CSS DIV

# Div

Div (short for division) divides the content into individual sections. Each section can then have its own formatting, as specified by the CSS. Div is a block-level container, meaning that there is a line feed after the </div> tag.

For example, if we have the following CSS declaration:

|  |
| --- |
| .large {    color: #00FF00;    font-family:arial;    font-size: 4pt;  } |

The HTML code

|  |
| --- |
| <div class="large">    This is a DIV sample.  </div> |

gets displayed as

|  |
| --- |
| This is a DIV sample. |

# Span

Span is similar to div in that they both divide the content into individual sections. The difference is that span goes into a finer level, so we can span to format a single character if needed. There is no line feed after the </span> tag.

For example, if we have the following CSS declaration:

|  |
| --- |
| .largefont {    color: #0066FF;    font-family:arial;    font-size: 6px;  } |

The HTML code

|  |
| --- |
| Span is not at the <span class="largefont">block level</span>. |

Gets displayed as

|  |
| --- |
| Span is not at the block level. |

## HTML Layout Using <div> Elements

|  |  |
| --- | --- |
| **Note** | The <div> element is often used as a layout tool, because it can easily be positioned with CSS. |

This example uses 4 <div> elements to create a multiple column layout:

## **Website Layout Using HTML5**

HTML5 offers new semantic elements that define different parts of a web page:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HTML5 Semantic Elements | |  |  | | --- | --- | | header | Defines a header for a document or a section | | nav | Defines a container for navigation links | | section | Defines a section in a document | | article | Defines an independent self-contained article | | aside | Defines content aside from the content (like a sidebar) | | footer | Defines a footer for a document or a section | | details | Defines additional details | | summary | Defines a heading for the details element | |

This example uses <header>, <nav>, <section>, and <footer> to create a multiple column layout:

# CSS Position

The position property defines how an element will be positioned on a page.

# CSS Positioning Methods

Positioning elements appropriately on the web pages is a necessity for a good layout design. There are several methods in CSS that you can use for positioning elements. The following section will describe you these positioning methods one by one.

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.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of CSS Static Positioning</title>

<style type="text/css">

p {

padding: 15px;

background: #ffdab9;

}

</style>

</head>

<body>

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

</body>

</html>

## Relative Positioning

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

In the relative positioning scheme the element's box position is calculated according to the normal flow. Then the box is shifted from this normal position according to the properties —top or bottom and/or left or right.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of CSS Relative Positioning</title>

<style type="text/css">

p { background: #FFDEAD; padding: 10px; }

p.positioned { position: relative; left: 50px; }

</style>

</head>

<body>

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

<p class="positioned">The left margin edge of this paragraph is shifted to right by 50px from its original position.</p>

</body>

</html>

## 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>:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of CSS Absolute Positioning</title>

<style type="text/css">

p { background: #FFB6C1; padding: 10px; }

p.positioned { position: absolute; top: 100px; }

</style>

</head>

<body>

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

<p class="positioned">The top margin edge of this paragraph is shifted below by 100px from the top edge of the document's viewport.</p>

</body>

</html>

## Fixed Positioning

An element with a fixed position is positioned relative to the browser window, and will not move even if the window is scrolled:

<!DOCTYPE html>

<html>

<head>

<style>

p.pos\_fixed { position: fixed; top: 30px; right: 5px; color: red; }

</style>

</head>

<body>

<p><b>Note:</b> IE7 and IE8 supports the fixed value only if a !DOCTYPE is specified.</p>

<p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p>

<p class="pos\_fixed">Some positioned text.</p> </body> </html>

# CSS Layers

*The CSS z-index property can be used in conjugation with the position property to create an effect of layers like Photoshop.*

# CSS z-index Property

Usually HTML pages are considered two-dimensional, because text, images and other elements are arranged on the page without overlapping. However, in addition to their horizontal and vertical positions, boxes can be stacked along the z-axis as well i.e. one on top of the other by using the CSS z-index property. This property specifies the stack level of a box whose [position](http://www.tutorialrepublic.com/css-tutorial/css-position.php) value is one of absolute, fixed, or relative.

The z-axis position of each layer is expressed as an integer representing the stacking order for rendering. An element with a larger z-index overlaps an element with a lower one.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of CSS layer effect</title>

<style type="text/css">

div {

top: 30px;

left: 30px;

width: 100px;

height: 100px;

position: absolute;

}

div.red {

background: #ff0000;

z-index: 1;

}

div.green {

background: #00ff00;

z-index: 2;

}

div.blue {

background: #0000ff;

z-index: 3;

}

</style>

</head>

<body>

<div class="red">

<div class="green">

<div class="blue"></div>

</div>

</div>

</body>

</html>

A z-index property can help you to create more complex webpage layouts. Following is the example which shows how to create layers in CSS.

# CSS Float

*The CSS float property specifies whether a box should float or not.*

# Floating Elements with CSS

You can float elements to the left or right, but only applies to the elements that generate boxes that are not [absolutely positioned](http://www.tutorialrepublic.com/css-tutorial/css-position.php#absolute-positioning). Any element that follows the floated element will flow around the floated element on the other side.

The float property may have one of the three values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| left | The element floats on the left side of its containing block. |
| right | The element floats on the right side of its containing block. |
| none | Removes the float property from an element. |