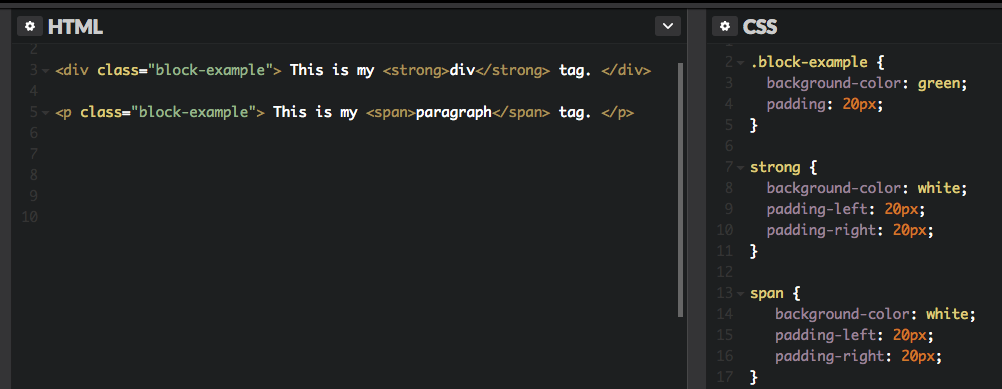
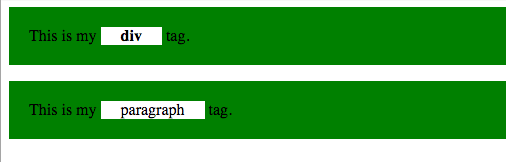
**CSS display properties: block, inline, and inline-block — & how to tell the difference**

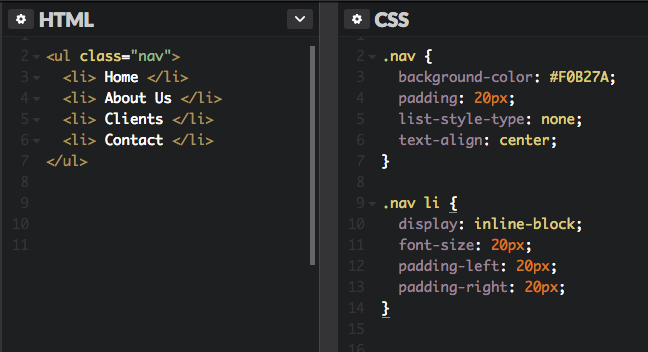
* **inline**: only margin-left, margin-right, padding-left, padding-right
* **inline-block**: margin, padding, height, width
* A block element always starts on a new line, and fills up the horizontal space left and right on the web page. You can add margins and padding on all four sides of any block element — top, right, left, and bottom.
* Some examples of block elements are **<div>** and **<p>** tags.
* Inline elements don’t start on a new line, they appear on the same line as the content and tags beside them. Some examples of inline elements are **<span>** , **<strong>**, and **<img>** tags.
* When it comes to margins and padding, browsers treat inline elements differently. You can add space to the left and right on an inline element, but you cannot add height to the top or bottom padding or margin of an inline element.
* Inline elements can actually appear within block elements, as shown below. I’ve added white padding on the left and right side of each inline element.

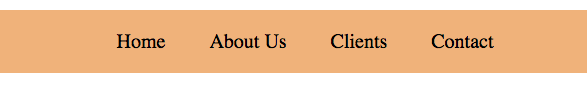




Inline-block elements are similar to inline elements, except they can have padding and margins added on all four sides. You’ll have to declare display: inline-block in your CSS code.

One common use for using inline-block is for creating navigation links horizontally





========================

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

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

===========

how many ways insert css?

## Three Ways to Insert CSS

* External style sheet
* Internal style sheet
* Inline style

how to put the div element into center?

To horizontally center a block element (like <div>), use margin: auto;

============================

how to put an image in center?

To center an image, set left and right margin to auto and make it into a block element:

display: block;  
    margin-left: auto;  
    margin-right: auto;  
    width: 40%;

=============================

why The position Property in css ?

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.

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

## position: static;

HTML elements are positioned static by default.

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

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

## position: relative;

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

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position

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

## position: fixed;

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.

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

## position: absolute;

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

## position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

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

===============================

In this example, the image is taller than the element containing it, and it is floated, so it overflows outside of its container:

.clearfix {

overflow: auto;

}

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

Add a clearfix class with overflow: auto; to the containing element, to fix this problem:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

==============

## Simple Tooltip Hover in css?

<style>

p {

display: none;

background-color: yellow;

padding: 20px;

}

div:hover p {

display: block;

}

</style>

</head>

<body>

<div>Hover over me to show the p element

<p>Tada! Here I am!</p>

</div>

===========================

oops

closures

anoynomus function

ngfor

ngif

angular2 vs angular4

ng template vs template

## All CSS Pseudo Elements

|  |  |  |
| --- | --- | --- |
| [after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert content after every <p> element |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Insert content before every <p> element |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Selects the first letter of every <p> element |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Selects the first line of every <p> element |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | p::selection | Selects the portion of an element that is selected by a user |

how to remove border default settings?

* set margin: 0; and padding: 0; to remove browser default settings

================

## Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This <div> element has a width of 500px, and margin set to auto.

**Note:** The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This <div> element has a max-width of 500px, and margin set to auto.

**Tip:** Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

div.ex1 {  
    width: 500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}  
  
div.ex2 {  
    max-width: 500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}

==================================

CSS Backgrounds

background-color

background-color: lightblue;

background-image

background-repeat

background-attachment

background-position

## Background Image

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

1em equal to 10px

background-image: url("gradient\_bg.png");  
    background-repeat: repeat-x;// only horizontally

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

background-repeat: no-repeat;

background-position:

left top  
left center  
left bottom  
right top  
right center  
right bottom  
center top  
center center  
center bottom

background-attachment: scroll|fixed|local|initial|inherit;

The background image will scroll with the page. This is default

## CSS Border Properties

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

## Border Style

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

* dotted - Defines a dotted border
* dashed - Defines a dashed border
* solid - Defines a solid border

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

p {border-style: dotted;}

p {

border-top-style: dotted;

border-right-style: solid;

border-bottom-style: dotted;

border-left-style: solid;

}

## Border Width

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

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

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

by using one of the three pre-defined values: thin, medium, or thick. **border-width: thin medium thick 10px;**

border-width: medium|thin|thick|*length*|initial|inherit;

## Border Color

p.three {  
    border-style: solid;  
    border-color: red green blue yellow;  
}

**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 - Shorthand Property

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

## Rounded Borders

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

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

border-bottom-style: none;

border-top: thick double #ff0000;

border-left-width: 15px;

border-right-color: #ff0000;

**Four values - border-radius: 15px 50px 30px 5px;**(first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner):

CSS Margins

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

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

* margin-top
* margin-right
* margin-bottom
* margin-left
* auto - the browser calculates the margin

**Tip:** Negative values are allowed.

## The auto Value

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:

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

## Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

h1 {  
    margin: 0 0 50px 0;  
}  
  
h2 {  
    margin: 20px 0 0 0;  
}

In the example above, the <h1> element has a bottom margin of 50px and the <h2> element has a top margin set to 20px.

Common sense would seem to suggest that the vertical margin between the <h1> and the <h2> would be a total of 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

## CSS Padding

The CSS padding properties are used to generate space around an element's content.

* padding-top
* padding-right
* padding-bottom
* padding-left

**Note:** Negative values are not allowed.

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

## Padding and Element Width

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element  
height + padding + border = actual height of an element

The CSS width property specifies the width of the element's content area.

The content area is the portion inside the padding, border, and margin of an element ([the box model](https://www.w3schools.com/css/css_boxmodel.asp)).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

In the following example, the <div> element is given a width of 300px. However, the actual rendered width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

div.ex1 {

width: 300px;

background-color: yellow;

}

div.ex2 {

width: 300px;

padding: 25px;

background-color: lightblue;

}

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease. Here is an example:

div {  
    width: 300px;  
    padding: 25px;  
    box-sizing: border-box;  
}

box-sizing: content-box|border-box|initial|inherit;

what is the difference between content-box and border-box in css?

|  |  |
| --- | --- |
| content-box | Default. The width and height properties (and min/max properties) includes only the content. Border and padding are not included |
| border-box | The width and height properties (and min/max properties) includes content, padding and border |

## Setting height and width

The height and width can be set to auto (this is default. Means that the browser calculates the height and width).

what is the drawback of width in css?

The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

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

Using max-width instead, in this situation, will improve the browser's handling of small windows.

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

If the content is smaller than the minimum height, the minimum height will be applied.

If the content is larger than the minimum height, the min-height property has no effect.

## 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.

It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

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

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

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

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

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

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

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

## CSS Outline

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

CSS Text

## Text Color

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

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

If you define the color property, you must also define the background-color.

body {  
    color: blue;  
}  
  
h1 {  
    color: green;  
}

## Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

When the text-align property 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):

div {  
    text-align: justify;  
}

## Text Decoration

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

The value text-decoration: none; is often used to remove underlines from links:

a {  
    text-decoration: none;  
}

h1 {  
    text-decoration: overline;  
}  
  
h2 {  
    text-decoration: line-through;  
}  
  
h3 {  
    text-decoration: underline;  
}

how to remove underline in links in css?

a {  
    text-decoration: none;  
}

## 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:

p.uppercase {  
    text-transform: uppercase;  
}  
  
p.lowercase {  
    text-transform: lowercase;  
}  
  
p.capitalize {  
    text-transform: capitalize;  
}

how to put capitalize the first letter of each word in css?

p.capitalize {  
    text-transform: capitalize;  
}

## Text Indentation

The text-indent property is used to specify the indentation of the first line of a text:

p {  
    text-indent: 50px;  
}

how to give specify the indentation of the firstline of text in css?

## Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

h1 {  
    letter-spacing: 3px;  
}  
  
h2 {  
    letter-spacing: -3px;  
}

## Line Height

The line-height property is used to specify the space between lines:

p.small {  
    line-height: 0.8;  
}  
  
p.big {  
    line-height: 1.8;  
}

## Text Direction

The direction property is used to change the text direction of an element:

p {  
    direction: rtl;  
}

This is the default text direction.

## Word Spacing

The word-spacing property is used to specify the space between the words in a text.

h1 {  
    word-spacing: 10px;  
}  
  
h2 {  
    word-spacing: -5px;  
}

## Text Shadow

The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

h1 {  
    text-shadow: 3px 2px red;  
}

CSS white-space

how to disable text wrapping inside an element in css ?

p.a {  
    white-space: nowrap;}

p.a {  
    white-space: nowrap; Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a <br> tag is encountered  
}  
  
p.b {  
    white-space: normal;  
}  
  
p.c {  
    white-space: pre; it is same pre tag in html what you write as same it show  
}

vertical-align

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

|  |  |
| --- | --- |
| **Default value:** | Baseline |

how to set the vertical align of an image in a text.

img.top {

vertical-align: text-top;

}

img.bottom {

vertical-align: text-bottom;

}

n W3Schools image with a default alignment.

An W3Schools image with a text-top alignment.

An W3Schools image with a text-bottom alignment.

CSS text-overflow

The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped, display an ellipsis (...), or display a custom string.

Both of the following properties are required for text-overflow:

* white-space: nowrap;
* overflow: hidden;

text-overflow: clip|ellipsis|*string.*

*default value is clip.*

*div.a {*

*white-space: nowrap;*

*width: 50px;*

*overflow: hidden;*

*text-overflow: clip;*

*border: 1px solid #000000;*

*}*

*div.b {*

*white-space: nowrap;*

*width: 50px;*

*overflow: hidden;*

*text-overflow: ellipsis;*

*border: 1px solid #000000;*

*}*

<h2>text-overflow: clip (default):</h2>

<div class="a">Hello world!</div>

CSS Fonts

The CSS font properties define the font family, boldness, size, and the style of a text.

sans-serif fonts are considered easier to read than serif fonts.

## Font Family

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, and so on.

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:

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)

p.normal {  
    font-style: normal;  
}  
  
p.italic {  
    font-style: italic;  
}  
  
p.oblique {  
    font-style: oblique;  
}

 If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

## Set Font Size With Em

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.

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

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:

body {  
    font-size: 100%;  
}  
  
h1 {  
    font-size: 2.5em;  
}

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

## Font Weight

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

p.normal {

font-weight: normal;

}

p.light {

font-weight: lighter;

}

p.thick {

font-weight: bold;

}

p.thicker {

font-weight: 900;

}

## Responsive Font Size

The text size can be set with a vw unit, which means the "viewport width".

way the text size will follow the size of the browser window:

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

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

## Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}

CSS Icons

<i class="fa fa-cloud"></i>  
<i class="fa fa-heart"></i>  
<i class="fa fa-car"></i>  
<i class="fa fa-file"></i>  
<i class="fa fa-bars"></i>

CSS Links

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

a {  
    color: hotpink;  
}

/\* unvisited link \*/  
a:link {  
    color: red;  
}  
  
/\* visited link \*/  
a:visited {  
    color: green;  
}  
  
/\* mouse over link \*/  
a:hover {  
    color: hotpink;  
}  
  
/\* selected link \*/  
a:active {  
    color: blue;  
}

## Text Decoration

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

 text-decoration: none;

text-decoration: underline;

===================

cursor:hand,pointer;

## HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

* unordered lists (<ul>) - the list items are marked with bullets
* ordered lists (<ol>) - the list items are marked with numbers or letters

## Different List Item Markers

The list-style-type property specifies the type of list item marker.

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;  
}

## Position The List Item Markers

The list-style-position property specifies the position of the list-item markers (bullet points).

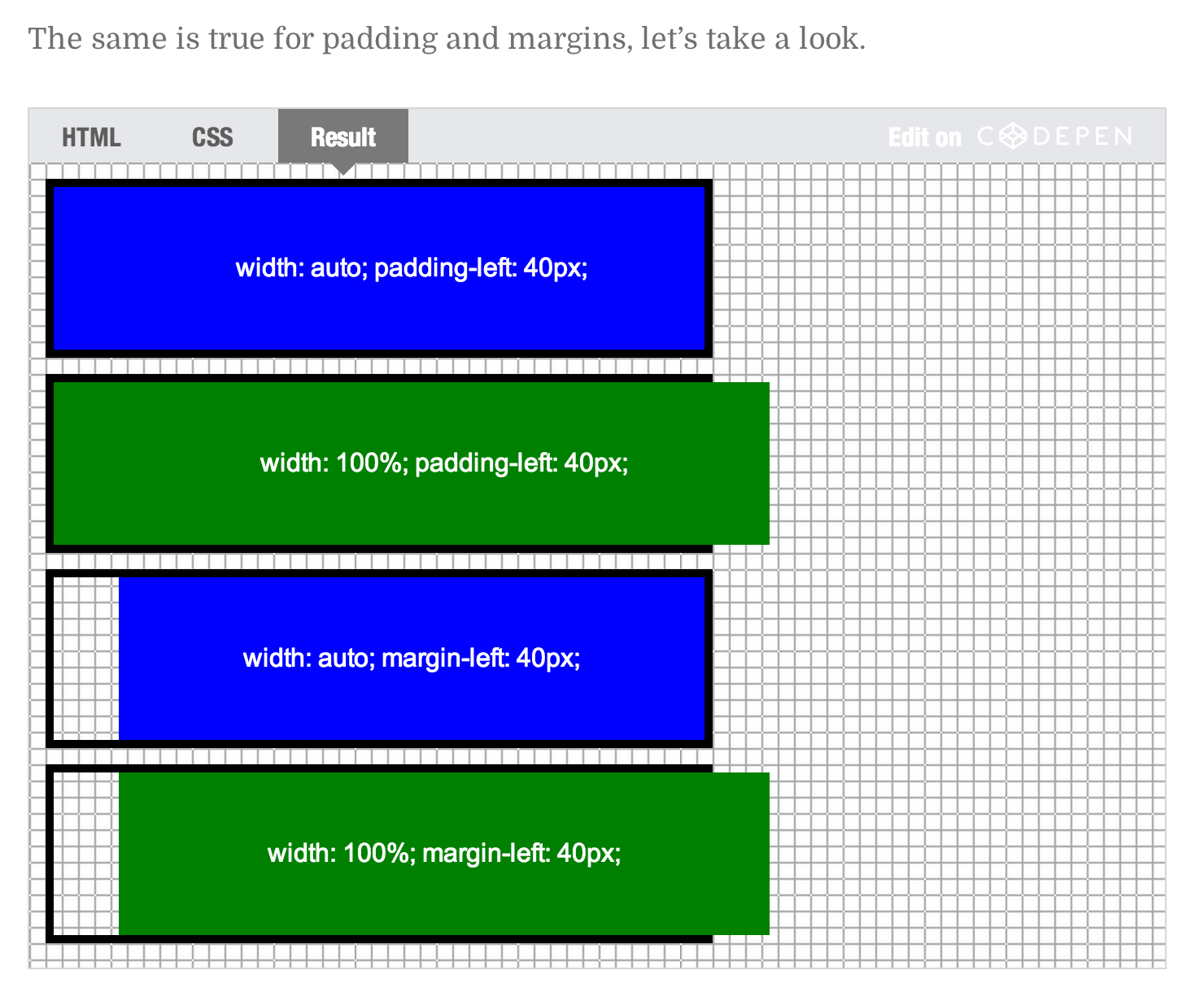
"list-style-position: outside;" This is default:

list-style-position: inside;

## Remove Default Settings

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

ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}



CSS Tables

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

table {  
    border-collapse: collapse;  
}

If you only want a border around the table, only specify the border property for <table>:

table {  
    border: 1px solid black;  
}

## Striped Tables

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

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

This example demonstrates how to position the table caption.

caption {

caption-side: top|bottom;

}

<table>

<caption>Table 1.1 Customers</caption>

</table>

CSS Layout - The display Property

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

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; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

## 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:

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:

h1.hidden {  
    visibility: hidden;  
}

### Difference between display:none and visiblity: hidden?

**visibility:hidden** hides the element, but it still takes up space in the layout.

**display:none** removes the element from the document. It does not take up any space.

## 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;

}

<h1>This is a heading</h1>

<img src="w3css.gif" width="100" height="140">

<p>Because the image has a z-index of -1, it will be placed behind the text.</p>

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

## Positioning Text In an Image

How to position text over an image:

.container {

position: relative;

}

.topleft {

position: absolute;

top: 8px;

left: 16px;

font-size: 18px;

}

-----

.container {

position: relative;

}

.center {

position: absolute;

left: 0;

top: 50%;

width: 100%;

text-align: center;

font-size: 18px;

}

==================================

Overflow

The CSS overflow property controls what happens to content that is too big to fit into an area.

The overflow property  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
* The auto value is similar to scroll, only it add scrollbars when necessary:
* div {  
      overflow: auto;  
  }

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

## overflow-x and overflow-y

The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

overflow-x specifies what to do with the left/right edges of the content.  
overflow-y specifies what to do with the top/bottom edges of the content.

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

CSS Layout - float and clear

## The float Property

The float property is used for positioning and layout on web pages.

The float property can have one of the following values:

* left - The element floats to the left of its container
* right- The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

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

if both image and text contain in one div then use float

<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim

</p>

img {  
    float: left;  
}

## The clear Property

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

The clear property can have one of the following values:

* none - Allows floating elements on both sides. This is default
* left - No floating elements allowed on the left side
* right- No floating elements allowed on the right side
* both - No floating elements allowed on either the left or the right side
* inherit - The element inherits the clear value of its parent
* The most common way to use the clear property is after you have used a float property on an element.
* When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.
* The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

two different divs using then use clear:float;

.div3 {

float: left;

width: 100px;

height: 50px;

margin: 10px;

border: 3px solid #73AD21;

}

.div4 {

border: 1px solid red;

clear: left;

}

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

.clearfix {

overflow: auto;

}

.img2 {

float: right;

}

<div class="clearfix">

<img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170">

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...</div>

## Equal Width Boxes

\* {  
    box-sizing: border-box;  
}  
  
.box {  
  float: left;  
  width: 33.33%; /\* three boxes (use 25% for four, and 50% for two, etc) \*/  
  padding: 50px; /\* if you want space between the images \*/  
}

**What is box-sizing?**

You can easily create three floating boxes side by side. However, when you add something that enlarges the width of each box (e.g. padding or borders), the box will break. The box-sizing property allows us to include the padding and border in the box's total width (and height), making sure that the padding stays inside of the box and that it does not break.

## Navigation Menu

Use float with a list of hyperlinks to create a horizontal menu:

li {

float: left;

}

li a {

display: inline-block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

=======================

## Center Align Elements

To horizontally center a block element (like <div>), use margin: auto;

.center {  
    margin: auto;  
    width: 50%;  
    border: 3px solid green;  
    padding: 10px;  
}

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

## Center Align Text

To just center the text inside an element, use text-align: center;

.center {  
    text-align: center;  
    border: 3px solid green;  
}

## Center an Image

To center an image, set left and right margin to auto and make it into a block element:

img {  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
    width: 40%;  
}

## Left and Right Align - Using position

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

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

## Left and Right Align - Using float

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

**Note:** If an element is taller than the element containing it, and it is floated, it will overflow outside of its container. You can use the "**clearfix**" hack to fix this (see example below).

.clearfix {

overflow: auto;

}

.img2 {

float: right;

}