Subject Code	CS3	Client-side Web Development
Module Code	7.0	Advanced CSS
Lesson Code	7.2	Events in CSS
Time Frame	30 mins.	

TARGET

Time Allocation: 1 minute

After completing this module, you are expected to:

- Enhance web pages by incorporating links styles and events using CSS
- Use appropriate techniques in correcting errors (debugging)
- Identify and practice the latest updates in CSS

HOOK

Time Allocation: 2 minutes

Introduction

In the previous module, you have learned that links have different states representing an event done by the user (i.e., hover) or an occurrence in the website (i.e., visited). In this module, you will also learn about **CSS Events** – events that happen because of user action. But it will not be exclusively for links. This is done by pseudo-classes. These are used to define the state of an HTML element. You will learn how to style a webpage when you hover over a paragraph or click on an image, among some examples.

IGNITE

Time Allocation: 12 minutes

Syntax

Given below is the basic syntax of using pseudo-classes. Take note that in this module, you are not only limited to links for selectors.

```
selector:pseudo-class {
  property: value;
}
```

Pseudo-classes can also be combined with other CSS classes. Given below is an example of a `div` tag with class `content`. It will change color upon hover,

```
div.content:hover {
  color: #ff0000;
}
```

Another combination that can be done with pseudo-classes is with CSS `element` selectors. Take, for example, a `div` element that a user will hover upon. But instead of styling the `div` element upon hover, another HTML element will be styled, which in this example, will be a `p` element. The code listed below is an example of a `p` element becoming visible when a `div` element is hovered upon. You are encouraged to copy and save the code into an HTML file in your text editor to see the result.

```
<!DOCTYPE html>
<html>
      <head>
            <style>
                  p {
                    display: none;
                    background-color: green;
                    padding: 20px;
                  }
                  div:hover p {
                    display: block;
                  }
            </style>
      </head>
      <body>
            <div>Hover over me to show the p element
                  <p>This is a p element</p>
            </div>
      </body>
</html>
```

There are many pseudo-classes available in CSS. You can check them out here: CSS Pseudo Classes (https://www.w3schools.com/css/css_pseudo_classes.asp).

Here are some pseudo-classes from the link given above. You will encounter them more often than other pseudo-classes during our activities for this class.

Pseudo-class	Description
`selector:active`	Selects the active selector
`selector:checked`	Selects all checked elements. Example is in checkboxes
`selector:disabled`	Selects all disabled elements. Example is in textboxes that cannot be edited or buttons that cannot be clicked unless an action is done
`selector:focus`	Selects the element that has focus (i.e., a textbox where the cursor is currently at)
`selector:valid`	Selects all elements with a valid value (i.e., a textbox for passwords with a valid input)

For more examples of how pseudo-classes are used, try copying and saving HTML files in your text editors the following codes.

Using the :focus pseudo-class

```
<!DOCTYPE html>
<html>
        <head>
                <style>
                        input:focus {
                          background-color: yellow;
                        }
                </style>
        </head>
                <body>
                        First name: <input type="text" name="fname"><br>
                        Last name: <input type="text" name="lname"><br>
                        <input type="submit" value="Submit">
                </body>
</html>
```

Using pseudo-class with other classes in links

```
<!DOCTYPE html>
<html>
        <head>
                <style>
                        a.one:link {color:red;}
                        a.one:visited {color:blue;}
                        a.one:hover {color:#orange;}

                        a.two:link {color:red;}
                        a.two:visited {color:blue;}
                        a.two:hover {font-size:150%;}

                        a.three:link {color:red;}
                        a.three:visited {color:blue;}
                        a.three:hover {background:orange;}

                        a.four:link {color:red;}
                        a.four:visited {color:blue;}
                        a.four:hover {font-family:monospace;}

                        a.five:link {color:red;text-decoration:none;}
                        a.five:visited {color:blue;text-decoration:none;}
                        a.five:hover {text-decoration:underline;}

                </style>
        </head>
        <body>

                <p>Mouse over the links:</p>

                <p><b><a class="one" href=# >This link changes color</a></p>
                <p><b><a class="two" href=#>This link changes font-size</a></p>
                <p><b><a class="three" href=#>This link changes background-color</
                a></b></p>
                <p><b><a class="four" href=#>This link changes font-family</a></p>
                <p><b><a class="five" href=#>This link changes text-decoration</a></
                p>

        </body>
</html>
```

Debugging Techniques in CSS

CSS is like any of the programming languages that you have previously encountered. You will encounter errors in your code along the way. But unlike in your G8 C++ or Java (for some campuses) where the program stops running when a problem is encountered, in CSS, the page will still render even with errors, but it will not show in the way that you would have wanted. And unlike in C++ or Java (for some campuses) where the line of the error can be traced, CSS will not give you a clue as to where the error is.

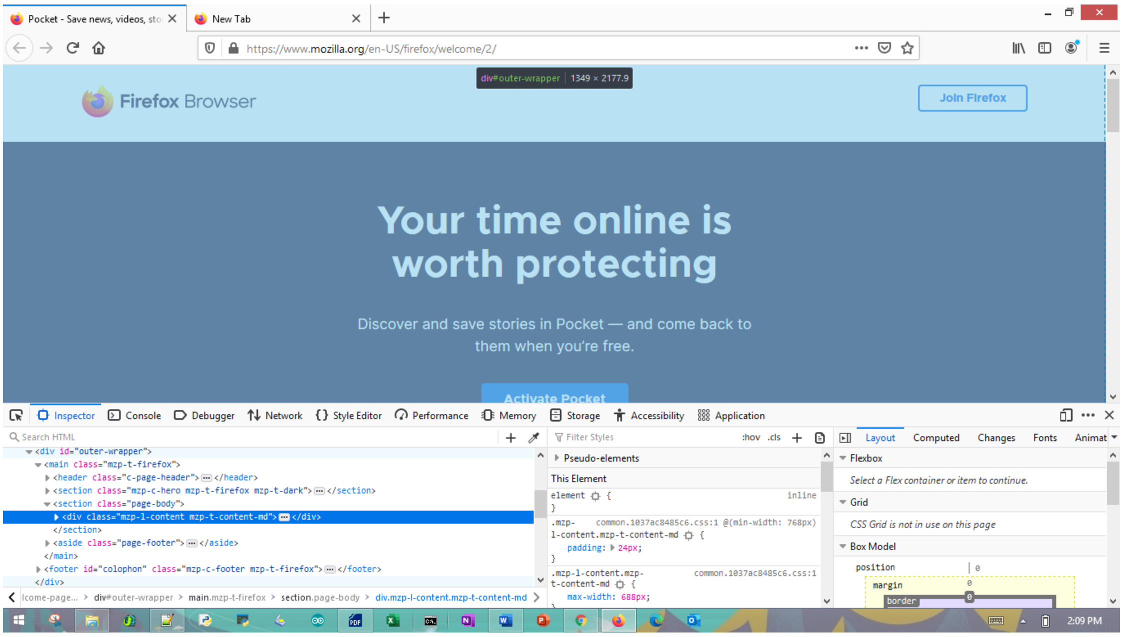
Debugging in CSS is a skill that you will acquire along with experience but listed below are some pointers on where you can start to debug.

- Make sure the proper selector, property, and value where appropriate.
  - Some beginners tend to confuse the different CSS properties and use them on selectors where the property is not applicable. One common example is using the text-align property on a div element to center the whole div container. Text-align only works on text, not whole HTML containers.

  - Another common mistake is using the wrong values on properties. One common example is setting the value of float to “center.” Note that “center” is not a possible value for float.

- Learn the box model
  - The most common frustration in using CSS is the elements never seem to quite align, no matter how many margin pixels you are adding. One example is when you have two div tags that you have set to have the same width, but the other seems larger. The reason for the difference in size is that the other div tag has a padding value. The padding value is not calculated into the actual width property unless you add other CSS properties to calculate the padding into the final width (i.e., box-sizing, flex-box). Understanding the box model can help you a great deal.

- Use the inspect element feature in browsers
  - When you are viewing a website, you can right-click anywhere on the webpage and select “Inspect” (for Chrome) or “Inspect Element” for Mozilla. When you hover over a section of the HTML code, the corresponding part of the webpage is highlighted.


Pocket - Save news, videos, sto
New Tab
https://www.mozilla.org/en-US/firefox/welcome/2/
div#outer-wrapper | 1349 × 2177.9
Firefox Browser
Join Firefox
Your time online is worth protecting
Discover and save stories in Pocket — and come back to them when you're free.
Activate Pocket
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Search HTML
Filter Styles
:hov
.cls
Layout
Computed
Changes
Fonts
Animat
2:09 PM

```
<div id="outer-wrapper">
  <main class="mzp-t-firefox">
    <header class="c-page-header"> … </header>
    <section class="mzp-c-hero mzp-t-firefox mzp-t-dark"> … </section>
    <section class="page-body">
      <div class="mzp-l-content mzp-t-content-md"> … </div>
    </section>
    <aside class="page-footer"> … </aside>
  </main>
  <footer id="colophon" class="mzp-c-footer mzp-t-firefox"> … </footer>
</div>
```

lcome-page… > div#outer-wrapper > main.mzp-t-firefox > section.page-body > div.mzp-l-content.mzp-t-content-md

Pseudo-elements

This Element

```
element {                                    inline
}
.mzp-    common.1037ac8485c6.css:1 @(min-width: 768px)
l-content.mzp-t-content-md {
  padding: 24px;
}
.mzp-l-content.mzp-             common.1037ac8485c6.css:1
t-content-md {
  max-width: 688px;
```

Flexbox

Select a Flex container or item to continue.

Grid

CSS Grid is not in use on this page

Box Model

position 0
margin 0
border 0

Figure 1. Inspect Element of Mozilla Firefox

- Make use of available plugins and developer tools.
    - When you already have a good grasp of CSS concepts, you can take the next step and use more advanced equipment for coding in CSS. There are several plugins that you can use to help ease your debugging.
    - Check out the developer tools that Mozilla is offering in this link: Debugging CSS.zz (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Debugging_CSS)

Latest Updates in CSS

The latest version of CSS as of now is CSS3. The main feature of this new version is that it makes web pages (1) more interactive with users by supporting animations and (2) more responsive and adaptive by being able to adjust to the device where the webpage is being viewed.

Because of CSS, there are also many tags in HTML that are no longer used. Some of them include text-formatting tags like `<bold>. <i>,` and `<ul>.` Some tags used for layout are also rendered obsolete, such as `<center>.`

Best Practices in CSS

When coding in CSS, there are some practices that will make work easier. It will make debugging the code faster and makes collaboration with others easier. Given below are some best practices that you can do when coding:

- Code your HTML first.

  CSS is used to design your web page. It will be difficult and very unorganized to do a bit on HTML, then do CSS, then return to HTML, then CSS again.

- Make your code readable.

  Always observe proper indentation and new line spacing where needed. Given below are two versions of the same code with different line spacing.

  Version 1:

```
h1{
    color: red;
    text-align: center;
    border: 1px solid black;
}
```

  Version 2:

```
h1{color: red; text-align: center; border: 1px solid black;}
```

  Which line spacing technique do you think allows for easier and faster reading of code? Version 1 allows for better reading. Assume also that you have to edit the code later on. Which line spacing technique do you think allows for faster location of the line of code that needs to be modified? It's version 1.

- Group selectors with the same formatting together

Let’s say you have an <h1> tag, a <p> tag, and an <h2> tag with the same formatting. You can group them by using commas so that the set of properties you will use for their formatting will only be declared once. Look at the code snippet below for an example:

```
 h1, h2, p {
    color: red;
    text-align: center;
    border: 1px solid black;
}
```

- Learn to use the short-hand property of CSS properties

There are multiple CSS properties whose values can be declared in a single line by using the shorthand for these CSS properties. Examples include margin, border, and padding. Look at the code snippets below for an example:

```
div{

    margin-top: 10px;
    margin-right: 5px;
    margin-bottom: 10px;
    margin-left: 5px;
}
```

```
div{
    margin: 10px 5px 10px 5px;
}
```

```
span{
    border-width: 1px;
    border-style: solid;
    border-color: black;
}
```

```
span{
    border: 1px solid black
}
```

- Avoid using inline CSS and use external stylesheets as much as possible.

This provides a more unified location for all CSS formatting for your project. Whenever you need to change or modify the whole website's formatting, using external style sheets allows you to modify only a single file. Using internal and inline CSS would mean that you have to visit all the HTML files in your website to modify the CSS code in each file.

- Organize your CSS code

  It's good practice to always start from the top and work your way down to the bottom. For example, at the top of your CSS code, the formatting for header elements should be found, then below them are formatting for the content, and then at the bottom of the CSS code is the formatting for the footer section of the website.

  Aside from top to bottom, structure your code to have the general formatting for containers first, and then work your way inside into the container's specific contents. For example, if you have a div element that contains an image and a paragraph, type the code first for the div (i.e., padding, margin, border) and then type the code for the paragraph and the image after the div.

NAVIGATE

Time allotment: 14 minutes

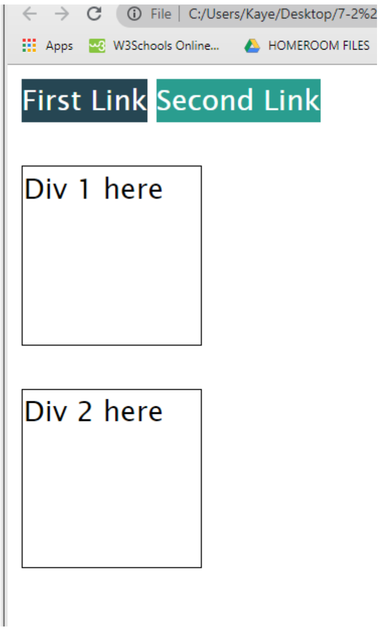
Exercise (Graded)

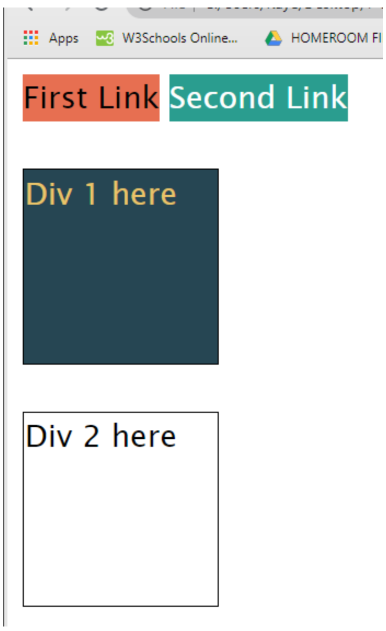
This is a graded exercise. Make sure to submit your output to your teacher.

1. Create a webpage with two `a` elements and two `div` elements. Add <br> in between where it is necessary for aesthetic purposes.
2. Add styles to the four link states for each `a` element. Make sure to style each state distinctively, particularly in `color` and `background-color`.
3. Use **internal style sheets.**
4. When the user clicks on the first `a` element, change the background color and text color of the first `div` element.
5. When the user clicks on the second `a` element, the border of the second `div` element becomes dotted with 10px width and border color green.

Clue: Aside from web pages and images, you can also use links to target a specific HTML tag. In your CSS code, the selector for the HTML tag that you want to modify is `tagName:`**`target`**`{ }.`

Sample Output





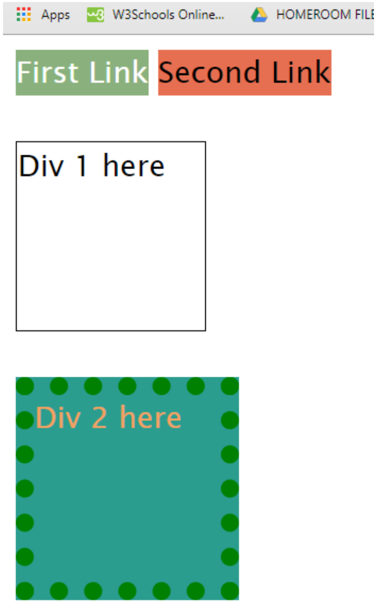


Figure 2. Upon loading	Figure 3. When First Link is clicked	Figure 4. When Second Link is clicked

Note: The screenshots are zoomed in for a better view. Ratio on your final output may not correspond to the screenshots above.

Grading Rubric

Perfect Score: 30/30

HTML	**8-10**	**5-7**	**1-4**
	All HTML tags listed in the specs are present in the final output	1 or 2 HTML tags required in the specs are missing/wrongly implemented in the code	Three or more HTML tags in the specs are missing/wrongly implemented
	Placement of HTML tags is according to specs.	1 or 2 HTML tags are wrongly placed	Three or more HTML tags are wrongly placed.
	Necessary meta tags are also present in the HTML file	Only two meta tags are present	1 or no meta tags used.
Neatness of code	**8-10**	**5-7**	**1-4**
	Code observes proper indentation.	1 or 2 lines of code are improperly indented	Three or more lines of code are improperly indented
	All basic HTML tags are present (i.e. head, title, etc)	1 or 2 HTML tags with no closing pair (where applicable)	Three or more HTML tags with no closing pair (where applicable)
	All HTML tags have their closing counterpart where applicable		
CSS	**8-10**	**5-7**	**1-4**
	All specifications in layout and color are followed	1 or 2 specifications in layout and color are not followed.	Three or more specifications in layout and color are not followed.
	All link states are formatted	1 or 2 link states are not formatted	Links are not formatted.
	The event stated in the activity is demonstrated by the webpage (clicking on a link triggers the specified event)	1 or 2 events stated in the activity are not demonstrated by the webpage	No events are triggered in the webpage

KNOT

Time allotment: 1 minute

Summary

CSS events provide a way to make the website more interactive for the user. This is achieved by using pseudo-classes that define the state of an HTML element for styling. Pseudo-classes can be used with other classes and other selectors also to make styling HTML elements easier.

As more CSS are added to a web page, a good grasp of several best practices and debugging techniques can help you check your CSS code when the output is not turning out the way you prefer it to look.

REFERENCES

Advantage of css3 over css - geeksforgeeks. (2019). *GeeksforGeeks*. https://www.geeksforgeeks.org/advantage-of-css3-over-css/#:~:text=It helps in creating complex,the table rows and columns.&text=Animations: CSS requires the developer,animation features like text-shadow

CSS element element Selector. (n.d.). https://www.w3schools.com/cssref/sel_element_element.asp

CSS Pseudo-classes. (n.d.). https://www.w3schools.com/css/css_pseudo_classes.asp

Narayan, A. (2017). 10 Best Practices in CSS | TO THE NEW Blog. TO the NEW BLOG. https://www.tothenew.com/blog/10-best-practices-in-css/

Stansberry, G. (n.d.). 30 css best practices for beginners. *Code Envato Tuts+*. https://code.tutsplus.com/tutorials/30-css-best-practices-for-beginners--net-6741

Prepared by:	Reviewed by:
KAYE B. ALAMAG SST II PSHS-CARC	ALINE TERESA MENDOZA SST V PSHS- MC