CSS Introduction

## **What is CSS?**

* **CSS** stands for **C**ascading **S**tyle **S**heets
* CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
* CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
* External stylesheets are stored in **CSS files**

## **Why Use CSS?**

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

## **CSS Solved a Big Problem**

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

# CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



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.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p {

color: red;

text-align: center;

}

</style>

</head>

<body>

<p>Hello World!</p>

<p>These paragraphs are styled with CSS.</p>

</body>

</html>

#### **Example Explained**

* p is a **selector** in CSS (it points to the HTML element you want to style: <p>).
* color is a property, and red is the property value
* text-align is a property, and center is the property value

# How To Add CSS

## **Three Ways to Insert CSS**

There are three ways of inserting a style sheet:

* External CSS
* Internal CSS
* Inline CSS

## **External CSS**

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

### **Example**

External styles are defined within the <link> element, inside the <head> section of an HTML page

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks like:

### **"mystyle.css"**

body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: navy;  
  margin-left: 20px;  
}

## **Internal CSS**

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

### **Example**

Internal styles are defined within the <style> element, inside the <head> section of an HTML page

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: linen;

}

h1 {

color: maroon;

margin-left: 40px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

## **Inline CSS**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

### **Example**

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>

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

</body>

</html>

# CSS Selectors

## **CSS Selectors**

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)
* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)

This page will explain the most basic CSS selectors.

## **The CSS element Selector**

The element selector selects HTML elements based on the element name.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p>Every paragraph will be affected by the style.</p>

<p id="para1">Me too!</p>

<p>And me!</p>

</body>

</html>

## **The CSS id Selector**

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#para1 {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the style.</p>

</body>

</html>

## **The CSS class Selector**

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.center {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1 class="center">Red and center-aligned heading</h1>

<p class="center">Red and center-aligned paragraph.</p>

</body>

</html>

You can also specify that only specific HTML elements should be affected by a class.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.center {

text-align: center;

color: red;

}

</style>

</head>

<body>

<h1 class="center">This heading will not be affected</h1>

<p class="center">This paragraph will be red and center-aligned.</p>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.center {

text-align: center;

color: red;

}

p.large {

font-size: 300%;

}

</style>

</head>

<body>

<h1 class="center">This heading will not be affected</h1>

<p class="center">This paragraph will be red and center-aligned.</p>

<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>

</body>

</html>

# CSS Comments

## **CSS Comments**

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /\* and ends with \*/

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

/\* This is a single-line comment \*/

p {

color: red;

}

</style>

</head>

<body>

<p>Hello World!</p>

<p>This paragraph is styled with CSS.</p>

<p>CSS comments are not shown in the output.</p>

</body>

</html>

# CSS Colors

## **CSS Background Color**

You can set the background color for HTML elements:

### **Example**

<!DOCTYPE html>

<html>

<body>

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">

hello

</p>

</body>

</html>

# CSS Backgrounds

The CSS background properties are used to define the background effects for elements.

In these chapters, you will learn about the following CSS background properties:

* background-color
* background-image
* background-repeat
* background-attachment
* background-position

## **CSS background-color**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: lightblue;

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>This page has a light blue background color!</p>

</body>

</html>

## **CSS background-image**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-image: url(2.jpg);

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>This page has an image as the background!</p>

</body>

</html>

## **CSS background-repeat**

By default, the background-image property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-image: url("gradient\_bg.png");

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>Strange background image...</p>

</body>

</html>

## **CSS background-attachment**

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page)

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-image: url("img\_tree.png");

background-repeat: no-repeat;

background-position: right top;

margin-right: 200px;

background-attachment: fixed;

}

</style>

</head>

<body>

<h1>The background-attachment Property</h1>

</body>

</html>

## **CSS background - Shorthand property**

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

You can use the shorthand property background

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background: #ffffff url("img\_tree.png") no-repeat right top;

margin-right: 200px;

}

</style>

</head>

<body>

<h1>The background Property</h1>

<p>The background property is a shorthand property for specifying all the background properties in one declaration.</p>

</body>

</html>

# CSS Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

## **CSS Background Color**

### **Example**

<!DOCTYPE html>

<html>

<body>

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">

Welcome to pyspiders…

</p>

</body>

</html>

## **CSS Text Color**

### **Example**

<!DOCTYPE html>

<html>

<body>

<h3 style="color:Tomato;">Hello World</h3>

<p style="color:DodgerBlue;">hello pyspiders.</p>

<p style="color:MediumSeaGreen;">Hi </p>

</body>

</html>

## **CSS Border Color**

### **Example**

<!DOCTYPE html>

<html>

<body>

<h1 style="border: 2px solid Tomato;">Hello World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>

<h1 style="border: 2px solid Violet;">Hello World</h1>

</body>

</html>

# CSS Borders

## **CSS Border Properties**

The CSS border properties allow you to specify the style, width, and color of an element's border.

## **CSS Border Style**

The border-style property specifies what kind of border to display.

The following values are allowed:

* dotted - Defines a dotted border
* dashed - Defines a dashed border
* solid - Defines a solid border
* double - Defines a double border
* groove - Defines a 3D grooved border. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* none - Defines no border
* hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.dotted {border-style: dotted;}

# p.dashed {border-style: dashed;}

# p.solid {border-style: solid;}

# p.double {border-style: double;}

# p.groove {border-style: groove;}

# p.ridge {border-style: ridge;}

# p.inset {border-style: inset;}

# p.outset {border-style: outset;}

# p.none {border-style: none;}

# p.hidden {border-style: hidden;}

# p.mix {border-style: dotted dashed solid double;}

# </style>

# </head>

# <body>

# <h2>The border-style Property</h2>

# <p>This property specifies what kind of border to display:</p>

# <p class="dotted">A dotted border.</p>

# <p class="dashed">A dashed border.</p>

# <p class="solid">A solid border.</p>

# <p class="double">A double border.</p>

# <p class="groove">A groove border.</p>

# <p class="ridge">A ridge border.</p>

# <p class="inset">An inset border.</p>

# <p class="outset">An outset border.</p>

# <p class="none">No border.</p>

# <p class="hidden">A hidden border.</p>

# <p class="mix">A mixed border.</p>

# </body>

# </html>

## **CSS Border Width**

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.one {

# border-style: solid;

# border-width: 5px;

# }

# p.two {

# border-style: solid;

# border-width: medium;

# }

# p.three {

# border-style: dotted;

# border-width: 2px;

# }

# p.four {

# border-style: dotted;

# border-width: thick;

# }

# p.five {

# border-style: double;

# border-width: 15px;

# }

# p.six {

# border-style: double;

# border-width: thick;

# }

# </style>

# </head>

# <body>

# <h2>The border-width Property</h2>

# <p>This property specifies the width of the four borders:</p>

# <p class="one">Some text.</p>

# <p class="two">Some text.</p>

# <p class="three">Some text.</p>

# <p class="four">Some text.</p>

# <p class="five">Some text.</p>

# <p class="six">Some text.</p>

# <p><b>Note:</b> The "border-width" property does not work if it is used alone.

# Always specify the "border-style" property to set the borders first.</p>

# </body>

# </html>

## **Specific Side Widths**

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.one {

# border-style: solid;

# border-width: 5px 20px; /\* 5px top and bottom, 20px on the sides \*/

# }

# p.two {

# border-style: solid;

# border-width: 20px 5px; /\* 20px top and bottom, 5px on the sides \*/

# }

# p.three {

# border-style: solid;

# border-width: 25px 10px 4px 35px; /\* 25px top, 10px right, 4px bottom and 35px left

# }

# </style>

# </head>

# <body>

# <h2>The border-width Property</h2>

# <p>The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):</p>

# <p class="one">Some text.</p>

# <p class="two">Some text.</p>

# <p class="three">Some text.</p>

# </body>

# </html>

## **CSS Border Color**

The border-color property is used to set the color of the four borders.

The color can be set by:

* name - specify a color name, like "red"
* HEX - specify a HEX value, like "#ff0000"
* RGB - specify a RGB value, like "rgb(255,0,0)"
* HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
* transparent

**Note:** If border-color is not set, it inherits the color of the element.

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.one {

# border-style: solid;

# border-color: red;

# }

# p.two {

# border-style: solid;

# border-color: green;

# }

# p.three {

# border-style: dotted;

# border-color: blue;

# }

# </style>

# </head>

# <body>

# <h2>The border-color Property</h2>

# <p>This property specifies the color of the four borders:</p>

# <p class="one">A solid red border</p>

# <p class="two">A solid green border</p>

# <p class="three">A dotted blue border</p>

# <p><b>Note:</b> The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

# </body>

# </html>

## **Specific Side Colors**

The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.one {

# border-style: solid;

# border-color: red green blue yellow; /\* red top, green right, blue bottom and yellow left \*/

# }

# </style>

# </head>

# <body>

# <h2>The border-color Property</h2>

# <p class="one">A solid multicolor border</p>

# </body>

# </html>

# CSS Border Sides

## **CSS Border - Individual Sides**

From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p {

# border-top-style: dotted;

# border-right-style: solid;

# border-bottom-style: dotted;

# border-left-style: solid;

# }

# </style>

# </head>

# <body>

# <h2>Individual Border Sides</h2>

# <p>2 different border styles.</p>

# </body>

# </html>

## **CSS Border - Shorthand Property**

Like you saw in the previous page, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

* border-width
* border-style (required)
* border-color

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p {

# border: 5px solid red;

# }

# </style>

# </head>

# <body>

# <h2>The border Property</h2>

# <p>This property is a shorthand property for border-width, border-style, and border-color.</p>

# </body>

# </html>

## **CSS Rounded Borders**

The border-radius property is used to add rounded borders to an element:

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# p.normal {

# border: 2px solid red;

# }

# p.round1 {

# border: 2px solid red;

# border-radius: 5px;

# }

# p.round2 {

# border: 2px solid red;

# border-radius: 8px;

# }

# p.round3 {

# border: 2px solid red;

# border-radius: 12px;

# }

# </style>

# </head>

# <body>

# <h2>The border-radius Property</h2>

# <p>This property is used to add rounded borders to an element:</p>

# <p class="normal">Normal border</p>

# <p class="round1">Round border</p>

# <p class="round2">Rounder border</p>

# <p class="round3">Roundest border</p>

# </body>

# </html>

## **All CSS Border Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [border](https://www.w3schools.com/cssref/pr_border.asp) | Sets all the border properties in one declaration |
| [border-bottom](https://www.w3schools.com/cssref/pr_border-bottom.asp) | Sets all the bottom border properties in one declaration |
| [border-bottom-color](https://www.w3schools.com/cssref/pr_border-bottom_color.asp) | Sets the color of the bottom border |
| [border-bottom-style](https://www.w3schools.com/cssref/pr_border-bottom_style.asp) | Sets the style of the bottom border |
| [border-bottom-width](https://www.w3schools.com/cssref/pr_border-bottom_width.asp) | Sets the width of the bottom border |
| [border-color](https://www.w3schools.com/cssref/pr_border-color.asp) | Sets the color of the four borders |
| [border-left](https://www.w3schools.com/cssref/pr_border-left.asp) | Sets all the left border properties in one declaration |
| [border-left-color](https://www.w3schools.com/cssref/pr_border-left_color.asp) | Sets the color of the left border |
| [border-left-style](https://www.w3schools.com/cssref/pr_border-left_style.asp) | Sets the style of the left border |
| [border-left-width](https://www.w3schools.com/cssref/pr_border-left_width.asp) | Sets the width of the left border |
| [border-radius](https://www.w3schools.com/cssref/css3_pr_border-radius.asp) | Sets all the four border-\*-radius properties for rounded corners |
| [border-right](https://www.w3schools.com/cssref/pr_border-right.asp) | Sets all the right border properties in one declaration |
| [border-right-color](https://www.w3schools.com/cssref/pr_border-right_color.asp) | Sets the color of the right border |
| [border-right-style](https://www.w3schools.com/cssref/pr_border-right_style.asp) | Sets the style of the right border |
| [border-right-width](https://www.w3schools.com/cssref/pr_border-right_width.asp) | Sets the width of the right border |
| [border-style](https://www.w3schools.com/cssref/pr_border-style.asp) | Sets the style of the four borders |
| [border-top](https://www.w3schools.com/cssref/pr_border-top.asp) | Sets all the top border properties in one declaration |
| [border-top-color](https://www.w3schools.com/cssref/pr_border-top_color.asp) | Sets the color of the top border |
| [border-top-style](https://www.w3schools.com/cssref/pr_border-top_style.asp) | Sets the style of the top border |
| [border-top-width](https://www.w3schools.com/cssref/pr_border-top_width.asp) | Sets the width of the top border |
| [border-width](https://www.w3schools.com/cssref/pr_border-width.asp) | Sets the width of the four borders |

## **CSS Margins**

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

## **Margin - Individual Sides**

CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 1px solid black;

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

background-color: lightblue;

}

</style>

</head>

<body>

<h2>Using individual margin properties</h2>

<div>hello world.</div>

</body>

</html>

## **Margin - Shorthand Property**

To shorten the code, it is possible to specify all the margin properties in one property.

The margin property is a shorthand property for the following individual margin properties:

* margin-top
* margin-right
* margin-bottom
* margin-left

So, here is how it works:

If the margin property has four values:

* **margin: 25px 50px 75px 100px;**
  + top margin is 25px
  + right margin is 50px
  + bottom margin is 75px
  + left margin is 100px

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 1px solid black;

margin: 25px 50px 75px 100px;

background-color: lightblue;

}

</style>

</head>

<body>

<h2>The margin shorthand property - 4 values</h2>

<div>hello.</div>

<hr>

</body>

</html>

# CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

## **Padding - Individual Sides**

CSS has properties for specifying the padding for each side of an element:

* padding-top
* padding-right
* padding-bottom
* padding-left

All the padding properties can have the following values:

* length - specifies a padding in px, pt, cm, etc.
* % - specifies a padding in % of the width of the containing element
* inherit - specifies that the padding should be inherited from the parent element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 1px solid black;

background-color: lightblue;

padding-top: 50px;

padding-right: 30px;

padding-bottom: 50px;

padding-left: 80px;

}

</style>

</head>

<body>

<h2>Using individual padding properties</h2>

<div>Welcome </div>

</body>

</html>

## **Padding - Shorthand Property**

To shorten the code, it is possible to specify all the padding properties in one property.

The padding property is a shorthand property for the following individual padding properties:

* padding-top
* padding-right
* padding-bottom
* padding-left

So, here is how it works:

If the padding property has four values:

* **padding: 25px 50px 75px 100px;**
  + top padding is 25px
  + right padding is 50px
  + bottom padding is 75px
  + left padding is 100px

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

border: 1px solid black;

padding: 25px 50px 75px 100px;

background-color: lightblue;

}

</style>

</head>

<body>

<h2>The padding shorthand property - 4 values</h2>

<div>This div element has a top padding of 25px, a right padding of 50px, a bottom padding of 75px, and a left padding of 100px.</div>

</body>

</html>

## **CSS Setting height and width**

The height and width properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

## **CSS height and width Values**

The height and width properties may have the following values:

* auto - This is default. The browser calculates the height and width
* length - Defines the height/width in px, cm etc.
* % - Defines the height/width in percent of the containing block
* initial - Sets the height/width to its default value
* inherit - The height/width will be inherited from its parent value

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

height: 200px;

width: 50%;

background-color: powderblue;

}

</style>

</head>

<body>

<h2>Set the height and width of an element</h2>

<p>This div element has a height of 200px and a width of 50%:</p>

<div></div>

</body>

</html>

**Note:** Remember that the height and width properties do not include padding, borders, or margins! They set the height/width of the area inside the padding, border, and margin of the element!

## **The CSS Box Model**

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: lightgrey;

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

</style>

</head>

<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>

</body>

</html>

## **CSS Outline**

## **CSS Outline**

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

CSS has the following outline properties:

* outline-style
* outline-color
* outline-width
* outline-offset
* outline

## **CSS Outline Style**

The outline-style property specifies the style of the outline, and can have one of the following values:

* dotted - Defines a dotted outline
* dashed - Defines a dashed outline
* solid - Defines a solid outline
* double - Defines a double outline
* groove - Defines a 3D grooved outline
* ridge - Defines a 3D ridged outline
* inset - Defines a 3D inset outline
* outset - Defines a 3D outset outline
* none - Defines no outline
* hidden - Defines a hidden outline

The following example shows the different outline-style values

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p {outline-color:red;}

p.dotted {outline-style: dotted;}

p.dashed {outline-style: dashed;}

p.solid {outline-style: solid;}

p.double {outline-style: double;}

p.groove {outline-style: groove;}

p.ridge {outline-style: ridge;}

p.inset {outline-style: inset;}

p.outset {outline-style: outset;}

</style>

</head>

<body>

<h2>The outline-style Property</h2>

<p class="dotted">A dotted outline</p>

<p class="dashed">A dashed outline</p>

<p class="solid">A solid outline</p>

<p class="double">A double outline</p>

<p class="groove">A groove outline. The effect depends on the outline-color value.</p>

<p class="ridge">A ridge outline. The effect depends on the outline-color value.</p>

<p class="inset">An inset outline. The effect depends on the outline-color value.</p>

<p class="outset">An outset outline. The effect depends on the outline-color value.</p>

</body>

</html>

## **CSS Outline Width**

The outline-width property specifies the width of the outline, and can have one of the following values:

* thin (typically 1px)
* medium (typically 3px)
* thick (typically 5px)
* A specific size (in px, pt, cm, em, etc)

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.ex1 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: thin;

}

p.ex2 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: medium;

}

p.ex3 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: thick;

}

p.ex4 {

border: 1px solid black;

outline-style: solid;

outline-color: red;

outline-width: 4px;

}

</style>

</head>

<body>

<h2>The outline-width Property</h2>

<p class="ex1">A thin outline.</p>

<p class="ex2">A medium outline.</p>

<p class="ex3">A thick outline.</p>

<p class="ex4">A 4px thick outline.</p>

</body>

</html>

## **CSS Outline Color**

The outline-color property is used to set the color of the outline.

The color can be set by:

* name - specify a color name, like "red"
* HEX - specify a hex value, like "#ff0000"
* RGB - specify a RGB value, like "rgb(255,0,0)"
* HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
* invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.ex1 {

border: 2px solid black;

outline-style: solid;

outline-color: red;

}

p.ex2 {

border: 2px solid black;

outline-style: dotted;

outline-color: blue;

}

p.ex3 {

border: 2px solid black;

outline-style: outset;

outline-color: grey;

}

</style>

</head>

<body>

<h2>The outline-color Property</h2>

<p>The outline-color property is used to set the color of the outline.</p>

<p class="ex1">A solid red outline.</p>

<p class="ex2">A dotted blue outline.</p>

<p class="ex3">An outset grey outline.</p>

</body>

</html>

## **CSS Outline Offset**

The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p {

margin: 30px;

border: 1px solid black;

outline: 1px solid red;

outline-offset: 15px;

}

</style>

</head>

<body>

<h2>The outline-offset Property</h2>

<p>This paragraph has an outline 15px outside the border edge.</p>

</body>

</html>

# CSS Fonts

## **Font Family**

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.serif {

font-family: "Times New Roman", Times, serif;

}

p.sansserif {

font-family: Arial, Helvetica, sans-serif;

}

</style>

</head>

<body>

<h1>CSS font-family</h1>

<p class="serif">This is a paragraph, shown in the Times New Roman font.</p>

<p class="sansserif">This is a paragraph, shown in the Arial font.</p>

</body>

</html>

## **Font Style**

The font-style property is mostly used to specify italic text.

This property has three values:

* normal - The text is shown normally
* italic - The text is shown in italics
* oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.normal {

font-style: normal;

}

p.italic {

font-style: italic;

}

p.oblique {

font-style: oblique;

}

</style>

</head>

<body>

<p class="normal">Hello</p>

<p class="italic">Welcome</p>

<p class="oblique">Hi</p>

</body>

</html>

## **Font Size**

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

* Sets the text to a specified size
* Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
* Absolute size is useful when the physical size of the output is known

Relative size:

* Sets the size relative to surrounding elements
* Allows a user to change the text size in browsers

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

font-size: 40px;

}

h2 {

font-size: 30px;

}

</style>

</head>

<body>

<h1>Hi</h1>

<h2>Welcome to pyspiders</h2>

</body>

</html>

# CSS Links

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

* a:link - a normal, unvisited link
* a:visited - a link the user has visited
* a:hover - a link when the user mouses over it
* a:active - a link the moment it is clicked

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

/\* unvisited link \*/

a:link {

color: red;

}

</style>

</head>

<body>

<p><b><a href="default.asp" target="\_blank">This is a link</a></b></p>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

/\* visited link \*/

a:visited {

color: green;

}

</style>

</head>

<body>

<p><b><a href="default.asp" target="\_blank">This is a link</a></b></p>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

/\*hover link \*/

button:hover {

color: hotpink;

}

</style>

</head>

<body>

Username : <input type=”text” placeholder=”enter your name”></br>

<button>Log in</button>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

/\*active link \*/

button:active {

color: orange;

}

</style>

</head>

<body>

Username : <input type=”text” placeholder=”enter your name”></br>

<button>Log in</button>

</body>

</html>

## **The display Property**

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

## **Block-level Elements**

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

## **Inline Elements**

An inline element does not start on a new line and only takes up as much width as necessary.

Examples of inline elements:

* <span>
* <a>
* <img>

## **Display: none;**

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The <script> element uses display: none; as default.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

li {

display: inline;

}

</style>

</head>

<body>

<p>Display a list of links as a horizontal menu:</p>

<ul>

<li>HTML</li>

<li>CSS<li>

<li>JavaScript</li>

</ul>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

li {

display: block;

}

</style>

</head>

<body>

<p>Display a list of links as a horizontal menu:</p>

<ul>

<li>HTML</li>

<li>CSS<li>

<li>JavaScript</li>

</ul>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

li {

display: none;

}

</style>

</head>

<body>

<p>Display a list of links as a horizontal menu:</p>

<ul>

<li>HTML</li>

<li>CSS<li>

<li>JavaScript</li>

</ul>

</body>

</html>

## **The position Property**

The position property specifies the type of positioning method used for an element.

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

## **position: static;**

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div.static {

position: static;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: static;</h2>

<div class="static">

Hello

</div>

</body>

</html>

## **position: relative;**

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div.relative {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: relative;</h2>

<div class="relative">

Hello

</div>

</body>

</html>

## **position: fixed;**

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div.fixed {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: fixed;</h2>

<div class="fixed">

Hello world!

</div>

</body>

</html>

## **position: absolute;**

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

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;

}

</style>

</head>

<body>

<h2>position: absolute;</h2>

<div class="relative">Hello

<div class="absolute">world</div>

</div>

</body>

</html>

## **position: sticky;**

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div.sticky {

position: -webkit-sticky;

position: sticky;

top: 0;

padding: 5px;

background-color: #cae8ca;

border: 2px solid #4CAF50;

}

</style>

</head>

<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<p>Note: IE/Edge 15 and earlier versions do not support sticky position.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">

<p>Scroll back up to remove the stickyness.</p>

<p> hello

Welcome to pyspiders.

hello

Welcome to pyspiders.

hello

Welcome to pyspiders..</p>

<p>hello

Welcome to pyspiders.

hello

Welcome to pyspiders.

hello

Welcome to pyspiders.</p>

</div>

</body>

</html>

## **Overlapping Elements**

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<img src="2.jpg" width="100" height="140">

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

</body>

</html>

## **CSS Overflow**

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

## **overflow: visible**

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: #eee;

width: 200px;

height: 50px;

border: 1px dotted black;

overflow: visible;

}

</style>

</head>

<body>

<h2>CSS Overflow</h2>

<p>By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>

</html>

## **overflow: hidden**

With the hidden value, the overflow is clipped, and the rest of the content is hidden.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: #eee;

width: 200px;

height: 50px;

border: 1px dotted black;

overflow: hidden;

}

</style>

</head>

<body>

<h2>CSS Overflow</h2>

<p>With the hidden value, the overflow is clipped, and the rest of the content is hidden:</p>

<p>Try to remove the overflow property to understand how it works.</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>

</html>

## **overflow: scroll**

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: #eee;

width: 200px;

height: 100px;

border: 1px dotted black;

overflow: scroll;

}

</style>

</head>

<body>

<h2>CSS Overflow</h2>

<p>Setting the overflow value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>

</html>

## **overflow: auto**

The auto value is similar to scroll, but it adds scrollbars only when necessary

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: #eee;

width: 200px;

height: 50px;

border: 1px dotted black;

overflow: auto;

}

</style>

</head>

<body>

<h2>CSS Overflow</h2>

<p>The auto value is similar to scroll, only it add scrollbars when necessary:</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>

</html>

# CSS Layout - float and clear

The CSS float property specifies how an element should float.

The CSS clear property specifies what elements can float beside the cleared element and on which side.

## **The float Property**

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

In its simplest use, the float property can be used to wrap text around images.

## **Example - float: right;**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

float: right;

}

</style>

</head>

<body>

<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="7.jpg" alt="image" style="width:170px;height:170px;margin-left:15px;">

Hello,Welcome to pyspiders.</p>

</body>

</html>

## **Example - float: left;**

The following example specifies that an image should float to the **left** in a text

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

float: left;

}

</style>

</head>

<body>

<p>In this example, the image will float to the left in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="8.jpg" alt="image2" style="width:170px;height:170px;margin-right:15px;">

Hello world!.</p>

</body>

</html>

## **The display: inline-block Value**

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

span.a {

display: inline; /\* the default for span \*/

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.b {

display: inline-block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.c {

display: block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

</style>

</head>

<body>

<h1>The display Property</h1>

<h2>display: inline</h2>

<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

<h2>display: inline-block</h2>

<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

<h2>display: block</h2>

<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="c">Aliquam</span> <span class="c">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

</body>

</html>

## **Using inline-block to Create Navigation Links**

One common use for display: inline-block is to display list items horizontally instead of vertically. The following example creates horizontal navigation links

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.nav {

background-color: yellow;

list-style-type: none;

text-align: center;

margin: 0;

padding: 0;

}

.nav li {

display: inline-block;

font-size: 20px;

padding: 20px;

}

</style>

</head>

<body>

<h1>Horizontal Navigation Links</h1>

<ul class="nav">

<li><a href="#home">Home</a></li>

<li><a href="#about">About Us</a></li>

<li><a href="#clients">Our Clients</a></li>

<li><a href="#contact">Contact Us</a></li>

</ul>

</body>

</html>

# CSS Layout - Horizontal & Vertical Align

## **Center Align Elements**

To horizontally center a block element (like <div>), use margin: auto;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.center {

margin: auto;

width: 60%;

border: 3px solid #73AD21;

padding: 10px;

}

</style>

</head>

<body>

<h2>Center Align Elements</h2>

<p>To horizontally center a block element (like div), use margin: auto;</p>

<div class="center">

<p><b>Note: </b>Using margin:auto will not work in IE8, unless a !DOCTYPE is declared.</p>

</div>

</body>

</html>

## **Center an Image**

To center an image, set left and right margin to auto and make it into a block element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

display: block;

margin-left: auto;

margin-right: auto;

}

</style>

</head>

<body>

<h2>Center an Image</h2>

<p>To center an image, set left and right margin to auto, and make it into a block element.</p>

<img src="paris.jpg" alt="Paris" style="width:40%">

</body>

</html>

## **Left and Right Align - Using position**

One method for aligning elements is to use position: absolute;

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.right {

position: absolute;

right: 0px;

width: 300px;

border: 3px solid #73AD21;

padding: 10px;

}

</style>

</head>

<body>

<h2>Right Align</h2>

<p>An example of how to right align elements with the position property:</p>

<div class="right">

<p>In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.</p>

</div>

</body>

</html>

## **Left and Right Align - Using float**

Another method for aligning elements is to use the float property

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.right {

float: right;

width: 300px;

border: 3px solid #73AD21;

padding: 10px;

}

</style>

</head>

<body>

<h2>Right Align</h2>

<p>An example of how to right align elements with the float property:</p>

<div class="right">

<p>In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.</p>

</div>

</body>

</html>

# CSS Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

* descendant selector (space)
* child selector (>)
* adjacent sibling selector (+)
* general sibling selector (~)

## **Descendant Selector**

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

The following example selects all <p> elements inside <div> elements

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div p {

background-color: yellow;

}

</style>

</head>

<body>

<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>

## **Child Selector**

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

The following example selects all <p> elements that are children of a <div> element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div > p {

background-color: yellow;

}

</style>

</head>

<body>

<div>

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

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

<section><p>Paragraph 3 in the div.</p></section> <!-- not Child but Descendant -->

<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>

## **Adjacent Sibling Selector**

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all <p> elements that are placed immediately after <div> elements

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div + p {

background-color: yellow;

}

</style>

</head>

<body>

<div>

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

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

</div>

<p>Paragraph 3. Not in a div.</p>

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

</body>

</html>

## **General Sibling Selector**

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of <div> elements

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div ~ p {

background-color: yellow;

}

</style>

</head>

<body>

<p>Paragraph 1.</p>

<div>

<p>Paragraph 2.</p>

</div>

<p>Paragraph 3.</p>

<code>Some code.</code>

<p>Paragraph 4.</p>

</body>

</html>

# CSS Pseudo-elements

## **What are Pseudo-Elements?**

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

* Style the first letter, or line, of an element
* Insert content before, or after, the content of an element

## **The ::first-line Pseudo-element**

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p::first-line {

color: #ff0000;

font-variant: small-caps;

}

</style>

</head>

<body>

<p>You can use the ::first-line pseudo-element to add a special effect to the first line of a text. Some more text. And even more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more.</p>

</body>

</html>

## **The ::first-letter Pseudo-element**

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all <p> elements

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p::first-letter {

color: #ff0000;

font-size: xx-large;

}

</style>

</head>

<body>

<p>You can use the ::first-letter pseudo-element to add a special effect to the first character of a text!</p>

</body>

</html>

## **CSS - The ::before Pseudo-element**

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1::before {

content: url(smiley.gif);

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p>The ::before pseudo-element inserts content before the content of an element.</p>

<h1>This is a heading</h1>

<p><b>Note:</b> IE8 supports the content property only if a !DOCTYPE is specified.</p>

</body>

</html>

## **CSS - The ::after Pseudo-element**

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each <h1> element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1::after {

content: url(smiley.gif);

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p>The ::after pseudo-element inserts content after the content of an element.</p>

<h1>This is a heading</h1>

<p><b>Note:</b> IE8 supports the content property only if a !DOCTYPE is specified.</p>

</body>

</html>

# CSS Opacity / Transparency

The opacity property specifies the opacity/transparency of an element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

opacity: 0.5;

}

</style>

</head>

<body>

<h1>Image Transparency</h1>

<p>The opacity property specifies the transparency of an element. The lower the value, the more transparent:</p>

<p>Image with 50% opacity:</p>

<img src="4.jpg" alt="Forest" width="170" height="100">

</body>

</html>

## **Transparent Hover Effect**

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

img {

opacity: 0.5;

}

img:hover {

opacity: 1.0;

}

</style>

</head>

<body>

<h1>Image Transparency</h1>

<p>The opacity property is often used together with the :hover selector to change the opacity on mouse-over:</p>

<img src="2.jpg" alt="Forest" width="170" height="100">

<img src="4.jpg" alt="Mountains" width="170" height="100">

<img src="6.jpg" alt="Italy" width="170" height="100">

</body>

</html>

## **Basic Dropdown**

Create a dropdown box that appears when the user moves the mouse over an element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

padding: 12px 16px;

z-index: 1;

}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<h2>Hoverable Dropdown</h2>

<p>Move the mouse over the text below to open the dropdown content.</p>

<div class="dropdown">

<span>Mouse over me</span>

<div class="dropdown-content">

<p>Hello World!</p>

</div>

</div>

</body>

</html>

## **Dropdown Menu**

Create a dropdown menu that allows the user to choose an option from a list

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.dropbtn {

background-color: #4CAF50;

color: white;

padding: 16px;

font-size: 16px;

border: none;

cursor: pointer;

}

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

.dropdown-content a:hover {background-color: #f1f1f1}

.dropdown:hover .dropdown-content {

display: block;

}

.dropdown:hover .dropbtn {

background-color: #3e8e41;

}

</style>

</head>

<body>

<h2>Dropdown Menu</h2>

<p>Move the mouse over the button to open the dropdown menu.</p>

<div class="dropdown">

<button class="dropbtn">Dropdown</button>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</div>

</body>

</html>

### **Dropdown Navbar**

How to add a dropdown menu inside a navigation bar.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

background-color: #333;

}

li {

float: left;

}

li a, .dropbtn {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover, .dropdown:hover .dropbtn {

background-color: red;

}

li.dropdown {

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

background-color: #f9f9f9;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

z-index: 1;

}

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

text-align: left;

}

.dropdown-content a:hover {background-color: #f1f1f1;}

.dropdown:hover .dropdown-content {

display: block;

}

</style>

</head>

<body>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li class="dropdown">

<a href="javascript:void(0)" class="dropbtn">Dropdown</a>

<div class="dropdown-content">

<a href="#">Link 1</a>

<a href="#">Link 2</a>

<a href="#">Link 3</a>

</div>

</li>

</ul>

</body>

</html>

# CSS Text Effects

## **CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes**

In this chapter you will learn about the following properties:

* text-overflow
* word-wrap
* word-break
* writing-mode

## **CSS Text Overflow**

The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

p.test1 {

white-space: nowrap;

width: 200px;

border: 1px solid #000000;

overflow: hidden;

text-overflow: clip;

}

p.test2 {

white-space: nowrap;

width: 200px;

border: 1px solid #000000;

overflow: hidden;

text-overflow: ellipsis;

}

</style>

</head>

<body>

<h1>The text-overflow Property</h1>

<p>The following two paragraphs contains a long text that will not fit in the box.</p>

<h2>text-overflow: clip:</h2>

<p class="test1">This is some long text that will not fit in the box</p>

<h2>text-overflow: ellipsis:</h2>

<p class="test2">This is some long text that will not fit in the box</p>

</body>

</html>

# CSS Shadow Effects

## **CSS Shadow Effects**

With CSS you can add shadow to text and to elements.

In this chapter you will learn about the following properties:

* text-shadow
* box-shadow

## **CSS Text Shadow**

The CSS text-shadow property applies shadow to text.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px;

}

</style>

</head>

<body>

<h1>Text-shadow effect!</h1>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px red;

}

</style>

</head>

<body>

<h1>Text-shadow effect!</h1>

</body>

</html>

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px 5px green;

}

</style>

</head>

<body>

<h1>Text-shadow effect!</h1>

</body>

</html>

# CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

* **Linear Gradients (goes down/up/left/right/diagonally)**
* **Radial Gradients (defined by their center)**

## **CSS Linear Gradients**

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

### **Syntax**

background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: linear-gradient(red, yellow); /\* Standard syntax (must be last) \*/

}

</style>

</head>

<body>

<h1>Linear Gradient - Top to Bottom</h1>

<div id="grad1"></div>

</body>

</html>

## **Using Multiple Color Stops**

The following example shows a linear gradient (from top to bottom) with multiple color stops

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: linear-gradient(red, yellow, green); /\* Standard syntax (must be last) \*/

}

#grad2 {

height: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: linear-gradient(red, orange, yellow, green, blue, indigo, violet); /\* Standard syntax (must be last) \*/

}

#grad3 {

height: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: linear-gradient(red 10%, green 85%, blue 90%); /\* Standard syntax (must be last) \*/

}

</style>

</head>

<body>

<h1>Linear Gradients - Multiple Color Stops</h1>

<h3>3 Color Stops (evenly spaced):</h3>

<div id="grad1"></div>

<h3>7 Color Stops (evenly spaced):</h3>

<div id="grad2"></div>

<h3>3 Color Stops (not evenly spaced):</h3>

<div id="grad3"></div>

<p><strong>Note:</strong> Color stops are automatically spaced evenly when no percents are specified.</p>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support gradients.</p>

</body>

</html>

# CSS 2D Transforms

## **CSS 2D Transforms**

CSS transforms allow you to move, rotate, scale, and skew elements.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: rotate(20deg); /\* IE 9 \*/

transform: rotate(20deg); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The rotate() Method</h1>

<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>

<div>

This a normal div element.

</div>

<div id="myDiv">

This div element is rotated clockwise 20 degrees.

</div>

</body>

</html>

## **CSS 2D Transforms Methods**

With the CSS transform property you can use the following 2D transformation methods:

* translate()
* rotate()
* scaleX()
* scaleY()
* scale()
* skewX()
* skewY()
* skew()
* matrix()

## **The translate() Method**

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: translate(50px,100px); /\* IE 9 \*/

transform: translate(50px,100px); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The translate() Method</h1>

<p>The translate() method moves an element from its current position:</p>

<div>

This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

</div>

</body>

</html>

## **The rotate() Method**

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

width: 300px;

height: 100px;

background-color: yellow;

border: 1px solid black;

}

div#myDiv {

-ms-transform: rotate(20deg); /\* IE 9 \*/

transform: rotate(20deg); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The rotate() Method</h1>

<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>

<div>

This a normal div element.

</div>

<div id="myDiv">

This div element is rotated clockwise 20 degrees.

</div>

</body>

</html>

## **The scale() Method**

The scale() method increases or decreases the size of an element

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

transform: scale(2,3);

}

</style>

</head>

<body>

<h1>The scale() Method</h1>

<p>The scale() method increases or decreases the size of an element.</p>

<div>

This div element is two times of its original width, and three times of its original height.

</div>

</body>

</html>

## **The scaleX() Method**

The scaleX() method increases or decreases the width of an element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleX(2); /\* IE 9 \*/

transform: scaleX(2); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The scaleX() Method</h1>

<p>The scaleX() method increases or decreases the width of an element.</p>

<div>

This div element is two times of its original width.

</div>

</body>

</html>

## **The scaleY() Method**

The scaleY() method increases or decreases the height of an element.

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

div {

margin: 150px;

width: 200px;

height: 100px;

background-color: yellow;

border: 1px solid black;

-ms-transform: scaleY(3); /\* IE 9 \*/

transform: scaleY(3); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h1>The scaleY() Method</h1>

<p>The scaleY() method increases or decreases the height of an element.</p>

<div>

This div element is three times of its original height.

</div>

</body>

</html>

## **CSS 3D Transforms**

CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation

In this chapter you will learn about the following CSS property:

* transform

## **CSS 3D Transforms Methods**

With the CSS transform property you can use the following 3D transformation methods:

* rotateX()
* rotateY()
* rotateZ()

## **The rotateX() Method**

# The rotateX() method rotates an element around its X-axis at a given degree.

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 300px;

# height: 100px;

# background-color: yellow;

# border: 1px solid black;

# }

# #myDiv {

# transform: rotateX(150deg);

# }

# </style>

# </head>

# <body>

# <h1>The rotateX() Method</h1>

# <p>The rotateX() method rotates an element around its X-axis at a given degree.</p>

# <div>

# This a normal div element.

# </div>

# <div id="myDiv">

# This div element is rotated 150 degrees.

# </div>

# <p><b>Note:</b> Internet Explorer 9 (and earlier versions) does not support the rotateX() method.</p>

# </body>

# </html>

## **The rotateY() Method**

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 300px;

# height: 100px;

# background-color: yellow;

# border: 1px solid black;

# }

# #myDiv {

# transform: rotateY(150deg);

# }

# </style>

# </head>

# <body>

# <h1>The rotateY() Method</h1>

# <p>The rotateY() method rotates an element around its Y-axis at a given degree.</p>

# <div>

# This a normal div element.

# </div>

# <div id="myDiv">

# This div element is rotated 150 degrees.

# </div>

# <p><b>Note:</b> Internet Explorer 9 (and earlier versions) does not support the rotateY() method.</p>

# </body>

# </html>

## **The rotateZ() Method**

The rotateZ() method rotates an element around its Z-axis at a given degree

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 300px;

# height: 100px;

# background-color: yellow;

# border: 1px solid black;

# }

# #myDiv {

# transform: rotateZ(90deg);

# }

# </style>

# </head>

# <body>

# <h1>The rotateZ() Method</h1>

# <p>The rotateZ() method rotates an element around its Z-axis at a given degree.</p>

# <div id="myDiv">

# This div element is rotated 90 degrees.

# </body>

# </html>

## **CSS Transitions**

CSS transitions allows you to change property values smoothly, over a given duration.

In this chapter you will learn about the following properties:

* transition
* transition-delay
* transition-duration
* transition-property
* transition-timing-function

## **How to Use CSS Transitions?**

To create a transition effect, you must specify two things:

* the CSS property you want to add an effect to
* the duration of the effect

**Note:** If the duration part is not specified, the transition will have no effect, because the default value is 0.

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background: red;

# transition: width 2s;

# }

# div:hover {

# width: 300px;

# }

# </style>

# </head>

# <body>

# <h1>The transition Property</h1>

# <div></div>

# </body>

# </html>

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background: red;

# transition: width 2s, height 4s;

# }

# div:hover {

# width: 300px;

# height: 300px;

# }

# </style>

# </head>

# <body>

# <h1>The transition Property</h1>

# <div></div>

# </body>

# </html>

## **Transition + Transformation**

The following example adds a transition effect to the transformation

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background: red;

# transition: width 2s, height 2s, transform 2s;

# }

# div:hover {

# width: 300px;

# height: 300px;

# transform: rotate(180deg);

# }

# </style>

# </head>

# <body>

# <div></div>

# </body>

# </html>

## **CSS Animations**

In this chapter you will learn about the following properties:

* @keyframes
* animation-name
* animation-duration
* animation-delay
* animation-iteration-count
* animation-direction
* animation-timing-function
* animation-fill-mode
* animation

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background-color: red;

# animation-name: example;

# animation-duration: 4s;

# }

# @keyframes example {

# from {background-color: red;}

# to {background-color: yellow;}

# }

# </style>

# </head>

# <body>

# <div></div>

# </body>

# </html>

## **What are CSS Animations?**

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

## **The @keyframes Rule**

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background-color: red;

# animation-name: example;

# animation-duration: 4s;

# }

# @keyframes example {

# 0% {background-color: red;}

# 25% {background-color: yellow;}

# 50% {background-color: blue;}

# 100% {background-color: green;}

# }

# </style>

# </head>

# <body>

# <div></div>

# </body>

# </html>

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# div {

# width: 100px;

# height: 100px;

# background-color: red;

# position: relative;

# animation-name: example;

# animation-duration: 4s;

# }

# @keyframes example {

# 0% {background-color:red; left:0px; top:0px;}

# 25% {background-color:yellow; left:200px; top:0px;}

# 50% {background-color:blue; left:200px; top:200px;}

# 75% {background-color:green; left:0px; top:200px;}

# 100% {background-color:red; left:0px; top:0px;}

# }

# </style>

# </head>

# <body>

# <div></div>

# </body>

# </html>

## **CSS Box Sizing**

The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .div1 {

# width: 300px;

# height: 100px;

# border: 1px solid blue;

# }

# .div2 {

# width: 300px;

# height: 100px;

# padding: 50px;

# border: 1px solid red;

# }

# </style>

# </head>

# <body>

# <div class="div1">This div is smaller (width is 300px and height is 100px).</div>

# <br>

# <div class="div2">This div is bigger (width is also 300px and height is 100px).</div>

# </body>

# </html>

## **With the CSS box-sizing Property**

The box-sizing property allows us to include the padding and border in an element's total width and height.

If you set box-sizing: border-box; on an element, padding and border are included in the width and height

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .div1 {

# width: 300px;

# height: 100px;

# border: 1px solid blue;

# box-sizing: border-box;

# }

# .div2 {

# width: 300px;

# height: 100px;

# padding: 50px;

# border: 1px solid red;

# box-sizing: border-box;

# }

# </style>

# </head>

# <body>

# <div class="div1">Both divs are the same size now!</div>

# <br>

# <div class="div2">Hooray!</div>

# </body>

# </html>

# CSS Flexbox

## **CSS Flexbox Layout Module**

Before the Flexbox Layout module, there were four layout modes:

* Block, for sections in a webpage
* Inline, for text
* Table, for two-dimensional table data
* Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# margin: 10px;

# padding: 20px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>

## **Parent Element (Container)**

The flex container becomes flexible by setting the display property to flex

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-direction: column;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-direction: column-reverse;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-direction: row;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-direction: row-reverse;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>

## **The flex-wrap Property**

The flex-wrap property specifies whether the flex items should wrap or not

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-wrap: wrap;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# <div>4</div>

# <div>5</div>

# <div>6</div>

# <div>7</div>

# <div>8</div>

# <div>9</div>

# <div>10</div>

# <div>11</div>

# <div>12</div>

# </div>

# </body>

# </html>

### **Example**

The nowrap value specifies that the flex items will not wrap (this is default)

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# flex-wrap: nowrap;

# background-color: DodgerBlue;

# }

# .flex-container>div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# <div>4</div>

# <div>5</div>

# <div>6</div>

# <div>7</div>

# <div>8</div>

# <div>9</div>

# <div>10</div>

# <div>11</div>

# <div>12</div>

# </div>

# </body>

# </html>

## **The justify-content Property**

### **Example**

# <!DOCTYPE html>

# <html>

# <head>

# <style>

# .flex-container {

# display: flex;

# justify-content: center;

# background-color: DodgerBlue;

# }

# .flex-container > div {

# background-color: #f1f1f1;

# width: 100px;

# margin: 10px;

# text-align: center;

# line-height: 75px;

# font-size: 30px;

# }

# </style>

# </head>

# <body>

# <div class="flex-container">

# <div>1</div>

# <div>2</div>

# <div>3</div>

# </div>

# </body>

# </html>