

CHAPTER 7 – CSS PT3: BOX MODEL + POSITIONING



TOPICS TO BE COVERED

- » **Box Model**

For styling and layout HTML elements.

- » **Display and Positioning**

Adjusting the position of HTML elements.

More info on this topic can be found in e-books provided in LMS.
Presentation slides made by Fifah S., M. Khalid.



BOX MODEL

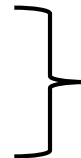
- » A concept to understand how elements are positioned and displayed on a web page.
- » Box Model can be applied to different block elements.
- » Difference between block elements:

```
<p>Test</p>  
<p>Test</p>  
<a href="">test</a>  
<a href="">test</a>
```

Test

Test

test test



Block element



Inline element

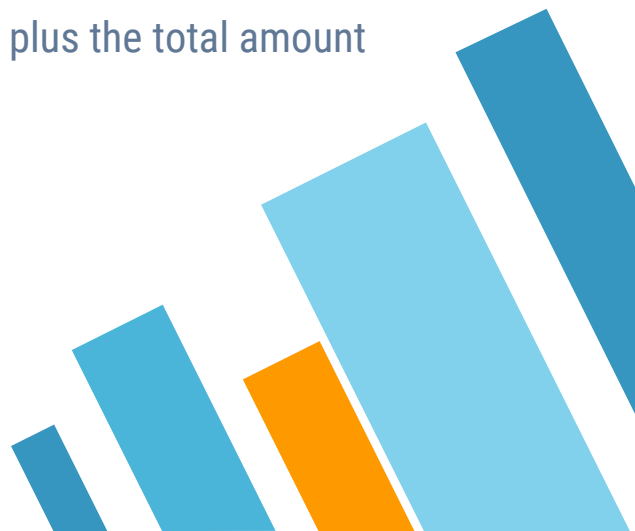
- » Block element takes 100% of the width by default, whereas inline does not.

BOX MODEL





CONTENT AREA

- » At the core of the element box is the content itself, which is indicated by text in a white box.
 - » Content area can be presented by paragraph, headings, images, videos and a lot more HTML elements.
 - » N.B. The total size of an element box includes the content area plus the total amount of padding, borders, and margin applied to the element.
- 



BOX DIMENSION

- » By default, the width and height of a **block** element is calculated automatically by the browser (default *auto* value).
- » CSS3 provides two ways to specify the size of an element:

- 1. Apply the width and height values to the content box.**

Resulting size = dimensions (w+h) + padding + borders

- 2. Box sizing property in CSS3**

Applies the width and height values to the border box (content, padding and border). Hence resulting in exact dimensions you specify.

*You can only specify the width and height for block-level elements and non-text inline element such as images.



WIDTH + HEIGHT

- » By default, **box-sizing: content-box**, where we can apply the width and height properties of the content area.

```
p, a{  
  background-color: purple; ●  
  width: 50%;  
  height: 40px;  
}
```

Test

Test

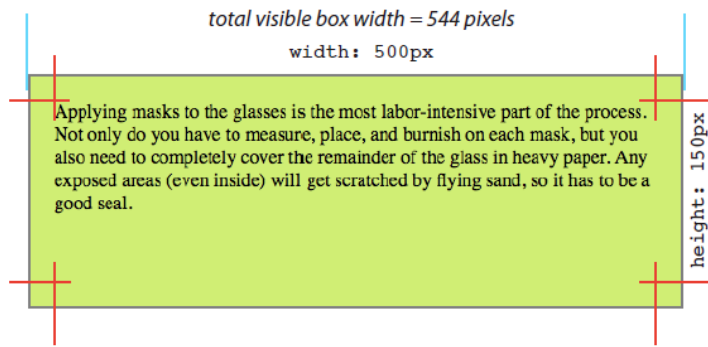
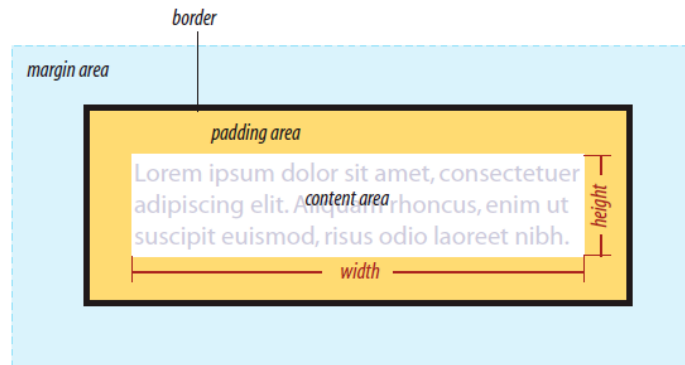
40px 50%

- » Since it only applies to **block** element, only it will be affected. Inline element (i.e. `<a>` will not be affected).
- » Width and height value can be in percentage (%) or pixels (px).

WIDTH + HEIGHT

```
p {  
  background: green; •  
  width: 500px;  
  height: 150px;  
  padding: 20px;  
  border: 2px solid gray; •  
  margin: 20px;  
}
```

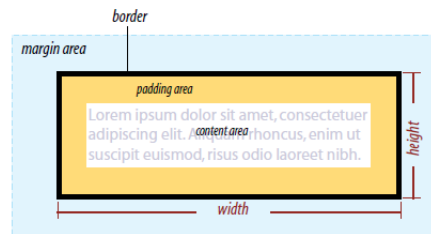
- » Total visible box width = 544px
= margin + border + padding + width + border
= 20px + 2px + 20px + 500px + 2px
= **544px**



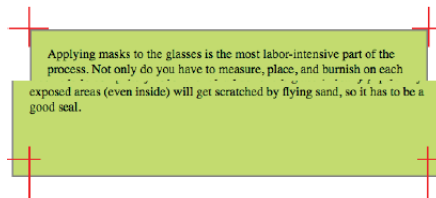
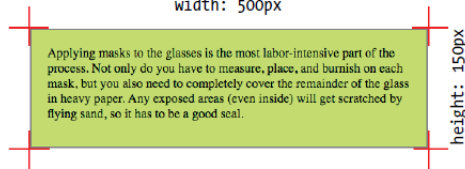
BORDER-BOX

- » Applying width and height dimensions to the entire visible box, including the padding and border.
- » You may need to set **box-sizing: border-box;**

```
p {  
  background: green;  
  -webkit-box-sizing: border-box; /*for Chrome, Safari */  
  -moz-box-sizing: border-box; /*for Mozilla */  
  box-sizing: border-box;  
  width: 500px;  
  height: 150px;  
}
```



total visible box width = 500 pixels
width: 500px



box-sizing: border-box;
width: 500;

box-sizing: content-box;
width: 500;

OVERFLOW

- » When an element is set to size that is too small for its contents, use **overflow** property to specify what to do with the content.

visible

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass in heavy paper. Any exposed areas (even inside) will get scratched by the flying sand, so it has to be a good seal.

Overflow: visible

A default value, allows the content to hang out over the element box.

hidden

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely

Overflow: hidden

The content that does not fit gets clipped off.

scroll

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also

Overflow: scroll

Scrollbars are added to element box.

auto (short text)

Applying the masks to the glasses is the most labor-intensive part of the process.

Overflow: auto

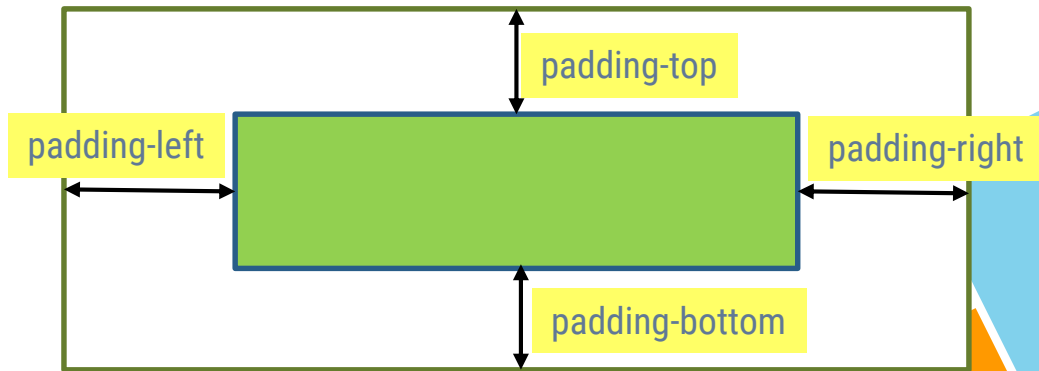
Allows the browsers to decide how to handle overflow. In most cases, scrollbars are added only when the content doesn't fit and are needed.

auto (long text)

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also

PADDING

- » The space between the content area and the border. It gives the content a little breathing room, and prevents the border or edge of the background from bumping right up against the text.
- » There are FOUR types of padding properties:
 - ◇ padding-top
 - ◇ padding-bottom
 - ◇ padding-right
 - ◇ padding-left



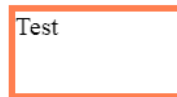
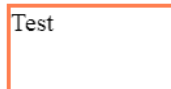
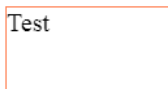
BORDER

- » Specifies the thickness and style of the border surrounding the content area and padding.

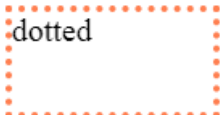
```
border: 3px solid coral; ●
```

- » Attributes:

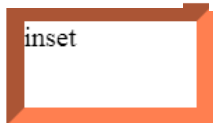
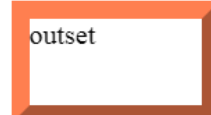
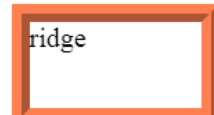
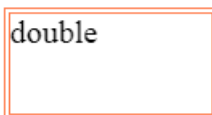
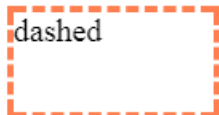
- ◇ **color** – the color of the border.
- ◇ **width** – the thickness of the border. It is either *thin*, *medium*, *thick* or in px, cm, pt, em.



- ◇ **style** – the design of the border. It can be *none* (default), *dotted*, *solid*, *inset* etc.

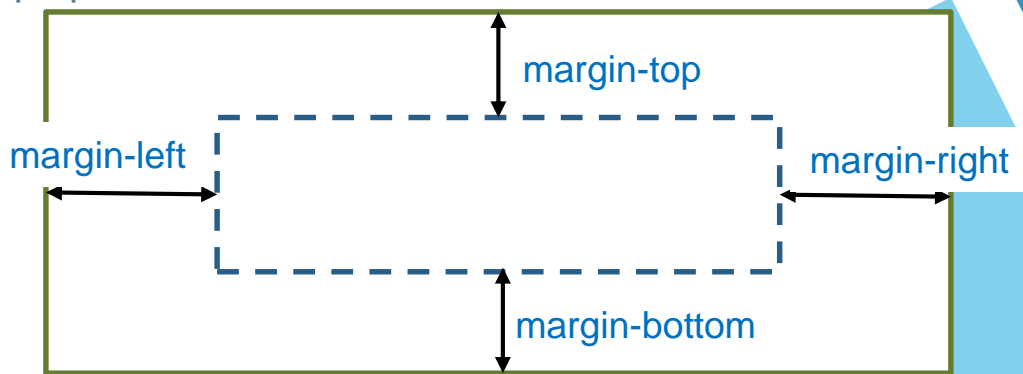


hidden/none



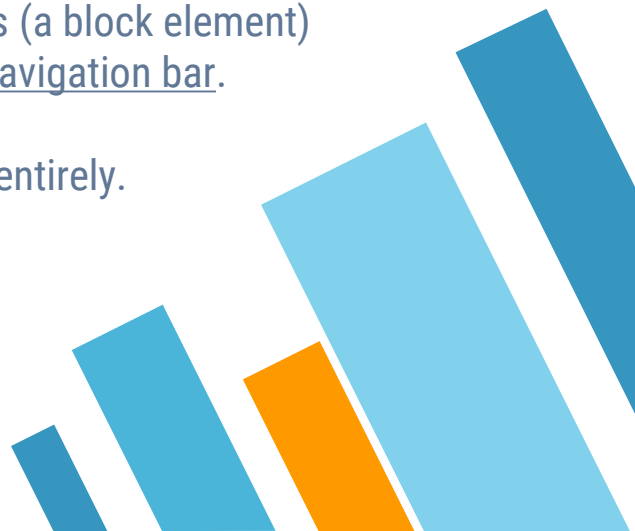
MARGIN

- » Specifies the amount of space between the border and the outside edge of the element.
- » Margins keep elements from bumping into one another or the edge of the browser window.
- » There are FOUR types of margin properties:
 - ◇ margin-top
 - ◇ margin-bottom
 - ◇ margin-right
 - ◇ margin-left





DISPLAY

- » The **display** property defines the type of element box an element generates in the layout.
 - » In addition to inline and block display roles, you can also make elements display as list items or the various parts of a table.
 - ♦ For example, it is a common practice to make **li** elements (a block element) display as inline elements to turn a list into a horizontal navigation bar.
 - ♦ `li { display: inline; }`
 - ♦ `display: none;` removes the content from the normal flow entirely.
- 

BASIC BOX PROPERTIES: SUMMARY

PROPERTY	DESCRIPTION
border	A shorthand property that combines border properties: Border: top right bottom left;
border-top, border-right, border-bottom, border-left	Combine border properties for each side of the element.
border-color	Shorthand property for specifying the color for each side of the element. border-color: top right bottom left;
border-top-color, border-right-color, border-bottom-color, border-left-color	Specify the border color for each side of the element.

BASIC BOX PROPERTIES: SUMMARY

PROPERTY	DESCRIPTION
border	A shorthand property that combines border properties: Border: top right bottom left;
border-top, border-right, border-bottom, border-left	Combine border properties for each side of the element.
border-color	Shorthand property for specifying the color for each side of the element. border-color: top right bottom left;
border-top-color, border-right-color, border-bottom-color, border-left-color	Specify the border color for each side of the element.

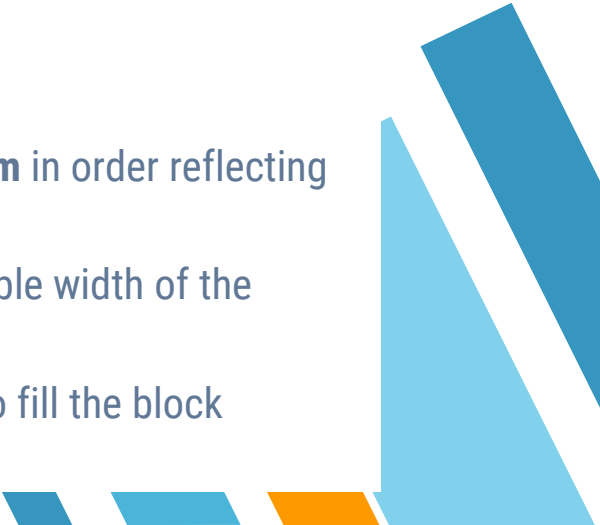


FLOATING AND POSITIONING

- » The **floating** property allows an element move to the left or right, and allows the following text to wrap around it.
 - » The **positioning** property is a way to specify the location of an element anywhere on the page with pixel precision.
 - » Take note:

In CSS layout model, text elements are laid out from **top to bottom** in order reflecting the which they appear in the source code.

Block elements stack up on top of one another and fill the available width of the browser window / containing element.

Inline elements and text characters line up next to one another to fill the block elements.
- 

FLOATS

- » This property moves an element as far as possible to the left or right, allowing the following content to wrap around it.
- » They are one of the primary tools used to create multicolumn layouts, navigation toolbars from lists, and table like alignment without tables.
- » Values: *left, right, none, inherit.*
- » Example: `<p>They went down, down..</p>`

Inline image in the normal flow

space next to the image is held clear



They went down, down, down, till at last they came to a passage with a door at one end, which was only fastened with a latch. The eldest Princess opened it, and they found themselves immediately in a lovely little wood, where the leaves were spangled with drops of silver which shone in the brilliant light of the moon. They next crossed another wood where the leaves were sprinkled with gold, and after that another still, where the leaves glittered with diamonds.

FLOATS

» `img { float: right;}`

Inline image floated to the right.

They went down, down, down, till at last they came to a passage with a door at one end, which was only fastened with a latch. The eldest Princess opened it, and they found themselves immediately in a lovely little wood, where the leaves were spangled with drops of silver which shone in the brilliant light of the moon. They next crossed another wood where the leaves were sprinkled with gold, and after that another still, where the leaves glittered with diamonds.



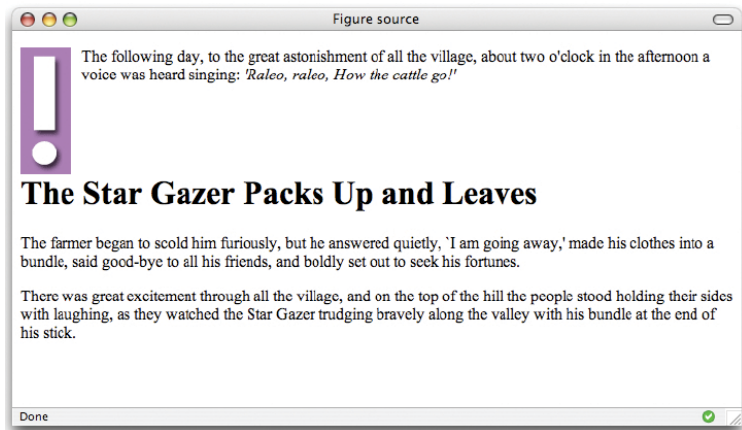
image moves over and text wraps around it

» As you can see, space are no longer wasted, as the text has wrapped around the image.

CLEAR

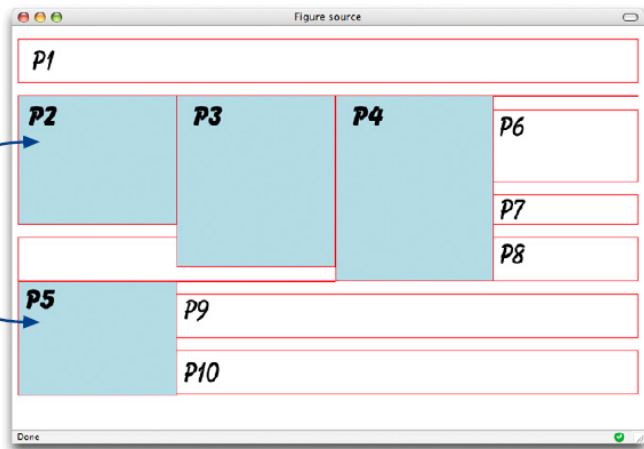
- » Applying **clear** property to an element prevents it from appearing next to a floated element and forces it to start against the next available “clear” space below the float.
- » Values: *left, right, both, none, inherit*.
- » Take note that you apply clear property **if** you want to start below the floated element. Not the floated element itself.

```
img {  
  float: left;  
  margin-right: 10px;  
}  
h2 {  
  clear: left;  
  margin-top: 2em;  
}
```



FLOATING MULTIPLE ELEMENTS

- » When multiple elements are floated, there is a complex system of rendering rules that ensures floated elements do not overlap.



```
<p>P1</p>
<p class="float">P2</p>
<p class="float">P3</p>
<p class="float">P4</p>
<p class="float">P5</p>
<p>P6</p>
<p>P7</p>
<p>P8</p>
<p>P9</p>
<p>P10</p>
```

```
p.float {
  float: left;
  width: 200px;
  margin: 0px;
  background: #CCC;
}
p {
  border: 1px solid red;
}
```

- » Example above shows if sequential paragraphs are floated to the same side. The first three floats start stacking up from *left edge* but when there is no room for the fourth one, it moves down and to the left.

DO TOGETHER!

- » Let's create a new file **navigation.html**.
- » Add this:

```
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">Products</a></li>
  <li><a href="#">Services</a></li>
  <li><a href="#">Shopping Cart</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

- » In the CSS file:
 1. Turn off bullets, and set padding and margins to 0.
 2. Float each list item to the left so that they line up.
 3. Make anchor elements in the list items (a) display as block elements, so you can set dimensions, padding, margins and other styles.

EXERCISE!

» Download and open index.html from Chap7 folder.

1. Make the navigation lists into a proper navigation bar by clearing the margin and padding.



2. Change the appearance of the links in the nav bar by:
 - i. make **a** elements into block instead of inline.
 - ii. add on padding of 0.5 em and margins 0.25em
 - iii. border color change to lavender #ba89a8.
 - iv. Once hovered and focused, border changed color to white.



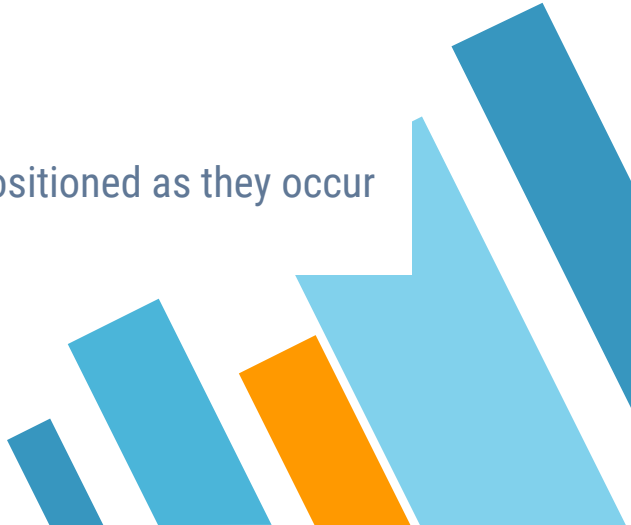
POSITIONING

- » The **position** property indicates that element is to be positioned and specifies which positioning method to use.
- » Value: *static, relative, absolute, fixed, inherit*

A. static

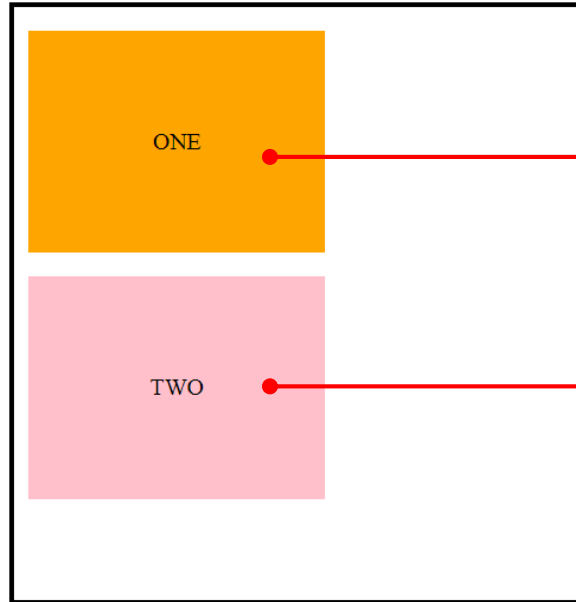
The default.

This is the normal positioning scheme in which elements are positioned as they occur in the normal document flow.



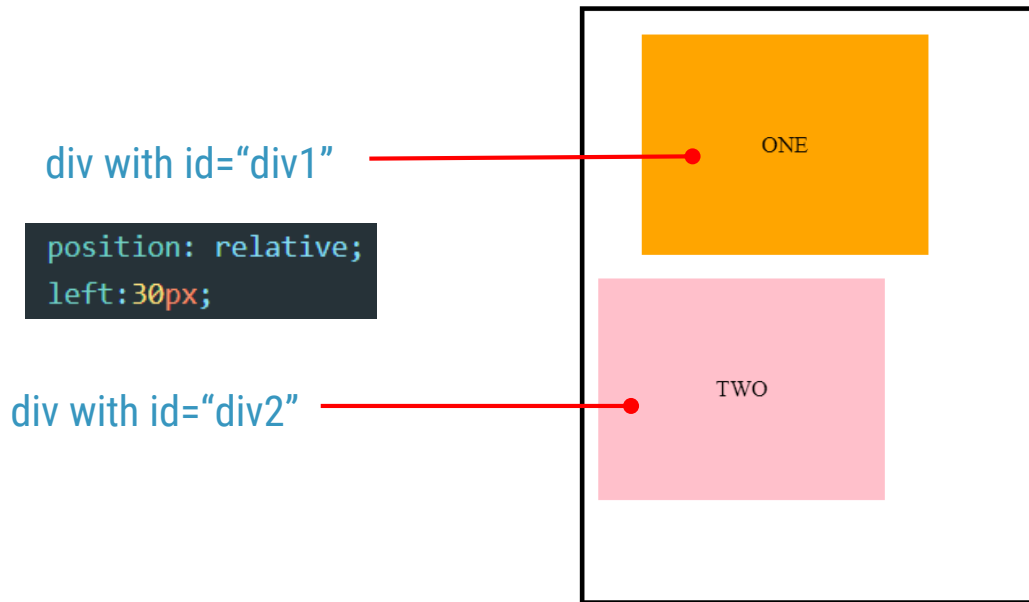
A. STATIC POSITIONING

```
position: static;
```



B. RELATIVE POSITIONING

- » Moves the box relative to its original position in the flow.
- » The space the element would have occupies in the normal flow is preserved as an empty space.



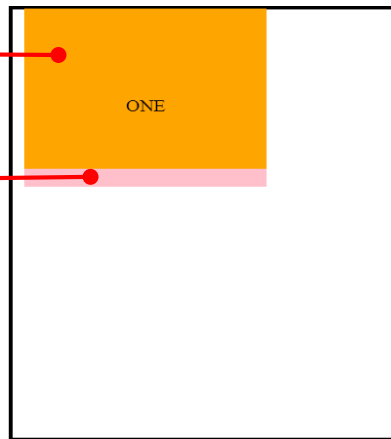
C. ABSOLUTE POSITIONING

- » Elements with this property are removed from the document flow entirely and positioned with respect to the browser window or a containing element.
- » The space of the element has occupied is closed up.
- » No influence at all on the layout of the surrounding elements.

```
position: absolute;  
top: 10px;
```

div with id="div1"

div with id="div2"



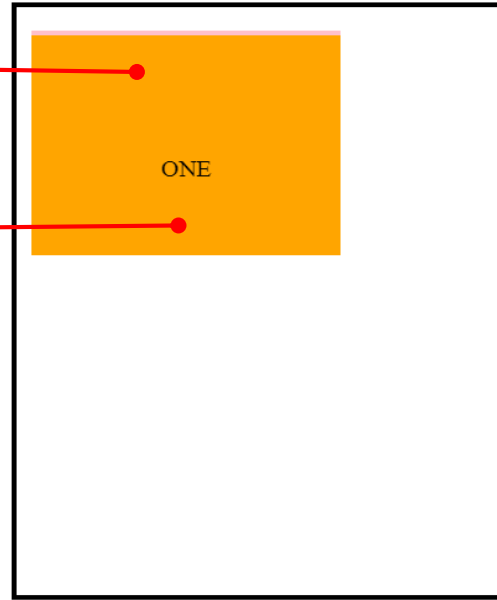
D. FIXED POSITIONING

- » The element with this property stayed in one position even when the document scrolls.
- » Fixed elements are removed from the document flow and positioned relative to the browser window rather than another element in the document.

div with id="div1"

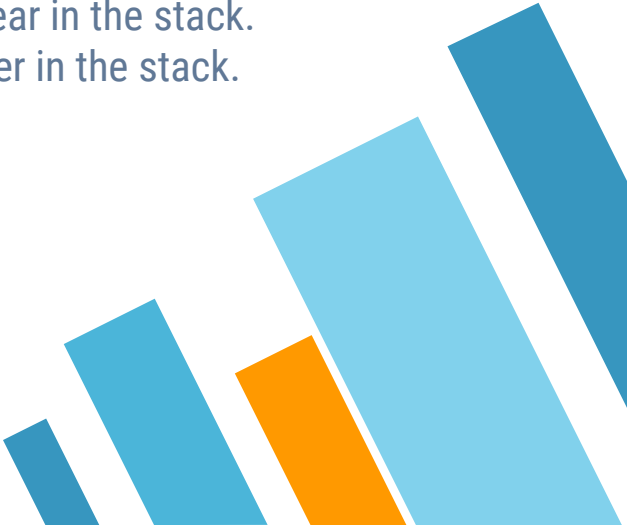
div with id="div2"

```
position: fixed;  
top: 30px;
```





STACKING ORDER

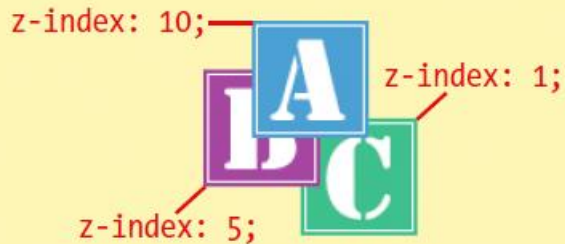
- » By default, elements stack up in the order in which they appear in the document, but you can change the order with **z-index** property.
 - » Values: *number, auto, inherit*.
 - ◇ The number can either be positive or negative.
 - ◇ The higher the number, the higher the element would appear in the stack.
 - ◇ Lower number and negative values move the element lower in the stack.
- 

STACKING ORDER

```
<body>
  <p id="A"></p>
  <p id="B"></p>
  <p id="C"></p>
</body>
```



By default, elements later in the document stack on top of preceding elements.



You can change the stacking order with the z-index property. Higher values stack on top of lower values.

```
<style>
  #A{
    z-index:10;
    position: absolute;
    top: 200px;
    left:200px;
  }
  #B{
    z-index:5;
    position: absolute;
    top: 225px;
    left:175px;
  }
  #C{
    z-index:1;
    position: absolute;
    top: 250px;
    left:225px;
  }
}
```