
CSS 5 TRAINING

Guide By Skyline Ict Consult Ltd
Olisa Macaulay

This HTML5 Course Covers the following.



CSS Introduction	01
CSS selectors	02
Adding css to html	03
CSS Comment	04
CSS Design color	05
Css Background	06
CSS Borders	07
CSS Margin	08
CSS Padding	09
Width and Height	10

CSS Box Model	11
Text CSS	12
CSS Fonts	13
CSS Icons	14
Css links	15
Lists Css	16
Table Css	17
CSS Layout	18
width and max-width	19
position Property	20

This HTML5 Course Covers the following.(2)



float and clear	21
display: inline-block	22
CSS Transparency	23
CSS Navigation Bar	24
Vertical Navigation Bar	25
Horizontal Navigation Bar	26
CSS Dropdowns	27
CSS Image Gallery	28

CSS Forms	29
CSS Counters	30
CSS Website Layout	31
CSS The !important Rule	32

- What you should expect from me

I take my courses very seriously but at the same time I try to make it fun since I know how difficult learning from an instructor This course is fun, and when you need some energy to keep going, you will get it from me.



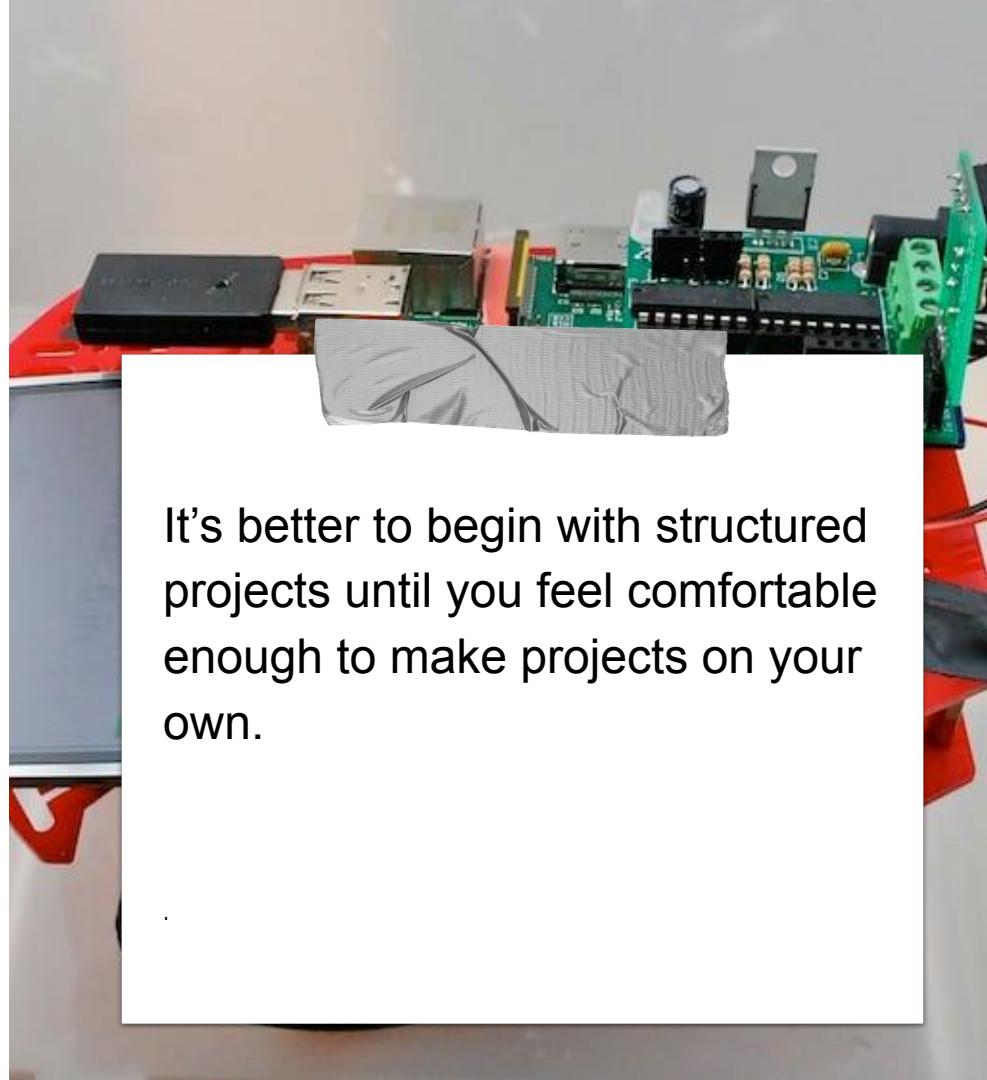
-

Practice, practice and more practice. Every section inside this course has a practice lecture and exercise at the end, reinforcing everything with went over in the lectures.



What's the lesson here? You need to find what motivates you and get excited about it! To get started, find one or two areas that interest you

Once you've learned the basic CSS syntax, start doing projects. Applying your knowledge right away will help you remember everything you've learned.



FRONT-END DEVELOPER

A front end developer, is a professional responsible for the design and implementation of the website interface.

Have you ever looked at your favorite website and wondered why it looked like that, how the buttons worked, or thought, “I wonder how complicated that is?” or, I wish I could do that? While web design determines the way a website looks, front end development is how that design actually gets implemented on the web.

A front end web developer is a **software engineer** who implements web designs through coding languages like **HTML, CSS, and JavaScript**. If you head to any site, you can see the work of a front end developer in the navigation, layouts (including this article), and in the way that a site looks different on your phone (thanks to mobile-first or responsive design).

Key Front End Developers Skills



Front end web developers use three primary coding languages to code the website

HTML,CSS, JavaScript

The code front end developers write runs inside the user's web browser (known as client-side,

as opposed to a back end developer, whose code runs server-side using open source runtime environments like **Django** or with programming languages like Python). **Full stack developers** are comfortable programming with **both front end and back end languages**.

A back end developer is like the engineer who designs and creates the systems that make a city work (electricity, water and sewer, zoning, etc.), while the front end developer is the one who lays out the streets and makes sure everything is connected properly so people can live their lives.

HTML



CSS



JS



Before we begin, install:



1. Web Browser
(lets us view websites)



2. Code Editor
(helps us write code)



VS Code

Cascading Style Sheets,Section: 1

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, css uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.



Why CSS?

CSS saves time: You can write CSS once and reuse the same sheet in multiple HTML pages.

Easy Maintenance: To make a global change simply change the style, and all elements in all the webpages will be updated automatically.



Search Engines: CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.

Superior styles to HTML: CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Offline Browsing: CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

Installing Visual Studio Code on Windows

Visual Studio Code is the most popular code editor and the IDEs provided by Microsoft for writing different programs and languages. It allows the users to develop new code bases for their applications and allow them to successfully optimize them and debug them properly.

It is a very user-friendly code editor and it is supported on all the different types of operating systems like Windows, macOS, and Linux.

It has the support for all the languages like HTML, CSS, Java, Python, JavaScript, React, Node JS, etc.

Follow the below steps to install **Visual Studio Code** on Windows:

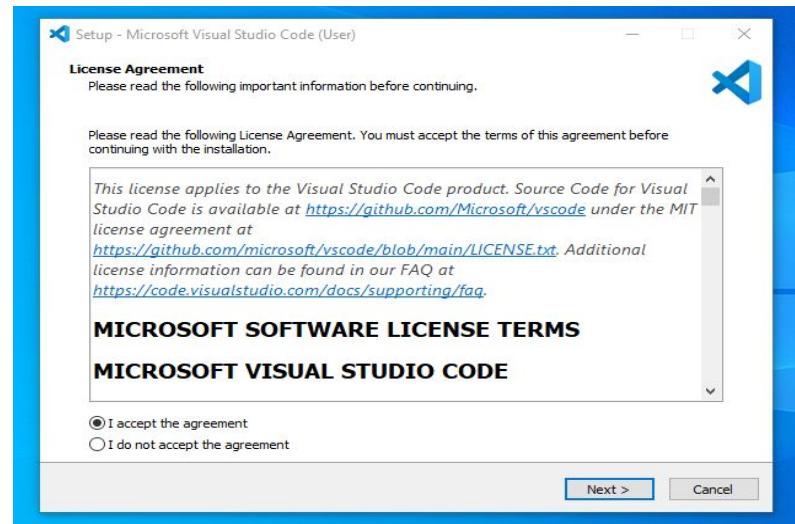
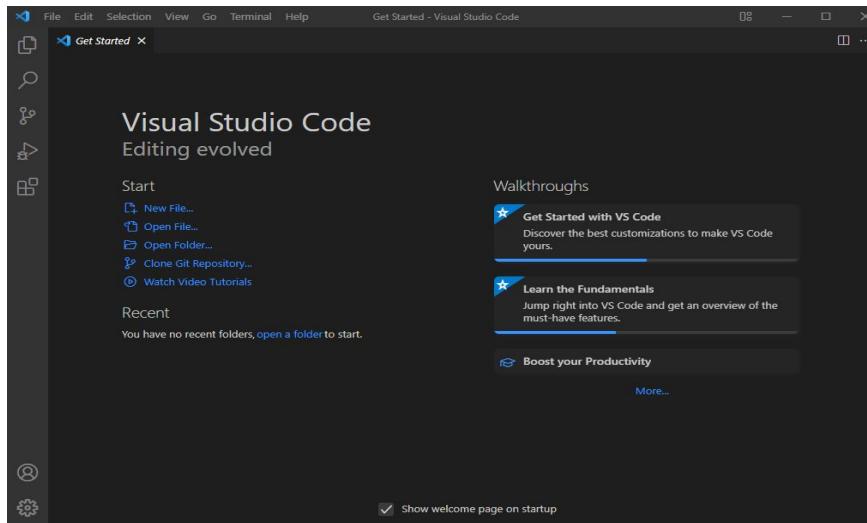
Step 1: Visit the [official website](#) of the **Visual Studio Code** using any web browser like Google Chrome, Microsoft Edge, etc.



Step 2: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 3: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

Step 4: Choose the location data for running the Visual Studio Code. It will then ask you for browsing the location. Then click on Next button.



Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

[Stylesheet 2](#)

[Stylesheet 3](#)

[Stylesheet 4](#)

[No Stylesheet](#)

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:

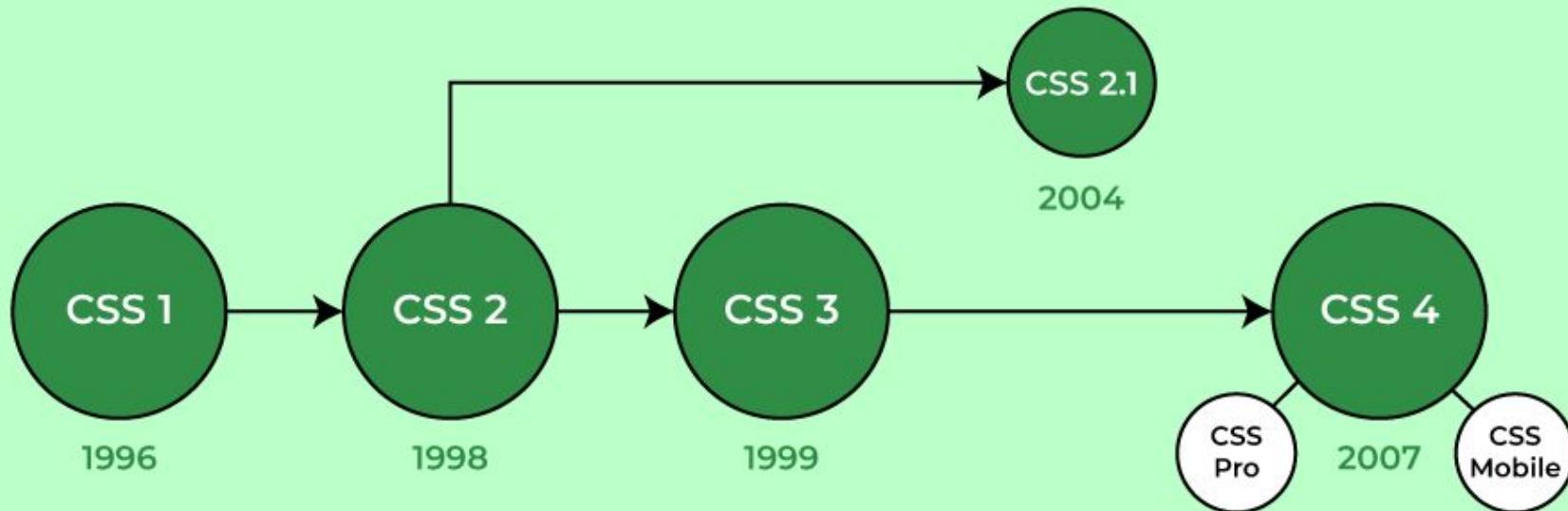
[Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#),
[Stylesheet4](#).

No Styles

Side-Bar

Loreum ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

CSS Released Versions



Why Use CSS?

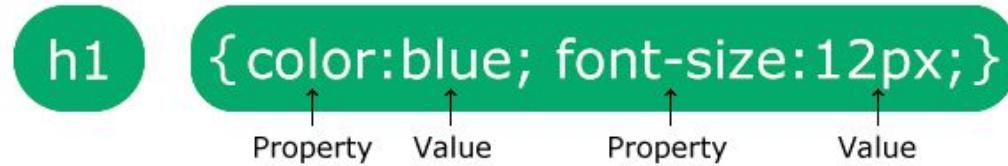
CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Example

```
body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: white;  
    text-align: center;  
}  
  
p {  
    font-family: verdana;  
    font-size: 20px;  
}
```

CSS Syntax

Selector Declaration Declaration



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

In this example all `<p>` elements will be center-aligned, with a red text color:

```
p {  
  color: red;  
  text-align: center;  
}
```



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

You will learn much more about CSS selectors and CSS properties in the next chapters!

CSS selectors: section 2.

CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

CSS Element Selector

CSS Id Selector

CSS Class Selector

CSS Universal Selector

CSS Group Selector



A CSS selector selects the HTML element(s) you want to style.

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 (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (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

Here, all `<p>` elements on the page will be center-aligned, with a red text color:

```
p {  
    text-align: center;  
    color: red;  
}
```

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

The CSS rule below will be applied to the HTML element with `id="para1"`:

```
#para1 {  
    text-align: center;  
    color: red;  
}
```



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

In this example all HTML elements with `class="center"` will be red and center-aligned:

```
.center {  
    text-align: center;  
    color: red;  
}
```

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

Example

In this example only `<p>` elements with `class="center"` will be red and center-aligned:

```
p.center {  
    text-align: center;  
    color: red;  
}
```

HTML elements can also refer to more than one class.

Example

In this example the `<p>` element will be styled according to `class="center"` and to `class="large"`:

```
<p class="center large">This paragraph refers to two classes.</p>
```

Note: A class name cannot start with a number!



The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

```
* {  
    text-align: center;  
    color: blue;  
}
```

The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {  
    text-align: center;  
    color: red;  
}  
  
h2 {  
    text-align: center;  
    color: red;  
}  
  
p {  
    text-align: center;  
    color: red;  
}
```



It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

```
h1, h2, p {  
    text-align: center;  
    color: red;  
}
```



ADDING CSS IN HTML: SECTION 3

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

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" 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:

"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>
```

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

Inline styles are defined within the "style" attribute of the relevant element:

```
<!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>
```



Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Assume that an **external style sheet** has the following style for the `<h1>` element:

```
h1 {  
    color: navy;  
}
```

Then, assume that an **internal style sheet** also has the following style for the `<h1>` element:

```
h1 {  
    color: orange;  
}
```

Example

If the internal style is defined **after** the link to the external style sheet, the <h1> elements will be "orange":

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>
```



Example

However, if the internal style is defined **before** the link to the external style sheet, the <h1> elements will be "navy":

```
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```



CSS Comment: Section 4.

CSS comments are not displayed in the browser, but they can help document your source code.

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 is placed inside the `<style>` element, and starts with `/*` and ends with `*/`:

Example

```
/* This is a single-line comment */  
p {  
    color: red;  
}
```





You can add comments wherever you want in the code:

Example

```
p {  
    color: red; /* Set text color to red */  
}
```

Comments can also span multiple lines:

Example

```
/* This is  
a multi-line  
comment */  
  
p {  
    color: red;  
}
```



HTML and CSS Comments

From the HTML tutorial, you learned that you can add comments to your HTML source by using the `<!--...-->` syntax.

In the following example, we use a combination of HTML and CSS comments:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red; /* Set text color to red */
}
</style>
</head>
<body>

<h2>My Heading</h2>

<!-- These paragraphs will be red -->
<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 Design color Section 5.

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

CSS Color Names

In CSS, a color can be specified by using a predefined color name:

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

Violet

LightGray

CSS/HTML support 140 standard color names.



CSS Background Color

You can set the background color for HTML elements:

Hello World

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lore ipsum...</p>
```

CSS Text Color

You can set the color of text:

Hello World

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

 Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```



CSS Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<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>
```

CSS Color Values

In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

```
rgb(255, 99, 71)
```

```
#ff6347
```

```
hsl(9, 100%, 64%)
```

Same as color name "Tomato", but 50% transparent:

```
rgba(255, 99, 71, 0.5)
```

```
hsla(9, 100%, 64%, 0.5)
```

CSS Background Section 6.

The CSS background properties are used to add 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`
- `background` (shorthand property)

CSS `background-color`

The `background-color` property specifies the background color of an element.

With CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at [CSS Color Values](#) for a complete list of possible color values.



Other Elements

You can set the background color for any HTML elements:

Example

The background color of a page is set like this:

```
body {  
    background-color: lightblue;  
}
```

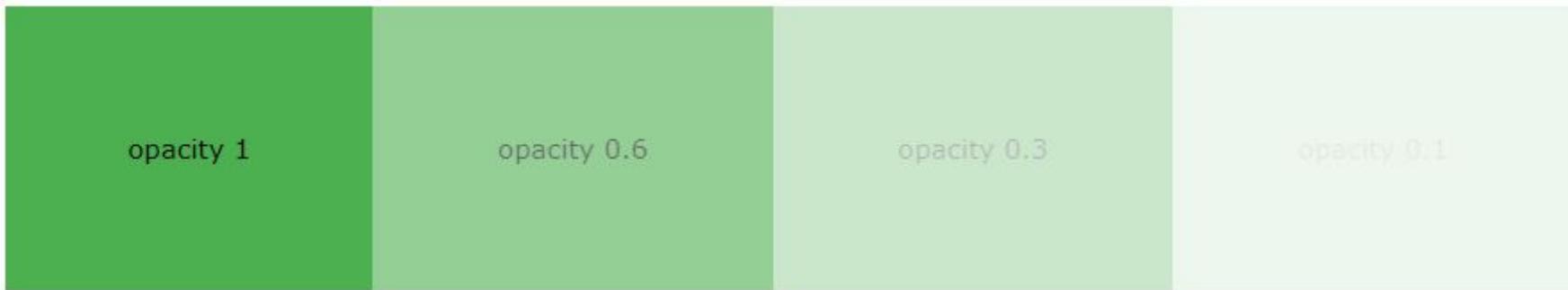
Example

Here, the `<h1>`, `<p>`, and `<div>` elements will have different background colors:

```
h1 {  
    background-color: green;  
}  
  
div {  
    background-color: lightblue;  
}  
  
p {  
    background-color: yellow;  
}
```

Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:



Example

```
div {  
    background-color: green;  
    opacity: 0.3;  
}
```



Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:



You learned from our [CSS Colors Chapter](#), that you can use RGB as a color value. In addition to RGB, you can use an RGB color value with an **alpha** channel (**RGBA**) - which specifies the opacity for a color.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Tip: You will learn more about RGBA Colors in our [CSS Colors Chapter](#).

CSS Borders Section 7.

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

I have borders on all sides.

I have a red bottom border.

I have rounded borders.

I have a blue left 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

Demonstration of the different border styles:

```
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;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.



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

Demonstration of the different border widths:

```
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;  
}
```



Result:

5px border-width



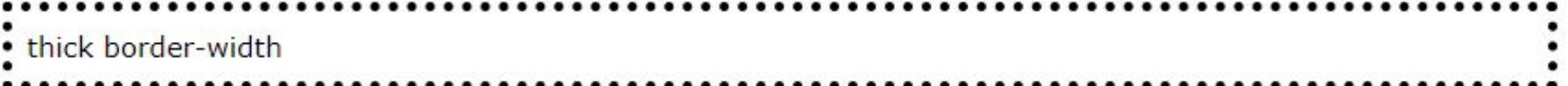
medium border-width



2px border-width



thick border-width



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

```
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 */  
}
```

CSS Rounded Borders

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

Normal border

Round border

Rounder border

Roundest border

Example

```
p {  
    border: 2px solid red;  
    border-radius: 5px;  
}
```

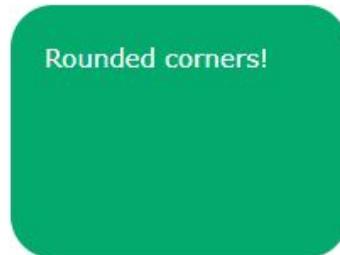
CSS border-radius Property

The CSS `border-radius` property defines the radius of an element's corners.

Tip: This property allows you to add rounded corners to elements!

Here are three examples:

1. Rounded corners for an element with a specified background color:



2. Rounded corners for an element with a border:



Example

```
#rcorners1 {  
    border-radius: 25px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners2 {  
    border-radius: 25px;  
    border: 2px solid #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners3 {  
    border-radius: 25px;  
    background: url(paper.gif);  
    background-position: left top;  
    background-repeat: repeat;  
    padding: 20px;  
    width: 200px;  
    height: 150px;
```

CSS border-radius - Specify Each Corner

The `border-radius` property can have from one to four values. Here are the rules:

Four values - border-radius: 15px 50px 30px 5px; (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner):



Three values - border-radius: 15px 50px 30px; (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner):



Two values - border-radius: 15px 50px; (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners):



One value - border-radius: 15px; (the value applies to all four corners, which are rounded equally):



Example

```
#rcorners1 {  
    border-radius: 15px 50px 30px 5px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners2 {  
    border-radius: 15px 50px 30px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners3 {  
    border-radius: 15px 50px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}
```

CSS border-image Property

The CSS `border-image` property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

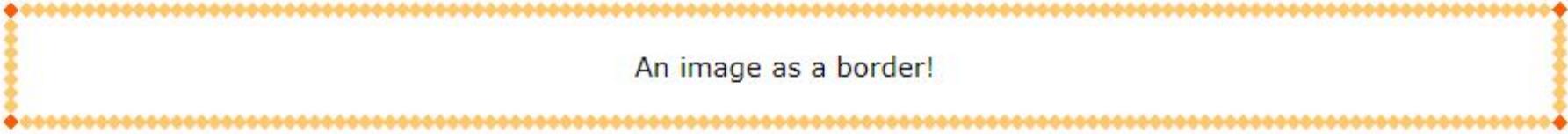
We will use the following image (called "border.png"):



The `border-image` property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

Note: For `border-image` to work, the element also needs the `border` property set!

Here, the middle sections of the image are repeated to create the border:



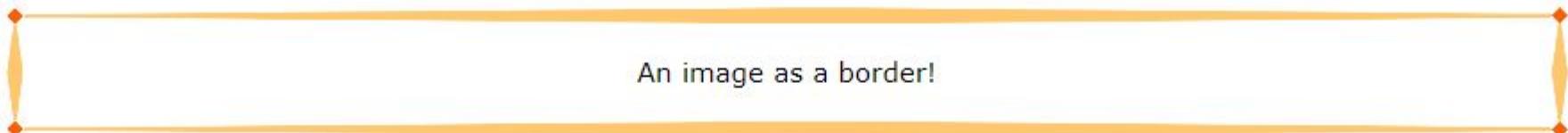
An image as a border!

Here is the code:

Example

```
#borderimg {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 30 round;  
}
```

Here, the middle sections of the image are stretched to create the border:



Here is the code:

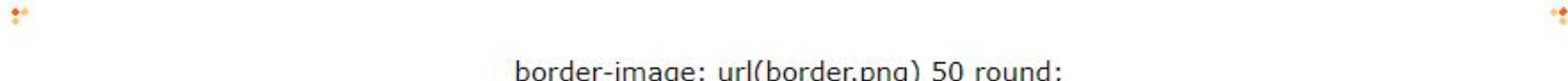
Example

```
#borderimg {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 30 stretch;  
}
```

CSS border-image - Different Slice Values

Different slice values completely changes the look of the border:

Example 1:



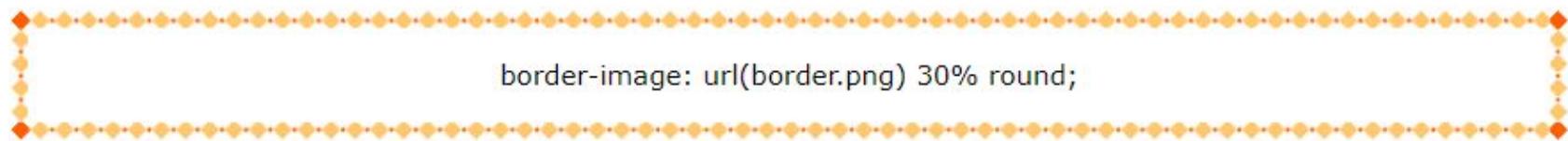
```
border-image: url(border.png) 50 round;
```

Example 2:



```
border-image: url(border.png) 20% round;
```

Example 3:



```
border-image: url(border.png) 30% round;
```

Here is the code:

Example

```
#borderimg1 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 50 round;  
}  
  
#borderimg2 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 20% round;  
}  
  
#borderimg3 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 30% round;  
}
```

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

Demonstration of the different border colors:

```
p.one {  
    border-style: solid;  
    border-color: red;  
}  
  
p.two {  
    border-style: solid;  
    border-color: green;  
}  
  
p.three {  
    border-style: dotted;  
    border-color: blue;  
}
```

Result:

Red border

Green border

Blue border

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

```
p.one {  
    border-style: solid;  
    border-color: red green blue yellow; /* red top, green right, blue bottom and yellow left */  
}
```

HEX Values

The color of the border can also be specified using a hexadecimal value (HEX):

Example

```
p.one {  
    border-style: solid;  
    border-color: #ff0000; /* red */  
}
```

RGB Values

Or by using RGB values:

Example

```
p.one {  
    border-style: solid;  
    border-color: rgb(255, 0, 0); /* red */  
}
```

HSL Values

You can also use HSL values:

Example

```
p.one {  
    border-style: solid;  
    border-color: hsl(0, 100%, 50%); /* red */  
}
```

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

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

Result:



Different Border Styles

The example above gives the same result as this:

Example

```
p {  
    border-style: dotted solid;  
}
```

So, here is how it works:

If the `border-style` property has four values:

- **`border-style: dotted solid double dashed;`**
 - top border is dotted
 - right border is solid
 - bottom border is double
 - left border is dashed

If the `border-style` property has three values:

- **`border-style: dotted solid double;`**
 - top border is dotted
 - right and left borders are solid
 - bottom border is double

If the `border-style` property has two values:

- **`border-style: dotted solid;`**
 - top and bottom borders are dotted
 - right and left borders are solid

If the `border-style` property has one value:

- **border-style: dotted;**
 - all four borders are dotted

Example

```
/* Four values */
p {
    border-style: dotted solid double dashed;
}

/* Three values */
p {
    border-style: dotted solid double;
}

/* Two values */
p {
    border-style: dotted solid;
}

/* One value */
p {
    border-style: dotted;
}
```

CSS Margin Section 8

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`

All the margin properties can have the following values:

- `auto` - the browser calculates the margin
- `length` - specifies a margin in px, pt, cm, etc.
- `%` - specifies a margin in % of the width of the containing element
- `inherit` - specifies that the margin should be `inherited` from the parent element

Tip: Negative values are allowed.

Example

Set different margins for all four sides of a <p> element:

```
p {  
    margin-top: 100px;  
    margin-bottom: 100px;  
    margin-right: 150px;  
    margin-left: 80px;  
}
```

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

Use the margin shorthand property with four values:

```
p {  
    margin: 25px 50px 75px 100px;  
}
```

If the `margin` property has three values:

- **margin: 25px 50px 75px;**
 - top margin is 25px
 - right and left margins are 50px
 - bottom margin is 75px



Example

Use the margin shorthand property with three values:

```
p {  
    margin: 25px 50px 75px;  
}
```



If the `margin` property has two values:

- **margin: 25px 50px;**
 - top and bottom margins are 25px
 - right and left margins are 50px

Example

Use the margin shorthand property with two values:

```
p {  
    margin: 25px 50px;  
}
```

The auto Value

You can set the margin property to `auto` to horizontally center the element within its container.

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

Example

Use `margin: auto`:

```
div {  
    width: 300px;  
    margin: auto;  
    border: 1px solid red;  
}
```

The inherit Value

This example lets the left margin of the `<p class="ex1">` element be inherited from the parent element (`<div>`):

Example

Use of the inherit value:

```
div {  
    border: 1px solid red;  
    margin-left: 100px;  
}  
  
p.ex1 {  
    margin-left: inherit;  
}
```

Sometimes two margins collapse into a single margin.

Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

Look at the following example:

Example

Demonstration of margin collapse:

```
h1 {  
    margin: 0 0 50px 0;  
}  
  
h2 {  
    margin: 20px 0 0 0;  
}
```

All CSS Margin Properties

Property	Description
<code>margin</code>	A shorthand property for setting the margin properties in one declaration
<code>margin-bottom</code>	Sets the bottom margin of an element
<code>margin-left</code>	Sets the left margin of an element
<code>margin-right</code>	Sets the right margin of an element
<code>margin-top</code>	Sets the top margin of an element

CSS Padding Section 9.

Padding is used to create space around an element's content, inside of any defined borders.

This element has a padding of 70px.

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

Note: Negative values are not allowed.

Example

Set different padding for all four sides of a <div> element:

```
div {  
    padding-top: 50px;  
    padding-right: 30px;  
    padding-bottom: 50px;  
    padding-left: 80px;  
}
```

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

Use the padding shorthand property with four values:

```
div {  
  padding: 25px 50px 75px 100px;  
}
```

If the `padding` property has three values:

- **`padding: 25px 50px 75px;`**
 - top padding is 25px
 - right and left paddings are 50px
 - bottom padding is 75px

Example

Use the padding shorthand property with three values:

```
div {  
  padding: 25px 50px 75px;  
}
```

If the `padding` property has two values:

- **`padding: 25px 50px;`**
 - top and bottom paddings are 25px
 - right and left paddings are 50px

Example

Use the padding shorthand property with two values:

```
div {  
  padding: 25px 50px;  
}
```

Padding and Element Width

The CSS `width` property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element ([the box model](#)).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

Example

Here, the `<div>` element is given a width of 300px. However, the actual width of the `<div>` element will be 350px (300px + 25px of left padding + 25px of right padding):

```
div {  
    width: 300px;  
    padding: 25px;  
}
```



To keep the width at 300px, no matter the amount of padding, you can use the `box-sizing` property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease.

Example

Use the `box-sizing` property to keep the width at 300px, no matter the amount of padding:

```
div {  
    width: 300px;  
    padding: 25px;  
    box-sizing: border-box;  
}
```

Width and Height Section 10.

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

The CSS `max-width` property is used to set the maximum width of an element.

This element has a height of 50 pixels and a width of 100%.

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
-



CSS height and width Examples

This element has a height of 200 pixels and a width of 50%



Example

Set the height and width of a <div> element:

```
div {  
    height: 200px;  
    width: 50%;  
    background-color: powderblue;  
}
```

This element has a height of 100 pixels and a width of 500 pixels.

Example

Set the `height` and `width` of another `<div>` element:

```
div {  
    height: 100px;  
    width: 500px;  
    background-color: powderblue;  
}
```



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!

Setting max-width

The `max-width` property is used to set the maximum width of an element.

The `max-width` can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the `<div>` above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

This element has a height of 100 pixels and a max-width of 500 pixels.

Note: If you for some reason use both the `width` property and the `max-width` property on the same element, and the value of the `width` property is larger than the `max-width` property; the `max-width` property will be used (and the `width` property will be ignored).

Example

This <div> element has a height of 100 pixels and a max-width of 500 pixels:

```
div {  
    max-width: 500px;  
    height: 100px;  
    background-color: powderblue;  
}
```

CSS Box Model Section 11.

The CSS Box Model

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:

Margin

Border

Padding

Content

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

Demonstration of the box model:

```
div {  
    width: 300px;  
    border: 15px solid green;  
    padding: 50px;  
    margin: 20px;  
}
```

Width and Height of an Element

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

Example

This <div> element will have a total width of 350px:

```
div {  
    width: 320px;  
    padding: 10px;  
    border: 5px solid gray;  
    margin: 0;  
}
```

Here is the calculation:

```
320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px
```

The total width of an element should be calculated like this:

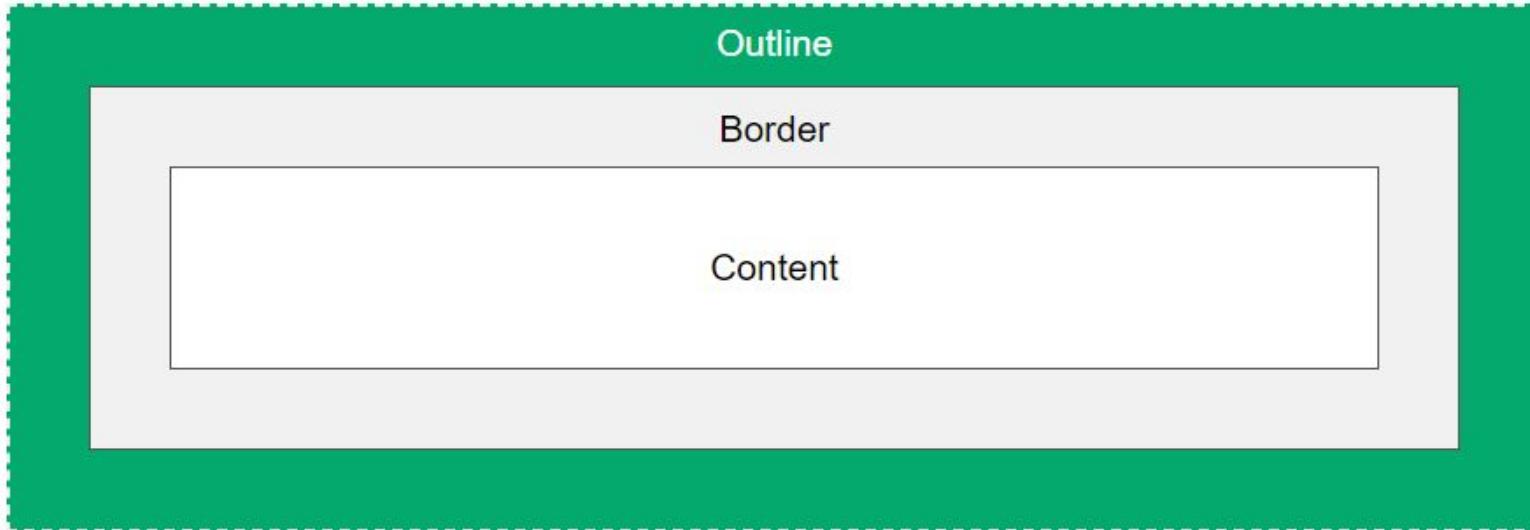
Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

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**

Note: Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the 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

Demonstration of the different outline styles:

```
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;}
```

Result:

A dotted outline.

A dashed outline.

A solid outline.

A double outline.

A groove outline. The effect depends on the outline-color value.

A ridge outline. The effect depends on the outline-color value.

An inset outline. The effect depends on the outline-color value.

An outset outline. The effect depends on the outline-color value.

CSS Text- Section 12

Text Color and Background Color

In this example, we define both the `background-color` property and the `color` property:

Example

```
body {  
    background-color: lightgrey;  
    color: blue;  
}  
  
h1 {  
    background-color: black;  
    color: white;  
}  
  
div {  
    background-color: blue;  
    color: white;  
}
```

Text Alignment and Text Direction

In this chapter you will learn about the following properties:

- `text-align`
- `text-align-last`
- `direction`
- `unicode-bidi`
- `vertical-align`

Text Alignment

The `text-align` property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

Example

```
h1 {  
    text-align: center;  
}  
  
h2 {  
    text-align: left;  
}  
  
h3 {  
    text-align: right;  
}
```

When the `text-align` property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

Example

```
div {  
    text-align: justify;  
}
```

Text Align Last

The `text-align-last` property specifies how to align the last line of a text.

Example

Align the last line of text in three `<p>` elements:

```
p.a {  
    text-align-last: right;  
}  
  
p.b {  
    text-align-last: center;  
}  
  
p.c {  
    text-align-last: justify;  
}
```

Text Decoration

In this chapter you will learn about the following properties:

- `text-decoration-line`
- `text-decoration-color`
- `text-decoration-style`
- `text-decoration-thickness`
- `text-decoration`



Add a Decoration Line to Text

The `text-decoration-line` property is used to add a decoration line to text.

Tip: You can combine more than one value, like `overline` and `underline` to display `lines` both over and under a text.

Example

```
h1 {  
  text-decoration-line: overline;  
}  
  
h2 {  
  text-decoration-line: line-through;  
}  
  
h3 {  
  text-decoration-line: underline;  
}  
  
p {  
  text-decoration-line: overline underline;  
}
```

The Shorthand Property

The `text-decoration` property is a shorthand property for:

- `text-decoration-line` (required)
- `text-decoration-color` (optional)
- `text-decoration-style` (optional)
- `text-decoration-thickness` (optional)

Example

```
h1 {  
    text-decoration: underline;  
}  
  
h2 {  
    text-decoration: underline red;  
}  
  
h3 {  
    text-decoration: underline red double;  
}  
  
p {  
    text-decoration: underline red double 5px;  
}
```



Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example

```
p.uppercase {  
    text-transform: uppercase;  
}  
  
p.lowercase {  
    text-transform: lowercase;  
}  
  
p.capitalize {  
    text-transform: capitalize;  
}
```



Text Spacing

In this chapter you will learn about the following properties:

- `text-indent`
- `letter-spacing`
- `line-height`
- `word-spacing`
- `white-space`



Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text:

Letter Spacing

The `letter-spacing` property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

Example

```
h1 {  
    letter-spacing: 5px;  
}  
  
h2 {  
    letter-spacing: -2px;  
}
```



Line Height

The `line-height` property is used to specify the space between lines:

Example

```
p.small {  
  line-height: 0.8;  
}  
  
p.big {  
  line-height: 1.8;  
}
```



Line Height

The `line-height` property is used to specify the space between lines:

Example

```
p.small {  
    line-height: 0.8;  
}  
  
p.big {  
    line-height: 1.8;  
}
```



Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Generic Font Families

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

All the different font names belong to one of the generic font families.

Difference Between Serif and Sans-serif Fonts

F

Sans-serif

F

Serif

F

Serif
(red serifs)



Some Font Examples

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	<i>Brush Script MT</i> <i>Lucida Handwriting</i>



The CSS font-family Property

In CSS, we use the `font-family` property to specify the font of a text.

Note: If the font name is more than one word, it must be in quotation marks, like: "Times New Roman".

Tip: The `font-family` property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. 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). The font names should be separated with comma. Read more about fallback fonts in the [next chapter](#).

Example

Specify some different fonts for three paragraphs:

```
.p1 {  
    font-family: "Times New Roman", Times, serif;  
}  
  
.p2 {  
    font-family: Arial, Helvetica, sans-serif;  
}  
  
.p3 {  
    font-family: "Lucida Console", "Courier New", monospace;  
}
```



What are Web Safe Fonts?

Web safe fonts are fonts that are universally installed across all browsers and devices.

Fallback Fonts

However, there are no 100% completely web safe fonts. There is always a chance that a font is not found or is not installed properly.

Therefore, it is very important to always use fallback fonts.

This means that you should add a list of similar "backup fonts" in the `font-family` property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

Example

Here, there are three font types: Tahoma, Verdana, and sans-serif. The second and third fonts are backups, in case the first one is not found.

```
p {  
    font-family: Tahoma, Verdana, sans-serif;  
}
```

Best Web Safe Fonts for HTML and CSS

The following list are the best web safe fonts for HTML and CSS:

- Arial (sans-serif)
- Verdana (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

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

```
p.normal {  
  font-style: normal;  
}  
  
p.italic {  
  font-style: italic;  
}  
  
p.oblique {
```

Font Weight

The `font-weight` property specifies the weight of a font:

Example

```
p.normal {  
    font-weight: normal;  
}  
  
p.thick {  
    font-weight: bold;  
}
```



Font Variant

The `font-variant` property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}
```



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



Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {  
    font-size: 40px;  
}  
  
h2 {  
    font-size: 30px;  
}  
  
p {  
    font-size: 14px;  
}
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: $\text{pixels}/16=\text{em}$

Example

```
h1 {  
    font-size: 2.5em; /* 40px/16=2.5em */  
}  
  
h2 {  
    font-size: 1.875em; /* 30px/16=1.875em */  
}  
  
p {  
    font-size: 0.875em; /* 14px/16=0.875em */  
}
```

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {  
    font-size: 100%;  
}  
  
h1 {  
    font-size: 2.5em;  
}  
  
h2 {  
    font-size: 1.875em;  
}  
  
p {  
    font-size: 0.875em;  
}
```

Google Fonts

If you do not want to use any of the standard fonts in HTML, you can use Google Fonts.

Google Fonts are free to use, and have more than 1000 fonts to choose from.

How To Use Google Fonts

Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

Example

Here, we want to use a font named "Sofia" from Google Fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
```

Result Result:

Sofia Font

 Lorem ipsum dolor sit amet.

 123456790

Example

Here, we want to use a font named "Audiotwide" from Google Fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiotwide">
<style>
body {
  font-family: "Audiotwide", sans-serif;
}
</style>
</head>
```

Result:

Audiotwide Font

Lorem ipsum dolor sit amet.

Example

Here, we want to use a font named "Audiotwide" from Google Fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiotwide">
<style>
body {
  font-family: "Audiotwide", sans-serif;
}
</style>
</head>
```

Result:

Audiotwide Font

Lorem ipsum dolor sit amet.

Use Multiple Google Fonts

To use multiple Google fonts, just separate the font names with a pipe character (|), like this:

Example

Request multiple fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">
<style>
h1.a {font-family: "Audiowide", sans-serif;}
h1.b {font-family: "Sofia", sans-serif;}
h1.c {font-family: "Trirong", serif;}
</style>
</head>
```

Result:

Use Multiple Google Fonts

To use multiple Google fonts, just separate the font names with a pipe character (|), like this:

Example

Request multiple fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">
<style>
h1.a {font-family: "Audiowide", sans-serif;}
h1.b {font-family: "Sofia", sans-serif;}
h1.c {font-family: "Trirong", serif;}
</style>
</head>
```

Result:

Styling Google Fonts

Of course you can style Google Fonts as you like, with CSS!

Example

Style the "Sofia" font:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
  font-size: 30px;
  text-shadow: 3px 3px 3px #ababab;
}
</style>
</head>
```

Result:

Sofia Font

Enabling Font Effects

Google has also enabled different font effects that you can use.

First add `effect=effectname` to the Google API, then add a special class name to the element that is going to use the special effect. The class name always starts with `font-effect-` and ends with the `effectname`.

Add the fire effect to the "Sofia" font:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia&effect=fire">
<style>
body {
  font-family: "Sofia", sans-serif;
  font-size: 30px;
}
</style>
</head>
<body>

<h1 class="font-effect-fire">Sofia on Fire</h1>

</body>
```

Result:

The image shows the text "Sofia on Fire" in a bold, yellow font. The font has a "fire" effect applied, which creates a horizontal gradient at the bottom of each character, transitioning from a bright yellow on the left to a vibrant orange on the right. The text is centered and appears to be part of a larger web page layout.

CSS Icons Section 14.

Icons can easily be added to your HTML page, by using an icon library.



How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like `<i>` or ``).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the `<head>` section of your HTML page:

```
<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>
```

Read more about how to get started with Font Awesome in our [Font Awesome 5 tutorial](#).

Note: No downloading or installation is required!

Example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
</head>
<body>

<i class="fas fa-cloud"></i>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>

</body>
</html>
```



Result:



Bootstrap Icons

To use the Bootstrap glyphicons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

Note: No downloading or installation is required!



Note: No downloading or installation is required!

Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>

<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>

</body>
</html>
```



Result:



Google Icons

To use the Google icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

Note: No downloading or installation is required!

Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>

</body>
</html>
```



Result:



CSS Links Section 15.

With CSS, links can be styled in many different ways.

Text Link Text Link

Link Button

Link Button

Styling Links

Links can be styled with any CSS property (e.g. `color`, `font-family`, `background`, etc.).

Example

```
a {  
  color: hotpink;  
}
```

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

```
/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}
```



```
/* selected link */  
a:active {  
    color: blue;  
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
 - a:active MUST come after a:hover
-

Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

Example

```
a:link {  
    text-decoration: none;  
}  
  
a:visited {  
    text-decoration: none;  
}  
  
a:hover {  
    text-decoration: underline;  
}  
  
a:active {  
    text-decoration: underline;  
}
```

Background Color

The `background-color` property can be used to specify a background color for links:

Example

```
a:link {  
    background-color: yellow;  
}  
  
a:visited {  
    background-color: cyan;  
}  
  
a:hover {  
    background-color: lightgreen;  
}  
  
a:active {  
    background-color: hotpink;  
}
```



Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Example

```
a:link, a:visited {  
    background-color: #f44336;  
    color: white;  
    padding: 14px 25px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
}  
  
a:hover, a:active {  
    background-color: red;  
}
```



More Examples

Example

This example demonstrates how to add other styles to hyperlinks:

```
a.one:link {color: #ff0000;}  
a.one:visited {color: #0000ff;}  
a.one:hover {color: #ffcc00;}  
  
a.two:link {color: #ff0000;}  
a.two:visited {color: #0000ff;}  
a.two:hover {font-size: 150%;}  
  
a.three:link {color: #ff0000;}  
a.three:visited {color: #0000ff;}  
a.three:hover {background: #66ff66;}  
  
a.four:link {color: #ff0000;}  
a.four:visited {color: #0000ff;}  
a.four:hover {font-family: monospace;}  
  
a.five:link {color: #ff0000; text-decoration: none;}  
a.five:visited {color: #0000ff; text-decoration: none;}
```



Example

Another example of how to create link boxes/buttons:

```
a:link, a:visited {  
    background-color: white;  
    color: black;  
    border: 2px solid green;  
    padding: 10px 20px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
}  
  
a:hover, a:active {  
    background-color: green;  
    color: white;  
}
```



Example

This example demonstrates the different types of cursors (can be useful for [links](#)):

```
<span style="cursor: auto">auto</span><br>
<span style="cursor: crosshair">crosshair</span><br>
<span style="cursor: default">default</span><br>
<span style="cursor: e-resize">e-resize</span><br>
<span style="cursor: help">help</span><br>
<span style="cursor: move">move</span><br>
<span style="cursor: n-resize">n-resize</span><br>
<span style="cursor: ne-resize">ne-resize</span><br>
<span style="cursor: nw-resize">nw-resize</span><br>
<span style="cursor: pointer">pointer</span><br>
<span style="cursor: progress">progress</span><br>
<span style="cursor: s-resize">s-resize</span><br>
<span style="cursor: se-resize">se-resize</span><br>
<span style="cursor: sw-resize">sw-resize</span><br>
<span style="cursor: text">text</span><br>
<span style="cursor: w-resize">w-resize</span><br>
<span style="cursor: wait">wait</span>
```



Lists Css Section 16.

Unordered Lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Ordered Lists:

1. Coffee
2. Tea
3. Coca Cola

- I. Coffee
- II. Tea
- III. Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (``) - the list items are marked with bullets
- ordered lists (``) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items



Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

The following example shows some of the available list item markers:

Example

```
ul.a {  
    list-style-type: circle;  
}  
  
ul.b {  
    list-style-type: square;  
}  
  
ol.c {  
    list-style-type: upper-roman;  
}  
  
ol.d {  
    list-style-type: lower-alpha;  
}
```



RC:1481530

An Image as The List Item Marker

The `list-style-image` property specifies an image as the list item marker:

Example

```
ul {  
    list-style-image: url('sqpurple.gif');  
}
```

Position The List Item Markers

The `list-style-position` property specifies the position of the list-item markers (bullet points).

"list-style-position: outside;" means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically. This is default:

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola

"list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start:

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola



Example

```
ul.a {  
    list-style-position: outside;  
}  
  
ul.b {  
    list-style-position: inside;  
}
```

Remove Default Settings

The `list-style-type:none` property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add `margin:0` and `padding:0` to `` or ``:

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

List - Shorthand property

The `list-style` property is a shorthand property. It is used to set all the `list` properties in one declaration:

Example

```
ul {  
    list-style: square inside url("sqpurple.gif");  
}
```

When using the shorthand property, the order of the property values are:

- `list-style-type` (if a `list-style-image` is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- `list-style-position` (specifies whether the list-item markers should appear inside or outside the content flow)
- `list-style-image` (specifies an image as the list item marker)

If one of the property values above are missing, the default value for the missing property will be inserted, if any.

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the `` or `` tag, affects the entire list, while properties added to the `` tag will affect the individual list items:

Example

```
ol {  
    background: #ff9999;  
    padding: 20px;  
}  
  
ul {  
    background: #3399ff;  
    padding: 20px;  
}  
  
ol li {  
    background: #ffe5e5;  
    padding: 5px;  
    margin-left: 35px;  
}  
  
ul li {  
    background: #cce5ff;  
    margin: 5px;  
}
```



Result:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

Table Css section 17.

The look of an HTML table can be greatly improved with CSS:

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Example

```
table, th, td {  
    border: 1px solid black;  
}
```

Full-Width Table

The table above might seem small in some cases. If you need a table that should span the entire screen (full-width), add `width: 100%` to the `<table>` element:

Example

```
table {  
    width: 100%;  
}
```

Double Borders

Notice that the table in the examples above have double borders. This is because both the table and the `<th>` and `<td>` elements have separate borders.

To remove double borders, take a look at the example below.

Collapse Table Borders

The `border-collapse` property sets whether the table borders should be collapsed into a single border:

Example

```
table {  
    border-collapse: collapse;  
}
```



Example

```
table {  
    border: 1px solid black;  
}
```

CSS Table Size

Table Width and Height

The width and height of a table are defined by the `width` and `height` properties.

The example below sets the width of the table to 100%, and the height of the `<th>` elements to 70px:

Example

```
table {  
    width: 100%;  
}  
  
th {  
    height: 70px;  
}
```



Example

```
table {  
    width: 50%;  
}  
  
th {  
    height: 70px;  
}
```

CSS Table Alignment

Horizontal Alignment

The `text-align` property sets the horizontal alignment (like `left`, `right`, or `center`) of the content in `<th>` or `<td>`.

By default, the content of `<th>` elements are center-aligned and the content of `<td>` elements are left-aligned.

To center-align the content of `<td>` elements as well, use `text-align: center`:

Example

```
td {  
    text-align: center;  
}
```

Example

```
td {  
    text-align: center;  
}
```

Example

```
th {  
    text-align: left;  
}
```

Vertical Alignment

The `vertical-align` property sets the vertical alignment (like top, bottom, or middle) of the content in `<th>` or `<td>`.

By default, the vertical alignment of the content in a table is middle (for both `<th>` and `<td>` elements).

The following example sets the vertical text alignment to bottom for `<td>` elements:

Example

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```

CSS Table Style

Table Padding

To control the space between the border and the content in a table, use the `padding` property on `<td>` and `<th>` elements:

Example

```
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

Horizontal Dividers

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Add the `border-bottom` property to `<th>` and `<td>` for horizontal dividers:

Example

```
th, td {  
    border-bottom: 1px solid #ddd;  
}
```

Hoverable Table

Use the `:hover` selector on `<tr>` to highlight table rows on mouse over:

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
tr:hover {background-color: #f5f5f5;}
```

Striped Tables

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

For zebra-striped tables, use the `nth-child()` selector and add a `background-color` to all even (or odd) table rows:

Example

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Table Color

The example below specifies the background color and text color of <th> elements:

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
th {  
    background-color: #04AA6D;  
    color: white;  
}
```

CSS Layout - section 18.

The `display` property is the most important CSS property for controlling layout.

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).

The `<div>` element is a block-level element.

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.

This is an inline `` element inside a paragraph.

Examples of inline elements:

- ``
 - `<a>`
 - ``
-

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.

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

Using The id Attribute in JavaScript

The `id` attribute can also be used by JavaScript to perform some tasks for that specific element.

JavaScript can access an element with a specific id with the `getElementById()` method:

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline `` elements for horizontal menus:

Example

```
li {  
    display: inline;  
}
```

The following example displays elements as block elements:

Example

```
span {  
    display: block;  
}
```

The following example displays <a> elements as block elements:

Example

```
a {  
    display: block;  
}
```

Hide an Element - display:none or visibility:hidden?

display:none



Remove

visibility:hidden



Hide

Reset



Reset All

Hiding an element can be done by setting the `display` property to `none`. The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {  
    display: none;  
}
```

`visibility:hidden;` also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

Example

```
h1.hidden {  
    visibility: hidden;  
}
```

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the `width` of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to `auto`, to horizontally center the element within its container. The element will take up the specified `width`, and the remaining space will be split equally between the two margins:

This `<div>` element has a width of 500px, and margin set to `auto`.

Note: The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

Example

```
div.ex1 {  
    width: 500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}  
  
div.ex2 {  
    max-width: 500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}
```

CSS The position Property 20.

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

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.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

This `<div>` element has `position: static;`

Here is the CSS that is used:

Example

```
div.static {  
    position: static;  
    border: 3px solid #73AD21;  
}
```



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.

This `<div>` element has `position: relative;`

Here is the CSS that is used:

Example

```
div.relative {  
    position: relative;  
    left: 30px;  
    border: 3px solid #73AD21;  
}
```

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.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {  
    position: fixed;  
    bottom: 0;  
    right: 0;  
    width: 300px;  
    border: 3px solid #73AD21;  
}
```



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.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Here is a simple example:

This `<div>` element has `position: relative;`

This `<div>` element has
`position: absolute;`

Here is the CSS that is used:

Example

```
div.relative {  
    position: relative;  
    width: 400px;  
    height: 200px;  
    border: 3px solid #73AD21;  
}  
  
div.absolute {  
    position: absolute;  
    top: 80px;  
    right: 0;  
    width: 200px;  
    height: 100px;  
    border: 3px solid #73AD21;  
}
```

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`).

Try to scroll inside this frame to understand how sticky positioning works.

I am sticky!

Lore ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Note: Internet Explorer does not support sticky positioning. Safari requires a `-webkit-` prefix (see example below). You must also specify at least one of `top`, `right`, `bottom` or `left` for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (`top: 0`), when you reach its scroll position.

Example

```
div.sticky {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}
```

In this example, the sticky element sticks to the top of the page (`top: 0`), when you reach its scroll position.

Example

```
div.sticky {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}
```

CSS Layout - float and clear 21.

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.

Float Left

Float Right

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;

The following example specifies that an image should float to the right in a text:

Etiam ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...



Example

```
img {  
    float: right;  
}
```

Example - float: left;

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



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Example

```
img {  
  float: left;  
}
```

Example - Float Next To Each Other

Normally div elements will be displayed on top of each other. However, if we use `float: left` we can let elements float next to each other:

Example

```
div {  
    float: left;  
    padding: 15px;  
}  
  
.div1 {  
    background: red;  
}  
  
.div2 {  
    background: yellow;  
}  
  
.div3 {  
    background: green;  
}
```



The clear Property

When we use the `float` property, and we want the next element below (not on right or left), we will have to use the `clear` property.

The `clear` property specifies what should happen with the element that is next to a floating element.

The `clear` property can have one of the following values:

- `none` - The element is not pushed below left or right floated elements. This is default
- `left` - The element is pushed below left floated elements
- `right` - The element is pushed below right floated elements
- `both` - The element is pushed below both left and right floated elements
- `inherit` - The element inherits the clear value from its parent

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.



Example

This example clears the float to the left. Here, it means that the <div2> element is pushed below the left floated <div1> element:

```
div1 {  
    float: left;  
}  
  
div2 {  
    clear: left;  
}
```

The clearfix Hack

If a floated element is taller than the containing element, it will "overflow" outside of its container. We can then add a clearfix hack to solve this problem:

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



Example

```
.clearfix {  
  overflow: auto;  
}
```

The `overflow: auto` clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

Example

```
.clearfix::after {  
    content: "";  
    clear: both;  
    display: table;  
}
```

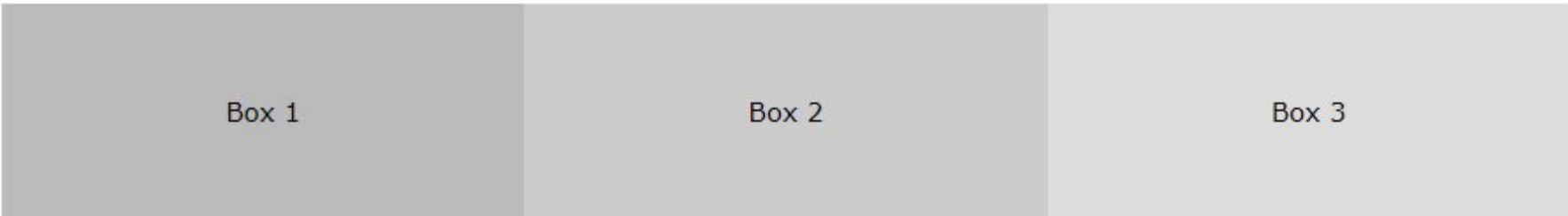
This page contains common float examples.

Grid of Boxes / Equal Width Boxes



Box 1

Box 2



Box 1

Box 2

Box 3

With the `float` property, it is easy to float boxes of content side by side:

With the `float` property, it is easy to float boxes of content side by side:

Example

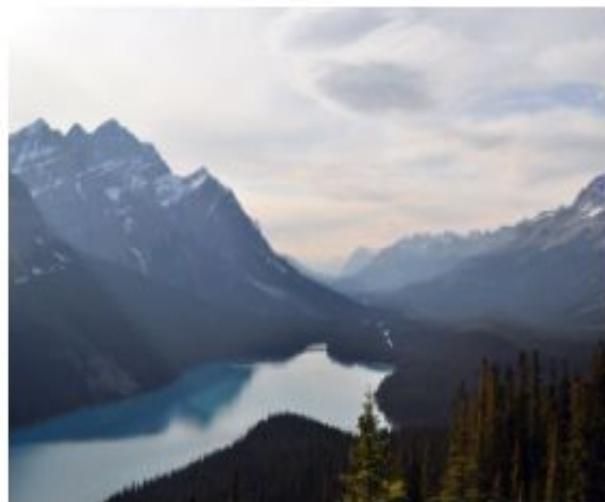
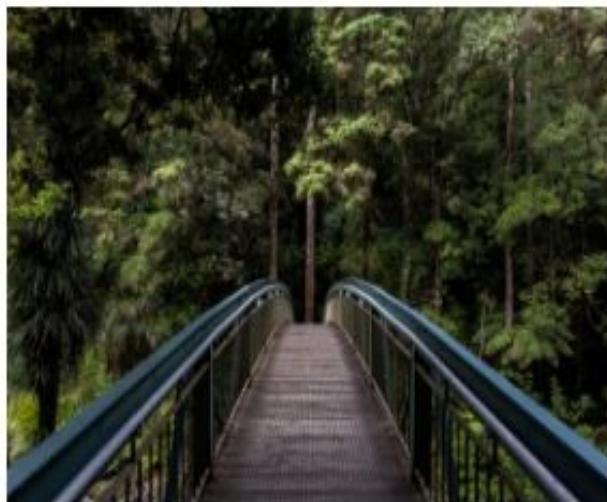
```
* {  
  box-sizing: border-box;  
}  
  
.box {  
  float: left;  
  width: 33.33%; /* three boxes (use 25% for four, and 50% for two, etc) */  
  padding: 50px; /* if you want space between the images */  
}
```

What is box-sizing?

You can easily create three floating boxes side by side. However, when you add something that enlarges the width of each box (e.g. padding or borders), the box will break. The `box-sizing` property allows us to include the padding and border in the box's total width (and height), making sure that the padding stays inside of the box and that it does not break.

You can read more about the box-sizing property in our [CSS Box Sizing Chapter](#).

Images Side By Side



The grid of boxes can also be used to display images side by side:

Example

```
.img-container {  
    float: left;  
    width: 33.33%; /* three containers (use 25% for four, and 50% for two, etc) */  
    padding: 5px; /* if you want space between the images */  
}
```

display: inline-block Section 22.

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.

The following example shows the different behavior of `display: inline`, `display: inline-block` and `display: block`:

```
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;
```

```
}
```

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

```
.nav {  
    background-color: yellow;  
    list-style-type: none;  
    text-align: center;  
    padding: 0;  
    margin: 0;  
}  
  
.nav li {  
    display: inline-block;  
    font-size: 20px;  
    padding: 20px;  
}
```

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:

This div element is centered.

Example

```
.center {  
    margin: auto;  
    width: 50%;  
    border: 3px solid green;  
    padding: 10px;  
}
```

Note: Center aligning has no effect if the `width` property is not set (or set to 100%).

Center Align Text

To just center the text inside an element, use `text-align: center;`

This text is centered.

Example

```
.center {  
    text-align: center;  
    border: 3px solid green;  
}
```

Center an Image

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



Example

```
img {  
  display: block;  
  margin-left: auto;  
  margin-right: auto;  
  width: 40%;  
}
```

Left and Right Align - Using position

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

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

Example

```
.right {  
    position: absolute;  
    right: 0px;  
    width: 300px;  
    border: 3px solid #73AD21;  
    padding: 10px;  
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Left and Right Align - Using float

Another method for aligning elements is to use the `float` property:

Example

```
.right {  
    float: right;  
    width: 300px;  
    border: 3px solid #73AD21;  
    padding: 10px;  
}
```

The clearfix Hack

Note: If an element is taller than the element containing it, and it is floated, it will overflow outside of its container. You can use the "clearfix hack" to fix this (see example below).

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



Then we can add the clearfix hack to the containing element to fix this problem:

Example

```
.clearfix::after {  
    content: "";  
    clear: both;  
    display: table;  
}
```

Center Vertically - Using padding

There are many ways to center an element vertically in CSS. A simple solution is to use top and bottom `padding`:

I am vertically centered.

Example

```
.center {  
    padding: 70px 0;  
    border: 3px solid green;  
}
```

To center both vertically and horizontally, use `padding` and `text-align: center`:

I am vertically and horizontally centered.

Example

```
.center {  
    padding: 70px 0;  
    border: 3px solid green;  
    text-align: center;  
}
```

Center Vertically - Using line-height

Another trick is to use the `line-height` property with a value that is equal to the `height` property:

I am vertically and horizontally centered.

Example

```
.center {  
    line-height: 200px;  
    height: 200px;  
    border: 3px solid green;  
    text-align: center;  
}  
  
/* If the text has multiple lines, add the following: */  
.center p {  
    line-height: 1.5;  
    display: inline-block;  
    vertical-align: middle;  
}
```

Center Vertically - Using position & transform

If `padding` and `line-height` are not options, another solution is to use positioning and the `transform` property:

I am vertically and horizontally centered.

Example

```
.center {  
    height: 200px;  
    position: relative;  
    border: 3px solid green;  
}  
  
.center p {  
    margin: 0;  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
}
```

Tip: You will learn more about the transform property in our [2D Transforms Chapter](#).

Center Vertically - Using Flexbox

You can also use flexbox to center things. Just note that flexbox is not supported in IE10 and earlier versions:

I am vertically and horizontally centered.

Example

```
.center {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 200px;  
  border: 3px solid green;  
}
```

CSS Opacity / Transparency Section 23.

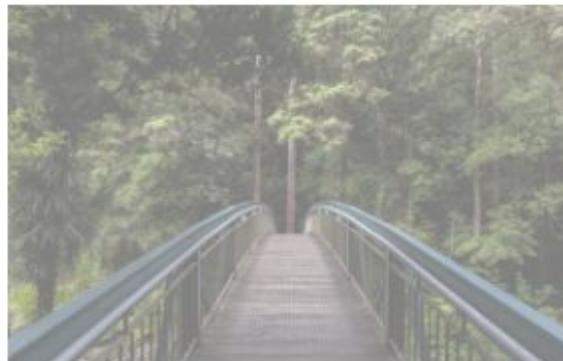
The `opacity` property specifies the opacity/transparency of an element.

Transparent Image

The `opacity` property can take a value from 0.0 - 1.0. The lower the value, the more transparent:



opacity 0.2



opacity 0.5



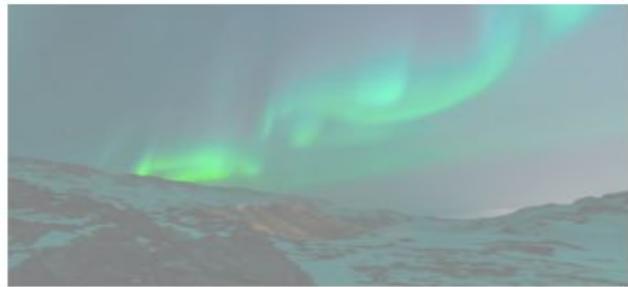
opacity 1
(default)

Example

```
img {  
  opacity: 0.5;  
}
```

Transparent Hover Effect

The `opacity` property is often used together with the `:hover` selector to change the opacity on mouse-over:



Example

```
img {  
  opacity: 0.5;  
}  
  
img:hover {  
  opacity: 1.0;  
}
```

Example explained

The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hovers over one of the images. In this case we want the image to NOT be transparent when the user hovers over it. The CSS for this is `opacity:1;`.

When the mouse pointer moves away from the image, the image will be transparent again.

An example of reversed hover effect:

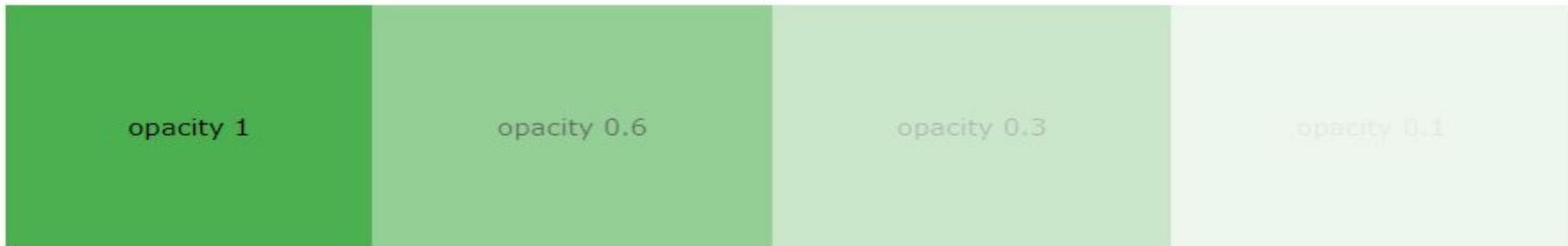


Example

```
img:hover {  
    opacity: 0.5;  
}
```

Transparent Box

When using the `opacity` property to add transparency to the background of an element, all of its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read:



Example

```
div {  
  opacity: 0.3;  
}
```

Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:



You learned from our [CSS Colors Chapter](#), that you can use RGB as a color value. In addition to RGB, you can use an RGB color value with an alpha channel (RGBA) - which specifies the opacity for a color.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Tip: You will learn more about RGBA Colors in our [CSS Colors Chapter](#).

Example

```
div {  
  background: rgba(76, 175, 80, 0.3) /* Green background with 30% opacity */  
}
```

Text in Transparent Box



This is some text that is placed in the transparent box.

Example

```
<html>
<head>
<style>
div.background {
    background: url(klematis.jpg) repeat;
    border: 2px solid black;
}

div.transbox {
    margin: 30px;
    background-color: #ffffff;
    border: 1px solid black;
    opacity: 0.6;
}

div.transbox p {
    margin: 5%;
    font-weight: bold;
    color: #000000;
}
</style>
</head>
<body>
```

```
<div class="background">
  <div class="transbox">
    <p>This is some text that is placed in the transparent box.</p>
  </div>
</div>

</body>
</html>
```

Example explained

First, we create a `<div>` element (`class="background"`) with a background image, and a border.

Then we create another `<div>` (`class="transbox"`) inside the first `<div>`.

The `<div class="transbox">` have a background color, and a border - the div is transparent.

Inside the transparent `<div>`, we add some text inside a `<p>` element.

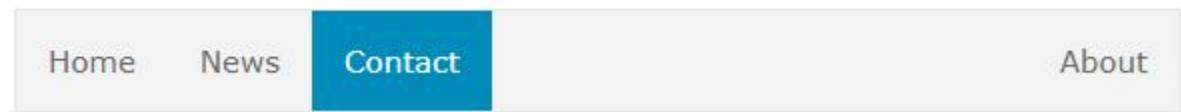
CSS Navigation Bar : Section 24.

Demo: Navigation Bars

Vertical



Horizontal



Navigation Bars

Having easy-to-use navigation is important for any web site.

With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the `` and `` elements makes perfect sense:

Example

```
<ul>
    <li><a href="default.asp">Home</a></li>
    <li><a href="news.asp">News</a></li>
    <li><a href="contact.asp">Contact</a></li>
    <li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

Example explained:

- `list-style-type: none;` - Removes the bullets. A navigation bar does not need list markers
- Set `margin: 0;` and `padding: 0;` to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars, which you will learn more about in the next chapters.

CSS Vertical Navigation Bar Section 25.

Vertical Navigation Bar

Home

News

Contact

About

To build a vertical navigation bar, you can style the `<a>` elements inside the list, in addition to the code from the previous page:

Example

```
li a {  
    display: block;  
    width: 60px;  
}
```

Example explained:

- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width (and padding, margin, height, etc. if you want)
- `width: 60px;` - Block elements take up the full width available by default. We want to specify a 60 pixels width

You can also set the width of ``, and remove the width of `<a>`, as they will take up the full width available when displayed as block elements. This will produce the same result as our previous example:

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    width: 60px;  
}  
  
li a {  
    display: block;  
}
```

Vertical Navigation Bar Examples

Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    width: 200px;  
    background-color: #f1f1f1;  
}  
  
li a {  
    display: block;  
    color: #000;  
    padding: 8px 16px;  
    text-decoration: none;  
}  
  
/* Change the link color on hover */  
li a:hover {  
    background-color: #555;  
    color: white;  
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:



Example

```
.active {  
    background-color: #04AA6D;  
    color: white;  
}
```

Center Links & Add Borders

Add `text-align:center` to `` or `<a>` to center the links.

Add the `border` property to `` add a border around the navbar. If you also want borders inside the navbar, add a `border-bottom` to all `` elements, except for the last one:

Home
News
Contact
About

Example

```
ul {  
    border: 1px solid #555;  
}  
  
li {  
    text-align: center;  
    border-bottom: 1px solid #555;  
}  
  
li:last-child {  
    border-bottom: none;  
}
```

Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

The screenshot shows a web page with a fixed vertical sidebar on the left and a main content area on the right.

Sidebar (Left):

- Home (highlighted in green)
- News
- Contact
- About

Main Content Area (Right):

Fixed Full-height Side Nav

Try to scroll this area, and see how the sidenav sticks to the page

Some text..

Some text..

Some text..

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    width: 25%;  
    background-color: #f1f1f1;  
    height: 100%; /* Full height */  
    position: fixed; /* Make it stick, even on scroll */  
    overflow: auto; /* Enable scrolling if the sidenav has too much content */  
}
```

CSS Horizontal Navigation Bar Section 26.

Horizontal Navigation Bar

Home News Contact

About

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the elements as **inline**, in addition to the "standard" code from the previous page:

Example

```
li {  
    display: inline;  
}
```

Example explained:

- `display: inline;` - By default, `` elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

Another way of creating a horizontal navigation bar is to float the `` elements, and specify a layout for the navigation links:

Example

```
li {  
    float: left;  
}  
  
a {  
    display: block;  
    padding: 8px;  
    background-color: #dddddd;  
}
```

Example explained:

- `float: left;` - Use float to get block elements to float next to each other
- `display: block;` - Allows us to specify padding (and height, width, margins, etc. if you want)
- `padding: 8px;` - Specify some padding between each `<a>` element, to make them look good
- `background-color: #dddddd;` - Add a gray background-color to each `<a>` element

Tip: Add the background-color to `` instead of each `<a>` element if you want a full-width background color:

Example

```
ul {  
    background-color: #dddddd;  
}
```

Horizontal Navigation Bar Examples

Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

```
Home News Contact About
```

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    background-color: #333;  
}  
br/>  
li {  
    float: left;  
}  
br/>  
li a {  
    display: block;  
    width: 120px;  
    height: 30px;  
    background-color: inherit;  
    color: inherit;  
    text-decoration: none;  
    text-align: center;  
    line-height: 30px;  
}  
br/>  
li a:hover {  
    background-color: #ccc;  
    color: black;  
}
```

```
li {  
    float: left;  
}  
  
li a {  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}  
  
/* Change the link color to #111 (black) on hover */  
li a:hover {  
    background-color: #111;  
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

Home News Contact About

Example

```
.active {  
    background-color: #04AA6D;  
}
```

Right-Align Links

Right-align links by floating the list items to the right (`float:right;`):

Home News Contact

About

Example

```
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="#about">About</a></li>
</ul>
```

Border Dividers

Add the `border-right` property to to create link dividers:



Example

```
/* Add a gray right border to all list items, except the last item (last-child) */
li {
    border-right: 1px solid #bbb;
}

li:last-child {
    border-right: none;
}
```

Fixed Navigation Bar

Make the navigation bar stay at the top or the bottom of the page, even when the user scrolls the page:

A screenshot of a web page featuring a fixed top navigation bar. The bar is dark grey and contains three items: "Home", "News", and "Contact". Below the bar, the main content area has a teal background. In the center of this area, the text "Fixed Top Nav" is displayed in large, bold, black font. Below it, another text block reads: "The navigation bar will stay at the top of the page while scrolling". A vertical scrollbar is visible on the right side of the content area.

Fixed Bottom Nav

The navigation bar will stay at the bottom of the page while scrolling

A screenshot of a web page featuring a fixed bottom navigation bar. The bar is dark grey and contains three items: "Home", "News", and "Contact". Above the bar, the main content area has a teal background. In the center of this area, the text "Fixed Bottom Nav" is displayed in large, bold, black font. Below it, another text block reads: "The navigation bar will stay at the bottom of the page while scrolling". A vertical scrollbar is visible on the right side of the content area.

Fixed Top

```
ul {  
  position: fixed;  
  top: 0;  
  width: 100%;  
}
```

Fixed Bottom

```
ul {  
  position: fixed;  
  bottom: 0;  
  width: 100%;  
}
```

Note: Fixed position might not work properly on mobile devices.

Gray Horizontal Navbar

An example of a gray horizontal navigation bar with a thin gray border:

Home

News

Contact

About

Example

```
ul {  
    border: 1px solid #e7e7e7;  
    background-color: #f3f3f3;  
}  
  
li a {  
    color: #666;  
}
```

Sticky Navbar

Add `position: sticky;` to `` to create a sticky navbar.

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`).

Scroll Down

Scroll down to see the sticky effect.

Home

News

Contact

Example

```
ul {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
}
```

Note: Internet Explorer do not support sticky positioning. Safari requires a -webkit- prefix (see example above). You must also specify at least one of `top`, `right`, `bottom` or `left` for sticky positioning to work.

Responsive Sidenav

How to use CSS media queries to create a responsive side navigation.

The image displays three devices illustrating a responsive sidebar navigation example. On the left is a desktop monitor showing a wide sidebar with 'Home', 'News', 'Contact', and 'About' links. In the center is a tablet device showing a top navigation bar with 'Home', 'News', 'Contact', and 'About' links. On the right is a smartphone showing a top navigation bar with 'Home', 'News', 'Contact', and 'About' links. All devices display the same content area with a heading 'Responsive Sidenav Example' and descriptive text about media queries.

Responsive Sidenav Example

This example uses media queries to transform the sidebar to a top navigation bar when the screen size is 600px or less. We have also added a media query for screens that are 400px or less, which will vertically stack and center the navigation links. You will learn more about media queries and responsive web design later in our CSS Tutorial. Resize the browser window to see the effect.

Responsive Sidenav Example

This example uses media queries to transform the sidebar to a top navigation bar when the screen size is 600px or less. We have also added a media query for screens that are 400px or less, which will vertically stack and center the navigation links. You will learn more about media queries and responsive web design later in our CSS Tutorial. Resize the browser window to see the effect.

Responsive Sidenav Example

This example uses media queries to transform the sidebar to a top navigation bar when the screen size is 600px or less. We have also added a media query for screens that are 400px or less, which will vertically stack and center the navigation links. You will learn more about media queries and responsive web design later in our CSS Tutorial. Resize the browser window to see the effect.

CSS Dropdowns Section 27

Create a hoverable dropdown with CSS.

Demo: Dropdown Examples

Move the mouse over the examples below:

Dropdown Text

Dropdown Menu

Other:



Basic Dropdown

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

Example

```
<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>  
  
<div class="dropdown">  
    <span>Mouse over me</span>  
    <div class="dropdown-content">  
        <p>Hello World!</p>  
    </div>  
</div>
```

Example Explained

HTML) Use any element to open the dropdown content, e.g. a , or a <button> element.

Use a container element (like <div>) to create the dropdown content and add whatever you want inside of it.

Wrap a <div> element around the elements to position the dropdown content correctly with CSS.

CSS) The `.dropdown` class uses `position:relative`, which is needed when we want the dropdown content to be placed right below the dropdown button (using `position:absolute`).

The `.dropdown-content` class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the `min-width` is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the `width` to 100% (and `overflow:auto` to enable scroll on small screens).

Instead of using a border, we have used the CSS `box-shadow` property to make the dropdown menu look like a "card".

The `:hover` selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu

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

Dropdown Menu

This example is similar to the previous one, except that we add links inside the dropdown box and style them to fit a styled dropdown button:

Example

```
<style>
/* Style The Dropdown Button */
.dropbtn {
    background-color: #4CAF50;
    color: white;
    padding: 16px;
    font-size: 16px;
    border: none;
    cursor: pointer;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
    position: relative;
    display: inline-block;
}
```

Example

```
<style>
/* Style The Dropdown Button */
.dropbtn {
    background-color: #4CAF50;
    color: white;
    padding: 16px;
    font-size: 16px;
    border: none;
    cursor: pointer;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
    position: relative;
    display: inline-block;
}
```

```
/* Change color of dropdown links on hover */
.dropdown-content a:hover {background-color: #f1f1f1}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
    display: block;
}

/* Change the background color of the dropdown button when the dropdown content is shown */
.dropdown:hover .dropbtn {
    background-color: #3e8e41;
}
</style>
```

```
<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>
```

```
/* Dropdown Content (Hidden by Default) */
.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;
}
```

```
/* Links inside the dropdown */
.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}
```

```
/* Change the background color of the dropdown button when the dropdown content is shown */
.dropdown:hover .dropbtn {
  background-color: #3e8e41;
}
</style>

<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>
```

Right-aligned Dropdown Content

Left

Right

If you want the dropdown menu to go from right to left, instead of left to right, add `right: 0;`

Example

```
.dropdown-content {  
  right: 0;  
}
```

CSS Image Gallery. Section 28

CSS can be used to create an image gallery.



Add a description of
the image here



Add a description of
the image here



Add a description of
the image here



Add a description of
the image here

Image Gallery

The following image gallery is created with CSS:

Example

```
<html>
<head>
<style>
div.gallery {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 180px;
}

div.gallery:hover {
    border: 1px solid #777;
}

div.gallery img {
    width: 100%;
    height: auto;
}
```

```
div.desc {  
    padding: 15px;  
    text-align: center;  
}  
</style>  
</head>  
<body>  
  
<div class="gallery">  
    <a target="_blank" href="img_5terre.jpg">  
          
    </a>  
    <div class="desc">Add a description of the image here</div>  
</div>  
  
<div class="gallery">  
    <a target="_blank" href="img_forest.jpg">  
          
    </a>  
    <div class="desc">Add a description of the image here</div>  
</div>
```

```
<div class="gallery">
  <a target="_blank" href="img_lights.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_mountains.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

CSS Forms Section 29.

The look of an HTML form can be greatly improved with CSS:

First Name

Your name..

Last Name

Your last name..

Country

Australia



Styling Input Fields

Use the `width` property to determine the width of the input field:

First Name

Example

```
input {  
  width: 100%;  
}
```

The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:

- `input[type=text]` - will only select text fields
- `input[type=password]` - will only select password fields
- `input[type=number]` - will only select number fields
- etc..

Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"] {  
    width: 150px;  
    display: block;  
    margin-bottom: 10px;  
    background-color: yellow;  
}  
  
input[type="button"] {  
    width: 120px;  
    margin-left: 35px;  
    display: block;  
}
```

Padded Inputs

Use the `padding` property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some `margin`, to add more space outside of them:

First Name

Last Name

Example

```
input[type=text] {  
    width: 100%;  
    padding: 12px 20px;  
    margin: 8px 0;  
    box-sizing: border-box;  
}
```

Activat
Go to Se

Note that we have set the `box-sizing` property to `border-box`. This makes sure that the padding and eventually borders are included in the total width and height of the elements.

Read more about the `box-sizing` property in our [CSS Box Sizing](#) chapter.

Bordered Inputs

Use the `border` property to change the border size and color, and use the `border-radius` property to add rounded corners:

First Name

Example

```
input[type=text] {  
    border: 2px solid red;  
    border-radius: 4px;  
}
```

Activat

If you only want a bottom border, use the `border-bottom` property:

First Name

Example

```
input[type=text] {  
    border: none;  
    border-bottom: 2px solid red;  
}
```

Colored Inputs

Use the `background-color` property to add a background color to the input, and the `color` property to change the text color:

John

Example

```
input[type=text] {  
    background-color: #3CBC8D;  
    color: white;  
}
```

Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding `outline: none;` to the input.

Use the `:focus` selector to do something with the input field when it gets focus:

Example

```
input[type=text]:focus {  
    background-color: lightblue;  
}
```

Example

```
input[type=text]:focus {  
    border: 3px solid #555;  
}
```

Input with icon/image

If you want an icon inside the input, use the `background-image` property and position it with the `background-position` property. Also notice that we add a large left padding to reserve the space of the icon:

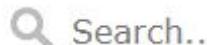


Example

```
input[type=text] {  
  background-color: white;  
  background-image: url('searchicon.png');  
  background-position: 10px 10px;  
  background-repeat: no-repeat;  
  padding-left: 40px;  
}
```

Animated Search Input

In this example we use the CSS `transition` property to animate the width of the search input when it gets focus. You will learn more about the `transition` property later, in our [CSS Transitions](#) chapter.



Example

```
input[type=text] {  
    transition: width 0.4s ease-in-out;  
}  
  
input[type=text]:focus {  
    width: 100%;  
}
```

Styling Textareas

Tip: Use the `resize` property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

```
Some text...
```

Example

```
textarea {  
    width: 100%;  
    height: 150px;  
    padding: 12px 20px;  
    box-sizing: border-box;  
    border: 2px solid #ccc;  
    border-radius: 4px;  
    background-color: #f8f8f8;  
    resize: none;  
}
```

Styling Select Menus

Australia

Example

```
select {  
    width: 100%;  
    padding: 16px 20px;  
    border: none;  
    border-radius: 4px;  
    background-color: #f1f1f1;  
}
```

Styling Input Buttons

Button

Button

Example

```
input[type=button], input[type=submit], input[type=reset] {  
    background-color: #04AA6D;  
    border: none;  
    color: white;  
    padding: 16px 32px;  
    text-decoration: none;  
    margin: 4px 2px;  
    cursor: pointer;  
}  
  
/* Tip: use width: 100% for full-width buttons */
```

For more information about how to style buttons with CSS, read our [CSS Buttons Tutorial](#).

Responsive Form

Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.

Advanced: The following example uses [media queries](#) to create a responsive form. You will learn more about this in a later chapter.

First Name

Your name..

Last Name

Your last name..

Country

Australia

Subject

Write something..

Submit

CSS Counters Section 30

1 Pizza

2 Hamburger

3 Hotdogs

CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.

Automatic Numbering With Counters

CSS counters are like "variables". The variable values can be incremented by CSS rules (which will track how many times they are used).

To work with CSS counters we will use the following properties:

- `counter-reset` - Creates or resets a counter
- `counter-increment` - Increments a counter value
- `content` - Inserts generated content
- `counter()` or `counters()` function - Adds the value of a counter to an element

To use a CSS counter, it must first be created with `counter-reset`.

The following example creates a counter for the page (in the body selector), then increments the counter value for each `<h2>` element and adds "Section <value of the counter>:" to the beginning of each `<h2>` element:

Example

```
body {  
    counter-reset: section;  
}  
  
h2::before {  
    counter-increment: section;  
    content: "Section " counter(section) ":";  
}
```

Nesting Counters

The following example creates one counter for the page (section) and one counter for each `<h1>` element (subsection). The "section" counter will be counted for each `<h1>` element with "Section *<value of the section counter>*.", and the "subsection" counter will be counted for each `<h2>` element with "*<value of the section counter>. <value of the subsection counter>*":

Example

```
body {  
  counter-reset: section;  
}  
  
h1 {  
  counter-reset: subsection;  
}  
  
h1::before {  
  counter-increment: section;  
  content: "Section " counter(section) ". ";  
}  
  
h2::before {  
  counter-increment: subsection;  
  content: counter(section) "." counter(subsection) " ";  
}
```

A counter can also be useful to make outlined lists because a new instance of a counter is automatically created in child elements. Here we use the `counters()` function to insert a string between different levels of nested counters:

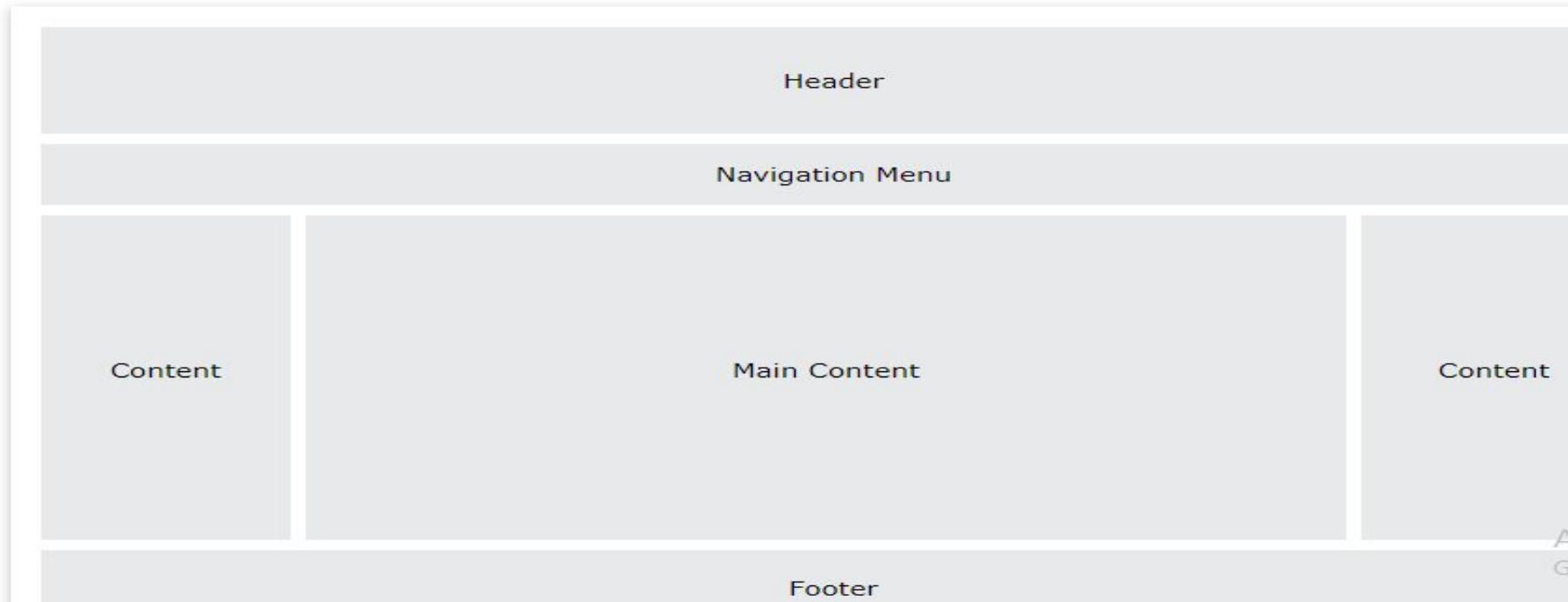
Example

```
ol {  
  counter-reset: section;  
  list-style-type: none;  
}  
  
li::before {  
  counter-increment: section;  
  content: counters(section,".") " ";  
}
```

CSS Website Layout. Section 31.

Website Layout

A website is often divided into headers, menus, content and a footer:



There are tons of different layout designs to choose from. However, the structure above, is one of the most common, and we will take a closer look at it in this tutorial.

Header

A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name:

Example

```
.header {  
    background-color: #F1F1F1;  
    text-align: center;  
    padding: 20px;  
}
```

Navigation Bar

A navigation bar contains a list of links to help visitors navigating through your website:

Example

```
/* The navbar container */
.topnav {
  overflow: hidden;
  background-color: #333;
}

/* Navbar links */
.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
```

```
/* Links - change color on hover */
.topnav a:hover {
    background-color: #ddd;
    color: black;
}
```

Result

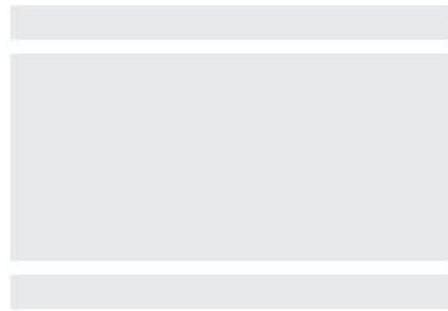
Link Link Link

Content

The layout in this section, often depends on the target users. The most common layout is one (or combining them) of the following:

- **1-column** (often used for mobile browsers)
- **2-column** (often used for tablets and laptops)
- **3-column layout** (only used for desktops)

1-column:



2-column:



3-column:



We will create a 3-column layout, and change it to a 1-column layout on smaller screens:

Example

```
/* Create three equal columns that float next to each other */
.column {
  float: left;
  width: 33.33%;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other instead of next to each
other on smaller screens (600px wide or less) */
@media screen and (max-width: 600px) {
  .column {
    width: 100%;
  }
}
```

Result

Column

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Maecenas sit amet pretium
urna. Vivamus venenatis velit
nec neque ultricies, eget
elementum magna tristique.

Column

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Maecenas sit amet pretium
urna. Vivamus venenatis velit
nec neque ultricies, eget
elementum magna tristique.

Column

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Maecenas sit amet pretium
urna. Vivamus venenatis velit
nec neque ultricies, eget
elementum magna tristique.

Tip: To create a 2-column layout, change the width to 50%. To create a 4-column layout, use 25%, etc.

Tip: Do you wonder how the @media rule works? [Read more about it in our CSS Media Queries chapter](#).

Tip: A more modern way of creating column layouts, is to use CSS Flexbox. However, it is not supported in Internet Explorer 10 and earlier versions. If you require IE6-10 support, use floats (as shown above).

To learn more about the Flexible Box Layout Module, [read our CSS Flexbox chapter](#).

Unequal Columns

The main content is the biggest and the most important part of your site.

It is common with **unequal** column widths, so that most of the space is reserved for the main content. The side content (if any) is often used as an alternative navigation or to specify information relevant to the main content. Change the widths as you like, only remember that it should add up to 100% in total:

Example

```
.column {  
  float: left;  
}  
  
/* Left and right column */  
.column.side {  
  width: 25%;  
}  
  
/* Middle column */  
.column.middle {  
  width: 50%;  
}  
  
/* Responsive layout - makes the three columns stack on top of each other instead of next to each  
other */  
@media screen and (max-width: 600px) {  
  .column.side, .column.middle {  
    width: 100%;  
  }  
}
```

Result

Side

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Main Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.

Side

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Footer

The footer is placed at the bottom of your page. It often contains information like copyright and contact info:

Example

```
.footer {  
    background-color: #F1F1F1;  
    text-align: center;  
    padding: 10px;  
}
```

Result

Footer

Responsive Website Layout

By using some of the CSS code above, we have created a responsive website layout, which varies between two columns and full-width columns depending on screen width:

My Website

Resize the browser window to see the effect.

Link

Link

Link

Link

TITLE HEADING

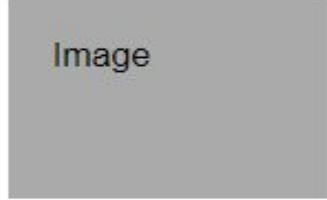
Title description, Dec 7, 2017

Image



About Me

Image



Some text about me
in culpa qui officia
deserunt mollit
anim..

CSS The !important Rule. Section 32.

What is !important?

The `!important` rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the `!important` rule, it will override ALL previous styling rules for that specific property on that element!

Let us look at an example:

Example

```
#myid {  
    background-color: blue;  
}  
  
.myclass {  
    background-color: gray;  
}  
  
p {  
    background-color: red !important;  
}
```

Example Explained

In the example above, all three paragraphs will get a red background color, even though the ID selector and the class selector have a higher specificity. The `!important` rule overrides the `background-color` property in both cases.

Important About !important

The only way to override an `!important` rule is to include another `!important` rule on a declaration with the same (or higher) specificity in the source code - and here the problem starts! This makes the CSS code confusing and the debugging will be hard, especially if you have a large style sheet!

Here we have created a simple example. It is not very clear, when you look at the CSS source code, which color is considered most important:

Example

```
#myid {  
    background-color: blue !important;  
}  
  
.myclass {  
    background-color: gray !important;  
}  
  
p {  
    background-color: red !important;  
}
```

Maybe One or Two Fair Uses of !important

One way to use `!important` is if you have to override a style that cannot be overridden in any other way. This could be if you are working on a Content Management System (CMS) and cannot edit the CSS code. Then you can set some custom styles to override some of the CMS styles.

Another way to use `!important` is: Assume you want a special look for all buttons on a page. Here, buttons are styled with a gray background color, white text, and some padding and border:

Example

```
.button {  
background-color: #8c8c8c;  
color: white;  
padding: 5px;  
border: 1px solid black;  
}
```

The look of a button can sometimes change if we put it inside another element with higher specificity, and the properties get in conflict. Here is an example of this:

Example

```
.button {  
    background-color: #8c8c8c;  
    color: white;  
    padding: 5px;  
    border: 1px solid black;  
}  
  
#myDiv a {  
    color: red;  
    background-color: yellow;  
}
```

To "force" all buttons to have the same look, no matter what, we can add the `!important` rule to the properties of the button, like this:

Example

```
.button {  
    background-color: #8c8c8c !important;  
    color: white !important;  
    padding: 5px !important;  
    border: 1px solid black !important;  
}
```

```
#myDiv a {  
    color: red;  
    background-color: yellow;  
}
```