**Responsive web design course Notes**

* The lower the number, the higher the importance, so h2 elements have less importance than h1 elements.
* HTML5 has some elements that identify different content areas. These elements make your HTML easier to read and help with Search Engine Optimization (SEO) and accessibility.
* The strong element is used to indicate that some text is of strong importance or urgent.
* The em element is used to indicate that emphasis should be put on the object text
* he div element is used mainly for design layout purposes unlike the other content elements you have used so far
* article elements commonly contain multiple elements that have related information.
* You can add a fallback value for the font-family by adding another font name separated by a comma. Fallbacks are used in instances where the initial is not found/available.
* #FF0000 and #FF0000CC . The second one defines the alpha value of the color..

**Registration Form**

* The vh unit stands for viewport height, and is relative to 1% of the height of the viewport.
* The method attribute specifies how to send form-data to the URL specified in the action attribute. The form-data can be sent via a GET request as URL parameters (with method="get") or via a POST request as data in the request body (with method="post").
* The rem unit stands for root em, and is relative to the font size of the html element.
* Minlength = 8 for min length
* With type="password" you can use the pattern attribute to define a regular expression that the password must match to be considered valid.

**[a-z0-5]{8,}**

* The above is a regular expression which matches eight or more lowercase letters or the digits 0 to 5
* To relate the radio inputs, give them the same name attribute with a value of account-type. Now, it is not possible to select both radio inputs at the same time.
* Select only the .inline elements, and give them width of unset. This will remove the earlier rule which set all the input elements to width: 100%. Unset is used to remove any previously applied rule. In this case the width was unsetted
* Color by default is used to designate text.

**Box Model**

* margin: 0 auto 20px; This will remove its top margin, horizontally center it, and set its bottom margin to 20 pixels.
* border-radius: 8px 10px; Set’s its top-left radius and bottom-right radius to 8px, and top-right radius and bottom-left radius to 10px

**Flex Box**

* Giving an element a display property of flex will make it a flex container. Any direct children of a flex container are called **flex items.**
* **Flexbox** has a main and cross axis. The main axis is defined by the flex-direction property, which has four possible values:
* row (default): horizontal axis with flex items from left to right
* row-reverse: horizontal axis with flex items from right to left
* column: vertical axis with flex items from top to bottom
* column-reverse: vertical axis with flex items from bottom to top

**Note**: The axes and directions will be different depending on the text direction. The values shown are for a left-to-right text direction.

* The**flex-wrap** property determines how your flex items behave when the flex container is too small. Setting it to wrap will allow the items to wrap to the next row or column. nowrap (default) will prevent your items from wrapping and shrink them if needed.
* The **justify-content** property determines how the items inside a flex container are positioned along the main axis, affecting their position and the space around them. Consider the snippet below

*Flex-direction : row;  
justify-content : center;*

It will imply that elements being aligned horizontally ( row) and being centered **horizontally** (row)

* The **align-items** property positions the flex content along the cross axis. In this case, with your flex-direction set to row, your cross axis would be vertical
* The **gap**CSS shorthand property sets the gaps, also known as gutters, between rows and columns. The gap property and its row-gap and column-gap sub-properties provide this functionality for flex, grid, and multi-column layout. You apply the property to the container element.
* The**::after**pseudo-element creates an element that is the last child of the selected element. You can use it to add an empty element after the last image. If you give it the same width as the images it will push the last image to the left when the gallery is in a two-column layout.

- Rather than setting each aspect ratio individually, you can use the object-fit property to determine how images should behave.

Give your .gallery img selector the object-fit property and set it to cover. This will tell the image to fill the img container while maintaining aspect ratio, resulting in cropping to fit.

- **Span**  is highly used to group elements (especially text ) for alignment. It’s like grouping in Adobe

**Typography**

* The **rem** unit stands for root em, and is relative to the font size of the html element. In other words setting an element’s font size to 1em is setting it to the font-size assigned in the html selector, or if not prescised, the font-size of the browser. Like the word indicates “**Root** element(rem)”
* The **:not** pseudo-selector can be used to select all elements that do not match the given CSS rule.

div:not(#example) {

color: red;

}

ie all the divs except that with the ‘example’ id

.daily-value p:not(.no-divider) {

   border-bottom: 1px solid #888989;

}

* **50vw** means 50% of the view width
* Even though you added a placeholder to the first input element in the previous lesson, this is actually not a best-practice for accessibility; too often, users confuse the placeholder text with an actual input value - they think there is already a value in the input.
* Remove the placeholder text from the first input element, relying on the label being the best-practice.