APCO/IASC 1P01 – Lab 9 – CSS

This lab is an extension of Lab 8. If you completed that lab and followed instructions carefully, this week could end up being atypically easy.

Tasks:
The requirements of Lab 9 are as follows:

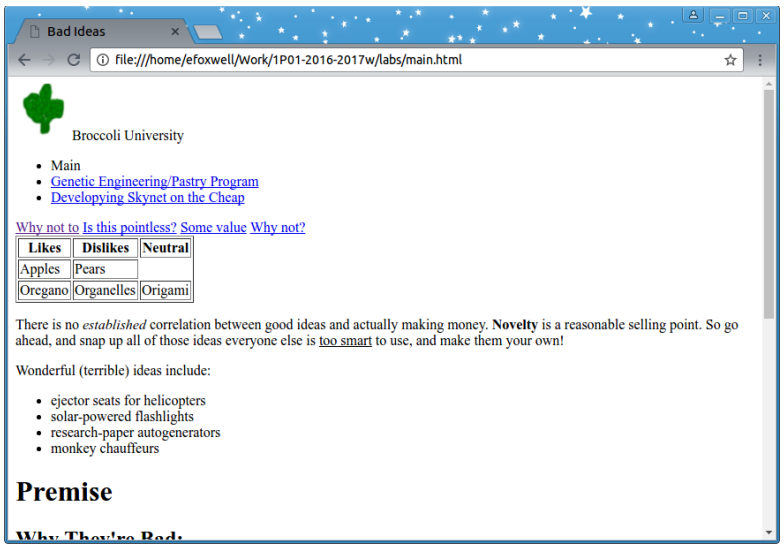
- All of the requirements of Lab 8 (refer to that lab for details)
- Add some CSS

Beyond that, how much time you spend on this is strictly a matter of how much practice you'd like.

Styling:
First off, let's look at the `table` tag that we skipped last week.
Assuming we're building on what we wrote last week, let's try temporarily putting a table immediately before the first paragraph:

```
<table border="1">
	<tr>
		<th>Likes</th><th>Dislikes</th><th>Neutral</th>
	</tr>
	<tr>
		<td>Apples</td><td>Pears</td>
	</tr>
	<tr>
		<td>Oregano</td><td>Organelles</td><td>Origami</td>
	</tr>
</table>
```

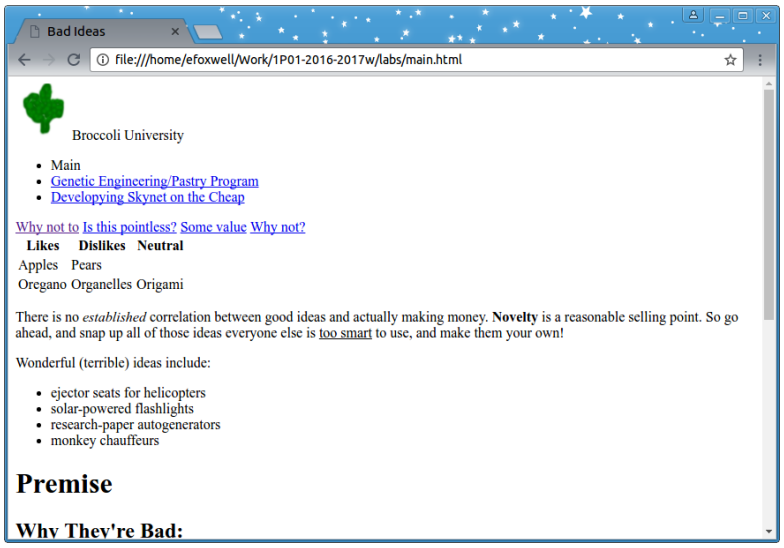
which gives us:



There are a few new things to explain here:

- The `table` tag creates a table
  - In this context, the table's effectively a container (the outer box only)
- A table contains some number of rows (`tr`)
- Each row contains either *table data* (`td`) or *table header* (`th`) cells
- The `border` attribute is used to tell it to include outlines
  - This is now *deprecated*; we'll be looking at a better replacement

What happens if we take out **`border="1"`**?

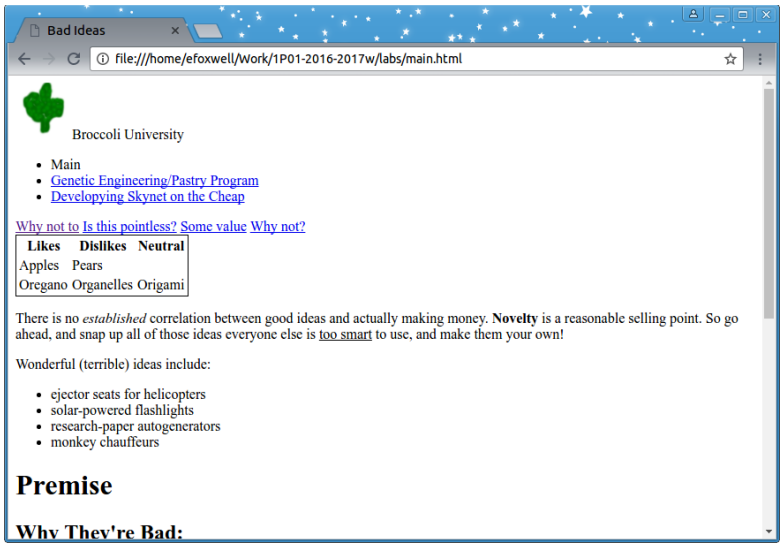


It's still tabular; it's simply missing the border. Instead of relying on HTML, which is supposed to designate *structure*, let's try *styling* it!

Try using this for the first half of the document:

```
<html>
<head>
	<title>Bad Ideas</title>
	<style>
		table {
			border: 1px solid black;
		} /* We now start and end comments differently. */
	</style>
</head>
<body><!-- Student A Studentson, 9999999 -->
<header>
	<img src="broccoli.png"/>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>
<table>
	<tr>
		<th>Likes</th><th>Dislikes</th><th>Neutral</th>
	</tr>
	<tr>
		<td>Apples</td><td>Pears</td>
	</tr>
	<tr>
		<td>Oregano</td><td>Organelles</td><td>Origami</td>
	</tr>
</table>
<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>
…
```

What does it look like now?



This isn't what we wanted, but it's enough to explain what's going on.

- We can use the `style` tag within the document's `head` to define new *styling* rules
  - This is what we call an ***embedded stylesheet***, because it's a listing of style stuck within the document
- A rule has three parts:
  - A selector – to let you know to which elements it applies
  - Properties – aspects of the elements' styling you wish to change
  - Values – what you wish to set those properties to
  - In this case, the rule *selects* all `table` tags, and assigns the values of *1px solid black* to it (this is actually shorthand for setting *black* to the *border-color* property, etc, but that doesn't matter here)
  - Note that you need a *semicolon* to terminate each list of values (before starting the next property), and property/value pairs are enclosed within *braces*

So, obvious question: how do we add the border to the `th` and `td` cells? Do we just add additional rules?
We could. That would work. But it would be overkill.

What we *really* want to do is assign the same property-value tuple to multiple different tags, right?
That means we need a wider *selector*.

- We can use the comma (`,`) to combine our selectors
  - e.g. **`table, th, td`** means the corresponding rule applies to **`table`** *or* **`th`** *or* **`td`**

So let's try that:

```
table, th, td {
      border: 1px solid black;
} /* We now start and end comments differently. */
```

This displays the table correctly. However we can have more fun with it this time.

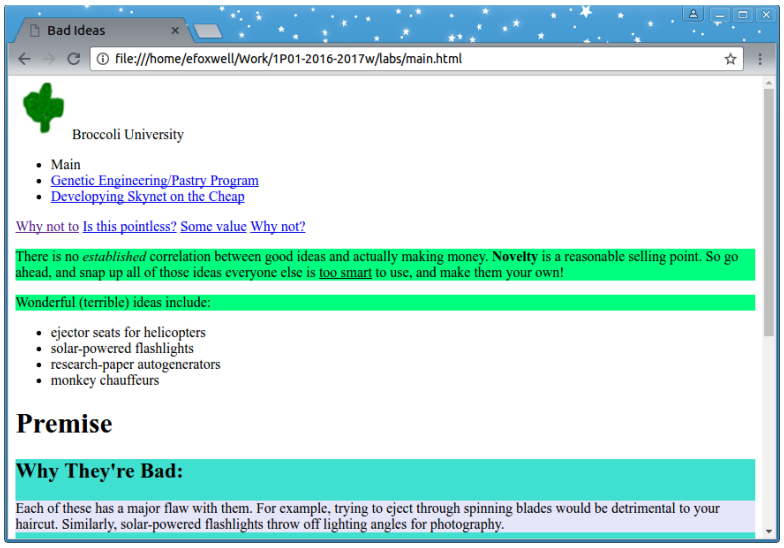
- We could change `black` to `red`, or `blue`
- Instead of `solid`, we could try `dotted` or `dashed` or `double`
- We could pick a different thickness, other than 1 pixel wide

Still, this was just to introduce the concept of a stylesheet. Feel free to delete the table and its corresponding style, or leave it in. I'll be deleting mine before continuing.

Remember that the paragraphs, `articles`, and `sections` all define their own blocks. But actually seeing proof of that isn't easy with HTML alone. Let's try adding some style:

```
p {
	background-color: springgreen;
}
article {
	background-color: turquoise;
}
section {
	background-color: lavender;
}
```

What's the result?



Disclaimer: I realize this is a terrible colour scheme. Feel free to pick other names from here: https://www.w3schools.com/cssref/css_colors.asp

While we're at it, maybe we could set a few more properties:

```
p {
	background-color: springgreen;
	margin-left: 16pt;
}
article {
	background-color: turquoise;
	padding: 16pt;
	border-radius: 8pt;
}
section {
	background-color: lavender;
	border-radius: 4pt;
	padding: 8pt;
}
h2 {
	margin: 4pt 12pt;
}
```

- The *padding* is the space requested *within* a block before the contents start
- The *margin* is the space requested *outside* the block
- The *border-radius* property can add rounded corners (larger number → rounder corners)
- Many properties are shorthand for more complicated collections of properties
  - *margin* with a single number is shorthand for *margin-top*, *margin-bottom*, *margin-left*, *margin-right*
    - Of course, this also applies to *padding*, etc.
  - If you specify two numbers, the first is applied to *-top* and *-bottom*; the second to *-left* and *-right*

This is probably a good time to remind you that you can right-click on any element, and *inspect* it within your browser. The CSS portion should be on the right.

Before we move on, just for ha-ha's, let's also change the colour of the bulleted lists:

```
ul { color: rgb(150,40,255); }
```

- It didn't really matter whether we styled the *unordered lists* (`ul`) or the *list items* (`li`) themselves
  - The list items will inherit it from the list anyway
- Note that this is a different way to specify colour:
  - First choose red, then green, then blue
  - No intensity in a colour channel is reflected by 0; maximum intensity by 255
  - If all three channels are the same, you get a shade of grey
    - By this definition, black (0,0,0) and white (255,255,255) are considered greys

Suppose I want to make some text be *very* special. Yes, we have emphasis (`em`) and strong (strong), but what if we want… fancy!
There's no `fancy` tag in HTML, unfortunately.

Does that mean we can't designate a portion of text as being "fancy"? Of course not!

Consider the following:

```
<p class="fancy">
      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>
```

What did it do? Try saving the document and refreshing the browser to see.
I'll wait.
…
Back yet?
Great! Didn't do anything, did it?
Ah, but it did!

- The **`class`** attribute is used to designate any arbitrary *role* or *category* you like
  - You can use it to fulfill semantic meanings not covered by existing HTML tags
- You can add as many members to the same class as you like, whether block, inline, or combined

Before we add meaning to being "fancy", I just want to make one more change:

```
Wonderful (<span class="fancy">terrible</span>) ideas include:
```

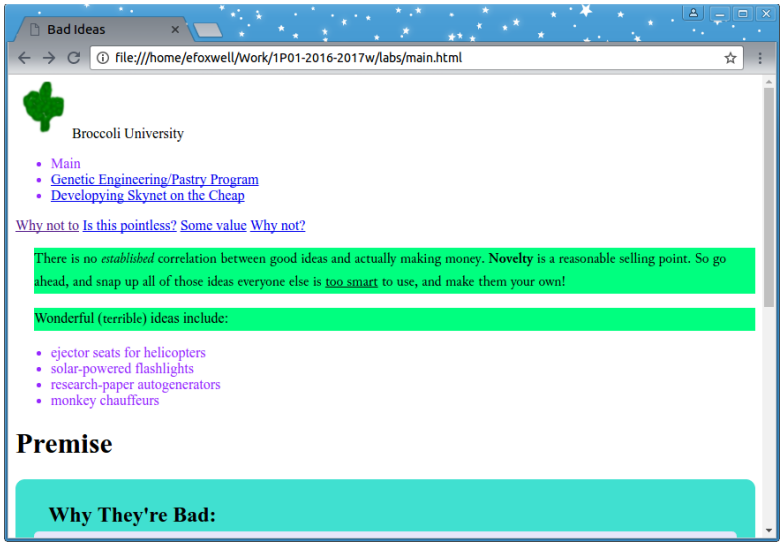
The reason we have a **`span`** tag is to have a ***generic*** *inline tag* for identifying a portion of a line of text; generally to add a class to it. Until you do that, it basically doesn't do anything at all.
Similarly, the **`div`** tag is the *block-level* counterpart (though this can be overridden for each, anyway).

How can we make something look "fancy"? Let's try a different font! Hmm… *Junicode* looks pretty fancy!

- What? You don't *have* Junicode? Oh, then how about Comic Sans MS? Comic Sans is pretty fancy
  - What? You deleted Comic Sans in a fit of design-mourning rage? How about *some* cursive?
    - Failing that, any sans-serif?

```
.fancy { font-family: "Junicode", "Comic Sans MS", cursive, sans-serif; }
```

Again, there's a **`font`** shorthand for combining properties. Feel free to read more here:
https://www.w3schools.com/css/css_font.asp



Note that I *do* have Junicode, so yours will probably look a little different.
However, the point remains: We can now designate any tag, block, or portion of text that we like as being *fancy*! Neat!

(If it wasn't clear, all we need to do to *select* a class is to put a period before its name in the CSS selector)

Hmm. What if we wanted that first heading ("Premise") to use the *Impact* font? And *only* that bit?
There's no need to define a class for that, right?

We have two choices. Let's briefly explore both.

```
<h1 style="font-family: Impact;">Premise</h1>
```

This approach is known as an ***inline style***. We simply assign the style like any other *attribute*.
Inline styles are discouraged, in part because:

- If we want to restyle the document, we need to explicitly search out each instance
- It may conflict with styles defined at the top (not seeing both simultaneously may confuse things)
- If you ever do decide to start reusing styles, you'll want to undo it anyway

There's another solution, instead:

```
<h1 id="premiseh">Premise</h1>
```

What does that do? Nothing, directly. However, we can create a style rule specifically for that element:

```
#premiseh { font-family: Impact; }
```

There! All you do is use an *octothorpe* before the name of the `id`.
This way, we can make a single-use rule, but we can still keep all of the styling organized together.

However, on that note, what about our other files? In theory, you should have a couple more simple pages you created last week. Oftentimes, we wish to use the same basic styling across multiple pages. We *can* simply copy the rules from one embedded stylesheet into another, but that isn't really ideal. In addition to the other reasons we covered in lecture, if we ever want to change our site's theme, we'd need to edit all of the files! That isn't terribly efficient, is it?

We can use an ***external stylesheet*** to define rules once, and then reference that stylesheet within each document.

Within the document's `head`, in between the `title` and the `style` tags, try adding this:

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

What does that do? It tells the browser to load an additional file (*theme.css*) so it can know additional styling. By putting it ***before*** the `style` tag, we:

- Conform to best practices
- Ensure that the embedded stylesheet applies *after* the external sheet
  - The external sheet is for the general (across the whole site); the embedded is for the specific (*this* page). Newer definitions override older ones, so we want *this* page's rules to trump general ones

And, by putting them in a separate file, we make it easier for multiple web pages to all refer to the same file's style rules.

Let's add a bit more to this sheet. First, maybe we want to *always* see the disclaimer.

```
footer { position: fixed; bottom: 0pt; left: 0pt; background-color: cyan; }
```

The `position:fixed` rule says that we wish to be able to explicitly declare the positioning of the block, relative to the window (instead of its normal flow within the document).
Because we're manually positioning, we can set the `bottom` and `left` as our reference edges.
The background was added, because otherwise the text would be hard to read.

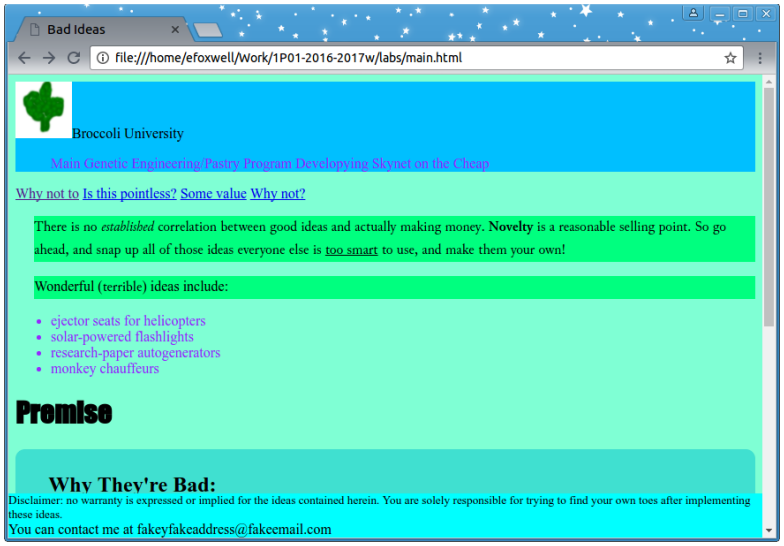
Within our header is a `nav`igation bar. We might like to have the *list items* beside each other, but we don't want to change ***all*** list items.
It's time to revisit our selectors:

```
body { background-color: aquamarine; }
footer { position: fixed; bottom: 0pt; left: 0pt; background-color: cyan; }
header { background-color: deepskyblue; }
header li { display: inline; }
li>a { color: inherit; text-decoration: none; }
```

- Listing two tags, separated only by a space, selects any instance of the latter tag that is a *descendant* of an instance of the former
  - Yeah, that sounds confusing. In this case, we're applying to every **`li`** that is ***somewhere*** within a `header`
  - Setting the `display` to `inline` is simply how we change it from a *block* tag into an *inline* tag
- Listing two tags, separated by a **>**, selects only instances of the latter that are the *immediate child* of an instance of the former
  - In this case, we're only selecting anchor tags that are immediately within list items
  - For example, an anchor within a span, within a list item ***would not*** be selected for this rule

I'll put one final screenshot, followed by everything we've written so far, at the end. But what else can we do? Actually, quite a lot. Much of the appearance of a 'modern' webpage comes from its CSS. We've only scratched the surface, but by adding more and more additional rules, with more careful selectors, you can create a pretty rich document; but still extensible, modifiable, and can share that look with additional pages.

Submission: As a reminder, assuming you finished last week's lab, all you need to do is to show your lab demonstrator that you've added some CSS. Nothing more.



theme.css

```
body { background-color: aquamarine; }
footer { position: fixed; bottom: 0pt; left: 0pt; background-color: cyan; }
header { background-color: deepskyblue; }
header li { display: inline; }
li>a { color: inherit; text-decoration: none; }
```

main.html

```
<html>
<head>
	<title>Bad Ideas</title>
	<link rel="stylesheet" type="text/css" href="theme.css"/>
	<style>
		p {
			background-color: springgreen;
			margin-left: 16pt;
		}
		article {
			background-color: turquoise;
			padding: 16pt;
			border-radius: 8pt;
		}
		section {
			background-color: lavender;
			border-radius: 4pt;
			padding: 8pt;
		}
		h2 {
			margin: 4pt 12pt;
		}
		ul { color: rgb(150,40,255); }
		.fancy { font-family: "Junicode", "Comic Sans MS", cursive, sans-serif; }
		#premiseh { font-family: Impact; }
	</style>
</head>
<body><!-- Student A Studentson, 9999999 -->
<header>
	<img src="broccoli.png"/>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 class="fancy">
	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 (<span class="fancy">terrible</span>) 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 id="premiseh">Premise</h1>
<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>
```

Additional Reading Material:
You should give w3schools a look (http://www.w3schools.com/tags/default.asp and http://www.w3schools.com/cssref/default.asp). They have live previews of modifiable sample code.