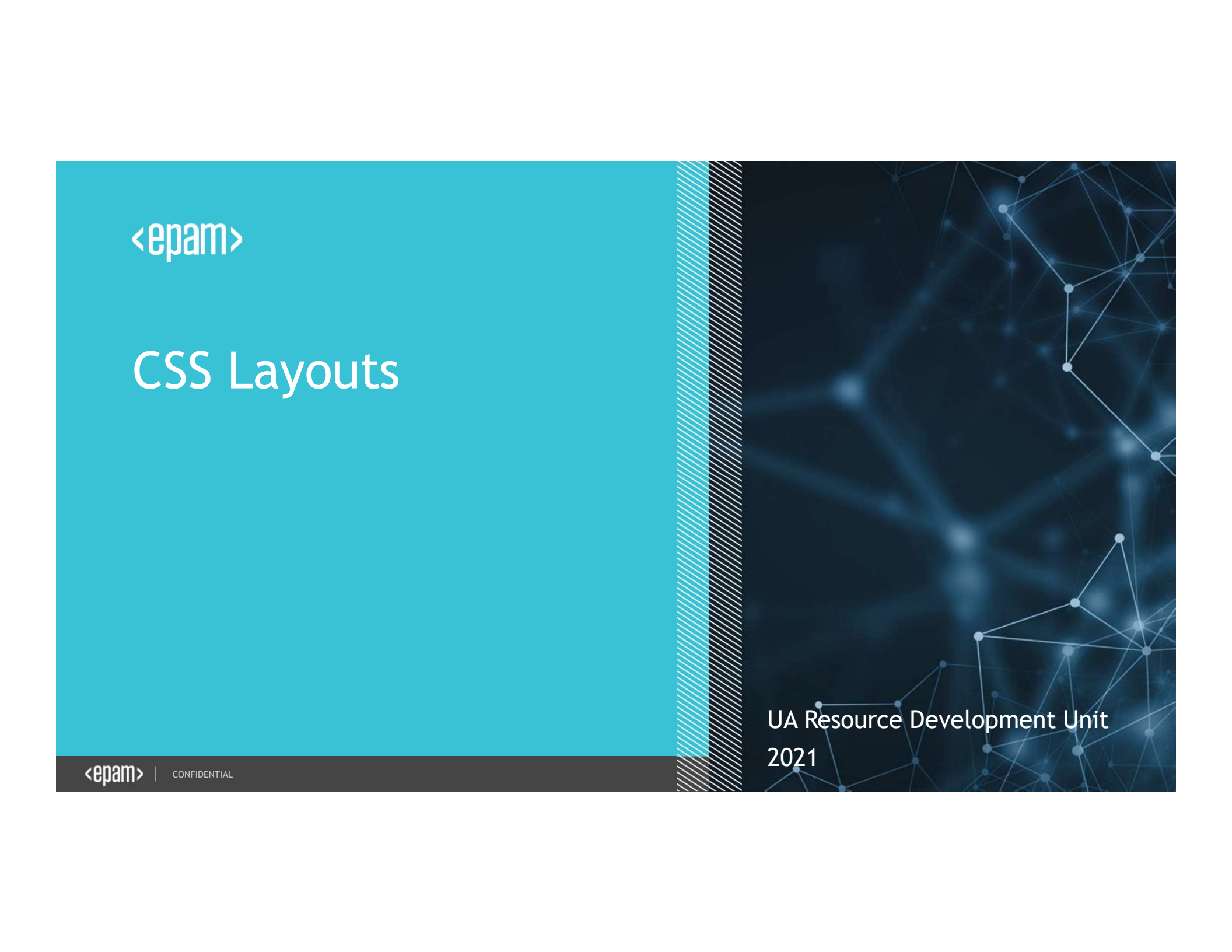
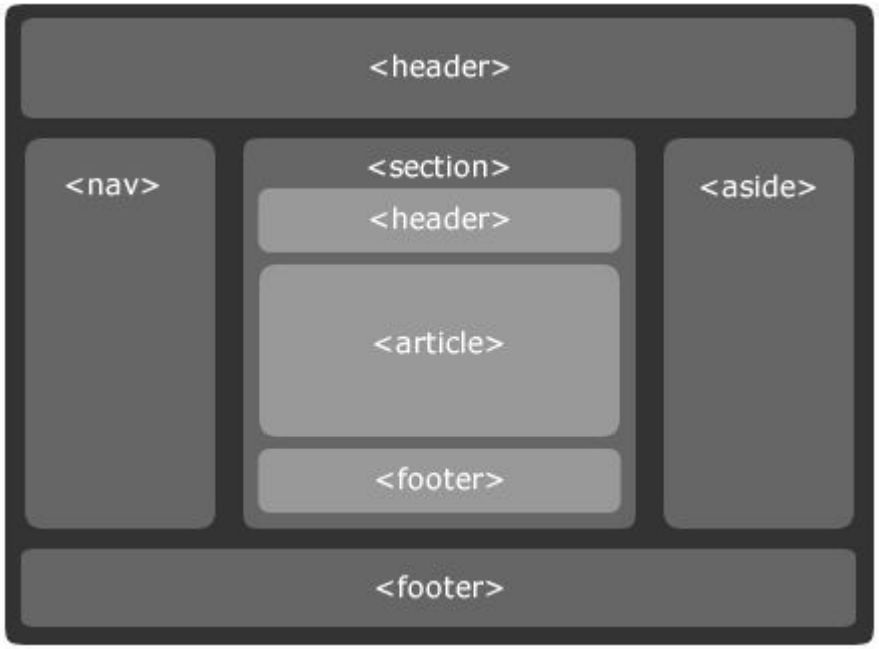

epam
CSS Layouts
UA Resource Development Unit
2021
epam | CONFIDENTIAL

AGENDA

1. Box displays
2. Box model
3. Overflow
4. CSS Calc
5. Positioning

CSS Layouts: Intent



Flow

HTML document contains its own rules:

- **fluidity:** how the content adapts to browser dimensions
- **ordering:** in which order elements appear
- **stacking:** how elements appear on top of each other

Normal document flow

- 100% parent's width and content's height (block)
- content's width and content's height (inline)
- add layers of box model
- vertically - block
- horizontally or wrapped - inline
- margin collapsing

Flow disruption

Some CSS properties allow to disrupt the flow:

- height and width can change element variability;
- float disrupt the behavior of an element, and the elements around it;
- values absolute and fixed of the position property remove the element from the flow;
- z-index can change the order of elements imposition.

Box displays (Part 1)

Display is a CSS rule that helps us redefine the default box treatment of elements.

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

```
// Make any element a block-level element.
This  span will have all the properties of a
block  element and will be taking the whole
container  width.
```

```
div {display: inline;}
```

```
// Make any element an inline-level
element.  This div will lose all it's block
properties.
```

```
a {display: inline-block;}
```

```
// Gives the block properties to an element
but doesn't take up the whole container
width.  Treats spaces within element like
spaces in  the text.
```

```
section {display: table;}
```

```
// Makes a box to behave like a table. Then
it's  children can be styled using properties
like
{display: table-cell}, {display:table-row} etc.
```

Box displays (Part 2)

Display is a CSS rule that helps us redefine the default box treatment of elements.

```
p {display: none;}
```

```
// This just removes an element from
a  document flow.
```

```
div {display: flex;}
```

```
// Enables the flexbox model. Flexbox is a
new layout mode in CSS3. Use of flexbox
ensures that elements behave predictably
when  the page layout must accommodate
different  screen sizes and different
display devices.
```

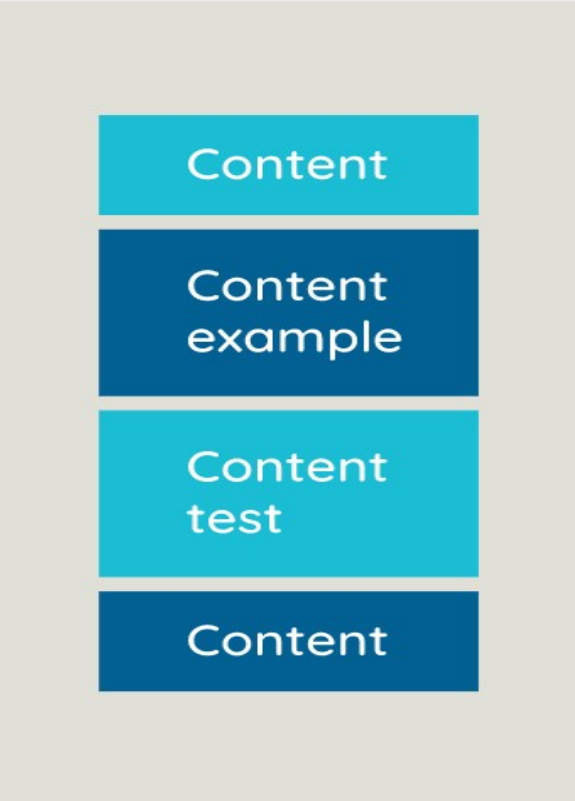
Read more about flexlayout

```
.container { display: grid; }
.container {
  display: inline-grid;
}
```

```
// Enables the grid model. This CSS module
defines a two-dimensional grid-based layout
system, optimized for user interface design.
In the grid layout model, the children of a
grid container can be positioned into
arbitrary  slots in a predefined flexible or
fixed-size  layout grid.
```

Read more about gridlayout

display: block

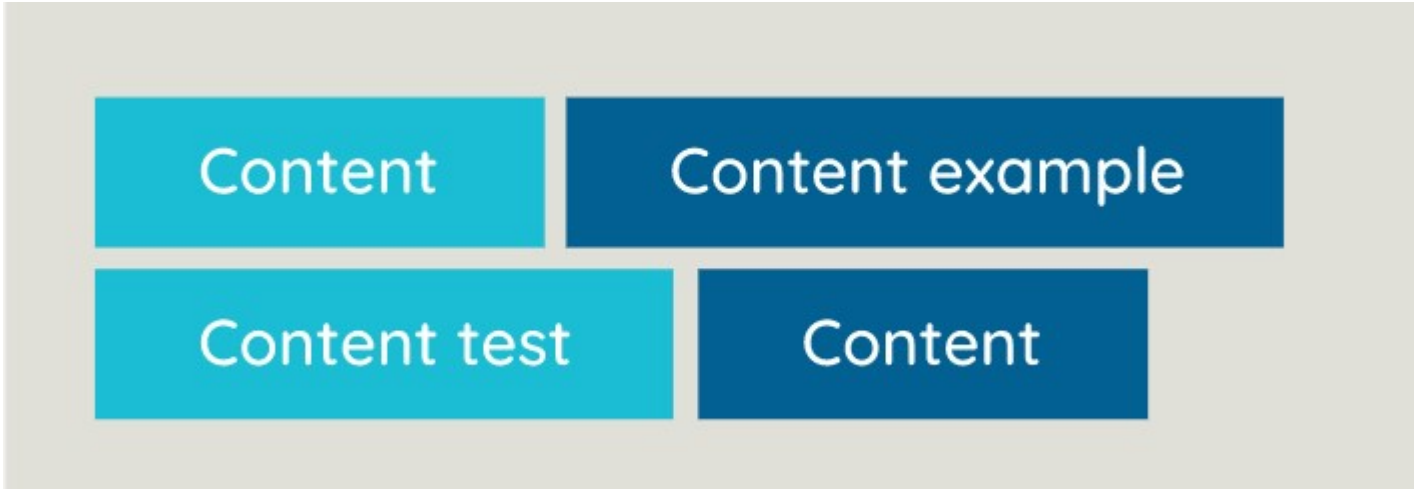



```
#navigation a {
  display: block;
  padding: 20px 10px;
}
```

display: block

- by default: p, div, form, ul, h1 …
- on the row on which the block is located can not add another element, even when there is an empty space
- if the width value is not specified, the element is stretched to the entire parent container
- vertical-align property does not work
- if the height is not specified, expands naturally, to fit the child elements
- can have margin and padding

display: inline

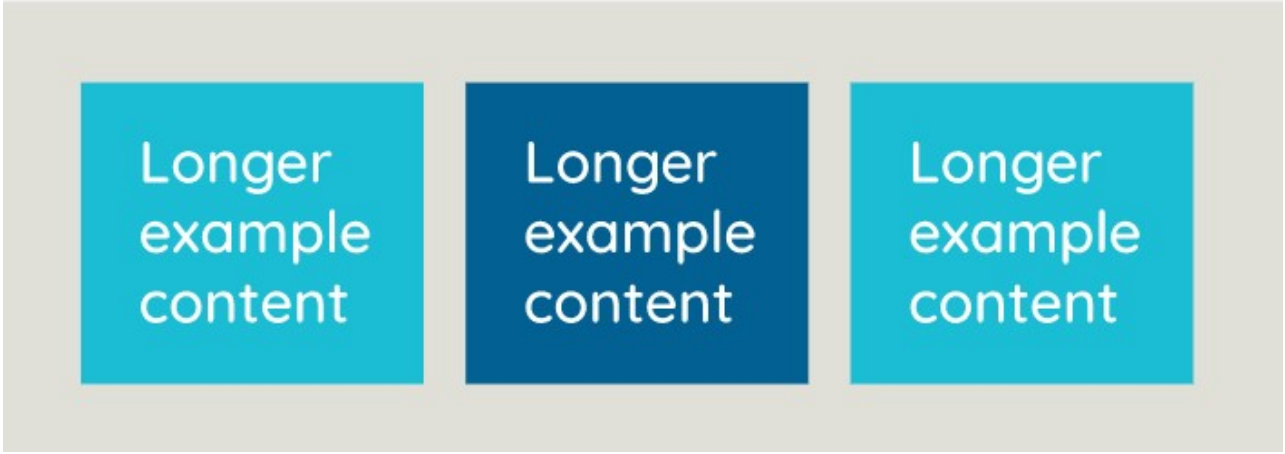



```
li { display: inline }
```

display: inline

- by default: a, span, b, em, etc.
- elements follow each other
- width and height are ignored.
- if the inline element is bordered by a block element, then a line transfer is required between them
- vertical-align property applies
- margin and padding can be applied to the right and left, but margin can not be applied from above and below

display: inline-block




```
.floating-box {
    display: inline-block;
    width: 75px;
    height: 75px;
    margin: 10px;
    border: 3px solid red;
}
```

display: inline-block

- by default: img, input, etc. In relation to external elements behaves like inline, to internal - block.
- elements follow each other
- has width and height, margin-top and margin-bottom.
- if the width value is not specified, it stretches along the width of the longest element inside.

Inherent whitespaces

```
<nav>
     <a href="#">One</a>
     <a href="#">Two</a>
     <a href="#">Three</a>
</nav>
```

```
nav a {
     display: inline-block;
     padding: .5em 1em;
      color: #FFFAD5;
     background-color: #BD4932;
}
```

REMOVE WHITESPACES

```
<nav>
   <a href="#">
   One</a><a href="#">
   Two</a><a href="#">
   Three</a>
</nav>
```

```
<nav>
   <a href="#">One</a
   ><a href="#">Two</a
   ><a href="#">Three</a>
</nav>
```

```
<nav>
   <a href="#">One</a><!--
   --><a href="#">Two</a><!--
   --><a href="#">Three</a>
</nav>
```

```
<nav>
   <a href="#">One
   <a href="#">Two
   <a href="#">Three
</nav>
```

REMOVE WHITESPACES

```
nav {
     font-size: 0;
}
nav a{
      font-size: 16px;
}
```

```
nav a{
     display: inline-block;
     margin-right: -4px;
}
```

display:none or visibility:hidden

display:none

- the value none lets you turn off the display of an element;
- the document is rendered as though the element doesn't exist in the document tree.

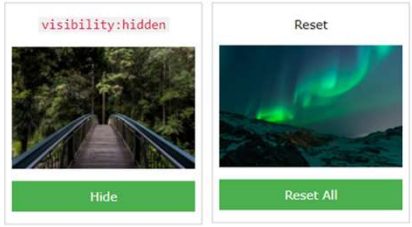
visibility:hidden

- the element will be hidden, but still affect the layout

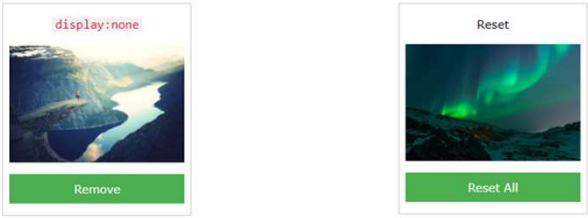
property	Description
display	Specifies how an element should be displayed
visibility	Specifies whether or not an element should be visible

Hidden Elements


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




```
h2.hidden {
    visibility: hidden;
}
```



display: table

```html
<ul class="table">
  <li><a href="#">link 1</a></li>
  <li><a href="#">link 2</a></li>
  <li><a href="#">link 3</a></li>
  <li><a href="#">link 4</a></li>
</ul>
```

Align the list with links in the center of the page horizontally.

```css
ul {
  list-style: none;
  margin: 0;
  padding: 0;
}
li {
  float: left;
  background-color: #BD4932;
}
li a {
  display: inline-block;
  padding: .5em 1em;
  color: #FFFAD5;
}
.table {
  display: table;
  margin: auto
}
```

HTML elements

Block HTML elements

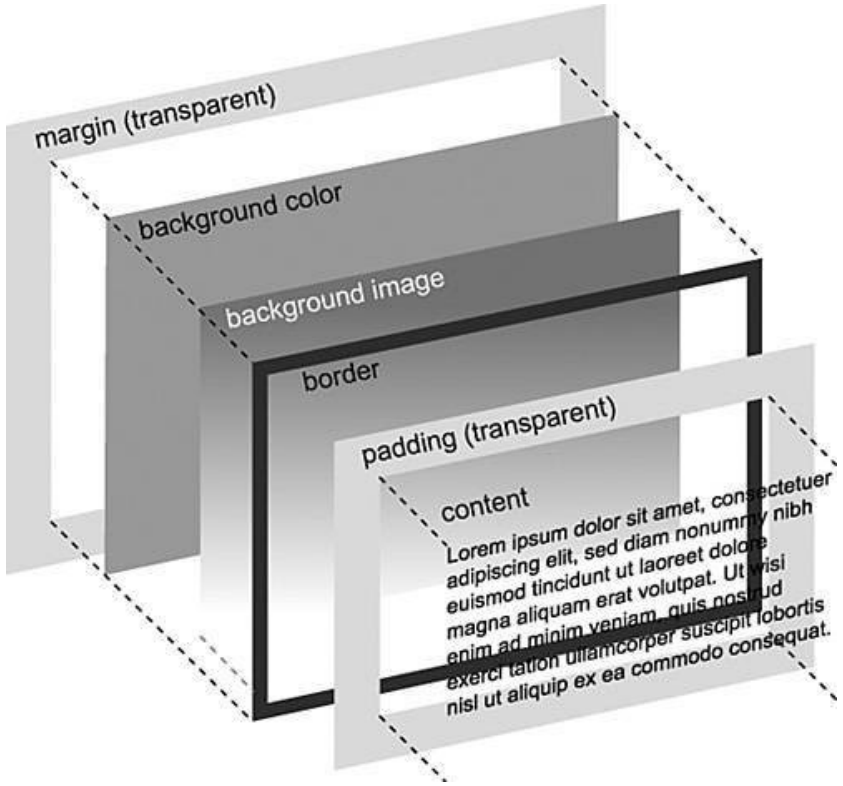
- p
- h1, h2, h3, h4, h5, h6
- ol, ul
- address
- blockquote
- dl
- div
- fieldset
- Form
- table

Inline HTML elements

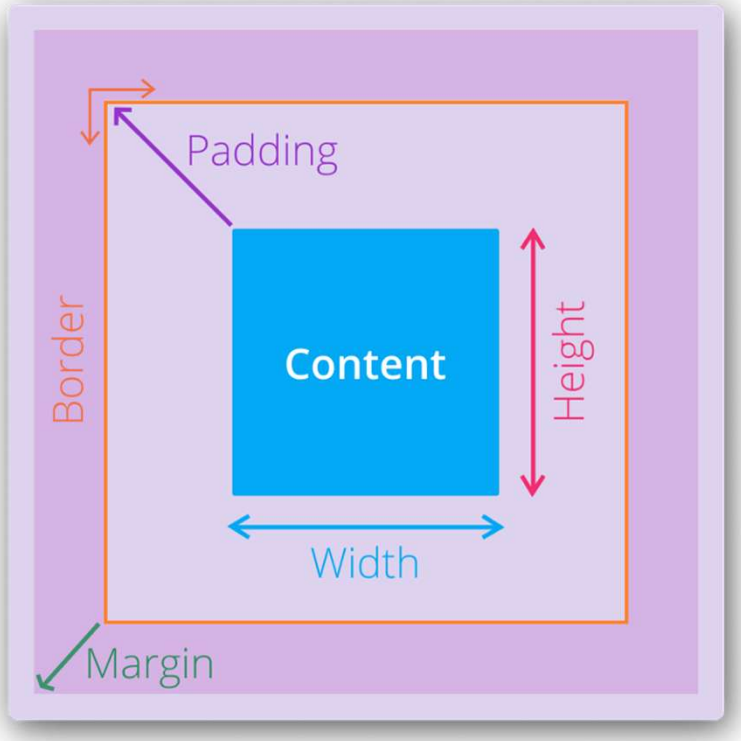
- b, big, i, small,
- abbr, acronym, cite, code, dfn, kbd, strong,
- a, bdo, br, img, q, span, sub, sup
- button, input, label, select, textarea

Box Model

Every element is a rectangle box can be represented with figure:



BOX MODEL



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

Total width

margin-right + border-right + padding-right + width +
padding-left + border-left + margin-left

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

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

Total height

margin-top + border-top + padding-top + height +
padding-bottom + border-bottom + margin-bottom

```
div {
    height: 100px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0;
}
```

100px (height)
+ 20px (top + bottom)
+ 10px (top + bottom border)
+ 0px (top+ bottom margin)
= 130px

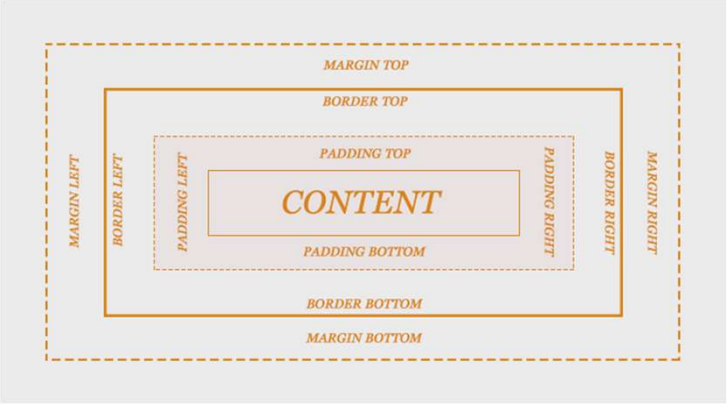
Margin

margin-top: <length> | <percentage> | auto

margin-right: <length> | <percentage> | auto

margin-bottom: <length> | <percentage> | auto

margin-left: <length> | <percentage> | auto



Margin

- margin: [<length> | <percentage> | auto]{1,4} inherit
- margin: all
- margin: vertical horizontal
- margin: top horizontal bottom
- margin: top right bottom left

Margin Collapsing

Top and bottom margins of blocks are sometimes combined into a single margin whose size is the largest of the margins combined into it, a behavior known as **margin collapsing**.

Proof

Parent and first/last child

If there is no border, padding, inline content, or clearance to separate the margin-top of a block with the margin-top of its first child block, or no border, padding, inline content, height, min-height, or max-height to separate the margin-bottom of a block with the margin-bottom of its last child, then those margins collapse. **The collapsed margin ends up outside the parent**.

Proof

Do not collapse

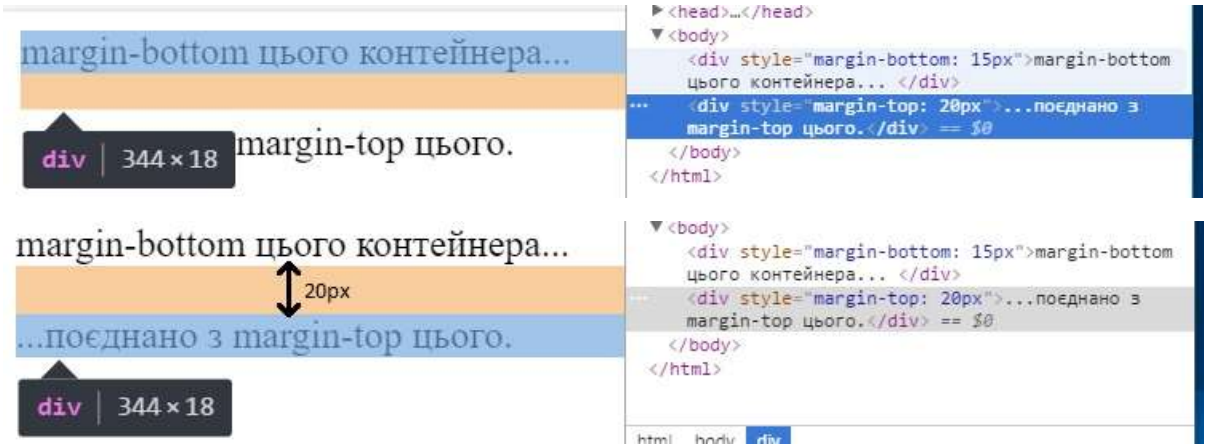
Margins of floating and absolutely positioned elements never collapse.

Margin Collapsing: Adjacent siblings

```
<div style="margin-bottom: 15px">bottom margin of this container... </div>
<div style="margin-top: 15px">...will be combined with top margin of this one.</div>
```

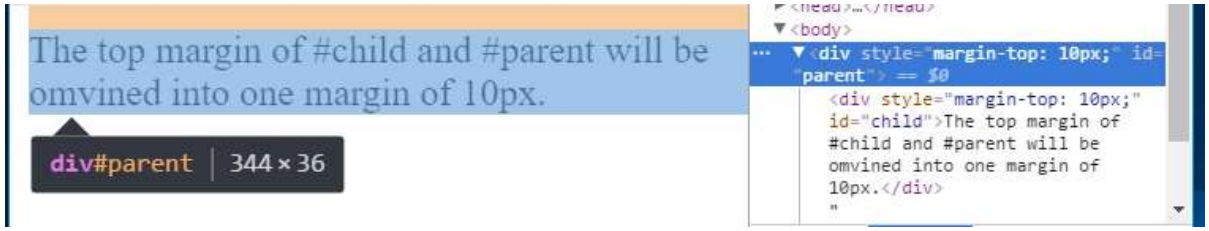
Margin Collapsing: Adjacent siblings

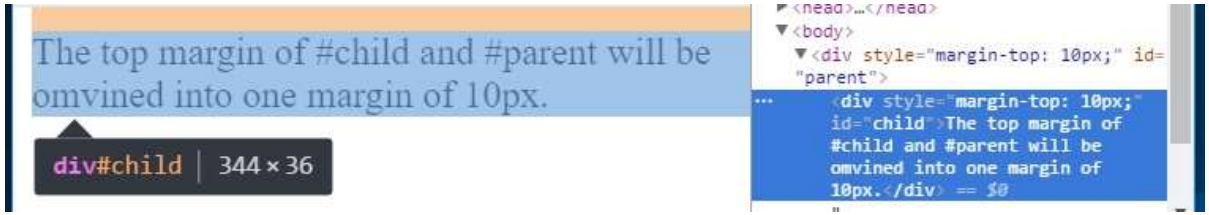
```
<div style="margin-bottom: 15px">bottom margin of this container... </div>
<div style="margin-top: 20px">...will be combined with top margin of this one.</div>
```



Margin Collapsing: Parent and first/last child

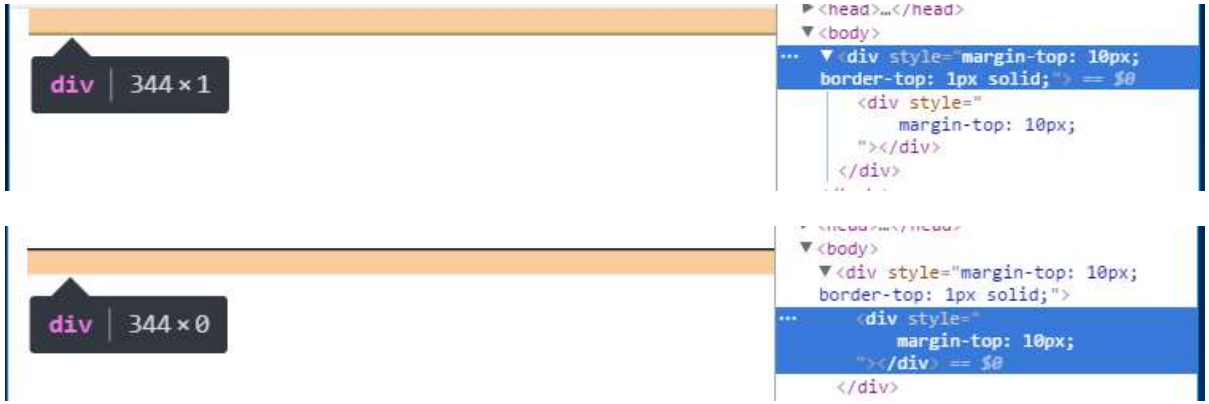
```
<div style="margin-top: 10px;" id="parent">
<div style="margin-top: 10px;" id="child">
The top margin of #child and #parent will be combined into one margin of 10px.
</div>
</div>
```





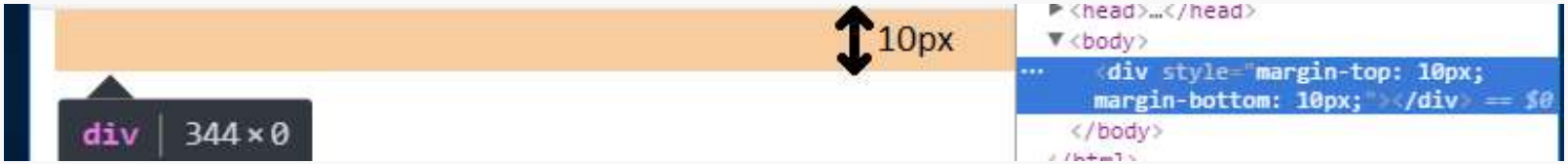
When the parent and first/child margin collapsing will not apply?

- border, padding, inline content or clearance on top
- border, padding, inline-content, height, min-height, max-height on bottom



Margin collapsing: Empty blocks

```
<div style="margin-top: 10px; margin-bottom: 10px;"></div>
```

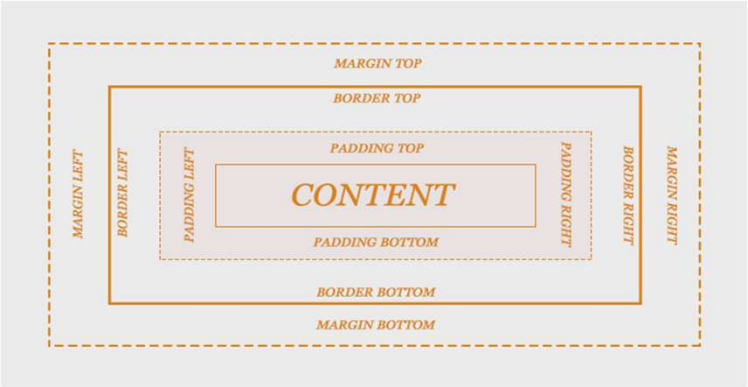


When the empty blocks margin collapsing will not apply?

- border
- padding
- height
- min-height

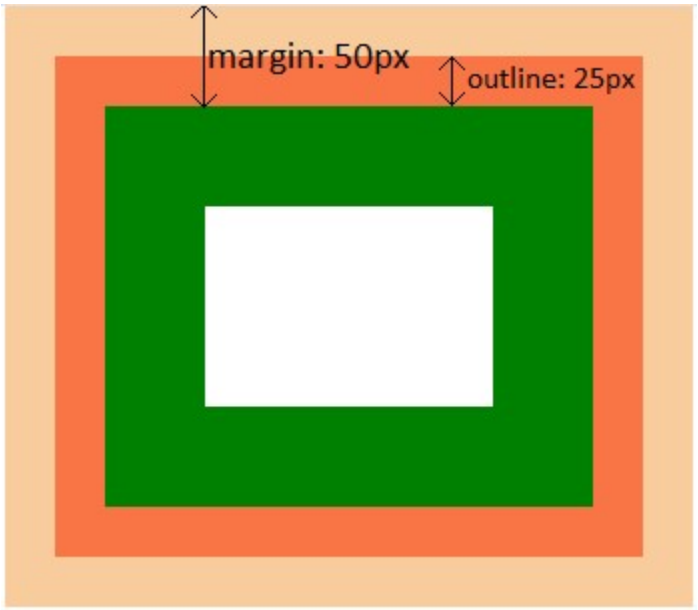
Padding

- padding: [<length> | <percentage>]{1,4}
- padding : all
- padding : vertical horizontal
- padding : top horizontal bottom
- padding : top right bottom left



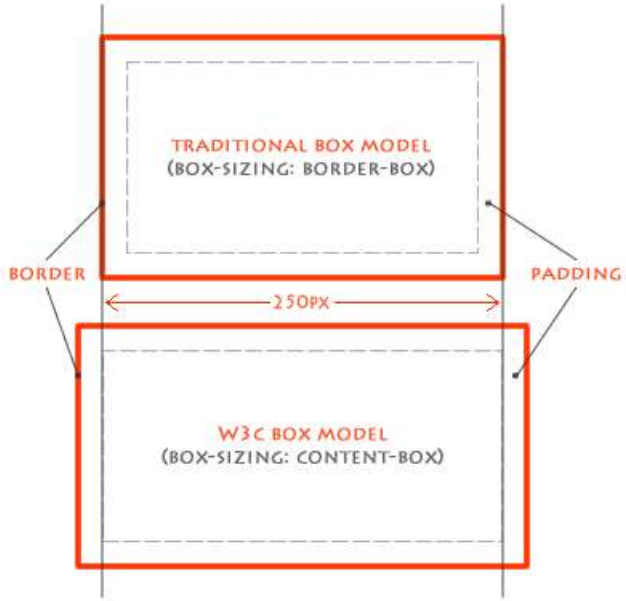
Outline

padding: 50**px**;
border: 50**px solid green**;
margin: 50**px**;
outline: 25**px solid red**;



Box Dimensions

1. Internet Explorer. IE in quirks mode defines box dimensions by adding together dimensions of the content, padding and border.
2. Other browsers define box dimensions by the content width only, by the W3C standards.



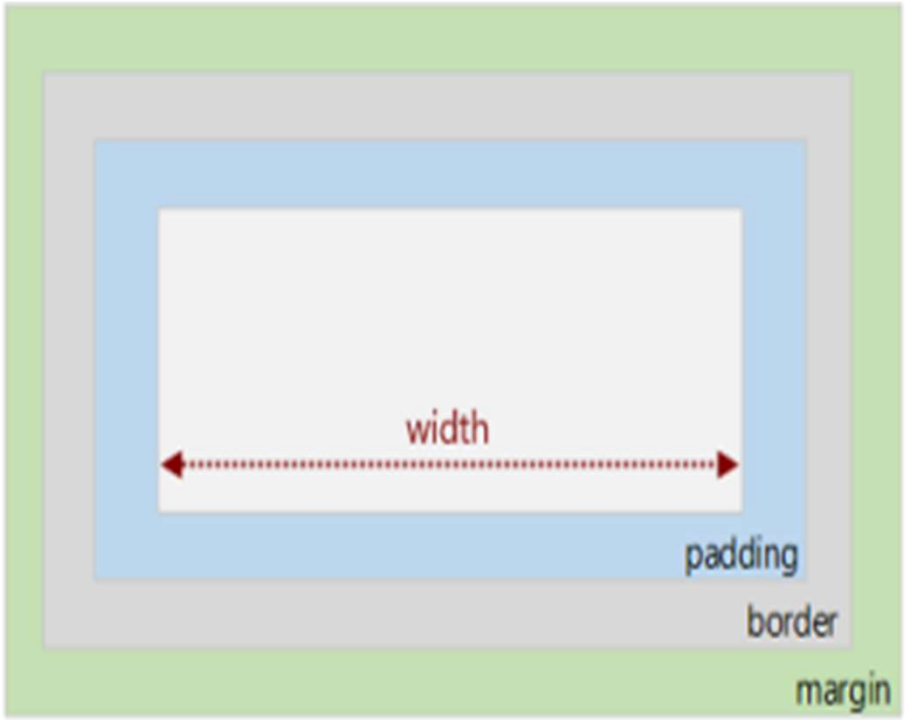
Box-sizing

`content-box`

`padding-box`

`border-box`

BOX-SIZING: CONTENT-BOX



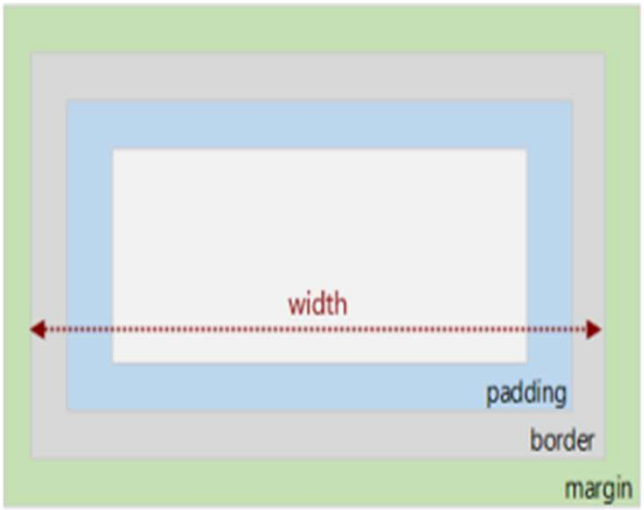
Default. The width and height properties (and min/max properties) includes only the content. Border, padding, or margin are not included

BOX-SIZING: BORDER-BOX

There is a rule in CSS that tells all browsers to follow the more logical IE approach.

```
box-sizing: border-box;
```

This is particularly helpful when dealing with fluid layouts or creating mobile-friendly sites.



The width and height properties (and min/max properties) includes content, padding and border, but not the margin

Live example

CSS Calc

```
/* property: calc(expression)
*/  width: calc(100% - 80px);
```

The calc() CSS function can be used anywhere with length, frequency, angle, time, number, or integer is required.

```
<div class="outer">
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
</div>
```

```
.inner {
    width: calc(99%/3);
    float: left;
    border: 1px solid black;
}
```

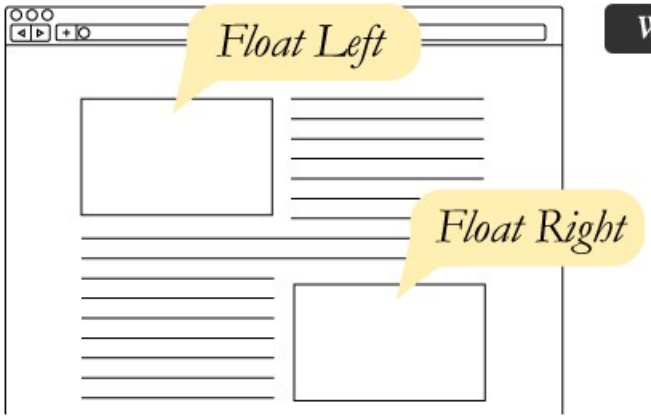
https://css-tricks.com/a-couple-of-use-cases-for-calc/.

https://habr.com/ru/company/ruvds/blog/493660/

https://jsbin.com/konodojici/edit?html,css,output

Floats

The float property specifies whether or not a box (an element) should float.



Web Layout

The float CSS property specifies that an element should be taken from the normal flow and placed along the left or right side of its container, where text and inline elements will wrap around it.

A floating element is one where the computed value of float is not none.

How it looks

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

Demo:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa.

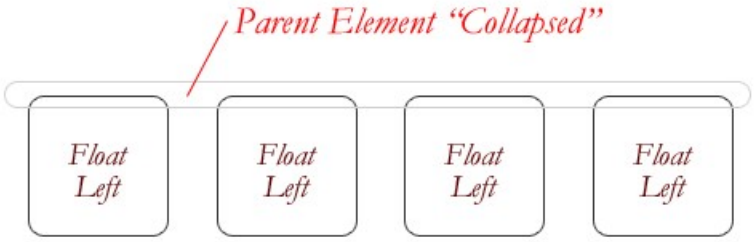

```
<style>
img {
  float: right;
  margin: 0 0 10px 10px;
}
</style>
<p>text</p>
<p>
  <img src="w3css.gif" alt="W3Schools.com" width="100"
  height="140">  Lorem ipsum dolor sit amet, consectetur adipiscing
  elit...
</p>
```

Setting up the float

```
div {
    float: left; // can be the following: "left", "right", "inherit", "none"
    width: 100px;
}
```

The floated element automatically asumess **display: block**.

If the parent element contains only floated elements, it's **height will equal to 0**

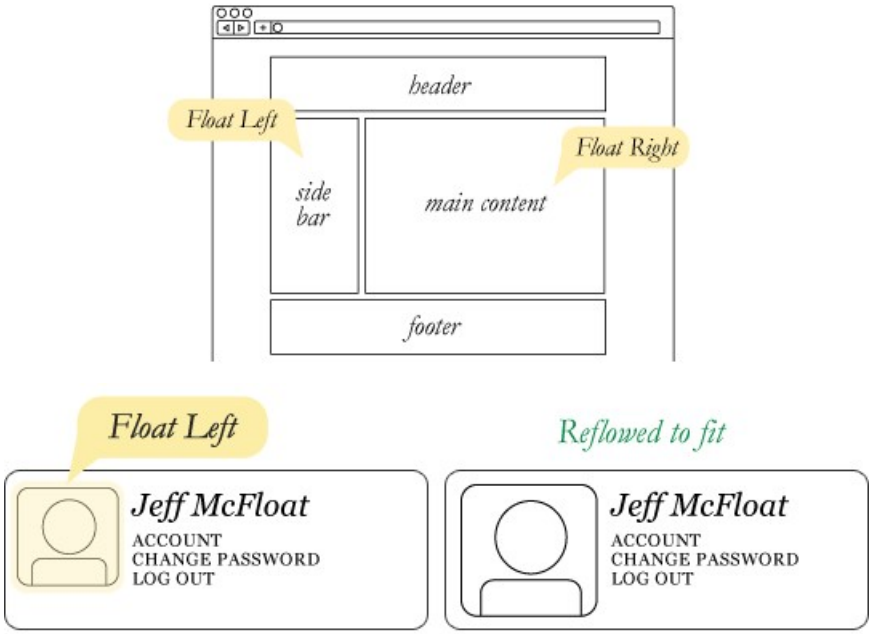


float

float: left | right | none | inherit

Value	Description
none	The element is not floated, and will be displayed just where it occurs in the text. This is default
left	The element floats to the left
right	The element floats the right
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

When do we use floats?



Clearing floats

To solve the issue of collapsing parents we can use several techniques:

```
.parent {overflow: hidden;}
```

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

```
.parent:after {
   content: "";
   visibility: hidden;
   display: block;
   height: 0;
   clear: both;
}
```

Further reading at
https://css-tricks.com/is-css-float-deprecated/

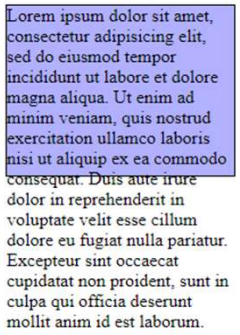
Overflow

Specifies what happens if content overflows an element's box

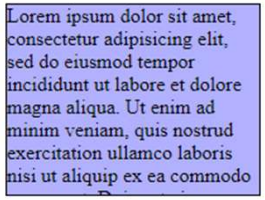
The overflow property only works for block elements with a specified height.

Value	Description
visible	The overflow is not clipped. It renders outside the element's box. This is default
hidden	The overflow is clipped, and the rest of the content will be invisible
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content
auto	If overflow is clipped, a scroll-bar should be added to see the rest of the content
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

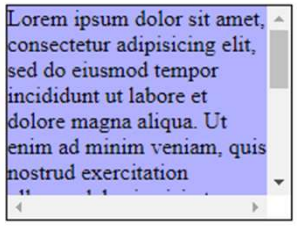
Overflow



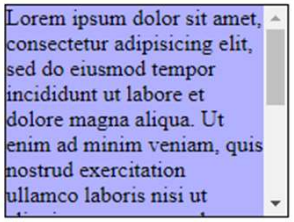

```
div {
    width: 200px;
    height: 50px;
    background-color: rgba(100,100,255,.5);
    overflow: visible;
}
```




```
div {
    overflow: hidden;
}
```




```
div {
    overflow: scroll;
}
```




```
div {
    overflow: auto;
}
```

clear

Specifies on which sides of an element where floating elements are not allowed to float

```
div {
    clear: left;
}
```

none	Default. Allows floating elements on both sides
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
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

The clearfix Hack

Without Clearfix

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

With Clearfix

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

The clearfix Hack

```
.clearfix::after {
  content : "";
  visibility : hidden;
  display : block;
  height : 0;
  clear : both;
}
```

```
.clearfix:before,
.clearfix:after {
  content: "";
  display: table;
  clear: both;
}
```

```
.clearfix {
    overflow: auto;
}
```

```
.clearfix {
    overflow: hidden;
}
```

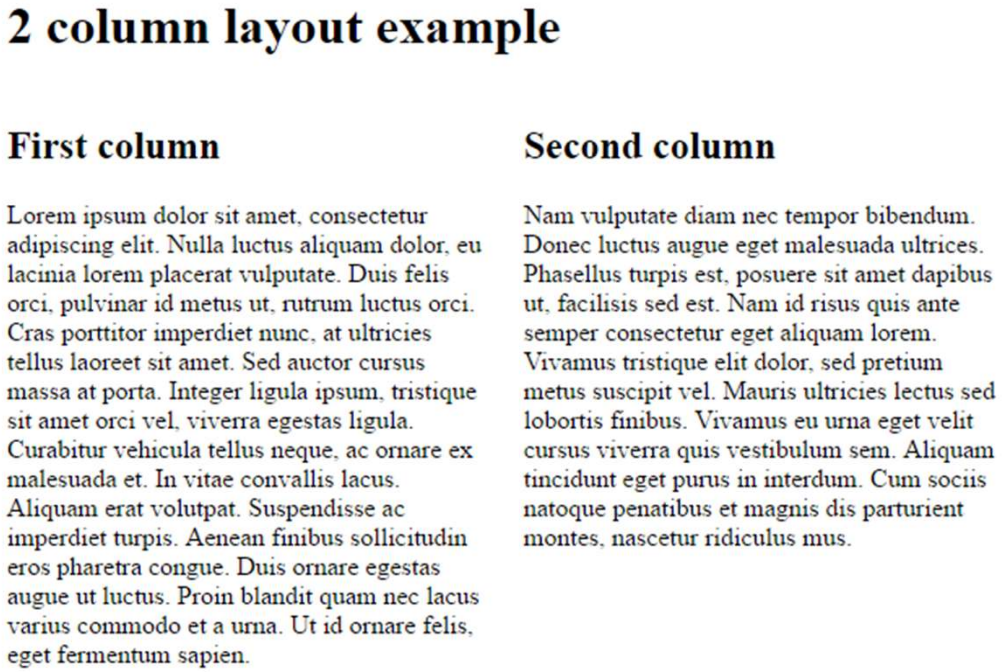
float

- Absolutely positioned elements ignores the float property!
- Elements after a floating element will flow around it. To avoid this, use the clear property or the clearfix hack.
- As float implies the use of the block layout, it modifies the computed value of the display values in some cases

Specified value	Computed value
inline	block
inline-block	block
inline-table	table
table-row	block...

Floating: two columns

```
div:nth-of-type(1) {
width: 48%;
float: left;
}
div:nth-of-type(2) {
width: 48%;
float: right;
}
```



2 column layout example

First column

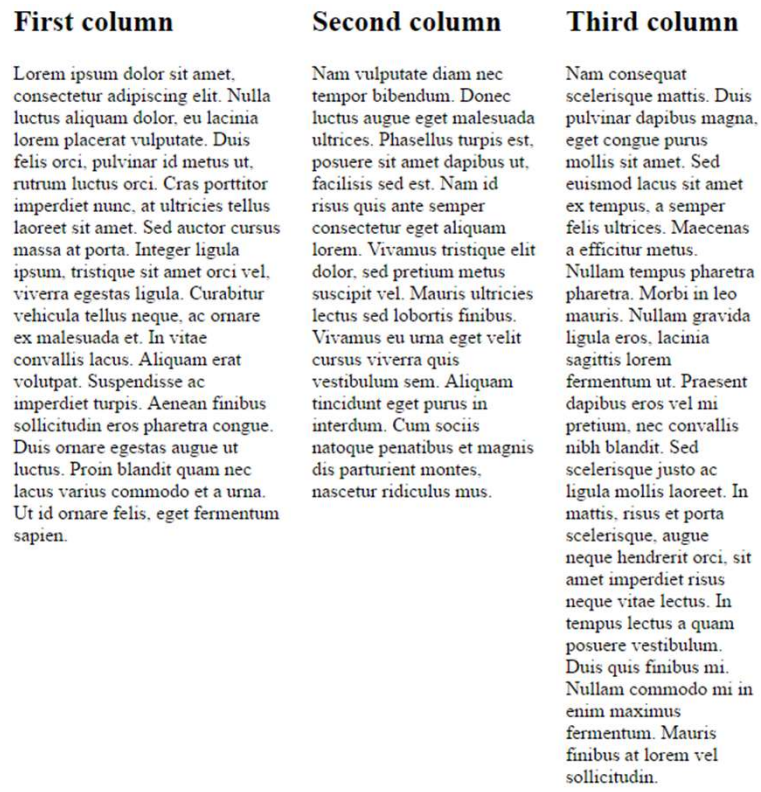
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Second column

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Floating: three columns

```
div:nth-of-type(1) {
width: 36%;
float: left;
}
div:nth-of-type(2) {
width: 36%;
float: left;
}
div:nth-of-type(3) {
width: 26%;
float: right;
}
```



First column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Second column

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Third column

Nam consequat scelerisque mattis. Duis pulvinar dapibus magna, eget congue purus mollis sit amet. Sed euismod lacus sit amet ex tempus, a semper felis ultrices. Maecenas a efficitur metus. Nullam tempus pharetra pharetra. Morbi in leo mauris. Nullam gravida ligula eros, lacinia sagittis lorem fermentum ut. Praesent dapibus eros vel mi pretium, nec convallis nibh blandit. Sed scelerisque justo ac ligula mollis laoreet. In mattis, risus et porta scelerisque, augue neque hendrerit orci, sit amet imperdiet risus neque vitae lectus. In tempus lectus a quam posuere vestibulum. Duis quis finibus mi. Nullam commodo mi in enim maximus fermentum. Mauris finibus at lorem vel sollicitudin.

Positioning

- override normal flow
- change element's position
- place one element over another one
- fix element's position

in a browser window when a page is scrolled

position

position: static | relative | absolute | fixed | sticky

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

position: static;

HTML elements are positioned static by default.

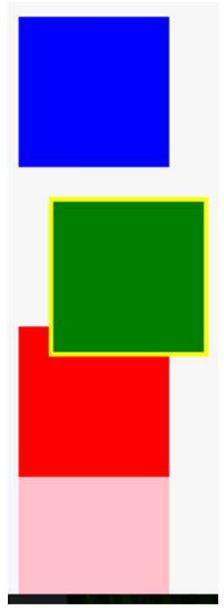
Static positioned elements are not affected by the top, bottom, left, and right properties.

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

relative

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

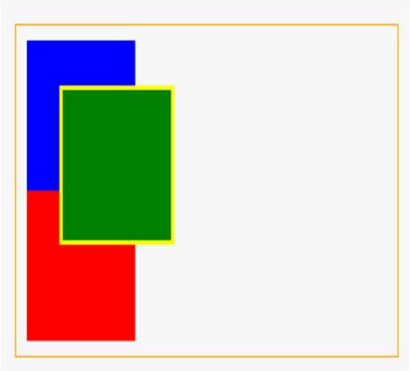
Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.



```
.box2 {
  width: 100px;
  height: 100px;
  background: green;
  border: yellow solid;
  position: relative;
  left: 20px;
  top: 20px;
}
```

absolute

An element with position: absolute; is positioned relative to the nearest positioned ancestor.
However; if an absolute positioned element has no positioned ancestors,
it uses the document body, and moves along with page scrolling.
A "positioned" element is one whose position is anything except static.
Element has display:block

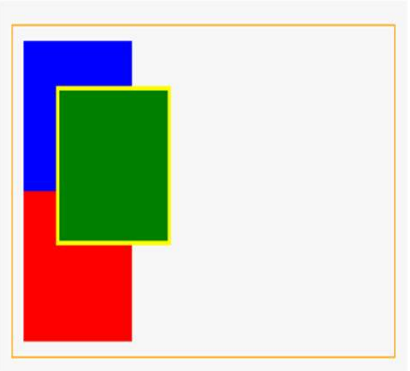


absolute

```
.container {
  position: relative;
  margin: 30px;
  border: solid 1px
orange;
  padding: 10px;
}

.box1 {
  width: 100px;
  height: 100px;
  background: blue;
}
```

```
.box2 {
  width: 100px;
  height: 100px;
  background: green;
  border: yellow
solid;
  position: absolute;
  left: 40px; top:
40px;
}
```



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.

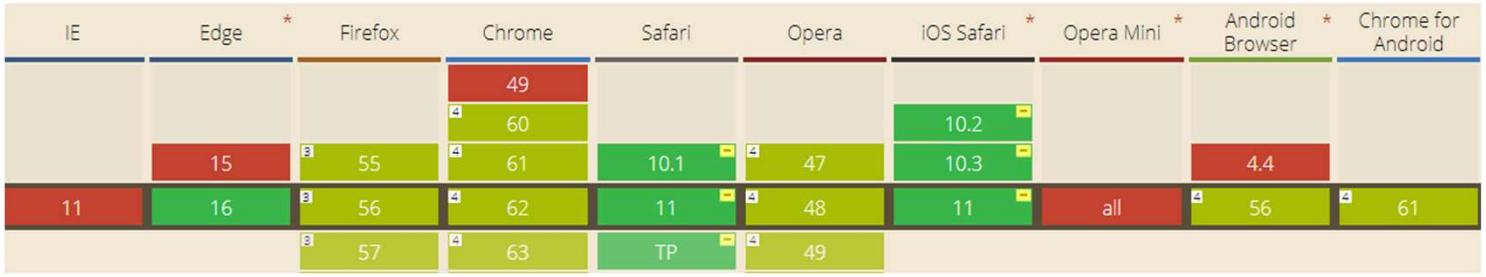
A fixed element does not leave a gap in the page where it would normally have been located.

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

sticky

An element with `position: sticky;` is positioned based on the user's scroll position.
A sticky element toggles between `relative` and `fixed`, depending on the scroll position.
It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed)

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



POSITION: ABSOLUTE AND NEGATIVE MARGIN-TOP


```
.content {
        position: absolute;
        top: 50%;
        margin-top: -8px;
    height : 16px
      }
```

VERTICAL ALIGNMENT

```
<div class="box">
        <span>
                <!-- content -->
        </span>
</div>
```

```
.box {
      height: 200px;
      width: 200px;
      white-space: nowrap;
}
.box:after {
      content: '';
      display: inline-block;
      height: 100%;
      width: 1px;
      overflow: hidden;
      margin: 0 0 0 -5px;
      vertical-align: middle;
}
  .box span {
        vertical-align: middle;
        display: inline-block;
        white-space: normal;
  }
```

LINE-HEIGHT PROPERTY

```
<div class="box">
        <span>
                <!-- content -->
        </span>
</div>
```

```
span {
        height: 100px;
        line-height: 100px;
}
```

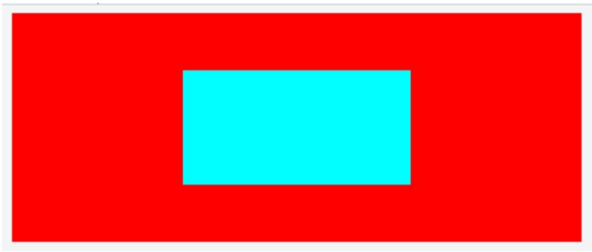
HORIZONTAL ALIGNMENT


```
.box {
        margin: 0 auto;
        width: 200px;
}
```

TEXT-ALIGN: CENTER; AND DISPLAY: INLINE-BLOCK;

```
<div class="box">
    <div class="holder">
        <!-- content -->
    </div>
</div>
```

```
.box {
    text-align: center;
}
.holder {
    display: inline-block;
}
```

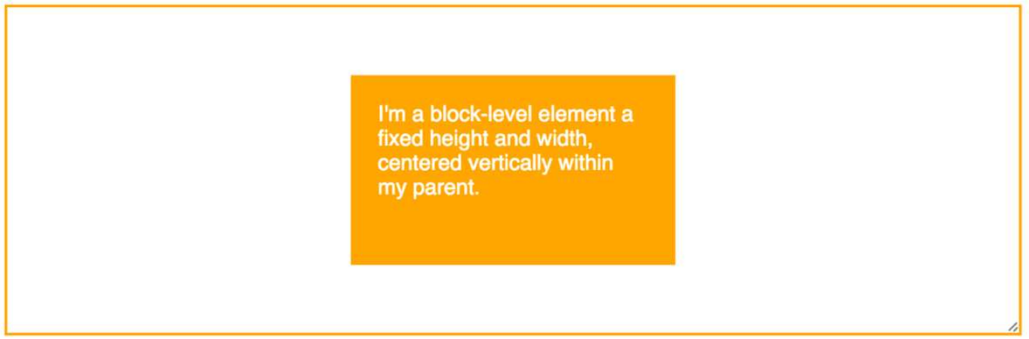


POSITION: ABSOLUTE;

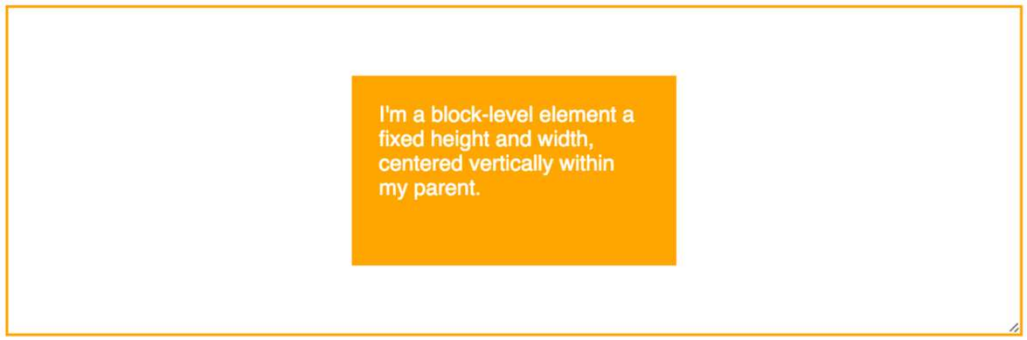

```
.parent {
  position: relative;
}
.box {
    width: 200px;
    height: 100px;
     position: absolute;
     left: 50%;
     margin-left: -100px;
}
```

BOTH VERTICAL & HORIZONTAL

```
.parent {
  position: relative;
}
.child {
  width: 200px;
  height: 100px;
  position: absolute;
  top: 50%;
  left: 50%;
  margin: -50px 0 0 -100px;
}
```



BOTH VERTICAL & HORIZONTAL

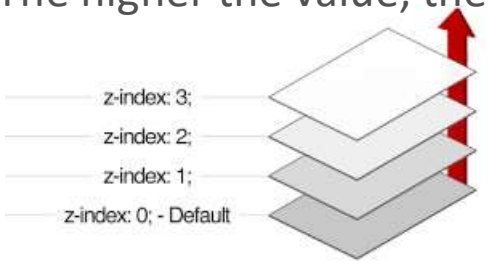


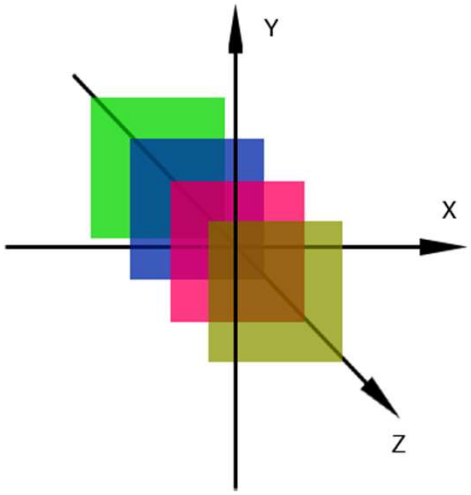

```css
.child {
  width: 300px;
  height: 100px;
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin: auto;
}
```

Z-Index

Z-index controls the vertical stacking of non-static blocks.
The higher the value, the closer the item is on stack to the user.

```
div {
  z-index: 1;
}
```





z-index: auto|*number*|initial|inherit;

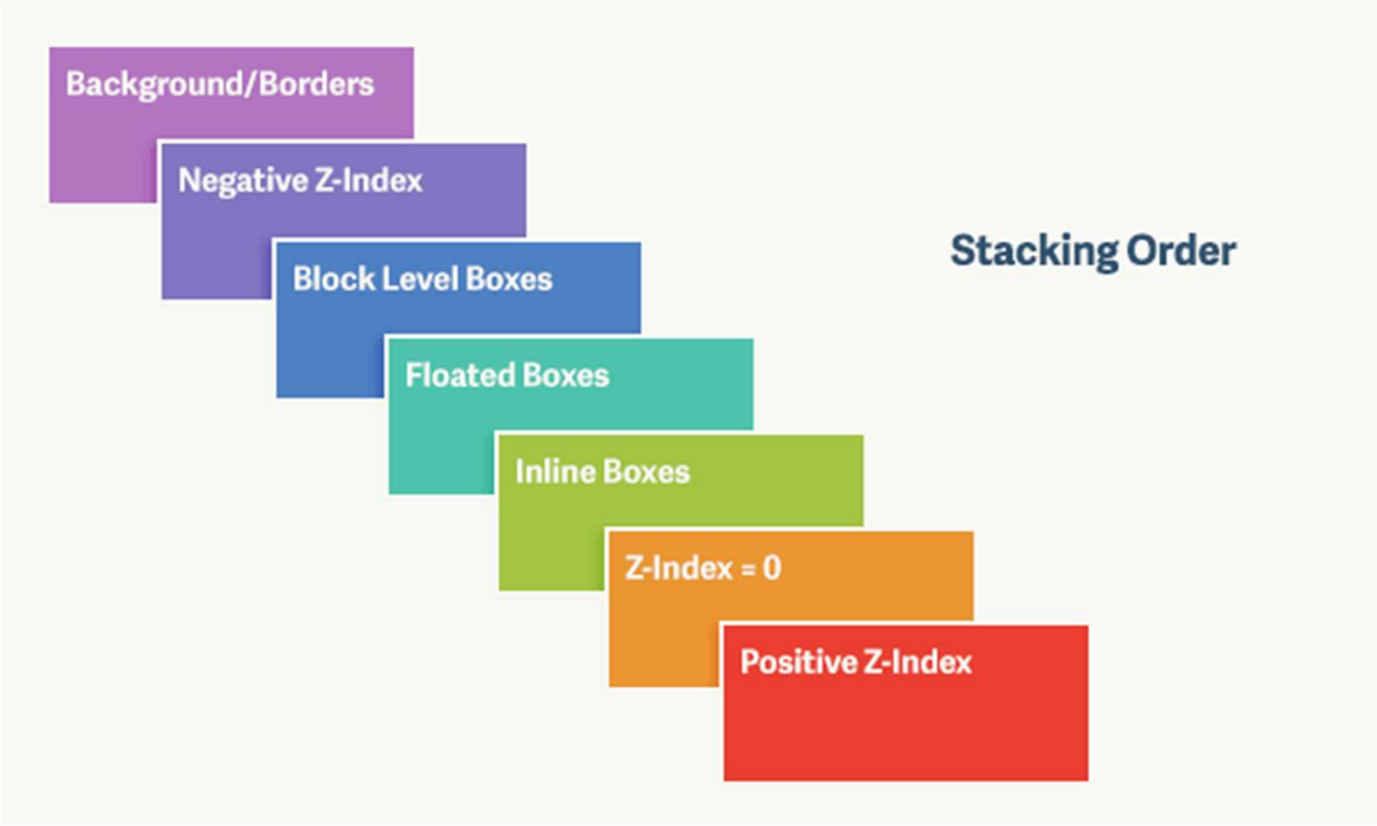
position: absolute | fixed | relative

Example 1

Example 2

Further reading at css-tricks

Stacking Order

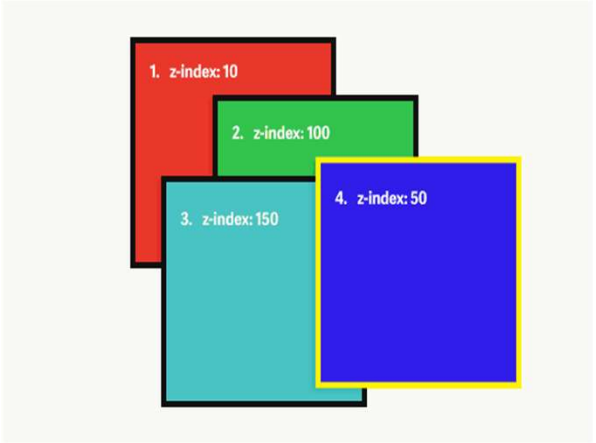


example

```
<div class="one">
  <div class="two"> </div>
<div class="three"></div>
</div>
<div class="four"></div>
```

```
div {
  width: 200px;
  height: 200px;
  padding: 20px;
}
.one, .two, .three, .four {
  position: absolute;
}
.one {
  background: #f00;
  outline: 5px solid #000;
  top: 100px;
  left: 200px;
  z-index: 10;
}
```

```
.two {
  background: #0f0;
  outline: 5px solid #000;
  top: 50px;
  left: 75px;
  z-index: 100;
}
.three {
  background: #0ff;
  outline: 5px solid #000;
  top: 125px;
  left: 25px;
  z-index: 150;
}
.four {
  background: #00f;
  outline: 5px solid #ff0;
  top: 200px;
  left: 350px;
  z-index: 50;
}
```



Modern layouts: intent

- size
- order

may depend on

- content
- screen resolution
- screen orientation

Related resources & Useful links

General

- CSS-Tricks Almanac
- Learn Layout
- .clear-fix

Flex Module

- FlexBox Cheatsheet
- Flexbox Playground
- Practical tricks

Grid Module

- Grid Cheatsheet
- Grid by Example
- CSS Fractional Unit in a simple way
- Grid Layout examples+

- Tutorials at **smashingmagazine.com**
- Presentation about work with retina displays **pepelsbey.net**
- The first famous article about responsive web design **alistapart.com**
- Responsive design inspiration at **mediaqueri.es**
- Image scaling trick with aspect ratio
- CSS-Tricks Almanac
- Codecademy CSS lessons Learn Layout
- Fun with flexbox

Q&A

UA Frontend Online LAB