## **CSS Basics**

* ****What is CSS?****: Cascading Style Sheets (CSS) is a markup language used to apply styles to HTML elements. CSS is used for colors, background images, layouts and more.
* ****Basic Anatomy of a CSS Rule****: A CSS rule is made up of two main parts: a selector and a declaration block. A selector is a pattern used in CSS to identify and target specific HTML elements for styling. A declaration block applies a set of styles for a given selector or selectors.

Here is the general syntax of a CSS rule:

selector {

property: value;}

* **meta name="viewport"**Element****: This meta element gives the browser instructions on how to control the page's dimensions and scaling on different devices, particularly on mobile phones and tablets.
* ****Default Browser Styles****: Each HTML element will have default browser styles applied to them. This usually includes items like default margins and paddings.

## **Inline, Internal, and External CSS**

* ****Inline CSS****: These styles are written directly within an HTML element using the style attribute. Most of the time you will not be using inline CSS due to separation of concerns.

Here is an example of inline CSS:

<p style="color: red;">This is a red paragraph.</p>

* ****Internal CSS****: These styles are written within the <style> tags inside the head section of an HTML document. This can be useful for creating short code examples, but usually you will not need be using internal CSS.
* ****External CSS****: These styles are written in a separate CSS file and linked to the HTML document using the link element in the head section. For most projects, you will use an external CSS file over internal or inline CSS.

## **Working With the width and height Properties**

* **width**Property****: This property specifies the width of an element. If you do not specify a width, then the default is set to auto. This means the element will take up the full width of its parent container.
* **min-width**Property****: This property specifies the minimum width for an element.
* **max-width**Property****: This property specifies the maximum width for an element.
* **height**Property****: This property specifies the height of an element. Similarly, the height is auto by default, which means it will adjust to the content inside.
* **min-height**Property****: This property specifies the minimum height for an element.
* **max-height**Property****: This property specifies the maximum height for an element.

## **Different Types of CSS Combinators**

* ****Descendant Combinator****: This combinator is used to target elements that are descendants of a specified parent element. The following example will target all li items inside ul elements.

<ul>

<li>Example item one</li>

<li>Example item two</li>

<li>Example item three</li></ul>

ul li {

background-color: yellow;}

* ****Child Combinator (**>**)****: This combinator is used to select elements that are direct children of a specified parent element. The following example will target all p elements that are direct children of the container class.

<div class="container">

<p>This will get styled.</p>

<div>

<p>This will not get styled.</p>

</div></div>

.container > p {

background-color: black;

color: white;}

* ****Next-sibling Combinator (**+**)****: This combinator selects an element that immediately follows a specified sibling element. The following example will select the paragraph element that immediately follows the h2 element.

<h2>I am a sub heading</h2>

<p>This paragraph element will get a red background.</p>

h2 + p {

background-color: red;}

* ****Subsequent-sibling Combinator (**~**)****: This combinator selects all siblings of a specified element that come after it. The following example will style only the second paragraph element because it is the only one that is a sibling of the ul element and shares the same parent.

<div class="container">

<p>This will not get styled.</p>

<ul>

<li>Example item one</li>

<li>Example item two</li>

<li>Example item three</li>

</ul>

<p>This will get styled.</p></div><p>This will not get styled.</p>

ul ~ p {

background-color: green;}

## **Inline, Block, and Inline-Block Level Elements**

* ****Inline Level Elements****: Inline elements only take up as much width as they need and do not start on a new line. These elements flow within the content, allowing text and other inline elements to appear alongside them. Common inline elements are span, anchor, and img elements.
* ****Block Level Elements****: Block-level elements start on a new line and take up the full width available to them by default, stretching across the width of their container. Some common block-level elements are div, paragraph, and section elements.
* ****Inline-Block Level Elements****: You can set an element to inline-block by using the display property. These elements behave like inline elements but can have a width and height set like block-level elements.

## **Margin and Padding**

* **margin**Property****: This property is used to apply space outside the element, between the element's border and the surrounding elements.
* **padding**Property****: This property is used to apply space inside the element, between the content and its border.
* **margin**Shorthand****: You can specify 1–4 values to set the margin sides. One value applies to all four sides; two values set top and bottom, then right and left; three values set top, horizontal (right and left), then bottom; four values set top, right, bottom, left.
* **padding**Shorthand****: You can specify 1–4 values to set the padding sides. One value applies to all four sides; two values set top and bottom, then right and left; three values set top, horizontal (right and left), then bottom; four values set top, right, bottom, left.

## **CSS Specificity**

* ****Inline CSS Specificity****: Inline CSS has the highest specificity because it is applied directly to the element. It overrides any internal or external CSS. The specificity value for inline styles is (1, 0, 0, 0).
* ****Internal CSS Specificity****: Internal CSS is defined within a style element in the head section of the HTML document. It has lower specificity than inline styles but can override external styles.
* ****External CSS Specificity****: External CSS is linked via a link element in the head section and is written in separate .css files. It has the lowest specificity but provides the best maintainability for larger projects.
* ****Universal Selector (**\***)****: a special type of CSS selector that matches any element in the document. It is often used to apply a style to all elements on the page, which can be useful for resetting or normalizing styles across different browsers. The universal selector has the lowest specificity value of any selector. It contributes 0 to all parts of the specificity value (0, 0, 0, 0).
* ****Type Selectors****: These selectors target elements based on their tag name. Type selectors have a relatively low specificity compared to other selectors. The specificity value for a type selector is (0, 0, 0, 1).
* ****Class Selectors****: These selectors are defined by a period (.) followed by the class name. The specificity value for a class selector is (0, 0, 1, 0). This means that class selectors can override type selectors, but they can be overridden by ID selectors and inline styles.
* ****ID Selectors****: ID selectors are defined by a hash symbol (#) followed by the ID name. ID selectors have a very high specificity, higher than type selectors and class selectors, but lower than inline styles. The specificity value for an ID selector is (0, 1, 0, 0).
* **!important**keyword****: used to give a style rule the highest priority, allowing it to override any other declarations for a property. When used, it forces the browser to apply the specified style, regardless of the specificity of other selectors. You should be cautious when using !important because it can make your CSS harder to maintain and debug.
* ****Cascade Algorithm****: An algorithm used to decide which CSS rules to apply when there are multiple styles targeting the same element. It ensures that the most appropriate styles are used, based on a set of well-defined rules.
* ****CSS Inheritance****: The process by which styles are passed down from parent elements to their children. Inheritance allows you to define styles at a higher level in the document tree and have them apply to multiple elements without explicitly specifying them for each element.

## **Styling Lists**

* **line-height**Property****: This property is used to create space between lines of text. The accepted line-height values include the keyword normal, numbers, percentages and length units like the em unit.
* **list-style-type**Property****: This property is used to specify the marker for a list item. Acceptable values can include a circle, disc, or decimal.
* **list-style-position**Property****: This property is used to set the position for the list marker. The only two acceptable values are inside and outside.
* **list-style-image**Property****: This property is used to use an image for the list item marker. A common use case is to use the url function with a value set to a valid image location.

## **Spacing list items using margin**

* Apart from line-height, margins can also be used in CSS to enhance the spacing and readability of list items.
* Margins create space outside each li element, allowing control over the gap between list items.
* margin-bottom is used to create space below each list item. For example, margin-bottom: 10px; will create a 10-pixel gap below each list item.

## **Styling Links**

* **pseudo-class**: This is a keyword added to a selector that allows you to select elements based on a particular state. Common states would include the :hover, :visited and :focus states.
* **:link pseudo-class**: This pseudo-class is used to style links that have not be visited by the user.
* **:visited pseudo-class**: This pseudo-class is used to style links where a user has already visited.
* **:hover pseudo-class**: This pseudo-class is used to style an elements where a user is actively hovering over them.
* **:focus pseudo-class**: This pseudo-class is used to style an element when it receives focus. Examples would include input or select elements where the clicks or tabs on the element to focus it.
* **:active pseudo-class**: This pseudo-class is used to style an element that was activated by the user. A common example would be when the user clicks on a button.

## **Working with Backgrounds and Borders**

* **background-size**Property****: This property is used to set the background size for an element. Some common values include cover for the background image to cover the entire element and contain for the background image to fit within the element.
* **background-repeat**Property****: This property is used to determine how background images should be repeated along the horizontal and vertical axes. The default value for background-repeat is repeat meaning the image will repeat both horizontally and vertically. You can also specify that there should be no repeat by using the no-repeat property.
* **background-position**Property****: This property is used to specify the position of the background image. It can be set to a specific length, percentage, or keyword values like top, bottom, left, right, and center.
* **background-attachment**Property****: This property is used to specify whether the background image should scroll with the content or remain fixed in place. The main values are scroll (default), where the background image scrolls with the content, and fixed, where the background image stays in the same position on the screen.
* **background-image**Property****: This property is used to set the background image of an element. You can set multiple background images at the same time and use either the url, radial-gradient or linear-gradient functions as values.
* **background**Property****: This is the shorthand property for setting all background properties in one declaration. Here is an example of setting the background image and setting it to not repeat: background: no-repeat url("example-url-goes-here");
* ****Good Contrast for Background and Foreground Colors****: It is important to ensure that the background and foreground colors have good contrast to make the text readable. The Web Content Accessibility Guidelines (WCAG) recommend a minimum contrast ratio of 4.5:1 for normal text and 3:1 for large text.

## **Borders**

* **border-top**Property****: This property is used to set the styles for the top border of an element. border-top: 3px solid blue; sets a 3-pixel-wide solid blue border on the top side of the element.
* **border-right**Property****: This property is used to set the styles for the right border of an element. border-right: 2px solid red; sets a 2-pixel-wide solid red border on the right side of the element.
* **border-bottom**Property****: This property is used to set the styles for the bottom border of an element. border-bottom: 1px dashed green; sets a 1-pixel-wide dashed green border on the bottom side of the element.
* **border-left**Property****: This property is used to set the styles for the left border of an element. border-left: 4px dotted orange; sets a 4-pixel-wide dotted orange border on the left side of the element.
* **border**Property****: This is the shorthand property for setting the width, style, and color of an element's border. border: 1px solid black; sets a 1-pixel-wide solid black border.
* **border-radius**Property****: This property is used to create rounded corners for an element's border.
* **border-style**Property****: This property is used to set the style of an element's border. Some accepted values include solid, dashed, dotted, and double.

## **Gradients**

* **linear-gradient()**Function****: This CSS function is used to create a transition between multiple colors along a straight line.
* **radial-gradient()**Function****: This CSS function creates an image that radiates from a particular point, like a circle or an ellipse, and gradually transitions between multiple colors.

## **User Action Pseudo-classes**

* ****Pseudo-classes Definition****: These are special CSS keywords that allow you to select an element based on its specific state or position.
* ****User Action Pseudo-classes****: These are special keywords that allow you to change the appearance of elements based on user interactions, improving the overall user experience.
* **:active**Pseudo-class****: This pseudo-class lets you select the active state of an element, like clicking on a button.
* **:hover**Pseudo-class****: This pseudo-class defines the hover state of an element.
* **:focus**Pseudo-class****: This pseudo-class applies styles when an element gains focus, typically through keyboard navigation or when a user clicks into a form input.
* **:focus-within**Pseudo-class****: This pseudo-class is used to apply styles to an element when it or any of its descendants have focus.

## **Input Pseudo-classes**

* ****Input Pseudo-classes****: These pseudo-classes are used to target HTML input elements based on the state they are in before and after user interaction.
* **:enabled**Pseudo-class****: This pseudo-class is used to target form buttons or other elements that are currently enabled.
* **:disabled**Pseudo-class****: This pseudo-class lets you style an interactive element in disabled mode.
* **:checked**Pseudo-class****: This pseudo-class is used to indicate to the user that it is checked.
* **:valid**Pseudo-class****: This pseudo-class targets the input fields that meet the validation criteria.
* **:invalid**Pseudo-class****: This pseudo-class targets the input fields that do not meet the validation criteria.
* **:in-range**and**:out-of-range**Pseudo-classes****: These pseudo-classes apply styles to elements based on whether their values are within or outside specified range constraints.
* **:required**Pseudo-class****: This pseudo-class targets input elements that have the required attribute. It signals to the user that they must fill out the field to submit the form.
* **:optional**Pseudo-class****: This pseudo-class applies styles input elements that are not required and can be left empty.
* **:autofill**Pseudo-class****: This pseudo-class applies styles to input fields that the browser automatically fills with saved data.

## **Location Pseudo-classes**

* ****Location Pseudo-classes****: These pseudo-classes are used for styling links and elements that are targeted within the current document.
* **:any-link**Pseudo-class****: This pseudo-class is a combination of the :link and :visited pseudo-classes. So, it matches any anchor element with an href attribute, regardless of whether it's visited or not.
* **:link**Pseudo-class****: This pseudo-class allows you to target all unvisited links on a webpage. You can use it to style links differently before the user clicks on them.
* **:local-link**Pseudo-class****: This pseudo-class targets links that point to the same document. It can be useful when you want to differentiate internal links from external ones.
* **:visited**Pseudo-class****: This pseudo-class targets a link the user has visited.
* **:target**Pseudo-class****: This pseudo-class is used to apply styles to an element that is the target of a URL fragment.
* **:target-within**Pseudo-class****: This pseudo-class applies styles to an element when it or one of its descendants is the target of a URL fragment.

## **Tree-structural Pseudo-classes**

* ****Tree-structural Pseudo-classes****: These pseudo-classes allow you to target and style elements based on their position within the document tree.
* **:root**Pseudo-class****: This pseudo-class is usually the root html element. It helps you target the highest level in the document so you can apply a common style to the entire document.
* **:empty**Pseudo-class****: Empty elements, that is, elements with no children other than white space, are also included in the document tree. That's why there's an :empty pseudo-class to target empty elements.
* **:nth-child(n)**Pseudo-class****: This pseudo-class allows you to select elements based on their position within a parent.
* **:nth-last-child(n)**Pseudo-class****: This pseudo-class enables you to select elements by counting from the end.
* **:first-child**Pseudo-class****: This pseudo-class selects the first element in a parent element or the document.
* **:last-child**Pseudo-class****: This pseudo-class selects the last element in a parent element or the document.
* **:only-child**Pseudo-class****: This pseudo-class selects the only element in a parent element or the document.
* **:first-of-type**Pseudo-class****: This pseudo-class selects the first occurrence of a specific element type within its parent.
* **:last-of-type**Pseudo-class****: This pseudo-class selects the last occurrence of a specific element type within its parent.
* **:nth-of-type(n)**Pseudo-class****: This pseudo-class allows you to select a specific element within its parent based on its position among siblings of the same type.
* **:only-of-type**Pseudo-class****: This pseudo-class selects an element if it's the only one of its type within its parent.

## **Functional Pseudo-classes**

* ****Functional Pseudo-classes****: Functional pseudo-classes allow you to select elements based on more complex conditions or relationships. Unlike regular pseudo-classes which target elements based on a state (for example, :hover, :focus), functional pseudo-classes accept arguments.
* **:is()**Pseudo-class****: This pseudo-class takes a list of selectors (ex. ol, ul) and selects an element that matches one of the selectors in the list.

<p class="example">This text will change color.</p><p>This text will not change color.</p><p>This text will not change color.</p><p class="this-works-too">This text will change color.</p>

p:is(.example, .this-works-too) {

color: red;}

* **:where()**Pseudo-class****: This pseudo-class takes a list of selectors (ex. ol, ul) and selects an element that matches one of the selectors in the list. The difference between :is and :where is that the latter will have a specificity of 0.

:where(h1, h2, h3) {

margin: 0;

padding: 0;}

* **:has()**Pseudo-class****: This pseudo-class is often dubbed the "parent" selector because it allows you to style elements that contain child elements specified in the selector list.

article:has(h2) {

border: 2px solid hotpink;}

* **:not()**Pseudo-class****: This pseudo-class is used to select elements that do not match the provided selector.

p:not(.example) {

color: blue;}

## **Pseudo-elements**

* **::before**Pseudo-element****: This pseudo-element uses the content property to insert cosmetic content like icons just before the element.
* **::after**Pseudo-element****: This pseudo-element uses the content property to insert cosmetic content like icons just after the element.
* **::first-letter**Pseudo-element****: This pseudo-element targets the first letter of an element's content, allowing you to style it.
* **::marker**Pseudo-element****: This pseudo-element lets you select the marker (bullet or numbering) of list items for styling.

## **Color Theory**

* ****Color Theory Definition****: This is the study of how colors interact with each other and how they affect our perception. It covers color relationships, color harmony, and the psychological impact of color.
* ****Primary Colors****: These colors which are yellow, blue, and red, are the fundamental hues from which all other colors are derived.
* ****Secondary Colors****: These colors result from mixing equal amounts of two primary colors. Green, orange, and purple are examples of secondary colors.
* ****Tertiary Colors****: These colors result from combining a primary color with a neighboring secondary color. Yellow-Green, Blue-Green, and Blue-Violet are examples of tertiary colors.
* ****Warm Colors****: These colors which include reds, oranges, and yellows, evoke feelings of comfort, warmth, and coziness.
* ****Cool Colors****: These colors which include blues, green, and purples, evoke feelings of calmness, serenity, and professionalism.
* ****Color Wheel****: The color wheel is a circular diagram that shows how colors relate to each other. It's an essential tool for designers because it helps them to select color combinations.
* ****Analogous Color Schemes****: These color schemes create cohesive and soothing experiences. They have analogous colors, which are adjacent to each other in the color wheel.
* ****Complementary Color Schemes****: These color schemes create high contrast and visual impact. Their colors are located on the opposite ends of the color wheel, relative to each other.
* ****Triadic Color Scheme****: This color scheme has vibrant colors. They are made from colors that are approximately equidistant from each other. If they are connected, they form an equilateral triangle on the color wheel.
* ****Monochromatic Color Scheme****: For this color scheme, all the colors are derived from the same base color by adjusting its lightness, darkness, and saturation. This evokes a feeling of unity and harmony while still creating contrast.

## **Different Ways to Work with Colors in CSS**

* ****Named Colors****: These colors are predefined color names recognized by browsers. Examples include blue, darkred, lightgreen.
* **rgb()**Function****: RGB stands for Red, Green, and Blue — the primary colors of light. These three colors are combined in different intensities to create a wide range of colors. the rgb() function allows you to define colors using the RGB color model.

p {

color: rgb(255, 0, 0);}

* **rgba()**Function****: This function adds a fourth value, alpha, that controls the transparency of the color. If not provided, the alpha value defaults to 1.

div {

background-color: rgba(0, 0, 255, 0.5);}

* **hsl()**Function****: HSL stands for Hue, Saturation, and Lightness — three key components that define a color.

p {

color: hsl(120, 100%, 50%);}

* **hsla()**Function****: This function adds a fourth value, alpha, that controls the opacity of the color.

div {

background-color: hsla(0, 100%, 50%, 0.5);}

* ****Hexadecimal****: A hex code (short for hexadecimal code) is a six-character string used to represent colors in the RGB color model. The "hex" refers to the base-16 numbering system, which uses digits 0 to 9 and letters A to F.

h1 {

color: #FF5733; /\* A reddish-orange color \*/}

p {

background-color: #4CAF50; /\* A shade of green \*/}

## **Linear and Radial Gradients**

* ****Linear Gradients****: These gradients create a gradual blend between colors along a straight line. You can control the direction of this line using keywords like to top, to right, to bottom right, or angles like 45deg. You can use any valid CSS color and as many color stops as you would like.

.linear-gradient {

background: linear-gradient(45deg, red, #33FF11, rgba(100, 100, 255, 0.5));

height: 40vh;}

* ****Radial Gradients****: These gradients create circular or elliptical gradients that radiate from a central point.

.radial-gradient {

background: radial-gradient(circle, red, blue);

height: 40vh;}

## **Best Practices for Styling Inputs**

* ****Styling Inputs****: As with all text elements, you need to ensure the styles you apply to text inputs are accessible. This means the font needs to be adequately sized, and the color needs to have sufficient contrast with the background. Input elements are also focusable. When you are editing your styles, you should take care that you preserve a noticeable indicator when the element has focus, such as a bold border.

## **Using appearance: none for Inputs**

* **appearance: none**: Browsers apply default styling to a lot of elements. The appearance: none CSS property gives you complete control over the styling, but comes with some caveats. When building custom styles for input elements, you will need to make sure focus and error indicators are still present.

## **Commons Issues Styling datetime-local and color Properties**

* ****Common Issues****: These special types of inputs rely on complex pseudo-elements to create things like date and color pickers. This presents a significant challenge for styling these inputs. One challenge is that the default styling is entirely browser-dependent, so the CSS you write to make the picker look the way you intend may be entirely different on another browser.

# **CSS Layouts and Effects Review**

## **CSS Overflow Property**

* ****Definition****: Overflow refers to the way elements handle content that exceeds, or "overflows", the size of the containing element. Overflow is two-dimensional.
* **overflow-x**: The x-axis determines horizontal overflow.
* **overflow-y**: the y-axis determines vertical overflow.
* **overflow**: Shorthand property for overflow-x and overflow-y. If given one value, both overflows will use it. If given two values, the overflow-x will use the first, and the overflow-y will use the second.

## **CSS Transform Property**

* ****Definition****: This property enables you to apply various transformations to elements, such as rotating, scaling, skewing, or translating (moving) them in 2D or 3D space.
* **translate()**Function****: This function is used to move an element from its current position.
* **scale()**Function****: This function allows you to change the size of an element.
* **rotate()**Function****: This function allows you to rotate an element.
* **skew()**Function****: This function allows you to skew an element.
* ****Transforms and Accessibility****: If you're using transform to hide or reveal content, make sure that the content is still accessible to screen readers and keyboard navigation. Hidden content should be truly hidden, such as by using display: none or visibility: hidden, rather than simply being visually moved off-screen.

## **The Box Model**

* ****Definition****: In the CSS box model, every element is surrounded by a box. This box consists of four components: the content area, padding, border, margin.
* ****Content Area****: The content area is the innermost part of the box. It's the space that contains the actual content of an element, like text or images.
* **padding**: The padding is the area immediately after the content area. It's the space between the content area and the border of an element.
* **border**: The border is the outer edge or outline of an element in the CSS box model. It's the visual boundary of the element.
* **margin**: The margin is the space outside the border of an element. It determines the distance between an element and other elements around it.

## **Margin Collapse**

* ****Definition****: This behavior occurs when the vertical margins of adjacent elements overlap, resulting in a single margin equal to the larger of the two. This behavior applies only to vertical margins (top and bottom), not horizontal margins (left and right).

## **The content-box and border-box Property Values**

* **box-sizing**Property****: This property is used to determine how the final width and height are calculated for an HTML element.
* **content-box**Value****: In the content-box model, the width and height that you set for an element determine the dimensions of the content area but they don't include the padding, border, or margin.
* **border-box**Value****: With border-box, the width and height of an element include the content area, the padding, and the border, but they don't include the margin.

## **CSS Reset**

* ****Definition****: A CSS reset is a stylesheet that removes all or some of the default formatting that web browsers apply to HTML elements. Third party options for CSS resets include sanitize.css and normalize.css.

## **CSS Filter Property**

* ****Definition****: This property can be used to create various effects such as blurring, color shifting, and contrast adjustment.
* **blur()**Function****: This function applies a Gaussian blur to the element. The amount is defined in pixels and represents the radius of the blur.
* **brightness()**Function****: This function adjusts the brightness of the element. A value of 0% will make the element completely black, while values over 100% will increase the brightness.
* **contrast()**Function****: This function adjusts the contrast of the element. A value of 0% will make the element completely grey, 100% will make the element appear normally, and values greater than 100% will increase the contrast.
* **grayscale()**Function****: This function converts the element to grayscale. The amount is defined as a percentage, where 100% is completely grayscale and 0% leaves the image unchanged.
* **sepia()**Function****: This function applies a sepia tone to the element. Like grayscale, it uses a percentage value.
* **hue-rotate()**Function****: This function applies a hue rotation to the element. The value is defined in degrees and represents a rotation around the color circle.

# **CSS Flexbox Review**

## **Introduction to CSS Flexbox and Flex Model**

* ****Definition****: CSS flexbox is a one-dimensional layout model that allows you to arrange elements in rows and columns within a container.
* ****Flex Model****: This model defines how flex items are arranged within a flex container. Every flex container has two axes: the main axis and the cross axis.

## **The flex-direction Property**

* ****Definition****: This property sets the direction of the main axis. The default value of flex-direction is row, which places all the flex items on the same row, in the direction of your browser's default language (left to right or right to left).
* **flex-direction: row-reverse;**: This reverses the items in the row.
* **flex-direction: column;**: This will align the flex items vertically instead of horizontally.
* **flex-direction: column-reverse;**: This reverses the order of the flex items vertically.

## **The flex-wrap Property**

* ****Definition****: This property determines how flex items are wrapped within a flex container to fit the available space. flex-wrap can take three possible values: nowrap, wrap, and wrap-reverse.
* **flex-wrap: nowrap;**: This is the default value. Flex items won't be wrapped onto a new line, even if their width exceed the container's width.
* **flex-wrap: wrap;**: This property will wrap the items when they exceed the width of their container.
* **flex-wrap: wrap-reverse;**: This property will wrap flex items in reverse order.
* **flex-flow**Property****: This property is a shorthand property for flex-direction and flex-wrap.

Example Code

flex-flow: column wrap-reverse;

## **The justify-content Property**

* ****Definition****: This property aligns the child elements along the main axis of the flex container.
* **justify-content: flex-start;**: In this case, the flex items will be aligned to the start of the main axis. This could be horizontal or vertical.
* **justify-content: flex-end;**: In this case, the flex items are aligned to the end of the main axis, horizontally or vertically.
* **justify-content: center;**: This centers the flex items along the main axis.
* **justify-content: space-between;**: This will distribute the elements evenly along the main axis.
* **justify-content: space-around;**: This will distribute flex items evenly within the main axis, adding a space before the first item and after the last item.
* **justify-content: space-evenly;**: This will distribute the items evenly along the main axis.

## **The align-items Property**

* ****Definition****: This property is used to distribute items along the cross axis. Remember that the cross axis is perpendicular to the main axis.
* **align-items: center;**: This is used to center the items along the cross axis.
* **align-items: flex-start;**: This aligns the items to the start of the cross axis.
* **align-items: stretch;**: This is used to stretch the flex items along the cross axis.