brocku.ca/~`*`(your username)`*`/1/index.html`

Final Requirement (for full credit):

In addition to the basic requirements above, you simply need to take one step towards making your page more flexible for mobile devices.

- Hint: Responsive web design
- Less-subtle hint: Week 02, Slide 37
- Even-less-subtle hint: https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

If you run out of time, you can email a link to your hosted page to *your* lab demonstrator *prior to* the ***beginning*** of next week's lab.
Do not email it to your instructor. You will get a grumpy response.

Assuming you can meet the requirements of this page, you don't have to read any further.

Introduction:

For this lab (or these labs, depending on perspective), you'll be learning some basic HTML. Some things might not seem very helpful until later labs. At first glance, the tasks will seem *very* easy. Because they are. The idea is to build up incrementally, so we're starting simple.

In HTML, several tags are largely semantic until you give them a more expressive meaning. For this lab, you'll be marking different types of text appropriately (even when it doesn't appear to have any real use), but later on, you'll be adding styling (and more) to them.

Before we begin:

As mentioned in class, it's always nice to have a reference.
If you're interested in "official" documentation, here's an example:
https://www.w3.org/TR/html5/

Ain't that just painful to read? That's why I prefer using this as a reference:
https://www.w3schools.com/tags/

However, there are myriad options online, and you can go with whatever you like.

Our first page, and *hosting*:

Before we can view a webpage, we first need to *create* one. That, of course, means we need to be able to *edit* it. The lab computer should have at least VS code, but seriously any plaintext editor will do.
If you want to follow-up with some of this from home, you might also wish to consider SciTE/Notepad++.

(There's also Sublime, Atom, etc. You can even use straight-up Notepad (or TextEdit on a Mac) if you like.) You'll also need the ability to put it onto sandcastle; again if you need to repeat the process at home there's a tool for that, but *for now* let's just focus on getting a document.

You'll also need the ability to put it onto sandcastle, but first let's just get a document.

For whatever editor you're using, type in the following:

```
<html>
<head>
</head>
<body>
Test
</body>
</html>
```

Now, save the file somewhere you can easily find it. The location and filename don't matter just yet, as we're going to use this to get everything else set up, but I'd call mine `index.html` .

First, drag the file onto Firefox (or whatever), just to confirm that the page actually saved.

This works, and would be fine for most of the lab, but we want to get hosting out of the way.

We actually have official instructions on how to set pretty much all of this stuff up here:

https://brocku.ca/mathematics-science/computer-science/sandcastle/#personal-web-page

And there's similarly documentation on logging in here:

https://brocku.ca/mathematics-science/computer-science/sandcastle/#SSH

and for file transfers here:

https://brocku.ca/mathematics-science/computer-science/sandcastle/#file-transfer

(Yes, I realize these are all the same help page)

However, just for ha-ha's it's also explained here.

Connecting

You can connect to the sandcastle server (Computer Science Linux server) using a *secure shell*. You have a few choices for that.

On Windows:

- Download and install PuTTY (https://www.putty.org/), and connect to sandcastle.cosc.brocku.ca (or 139.57.100.6), on port 22; providing your username and **Computer Science** password
- Get MobaXterm (https://mobaxterm.mobatek.net/download.html), start up a terminal, and type: `ssh username@sandcastle.cosc.brocku.ca` and enter your Computer Science password
    - Decent option, because it'll take care of the *file transfer* section below as well
- Install WSL (Windows Subsystem for Linux) for, uh… honestly kinda overkill

On Mac:

Bring up the spotlight and start Terminal.

Type: `ssh username@sandcastle.cosc.brocku.ca` and enter your Computer Science password.

On Linux:

Like the Mac instructions, but for whatever terminal you have.

If you forget your Computer Science password:

You should be able to rest it here: https://www.cosc.brocku.ca/support/rstpass.phtml

Otherwise, you'll need to contact Cale Fairchild.

File transfers

File transfers use 'secure FTP', which is actually just another way of using secure shell, so the address, port, login credentials, etc. are all the same.

On Windows:

- You can download Filezilla, Core FTP, or any of a couple other similar programs, and connect to the same address as for the shell above
- If you went with MobaXterm above, then the moment you connect via ssh, it will also provide a file transfer in the left panel (you might need to click an 'sftp' tab or similar)
    - It has little buttons at the top for transfers, but drag&drop should also work

On Mac:

Try Cyberduck

On Linux:

Uh… `scp`? You knew what you were getting into...

<u>Waaait! Isn't this goofy?</u>

Yeah, much of it is. Or at least *premature*? You're in the lab, so 'sandcastle' is just your Z:\ drive. And PuTTY is already there for when you need it. Still, good to know!

Folder and permissions

So you have that file (e.g. index.html), and just need to upload it to get something basic going.
Before you can upload it, you'll need a folder, so the first step is just connecting for a shell.

You could *probably* just do this from Z:\ drive, but access permissions would honestly be tedious, so maybe do this ‘the hard way’?

Follow the instructions above (PuTTY, or MobaXterm, ssh, etc.).

Both files and folders have *permissions*.
The basic permissions are *read*, *write*, and *execute*.
For files, they're self-explanatory, but folders have a bit of nuance.

- Having the *read* permission for a folder means you can see a *listing* of it
- Having the *execute* permission for a folder means you can *enter into it*

Additionally, for all three permissions, you can assign them for three different categories of users:

- Yourself
- Other users within your group
- Any other users not in your group

All together, this is 9 decisions to make, or 512 possible combinations.
For each category, we use:

- **4** for *read*
- **2** for *write*
- **1** for *execute*

These are just bit flags, so they can be added. e.g. 7 means you can do anything with the file/folder, 0 means you can't do anything, or 5 means you can read and execute (but not make changes).

Altogether, a file with 777 means *anyone* can do *anything*.
(Obviously, this would be a tremendously bad idea, and very open to malicious attacks)

For our purposes, your folders should normally be fine for you to do whatever you like, and only allow other users to enter into them (i.e. execute them).

- If you're wondering why you'd want to allow even *that* much: the services (e.g. web server) have to be executed by *some* user
  - Some actually run *as you* (e.g. via `su` — *substitute user*)
  - Others run as a separate daemon

First, you want to ensure the permissions are correct on your *home folder* (aliased as ~):

```
chmod 711 ~
```

You can confirm the results by seeing a listing:

```
ls -ld
```

The 'current folder' is just a period (`..` means 'one folder up'). Normally it tries to show the contents of folders, so the `-d` confirms you want the directory itself. The `-l` just puts it into a more descripting listing format.

- You'll notice you only need a single hyphen (you *can* have two separate hyphenated arguments)

Next, let's make the folder for hosting:

```
mkdir public_html
```

```
chmod 711 public_html
```

We needed to set the permissions on that as well.
Just for ha-ha's, let's see what we have so far:

```
ls -hl
```

- It doesn't matter which order you put the flags in (e.g. `hl` or `lh`)
- The `-h` puts the file sizes into a human-friendly formatting (KB/MB/etc.)

There are a *ton* of options, so feel free to check the manual:

```
man ls
```

(Hit the **q** key when you're done)

Anyhoo, back on track...

Once you make that folder, you could *try* going to: https://www.cosc.brocku.ca/~(your username)/

but there's nothing to display yet.

Create a folder just called `1` in your `public_html` folder:

```
mkdir ~/public_html/1
```

Now, put that file you created (e.g. index.html) into that 1 folder.

- In the lab, seriously just drop it in
- If you're in MobaXterm, navigate there in the panel on the side, and you can just drag the file onto it
- In most other sftp programs, the left side shows your local files (your computer), and the right shows the remote end (sandcastle), so just navigate to the right place on both sides, and click the →

It's *possible* this already works, and you can see the file by going to:

https://www.cosc.brocku.ca/~yourusername/1/index.html

but since this'll come up eventually, I'm going to assume everything is broken, and cover how to make it work.

Let's have a chat about Bash, shall we?

(Super-)Quick Shell Primer:

This is just a brief interjection into how to navigate within a Linux server using Bash (the proper name for the command-line shell we'll be using). If you're already good with this stuff, you might not need this.

Somehow, connect to sandcastle.cosc.brocku.ca .

By now, you've probably guessed the `ls` is a *listing* command.

You can confirm where you are via the `pwd` command (for *print working directory*).

If you have lots of stuff in your Z:\ drive, you might not easily see `public_html`, but typing:

```
ls public_html
```

will show you the *contents* of it, rather than showing the folder itself. On the other hand:

```
ls -lahd public_html
```

might change that.

To reiterate from above (and expand upon it):

- It doesn't matter whether you say `lahd`, `lhda`, `ahdl`, or whatever; we're just adding *flags*
- `a` says to include all files, even hidden ones
  - You hide files under linux by having them start with a `.`
- `h` simply says to use human-friendly sizes (e.g. 1K, instead of 1024)
- `d` tells it to not dereference links, or follow into the directory
- `l` says to use a *long listing* format
  - This is why we see so much information
  - Notice that this format shows permission bits, the owner, the group, etc.

There are lots of other options. e.g. `-t` sorts by time, `-S` by filesize, and `-r` reverses the listing sequence.

Again: `man ls`

Let's find that file?
In case you aren't already there, let's ensure you're in the top-level home folder:

```
cd ~
```

`cd` means change directory.

Note that the `~/` prefixes aren't in any way necessary. If you don't specify a path, everything's just "relative to where you are". e.g.:

```
chmod 711 .

cd public_html

chmod 711 .

chmod 711 1

chmod 744 1/index.html
```

Most of that won't do anything if you were following along above. The 744 on the html file says "anyone can read this", and that's pretty normal for hosting effectively a text file.

Anyhoo, try seeing if you can load the page. Again, the URL should be something like:

https://www.cosc.brocku.ca/~username/1/index.html

So, if your username were `ef99ab`, you'd go to:

https://www.cosc.brocku.ca/~ef99ab/1/index.html

If that doesn't work, go back to the help page linked above, and try to work out what went wrong.

Once you can definitely access that page, everything else becomes a snap!

Later on, we'll expand on hosting a little, including `.htaccess` files, but we don't need that right now.

Back to editing!

Let's go back to edit our `index.html` file, to add some (any) content.

For the most part, we won't be re-explaining the lecture, but as an initial reminder:

- Our `html` tag marks the beginning and end of the entire document
- The `head` tag is where we describe attributes *about* the document
- The `body` tag is where the actual content for the page goes
- All of these tags have both an *opening* and a *closing* tag, to mark where they begin and end

What can we add next? Before we get to the content, let's fix that "index.html" in the tab bar.

```
<html>
<head>
	<title>Bad Ideas</title>
</head>
<body>
	Test
</body>
</html>
```

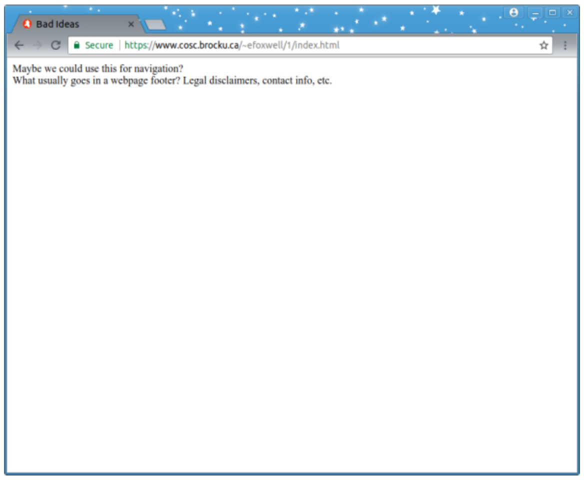
So, what's different? Well, first off, I've indented in a couple of places. That won't affect anything at all, since *additional* whitespace is just ignored. Besides that, I've added the `title` tag.

To see the change, we could drag the file back onto the browser again, but it's easier to just *refresh* the page (either via the reload button, or by pressing control+r). We now wee "Bad Ideas" instead of "main.html", which is nicer.

Before we start filling in all of our content, a bit of *organization* is probably a good idea. What will we want? I like to start files with notes to myself so I don't forget anything. We'll also have a header at the top, some content, and a footer. What will be inside them? We'll figure that out later. Let's just get this much sketched out:

```
<html>
<head>
	<title>Bad Ideas</title>
</head>
<body>
<!-- Student A Studentson, 9999999 -->
<!-- This is a comment. I can have it span multiple lines if I like.
  Or I can have it on one line. The browser won't render this, so it's
  all up to what's most convenient for me. -->
<header>
	Maybe we could use this for navigation?
</header>
<!-- I should add some sort of block for content here. -->
<footer>
	What usually goes in a webpage footer? Legal disclaimers, contact info, etc.
</footer>
</body>
</html>
```

Try reloading the page, to see the difference between a header and a footer.



Maybe we could use this for navigation?
What usually goes in a webpage footer? Legal disclaimers, contact info, etc.

Huh. Couple things worth noting:

- The header and footer look the same
  - This is because, before you start adding some significance to the tags (next week), these both start out as normal *block-level* tags; nothing more, nothing less. So they define new blocks of text, but aren't special. We're just setting ourselves up for later when we'll eventually want to define a connected decoration or *style* to the semantic hint of being whatever "header" or "footer" means
- The comments aren't rendered by the browser
  - e.g. that note about adding a block for content doesn't show up at all
- I mentioned a student name/number in the comments at the top. You should do that. But with yours

Let's start adding some content. Again, write whatever you like; it doesn't have to be what I wrote.

I'm going to start only commenting when there's something useful to point out. You can copy that into your own document, or not; again, it doesn't actually *do* anything.

```
<html>
<head>
      <title>Bad Ideas</title>
</head>
<body><!-- Student A Studentson, 9999999 -->
<header>
      Maybe we could use this for navigation?
</header>
<p><!--The most common block is the paragraph (p)-->
      There is no established correlation between good ideas and actually making
      money. Novelty is a reasonable selling point. So go ahead, and snap up all
      of those ideas everyone else is too smart to use, and make them your own!
</p><!--End first paragraph-->
<p><!--We want to start AND end each block-->
      Wonderful (terrible) ideas include: ejector seats for helicopters, solar
      powered flashlights, research-paper autogenerators, and monkey chauffeurs!
</p><!--End second paragraph-->
<footer>
      Disclaimer: no warranty is expressed or implied for the ideas contained herein.
      You are solely responsible for trying to find your own toes after implementing
      these ideas.
</footer>
</body>
</html>
```

Feel free to take a look at how it renders, but it should be pretty predictable by now.

Pretty plain, isn't it?

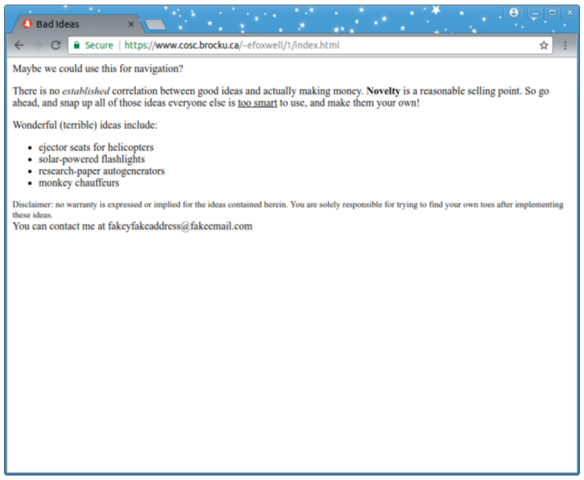
For the most part, that won't be changing until next week. However, there are still a few semantic designations (and structural organization) we can apply to improve it.

What's missing? Well, any notion of emphasis is the most obvious. Also, the second paragraph includes a list, and we have a way of presenting proper lists. Finally, that bit at the bottom is what we'd normally call "fine print", and we actually have a tag for that!

- We can make text stand out via any of several tags:
  - **`i`** will make text *italicized*
  - **`em`** will make text *emphasized*. By default, this will usually be the same thing (until you change it)
  - **`b` bolds**
  - **`strong`** mark text as important (by default, bolded)
  - **`u`** marks text as "stylistically different" (by default, underlined)
  - **`small`** marks text that should be smaller than normal (e.g. for fine print)
- An unordered (bulleted) list is achieved with **`ul`**; an ordered (numbered) list with **`ol`**
  - Either way, a list item within the list is marked with **`li`**
- Fine print is easy; just use **`small`**
  - I also felt like adding contact info to the footer, so I simply added a *line break* (**`br`**)

```
<html>
<head>
      <title>Bad Ideas</title>
</head>
<body><!-- Student A Studentson, 9999999 -->
<header>
      Maybe we could use this for navigation?
</header>
<p>
      There is no <em>established</em> correlation between good ideas and actually making
      money. <strong>Novelty</strong> is a reasonable selling point. So go ahead, and snap
      up all of those ideas everyone else is <u>too smart</u> to use, and make them your
      own!
</p>
<p>
      Wonderful (terrible) ideas include:
      <ul><!--ul = unordered (bulleted list)-->
            <li>ejector seats for helicopters</li><!--Each 'list item' is a bullet-->
            <li>solar-powered flashlights</li><!--Please remember the </li>-->
            <li>research-paper autogenerators</li>
            <li>monkey chauffeurs</li>
      </ul><!--If we wanted a numbered list instead: ol (ordered list)-->
</p>
<footer>
      <small>Disclaimer: no warranty is expressed or implied for the ideas contained herein.
      You are solely responsible for trying to find your own toes after implementing
      these ideas.</small><br/><!--br is for a line break. Use them sparingly-->
      You can contact me at fakeyfakeaddress@fakeemail.com
</footer>
</body>
</html>
```

What does this look like? Let's find out!



Maybe we could use this for navigation?

There is no *established* correlation between good ideas and actually making money. **Novelty** is a reasonable selling point. So go ahead, and snap up all of those ideas everyone else is too smart to use, and make them your own!

Wonderful (terrible) ideas include:

- ejector seats for helicopters
- solar-powered flashlights
- research-paper autogenerators
- monkey chauffeurs

Disclaimer: no warranty is expressed or implied for the ideas contained herein. You are solely responsible for trying to find your own toes after implementing these ideas.
You can contact me at fakeyfakeaddress@fakeemail.com

There's something else worth pointing out: while the **`p`**, **`header`**, and **`footer`** tags were all *block-level* — they actually define their own blocks — that isn't true for the **`i`**, **`em`**, **`b`**, **`strong`**, **`u`**, and **`small`** tags.

These tags (and some other similar tags) don't define their own blocks, because they aren't intended to create separations, or designate independent sections of text. Rather, they're meant to identify portions (or selections) of text that should be treated as special. These are what we call *inline* tags.

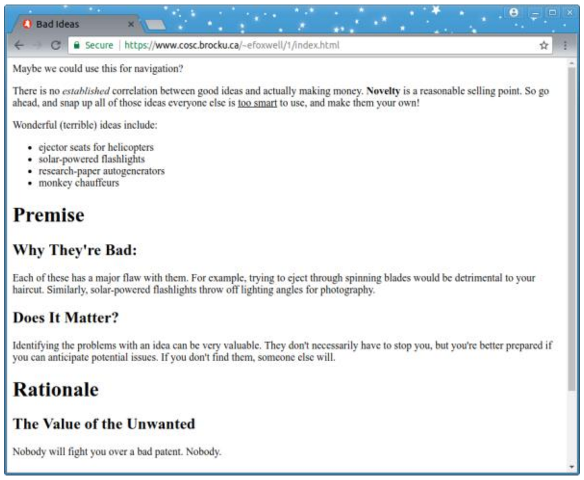
We're doing far better now. By this point, I think we can start adding a bit more detail. However, as the page gets bigger, our organization should also match.

This page is going to have two `articles` (one to address the premise, and one to explain a rationale). Each article will have a couple of `sections`. To best identify each section, I'll be putting in *headings*, very similar to the heading styles we used in Word.

Let's put this between the list and the `footer`:

```
<h1>Premise</h1> <!--By default, a separate block-->
<article>
      <h2>Why They're Bad:</h2>
      <section>
            Each of these has a major flaw with them. For example, trying to eject through
            spinning blades would be detrimental to your haircut.
            Similarly, solar-powered flashlights throw off lighting angles for photography.
      </section>
      <h2>Does It Matter?</h2>
      <section>
            Identifying the problems with an idea can be very valuable. They don't
            necessarily have to stop you, but you're better prepared if you can
            anticipate potential issues. If you don't find them, someone else will.
      </section>
</article>
<h1>Rationale</h1>
<article>
      <h2>The Value of the Unwanted</h2>
      <section>
            Nobody will fight you over a bad patent. Nobody.
      </section>
      <h2>So What?</h2>
      <section>
            So you can repurpose them, to make <em>good</em> ideas! Maybe you can
            ditch out the side of the helicopter and deploy gliding wings? Maybe you can
            add a battery to the flashlight for the solar panel to charge?
      </section>
</article>
```

What does this look like?



Maybe we could use this for navigation?

There is no *established* correlation between good ideas and actually making money. **Novelty** is a reasonable selling point. So go ahead, and snap up all of those ideas everyone else is too smart to use, and make them your own!

Wonderful (terrible) ideas include:

- ejector seats for helicopters
- solar-powered flashlights
- research-paper autogenerators
- monkey chauffeurs

Premise

Why They're Bad:

Each of these has a major flaw with them. For example, trying to eject through spinning blades would be detrimental to your haircut. Similarly, solar-powered flashlights throw off lighting angles for photography.

Does It Matter?

Identifying the problems with an idea can be very valuable. They don't necessarily have to stop you, but you're better prepared if you can anticipate potential issues. If you don't find them, someone else will.

Rationale

The Value of the Unwanted

Nobody will fight you over a bad patent. Nobody.

We're definitely making good progress here.

As the page gets longer, it might be nice to be able to jump between different points in the page.

Let's try something a little odd. Below all four of the **`h2`** tags, let's add some *empty* ***anchor tags***, with the **`id`** attribute assigned:

```
<h2>Why They're Bad:</h2>

<a id="whybad"/>
```

…

```
<h2>Does It Matter?</h2>

<a id="doesmatter"/>
```

…

```
<h2>The Value of the Unwanted</h2>

<a id="value"/>
```

…

```
<h2>So What?</h2>

<a id="sowhat"/>
```

Note the quotation marks around each `id` label.

What does this do? It creates a ***bookmark***; a place in the page to which you can jump. In the browser's address bar, try adding `#value` (in other words, change `main.html` to `main.html#value`).

The octothorpe is how you identify the separation between filename and bookmark name. Actually, you can assign the `id` attribute to any HTML tag you like, but this is pretty much the only use for the empty version of the anchor tag.

What about the non-empty version? That's the primary use for anchors: *linking to documents*. In this case, we can link to the *same* document.

We'll get to a more interesting example of anchors in a moment, but first let's just make a simple table of contents. We don't need any special tags for this, but feel free to if you like (e.g. bulleted list).

I'll go simple, between the `header` and the first paragraph:

```
<a href="#whybad">Why not to</a> <a href="#doesmatter">Is this pointless?</a>
<a href="#value">Some value</a> <a href="#sowhat">Why not?</a>
```

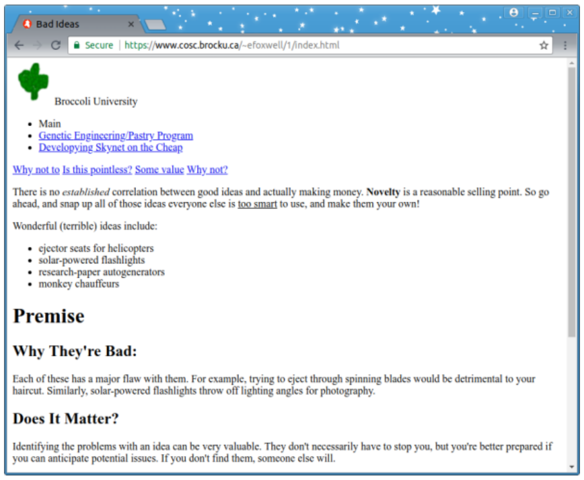
Reload the page, and we now have clickable links!

The **`href`** attribute is a *hypertext reference* (i.e. an identifier for the corresponding page's address). Normally, these would include a filename (possibly an entire path to another website/domain), but since we're linking to the *same page*, we can skip the filename.

Although, that brings up the issue of the other two pages we'd planned to create. Also, we haven't *really* filled in the `header` yet. Additionally, we don't even have any pictures yet!

Let's take care of all of this together, with a **`nav`**igation bar:

```
<header>
        <img src="broccoli.png" alt="Broccoli Logo"/>Broccoli University
        <nav>
                <ul>
                        <li>Main</li>
                        <li><a href="geneteclair.html"/>Genetic Engineering/Pastry Program</a></li>
                        <li><a href="cyberdyne.html"/>Developying Skynet on the Cheap</a></li>
                </ul>
        </nav>
</header>
```



Broccoli University

- Main
- Genetic Engineering/Pastry Program
- Developying Skynet on the Cheap

Why not to Is this pointless? Some value Why not?

There is no *established* correlation between good ideas and actually making money. **Novelty** is a reasonable selling point. So go ahead, and snap up all of those ideas everyone else is too smart to use, and make them your own!

Wonderful (terrible) ideas include:

- ejector seats for helicopters
- solar-powered flashlights
- research-paper autogenerators
- monkey chauffeurs

Premise

Why They're Bad:

Each of these has a major flaw with them. For example, trying to eject through spinning blades would be detrimental to your haircut. Similarly, solar-powered flashlights throw off lighting angles for photography.

Does It Matter?

Identifying the problems with an idea can be very valuable. They don't necessarily have to stop you, but you're better prepared if you can anticipate potential issues. If you don't find them, someone else will.

So many new things!

- First, notice that we're using **`a`**nchor tags to link to other documents now
  - I didn't bother including a link to the same page, but there wouldn't have been any harm in it
- We've used the **`img`** tag (another empty element!) to include an image (draw one, or download one)
  - The **`src`** attribute lists the image's source (filename). Unless you specify a path, it assumes the image is in the same folder as the .html file
  - We *could* have also included **`width`** or **`height`** attributes, but they didn't seem necessary here
- The **`nav`** tag is for a navigational bar

Again, we've introduced new tags that are identical in result to what we've already seen (e.g. a navigational bar is apparently identical to a section?). We'll be adding significance next week.

Naturally, the `geneteclair.html` and `cyberdyne.html` documents don't really exist (yet). Create whatever two pages you like, and link to those. Just make sure that they include links to each other, and back to this document (for easier navigation).

So, what did we end up with altogether?

```
<html>
<head>
       <title>Bad Ideas</title>
</head>
<body><!-- Student A Studentson, 9999999 -->
<header>
       <img src="broccoli.png" alt="Broccoli logo"/>Broccoli University
       <nav>
              <ul>
                     <li>Main</li>
                     <li><a href="geneteclair.html"/>Genetic Engineering/Pastry Program</a></li>
                     <li><a href="cyberdyne.html"/>Developying Skynet on the Cheap</a></li>
              </ul>
       </nav>
</header>
<a href="#whybad">Why not to</a> <a href="#doesmatter">Is this pointless?</a>
<a href="#value">Some value</a> <a href="#sowhat">Why not?</a>
<p>
       There is no <em>established</em> correlation between good ideas and actually making
       money. <strong>Novelty</strong> is a reasonable selling point. So go ahead, and snap
       up all of those ideas everyone else is <u>too smart</u> to use, and make them your own!
</p>
<p>
       Wonderful (terrible) ideas include:
       <ul><!--ul = unordered (bulleted list)-->
              <li>ejector seats for helicopters</li><!--Each 'list item' is a bullet-->
              <li>solar-powered flashlights</li><!--Please remember the </li>-->
              <li>research-paper autogenerators</li>
              <li>monkey chauffeurs</li>
       </ul><!--If we wanted a numbered list instead: ol (ordered list)-->
</p>
<h1>Premise</h1> <!--By default, a separate block-->
<article>
       <h2>Why They're Bad:</h2>
       <a id="whybad"/>
       <section>
              Each of these has a major flaw with them. For example, trying to eject through
              spinning blades would be detrimental to your haircut.
              Similarly, solar-powered flashlights throw off lighting angles for photography.
       </section>
       <h2>Does It Matter?</h2>
       <a id="doesmatter"/>
       <section>
              Identifying the problems with an idea can be very valuable. They don't
              necessarily have to stop you, but you're better prepared if you can
              anticipate potential issues. If you don't find them, someone else will.
       </section>
</article>
<h1>Rationale</h1>
<article>
       <h2>The Value of the Unwanted</h2>
       <a id="value"/>
       <section>
              Nobody will fight you over a bad patent. Nobody.
       </section>
       <h2>So What?</h2>
       <a id="sowhat"/>
       <section>
              So you can repurpose them, to make <em>good</em> ideas! Maybe you can
              ditch out the side of the helicopter and deploy gliding wings? Maybe you can
              add a battery to the flashlight for the solar panel to charge?
       </section>
</article>
<footer>
       <small>Disclaimer: no warranty is expressed or implied for the ideas contained herein.
       You are solely responsible for trying to find your own toes after implementing
       these ideas.</small><br/><!--br is for a line break. Use them sparingly-->
       You can contact me at fakeyfakeaddress@fakeemail.com
```

```
</footer>
</body>
</html>
```

There should only be two more requirements left for this lab: a `table`, and a *clickable image*.

For the latter, simply put an `img` tag within a clickable anchor tag. You can even just wrap an anchor around the logo, if you want.

For the table, try something like:

```
<table border="1">
        <tr>
                <th colspan="2">Broccoli</th>
        </tr>
        <tr>
                <th>Observation</th><th>Explanation</th>
        </tr>
        <tr>
                <td>We use "Broccoli" as an example pretty often</td>
                <td>Broccoli is delicious, as are most universities</td>
        </tr>
</table>
```

We still haven't used `div` or `span` yet, but those will probably be more fun next week.