The Type Selectors

**The Universal Selectors**

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type:

\* {

color: #000000;

}

**The Descendant Selectors**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, the style rule will apply to <em> element only when it lies inside the <ul> tag.

ul em {

color: #000000;

}

**The Class Selectors**

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

.black {

color: #000000;

}

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example:

h1.black {

color: #000000;

}

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

You can apply more than one class selectors to a given element. Consider the following example:

<p class="center bold">

This para will be styled by the classes center and bold.

</p>

**The ID Selectors**

You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

#black {

color: #000000;

}

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example:

h1#black {

color: #000000;

}

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors. For example:

#black h2 {

color: #000000;

}

In this example, all level 2 headings will be displayed in black color when those headings will lie within tags having *id* attribute set to *black*.

**The Child Selectors**

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example:

body > p {

color: #000000;

}

**The Attribute Selectors**

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text*:

input[type="text"]{

color: #000000;

}

The advantage to this method is that the <input type="submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

 **p[lang]** - Selects all paragraph elements with a *lang* attribute.

 **p[lang="fr"]** - Selects all paragraph elements whose *lang* attribute has a value of exactly "fr".

 **p[lang~="fr"]** - Selects all paragraph elements whose *lang* attribute contains the word "fr".

 **p[lang|="en"]** - Selects all paragraph elements whose *lang* attribute contains values that are exactly "en", or begin with "en-".

**Grouping Selectors**

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example:

h1, h2, h3 {

color: #36C;

font-weight: normal;

letter-spacing: .4em;

margin-bottom: 1em;

text-transform: lowercase;

}

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various *class* selectors together as shown below:

#content, #footer, #supplement {

position: absolute;

left: 510px;

width: 200px;

}

**Imported CSS -@import Rule**

@import is used to import an external stylesheet in a manner similar to the <link> element. Here is the generic syntax of @import rule.

<head>

<@import "URL";

</head>

Here URL is the URL of the style sheet file having style rules. You can use another syntax as well:

<head>

<@import url("URL");

</head>

BACKGROUND

The **background-color** property is used to set the background color of an element.

 The **background-image** property is used to set the background image of an element.

 The **background-repeat** property is used to control the repetition of an image in the background.

background-repeat: repeat;

repeat the background image vertically.

background-repeat: repeat-y;

The following example demonstrates how to repeat the background image horizontally.

background-repeat: repeat-x;

 The **background-position** property is used to control the position of an image in the background.

The following example demonstrates how to set the background image position 100 pixels away from the left side.

background-position:100px;

The following example demonstrates how to set the background image position 100 pixels away from the left side and 200 pixels down from the top.

background-position:100px 200px;

 The **background-attachment** property is used to control the scrolling of an image in the background.

Background attachment determines whether a background image is fixed or scrolls with the rest of the page.

The following example demonstrates how to set the fixed background image.

background-attachment:fixed;

The following example demonstrates how to set the scrolling background image.

background-attachment:scroll;

 The **background** property is used as a shorthand to specify a number of other background properties.

You can use the *background* property to set all the background properties at once. For example:

background: #ffffff url("img\_tree.png") no-repeat right top;

<p style="background:url(/images/pattern1.gif) repeat fixed;">

This parapgraph has fixed repeated background image.

</p>

Font

The **font-family** property is used to change the face of a font.

 The **font-style** property is used to make a font italic or oblique.

 The **font-variant** property is used to create a small-caps effect.

 The **font-weight** property is used to increase or decrease how bold or light a font appears.

 The **font-size** property is used to increase or decrease the size of a font.

 The **font** property is used as shorthand to specify a number of other font properties.

**Set the Font Family**

Following is the example, which demonstrates how to set the font family of an element. Possible value could be any font family name.

<p style="font-family:georgia,garamond,serif;">

This text is rendered in either georgia, garamond, or the default

serif font depending on which font you have at your system.

</p>

It will produce the following result:

This text is rendered in either georgia, garamond, or the default

serif font depending on which font you have at your system.

**Set the Font Style**

The following example demonstrates how to set the font style of an element. Possible values are *normal, italic and oblique*.

<p style="font-style:italic;">

**Set the Font Variant**

The following example demonstrates how to set the font variant of an element. Possible values are *normal and small-caps*.

<p style="font-variant:small-caps;">

This text will be rendered as small caps

</p>

**Set the Font Weight**

The following example demonstrates how to set the font weight of an element. The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*.

<p style="font-weight:bold;">

This font is bold.

</p>

<p style="font-weight:bolder;">

This font is bolder.

</p>

<p style="font-weight:900;">

This font is 900 weight.

</p>

**Set the Font Size**

The following example demonstrates how to set the font size of an element. The font-size property is used to control the size of fonts. Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %.*

<p style="font-size:20px;">

This font size is 20 pixels

</p>

<p style="font-size:small;">

This font size is small

</p>

<p style="font-size:large;">

This font size is large

</p>

**Set the Font Size Adjust**

The following example demonstrates how to set the font size adjust of an element. This property enables you to adjust the x-height to make fonts more legible. Possible value could be any number.

<p style="font-size-adjust:0.61;">

This text is using a font-size-adjust value.

</p>

**Set the Font Stretch**

The following example demonstrates how to set the font stretch of an element. This property relies on the user's computer to have an expanded or condensed version of the font being used.

Possible values could be *normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded*.

<p style="font-stretch:ultra-expanded;">

If this doesn't appear to work, it is likely that

your computer doesn't have a condensed or expanded

version of the font being used.

</p>

**Shorthand Property**

You can use the *font* property to set all the font properties at once. For example:

<p style="font:italic small-caps bold 15px georgia;">

Applying all the properties on the text at once.

</p>

Text

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

 The **direction** property is used to set the text direction.

<p style="direction:rtl;">

This text will be renedered from right to left

</p>

 The **letter-spacing** property is us to adedd or subtract space between the letters that make up a word.

<p style="letter-spacing:5px;">

This text is having space between letters.

</p>

 The **word-spacing** property is used to add or subtract space between the words of a sentence.

<p style="word-spacing:5px;">

This text is having space between words.

</p>

 The **text-indent** property is used to indent the text of a paragraph.

The following example demonstrates how to indent the first line of a paragraph. Possible values are *% or a number specifying indent space.*

<p style="text-indent:1cm;">

This text will have first line indented by 1cm

and this line will remain at its actual position

this is done by CSS text-indent property.

</p>

 The **text-align** property is used to align the text of a document.

<p style="text-align:right;">

This will be right aligned.

</p>

<p style="text-align:center;">

This will be center aligned.

</p>

<p style="text-align:left;">

This will be left aligned.

</p>

 The **text-decoration** property is used to underline, overline, and strikethrough text.

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:

h1 {  
    text-decoration: overline;  
}  
  
h2 {  
    text-decoration: line-through;  
}  
  
h3 {  
    text-decoration: underline;  
}<p style="text-decoration:underline;">

This will be underlined

</p>

<p style="text-decoration:line-through;">

This will be striked through.

</p>

<p style="text-decoration:overline;">

This will have a over line.

</p>

<p style="text-decoration:blink;">

This text will have blinking effect

</p>

 The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.

<p style="text-transform:capitalize;">

This will be capitalized

</p>

<p style="text-transform:uppercase;">

This will be in uppercase

</p>

<p style="text-transform:lowercase;">

This will be in lowercase

</p>

 The **white-space** property is used to control the flow and formatting of text.

<p style="white-space:pre;">This text has a line break

and the white-space pre setting tells the browser to honor it

just like the HTML pre tag.</p>

 The **text-shadow** property is used to set the text shadow around a text.

<p style="text-shadow:4px 4px 8px blue;">

If your browser supports the CSS text-shadow property,

this text will have a blue shadow.</p>

IMAGES

The **border** property is used to set the width of an image border.

<img style="border:0px;" src="/images/css.gif" />

<br />

<img style="border:3px dashed red;" src="/images/css.gif" />

 The **height** property is used to set the height of an image.

<img style="border:1px solid red; height:100px;"

src="/images/css.gif" />

 The **width** property is used to set the width of an image.

<img style="border:1px solid red; width:100px;"

src="/images/css.gif" />

 The **-moz-opacity** property is used to set the opacity of an image.

The *-moz-opacity* property of an image is used to set the opacity of an image. This property is used to create a transparent image in Mozilla. IE uses **filter:alpha(opacity=x)** to create transparent images.

In Mozilla (-moz-opacity:x), x can be a value from 0.0 - 1.0. A lower value makes the element more transparent (The same things goes for the CSS3-valid syntax opacity:x).

In IE (filter:alpha(opacity=x)), x can be a value from 0 - 100. A lower value makes the element more transparent.

<img style="border:1px solid red;-moz-opacity:0.4;filter:alpha(opacity=40);"

src="/images/css.gif" />

Link

The **:link** signifies unvisited hyperlinks.

 The **:visited** signifies visited hyperlinks.

 The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.

 The **:active** signifies an element on which the user is currently clicking.

<style type="text/css">

a:link {color: #000000}

a:visited {color: #006600}

a:hover {color: #FFCC00}

a:active {color: #FF00CC}

</style>

**Table Borders**

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

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

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

## Collapse Table Borders

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

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

## Table Width and Height

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

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

table {  
    width: 100%;  
}  
  
th {  
    height: 50px;  
}

## Horizontal Alignment

The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

th {  
    text-align: left;  
}

## Vertical Alignment

The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

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

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

## Horizontal Dividers

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

## Hoverable Table

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

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}

## Table Color

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

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

## Responsive Table

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

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

### Example

<div style="overflow-x:auto;">  
  
<table>  
... table content ...  
</table>  
  
</div>

## The CSS Box Model

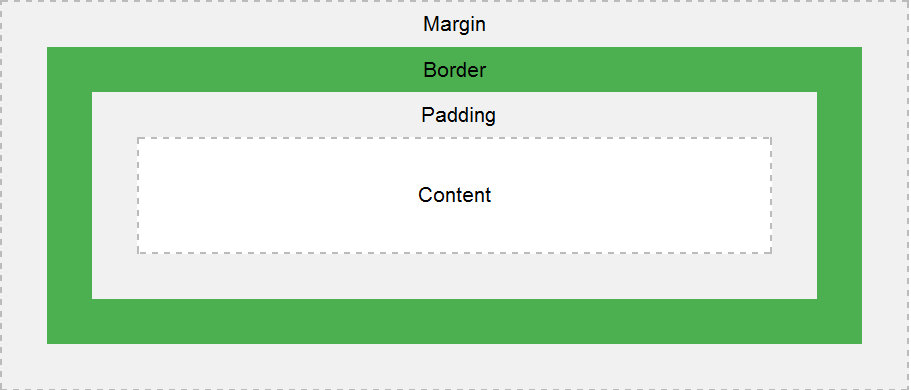
All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The image below illustrates the box model:

Explanation of the different parts:

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

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

 Explanation of the different parts:

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

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

### Example

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

MARGINS

The **margin** specifies a shorthand property for setting the margin properties in one declaration.

 The **margin-bottom** specifies the bottom margin of an element.

 The **margin-top** specifies the top margin of an element.

 The **margin-left** specifies the left margin of an element.

 The **margin-right** specifies the right margin of an element.

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

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

This element has a margin of 80px.

All the margin properties can have the following values:

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

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

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

Example Margin Example

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

### Example

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

Example Div Container

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

If the margin property has four values:

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

If the margin property has three values:

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

If the margin property has two values:

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

If the margin property has one value:

* **margin: 25px;**
  + all four margins are 25px
* You can set the margin property to auto to horizontally center the element within its container.
* The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

### Example

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

PADDINGS

The **padding-bottom** specifies the bottom padding of an element.

 The **padding-top** specifies the top padding of an element.

 The **padding-left** specifies the left padding of an element.

 The **padding-right** specifies the right padding of an element.

 The **padding** serves as shorthand for the preceding properties

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

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

This element has a padding of 50px.

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

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

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

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

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

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

OR

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

Example Padding Example 1

If the padding property has four values:

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

If the padding property has three values:

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

If the padding property has two values:

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

If the padding property has one value:

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

CSS Layout - The display Property

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

**The display Property**

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

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

## Block-level Elements

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

The <div> element is a block-level element.

Examples of block-level elements:

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

## Inline Elements

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

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

## Display: none;

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

The <script> element use display: none; as its default.

## Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline <li> elements for horizontal menus:

### Example

li {  
    display: inline;  
}

Example Inline

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

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

### Example

span {  
    display: block;  
}

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

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

### Example

h1.hidden {  
    display: none;  
}

visibility:hidden; also hides an element.

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

### Example

h1.hidden {  
    visibility: hidden;  
}

CSS Layout - The position Property

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

**The position Property**

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

There are four different position values:

* static
* relative
* fixed
* absolute

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

## position: static;

HTML elements are positioned static by default.

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

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

This <div> element has position: static;

Here is the CSS that is used:

### Example

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

Example Position Example

## position: relative;

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

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

This <div> element has position: relative;

Here is the CSS that is used:

### Example

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

Example Position Relative

## position: fixed;

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

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

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

### Example

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

Example Position Fixed

**position: absolute;**

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

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

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

Here is a simple example:

div.relative {  
    position: relative;  
    width: 400px;  
    height: 200px;  
    border: 3px solid #73AD21;  
}   
  
div.absolute {  
    position: absolute;  
    top: 80px;  
    right: 0;  
    width: 200px;  
    height: 100px;  
    border: 3px solid #73AD21;  
}

**Overlapping Elements**

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:

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

example position Over lap

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

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

CSS Layout - float and clear

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

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



**The float Property**

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

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

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

Example Float

## The clear Property

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

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

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

### Example

div {  
    clear: left;  
}

## The clearfix - overflow: auto;

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

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

### Example

.clearfix {  
    overflow: auto;  
}

## Web Layout Example

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

### Example

div {  
    border: 3px solid blue;  
}  
  
.clearfix {  
    overflow: auto;  
}  
  
nav {  
    float: left;  
    width: 200px;  
    border: 3px solid #73AD21;  
}  
  
section {  
    margin-left: 206px;  
    border: 3px solid red;  
}Example Web Layot

**All CSS Float Properties**

|  |  |
| --- | --- |
| **Property** | **Description** |
| [clear](http://www.w3schools.com/cssref/pr_class_clear.asp) | Specifies on which sides of an element where floating elements are not allowed to float |
| [float](http://www.w3schools.com/cssref/pr_class_float.asp) | Specifies whether or not an element should float |
| [overflow](http://www.w3schools.com/cssref/pr_pos_overflow.asp) | Specifies what happens if content overflows an element's box |
| [overflow-x](http://www.w3schools.com/cssref/css3_pr_overflow-x.asp) | Specifies what to do with the left/right edges of the content if it overflows the element's content area |
| [overflow-y](http://www.w3schools.com/cssref/css3_pr_overflow-y.asp) | Specifies what to do with the top/bottom edges of the content if it overflows the element's content area |

CSS Layout - inline-block

## The inline-block Value

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

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

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

### Examples

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

### Example

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

Example Inline Block

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

### Example

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

# CSS Layout - Horizontal Align

## Center Align - Using margin

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Use margin: auto;, to horizontally center an element within its container.

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

### Example

.center {  
    margin: auto;  
    width: 60%;  
    border: 3px solid #73AD21;  
    padding: 10px;  
}

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

Example Center align using Margin

## Left and Right Align - Using position

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

### Example

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

Example Left right using Positon

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

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

There is also a problem with IE8 and earlier, when using position. If a container element (in our case <div class="container">) 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. So, always set the !DOCTYPE declaration when using position:

### Example

body {  
    margin: 0;  
    padding: 0;  
}  
  
.container {  
    position: relative;  
    width: 100%;  
}  
  
.right {  
    position: absolute;  
    right: 0px;  
    width: 300px;  
    background-color: #b0e0e6;  
}

## Left and Right Align - Using float

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

### Example

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

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

There is also a problem with IE8 and earlier, when using float. 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. So, always set the !DOCTYPE declaration when using float:

### Example

body {  
    margin: 0;  
    padding: 0;  
}  
  
.right {  
    float: right;  
    width: 300px;  
    background-color: #b0e0e6;  
}

Example Left