





What is the Box Model?

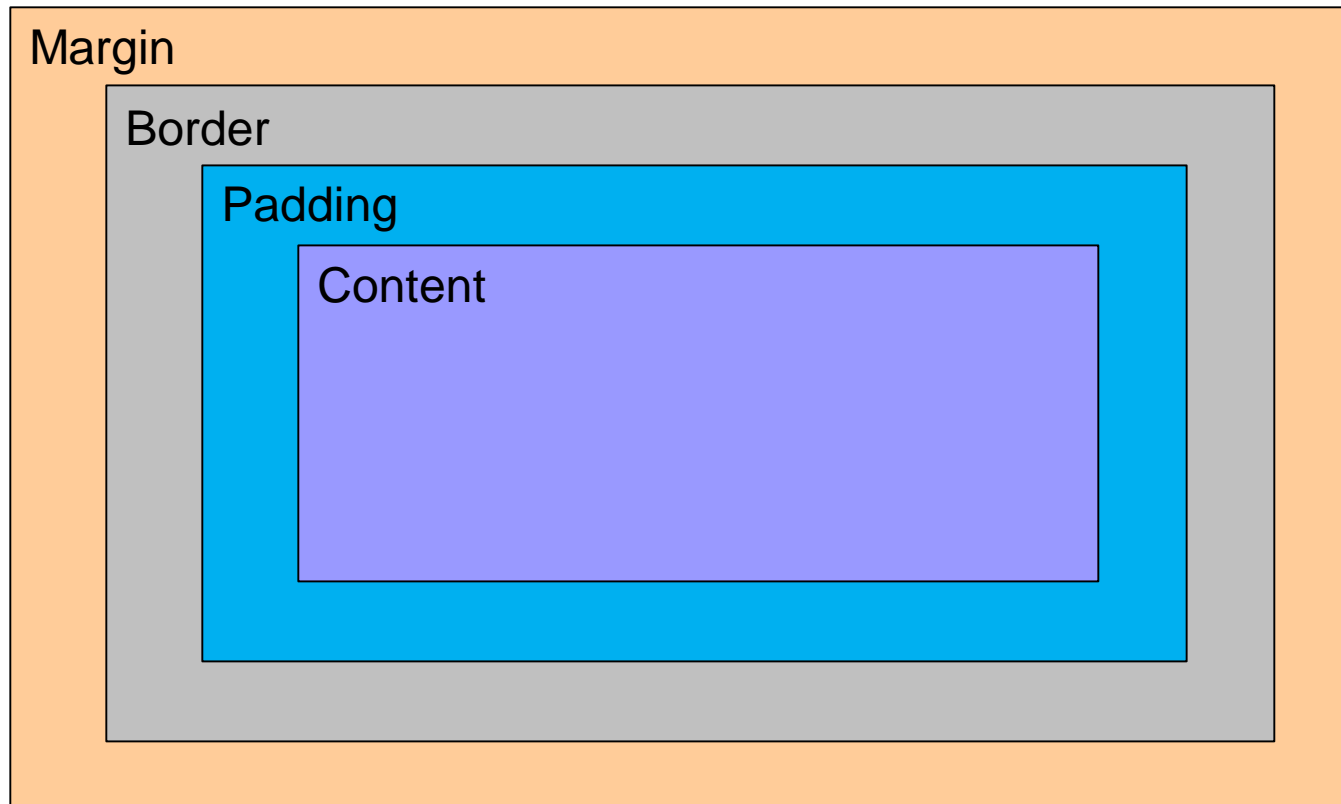
The box model is a tool we use to understand how our content will be displayed on a web page.

- Each XHTML element appearing on our page takes up a "box" or "container" of space.
- Each box size is affected not only by content but also by padding, borders, and margins.
- By knowing how to calculate the dimensions of each box, we can accurately predict how elements will lay out on the screen.
- As we build a new page, we can arrange these boxes on the screen, creating a balanced layout with white space around the content.

The importance of the box model concept cannot be overemphasized. It would be difficult and frustrating to create a website without understanding this concept.

Box Components

Each element on the page has the following components:



The easiest way to understand these components is to use one of the most versatile tools available to us as web designers: the `<div>` element.



Explanation of the different parts

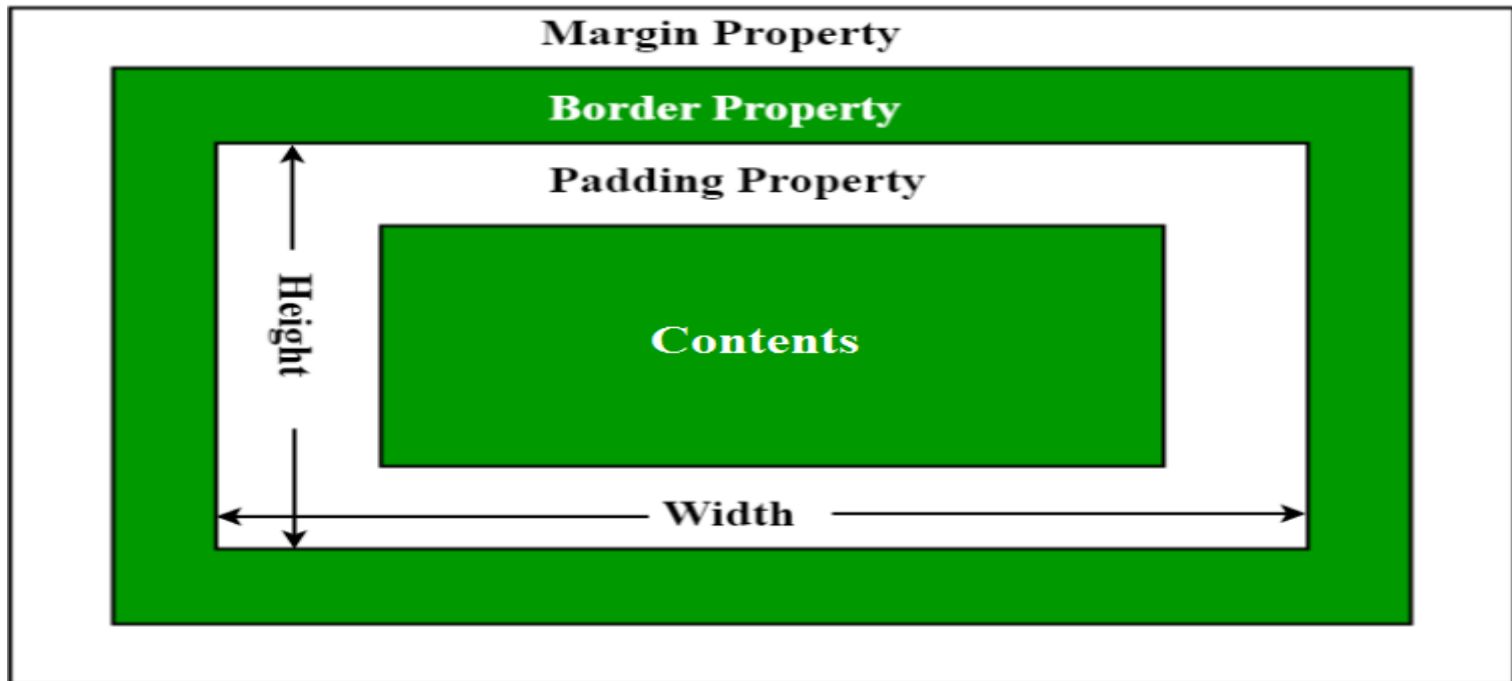
Margin - Clears an area around the border. The margin does not have a background color, it is completely transparent.

Border - A border that goes around the padding and content. The border is affected by the background color of the box.

Padding - Clears an area around the content. The padding is affected by the background color of the box.


Content - The content of the box, where text and images appear. In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Box Components



Property	Defines...
<code>margin</code>	Distance between the current box and those around it.
<code>padding</code>	Distance between the content in the box and the inner edge of its border.
<code>border</code>	The size and style of the border of an individual box.

The defaults for each property are for them to hold no value – you have to set the ones you wish to use.



Introducing the <div> Element

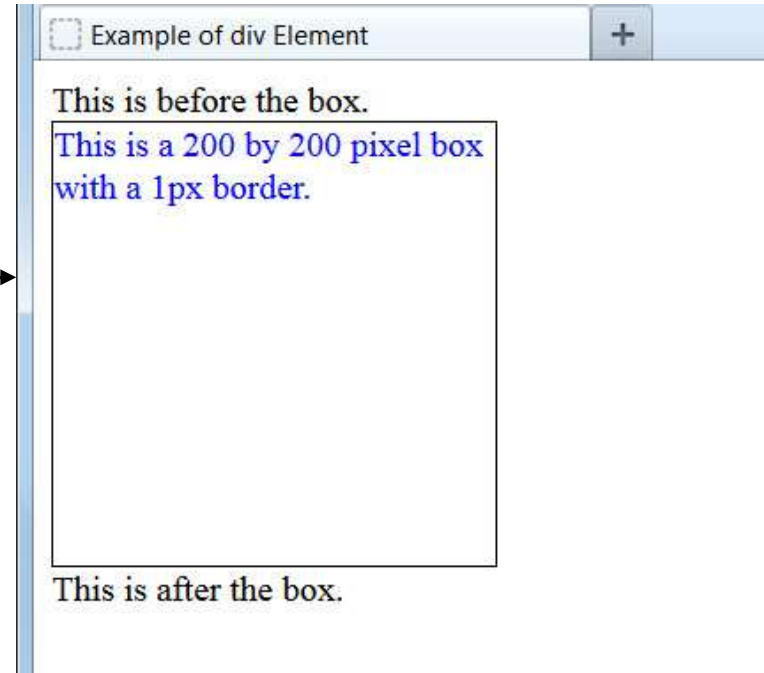
The <div> ("division") element groups other elements on the screen.

- By setting the **width** and **height** attributes via CSS, we can reserve a precise amount of space on our page for specific content.
- The actual content is nested and contained within the opening <div> and closing </div> tags.
- When we apply CSS styling directly to the <div> element, all the elements contained within that <div> will inherit that style.
- By using multiple <div> elements as building blocks, we can design an entire web page layout.

Example <div> Element

Let's create a box on the screen and add a border around it:

```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    border: 1px solid black;
    color: blue;
  }
</style>
...
This is before the box.
  <div class="box200">
This is a 200 by 200 pixel box with
  a 1px border.
  </div>
This is after the box.
...
```



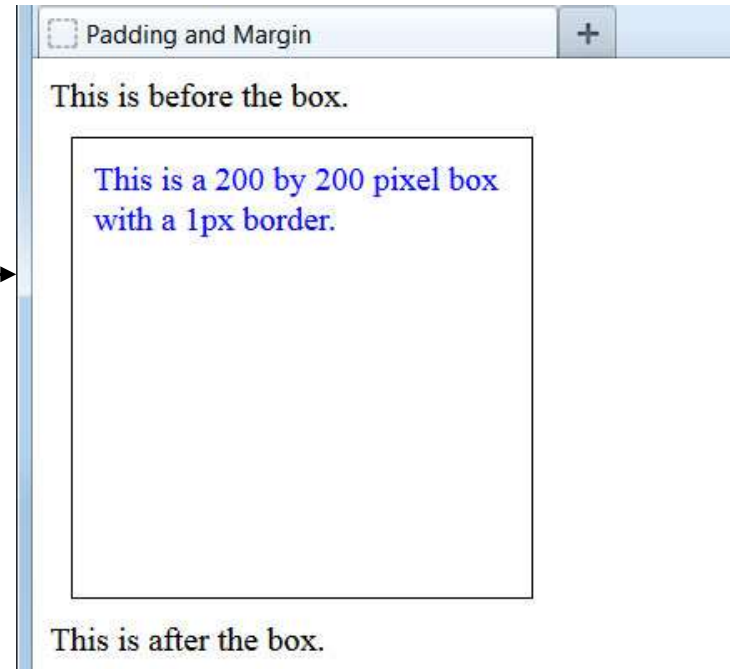
With the border set, we can see the exact space taken up on the page.

Notice that there is almost no space separating the text from the box border. Elements such as paragraphs, headers, and lists automatically insert padding and margins, but plain text does not do so.

Adding Padding and Margin

Let's create some space between elements by adding both padding and margin:

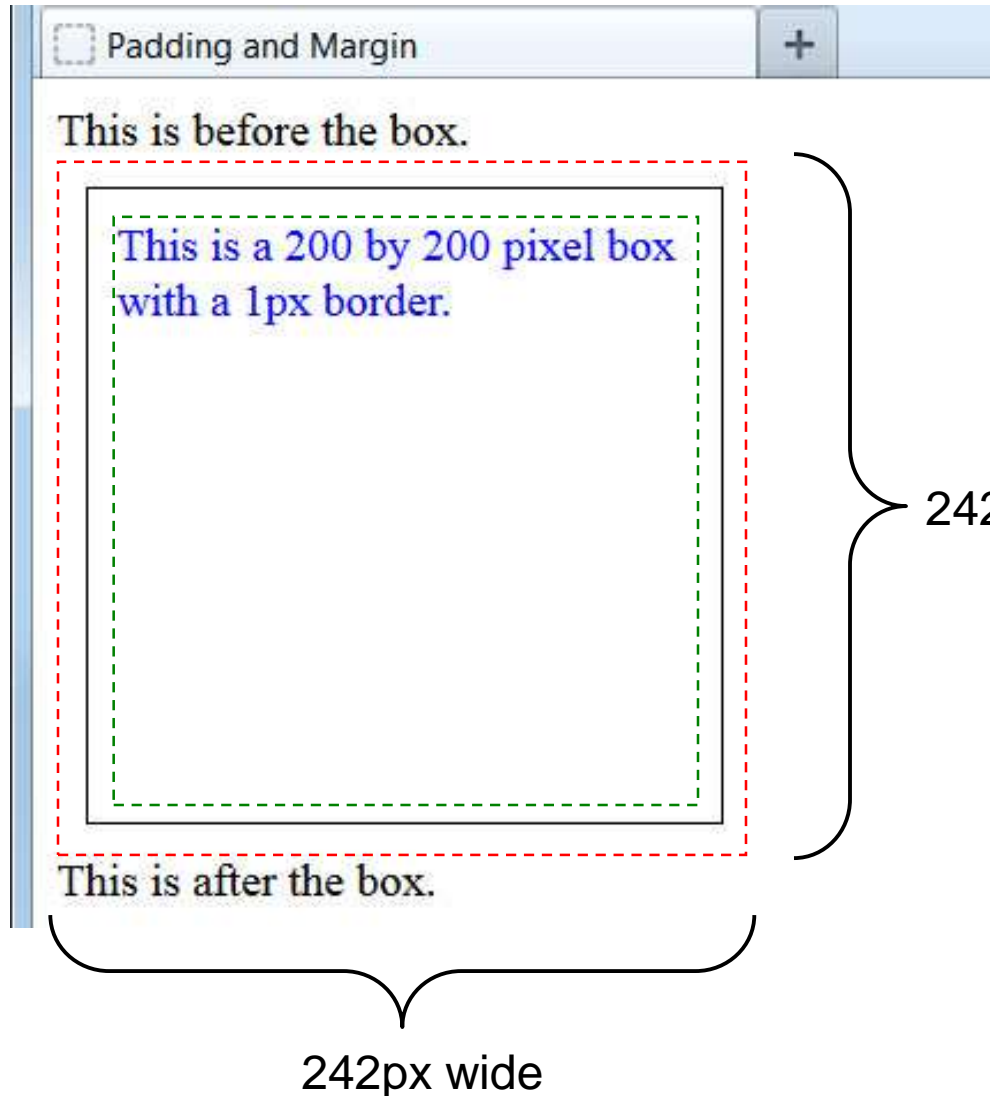
```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    border: 1px solid black;
    color: blue;
    padding: 10px;
    margin: 10px;
  }
</style>
...
```



The 10 pixel padding adds buffer space, on all four sides, between the content and border. The 10 pixel margin adds buffer space, on all four sides, between the border and surrounding elements.

Let's examine this a bit closer up to see exactly what is happening.

Padding and Margin Illustrated



The dotted red line shows the margin's outer boundary and the dotted green line shows the padding's inner boundary.

When we define the width and height of a `<div>` element, these dimensions apply only to the actual content, not to the padding, border, or margin.

Calculating Total Dimensions

When we are planning our page, we have to calculate exactly how much screen space a `<div>` or any other element will use. The formula is:

- Total element width = defined width + left padding + right padding + left border + right border + left margin + right margin.
- Total element height = defined height + top padding + bottom padding + top border + bottom border + top margin + bottom margin.

In our previous example:

- Total `<div>` width = 200px + 10px + 10px + 1px + 1px + 10px + 10px = 242px.
- Total `<div>` height = 200px + 10px + 10px + 1px + 1px + 10px + 10px = 242px.

Padding, borders, and margins do not have to be the same on all four sides, as they are in this example. Let's see how to set these individually.

Setting Individual Margins

We can set the specific margin sizes by using these four properties:

margin-top:

margin-right:

margin-bottom:

margin-left:

In practice, few web designers use these properties, preferring a **shorthand** method that condenses the declaration to a single line:

margin: 10px; →

Sets all four margins to be 10px.

margin: 10px 5px; →

Sets the top and bottom margins as 10px and the left and right margins as 5px.

margin: 20px 10px 5px 15px; →

Sets the top margin as 20px, the right margin as 10px, the bottom margin as 5px, and the left margin as 15px.

Setting Individual Padding

We can set specific padding sizes by using these four properties:

padding-top:
padding-right:
padding-bottom:
padding-left:

Again, these are rarely used. Instead, the same shorthand method is employed:

padding: 5px; →

Sets padding as 5px on all four sides.

padding: 25px 5px; →

Sets top and bottom padding as 25px and left and right padding as 5px.

padding: 50px 10px 15px 5px; →

Sets top padding as 50px, right padding as 10px, bottom padding as 15px, and the left padding as 5px.



Interrupt the Flow

- Absolute
- Relative
- Float

When you want to do fancier layout, you can **position** “boxes” or “containers.” By doing this, you interrupt the normal (top to bottom, left to right) flow. You can do this in three ways; **Float**, **Absolute**, and **Relative**.

```
<div>
```

```
<p> This is the  
normal...</p>
```

```
<p class="float">This text is  
floated right.</p>
```

```
</div>
```

```
.float {float:right;}
```

This is the normal flow of a document; from top to bottom, left to right. When the floated text is added, it moves to the top right corner of the containing element, in this case the <div>. Normal text flows around the floated text.

This text is
floated right.

Absolute

```
<div>
<p> This is the normal...
<span class="abs"> This text
is absolutely
positioned.</span>...top to
bottom... </p>
</div>
```

```
.abs {position: absolute;
      top: 40px;
      left: 80px;}
```

```
#two {
  top: 20px;
  left: 20px;
  background: green;
  position: absolute;
}
```

This is the normal flow of a document; from top to bottom, left to right. When you add the absolutely positioned text, it moves to the coordinates you set based on the top left corner of the containing element, in this case the <div>. Normal text flows over the absolutely positioned text. There is no gap where the text is taken from. This text is absolutely positioned.



This text is relatively positioned.

Relative

```
<div>
```

```
<p> This is the normal...
```

```
<span class="rel"> This text is relatively positioned. </span>
```

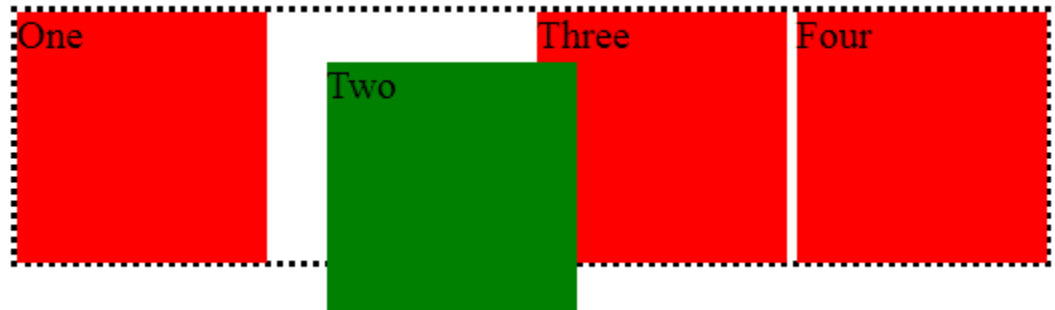
```
... from top to bottom...</p>
```

```
</div>
```

```
.rel {position: relative;  
      top: -50px;  
      left: -150px}
```

This is the normal flow of a document; from top to bottom, left to right. When you add the relatively positioned text, it moves to the coordinates you set based on the **top left corner** of the containing element, in this case the `<div>`. Normal text flows **as normal**, but a gap is left where the relative text used to be, and the text **overlaps** with the newly positioned relative text if they are in the same area.

```
#two {  
  top: 20px;  
  left: 20px;  
  background: green;  
  position: relative;  
}
```



Setting Individual Borders

Customizing borders is more involved. Since they are visible on the page, we have to specify style, thickness, and color. Accordingly, there are more properties available:

<code>border-top-width:</code>	<code>border-top-style:</code>	<code>border-top-color:</code>
<code>border-right-width:</code>	<code>border-right-style:</code>	<code>border-right-color:</code>
<code>border-bottom-width:</code>	<code>border-bottom-style:</code>	<code>border-bottom-color:</code>
<code>border-left-width:</code>	<code>border-left-style:</code>	<code>border-left-color:</code>

To make things easier, we can use the shorthand method as before:

```
border-width: 10px;  
border-style: solid dashed;  
border-color: blue red orange gray;
```

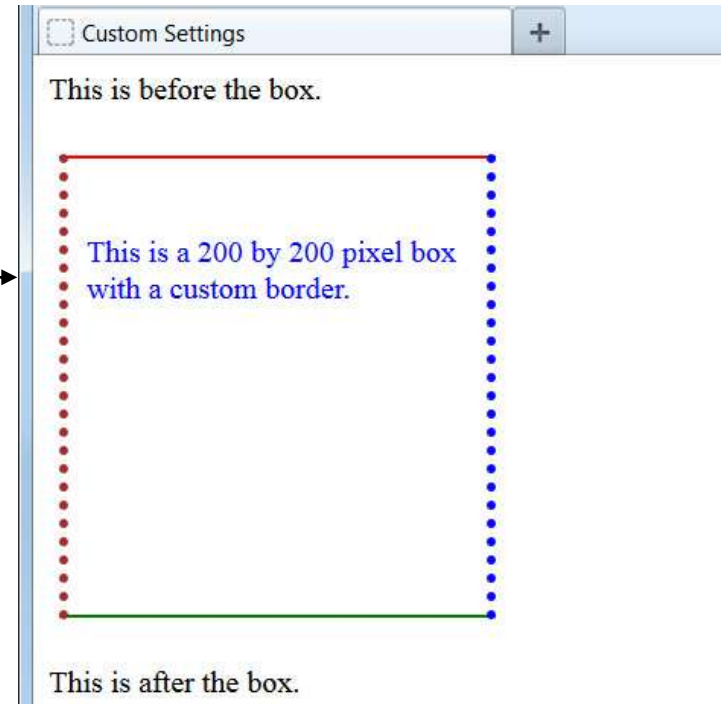
If the three border properties will be identical on all four sides, we can use a single-line border shorthand, as we did in an earlier lesson:

```
border: 5px solid blue;
```

Example of Customized Settings

Here we have set custom padding, borders, and margins:

```
<style type="text/css">
  .box200 {
    width: 200px;
    height: 200px;
    color: blue;
    padding: 40px 10px 0px 10px;
    margin: 25px 5px;
    border-width: 2px 5px;
    border-style: solid dotted;
    border-color: red blue green maroon;
  }
</style>
```



A helpful way to remember the order of shorthand settings is that it starts at noon and goes clockwise around: top, right, bottom, left.



CSS Font-Size: em vs. px vs. pt vs. percent

It's easy to understand the difference between font-size units when you see them in action. Generally, **1em = 12pt = 16px = 100%**.

“Ems” (em): The “em” is a scalable unit that is used in web document media

Pixels (px): Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen). One pixel is equal to one dot on the computer screen (the smallest division of your screen's resolution)

Points (pt): Points are traditionally used in print media (anything that is to be printed on paper, etc.). One point is equal to 1/72 of an inch. Points are much like pixels, in that they are fixed-size units and cannot scale in size.

Percent (%): The percent unit is much like the “em” unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%). While using the percent unit, your text remains fully scalable for mobile devices and for accessibility.

Em vs. Percent

We've decided that point and pixel units are not necessarily best suited for web documents, which leaves us with the em and percent units. In theory, both the em and the percent units are identical, but in application, they actually have a few minor differences that are important to consider.

boxmodel_ex.html

```
<html>
<head>
  <title>Box Model</title>
  <link href="box_ex2.css" rel="stylesheet"
type="text/css">
</head>
<body>
  <div id="content">
    <h1>CSS Box Model</h1>

    <p>The CSS margin properties dictate the
spacing between elements on a web page.They
are typically used to offset content to create an
indented or centred effect.</p>
  </div>
</body>
</html>
```

box_ex2.css

```
#content {
  width:220px;
  border:5px solid gray;
  margin:20px;
  padding:20px 15%; }
```



CSS Box Model

The CSS margin properties dictate the spacing between elements on a web page. They are typically used to offset content to create an indented or centred effect.

CSS Box Model

The CSS margin properties dictate the spacing between elements on a web page. They are typically used to offset content to create an indented or centred effect.

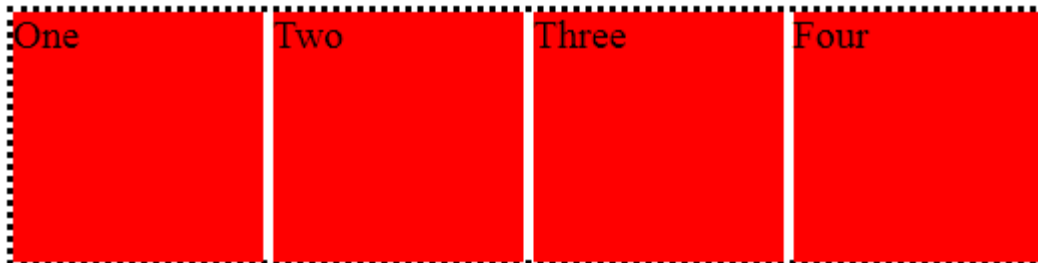
css_abs_rel.html

```
<html>
<head>
<title>Positioning of BOX</title>
<link href="css_abs_rel.css" rel="stylesheet"
      type="text/css">
</head>
<body>
<div class="parent">
  <div class="box" id="one">One</div>
  <div class="box" id="two">Two</div>
  <div class="box" id="three">Three</div>
  <div class="box" id="four">Four</div>
</div>
</body>
</html>
```

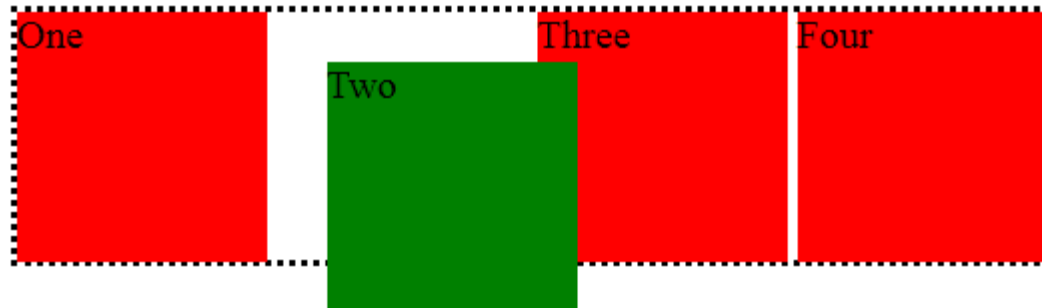
css_abs_rel.css

```
.parent {
  border: 2px black dotted;
  display: inline-block;
}
.box {
  display: inline-block;
  background: red;
  width: 100px;
  height: 100px;
}
#two {
  top: 20px;
  left: 20px;
  background: green;
  position: relative;
}

/*#two {
  top: 20px;
  left: 20px;
  background: green;
  position: absolute;
}*/
```



position: absolute;



position: relative;

CSS CHEAT SHEET

Shorthand*

background
border
border-bottom
border-left
border-right
border-top
font
list-style
margin
padding

Comments

/* Comment */

Pseudo Selectors

:hover
:active
:focus
:link
:visited
:first-line
:first-letter

Media Types

all
braille
embossed
handheld
print
projection
screen
speech
tty
tv

Units

Length %
em
pt

SYNTAX

Syntax

selector {property: value;}

External Style Sheet

<link rel="stylesheet" type="text/css" href="style.css" />

Internal Style

<style type="text/css">

selector {property: value;}

</style>

Inline Style

<tag style="property: value">

GENERAL

Class	String preceded by a period
ID	String preceded by a hash mark
div	Formats structure or block of text
span	Inline formatting
color	Foreground color
cursor	Appearance of the cursor
display	block; inline; list-item; none
overflow	How content overflowing its box is handled visible, hidden, scroll, auto
visibility	visible, hidden

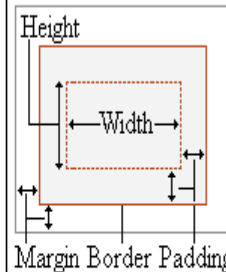
FONT

font-style	Italic, normal
font-variant	normal, small-caps
font-weight	bold, normal, lighter, bolder, integer (100-900)
font-size	Size of the font
font-family	Specific font(s) to be used

TEXT

letter-spacing	Space between letters
line-height	Vertical distance between baselines

BOX MODEL



height; width; margin-top; margin-right; margin-bottom; margin-left; padding-top; padding-right; padding-bottom; padding-left;

BORDER

border-width	Width of the border
border-style	dashed; dotted; double; groove; inset; outset; ridge; solid; none
border-color	Color of the border

POSITION

clear	Any floating elements around the element? both, left, right, none
float	Floats to a specified side left, right, none
left	The left position of an element auto, length values (pt, in, cm, px)
top	The top position of an element auto, length values (pt, in, cm, px)
position	static, relative, absolute
z-index	Element above or below overlapping elements? auto, integer (higher numbers on top)

BACKGROUND

background-color	Background color
background-image	Background image
background-repeat	

Exercise

#container

#banner

#nav

#content

#footer

Create a web page with the following layout by using css box model.



[Home](#)
[Contact](#)

CSS Box Model

The CSS margin properties dictate the spacing between elements on a web page. They are typically used to offset content to create an indented or centred effect.

Copyright info here