

Diary Entry

Link to SCIS website: <http://student.athabasca.ca/~matsph/>

Work for this Unit:

Activities:

Read the Unit 3 FAQ

Watched Derek Banas' CSS tutorial: <https://www.youtube.com/watch?v=l-rTKuEhrCM>

Watched Derek Banas' "Create A Website Menubar" video:
<https://www.youtube.com/watch?v=l3lo7flv7Q>

Watched EJ Media's Horizontal menu tutorial:
https://www.youtube.com/watch?v=dZYy72ObKf0&index=43&list=PLr6-GrHUIVf8JlgLcu3sHigvQjTw_aC9C

Looked up in Yahoo Search certain features of CSS I wanted, like scaling font-size according to viewport width.

Read about the position property from W3schools:
http://www.w3schools.com/css/css_positioning.asp

Browsed through selectors listed here: <http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/>

How I have met the Learning Outcomes:

☐ Be able to write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.

I wrote an external CSS file that is applied to all webpages, and it structures and styles the pages to make them good-looking and organized. The CSS is mostly easily maintainable, except for the inline CSS on the 'other_puzzles.html' page. All CSS is compliant with the WCAG 2.0 standards. The following CSS effect changes were used:

Background color

Font type, size, and color

Padding, margin, and border thickness

Text alignment

Absolute and fixed positioning

Inline, block, and 'invisible' (none) display

Border-radiuses

Z-index

Hover over elements' color

Selectors used include:

Body

Id property

Class property

Lists

Tables

Images

Anchors

First and last elements

Headers

Spans

Divs

Other Requirements

Explain why your changes are improvements by relating them to the personas and scenarios identified in Unit 1

Persona 1: Saun Simmerling

Scenario 2: Interesting Facts about the Rubik's Cube

Saun is easily able to browse through all of the pages using the large clear menu across the top of the page. The front page is also accessible through the logo image.

Scenario 3: How to solve the Rubik's Cube

Because of the solving methods table layout, all the methods are organized and easy to find. The introduction is highlighted as well.

Persona 2: Janice Miranda

Scenario 1: Finding Special Rubik's Cube patterns

The new table borders and resized images make the special patterns easy to read and find.

Scenario 5: The 5*5*5

The new table borders and resized images make the puzzles easy to browse through, and the fixed menu on the side gives her an easy way to jump to different categories to find the puzzle she likes best.

Scenario 7: Solving away from home

Due to dynamic resizing of the menu and special patterns table, Janice can scroll through the special patterns without having to resize the page or move it horizontally.

Persona 3: Saun Simmerling

Scenario 4: Fastest solving methods

Because of the solving methods table layout, all the methods are organized and easy to find. The advanced methods are all clearly under the 'Advanced Methods' title.

Scenario 6: Buying a new puzzle

Robert can use the fixed categories menu to select the category he likes while browsing the most popular puzzles, allowing him to skip between puzzle categories with ease.

Notes:

Went well:

The menu bar went very well, and was much easier than I expected. I was able to format in almost every way I wanted to.

Didn't go well:

The fixed categories navigation panel. I wanted it to stay in a relative position beside the most popular puzzles table, until the user scrolled down, and then it would turn into a fixed position panel, which would move with the user as the scrolled down. I found out this feature isn't reasonably possible without using JavaScript, so for now it hangs in the top left corner covering the menu bar when first navigating to the page.

What was most difficult and why:

Getting the internal CSS to override the external CSS, as I tried to use the style tag with the attribute type="text/css", which for some reason wouldn't work. Then I removed the attribute, and it still didn't work because it wouldn't override the external CSS unless it used the exact same selector.

If done again, would it be done differently and why:

It would be done differently, I would make sure I didn't add or remove any html content unless I needed to in order to fulfill a learning outcome or required task.

How did previous experience help/hinder completing the tasks:

Previous experience with writing HTML and complying with standards made this part much easier, as I followed the standards I already knew, and followed the same steps and sequences as I did with the HTML unit.

Most surprising thing learned:

CSS can select elements with much more precision and exactness than I imagined it did, as it could select elements using filters that can specify blocks, multiple ids and classes, and children in a certain placing (last, first, nth from last, etc.).

Most useful thing learned:

Any element, can be moved to anywhere, be dynamically and statically resized, and have all of its properties changed with ease. CSS is a very easy language to use, almost as easy as HTML.

Map to Course Outcomes

The following shows the relation of the CSS to the Unit 1 site mock-up plan by showing how each piece of code meets the original plan. It has the following structure:

- Page
 - CSS selector
 - CSS property change

- All pages:
 - body
 - The font-size was set to 20px to match the intended description text size.
 - The background was changed to a sky-like color to match the smooth layout and style of the site.
 - #header
 - Georgia font is the closest web-font to the font used by the menu bar items.
 - #header ul
 - Padding, margin, and border are set to 0 to unify the menu into one constant bar.
 - Alignment is set to center, to match the menu positioning in the mock-up pages.
 - Positioning is absolute to keep the menu at the top of the page overtop of the logo.
 - #header li
 - Padding, margin, and border are set to 0 to remove any separation.
 - Font size is kept relative to the page width to ensure the menu bar is always one solid bar, and not word-wrapped when the screen is too small.
 - Vertical alignment is kept in the middle as the text is in the mock-up.
 - #header a
 - Color is set to yellow to match the menu text in the mock-up.
 - Padding of 10px separates the menu items, to distinguish them as buttons.
 - Vertical alignment is kept in the middle as the text is in the mock-up.
 - Background color is set to grey as in the mock-up.
 - #header first-child and last-child a
 - The top-left, top-right, bottom-right, and bottom-left borders are rounded to match the mock-up.
 - #header a.image
 - Z-index is lowered to keep the logo behind the menu.
 - #header a:hover
 - Color and background-color are changed to highlight the button, like it was meant to function.
 - Img.menu
 - Displayed as a block so it can be centered as in the mock-up.
 - span#content
 - Text is blocked, centered with margins, and padded to match the positioning and margins used in the mock-up pages.

- Background-color is added to emphasize the text is the main content.
 - Border-radius is added to match the smooth layout and style of the site.
 - #author-edit
 - Text is aligned to the right and put in italic to minimize it as a footnote.
 - Text is hidden to match the mock-up pages.
- Solving Methods, Solving Records
 - h1
 - Text is centered and set to 45px to approximate the size shown on the mock-up pages.
 - tr.title
 - Title text is enlarged to match the titles in the mock-up pages.
- Facts And History
 - h3
 - Text is indented and set to 25px to approximate the section title in the mock-up page.
- Other Puzzles
 - #navigation (extra positioning is planned with Javascript)
 - Position is fixed so that it can always be accessed (that was part of the reason it was put on the side in the mock-up).
 - Background color is changed to make it pop out as a navigation panel.
 - Border-radius is added to match the smooth layout and style of the site.
 - Margin and padding match the margin and padding used in the mock-up page.
 - Internal span#content
 - Redefinition of span#content with extra left margin to make room for the navigation panel.
- Special Patterns, Brand Versions, Other Puzzles
 - img:not(.menu)
 - Size is fixed to make the images uniform in the tables, as in the mock-up.
 - table, tr, th
 - Width is maximized to spread the contents out across the page.
 - Borders are added as in the mock-up, to separate content.

Self-assessment:

Learning Outcome	Evidence of Meeting the Learning Outcome	Grade	Tutor's Justification of Grading (optional)
Write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.	The CSS I wrote is either in the external CSS file, or in a one-time internal CSS in the 'other_puzzles.html' page. In both, the code is organized with	B	

	<p>comments and well-structured using proper spacing, character case, property ordering, and naming schemes.</p> <p>It is easily maintained as the external CSS file is less than 130 lines long, and the internal CSS only defines styling for one selector.</p> <p>It is standards compliant with WCAG 2.0 by having no obscure or hard to read text, and proper contrast.</p>		
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	--

CSS Notes

It's called CSS because each style sheet builds on top of the other, as they cascade their stylings. Rules built on top of rules.

selector1, selector2 {property: value; property2: value2}

The selectors can be something like "p", which means all paragraphs (with the tag <p>).

It can also be something like ".classname", which means all tags with "class="classname".

It can also be something like "#idname", which means all tags with "id="idname".

The selector can specify a tag and a class or id name with something like "p.classname".

The property is the property to set for the items inside the selector, and the value is the value to set the property to (e.g. color:pink).

selector {property: value1, value2, value3; }

In this case, a property can have multiple values, in case value1 can't be used, it uses value2, and in case value2 can't be used, it uses value3.

CSS can be written and applied in 3 ways:

Inline (embedded in the HTML tags)

Embedded (embedded in the HTML file)

External (has its own separate file, linked to by the HTML files that use it)

Stylesheets can be imported within other stylesheets by using:

@import url ("stylesheet2css");

To link external CSS:

CSS inclusion styles:

Inline:

<p style="color: red"> paragraph text </p>

Internal:

<style type="text/css">

/*CSS code here*/

</style>

External:

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

Positioning

The position property can have 4 different values:

- Static – Default, not affected by the 4 properties (top, bottom, left, and right), and positioned according to the normal flow of the page.
- Relative – The 4 properties position it relative to its normal (static) position.
- Fixed – Positioned relative to the viewport, even when the page is scrolled through.
- Absolute – Positioned relative to the nearest positioned ancestor (if no ancestor, uses document body).

Standards (unofficial): <https://make.wordpress.org/core/handbook/best-practices/coding-standards/css/>

Tabs, not spaces

Each selector should be on its own line (e.g. 'ul, li' should have 'ul,' on one line, and li on another).

Use lowercase and hyphens (-) for names (no camelCase or under_score).

Properties followed by ':' (colon and space).

All lowercase except font names and vendor-specific properties.

Shorten RGB if possible (e.g. #fff instead of #ffffff)

Condense multiple values of properties if possible (e.g. margin: 10px 20px; instead of margin: 10px 20px 10px 20px).

Have CSS selectors in an order (not random placing in the CSS file)

Properties should be listed in clock-wise order (e.g. top-left, top-right, bottom-right, bottom-left).