CSS Box Model

Courtesy:Nick Foxall

The Basis of CSS layout

The 3 core concepts to understand about CSS layout are:

1. *The CSS box model*
2. *Floating*
3. *Positioning*

Together, these 3 concepts control the way elements are arranged and displayed on a page.

The CSS Box Model

Every block element in CSS is effectively inside a box, and can have margins, padding and borders applied to it.

Box widths can be specified in absolute values (e.g. px) or in relative values, usually:

> ***em*** *- width values relative to the size of the font in ems*
>
> ***percentage*** *- width values relative the containing box's content region*

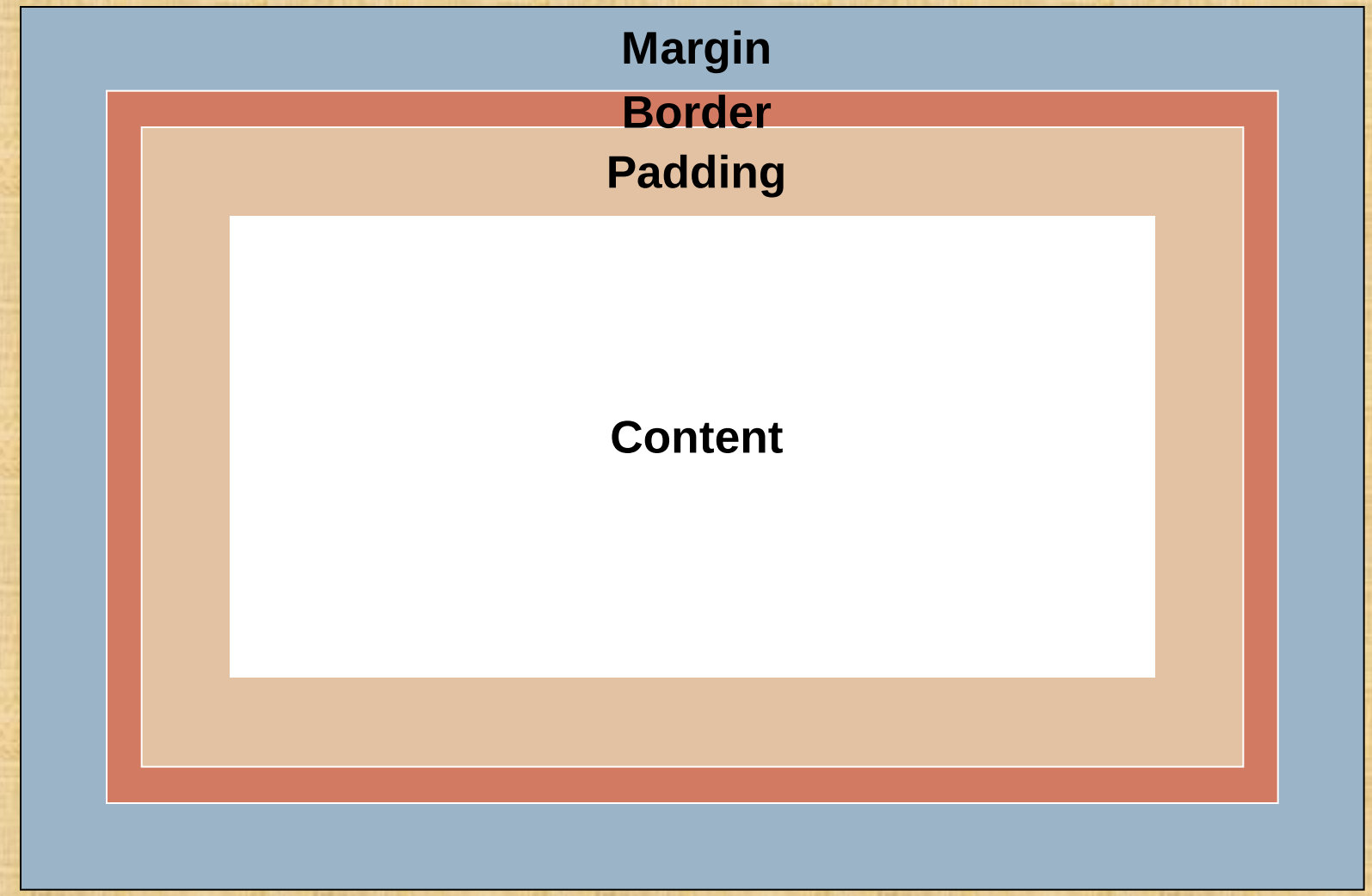
The root (or top-most) element's containing box is effectively the browser window.

The CSS Box Model

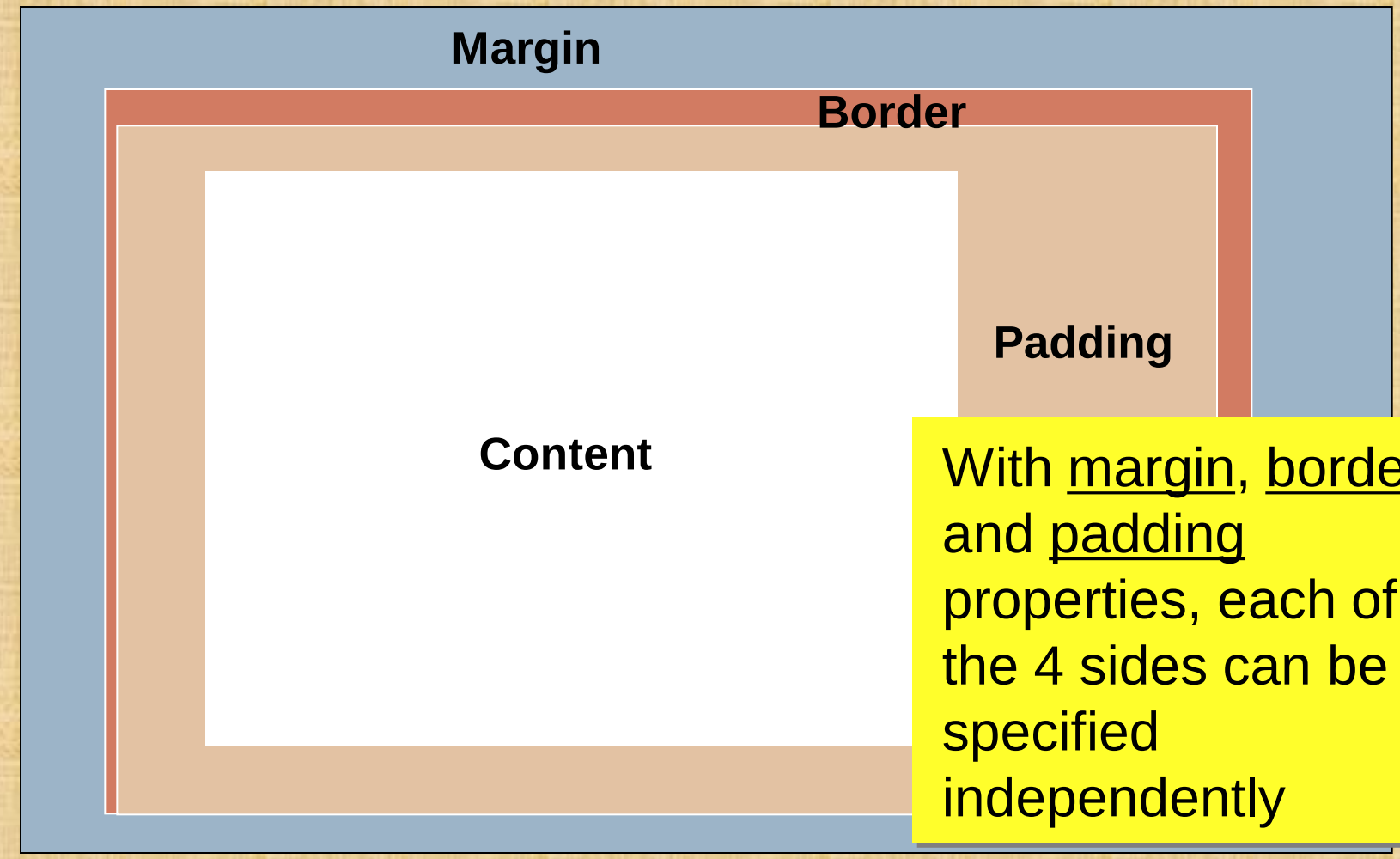
Every CSS box is divided into regions, consisting of:

1. *Content*
2. *Padding*
3. *Border*
4. *Margins*

The CSS Box Model



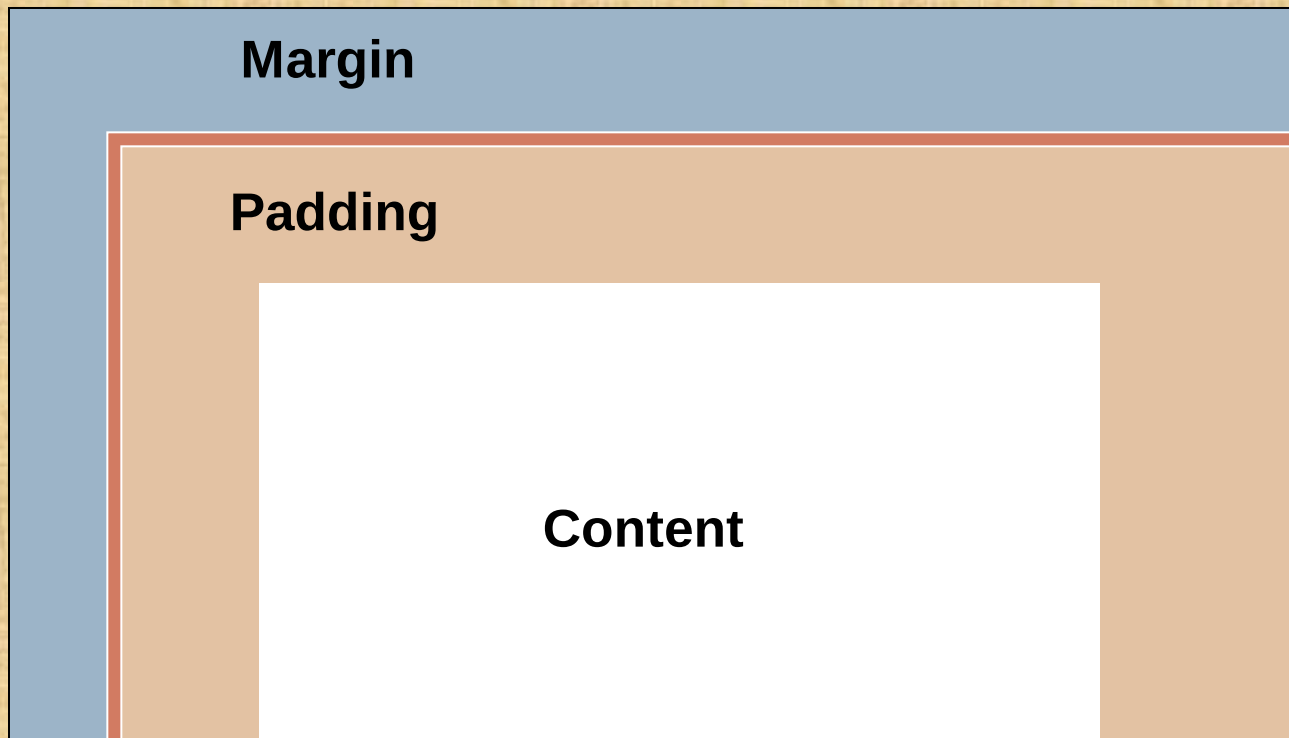
The CSS Box Model



With margin, border and padding properties, each of the 4 sides can be specified independently

Margins & Padding

Margins and Padding may seem similar at first glance. But each has its own effect on content, particularly on any backgrounds assigned to block and div elements.



Margins & Padding

Margins

Margins define the space around elements outside the border

Margin properties can have negative values in order to deliberately overlap content

Margin properties will affect the position of background elements (graphics and/or colours) in relation to the edges of the containing block element

Margin properties can be defined independently on top, right, bottom and left, or all-at-once using CSS shorthand

Margins & Padding

Padding

Padding defines the space around elements inside the border; i.e between the border and the content itself

Padding properties cannot have negative values

Padding properties do not affect the position of background elements (graphics and/or colours) in the containing block element; only the position of content.

Padding properties can be defined independently on top, right, bottom and left, or all-at-once using CSS shorthand

Margins and Padding: Margin Collapse

When 2 or more vertical margins meet, they will collapse to form a single margin

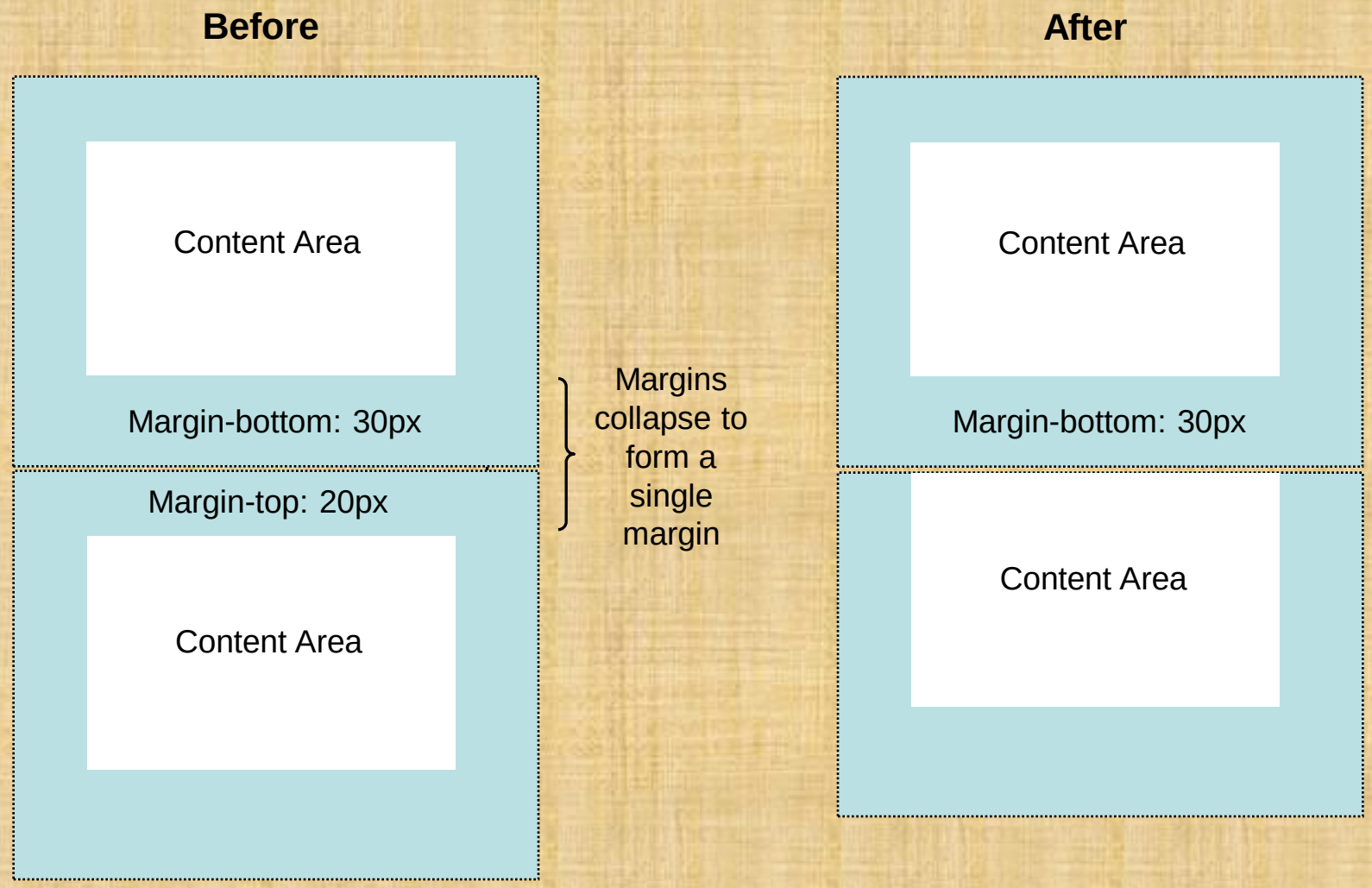
The height of this combined margin will be equal the height of the larger of the 2 margins

Margin collapse applies when:

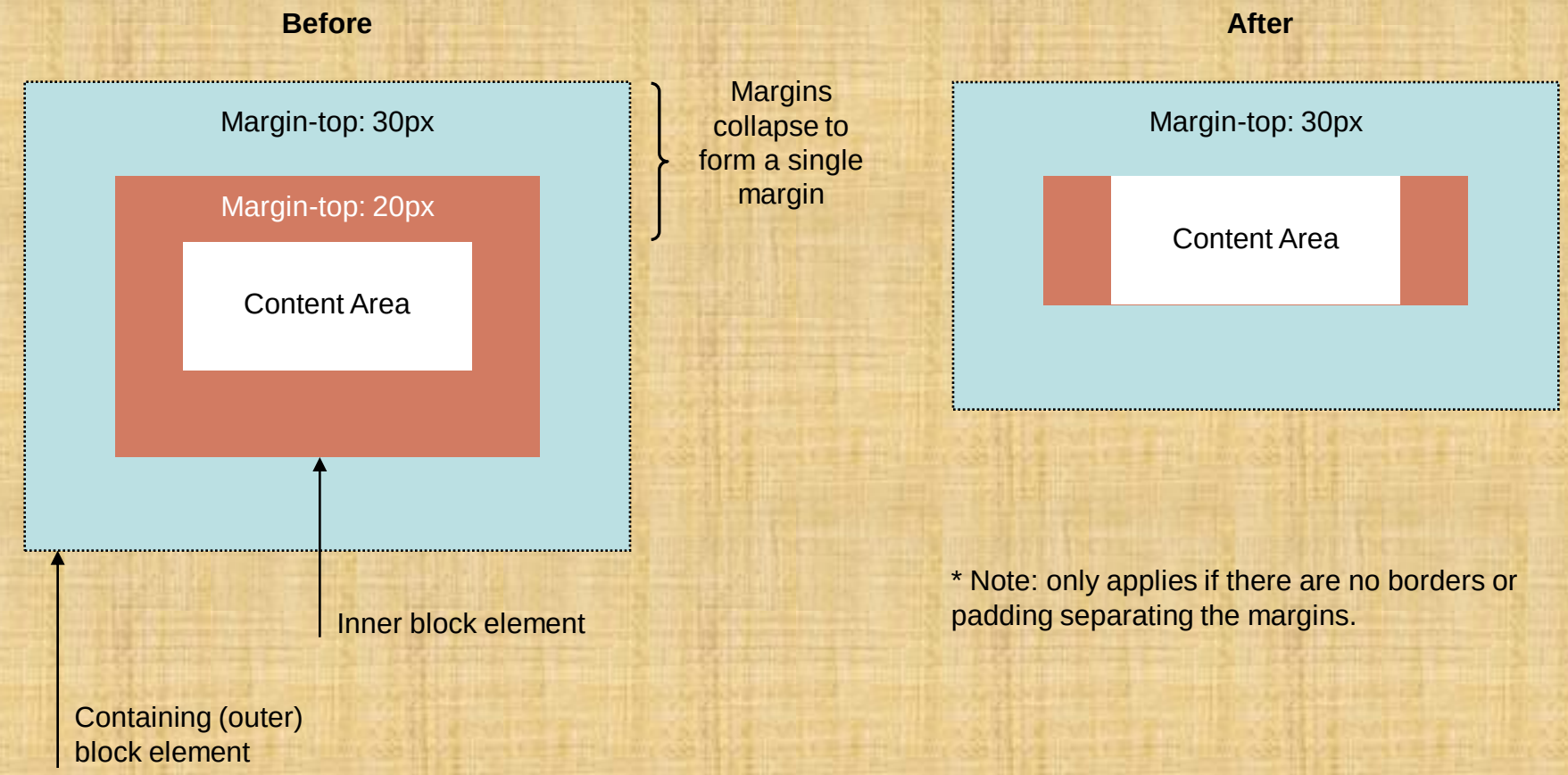
2 or more block elements are stacked one above the other,

or when one block element is contained within another block element

Margin Collapse: Stacked Elements



Margin Collapse: Contained Elements

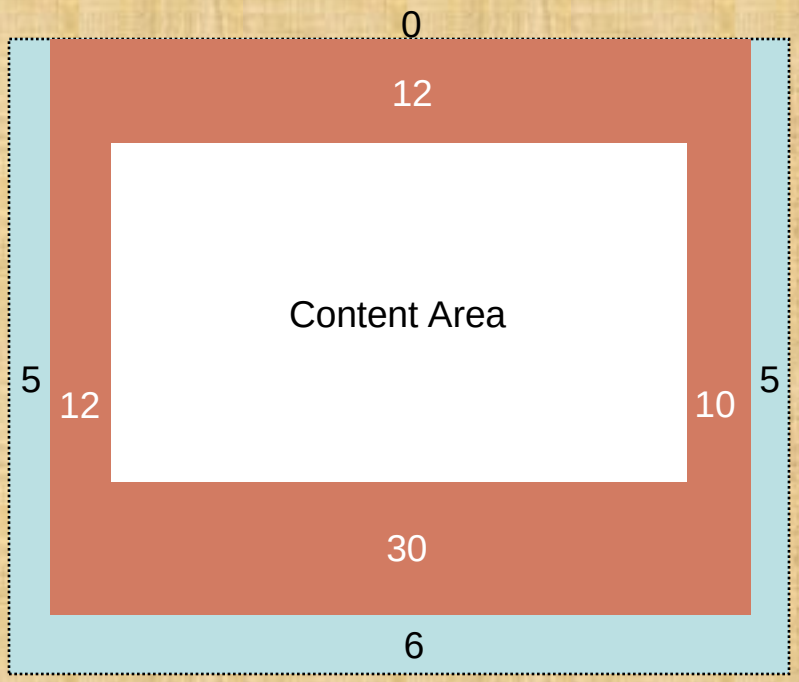


* Note: only applies if there are no borders or padding separating the margins.

CSS Shorthand: Margin & Padding

For margin and padding (and others), CSS provides a number of shorthand properties that can save on writing lines and lines of code. Instead of writing this:

```
#container {
        margin-top: 0;
        margin-right: 5px;
        margin-bottom: 6px;
        margin-left: 5px;
        padding-top: 20px;
        padding-right: 10px;
        padding-bottom: 30px;
        padding-left: 12px;
}
```

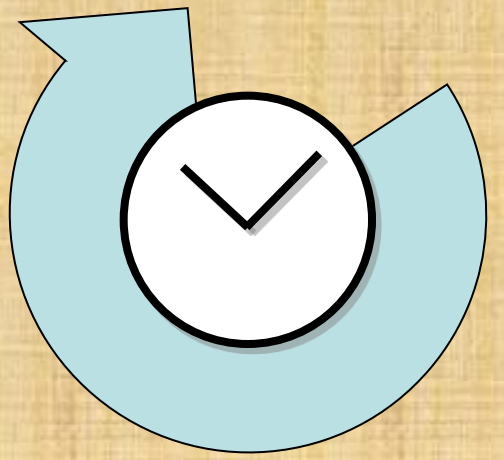


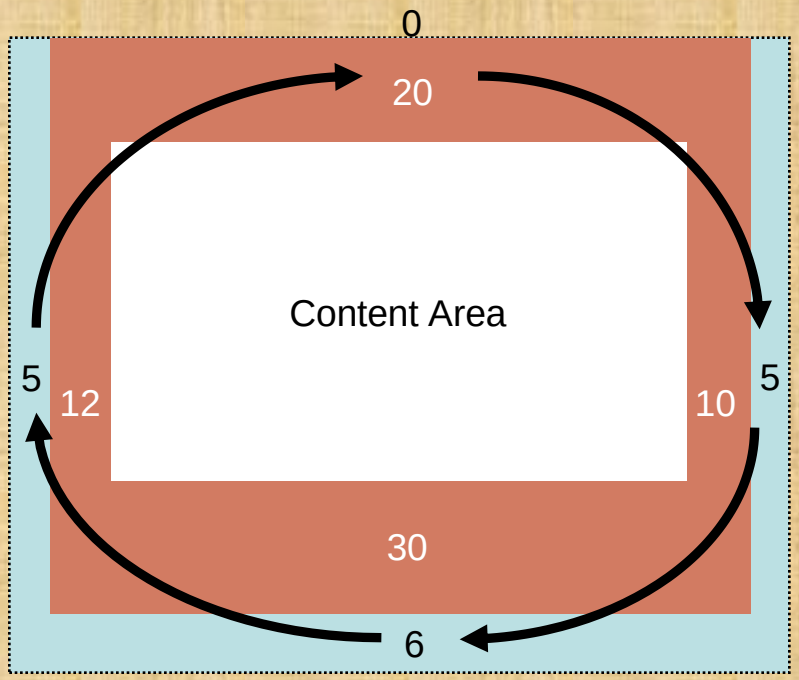
CSS Shorthand: Margin & Padding

…Its much easier to write this:

```
#container {
        padding: 20px 10px 30px 12px;
        margin: 0px 5px 6px 5px;
}
```

The sequence order is always clockwise, starting from the top



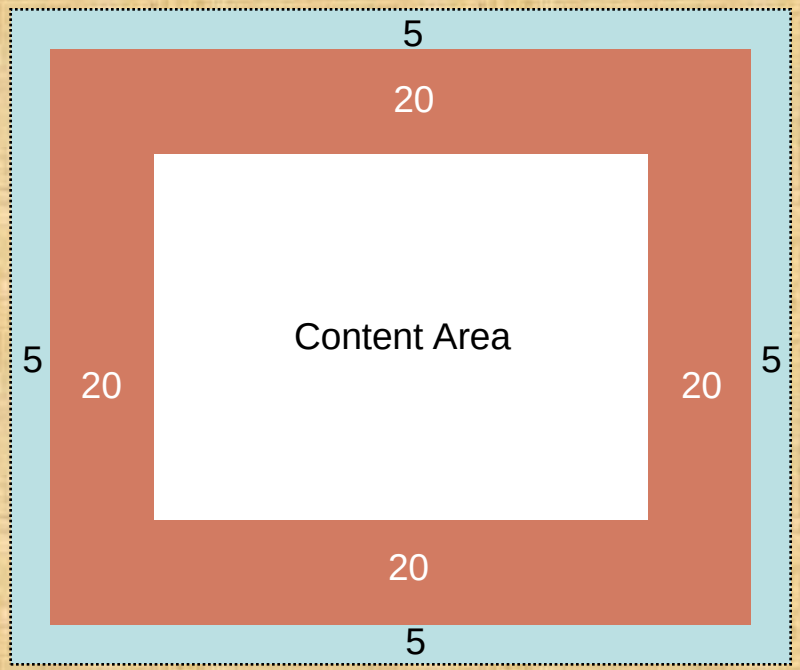


CSS Shorthand: Margin and Padding

You can also apply just one value, example:

```
#container {
        padding: 20px;
        margin: 5px;
}
```

Which will apply the value specified equally on all 4 sides



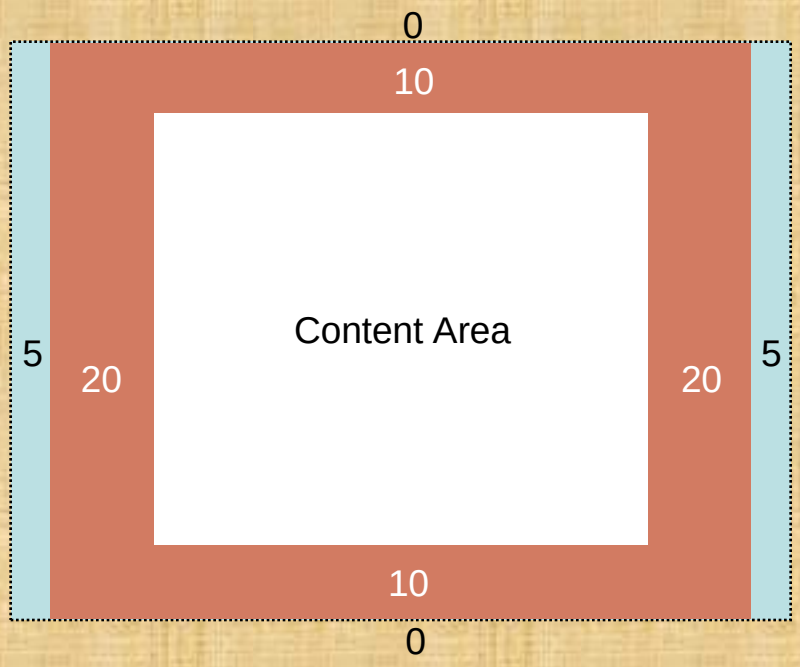
CSS Shorthand: Margin and Padding

And you can apply two values, example:

```
#container {
        padding: 10px 20px;
        margin: 0px 5px;
}
```

The first value is applied to the top and bottom

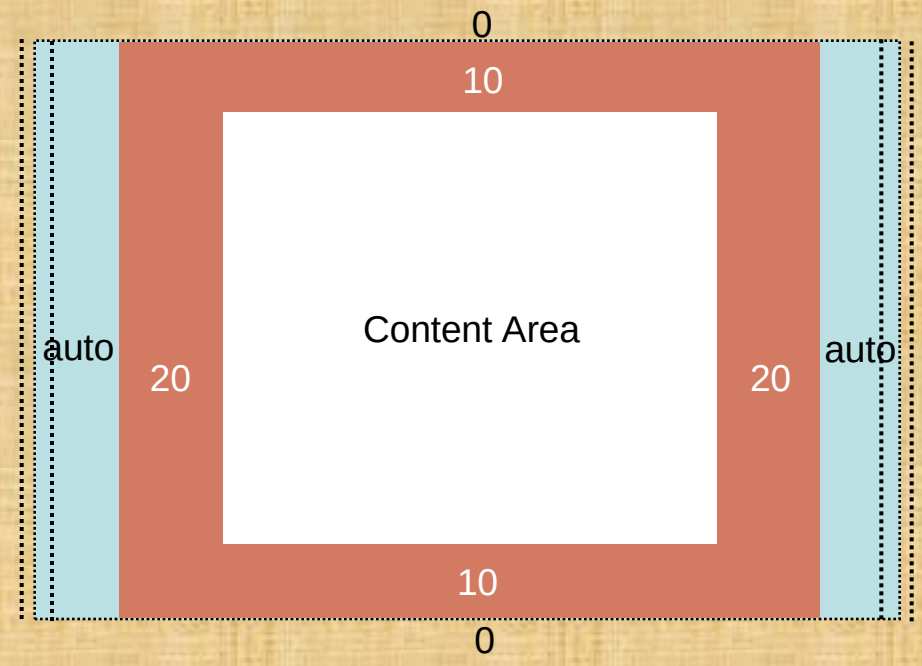
The second value is applied to the left and right



CSS Shorthand: Margin and Padding: auto

A useful value to remember is ‘auto’:

```
#container {
        padding: 10px 20px;
        margin: 0px auto;
}
```



Usually applied to the left & right areas of the margin property, auto is useful for centering a block container element in the browser window

Borders

Borders can be applied to any block element

Borders always come outside the padding area, but inside the margin area.

Border properties cannot have negative values

If a border is not specified, the default is no-border (i.e. no border appears and no space between any margin and padding is allocated for a border)

Border properties can be defined independently on top, right, bottom and left, or all-at-once using CSS shorthand

Borders

The core border properties are:

Width: *absolute (px, in, cm, or 'thin', 'medium', 'thick'), or relative (ems)*

Style: *dotted, dashed, solid, double, groove, ridge, inset, outset, hidden, etc*

Color: *'blue', 'red', #FF9900, etc*

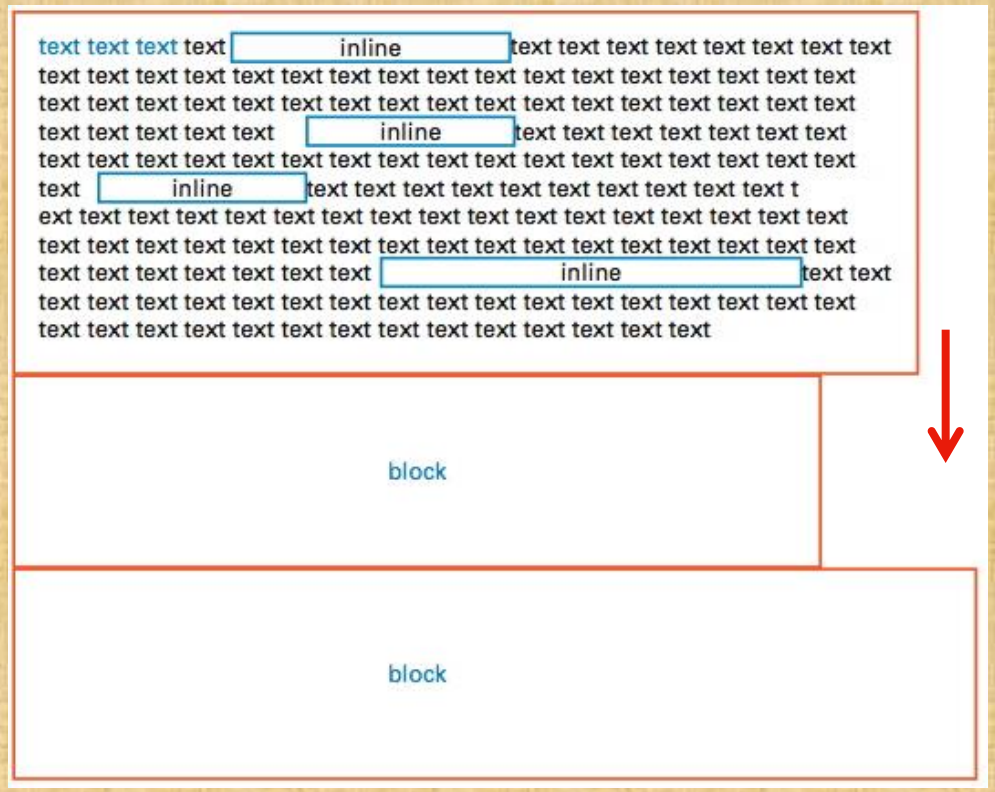
You can also create the effect of a border by using a graphic image in a CSS background property, instead of the border property

CSS Floats: “Normal Flow”

CSS boxes for block elements are stacked, one above the other, so that they’re read from top to bottom.

In CSS, this is said to be the “normal flow”.

(Note that CSS boxes for inline elements are placed next to each other, within boxes for block elements, and will normally wrap according to the width of the containing block.)

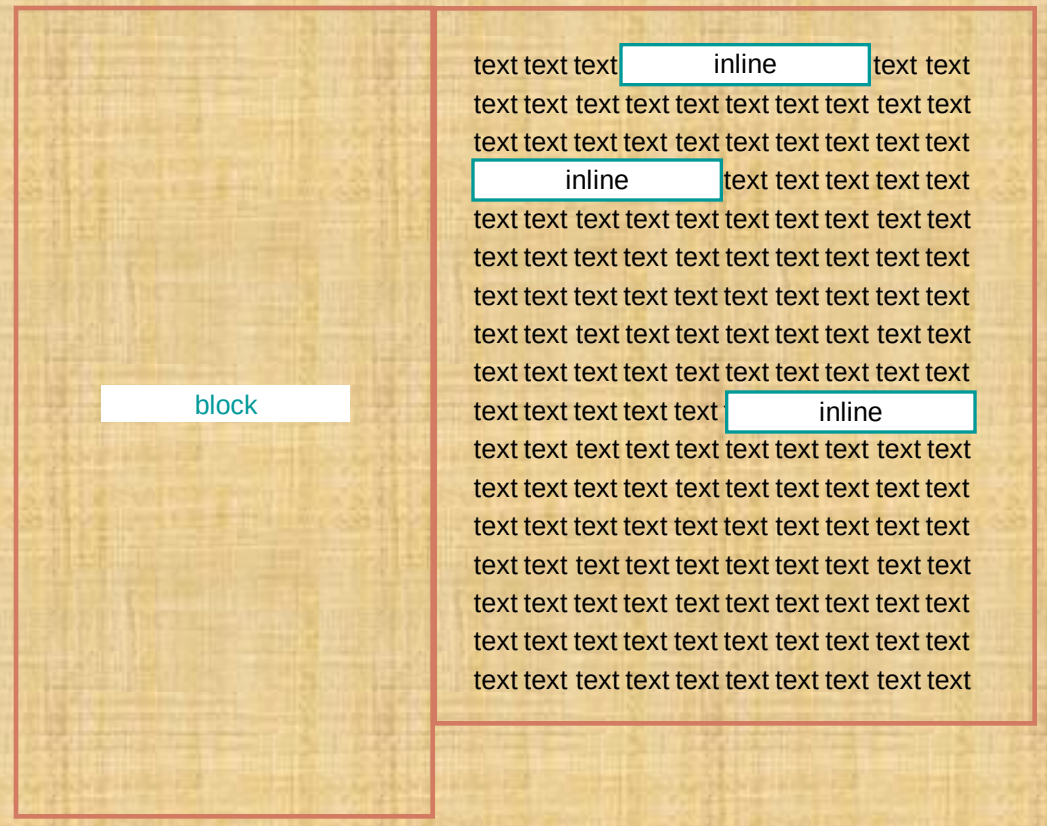


But…

Floats: Positioning CSS Boxes

…we can position block element boxes side-by-side in CSS using floats.

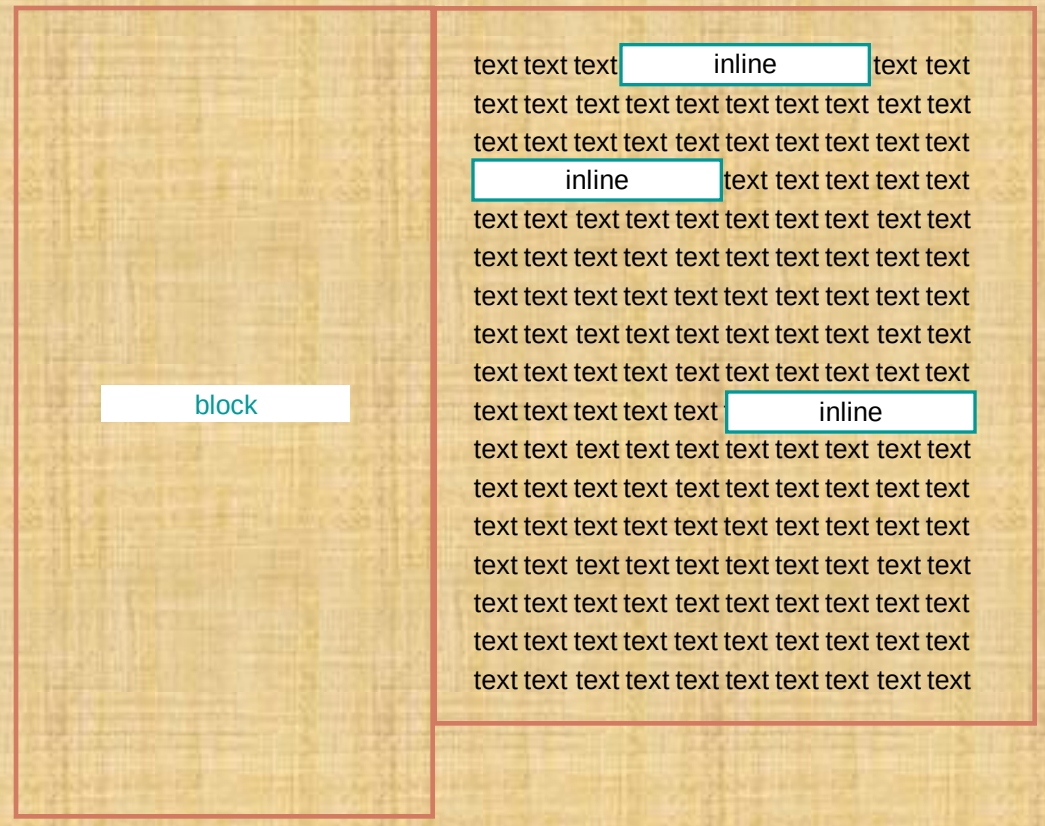
Setting the float property to left or right causes a box to be ***taken out of its position in the normal flow*** and moved as far left or right as possible.



Float Values

The Float property has three value options:

- float: left;
- float: right;
- float: none;



Restoring the Normal Flow: “Clear”

To restore the “normal flow”, we can use the clear property.

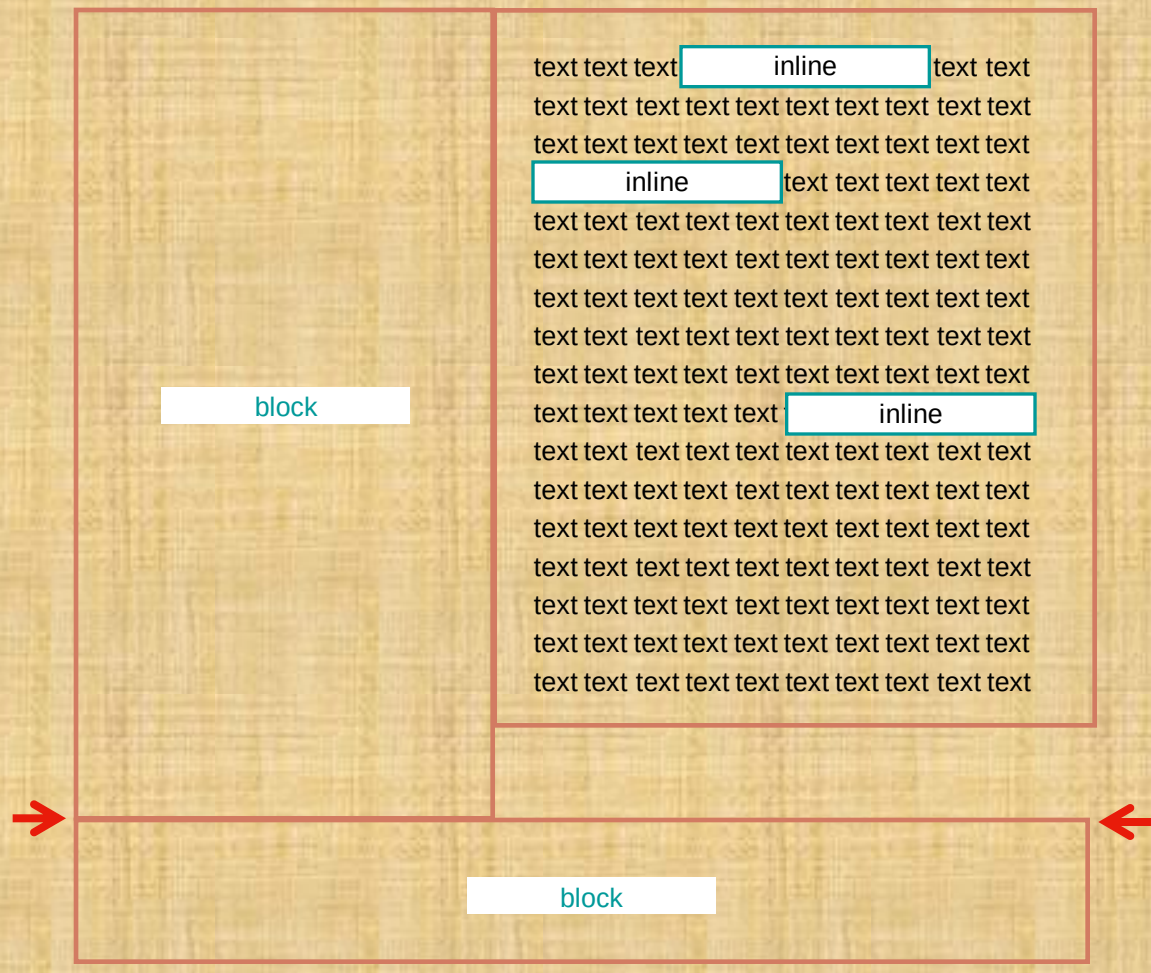
The clear property has three value options:

clear: left;

clear: right;

clear: both;

These specify which side of the element’s box may ***not*** have a float next to it.



CSS Positioning

The third core concept to understand in CSS layout (after the ‘box model’ and ‘floats’), is positioning.

There are two types of positioning that can be applied to CSS boxes:

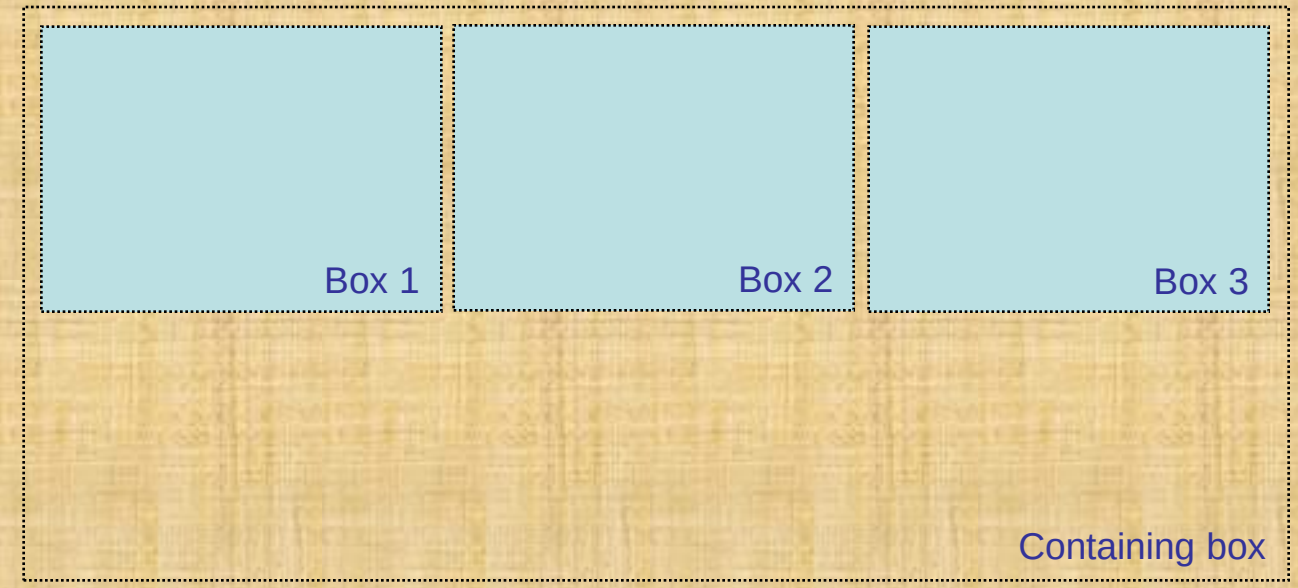
- **Relative Positioning**
- **Absolute Positioning**

Understanding the differences between the two is difficult at first, but important!

CSS Positioning: Relative Positioning

A relatively positioned element will stay exactly where it is, in relation to the normal flow.

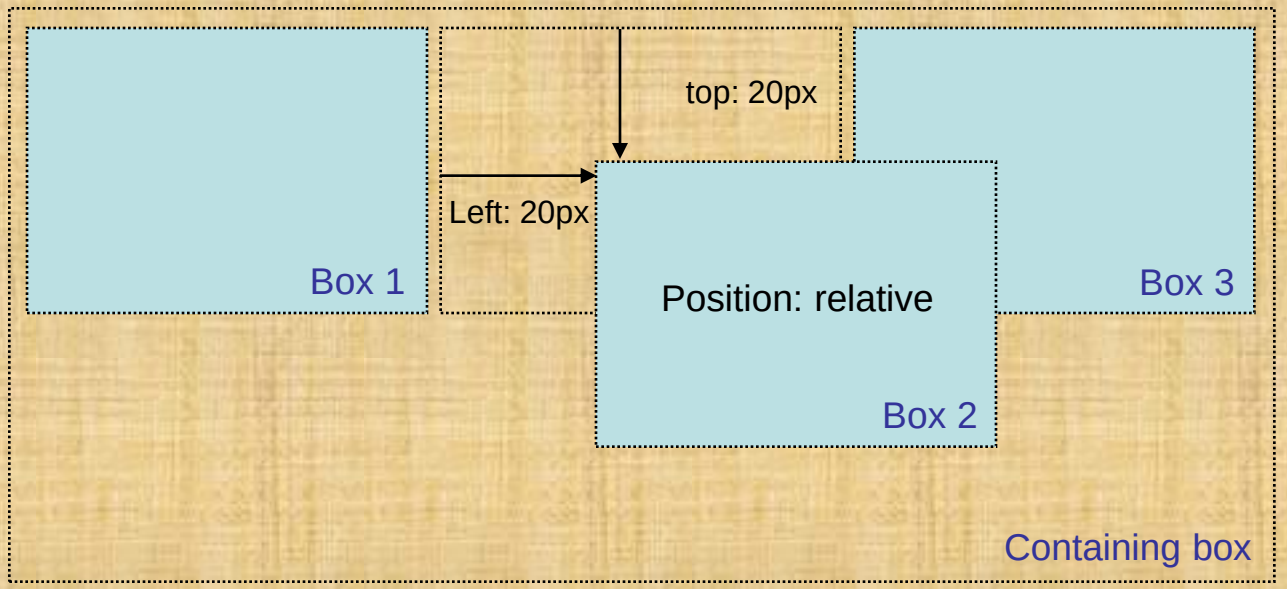
You can then offset its position “relative” to its starting point in the normal flow:



CSS Positioning: Relative Positioning

In this example, **box 2** is offset 20px, top and left. The result is the box is offset 20px from its original position in the normal flow. Box 2 may overlap other boxes in the flow, but other boxes still recognise its original position in the flow.

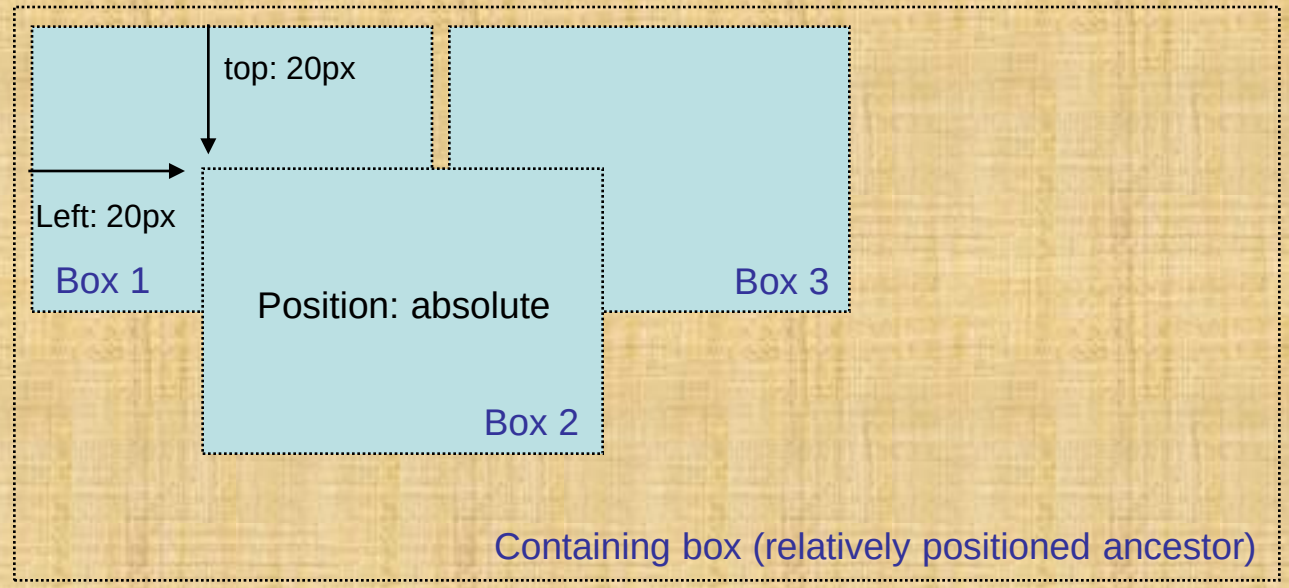
```
#myBox {
    position: relative;
    left: 20px;
    top: 20px;
}
```



CSS Positioning: Absolute Positioning

An absolutely positioned box is taken out of the normal flow, and positioned in relation to its nearest positioned ancestor (i.e. its containing box).

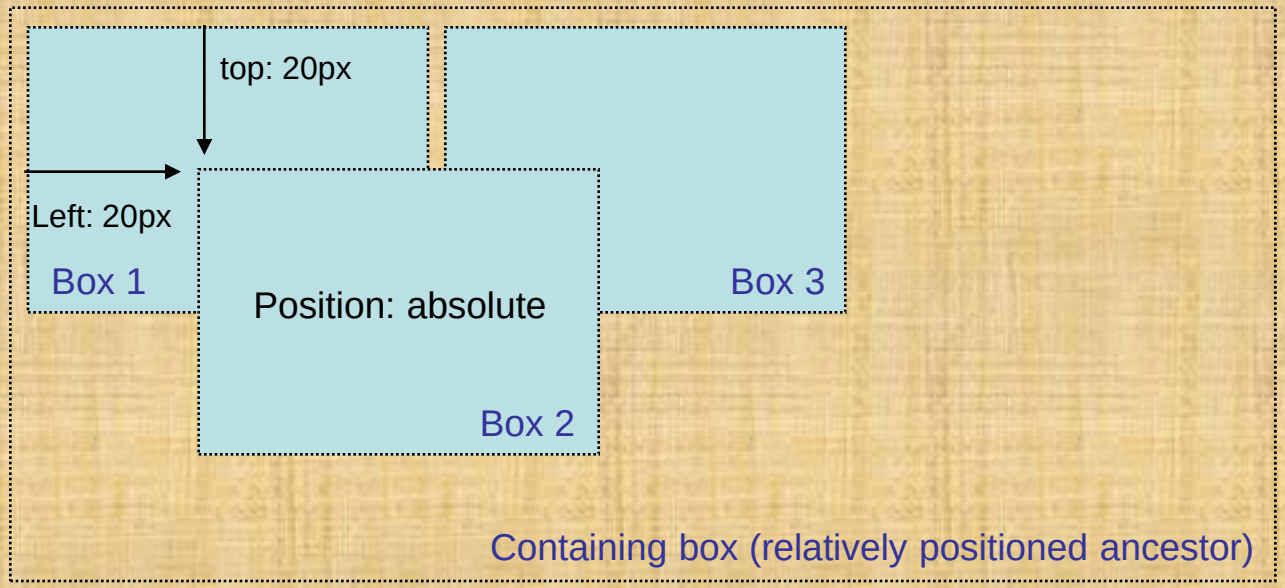
If there is no ancestor box, it will be positioned in relation to the initial containing block, usually the browser window.



CSS Positioning: Absolute Positioning

An absolutely positioned box can be offset from its initial position inside the containing block, but other boxes within the block (and still within the normal flow) act as if the box wasn't there.

```
#myBox {
    position: absolute;
    left: 20px;
    top: 20px;
}
```



CSS Positioning: Fixed Positioning

Fixed Positioning is a sub-category of Absolute Positioning

Allows the creation of floating elements that are always fixed in the same position in the browser window, while the rest of the content scrolls as normal

(rather like the effect of fixed background attachments)

PROBLEM: fixed positioning is not supported in IE5 and IE6(!), but can be made to work with javascript for those browsers

Floats & Positioning

Summary:

Floats (also a form of positioning) takes boxes out of the normal flow and “floats” them left or right edges of the containing block

Relative Positioning is “relative” to the element’s initial position in the normal flow.

Absolute Positioning is “relative” to the nearest positioned ancestor, or (if one doesn’t exist) the initial container block (usually the browser window)

Fixed Positioning is fixed in one position “relative” to the browser window only — does not scroll with content (Not supported in IE5, IE6)

Background Images in CSS

It is also possible to use the background CSS property any block element (including div's) to place a background image behind other elements.

Background images can be…

- *small images that repeat horizontally or vertically to fill a flexible background space, or*
- *one single image that fills a space of fixed size.*

Background Images in CSS: Fixed Position

Background images will normally scroll with the containing box, and the rest of the page

But they can also be “fixed”, staying in the same position in the layout, while the rest of the content scrolls.

```
#sidebar {
        float: right;
        width: 300px;
        margin-left: 25px;
        background-image: url(images/harbour.jpg);
        background-attachment: fixed;
}
```

Using Background Images

Background images are useful in allowing us to visually define a page, and separate content into a deliberate visual hierarchy.

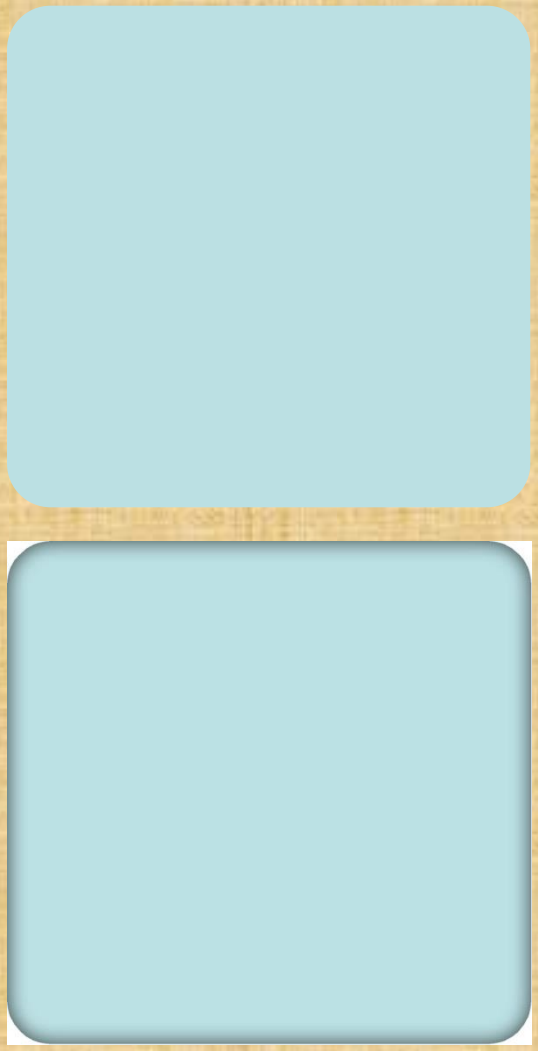
The ability to to repeat images in a background box, *and* reuse the SAME images across a number of boxes, means we can make very efficient use of images.

Wherever possible, background images should be used in conjunction with background colours.

Rounded Corner Boxes

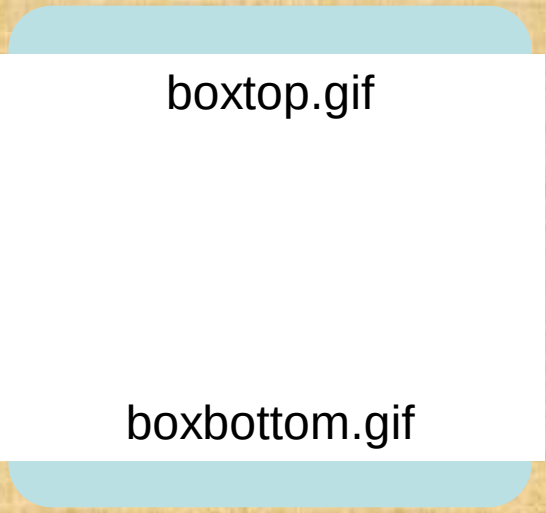
Rounded corner boxes are very popular. Unfortunately, the current version of CSS does not have any properties that can define a corner radius (CSS 3 will).

However, simple rounded corner boxes are very easy to create in CSS 2 with a couple of background images.



Fixed-Width Rounded Corner Boxes

Simple flat-colour, fixed-width rounded corner boxes require only 2 images:

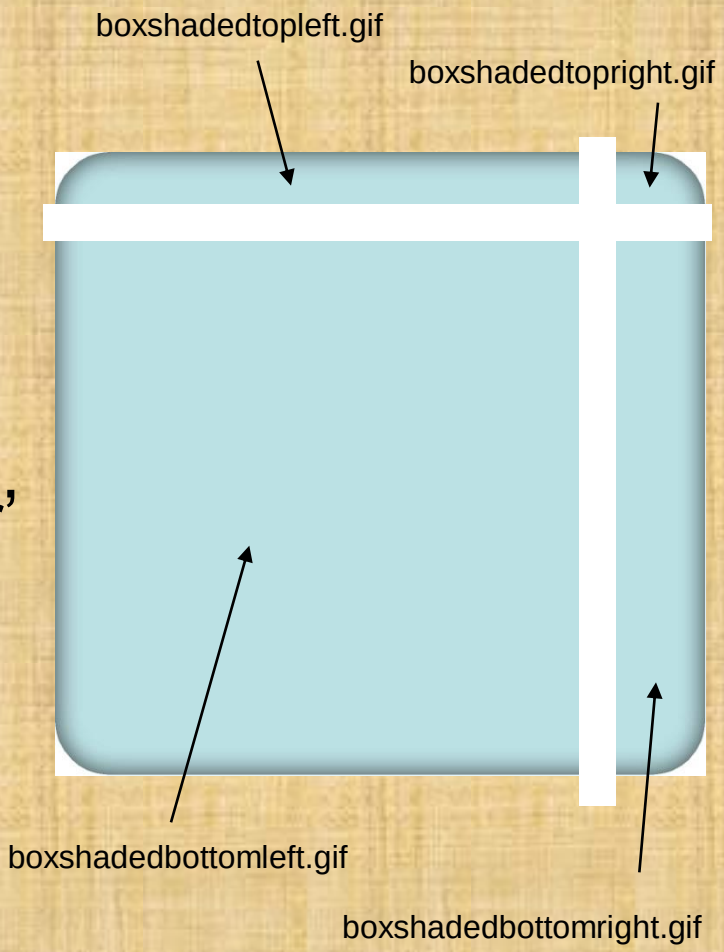


Fixed-width boxes with a shadow require 3 images, with the centre one set to repeat:

Flexible-Width Rounded Corner Boxes

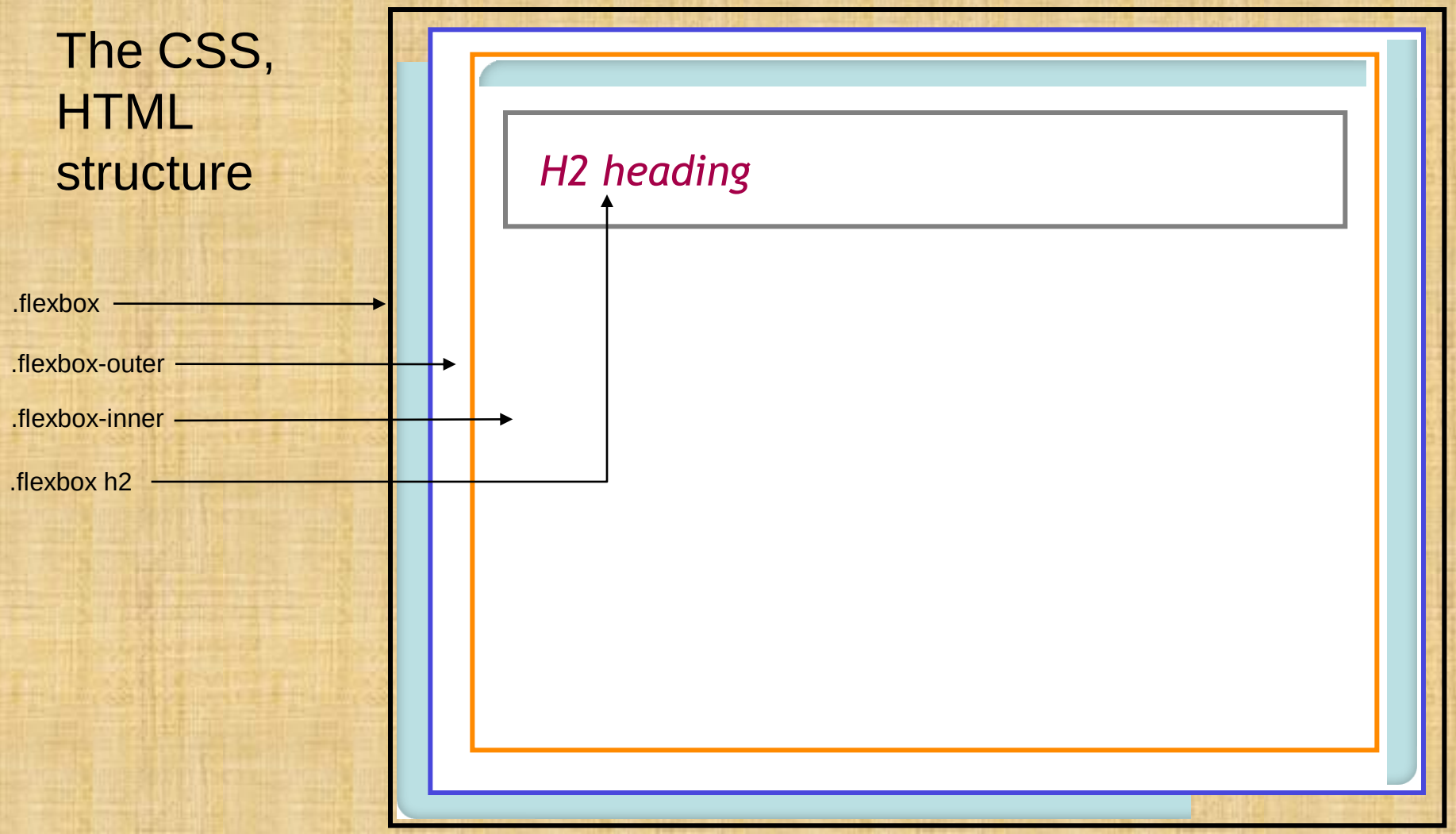
Flexible-width rounded corner boxes are a little more complicated...

You to break a large box into 4 images, then use CSS to have the right side pieces ‘slide over’ the larger (fixed-position) left pieces.



Flexible-Width Rounded Corner Boxes

The CSS, HTML structure



Centering in the Browser Window

We've seen how to center a box horizontally in the window:

Set the body to center

Create a container div (fixed or relative width), with left and right margins set to auto

So how do we float a CSS box vertically in the browser window?

Centering in the Browser Window

Like this:

Set the body to center

Create a container div (fixed or relative width), with..

- Position: absolute;
- top: 50%;
-
- left: 50%;
- margin-top: -300px;
-
- margin-left: -400px;

Note, the box must have a height specified, as well as a width, and it must be a measured unit (i.e. px, mm, com, etc.)

Using divs to Define CSS boxes

We've seen that CSS box model attributes can be applied to any simple XHTML block element (p, h1, etc.)

However, we often use specific elements within the XHTML, each identified with an ID or class name

We then apply width, float, margin, padding and border properties to those div's in CSS.

This gives us more options to create and control a range of layout elements, on top of core content elements.

Using divs to Define CSS boxes

id example: in the XHTML:

```
<div id="sidebar">
    <p> blah, blah, blah,
blah, blah, blah. </p>
</div>
```

In the CSS:

```
#sidebar {
    float: right;
    width: 300px;
    margin-left: 25px;
}
```

class example: in the XHTML:

```
<div class="sidebar">
    <p> blah, blah, blah,
blah.</p>

</div>
```

In the CSS:

```
.sidebar {
    float: right;
    width: 300px;
    margin-left: 25px;
}
```