CSS

Tags	

Margin padding

Here is an example of how to use margin and padding in CSS:

```
.element {
  margin: 10px; /* Adds a 10px margin around the element */
  padding: 20px; /* Adds a 20px padding inside the element */
}
```

You can also specify different values for the top, right, bottom, and left sides of an element's margin or padding. For example:

```
.element {
  margin: 10px 20px 30px 40px; /* Adds a 10px top margin, a 20px right margin, a 30px bottom margin, and a 40px left margin */
  padding: 5px 10px 15px 20px; /* Adds a 5px top padding, a 10px right padding, a 15px bottom padding, and a 20px left padding */
}
```

In the above example, the values are specified in the order: top, right, bottom, left. You can also specify the values using the keywords "top", "right", "bottom", and "left", like this:

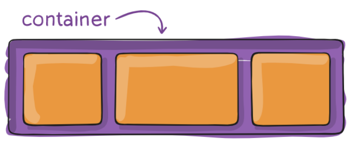
```
.element {
  margin: 10px 20px 30px 40px;
  padding: 5px 10px 15px 20px;
}
```

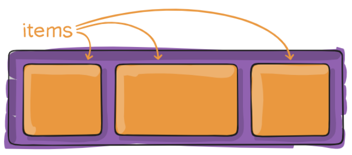
Margin and padding can be used to add space around elements and help to separate them from other elements on the page. They are an important part of the layout and design of a webpage.

Flexbox stuff

Best website for this: https://css-tricks.com/guides/

▾ Flexbox properties





Properties for the Parent (flex container)

display

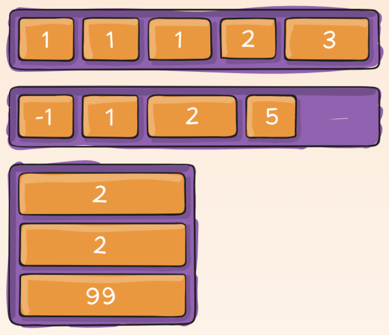
This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

```css
.container {
  display: flex; /* or inline-flex */
}
```

Note that CSS columns have no effect on a flex container.

Properties for the Children (flex items)

order



Flexbox is a layout module in CSS that makes it easier to design flexible and responsive layouts. It allows you to align elements in a flexible and dynamic way, and to control the layout of elements on the page.

Here are a few examples of how to use flexbox in your CSS:

- To create a flex container and make its children flex items, you can use the following CSS:

```
.container {
  display: flex; /* Creates a flex container */
}
```

```
<div class="container">
  <div>Flex item 1</div>
  <div>Flex item 2</div>
  <div>Flex item 3</div>
</div>
```

- To align the flex items horizontally, you can use the `justify-content` property:

```
.container {
  display: flex;
  justify-content: space-between; /* Aligns the flex items horizontally with equal space b
etween them */
}
```

- To align the flex items vertically, you can use the `align-items` property:

```
.container {
  display: flex;
  align-items: center; /* Aligns the flex items vertically at the center */
}
```

- To wrap the flex items onto multiple lines, you can use the `flex-wrap` property:

```
.container {
  display: flex;
  flex-wrap: wrap; /* Wraps the flex items onto multiple lines */
}
```

These are just a few examples of how you can use flexbox in your CSS. Flexbox provides a lot of other features and properties that you can use to control the layout of your page.

Children - Parent

To add CSS styles to the children of an element, you can use the child selector (">") in your CSS rules.

For example, let's say you have the following HTML structure:

```
<div class="parent">
  <p>Child element 1</p>
  <p>Child element 2</p>
</div>
```

To apply a style to the `p` elements that are children of the `div` element, you can use the following CSS:

```
.parent > p {
  /* CSS styles for the child elements go here */
}
```

This will apply the styles only to the `p` elements that are direct children of the `div` element with a class of "parent".

You can also use the child selector to apply styles to specific child elements. For example:

```
.parent > p:first-child {
  /* CSS styles for the first child element go here */
}

.parent > p:last-child {
  /* CSS styles for the last child element go here */
}
```

This will apply the styles only to the first and last `p` elements that are children of the `div` element.

Note that the child selector will only select direct children of an element, not all descendants. If you want to apply styles to all descendants of an element, you can use the descendant selector (a space) instead of the child selector.

For example:

```
.parent p {
  /* CSS styles for all p elements that are descendants of the parent element go here */
}
```

Media queries - conditionals

There are a few ways to add conditionals to your CSS, depending on the type of condition you want to create.

- Media queries: Media queries allow you to apply different styles based on the characteristics of the device or viewport. For example, you can use a media query to apply different styles for different screen sizes or orientations.

```
/* Apply these styles for screens that are wider than 480px */
@media (min-width: 480px) {
  /* CSS styles go here */
}

/* Apply these styles for screens that are narrower than 480px */
@media (max-width: 480px) {
  /* CSS styles go here */
}
```

- Feature queries: Feature queries allow you to apply different styles based on the support for a specific CSS feature. For example, you can use a feature query to apply different styles for browsers that do or do not support a particular property.

```
@supports (display: flex) {
  /* Apply these styles if the browser supports the flex layout */
}

@supports not (display: flex) {
  /* Apply these styles if the browser does not support the flex layout */
}
```

- The `:has` pseudo-class: The `:has` pseudo-class is a proposed feature of CSS that allows you to apply styles based on the presence of a specific element or attribute.

```
/* Apply these styles if the element has a child element with a class of "warning" */
.element:has(.warning) {
  /* CSS styles go here */
}
```

Note that the `:has` pseudo-class is not yet widely supported by browsers, so you may want to consider using other methods for adding conditionals to your CSS.

Responsive design

To add responsive design to your CSS, you can use media queries. Media queries allow you to apply different styles based on the characteristics of the device or viewport, such as the width or height of the screen.

Here is an example of how to use media queries to create a responsive design:

```
/* Apply these styles for screens that are wider than 480px */
@media (min-width: 480px) {
  /* CSS styles for larger screens go here */
}

/* Apply these styles for screens that are narrower than 480px */
@media (max-width: 480px) {
  /* CSS styles for smaller screens go here */
}
```

In the above example, the styles within the first media query will be applied for screens that are wider than 480px, while the styles within the second media query will be applied for screens that are narrower than 480px.

You can use media queries to create responsive designs for different screen sizes and orientations. For example, you might have separate styles for screens that are in landscape orientation, or for screens that are used with high-density displays.

In addition to media queries, there are other techniques you can use to create responsive designs, such as using flexible layout methods like flexbox and grid, or using responsive design frameworks like Bootstrap or Foundation.

CSS gradients

CSS gradients allow you to create smooth transitions between two or more colors. You can use gradients as backgrounds for elements, or as borders.

Here is an example of how to add a linear gradient as a background for an element:

```
.element {
  background: linear-gradient(to right, red, yellow); /* Creates a linear gradient from re
d to yellow */
}
```

You can also specify multiple color stops and angles for your gradient. For example:

```
.element {
  background: linear-gradient(45deg, red, yellow, green); /* Creates a linear gradient fro
m red to yellow to green at a 45 degree angle */
}
```

To create a radial gradient, you can use the `radial-gradient` function:

```
.element {
  background: radial-gradient(red, yellow); /* Creates a radial gradient from red to yello
w */
}
```

You can also specify the size and position of the gradient using keywords like `closest-side`, `farthest-side`, `closest-corner`, and `farthest-corner`, or by using lengths and percentages.

```
.element {
  background: radial-gradient(closest-side, red, yellow); /* Creates a radial gradient fro
m red to yellow, with the center of the gradient at the point closest to the side of the e
lement */
}
```

These are just a few examples of how you can use CSS gradients. There are many other options and techniques you can use to create more complex and custom gradients.

Resources

1. W3Schools: W3Schools is a website that offers a range of tutorials and references for web technologies, including CSS. It provides clear and concise explanations of

different CSS concepts and techniques, along with interactive examples that you can try out.

2. MDN Web Docs: MDN Web Docs is a comprehensive resource for web developers, maintained by the Mozilla Foundation. It includes detailed documentation and guides on CSS, as well as other web technologies.
3. CSS Tricks: CSS Tricks is a blog and resource site for web developers and designers, with a focus on CSS. It includes a range of articles, tutorials, and tips on different CSS techniques and approaches.
4. Codecademy: Codecademy is an online platform that offers interactive courses and exercises for learning a range of programming languages and technologies, including CSS. It provides step-by-step instructions and challenges to help you learn and practice CSS concepts.
5. The Complete CSS Course 2021: From Zero to Hero: This is an online course on Udemy that covers a range of CSS concepts and techniques, from the basics to advanced topics. The course includes video lectures, exercises, and projects to help you learn CSS and build practical skills.