CSS Introduction

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

CSS Demo

An HTML document can be displayed with different styles:

Styles Solved a Big Problem

HTML was never intended to contain tags for formatting a document.

HTML was intended to define the content of a document, 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 web sites, 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.

In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file.

All browsers support CSS today.

CSS Saves a Lot of Work!

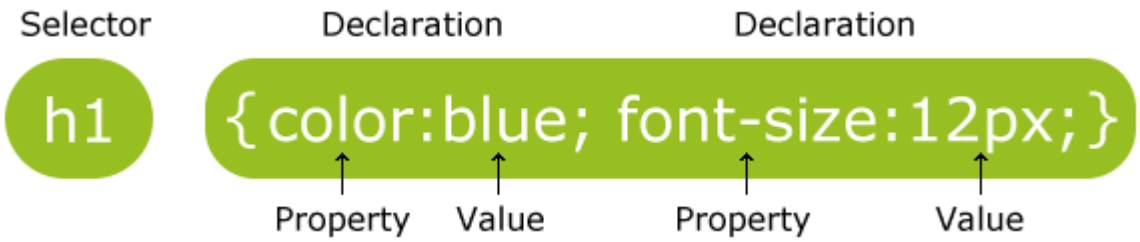
CSS defines HOW HTML elements are to be displayed.

Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single 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 property name and a value, separated by a colon.

CSS Example

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly braces:

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

To make the CSS code more readable, you can put one declaration on each line, like this:

Example

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

CSS Comments

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

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

Example

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

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

CSS Selectors

CSS Selectors

CSS selectors allow you to select and manipulate HTML element(s).

CSS selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much 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: (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 tag to find the specific element.

An id should be unique within a page, so you should use the id selector when you want to find a single, unique element.

To find 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;
}
```

 Do **NOT** start an ID name with a number!

The class Selector

The class selector finds elements with the specific class.

The class selector uses the HTML class attribute.

To find 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 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, all p elements with class="center" will be center-aligned:

Example

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

 Do **NOT** start a class name with a number!

Grouping Selectors

In style sheets there are often elements with the same style:

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

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

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

To minimize the code, you can group selectors.

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 How To...

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

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

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing just one file.

Each page must include a link to the style sheet with the <link> tag. The <link> tag goes inside the head section:

```
<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. An example of a style sheet file is shown below:

"myStyle.css":

```
hr {color: sienna;}
p {margin-left: 20px;}
body {background-image: url("images/background.gif");}
```

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 should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, inside the <style> tag, like this:

```
<head>
<style>
hr {color: sienna;}
p {margin-left: 20px;}
body {background-image: url("images/background.gif");}
</style>
</head>
```

Inline Styles

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

To use inline styles, add the style attribute to the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px;">This is a paragraph.</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, assume that an external style sheet has the following properties for the h3 selector:

```
h3 {
  color: red;
  text-align: left;
```

```
  font-size: 8pt;
}
```

then, assume that an internal style sheet also has the following properties for the h3 selector:

```
h3 {
  text-align: right;
  font-size: 20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for the h3 element will be:

```
color: red;
text-align: right;
font-size: 20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Multiple Styles Will Cascade into One

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

Tip: Even multiple external style sheets can be referenced inside a single HTML document.

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 four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an 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 in a browser (a default value).

Note: If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- 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 defined in the body selector:

Example

```
body {
  background-color: #boc4de;
}
```

With CSS, a color is most often specified by:

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

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

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

Example

```
h1 {
  background-color: #6495ed;
}

p {
  background-color: #e0ffff;
}

div {
  background-color: #b0c4de;
}
```

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 almost not readable:

Example

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

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

Example

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

Background Image - Set position and no-repeat

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

Showing the image only once is 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 - Shorthand property

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

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

The shorthand property for background is simply "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

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

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated

SS Text

TEXT FORMATTING

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified.

Text Color

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

With CSS, a color is most often specified by:

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

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

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

Example

```
body {
  color: blue;
}

h1 {
  color: #00ff00;
}

h2 {
  color: rgb(255,0,0);
}
```

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.

Text can be centered, or aligned to the left or right, or justified.

When text-align 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

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

p.date {
  text-align: right;
}

p.main {
  text-align: justify;
}
```

Text Decoration

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

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

Example

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

It can also be 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 users.

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;
}
```

All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
unicode-bidi	Used together with the direction property to set or return whether the text

	should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

CSS Font

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	All monospace characters have the same

	Lucida Console	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.

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;
}
```

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;
}
```

The example above allows Internet Explorer 9, Firefox, Chrome, Opera, and Safari to resize the text.

Note: The example above does not work in IE, prior version 9.

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Set Font Size With Em

To avoid the resizing problem with older versions of Internet Explorer, 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!

All CSS Font Properties

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

CSS Links

Links can be styled in different ways.

Styling Links

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

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: #FFoooo;
}

/* visited link */
a:visited {
  color: #ooFFoo;
}

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

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

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

Common Link Styles

In the example above the link changes color depending on what state it is in.

Lets go through some of the other common ways to style links:

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 specifies the background color for links:

Example

```
a:link {
  background-color: #B2FF99;
}

a:visited {
  background-color: #FFFF85;
}

a:hover {
  background-color: #FF704D;
}
```

```
a:active {
  background-color: #FF704D;
}
```

CSS Lists

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

List

In HTML, there are two types of lists:

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

With CSS, lists can be styled further, and images can be used as the list item marker.

Different List Item Markers

The type of list item marker is specified with the list-style-type property:

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;
}
```

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

An Image as The List Item Marker

To specify an image as the list item marker, use the list-style-image property:

Example

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

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

Crossbrowser Solution

The following example displays the image-marker equally in all browsers:

Example

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

ul li {
  background-image: url(sqpurple.gif);
  background-repeat: no-repeat;
  background-position: opx 5px;
  padding-left: 14px;
}
```

Example explained:

- For ul:
  - Set the list-style-type to none to remove the list item marker
  - Set both padding and margin to opx (for cross-browser compatibility)
- For all li in ul:
  - Set the URL of the image, and show it only once (no-repeat)
  - Position the image where you want it (left opx and down 5px)
  - Position the text in the list with padding-left

List - Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property.

The shorthand property used for lists, is the list-style property:

Example

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

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

- list-style-type
- list-style-position (for a description, see the CSS properties table below)
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

All CSS List Properties

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow
list-style-type	Specifies the type of list-item marker

CSS Tables

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/td elements have separate borders.

To display a single border for the table, use the border-collapse property.

Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

Example

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

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

Table Width and Height

Width and height of a table is 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;
}
```

Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties.

The text-align property sets the horizontal alignment, like left, right, or center:

Example

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

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

Example

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

Table Padding

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

Example

```
td {
  padding: 15px;
}
```

Table Color

The example below specifies the color of the borders, and the text and background color of th elements:

Example

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

th {
  background-color: green;
  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 HTML elements, and it consists of: margins, borders, padding, and the actual content.

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

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

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 the padding, borders and margins.

Let's make a div element with a total width of 350px:

Example

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

Let's do 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

---

Browsers Compatibility Issue

Internet Explorer 8 and earlier versions, include padding and border in the width property.

To fix this problem, add a <!DOCTYPE html> to the HTML page.

CSS Border

CSS Border Properties

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

Border Style

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

Note: None of the border properties will have ANY effect unless the **border-style** property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

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

Border Width

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

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

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

Border Color

The border-color property is used to set the color of the border. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

If the border color is not set it is inherited from the color property of the element.

Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

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

p.two {
  border-style: solid;
  border-color: #98bf21;
}
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

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

The example above can also be set with a single property:

Example

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

The border-style property can have from one to four values.

- **border-style: dotted solid double dashed;**

    - top border is dotted
    - right border is solid
    - bottom border is double
    - left border is dashed

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

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

- **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. This is called a shorthand property.

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

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

Example

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

All CSS Border Properties

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

CSS Margin

The CSS margin properties define the space around elements.

Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent 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, to overlap content.

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides of an element:

Example

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 50px;
}
```

Margin - Shorthand property

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

The shorthand property for all the margin properties is "margin":

Example

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

The margin property can have from one to four values.

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

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

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

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

CSS Padding

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

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

```
p {
  padding-top: 25px;
  padding-bottom: 25px;
  padding-right: 50px;
  padding-left: 50px;
}
```

Padding - Shorthand property

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

The shorthand property for all the padding properties is "padding":

Example

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

The padding property can have from one to four values.

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

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

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

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

All CSS Padding Properties

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

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

CSS Dimension

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

All CSS Dimension Properties

Property	Description	Values
height	Sets the height of an element	auto *length* *%* inherit
max-height	Sets the maximum height of an element	none *length* *%* inherit
max-width	Sets the maximum width of an element	none *length* *%* inherit
min-height	Sets the minimum height of an element	*length* *%* inherit
min-width	Sets the minimum width of an element	*length* *%* inherit
width	Sets the width of an element	auto *length* *%* inherit

CSS Display and Visibility

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

Hiding an Element - display:none or visibility:hidden

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

visibility:hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

Example

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

display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as if the element is not there:

Example

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

CSS Display - Block and Inline Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-
-

An inline element only takes up as much width as necessary, and does not force line breaks.

Examples of inline elements:

-
- <a>

Changing How an Element is Displayed

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 web standards.

The following example displays <li> elements as inline elements:

Example

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

The following example displays elements as block elements:

Example

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

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 of it.

CSS Positioning

Positioning can be tricky sometimes!

Decide which element to display in front!

Elements can overlap!

Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be 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 positioning method.

There are four different positioning methods.

Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

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

Fixed Positioning

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled:

Example

```
p.pos_fixed {
  position: fixed;
  top: 30px;
  right: 5px;
}
```

 **Note:** IE7 and IE8 support the fixed value only if a !DOCTYPE is specified.

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.

Fixed positioned elements can overlap other elements.

Relative Positioning

A relative positioned element is positioned relative to its normal position.

Example

```
h2.pos_left {
  position: relative;
  left: -20px;
}

h2.pos_right {
  position: relative;
  left: 20px;
}
```

The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

Example

```
h2.pos_top {
  position: relative;
  top: -50px;
}
```

Relatively positioned elements are often used as container blocks for absolutely positioned elements.

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

Example

```
h2 {
  position: absolute;
  left: 100px;
  top: 150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.

Absolutely positioned elements can overlap other elements.

Overlapping Elements

When elements are positioned outside the normal flow, 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: opx;
  top: opx;
  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.

All CSS Positioning Properties

Property	Description	Values
bottom	Sets the bottom margin edge for a positioned box	auto *length*

		% inherit
clip	Clips an absolutely positioned element	*shape* auto inherit
cursor	Specifies the type of cursor to be displayed	*url* auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help
left	Sets the left margin edge for a positioned box	auto *length* *%* inherit
overflow	Specifies what happens if content overflows an element's box	auto hidden scroll visible inherit
position	Specifies the type of positioning for an element	absolute fixed relative static inherit
right	Sets the right margin edge for a positioned box	auto *length* *%* inherit
top	Sets the top margin edge for a positioned box	auto *length* *%*

		inherit
z-index	Sets the stack order of an element	*number* auto inherit

CSS Float

What is CSS Float?

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.

Float is very often used for images, but it is also useful when working with layouts.

How Elements Float

Elements are floated horizontally; this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

If an image is floated to the right, a following text flows around it, to the left:

Example

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

Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.

Here we have made an image gallery using the float property:

Example

```
.thumbnail {
  float: left;
  width: 110px;
  height: 90px;
  margin: 5px;
}
```

Turning off Float - Using Clear

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

The clear property specifies which sides of an element other floating elements are not allowed.

Add a text line into the image gallery, using the clear property:

Example

```
.text_line {
  clear: both;
}
```

CSS Horizontal Align

In CSS, several properties are used to align elements horizontally.

Aligning Block Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-

In this chapter we will show you how to horizontally align block elements for layout purposes.

Center Aligning Using the margin Property

Block elements can be center-aligned by setting the left and right margins to "auto".

Note: Using margin:auto; will not work in IE8 and earlier**, unless a !DOCTYPE is declared.**

Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element:

Example

```
.center {
  margin-left: auto;
  margin-right: auto;
  width: 70%;
  background-color: #b0e0e6;
}
```

Tip: Center-aligning has no effect if the width is 100%.

Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning:

Example

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

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

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is a problem with IE8 and earlier, when using the position property. If a container element (in our case) has a specified width, and the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property:

Example

```
body {
  margin: 0;
```

```
  padding: 0;
}

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


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

Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property:

Example

```
.right {
  float: right;
  width: 300px;
  background-color: #b0e0e6;
}
```

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is a problem with IE8 and earlier when using the float property. If the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the float property:

Example

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

.right {
  float: right;
  width: 300px;
  background-color: #b0e0e6;
}
```

CSS Combinators

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 CSS3:

- descendant selector
- child selector
- adjacent sibling selector
- general sibling selector

Descendant Selector

The descendant selector matches all element 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 immediate children of a specified element.

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

Example

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

Adjacent Sibling Selector

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

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

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

Example

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

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 Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

Syntax

The syntax of pseudo-classes:

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

CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class {
  property:value;
}
```

Anchor Pseudo-classes

Links can be displayed in different ways in a CSS-supporting browser:

Example

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

/* visited link */
a:visited {
  color: #ooFFoo;
}

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

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

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:

CSS:

```
a.red:visited {
  color: #FFoooo;
}
```

HTML:

```
<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

If the link in the example above has been visited, it will be displayed in red.

CSS - The :first-child Pseudo-class

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

 **Note:** For :first-child to work in IE8 and earlier, a <!DOCTYPE> must be declared.

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.

 **Note:** IE8 supports the :lang pseudo-class only if a <!DOCTYPE> is specified.

In the example below, the :lang class 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>
```

All CSS Pseudo Classes/Elements

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focus	input:focus	Selects the input element which has focus

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.

The CSS opacity property is a part of the W3C CSS3 recommendation.

Example 1 - Creating a Transparent Image

The CSS3 property for transparency is **opacity**.

Look at the following CSS:

```
img {
  opacity: 0.4;
```

```
  filter: alpha(opacity=40); /* For IE8 and earlier */
}
```

IE9, Firefox, Chrome, Opera, and Safari use the property **opacity** for transparency. The opacity property can take a value from 0.0 - 1.0. A lower value makes the element more transparent.

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

Example 2 - Image Transparency - Hover Effect

```
img {
  opacity: 0.4;
  filter: alpha(opacity=40); /* For IE8 and earlier */
}

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

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

The CSS for this is: **opacity=1**.

IE8 and earlier: **filter:alpha(opacity=100)**.

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

Example 3 - Text in Transparent Box

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

The source code looks like this:

Example

```
<html>
<head>
<style>
div.background {
  width: 500px;
  height: 250px;
  background: url(klematis.jpg) repeat;
  border: 2px solid black;
}

div.transbox {
  width: 400px;
  height: 180px;
  margin: 30px 50px;
  background-color: #ffffff;
  border: 1px solid black;
  opacity: 0.6;
  filter: alpha(opacity=60); /* For IE8 and earlier */
}

div.transbox p {
  margin: 30px 40px;
  font-weight: bold;
  color: #000000;
}
</style>
</head>
<body>

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

```
   This is some text that is placed in the transparent box.</p>
  </div>
</div>

</body>
</html>
```

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