What is CSS?

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

CSS Demo - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" links below to see the different styles:

Why Use CSS?

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

CSS Solved a Big Problem

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

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

<h1>This is a heading</h1>

This is a paragraph.

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

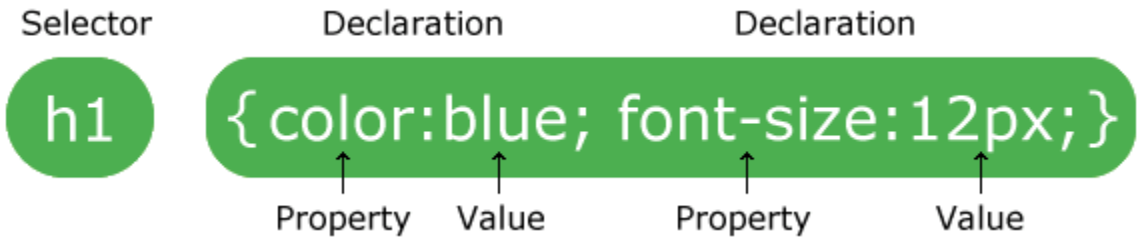
To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

CSS Syntax and Selectors

CSS Syntax

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



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

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

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

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

Example

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

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

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

Example

You can select all <p> elements on a page like this (here, all <p> elements will be center-aligned, with a red text color):

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

The id Selector

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

The id of an element should be 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 style rule below will be applied to the HTML element with id="para1":

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

The class Selector

The class selector selects elements with a specific class attribute.

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

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 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!

Grouping Selectors

If you have elements with the same style definitions, like this:

```
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;
}
```

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.

Example

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

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

/* This is
a multi-line
comment */
```

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

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

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

Example

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

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

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

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

"mystyle.css"

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

h1 {
  color: navy;
  margin-left: 20px;
}
```

Note: Do not add a space between the property value and the unit (such as `margin-left: 20 px;`). The correct way is: `margin-left: 20px;`

Internal Style Sheet

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

Example

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

```
<head>
<style>
body {
  background-color: linen;
}
```

```
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
```

Inline Styles

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:

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

Tip: An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method sparingly.

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>
```

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

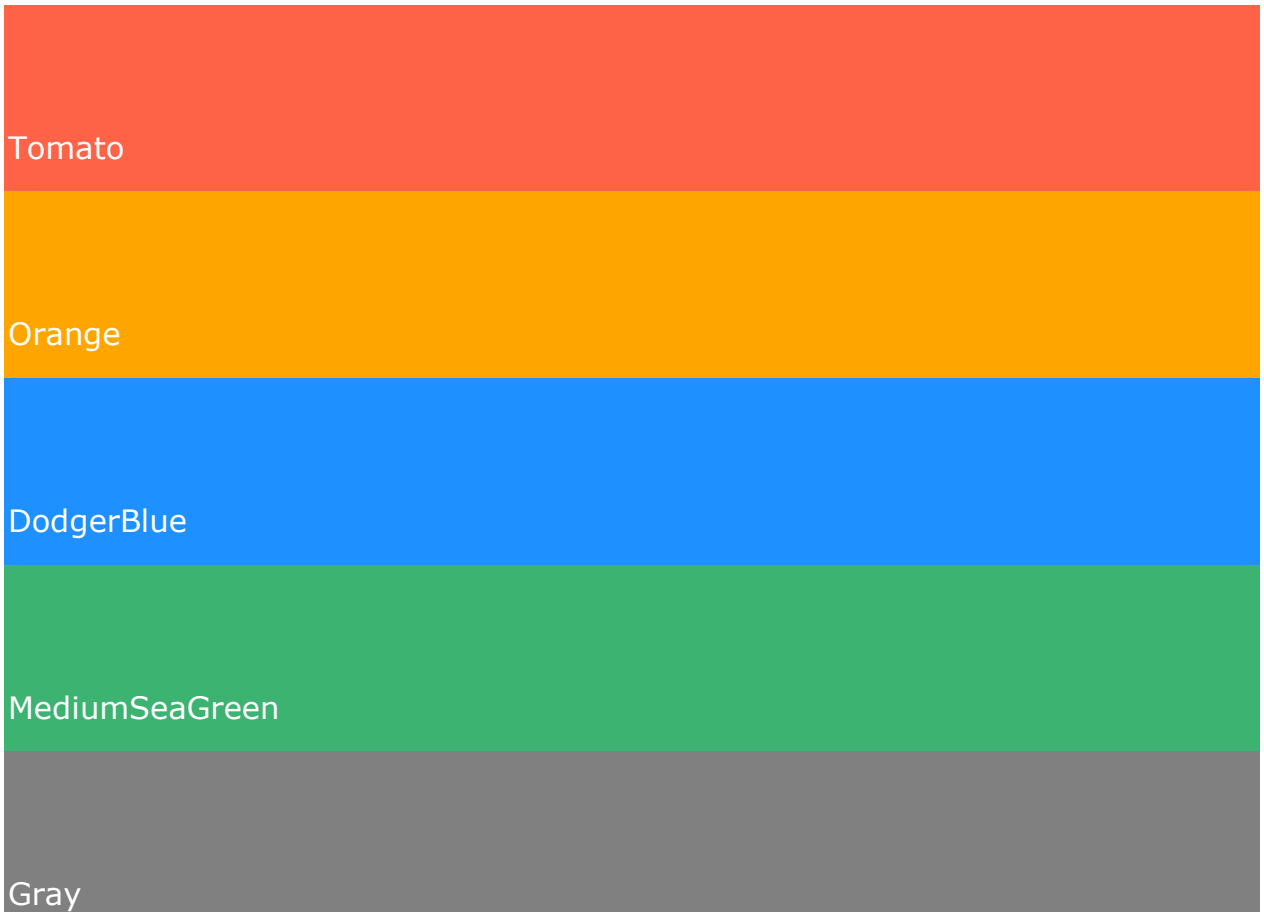
So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Colors

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

Color Names

In HTML, a color can be specified by using a color name:



SlateBlue

Violet

LightGray

HTML supports [140 standard color names](#).

Background Color

You can set the background color for HTML elements:

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

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>
```

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>
```

Color Values

In HTML, 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:

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

Try it Yourself »

RGB Value

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0).

To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:

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

RED

255

GREEN

99

BLUE

71

Example

```
rgb(255, 0, 0)
```

```
rgb(0, 0, 255)
rgb(60, 179, 113)
rgb(238, 130, 238)
rgb(255, 165, 0)
rgb(106, 90, 205)
```

Try it Yourself »

Shades of gray are often defined using equal values for all the 3 light sources:

Example

```
rgb(0, 0, 0)
rgb(60, 60, 60)
```

```
rgb(120, 120, 120)
```

```
rgb(180, 180, 180)
```

```
rgb(240, 240, 240)
```

```
rgb(255, 255, 255)
```

Try it Yourself »

HEX Value

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

Example

```
#ff0000
```

#0000ff

#3cb371

#ee82ee

#ffa500

#6a5acd

Try it Yourself »

Shades of gray are often defined using equal values for all the 3 light sources:

Example

#000000

#3c3c3c

#787878

#b4b4b4

#f0f0f0

#ffffff

Try it Yourself »

HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

Example

hsl(0, 100%, 50%)

hsl(240, 100%, 50%)

hsl(147, 50%, 47%)

hsl(300, 76%, 72%)

hsl(39, 100%, 50%)

hsl(248, 53%, 58%)

Try it Yourself »

Saturation

Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

Example

```
hsl(0, 100%, 50%)
hsl(0, 80%, 50%)
hsl(0, 60%, 50%)
hsl(0, 40%, 50%)
hsl(0, 20%, 50%)
hsl(0, 0%, 50%)
```

Try it Yourself »

Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Example

```
hsl(0, 100%, 0%)

hsl(0, 100%, 25%)

hsl(0, 100%, 50%)

hsl(0, 100%, 75%)

hsl(0, 100%, 90%)

hsl(0, 100%, 100%)
```

Try it Yourself »

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

Example

```
hsl(0, 0%, 0%)

hsl(0, 0%, 24%)

hsl(0, 0%, 47%)

hsl(0, 0%, 71%)

hsl(0, 0%, 94%)

hsl(0, 0%, 100%)
```

Try it Yourself »

RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - 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 (not transparent at all):

Example

```
rgba(255, 99, 71, 0)
```

```
rgba(255, 99, 71, 0.2)
```

```
rgba(255, 99, 71, 1)
```

Try it Yourself »

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue, saturation, lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

```
hsla(9, 100%, 64%, 0)
```

```
hsla(9, 100%, 64%, 0.2)
```

```
hsla(9, 100%, 64%, 1)
```

CSS Backgrounds

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

CSS background properties:

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

Background Color

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

Example

The background color of a page is set like this:

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

Try it Yourself »

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.

Example

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

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

Try it Yourself »

Background Image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

The background image for a page can be set like this:

```
body {
  background-image: url("paper.gif");
}
```

Try it Yourself »

Example

This example shows a **bad combination** of text and background image. The text is hardly readable:

```
body {
  background-image: url("bgdesert.jpg");
}
```

Try it Yourself »

Note: When using a background image, use an image that does not disturb the text.

Background Image - Repeat Horizontally or Vertically

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

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

Example

```
body {
  background-image: url("gradient_bg.png");
}
```

Try it Yourself »

If the image above is repeated only horizontally (`background-repeat: repeat-x;`), the background will look better:

Example

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Try it Yourself »

Tip: To repeat an image vertically, set `background-repeat: repeat-y;`

Background Image - no-repeat

Showing the background image only once is also specified by the `background-repeat` property:

Example

Show the background image only once:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

Try it Yourself »

In the example above, the background image is placed in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

Background Image - Set position

The `background-position` property is used to specify the position of the background image.

Example

Position the background image in the top-right corner:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

Try it Yourself »

Background Image - Fixed or Scroll?

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

Example

Specify that the background image should be fixed:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Try it Yourself »

Example

Specify that the background image should scroll with the rest of the page:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

Try it Yourself »

Background - Shorthand property

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

The shorthand property for background is `background`.

Example

Use the shorthand property to set all the background properties in one declaration:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

Try it Yourself »

When using the shorthand property the order of the property values is:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`

It does not matter if one of the property values is missing, as long as the other ones are in this order.

Test Yourself with Exercises!

Exercise 1 »Exercise 2 »Exercise 3 »Exercise 4 »Exercise 5 »

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with th
background-clip	Specifies the painting area of the background
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-origin	Specifies where the background image(s) is/are positioned

background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated
background-size	Specifies the size of the background image(s)

CSS Borders

CSS Border Properties

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.

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 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.

Try it Yourself »

Note: None of the OTHER CSS border properties described below will have ANY effect unless the `border-style`property is set!

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.

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

5px border-width

Example

```
p.one {
  border-style: solid;
```

```
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: solid;
  border-width: 2px 10px 4px 20px;
}
```

Try it Yourself »

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)"
- transparent

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

If `border-color` is not set, it inherits the color of the element.

Red border

Example

```
p.one {
  border-style: solid;
  border-color: red;
}

p.two {
```

```
  border-style: solid;
  border-color: green;
}

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

Try it Yourself »

Border - Individual Sides

From the examples above 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):

Different Border Styles

Example

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

Try it Yourself »

The example above gives the same result as this:

Example

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

Try it Yourself »

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

The `border-style` property is used in the example above. However, it also works with `border-width` and `border-color`.

Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

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

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

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

Example

```
p {
  border: 5px solid red;
}
```

Result:

Some text

Try it Yourself »

You can also specify all the individual border properties for just one side:

Left Border

```
p {
  border-left: 6px solid red;
  background-color: lightgrey;
}
```

Result:

Some text

Try it Yourself »

Bottom Border

```
p {
  border-bottom: 6px solid red;
  background-color: lightgrey;
}
```

Result:

Some text

Try it Yourself »

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;
}
```

Try it Yourself »

Note: The `border-radius` property is not supported in IE8 and earlier versions.

More Examples

All the top border properties in one declaration
This example demonstrates a shorthand property for setting all of the properties for the top border in one declaration.

Set the style of the bottom border
This example demonstrates how to set the style of the bottom border.

Set the width of the left border
This example demonstrates how to set the width of the left border.

Set the color of the four borders
This example demonstrates how to set the color of the four borders. It can have from one to four colors.

Set the color of the right border
This example demonstrates how to set the color of the right border.

All CSS Border Properties

Property	Description
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-left	Sets all the left border properties in one declaration

border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	Sets all the four border-*-radius properties for rounded corners
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border

border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders

CSS Margins

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

If the `margin` property has one value:

- **margin: 25px;**
  - all four margins are 25px

Example

Use the margin shorthand property with one value:

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

Try it Yourself »

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;
}
```

Try it Yourself »

The inherit Value

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

Example

Use of the inherit value:

```
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
}
```

Try it Yourself »

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;
}
```

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 element:

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

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

If the `padding` property has one value:

- **padding: 25px;**
  - all four paddings are 25px

Example

Use the padding shorthand property with one value:

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

Try it Yourself »

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

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

Try it Yourself »

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;
}
```

Try it Yourself »

More Examples

Set the left padding
This example demonstrates how to set the left padding of a <p> element.

Set the right padding
This example demonstrates how to set the right padding of a <p> element.

Set the top padding
This example demonstrates how to set the top padding of a <p> element.

Set the bottom padding
This example demonstrates how to set the bottom padding of a <p> element.

Test Yourself with Exercises!

Exercise 1 »Exercise 2 »Exercise 3 »

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element

padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

CSS Height and Width

Setting height and width

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

The `height` and `width` can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

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

Example

Set the height and width of a element:

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

Try it Yourself »

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

Example

Set the height and width of another element:

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

Try it Yourself »

Note: 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: The value of the `max-width` property overrides `width`.

Example

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

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

Try it Yourself »

Try it Yourself - Examples

Set the height and width of elements
This example demonstrates how to set the height and width of different elements.

Set the height and width of an image using percent
This example demonstrates how to set the height and width of an image using a percent value.

Set min-width and max-width of an element
This example demonstrates how to set a minimum width and a maximum width of an element using a pixel value.

Set min-height and max-height of an element
This example demonstrates how to set a minimum height and a maximum height of an element using a pixel value.

All CSS Dimension Properties

Property	Description
height	Sets the height of an element

max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

CSS Box Model

The CSS Box Model

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

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

Explanation of the different parts:

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

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

Example

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 element will have a total width of 350px:

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

Try it Yourself »

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

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.

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:

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.

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;}
```

Note: None of the other outline properties will have any effect, unless the `outline-style` property is set!

Outline Color

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

The color can be set by:

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

The following example shows some different outlines with different colors. Also notice that these elements also have a thin black border inside the outline:

A solid red outline.

A double green outline.

An outset yellow outline.

Example

```
p.ex1 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
}

p.ex2 {
  border: 1px solid black;
  outline-style: double;
  outline-color: green;
}

p.ex3 {
  border: 1px solid black;
  outline-style: outset;
```

```
  outline-color: yellow;
}
```

The following example uses `outline-color: invert`, which performs a color inversion. This ensures that the outline is visible, regardless of color background:

A solid invert outline.

Example

```
p.ex1 {
  border: 1px solid yellow;
  outline-style: solid;
  outline-color: invert;
}
```

Outline Width

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

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

The following example shows some outlines with different widths:

A thin outline.

A medium outline.

A thick outline.

A 4px thick outline.

Example

```
p.ex1 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
```

```
  outline-width: thin;
}

p.ex2 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: medium;
}

p.ex3 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thick;
}

p.ex4 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: 4px;
}
```

Outline - Shorthand property

The `outline` property is a shorthand property for setting the following individual outline properties:

- `outline-width`
- `outline-style` (required)
- `outline-color`

The `outline` property is specified as one, two, or three values from the list above. The order of the values does not matter.

The following example shows some outlines specified with the shorthand `outline` property:

A dashed outline.

A dotted red outline.

A 5px solid yellow outline.

A thick ridge pink outline.

Example

```
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
```

Outline Offset

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

The following example specifies an outline 15px outside the border edge:

This paragraph has an outline 15px outside the border edge.

Example

```
p {
  margin: 30px;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
```

The following example shows that the space between an element and its outline is transparent:

This paragraph has an outline of 15px outside the border edge.

Example

```
p {
  margin: 30px;
  background: yellow;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
```

CSS Text

Text Color

The `color` property is used to set the color of the text. The color is specified by:

- a 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.

The default text color for a page is defined in the body selector.

Example

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

Note: For W3C compliant CSS: If you define the `color` property, you must also define the `background-color`.

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 Decoration

The `text-decoration` property is used to set or remove decorations from text.

The value `text-decoration: none;` is often used to remove underlines from links:

Example

```
a {
  text-decoration: none;
}
```

The other `text-decoration` values are used to decorate text:

Example

```
h1 {
  text-decoration: overline;
}

h2 {
  text-decoration: line-through;
}

h3 {
  text-decoration: underline;
}
```

Note: It is not recommended to underline text that is not a link, as this often confuses the reader.

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 Indentation

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

Example

```
p {
  text-indent: 50px;
}
```

Try it Yourself »

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: 3px;
}

h2 {
  letter-spacing: -3px;
}
```

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;
}
```

Text Direction

The `direction` property is used to change the text direction of an element:

Example

```
p {
  direction: rtl;
}
```

Word Spacing

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

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

Example

```
h1 {
  word-spacing: 10px;
}

h2 {
  word-spacing: -5px;
}
```

Text Shadow

The `text-shadow` property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

Example

```
h1 {
  text-shadow: 3px 2px red;
}
```

Try it Yourself »

More Examples

Disable text wrapping inside an element
This example demonstrates how to disable text wrapping inside an element.

Vertical alignment of an image
This example demonstrates how to set the vertical align of an image in a text.

Tip: Go to our CSS Fonts chapter to learn about how to change fonts, text size and the style of a text.

All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a t
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text

text-overflow	Specifies how overflowed content that is not displayed shou the user
unicode-bidi	Used together with the direction property to set or return w should be overridden to support multiple languages in the s
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

CSS Fonts

The CSS font properties define the font family, boldness, size, and the style of a text.

Difference Between Serif and Sans-serif Fonts

CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the e characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

Font Family

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

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

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

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

More than one font family is specified in a comma-separated list:

Example

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

For commonly used font combinations, look at our Web Safe Font Combinations.

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-style: oblique;
}
```

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

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

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;
}
```

Tip: If you use pixels, you can still use the zoom tool to resize the entire page.

Set Font Size With Em

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

The em size unit is recommended by the W3C.

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: *pixels*/16=*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 */
}
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

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;
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Font Weight

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

Example

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

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

Responsive Font Size

The text size can be set with a `vw` unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

Hello World

Resize the browser window to see how the font size scales.

Example

```
<h1 style="font-size:10vw">Hello World</h1>
```

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

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;
}
```

CSS Icons

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 `<span>`).

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

```
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.0/css/all.css"
integrity="sha384-
lZN37f5QGtY3VHgisS14W3ExzMWZxybE1SJSEsQp9S+oqd12jhcu+A56Ebc1zFSJ"
crossorigin="anonymous">
```

Note: No downloading or installation is required!

Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.0/css/all.css"integrity="sha384-lZN37f5QGtY3VHgisS14W3ExzMWZxybE1SJSEsQp9S+oqd12jhcu+A56Ebc1zFSJ"crossorigin="anonymous">
</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>
```

For a complete reference of all Font Awesome icons, visit our Icon Reference.

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!

Example

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

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=Materia
l+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>
```

CSS Links

With CSS, links can be styled in 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;
}
```

Try it Yourself »

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;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

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

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Advanced - 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;
}
```

CSS Lists

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 (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - 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;
}
```

Try it Yourself »

Note: Some of the values are for unordered lists, and some for ordered lists.

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');
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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 <ul> or <ol>:

Example

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

Try it Yourself »

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");
}
```

Try it Yourself »

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 <ol> or <ul> tag, affects the entire list, while properties added to the <li> 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

CSS Tables

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

Company	Contact
Alfreds Futterkiste	Maria Anders
Berglunds snabbköp	Christina Berglund
Centro comercial Moctezuma	Francisco Chang
Ernst Handel	Roland Mendel
Island Trading	Helen Bennett
Königlich Essen	Philip Cramer

Laughing Bacchus Winecellars	Yoshi Tannamuri
Magazzini Alimentari Riuniti	Giovanni Rovelli

Try it Yourself »

Table Borders

To specify table borders in CSS, use the `border` property.

The example below specifies a black border for <table>, <th>, and <td> elements:

Example

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

Try it Yourself »

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

Collapse Table Borders

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

Example

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

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

Try it Yourself »

If you only want a border around the table, only specify the `border` property for <table>:

Example

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

Try it Yourself »

Table Width and Height

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 50px:

Example

```
table {
  width: 100%;
}

th {
  height: 50px;
}
```

Try it Yourself »

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.

The following example left-aligns the text in <th> elements:

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;}
```

Try it Yourself »

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;}
```

Try it Yourself »

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: #4CAF50;
  color: white;
}
```

Try it Yourself »

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

First Name	Last Name	Points	Points	Points	Points	Points	Points	Points	Points	P
Jill	Smith	50	50	50	50	50	50	50	50	5

Eve	Jackson	94	94	94	94	94	94	94	94	9
Adam	Johnson	67	67	67	67	67	67	67	67	6

Add a container element (like) with `overflow-x:auto` around the <table> element to make it responsive:

Example

```
<div style="overflow-x:auto;">

<table>
... table content ...
</table>

</div>
```

CSS Layout - The display Property

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`.

Click to show panel

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 element is a block-level element.

Examples of block-level elements:

-
- <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>
- <img>

Display: none;

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

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

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 `<li>` elements for horizontal menus:

Example

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

Try it Yourself »

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with `display: block;` is not allowed to have other block elements inside it.

The following example displays elements as block elements:

Example

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

Try it Yourself »

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

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

`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;
}
```

Try it Yourself »

More Examples

Differences between display: none; and visibility: hidden;
This example demonstrates display: none; versus visibility: hidden;

Using CSS together with JavaScript to show content
This example demonstrates how to use CSS and JavaScript to show an element on click.

Test Yourself with Exercises!

Exercise 1 »Exercise 2 »Exercise 3 »Exercise 4 »

CSS Display/Visibility Properties

Property	Description
display	Specifies how an element should be displayed
visibility	Specifies whether or not an element should be visible

CSS Layout - width and max-width

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 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:

This element has a max-width of 500px, and margin set to auto.

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;
}
```

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

CSS Layout - The position Property

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 element has position: static;

Here is the CSS that is used:

Example

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

Try it Yourself »

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 element has position: relative;

Here is the CSS that is used:

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

This element has `position: fixed;`

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: A "positioned" element is one whose position is anything except `static`.

Here is a simple example:

This element has position: relative;

This 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;
}
```

Try it Yourself »

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).

Note: Internet Explorer, Edge 15 and earlier versions do 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;
}
```

Try it Yourself »

Overlapping Elements

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:

This is a heading

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

Example

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Try it Yourself »

Positioning Text In an Image

How to position text over an image:

Example

Bottom Left

Top Left

Top Right

Bottom Right

Centered

CSS Layout - Overflow

The CSS `overflow` property controls what happens to content that is too big to fit into an area.

CSS Overflow

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

The `overflow` property has the following values:

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

Note: The `overflow` property only works for block elements with a specified height.

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

overflow: visible

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

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

Example

```
div {
  width: 200px;
  height: 50px;
  background-color: #eee;
  overflow: visible;
}
```

Try it Yourself »

overflow: hidden

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

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

Example

```
div {
  overflow: hidden;
}
```

Try it Yourself »

overflow: scroll

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

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

Example

```
div {
  overflow: scroll;
}
```

Try it Yourself »

overflow: auto

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

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

Example

```
div {
  overflow: auto;
}
```

Try it Yourself »

overflow-x and overflow-y

The `overflow-x` and `overflow-y` properties specifies whether to change the overflow of content just horizontally or vertically (or both):

`overflow-x` specifies what to do with the left/right edges of the content.
`overflow-y` specifies what to do with the top/bottom edges of the content.

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

Example

```
div {
  overflow-x: hidden; /* Hide horizontal scrollbar */
  overflow-y: scroll; /* Add vertical scrollbar */
}
```

Try it Yourself »

Test Yourself with Exercises!

Exercise 1 »Exercise 2 »Exercise 3 »

All CSS Overflow Properties

Property	Description
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it over content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it ov element's content area

CSS Layout - float and clear

The CSS `float` property specifies how an element should float.

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

The float Property

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

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

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

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

Example - float: right;

The following example specifies that an image should float to the **right** 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: right;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Example - No float

In the following example the image will be displayed just where it occurs in the text (float: none;):


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: none;
}
```

Try it Yourself »

The clear Property

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

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

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side

- inherit - The element inherits the clear value of its parent

The most common way to use the `clear` property is after you have used a `float` property on an element.

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.

The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

Example

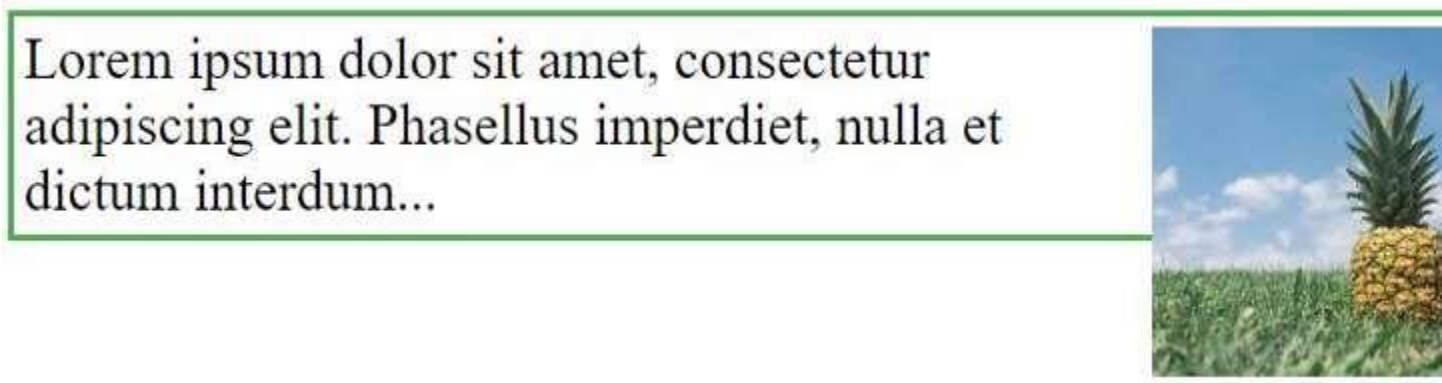
```
div {
  clear: left;
}
```

Try it Yourself »

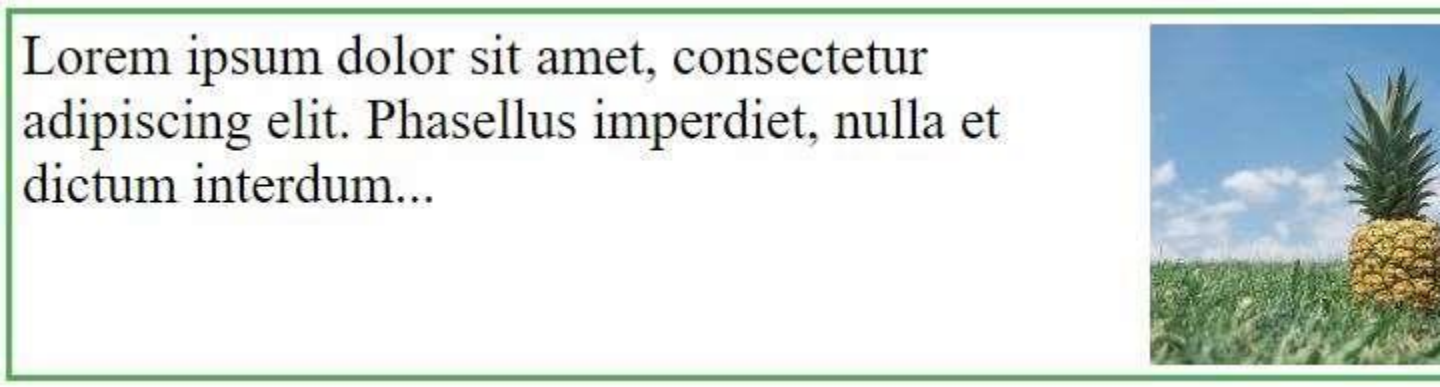
The clearfix Hack

If an element is taller than the element containing it, and it is floated, it will "overflow" outside of its container:

Without Clearfix



With Clearfix



Then we can add `overflow: auto;` to the containing element to fix this problem:

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

You will learn more about the `::after` pseudo-element in a later chapter.

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:

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 */
}
```

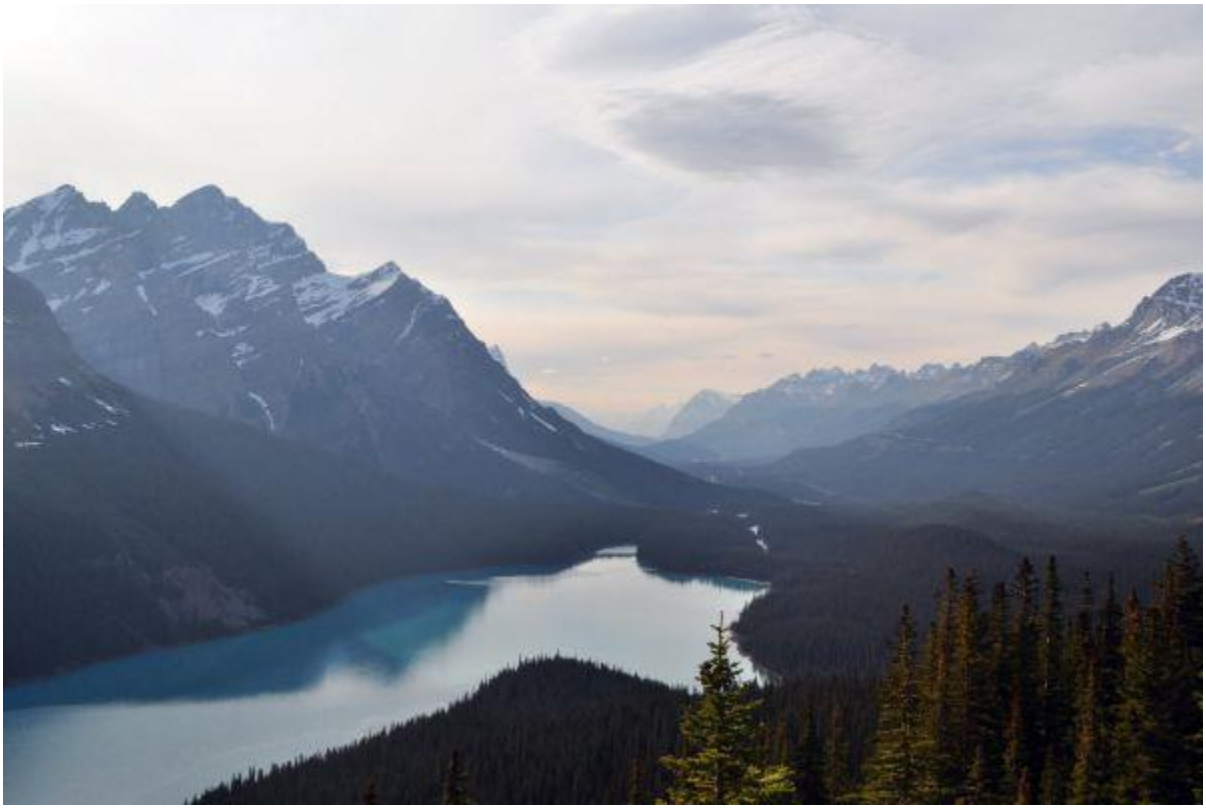
Try it Yourself »

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 */
}
```

Try it Yourself »

Equal Height Boxes

In the previous example, you learned how to float boxes side by side with an equal width. However, it is not easy to create floating boxes with equal heights. A quick fix however, is to set a fixed height, like in the example below:

Box 1

Some content, some content, some content

Box 2

Some content, some content, some content

Some content, some content, some content

Some content, some content, some content

Example

```
.box {
  height: 500px;
}
```

Try it Yourself »

However, this is not very flexible. It is ok if you can guarantee that the boxes will always have the same amount of content in them. But many times, the

content is not the same. If you try the example above on a mobile phone, you will see that the second box's content will be displayed outside of the box. This is where CSS3 Flexbox comes in handy - as it can automatically stretch boxes to be as long as the longest box:

Example

Using **Flexbox** to create flexible boxes:

Box 1 - This is some text to make sure that the content gets really tall. This is some text to make sure that the content gets really tall. This is some text to make sure that the content gets really tall.

Box 2 - My height will follow Box 1.

Try it Yourself »

The only problem with Flexbox is that it does not work in Internet Explorer 10 or earlier versions. You can read more about the Flexbox Layout Module in our CSS Flexbox Chapter.

Navigation Menu

Use `float` with a list of hyperlinks to create a horizontal menu:

Example

- Home
- News
- Contact
- About

Try it Yourself »

Web Layout Example

It is also common to do entire web layouts using the `float` property:

Example

```
.header, .footer {
  background-color: grey;
  color: white;
  padding: 15px;
}

.column {
  float: left;
  padding: 15px;
}

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

.menu {
  width: 25%;
}

.content {
  width: 75%;
}
```

Try it Yourself »

More Examples

An image with border and margins that floats to the right in a paragraph
Let an image float to the right in a paragraph. Add border and margins to the image.

An image with a caption that floats to the right
Let an image with a caption float to the right.

Let the first letter of a paragraph float to the left
Let the first letter of a paragraph float to the left and style the letter.

Creating a website with float
Use float to create a homepage with a navbar, header, footer, left content and main content.

All CSS Float Properties

Property	Description
box-sizing	Defines how the width and height of an element are calculated: should padding and borders, or not
clear	Specifies what elements can float beside the cleared element and on w
float	Specifies how an element should float
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it overflo content area

overflow-y	Specifies what to do with the top/bottom edges of the content if it over content area

CSS Layout - display: inline-block

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`:

Example

```
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;
}
```

Try it Yourself »

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;
}
```

CSS Layout - Horizontal & Vertical Align

Center Align Elements

To horizontally center a block element (like), 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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Tip: For more examples on how to align text, see the CSS Text chapter.

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%;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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).

The clearfix Hack

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 `overflow: auto;` to the containing element to fix this problem:

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Center Vertically - Using position & transform

If `padding` and `line-height` are not options, a third 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%);
}
```

CSS Combinators

A combinator is something that explains the relationship between the selectors.

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

There are four different combinators in CSS:

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

Descendant Selector

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

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

Example

```
div p {
  background-color: yellow;
}
```

Try it Yourself »

Child Selector

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

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

Example

```
div > p {
  background-color: yellow;
}
```

Try it Yourself »

Adjacent Sibling Selector

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

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

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

Example

```
div + p {
  background-color: yellow;
}
```

Try it Yourself »

General Sibling Selector

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

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

Example

```
div ~ p {
  background-color: yellow;
}
```

Try it Yourself »

CSS Opacity / Transparency

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 value, the more transparent:

opacity 0.2

opacity 0.5

opacity 1
(default)

Note: IE8 and earlier use `filter:alpha(opacity=x)`. The x can take a value from 0 - 100. A lower value makes the element more transparent.

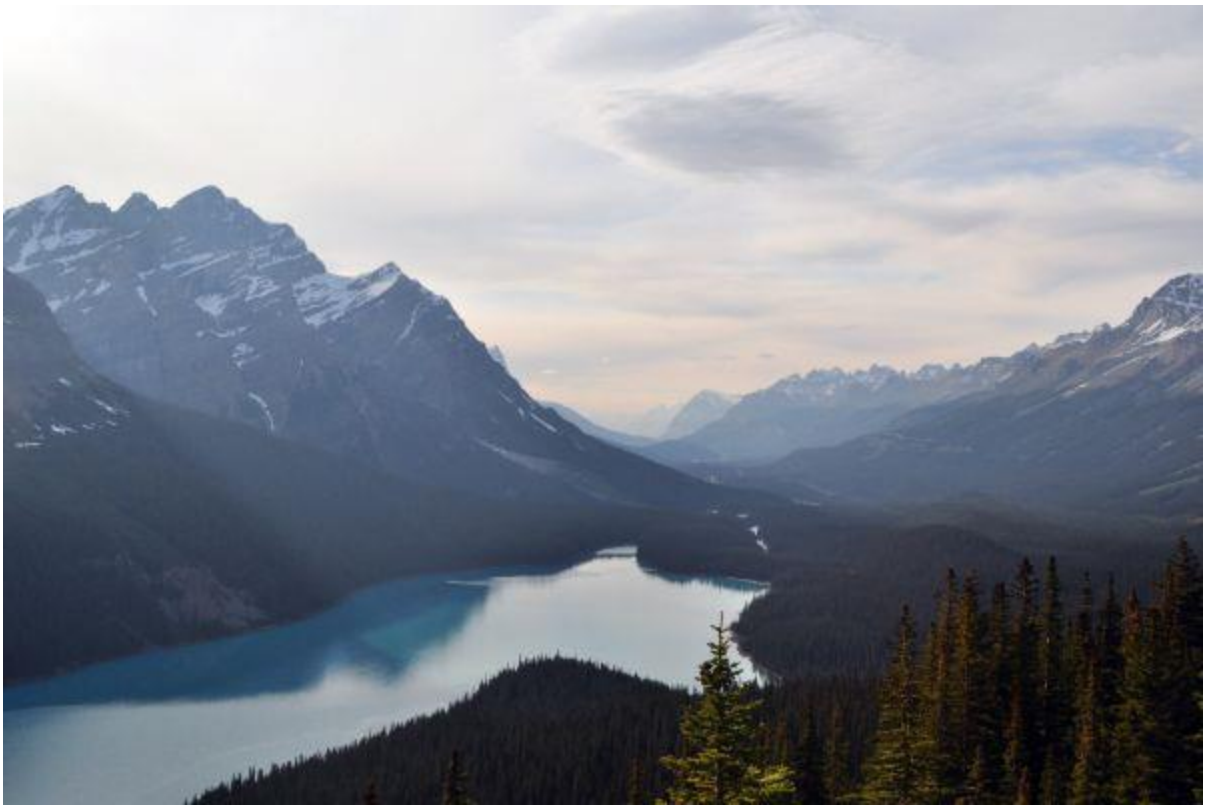
Example

```
img {
  opacity: 0.5;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}
```

Try it Yourself »

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;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}

img:hover {
  opacity: 1.0;
  filter: alpha(opacity=100); /* For IE8 and earlier */
}
```

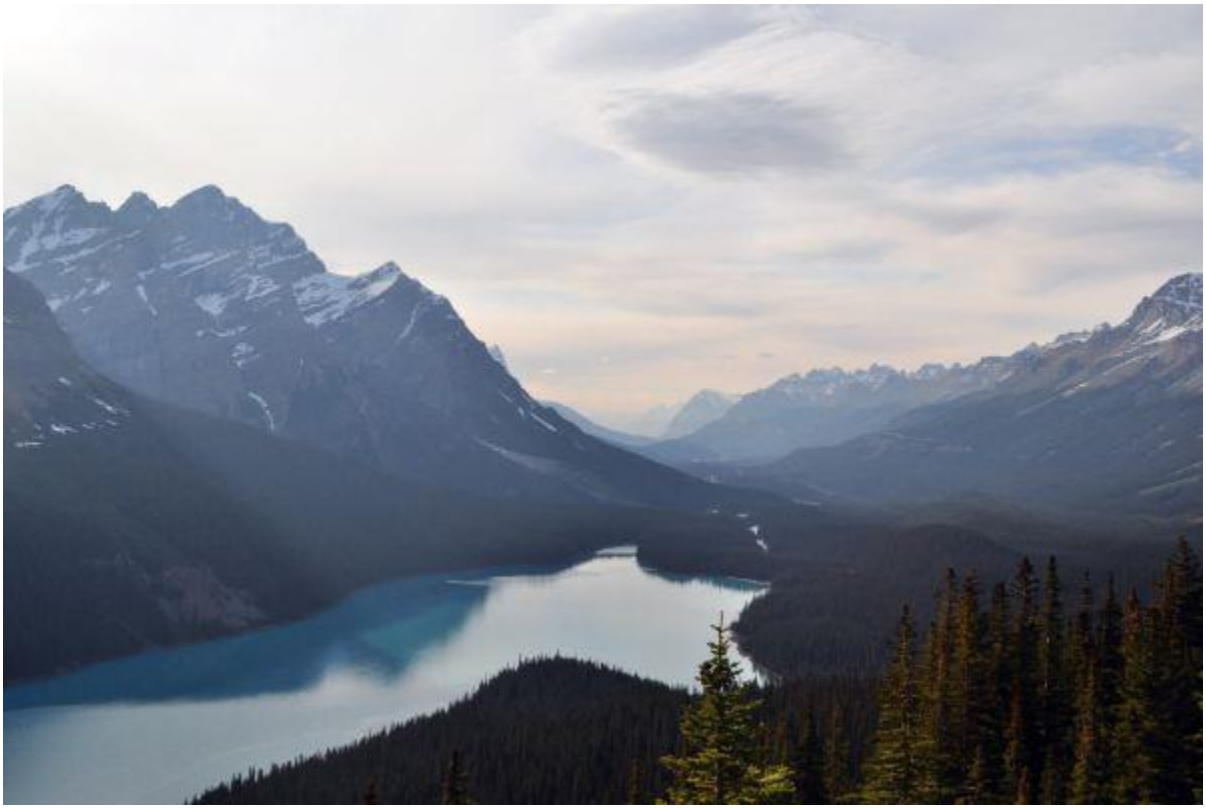
Try it Yourself »

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;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}
```

Try it Yourself »

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:

opacity 1

opacity 0.6

opacity 0.3

opacity 0.1

Example

```
div {
  opacity: 0.3;
  filter: alpha(opacity=30); /* For IE8 and earlier */
}
```

Try it Yourself »

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:

100% opacity

60% opacity

30% opacity

10% opacity

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 */
}
```

Try it Yourself »

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;
  filter: alpha(opacity=60); /* For IE8 and earlier */
}

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>
```

CSS Navigation Bar

Demo: Navigation Bars

Vertical

- Home
- News
- Contact
- About

Horizontal

- Home
- News
- Contact
- About

- Home
- News
- Contact
- About

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 <ul> and <li> 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>
```

Try it Yourself »

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

Example

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

Try it Yourself »

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.

Vertical Navigation Bar

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

Example

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

Try it Yourself »

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 <ul>, 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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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: #4CAF50;
  color: white;
}
```

Try it Yourself »

Center Links & Add Borders

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

Add the `border` property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a `border-bottom` to all <li> 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;
}
```

Try it Yourself »

Full-height Fixed Vertical Navbar

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

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 */
}
```

Try it Yourself »

Note: This example might not work properly on mobile devices.

Horizontal Navigation Bar

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 <li> elements as inline, in addition to the "standard" code above:

Example

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

Try it Yourself »

Example explained:

- `display: inline;` - By default, <li> 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 <li> elements, and specify a layout for the navigation links:

Example

```
li {
  float: left;
}

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

Try it Yourself »

Example explained:

- `float: left;` - use float to get block elements to slide next to each other
- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify padding (and height, width, margins, etc. if you want)
- `padding: 8px;` - Since block elements take up the full width available, they cannot float next to each other. Therefore, specify some padding to make them look good

- `background-color: #dddddd;` - Add a gray background-color to each a element

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

Example

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

Try it Yourself »

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;
}

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;
}
```

Try it Yourself »

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: #4CAF50;
}
```

Try it Yourself »

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>
```

Try it Yourself »

Border Dividers

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

- Home
- News
- Contact
- About

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;
}
```

Try it Yourself »

Fixed Navigation Bar

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

Fixed Top

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

Try it Yourself »

Fixed Bottom

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

Try it Yourself »

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;
}
```

Try it Yourself »

Sticky Navbar

Add `position: sticky;` to <ul> 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).

Example

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

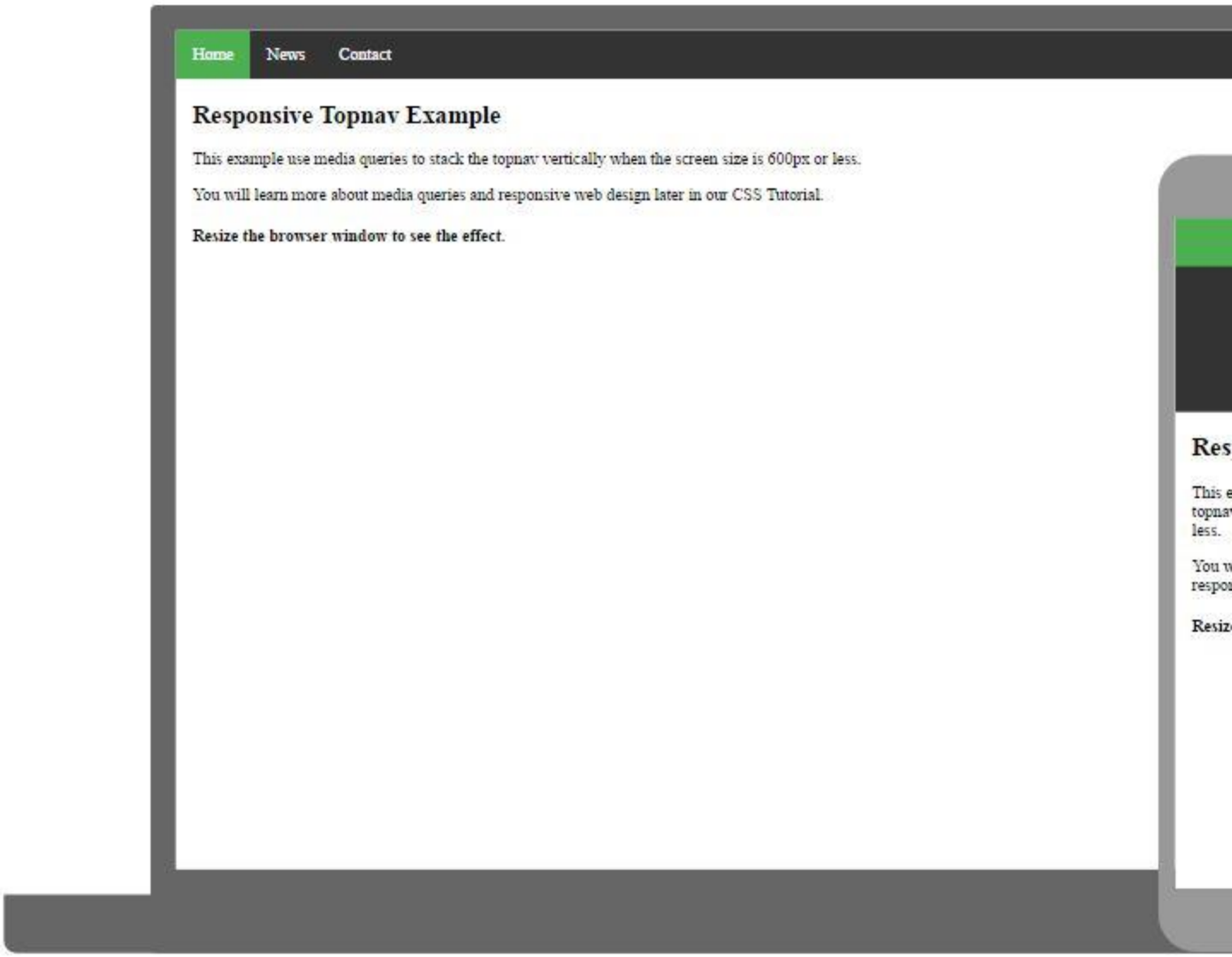
Try it Yourself »

Note: Internet Explorer, Edge 15 and earlier versions 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.

More Examples

Responsive Topnav

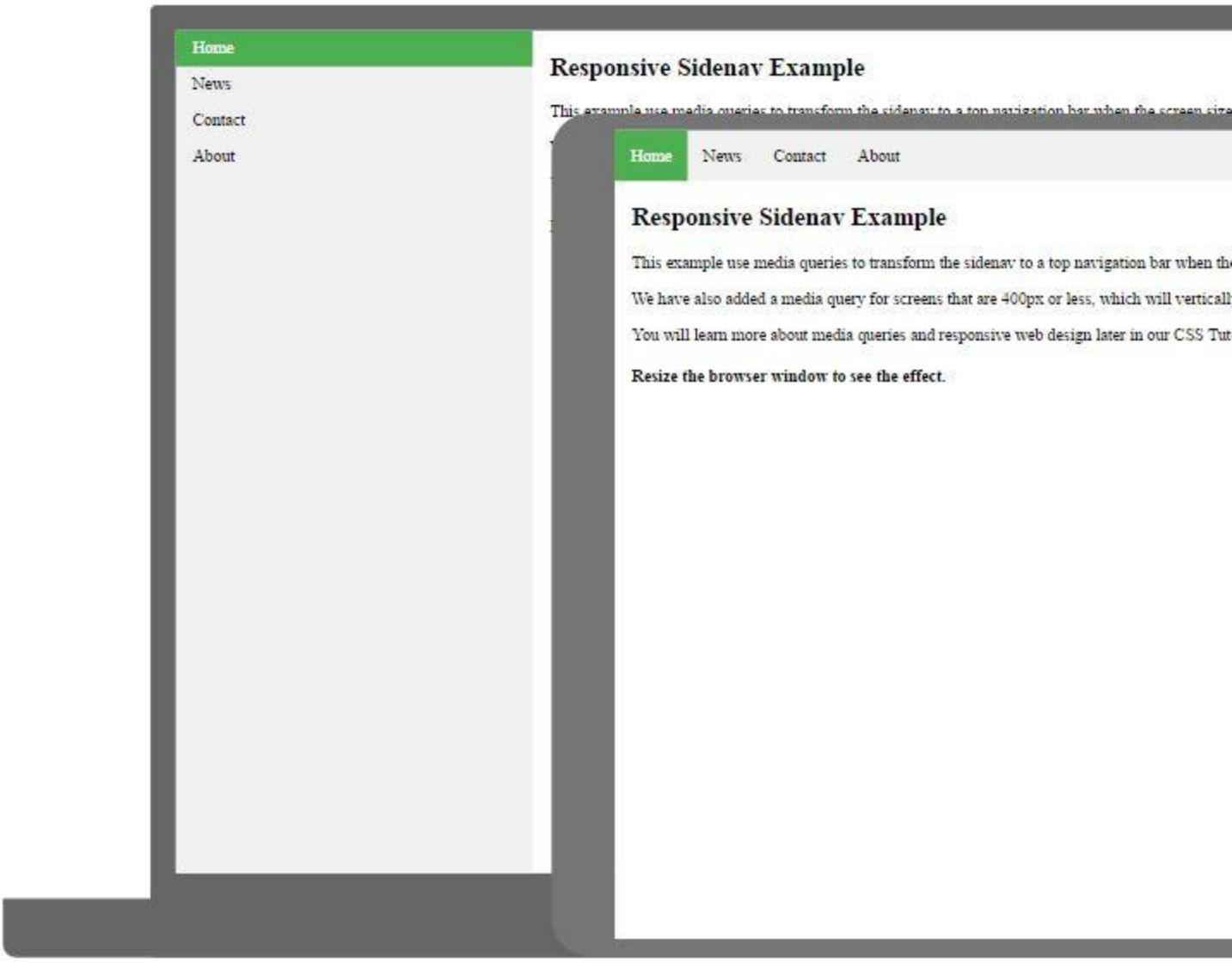
How to use CSS media queries to create a responsive top navigation.



Try it Yourself »

Responsive Sidenav

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



Try it Yourself »

Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

CSS Dropdowns

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>
```

Try it Yourself »

Example Explained

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

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

Wrap a 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;
}

/* 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 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>
```

Try it Yourself »

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;
}
```

Try it Yourself »

More Examples

Dropdown Image

How to add an image and other content inside the dropdown box.

Hover over the image:

Try it Yourself »

Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

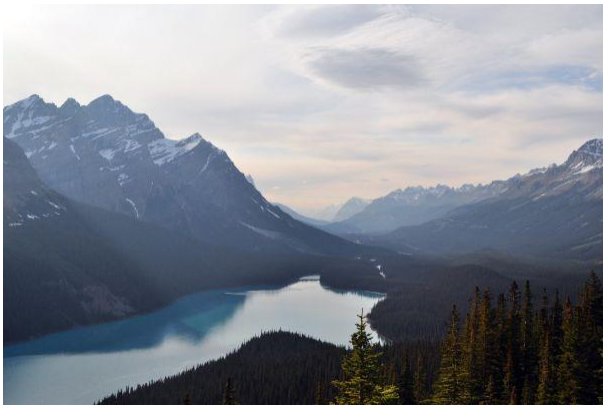
CSS Image Gallery

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">
    <img src="img_5terre.jpg" alt="Cinque Terre" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_forest.jpg">
    <img src="img_forest.jpg" alt="Forest" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
```

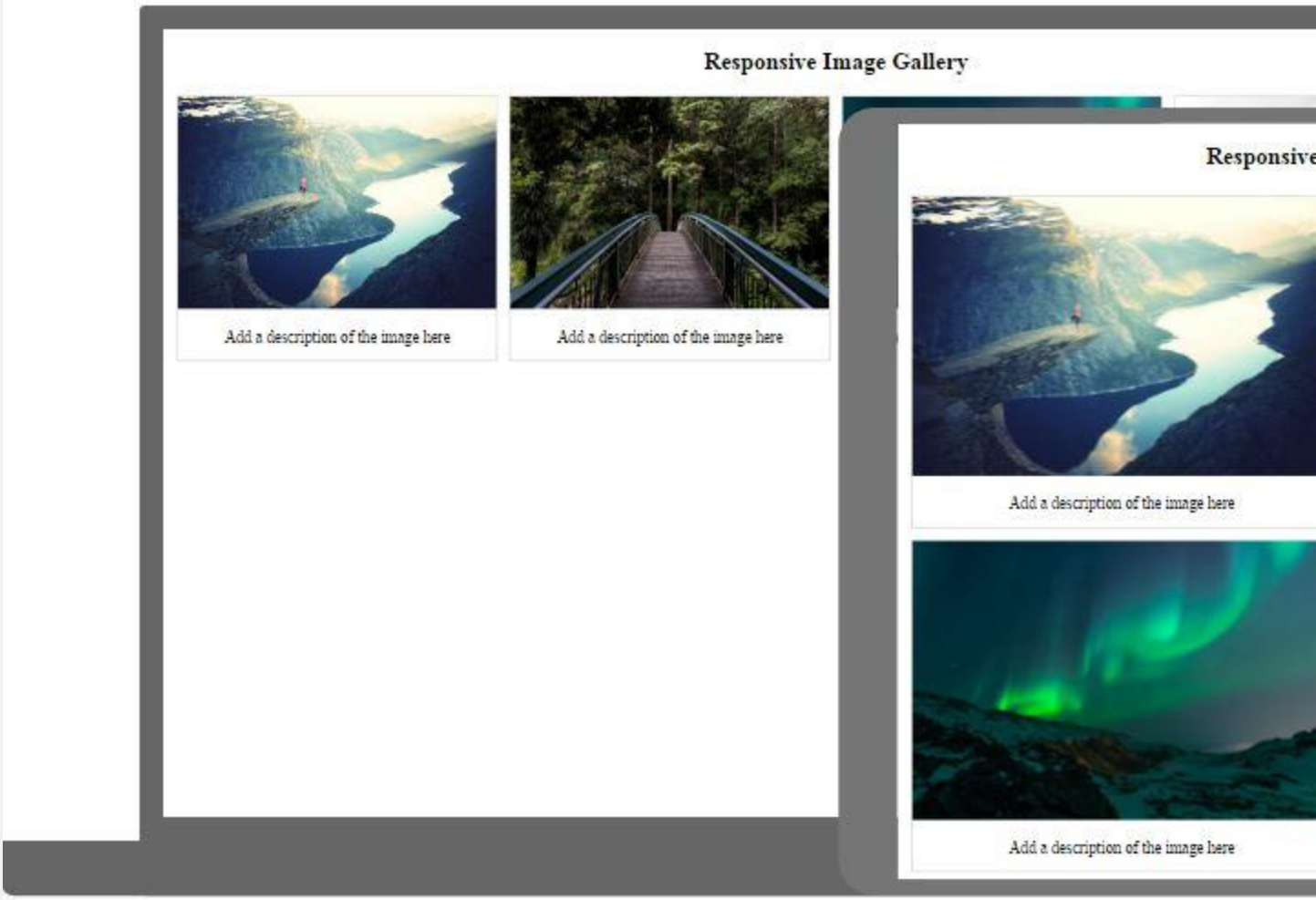
```
</div>

<div class="gallery">
  <a target="_blank" href="img_lights.jpg">
    <img src="img_lights.jpg" alt="Northern Lights" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_mountains.jpg">
    <img src="img_mountains.jpg" alt="Mountains" width="600" height="400">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

Try it Yourself »

More Examples

Responsive Image Gallery

How to use CSS media queries to create a responsive image gallery that will look good on desktops, tablets and smart phones.



CSS Image Sprites

Image Sprites

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

Image Sprites - Simple Example

Instead of using three separate images, we use this single image ("img_navsprites.gif"):

With CSS, we can show just the part of the image we need.

In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:

Example

```
#home {
  width: 46px;
  height: 44px;
  background: url(img_navsprites.gif) 0 0;
}
```

Try it Yourself »

Example explained:

- `<img id="home" src="img_trans.gif">` - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- `width: 46px; height: 44px;` - Defines the portion of the image we want to use
- `background: url(img_navsprites.gif) 0 0;` - Defines the background image and its position (left 0px, top 0px)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

Image Sprites - Create a Navigation List

We want to use the sprite image ("img_navsprites.gif") to create a navigation list.

We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist {
  position: relative;
}

#navlist li {
  margin: 0;
  padding: 0;
  list-style: none;
  position: absolute;
  top: 0;
}

#navlist li, #navlist a {
  height: 44px;
  display: block;
}

#home {
  left: 0px;
  width: 46px;
  background: url('img_navsprites.gif') 0 0;
}

#prev {
  left: 63px;
  width: 43px;
  background: url('img_navsprites.gif') -47px 0;
}

#next {
  left: 129px;
  width: 43px;
  background: url('img_navsprites.gif') -91px 0;
}
```

Try it Yourself »

Example explained:

- #navlist {position:relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li {margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin and padding are set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a {height:44px;display:block;} - the height of all the images are 44px

Now start to position and style for each specific part:

- #home {left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home {background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev {left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev {background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next {left:129px;width:43px;}- Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next {background:url('img_navsprites.gif') -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

Now we want to add a hover effect to our navigation list.

Tip: The `:hover` selector can be used on all elements, not only on links.

Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:

Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.

We only add three lines of code to add the hover effect:

Example

```
#home a:hover {
  background: url('img_navsprites_hover.gif') 0 -45px;
}

#prev a:hover {
  background: url('img_navsprites_hover.gif') -47px -45px;
}

#next a:hover {
  background: url('img_navsprites_hover.gif') -91px -45px;
}
```

CSS Attribute Selectors

Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes or attribute values.

CSS [attribute] Selector

The `[attribute]` selector is used to select elements with a specified attribute.

The following example selects all <a> elements with a target attribute:

Example

```
a[target] {
  background-color: yellow;
}
```

Try it Yourself »

CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

Example

```
a[target="_blank"] {
  background-color: yellow;
}
```

Try it Yourself »

CSS [attribute~="value"] Selector

The `[attribute~="value"]` selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"] {
  border: 5px solid yellow;
}
```

Try it Yourself »

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|="value"] Selector

The `[attribute|="value"]` selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|="top"] {
  background: yellow;
}
```

Try it Yourself »

CSS [attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"] {
  background: yellow;
}
```

Try it Yourself »

CSS [attribute$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example

```
[class$="test"] {
  background: yellow;
}
```

Try it Yourself »

CSS [attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {
  background: yellow;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Tip: Visit our CSS Forms Tutorial for more examples on how to style forms with CSS.

Test Yourself with Exercises!

Exercise 1 »Exercise 2 »Exercise 3 »Exercise 4 »Exercise 5 »Exercise 6 »

More Examples of CSS Selectors

Use our CSS Selector Tester to demonstrate the different selectors.

For a complete reference of all the CSS selectors, please go to our CSS Selectors Reference.

CSS Forms

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

First Name Last Name Country Try it Yourself »

Styling Input Fields

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

First Name

Example

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

Try it Yourself »

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..

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

Colored Inputs

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

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

Example

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

Try it Yourself »

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;
}
```

Try it Yourself »

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] {
  -webkit-transition: width 0.4s ease-in-out;
  transition: width 0.4s ease-in-out;
}

input[type=text]:focus {
  width: 100%;
}
```

Try it Yourself »

Styling Textareas

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

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;
}
```

Try it Yourself »

Styling Select Menus

Example

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

Try it Yourself »

Styling Input Buttons

Example

```
input[type=button], input[type=submit], input[type=reset] {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 16px 32px;
  text-decoration: none;
  margin: 4px 2px;
  cursor: pointer;
}

/* Tip: use width: 100% for full-width buttons */
```

Try it Yourself »

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 use media queries to create a responsive form. You will learn more about this in a later chapter.

CSS Counters

Pizza

Hamburger

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) ": ";
}
```

Try it Yourself »

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) " ";
}
```

Try it Yourself »

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,".") " ";
}
```

Try it Yourself »

CSS Counter Properties

Property	Description
content	Used with the ::before and ::after pseudo-elements, to insert generate

counter-increment	Increments one or more counter values
counter-reset	Creates or resets one or more counters

CSS Website Layout

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;
}
```

Result

Header

Try it Yourself »

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

LinkLinkLink

Try it Yourself »

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 floats 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.

Try it Yourself »

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...

Try it Yourself »

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

Try it Yourself »

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:

Try it Yourself »

CSS Units

CSS Units

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as `width`, `margin`, `padding`, `font-size`, etc.

Length is a number followed by a length unit, such as 10px, 2em, etc.

A whitespace cannot appear between the number and the unit. However, if the value is 0, the unit can be omitted.

For some CSS properties, negative lengths are allowed.

There are two types of length units: absolute and relative.

Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description
cm	centimetersTry it
mm	millimetersTry it

in	inches (1in = 96px = 2.54cm)Try it
px *	pixels (1px = 1/96th of 1in)Try it
pt	points (1pt = 1/72 of 1in)Try it
pc	picas (1pc = 12 pt)Try it

* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scales better between different rendering mediums.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)

ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

CSS Specificity

What is Specificity?

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out.

Think of specificity as a score/rank that determines which style declarations are ultimately applied to an element.

The universal selector (*) has low specificity, while ID selectors are highly specific!

Note: Specificity is a common reason why your CSS-rules don't apply to some elements, although you think they should.

Specificity Hierarchy

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

Inline styles - An inline style is attached directly to the element to be styled. Example: <h1 style="color: #ffffff;">.

IDs - An ID is a unique identifier for the page elements, such as #navbar.

Classes, attributes and pseudo-classes - This category includes .classes, [attributes] and pseudo-classes such as :hover, :focus etc.

Elements and pseudo-elements - This category includes element names and pseudo-elements, such as h1, div, :before and :after.

How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element.

Consider these three code fragments:

Example

```
A: h1
B: #content h1
C: <div id="content"><h1 style="color: #ffffff">Heading</h1></div>
```

The specificity of A is 1 (one element)
The specificity of B is 101 (one ID reference and one element)
The specificity of C is 1000 (inline styling)

Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

Specificity Rules

Equal specificity: the latest rule counts - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied:

Example

```
h1 {background-color: yellow;}
h1 {background-color: red;}
```

Try it Yourself »

the latter rule is always applied.

ID selectors have a higher specificity than attribute selectors - Look at the following three code lines:

Example

```
div#a {background-color: green;}
#a {background-color: yellow;}
div[id=a] {background-color: blue;}
```

Try it Yourself »

the first rule is more specific than the other two, and will be applied.

Contextual selectors are more specific than a single element selector - The embedded style sheet is closer to the element to be styled. So in the following situation

Example

```
From external CSS file:
#content h1 {background-color: red;}

In HTML file:
<style>
#content h1 {
  background-color: yellow;
}
</style>
```

the latter rule will be applied.

A class selector beats any number of element selectors - a class selector such as .intro beats h1, p, div, etc:

Example

```
.intro {background-color: yellow;}
h1 {background-color: red;}
```