UNIVERSITY OF
Baguio
SCHOOL OF INFORMATION TECHNOLOGY

HUMCOM1

HUMAN-COMPUTER INTERACTION - LECTURE

A Self-regulated Learning Module

Prepared by:

Erna-kristi N. Martinez

UNIVERSITY OF Baguio

Table of Contents

UNIVERSITY OF Baguio

Course Overview

Course Code: HUMCOM1

Course Description: Human-Computer Interaction 1

This course teaches students to design user interfaces based on the capabilities of computer technology and the needs of human factors. The course covers human capabilities, design principles, prototyping techniques, evaluation techniques, and the implementation of graphical user interfaces. Deliverables include short programming assignments and a semester-long individual/group project. Students design a user interface for a system and implement a prototype from a list of informal requirements. Students design a user interface by a design process based on current human–computer interaction principles.

Learning Competencies:

At end of the course, you are expected to

First Grading:

1. Setup the development environment with appropriate front-end development tools that will help in

Midterms:

2. Define cognition and its relevance in interaction design.
3. Identify means on how memory can be enhanced through technology aids
4. Enumerate ways on ways which attention affects people's ability to multitask.
5. Explain through multimedia (i.e. video, interactive presentation, etc.) what is meant by social interaction and its importance in interaction design
6. Describe how technologies can be designed to change people's attitudes and behavior.

Finals:

7. Explain the rationale and rules for an effective interface design methodology in order to establish a criteria for evaluating the quality of user interfaces.
8. Explain how to design user interfaces that anticipate what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.
9. Describe how to bring together concepts from interaction design, visual design, and information architecture to create an interface with elements that are consistent and predictable in their choices and their layout.

Module Requirements:

At the end of each module, you are expected to complete each:

1. Self-check (Quiz)
2. Assignments
3. Hands-on Activities
4. Recitation

*** All module quizzes and other activities shall be placed in a long brown envelope and submitted at the end of each module.

UNIVERSITY OF Baguio

Contact Information:

emartinez@e.ubaguio.edu
https://www.facebook.com/krejpinoy

Consultation Hours:

Course schedule

Course Schedule and Outline

	Lecture	Assigned Reading	Requirements
Week 1	Development Environment Set-up Introduction to HTML5	Lesson 1 Lesson 2	✓ GitHub Account ✓ Published home page
Week 2	HTML Elements Part 1	Lesson 3	✓ Hands-on Activity – Exercise1.html ✓ Exercise2.html
Week 3	Semantic Elements HTML Elements – Tables	Lesson 4 Lesson 5	✓ Exercise3.html ✓ Exercise4.html ✓ Exercise5.html ✓ Exercise6.html
Week 4	HTML Elements – Links HTML Elements – Media	Lesson 6 Lesson 6	✓ Exercise7.html ✓ Exercise8.html
Week 5	HTML Elements – Forms	Lesson 7	✓ Exercise9.html
Week 6	First Grading Exam		
Week 7	Introduction to CSS CSS Implementation	Lesson 8	✓ Hands-on Activity
Week 8	CSS Selectors	Lesson 9	✓
Week 9	CSS Layout	Lesson 10	✓ Hands-on Activity
Week 10	CSS Box Properties CSS Responsive Web	Lesson 11 Lesson 12	✓ Hands-on Activity
Week 11			
Week 12	Midterm Exam		
Week 13	Introduction to JavaScript JavaScript Variables and Types	Lesson 13	✓ Hands-on Activity
Week 14	Expressions and Operators	Lesson 14	✓ Hands-on Activity
Week 15	Control Flow Statements	Lesson 15	✓ Hands-on Activity
Week 17	Functions	Lesson 16	✓ Hands-on Activity
Week 18	Final Exam		Final static website

UNIVERSITY OF Baguio

LESSON 1: Development Environment Set-up

Duration: 2 hours

About this lesson: At the end of this lesson you are expected to:

- Setup and configure your development environment
- Setup a GitHub account and configure local and remote repositories.
- Familiarize the use of basic Git commands and use of code editors.

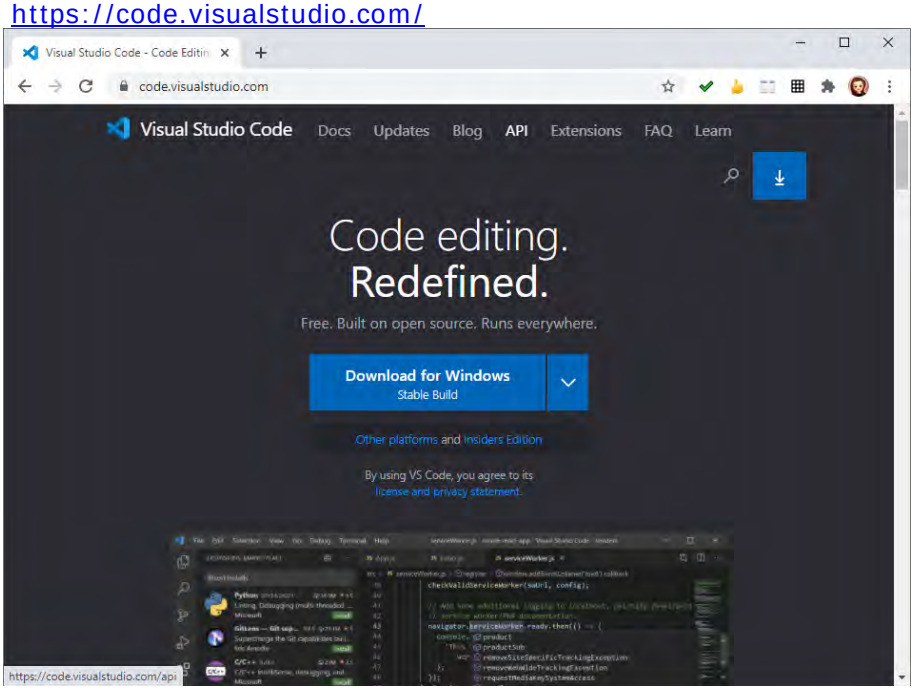
1.1 What you'll need

1. Code editor
    - Visual Studio Code or Notepad++ -> you can use any of these code edits. Both are free with support for various programming languages.
2. Git
3. Nodejs
4. Browser-sync
5. Github
6. Browsers
    - Chrome, Safari, Opera, and/or Edge.

1.2 Development environment setup

a. Visual Studio Code (or Notepad++)

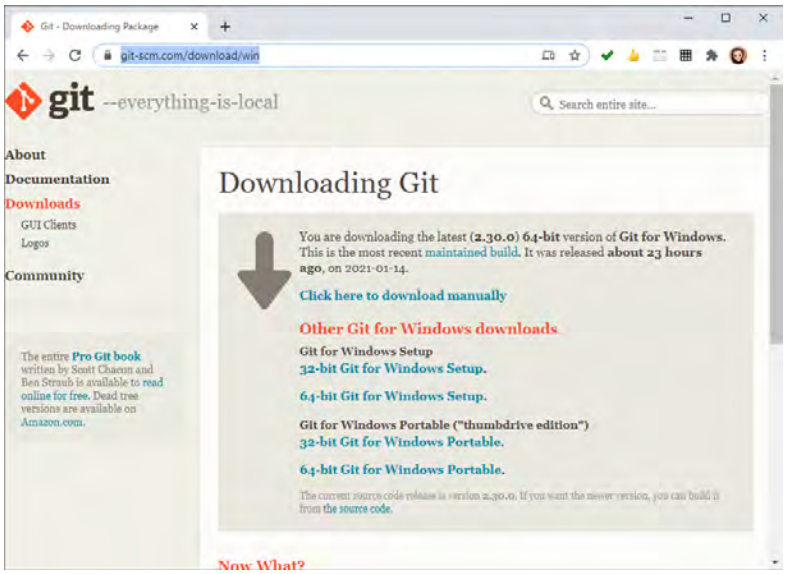
- Go to https://code.visualstudio.com/



- *Download* the *stable* Visual Studio Code installer
- *Install*. Follow the set-up (installation) wizard.
  **** You may use Notepad++ as your code editor. Notepad++ is a lightweight code editor (has lower system requirements) than Visual Studio Code, but is very efficient as a code editor.*

b. Git - is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. (Git and Software Freedom Conservancy, n.d.)

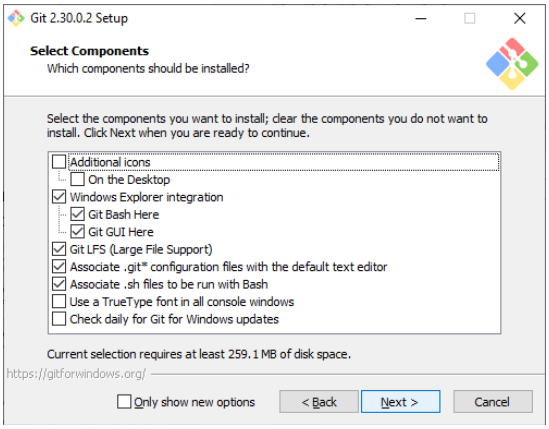
- Go to https://git-scm.com/download/win (for windows)



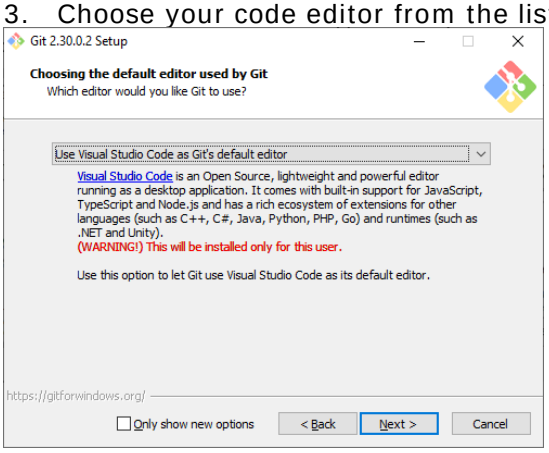
- Download either 32bit or 64bit for Windows, (make sure you download the version compatible for your computer)
- When completed, install the software. Follow the installation steps.

1. Just Click Next in this window

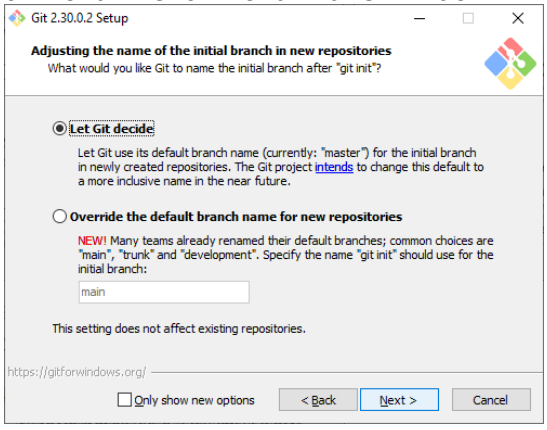
2. Just Click Next in this window



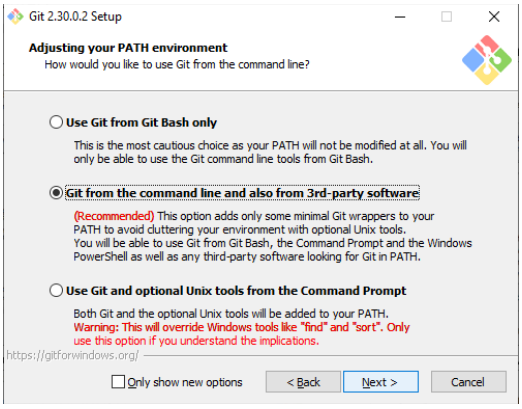
3. Choose your code editor from the list then click Next



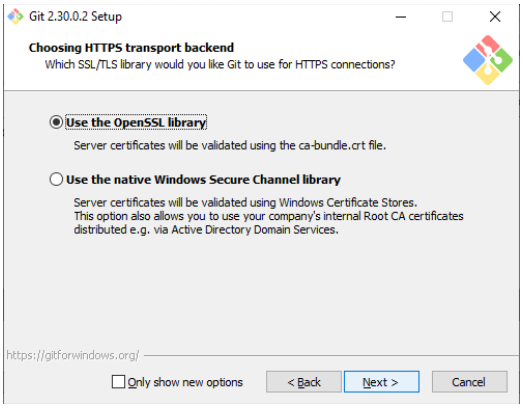
4. Click Next in this Window



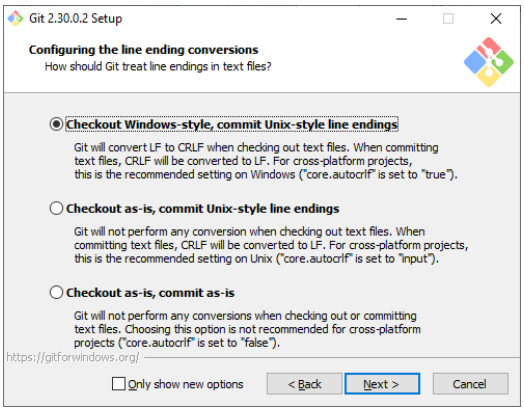
5. Choose the **2nd option,** then click NEXT in this window 6. Click Next in this Window



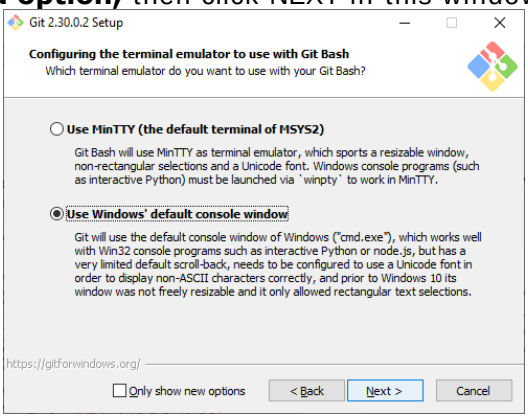
7. Click Next in this Window



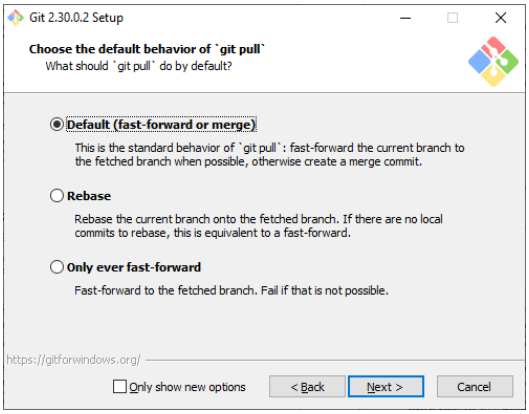
8. Choose the **2nd option,** then click NEXT in this window



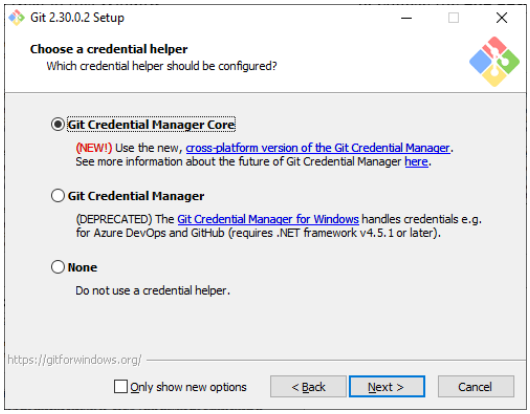
9. Click Next in this Window



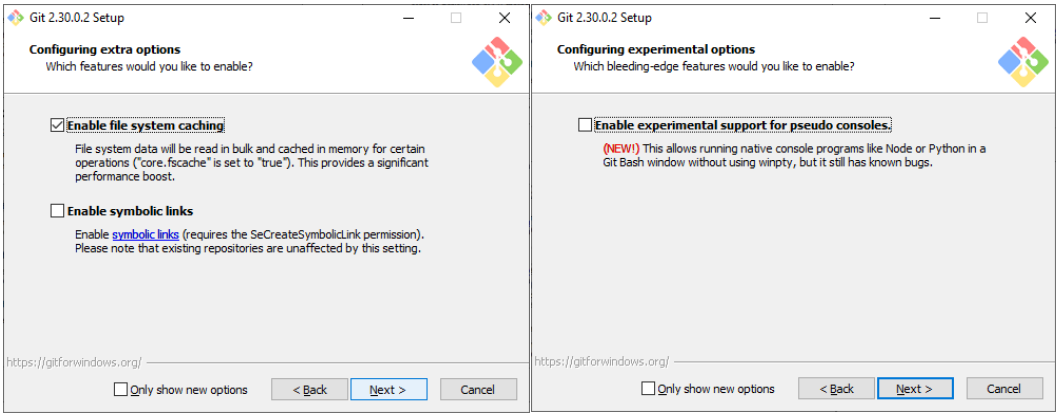
10. Click Next in this Window



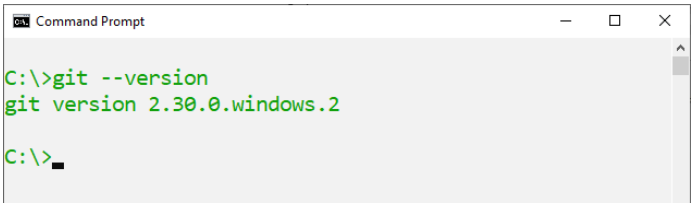
11. Click Next in this Window



12. Click Next in this Window

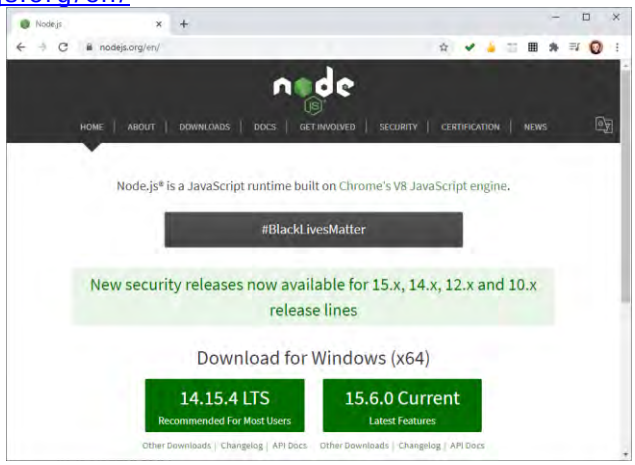


- Finish the installation process from there.
- Check if the application is installed through the Windows command prompt.
- Type git --version on the command prompt.

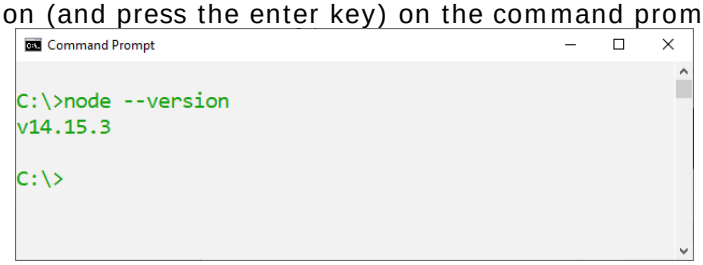


c. Nodejs - As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently.

- Go to https://nodejs.org/en/



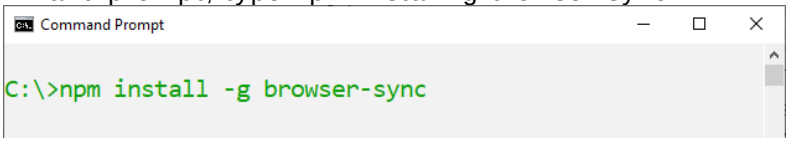
- Download and install the *Nodejs LTS* Windows installer.
- After installation, check if it was installed through the Windows command prompt.
- Type, node --version (and press the enter key) on the command prompt.



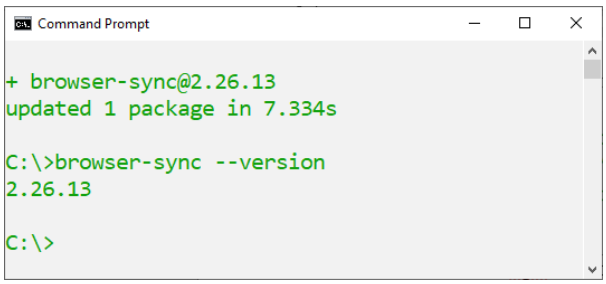
- Then, type, npm –version

d. Browser-sync - is an automation tool that makes web development faster by synchronizing file changes and interactions across many devices.

- This step should only be done after installing NodeJs.
- On the command prompt, type npm install -g browser-sync

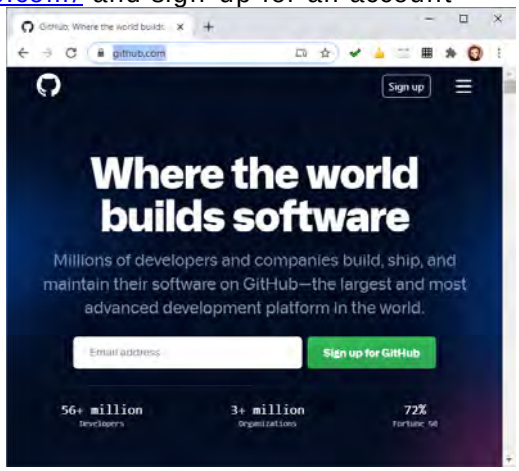


- Verify that it is installed by typing this command on the command prompt: browser-sync --version

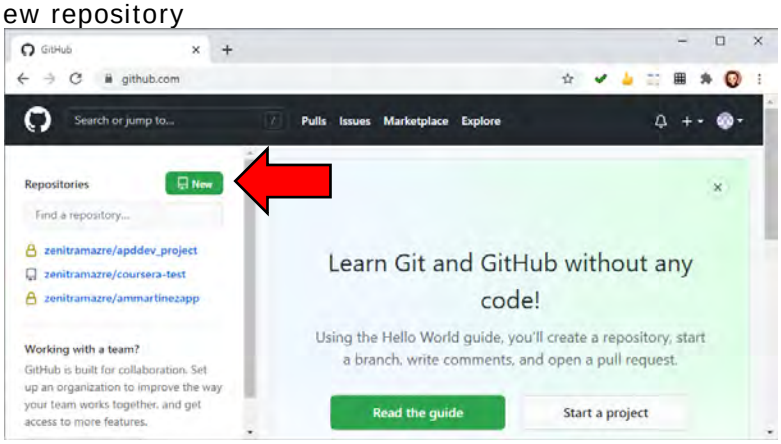


e. Github - GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. You will be using pushing all your activities in your Github repository for this course.

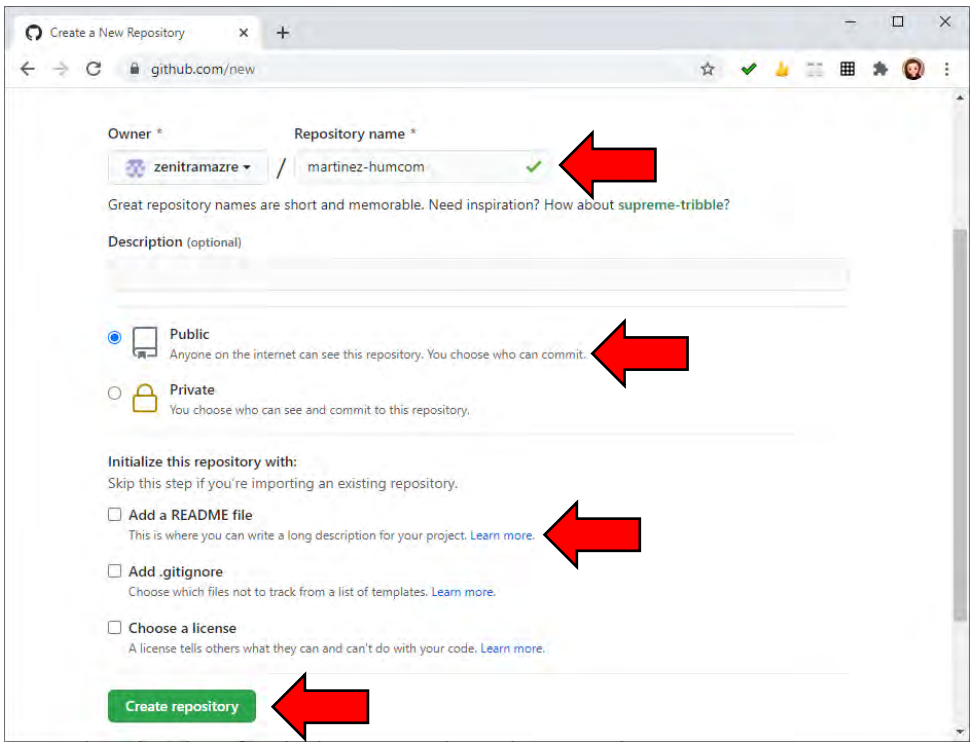
- Go to https://github.com/ and sign-up for an account



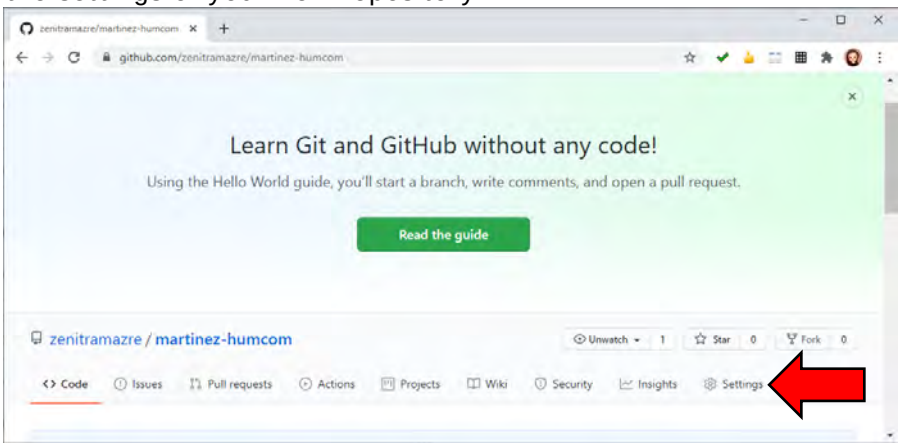
- After signing-up, sign-in to your account.
- Create a new repository



- Name the repository: lastname-humcom
- Set the repository to public. (Setting it to private requires a paid account)
- Enable 'Add a README file' for documentation purposes.
- Click 'Create repository'

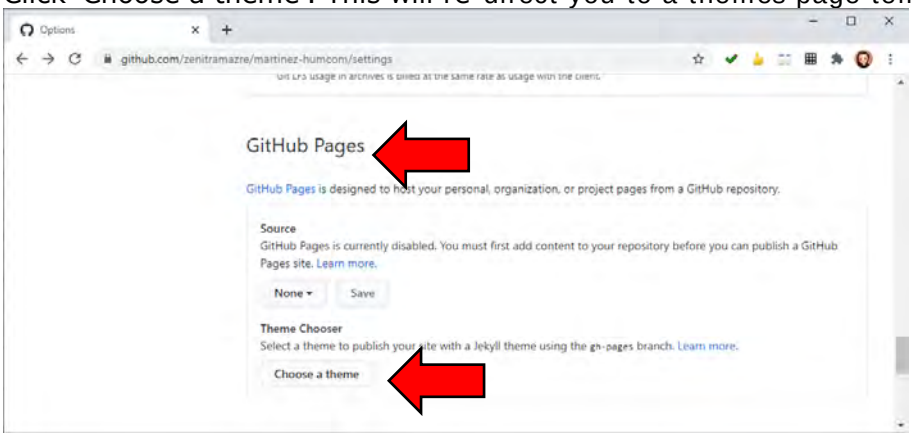


a. Go to the settings of your new repository

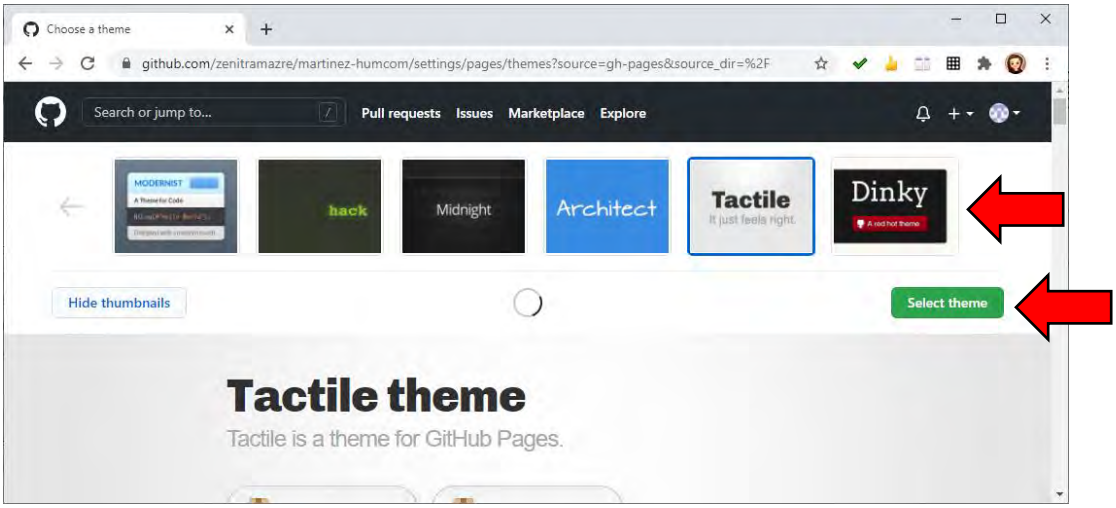


b. Scroll down to GitHub Pages

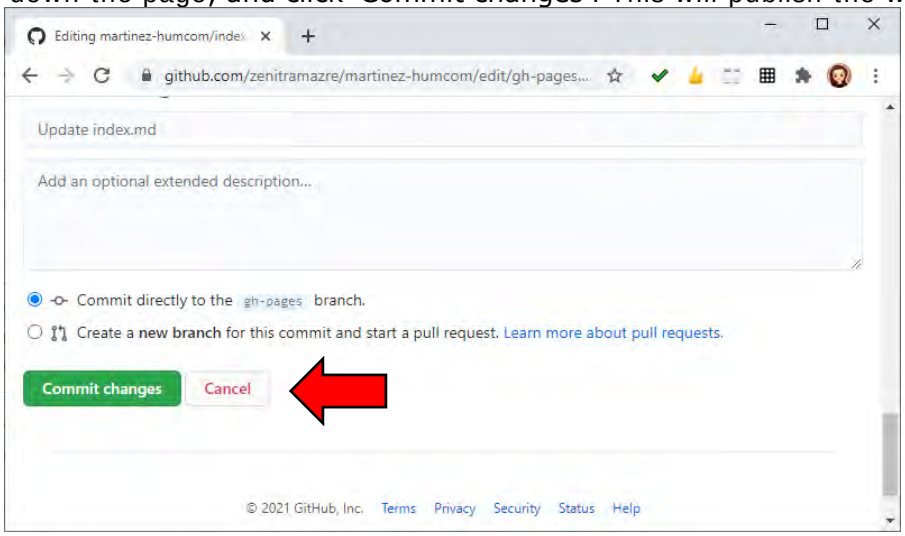
c. Click 'Choose a theme'. This will re-direct you to a themes page template for a homepage.



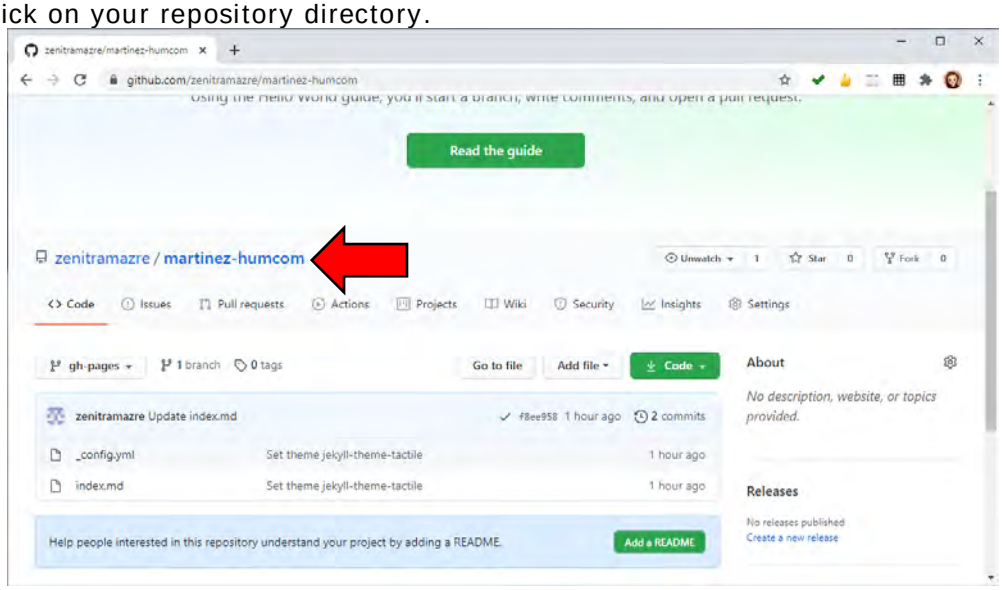
- Choose from any of the templates. Then click on 'Select theme'



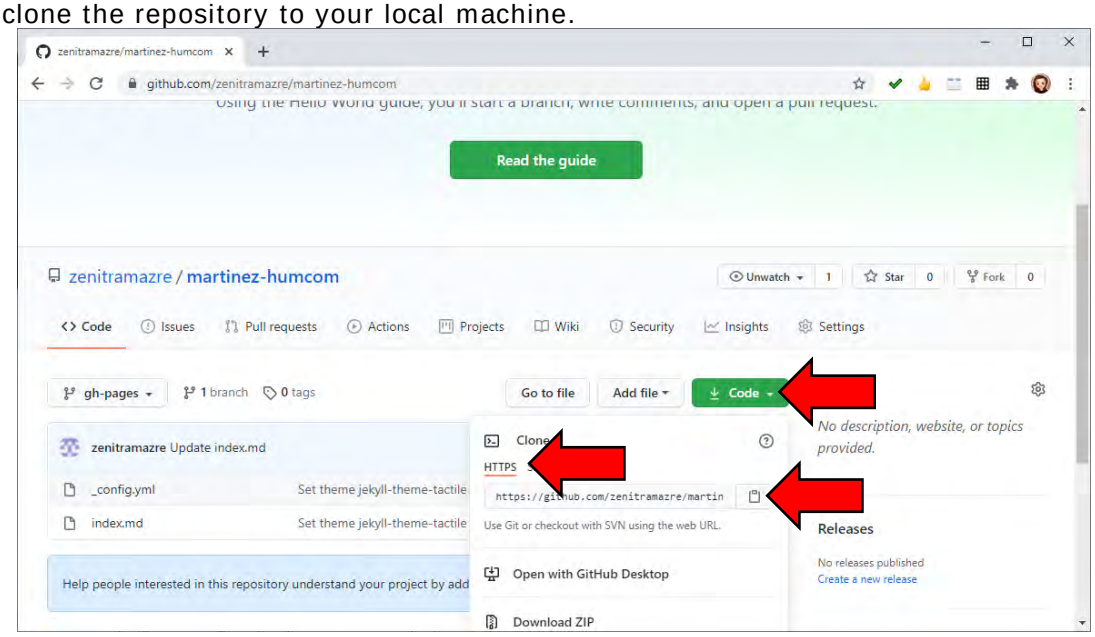
- You may edit this now (or later). If you want. This is not our purpose for now.
- Scroll down the page, and click 'Commit changes'. This will publish the web page.



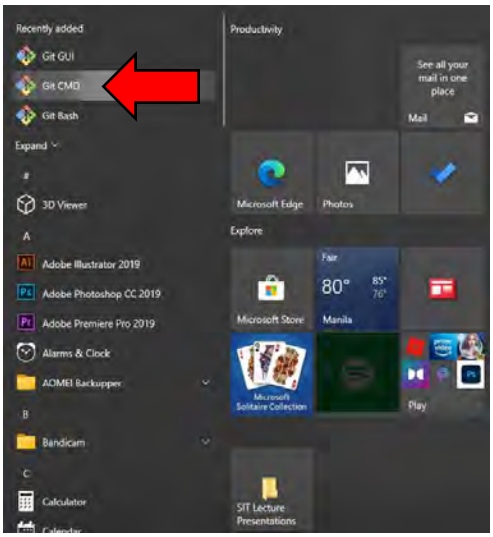
- Click on your repository directory.



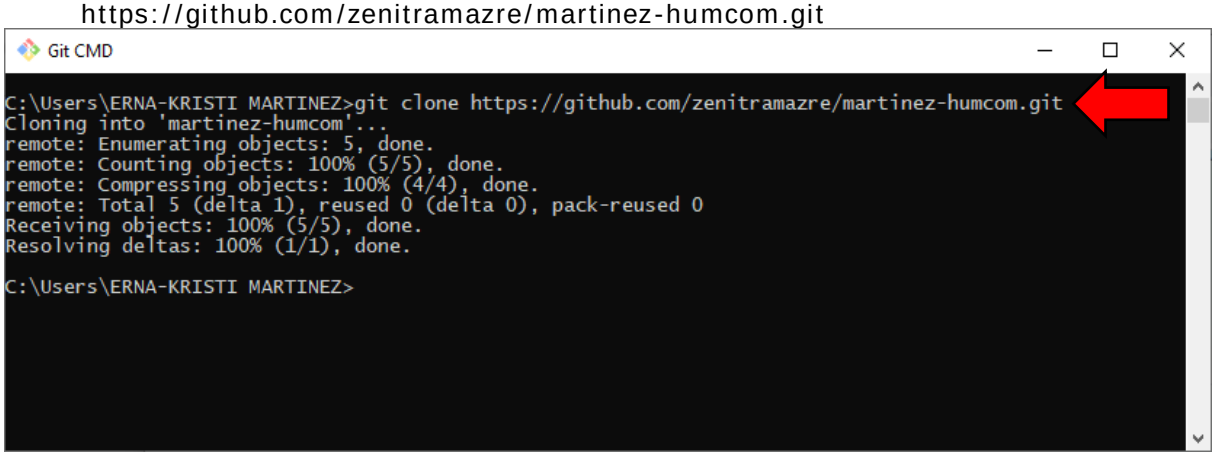
- Then click on the 'Code' button. Select HTTPS. Copy to clipboard. You will need this URL to clone the repository to your local machine.



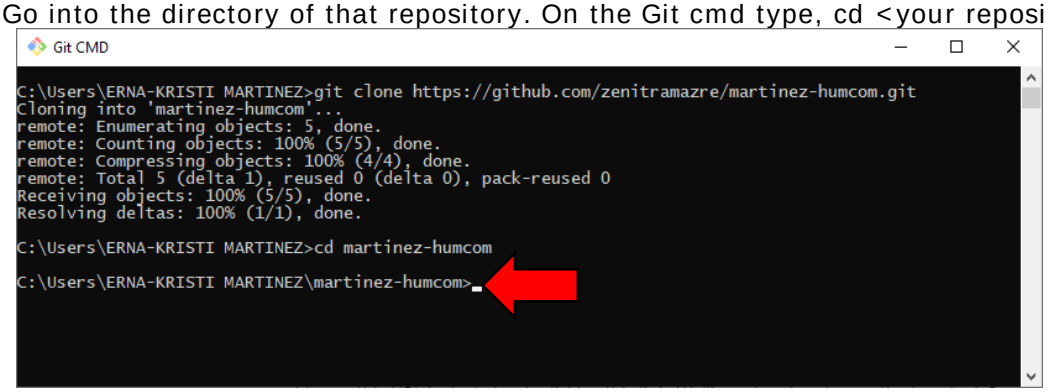
d. Setting up locally – cloning the repository on your pc to make it easier to sync your projects to the web repository.
   - Open the Git cmd from the Windows Programs menu



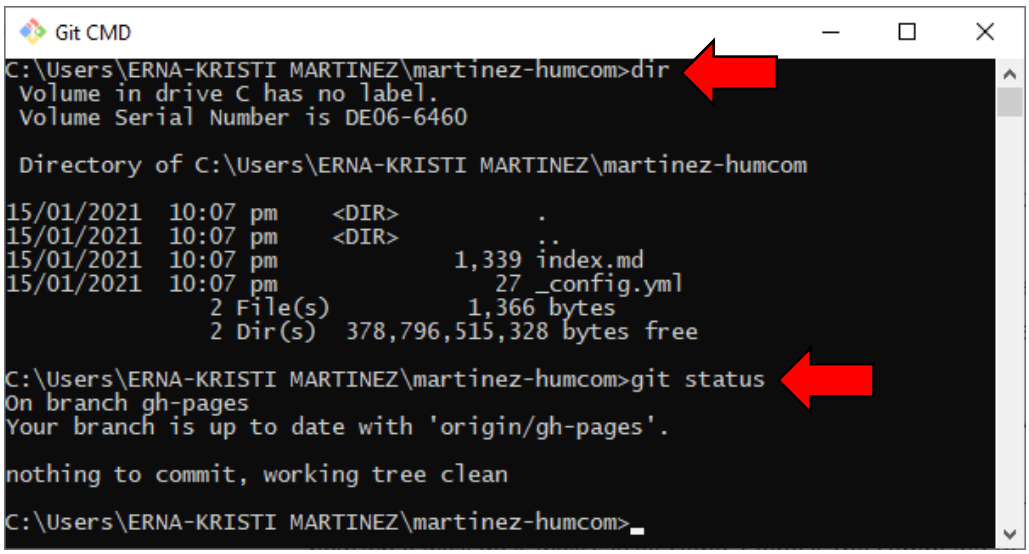
- Once it is open, on the command prompt type, git clone (paste the copied URL) https://github.com/zenitramazre/martinez-humcom.git



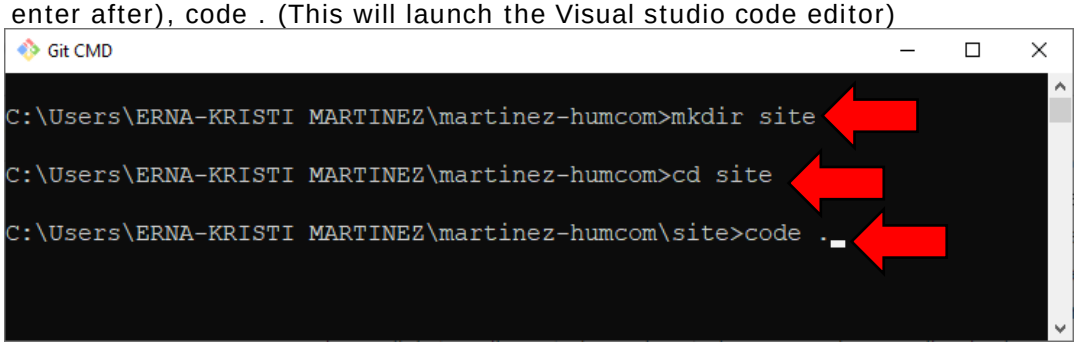
- The repository should be cloned in your local machine.
- Go into the directory of that repository. On the Git cmd type, cd <your repository name>



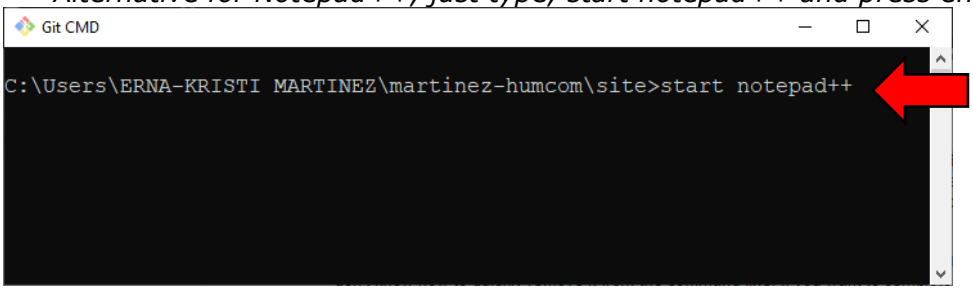
- You may check what is in the repository (or directory) by typing, dir
- Go ahead and check the status by typing, git status



- Create a new folder in the current directory by typing, mkdir site
- Go to that new folder by typing, cd site
- This step is only for Visual studio code users. On the command prompt type (and press enter after), code . (This will launch the Visual studio code editor)

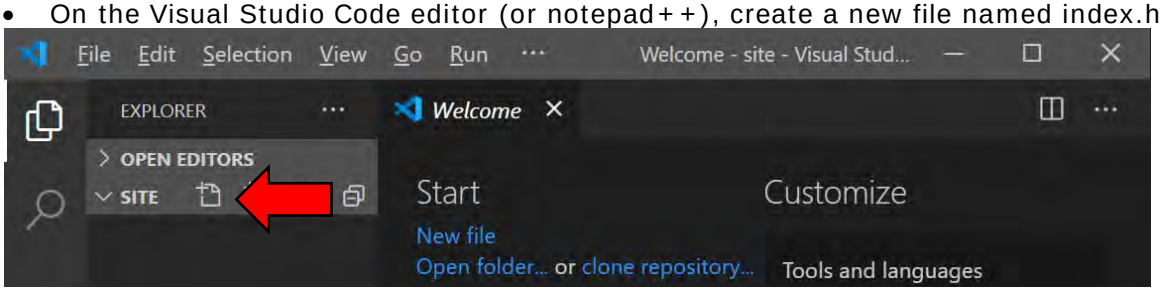


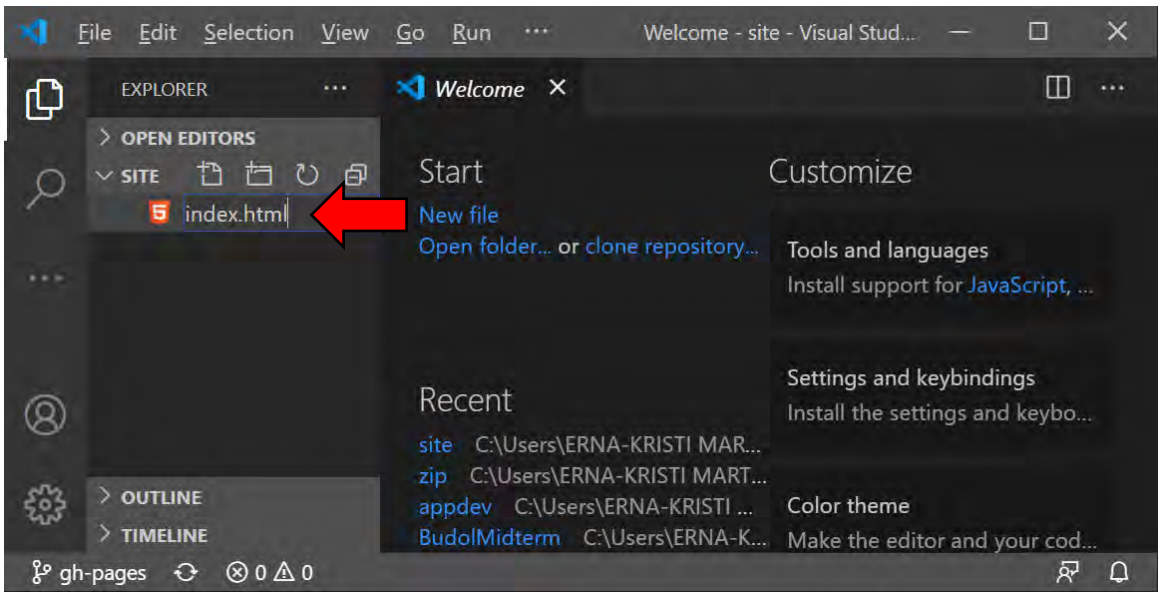
****Alternative for Notepad++, just type, start notepad++ and press enter.*



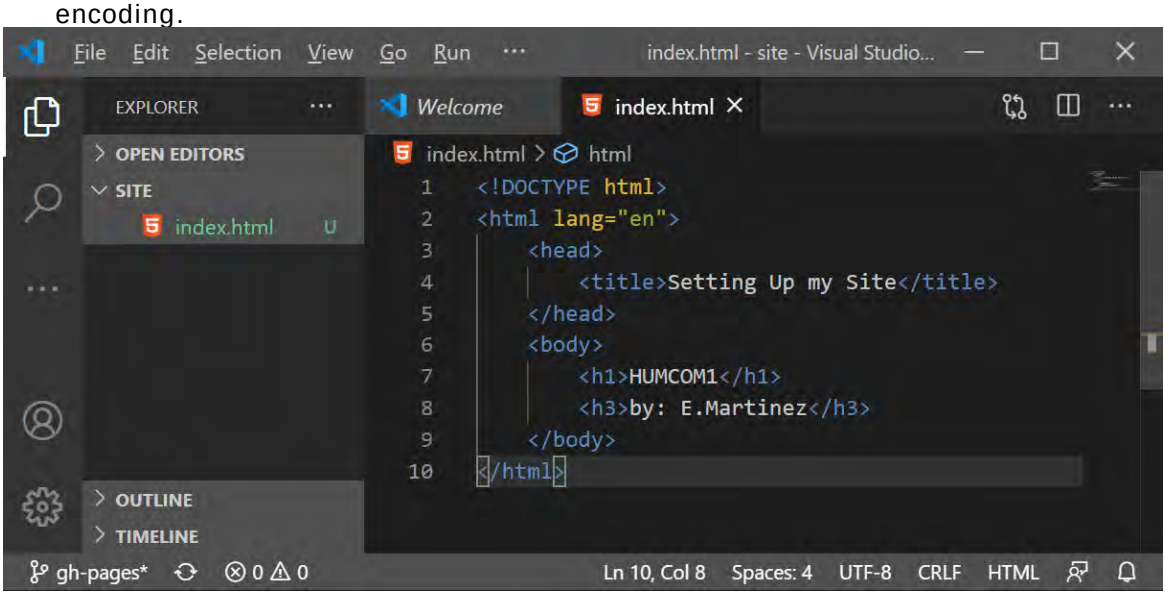
e. GitHub and Browser

- On the Visual Studio Code editor (or notepad++), create a new file named index.html.

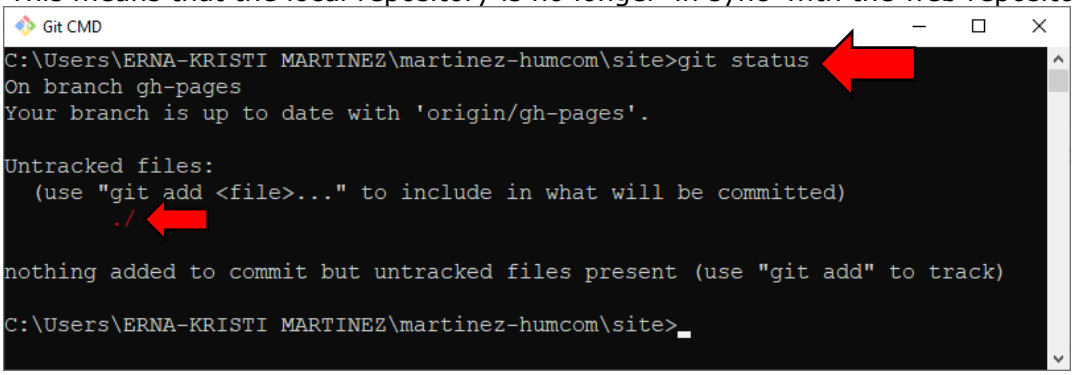




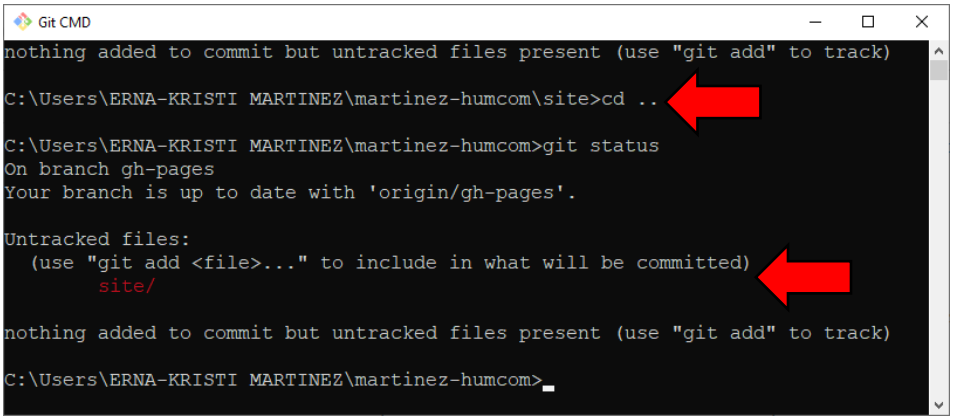
- On the page editor, type the following code for now. The context will be explained in the next lessons. This is for testing purposes for now. Make sure you save the file after encoding.



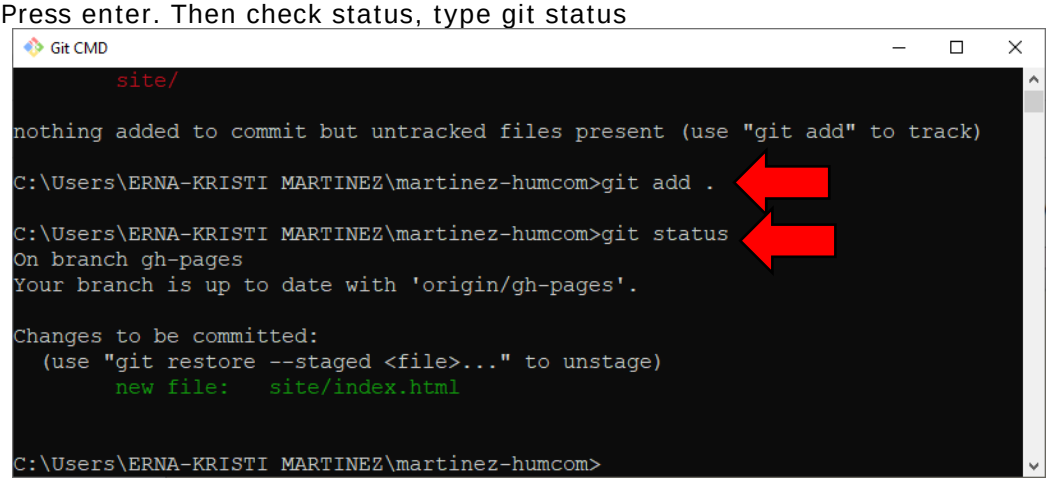
- Go back to your Git cmd and check the repository status. Type: git status
- It will indicate that a new file has been added in the site directory as indicated by the '**./**'. This means that the local repository is no longer 'in sync' with the web repository.



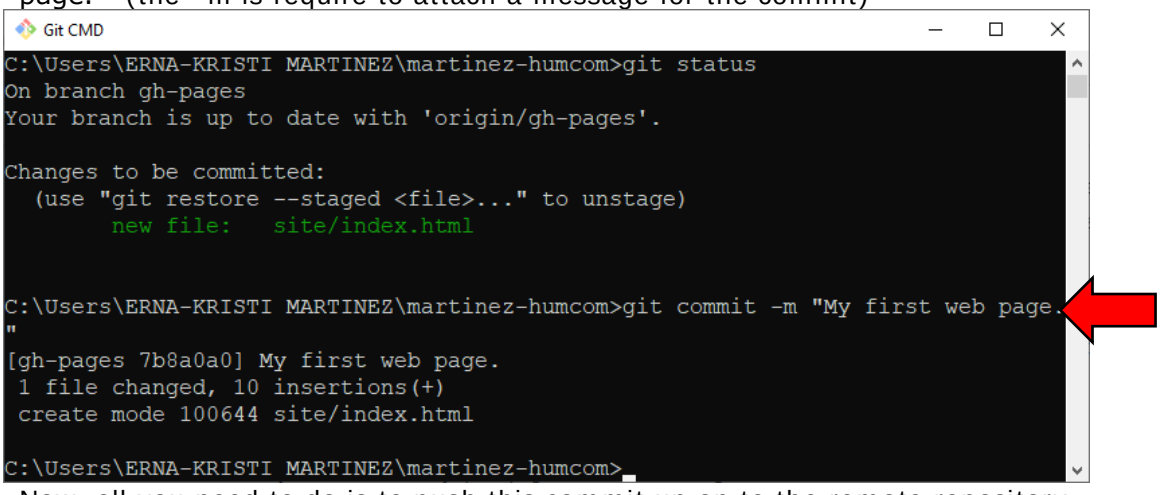
- Go up one directory and check the status again. Type, cd ..



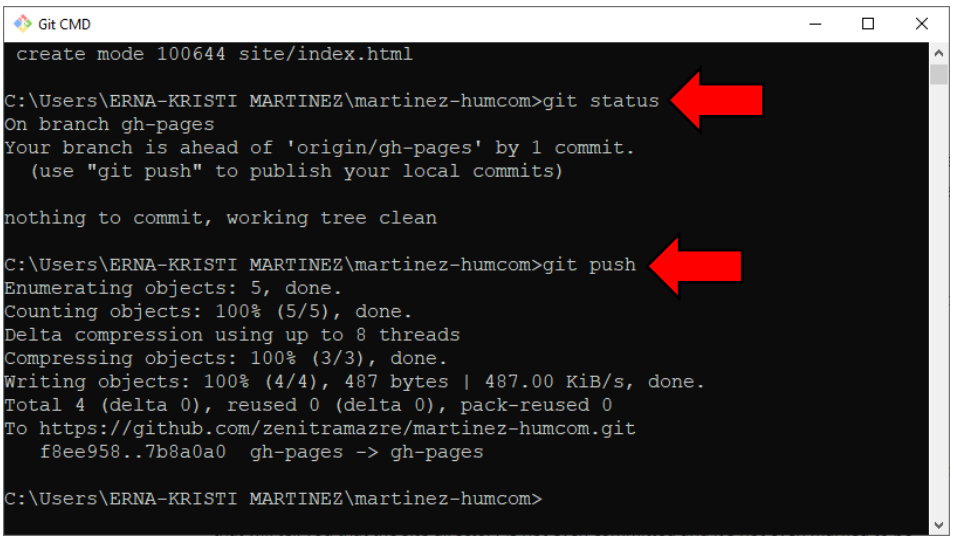
- To sync the local and remote repository, we need to mark something for committing and push the changes in the repository.
- So, on the command line type, git add . (the dot includes everything in the local machine)
- Press enter. Then check status, type git status



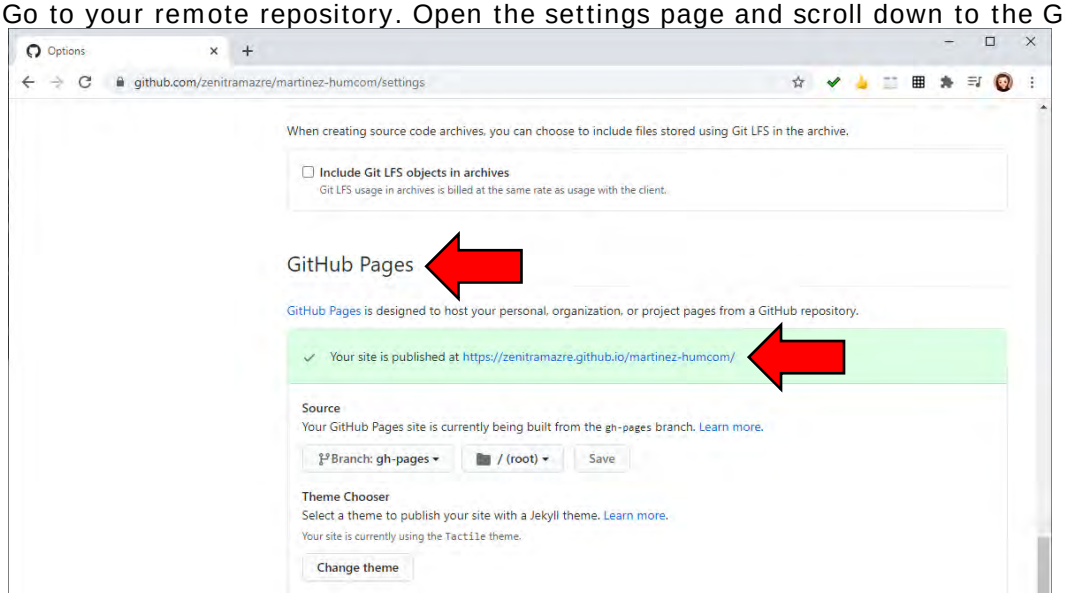
- It is now marked as committed. To commit the new file, type: git commit –m "My first web page." (the –m is require to attach a message for the commit)



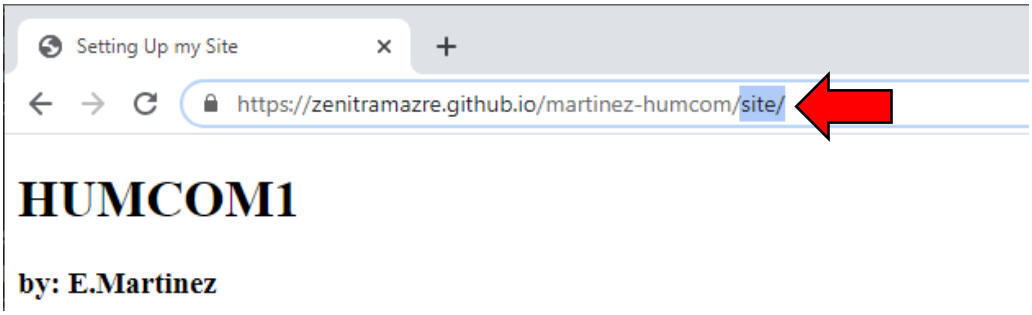
- Now, all you need to do is to push this commit up on to the remote repository.
- Type, git status and it will tell you that the local repo is ahead by 1 commit and to 'push' the new commit to publish. So, we'll go ahead and do that.



- Now the local and remote repo are in sync.
- Go to your remote repository. Open the settings page and scroll down to the GitHub Pages



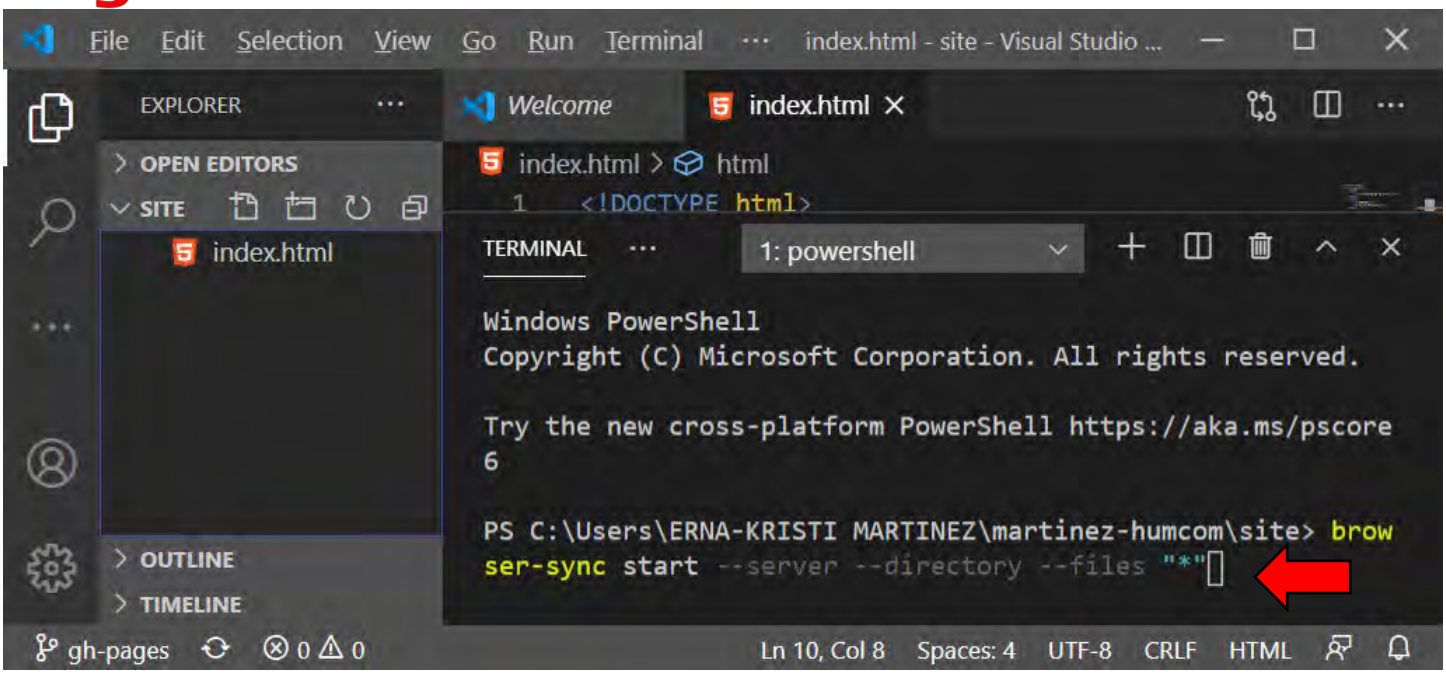
- Click on the highlighted published URL shown above.
- On the address bar of the browser on the new page that is opened. Add 'site/' on the end. (Don't include the apostrophes). This will open the web page we created earlier on the code editor.



- Congratulations! You just published your first web page online.

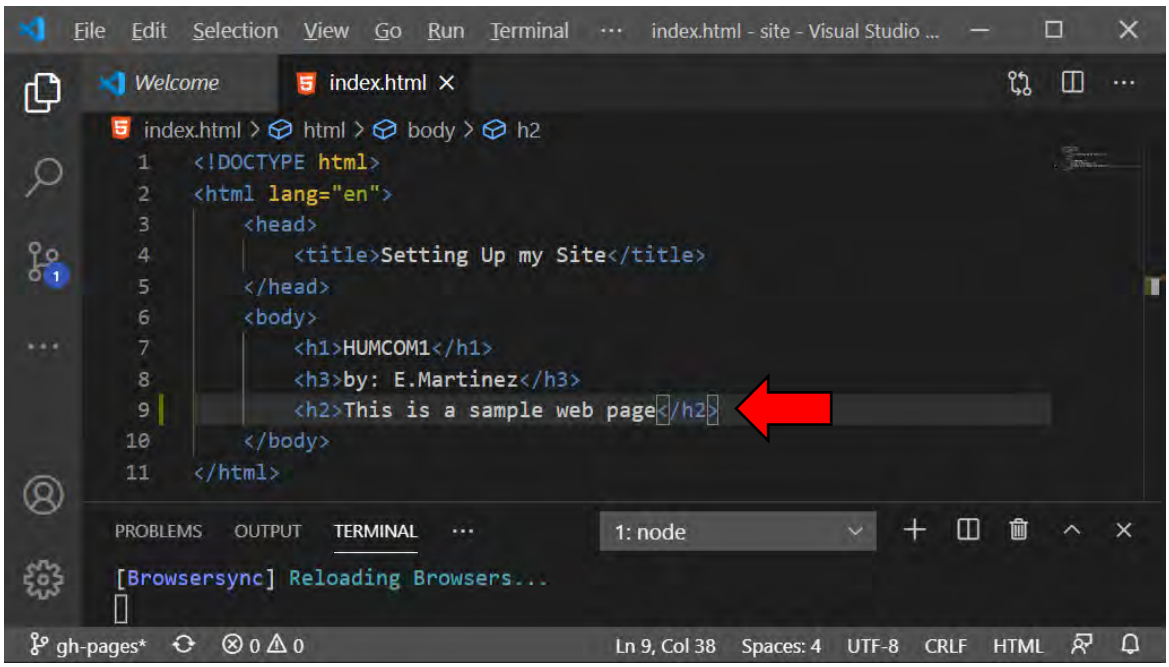
f. Using Browser-sync

- On visual studio code, open the terminal from the menu (or press Ctrl+Shift+` on the keyboard).
  *** or just use the Windows cmd prompt (not the Git CMD)
- The Terminal is Visual studio code's built in command line interface.
- On the terminal type, browser-sync --start --directory --files "*"



- This will launch or open a new web page showing your directory listing (or repository directory) served on a localhost:3000.

- Click on 'index.html' file from the page. While the page is open, go back to your code editor and edit some of the content in the web page as shown below. Observe the web page as you are typing the new line in your code.



- So, browser-sync --start --directory --files "*",
  - --start: starts the browser-sync
  - --directory: automatically syncs your directory, once you add any new files or folders in the directory
  - --files: automatically reloads any changes or modifications on local files

Name:		Date:	
Development Environment Set-up		Lesson 1 **Activity**	Score: (HPS 20points)

1. Signup for GitHub account.
2. Publish your first sample web page as we have done in the lesson.
3. Submit your URL to your instructor.

LESSON 2: Introduction to HTML

Duration: 1 hour

About this lesson: At the end of this lesson you are expected to:

1. Define HTML and discuss some relevant facts on the history and evolution of HTML.
2. Identify define the parts and meanings of the HTML markup, document structure, and content model in order to create web pages more efficiently.

2.1. What is HTML?

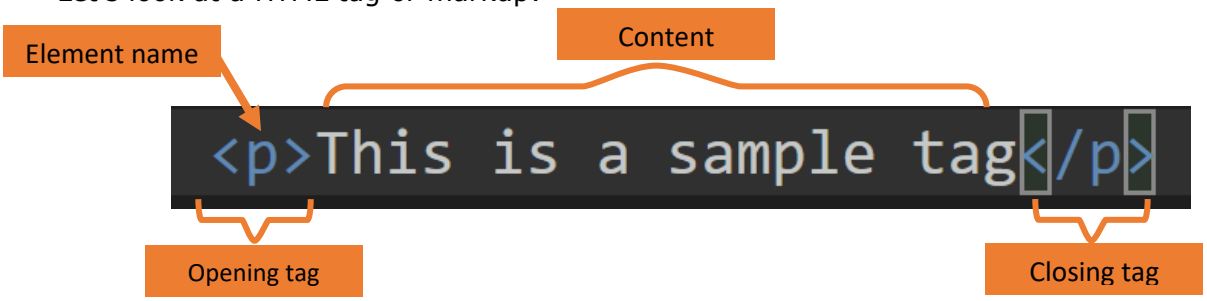
- HTML stands for HyperText Markup Language
- It is the language that has been used to create and structure documents on the web.
- It is plain text that uses a variety of tags that define the structure of the document.

2.2. Relevant History

- 1993 – first version of HTML was written
- Pre 1997 HTML was non-standardized
- 1997 – World Wide Web Consortium (W3C) standardized HTML4.0 and HTML4.1
- 2000 – XHTML 1.0 was standardized
- 2004 – Web Hypertext Application Technology Working Group (WHATWG) introduced HTML5 specification for Web applications
- 2007 – W3C and WHATWG working together, W3C for HTML5 standardization and WHATWG for HTML evolution

2.3. Anatomy of an HTML Tag

- Understanding the components of the HTML tag is important since it is the core of any HTML document.
- Let's look at a HTML tag or markup:



- HTML documents are made up elements, which will be discussed in the next lessons. This elements are always enclosed in angle brackets '<' and '>' and are called tags, or elements.
- Not all elements have closing tags.
- Elements may have 1 or more attributes, which will also be discussed in the next lessons.
- Place between the elements is the content. This is what you see displayed on the browser page.

2.4. Basic HTML Document Structure

- The next important item that we need to understand about the HTML document is the structure. The HTML document will be interpreted by the browser. Therefore, it is important that it understood by the browser. We can do that by correctly structuring our HTML document.
- Let's look at a HTML document.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <meta name="author" content="E.Martinez" />
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>Setting Up my Site</title>
8   </head>
```

```
9   <body>
10      <h1>HUMCOM1</h1>
11     <h3>by: E.Martinez</h3>
12      <p>This is a sample tag</p>
13   </body>
14 </html>
```

- Line 1 <!DOCTYPE html>
  - Every HTML document start with a document declaration. Our document uses the HTML5 document type declaration. (HTML uses a different document declaration)
  - This tells the browser the type and version of HTML used in building the document. Web browsers also use it to determine the rendering mode to use.
- Line 2 <html> and Line 14 </html>
  - The <html> tag denotes the start of the of HTML document
  - A closing </html> tag is placed at line 14 which denotes the END of the document.
- Line 3 and Line 8 <head> … </<head>
  - The <head> ..</head> tags contains information about the web page.
  - Character coding, author description, page description, keywords, viewport size and scale can be added here using the <meta> tag (line 4-5) for the metadata
  - A page <title>..</title> is also added in the <head> of the HTML document. This appears on the browsers title bar.
- Line 9 <body> and line 13 </body>
  - The <body> tag is the root of all the content that is visible to the user. We refer to the visible content as the *viewport.*
  - All visible content should be placed between the <body> and </body> tags.
  - Always close the body.
  - The <body> always comes after the <head></head>
- Line 10 to line 12
  - These are the page elements.
  - We have different types of page elements, representing different types of content.
  - In the example used, <h1> and <h3> are headings, <p> is a paragraph.
  - There are more elements that will be discussed later.
  - You will be using other types of elements to structure and organize content in your pages better.
  - Other elements you may use are tables, lists, forms, and others.

2.5. General Guidelines

- HTML is NOT case sensitive, but it helps if you make your code consistent throughout your entire project to make it easier to read and debug.
- Excess white spaces (including spaces, new lines, tabs, carriage returns) is generally ignored. It is encouraged to use the extra space (indents for each line) in your code to improve readability, without affecting the output on the browser.
- Save your document as .htm or .html file.
- Element attributes may be enclosed in either single or double quotes, just make sure that you use a matching pair and use it consistently in your document.
- When nesting html tags, close the last opened tag before you close its parent tag.

2.6. HTML Content Models

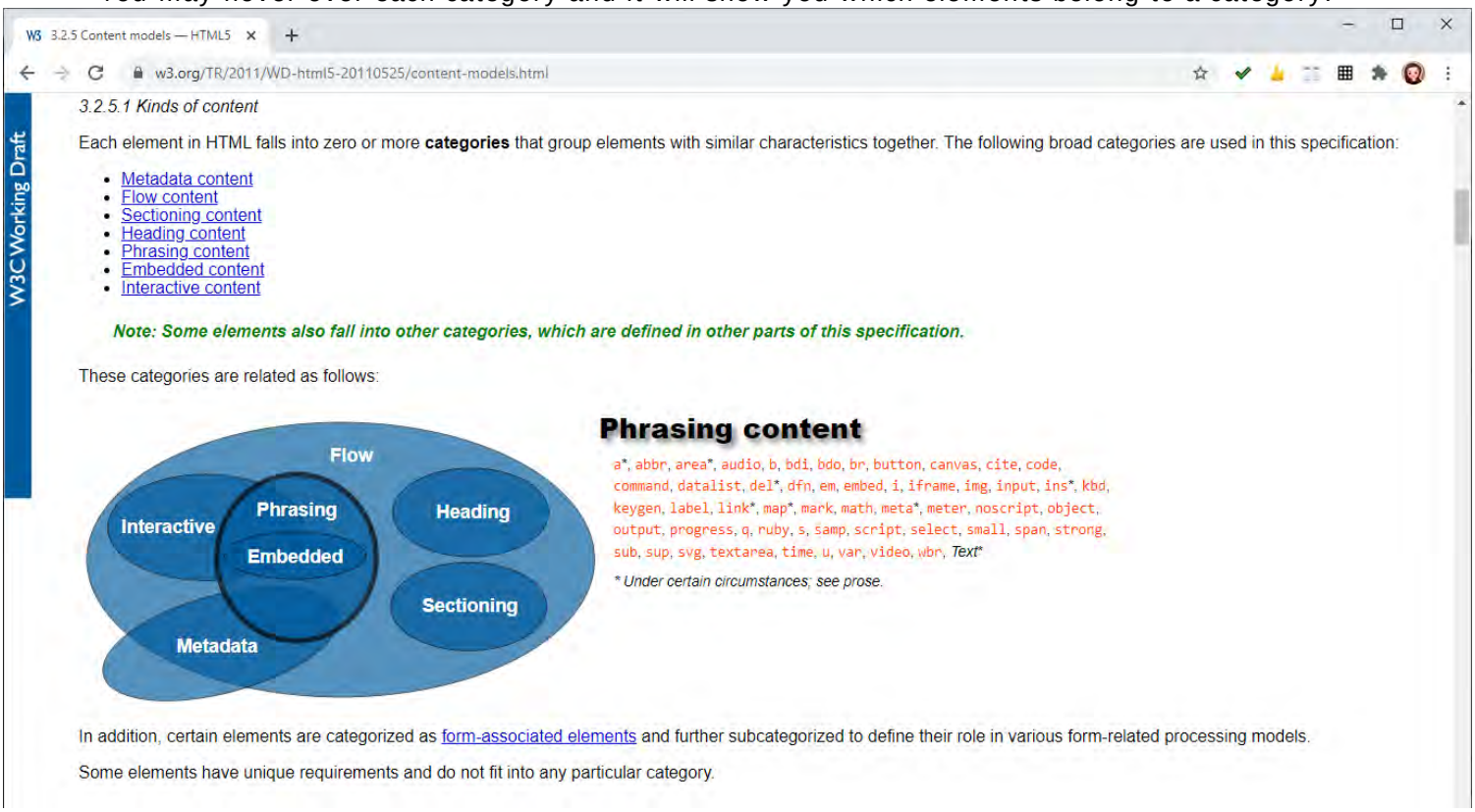
- Content model refers to the set of rules that define what type of content each element is allowed to have. This translates into what other elements are allowed to be nested inside which other elements.
- Traditionally (HTML4.1), elements can either be:
  - Block-level elements – render to begin on a new line (by default); may contain inline or other block-level elements
  - Inline Elements – render on the same line (by default); may contain other inline elements
- Example

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="author" content="E.Martinez" />
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <title>Div and Span Elements</title>
  </head>
  <body>
    <h1>Div and Span Elements</h1>
    <div>*** DIV 1: Some content here *** </div>
    <div>*** DIV 2: Following right after div 1 ***</div>

    <span>*** SPAN 1: Following right after div 2***</span>
    <div>
      *** DIV 3: Following right after span 1
        <span>*** SPAN 2: INSIDE div 3 ***</span>
        Continue content of div 3 ***
    </div>
  </body>
</html>
```

- Output

  - element stands for division. This is the most generic block level element. As seen in the output, starts or pushes a new line for every new   block
  - element stands for span. This is another generic inline element. In the output, although span 1 started a new line, it was because div 2 pushed span 1 to the next line. But if we look at span 2, it is nested in div 3. Even if we removed the white spaces in our code, the output will remain the same.

- HTML5 has replace these two with 7 sets of content categories which is roughly equivalent to flow content (block-level) and phrasing content (inline).
- To check out this 7 content categories and which elements belong to this categories, go to https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html
- You may hover over each category and it will show you which elements belong to a category.



UNIVERSITY OF Baguio

LESSON 3: HTML Elements Part 1

Duration: 1 hour

About this lesson: At the end of this lesson you are expected to:

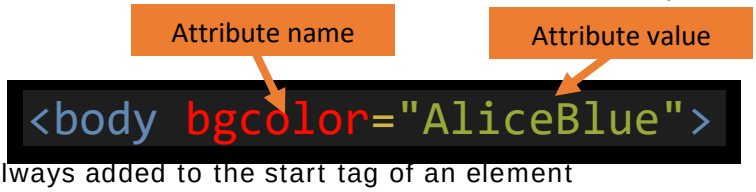
1. Define basic HTML elements and functions.
2. Use the html headings, paragraph, and lists to structure webpage content.

3.1. HTML elements

- HTML elements or tags may represent headings, paragraphs, hypertext links, lists, embedded media, and a variety of other HTML5 semantic elements.
- Let's start with some basic HTML elements.

3.1.1.Element attributes

- Elements can be basic or could be defined with attributes. Elements may have more than 1 attribute.
- Attributes are added inside the element as attribute name/value pairs



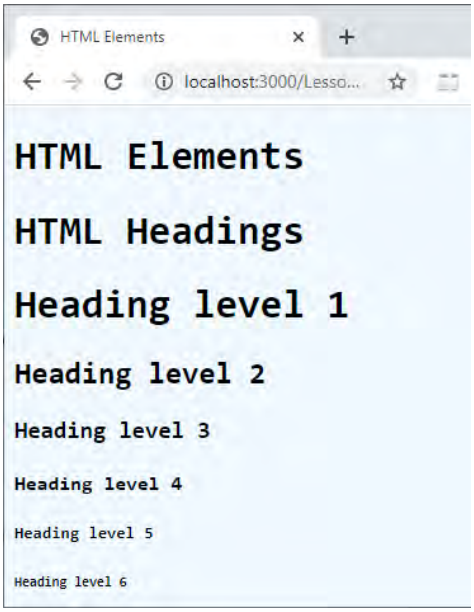
- Attributes are always added to the start tag of an element
- Attribute values are either enclosed in single or double quotes

3.1.2.HTML Headings

- Are used to describe the topic of the section you want to introduce.
- Headings have six levels with H1 as the most important (biggest in sizes) and H6 as the least (smallest).
- <h1> elements defines the level 1 heading. It has a start tag <h1> and an end tag </h2>. This is the same for the next 5 levels, <h5> to <h6>
- Headings are block-level or flow content elements.
- Example (headings.html)

```
<h1>HTML Elements</h1>

<h1>HTML Headings</h1>
<h1>Heading level 1</h1>
<h2>Heading level 2</h2>
<h3>Heading level 3</h3>
<h4>Heading level 4</h4>
<h5>Heading level 5</h5>
<h6>Heading level 6</h6>
```

- Output



3.1.3.HTML Paragraphs

- Are used to structure paragraph content in a web page using the opening tag and closing tag.
- Paragraph, like div and heading, is a block-level or flow content element.
- are usually represented in visual media by blocks of text that are physically separated from adjacent blocks through blank lines
- Example

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="author" content="E.Martinez" />
        <meta name="viewport" content="width=device-width,
        initial-scale=1"/>
        <title>HTML Elements</title>
    </head>
    <body bgcolor="AliceBlue">
        <h1>HTML Elements</h1>
        <h1>Heading level 1</h1>
        <p>The little kitten gently seated himself on a piece of
            carpet. Later in his life, this would be referred to as
            the time the
            cat sat on the mat.</p>
    </body>
</html>
```

- Output

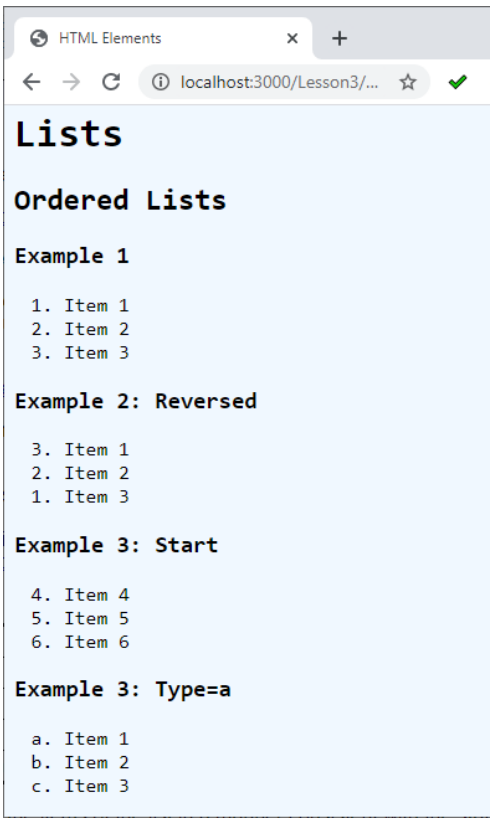
3.1.4.HTML Lists

- Are used to group related pieces of information together, so that they are clearly associated with each other and easy to read.
- Lists are good from a structural point of view as they help create a well structures, more accessible, easy to-to-maintain document.
- Lists can be nested.
- There are two types of lists that can be used in HTML, the ordered and unordered list.

3.1.4.1.Ordered lists

- Defines an ordered list of items, where the items have been intentionally ordered.
- The list is created using the … tags
- The items in the list are the … element child nodes of the <ol> element
- The <ol> has several attributes:
  - reversed – if present, it indicates that the list is a descending list
  - start – if present, must be a valid integer giving the ordinal value of the first item.
  - type – can be used to specify the kind of maker to use in the list. Assignable type are 1, a, A, i, and I
- Example (ordered-list) Output

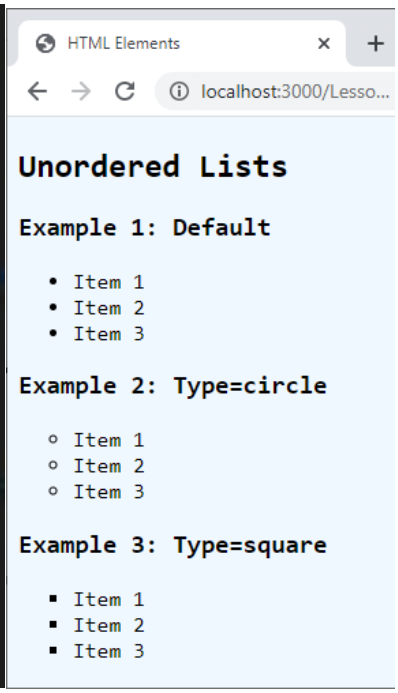
```
<h1>Lists</h1>
    <h2>Ordered Lists</h2>
        <h3>Example 1</h3>
            <ol>
                <li>Item 1</li>
                <li>Item 2</li>
                <li>Item 3</li>
            </ol>
        <h3>Example 2: Reversed</h3>
            <ol reversed>
                <li>Item 1</li>
                <li>Item 2</li>
                <li>Item 3</li>
            </ol>
        <h3>Example 3: Start</h3>
            <ol start=4>
                <li>Item 4</li>
                <li>Item 5</li>
                <li>Item 6</li>
            </ol>
        <h3>Example 3: Type=a</h3>
            <ol type='a'>
                <li>Item 1</li>
                <li>Item 2</li>
                <li>Item 3</li>
            </ol>
```



3.1.4.2.Unordered lists

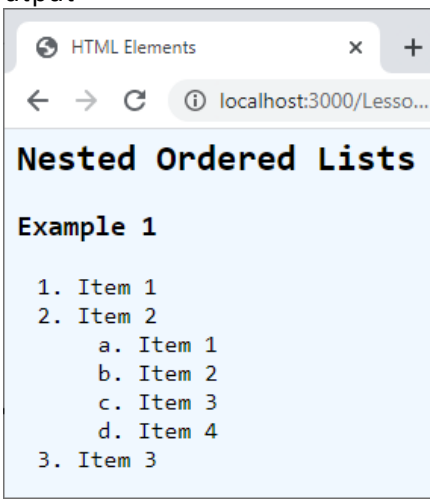
- The <ul> element represents a list of items, where the order of the elements is not important.
  - The list is created using the … tags
  - The items in the list are the … element child nodes of the <ol> element
  - The <ul> has one attribute:
    - type - can be used to specify the kind of maker to use in the list. Assignable values are circle, disc, and square
  - Example (unordered-lists) Output

```
<h2>Unordered Lists</h2>
    <h3>Example 1: Default</h3>
        <ul>
            <li>Item 1</li>
            <li>Item 2</li>
            <li>Item 3</li>
        </ul>
    <h3>Example 2: Type=circle</h3>
        <ul type='circle'>
            <li>Item 1</li>
            <li>Item 2</li>
            <li>Item 3</li>
        </ul>
    <h3>Example 3: Type=square</h3>
        <ul type='square'>
            <li>Item 1</li>
            <li>Item 2</li>
            <li>Item 3</li>
        </ul>
```



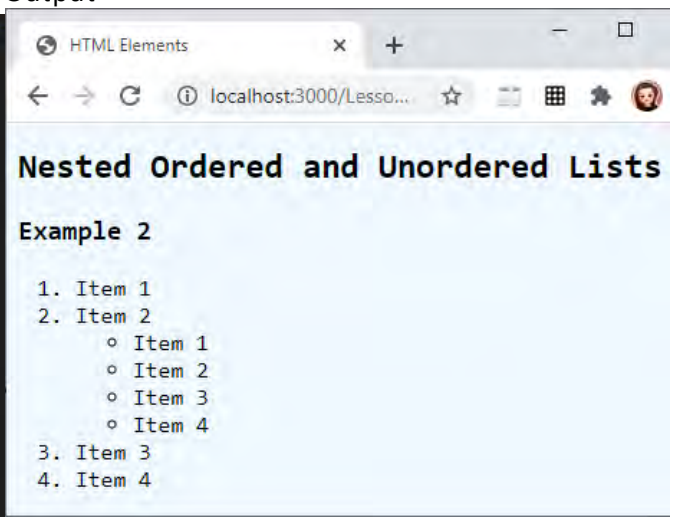
- Example(nested ordered-list) Output

```
<h2>Nested Ordered Lists</h2>
    <h3>Example 1</h3>
        <ol>
            <li>Item 1</li>
            <li>Item 2</li>
                <ol type='a'>
                    <li>Item 1</li>
                    <li>Item 2</li>
                    <li>Item 3</li>
                    <li>Item 4</li>
                </ol>
            <li>Item 3</li>
        </ol>
```



- Example(nested ordered-list) Output

```
<h2>Nested Ordered and Unordered Lists</h2>
    <h3>Example 2</h3>
        <ol>
            <li>Item 1</li>
            <li>Item 2</li>
                <ul >
                    <li>Item 1</li>
                    <li>Item 2</li>
                    <li>Item 3</li>
                    <li>Item 4</li>
                </ul>
            <li>Item 3</li>
            <li>Item 4</li>
        </ol>
```



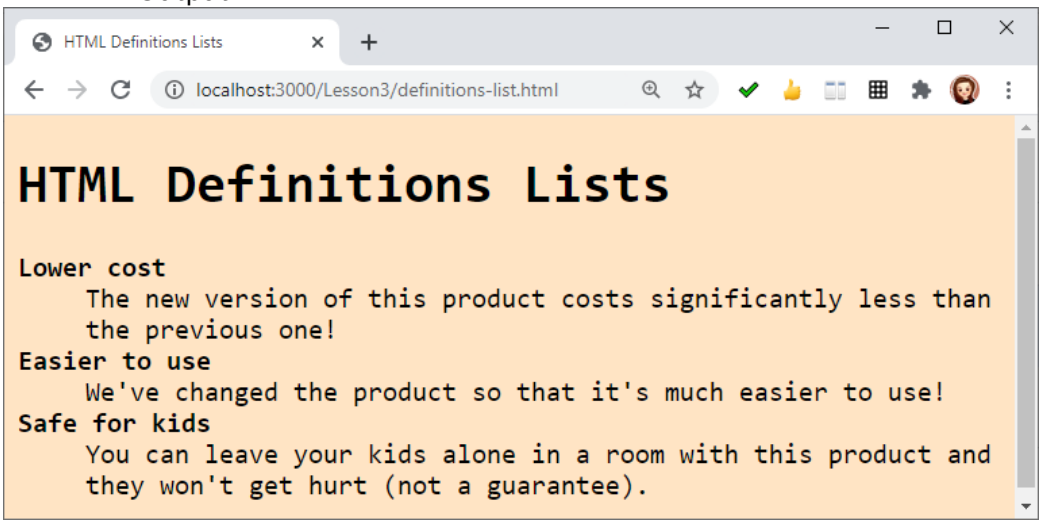
3.1.1.1. Definitions lists

- Definition lists, created using the DL element, generally consist of a series of term/definition pairs (although definition lists may have other applications). Thus, when advertising a product, one might use a definition list.

- Example

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="author" content="E.Martinez" />
        <meta name="viewport" content="width=device-width, initial-scale=1"/>
        <title>HTML Definitions Lists</title>
    </head>
    <body bgcolor="bisque">
            <h1>HTML Definitions Lists</h1>
        <dl>
            <dt><STRONG>Lower cost</STRONG></dt>
            <dd>The new version of this product costs significantly less than the previous one!</dd>
            <dt><STRONG>Easier to use</STRONG></dt>
            <dd>We've changed the product so that it's much easier to use! </dd>
            <dt><STRONG>Safe for kids</STRONG></dt>
            <dd>You can leave your kids alone in a room with this product and they won't get hurt (not a
            guarantee). </dd>
        </dl>
    </body>
</html>
```

- Output



3.1.2. Formatting tags

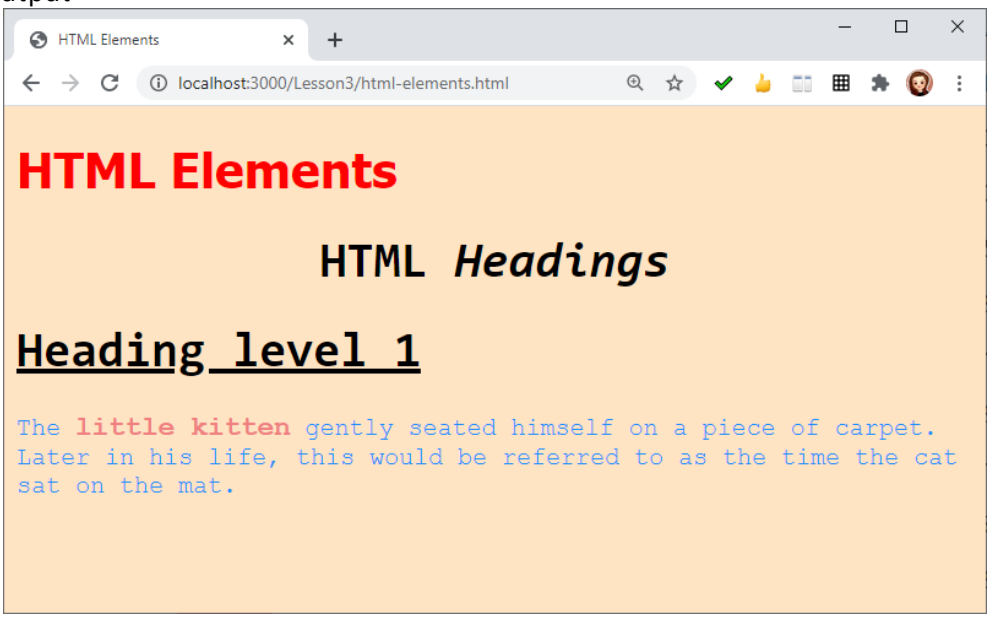
- Aside from attribute there are also tags that you can use to format html elements. This are formatting tags, which used to format the appearance of some elements on your web page.
- Most of this are no longer used since we already have CSS to do the page and content styling for us.
- Examples of these tags are:
  - Italic *italic*
  - Bold **bold**
  - Underline <u>underline</u>
  - Center

- Font <font face=’‘
  size=’‘
  color=’‘ >

- Example of formatted context

```
Lesson3 > html-elements.html > html > body > font
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="utf-8" />
5          <meta name="author" content="E.Martinez" />
6          <meta name="viewport" content="width=device-width, initial-scale=1"/>
7          <title>HTML Elements</title>
8      </head>
9      <body bgcolor="bisque">
10         <font face='Tahoma' size='3pt' color='red'>
11             <h1>HTML Elements</h1>
12         </font>
13         <center><h1>HTML <i>Headings</i></h1></center>
14         <h1><u>Heading level 1</u></h1>
15
16         <font face='Courier New' size='3pt' color='#1E90FF'>
17             <p>The <font color='#F08080' size='+1'><b>little kitten</b></font> gently
18             seated himself on a piece of carpet. Later in his life, this would be
19             referred to as the time the cat sat on the mat.</p>
20         </font>
21     </body>
22 </html>
pages*   ⊗ 0 ⚠ 0                                   Ln 10, Col 44   Spaces:
```

- Output



UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements Part1:Creating your first web page		Lesson 2-3 **Exercise1-2**	Score: 25

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.
- Save First grading exercises in the FirstGrading directory.

Objectives

- Use Basic HTML structure
- Use Basic HTML formatting tags/mark-up
- Organize a webpage content using HTML Lists

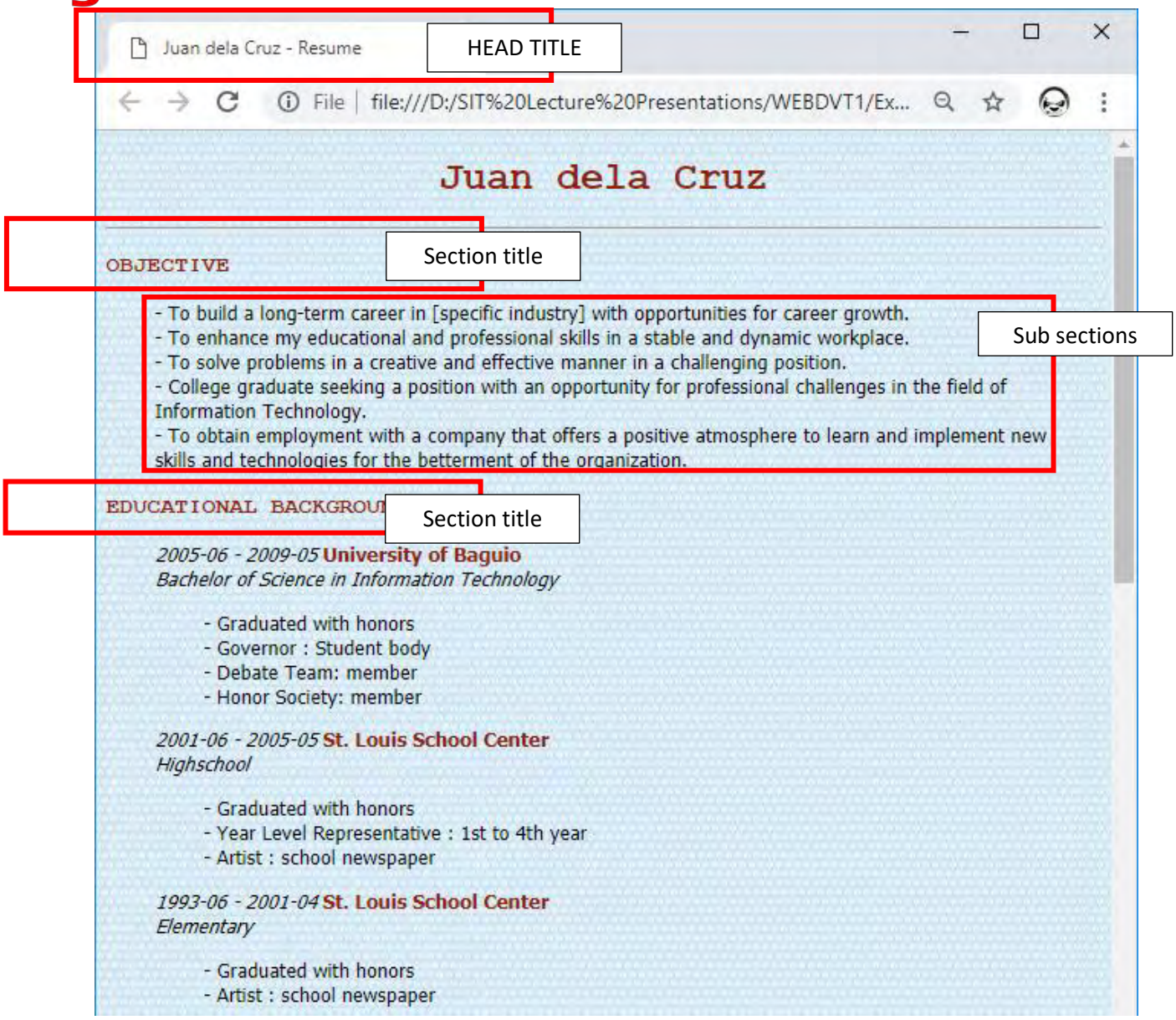
Exercise1 (Exercise1.html) 25 points

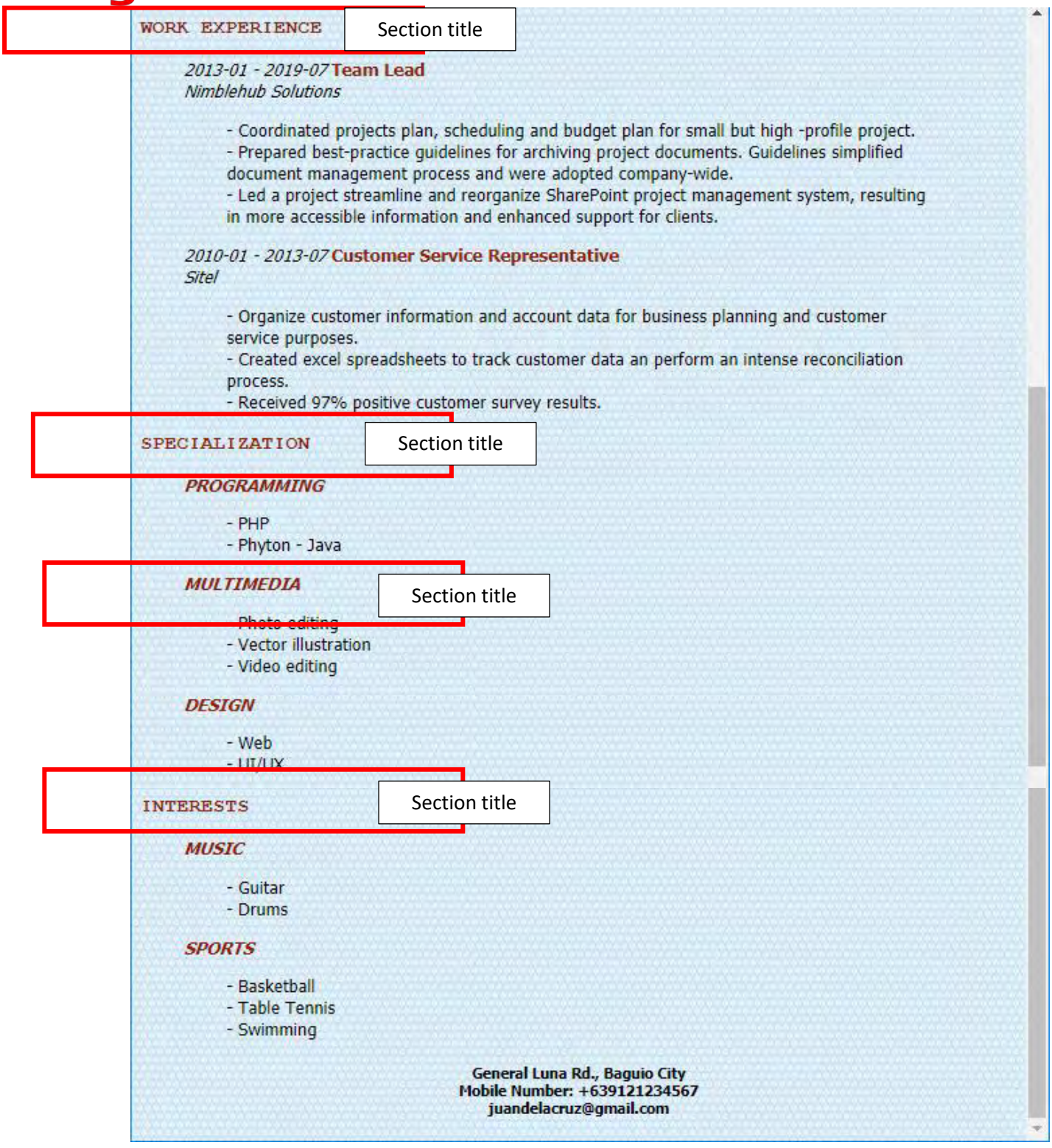
- Print a paragraph that is a description of your favorite book, include the title of the book as well as its author. Names and titles should be underlined, adjectives should be in red, italicized and bolded.

Exercise2 (Exercise2.html) 25 points

Instructions:

1. Create a new web page document that will display your resume on a web page. Refer to the sample output Figure E1. Observe the format of the document
   e.g. bold, italics, lines, etc. and recreate this in your document.
2. Include meta tag containing your name as the page author, content, and keywords to describe the content of your web page.
3. Font.
   a. Document Body: Tahoma, 2pt.
   b. Section titles (name and headings): Courier New
      - Name = h1
      - Headings = h3
      - Color = any color except black
4. Background Color:
   a. Assign a background color or image for the webpage.
   b. Background color or image should be pastel or anything subtle.
5. Do not copy the content from the example, only the format.
6. The title of the webpage should be Your Name + Resume e.g: Juan Sarmiento Resume


Juan dela Cruz - Resume
HEAD TITLE
File | file:///D:/SIT%20Lecture%20Presentations/WEBDVT1/Ex...
Juan dela Cruz
OBJECTIVE
Section title
- To build a long-term career in [specific industry] with opportunities for career growth.
- To enhance my educational and professional skills in a stable and dynamic workplace.
- To solve problems in a creative and effective manner in a challenging position.
- College graduate seeking a position with an opportunity for professional challenges in the field of Information Technology.
- To obtain employment with a company that offers a positive atmosphere to learn and implement new skills and technologies for the betterment of the organization.
Sub sections
EDUCATIONAL BACKGROU
Section title
2005-06 - 2009-05 University of Baguio
Bachelor of Science in Information Technology
- Graduated with honors
- Governor : Student body
- Debate Team: member
- Honor Society: member
2001-06 - 2005-05 St. Louis School Center
Highschool
- Graduated with honors
- Year Level Representative : 1st to 4th year
- Artist : school newspaper
1993-06 - 2001-04 St. Louis School Center
Elementary
- Graduated with honors
- Artist : school newspaper


WORK EXPERIENCE
Section title
2013-01 - 2019-07 Team Lead
Nimblehub Solutions
- Coordinated projects plan, scheduling and budget plan for small but high -profile project.
- Prepared best-practice guidelines for archiving project documents. Guidelines simplified document management process and were adopted company-wide.
- Led a project streamline and reorganize SharePoint project management system, resulting in more accessible information and enhanced support for clients.
2010-01 - 2013-07 Customer Service Representative
Sitel
- Organize customer information and account data for business planning and customer service purposes.
- Created excel spreadsheets to track customer data an perform an intense reconciliation process.
- Received 97% positive customer survey results.
SPECIALIZATION
Section title
PROGRAMMING
- PHP
- Phyton - Java
MULTIMEDIA
Section title
- Photo editing
- Vector illustration
- Video editing
DESIGN
- Web
- UI/UX
INTERESTS
Section title
MUSIC
- Guitar
- Drums
SPORTS
- Basketball
- Table Tennis
- Swimming
General Luna Rd., Baguio City
Mobile Number: +639121234567
juandelacruz@gmail.com

UNIVERSITY OF Baguio

Grading Rubric

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
	The background, colors and layout are artful and consistent across the website and enhance the readability of the information presented.	The background, colors and layout are consistent across the website and make it easy to read the information presented.	The background, colors and layout are distracting and make it difficult to read the information presented	The background, colors and layout make the site unattractive, and it is difficult to read the information presented.	

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements Part1 : How to <_____>		Lesson 2-3 **Exercise3**	Score:

Create a new 'how to' web page using lists.

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.
- Save First grading exercises in the FirstGrading directory.

Exercise3.html

Instructions.

1. Create a web page that teaches or instructs the viewer to do a task through a lists of instructions.
Examples:
    - How to assemble a computer safely.
    - How to cook Filipino style spaghetti.
    - How to groom your pet.
2. The instructions should be clear and specific.
3. Include all the required materials or tools needed to accomplish the task.
4. Use ordered and/or unordered lists, a combination of, and or nested html lists.
5. Do not just copy and paste content from other websites. Be original.
6. Format the content as you deem necessary to make it easier to read.

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	**4**	**3**	**0-2**	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Originality	Shows original thought	Has some originality	Is simple and shows little creative effort	shows no creative effort	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
	The background, colors and layout are artful and consistent across the website and enhance the readability of the information presented.	The background, colors and layout are consistent across the website and make it easy to read the information presented.	The background, colors and layout are distracting and make it difficult to read the information presented	The background, colors and layout make the site unattractive, and it is difficult to read the information presented.	

UNIVERSITY OF Baguio

LESSON 4: HTML Semantic Elements

Duration: 3 hours

About this lesson: At the end of this lesson you are expected to:

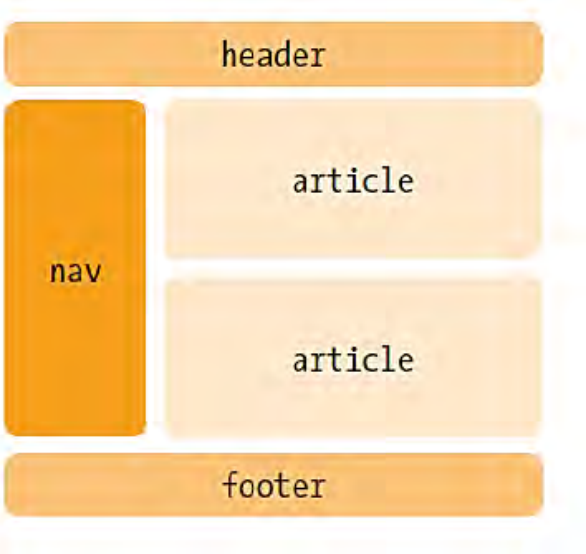
1. Define Semantic Elements.
2. Use semantic elements in web pages.

4.1. HTML Semantic Elements

- Semantic elements = elements with a meaning.
- Semantic HTML elements are those that clearly describe their meaning in a human- and machine-readable way. (freeCodeCamp.org, 2020)

4.2. List of Semantic Elements

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



- Elements such as <header>, <nav>, <section>, <article>, <aside>, and <footer> act more or less like elements.
- They group other elements together into page sections. However where a tag could contain any type of information, it is easy to identify what sort of information would go in a semantic <header> region.

Example: Semantic Vs. Non-Semantic Web

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="author" content="E.Martinez" />
        <meta name="viewport" content="width=device
        <title>Semantic Webpage Elements</title>
    </head>
    <body>
    <header></header>
    <section>
        <article>
            <figure>
                <img>
                <figcaption></figcaption>
            </figure>
        </article>
    </section>
    <footer></footer>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="author" content="E.Martinez" />
        <meta name="viewport" content="width=device-wid
        <title>Non-Semantic Webpage Elements</title>
    </head>
    <body>
    <div id="header"></div>
    <div class="section">
        <div class="article">
            <div class="figure">
                <img>
                <div class="figcaption"></div>
            </div>
        </div>
    </div>
    <div id="footer"></div>
    </body>
</html>
```

<header>

- element is generally found at the top of a document, a section, or an article and usually contains the main heading and some navigation and search tools.
-
- Example

```
<header>
    <h1>TheCompany</h1>
        <ul>
            <li><a href="/home">Home</a></li>
            <li><a href="/about">About</a></li>
            <li><a href="/contact">Contact us</a></li>
        </ul>
</header>
```

<nav>

- element defines a set of navigation links.

```
    <nav>
        <a href="/html/">HTML</a> |
        <a href="/css/">CSS</a> |
        <a href="/js/">JavaScript</a> |
        <a href="/jquery/">jQuery</a>
    </nav>
```

<section>

- It element defines a section in a document.
- According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."
- A web page could normally be split into sections for introduction, content, and contact information.
- Example

```
<section>
    <h2>Semantic Elements</h2>
    <p>Elements such as &lt;header&gt;, &lt;footer&gt; and &lt;article&gt; are all
    considered semantic because they accurately describe the purpose of the element
    and the type of content that is inside them.</p>
</section>
```

<article>

- element specifies independent, self-contained content.
- An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.
- Examples of where an <article> element can be used: Forum post, Blog post, Newspaper article

```
<section>
    <h3>News Story 1</h3>
    <article>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ac lectus in turpis tristique
        consequat. Aenean vel mollis ante. Ut vitae ligula justo. Sed sit amet pulvinar quam. Nulla ut tortor
        vitae risus cursus posuere. Ut eu orci sed mi dignissim auctor et quis nisi. Nullam mattis, mi nec
        varius sagittis, metus orci feugiat leo, a venenatis orci elit eu enim. Integer auctor massa nisi,
        quis molestie orci malesuada eget. Proin euismod risus ac sem cursus lacinia. Praesent in diam a enim
        venenatis dapibus. Curabitur magna eros, rutrum eu enim sit amet, vulputate varius justo. Duis
        consequat eleifend justo vel sagittis. Sed ultrices, nisl vel aliquam semper, purus nibh tincidunt
        nunc, in scelerisque orci diam malesuada nibh. Aenean rhoncus elit eu risus vehicula rutrum.
        Vestibulum luctus mauris id nisi vestibulum imperdiet.
        </p>
    </article>
    <h3>News Story 2</h3>
    <article>
        <p>Proin eget arcu at turpis sollicitudin sagittis. Morbi fermentum quis ipsum eu commodo. Nam
        eleifend, sapien ut pharetra rutrum, enim justo elementum elit, et tempus odio est vel sem. Nam
        semper dignissim elit, in mollis nibh tincidunt in. Aliquam ornare lectus nisl. Quisque eget lacinia
        lectus. Praesent a purus suscipit, dapibus massa ac, fringilla mauris. Cras nec viverra tellus.
        Quisque non consectetur libero, id dictum mi.
        </p>
    </article>
</section>
```

<figure> and <figcaption>

- tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
- tag defines a caption for a <figure> element. The <figcaption> element can be placed as the first or as the last child of a <figure> element.

<footer>

- element defines a footer for a document or section.
- element typically contains:
    - authorship information
    - copyright information
    - contact information
    - sitemap
    - back to top links
    - related documents

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements Part1: Semantic Wiki		Lesson 4 Exercise4	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.

Instructions

1. Create a Semantic Wiki about a specific interest. (i.e. chocolate, video game, specific band, etc.)
2. The page will have a title and a list of internal page links.
3. The page should include a short intro, definition, history, and other information about your topic.
4. Use appropriate *semantic html5 elements* to structure the content of your webpage.
5. The bottom of the page will be the references which is a list of the external links to the Wikipedia or other pages where you got your information.
6. Format the content as you deem necessary to make it easier to read the content.
7. Save the file as Exercise4.html in the ~/FirstGrading folder.

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Originality	Shows original thought	Has some originality	Is simple and shows little creative effort	shows no creative effort	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Semantic Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
	The background, colors and layout are artful and consistent across the website and enhance the readability of the information presented.	The background, colors and layout are consistent across the website and make it easy to read the information presented.	The background, colors and layout are distracting and make it difficult to read the information presented	The background, colors and layout make the site unattractive, and it is difficult to read the information presented.	

UNIVERSITY OF Baguio

LESSON 5: HTML Elements: Tables

Duration: 1.5 hour

About this lesson: At the end of this lesson you are expected to:

1. Define HTML table elements.
2. Use table elements to structure webpage content.

5.1. HTML Table Element

- It is an HTML structure for creating rows and columns. It is used for lists, specifications and other tabular data as well as to locate elements on the page.
- The <tr> elements are the data containers in the table.
- The <td> elements can contain all sorts of HTML elements like text, images, lists, other tables, etc.
- A table header is defined with the <th> tag. By default, table headings are bold and centered.

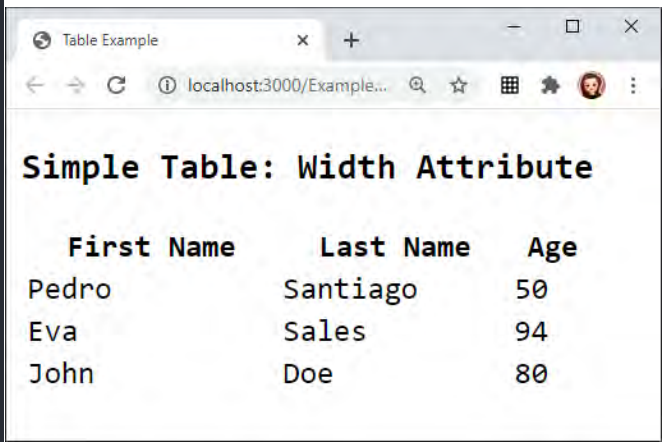
5.2. HTML Table Attributes

Property name	values	Description
align	left center right	Alignment of table according to surrounding text
bgcolor	rgb(x,x,x) ->x=0-255 #xxxxxx ->x=0-9,a-f colorname	Background color for a table or cell
border	integer (pt, px, %)	Adds a border or specifies the weight of the border
cellpadding	px	Space between the cell wall and content
cellspacing	px	Space between cells
frame	void above below hsides	Which borders should be visible
colspan	integer	Specifies how many columns to merge
rowspan	integer	Specifies how many rows to merge
caption	<caption></caption>	Add a caption to a table
width	px, %	Width of the table

5.3. Example

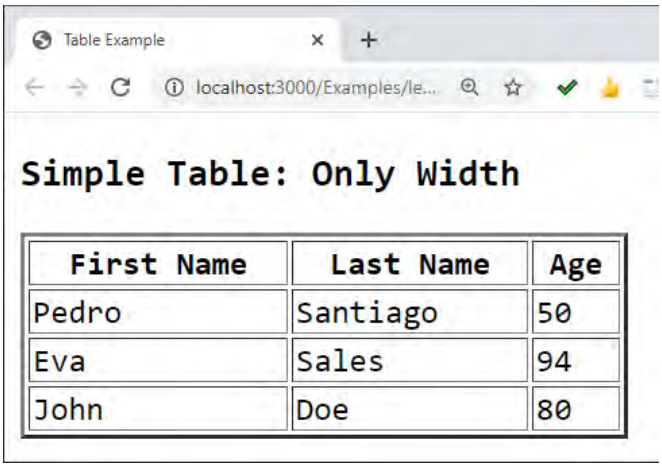
Example 1. HTML Table with only width attribute

```
<h3>Simple Table: Width Attribute</h3>
<table width='300px'>
    <thead>
        <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Age</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Pedro</td>
            <td>Santiago</td>
            <td>50</td>
        </tr>
        <tr>
            <td>Eva</td>
            <td>Sales</td>
            <td>94</td>
        </tr>
        <tr>
            <td>John</td>
            <td>Doe</td>
            <td>80</td>
        </tr>
    </tbody>
</table>
```



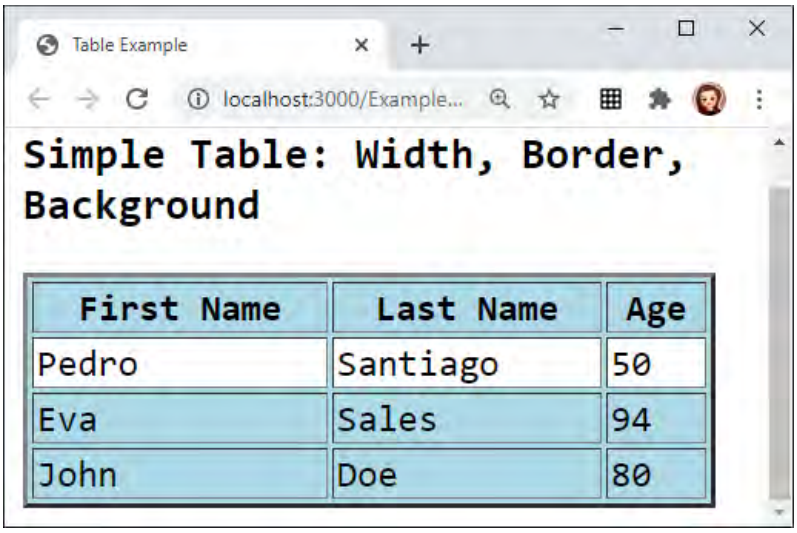
Example 2. HTML Table with border attributes

```
<h3>Simple Table: Width and Border</h3>
<table width='300px' border='2pt'>
    <thead>
        <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Age</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Pedro</td>
            <td>Santiago</td>
            <td>50</td>
        </tr>
        <tr>
            <td>Eva</td>
            <td>Sales</td>
            <td>94</td>
        </tr>
        <tr>
            <td>John</td>
            <td>Doe</td>
            <td>80</td>
        </tr>
    </tbody>
</table>
```



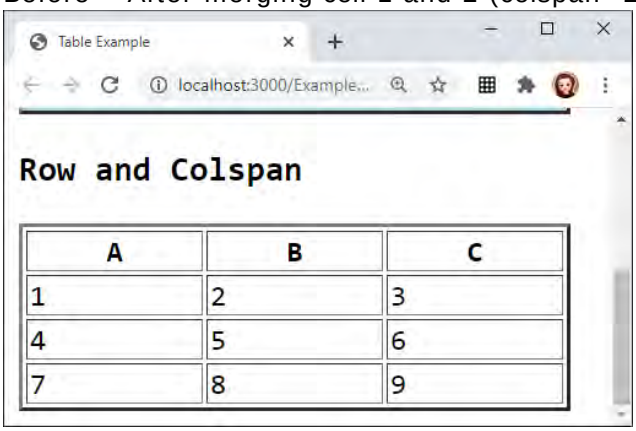
Example 3. HTML Table with border and bgcolor attributes

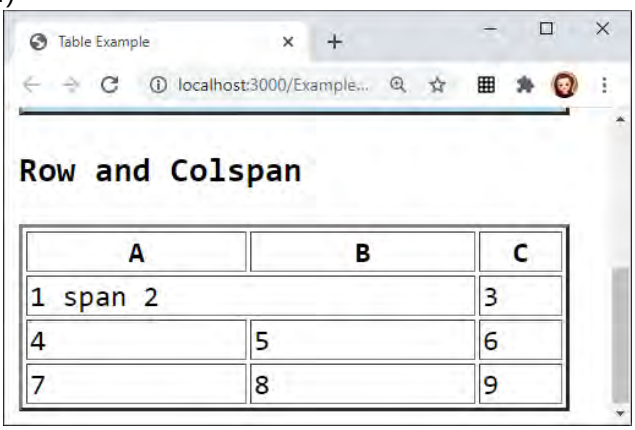
```
<table width='300px' border='2pt' bgcolor='lightblue'>
    <thead>
        <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Age</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td bgcolor='white'>Pedro</td>
            <td bgcolor='white'>Santiago</td>
            <td bgcolor='white'>50</td>
        </tr>
        <tr>
            <td>Eva</td>
            <td>Sales</td>
            <td>94</td>
        </tr>
        <tr>
            <td>John</td>
            <td>Doe</td>
            <td>80</td>
        </tr>
    </tbody>
</table>
```



Example 4-5. HTML Table with colspan and rowspan attributes

Before After merging cell 1 and 2 (colspan=2)

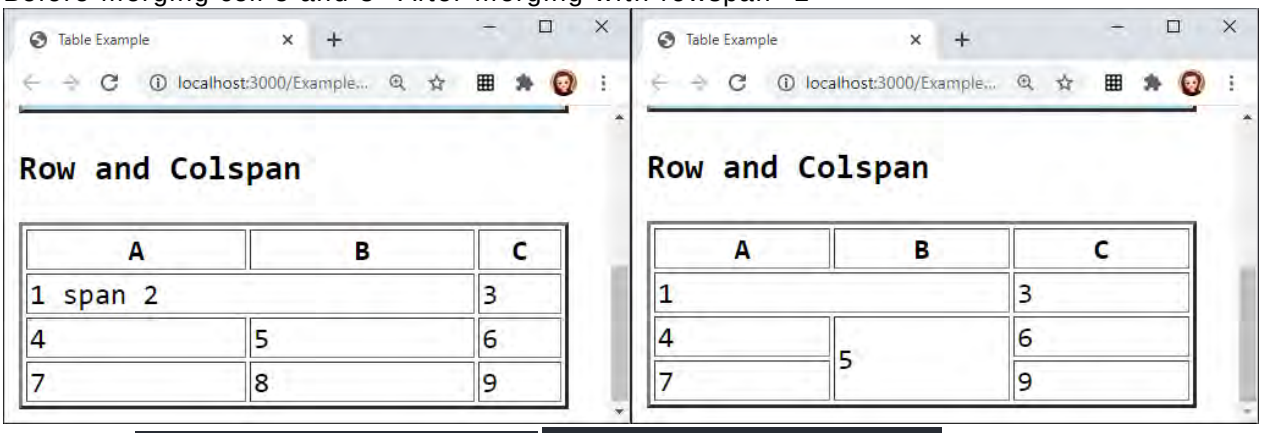


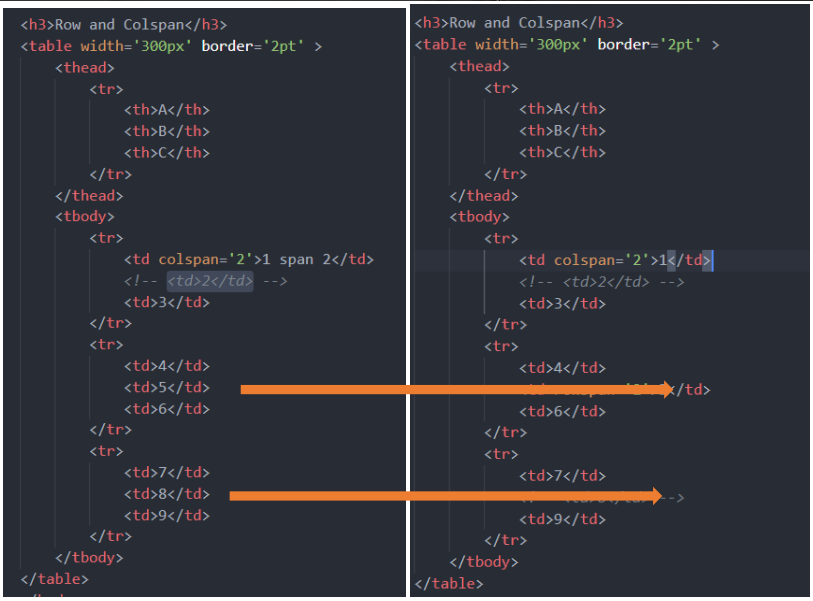


```
<h3>Row and Colspan</h3>
<table width='300px' border='2pt' >
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>5</td>
            <td>6</td>
        </tr>
        <tr>
            <td>7</td>
            <td>8</td>
            <td>9</td>
        </tr>
    </tbody>
</table>
```

```
<h3>Row and Colspan</h3>
<table width='300px' border='2pt' >
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td colspan='2'>1 span 2</td>
            <!-- <td>2</td> -->
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>5</td>
            <td>6</td>
        </tr>
        <tr>
            <td>7</td>
            <td>8</td>
            <td>9</td>
        </tr>
    </tbody>
</table>
```

Before merging cell 5 and 8 After merging with rowspan=2

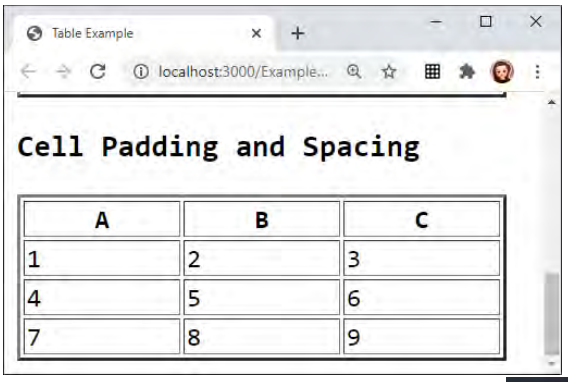


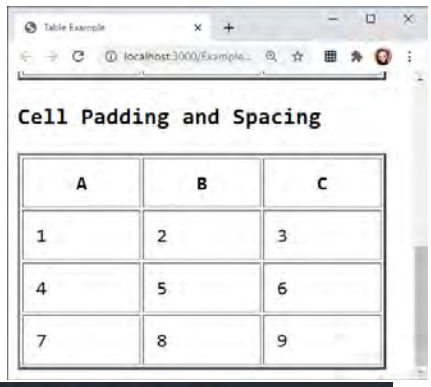


```
<h3>Row and Colspan</h3>
<table width='300px' border='2pt' >
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td colspan='2'>1 span 2</td>
            <!-- <td>2</td> -->
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>5</td>
            <td>6</td>
        </tr>
        <tr>
            <td>7</td>
            <td>8</td>
            <td>9</td>
        </tr>
    </tbody>
</table>
```

```
<h3>Row and Colspan</h3>
<table width='300px' border='2pt' >
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td colspan='2'>1</td>
            <!-- <td>2</td> -->
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>6</td>
        </tr>
        <tr>
            <td>7</td>
            <td>9</td>
        </tr>
    </tbody>
</table>
```

Example 5-6. HTML Table with cellpadding and cellspacing attribute
Before adding the cellpadding After adding the cellpadding



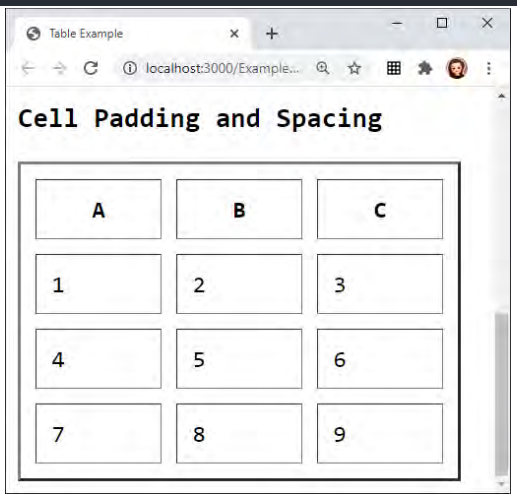



```
<h3>Row and Colspan</h3>
<table width='300px' border='2pt' >
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
```

```
<h3>Cell Padding and Spacing</h3>
border='2pt' cellpadding='10px'>
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
    <tbody>
```

After adding the cellpadding

```
<h3>Cell Padding and Spacing</h3>
<table width='300px' border='2pt' cellpadding='10px' cellspacing='10px'>
    <thead>
        <tr>
            <th>A</th>
            <th>B</th>
            <th>C</th>
        </tr>
    </thead>
```



UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements: Tables		Lesson 5 Exercise5	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.

Objectives

- Use Basic HTML structure
- Use Basic HTML formatting tags/mark-up
- Use html table tags (table, td, th, colspan, rowspan, etc)

Instructions:

1. Create a new HTML document and save it as Exercise5.html.
2. Recreate the different table configurations in a single web page (Tables A to H) as shown in the next page.

A	B	C
D	E	F

Table A

A	D
B	E
C	F

Table B

Title goes here		
A	C	E
B	D	F

Table C

Title goes here	A	D
	B	E
	C	F

Table E

Title goes here	A	D
	B	E
	C	F

Table F

Title goes here	A	B
C	D	E	F	G
	H	I	
	K	L	M	J
N	O			

Table H

Title goes here	A	D
	B	E
	C	F

Table G

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Originality	Shows original thought	Has some originality	Is simple and shows little creative effort	shows no creative effort	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Semantic Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
Table attributes • **rowspan** • **colspan** • **cellspacing** • **cellpadding**	Appropriate table **attributes and attribute values** were used to table contents. Format was followed.	Most table **attributes and attribute values were** used to table contents. Format was mostly correct and followed.	Some table **attributes and attribute values were** used to table contents. Some appropriate and correct. Format was somewhat followed.	Wrong or inappropriate table **attributes and attribute values were** used to table contents. Table **attributes and attribute values** used to table contents were inconsistent. Format was not followed.	

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements: Tables		Lesson 5 Exercise6	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.

Instructions:

1. Create a new HTML document and save it as Exercise6.html.
2. Recreate the webpage shown in the next section.
    a. Line 1-2 -> heading size 1, font face = Verdana
    b. Line 3 -> Table caption (Course and Year Level)
3. You may choose the border, background/BACKGROUND IMAGE and forecolor for the table.
4. You may also define your own the font attributes for the table.

** Be guided by the grading rubric.*

2nd Semester Schedule
Your Name
Course and Year Level

	MON	TUES	WED	THUR	FRI	SAT
7:30						
8:00						NSTPRO 1 UB Gym
8:30						
9:00		TRBSHT1 FA01		TRBSHT1 FA01		
9:30						
10:00						
10:30	ITFNDS1 F309		ITFNDS1 F309		ITFNDS1 F309	
11:00						
11:30	ITLAWS1 F306	COMPRG1 F217	ITLAWS1 F306	COMPRG1 F217		
12:00						
12:30						
1:00						
1:30		ITFNDL1 F111		OPSYSL1 F112	WEBDVT1 D303	
2:00						
2:30						
3:00						
3:30						
4:00						
4:30		WEBDVT1 D303		COMPRG1 F110		
5:00						
5:30						
6:00						
6:30						
7:00						
7:30						
8:00						

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Originality	Shows original thought	Has some originality	Is simple and shows little creative effort	shows no creative effort	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Semantic Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
Table attributes • **rowspan** • **colspan** • **cellspacing** • **cellpadding**	Appropriate table **attributes and attribute values** were used to table contents. Format was followed.	Most table **attributes and attribute values were** used to table contents. Format was mostly correct and followed.	Some table **attributes and attribute values were** used to table contents. Some appropriate and correct. Format was somewhat followed.	Wrong or inappropriate table **attributes and attribute values were** used to table contents. Table **attributes and attribute values** used to table contents were inconsistent. Format was not followed.	

UNIVERSITY OF Baguio

LESSON 6: HTML Elements: Links

Duration: 1.5 hour

About this lesson: At the end of this lesson you are expected to:

1. Define HTML hyperlinks.
2. Use hyperlinks to link a webpage to internal and external webpages.

6.1. HTML hyperlinks

According to web standards (w3schools, n.d.),

- A hyperlink (or link) is a word, group of words, or image that you can click on to jump to another document. A link, or an anchor, is a word, a group of words or an image you can click to jump to another document or a specific part of the current document.
- The element for links, both internal and external is as simple as <a>.
- The <a> tag defines a hyperlink, which is used to link from one page to another.
- The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
- By default, links will appear as follows in all browsers:
  - An unvisited link is underlined and blue
  - A visited link is underlined and purple
  - An active link is underlined and red

6.2. Hyperlink syntax

- The HTML code for a link is simple. It looks like this:

```
<a href="url">Link text</a>
```

6.3. Hyperlink Attributes

Property name	values	Description
download	ilename Example: Download File	Specifies that the target will be downloaded when a user clicks on the hyperlink
href	URL Example: Mozilla Homepage	Specifies the URL of the page the link goes to
target	_blank _parent _self _top	Specifies where to open the linked document
Title	Text description Example: the Mozilla homepage	hovering over the link shows a tooltip

6.4. Hyperlink Implementation

6.4.1. Absolute Links (absolute urls)

- "Points to a location defined by its absolute location on the web, including protocol and domain name." (Link Types - HTML: HyperText Markup Language | MDN, 2021)
- Examples:
  - https://www.example.com
  - https://www.example.com/projects/index.html

6.4.2. Relative Links (relative urls)

- "Points to a location that is relative to the file you are linking from, more like what we looked at in the previous section." (Link Types - HTML: HyperText Markup Language | MDN, 2021)
- A relative URL will point to different places depending on the actual location of the file you refer from.
- Examples:
  - The page containing the link ->`https://www.example.com/projects/index.html`
  - The link in the page to a PDF file in the same directory: `<a filename="project-brief.pdf"/>File</a>`

6.4.3. Document fragments

- Creates a link to a specific part of an HTML another or the same document, known as a document fragment, another section of the same document.

Linking to a part of another document:

- To do this you first have to assign an id attribute to the element you want to link to, like so:

```
<a id="part2">Part 2</a>
```

- Then to link to that specific id, you'd include it at the end of the URL, preceded by a hash/pound symbol (#),like so:

```
<a href="lesson2.html/#part2">Go to Part 2 </a>of Lesson 2
```

Linking to a part the same document:

- To do this you first have to assign an id attribute to the element you want to link to, like so:

```
<a id="part2">Part 2</a>
```

- Then to link to that specific id, you'd include it at the end of the URL, preceded by a hash/pound symbol (#),like so:

```
<a href="part2">Go to Part 2</a>
```

6.4.4. Download link

- A Download Link links a file for the visitor to download
- Example:

```
<a href="File URL.zip"> Click here to download this file </a>
```

6.4.5. mailto link

- E-Mail links, otherwise known as a "mailto" links, use the following format:
- Example

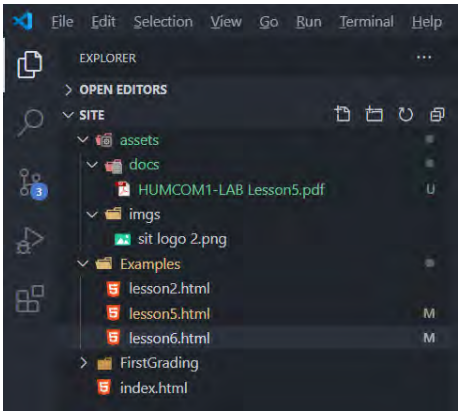
```
<a href="mailto:myname@mysite.com">Click Here to Email Me</a>
```

- Example: mailto with subject line (may not work on all browsers)

```
<a href="mailto:myname@mysite.com?subject=Inquiry From Website">Click Here to Email Me</a>
```

Try this example:

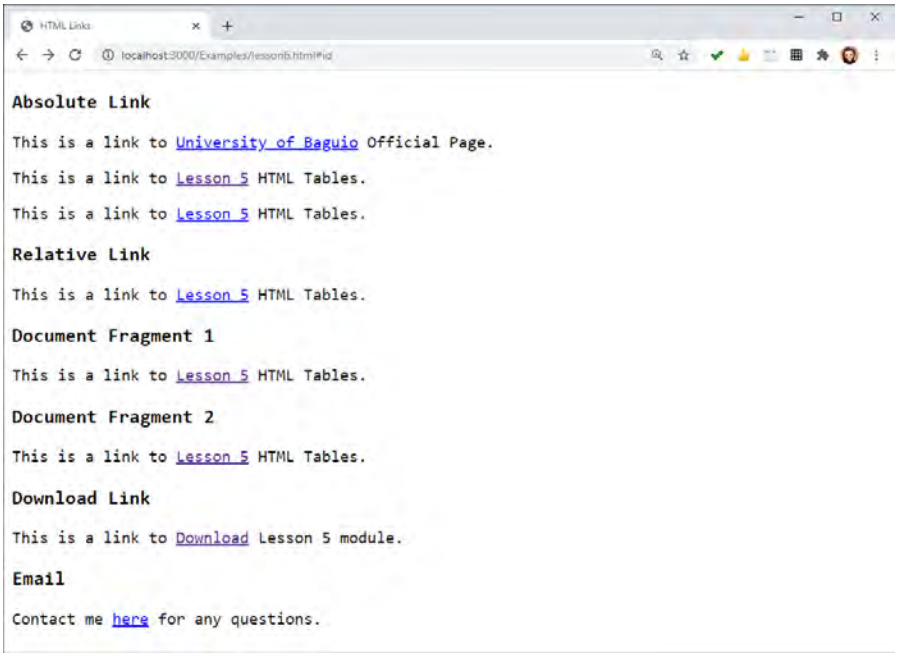
- site directories and documents



- lesson6.html ->body containing links

```
<body>
    <h3><a id="top">Absolute Link</a></h3>
        <p>
            This is a link to <a href="http://www.ubaguio.edu/main/" target='_blank'>University of Baguio</a> Official Page.
        </p>
        <p>
            This is a link to <a href="https://zenitramazre.github.io/martinez-humcom/site/Examples/lesson5" target='_blank'>Lesson 5</a> HTML Tables.
        </p>
        <p>
            This is a link to <a href="site\Examples\lesson5.html" target='_blank'>Lesson 5</a> HTML Tables.
        </p>

    <h3>Relative Link</h3>
        <p>
            This is a link to <a href="lesson5.html\#table3" target='_blank'>Lesson 5</a> HTML Tables.
        </p>
    <h3>Document Fragment 1</h3>
        <p>
            This is a link to <a href="lesson5.html" target='_blank'>Lesson 5</a> HTML Tables.
        </p>
    <h3>Document Fragment 2</h3>
        <p>
            This is a link to <a href="#top">Lesson 5</a> HTML Tables.
        </p>
    <h3>Download Link</h3>
        <p>
            This is a link to <a href="..\assets\docs\HUMCOM1-LAB Lesson5.pdf">Download</a> Lesson 5 module.
        </p>
    <h3>Email</h3>
        <p>
            Contact me <a href="mailto:emartinez@e.ubaguio.edu">here</a> for any questions.
        </p>
</body>
```

- output



UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML HTML Elements: Hyperlinks		Lesson 6 Exercise7	Score:

General Instructions

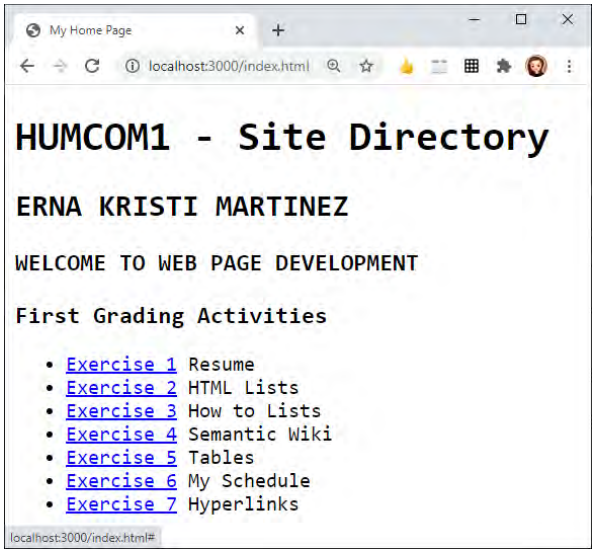
- Always complete the general information in the heading section
- Commit and push the local changes in repository to your remote GitHub repository.

Instructions:

1. Open your index.html document.
2. Edit the page so that it will contain a link:
    a. To all your first grading activities.
    b. Add a title attribute on all the links
    c. Use relative url to link your pages.
3. Use a list to create the link items.
4. All linked pages must open on a new page.

** Be guided by the grading rubric.*

Sample Output



Grading Rubric

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Originality	Shows original thought	Has some originality	Is simple and shows little creative effort	shows no creative effort	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Layout and Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
Link attributes • **href** • **title** • **target**	Appropriate table **attributes and attribute values** were used to table contents. Format was followed.	Most table **attributes and attribute values were** used to table contents. Format was mostly correct and followed.	Some table **attributes and attribute values were** used to table contents. Some appropriate and correct. Format was somewhat followed.	Wrong or inappropriate table **attributes and attribute values were** used to table contents. Table **attributes and attribute values** used to table contents were inconsistent. Format was not followed.	

UNIVERSITY OF Baguio

LESSON 7: HTML Elements: Media

Duration: 1.5 hour

About this lesson: At the end of this lesson you are expected to:

1. Define HTML media elements.
2. Add multimedia elements to a webpage.

7.1. HTML media

- Multimedia elements (like sounds or videos) are stored in media files.

7.1.1. HTML image

- In HTML, images are defined with the <img> tag.
- The <img> tag is empty, which means that it contains attributes only, and has no closing tag.
- To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.

  Syntax:

```
<img src="url" alt="some_text">
```

  Example

7.1.2. HTML video

- HTML5 defines a new element which specifies a standard way to embed a video/movie on a web page: the <video> element and its attributes as shown in the following table

Tag	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in mediaplayers

  Example

```
<video width="320" height="240" controls="controls">
   <source src="movie.mp4" type="video/mp4">
   <source src="movie.ogg" type="video/ogg">
 Your browser does not support the video tag.
 </video>
```

- Video formats supported by browsers (source: w3schools)

Browser	MP4	WebM	Ogg
Edge	yes	yes	yes
Firefox	yes	yes	yes
Google Chrome	yes	yes	yes
Apple Safari	yes	yes	no
Opera	yes	yes	yes

7.1.3. HTML audio

- The <audio> element specifies a standard way to embed an audio file on a web page.

  Example

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

- The following table gives a listing of the tags associated in incorporating audio in a website

Tag	Description

<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	

- Audio formats supported by browsers (source: w3schools)

Browser	MP3	WAV	OGG
Edge	yes	NO	NO
Firefox	yes	yes	yes
Google Chrome	yes	yes	yes
Apple Safari	yes	yes	no
Opera	yes	yes	yes

Name:		Date:	
Introduction to HTMLHTML Elements: Media		Lesson 7 Exercise8	Score:

General Instructions

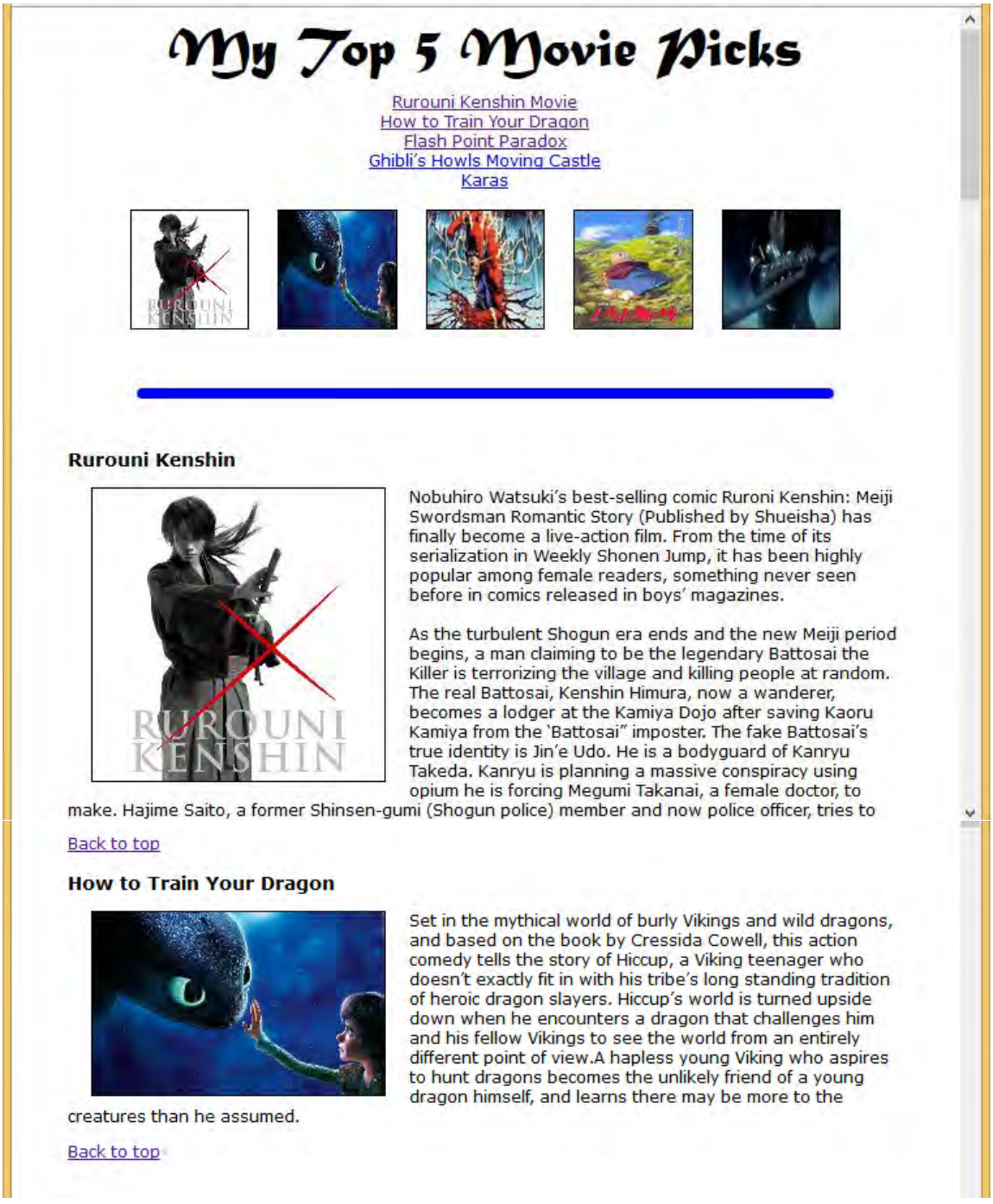
- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save First grading exercises in the FirstGrading directory.

Instructions:

1. Create a list of the titles of 5 of your favorite movies.
2. In the same page, write a synopsis about each movie and include a thumbnail for each.
    a. The thumbnail is 150px by 150px

3. Link each item on the list to its corresponding movie synopsis within the pages as seen in the sample output.
4. Cite all the sources of your content within the page.
5. At the footer of the page, add all the resources or references you used for the page content.
6. Refer to the sample output for your reference.

Sample Output



Grading Rubric

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Text Elements	Use of bullets, italics, bold, and indentations enhances readability.	Bullets, italics, bold, and indentations for headings and subheadings sometimes detract or do not enhance readability.	Inconsistent use of bullets, italics, bold, and indentations of text detracts from readability.	Lack of bullets, indentations for headings and sub-headings and body text make comprehension difficult.	
Link attributes • **href** • **title** • **target**	Appropriate table **attributes and attribute values** were used to table contents. Format was followed.	Most table **attributes and attribute values were** used to table contents. Format was mostly correct and followed.	Some table **attributes and attribute values were** used to table contents. Some appropriate and correct. Format was somewhat followed.	Wrong or inappropriate table **attributes and attribute values were** used to table contents. Table **attributes and attribute values** used to table contents were inconsistent. Format was not followed.	
Media elements	All of the photographs, graphics, sound and/or video enhance the content and create interest.	Most of the photographs, graphics, sound and/or video enhance the content and create interest.	A few of the photographs, graphics, sound and/or video are inappropriate for the content and do not create interest.	The photographs, graphics, sounds, and/or videos are inappropriate for the content or are distracting decorations that create a busy feeling and detract from the content.	
Fair Use Guidelines	Fair use guidelines are followed with proper use of citations throughout the Web page.	Fair use guidelines are frequently followed and most non original material uses proper citations.	Sometimes fair use guidelines are followed and some non-original material uses proper citations.	Fair use guidelines are not followed. Non original material is improperly cited.	

UNIVERSITY OF Baguio

LESSON 8: HTML Elements: Form

Duration: 1.0 hour

About this lesson: At the end of this lesson you are expected to:

1. Define HTML form elements and corresponding attributes.
2. Create forms in a webpage.

8.1. HTML form

A form is a part of the HTML document that allows the page to collect any kind of information from the user.

8.1.1. HTML form tag

- A form is defined with the <form> tag, which tells the browser where the form starts and ends. All kinds of HTML tags can then be added within the <form> tag.
- Syntax:

```
<form>
      :
      form elements/html elements
      :
</form>
```

8.1.2. HTML form input tags

- Most of the form elements are defined with the <input> tag. The type of element is specified through the type attribute.

8.1.2.1. Text field

- It is used to define a one-line input field that a user can enter text into.
- Syntax:

```
<form>
      First name: <input type="text" name="firstname" /><br />
      Last name: <input type="text" name="lastname" />
</form>
```

8.1.2.2. Password field

- Password fields are similar to text fields. The only difference is that the characters entered into a password field shows up as dots on the screen. This is, of course, to prevent others from reading the password on the screen.
- Password field properties are the same as text field properties.

Example

```
<form>
      Password: <input type="password" name="pwd" />
</form>
```

8.1.2.3. Radio button

- Radio buttons are used when a user is limited to selecting only one option from a set of alternatives. The properties of radio buttons are the same as checkboxes

Example

```
<form>
      <input type="radio" name="sex" value="male" /> Male<br />
      <input type="radio" name="sex" value="female" /> Female
</form>
```

8.1.2.4. Checkbox

- Checkboxes are used when a user is allowed to select one or more options from a set of alternatives.

Example

```
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>
```

8.1.2.5. Submit Button

- Submit buttons send form data to a back-end process or application. The back-end process then verifies and processes the data, eventually passing the information into some database application.

  Example

```
<form action="demo_form.asp">
  Select your favorite color: <input type="color" name="favcolor" /><br />
  <input type="submit" />
</form>
```

8.1.2.6. Color

- The color type is used for input fields that should contain a color.

  Example

```
<form action="demo_form.asp">
  Select your favorite color: <input type="color" name="favcolor" /><br />
  <input type="submit" />
</form>
```

8.1.2.7. Date

- The date type allows the user to select a date.

  Example

```
<form
  <label>Birthday</label>
   <input type="date" name="bday" />
  <input type="submit" />
</form>
```

8.1.2.8. Number

- The input element with a type attribute whose value is "number" represents a precise control for setting the element's value to a string representing a number.
- The number type is used for input fields that should contain a numeric value. You can also set restrictions on what numbers are accepted. Use the following attributes to specify restrictions:
  - max - specifies the maximum value allowed
  - min - specifies the minimum value allowed
  - step - specifies the legal number intervals
  - value - Specifies the default value

  Example

```
<form>
  <label>Quantity (between 1 and 5):</label>
      <input type="number" name="quantity" min="1" max="5" />
  <input type="submit" />
</form>
```

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to HTML: HTML Elements: Forms		Lesson 8 Exercise9	Score:

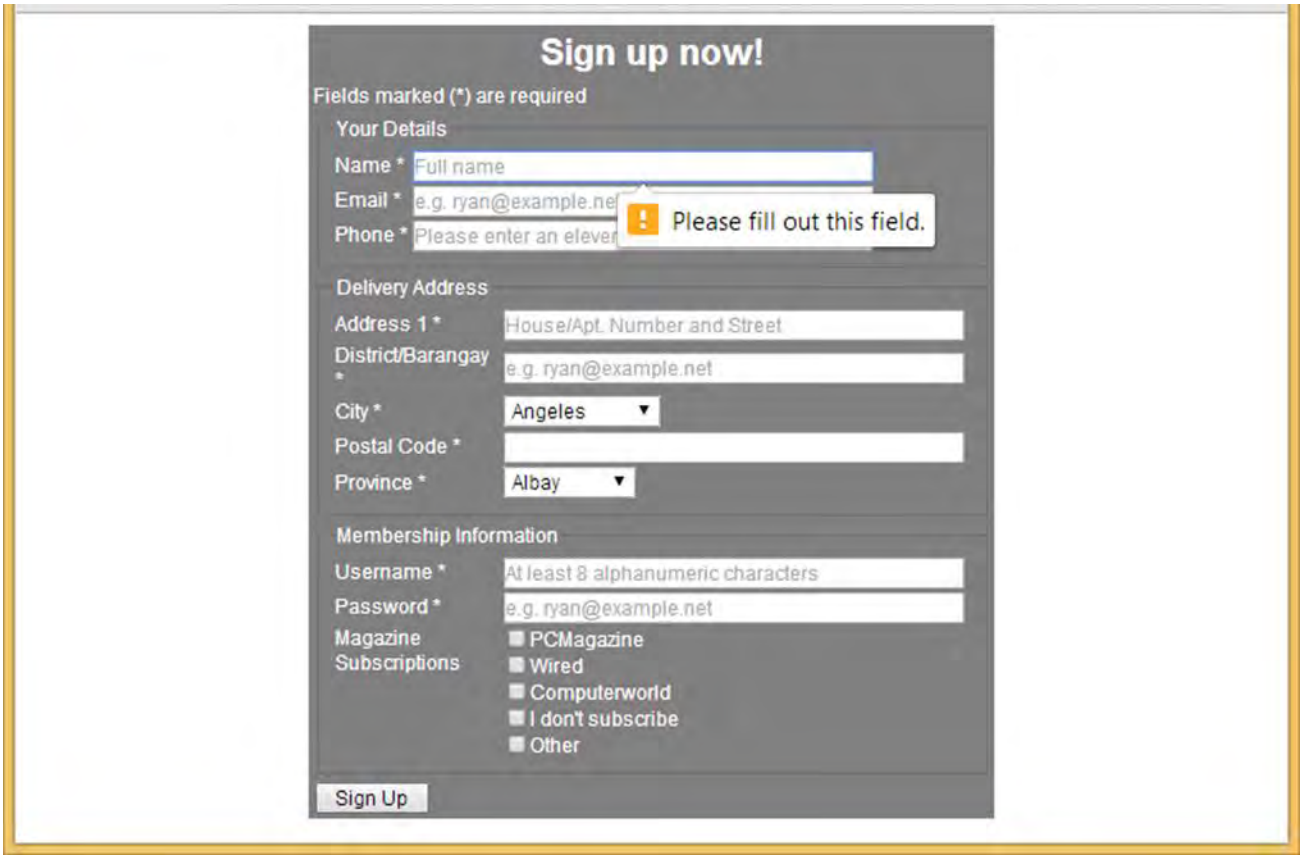
General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save First grading exercises in the FirstGrading directory.

Instructions:

1. Set the title in the title bar to "Registration Form" + Your Name.
2. Use a <label> tag for all the labels.
3. Use the appropriate input type for each field.
4. Use a <fieldset> to group information.
5. Set the appropriate sizes and/or ranges for each input.
6. Validate missing input when "Sign up" button is clicked.
7. Add a placeholder where it applies.
8. Add html5 attributes that will validate user inputs as soon as they click on the 'submit' button.
    - required input
    - pattern for telephone numbers, passwords, and email
    - min and max values for dates and character lengths
9. Refer to the sample output for your reference.

Sample output



Grading Rubric

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	**4**	**3**	**0-2**	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
Text Elements	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
	10-9	**8-7**	**6-4**	**3-0**	
Form Element and validation	All the required form elements were added and validation attributes are working.	Most of the form elements were added and validation attributes are working.	Some of the form elements were added and with some attributes are working.	Form elements were added and but the validations are not working.	

LESSON 9: Introduction to CSS

Duration: 1.0 hour

About this lesson: At the end of this lesson you are expected to:

1. Understand the importance of separating structure from design
2. Explain Cascading Style Sheets (CSS)
3. Differentiate inline, Embedded, and external styles.
4. Explain the difference between a selector, property, and value.
5. Explain CSS rules and syntax.

9.1.Introduction to CSS

According to W3C, "CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers." (World Wide Web Consortium, 2016)

9.1.a.CSS Implementation

9.1.a.1.Inline

- Adding CSS style inside html line is called inline styling.
- Inline styling uses CSS attribute named as style.
- inline styles take precedence over other types of styles and affect the style for individual pieces of content
- Following is the example of using inline style sheet.

```
<p style='color: red;'>Inline stylesheet </p>
```

9.1.a.2.Internal/Embedded

- Includes the style sheet within the opening <head> and closing </head> tags of the HTML document.
- Use an embedded style sheet when you want to create styles for a single webpage that are different from the rest of the website.
- An embedded style sheet takes precedence over an external style sheet.

```
<head>
    <title>Internal CSS</title>
    <meta charset="utf-8">
    <meta name="author" content="E.Martinez" >
    <meta name="viewport" content="width=device-width">
    <style>
        body
        {
            background-color: beige;
        }
    </style>
</head>
```

9.1.a.3.External

- is a CSS file that contains all of the styles you want to apply to more than one page in the website.
- An external style sheet is a text file with the .css file extension.

Html document

```
    <head>
        <title>Internal CSS</title>
        <meta charset="utf-8">
        <meta name="author" content="E.Martinez" >
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" type="text/css"
        href="styles\styles.css" >
    </head>
<body>
    CSS style instructions stored in 'style.css' will be
    applied to this page.
</body>
```

style.css

```
body
{
    background-color: antiquewhite;
}
```

9.1.b. Anatomy of CSS Rules

- A CSS rule determines how the content of the element or elements that it is associated with should be displayed.
- Has two parts:
  - a **selector** - selects the element or elements to which the instructions specified in the declaration block are to be applied.
  - **a declaration block** - contain one or more declarations and is enclosed in curly brackets, with the declarations separated by semicolons.

- Each declaration comprises a **property** and a **value**, separated by a **colon**.
  - **Property** specifies the aspects of the selected element to change, such as font, color, width, height, and border, and
  - **Value** specifies the setting for the property.

9.1.3. Grouping Selectors

- You may apply a style to many selectors if you like. Just separate the selectors with a comma.

```
h1, h2, h3, h4, h5, h6 { color: blue }
```

9.1.4. Styling Elements

a. Common CSS property and value combinations

- **color: #FFFFFF;** *Set font color to white*
- **background-color: #000000;** *Set background color to black*
- **border: 1px solid blue;** *Create 1px-thick blue border around the element*
- **font-family: Arial, sans-serif;** *Set font to Arial*
- **font-size: 16px;** *Set font size to 16px*

- **padding: 32px;** *Add padding 32px thickness in size*
- **margin: 16px;** *Add 16 pixels of margin around the content area*

b. Shorthand properties (Example)

- Original notation
  - background-color: #000000;
  - background-image: url('image.jpg');
  - background-repeat: no-repeat;
- Shorthand
  The same can be written by using *shorthand* property **background,** separated by space:
  - background: #000000 url('image.jpg') no-repeat;

c. Background

- The CSS background properties are used to define the background effects for elements.
- Css background properties are as follows with some given use and examples :
  1.Background-color
  2.Background-image
  3.Background-repeat
  4.Background-attachment
  5.Background-position

BACKGROUND-COLOR :

- This property specifies the background-color of an element.
- A color name can also be given as : "green", a HEX value as "#5570f0", an RGB value as "rgb(25, 255, 2)".
- background-color:color name;

BACKGROUND-IMAGE :

- This property specify an image to use as the background of an element. By default, the image is repeated so it covers the entire element.
- background-image :url(' link');

BACKGROUND-POSITION :

- This property is used to set the image to a particular position.
- background-position:left top;

d. Font

- The CSS font property is used to set the fonts content of HTML element. There are many font property in CSS which are discussed below with some examples:
  1.font-family
  2.font-style
  3.font-weight
  4.font-variant
  5.font-size

FONT-FAMILY:

- It is used to set the font type of an HTML element. It holds several font names as a fallback system.
- font-family: "font family name";

FONT-STYLE: It is used to specify the font style of an HTML element.

- It can be "normal, italic or oblique".
- font-style: style name;

FONT-WEIGHT:

- It is used to set the boldness of the font. Its value can be "normal, bold, lighter, bolder".
- font-weight: font weight value;

a. Text Formatting

- CSS text formatting properties is used to format text and style text.
- CSS text formatting include following properties and some examples:
  1. Text-color
  2. Text-alignment
  3. Text-decoration
  4. Text-transformation
  5. Text-indentation

6. Letter spacing
7. Line height
8. Text-direction
9. Text-shadow
10. Word spacing

TEXT-COLOR:

- Text-color property is used to set the color of the text.
- Text-color can be set by using the name "red", hex value "#ff0000" or by its RGB value"rgb(255, 0, 0).

TEXT-ALIGNMENT:

- Text alignment property is used to set the horizontal alignment of the text.
- The text can be set to left, right, centered and justified alignment.

TEXT INDENTATION:

- Text indentation property is used to indent the first line of the paragraph.
- The size can be in px, cm, pt.

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to CSS: Implementing Styles		Lesson 9 Exercise10	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Midterm exercises in the Midterm directory.

Instructions:

1. Create a Midterm directory for your midterm exercises.
2. Create a 'styles' for your external css files in the Midterm folder.
3. Create a new css file and name it Exercise10.css in the styles directory.
4. Open the Semantic Wiki activity from the first grading exercises, make sure to save it in the Midterm folder as Exercise10.html.
5. Cite all the sources of your content within the page.
6. Apply style on the page through the external css and link it to your web page.

Grading Rubric

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	**4**	**3**	**0-2**	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
CSS Implementation	Correctly implemented external CSS, no inline or internal styles used.	Implemented external CSS, with some inline or internal styles used.	Implemented external CSS from the wrong location/directory, no inline or internal styles used.	Did not implemented external CSS, used inline or internal styles used.	
Media elements	All of the photographs, graphics, sound and/or video enhance the content and create interest.	Most of the photographs, graphics, sound and/or video enhance the content and create interest.	A few of the photographs, graphics, sound and/or video are inappropriate for the content and do not create interest.	The photographs, graphics, sounds, and/or videos are inappropriate for the content or are distracting decorations that create a busy feeling and detract from the content.	
Fair Use Guidelines	Fair use guidelines are followed with proper use of citations throughout the Web page.	Fair use guidelines are frequently followed and most non original material uses proper citations.	Sometimes fair use guidelines are followed and some non-original material uses proper citations.	Fair use guidelines are not followed. Non original material is improperly cited.	
	10-9	**8-7**	**6-4**	**3-0**	
CSS Selectors, properties and values	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	

	The background, colors and layout are artful and consistent across the website and enhance the readability of the information presented.	The background, colors and layout are consistent across the website and make it easy to read the information presented.	The background, colors and layout are distracting and make it difficult to read the information presented	The background, colors and layout make the site unattractive, and it is difficult to read the information presented.	

UNIVERSITY OF Baguio

LESSON 10: CSS Selectors

Duration: 1.0 hour

About this lesson: At the end of this lesson you are expected to:

1. Explain the difference among the CSS selectors.
2. Create styles that use text and color properties.
3. Add comments to an external stylesheet.

10.1. CSS Selectors

- species the element to style
- apply rules by selecting the specific Element from html.
- A selector can be an HTML element name, an id attribute value, or a class attribute value.

10.1.a. Basic Selectors

1. Universal Selector
   - Selects all elements. Optionally, it may be restricted to a specific namespace or to all namespaces.
     Syntax: * { property: value; }
     Example: * { background : #0000;}
2. Type (Element) Selector
   - Selects all elements that have the given ***element*** name.
     Syntax: *elementname* { property: value; }
     Example: h1 { color : #f1342a;}
3. Class Selector
   - Selects all elements that have the given ***class*** name.
     Syntax: *.classname* { property: value; }
     Example: .navis { color : #f1342a;}
4. ID Selector
   - Selects an element based on the value of its ***id*** attribute. There should be only one element with a given ID in a document.
     Syntax: *#idname* { property: value; }
     Example: #toc { color : #f1342a;}

10.1.b. Combinators

5. Decendant combinator
   - The (space) combinator selects nodes that are descendants of the first element.
     Syntax: *A B* { property: value; }
     Example: div span { background : #0000;}
     **will match all elements that are inside a element.*
6. Child combinator
   - The **>** combinator selects nodes that are direct children of the first element
     Syntax: *A > B* { property: value; }
     Example: ul > li { color : #f1342a;}
     **will match all <li> elements that are nested directly inside a <ul> element.*
7. General sibling combinator
   The ~ combinator selects siblings. This means that the second element follows the first (though not necessarily immediately), and both share the same parent.
   Syntax: *A ~ B* { property: value; }
   Example: p ~ span { background : #0000;}
   ** will match all elements that follow a <p>, immediately or not.*
8. Adjacent sibling combinator
   The + combinator selects adjacent siblings. This means that the second element directly follows the first, and both share the same parent.
   Syntax: *A + B* { property: value; }
   Example: h2 > p { color : #f1342a;}
   ** will match all <p> elements that directly follow an <h2>.*

10.1.c.Pseudo

9. Pseudo classes

- The **:** pseudo allow the selection of elements based on state information that is not contained in the document tree.
  Syntax: *A:B* { property: value; }
  Example: a:visited { background : #0000;}
  * *will match all <a> elements that have been visited by the user.*

```
a:link    { color: blue; }
a:hover   { background: red; }
a:visited { border:bottom solid 1px; }
a:active  { color: green; }
input[type="text"]:focus { font-family: Georgia; }
```

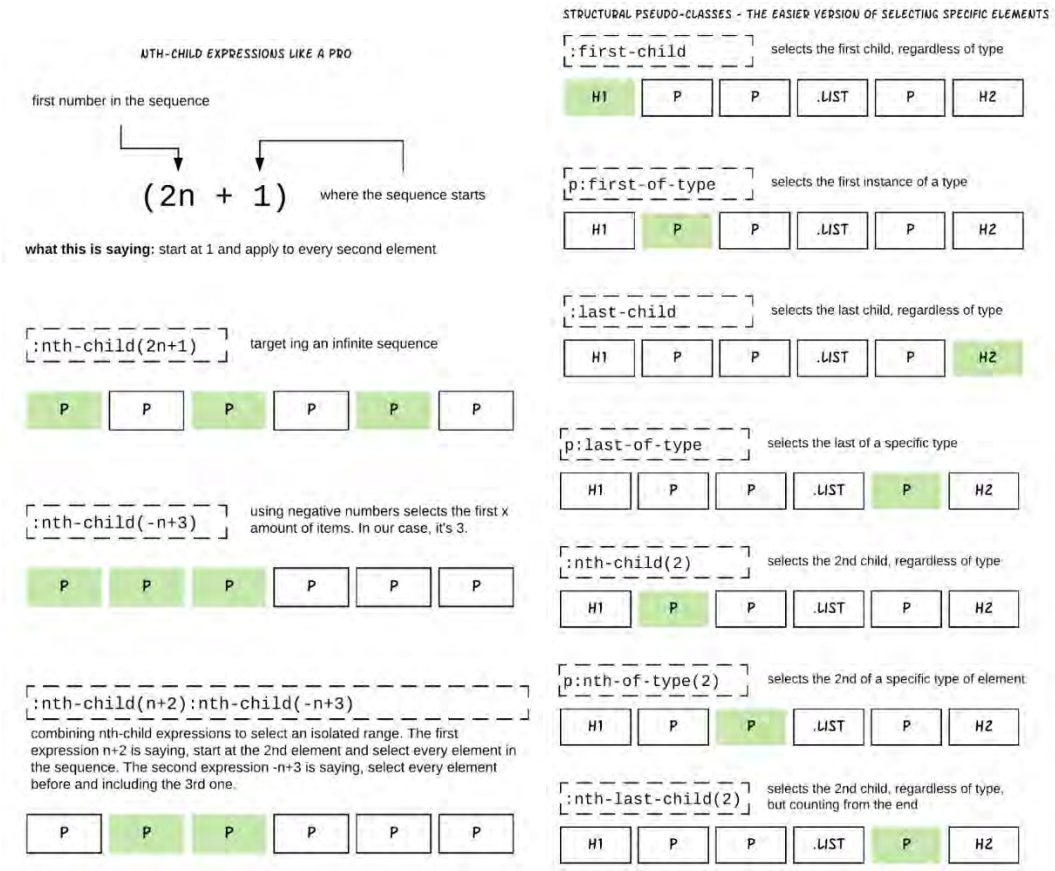
State-based Pseudo Selector
(Dechalert, 2020)

10. Pseudo elements

- The :: pseudo represent entities that are not included in HTML.
  Example: p:first-line { background : #0000;}
  * *will match the first line of all <p> elements.*

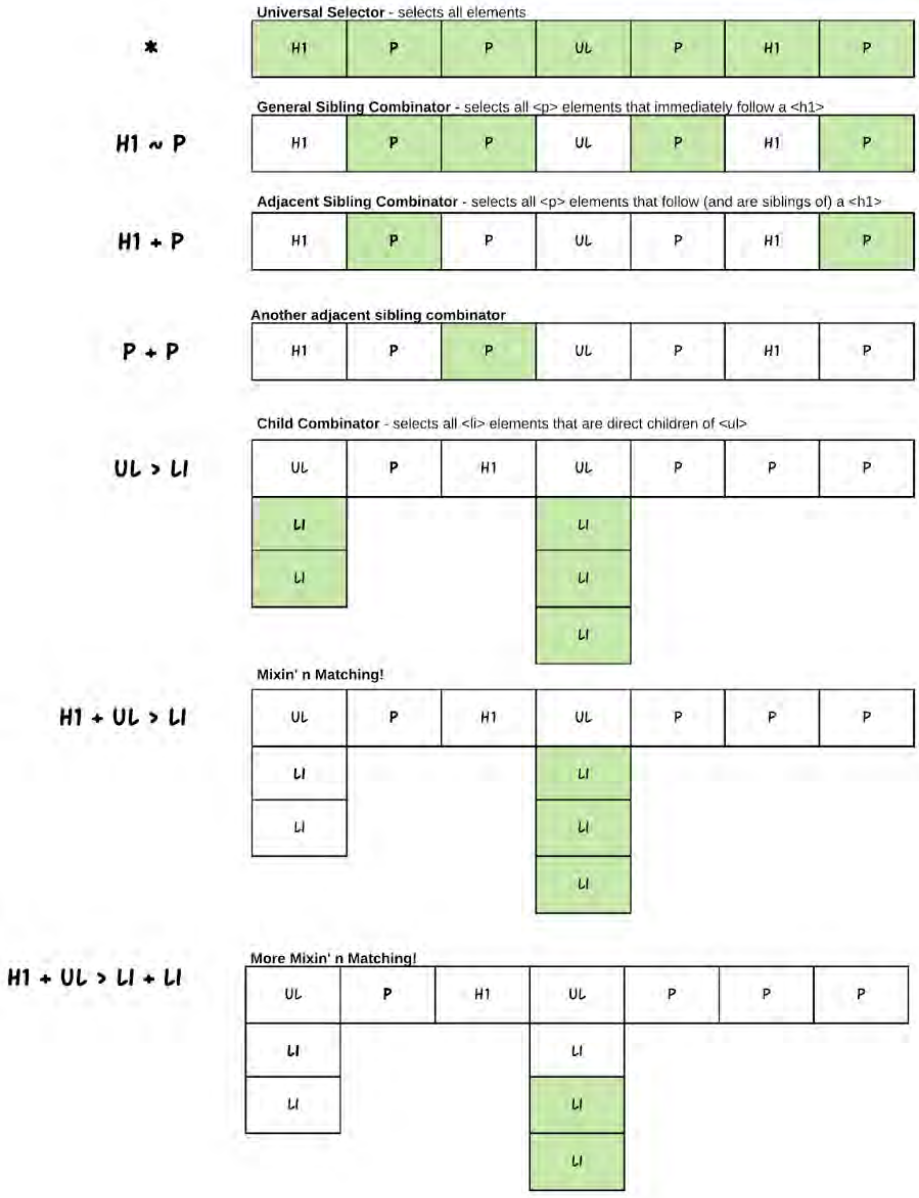
  Example: li:nth-child(2) { background : #0000;}
  * *will match the 2nd of all <li> elements.*

  Example: li:nth-child(odd) { background : #0000;}
  * *will match all the odd of all <li> elements in a list.*



Structural-based Pseudo Selector
(Dechalert, 2020)

CSS Selector Visual Reference



Child Combinator Selector Diagram
(Dechalert, 2020)

CSS Selector Reference List

- https://www.w3schools.com/cssref/css_selectors.asp
- https://www.geeksforgeeks.org/css-selectors-complete-reference/

UNIVERSITY OF Baguio

LESSON 11: CSS Box Model

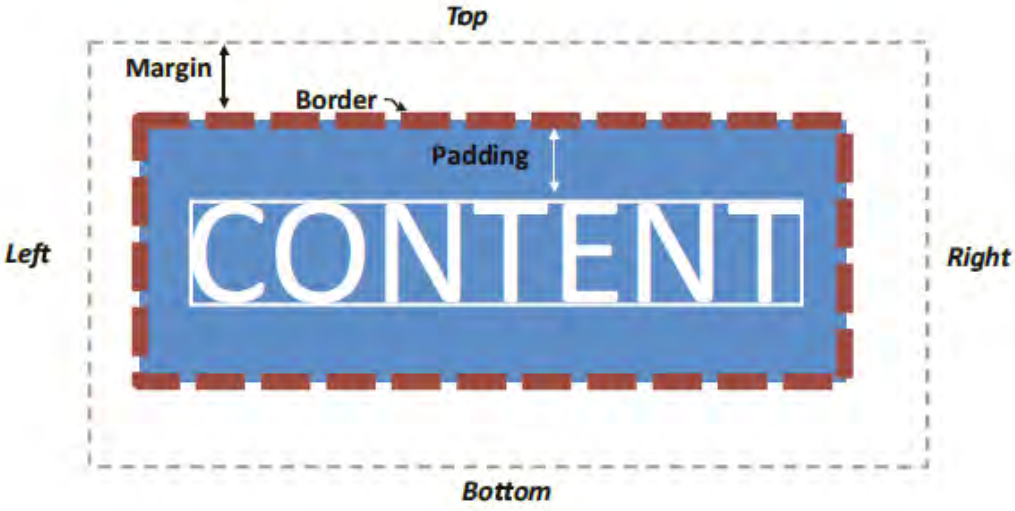
Duration: 1.0 hour

About this lesson: At the end of this lesson you are expected to:

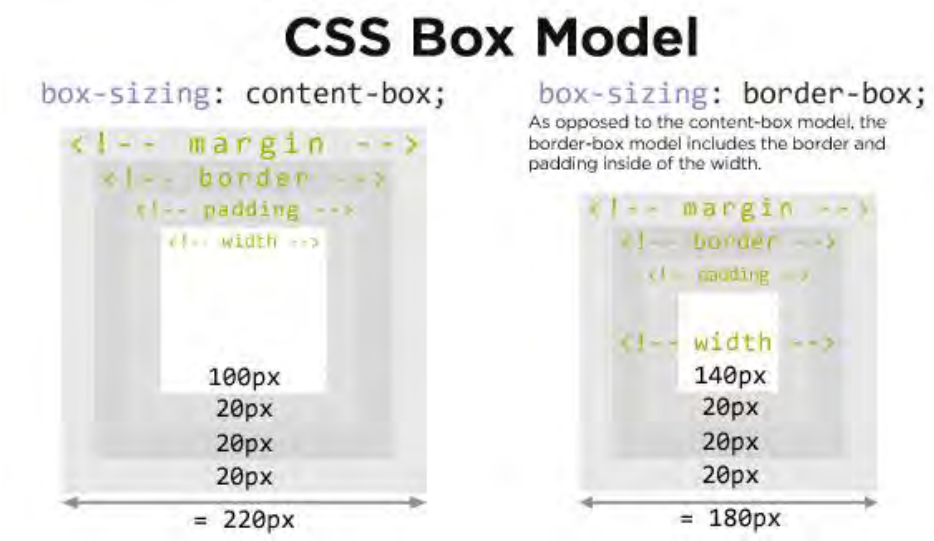
1. Describe the CSS box model and how to apply margins, padding, and borders.
2. Create styles that use padding, border, and margin properties.

11.1. CSS Box Model

- Each block element such as a header, nav, main, and footer element has a top, right, bottom, and left side.
- The box model is the fundamental structure behind every HTML element.
- Traditionally, it consists of the content area, with padding, border and margin areas surrounding it.



- The **margin**
  - provides passive white space between block elements or between the top or bottom of a webpage.
  - define margins at the top, right, bottom, and left of a block element.
  - Margins are transparent and are measured in pixels (px), ems (em), or percentages (%)
- The **border**
  - separates the padding and the margin of the block element.
  - can vary in thickness and color and can be defined at the top, right, bottom, and left sides of a block element. A border can also have a style such as solid, dashed, or dotted.
- **Padding**
  - is the passive white space between the content and the border of a block element.
  - Padding is typically measured in pixels (px), ems (em), or percentages (%).
- The most important thing about the box model is that by default its **box-sizing** property is set to **content-box**.
  - Any padding or border that's applied will not be added to the rendered width. Instead, it will automatically subtract from the space that's available in the content area.



Example

For example, consider the following <p>, which is wrapped inside a :

```
<div>
    <p>This is a paragraph. It is a very short paragraph.</p>
  </div>
```

We can apply the following CSS to the paragraph tag in order to control the size of the padding, border, and margin of the paragraph:

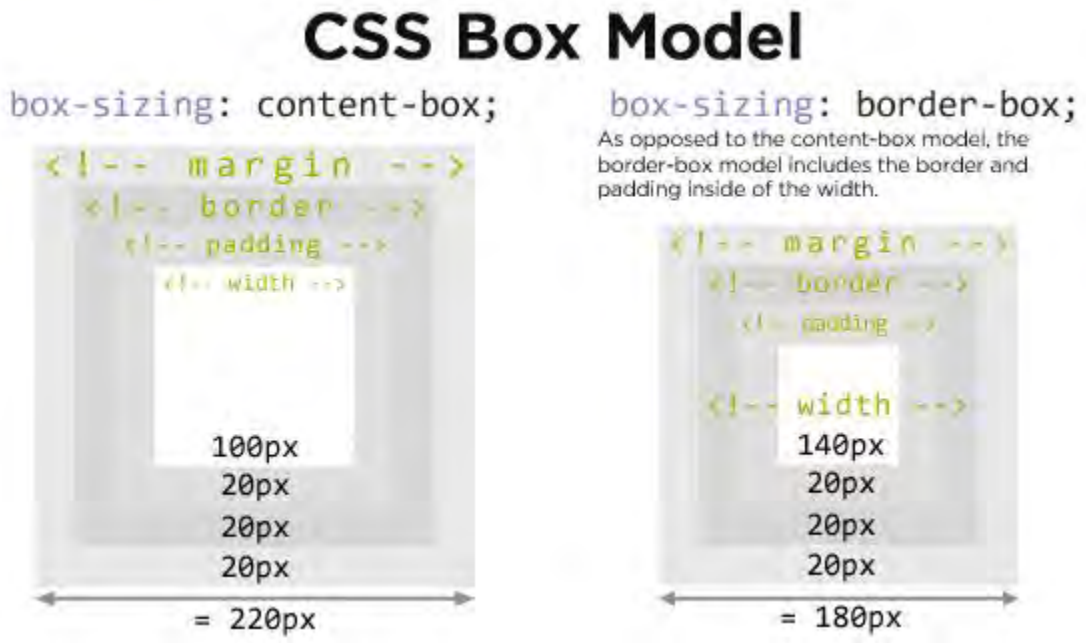
```
div {
    background-color: red;
    padding: 0;
    border: 1px solid black;
    margin: 0;
  }
  p {
    background-color: white;
    padding: 1em;
    border-width: 10px;
    border-style: solid;
    border-color: blue;
    margin: 10px !important;
   }
```

Output:

This is a paragraph. It is a very short paragraph.

11.1.a Common Box Model Properties

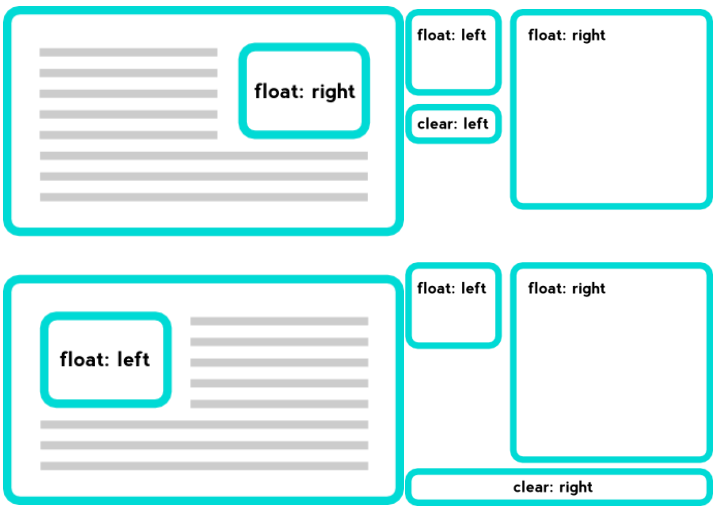
Property	Description	Examples
margin	Sets the amount of space around the block element (top, right, bottom, left)	margin: 20px; margin-top: 2em; margin-bottom: 150%;
padding	Sets the amount of space between content and the border of its block element	padding: 10px; padding-left: 1.5em; padding-right: 125%;
border	Sets the format of the block element's border	border: solid 1px #000000;
border-style	Designates the style of a border	border-top-style: solid; border-top-style: dotted;
border-width	Designates the width of a border	border-top-width: 1px; border-bottom-width: thick;
border-color	Designates the border color	border-top-color: #000000; border-bottom-color: gray;
border-radius	Rounds the corners of a block element's border	border-radius: 10px;
box-shadow	Adds a shadow to a block element's border	box-shadow: 8px 8px 8px #000000;



11.1.b. CSS Float

Float is a CSS property written in CSS file or directly in the style of an element. The float property defines the flow of content. Below are the types of floating properties:

float: left Element floats on left side of the container
float: right Element floats on right side of container
float: inherit Element inherits floating property of it's parent (div, table etc…)
float: none Element is displayed as it is (Default).



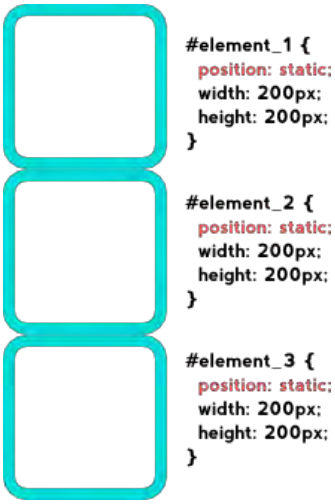
The 3 Rules

- Rule #1 Use float for its intended purpose
- Rule #2 Parents containing float elements will collapse
- Rule #3 Use clear in conjunction with float

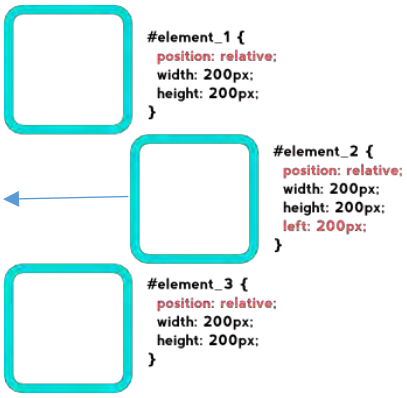
11.1.c. Positioning

The following gives basic information about the CSS position property and other related properties. The **position** CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

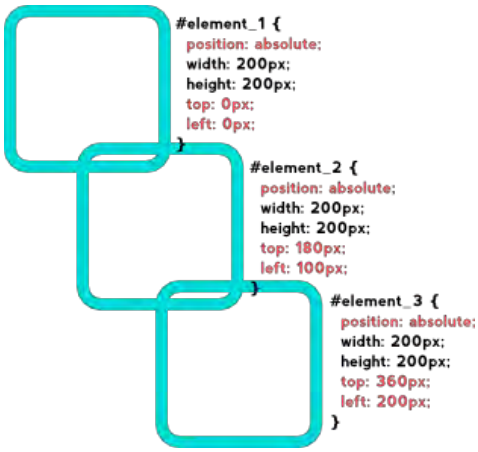
- **position:static**;
  - The element is positioned according to the normal flow of the document. The top, right, bottom, left, and z-index properties have no effect. This is the default value.



- **position:relative;**
  - The element is positioned according to the normal flow of the document, and then **offset** relative to itself based on the values of top, right, bottom, and left.
  - The offset does not affect the position of any other elements; thus, the space given for the element in the page layout is the same as if position were static.



- **position:absolute;**
  - The element is removed from the normal document flow, and no space is created for the element in the page layout.
  - It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block.
  - Its final position is determined by the values of top, right, bottom, and left.



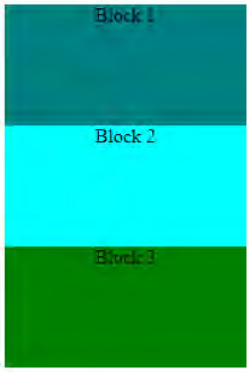
- **position: fixed;**
  - The element is removed from the normal document flow, and no space is created for the element in the page layout.

11.1.d. Display Property

The Display property in CSS defines how the components (div, hyperlink, heading, etc) are going to be placed on the web page. As the name suggests, this property is used to define the display of the different parts of a web page.

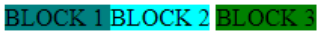
- **Display:block;**
  - Block: This property is used as the default property of div.
  - This property places the div one after another vertically.
  - The height and width of the div can be changed using the block property if the width is not mentioned, then the div under block property will take up the width of the container.

display: block; property



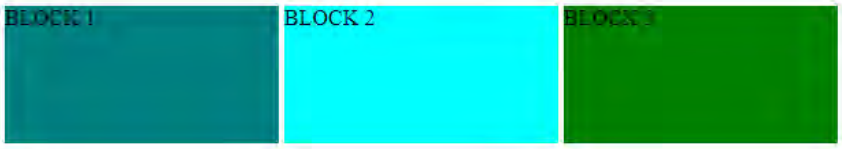
- **Display:inline;**
  - This property is the default property of anchor tags. This is used to place the div inline i.e. in a horizontal manner.
  - The inline display property ignores the height and the width set by the user.

display: inline; property



- **Display:inline-block;**
  - Inline-block: This features uses the both properties mentioned above, block and inline.
  - This property aligns the div inline but the difference is it can edit the height and the width of block.
  - Basically, this will align the div both in block and inline fashion.

display: Inline-block; property



UNIVERSITY OF Baguio

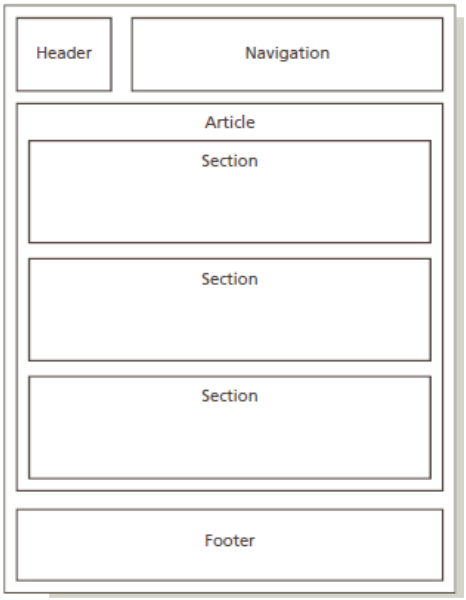
Name:		Date:	
Introduction to CSS: Box Model Layouts		Lesson 11 Exercise11	Score:

General Instructions

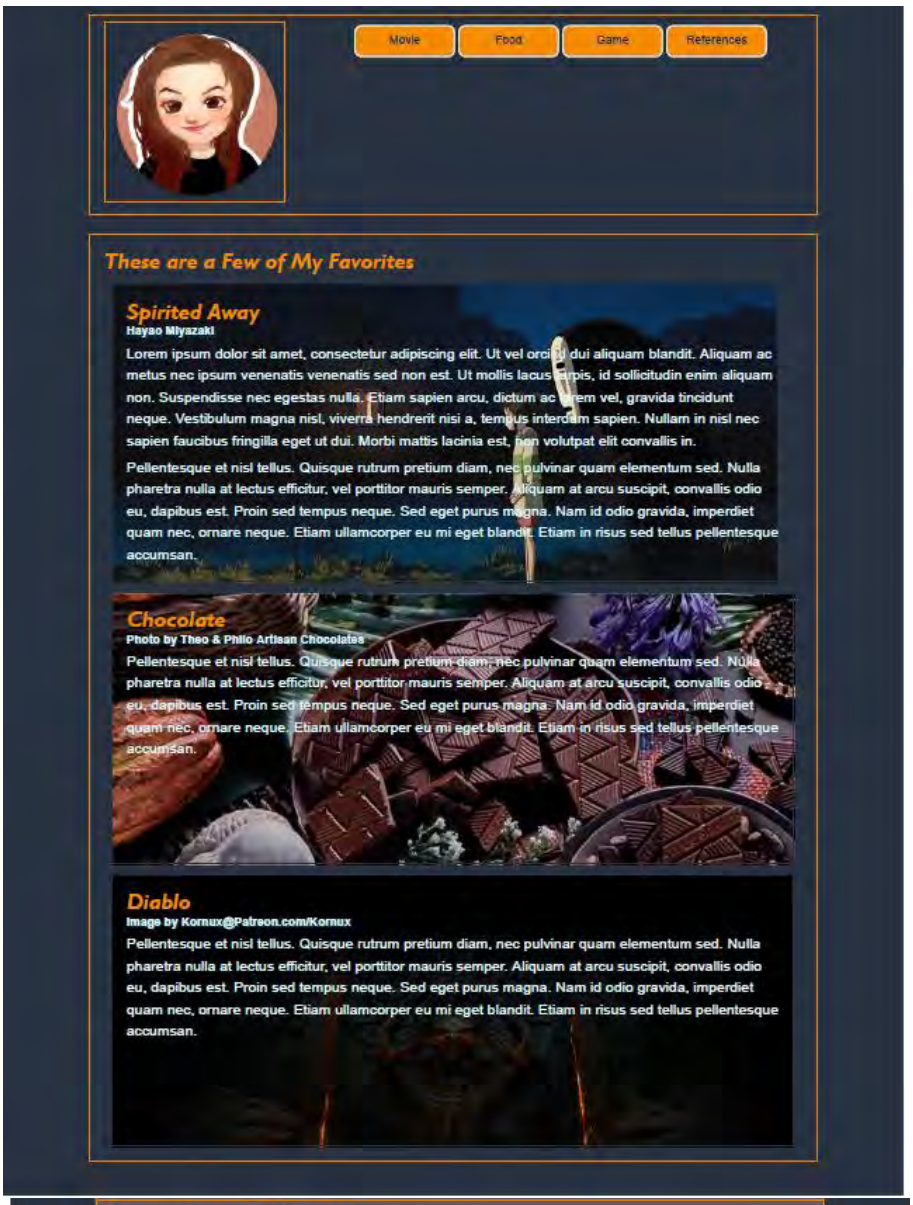
- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Midterm exercises in the Midterm directory.

Instructions:

1. Create a new css file and name it Exercise11.css in the styles directory.
2. Create a new webpage and save it as Exercise11.html.
3. Recreate the web page wireframe as shown in the image.
4. **Header** will contain a logo
5. The **navigation** should contain 4 links. 1 for each section in the page, and 1 for the footer.
6. The article will contain **3 sections** that discusses three of your favorite things/people in your life.
7. The footer should list all references you have used.
8. Add media content and make sure you cite all your references.
9. Use semantic and appropriate html elements for your page.



Sample Output


Movie
Food
Game
References
These are a Few of My Favorites
Spirited Away
Hayao Miyazaki
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vel orci ut dui aliquam blandit. Aliquam ac metus nec ipsum venenatis venenatis sed non est. Ut mollis lacus turpis, id sollicitudin enim aliquam non. Suspendisse nec egestas nulla. Etiam sapien arcu, dictum ac rutrum vel, gravida tincidunt neque. Vestibulum magna nisl, viverra hendrerit nisi a, tempus interdum sapien. Nullam in nisl nec sapien faucibus fringilla eget ut dui. Morbi mattis lacinia est, non volutpat elit convallis in.
Pellentesque et nisl tellus. Quisque rutrum pretium diam, nec pulvinar quam elementum sed. Nulla pharetra nulla at lectus efficitur, vel porttitor mauris semper. Aliquam at arcu suscipit, convallis odio eu, dapibus est. Proin sed tempus neque. Sed eget purus magna. Nam id odio gravida, imperdiet quam nec, ornare neque. Etiam ullamcorper eu mi eget blandit. Etiam in risus sed tellus pellentesque accumsan.
Chocolate
Photo by Theo & Philo Artisan Chocolates
Pellentesque et nisl tellus. Quisque rutrum pretium diam, nec pulvinar quam elementum sed. Nulla pharetra nulla at lectus efficitur, vel porttitor mauris semper. Aliquam at arcu suscipit, convallis odio eu, dapibus est. Proin sed tempus neque. Sed eget purus magna. Nam id odio gravida, imperdiet quam nec, ornare neque. Etiam ullamcorper eu mi eget blandit. Etiam in risus sed tellus pellentesque accumsan.
Diablo
Image by Kornux@Patreon.com/Kornux
Pellentesque et nisl tellus. Quisque rutrum pretium diam, nec pulvinar quam elementum sed. Nulla pharetra nulla at lectus efficitur, vel porttitor mauris semper. Aliquam at arcu suscipit, convallis odio eu, dapibus est. Proin sed tempus neque. Sed eget purus magna. Nam id odio gravida, imperdiet quam nec, ornare neque. Etiam ullamcorper eu mi eget blandit. Etiam in risus sed tellus pellentesque accumsan.
Resources and References
Credits to art owned by artist Rodngd Alexandrino
Credits to Theo & Philo Artisan Chocolates
Credits to Kornux@Patreon.com/Kornux

Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	4	3	0-2	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
CSS Implementation	Correctly implemented external CSS, no inline or internal styles used.	Implemented external CSS, with some inline or internal styles used.	Implemented external CSS from the wrong location/directory, no inline or internal styles used.	Did not implemented external CSS, used inline or internal styles used.	
Media elements	All of the photographs, graphics, sound and/or video enhance the content and create interest.	Most of the photographs, graphics, sound and/or video enhance the content and create interest.	A few of the photographs, graphics, sound and/or video are inappropriate for the content and do not create interest.	The photographs, graphics, sounds, and/or videos are inappropriate for the content or are distracting decorations that create a busy feeling and detract from the content.	
Fair Use Guidelines	Fair use guidelines are followed with proper use of citations throughout the Web page.	Fair use guidelines are frequently followed and most non original material uses proper citations.	Sometimes fair use guidelines are followed and some non-original material uses proper citations.	Fair use guidelines are not followed. Non original material is improperly cited.	
	10-9	8-7	6-4	3-0	
CSS Selectors, properties and values	The typography is easy to read, and point size varies appropriately for headings and text while maintaining a consistent style throughout.	Sometimes the typography is easy to read, but in a few places the use of fonts, point size, and color are inconsistent.	The typography is difficult to read and uses too many different fonts and colors, overuse of bold and/or underlines.	The text is extremely difficult to read due to inappropriate use of fonts, point size, italics, bold, and underlines.	
	The background, colors and layout are artful and consistent across the website and enhance the readability of the information presented.	The background, colors and layout are consistent across the website and make it easy to read the information presented.	The background, colors and layout are distracting and make it difficult to read the information presented	The background, colors and layout make the site unattractive, and it is difficult to read the information presented.	

UNIVERSITY OF Baguio

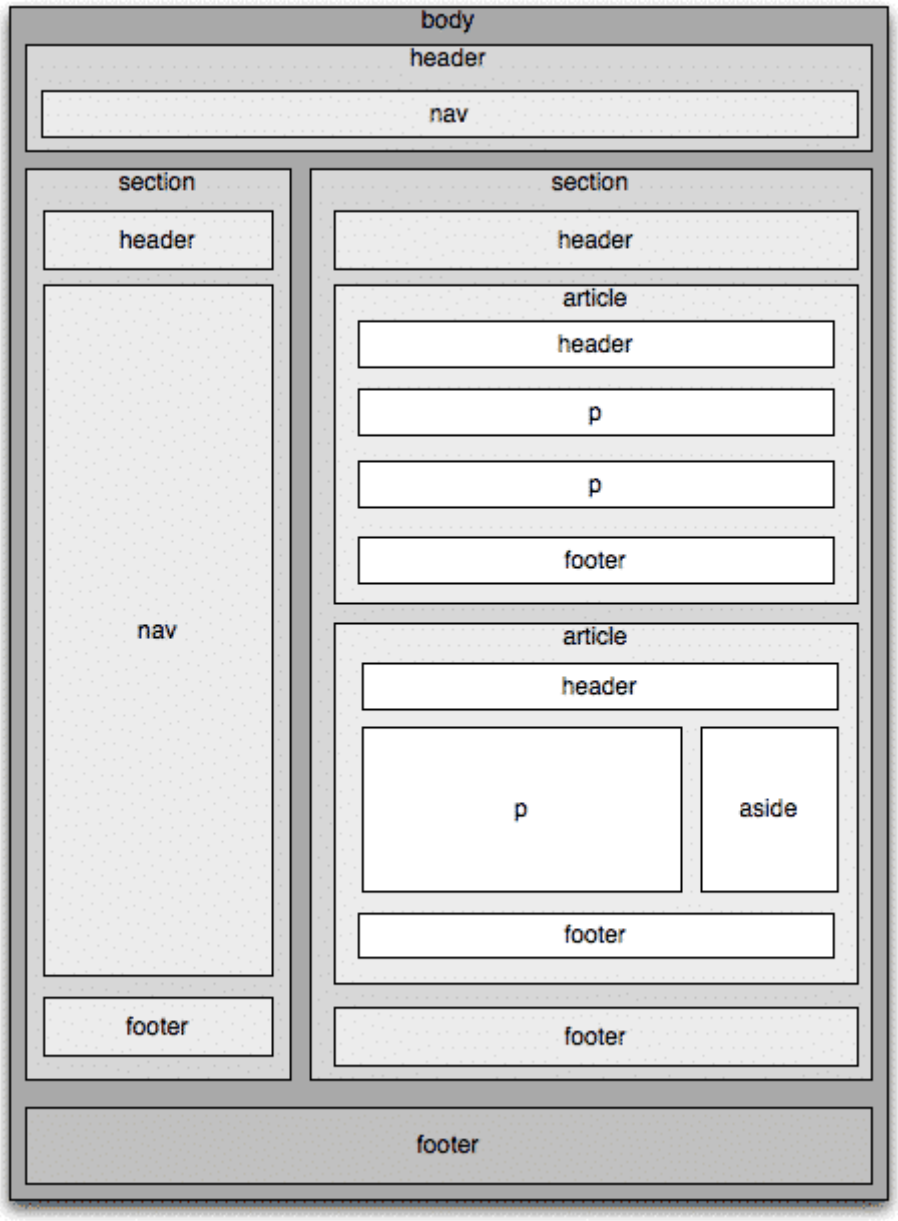
Name:		Date:	
Introduction to CSS: Box Model Layouts		Lesson 11 Exercise12	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Midterm exercises in the Midterm directory.

Instructions:

1. Create a new css file and name it Exercise12.css in the styles directory.
2. Create a new webpage and save it as Exercise12.html.
3. Recreate the web page wireframe as shown in the image.
4. You may use semantic elements or assign classes or ids to your html elements.
5. Define your own dimensions.



Criteria	Exemplary	Proficient	Partially Proficient	Unsatisfactory	Score
	5	**4**	**3**	**0-2**	
Code Validation	There are no errors in the HTML or other coding on the site as found by me or an online validator.	There are 1-3 coding errors on the site as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.	
Instructions • **Filename** • **File path** • **Page title** • **Meta tags** • **GitHub**	Followed all instructions. Implemented all of the instructions and/or requirements.	Followed most of the instructions. Implemented more than 5-6 of the instructions and/or requirements.	Followed some of the instructions. (3-4 of the instructions and/or requirements.)	Instructions were not followed/ implemented. (Less than 3 of the instructions and/or requirements.)	
CSS Implementation	Correctly implemented external CSS, no inline or internal styles used.	Implemented external CSS, with some inline or internal styles used.	Implemented external CSS from the wrong location/directory, no inline or internal styles used.	Did not implemented external CSS, used inline or internal styles used.	
	10-9	**8-7**	**6-4**	**3-0**	
CSS Selectors, properties and values	Correct choice of CSS box properties and values were used to achieve required layout.	Mostly correct choice of CSS box properties and values were used to achieve required layout.	Some CSS box properties and values were correct achieve required layout	Was not able to achieve required layout	

UNIVERSITY OF Baguio

LESSON 12: CSS Responsive Web

Duration: 1.0 hour

About this lesson: At the end of this lesson you are expected to:

1. Explain the role of different screen sizes and the need for the different layout approaches.
2. Create fixed and responsive layouts.
3. Differentiate between responsive and adoptive layout design.

12.1. Responsive

- The term responsive design was coined by Ethan Marcotte in 2010. (MDN Contributors, 2021)
- The layout changes based on the size and capabilities of the device. (LePage & Andrew, 2020)
- "Responsive layouts are also known as Responsive Web Design (RWD), while adaptive layouts as Adaptive Web Design (AWD)." (Carey, 2017, p. 365)

12.2. Three Primary Components of Responsive Design Theory

1. **Flexible Layout** - the page layout automatically adjusts to screens of different widths
2. **Responsive images** - rescale based on the size of the viewing device
3. **Media queries -** determine the properties of the device rendering the page so that appropriate designs can be delivered to specific devices.

12.3. Starting Responsive Design

1. **Set the viewport**
   - Pages optimized for a variety of devices must include a meta viewport tag in the head of the document.
   - A meta viewport tag gives the browser instructions on how to control the page's dimensions and scaling.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Responsive Web</title>
        <meta charset="utf-8">
        <meta name="author" content="E.Martinez" >
        <meta name="viewport"
              content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet"
              type="text/css"
              href="styles\responsivestyle.css"
```

   - Using the meta viewport value ***width=device-width*** instructs the page to match the screen's width in device-independent pixels.
   - Adding the value initial-scale=1 instructs browsers to:
     - *initial-scale*: Sets the initial zoom of the page, which we set to 1.0
     - *height*: Sets a specific height for the viewport.
     - *minimum-scale*: Sets the minimum zoom level.
     - *maximum-scale*: Sets the maximum zoom level.
     - *user-scalable*: Prevents zooming if set to no.
2. **Creating the Layouts for Different Screen Sizes**
   - Create a wire frame layout for the target device classes you want to design for.
   - The table below show the viewport sizes, also known as your CSS breakpoints, of common devices used by users you may use as a reference.
     - Mobile: 360 x 640
     - Mobile: 375 x 667
     - Mobile: 360 x 720
     - iPhone X: 375 x 812
     - Pixel 2: 411 x 731
     - Tablet: 768 x 1024
     - Laptop: 1366 x 768
     - High-res laptop or desktop: 1920 x 1080

Category	Code	Width	For
Extra Small	xs	<768px	Phones
Small	sm	>=768px	Small Devices
Medium	md	>=992px	Desktops
Large	lg	>=1200px	Large Desktops

3. **Use CSS media queries for responsiveness**

- Media queries are used to associate a style sheet or style rule with a specific device or list of device features.
- To create a media query within an HTML file, add the following **media** attribute to either the link or style element in the document head where **devices** is a comma-separated list of supported media types associated with a specified style sheet.

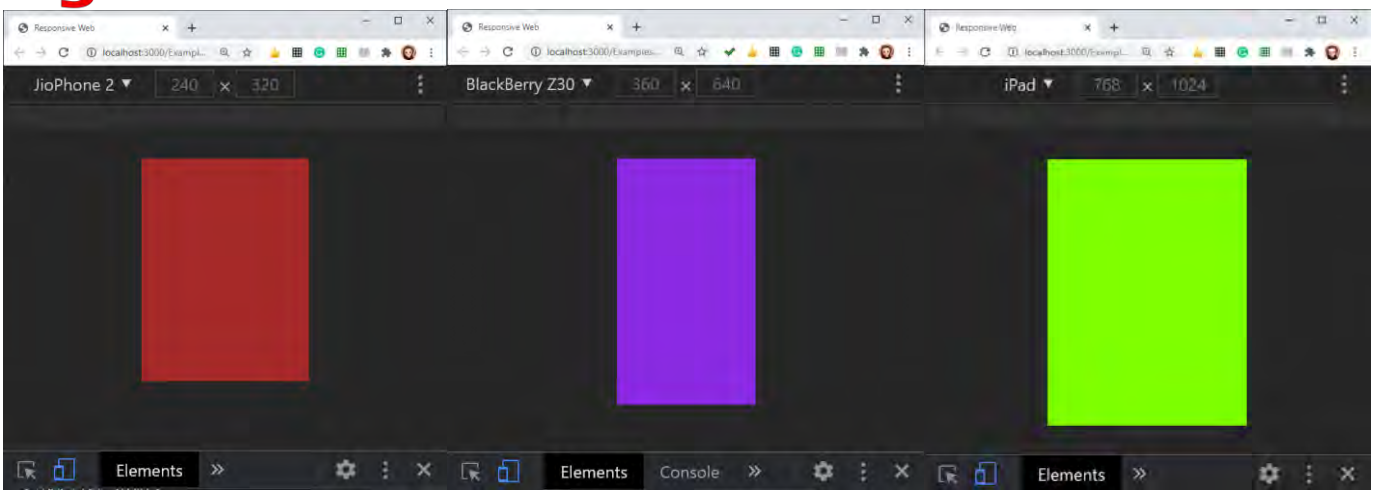
```
<link rel="stylesheet"
      type="text/css"
      href="styles\responsivestyle.css"
```

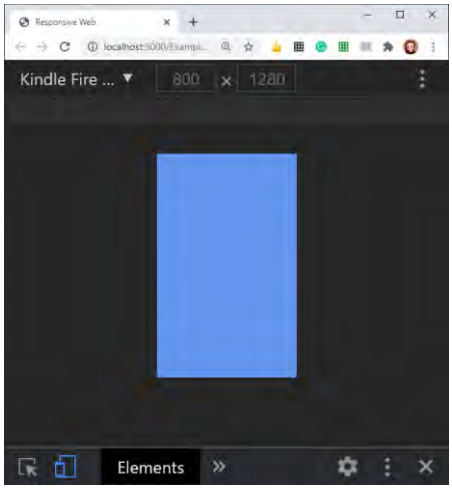
4. **The @media Rule**
   - Media queries can also be used to associate specific style rules with specific devices by including the following `@media` rule in a CSS style sheet file
   - use the @media rule to include a block of CSS properties only if a certain condition is true.

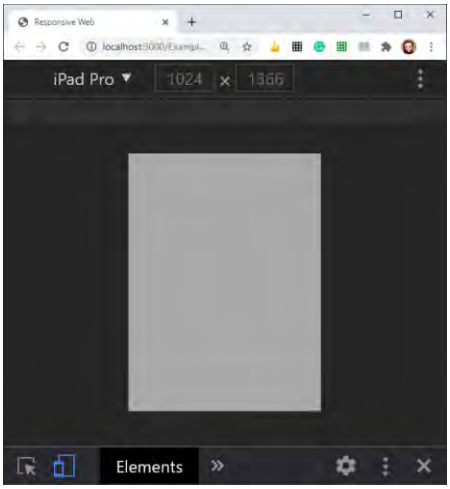
Example: This is the stylesheet linked to the HTML document. *responsivestyle.css*

```
*{
    width:100%;
}
@media screen and (max-width:1024px){
    body{
        background-color: darkgrey;
    }
}
@media screen and (max-width:980px){
    body{
        background-color: cornflowerblue;
    }
@media screen and (max-width:768px){
    body{
        background-color: chartreuse;
    }
}
@media screen and (max-width:481px){
    body{
        background-color: blueviolet;
    }
}
@media screen and (max-width:320px)
{
    body{
        background-color: brown;
    }
}
```

These are the outputs on a browser at the different breakpoints or screen widths







Breakpoints

- A breakpoint is a key to determine when to change the layout and adapt the new rules inside the media queries.
- common breakpoints for widths of devices:
    - 320px—480px: Mobile devices
    - 481px—768px: iPads, Tablets
    - 769px—1024px: Small screens, laptops
    - 1025px—1200px: Desktops, large screens
    - 1201px and more— Extra large screens, TV

12.4. PX vs EM vs REM vs Viewport Units for responsive design

- CSS has both absolute and relative units of measurement. An example of an absolute unit of length is a cm or a px.
- Relative units or dynamic values depend on the size and resolution of the screen or the font sizes of the root element.
    - PX – a single pixel (always use for text)
    - EM – relative unit based on the font-size of the element.
    - REM – relative unit based on the font-size of the element.
    - VH, VW – % of the viewport's height or width.
    - % – the percentage of the parent element. (best for image and element widths)

12.5. Build a Responsive Grid-View for a responsive layout (Option 1)

Step 1. use a responsive grid-view with 12 columns, to have more control over the web page.

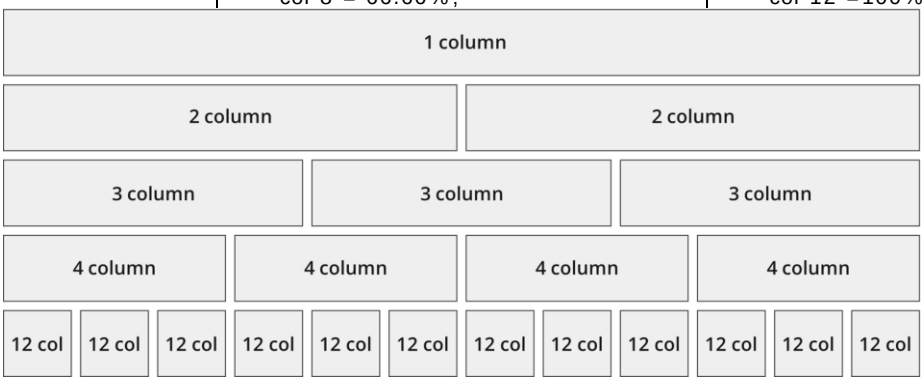
Step 2. calculate the percentage for one column: 100% / 12 columns = 8.33%.

Step 3. make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span:

col-1 = 8.33%;
col-2 =16.66%;
col-3 = 25%;
col-4 = 33.33%;

col-5 = 41.66%;
col-6 = 50%;
col-7 = 58.33%;
col-8 = 66.66%;

col-9 = 75%;
col-10 = 83.33%;
col-11 = 91.66%;
col-12 =100%;



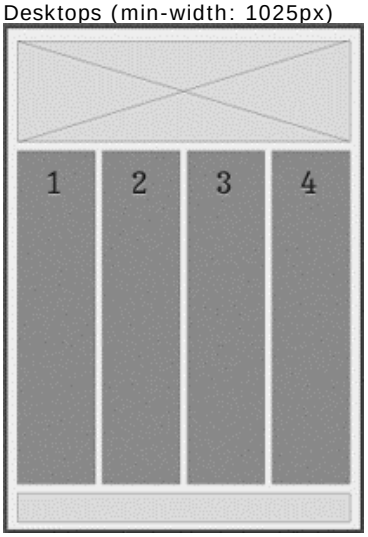
12.6. Create your HTML wireframe for the different viewport you want to design for

- Set your target devices and determine the breakpoints for these devices
- Draw your wireframe

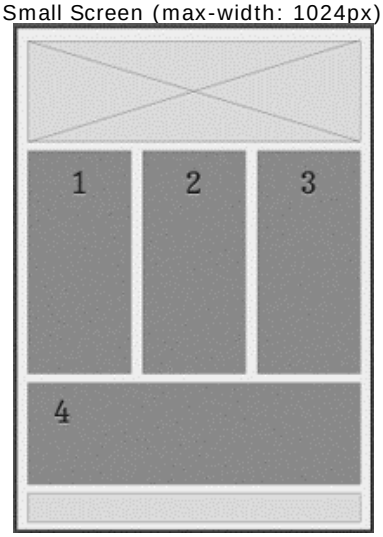
Example:

Layout for:

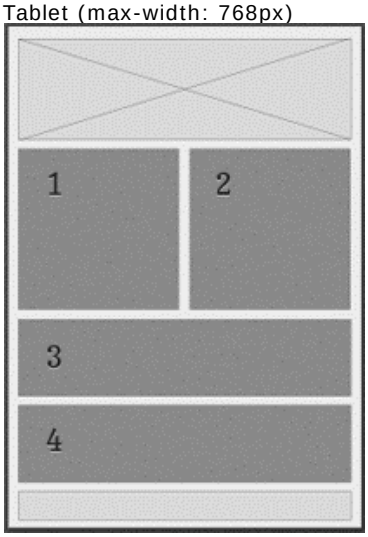
Desktops (min-width: 1025px)



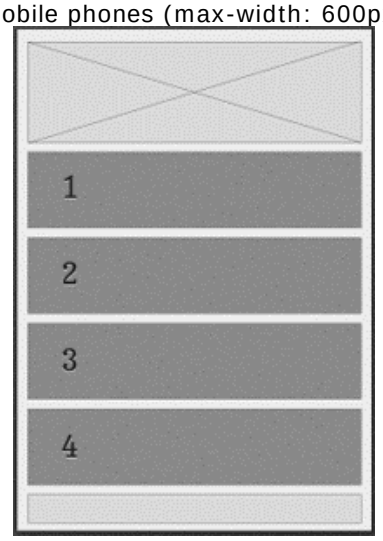
Small Screen (max-width: 1024px)



Tablet (max-width: 768px)



Mobile phones (max-width: 600px)



12.7. Code the HTML structure and add your classes for the columns.

The head

```
<!DOCTYPE html>
<html>
    <head>
        <title>Responsive Web With CSS Grid</title>
        <meta charset="utf-8">
        <meta name="author" content="E.Martinez" >
        <meta name="viewport"
              content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet"
              type="text/css"
              href="styles\cssgrid.css">
    </head>
```

The body

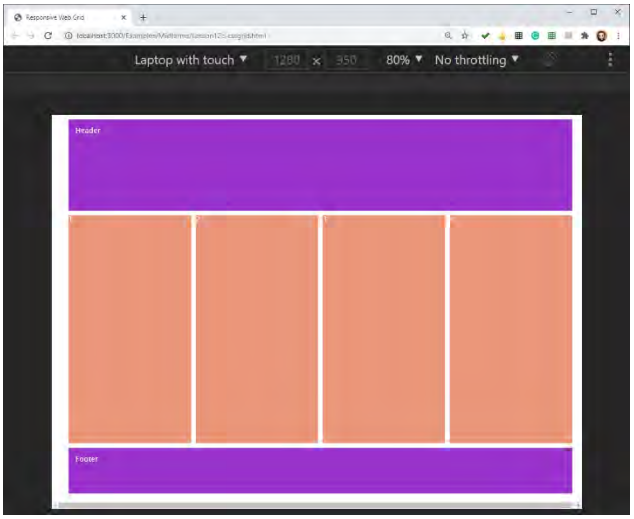
```
    <body>
        <div id="wrapper">
            <header>
                Header
            </header>
            <div class="grid-container">
                <div class="col-1 col1">
                    1
                </div>
                <div class="col-1 col2">
                    2
                </div>
                <div class="col-1 col3">
                    3
                </div>
                <div class="col-1 col4">
                    4
                </div>
            </div>
            <footer>
                    Footer
            </footer>
        </div>
    </body>
    </html>
```

12.8. Build a Responsive Grid-Layout for a responsive layout (Option 2)

Step 1. Use the CSS Grid layout which is similar to the fluid grid but with its own built-in property and values.

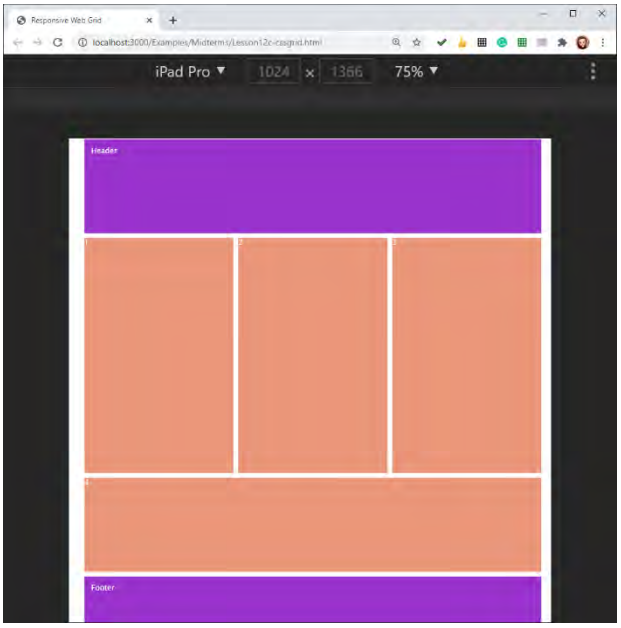
Step 2. For viewports larger than 1024px you don't need to add @media rules

```css
*{
    box-sizing: border-box;  color: #ffff;
    font-family: 'Lucida Sans', 'Lucida Sans Regular',  sans-serif;
}
body{ width: 100%; }

#wrapper, header, footer{
    width:95%;  margin: 10px auto;
}
header, footer{
    height: 200px;
    background-color:  #9933cc;
    width: 100%;
    padding: 15px;
}
.grid-container{
    width: 100%;
    display: grid;
    grid-template-columns: auto auto auto auto;
    grid-gap: 10px;
}
.grid-container>.col-1{
    height: 500px;
    background-color: darksalmon;
}
footer{  height: 100px; }
```



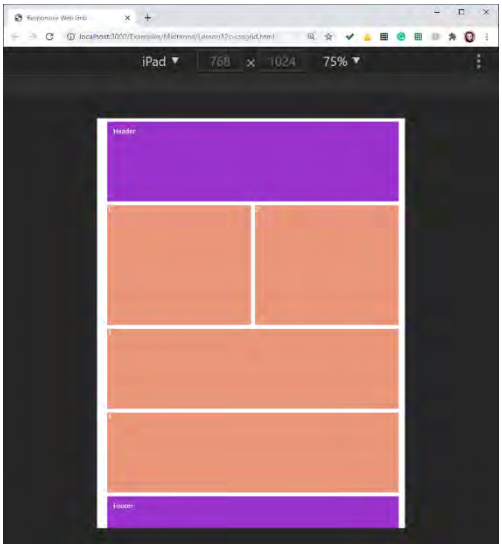
Step 3. Insert @media rule for next breakpoint

```css
@media screen and (max-width:1024px){
    .grid-container{
        width: 100%;
        display: grid;
        grid-auto-flow: row;
        grid-template-columns: auto auto auto;

    }
    .grid-container>.col1, .grid-container>.col2, .grid-container>.col3{
        height: 500px;

        background-color: darksalmon;
    }
    .grid-container>.col4{
        height: 200px;
        grid-column-end: span 3;
    }
```



Step 4. Proceed with the next breakpoint

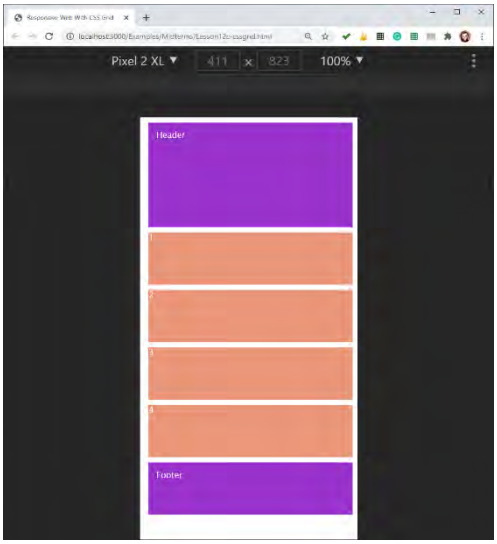
```css
@media screen and (max-width:768px){
    .grid-container{
        width: 100%;
        display: grid;
        grid-template-columns: auto auto auto auto;
        grid-gap: 10px;
    }
    .grid-container>.col1, .grid-container>.col2{
        height: 300px;
        grid-column-end: span 2 ;
        background-color: darksalmon;
    }
    .grid-container>.col3, .grid-container>.col4{
        height: 200px;
        grid-column-end: span 4 ;
    }

}
```



Step 5. And proceed with the last breakpoint

```
@media screen and (max-width:600px){
    .grid-container{
        width: 100%;
        display: grid;
        grid-template-columns: auto auto auto auto;
        grid-gap: 10px;
    }
    .grid-container>.col-1{
        height: 200px;
        grid-column-end: span 4 ;
        background-color: darksalmon;
    }
}
```

Step 6. Test your output.



UNIVERSITY OF Baguio

Name:		Date:	
Introduction to CSS: Responsive Layouts		Lesson 12 Exercise13	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Midterm exercises in the Midterm directory.

Instructions:

1. Create a new css file and name it Exercise13.css in the styles directory.
2. Create a new webpage and save it as Exercise13.html.
3. Recreate the web page wireframes for the three breakpoints for mobile, tablet, and desktop screen displays shown in the image below.
    - The webpage should automatically change layout when the viewport is reduced to the breakpoint range.
    - You may use semantic elements for your containers.
    - Assign classes and/or ids to your html elements.
    - Define your unknown dimensions.

UNIVERSITY OF Baguio

Lesson 13: Introduction to JavaScript

Time: 3 hours

About this lesson: This lesson will introduce you to the fundamentals of JavaScript. It will allow you to add interactive features to your website. You will learn introductory code concepts which are essential knowledge for future web developers.

Objectives:

In this lesson, you are expected to:

1. Understand the use of JavaScript in web development
2. Define JavaScript

13.1. What is JavaScript?

- High level, interpreted programming language
- Conforms to the **ECMAScript** specification
- Runs on the client/browser as well as on the server
- Was created by Brendan Eich in 1995

13.2. Why Learn JavaScript?

- it is one of programming language used on a web browser
- builds very interactive user interfaces with JavaScript frameworks
- used in building very fast server side (back-end) and full stack applications
- used in mobile development
- used in desktop application development

13.3JavaScript Implementation

1. The **<script> </script>** tag
   - The JavaScript code must be inserted between the **<script>** and **</script>** tags.
   - **Example**

```
<script>

// Javascipt code goes here

</script>
```

Tells the browser that the java script code starts here

Tells the browser that the java script code ends here

2. JavaScript in the <head> </head> or <body> </body>
   - You can place any number of scripts in the <head> or <body> section of the html document
   - Placing the scripts at the bottom of the <body element improves the display speed, because scripts interpretation slows down the display.

   Example

```
<script type="text/javascript">

//javascript code goes here

</script>
```

Tells the browser that the language will be JavaScript

3. External JavaScript
   - Are practical when the same code is used in many different web pages
   - Has some advantages
     - ✓Separates html and code
     - ✓Makes html, css, and JavaScript easier to read and maintain
     - ✓Cached js files can speed up page loads

   Example

Tells the browser that the directory and name of the JavaScript file

```
<script type="text/javascript" src="js/yourfile.js">

//javascript code goes here

</script>
```

13.4JavaScript Syntax

White space

- Are added for readability.

Case sensitive

- JavaScript is **case-sensitive**
- a variable named 'aName' is different than 'Aname'

Identifiers

- an *identifier* is a sequence of characters that can be used to identify a variable, a function, or an object.
- It may start with a letter, the dollar sign ($), an underscore (_), and may contain digits (0-9)
- Reserved words (like JavaScript keywords) cannot be used as identifiers

Semicolons

- Statements are ***optionally*** terminated with a *semicolon (;)*

How to display data with JavaScript?

1. **innerHTML**
   - **innerHTML** property is used to get or set the HTML content of an element that will allow the JavaScript code to manipulate a webpage content being displayed.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using innerHTML</title>
    </head>
    <body>
        <h2 id="target">Hello</h2>
        <script>
            document.getElementById("target").innerHTML="Goodbye";
        </script>
    </body>
</html>
```

getElementById method will access the HTML element with id="target"

The innerHTML property defines the HTML content = "Goodbye"

Output (Try out the code yourself and write the output in the box below.)

2. **document.write()**

- **document.write** method writes a string of text in a HTML document. Using document.write() after a HTML document is loaded will delete all its content. It is mostly used for testing purpose.

Example1:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using document.write</title>
    </head>
    <body>
        <h2>Hello</h2>
        <h2>How are you today?</h2>
        <script>
            document.write("Goodbye");
        </script>
    </body>
</html>
```

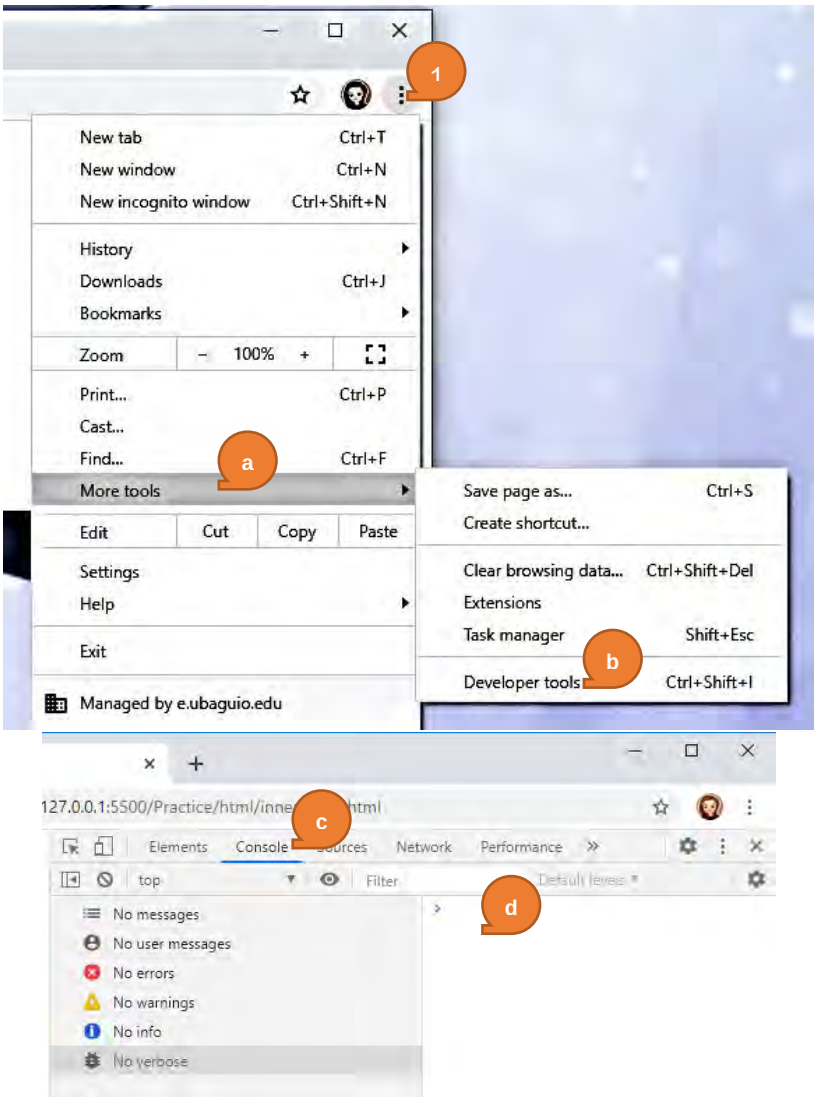
Output (Try out the code yourself and write the output in the box below.)

Example2

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using document.write</title>
    </head>
    <body>
        <h2>Hello</h2>
        <h2>How are you today?</h2>
        <button type="button" onclick="document.write('Have a nice day!');">
                Click Here
        </button>
    </body>
</html>
```

Output (Try out the code yourself and write the output in the box below.)

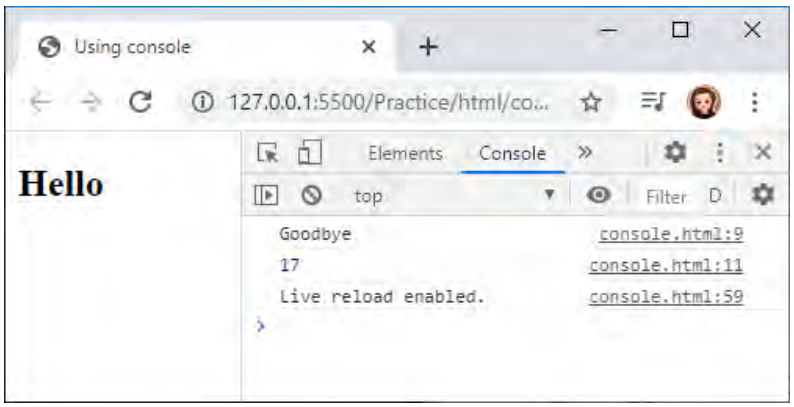
3. **console.log()**
   - a method used to write a message to the browsers' debugging console which is useful for testing.
   - To open the browser console press *ctrl*+*shif*+i (for chrome browser this will open the Developer tools), then go to console or
   - From the menu open (a)More Tools>(b)Developer Tools



Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using console</title>
    </head>
    <body>
        <h2 id="target">Hello</h2>
        <script>
            console.log("Goodbye");
            //We can do computations too!
            console.log(5+12);
        </script>
    </body>
</html>
```

Output

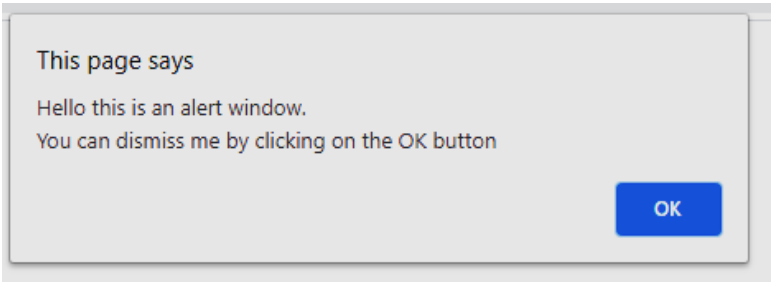


4. **alert(), confirm(), and prompt()**
   - You can use an alert box to display data

   Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using alert</title>
    </head>
    <body>
        <h2 id="target">Hello</h2>
        <script>
            alert("Hello this is an alert window.
      \nYou can dismiss me by clicking on the OK button");
        </script>
    </body>
</html>
```

Output



JavaScript Comments – are statements that will not be executed by the interpreter, these are used to mark annotations of what your code does to make it easier for others to understand.

a. **Single line comment** will comment out a single line from your code by preceding the comment with two forward slashes //

   Example:

```
let m1 = "Hello";
      let m2 = "World";

      // msg equals "Hello World"
      let msg = m1 + " " + m2;
```

b. **Multi-line comments** will comment out multiple lines and is denoted with /* to begin the comment, and */ to end the comment.

Example:

```
/*
            This is all commented
            let m1 = "Hello";
            let m2 = "World";
            let msg = m1 + " " + m2;
            None of this is going to run!
            */
```

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to JavaScript		Lesson 13 Quiz	Score:

After reading, answer the following questions. **Encircle** the *letter* of your choice from the given list of options.

1. What is the purpose of the <script> and </script> tags in a HTML document?
    a. It tells the browser where a JavaScript begins and ends in the HTML document
    b. To tell the browser the scripting language to be used
    c. To point to a scripting language file
    d. All of the above
2. When would it be a good idea to use an external JavaScript file?
    a. If the script is short and/or will just be used by one HTML document
    b. When the target audience use older browsers
    c. When the script is very long and will be used by more than one HTML document
    d. External JavaScript files are not recommended in general
3. JavaScript is case-sensitive
    a. Yes
    b. No
4. All JavaScript statements are terminated with a semicolon (;).
    a. Yes
    b. No
5. An *external JavaScript* file uses a filename extension of _____?
    a. .html
    b. .java
    c. .css
    d. .js
6. Which of the following html <script> attributes correctly points you to an external JavaScript?
    a. type
    b. src
    c. link
    d. language
7. What JavaScript method is used to output a string of text on a Web page?
    a. document.write()
    b. document.print()
    c. document.type()
8. A method property used to get / set the HTML content of an element that will allow the JavaScript code to manipulate a webpage content being displayed.
    a. getElementById
    b. log
    c. innerHTML
    d. print
9. Which of the following items would you use to add a single line of comment in your JavaScript code?
    a. /*
    b. /-
    c. //
    d. <!—
10. Which of the following items indicates that a multiline of comment is added in a JavaScript code?
    a. /* */
    b. /-
    c. //
    d. <!--

UNIVERSITY OF Baguio

Name:		Date:	
Introduction to JavaScript		Lesson 13 Exercise14	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Finals exercises in the Finals directory.
- Perform the set of activity on your computer. Print the HTML code and submit on the set deadline. (Modular Students)

1. (10 Points each) Create new HTML documents with **Internal script** that will display your complete name, course, year, subject code and section, and subject description. Use the four JavaScript methods (innerHTML, document.write(), console.log, and alert()) to display information on a browser. Save the html documents follows:
    a. Exercise14a.html (innerHTML)
    b. Exercise14b.html (document.write)
    c. Exercise14c.html (console.log)
    d. Exercise14d.html (alert())

    Example for the alert() is shown below.

.

2. (10 Points each) Create new HTML and **External JavaScript** documents that will display your complete name, course, year, subject code and section, and subject description. Use the four JavaScript methods (innerHTML, document.write(), console.log, and alert()) to display information on a browser. Save the html documents follows:
    a. Exercise14a.html and 2a.js (innerHTML)
    b. Exercise14b.html and 2b.js (document.write)
    c. Exercise14c.html and 2c.js (console.log)
    d. Exercise14d.html and 2d.js (alert())

Lesson 14: Variables and Types

Time: 3 hours

About this lesson: This lesson will introduce you JavaScript's **var**, **let**, and **const** keywords. You will learn how to create and/or declare variables correctly with the appropriate JavaScript keywords.

Objectives:

In this lesson, you are expected to:

- Understand variables
- Properly create or declare variables
- Define variable types
- Use variables

14.1Variables

In programming variables are containers for different kinds of values. You may create a name for your variable as well as assign values for that variable.

A *variable* is a container for a value, like a number, or a string whose values may or may not change depending on how they are declared (JavaScript First Steps, 2020).

What you'll need

1. **Lesson14.html** – create a new html document in the Finals folder and copy the code shown below.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Using Variables</title>
    </head>
    <body>
        <h2 Using Variables</h2>
        <script src="../js/main.js">
        </script>
    </body>
</html>
```

2. **main.js** create a new JavaScript document in the Finals/js folder in your Finals folder and perform the practice codes in the following sections.

14.2Variable Naming

There are two limitations on variable names in JavaScript:

1. The name must contain only letters, digits, or the symbols `$` and `_`.
2. The first character must not be a digit.

Examples of valid names:

```
let userName;

let test123;
```

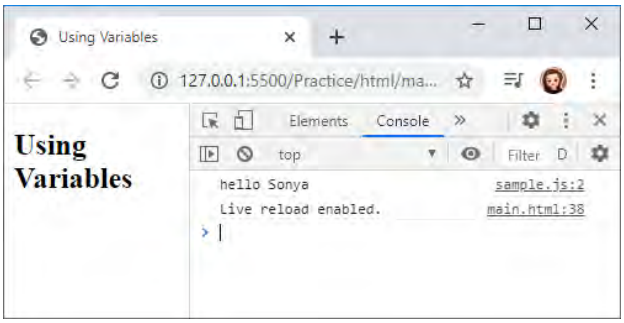
14.3var, let, and const

14.3.1var

- declarations are globally scoped
- A *variable* hold reusable data, which stored in memory, in a program and associate it with an *identifier*.
- var keyword is a pre-ES6 version of JavaScript used to declare a variable.

Example(main.js) Output

```
var myName='Sonya';

        console.log("hello "+myName);
```

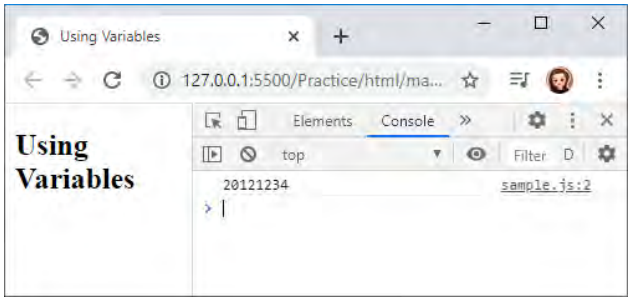


14.3.2const

- defines a constant reference to a value
- Variables declared via **const** are *immutable (***const** variables can neither be updated nor re-declared); You must always initialize immediately.
- you may not *reassign* a new value to a *const*

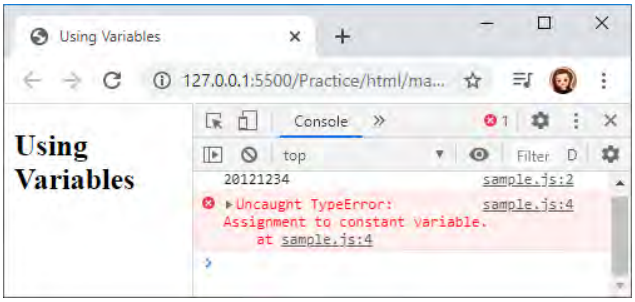
Example Output

```
const idNumber='20121234';
        console.log(idNumber);
```



Example Output

```
        const idNumber='20121234';
        console.log(idNumber);
        /*when the value of idNumber
        is updated */
        idNumber='210323432';
```

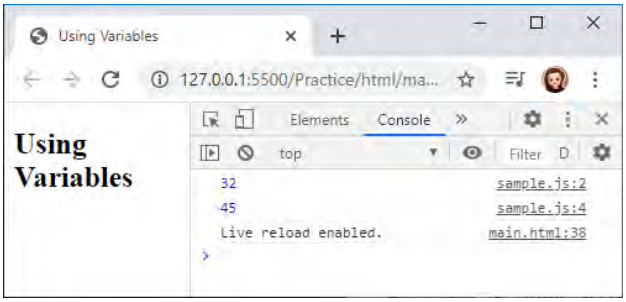


14.3.3let

- **let** and **const** was added in ES6 (ECMAScript); have a block level scope
- **let** allows you to re-assign values within its scope;
- variables declared with **let** are *mutable*
- **let** variables can be updated but not re-declared;

Example(main.js) Output

```
   let age = 32;
                    console.log(age);
/* when the value of age is
            re-assigned
                age = 45;

                 console.log(age);
```



14.4Variable Scope

Scope – determines the accessibility (visibility) of variables

a. Local variable

- variables declared within a JavaScript function, become LOCAL to the function
- local variables have **Function scope**: They can only be accessed from within the function
- variables defined inside a function are not accessible (visible) from outside the function.
- variables with the same name can be used in different functions.
- Local variables are created when a function starts, and deleted when the function is completed.

b. Global variable

- A variable declared outside a function, becomes GLOBAL.
- A global variable has global scope: All scripts and functions on a web page can access it.

(JavaScript Scope, 2020)

Example

```
// Initialize a global variable
var species = "human";

function transform() {
  // Initialize a local, function-scoped variable
  var species = "werewolf";
  console.log(species);
}

// Log the global and local variable
console.log(species);
transform();
console.log(species);
```

14.5Types

A. Primitive Data Types

- **String** – may contain letters, words, spaces, numbers, symbols but it must be enclosed in either single or double quotes.

```
const name='Dolly' //string
```

- **Numbers** – can be written with or without decimals; does not need to be declared as integers as integers or floating-point (decimal) numbers.

```
const age=55; //integer number

const length=16.75;  //float number
```

- Long numbers can be declared in exponential notation

```
let bignumber=4.7e5;  //exponential float number
```

- **Boolean** – one with either a value of true or false.

```
const isCool=true;
```

- **null** – it means that the variable has no value.

```
const x=null;
```

- **undefined**

```
const x=undefined; //explicitly defined

let z;  // undefined
```

Try this code (main.js):

```
const name = "Dyanna"
            const age  = "18"
            const subj = "APPDEV1"

            //concatenation
            console.log('I am ' + name + ' and I am ' + age + ' learning ' + subj +'.');
            //template string
            console.log(`I am ${name} and I am ${age} learning ${subj}.`);
            //or assign it to a variable
            const hello=`I am ${name} and I am ${age} learning ${subj}.`;
            console.log(hello);
```

B. Other Types

- **Arrays** –written with brackets
  - is a type of object used for storing multiple values in single variable.
  - Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays.

```
var colors = ["red", "white", "green"] //array

var x=colors.length; //the length property returns the number of array elements

myArray[0]=Date.now;
```

- **Objects** – written with curly brackets
  - is a complex data type that allows you to store collections of data.
  - contains properties, defined as a key-value pair.

```
var student ={id:"2012121", fn:"Linda", ln:"Santos"}; //object
```

Summary:

This lesson introduced you to variables, a powerful concept you will use in your future programming endeavors.

Let's review what you've learned:

- ✓ We can declare variables using var, let, and const
- ✓ Variables hold reusable data in a program and associate it with a name.
- ✓ Variables are stored in memory.
- ✓ The var keyword was used in pre-ES6 versions of JavaScript.
- ✓ Let is the preferred way

UNIVERSITY OF Baguio

Name:		Date:	
Variables and Types		Lesson14 Quiz	Score:

After reading, answer the following questions. **Encircle** the *letter* of your choice from the given list of options.

1. Which of the following type of variable is visible only within a function where it is defined?
    a. global variable
    b. local variable
    c. functional variable
    d. none of these
2. When naming variables in JavaScript, which of the following items is correct?
    a. The name must contain only letters, digits, or the symbols $ and _
    b. The name must contain only letters and digits
    c. The name must contain only letters, and the symbols %, #, $ and _
    d. The name must contain only letters, digits, and start with digits and the symbols $ and _
3. What symbol is used to separate multiple variables in one line?
    a. Period .
    b. Colon :
    c. Semicolon ;
    d. Comma ,
4. Which of the following is not a correct method of declaring a variable in JavaScript?
    a. var
    b. int
    c. const
    d. let
5. What will be the output of the following code:

```
<body>
    <h2 id="target">Loops</h2>
    <h3>Student</h3>
    <p id="stdnt"></p>
    <script>
        let a = 12;
        a = "12";
        console.log("a is of type " + typeof(a));
    </script>
</body>
```

    a. a is of type typeof12
    b. a is of type number
    c. a is of type string
    d. error
6. To declare a variable, which of the following reserved words should you use?
    a. var, let, const
    b. for
    c. function
    d. switch
7. JavaScript reserved words can be used as variable names.
    a. True
    b. False
8. Variable names are case sensitive.
    a. True
    b. False
9. Which of the following items would be a valid variable name in JavaScript?
    a. #Name
    b. @Name
    c. ()Name
    d. $Name
10. What data type is *emp* in the given declaration: var emp={name:"Daniel", age:15, gender:"male"};?
    a. array
    b. string
    c. object
    d. none of these

UNIVERSITY OF Baguio

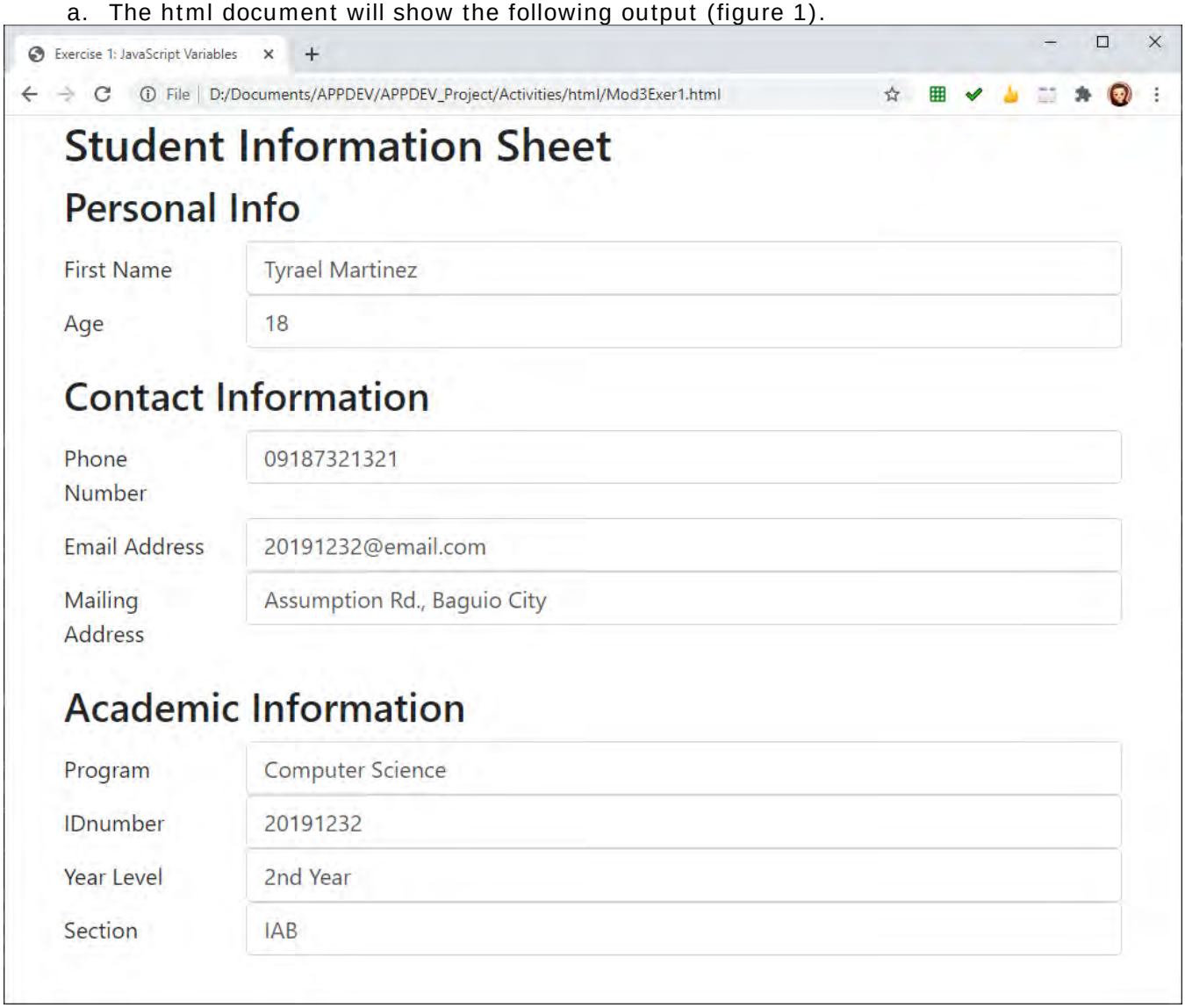
Name:		Date:	
Variables and Types		Module 14 Exercise15	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Finals exercises in the Finals directory.
- Perform the set of activity on your computer. Print the HTML code and submit on the set deadline. (Modular Students)

Perform the set of exercise on your computer. Write the output on the space provided for your output.

1. (25 points) Create a new HTML document named Exercise15.html. Add an external link to an external script file named Script15.js.
   a. The html document will show the following output (figure 1).



   b. In the external JavaScript file, declare the necessary variable/s and the appropriate type/s t for all the data displayed on the webpage shown. You may assign your own values to your variables.
   c. Output the assigned values on the webpage as shown in the figure above.
   d. Save and upload your files on your shared google drive.

2. (15 points) Create a new HTML document named Exercise16.html. Add an internal script that will perform the following:
    a. On one single line, declare three variables with the following names and assign your own first name, last name and age for the values: (Display the variable name and value on the console)
        - first_name
        - last_name
        - age
    b. Create a single variable that will store the same details from the previous item (letter c).
    c. Create a variable that will store the days of the week. (Monday to Sunday).
2. (15 points) Create a new HTML document named Exercise17.html. Add an internal script that will perform the following. Answer the corresponding questions after each item. Write your answer on the space provided.
    a. Add the following statements in the script. Launch the html document and describe the output.

```
var mytext='Here is \nsome text!';

        mytext= Today is another busy day!';

        console.log(mytext);
```

    b. What is the output? Explain.

    _____

    _____

    _____

    c. Replace **var** with **let**. Save the html document and refresh the web page. Describe the output.

    _____

    d. Replace **let** with **const**. Save the html document and refresh the web page. Describe the output. Explain

    _____

    _____

    _____

    _____

UNIVERSITY OF Baguio

Lesson 15: Expressions and Operators

Time: 6 hours

About this lesson: This lesson will introduce and define JavaScript **statements, expressions** and **operators**. This lesson will distinguish between JavaScript's statements and expressions, which are very important in several JavaScript frameworks you may want to learn later.

Objectives:

In this lesson, you are expected to:

1. Differentiate the operator types
2. Identify and use the Mathematical operators
3. Identify and use Assignment operators
4. Identify and use Logical operators and other JavaScript operators

15.1Statements

- A statement is a snippet of code that performs an action.
- Composed of values, operators, expressions, keywords, and comments
- Each statement is terminated by a semicolon

Examples

```
// 'if' statement
      if (score  !== 0) { score = 0 }
      // 'for' is a statement, it does not evaluate to a value
      for (;;) console.log('Sayonara!'); }
      // declaring a variable is statement
      let score = 0;
      score++; // this is a statement
```

15.2Expression

- A unit of code that can be **evaluated** to a **value**.
- Can be simple or complex

1. **Arithmetic Expressions**
   - are expressions that take a variable and an operator and result in a number
2. **String Expressions**
   - are expressions that result in a string
3. **Logical Expressions**
   - make use of logical operators and resolve to a Boolean value

15.3Operators

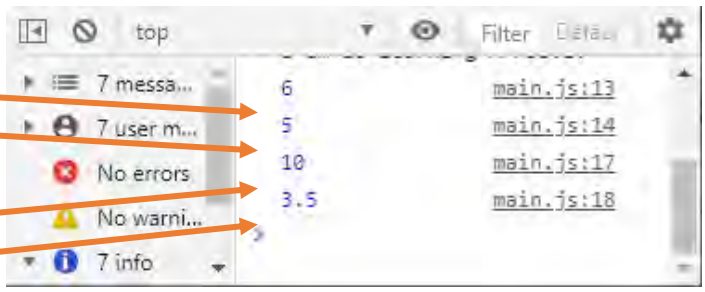
Operators allow you to get two simple expressions and combine them to form a more complex expression. An *operator* is a symbol or word in JavaScript that performs calculations, comparisons, or assignments on one or more values.

a. Mathematical –> used to perform mathematical or arithmetic operations.

Operator	Symbol	Function
Addition	+	Adds two values
Subtraction	-	Subtracts one value from another
Multiplication	*	Multiplies two values
Division	/	Divides one value by another
Modulus	%	Divides one value by another and returns the remainder
Increment	+ +	Unary operator that returns the value incremented
Decrement	--	Unary operator that returns the value decremented
Unary	-	Returns a negative number

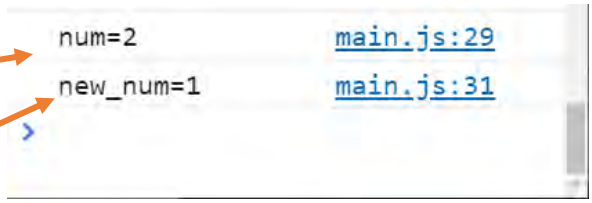
Example

```
console.log(1+5);
console.log(1*5);
let x = 2.5;
let y = 6;
console.log(25/x);
console.log(y-x);
```




```
alert(5%2);
```


```
var num=2;
console.log(`num=${num}`);
var new_num=--num;
console.log(`new_num=${new_num}`);
```

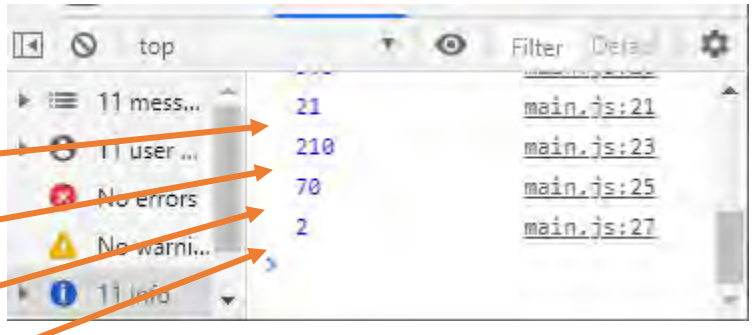


b. Assignment –> used to assign values to variables.

- = (assignment) : assigns right operand value to the left operand,
- += (add and assign): Sums up left and right operand values and assigns the result to the left operand;
- -= (subtract and assign): subtract right operand value from the left operand value then assign the result to the left operand,
- *= (multiply and assign): multiply the left and right operand values and assign the product to the left operand,
- /= (divide and assign): divide the left operand value by the right operand value, then assign the result to the left operand,
- %= (modulus and assign): get the modulus of the left operand by dividing the right operand and assigning the result to the left operand:

Example

```
let x = 2.5;
let y = 6;
y+=15; //y=y+15
console.log(y);
y*=10; //y=y*10;
console.log(y);
y/=3; //y=y/3
console.log(y);
```



```
y%=4; //y=y%4
console.log(y);
```

c. **Comparison** – used to compare two values, two variables, or two statements.

Name	Symbol	Description
equal to	==	Compare the equality of two operands w/o considering the type
not equal to	!=	Compare inequality of two operands
greater than	>	Checks whether the left value has a greater value than the one on the right side of the operator. If yes, it evaluates to true. Otherwise, false
less than	<	Checks if the left value is less than the right operand. If yes, then the expression evaluates to true. Otherwise, false
greater than OR equal to	>=	Checks if the left operand is either GREATER or EQUAL to the right operand. If yes, then the expression evaluates to true. Otherwise, false.
less than or equal to	<=	Checks if the left operand is either LESS or EQUAL to the right operand. If yes, then the expression evaluates to TRUE. Otherwise, FALSE.
equal value and equal type	===	Compares equality of two operands type
not equal value or not equal type	!==	

Example

```
4==’4’; //returns true
      (4+2)!=(3+5); //returns true
      "my friend"=="My Friend"; //returns false
      4<5; //returns true
      14>5; //returns true
      4<=5; //returns true
      let x =5;
      x==="5"; //returns false statements on each side must be equal and must be
of the same type
```

d. **Logical** – are used to compare two logical statements to determine if the result is true.

Operator	Symbol	
AND	&&	returns true if both statements on both sides of the operator are true
OR	||	returns true if a statement on either side of the statement of the true
NOT	!	Returns true if the statement to the right side of the operator is not true

Example

```
(1==2)&&(2==2) //returns false
      (1<2)&&(2>=3) //returns true
      (2>0)||(5==0) //returns true
      (2>0)||(5<0)  //returns true
      ("a"!="a")&&("c"!="d") //true
      !(2>5); // evaluates to true
```

Summary:

Let's review what you've learned:

- ✓ Expressions evaluate to a value.
- ✓ Statements can be composed of values, operators, expressions, keywords, and comments
- ✓ There are different types of operators

Name:		Date:

Expressions and Operators	Lesson 15 Quiz	Score:

After reading, answer the following questions. **Encircle** the *letter* of your choice from the given list of options.

1. What operator is used to assign a value to a variable in JavaScript?
   a. +
   b. –
   c. = assignment operator
   d. :
2. Which of the following is NOT a JavaScript operator?
   a. =
   b. ==
   c. ===
   d. +==
3. Which of the following comparisons will return false?
   a. 'A'!='B' T
   b. 5==5 T
   c. 15<15 F
   d. 0<=100 T
4. Which of the following statements will return true?
   a. ('z'=='z')&&(5<0) T && F =F
   b. !(15>=100) !F = T
   c. (5<1)&&(3==3) F && T = F
   d. (6!=3)||(7<2) T || F = T
5. What does a comparison operator do?
   a. Performs arithmetic operations
   b. Assigns values to variables
   c. Compares a number on the right side of the operator TO the number on the left
   d. Compares two values or statements, and returns a value of true or false
6. It is a JavaScript special operator that returns a string that tells you the type of the value being evaluated.
   a. this
   b. typeof
   c. new
   d. instanceof
7. It refers to JavaScript operators that are often used with conditional statement and loops to perform actions only when a certain condition is met.
   a. Mathematical operators
   b. Comparison operators
   c. Logical operators
   d. Assignment operator
8. What value will the comparison **"4"===(1+3)** statement yield?
   a. true
   b. false
9. What value will the comparison **"c">"C"** statement yield?
   a. true
   b. false
10. What value will the statement **"c"<="C"** statement yield?
    a. true
    b. false

UNIVERSITY OF Baguio

Name:		Date:	
Expressions and Operators		Lesson 15 Hands-on Activities	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Finals exercises in the Finals directory.
- Perform the set of activity on your computer. Print the HTML code and submit on the set deadline. (Modular Students)

Perform the set of activities on your computer.

1. Create a new HTML document named Exercise16.html. Insert a JavaScript code that will compute and display the sum of 8 + 7.5, using two variables: a and b. (10 points).

2. Create a new HTML document named Exercise17.html. Insert a JavaScript code that declares a variable called sum, assign numA + numB to it, and display the result in an alert box. (10 points).

3. Create a new HTML document named Exercise18.html. Insert a JavaScript code that will alert the remainder of any two variables. (5 points)

4. Create a new HTML document named Exercise19.html. Insert a JavaScript code that uses the correct assignment operator that will result in ***ans*** being 25 (same as ***ans*** = x + y). (10 points)

5. Create a new HTML document named Mod4Exer5.html. Insert a JavaScript code that uses comments to describe the correct data type of the following variables (10 points):

```
let length = 16;          // _____________________________________

let lastName = "Johnson"; // _____________________________________

let x = {

  fName: "Jenny",

  lName: "Deluna"

};                        // _____________________________________________________
```

UNIVERSITY OF Baguio

Lesson 16: Control Flow Statements

Time: 12 hours

About this lesson: This lesson will introduce and define JavaScript *control flow and conditional statements* which are necessary when you need to perform different actions for different decisions in your program. You will learn how to execute the same code over and over correctly and efficiently again using loops.

Objectives:

In this lesson, you are expected to:

- Identify and define conditional statements and control flow
- Use conditional statements and control flow
- Identify, define and use loops

16.1 What is control flow?

Control flow is the order in which the computer executes statements in a script (Control Flow, 2019)

Control Flow Statements:

- if statement (ES1)
- switch statement (ES3)
- while loop (ES1)
- do-while (ES3)
- for loop (ES6)
- for-of loop (ES6)
- for-await-of loop (ES2018)
- for-in loop (ES1)

** ES –> ECMA Script version

16.2 What is a conditional statement?

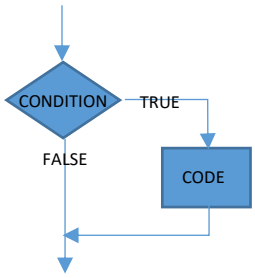
A *conditional statement* is a statement that you can use to execute a line or block of code based on the result of a condition or do something else if the result of that condition is not met (Pollock, 2009, pp. 116).

16.2.1 Conditional Statements

16.2.1.aif Statement

- you may use this to identify a block of code to be executed, if a specified condition is true.

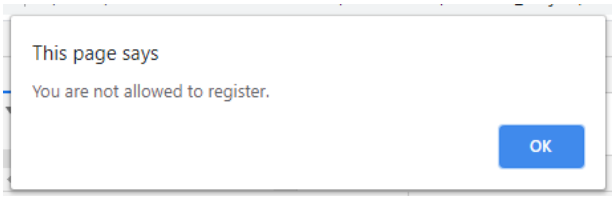
Flowchart: Syntax:




```
if (condition) {

    //block of code to be executed if
 the condition is true

}
```

Example **Output:**

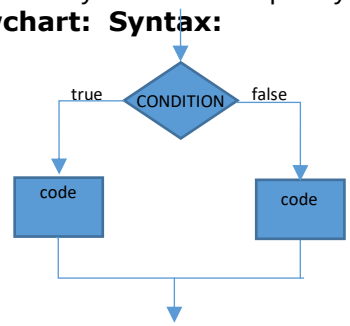
```
<body>
    <script>
    let age  = 11;
        if (age <18) {
        alert("You are not
        allowed to register.");
    }
    </script>
</body>
```



16.2.1.b.if-else statement

- you may use this to specify the block of code to be executed if the condition is false.

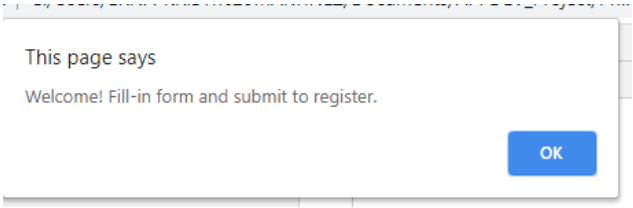
Flowchart: Syntax:




```
if (condition) {

  //block of code to be executed if the condition
 is true

} else {
```

Example Output

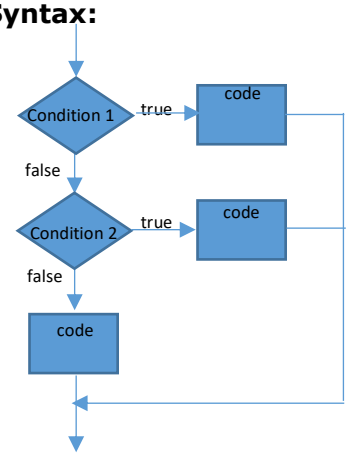
```
<body>
    <script>
        let age  = 18;
           if (age <18) {
              alert("You are not allowed to
              register.");
           } else {
              alert("Welcome! Fill-in form
              and submit to register.")
           }
    </script>
```



16.2.1.c.if-else-if

- you may use this to specify a new condition if the first condition is false

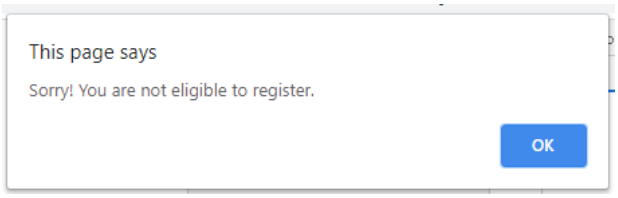
Flowchart: Syntax:




```
if (condition) {
 //block of code to be executed if the condition
is true
} else if (2nd condition) {
 //block of code if the 1st condition is false and
2nd condition is true
} else {
 //block of code if both the 1st and 2nd
conditions are false
}
```

Example Output

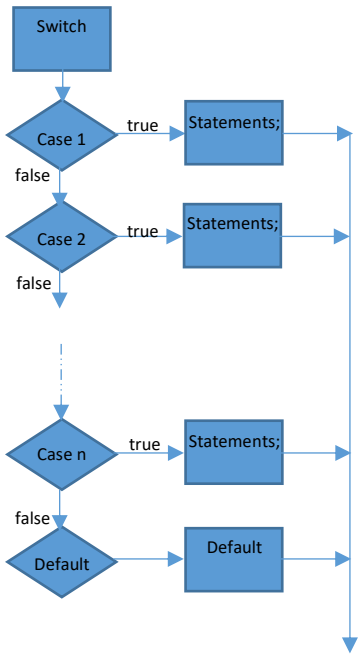
```
<body>
    <h2 id="target">Conditional Statements</h2>
    <script>
        let age  = 20;
        if (age<=17) {
            alert("You are not allowed to register.");
        }else if(age==18){
            alert("Welcome! Fill-in form and submit
            to register.");
        }else{
            alert("Sorry! You are not eligible to
            register.");
        }
    </script>
</body>
```



16.2.1.d.switch statement

- allows you to execute different actions based on different conditions by comparing value of an expression with a value in each case.

Flowchart: Syntax:




```
switch(expression){
        case a:
              //code block
              break;
         case b:
              //code block
              break;
         default:
              // code block
    }
```

Example: Output:

```
<body>
    <h2 id="target">Conditional Statements</h2>
    <script>
        let day  = 1;
        switch (day) {
            case 1:
            alert("Today is Monday,");
            break;
            case 2:
            alert("Twos-day Tuesday!,");
            break;
            case 3:
            alert("Middle of the work week Wednesday,")
            ;
            break;
            default:
             alert("It's finally Sunday!");
        }
    </script>
</body>
```

16.3. Loops

A *loop* is a block of code that will allow you to repeat a section of code a certain number of times.

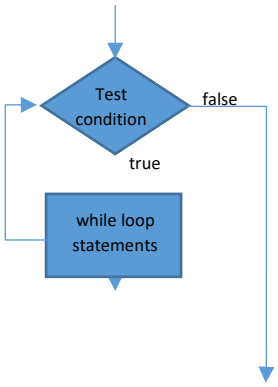
Kinds of loops

- while
- do-while
- for
- for-in
- for-of

16.3.1. while loop

- loops through a block of code as long as a specified condition is true.
- If the condition evaluates to *false*, the loop terminates
- If the test condition evaluates to *true*, the while body is executed one more time

Flowchart: Syntax:




```
while (condition) {
    //statements (while body)
}
```

Example: Output:

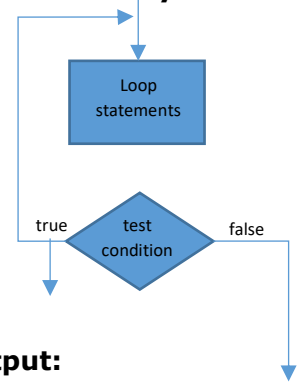
```
<body>
    <h2 id="target">Conditional Statements</h2>
    <script>
        let i=0;
        while(i<5){
            console.log(`i is ${i}`);
            i++;
        }
    </script>
</body>
```

```
i is 0          main copy.html:11
i is 1          main copy.html:11
i is 2          main copy.html:11
i is 3          main copy.html:11
i is 4          main copy.html:11
```

16.3.2. do-while loops

- works much like the *while-loop*, but it tests the condition *after* each loop iteration, not before.

Flowchart: Syntax:




```
do {
    //statements
}while (condition);
```

Example: Output:

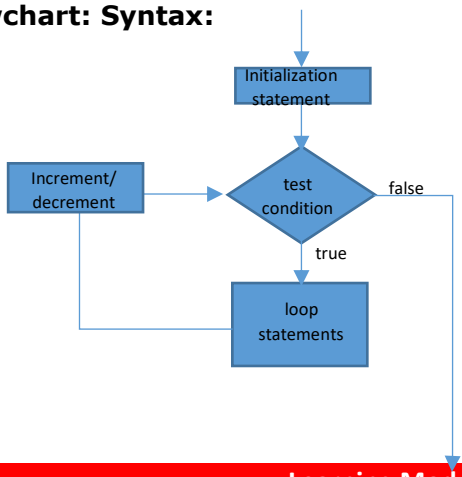
```
<body>
    <h2 id="target">Conditional Statements</h2>
    <script>
        let i=0;
        do{
            console.log(`i is ${i}`);
            i++;
        } while(i<5);
    </script>
</body>
```

```
i is 0          main copy.html:11
i is 1          main copy.html:11
i is 2          main copy.html:11
i is 3          main copy.html:11
i is 4          main copy.html:11
```

16.3.3. for loop

- loops through a block of code a specified number of times

Flowchart: Syntax:




```
for (statement 1; statement 2; statement 3)
    //statements to be executed
```

Example1: Output1:

```
<body>
    <h2 id="target">Conditional Statements</h2>
    <script>
        for(i = 0; i<5; i++){
            console.log(`The value of i is ${i}`);
        }
    </script>
</body>
```

```
The value of i is 0        main copy.html:10
The value of i is 1        main copy.html:10
The value of i is 2        main copy.html:10
The value of i is 3        main copy.html:10
The value of i is 4        main copy.html:10
>
```

Example2: Output2:

```
<body>
    <h2 id="target">Loops</h2>
    <h3>Class List</h3>
    <p id="cl"></p>
    <script>
        let students=["jen", "vida", "jun","jon","hya","edz",
        "chad","onyx","net","jacinto","ver","yo","mon","mhyke"];
        let i, len, s_lists;
        s_lists="";
        len=students.length;
        for(i = 0; i<len; i++){
            s_lists +=i + ". "+ students[i] + "<br>";
        }
        document.getElementById("cl").innerHTML=s_lists;
    </script>
</body>
```

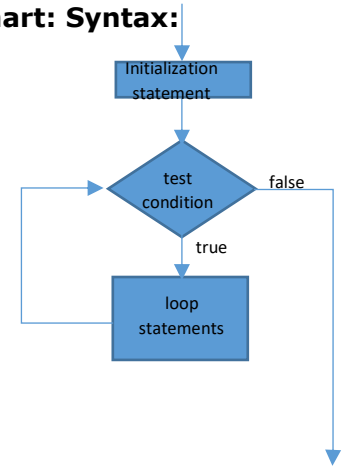
Class List

0. jen
1. vida
2. jun
3. jon
4. hya
5. edz
6. chad
7. onyx
8. net
9. jacinto
10. ver
11. yo
12. mon
13. mhyke

16.3.4. for-of loop

- allows you to loop over data structures that are *iterable* (*a data container that supports the* ***iteration protocol***) such as arrays and strings (Rauschmayer, 2020, pp. 191).

Flowchart: Syntax:




```
for (iteration_variable of iterable)

    //statements to be executed
```

Example: Output:

```
<body>
    <h2 id="target">Loops</h2>
    <h3>Color List</h3>
    <script>
        let colors = ["red", "teal", "orange", "turquoise"];
        var i;
        for(i of colors){
            document.write(i + "<br>");
        }
    </script>
</body>
```

Loops

Color List

red
teal
orange
turquoise

16.3.5. for-in loop

- allows you to loop through the properties of an object

Syntax:

```
for (iteration_variable in iterable)

    //statements to be executed
```

Example: Output:

```
<body>
    <h2 id="target">Loops</h2>
    <h3>Student</h3>
    <p id="stdnt"></p>
    <script>
        const student = {idnum:"20201451",fname:"Sophie",
        lname:"Ghi",age:17};
        var i;
        var studes ="";
        for(i in student){
            studes +=student[i] + " ";
        }
        document.getElementById("stdnt").innerHTML = studes;
    </script>
</body>
```

Loops

Student

20201451 Sophie Ghi 17

UNIVERSITY OF Baguio

Name:		Date:	
Control Flow Statements		Lesson 16 Quiz	Score:

After reading, answer the following questions. **Encircle** the *letter* of your choice from the given list of options.

1. Rather than executing every single line of code within the script, what allows you to execute certain sections of the script only when a particular condition is met
    a. Comparison statements
    b. Conditional statements
    c. Logical statements
    d. Looping statements
2. Which of the following would be valid as the first line of an if/else statement?
    a. if(aNum=0)
    b. if(aNum<10)
    c. else
    d. if((aNum==0&&)
3. What do you use to enclose the blocks of code in conditionals and loops?
    a. Parentheses ()
    b. Curly brackets {}
    c. Square brackets []
    d. Angle brackets <>
4. Which of the following items is a correct *for loop* expression?
    a. for(i=1;i<6;i+=1)
    b. for(i==1;i<6;i+=1)
    c. for(i=1;i=6;i+=1)
    d. for(i+=1;i<6;i=1)
5. Which of the following items is an **incorrect** *while loop* expression?
    a. while(a<=5)
    b. while(a=5)
    c. while(a<5)
    d. while(a!=5)
6. The following items are JavaScript supported loops, *except*
    a. for
    b. for-in
    c. do-while
    d. while-in
7. In the following for loop line: `for (statement 1; statement 2; statement 3);` which of the following items is NOT true?
    a. Statement 1 is executed (one time) before the execution of the code block.
    b. Statement 1 is optional.
    c. Statement 2 is used to evaluate the condition of the initial variable. It is always required.
    d. Statement 3 is executed (every time) after the code block has been executed.
8. In JavaScript Switch statements, which of the following items is NOT true?
    a. The switch expression is evaluated once.
    b. Omitting the break statement will stop the execution inside the block.
    c. The value of the expression is compared with the values of each case.
    d. It is not necessary to break the last case in a switch block.
9. Which of the following is true about JavaScript loops?
    a. Loops can execute a block of code as long as a specified condition is true
    b. Incrementing or decrementing the control variable in a loop will not affect the loop.
    c. The do-while loop first checks if the condition is true before executing the code block.
    d. The *while* loop and *for* loop will not give the same result.
10. Which of the following is NOT true about conditional statements?
    a. Use *if* statement to specify a block of JavaScript code to be executed if a condition is true.
    b. Uppercase letters (If or IF) will generate a JavaScript error.
    c. The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
    d. Nested if….else if.. statements will generate a JavaScript error.

UNIVERSITY OF Baguio

Name:		Date:	
Control Flow Statements		Lesson 16 Hands-on Activities	Score:

General Instructions

- Always complete the general information in the heading section
- Commit and push the local changes in your remote GitHub repository.
- Submit your updated published GitHub URL.
- Save Finals exercises in the Finals directory.
- Perform the set of activity on your computer. Print the HTML code and submit on the set deadline. (Modular Students)

Perform the set of activities on your computer.

1. Create a new HTML document named Exercise17.html. Insert a JavaScript code with a switch statement that will alert a greeting for every day of the week (i.e. "Hello" if days is "Monday", and "Welcome" if days is "Friday). (10 points).
2. Create a new HTML document named Exercise18.html. Insert a JavaScript code modify the previous activity by replacing switch conditional statement with the *if-else* statement (i.e. "Hello" if days is "Monday", and "Welcome" if days is "Friday). (10 points).
3. Create a new HTML document named Exercise19.html. Insert a JavaScript code with a *while loop* that runs from 0 to 9 and displays the output in an alert. (10 points).
4. Create a new HTML document named Exercise20.html. Insert a JavaScript code with *for loop* that runs from 9 to 0 and displays the output in the console. (10 points).
5. Create a new HTML document named Exercise21.html. Insert a JavaScript code with *for-of* loop that runs through the iterable array of months. Display the output using innerHTML. (10 points).

UNIVERSITY OF Baguio

Lesson 16: Using Functions

Time: **6 hours**

About this lesson: This lesson will introduce you JavaScript functions and how to use them in web application development. You will learn how to create and/or implement JavaScript functions on web pages.

Objectives:

In this lesson, you are expected to:

1. Define functions
2. Create and implement functions
3. Call functions in Scripts

17.1. What is a function?

According to Pollock (2010), "A function is basically a little script within a larger script. Its purpose is to perform a single task or a series of tasks. What a function does depends on what code you place inside it".

Kantor (2019) explains, "Functions are the main building blocks of the program. They allow the code to be called many times without repetition".

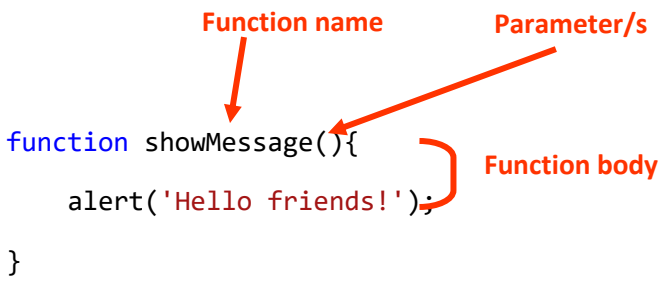
17.2. Why are functions useful?

- A function helps organize several parts of a script so that it is divided into different tasks that perform specific functions. Through functions it will be easier for you to understand and debug the each section of the script.
- Functions are reusable.
- Functions can be written to perform complex tasks.

17.3. How to declare functions

- To create a function you have to use a *function declaration.*
- The *function* keyword declares the new function
- The *function name* will allow you to call (or execute) the function
- The set of *parameters* will allow you to pass parameters to the function (or if you will allow the function to accepts any parameter/s)
- The *function body* placed between the opening and closing curly brackets ({ }) is where you will add the code that the function is supposed to accomplish. The browser will execute all of the code inside the curly brackets.
- For the example below, when called, will display the message *'Hello friends!'* through an alert window.

Example 1



Naming Functions

When naming functions there are some things that you need to remember.

- Function names are case sensitive, similar to you variable names. This is important because you will *calling* your functions by their names.
- There naming convention for variables also apply to naming functions. There are allowed characters and never using reserved JavaScript words.
- Give your functions meaningful names so that it will be easier for you to identify what each function does just by its *name*.

17.4. Local and global variables

- Variables may be declared inside (local variable) the function body and outside (outer or global variable).
- *Local variables* are only visible inside the function it was declared in (Example 2).
- *Global or outer variables* can be accessed or used within the function body (Example 3).
- *Global variables* are visible and can be used by any function. However, it is advised to minimize the use of global variables.

Example 2

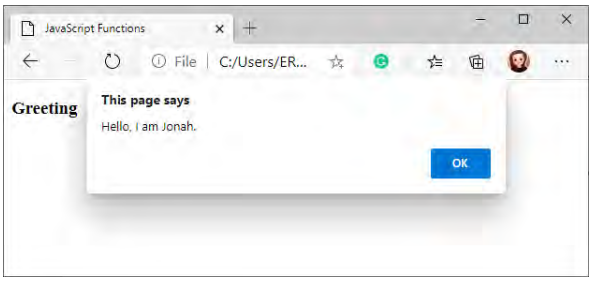
```
function showMessage(){
    let msg = "Hello, I am Jonah.";
    alert(msg);
}

showMessage();
```

Local variable

Function call

** Internal JavaScript *Output on a browser*

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Functions</title>
    </head>
    <body>
        <h3>Greeting</h3>
        <script>
            function showMessage(){
                let msg = "Hello, I am Jonah.";
                alert(msg);
            }
        showMessage();
        </script>
    </body>
</html>
```



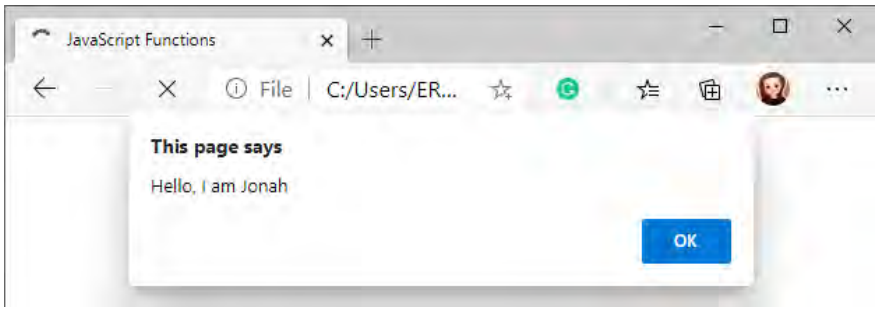
Example 3

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>JavaScript Functions</title>
5      </head>
6      <body>
7          <h3>Greeting</h3>
8          <script>
9              let aName="Jonah";
10             function showMessage(){
11                 let msg = "Hello, I am " + aName;
12                 alert(msg);
13             }
14         showMessage();
15         </script>
16     </body>
17 </html>
```

Global variable

Local variable and a global variable accessed within a function.

Output on a browser



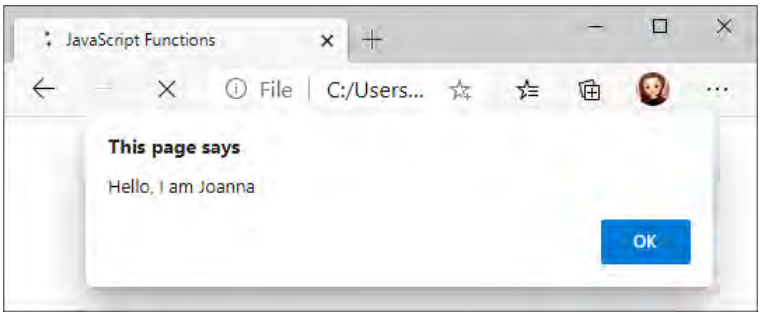
- The function can access or use the global/outer variable. You may also allow the function to re-assign a value to the variable (*Example 4*).

Example 4

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Functions</title>
    </head>
    <body>
        <h3>Greeting</h3>
        <script>
            let aName="Jonah";
            function showMessage(){
                aName = "Joanna";
                let msg = "Hello, I am " + aName;
                alert(msg);
            }
        showMessage();
        </script>
    </body>
</html>
```

The global variable is re-assigned a value inside the function body.

Output on a browser



Variable shadowing

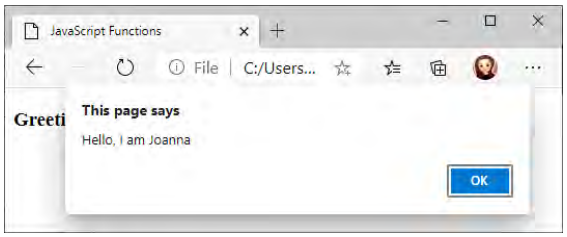
- *Variable shadowing* happens when a global variable is re-declared as a local variable. Simply, it happens when a global variable has the same name a local variable.
- When this is done, *variable scoping* overlaps but rules state that the inner variable gets priority and the global variable is ignored (in the function). This is not recommended.

Example 5 output on a browser

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>JavaScript Functions</title>
5       </head>
6       <body>
7           <h3>Greeting</h3>
8           <script>
9               let aName="Jonah";
10              function showMessage(){
11                  let aName = "Joanna";
12                  let msg = "Hello, I am " + aName;
13                  alert(msg);
14              }
15          showMessage();
16          </script>
17      </body>
18  </html>
```

global variable

Local variable with the same name as a global variable



17.5. Using Parameters in Functions

- You will use *parameters* to allow your function to accept one or more variables from outside the function.
- Syntax is shown below:

```
function functionName(parameter1, parameter2){

        //function body

}
```

Parameters are placed within the parentheses after the function name.

- Multiple parameters should be separated with a comma (,) as shown in Example 6.

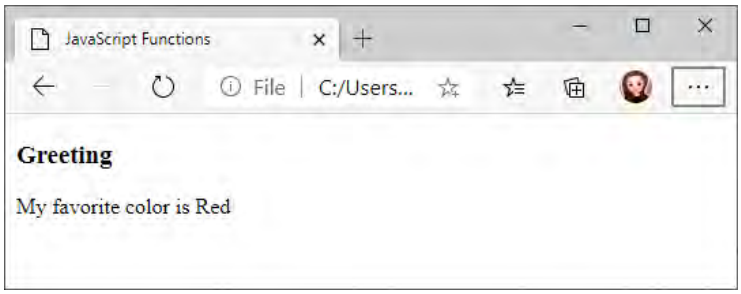
Example 6

```
<html>
    <head>
        <title>JavaScript Functions</title>
    </head>
    <body>
        <h3>Greeting</h3>
        <script>

            function showColors(colr1, colr2){
                var favColor=colr1;
                document.write(`My favorite color is ${favColor}`);
            }
        showColors('Red', 'Blue');
        </script>
    </body>
</html>
```

Parameters are placed within the parentheses after the function name.

Parameter values passed to the function.

Output of Example 6 on a browser



17.6. Returning a value

- You can write functions that can return a value back to the function call.
- The *return* directive can be placed anywhere within the function.
- Once the *return* statement is reached, the function stops, and returns the value to the function call.

Example 7

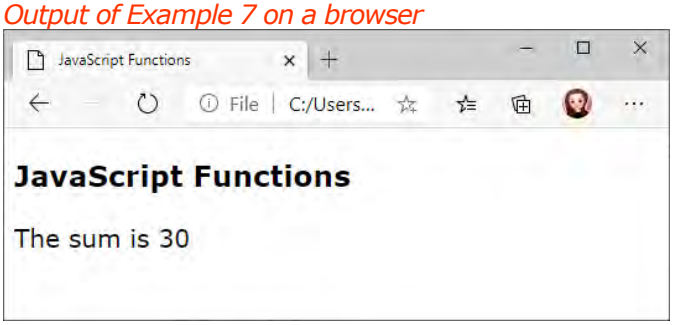
```
<html>
    <head>
        <title>JavaScript Functions</title>
    </head>
    <body>
        <h3>JavaScript Functions</h3>
        <script>
            function computeSum(num1, num2){
                return num1 + num2;
            }
            let sum=computeSum(12,18);
            document.write(`The sum is ${sum}`);
        </script>
    </body>
</html>
```

Parameters passed to the function

return directive computing for the sum of the 2 passed values

Function call that passes the values of the 2 parameters

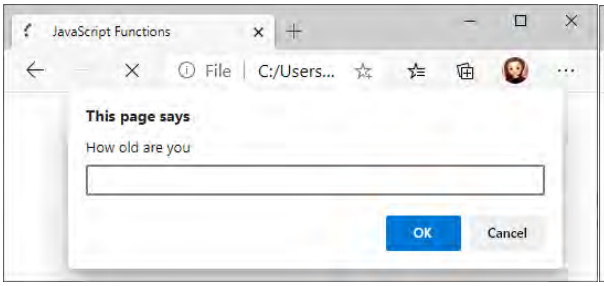
Output of Example 7 on a browser

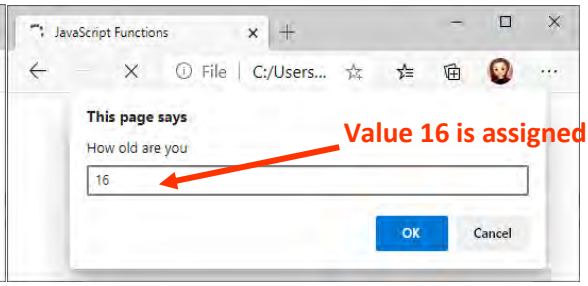


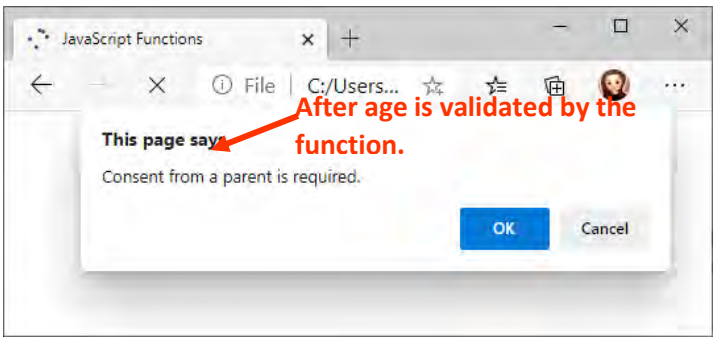
Example 8

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript Functions</title>
    </head>
    <body>
        <h3>JavaScript Functions</h3>
        <script>
            let age=prompt('How old are you',);
            function checkAge(age){
                if (age>=18){
                    return alert('Welcome to the group.')
                }else{
                    return confirm('Consent from a parent is required.');
                }
            }
            checkAge(age);
        </script>
    </body>
</html>
```

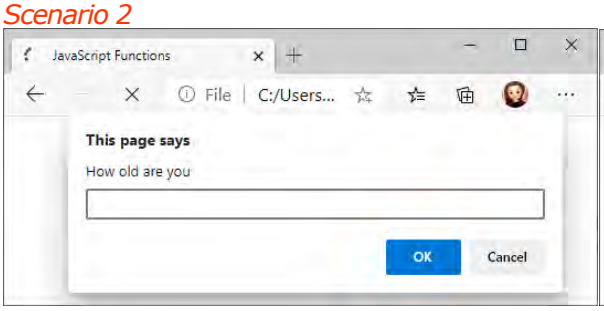
Output of Example 8 on a browser
Scenario 1

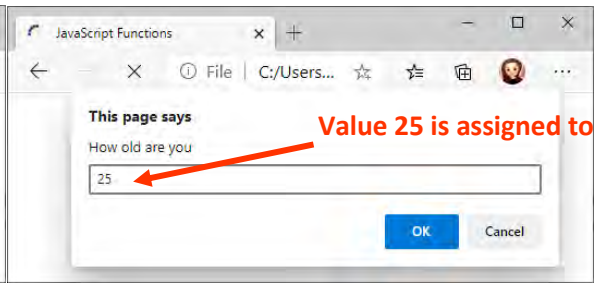


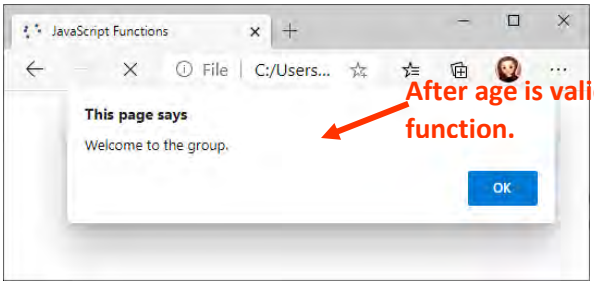




Scenario 2







17.7. Calling functions in your scripts

- To *call* a function in JavaScript is done by using the *function name* plus the set of parentheses (with or without parameters), then a semi-colon (;)
- Make sure you define your functions before calling it as a good coding practice.
- Your functions may be called anywhere within the JavaScript, but make sure the function is loaded by the browser to make it work.
- Examine the following examples of how to define and call functions.

Example 9: using external JavaScript

- HTML document with script tag within the <body> and an external JavaScript file containing the function.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Calling Functions</title>
    </head>
    <body>
        <h2 id="target">Calling Functions</h2>
        <script src="../js/main.js">
        </script>
    </body>
</html>
```

Figure a. HTML document

Practice > js > main.js > ...

```
1   //function
2   function showMessage(){
3       let msg = "Hello, I am Jonah.";
4       alert(msg);
5   }
6   showMessage();
7
```

Figure b. main.js function showMessage()

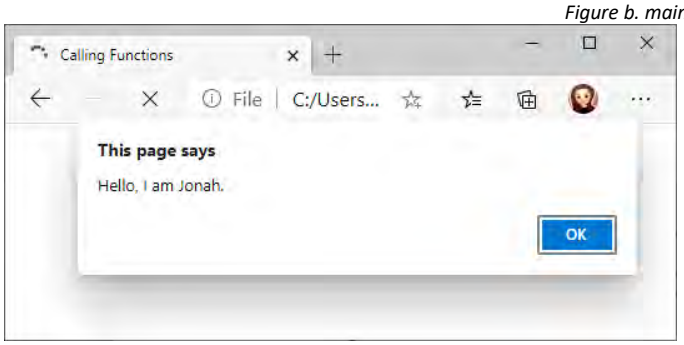


Figure c. HTML document opened on the browser.

Example 10: Calling a function from another function

Practice > js > main.js > ...

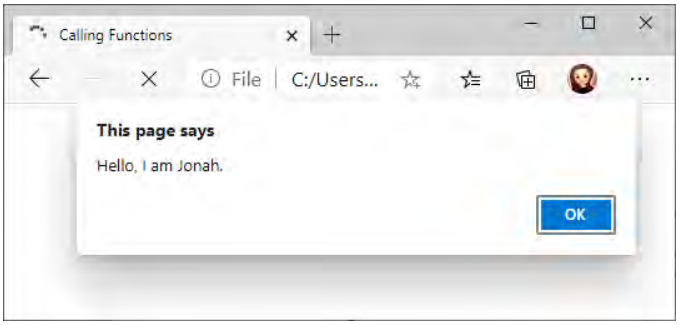
```
1   //functions
2   function updateMessage(){
3       let msg = "Hello, I am Jonah.";
4       alert(msg);
5   }
6   function showMessage(){
7       updateMessage();
8   }
9   showMessage();
10
```

Function that displays the message in an alert box.

Function that calls the updateMessage function.

Calls the showMessage function.

Output when HTML document is opened on a browser



Example 11

Input value of user is entered through prompt box. String input is parsed to integer and assigned to a variable.

Practice > js > main.js > computeNum

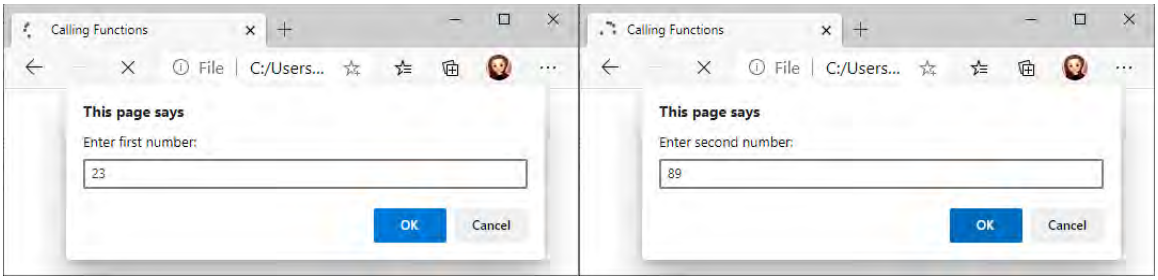
```
1   //functions
2   function getInput(){
3       let num1 =parseInt(prompt('Enter first number: ',));
4       let num2 =parseInt(prompt('Enter second number: ',));
5       computeNum(num1, num2);
6   }
7   function computeNum(num1, num2){
8       let numSum=num1+num2;
9       return document.write(`The sum of the 2 numbers is: ${numSum}`);
10  }
11  getInput();
12
```

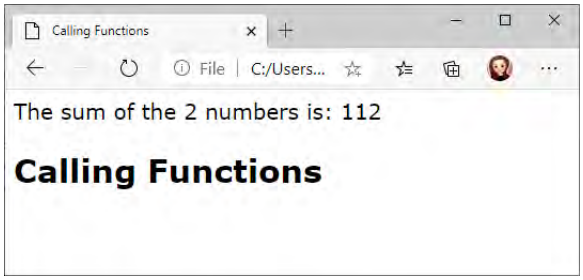
Function computeNum is called that passes two parameters to the called function.

Function computeNum accepts the two parameters, computes the sum and returns the result through document.write method.

Function getInput() is called once the script is loaded by the browser through the html document.

Output when HTML document is opened on a browser





References

Carey, P. M. (2017). Introducing Responsive Design: Comprehensive. In New Perspectives HTML5 and CSS3: Comprehensive (7th ed., pp. 360–367). Cengage Learning.

Dechalert, A. (2020, April 11). CSS Selector Diagram [Diagrams]. Dottedsquirrel.Com. https://dottedsquirrel.com/css/css-selectors/

freeCodeCamp.org. (2020, January 29). Semantic HTML5 Elements Explained. https://www.freecodecamp.org/news/semantic-html5-elements/

JavaScript First Steps. (2020, July 9). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps

JavaScript Introduction. (2020). W3schools.Com. https://www.w3schools.com/js/js_intro.asp

JavaScript Scope. (2020). W3schools.Com. https://www.w3schools.com/js/js_scope.asp

LePage, P., & Andrew, R. (2020, April 14). Responsive web design basics. Web.Dev. https://web.dev/responsive-web-design-basics/

Lindley, C. L. (2019). *Front-end Developer Handbook 2019 - Learn the entire JavaScript, CSS and HTML development practice!* Https://Frontendmasters.Com/Books/Front-End-Handbook/2019/. https://frontendmasters.com/books/front-end-handbook/2019/

Link types - HTML: HyperText Markup Language | MDN. (2021, January 27). Developer.Mozilla.Org. https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types

Macaulay, M. (2017). Responsive and Adaptive Web. In Introduction to Web Interaction Design: With HTML and CSS (1st ed., pp. 767–794). Chapman and Hall/CRC.

MDN Contributors. (2021, March 4). Responsive design - Learn web development | MDN. Https://Developer.Mozilla.Org/. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

Pollock, J. (2009). *JavaScript, A Beginner's Guide, Third Edition* (Third ed.). McGraw-Hill Education.

Rauschmayer, A. (2020). *JavaScript for Impatient Programmers* (2020 Edition). Independently Published.

w3schools. (n.d.). HTML Semantic Elements. W3schools.Com. Retrieved January 25, 2021, from https://www.w3schools.com/html/html5_semantic_elements.asp

Word Wide Web Consortium. (2015, October 23). 3.2.5 Content models — HTML5. Https://Www.W3.Org/. https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html#flow-content