commentato CSS

**BOX MODEL DISPLAY VISITBILITY POSITION OVERFLOW FLOAT INLINE-BLOC ALIGN**

**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.

INLINE

BLOCK

**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.

**visibility:hidden;** also hides an element.

HIDDEN LO NASCONDE MA è LI QUINDI IL LAYOUT PUò CAMBIARE

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

Override The Default Display Value

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

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

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

Example

li {

display: inline;

}

**The 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.

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:

## 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.

div.relative {

position: relative;

left: 10px;

border: 3px solid #73AD21;

## position: fixed;

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

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

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

## 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

## 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:

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

img {  
    position: absolute;  
    left: 0px;  
    top: 0px;  
    z-index: -1; // vuol dire che l'immagine resta dietro  
}

The CSS overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. It renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
* auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

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

The float property specifies whether or not an element should 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:

## 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.