Before you start reviewing this, please download **one** of following code editors. It is highly encouraged to code along and in Zybook so you know how the code is displayed.

[Visual Studio Code](Visual%20Studio%20Code)  
[Sublime Text](Sublime%20Text)  
[Notepad ++](Notepad%20++)

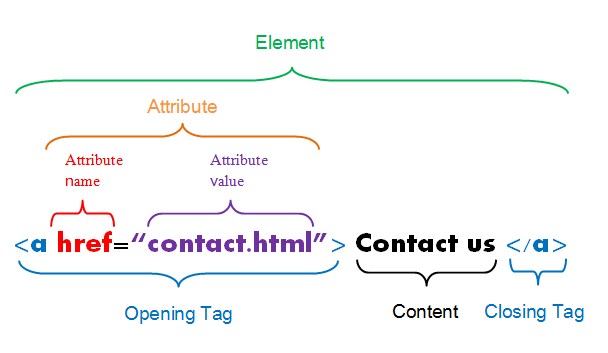
**Please pay attention to the highlighted text as it’s very important to the OA**

HTML document structure

* The **<!DOCTYPE>** declaration instructs the web browser about what type of document follows. All HTML documents must start with a <!DOCTYPE> declaration.
* The **<html>** is the container for all other HTML elements (except for the <!DOCTYPE> declaration).
* The **<head>** opening and closing tags contain the document title, document metadata, and various other elements that are typically not displayed in the webpage.
* The **<meta>** tag specifies metadata, which is data that describes the document's data. <meta charset="UTF-8"> describes how characters are represented in the HTML document. Additional <meta> tags may be used to indicate when the document was saved, who the author is, etc.
* The **<title>** opening and closing tags enclose the name of the document. The title is usually displayed in the browser's titlebar, is used by search engines, and is used for bookmarking.
* The **<body>** opening and closing tags enclose all elements and content to be rendered in the browser.

**HTML tags vs HTML elements vs HTML attributes**

* **HTML tags** - are used to enclose the HTML elements.
  + For example, <p></p>
* **HTML elements** - comprise the opening and closing tags along with the content.
  + For example, <p>this is a paragraph</p>
* **HTML attributes** - are used to describe the characteristic of an HTML element in detail.
  + For example, <img src="mydog.jpg" alt="A photo of my dog.">



***Please note you can look any website to see a basic website structure by inspecting website by right clicking choose inspect***

* **What are the required tags to make a website?**

<!DOCTYPE html>  
 <html>  
 <head>

</head>

<body>

</body>

</html>

* **Where does the title and meta data go in a basic website?**  
  (Inside the head tag)

<head>  
 <title>My first webpage</title>

<meta charset="UTF-8">  
</head>

* Validators
  + Used to guarantees compliance to HTML and CSS standards during website design and development.
  + Checks for syntax and logic errors
  + What website would you use?
    - HTML - <https://validator.w3.org/>
    - CSS - <https://jigsaw.w3.org/css-validator/>

Basic HTML tags

**Paragraph tag**

* <p> …. </p>
* Container tag which means it has an opening and closing tag
* Please type a paragraph and use <p></p>

**Whitespace**

* Is ignored/absorbs in HTML but **not** in XML

**Page Break**

* 1 <br> tag - Single line spacing
* 2 <br> tag - Double line spacing
* is a standalone or empty tag.
* used to move a line of code to the next line
* Please type the following example to using a code editor

<p>The White House<br>

1600 Pennsylvania Avenue Northwest<br>

Washington, DC 20500</p>

**How will this display?**

The White House  
1600 Pennsylvania Avenue Northwest  
Washington, DC 20500

**Do you know notice the difference with <br>**

<p>The White House<br><br>

1600 Pennsylvania Avenue Northwest<br>

Washington, DC 20500</p>

**How will this display?**

The White House  
  
1600 Pennsylvania Avenue Northwest  
Washington, DC 20500

**Semantic Elements**<https://www.w3schools.com/html/html5_semantic_elements.asp>

* A **semantic element** clearly defines its content.
  + For example: <form>, <table>, and <article>
* A **non-semantic element** tells nothing about its content.
  + For example: **<div>** and **<span>**
    - **<div> -** This stands for "division" and is a **block-level element**. It's used to create sections or divisions in an HTML document.  
      * **When do I use it?**
        + when you want to create larger structural sections or divisions within your webpage.
      * **For example**: <div class="header">

<h1>Welcome to My Website</h1>

<!-- Other header content -->

</div>

<div class="main-content">

<p>This is the main content of the page.</p>

<!-- Other main content -->

</div>

* + - **<span> -** This is an **inline element** and is used to apply styles to a small piece of content within a larger block-level element  
      * **When do I use it?**
        + when you want to apply styles or manipulate smaller portions of text or inline elements within larger blocks of content
      * **For example**: <p>This is a <span style="color: blue;">blue</span> word in a sentence.</p>
* Used to group content together.

**Heading**

* There are 6 Heading tags ranging from smallest to biggest.
  + <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>
* <h1> is the biggest and <h6> is the smallest
* Are container tags (opening and closing tags)
* The h4 tag <h4> is similar to the paragraph tag </p>

**Text formatting**

* **<b>** - **Bold text**
* **<strong>** - **Important text**
* *<i>* - *Italic text*
* *<em>* - *Emphasized text*
* <u> - underlines text
* <mark> - highlight text
* <cite> - define the title of a work

**<b>** and **<strong>** have the same results, but **<strong>** is meant to indicate stronger importance or emphasis for search engines and accessibility tools, while **<b>** simply stylizes text as bold

*<i>* and *<em>* have the same results, but *<em>* is more common to use in HTML5.

HTML vs CSS Comments

* **HTML comment (is the same XML comment)**
  + <! - - this is a comment - ->
* **CSS comment**
  + /\* this is a comment \*/
* The browser ignores the comments and are not seen (invisible) on the webpage
* However, you can find the comments by viewing page source
* Is used for web developers as notes for specific code

Lists

**Unordered lists**

* ul = unordered list and the tags <ul></ul>
* li = listed item and the tags are <li></li>
* The default object for an unordered list is a black disc
* Both ul and li are container tag (opening and closing tags)

Make an unordered list for apples, bananas, and oranges

<ul>  
 <li>Apples</li>  
 <li>Bananas</li>  
 <li>Oranges</li>  
</ul>

* Type attribute for an unordered list can change the shape of the bullet to a disc (default), circle, square, or none
* The syntax for a circle is <ul type="disc"> **(but by default, it’s a disc so this is not needed)**
* The syntax for a circle is <ul type="circle">
* The syntax for a square is <ul type="square">
* The syntax for a none is <ul type="none">

**Ordered lists.**

* ol = ordered list and the tags <ol></ol>
* li = listed item and the tags are <li></li>
* By default, for an ordered list it starts with 1
* Both ol and li are container tag (opening and closing tags)
* **Type attribute** for an ordered list can change list to:
  + <ol type="1"> = Numbers is the default
  + <ol type="A"> = uppercase letters
  + <ol type="a"> = lowercase, letters
  + <ol type="I"> = uppercase Roman numerals
  + <ol type="i"> = lowercase Roman numerals
* **Start attribute** only works with numbers.
  + For example, <ol start="100">
  + Please try code the following:

<ol start="50">

<li>Apples</li>

<li>Bananas</li>

<li>Oranges</li>

</ol>

Make an ordered list for apples, bananas, and oranges

<ol>  
 <li>Apples</li>  
 <li>Bananas</li>  
 <li>Oranges</li>  
</ol>

Now type using uppercase and lowercase Roman numerals

<ol type="I">  
 <li>Apples</li>  
 <li>Bananas</li>  
 <li>Oranges</li>  
</ol>

<ol type="i">  
 <li>Apples</li>  
 <li>Bananas</li>  
 <li>Oranges</li>  
</ol>

**Creating a table**

* You must first use the table tag <table></table>
* <tr> which is a table row is the next tag you need
* Then <th> which stands for table header and is **bolded** (While optional, it aids in labeling data.)
* <td> which stands for table data is required
* <caption> tag is optional and **always** above the table regardless where it’s coded in the table
* The **3 required tags** to create a table are:
  + **<table> <tr> and <td>**
* \*\*\* By default, tables have **no** borders around the table cells without CSS **(see below) \*\***

**Copy or type the following the code:**

**As you code it, how many rows and columns do you have??**

<!DOCTYPE html>  
<html>  
 <head>  
 <title>Creating a table</title>  
 </head>  
 <body>  
 <table>  
 <tr>  
 <th>Name</th>  
 <th>City</th>  
 <th>State</th>  
 </tr>  
 <tr>  
 <td>Josh</td>  
 <td>Dallas</td>  
 <td>Texas</td>  
 </tr>  
 <tr>  
 <td>Nancy</td>  
 <td>Boca Raton</td>  
 <td>Florida</td>  
 </tr>  
 </table>  
 </body>  
</html>

This is how the table should display:

|  |  |  |
| --- | --- | --- |
| **Name** | **City** | **State** |
| Josh | Dallas | Texas |
| Nancy | Boca Raton | Florida |

*\*\* You will notice it has no borders or padding because there is no CSS to style it. \*\**

**Adding borders and padding to Tables using CSS**

* The style tag used here (which is internal/embedded CSS) puts borders and padding in the table.
* The border is 1px (top, right, bottom, left) solid black all around the content.
* The border-collapse property is set to collapse the border.
* Padding is the area that surrounds the content in the table.

***\*\*\* Please note later on we will go over CSS \*\*\*\****

<!DOCTYPE html>  
<html>  
 <head>  
 <title>Creating a table</title>  
 **<style>  
 table, th, td, tr {  
 border: 1px solid black;   
 border-collapse: collapse;  
 padding: 10px;  
 }  
 </style>**  
 </head>  
 <body>  
 <table>  
 <tr>  
 <th>Name</th>  
 <th>City</th>  
 <th>State</th>  
 </tr>  
 <tr>  
 <td>Josh</td>  
 <td>Dallas</td>  
 <td>Texas</td>  
 </tr>  
 <tr>  
 <td>Nancy</td>  
 <td>Boca Raton</td>  
 <td>Florida</td>  
 </tr>  
 </table>  
 </body>  
</html>

A white rectangular grid with black text

Description automatically generated

Images

<img> tag

* Stand alone tag (also known as empty tag)
* Two attributes that are needed
  + **src** = what the location of the URL
  + **alt** = provides a description of image
    - it's considered best practice and is strongly encouraged to include the alt attribute to ensure your HTML conforms to accessibility standards
    - Validators may issue a warning or notification rather than a direct error.

Please copy or type the following code

<img src =<https://res.cloudinary.com/dk-find-out/image/upload/q_80,w_1920,f_auto/580540_mjznrj.jpg> alt="dog">

You’ll notice the image is huge!!

So how do we fix this?

We add width attribute and let’s 400px

<img src =<https://res.cloudinary.com/dk-find-out/image/upload/q_80,w_1920,f_auto/580540_mjznrj.jpg> width="400px" alt="dog">

If you copied or accidently deleted part of the image link, what will happen?

<img src =https://res.cloudinary.com/find-out/image/upload/q\_80,w\_1920,f\_auto/580540\_mjznrj.jpg width="400px" alt="dog">

You should see this below

A close-up of a logo

Description automatically generated

Image formats

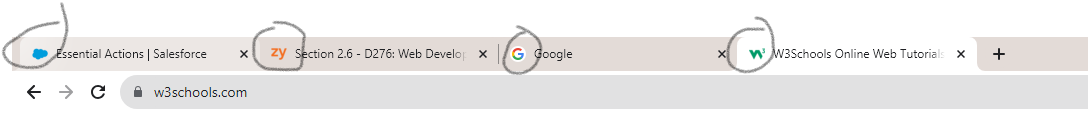
**A screenshot of a computer

Description automatically generated**

Please note: **Both GIF formats, 87a and 89a, support animation too.**

Please note**: SVG and TIFF also support transparency.**

**Favicon**



* It’s the small icon next to website name
* Can be svg, ico, png, jpg, or gif images and uses a <link> tag

W3schools example:

<link rel="icon" href="https://www.w3schools.com/favicon.ico">

Hyperlinks

**Anchor tag**

* <a></a> is an anchor tag which is need to create a hyperlink
  + **For example**: <a href="<https://www.w3schools.com>">W3schools</a>
  + **Target attribute**
    - <a href="<https://www.w3schools.com>" **target=**"\_blank">W3schools</a>
      * This will open a new tab or window
* **Opens email clients**
  + <a href="<mailto:someone@example.com>">Send email</a>
* **Hyperlink** is defined as one place in a document to another place in the same or a different document usually with a single mouse click
* Two types of URL
  + **Absolute** – provide the full web address, leading to an external location (e.g., "https://www.example.com/page").
  + **Relative** - Describes a link location in relation to the current page or the root of the website. It does not include the complete web address and often indicates a path within the same domain.

**Section ID**

* Fragment of a website is using the # symbol
  + <https://en.wikipedia.org/wiki/Computer_science#History>
    - This would take you directly to the History section.

**Creating a bookmark** (<https://www.w3schools.com/html/html_links_bookmarks.asp>)

* You need to use the ID attribute.
  + See example: <h2 id="C4">Chapter 4</h2>
* add a link to the bookmark ("Jump to Chapter 4"), from within the same page:
  + <a href="#C4">Jump to Chapter 4</a>

Special Characters

* XML requires using special characters for the following symbols
* Always start with a & and ends with ;
* HTML is not required

A screenshot of a computer

Description automatically generated

Forms

**<form> tag**

* Two type of attributes
  + **action** attribute – specifies the URL where the form data will be sent upon submission.
  + **method** attribute
    - tells the browser how to communicate with the server
    - Method is either GET or POST
    - By default, method attribute, is a GET
  + For example

<form action="https://wp.zybooks.com/form-viewer.php" method="POST">

**GET Method**

* is the default method
* is visible to anyone so it should not be used for sending private information
* **Query strings** - is a set of name=value pairs separated by the ampersand character (&)
  + After the question mark (?), is the query string
  + In this example,
    - <https://www.google.com/search?q=john+smith&>
    - John+smith& would be the query string

**POST Method**

* more secure

**Input Element**

* <input> tag - allows the user to enter information into a web page

Here are a couple of different typesA screenshot of a computer

Description automatically generated

* Input has 5 primary attributes (see form below for example)
  + The **type attribute** indicates the widget type. Common types include text, password, submit, and button.
  + The **name attribute** names the widget and sends the widget’s value when the widget’s form is submitted.
  + The **id attribute** is used to give a widget a unique identifier.
  + The **placeholder attribute** specifies text that first appears in a text widget, typically for giving the user a hint as to the expected value.
  + The **value attribute** specifies a default value for a widget.
* <labels> tags displays descriptive text associated with form.
  + <label for="fname">First Name:</label><br>

Either copy or type the following form.

<form action="https://wp.zybooks.com/form-viewer.php" method="POST">

<label for="fname">First Name:</label><br>

<input type="text" id="fname" name="fname" placeholder="Enter your first name" required><br><br>

<label for="lname">Last Name:</label><br>

<input type="text" id="lname" name="lname" placeholder="Enter your last name" required><br><br>

<label for="email">Email Address</label><br>

<input type="email" id="email" name="email" placeholder="Enter your email address" required><br><br>

<label for="confirm-email">Confirm Email</label><br>

<input type="email" id="confirm" name="confirm-email" placeholder="Please confirm your email address" required><br><br>

<label for="questions">Questions</label><br>

<textarea row="5" type="text" id="question" name="question" placeholder="Ask your question here" ></textarea><br><br>

<button>Submit</button>

</form>

<Textarea> tag

<textarea name="message" rows="4" cols="50">Write a message...</textarea>

A white rectangular frame with black border

Description automatically generated

Form input types

**Checkbox**

* <input type="checkbox" >
* Non-mutually exclusive **(name attribute can be different for each item)**
* Type of copy the following code:

<!DOCTYPE html>

<html>

<body>

<form>

<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">

<label for="vehicle1"> I have a bike</label><br>

<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">

<label for="vehicle2"> I have a car</label><br>

<input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">

<label for="vehicle3"> I have a boat</label><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

* The display should look like this:

Black text on a white background

Description automatically generated

**Radio Button**

* <input type="radio">
* Mutually exclusive **(name attribute has to be same for each item)**
* Type of copy the following code:

<!DOCTYPE html>

<html>

<body>

<form>

<p>Please select your favorite Web language:</p>

  <input type="radio" id="html" name="fav\_language" value="HTML">

  <label for="html">HTML</label><br>

  <input type="radio" id="css" name="fav\_language" value="CSS">

  <label for="css">CSS</label><br>

  <input type="radio" id="javascript" name="fav\_language" value="JavaScript">

  <label for="javascript">JavaScript</label>

</form>

</body>

</html>

* The display should look like this

A white background with black text

Description automatically generated

**Drop-down menu/list**

* <select></select> tag creates a drop-down menu/list for a user to select predefined values
* <option></option> tag create a value for the user to choose
* By default, the first option is selected

<!DOCTYPE html>

<html>

<body>

<label for="cars">Choose a car:</label>

<select id="cars" name="carlist" form="carform">

<option value="Telsa">Telsa</option>

<option value="Toyota">Toyota</option>

<option value="Nissan">Nissan</option>

<option value="Audi">Audi</option>

</select>

</body>

</html>

A screenshot of a computer

Description automatically generated

**List box**

* The **size attribute** tells you have many options you can have.
* If you have more size options listed, you will have a vertical scroll bar
* **Multiple attribute** allows to select multiple options by holding the ctrl key or on a mac holding the command key.

<!DOCTYPE html>

<html>

<body>

<label for="cars">Choose a car:</label>

<select id="cars" name="carlist" form="carform" size="4" multiple>

<option value="Telsa">Telsa</option>

<option value="Toyota">Toyota</option>

<option value="Nissan">Nissan</option>

<option value="Audi">Audi</option>

</select>

<p>Hold down the Ctrl (windows) or Command (Mac) button to select multiple options.</p>

</body>

</html>

A close-up of a computer

Description automatically generated

**Buttons**

* <button></button> tag – allows text and images
* <input type="button"> – only allows text
* Type attribute can be set to either:
  + Button or submit.
* If type attribute is **not** defined, nothing will happen when user clicks on button
* Button tag is used mostly with JavaScript to perform an action when clicked.
* The highlighted text is JavaScript

Please type this code:

<!DOCTYPE html>

<html>

<body>

<button type="button" onclick="alert('Hello world!')">Click Me!</button>

</body>

</html>

This should display

A white rectangle with black text

Description automatically generated

When user clicks this button, a pop-up appears saying hello world

A screenshot of a computer

Description automatically generated

**Password field**

* <input type="password"> - defines a password field using characters that are masked using asterisks (\*\*\*\*\*\*\*)
* **Size attribute** – limits the password width
* **Maxlength attribute** – limits how many characters a user can enter

For example:

<input type="password" name="secret" size="10" maxlength="10">

A white rectangular object with black dots

Description automatically generated

**Specialized text input**

Four input types exist for entering specific types of text:

url - For typing a URL

* <input type="url" id="homepage" name="homepage"><input type="url" id="homepage" name="homepage">

tel - For typing a telephone number

* <input type="tel" id="phone" name="phone" placeholder="123-456-7890">

email - For typing an email address

* <input type="email" id="email" name="email">

search - For typing search terms

* <input type="search" id="gsearch" name="gsearch">

Audio and Video

**Audio**

* Plays an audio file
* Audio file extensions are:
  + .aac
  + .mp3
  + .ogg **(which is both an audio and video file and doesn’t work on the safari browser)**
  + .wav

**Audio attributes**

* **autoplay** - Boolean attribute that makes the audio begin playing automatically.
* **controls** - Boolean attribute that displays audio controls for the user to play, pause, and control the volume.
* **loop** - Boolean attribute that replays the audio upon reaching the end of the audio.
* **muted** - Boolean attribute that initially mutes the audio.

Please type or copy the following:

<audio controls>

<source src="https://resources.zybooks.com/WebProgramming/tyops\_vivaldi-meets-hip-hop.mp3">

Your browser does not support the audio element.

</audio>

*\*\* Special note: Please note if you leave out or delete the word “controls”, the audio file will* ***not*** *display. \*\**

**Video**

* Plays a video file
* Video file extensions are:
  + .mp4
  + .webm
  + .ogg **(which is both an audio and video file and doesn’t work on the safari browser)**

**Video attributes**

* **autoplay** - Boolean attribute that makes the video begin playing automatically.
* **controls** - Boolean attribute that displays video controls for the user to play, pause, and control the volume.
* **loop** - Boolean attribute that replays upon reaching the end of the video.
* **muted** - Boolean attribute that initially mutes the video.
* **width** - Specifies the pixel width of the video's display area. **(video only)**

Please type or copy the following:

<video controls width="500">

<source src="https://resources.zybooks.com/WebProgramming/subway.mp4">

</video>

*\*\* Special note: Please note if you leave out or delete the word “controls”, the video file will display as an image. \*\**

**Embedding Web Pages**

* <iframe></iframe> tag allows a web page to be embedded in a rectangular area of the current web page.
* Must use a **src attribute** to specify the URL of the web page to display and the **width and height attributes** to define the width and height in pixels of the rectangular iframe

For example: YouTube

A screenshot of a computer

Description automatically generated

You click **Copy**

<iframe width="560" height="315" src="https://www.youtube.com/embed/qz0aGYrrlhU?si=ypbKhIgPJ9vBsLSy" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

You then copy the whole code and paste it in your code editor

Now let’s say you want to autoplay and mute this video (**by adding a 1, it turns it on**)

* You would add &autoplay=1&mute=1 add the end of the video url

<iframe width="560" height="315" src="https://www.youtube.com/embed/qz0aGYrrlhU?si=ypbKhIgPJ9vBsLSy &autoplay=1&mute=1" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

<script> and <style>

**Script**

* <script></script> tag is used to embed a client-side script (JavaScript).
* type attribute – is optional and indicates content type
* src attribute – provides the URL of the external file containing JavaScript
* This tag will not be displayed for in the browser.

See example below:

<script src="https://example.com/interactive\_content.js"></script>

<script type="text/javascript">

alert("Hello, World!");

</script>

**Style**

* <style></style> tag is used to define style information (CSS) for a document.
* type attribute – is optional and indicates content type
* This tag will not be displayed for in the browser.

<html>

<head>

<style>

p {color:blue;}

</style>

</head>

<body>

<p>A paragraph.</p>

</body>

</html>

HTML developer guidelines

Best practices

* Indenting code
  + For example:

<ul>  
 <li>Apples</li>  
 <li>Bananas</li>  
 <li>Oranges</li>  
</ul>

* Closing tags always have </>
  + <p>For example </p>
* Use double quotes for attribute values
  + For example: <ol type="A">
* Use all lowercase for tags and attributes
* Always have labels and placeholders

<label for="phone">Enter a phone number:</label><br><br>

<input type="tel" id="phone" name="phone" placeholder="123-456-7890">

XML

**Please take this quiz**

[D276 XML Fundamentals](https://wguashkin.github.io/D276-Practice-Quiz-3/)

* XML stands for eXtensible Markup Language.
* XML was designed to store and transport data.
* XML was designed to be both human- and machine-readable.

**XML Syntax rules vs HTML**

* XML Documents Must Have a Root Element

See example below

<root>  
 <child>  
 <subchild>.....</subchild>  
 </child>

</root>

* XML prolog is optional and does **NOT** have a closing tag
* XML doctype declaration is **NOT** required
  + <?xml version="1.0" encoding="UTF-8"?>
* All XML Elements Must Have a closing Tag **(except the XML prolog)**
* Not all HTML elements have a closing tag
  + For example - <br>, <hr>, <link>, etc
* XML Tags are case sensitive
  + For example - <mark>This is a test</mark>
* HTML tags are not case sensitive
* XML elements **must be** Properly Nested
* HTML elements **don’t have** to be properly nested
  + For example - <b><i>This text is bold and italic</i></b>
* XML Attribute Values must Always be Quoted
  + <note date="10/23/2023">
* XML Entities
  + Are required to use these symbols
    - For example - I&apos;m hungry

A screenshot of a phone

Description automatically generated

* HTML **does not** require these symbols
* XML Comments and HTML comments are the **same**
  + <!-- This is a comment -->
* XML White-space (it keeps all the spacing)  **\*\* White space is the spaces between words \*\***
  + Hello World
* HTML White-space (is ignored)
  + Hello World

Some other differences Between XML and HTML

* XML was designed to **carry** data - with focus on what data is
* HTML was designed to **display** data - with focus on how data looks
* XML tags are **not** predefined like HTML tags are

XML Elements vs XML attributes vs XML values

* XML element - basic building block of the XML document and It is used as a container to store text elements, attributes, and media objects
  + For example - <book>Harry Potter</book>
    - book is the XML element
* XML attributes – part of an XML element and it enhances the properties of the XML element.
  + For example - <book category="children">
    - category is the XML attribute
* XML values - is stored in a table as a value of a column defined with the XML data type must be a well-formed XML document and must always be quoted.
  + For example - <book category="children">
    - children is the XML value

CSS

* Stands for **C**ascading **S**tyle **S**heets
* describes how HTML elements are to be displayed on screen
* is the language we use to style a Web page.

**CSS Syntax**

**A green rectangular sign with white text

Description automatically generated**

* The selector points to the HTML element you want to style.
* The declaration block contains one or more declarations separated by semicolons.
* Each declaration includes a CSS property name and a value, separated by a colon.
* Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.
* The browser's default styling colors the text black.

**CSS Basic Selectors**

* Simple selectors (select elements based on name, id, class)
  + Element selector
    - p { color: blue; }
  + Class selector
    - .box { text-align: center; }
  + ID selector
    - #new {border: 1px solid black;}

**Three Ways to Insert CSS**

* **Inline CSS** (inside the tag)
  + For example – <p style="color:green;">I'm green</p>
* **Internal CSS** (also known as embedded css)
  + For example -   
    *\*\* special note: remember the style element is inside the head element \*\**  
    <style>

p {color: orange;}

</style>

<p>I'm orange.</p>

* **External CSS** (on a separate document)

For example   
*\*\* special note: remember the link element is inside the head element and the external css is on a separate document*

* + <link rel="stylesheet" href="mystyle.css">

p { color: purple; }

* The HTML element here would be a on a separate page

<p>I'm purple.</p>

**CSS Specificity**

<https://www.w3schools.com/css/css_specificity.asp>

**Please take this quiz**

[D276 CSS Specificity](https://wguashkin.github.io/D276-CSS-Specificity-Quiz/)

* Every CSS selector has its place in the specificity hierarchy.
* Think of specificity as a score/rank that determines which style declaration is ultimately applied to an element.

There are **four categories** which define the specificity level of a selector:  
  
Remember the phrase “**I**gloos **I**n **C**old **E**nvironments”

* **Inline styles** - Example: <h1 style="color: pink;">
* **IDs** - Example: #navbar
* **Classes, pseudo-classes, attribute selectors** - Example: .test, :hover, [href]
* **Elements and pseudo-elements** - Example: h1, ::before

See if you can figure out what happens here

Type or copy the following code:

<html>

<head>

<style>

#demo {color: blue;}

.test {color: green;}

p {color: red;}

</style>

</head>

<body>

<p id="demo" class="test" style="color: pink;">Hello World!</p>

</body>

</html>

In this example, Hello World would be pink because **inline style** ranks the highest over IDs, Classes, and Elements

This is what would display.

Hello World!

Type or copy the following code:

<!DOCTYPE html>

<html>

<head>

<style>

#demo {color: blue;}

.test {color: green;}

p {color: red;}

</style>

</head>

<body>

<p id="demo" class="test">Hello World!</p>

</body>

</html>

In this example, Hello World would be blue because the **ID** selector ranks the highest because there is **no** inline style and ID ranks above Classes, and Elements

This is what would display.

Hello World!

<!DOCTYPE html>

<html>

<head>

<style>

.test {color: green;}

p {color: red;}

</style>

</head>

<body>

<p class="test">Hello World!</p>

</body>

</html>

This is what would display.

Hello World!

What if we had multiple classes selector that are the same?

<!DOCTYPE html>

<html>

<head>

<style>

.test {color: blue;}

.test {color: green;}

.test {color: red;}

</style>

</head>

<body>

<p class="test">Hello World!</p>

</body>

</html>

***Then it would always take the last value listed***

This is what would display.

Hello World!

What if we had multiple classes selector but an element in front of it?

<!DOCTYPE html>

<html>

<head>

<style>

.test {color: blue;}

p.test {color: green;}

.test {color: red;}

</style>

</head>

<body>

<p class="test">Hello World!</p>

</body>

</html>

This is what would display.

Hello World!

Then it would override the last value and take on a higher value since it’s two elements.

|  |  |  |
| --- | --- | --- |
| **Selector** | **Specificity Value** | **Calculation** |
| p | 1 | 1 |
| p.test | 11 | 1 + 10 |
| .test | 10 | 10 |

* The selector with the highest specificity value will win and take effect!

**CSS :nth-child() Selector**

<!DOCTYPE html>

<html>

<head>

<style>

/\* Selects the second element of div siblings \*/

**div:nth-child(2)** {

background: red;

}

/\* Selects the second li element in a list \*/

**li:nth-child(2)** {

background: lightgreen;

}

/\* Selects every third element among any group of siblings \*/

**:nth-child(3)** {

background: yellow;

}

</style>

</head>

<body>

<div>

<p>This is some text.</p>

</div>

<div>

<p>This is some text.</p>

</div>

<div>

<p>This is some text.</p>

</div>

<ul>

<li>First list item</li>

<li>Second list item</li>

<li>Third list item</li>

<li>Fourth list item</li>

<li>Fifth list item</li>

</ul>

</body>

</html>

**A list of text on a white background

Description automatically generated**

**CSS :first-child Selector**

<!DOCTYPE html>

<html>

<head>

<style>

**p:first-child** {

background-color: yellow;

}

</style>

</head>

<body>

<p>This paragraph is the first child of its parent (body).</p>

<p>This paragraph is not the first child of its parent (body).</p>

<div>

<p>This paragraph is the first child of its parent (div).</p>

<p>This paragraph is not the first child of its parent (div).</p>

</div>

</body>

</html>

A close up of text

Description automatically generated

**CSS :last-child Selector**

<!DOCTYPE html>

<html>

<head>

<style>

**p:last-child {**

background: #ff0000;

}

</style>

</head>

<body>

<p>The first paragraph.</p>

<p>The second paragraph.</p>

<p>The third paragraph.</p>

<p>The fourth paragraph.</p>

</body>

</html>

**A text on a white background

Description automatically generated**

<!DOCTYPE html>

<html>

<head>

<style>

**p:nth-last-child(2)** {

background: orange;

}

</style>

</head>

<body>

<p>The first paragraph.</p>

<p>The second paragraph.</p>

<p>The third paragraph.</p>

<p>The fourth paragraph.</p>

</body>

</html>

A text on a white background

Description automatically generated

**CSS :nth-child(even) and :nth-child(odd)**

<!DOCTYPE html>

<html>

<head>

<style>

p:nth-child(odd) {

background: red;

}

p:nth-child(even) {

background: lightgreen;

}

</style>

</head>

<body>

<p>The first paragraph.</p>

<p>The second paragraph.</p>

<p>The third paragraph.</p>

<p>The fourth paragraph.</p>

<p>The fifth paragraph.</p>

</body>

</html>

**A red and green text

Description automatically generated**

**CSS :link Selector**

<https://www.w3schools.com/cssref/sel_link.php>

<!DOCTYPE html>

<html>

<head>

<style>

/\* unvisited link \*/

a:link {

color: green;

}

/\* visited link \*/

a:visited {

color: blue;

}

/\* mouse over link \*/

a:hover {

color: red;

}

/\* selected link \*/

a:active {

color: yellow;

}

</style>

</head>

<body>

<p>Mouse over and click the link: <a href="https://www.w3schools.com">w3schools.com</a></p>

</body>

</html>

**CSS comment**

* /\* This is a comment \*/

**Please take this quiz**

[D276 Finding the Total Width and Height](https://wguashkin.github.io/D276-Finding-Total-Width-and-Height/)

**Width** = left and right

**Height** = top and bottom

**CSS (inner space)**

* CSS has properties for specifying the padding for each side of an element:
  + padding-top
  + padding-right
  + padding-bottom
  + padding-left

If the padding property has **four values**:

* For example - padding: 25px 50px 75px 100px;
  + **top** padding is 25px.
  + **right** padding is 50px.
  + **bottom** padding is 75px.
  + **left** padding is 100px.

If the padding property has **three values**:

* For example - padding: 25px 50px 75px;
  + **top** padding is 25px.
  + **right** and **left** padding is 50px.
  + **bottom** padding is 75px.

If the padding property has **two values**:

* For example - padding: 25px 50px;
  + **top** and **bottom** padding is 25px.
  + **right** and **left** padding is 50px.

If the padding property has **one value**:

* For example - padding: 25px;
  + All **four** padding is 25px.

**CSS Margins (outer space)**

*Special note:* ***Same information as above only it’s outer space instead of inner space***

**CSS Box Model**

**A screenshot of a computer

Description automatically generated**

Explanation of the different parts:

* **Content** - The content of the box, where text and images appear.
* **Padding** – inner space around the around the content.
* **Border** - A border that goes around the padding and content.
* **Margin** – outer space around the border.

**CSS combinators**

<https://www.w3schools.com/css/css_combinators.asp>

**Please do this quiz**  
[D276 CSS Combinators](https://wguashkin.github.io/D276-Combinators-Quiz/)

There are four different combinators in CSS:

* descendant selector (**space**)
  + For example: div p
* child selector (**>**)
  + For example: div > p
* adjacent sibling selector (**+**)
  + For example: div + p
* general sibling selector (**~**)
  + For example: div ~ p

**Descendant selector** (space)

* must contain both elements to apply CSS property and value
* we have div and p elements so if we have a div and p elements it will have a background of yellow

For example

div p {

background-color: yellow;

}

Type or copy the following code below:

<!DOCTYPE html>

<html>

<head>

<style>

div p {

background-color: yellow;

}

</style>

</head>

<body>

<h2>Descendant Selector</h2>

<p>The descendant selector matches all elements that are descendants of a specified element.</p>

**<div>**

**<p>**Paragraph 1 in the div.**</p>**

**<p>**Paragraph 2 in the div.**</p>**

<section>**<p>**Paragraph 3 in the div.**</p>**</section>

**</div>**

<p>Paragraph 4. Not in a div.</p>

<p>Paragraph 5. Not in a div.</p>

</body>

</html>

This will display: *Notice how only paragraphs 1, 2, and 3 are yellow because they both have div and p elements. Paragraphs 4,5 are* ***not*** *surround by a div so they are* ***not*** *yellow*

**Descendant Selector**

The descendant selector matches all elements that are descendants of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

**Child Selector (>)**

* is the opposite of descendant selector.
* We have div greater than p element so this means, if an element is in front of the 2nd selector listed, it will **not** apply the CSS property and value.

For example

div **>** p {

background-color: yellow;

}

Type or copy the following code below:

<!DOCTYPE html>

<html>

<head>

<style>

div > p {

background-color: yellow;

}

</style>

</head>

<body>

<h2>Child Selector</h2>

<p>The child selector (>) selects all elements that are the children of a specified element.</p>

<div>

<p>Paragraph 1 in the div.</p>

<p>Paragraph 2 in the div.</p>

**<section>** <p>Paragraph 3 in the div (inside a section element).</p> **</section>**

<p>Paragraph 4 in the div.</p>

</div>

<p>Paragraph 5. Not in a div.</p>

<p>Paragraph 6. Not in a div.</p>

</body>

</html>

This will display: *Notice how only paragraphs 1, 2, and 4 are yellow because they both have div and p elements.* ***Paragraphs 3 has an element in front of the 2nd element selector.*** *Paragraphs 5 and 6 do not have div so the CSS property and values would* ***not*** *apply here.*

## Child Selector

The child selector (>) selects all elements that are the children of a specified element.

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div (inside a section element).

Paragraph 4 in the div.

Paragraph 5. Not in a div.

Paragraph 6. Not in a div.

**Adjacent Sibling Selector (+)**

* When the 1st CSS selector uses a closing tag, **only** the adjacent (right next to) 2nd CSS selector will be applied that property and value.

For example

div **+** p {

background-color: yellow;

}

Type or copy the following code below:

<!DOCTYPE html>

<html>

<head>

<style>

div + p {

background-color: yellow;

}

</style>

</head>

<body>

<h2>Adjacent Sibling Selector</h2>

<p>The + selector is used to select an element that is directly after another specific element.</p>

<p>The following example selects the first p element that are placed immediately after div elements:</p>

<div>

<p>Paragraph 1 in the div.</p>

<p>Paragraph 2 in the div.</p>

</div>

**<p>Paragraph 3. After a div.</p>**

<p>Paragraph 4. After a div.</p>

<div>

<p>Paragraph 5 in the div.</p>

<p>Paragraph 6 in the div.</p>

</div>

**<p>Paragraph 7. After a div.</p>**

<p>Paragraph 8. After a div.</p>

</body>

</html>

This will display: *Notice how only paragraphs 3 and 7 are yellow because when 1st element (which is div) is closed, the adjacent 2nd element (which is p) is yellow only. It’s right after the closing of the div element. Paragraphs 1, 2, 4, 5, 6, and 8 do* ***not*** *have a closing tag for the 1st element.*

## Adjacent Sibling Selector

The + selector is used to select an element that is directly after another specific element.

The following example selects the first p element that are placed immediately after div elements:

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. After a div.

Paragraph 4. After a div.

Paragraph 5 in the div.

Paragraph 6 in the div.

Paragraph 7. After a div.

Paragraph 8. After a div.

**General Sibling Selector (~)**

* Is the opposite of adjacent sibling selector
* Everything after the 1st element that closes **regardless** if it’s adjacent of 2nd element, applies the CSS property and value

For example

div **~** p {

background-color: yellow;

}

Type or copy the following code below:

<!DOCTYPE html>

<html>

<head>

<style>

div ~ p {

background-color: yellow;

}

</style>

</head>

<body>

<h2>General Sibling Selector</h2>

<p>The general sibling selector (~) selects all elements that are next siblings of a specified element.</p>

<p>Paragraph 1.</p>

<div>

<p>Paragraph 2.</p>

</div>

**<p>Paragraph 3.</p>**

<code>Some code.</code>

**<p>Paragraph 4.</p>**

</body>

</html>

This will display: *Notice how everything after paragraph 2 is yellow because the div is closed and the CSS property and value are applied. Paragraphs 1 and 2 do not have a closing tag so it does* ***not*** *take the CSS property and value.*

**General Sibling Selector**

The general sibling selector (~) selects all elements that are next siblings of a specified element.

Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

**Color properties**

* CSS defines 140 color names.
* RGB (stands for Red, Green, and Blue) color value and each intensity for red, green, and blue is between 0 and 255, where 0 is the lowest intensity and 255 is the highest.
  + For example: rgb(0, 0, 0) is black, rgb(0, 0, 255) is blue, rgb(255, 255, 0) is yellow, and rgb(255, 255, 255) is white.
* Hexadecimal color - specifies a color using the #RRGGBB format by indicating the red, green, and blue intensities and each intensity for red, green, and blue is between 00 and FF hexadecimal numbers, where 00 is the lowest intensity and FF is the highest
  + For example: Ex: #FF0000 is red, #00FF00 is green, #0000FF is blue, #000000 is black, and #FFFFFF is white.

**CSS Float property**

* Float – will float right or left and allowing text to flow around the image.
  + **none -** Element does not float (**default value**)
  + **left -** Element floats to parent container's left side
  + **right -** Element floats to parent container's right side

See example: see what happens when you type float:left;

<https://www.w3schools.com/css/tryit.asp?filename=trycss_layout_float>

* Clear – This property prevents elements from floating around an image
  + **none** - Elements allowed to float (**default value**)
  + **both** - No elements allowed to float
  + **left** - No element allowed to float on parent container's left side
  + **right** - No element allowed to float on parent container's right side

**CSS Display**

* The display property specifies the display behavior (the type of rendering box) of an element.
  + **none** - Hides the element from being displayed, like style elements.
  + **inline** - Displays the element as an inline element, like span or a elements.
  + **block** - Displays the element as a block element, like p, h1, or div elements.
  + **inline-block** - Displays the contents of the element as a block element, but formats the element as an inline element.
  + **list-item** - Displays the contents of the element as a list item element.

**Font Properties**

* The **font-family** property specifies the font family, such as "Times New Roman" or serif.
* The **font-size** property changes the font size, such as 120%, small, or 12px.
* The **font-weight** property specifies the font weight, such as normal or bold.
* The **font-style** property changes the text style, such as normal, italic, or oblique.
* The **font-variant** property specifies the variant of the text, such as normal or small-caps.
* The **font** property is shorthand for setting several font properties at the same time. Ex: font: italic 12pt Georgia, serif;

A screenshot of a computer

Description automatically generated

**CSS variables**

* Has access to the DOM
  + highest DOM element is the <html> element.
* you can create variables with local or global scope which is in the **:root** selector
* CSS variable is defined with **two dashes** preceding the variable name.
  + For example:  --my-variable: red;
* When accessing the css variable, you must use the var() function
  + For example var(--my-variable);

<!DOCTYPE html>

<html>

<head>

<style>

:root {

--blue: #1e90ff;

--white: #ffffff;

}

body {

background-color: var(--blue);

}

h2 {

border-bottom: 2px solid var(--blue);

}

.container {

color: var(--blue);

background-color: var(--white);

padding: 15px;

}

</style>

</head>

<body>

<h1>Using the var() Function</h1>

<div class="container">

<h2>Lorem Ipsum</h2>

<p>This a test</p>

<p>This a test</p>

</div>

</body>

</html>

**A blue rectangular box with white text

Description automatically generated**

**Font Size**

An **absolute size** is a size that is fixed and independent of other CSS sizes. Absolute size units include:

* cm - centimeters
* mm - millimeters
* in - inches
* px - pixels (1px = 1/96in) - **is the smallest size**
* pt - points (1pt = 1/72in)
* pc - pica (1pc = 12pt)

A **relative size** is a size that is relative to another size. Some common relative size units include:

* em - Relative to the element's font size. Ex: 2em = 2 × current font size.
* rem - Relative to the root element's font size. Ex: 1.5rem = 1.5 × <html> element's font size.
* vw - 1% of the viewport's width. Ex: 10vw = 10% of browser's width.
* vh - 1% of the viewport's height. Ex: 5vh = 5% of browser's height.
* % - Percentage of the element's font size. Ex: 120% = 20% larger than the current font size.
* Most web browsers use a default font size of 16px.

**How do you calculate the value of the line height in pixels?**

To calculate the value of the line height in pixels, you can multiply **font size** by **line height**:

p {

font-size: 9px;

line-height: 2em;

}

In this example,

the font size of the element is set to 9 px and the line height is set to 2em. To calculate the value of the line height in pixels, you can multiply font size by line height: 9px x 2em = 18px.

**Flexbox**<https://www.w3schools.com/css/css3_flexbox.asp>

* 2 types of layouts
  + **Fixed** - resizing the browser **does not** change the width of the web page contents.
    - Uses pixels (px)
  + **Fluid** - make better use of the available space than fixed layouts
    - Uses percentages
* **Wireframe** - a blueprint for a web page that shows how future content will be arranged.

**Flex-wrap** **property**- in flexbox determines whether the flex items should wrap or not when there's insufficient space within their container.

**nowrap**: Prevents items from wrapping to the next line, keeping them in a single line (default behavior).

A computer code with text

Description automatically generated with medium confidence

A blue and white squares with black numbers

Description automatically generated

**wrap**: Allows flex items to wrap onto multiple lines if necessary, accommodating smaller screen sizes or limited space.

A computer code on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

**wrap-reverse**: Similar to wrap, but wraps items in the reverse order.

A blue and white squares with black numbers

Description automatically generated

More information <https://www.w3schools.com/css/css3_flexbox.asp>

**Flexbox container and items**

* **Flexbox container** - Accurately defined as an element with the CSS property display set to flex to create a block-level flex container or inline-flex to create an inline flex container.
  + Example: **<div style="display: flex">** demonstrates how to create a flex container using inline styles.

**Flex container properties**

* The **flex-direction** property defines the direction of flex items within the container using values:

Values for flex-direction: (see below)

* **row (default)**: Places flex items in a row from left to right.
* **row-reverse:** Places flex items in a row from right to left.
* **column:** Places flex items in a column from top to bottom.
* **column-reverse:** Places flex items in a column from bottom to top.

**A screenshot of a computer

Description automatically generated**

**Grid container and grid items**

* **Grid layout** is a CSS layout mode that organizes a web page into a rectangular grid, facilitating the positioning of various page elements.
* **Grid container** is an element whose CSS property display is set to grid to create a block-level grid container or inline-grid to create an inline grid container..
  + **Two display types**
    - **Grid**  
       .grid-container {display: grid;}
    - **Inline-grid**  
       .grid-container {display: inline-grid;}
* **Grid-template-columns = width**
  + When using grid-template-columns, the auto value allows columns to adjust their width based on content or available space.
  + For example:

#grid-container {

width: 300px;

display: grid;

grid-template-columns: auto auto auto;

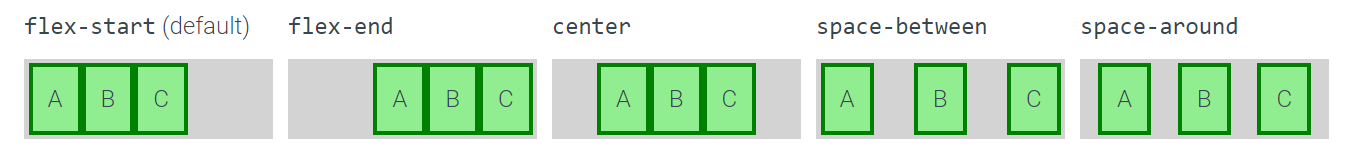
}

* **Gap** property defines the space between flex items  
  + If **two values** are specified, the first value represents the gap between **rows**, and the second value represents the gap between **columns**
* For example: gap: 5px 25px;
  + 5px is the **row** gap and 25px is the **column** gap.

A green rectangles on a white background

Description automatically generated

* The **justify-content** property justifies the flex items within the container using values:
  + **space-between**: Items are evenly distributed in the container, with the first item aligning to the start and the last item aligning to the end.
  + **space-around**: Items are evenly distributed in the container with equal space around them.



**CSS Positioning**

<https://www.w3schools.com/css/css_positioning.asp>

* **static** - Static positioning is the default positioning. It does **not** move up, down, left or right.

For example:

div.relative {

position: static;

border: 3px solid #73AD21;

}

* **relative** - Relative positioning positions the element relative to the element's default position. The element moves from **not** to the direction listed.

For example: This element move **from** the left to the right.

div.relative {

position: relative;

left: 30px

border: 3px solid #73AD21;

}

* **fixed** - Fixed positioning positions the element relative to the viewport in a fixed location. Think of how an ad on webpage follows you as you scroll.

For example:

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

* **absolute** - Absolute positioning positions the element relative to the nearest positioned ancestor. They are removed from the normal flow and can also overlap elements.

For example:

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

**Z-index property**

* specifies the stack order of an element.
* An element with greater stack order is always in front of an element with a lower stack order.
* Elements with **higher z-index** **values** are placed on top of elements with **lower z-index** **values**.

<!DOCTYPE html>  
<html>  
<head>  
<style>

img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

</style>

</head>

<body>

<h1>The z-index Property</h1>

<img src="w3css.gif" width="100" height="140">

<p>Because the image has a z-index of -1, it will be placed behind the heading.</p>

</body>

</html>

How will this display?

**A close-up of a logo

Description automatically generated**

Now change it to 0 or a positive number, how will it display?

A close-up of a logo

Description automatically generated

div {

width: 100px;

height: 100px;

position: absolute;

}

#orange {

background-color: orange;

z-index: 3;

left: 10px;

top: 10px;

}

#blue {

background-color: blue;

color: white;

z-index: 2;

left: 30px;

top: 30px;

}

#green {

background-color: green;

z-index: 1;

left: 50px;

top: 50px;

}

<div id="orange">Go orange!</div>

<div id="blue">Go blue!</div>

<div id="green">Go green!</div>

A stack of colorful squares

Description automatically generated

**CSS animation**

<https://www.w3schools.com/css/css3_animations.asp>

**Please take this quiz**

[D276 CSS Animation](https://wguashkin.github.io/D276-CSS-Animation-Quiz/)

* **@keyframes** rule - the animation will gradually change from the current style to the new style at certain times. There can be 2 types of keyframe rules
  + From and to
  + Percentages where 0% is from, 100% is to, and 50% is the halfway point
  + When an animation is finished, it goes back to its **original style outside of keyframes**.
* To create an animation, you must have 2 CSS properties.
  + **animation-name** – this property names the keyframe list associated with the animation
  + **animation-duration** – this property has 2 types of lengths
    - seconds (s)
    - milliseconds (ms)
* **animation-delay –** thispropertyspecifies a delay for the start of an animation.   
  *\*\* Special note: Negative delays are allowed. If using negative values, the animation will start as if it had already been playing for N seconds and start bottom right-hand corner \*\**
* **animation-iteration-count –** this propertyspecifies the number of times an animation should run. You can have 1 all the way to infinite (never stops). By default, it will only run once
* **animation-direction –** thisproperty specifies whether an animation should be played forwards, backwards or in alternate cycles.
  + This property has 4 types of directions
    - **normal** - The animation is played as normal (forwards). **This is default**
      * clockwise motion
    - **reverse** - The animation is played in reverse direction (backwards)
      * counter-clockwise motion
    - **alternate** - The animation is played **forwards first**, then **backwards**
      * **clockwise**, then counter-clockwise motion
        + ***\*\*\* special note: animation-iteration-count value must be 2 or more for this to work \*\*\****
    - **alternate-reverse** - The animation is played **backwards first**, then **forwards**
      * **counter-clockwise**, then clockwise motion
        + ***\*\*\* special note: animation-iteration-count value must be 2 or more for this to work \*\*\****
* **animation-timing-function –** this property specifies the speed curve of the animation.  
  + **ease** - Specifies an animation with a slow start, then fast, then end slowly (**this is default**)
  + **linear** - Specifies an animation with the same speed from start to end.
  + **ease-in** - Specifies an animation with a slow start.
  + **ease-out** - Specifies an animation with a slow end.
  + **ease-in-out** - Specifies an animation with a slow start and end.

**Mobile design**

* A responsive web design will automatically adjust for different screen sizes.
* Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, ensuring an appealing appearance across a range of devices, including desktops, tablets, and phones.

**CSS media queries and rules**

* CSS statement that starts with the @ character and instructs the CSS engine how to behave. When a media query in a stylesheet matches, the @media's inner rules are defined.

<!DOCTYPE html>

<html>

<head>

<style>

**body** {

**background-color: pink**;

}

**@media screen and (min-width: 480px) {**

**body {**

**background-color: lightgreen**;

}

}

</style>

</head>

<body>

<h1>Resize the browser window to see the effect!</h1>

<p>What will happen to the color</p>

</body>  
</html>

**Anything over 480px, it’s lightgreen**

A screenshot of a computer

Description automatically generated

**As soon as the width is less than 480px, it’s pink**

A screenshot of a computer error

Description automatically generated

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

**body {**

**background-color: yellow**;

}

@media only screen and (**min-width**: 600px) {

**body {**

**background-color: lightblue**;

}

}

@media only screen and (**max-width**: 400px) {

**body {**

**background-color: red**;

}

}

</style>

</head>

<body>

<h1>The @media Rule</h1>

<p>What color will it display</p>

</body>  
</html>

**What does this mean?**

**Min-width** means it **starts at a number**  
**Max-width** means it **ends at a number**

When the width is > 600px = lightblue  
When the width is < 400px = red   
When the width is between 400px and 600px = yellow

**Bootstrap**

* one of the most popular free, open-source frameworks
* solves the issue of responsive web design
* uses a **CDN** which stands for content delivery network
* Two main components of Bootstrap
  + Icons
  + Typography
    - overall appearance of the text on your website

**Debugging (Google Chrome)**

Browser developer tools are used to apply changes instantly.

* What are the three components of the debugging process?
  + **Find**: Discover the unexpected behavior you want to change
  + **Identify**: Determine what is causing the unexpected behavior
  + **Fix**: Implement a fix that will prevent the unexpected behavior from happening again.

Click à [Learn Chrome Web Development Tool](https://www.linkedin.com/learning/learning-chrome-web-developer-tools-2/streamlining-development-with-chrome-devtools-22186777?u=2045532)

To access the devtools for Google chrome: **right click and click on inspect**

**Element panel**

* The first panel is the **Elements panel**
* In the elements panel is the DOM
* The Document Object Model (**DOM** defines the logical structure of documents and the way a document is accessed and manipulated.
* You can change DOM by clicking the box with the arrow to edit what’s in the code.

A screenshot of a computer

Description automatically generated

**Console panel**

* Run and displays the JavaScript code
  + For example: alert("Hello World!");

A screenshot of a computer

Description automatically generated

* **Red means there is an error**
* **Yellow mean there is a warning**

**Sources Panel**

* View files.

A screenshot of a computer

Description automatically generated

* Debug JavaScript.

A screenshot of a computer

Description automatically generated

**Network Panel**

* Use to verify the resources of a site have been downloaded or uploaded correctly.
* Also used to verify the properties of the resources, for example, the size of image, HTP headers, etc.
* This section displays all network requests made by the webpage, including their status, size, and timing

**Performance Panel**

* This panel allows you to record and analyze the runtime performance of your web application
* You can identify performance bottlenecks, memory leaks, and optimize code execution.

**Memory Panel**

* It used to determine memory leak issues
* Click the 3 dots, then more tools, then task manager

**A screenshot of a computer

Description automatically generated**

You should see the Task Manager

**A screenshot of a computer

Description automatically generated**

**Application Panel**

* View and edit local storage
* View and edit cookies
* View cache data
* Debug background services

**Security Panel**

* Determines if website is secure using https.
* Website has a padlock in web address.

**A screenshot of a computer

Description automatically generated**

**Lighthouse Panel**

* is a tool that helps developers improve the quality and performance of their web pages by running audits and providing actionable suggestions

**Debugging (Firefox)**

To access the devtools for Firefox: click F12 (PC only) or right click and click on inspect

Click à [Learn Firefox Web Development Tool](https://www.linkedin.com/learning/learning-enterprise-web-application-performance/using-firefox-developer-tools?u=2045532)

**Inspector panel**

* The first panel is the **Inspector panel**
* In the inspector panel is the DOM
* The Document Object Model (**DOM** defines the logical structure of documents and the way a document is accessed and manipulated.
* You can change DOM by clicking the box with the arrow to edit what’s in the code.

A screenshot of a computer

Description automatically generated

**Debugger panel**

* enables you to step through JavaScript code and examine or modify its state to help track down bugs.

**A screen shot of a computer

Description automatically generated**

**Console panel**

* Run and displays the JavaScript code
  + For example: alert("Hello World!");

A screenshot of a computer

Description automatically generated

**Network Panel**

* Use to verify the resources of a site have been downloaded or uploaded correctly.
* Also used to verify the properties of the resources, for example, the size of image, HTP headers, etc.

**Style Editor Panel**

* provides you with a space to edit and view the CSS used on the site.
* You can create your own CSS file or import it from your system

**Performance Panel**

* Runtime performance is how your page performs when it is running, as opposed to loading and analyze frames per second

**Memory Panel**

* It used to determine memory leak issues

Click on the 3 bars and then **more tools**

A screenshot of a computer

Description automatically generated

Then click **Task Manager**

A screenshot of a computer

Description automatically generated

A white rectangular object with a black border

Description automatically generated

**Storage Panel**

* View and edit local storage
* View and edit cookies
* View cache data

**A screenshot of a computer

Description automatically generated**

**Accessibility Panel**

Check for accessibility issues.

You can check for accessibility issues by clicking the drop-down menu next to: Check for issues. The available menu items include:

* **None** — Don’t show the possible list of issues.
* **All Issues** — Check for all types of issues.
* **Contrast** — Check for issues with visual contrast.
* **Keyboard** — Check for issues with navigating via a keyboard.
* **Text Labels** — Check for issues with missing text labels.

**A screenshot of a computer

Description automatically generated**

**Application Panel**

* provides tools for inspecting and debugging modern web apps (aka Progressive Web Apps)

**A screenshot of a computer

Description automatically generated**

**Debugging (Safari)**

<https://developer.apple.com/safari/tools/>

Click à [Using Safari Developer Tools](https://www.linkedin.com/learning/learning-enterprise-web-application-performance/using-safari-developer-tools?u=2045532)

**Element panel**

* The first panel is the **Elements panel**
* In the elements panel is the DOM
* The Document Object Model (**DOM** defines the logical structure of documents and the way a document is accessed and manipulated.
* You can change DOM by clicking the box with the arrow to edit what’s in the code.

A screenshot of a computer

Description automatically generated

**Console panel**

* Run and displays the JavaScript code
  + For example: alert("Hello World!");
* Views logs, errors, and warnings displays to webpage
* **Red means there is an error**
* **Yellow mean there is a warning**

A screenshot of a computer

Description automatically generated

**Sources Panel**

* View files.

A screenshot of a computer

Description automatically generated

* Debug JavaScript and troubleshoot script files.

**Network Panel**

**A screenshot of a computer

Description automatically generated**

* Use to verify the resources of a site have been downloaded or uploaded correctly.
* Also used to verify the properties of the resources, for example, the size of image, HTP headers, etc.
* This section displays all network requests made by the webpage, including their status, size, and timing

**Timelines Panel**

**A screenshot of a computer

Description automatically generated**

* Understand all the activity that occurs on an open web page, such as network requests, layout and rendering, JavaScript events, memory, and CPU impact.
* Everything is neatly plotted on a timeline or recorded by frame, helping you discover ways to optimize your site.

**Storage Panel**

**A screenshot of a computer

Description automatically generated**

* View and edit local storage
* View and edit cookies
* View cache data
* View indexed databases
* View application cache

**Graphics Panel**

**A screenshot of a computer

Description automatically generated**

* Preview animation keyframes and their classes from HTML5 canvas, JavaScript animations, CSS animations, and CSS transitions.
* Dial in the motion and the visual design of web pages.

**Layers Panel**

**A screenshot of a computer

Description automatically generated**

* Visualize compositing layers in 3D to understand where layers are generated and in what order they'll render.
* Use layers to help find unexpected memory consumption or excessive repaints on a web page.

**Audit Panel**

**A screenshot of a computer

Description automatically generated**

* Performs audits against a web page to certify that common code and accessibility errors are addressed.
* Confirm that a web page follows design guidelines and specifications of modern web pages.

**Chart of each panel   
\*\*\* (**please note it’s **not** in the same order as listed in the browser**) \*\*\*\***

**\*\*\* (**please also note: Application panel for Firefox is **NOT** the same as Google Chrome) \*\*\*\*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | The DOM is located | JavaScript is located | View files  \*View JS only with Debugger\* | Displays all network requests to see if they downloaded and uploaded correctly | How your page performs when it is running | It used to determine memory leak issues | View and edit local storage  View and edit cookies  View cache data  Debug background services |
| **Chrome** | Elements | Console | Sources | Network | Performance | Memory | **Application** |
| **Firefox** | **Inspector** | Console | **Debugger** | Network | Performance | Memory | \* Storage |
| **Safari** | Elements | Console | Sources | Network | **Not used** | **Not used** | Storage |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Determines if website is secure using https. | provides you with a space to edit and view the CSS used on the site and import to browser | Check for issues for the following  Contrast Keyboard Text Labels | helps developers improve the quality and performance of their web pages | all the activity that occurs on an open web page, such as network requests, layout and rendering, JavaScript events, memory, and CPU impact | Preview animation keyframes and their classes from HTML5 canvas, JavaScript animations, CSS animations, and CSS transitions | Visualize compositing layers in 3D to understand where layers are generated and in what order they'll render. |
| **Chrome** | Security | **Not used** | **Not used** | Lighthouse | **Not Used** | **Not Used** | **Not Used** |
| **Firefox** | **Not used** | Style Editor | Accessibility | Audit | **Not Used** | **Not Used** | **Not Used** |
| **Safari** | **Not used** | **Not used** | **Not used** | Audit | Timelines | Graphics | Layers |

**Sample Questions:**

* How would code the following:  
  Sample 1  
    
  **Sample 2**  
  *Sample 3*
* What are the two main components of Bootstrap?
* What is the overall appearance of the text called?
* What 2 CSS properties must be defined to create an animation?
* The CSS property animation-duration can only be two types of lengths?
* Which XML entity reference is used for an apostrophe (')?
* Which heading tag is similar to the <p> tag?
* What default shape for unordered lists type?
* Difference between XML element XML attribute, and XML attribute value
* Which of the following is the smallest absolute CSS unit?
* How do calculate the value of the line height for example?

p {  
 font-size: 10px;

line-height: 3.5em;   
 }

* All HTML documents must start with a what?
* Which HTML form method is by default?
* Which syntax is used for a HTML and CSS comment?
* Where will the caption element always be located in a table?
* Where would you go to validate HTML code?
* What are the three components of the debugging process?

**Make sure to do these Quizzes as well**

* [D276 HTML Fundamentals](https://wguashkin.github.io/D276-Practice-Quiz--1--HTML-Fundamentals/)
* [D276 More HTML](https://wguashkin.github.io/D276-Practice-Quiz-2/)
* [D276 XML Fundamentals](https://wguashkin.github.io/D276-Practice-Quiz-3/)
* [D276 CSS Fundamentals](https://wguashkin.github.io/D276-Practice-Quiz-4/)
* [D276 Debugging](https://wguashkin.github.io/D276-Practice-Quiz-5/)
* [D276 CSS Specificity](https://wguashkin.github.io/D276-CSS-Specificity-Quiz/)
* [D276 CSS Combinators](https://wguashkin.github.io/D276-Combinators-Quiz/)
* [D276 CSS Animation](https://wguashkin.github.io/D276-CSS-Animation-Quiz/)
* [D276 Finding the Total Width and Height](https://wguashkin.github.io/D276-Finding-Total-Width-and-Height/)
* [D276 Media Rule](https://wguashkin.github.io/D276-CSS-Media-Rule/)
* [D276 Practice Questions](https://wguashkin.github.io/D276-Practice-Questions/)