UPCISS

CSS

Basic Tutorials

Video Tutorial

CSS Full Playlist Available on YouTube Channel **UPCISS**

Free Online Computer Classes on YouTube Channel **UPCISS**
www.youtube.com/upciss

Table of Contents

CSS Introduction

What is CSS?

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

Why Use CSS?

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

CSS Solved a Big Problem

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

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

<h1>This is a heading</h1>

This is a paragraph.

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

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

CSS removed the style formatting from the HTML page!

If you don't know what HTML is, we suggest that you read our HTML Tutorial.

CSS Saves a Lot of Work!

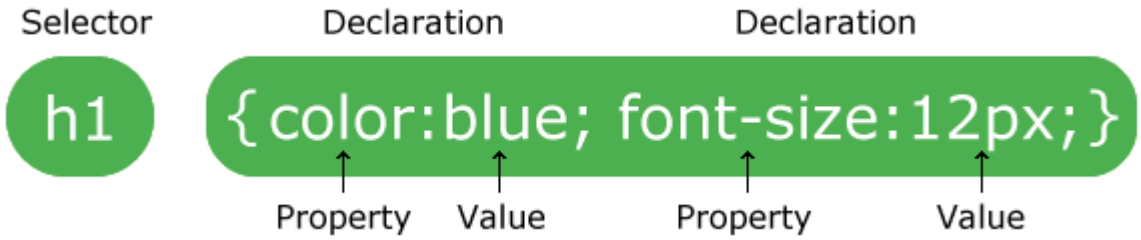
The style definitions are normally saved in external .css files.

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

CSS Syntax

CSS Syntax

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



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

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

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

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

Example

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

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

CSS Comments

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

Comments are ignored by browsers.

Example

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

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

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

CSS Selectors

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

The CSS element Selector

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

Example

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

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

The CSS id Selector

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

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

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

Example

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

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

Note: An id name cannot start with a number!

The CSS class Selector

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

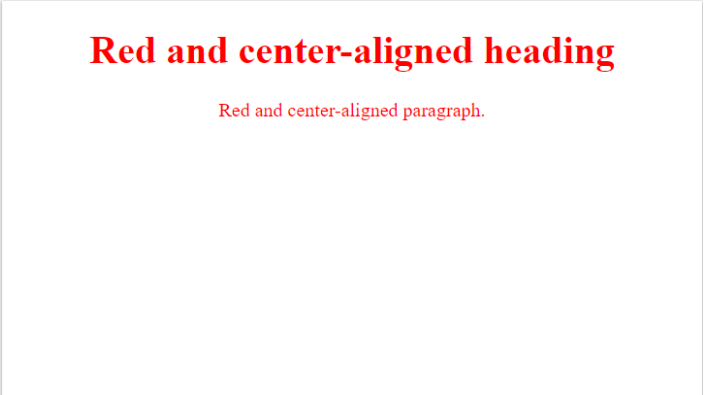
To select elements with a specific class, write a period (.) character, followed by the class name.

Example

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

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

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

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

</body>
</html>
```



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

Example

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

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

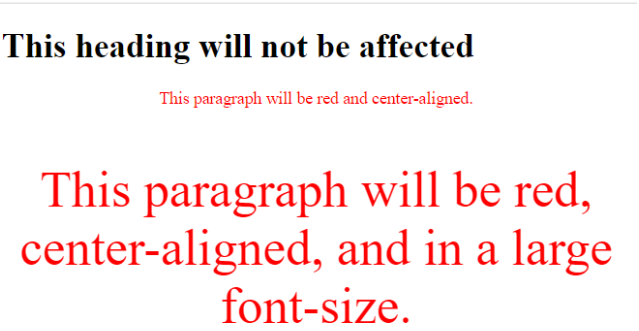
HTML elements can also refer to more than one class.

Example

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

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

```
<!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!

The CSS Universal Selector

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

Example

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

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

The CSS Grouping Selector

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

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

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

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

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

To group selectors, separate each selector with a comma.

Example

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

Types of CSS

Cascading Style Sheet (CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color,… etc property of elements on a web page.
There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

How To Add CSS

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

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

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

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

Example

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

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

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

</body>
</html>
```

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

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

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

"mystyle.css"

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

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

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 CSS

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

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

Example

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

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

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

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

</body>
</html>
```

Inline CSS

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

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

Example

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

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

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

</body>
</html>
```

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

Multiple Style Sheets

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

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

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

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

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

Example

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

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

Example

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

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

Cascading Order

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

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

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

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

CSS Comments

CSS comments are not displayed in the browser, but they can help document your source code. Comments are used to explain the code, and may help when you edit the source code at a later date. Comments are ignored by browsers.

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

HTML and CSS Comments

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

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

Example

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

<h2>My Heading</h2>

<!-- These paragraphs will be red -->
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>

</body>
</html>
```

CSS Colors

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

CSS Color Names

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

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

LightGray

Violet

CSS 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>
```

CSS 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>
```

CSS 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 Color Values

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

Same as color name "Tomato":

```
rgb(255, 99, 71)
#ff6347
```

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

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

```
rgba(255, 99, 71, 0.5)
hsla(9, 100%, 64%, 0.5)
```

Example

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

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

CSS background-image

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

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

Example

Set the background image for a page:

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

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

The background image can also be set for specific elements, like the <p> element:

Example

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

CSS background-repeat

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

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

Example

```
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;
}
```

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

CSS background-repeat: no-repeat

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

Example

Show the background image only once:

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

CSS background-position

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

Example

Position the background image in the top-right corner:

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

CSS background-attachment

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

Example

Specify that the background image should be fixed:

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

Tip: Specify that the background image should scroll with the rest of the page: set `background-attachment: scroll;`

CSS background - Shorthand property

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

You can use the shorthand property `background`:

Example

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

```
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

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

I have borders on all sides.

I have a red bottom border.

I have rounded borders.

I have a blue left border.

CSS Border Style

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

The following values are allowed:

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

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

Example

Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

CSS Border Width

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

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

Example

Demonstration of the different border widths:

```
p.one {
  border-style: solid;
  border-width: 5px;
}

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

p.three {
  border-style: dotted;
  border-width: 2px;
}

p.four {
  border-style: dotted;
  border-width: thick;
}
```

Specific Side Widths

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

Example

```
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}

p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}

p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and
35px left */
}
```

CSS Border Color

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

The color can be set by:

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

Specific Side Colors

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

Example

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

CSS Border - Individual Sides

From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.
In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

Example

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

CSS Border - Shorthand Property

Like you saw in the previous page, there are many properties to consider when dealing with borders.
To shorten the code, it is also possible to specify all the individual border properties in one property.

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

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

Example

```
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;
}
```

CSS Rounded Borders

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

Example

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

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.

CSS Margins

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

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

Margin - Individual Sides

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

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

All the margin properties can have the following values:

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

Tip: Negative values are allowed.

Example

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

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

Margin - Shorthand Property

Example

Use the margin shorthand property with four values:

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

CSS Padding

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

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

Padding - Individual Sides

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

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

All the padding properties can have the following values:

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

Note: Negative values are not allowed.

Example

Set different padding for all four sides of a element:

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

Padding - Shorthand Property

Example

Use the padding shorthand property with four values:

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

CSS Setting height and width

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

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

CSS height and width Values

The `height` and `width` properties may have the following values:

- `auto` - This is default. The browser calculates the height and width
- `length` - Defines the height/width in px, cm etc.
- `%` - Defines the height/width in percent of the containing block

Example

Set the height and width of a element:

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

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`.

Example

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

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

The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

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

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

Width and Height of an Element

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

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

Example

This element will have a total width of 350px:

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

Here is the calculation:

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

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

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

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

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

CSS Outline

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

CSS has the following outline properties:

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

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

CSS Outline Style

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

- `dotted` - Defines a dotted outline
- `dashed` - Defines a dashed outline
- `solid` - Defines a solid outline
- `double` - Defines a double outline
- `groove` - Defines a 3D grooved outline
- `ridge` - Defines a 3D ridged outline
- `inset` - Defines a 3D inset outline

- `outset` - Defines a 3D outset outline
- `none` - Defines no outline
- `hidden` - Defines a hidden outline

CSS Outline Color

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

The color can be set by:

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

CSS Outline Width

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

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

CSS Outline Offset

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

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

Example

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

CSS Outline - Shorthand property

- `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.

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.

Text Direction

The `direction` and `unicode-bidi` properties can be used to change the text direction of an element:

Example

```
p {
  direction: rtl;
  unicode-bidi: bidi-override;
}
```

Vertical Alignment

The `vertical-align` property sets the vertical alignment of an element.

This example demonstrates how to set the vertical alignment of an image in a text:

Example

```
img.top {
  vertical-align: top;
}

img.middle {
  vertical-align: middle;
}

img.bottom {
  vertical-align: bottom;
}
```

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;}

h1 {text-decoration: overline;}


h2 {text-decoration: line-through;}

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

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;}
```

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;}
```

White Space

The `white-space` property specifies how white-space inside an element is handled.

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

Example

```
p {white-space: nowrap;}
```

Text Shadow

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

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Text shadow effect!

Example

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

Next, add a color (red) to the shadow:

Text shadow effect!

Example

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

Then, add a blur effect (5px) to the shadow:

Text shadow effect!

Example

```
h1 {text-shadow: 2px 2px 5px red;}
```

What are Web Safe Fonts?

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

Best Web Safe Fonts for HTML and CSS

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

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

Font Style

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

- normal - The text is shown normally
- italic - The text is shown in italics

Font Weight

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

- normal - The text is shown normally
- bold - The text is shown in 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.

- normal - The text is shown normally
- small-caps - The text is shown in SMALL-CAPS

Font Size

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

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

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

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

Absolute size:

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

Relative size:

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

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;}
```

Set Font Size with Em

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

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

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

Example

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

h2 {font-size: 1.875em;}

p {font-size: 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 Internet Explorer. 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!

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:

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

Example

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

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

The CSS Font Property

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

The `font` property is a shorthand property for:

- `font-style`
- `font-variant`
- `font-weight`
- `font-size/line-height`
- `font-family`

Note: The `font-size` and `font-family` values are required. If one of the other values is missing, their default value are used.

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;}
```

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).

Example

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

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

Remove Default Settings

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

CSS Tables 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;}
```

Firstname	Lastname
Rahul	Kumar
Dharmendra	Kumar

Full-Width Table

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

Example

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

Double Borders

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

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

Example

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

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

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

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

Example

```
table {width: 100%;}

th {height: 70px;}
```

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.

Example

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

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;}
```

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

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

Striped Tables

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

Example

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

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) with `overflow-x:auto` around the <table> element to make it responsive:

Example

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

<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).

The display Property

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

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:

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

Inline Elements

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

Examples of inline elements:

-
- <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;}
```

The following example displays elements as block elements:

Example

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

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;}
```

The position Property

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

There are five different position values:

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

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.

Example

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

position: fixed;

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

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

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

Example

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

position: absolute;

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

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

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

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

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

Example

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

Overlapping Elements

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:

Example

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

CSS Overflow

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

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

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

overflow: visible

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

Example

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

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):

Example

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

overflow: auto

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

Example

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

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.

Example

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

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

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

The clear Property

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

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

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

- `both` - No floating elements allowed on either the left or the right side
- `inherit` - The element inherits the clear value of its parent

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

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

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

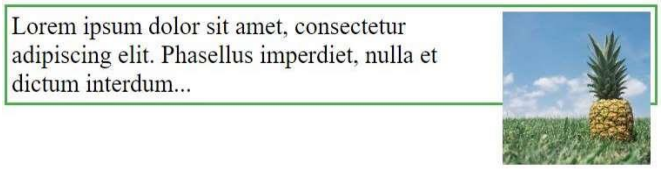
Example

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

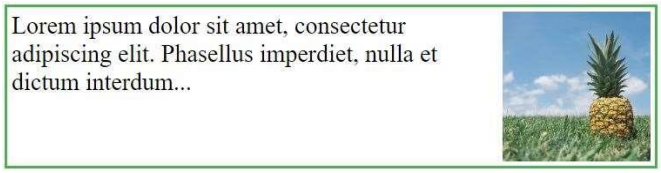
The clearfix Hack

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

Without Clearfix



With Clearfix



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

Example

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

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

Example

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

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`:

Using inline-block to Create Navigation Links

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

Example

```
.nav {
  background-color: yellow;
  list-style-type: none;
  text-align: center;
  padding: 0;
  margin: 0; }

.nav li {
  display: inline-block;
  font-size: 20px;
  padding: 20px; }
```

Center Align Elements

To horizontally center a block element (like), use `margin: auto;`

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%;
}
```

Center Vertically

```
vertical-align: top;
```

```
vertical-align: middle;
```

```
vertical-align: bottom;
```

CSS Combinators

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

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

There are four different combinators in CSS:

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

Descendant Selector

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

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

Example

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

Child Selector (>)

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

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

Example

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

Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

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

The following example selects the first <p> element that are placed immediately after elements:

Example

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

General Sibling Selector (~)

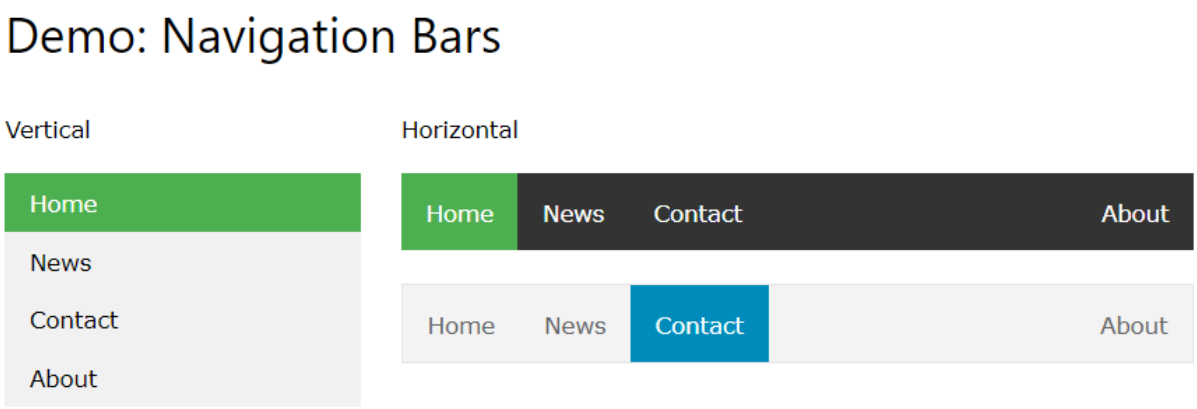
The general sibling selector selects all elements that are siblings of a specified element.

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

Example

```
div ~ p { background-color: 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>
```

Vertical Navigation Bar Examples

```
<style>
body {
  margin: 0;
}
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}
li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}
li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
</style>
```

CSS Horizontal Navigation Bar

```
<style>
body {
  font-size: 28px;
}
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
}
li {
  float: left;
}
li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
li a:hover {
  background-color: #111;
}
</style>
```

Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

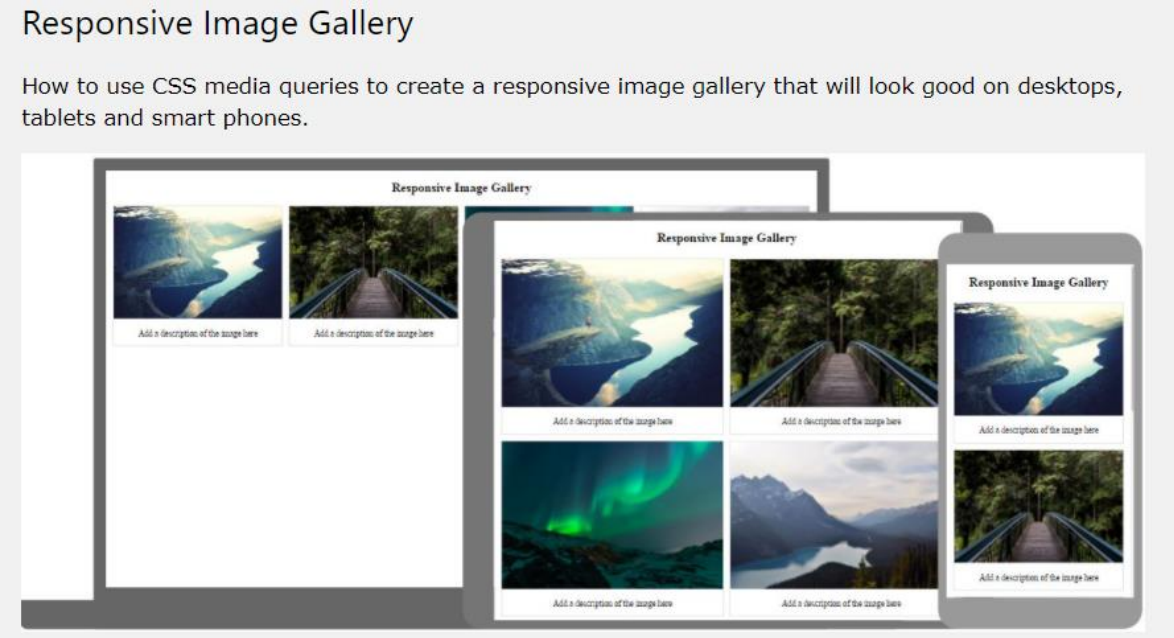
```
<style>
ul { list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333; }

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>

<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>
```

Dropdown Image

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

```
<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>

<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>
```

CSS Image Gallery




```
<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>
```

```
<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>
```

You will learn more about Advanced CSS later in this tutorial.
https://upcissyoutube.com/ "YouTube Channel-UPCISS"