CSS

What is CSS?

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

CSS Demo - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" .

Why Use CSS?

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

CSS Solved a Big Problem

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

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

<h1>This is a heading</h1>

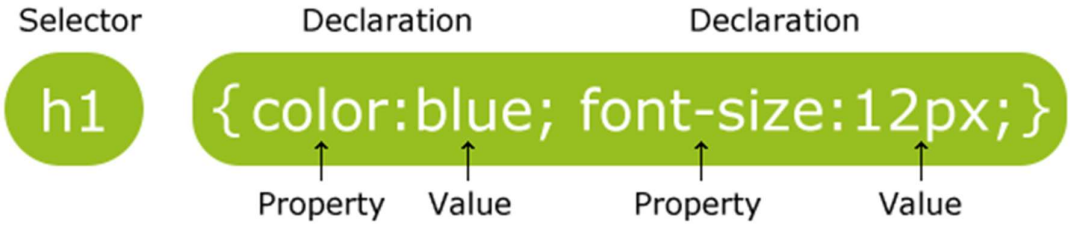
This is a paragraph.

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

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

CSS Syntax

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



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

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

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

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

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

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

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

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

 **Note:** A class name cannot start with a number!

Grouping Selectors

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

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

h2 {
    text-align: center;
    color: red;
}

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

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

To group selectors, separate each selector with a comma.

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;
}
```

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

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

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

Example

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

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

Here is how the "myStyle.css" looks:

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

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

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

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

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?

Generally speaking we can say that all the styles 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 (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

CSS Colors

Colors are displayed combining RED, GREEN, and BLUE light.

Colors in CSS are most often specified by:

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

Color Names

Colors set by using color names:

Example

Color	Name
	Red
	Green
	Blue
	Orange
	Yellow
	Cyan
	Black

RGB (Red, Green, Blue)

RGB color values can be specified using this formula: rgb(red, green, blue).

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

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

Hexadecimal Colors

RGB values can also be specified using **hexadecimal** color values in the form: *#RRGGBB*, where RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and FF (same as decimal 0-255).

For example, `#FF0000` is displayed as red, because red is set to its highest value (FF) and the others are set to the lowest value (00). **Note:** HEX values are case-insensitive: "#ff0000" is the same as "FF0000".

CSS Backgrounds

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

CSS background properties:

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

Background Color

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

The background color of a page is set like this:

Example

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

With CSS, a color is most often specified by:

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

Look at CSS Color Values for a complete list of possible color values.

In the example below, the <h1>, <p>, and 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

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

Below is an example of a bad combination of text and background image. The text is hardly readable:

Example

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

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

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

Note: 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

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

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

- `background-color`

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

CSS Borders

CSS Border Properties

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

This element has a groove border that is 15px wide and green.

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;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

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

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

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

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

No border.

A hidden border.

A mixed border.

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).

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

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

Example

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

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;
}
```

CSS Margins

CSS Margin Properties

The CSS `margin` properties are used to generate space around elements.

The margin properties set the size of the white space OUTSIDE the border.

CSS Margins

The CSS margin properties set the size of the white space OUTSIDE the border.

Note: The margins are completely transparent - and cannot have a background color!

With CSS, you have full control over the margins. There are CSS 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

Note: It is also possible to use negative values for margins; to overlap content.

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;
}
```

The following example lets the left margin be inherited from the parent element:

Example

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

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

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`

Example

```
p {
    margin: 100px 150px 100px 80px;
}
```

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

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

If the `margin` property has two values:

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

If the `margin` property has one value:

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

Use of 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;
}
```

CSS Padding

CSS Padding Properties

The CSS `padding` properties are used to generate space around content.

The padding properties set the size of the white space between the element content and the element border.

CSS Padding

The CSS padding properties define the white space between the element content and the element border.

The padding clears an area around the content (inside the border) of an element.

Note: The padding is affected by the background color of the element!

With CSS, you have full control over the padding. There are CSS 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

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

Example

```
p {
    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`

Example

```
p {
    padding: 50px 30px 50px 80px;
}
```

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

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

If the `padding` property has two values:

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

If the `padding` property has one value:

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

CSS Height and Width Dimensions

CSS Dimension Properties

The CSS dimension properties allow you to control the height and width of an element.

This element has a width of 100%.

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.

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!

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

Example

```
div {
    width: 500px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

Setting max-width

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

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

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

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

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

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

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

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;
    border: 3px solid #73AD21;
}
```

CSS Text

Text Color

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

With CSS, a color is most often specified by:

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

Look at CSS Color Values for a complete list of possible color values.

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

Example

```
body {
    color: blue;
}

h1 {
    color: green;
}
```

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

Text Alignment

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

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

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

Example

```
h1 {
    text-align: center;
}

h2 {
    text-align: left;
}

h3 {
    text-align: right;
}
```

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

Example

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

Text Decoration

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

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

Example

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

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

Example

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

h2 {
    text-decoration: line-through;
}

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

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

Text Transformation

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

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

Example

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

p.lowercase {
    text-transform: lowercase;
}

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

Text Indentation

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

Example

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

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

```
div {
    direction: rtl;
}
```

Word Spacing

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

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

Example

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

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

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

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

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

Font Style

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

This property has three values:

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

Example

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

p.italic {
    font-style: italic;
}

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

Font Size

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

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

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

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

Absolute size:

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

Relative size:

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

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

Set Font Size With Pixels

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

Example

```
h1 {
    font-size: 40px;
}

h2 {
    font-size: 30px;
}

p {
    font-size: 14px;
}
```

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

Set Font Size With Em

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

The em size unit is recommended by the W3C.

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

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {
    font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
    font-size: 1.875em; /* 30px/16=1.875em */
}

p {
    font-size: 0.875em; /* 14px/16=0.875em */
}
```

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

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

Use a Combination of Percent and Em

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

Example

```
body {
    font-size: 100%;
}

h1 {
    font-size: 2.5em;
}

h2 {
    font-size: 1.875em;
}

p {
```

```
    font-size: 0.875em;
}
```

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

Font Weight

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

Example

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

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

Font Variant

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

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

Example

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

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

CSS Links

With CSS, links can be styled in different ways.

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;
}
```

Advanced - Link Buttons

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

Example

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

a:hover, a:active {
    background-color: red;
}
```

CSS Lists

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

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

Different List Item Markers

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

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

Example

```
ul.a {
    list-style-type: circle;
}

ul.b {
    list-style-type: square;
}

ol.c {
    list-style-type: upper-roman;
}

ol.d {
    list-style-type: lower-alpha;
}
```

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 whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {
    list-style-position: inside;
}
```

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;
}
```

Result:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

CSS Tables

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

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

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;
}
```

Try it yourself »

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

Example

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

Table Width and Height

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

The example below sets the width of the table to 100%, and the height of the < th> elements to 50px:

Example

```
table {
    width: 100%;
}

th {
    height: 50px;
}
```

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

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

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

Example

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

Hoverable Table

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

First Name	Last Name	Savings
Peter	Griffin	$100

Lois	Griffin	$150
Joe	Swanson	$300

Example

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

Table Color

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

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

Example

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

CSS Box Model

The CSS Box Model

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

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

The image below illustrates the box model:

Explanation of the different parts:

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

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

Example

```
div {
    width: 300px;
    padding: 25px;
    border: 25px solid navy;
    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 element to have a total width of 350px:

Example

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

Here is the math:

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 Layout - The display Property

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

The display Property

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

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

Click to show panel

Block-level Elements

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

The element is a block-level element.

Examples of block-level elements:

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

Inline Elements

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

This is an inline element inside a paragraph.

Examples of inline elements:

-
- <a>
- <img>

Display: none;

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

The `<script>` element use `display: none;` as its 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;
}
```

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

The following example displays elements as block elements:

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

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;
}
```

CSS Layout - The position Property

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

The position Property

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

There are four different position values:

- `static`
- `relative`
- `fixed`
- `absolute`

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

position: static;

HTML elements are positioned static by default.

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

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

This element has position: static;

Here is the CSS that is used:

Example

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

---

position: relative;

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

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

This element has position: relative;

Here is the CSS that is used:

Example

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

---

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;
}
```

This element has position: fixed;

---

position: absolute;

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

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

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

Here is a simple example:

This element has position: relative;

This element has position: absolute;

Here is the CSS that is used:

Example

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

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

Overlapping Elements

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:

This is a heading

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

Example

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

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

Bottom Left

Top Left

Top Right

Bottom Right

Centered

CSS Layout- float and clear

The `float` property specifies whether or not an element should float.

The `clear` property is used to control the behavior of floating elements.

The float Property

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

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

Example

```
img {
    float: right;
    margin: 0 0 10px 10px;
}
```

The clear Property

The `clear` property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the `clear` property.

The `clear` property specifies on which sides of an element floating elements are not allowed to float:

Example

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

The clearfix Hack - overflow: auto;

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:

Example

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

Web Layout Example

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

Example

```
div {
    border: 3px solid blue;
}
```

```
.clearfix {
    overflow: auto;
}

nav {
    float: left;
    width: 200px;
    border: 3px solid #73AD21;
}

section {
    margin-left: 206px;
    border: 3px solid red;
}
```

CSS Layout- inline-block

The inline-block Value

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the `float` property.

However, the `inline-block` value of the `display` property makes this even easier.

inline-block elements are like inline elements but they can have a width and a height.

Examples

The old way - using `float` (notice that we also need to specify a `clear` property for the element after the floating boxes):

Example

```
.floating-box {
    float: left;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}

.after-box {
    clear: left;
}
```

The same effect can be achieved by using the `inline-block` value of the `display` property (notice that no `clear` property is needed):

Example

```
.floating-box {
    display: inline-block;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}
```

CSS Layout- Horizontal Align

Center Align - Using margin

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Use `margin: auto;`, to horizontally center an element within 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: 60%;
    border: 3px solid #73AD21;
    padding: 10px;
}
```

Left and Right Align - Using position

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

Example

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

Tip: When aligning elements with `position`, always define `margin` and `padding` for the `<body>` element. This is to avoid visual differences in different browsers.

Example

```
body {
    margin: 0;
    padding: 0;
}

.container {
    position: relative;
    width: 100%;
}

.right {
    position: absolute;
    right: 0px;
    width: 300px;
    background-color: #b0e0e6;
}
```

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;
}
```

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

An example of using the `:hover` pseudo-class on a element:

```
Example
div:hover {
    background-color: blue;
}
```

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

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

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;
}
```

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

```
<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>
</body>
</html>
```

CSS Pseudo-elements

What are Pseudo-Elements?

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

For example, it can be used to:

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

Syntax

The syntax of pseudo-elements:

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

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;
}
```

Try it yourself »

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;
}
```

Try it yourself »

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%;
}
```

Try it yourself »

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

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

p::first-line {
    color: #0000ff;
    font-variant: small-caps;
}
```

Try it yourself »

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);
}
```

Try it yourself »

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);
}
```

Try it yourself »

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

```
::selection {
    color: red;
    background: yellow;
}
```

CSS Navigation Bar

Navigation Bars

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

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

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

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

A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

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

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

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

Example explained:

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

You can also set the width of <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:a

Example

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

li a {
    display: block;
    color: #000;
    padding: 8px 0 8px 16px;
    text-decoration: none;
}

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

Active/Current Navigation Link

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

Example

```
.active {
    background-color: #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:

Example

```
ul {
    border: 1px solid #555;
}

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

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

Full-height Fixed Vertical Navbar

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

Example

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

```
content */
}
```

Horizontal Navigation Bar

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

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code above:

Example

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

Example explained:

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

Floating List Items

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

Example

```
li {
    float: left;
}

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

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:

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:

Example

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

Right-Align Links

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

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;
}
```