CSS

CSS Introduction

What is CSS?

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

Why Use CSS?

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

CSS Solved a Big Problem

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

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

<h1>This is a heading</h1>

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

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

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

CSS removed the style formatting from the HTML page!

CSS Saves a Lot of Work!

The style definitions are normally saved in external .css files.

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

CSS Syntax and Selectors

CSS Syntax

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

CSS selector

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.

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

Example

```
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.

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

Example

```
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.

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

Example

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

Note: An id name cannot start with a number!

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.

In the example below, all HTML elements with class="center" will be red and center-aligned:

Example

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

You can also specify that only specific HTML elements should be affected by a class. In the example below, only <p> elements with class="center" will be center-aligned:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
   text-align: center;
   color: red;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>

</body>
</html>
```

HTML elements can also refer to more than one class.

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
   text-align: center;
   color: red;
}

p.large {
   font-size: 300%;
}
</style>
</head>
<body>

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

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

</body>
</html>
```

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.

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

Example

```
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.

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

Example

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

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

CSS How To...

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

Three Ways to Insert CSS (style sheet)

1.External style sheet
2.Internal style sheet
3.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. The <link> element goes inside the <head> section:

Example

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

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

An external style sheet can be written in any text editor. 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" looks

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

Example

```
<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.

The example below shows how to change the color and the left margin of a <h1> element:

Example

```
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

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.

Example

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;
}
```

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

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

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

```
<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.

Background Color

You can set the background color for HTML elements:

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:

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:

Example

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

CSS Backgrounds

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

1.background-color
2.background-image
3.background-repeat
4.background-attachment
5.background-position

Background Color

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

The background color of a page is set like this:

Example

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

With CSS, a color is most often specified by:

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

CSS Colors

Colors in CSS can be specified by the following methods:

1.Hexadecimal colors

2.RGB colors
3.RGBA colors
4.HSL colors
5.HSLA colors
6.Predefined/Cross-browser color names

In the example below, the <h1>, <p>, and <div> elements have different background colors:
Example

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

div {
   background-color: lightblue;
}

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

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-image: url("paper.gif");
}
</style>
</head>
<body>

<h1>Hello World!</h1>

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

</body>
</html>
```

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");
}
```

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-image: url("gradient_bg.png");
   background-repeat: repeat-x;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Here, a background image is repeated only horizontally!</p>

</body>
</html>
```

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

Background Image - Set position and no-repeat

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

Example

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

In the example above, the background image is shown 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.

The position of the image is specified by the background-position property:

Example

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

Background Image - Fixed position

To specify that the background image should be fixed (will not scroll with the rest of the page), use the background-attachment property:

Example

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

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

```
<style>
body {
   background: #ffffff url("img_tree.png") no-repeat right top;
   margin-right: 200px;
}
</style>
```

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

1.background-color
2.background-image
3.background-repeat
4.background-attachment
5.background-position

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

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 the rest of the page
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

```
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;}
```
--- **for mixed border**

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;
}
```

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.

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;
}
```

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;
}
```

The example above gives the same result as this:

Example

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

So, here is how it works:

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

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

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

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

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

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

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

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

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;
}
```

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

Left Border

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

Bottom Border

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

Rounded Borders

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

```
<style>
p.normal {
   border: 2px solid red;              //Normal border

}

p.round1 {
   border: 2px solid red;                  //Round border

   border-radius: 5px;
}

p.round2 {
   border: 2px solid red;                   //Rounder border

   border-radius: 8px;
}

p.round3 {
   border: 2px solid red;                  //Roundest border

   border-radius: 12px;
}
</style>
```

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

More Examples

All the top border properties in one declaration

```
<style>
p {
   border-style: solid;
   border-top: thick double #ff0000;
}
</style>
```

Set the style of the bottom border

```
<style>
p {border-style: solid;}
p.none {border-bottom-style: none;}
p.dotted {border-bottom-style: dotted;}
</style>
```

Set the width of the left border

```
<style>
p {
   border-style: solid;
   border-left-width: 15px;
}
</style>
```

Set the color of the four borders

```
<style>
p.four {
   border-style: solid;
   border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255);
}
```

```
</style>
```

Set the color of the right border

```
<style>
p {
   border-style: solid;
   border-right-color: #ff0000;
}
</style>
```

.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.

The following example sets different margins for all four sides of a <p> element:
Example

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

Margin - Shorthand Property

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

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

margin-top
margin-right
margin-bottom
margin-left
So, here is how it works:

If the margin property has four values:

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

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

If the margin property has three values:

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

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

If the margin property has two values:

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

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

If the margin property has one value:

margin: 25px;
all four margins are 25px
Example

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

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

The inherit Value

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

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

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

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

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

In this example the h1 element has a bottom margin of 50px and the h2 element has a top margin of 20px. Then, the vertical margin between h1 and h2 should have been 70px (50px + 20px). However, due to margin collapse, the actual margin ends up being 50px.

All CSS Margin Properties

Property	Description

margin	A shorthand property for setting the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

CSS Padding

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.

The following example sets different padding for all four sides of a <div> element:

Example

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

Padding - Shorthand Property

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

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

padding-top
padding-right
padding-bottom
padding-left
So, here is how it works:

If the padding property has four values:

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

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

If the padding property has three values:

padding: 25px 50px 75px;
top padding is 25px
right and left paddings are 50px
bottom padding is 75px
Example

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

If the padding property has two values:

padding: 25px 50px;
top and bottom paddings are 25px
right and left paddings are 50px
Example

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

If the padding property has one value:

padding: 25px;
all four paddings are 25px
Example

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

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.

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

Example

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

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

Example

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

All CSS Padding Properties

Property	**Description**
padding	A shorthand property for setting all the padding properties in one declaration
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.

Example

```
<style>
div {
   width: 50%;
   border: 1px solid #4CAF50;
}
</style>
```

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.

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

The following example shows a <div> element with a height of 100 pixels and a max-width of 500 pixels:

Example

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

Set the height and width of an image using percent

```
<!DOCTYPE html>
<html>
<head>
<style>
html, body {
   height: 100%;
}

img.one {
   height: auto;
   width: auto;
}

img.two {
    height: 50%;
    width: 50%;
}
</style>
</head>
<body>

<h2>Set the height and width in %</h2>
<p>Resize the browser window to see the effect.</p>

<p>Original image:</p>
<img class="one" src="ocean.jpg" width="300" height="300"><br>

<p>Sized image (in %):</p>
<img class="two" src="ocean.jpg" width="300" height="300">

</body>
</html>
```

Set the height and width of elements

```
<!DOCTYPE html>
<html>
<head>
<style>
img.one {
    height: auto;
}

img.two {
    height: 200px;
    width: 200px;
}

div.three {
    height: 300px;
    width: 300px;
    background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the height and width of elements</h2>

<p>Original image:</p>
<img class="one" src="ocean.jpg" width="300" height="300"><br>

<p>Sized image (200x200 pixels):</p>
<img class="two" src="ocean.jpg" width="300" height="300"><br>

<p>The height and width of this div element is 300px:</p>
```

```
<div class="three"></div>

</body>
</html>
```

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.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   max-width: 400px;
   min-width: 100px;
   background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the max-width and min-width of an element</h2>
<p>Resize the browser window to see the effect.</p>

<div>This is some text.</div>

</body>
</html>
```

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.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   max-height: 600px;
   min-height: 400px;
   background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the max-height and min-height of an element</h2>
<p>Resize the browser window to see the effect.</p>

<div>This is some text.</div>
</body>
</html>
```

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.

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

```
div {
   width: 300px;
   border: 25px solid green;
   padding: 25px;
   margin: 25px;
}
```

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.

Assume we want to style a <div> element to have a total width of 350px:

Example

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

Here is the calculation:
320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px
The total width of an element should be calculated like this:

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

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

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

CSS Outline

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

CSS has the following outline properties:

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

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

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
   border: 1px solid yellow;
   outline-style: solid;
   outline-color: invert;
}
</style>
</head>
<body>

<h2>Using outline-color:invert</h2>

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

</body>
</html>
```

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)

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.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
</style>
</head>
<body>

<h2>The outline Property</h2>

<p class="ex1">A dashed outline.</p>
<p class="ex2">A dotted red outline.</p>
<p class="ex3">A 5px solid yellow outline.</p>
<p class="ex4">A thick ridge pink outline.</p>

</body>
</html>
```

Outline Offset

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

Example

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

All CSS Outline Properties

Property	Description
outline	A shorthand property for setting outline-width, outline-style, and outline-color in one declaration
outline-color	Sets the color of an outline
outline-offset	Specifies the space between an outline and the edge or border of an element
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline

CSS Text

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   border: 1px solid gray;
   padding: 8px;
}

h1 {
   text-align: center;
   text-transform: uppercase;
   color: #4CAF50;
}

p {
   text-indent: 50px;
   text-align: justify;
   letter-spacing: 3px;
}

a {
   text-decoration: none;
   color: #008CBA;
}
</style>
</head>
<body>

<div>
<h1>text formatting</h1>
```

```
<p>This text is styled with some of the text formatting properties. The heading uses the
text-align, text-transform, and color properties.
The paragraph is indented, aligned, and the space between characters is specified. The
underline is removed from this colored
<a target="_blank" href="tryit.asp?filename=trycss_text">"Try it Yourself"</a> link.</p>
</div>

</body>
</html>
```

Text Color

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

1.a color name - like "red"
2.a HEX value - like "#ff0000"
3.an RGB value - like "rgb(255,0,0)"

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

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
   text-transform: uppercase;
}
```

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

p.capitalize {
   text-transform: capitalize;
}
</style>
</head>
<body>

<p class="uppercase">This is some text.</p>
<p class="lowercase">This is some text.</p>
<p class="capitalize">This is some text.</p>

</body>
</html>
```

Text Indentation

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

Example

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
   direction: rtl;
}
</style>
</head>
<body>

<p>This is the default text direction.</p>
<p class="ex1"><bdo dir="rtl">This is right-to-left text direction.</bdo></p>

</body>
</html>
```

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;
}
```

More Examples

Disable text wrapping inside an element

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

```
<style>
p {
   white-space: nowrap;
}
</style>
```

Vertical alignment of an image

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

```
<style>
img.top {
   vertical-align: text-top;
}

img.bottom {
   vertical-align: text-bottom;
}
</style>
```

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

All CSS Text Properties

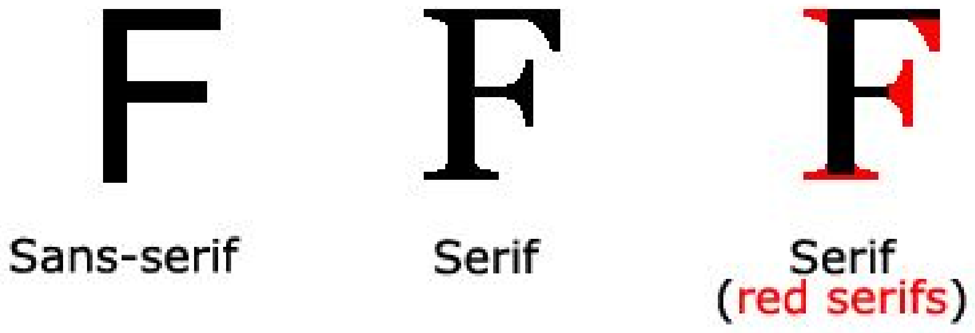
Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the 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 should be signaled to the user
unicode-bidi	Used together with the direction property to set or return whether the text should be

	overridden to support multiple languages in the same document
[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 ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters

Monospace	Courier New Lucida Console	All monospace characters have the same width

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

```
<style>
p.serif {
   font-family: "Times New Roman", Times, serif;
}

p.sansserif {
   font-family: Arial, Helvetica, sans-serif;
}
</style>
```

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:
Example

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<body>

<h1 style="font-size:10vw;">Responsive Text</h1>

<p style="font-size:5vw;">Resize the browser window to see how the text size scales.</p>

<p style="font-size:5vw;">Use the "vw" unit when sizing the text. 10vw will set the size to
10% of the viewport width.</p>

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

</body>
</html>
```

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;
}
```

More Examples

All the font properties in one declaration

This example demonstrates how to use the shorthand property for setting all of the font properties in one declaration.

```
<style>
p.ex1 {
   font: 15px arial, sans-serif;
}

p.ex2 {
   font:italic bold 12px/30px Georgia, serif;
}
</style>
```

All CSS Font Properties

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

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://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

Note: No downloading or installation is required!

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Font Awesome Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>

<p>Some Font Awesome icons:</p>
<i class="fa fa-cloud"></i>
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>

<p>Styled Font Awesome icons (size and color):</p>
<i class="fa fa-cloud" style="font-size:24px;"></i>
<i class="fa fa-cloud" style="font-size:36px;"></i>
<i class="fa fa-cloud" style="font-size:48px;color:red;"></i>
<i class="fa fa-cloud" style="font-size:60px;color:lightblue;"></i>

</body>
</html>
```

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>
```

```
<title>Bootstrap Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body class="container">

<p>Some Bootstrap icons:</p>
<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>
<br><br>

<p>Styled Bootstrap icons (size and color):</p>
<i class="glyphicon glyphicon-cloud" style="font-size:24px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:36px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:48px;color:red;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:60px;color:lightblue;"></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>
<title>Google Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<p>Some Google icons:</p>
<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>
<br><br>

<p>Styled Google icons (size and color):</p>
<i class="material-icons" style="font-size:24px;">cloud</i>
```

```
<i class="material-icons" style="font-size:36px;">cloud</i>
<i class="material-icons" style="font-size:48px;color:red;">cloud</i>
<i class="material-icons" style="font-size:60px;color:lightblue;">cloud</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;
}
```

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;
}
```

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

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

Text Decoration

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

Example

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

a:visited {
   text-decoration: none;
}

a:hover {
   text-decoration: underline;
}

a:active {
   text-decoration: underline;
}
```

Background Color

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

Example

```
a:link {
   background-color: yellow;
}

a:visited {
   background-color: cyan;
}

a:hover {
   background-color: lightgreen;
}

a:active {
   background-color: hotpink;
}
```

Order must be followed…

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;
}
```

More Examples

Add different styles to hyperlinks

This example demonstrates how to add other styles to hyperlinks.

```
<style>
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
```

Advanced - Create a link button with borders

Another example of how to create link boxes/buttons.

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

a:hover, a:active {
   background-color: green;
   color: white;
}
</style>
```

Change the cursor

The cursor property specifies the type of cursor to display. This example demonstrates the different types of cursors (can be useful for links).

```
<!DOCTYPE html>
<html>
<body>

<p>Mouse over the words to change the cursor.</p>
<span style="cursor:auto">auto</span><br>
<span style="cursor:crosshair">crosshair</span><br>
<span style="cursor:default">default</span><br>
<span style="cursor:e-resize">e-resize</span><br>
<span style="cursor:help">help</span><br>
<span style="cursor:move">move</span><br>
<span style="cursor:n-resize">n-resize</span><br>
<span style="cursor:ne-resize">ne-resize</span><br>
<span style="cursor:nw-resize">nw-resize</span><br>
<span style="cursor:pointer">pointer</span><br>
<span style="cursor:progress">progress</span><br>
<span style="cursor:s-resize">s-resize</span><br>
<span style="cursor:se-resize">se-resize</span><br>
<span style="cursor:sw-resize">sw-resize</span><br>
<span style="cursor:text">text</span><br>
<span style="cursor:w-resize">w-resize</span><br>
<span style="cursor:wait">wait</span><br>

</body>
```

```
</html>
```

CSS Lists

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;
}
```

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');
}
```

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;
}
```

First one is best...

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;
}
```

List - Shorthand property

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

Example

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

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

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

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

Styling List With Colors

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

Anything added to the <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;
}
```

More Examples

Customized list with a red left border

This example demonstrates how to create a list with a red left border.

```
<style>
ul {
   border-left: 5px solid red;
```

```
    background-color: #f1f1f1;
    list-style-type: none;
    padding: 10px 20px;
}
</style>
```

Full-width bordered list

This example demonstrates how to create a bordered list without bullets.

```
<style>
ul {
    list-style-type: none;
    padding: 0;
    border: 1px solid #ddd;
}

ul li {
    padding: 8px 16px;
    border-bottom: 1px solid #ddd;
}

ul li:last-child {
    border-bottom: none
}
</style>
```

All the different list-item markers for lists

This example demonstrates all the different list-item markers in CSS.

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {list-style-type: circle;}
ul.b {list-style-type: disc;}
ul.c {list-style-type: square;}

ol.d {list-style-type: armenian;}
ol.e {list-style-type: cjk-ideographic;}
ol.f {list-style-type: decimal;}
ol.g {list-style-type: decimal-leading-zero;}
ol.h {list-style-type: georgian;}
ol.i {list-style-type: hebrew;}
ol.j {list-style-type: hiragana;}
ol.k {list-style-type: hiragana-iroha;}
ol.l {list-style-type: katakana;}
ol.m {list-style-type: katakana-iroha;}
ol.n {list-style-type: lower-alpha;}
ol.o {list-style-type: lower-greek;}
ol.p {list-style-type: lower-latin;}
ol.q {list-style-type: lower-roman;}
ol.r {list-style-type: upper-alpha;}
ol.s {list-style-type: upper-latin;}
ol.t {list-style-type: upper-roman;}
ol.u {list-style-type: none;}
ol.v {list-style-type: inherit;}
</style>
</head>
<body>

<ul class="a">
  <li>Circle type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Disc type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="c">
  <li>Square type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ol class="d">
  <li>Armenian type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="e">
  <li>Cjk-ideographic type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="f">
  <li>Decimal type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
```

```
</ol>

<ol class="g">
  <li>Decimal-leading-zero type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="h">
  <li>Georgian type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="i">
  <li>Hebrew type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="j">
  <li>Hiragana type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="k">
  <li>Hiragana-iroha type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="l">
  <li>Katakana type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="m">
  <li>Katakana-iroha type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="n">
  <li>Lower-alpha type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="o">
  <li>Lower-greek type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="p">
  <li>Lower-latin type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="q">
  <li>Lower-roman type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="r">
  <li>Upper-alpha type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="s">
  <li>Upper-latin type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="t">
  <li>Upper-roman type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="u">
  <li>None type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
```

```
<ol class="v">
  <li>inherit type</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

</body>
</html>
```

All CSS List Properties

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies the position of the list-item markers (bullet points)
list-style-type	Specifies the type of list-item marker

CSS Tables

Example

```
<style>
#customers {
    font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
    border-collapse: collapse;
    width: 100%;
}

#customers td, #customers th {
    border: 1px solid #ddd;
    padding: 8px;
}

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

#customers tr:hover {background-color: #ddd;}

#customers th {
    padding-top: 12px;
    padding-bottom: 12px;
    text-align: left;
    background-color: #4CAF50;
```

```
   color: white;
}
</style>
```

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;
}
```

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;
}
```

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

Example

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

Table Width and Height

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

Example

```
table {
   width: 100%;
}

th {
   height: 50px;
}
```

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;
}
```

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;
}
```

Table Padding

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

Example

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

Horizontal Dividers

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

Example

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

Hoverable Table

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

Example

```
<style>
table {
    border-collapse: collapse;
    width: 100%;
}

th, td {
    padding: 8px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

tr:hover {background-color:#f5f5f5;}
</style>
```

Striped Tables

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

Example

```
<style>
table {
    border-collapse: collapse;
    width: 100%;
}
th, td {
    text-align: left;
    padding: 8px;
}
tr:nth-child(even) {background-color: #f2f2f2;}
</style>
```

Table Color

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

Example

```
<style>
table {
    border-collapse: collapse;
    width: 100%;
}
th, td {
    text-align: left;
    padding: 8px;
}
tr:nth-child(even){background-color: #f2f2f2}

th {
    background-color: #4CAF50;
    color: white;
}
</style>
```

Responsive Table

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

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

Example

```
<div style="overflow-x:auto;">          // written just before the table content...
<table>
... table content ...
</table>

</div>
```

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

More Examples

Make a fancy table

This example demonstrates how to create a fancy table.

```
<style>
#customers {
    font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
    border-collapse: collapse;
    width: 100%;
}

#customers td, #customers th {
    border: 1px solid #ddd;
    padding: 8px;
}

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

#customers tr:hover {background-color: #ddd;}

#customers th {
    padding-top: 12px;
    padding-bottom: 12px;
    text-align: left;
    background-color: #4CAF50;
    color: white;
}
</style>
```

Set the position of the table caption

This example demonstrates how to position the table caption.

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

caption {
   caption-side: bottom;
}
</style>
```

CSS Table Properties

Property	Description
border	Sets all the border properties in one declaration
border-collapse	Specifies whether or not table borders should be collapsed
border-spacing	Specifies the distance between the borders of adjacent cells
caption-side	Specifies the placement of a table caption
empty-cells	Specifies whether or not to display borders and background on empty cells in a table
table-layout	Sets the layout algorithm to be used for a table

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.

Block-level Elements

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

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>

- <footer>
- <section>
-

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.
This is an inline <span> element inside a paragraph.
Examples of inline elements:

- <span>
- <a>
- <img>
-

Display: none;

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

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;  // shows list in single line instead of multiple line...
}
```

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 <span> elements as block elements:
Example

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

**A display property with a value of "block" results in
a line break between the two elements.**

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
a {
   display: block;
}
</style>
</head>
<body>

<p>Display links as block elements:</p>

<a href="/html/default.asp" target="_blank">HTML</a>
<a href="/css/default.asp" target="_blank">CSS</a>
<a href="/js/default.asp" target="_blank">JavaScript</a>

</body>
</html>
```

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

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

Example

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

visibility:hidden; also hides an element.

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

Example

```
h1.hidden {
   visibility: hidden;                //don't use it...
}
```

More Examples

Using CSS together with JavaScript to show content

This example demonstrates how to use CSS and JavaScript to show an element on click.

```
<!DOCTYPE html>
<html>
<head>
<style>
#panel, .flip {
    font-size: 16px;
    padding: 10px;
    text-align: center;
    background-color: #4CAF50;
    color: white;
    border: solid 1px #a6d8a8;
    margin: auto;
}

#panel {
    display: none;
}
</style>
</head>
<body>

<p class="flip" onclick="myFunction()">Click to show panel</p>

<div id="panel">
  <p>This panel contains a div element, which is hidden by default (display: none).</p>
  <p>It is styled with CSS and we use JavaScript to show it (display: block).</p>
  <p>How it works: Notice that the p element with class="flip" has an onclick attribute
attached to it. When the user clicks on the p element, a function called myFunction() is
executed, which changes the style of the div with id="panel" from display:none (hidden) to
display:block (visible).</p>
  <p>You will learn more about JavaScript in our JavaScript Tutorial.</p>
</div>

<script>
function myFunction() {
    document.getElementById("panel").style.display = "block";
}
</script>

</body>
</html>
```

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:

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:

Example

```
div.ex1 {
   width: 500px;
   margin: auto;    //This <div> element has a width of 500px, and margin set to auto.
   border: 3px solid #73AD21;
}

div.ex2 {
   max-width: 500px;                           // this should be used...
   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:

Here is the CSS that is used:
Example
```
div.static {
   position: static;
   border: 3px solid #73AD21;          //This <div> element has position: static;
}
```

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.

Here is the CSS that is used:
Example
```
div.relative {
   position: relative;
   left: 30px;
   border: 3px solid #73AD21;     //This <div> element has position: relative;
}
```

position: fixed;

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

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

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:
Example
```
div.fixed {
   position: fixed;
   bottom: 0;
   right: 0;
   width: 300px;
   border: 3px solid #73AD21;
}
```

position: absolute;

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

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

Note: A "positioned" element is one whose position is anything except static.
Here is the CSS that is used:
Example

```
div.relative {
   position: relative;
   width: 400px;
   height: 200px;
   border: 3px solid #73AD21;
}

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

position: sticky;

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

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

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;
}
```

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

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

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Positioning Text In an Image

How to position text over an image:

Example

```
<style>
.container {
   position: relative;
}

.topright {
   position: absolute;            // top right
   top: 8px;
   right: 16px;
   font-size: 18px;
}
.topleft {
   position: absolute;
   top: 8px;                      // top left
   left: 16px;
   font-size: 18px;
}
```

```
.bottomright {
   position: absolute;
   bottom: 8px;                    // bottom right
   right: 16px;
   font-size: 18px;
}

.bottomleft {
   position: absolute;
   bottom: 8px;                     // bottom left
   left: 16px;
   font-size: 18px;
}
.center {
   position: absolute;
   left: 0;
   top: 50%;                         // center
   width: 100%;
   text-align: center;
   font-size: 18px;
}

img {
   width: 100%;
   height: auto;
   opacity: 0.3;
}
</style>
```

More Examples

Set the shape of an element

This example demonstrates how to set the shape of an element. The element is clipped into this shape, and displayed.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
   position: absolute;
   clip: rect(0px,60px,200px,0px);
}
</style>
</head>
<body>

<img src="w3css.gif" width="100" height="140">

</body>
</html>
```

All CSS Positioning Properties

Property	Description
bottom	Sets the bottom margin edge for a positioned box
clip	Clips an absolutely positioned element
left	Sets the left margin edge for a positioned box
position	Specifies the type of positioning for an element
right	Sets the right margin edge for a positioned box
top	Sets the top margin edge for a positioned box
z-index	Sets the stack order of an element

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 content or to add scrollbars when the content of an element is too big to fit in a specified area.
The overflow property has the following values:

- visible - Default. The overflow is not clipped. It 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, but a scrollbar is added to see the rest of the content
- auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

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;
     }
```

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

```
<head>
<style>
div {
   background-color: #eee;
   width: 200px;
   height: 50px;
   border: 1px dotted black;
   overflow: hidden;
}
</style>
</head>
```

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;
}
```

overflow: auto

The auto value is similar to scroll, only it add scrollbars 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;
}
```

The auto value is similar to scroll, only it add scrollbars when necessary:

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

```
<style>
div {
   background-color: #eee;
   width: 200px;
   height: 50px;
   border: 1px dotted black;
   overflow-x: hidden;
   overflow-y: scroll;
}
</style>
```

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 overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the 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

```
<head>
<style>
img {
   float: right;
}
</style>
</head>
```

Example

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

Example

```
img {
   float: none;
}
```

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

```
<style>
.div1 {
    float: left;
    width: 100px;
    height: 50px;
    margin: 10px;
    border: 3px solid #73AD21;
}

.div2 {
    border: 1px solid red;
}

.div3 {
    float: left;
    width: 100px;
    height: 50px;
    margin: 10px;
    border: 3px solid #73AD21;
}

.div4 {
    border: 1px solid red;
    clear: left;
}
</style>
</head>
```

The clearfix Hack

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

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

```
<style>
div {
    border: 3px solid #4CAF50;
    padding: 5px;
}

.img1 {
    float: right;
}

.clearfix {
    overflow: auto;
}

.img2 {
    float: right;
}
```

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 web pages:

Example

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

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

```
<style>
* {
    box-sizing: border-box;
}

.box {
    float: left;
    width: 33.33%;
    padding: 50px;
}

.clearfix::after {
    content: "";
    clear: both;
    display: table;
}
</style>
</head>
<body>

<h2>Grid of Boxes</h2>
<p>Float boxes side by side:</p>

<div class="clearfix">
  <div class="box" style="background-color:#bbb">
    <p>Some text inside the box.</p>
  </div>
  <div class="box" style="background-color:#ccc">
    <p>Some text inside the box.</p>
  </div>
  <div class="box" style="background-color:#ddd">
    <p>Some text inside the box.</p>
  </div>
</div>
```

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.

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 */
}
```

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:

Example

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

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:

Navigation Menu

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

Example

HomeNewsContactAbout

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
}

li {
    float: left;
}

li a {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

li a:hover {
    background-color: #111;
}

.active {
    background-color: red;
}
</style>
</head>
<body>

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

</body>
</html>
```

Web Layout Example

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

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
    box-sizing: border-box;
}
.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%;
}
.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
.menu li {
    padding: 8px;
    margin-bottom: 8px;
    background-color: #33b5e5;
    color: #ffffff;
}
.menu li:hover {
    background-color: #0099cc;
}
</style>
</head>
<body>

<div class="header">
  <h1>Chania</h1>
</div>

<div class="clearfix">
  <div class="column menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>
  <div class="column content">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>
    <p>You will learn more about web layout and responsive web pages in a later chapter.</p>
  </div>
</div>

<div class="footer">
  <p>Footer Text</p>
</div>

</body>
</html>
```

An image with border and margins that floats to the right in a paragraph.

```
<style>
img {
    float: right;
    border: 1px dotted black;
    margin: 0px 0px 15px 20px;
}
</style>
```

An image with a caption that floats to the right

```
<style>
div {
    float: right;
    width: 120px;
    margin: 0 0 15px 20px;
    padding: 15px;
    border: 1px solid black;
    text-align: center;
}
</style>
```

Let the first letter of a paragraph float to the left

```
<style>
span {
    float: left;
    width: 0.7em;
    font-size: 400%;
    font-family: algerian, courier;
    line-height: 80%;
}
</style>
```

Creating a website with float

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
   box-sizing: border-box;
}
body {
   margin: 0;
}
.header {
   background-color: #2196F3;
   color: white;
   text-align: center;
   padding: 15px;
}
.footer {
   background-color: #444;
   color: white;
   padding: 15px;
}
.topmenu {
   list-style-type: none;
   margin: 0;
   padding: 0;
   overflow: hidden;
   background-color: #777;
}
.topmenu li {
   float: left;
}
.topmenu li a {
   display: inline-block;
   color: white;
   text-align: center;
   padding: 16px;
   text-decoration: none;
}
.topmenu li a:hover {
   background-color: #222;
}
.topmenu li a.active {
    color: white;
    background-color: #4CAF50;
}
.column {
    float: left;
    padding: 15px;
}
.clearfix::after {
    content: "";
    clear: both;
    display: table;
}
.sidemenu {
    width: 25%;
}
.content {
    width: 75%;
}
.sidemenu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
.sidemenu li a {
    margin-bottom: 4px;
    display: block;
    padding: 8px;
    background-color: #eee;
    text-decoration: none;
    color: #666;
}
.sidemenu li a:hover {
    background-color: #555;
    color: white;
}
.sidemenu li a.active {
    background-color: #008CBA;
    color: white;
}
</style>
</head>
<body>

<ul class="topmenu">
  <li><a href="#home" class="active">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<div class="clearfix">
  <div class="column sidemenu">
    <ul>
      <li><a href="#flight">The Flight</a></li>
      <li><a href="#city" class="active">The City</a></li>
      <li><a href="#island">The Island</a></li>
      <li><a href="#food">The Food</a></li>
      <li><a href="#people">The People</a></li>
      <li><a href="#history">The History</a></li>
      <li><a href="#oceans">The Oceans</a></li>
    </ul>
  </div>

  <div class="column content">
    <div class="header">
      <h1>The City</h1>
    </div>
    <h1>Chania</h1>
    <p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>
    <p>You will learn more about responsive web pages in a later chapter.</p>
  </div>
</div>

<div class="footer">
  <p>Footer Text</p>
</div>

</body>
</html>
```

All CSS Float Properties

Property	Description
box-sizing	Defines how the width and height of an element are calculated: should they include padding and borders, or not
clear	Specifies what elements can float beside the cleared element and on which side

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 overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's 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;
}
```

Using inline-block to Create Navigation Links

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.nav {
   background-color: yellow;
   list-style-type: none;
   text-align: center;
   margin: 0;
   padding: 0;
}

.nav li {
   display: inline-block;
   font-size: 20px;
   padding: 20px;
}
</style>
</head>
<body>

<h1>Horizontal Navigation Links</h1>
<p>By default, list items are displayed vertically. In this example we use display: inline-block
to display them horizontally (side by side).</p>
<p>Note: If you resize the browser window, the links will automatically break when it
becomes too crowded.</p>

<ul class="nav">
  <li><a href="#home">Home</a></li>
  <li><a href="#about">About Us</a></li>
  <li><a href="#clients">Our Clients</a></li>
  <li><a href="#contact">Contact Us</a></li>
</ul>

</body>
</html>
```

CSS Layout - Horizontal & Vertical Align

Center Align Elements

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

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

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

Example

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

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

Center Align Text

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

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
   text-align: center;
   border: 3px solid green;
}
</style>
</head>
<body>

<h2>Center Text</h2>

<div class="center">
  <p>This text is centered.</p>
</div>

</body>
</html>
```

Center an Image

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

Example

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

Left and Right Align - Using position

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

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

Example

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

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

Left and Right Align - Using float

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

Example

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

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

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

Example

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

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:

Example

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

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

Example

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

Center Vertically - Using line-height

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

Example

```
.center {
   line-height: 200px;
   height: 200px;
   border: 3px solid green;
   text-align: center;
}

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

Center Vertically - Using position & transform

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
   height: 200px;
   position: relative;
   border: 3px solid green;
}

.center p {
   margin: 0;
   position: absolute;
   top: 50%;
   left: 50%;
   -ms-transform: translate(-50%, -50%);
   transform: translate(-50%, -50%);
}
</style>
</head>
<body>
```

```
<h2>Centering</h2>
<p>In this example, we use positioning and the transform property to vertically and
horizontally center the div element:</p>

<div class="center">
  <p>I am vertically and horizontally centered.</p>
</div>

<p>Note: The transform property is not supported in IE8 and earlier versions.</p>

</body>
</html>
```

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 <div> elements:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div p {
   background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
```

```
</html>
```

Child Selector

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

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

Example

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

Adjacent Sibling Selector

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

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

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

Example

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

General Sibling Selector

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

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

Example

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

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.
For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {
  Property:value;
}
```

Anchor Pseudo-classes
Links can be displayed in different ways:

Example

```
/* unvisited link */
a:link {
   color: #FF0000;
}

/* visited link */
a:visited {
   color: #00FF00;
}

/* mouse over link */
a:hover {
   color: #FF00FF;
}

/* selected link */
a:active {
   color: #0000FF;
}
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

When you hover over the link in the example, it will change color:
Example

```
a.highlight:hover {
   color: #ff0000;
}
```

Hover on <div>

An example of using the :hover pseudo-class on a <div> element:
Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   background-color: green;
   color: white;
   padding: 25px;
   text-align: center;
}
```

```
div:hover {
   background-color: blue;
}
</style>
</head>
<body>

<p>Mouse over the div element below to change its background color:</p>

<div>Mouse Over Me</div>

</body>
</html>
```

Simple Tooltip Hover

Hover over a <div> element to show a <p> element (like a tooltip):

Hover over me to show the <p> element.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   display: none;
   background-color: yellow;
   padding: 20px;
}

div:hover p {
   display: block;
}
</style>
</head>
<body>

<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>

</body>
</html>
```

CSS - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

```
p:first-child {
    color: blue;
}
```

Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p i:first-child {
    color: blue;
}
</style>
</head>
<body>

<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p>I am a <i>strong</i> person. I am a <i>strong</i> person.</p>
<p><b>Note:</b> For :first-child to work in IE8 and earlier, a DOCTYPE must be
declared.</p>

</body>
</html>
```

Result:

I am a ***strong*** person. I am a *strong* person.
I am a ***strong*** person. I am a *strong* person.

Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

Example

```
p:first-child i {
    color: blue;
}
```

Result:

I am a ***strong*** person. I am a ***strong*** person.
I am a *strong* person. I am a *strong* person.

CSS - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.

In the example below, :lang defines the quotation marks for <q> elements with lang="no":

Example

```
<!DOCTYPE html>
<html>
```

```
<head>
<style>
q:lang(no) {
   quotes: "~" "~";
}
</style>
</head>
<body>

<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
<p>In this example, :lang defines the quotation marks for q elements with lang="no":</p>
<p><b>Note:</b> IE8 supports the :lang pseudo class only if a !DOCTYPE is specified.</p>

</body>
</html>
```

Add different styles to hyperlinks

```
<!DOCTYPE html>
<html>
<head>
<style>
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
</head>
<body>

<p>Mouse over the links and watch them change layout:</p>

<p><b><a class="one" href="default.asp" target="_blank">This link changes
color</a></b></p>
<p><b><a class="two" href="default.asp" target="_blank">This link changes
font-size</a></b></p>
<p><b><a class="three" href="default.asp" target="_blank">This link changes
```

```
background-color</a></b></p>
<p><b><a class="four" href="default.asp" target="_blank">This link changes
font-family</a></b></p>
<p><b><a class="five" href="default.asp" target="_blank">This link changes
text-decoration</a></b></p>

</body>
</html>
```

Use of :focus

This example demonstrates how to use the :focus pseudo-class.

```
<!DOCTYPE html>
<html>
<head>
<style>
input:focus {
   background-color: yellow;
}
</style>
</head>
<body>

<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
</form>

<p><b>Note:</b> IE8 supports the :focus pseudo-class only if a !DOCTYPE is
specified.</p>

</body>
</html>
```

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element

:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

CSS Pseudo-elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.
For example, it can be used to:

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

Syntax
The syntax of pseudo-elements:

```
selector::pseudo-element {
  property:value;
}
```

Notice the double colon notation - ::first-line versus :first-line

The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between pseudo-classes and pseudo-elements.

The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.

For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements.

The ::first-line Pseudo-element

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

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

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

Note: The ::first-line pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-line` pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

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

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

Example

```
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}
```

Note: The `::first-letter` pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

Example

```
p.intro::first-letter {
    color: #ff0000;
    font-size:200%;
}
```

The example above will display the first letter of paragraphs with class="intro", in red and in a larger size.

Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}
```

```
p::first-line {
   color: #0000ff;
   font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can combine the ::first-letter and ::first-line pseudo-elements to add a special effect to the first letter and the first line of a text!</p>

</body>
</html>
```

CSS - The ::before Pseudo-element

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

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

Example

```
h1::before {
   content: url(smiley.gif);
}
```

CSS - The ::after Pseudo-element

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

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

Example

```
h1::after {
   content: url(smiley.gif);
}
```

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

The following example makes the selected text red on a yellow background:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
::-moz-selection { /* Code for Firefox */
   color: red;
```

```
  background: yellow;
}

::selection {
  color: red;
  background: yellow;
}
</style>
</head>
<body>

<h1>Select some text on this page:</h1>

<p>This is a paragraph.</p>
<div>This is some text in a div element.</div>
```

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element

:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

CSS Opacity / Transparency

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

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 */
}
```

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 */
}
```

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:

Northern Lights Mountains Italy

Example

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

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   background-color: #4CAF50;
   padding: 10px;
}

div.first {
   opacity: 0.1;
   filter: alpha(opacity=10); /* For IE8 and earlier */
}

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

div.third {
   opacity: 0.6;
```

```
  filter: alpha(opacity=60); /* For IE8 and earlier */
}
</style>
</head>
<body>

<h1>Transparent Box</h1>
<p>When using the opacity property to add transparency to the background of
an element, all of its child elements become transparent as well. This can make
the text inside a fully transparent element hard to read:</p>

<div class="first"><p>opacity 0.1</p></div>
<div class="second"><p>opacity 0.3</p></div>
<div class="third"><p>opacity 0.6</p></div>
<div><p>opacity 1 (default)</p></div>

</body>
</html>
```

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:

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).

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background: rgb(76, 175, 80);
  padding: 10px;
}

div.first {
  background: rgba(76, 175, 80, 0.1);
}

div.second {
  background: rgba(76, 175, 80, 0.3);
}

div.third {
  background: rgba(76, 175, 80, 0.6);
}
</style>
</head>
<body>
```

```
<h1>Transparent Box</h1>
<p>With opacity:</p>
<div style="opacity:0.1;"><p>10% opacity</p></div>
<div style="opacity:0.3;"><p>30% opacity</p></div>
<div style="opacity:0.6;"><p>60% opacity</p></div>
<div><p>opacity 1</p></div>

<p>With RGBA color values:</p>
<div class="first"><p>10% opacity</p></div>
<div class="second"><p>30% opacity</p></div>
<div class="third"><p>60% opacity</p></div>
<div><p>default</p></div>

<p>Notice how the text gets transparent as well as the background color when
using the opacity property.</p>

</body>
</html>
```

Text in 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>
```

First, we create a <div> element (class="background") with a background image, and a border. Then we create another <div> (class="transbox") inside the first <div>. The <div class="transbox"> have a background color, and a border - the div is transparent. Inside the transparent <div>, we add some text inside a <p> element.

CSS Navigation Bar

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>
```

Another example

```
<!DOCTYPE html>
<html>
<body>
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
<p>Note: We use href="#" for test links. In a real web site this would be URLs.</p>
</body>
</html>
```

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

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

Example explained:
list-style-type: none; - Removes the bullets. A navigation bar does not need list markers
Set margin: 0; and padding: 0; to remove browser default settings
The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

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

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
}

li a {
   display: block;
   width: 60px;
   background-color: #dddddd;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<p>A background color is added to the links to show the link area.</p>
<p>Notice that the whole link area is clickable, not just the text.</p>

</body>
</html>
```

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;
}
```

Vertical Navigation Bar Examples

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

Home
News
Contact
About

Example

```
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
   width: 200px;
   background-color: #f1f1f1;
}

li a {
   display: block;
   color: #000;
   padding: 8px 16px;
   text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
   background-color: #555;
   color: white;
}
```

Active/Current Navigation Link

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

Home
News
Contact

About

Example

```
.active {
   background-color: #4CAF50;
   color: white;
}
```

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
   width: 200px;
   background-color: #f1f1f1;
   border: 1px solid #555;
}

li a {
   display: block;
   color: #000;
   padding: 8px 16px;
   text-decoration: none;
}

li {
   text-align: center;
   border-bottom: 1px solid #555;
}

li:last-child {
   border-bottom: none;
}

li a.active {
   background-color: #4CAF50;
   color: white;
}

li a:hover:not(.active) {
   background-color: #555;
   color: white;
}
```

```
</style>
</head>
<body>

<h2>Vertical Navigation Bar</h2>
<p>In this example, we center the navigation links and add a border to the
navigation bar.</p>

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

</body>
</html>
```

Full-height Fixed Vertical Navbar

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

Example

```
<style>
body {
   margin: 0;
}
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 */
}
li a {
   display: block;
   color: #000;
   padding: 8px 16px;
   text-decoration: none;
}
li a.active {
   background-color: #4CAF50;
   color: white;
}
li a:hover:not(.active) {
   background-color: #555;
   color: white;
}
</style>
```

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

```
<html>
<head>
<style>
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
}

li {
   display: inline;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

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

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
   overflow: hidden;
}

li {
   float: left;
}

li a {
   display: block;
   padding: 8px;
   background-color: #dddddd;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<p><b>Note:</b> If a !DOCTYPE is not specified, floating items can
produce unexpected results.</p>
<p>A background color is added to the links to show the link area. The
whole link area is clickable, not just the text.</p>
<p><b>Note:</b> overflow:hidden is added to the ul element to
prevent li elements from going outside of the list.</p>

</body>
</html>
```

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;
}
```

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:

HomeNewsContactAbout

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;
}
```

Active/Current Navigation Link

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

HomeNewsContactAbout

```
<!DOCTYPE html>
<html>
<head>
<style>
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;
}

li a:hover:not(.active) {
    background-color: #111;
}

.active {
    background-color: #CAF50;
}
</style>
</head>
<body>

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

</body>
</html>
```

Right-Align Links

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

HomeNewsContactAbout

Example

```
<ul>
```

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

Border Dividers

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

Example

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

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

Fixed Navigation Bar

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

Fixed Top

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

Fixed Bottom

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

Gray Horizontal Navbar

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

Example

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

li a {
   color: #666;
}
```

Sticky Navbar

Use position: sticky; to <li> 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;
}
```

More Examples

Responsive Topnav

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

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {margin: 0;}

ul.topnav {
   list-style-type: none;
   margin: 0;
   padding: 0;
   overflow: hidden;
   background-color: #333;
}

ul.topnav li {float: left;}

ul.topnav li a {
   display: block;
   color: white;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
}

ul.topnav li a:hover:not(.active) {background-color: #111;}

ul.topnav li a.active {background-color: #4CAF50;}
ul.topnav li.right {float: right;}

@media screen and (max-width: 600px){
    ul.topnav li.right,
    ul.topnav li {float: none;}
}
</style>
</head>
<body>

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

<div style="padding:0 16px;">
  <h2>Responsive Topnav Example</h2>
  <p>This example use media queries to stack the topnav vertically when the screen size is 600px or less.</p>
  <p>You will learn more about media queries and responsive web design later in our CSS Tutorial.</p>
  <h4>Resize the browser window to see the effect.</h4>
</div>

</body>
</html>
```

Responsive Sidenav

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

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {margin: 0;}

ul.sidenav {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 25%;
    background-color: #f1f1f1;
    position: fixed;
    height: 100%;
    overflow: auto;
}

ul.sidenav li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}

ul.sidenav li a.active {
    background-color: #4CAF50;
    color: white;
}

ul.sidenav li a:hover:not(.active) {
    background-color: #555;
    color: white;
}

div.content {
    margin-left: 25%;
    padding: 1px 16px;
    height: 1000px;
}

@media screen and (max-width: 900px) {
    ul.sidenav {
        width: 100%;
        height: auto;
        position: relative;
    }
    ul.sidenav li a {
        float: left;
        padding: 15px;
    }
    div.content {margin-left: 0;}
}

@media screen and (max-width: 400px) {
    ul.sidenav li a {
        text-align: center;
        float: none;
    }
}
</style>
```

```
</head>
<body>

<ul class="sidenav">
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<div class="content">
  <h2>Responsive Sidenav Example</h2>
  <p>This example use media queries to transform the sidenav to a top navigation bar when the screen size is 900px or less.</p>
  <p>We have also added a media query for screens that are 400px or less, which will vertically stack and center the navigation links.</p>
  <p>You will learn more about media queries and responsive web design later in our CSS Tutorial.</p>
  <h3>Resize the browser window to see the effect.</h3>
</div>

</body>
</html>
```

Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
}

li {
    float: left;
}

li a, .dropbtn {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

li a:hover, .dropdown:hover .dropbtn {
    background-color: red;
}

li.dropdown {
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}

.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
}

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

.dropdown:hover .dropdown-content {
    display: block;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Dropdown</a>
    <div class="dropdown-content">
      <a href="#">Link 1</a>
      <a href="#">Link 2</a>
      <a href="#">Link 3</a>
    </div>
  </li>
</ul>

<h3>Dropdown Menu inside a Navigation Bar</h3>
<p>Hover over the "Dropdown" link to see the dropdown menu.</p>

</body>
</html>
```

CSS Dropdowns

Create a hoverable dropdown with CSS.

Basic Dropdown

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>
</head>
<body>
```

```
<h2>Hoverable Dropdown</h2>
<p>Move the mouse over the text below to open the dropdown content.</p>

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

</body>
</html>
```

Example Explained

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

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

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

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

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

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

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

Dropdown Menu

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

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>
```

Right-aligned Dropdown Content

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;
}
```

More Examples

Dropdown Image

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

```
<!DOCTYPE html>
<html>
<head>
<style>
.dropdown {
    position: relative;
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}

.dropdown:hover .dropdown-content {
    display: block;
}
```

```
.desc {
    padding: 15px;
    text-align: center;
}
</style>
</head>
<body>

<h2>Dropdown Image</h2>
<p>Move the mouse over the image below to open the dropdown content.</p>

<div class="dropdown">
  <img src="img_5terre.jpg" alt="Cinque Terre" width="100" height="50">
  <div class="dropdown-content">
    <img src="img_5terre.jpg" alt="Cinque Terre" width="300" height="200">
    <div class="desc">Beautiful Cinque Terre</div>
  </div>
</div>

</body>
</html>
```

Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
}

li {
    float: left;
}

li a, .dropbtn {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

li a:hover, .dropdown:hover .dropbtn {
    background-color: red;
}

li.dropdown {
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}

.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
}

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

.dropdown:hover .dropdown-content {
    display: block;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Dropdown</a>
    <div class="dropdown-content">
      <a href="#">Link 1</a>
      <a href="#">Link 2</a>
      <a href="#">Link 3</a>
    </div>
  </li>
</ul>

<h3>Dropdown Menu inside a Navigation Bar</h3>
<p>Hover over the "Dropdown" link to see the dropdown menu.</p>

</body>
</html>
```

CSS Image Gallery

CSS can be used to create an 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="fjords.jpg">
    <img src="5terre.jpg" alt="Cinque Terre" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="forest.jpg">
    <img src="forest.jpg" alt="Forest" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="lights.jpg">
    <img src="lights.jpg" alt="Northern Lights" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="mountains.jpg">
    <img src="mountains.jpg" alt="Mountains" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

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 smartphones.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.gallery {
    border: 1px solid #ccc;
}

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

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

div.desc {
    padding: 15px;
    text-align: center;
}

* {
    box-sizing: border-box;
}

.responsive {
    padding: 0 6px;
    float: left;
    width: 24.99999%;
}

@media only screen and (max-width: 700px) {
    .responsive {
        width: 49.99999%;
        margin: 6px 0;
    }
}

@media only screen and (max-width: 500px) {
    .responsive {
        width: 100%;
    }
}

.clearfix:after {
    content: "";
    display: table;
    clear: both;
}
</style>
</head>
<body>

<h2>Responsive Image Gallery</h2>
<h4>Resize the browser window to see the effect.</h4>

<div class="responsive">
  <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>


<div class="responsive">
  <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>

<div class="responsive">
  <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>

<div class="responsive">
  <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>
</div>

<div class="clearfix"></div>

<div style="padding:6px;">
  <p>This example use media queries to re-arrange the images on different
screen sizes: for screens larger than 700px wide, it will show four images side by
side, for screens smaller than 700px, it will show two images side by side. For
screens smaller than 500px, the images will stack vertically (100%).</p>
  <p>You will learn more about media queries and responsive web design later
in our CSS Tutorial.</p>
```

</div>

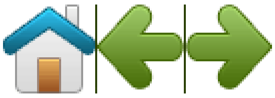
</body>
</html>

CSS 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

```
<!DOCTYPE html>
<html>
<head>
<style>
#home {
    width: 46px;
    height: 44px;
    background: url(img_navsprites.gif) 0 0;
}

#next {
    width: 43px;
    height: 44px;
    background: url(img_navsprites.gif) -91px 0;
}
</style>
</head>
<body>

<img id="home" src="img_trans.gif"><br><br>
<img id="next" src="img_trans.gif">

</body>
</html>
```

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

```
<!DOCTYPE html>
<html>
<head>
<style>
#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;
}
</style>
</head>
<body>

<ul id="navlist">
  <li id="home"><a href="default.asp"></a></li>
  <li id="prev"><a href="css_intro.asp"></a></li>
  <li id="next"><a href="css_syntax.asp"></a></li>
</ul>

</body>
</html>
```

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:

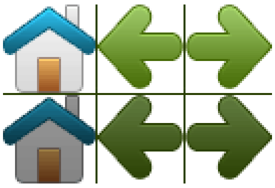
- #home {left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home {background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev {left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + 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;
}
```

Example explained:

#home a:hover {background: transparent url('img_navsprites_hover.gif') 0 -45px;} - For all three hover images we specify the same background position, only 45px further down

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

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
a[target] {
   background-color: yellow;
}
</style>
</head>
<body>

<p>The links with a target attribute gets a yellow background:</p>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

<p><b>Note:</b> For [<i>attribute</i>] to work in IE8 and earlier, a DOCTYPE must be declared.</p>

</body>
</html>
```

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

```
<!DOCTYPE html>
<html>
<head>
<style>
a[target=_top] {
   background-color: yellow;
}
</style>
</head>
<body>

<p>The link with target="_blank" gets a yellow background:</p>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

<p><b>Note:</b> For [<i>attribute</i>] to work in IE8 and earlier, a DOCTYPE must be declared.</p>
</body>
</html>
```

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;
}
```

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;
}
```

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;
}
```

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;
}
```

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;
}
```

Styling Forms

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

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type="text"] {
   width: 150px;
   display: block;
   margin-bottom: 10px;
   background-color: yellow;
}

input[type="button"] {
   width: 120px;
   margin-left: 35px;
   display: block;
}
</style>
</head>
<body>

<form name="input" action="" method="get">
  Firstname:<input type="text" name="Name" value="Peter" size="20">
  Lastname:<input type="text" name="Name" value="Griffin" size="20">
  <input type="button" value="Example Button">
</form>

</body>
</html>
```

CSS Forms

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

Styling Input Fields

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

Example

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

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

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

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
   width: 100%;
   padding: 12px 20px;
   margin: 8px 0;
   box-sizing: border-box;
}
</style>
</head>
<body>

<p>Padded text fields:</p>

<form>
  <label for="fname">First Name</label>
  <input type="text" id="fname" name="fname">
  <label for="lname">Last Name</label>
  <input type="text" id="lname" name="lname">
</form>

</body>
</html>
```

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.

Bordered Inputs

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

Example

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

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

Example

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

Colored Inputs

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

Example

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

Focused Inputs

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

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

Example

```
<style>
input[type=text] {
   width: 100%;
   padding: 12px 20px;
   margin: 8px 0;
   box-sizing: border-box;
   border: 1px solid #555;
   outline: none;

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

Example

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

Input with icon/image

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

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
   width: 100%;
   box-sizing: border-box;
   border: 2px solid #ccc;
   border-radius: 4px;
   font-size: 16px;
   background-color: white;
   background-image: url('searchicon.png');
   background-position: 10px 10px;
   background-repeat: no-repeat;
   padding: 12px 20px 12px 40px;
}
</style>
</head>
<body>

<p>Input with icon:</p>

<form>
  <input type="text" name="search" placeholder="Search..">
</form>

</body>
</html>
```

Animated Search Input

In this example we use the CSS transition property to animate the width of the search input when it gets focus.

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
   width: 130px;
   box-sizing: border-box;
   border: 2px solid #ccc;
```

```
    border-radius: 4px;
    font-size: 16px;
    background-color: white;
    background-image: url('searchicon.png');
    background-position: 10px 10px;
    background-repeat: no-repeat;
    padding: 12px 20px 12px 40px;
    -webkit-transition: width 0.4s ease-in-out;
    transition: width 0.4s ease-in-out;
}

input[type=text]:focus {
    width: 100%;
}
</style>
</head>
<body>

<p>Animated search input:</p>

<form>
  <input type="text" name="search" placeholder="Search..">
</form>

</body>
</html>
```

Styling Textareas

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

```
<!DOCTYPE html>
<html>
<head>
<style>
textarea {
    width: 100%;
    height: 150px;
    padding: 12px 20px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    background-color: #f8f8f8;
    font-size: 16px;
    resize: none;
}
</style>
</head>
<body>

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

```
<form>
  <textarea>Some text...</textarea>
</form>

</body>
</html>
```

Styling Select Menus

```
<!DOCTYPE html>
<html>
<head>
<style>
select {
    width: 100%;
    padding: 16px 20px;
    border: none;
    border-radius: 4px;
    background-color: #f1f1f1;
}
</style>
</head>
<body>

<p>A styled select menu.</p>

<form>
  <select id="country" name="country">
    <option value="au">Australia</option>
    <option value="ca">Canada</option>
    <option value="usa">USA</option>
  </select>
</form>

</body>
</html>
```

Styling Input Buttons

```
<!DOCTYPE html>
<html>
<head>
<style>
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;
}
</style>
</head>
<body>

<p>Styled input buttons.</p>

<input type="button" value="Button">
<input type="reset" value="Reset">
<input type="submit" value="Submit">

</body>
</html>
```

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
   box-sizing: border-box;
}

input[type=text], select, textarea {
   width: 100%;
   padding: 12px;
   border: 1px solid #ccc;
   border-radius: 4px;
   resize: vertical;
}

label {
   padding: 12px 12px 12px 0;
   display: inline-block;
}

input[type=submit] {
   background-color: #4CAF50;
   color: white;
   padding: 12px 20px;
   border: none;
   border-radius: 4px;
   cursor: pointer;
   float: right;
}

input[type=submit]:hover {
    background-color: #45a049;
}

.container {
    border-radius: 5px;
    background-color: #f2f2f2;
    padding: 20px;
}

.col-25 {
    float: left;
    width: 25%;
    margin-top: 6px;
}

.col-75 {
    float: left;
    width: 75%;
    margin-top: 6px;
}

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

/* Responsive layout - when the screen is less than 600px wide, make the two
columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
    .col-25, .col-75, input[type=submit] {
        width: 100%;
        margin-top: 0;
    }
}
</style>
</head>
<body>

<h2>Responsive Form</h2>
<p>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.</p>

<div class="container">
  <form action="/action_page.php">
    <div class="row">
```

```
      <div class="col-25">
        <label for="fname">First Name</label>
      </div>
      <div class="col-75">
        <input type="text" id="fname" name="firstname" placeholder="Your
name..">
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="lname">Last Name</label>
      </div>
      <div class="col-75">
        <input type="text" id="lname" name="lastname" placeholder="Your last
name..">
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="country">Country</label>
      </div>
      <div class="col-75">
        <select id="country" name="country">
          <option value="australia">Australia</option>
          <option value="canada">Canada</option>
          <option value="usa">USA</option>
        </select>
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="subject">Subject</label>
      </div>
      <div class="col-75">
        <textarea id="subject" name="subject" placeholder="Write something.."
style="height:200px"></textarea>
      </div>
    </div>
    <div class="row">
      <input type="submit" value="Submit">
    </div>
  </form>
</div>

</body>
</html>
```

CSS Counters

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    counter-reset: section;
}

h2::before {
    counter-increment: section;
    content: "Section " counter(section) ": ";
}
</style>
</head>
<body>

<h1>Using CSS Counters:</h1>
<h2>HTML Tutorial</h2>
<h2>CSS Tutorial</h2>
<h2>JavaScript Tutorial</h2>

<p><b>Note:</b> IE8 supports these properties only if a !DOCTYPE is
specified.</p>

</body>
</html>
```

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>":

```
<head>
<style>
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) " ";
}
</style>
</head>
```

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

```
<!DOCTYPE html>
<html>
<head>
<style>
ol {
    counter-reset: section;
    list-style-type: none;
}

li::before {
    counter-increment: section;
    content: counters(section,".") " ";
}
</style>
</head>
<body>

<ol>
  <li>item</li>
  <li>item
    <ol>
      <li>item</li>
      <li>item</li>
      <li>item
        <ol>
```

```
          <li>item</li>
          <li>item</li>
          <li>item</li>
        </ol>
      </li>
      <li>item</li>
    </ol>
  </li>
  <li>item</li>
  <li>item</li>
</ol>

<ol>
  <li>item</li>
  <li>item</li>
</ol>

<p><b>Note:</b> IE8 supports these properties only if a !DOCTYPE is
specified.</p>

</body>
</html>
```

CSS Counter Properties

Property	Description
content	Used with the ::before and ::after pseudo-elements, to insert generated content
counter-increment	Increments one or more counter values
counter-reset	Creates or resets one or more counters