Positioning content

d-r Ivan Chorbev

Introduction

One of the best things about CSS is that it gives us the ability to position content and elements on a page in nearly any imaginable way, bringing structure to our designs and helping make content more digestible.

There are a few different types of positioning within CSS, and each has its own application.

The position property

The position property specifies the type of positioning method used for an element.

There are four different position values:

- static
- relative
- fixed
- absolute

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.

position: static;

HTML elements are positioned static by default.

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

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.

Example (code)

```
<!DOCTYPE html>
<html>
	<head>
	<style>
		div.static {
		    position: static;
		    border: 3px solid #73AD21;
		}
	</style>
	</head>
	<body>

		<h2>position: static;</h2>

		<p>An element with position: static; is not positioned in any special way; it is
		always positioned according to the normal flow of the page:</p>

		<div class="static">
			This div element has position: static;
		</div>

	</body>
</html>
```

Example (result)

position: static;

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;

position: 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.

Example (code)

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			div.relative {
			    position: relative;
			    left: 30px;
			    border: 3px solid #73AD21;
			}
		</style>
	</head>
	<body>

		<h2>position: relative;</h2>

		<p>An element with position: relative; is positioned relative to its normal position:</p>

		<div class="relative">
			This div element has position: relative;
		</div>

	</body>
</html>
```

Example (result)

position: relative;

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

This div element has position: relative;

position: 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.

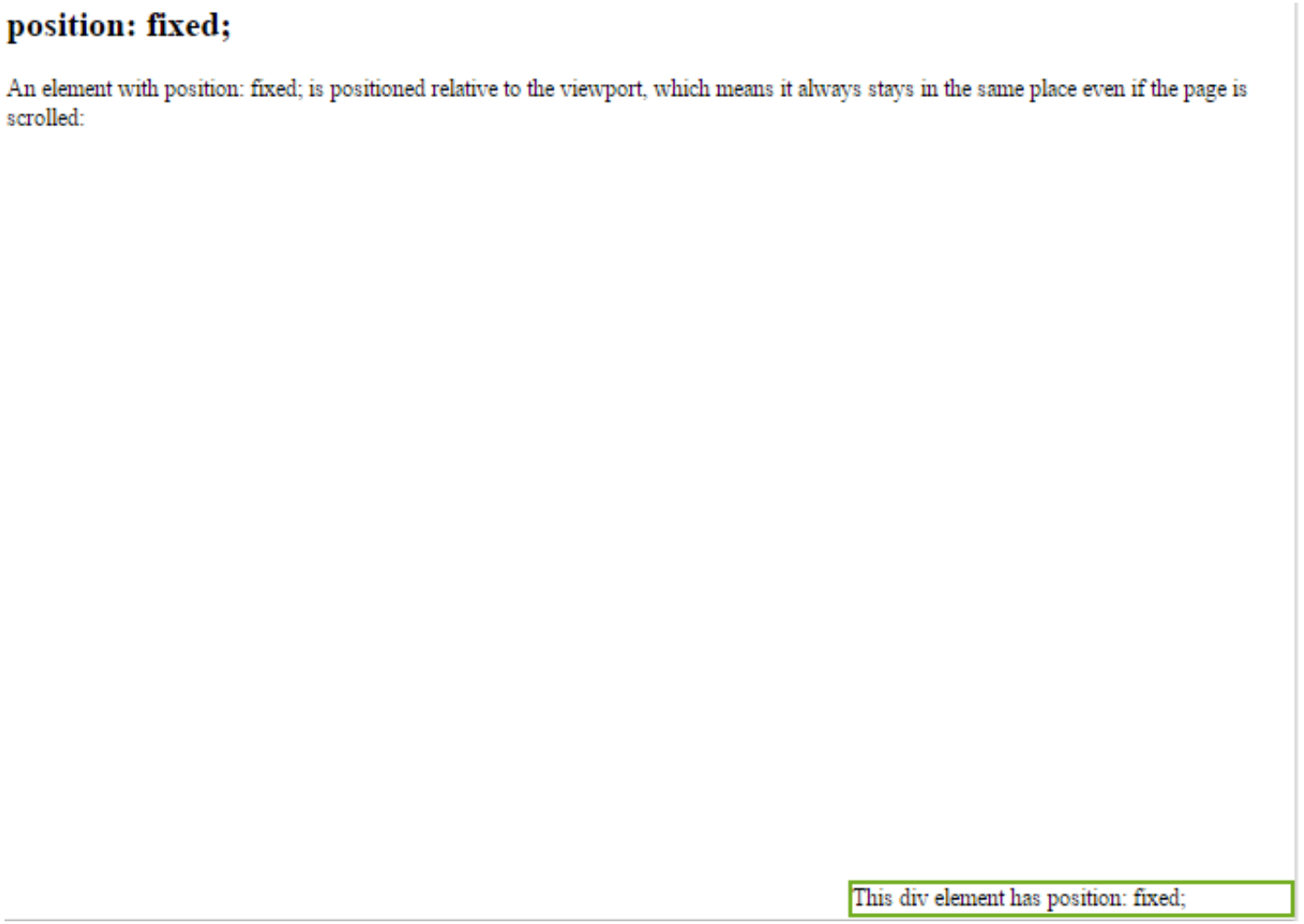
The top, right, bottom, and left properties are used to position the element.

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

Example (code)

```
<!DOCTYPE html>
<html>
	<head>
		<style>
			div.fixed {
			  position: fixed;
			  bottom: 0;
			  right: 0;
			  width: 300px;
			  border: 3px solid #73AD21;
			}
		</style>
	</head>
	<body>

		<h2>position: fixed;</h2>

		<p>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:</p>

		<div class="fixed">
			This div element has position: fixed;
		</div>

	</body>
</html>
```

Example (result)



position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

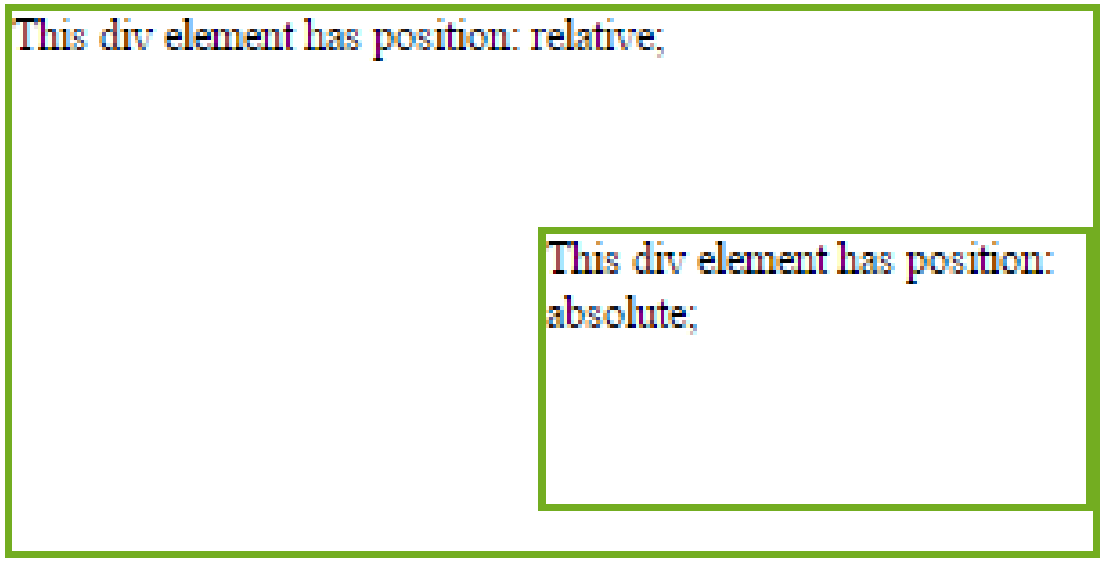
Example (code)

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

Example (result)

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):



More positioning examples (HTML)

```
<div id="example">

<div id="div-before">
<p>id = div-before</p>
</div>

<div id="div-1">
<div id="div-1-padding">

<p>id = div-1</p>

<div id="div-1a">
<p>id = div-1a</p>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis.
Integer sit amet diam. Phasellus ultrices viverra velit.</p>
</div>
```

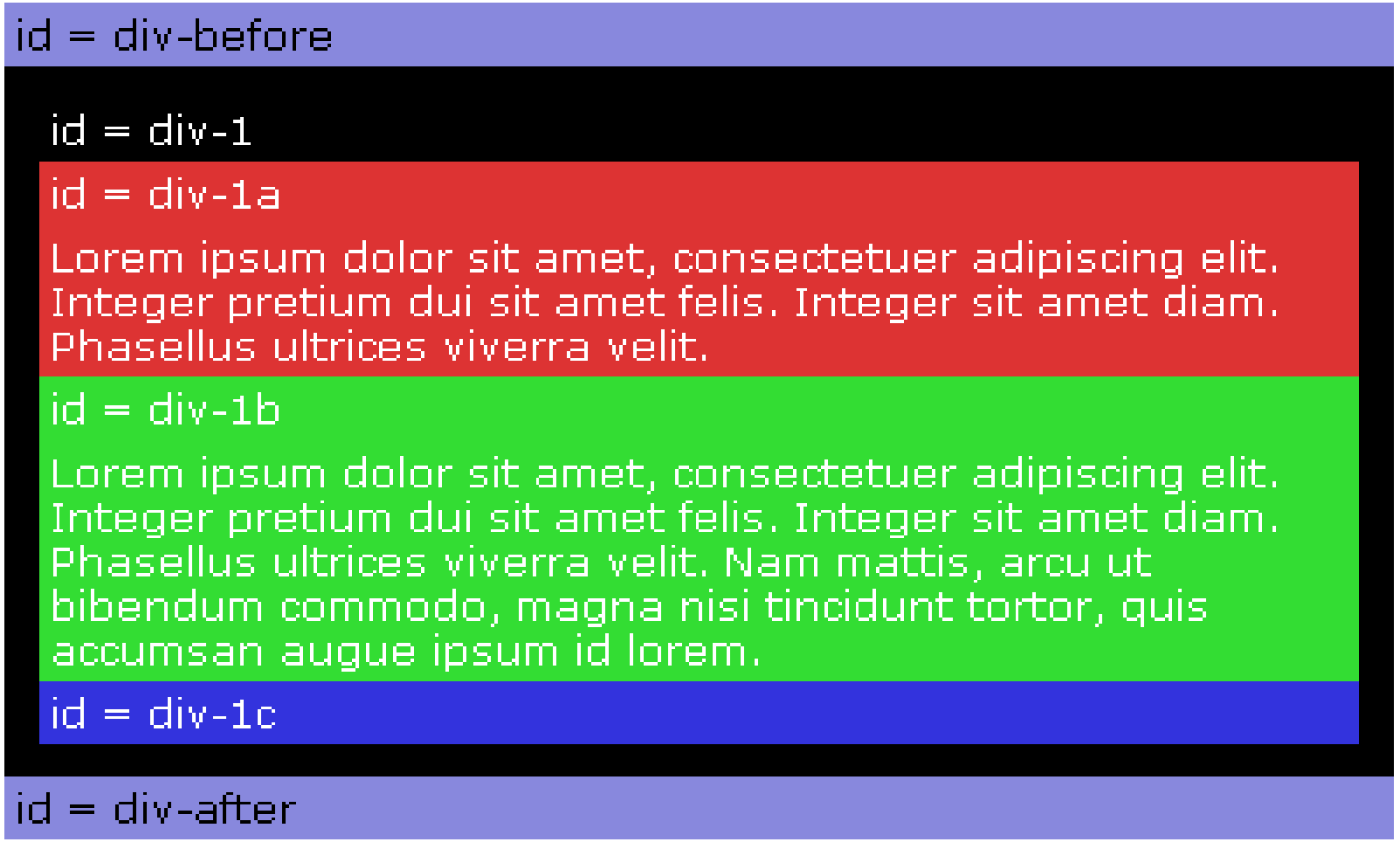
More positioning examples (HTML 2)

```
…
<div id="div-1b">
<p>id = div-1b</p>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis.
Integer sit amet diam. Phasellus ultrices viverra velit. Nam mattis, arcu ut bibendum commodo,
magna nisi tincidunt tortor, quis accumsan augue ipsum id lorem.</p>
</div>

<div id="div-1c"><p>id = div-1c</p></div>

</div></div><!-- /id=div-1-padding /id=div-1 -->

<div id="div-after">
<p>id = div-after</p>
</div>

</div>
```

position:static (Example)

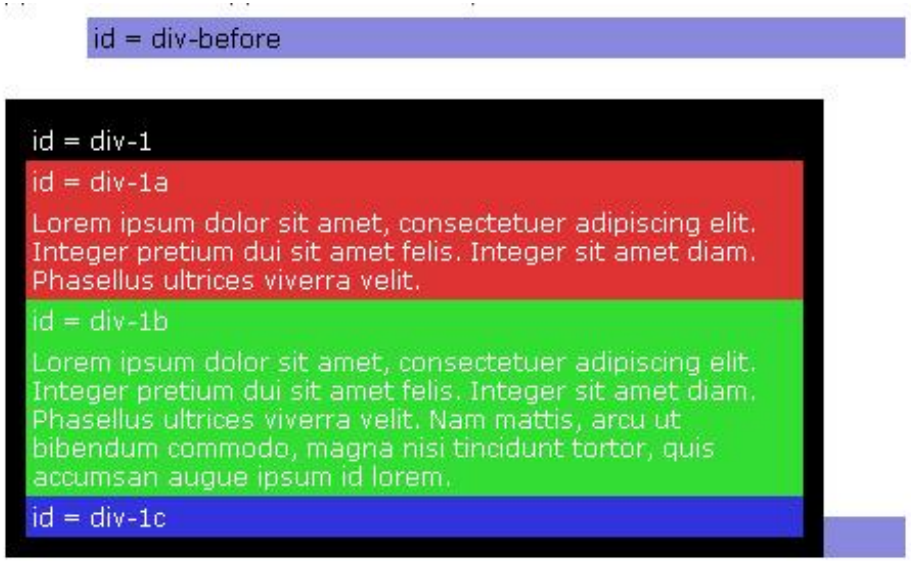
CSS:

```
#div-1 {
        position:static;
}
```



position:relative (Example)

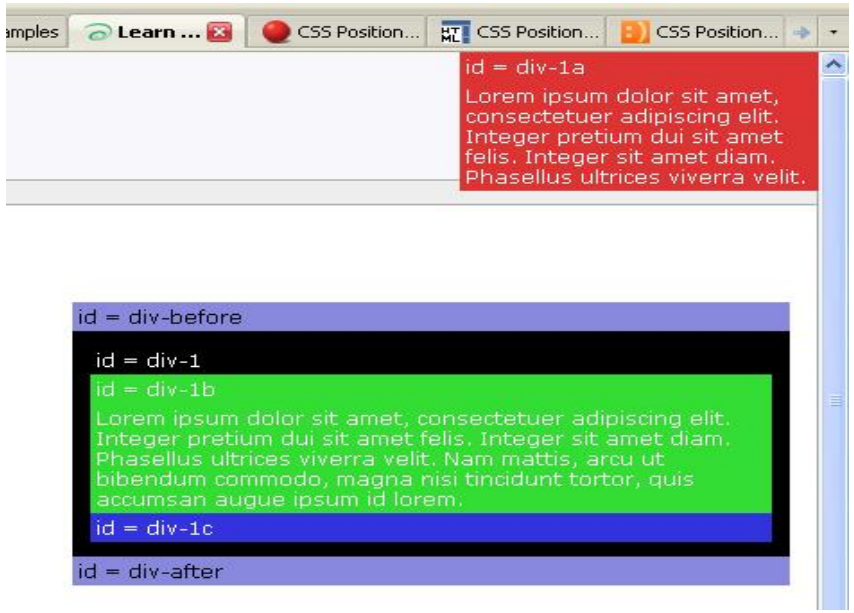
```
#div-1 {
        position:relative;
        top:20px;
        left:-40px;
}
```



Notice the space where div-1 normally would have been if we had not moved it: now it is an empty space. The next element (div-after) did not move when we moved div-1. That's because div-1 still occupies that original space in the document, even though we have moved it.

position:absolute (Example)

```
#div-1a {
        position:absolute;
        top:0;
        right:0;
        width:200px;

}
```



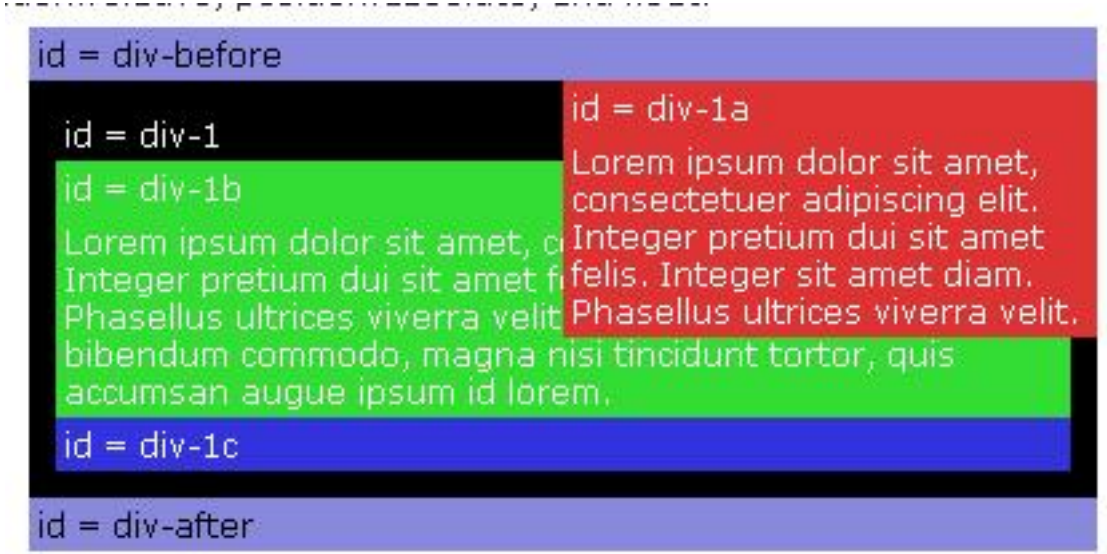
Notice that this time, since div-1a was removed from the document, the other elements on the page were positioned differently: div-1b, div-1c, and div-after moved up since div-1a was no longer there.

Also notice that div-1a was positioned in the top right corner of the page. It's nice to be able to position things directly the page, but it's of limited value.
What I really want is to position div-1a *relative* to div-1. And that's where relative position comes back into play.

position:relative + position:absolute

If we set *relative* positioning on div-1,
any elements within div-1 will be positioned relative to div-1.
Then if we set absolute positioning on div-1a,
we can move it to the top right of div-1:

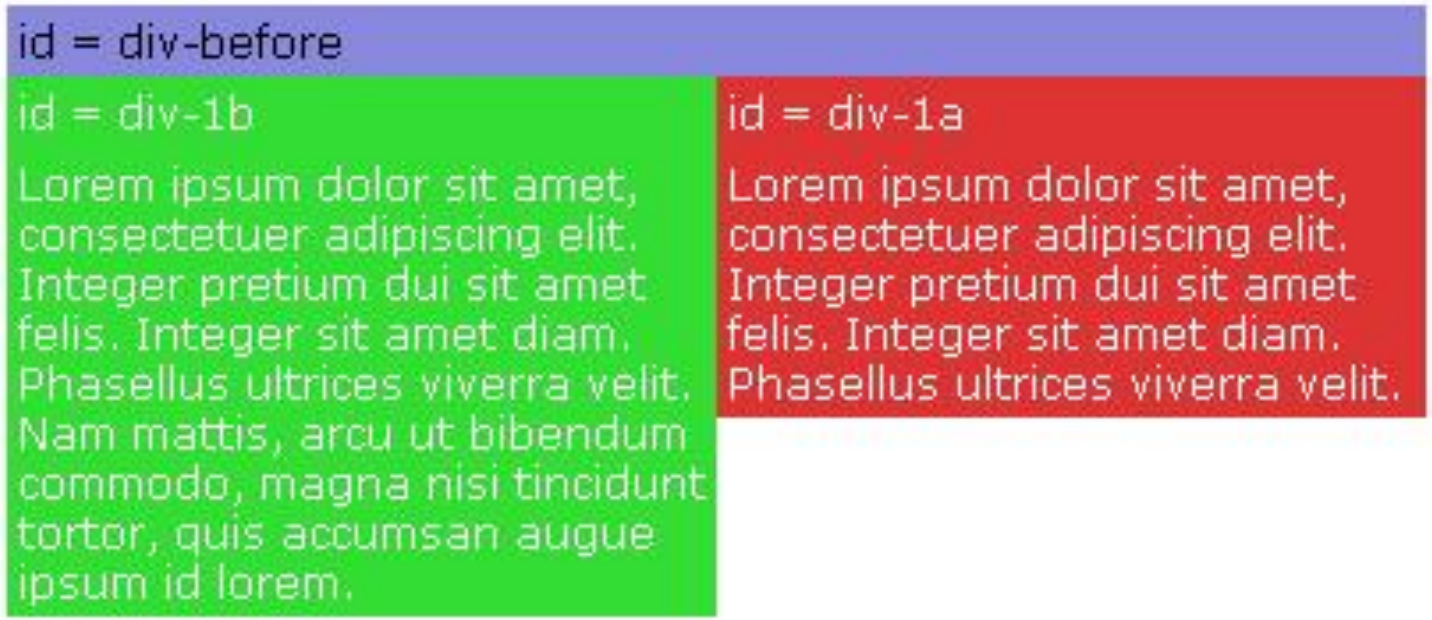
```
#div-1 {
        position:relative;
}

#div-1a {
        position:absolute;
        top:0;
        right:0;
        width:200px;
}
```



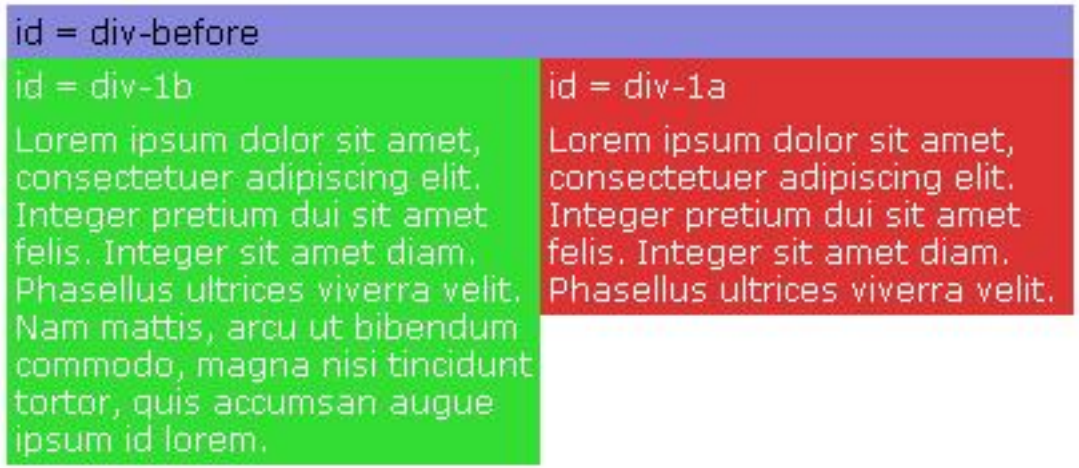
two column absolute

Now we can make a two-column layout using relative and absolute positioning!

```
#div-1 {
        position:relative;
}

#div-1a {
        position:absolute;
        top:0; right:0; width:200px;
}
#div-1b {
        position:absolute;
        top:0;
        left:0;
        width:200px;
}
```



two column absolute (Result)



One advantage to using absolute positioning is that we can position the elements in any order on the page, regardless of the order they appear in the HTML. So I put div-1b before div-1a.

But what happened to the other elements? They are being obscured by the absolutely positioned elements. What can we do about that?

two column absolute height (Result)

One solution is to set a fixed height on the elements.
But that is not a viable solution for most designs, because we usually do not know how much text will be in the elements, or the exact font sizes that will be used.

```
#div-1 {
        position:relative;
        height:250px;
}
#div-1a {
        position:absolute;
        top:0;
        right:0;
        width:200px;
}
#div-1b {
        position:absolute;
        top:0; left:0; width:200px;
}
```

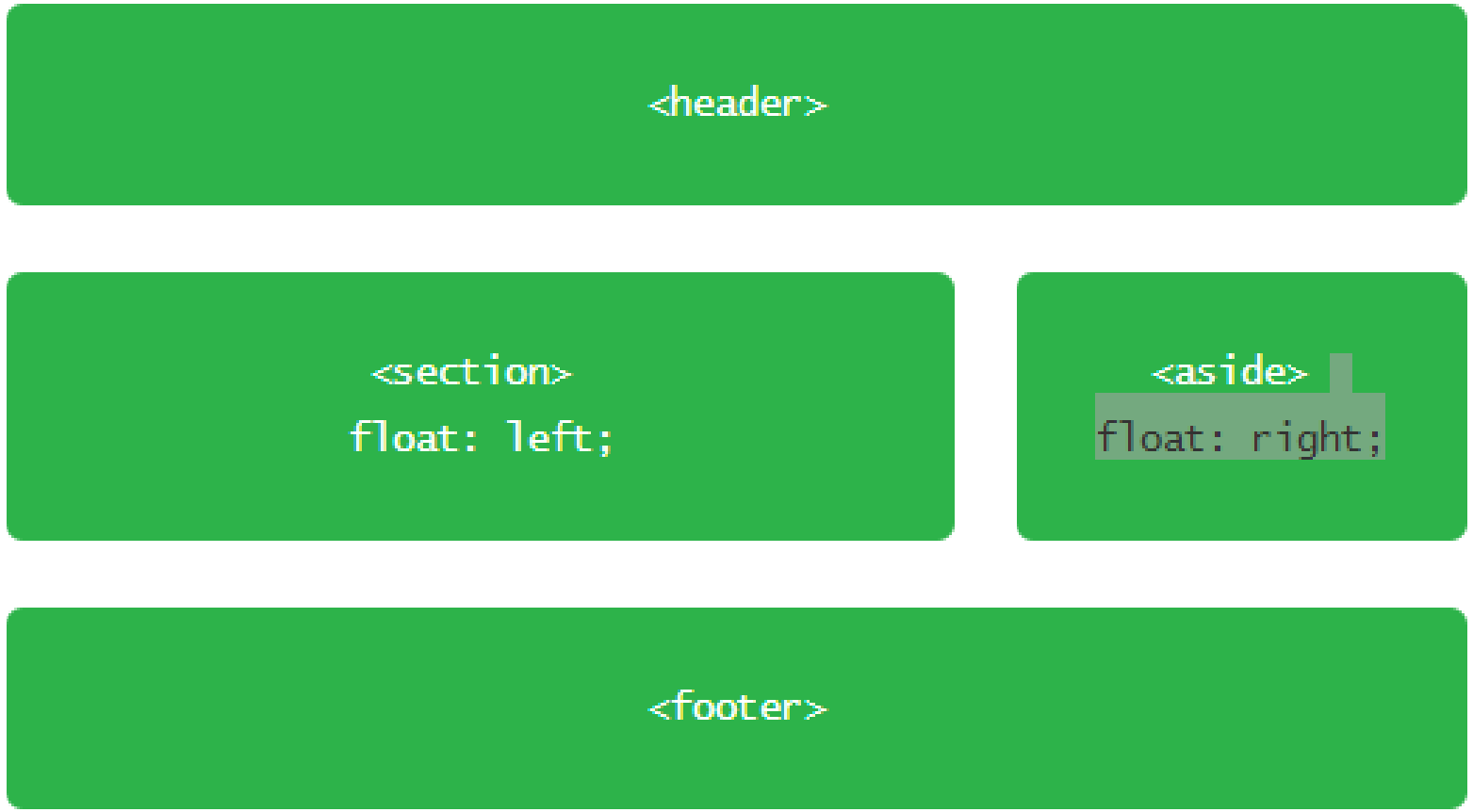
Positioning with Floats

One way to position elements on a page is with the float property.
The float property is pretty versatile and can be used in a number of different ways.

Essentially, the float property allows us to take an element, remove it from the normal flow of a page, and position it to the left or right of its parent element. All other elements on the page will then flow around the floated element.

An <img> element floated to the side of a few paragraphs of text, for example, will allow the paragraphs to wrap around the image as necessary.

Example 1



Example 1 (HTML)

```
<header>
        <code>&#60;header&#62;</code>
</header>

<section>
        <code>&#60;section&#62; <br> float: left;</code>
</section>

<aside>
        <code>&#60;aside&#62; <br> float: right;</code>
</aside>

<footer>
        <code>&#60;footer&#62;</code>
</footer>
```

Example 1 (CSS)

```
code {
        background: #2db34a;
        border-radius: 6px;
        color: #fff;
        display: block;
        font: 14px/24px "Source Code Pro", Inconsolata, "Lucida Console", Terminal, "Courier New", Courier;
        padding: 24px 15px;
        text-align: center;
}
header,
section,
aside,
footer {
        margin: 0 1.5% 24px 1.5%;
}
section {
        float: left;
        width: 63%;
}
aside {
        float: right;
        width: 30%;
}
footer {
        clear: both;
        margin-bottom: 0;
}
```

Example 2


id = div-before
id = div-1
id = div-1a
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit.
id = div-1b
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pretium dui sit amet felis. Integer sit amet diam. Phasellus ultrices viverra velit. Nam mattis, arcu ut bibendum commodo, magna nisi tincidunt tortor, quis accumsan augue ipsum id lorem.
id = div-1c
id = div-after

Example 2 (Code & Explanation)

```
#div-1a {
        float:left;
        width:150px;
}

#div-1b {
        float:left;
        width:150px;
}
```

If we float one column to the left, then also float the second column to the left, they will push up against each other. We can "float" an element to push it as far as possible to the right or to the left, and allow text to wrap around it.
This is typically used for images, but we will use it for more complex layout tasks (because it's the only tool we have).

Clearing floats

Clearing floats is accomplished using the clear property, which accepts a few different values: the most commonly used values being left, right, and both.

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

The left value will clear left floats, while the right value will clear right floats. The both value, however, will clear both left and right floats and is often the most ideal value.

Example

We can "clear" the floats to push down the rest of the content.

```
#div-1a {
        float:left;
        width:190px;
}

#div-1b {
        float:left; width:190px;
}

#div-1c {
        clear:both;
}
```

Containing Floats

Rather than clearing floats, another option is to contain the floats.

The outcomes of containing floats versus those of clearing them are nearly the same; however, containing floats does help to ensure that all of our styles will be rendered properly.

To contain floats, the floated elements must reside within a parent element. The parent element will act as a container, leaving the flow of the document completely normal outside of it.

Containing Floats 2

The CSS for that parent element, represented by the group class below, is shown here:

```
.group:before,
.group:after {
        content: "";
        display: table;
}
.group:after {
        clear: both;
}
.group {
        clear: both;
        *zoom: 1;
}
```

Containing Floats 3

Essentially what the CSS is doing is clearing any floated elements within the element with the class of group and returning the flow of the document back to normal.

More specifically, the :before and :after pseudo-elements are dynamically generated elements above and below the element with the class of group.

Those elements do not include any content and are displayed as table-level elements, much like block-level elements.

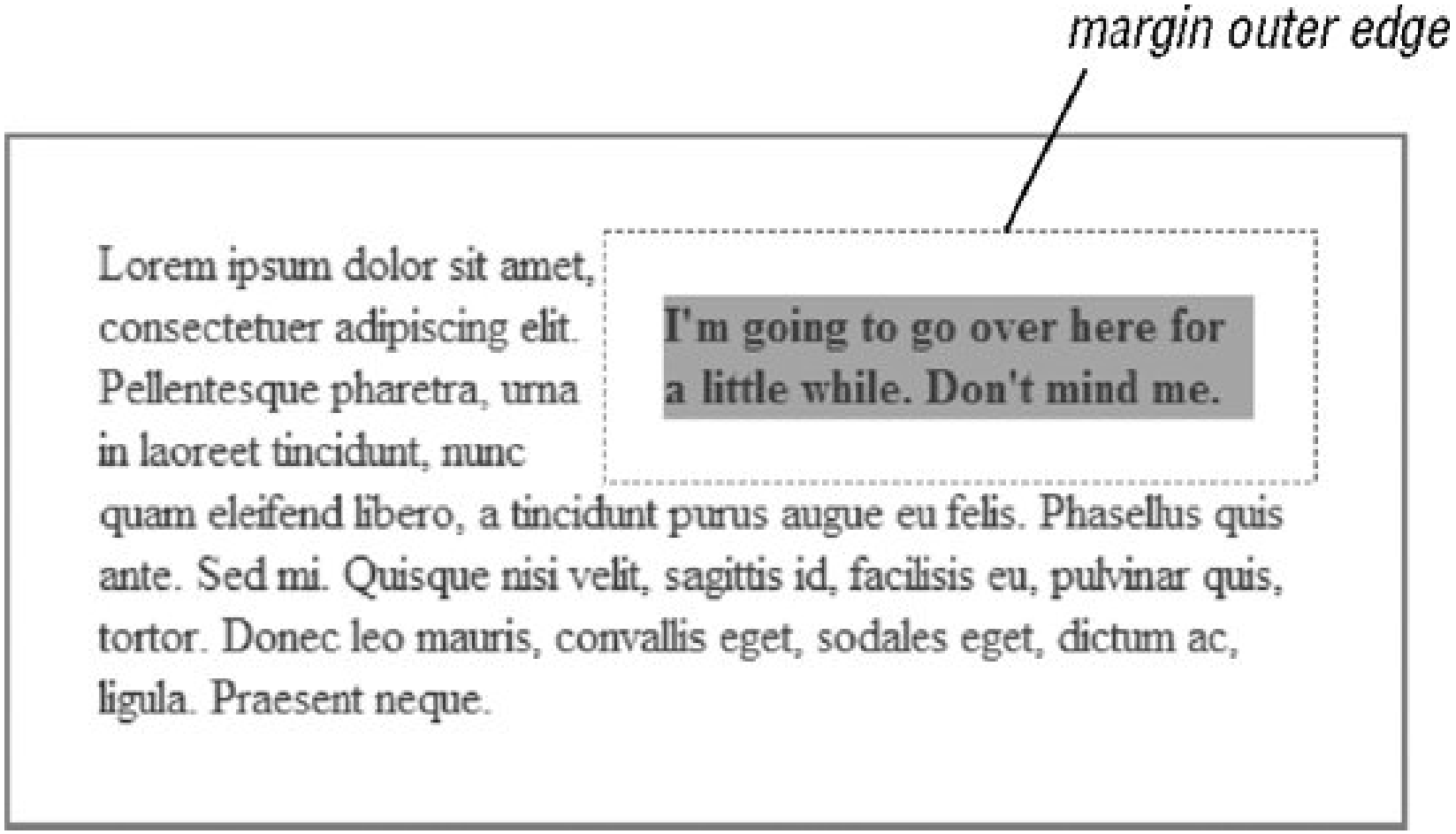
Properties classification

Property	Description	Values
clear	Defines the sides of the elements where other floating elements must not appear	Left right both none
cursor	Cursor type	url auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help

Properties classification 2

display	Defines how an element is displayed	None inline block list-item run-in compact marker table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption
float	Defines floating	Left right none
position	Set position	Static relative absolute fixed

Example


margin outer edge
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque pharetra, urna in laoreet tincidunt, nunc
I'm going to go over here for a little while. Don't mind me.
quam eleifend libero, a tincidunt purus augue eu felis. Phasellus quis ante. Sed mi. Quisque nisi velit, sagittis id, facilisis eu, pulvinar quis, tortor. Donec leo mauris, convallis eget, sodales eget, dictum ac, ligula. Praesent neque.

Example (code)

CSS:

```
span.note {
        float: right;
        width: 200px;
        margin: 20px;
        background-color: #999;
        font-weight: bold;
}

p {
                border: solid 2px #666; padding: 30px;
}
```

HTML:

```
<p><span class="note">I'm going to go over here for a little while. Don't
mind me. </span> Lorem ipsum dolor sit amet, consectetuer  . . . </p>
```

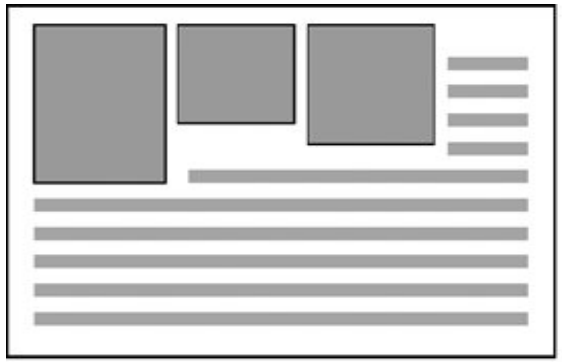
Rules

- All float elements become block elements
- When text floats, the width must be set, otherwise it will shrink as much as possible.
- A floating element does not pass over the containing block. Does not go over the padding.
- Margins do not collapse for floating elements.

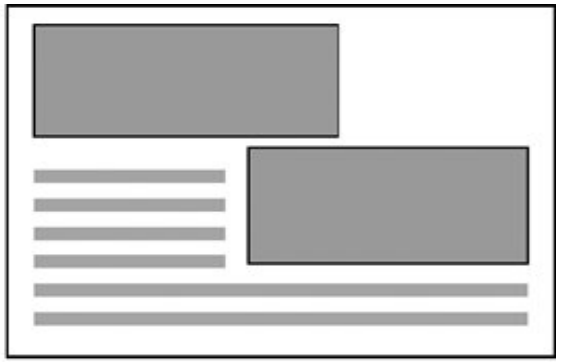
Rules 2

- When elements float in the same direction, every following element moves towards the same direction until it hits in the border of the block or in the preceding element.

- If there is not enough room for the floating elements to be beside each other, the second goes downwards looking for room.



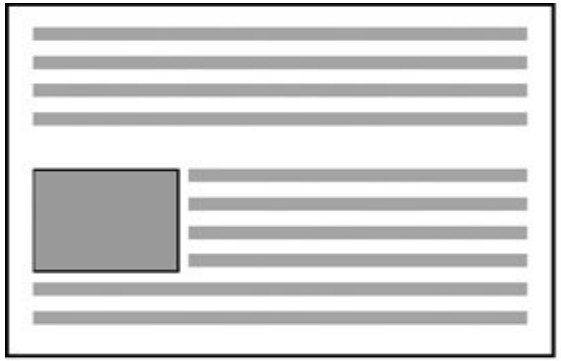
consecutive objects floated in the same direction



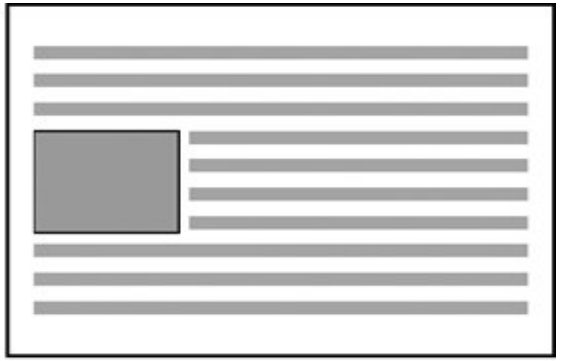
objects that exceed the width of the containing block, floated in opposite directions

Rules 3

- The top of a floating element must remain in the block.
- The top of a floating element can not go higher than the previous block.



Floated objects will not rise higher than the top of their parent elements and will not go above preceding block elements.



Floated objects will not rise higher than the line box from which they originate.

Positioning

property	values
bottom	Auto % length
clip	Shape auto
left	Auto % length
overflow	Visible hidden scroll auto
position	Static relative absolute fixed

Positioning

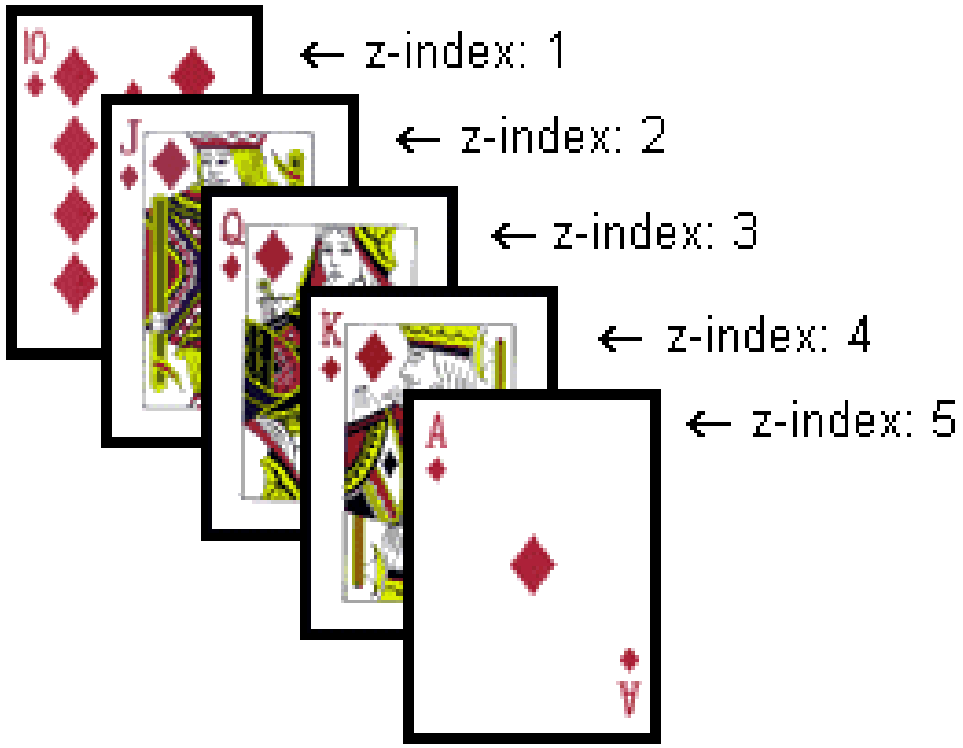
right	Auto % length
top	Auto % length
vertical-align	Baseline sub super top text-top middle bottom text-bottom length %
z-index	Auto number

Example

Example (Code)

```
<html>
	<head>
		<style type="text/css">
		h1.goreLevo { position:absolute; top: 50px;
				left: 50px;	}
		h1.goreDesno {	position:absolute; top: 50px;
				right: 50px; }
		h1.doluLevo { position:absolute; bottom:50px;
				left: 50px;	}
		h1.doluDesno {	position: absolute; bottom: 50px;
				right: 50px;}
		</style>
	</head>
	<body>
		<h1 class="goreLevo">GL</h1>
		<h1 class="goreDesno">GD</h1>
		<h1 class="doluLevo">DL</h1>
		<h1 class="doluDesno">DD</h1>
	</body>
</html>
```

Z-index



Z-index (code)

```
<html>
    <head>
        <style type="text/css">
            img.x
            {
                position:absolute;
                left:10px;
                top:10px;
                z-index:-1
            }
        </style>
    </head>
    <body>
        <h1>Наслов</h1>
        <img class="x" src="krug.gif" width="100" height="180">
        <p>Преддефинираната вредност на z-index е 0. Z-index = -1
            има понизок приоритет
        </p>
    </body>
</html>
```

Questions