

Lecture 10

Layering Absolutely Positioned Tags

- Absolutely positioned tags appear on top of other content.
- That includes other absolutely positioned tags.
- Normally, tags that appear farther down in your HTML file appear on top.

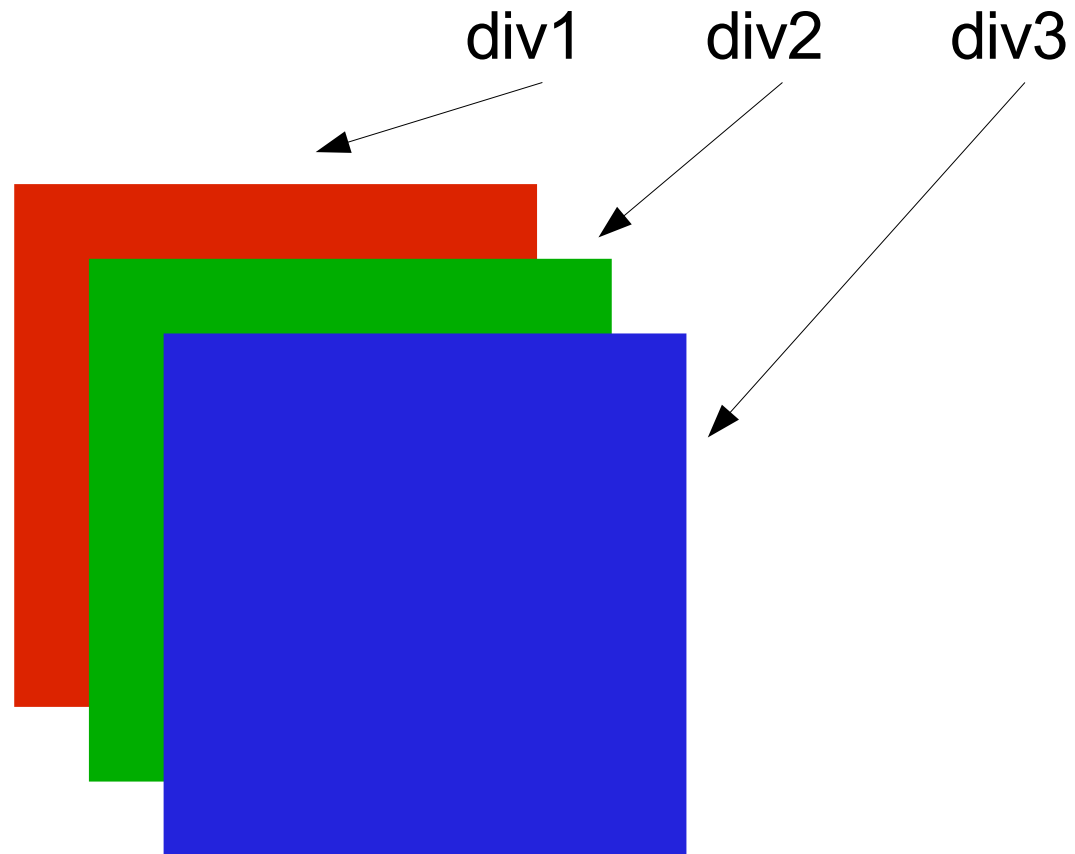
Layering Absolutely Positioned Tags

Example:

```
<div id="div1">      #div1, #div2, #div3
</div>              {
<div id="div2">      width: 100px;
</div>              height: 100px;
<div id="div3">      position: absolute;
</div>              }
                    #div1
                    {top: 0; left: 0; background: red;}
                    #div2
                    {top: 10px; left: 10px; background: green;}
                    #div3
                    {top: 20px; left: 20px; background: blue;}
```

Layering Absolutely Positioned Tags

Example:



Layering Absolutely Positioned Tags

- We can change the way that tags are ordered.
- The z-index CSS attribute sets the stacking order of a tag.
- It only works with positioned elements.

Layering Absolutely Positioned Tags

Example: Let's update the stylesheet:

```
#div1
```

```
{top: 0; left: 0; background: red; z-index: 3;}
```

```
#div2
```

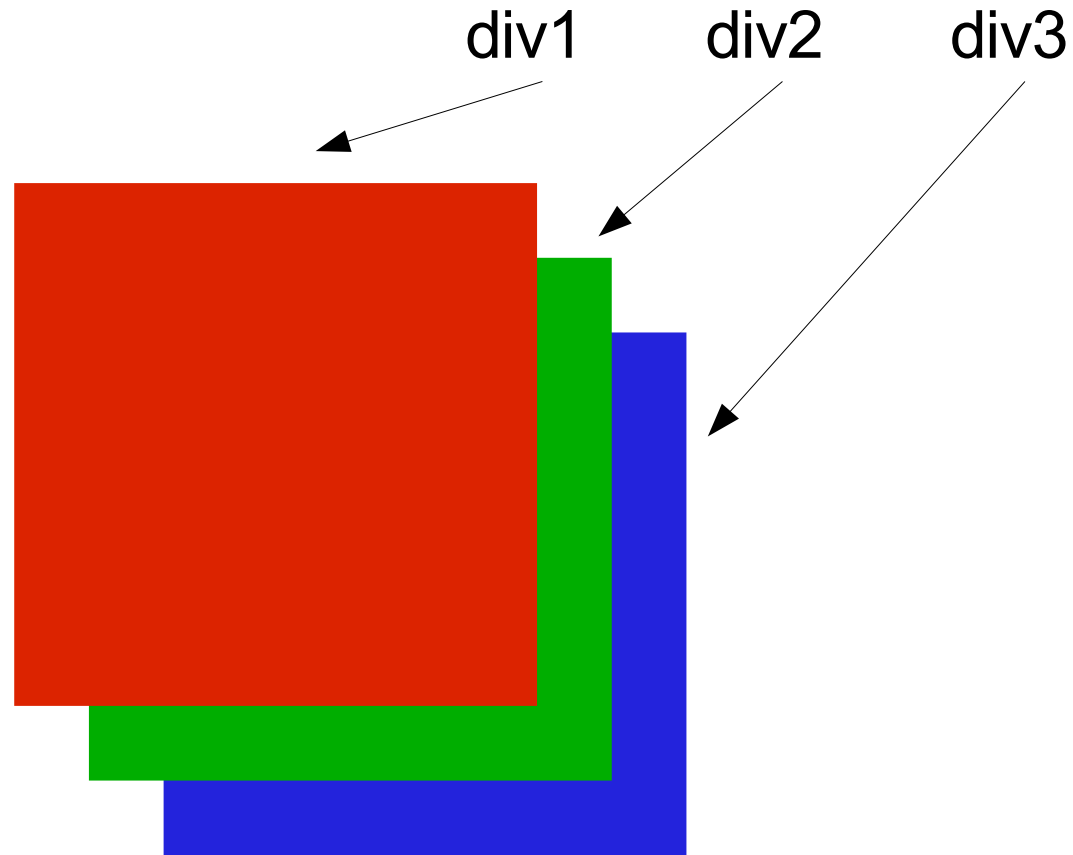
```
10px; left: 10px; background: green; z-index: 2;}
```

```
#div3
```

```
{top: 20px; left: 20px; background: blue; z-index: 1;}
```

Layering Absolutely Positioned Tags

Example:



The order has been changed, even though the HTML is the same!

Layering Absolutely Positioned Tags

- Tags with the same z-index are ordered according to their position in the document.
- By default, the z-index is 0.
- Tags with a negative z-index will appear below everything else.

Layering Absolutely Positioned Tags

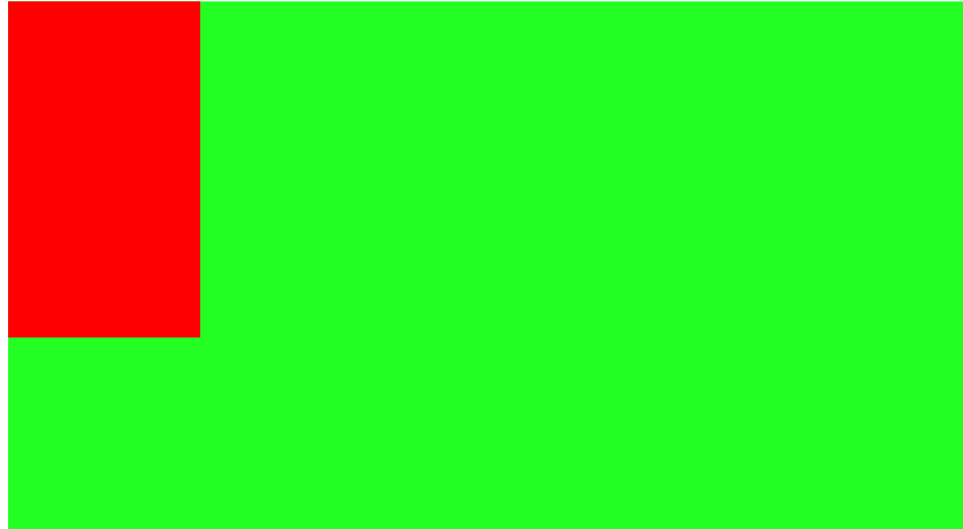
- The z-index depends on context.
- Setting the z-index for a tag affects all descendants
- If tag A has z-index 1 and tag B has z-index 2, and if C is a descendant of A that has z-index 200, it will still appear below B and all its descendants!
- Internet Explorer is buggy in its support for these rules.

Scaling

- If you fill a tag with text, it will scale to the right size to hold the content.
- Absolutely positioned tags are removed from the flow of content.
- This means that they do not scale their parent tags!

Scaling

Example: Suppose your layout looks like this:



- The green region holds the content of your page.
- The red region is an absolutely-positioned navigation bar.

Scaling

In code:

HTML:

```
<div id="content">  
  <div id="nav">  
  </div>  
</div>
```

CSS:

```
#content {width: 800px;}
```

```
#nav {position: absolute; width: 100px; height: 200px;}
```

Scaling

If you you're content does not exceed the height of your navigation bar, you'll get a situation like this:



We need to make sure that the green region is at least as tall as the red region.

Scaling

- In our stylesheet, we set the navigation bar to be 200 pixels tall:

```
#nav {position: absolute; width: 100px; height: 200px;}
```

- We can use the min-height attribute to make sure that the the green region is always at least 200 pixels tall:

```
#content {width: 800px; min-height: 200px;}
```

Warning!

- The min-height attribute does not work in some versions of Internet Explorer.
- To fix this, we will do our first IE hack.
- Most web pages are full of hacks to ensure support in different browsers.

An Internet Explorer Hack

We'll create another div to act as a spacer, and float it to the left.

HTML:

```
<div id="content">  
  <div id="nav">  
  </div>  
  <div id="spacer">  
  </div>  
</div>
```

An Internet Explorer Hack

We'll create another div to act as a spacer, and float it to the left.

CSS:

```
#spacer  
{  
    float: left;  
    width: 1px;  
    height: 200px;  
}
```

An Internet Explorer Hack

- The spacer is 200 pixels tall, and it forces Internet Explorer to make the green region at least 200 pixels high.
- This does not work in Firefox.
- For the green area to be the correct height in both IE and Firefox, you need both the IE hack and the min-height attribute.

Another Solution

- The body tag is an exception.
- It always scales to the size of the content.
- In this case, the green region has a different background color than the rest of the page.
- We can get the same effect using a background image.

Another Solution

Here's the syntax for adding a background image:

```
body
{
    background-image: url('[filename]');
}
```

- The green box is 800 pixels wide.
- We can make an 800 x 1 image that has the same shade of green and set it as the background.

Another Solution

Let's call the file green.gif.

```
body
{
    background-image: url('green.gif');
}
```

We're not done yet! The background image is tiled over the entire body unless we tell the browser otherwise. We do this with the background repeat attribute.

Background-Repeat

The background-repeat attribute can have for values:

repeat: The default. The background image is tiled over the whole page.

repeat-y: The background image is tiled only vertically.

repeat-x: The background image is tiled only horizontally.

no-repeat: The background image is not tiled in either direction.

Background-Repeat

The background-repeat attribute can have for values:

repeat: The default. The background image is tiled over the whole page.

repeat-y: The background image is tiled only vertically.

repeat-x: The background image is tiled only horizontally.

no-repeat: The background image is not tiled in either direction.

Background-Repeat

We'll make the green image tile only in the y direction:

```
body
{
    background-image: url('green.gif');
    background-repeat: repeat-y;
}
```

No the green region will always extend to the bottom edge of the browser window. Our problem is solved without a hack!