

# **Advanced CSS Training**

**Positioning and Visibility**

## Lesson 1, Activity 2: Element Flow

The flow is the way in which elements are laid out. By default, sibling elements are all in the same flow and positioned statically on the page; the order elements appear on the page is the same as the order they appear in the code. With CSS, we can move elements into a separate flow. This allows us more control over the design of the page and over the layout of our code.

### Position

The `position` property is used to determine how an element is positioned. The default value is `static`. Other options are:

- `absolute`
- `relative`
- `fixed`

### Absolute Positioning

When elements are positioned absolutely, they are removed from the normal flow. As a result, they do not affect the positioning of subsequent sibling elements. Elements are positioned absolutely by setting the `position` property to `absolute` and specifying one or more "offset" properties.

### Offset Properties: top, right, bottom, and left

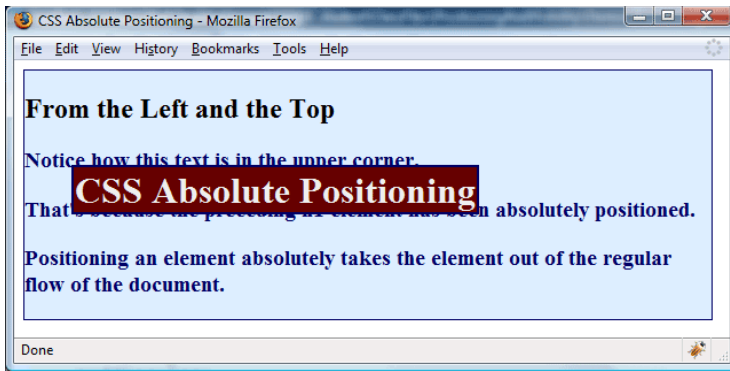
The "offset" properties are `top`, `right`, `bottom`, and `left`. Their values can be specified in number of units (e.g, 10px) or percentage of the containing block (e.g, 20%). These properties offset the element from its nearest non-statically positioned containing block element (i.e, with `position` set to `absolute`, `relative` or `fixed`). If it has no ancestor that is non-statically positioned, then it is offset from the browser window.

### Code Sample:

[PositioningAndVisibility/Demos/PositioningAbsolute.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
h1 {
  position:absolute;
  top:70px;
  left:50px;
  border:2px solid #006;
  padding:1px;
  background-color:#600;
  color:#eee;
}
#explanation {
  color:#006;
  font-weight:bold;
  font-size:1.2em;
}
#wrapper {
  width:600px;
  background-color:#def;
  border:1px solid #006;
}
</style>
<title>CSS Absolute Positioning</title>
</head>
<body>
<div id="wrapper">
  <h1>CSS Absolute Positioning</h1>
  <h2>From the Left and the Top</h2>
  <div id="explanation">
    <p>Notice how this text is in the upper corner.</p>
    <p>That's because the preceding h1 element has been absolutely positioned.</p>
    <p>Positioning an element absolutely takes the element out of the regular flow of the document.</p>
  </div>
</div>
</body>
</html>
```

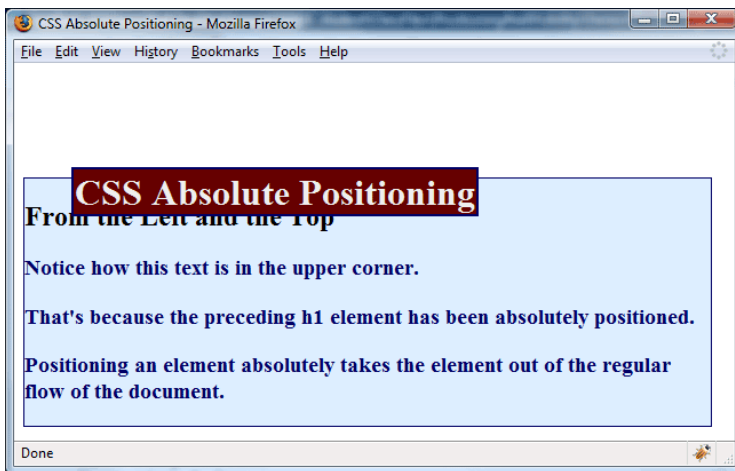
The code above will render the following results.



Notice how the positioning of the sibling h2 and div elements is not affected by the h1 element that precedes them. This is because the h1 element has been absolutely positioned.

Now watch what happens if we push the containing "wrapper" div down 100 pixels using the margin-top property like this:

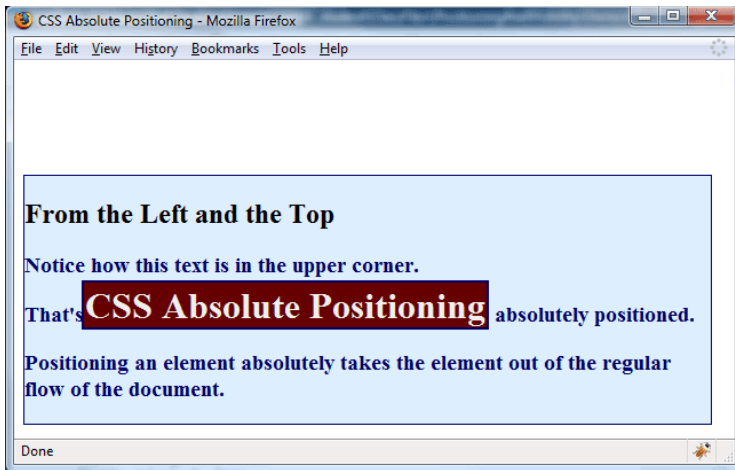
```
#wrapper {
  margin-top:100px;
  width:600px;
  background-color:#def;
  border:1px solid #006;
}
```



Notice that the h1 element was not affected at all. That's because the "wrapper" div is not in the same flow as the h1 element. To get it in the same flow, we need to absolutely position it as well. We do that in the code below and move the div using the top property rather than margin-top:

```
#wrapper {
  position:absolute;
  top:100px;
  width:600px;
  background-color:#def;
  border:1px solid #006;
}
```

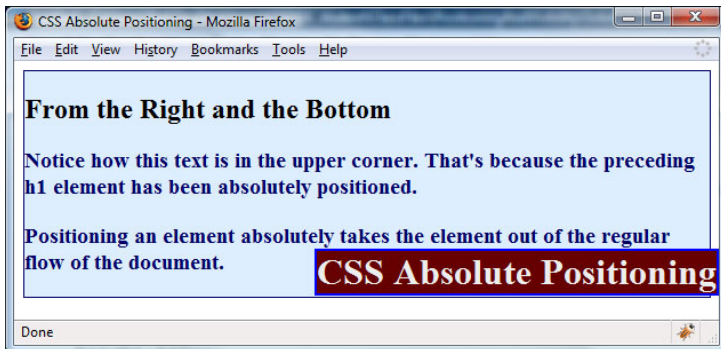
The result is shown below:



Now let's position the h1 element from the bottom right with this code:

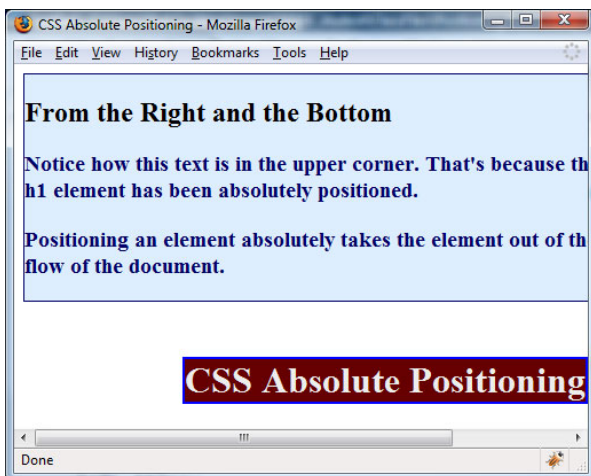
```
h1 {
  position: absolute;
  bottom: 0px;
  right: 0px;
  border: 2px solid #00f;
  background-color: #600;
  color: #eee;
}
```

The code above will render the following results:



You'll notice that Firefox doesn't put the element at the very bottom of the window. The distance from the bottom will vary between browsers, but all of them seem to give some buffer.

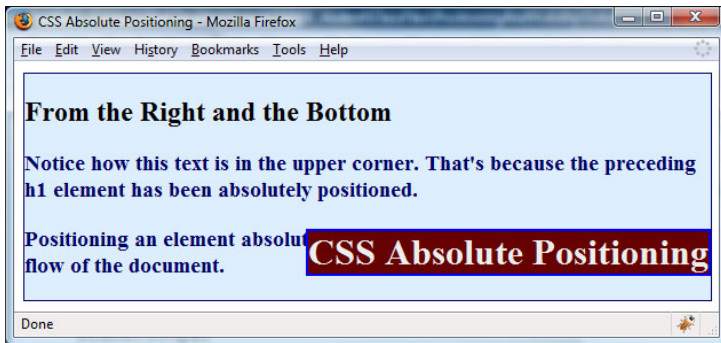
As you can see in the screen shot below the h1 element stays the same distance from the right bottom when the browser window is resized:



Again, if the "wrapper" div is also absolutely positioned then the h1 element is positioned within that div element. So the code below would render the page below it:

```
#wrapper {
  position: absolute;
  top: 10px;
```

```
width:600px;
background-color:#def;
border:1px solid #006;
}
```



Notice that the h1 element is fully contained in the "wrapper" div.

Note that when an inline element is positioned absolutely, it becomes a block-level element.

### Relative Positioning

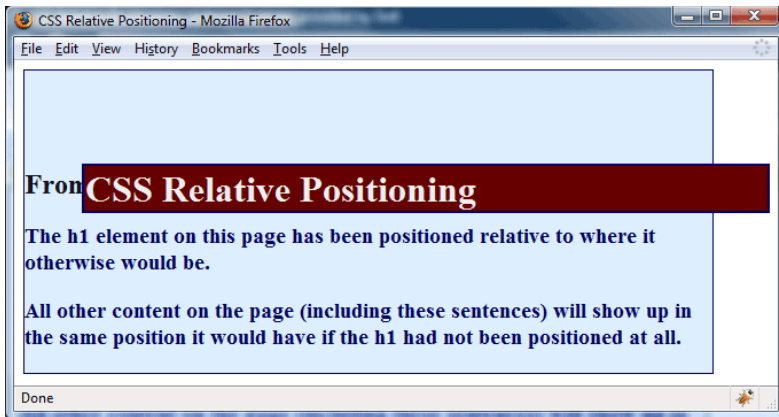
When elements are positioned relatively, they are positioned relative to where they would normally appear in the flow. Unlike absolutely positioned elements, relatively positioned elements *do* affect the positioning of subsequent sibling elements. Elements are positioned relatively by setting the `position` property to `relative` and specifying one or more "offset" properties as described above in ["Offset Properties: top, right, bottom, and left"](#).

### Code Sample:

[PositioningAndVisibility/Demos/PositioningRelative.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
h1 {
  position:relative;
  top:60px;
  left:50px;
  border:2px solid #006;
  padding:1px;
  background-color:#600;
  color:#eee;
}
#explanation {
  color:#006;
  font-weight:bold;
  font-size:1.2em;
}
#wrapper {
  width:600px;
  background-color:#def;
  border:1px solid #006;
}
</style>
<title>CSS Relative Positioning</title>
</head>
<body>
<div id="wrapper">
  <h1>CSS Relative Positioning</h1>
  <h2>From the Left and the Top</h2>
  <div id="explanation">
    <p>The h1 element on this page has been positioned relative to where it otherwise would be.</p>
    <p>All other content on the page (including these sentences) will show up in the same position it would have if the h1 had not been positioned at</p>
  </div>
</div>
</body>
</html>
```

The code above will render the following results.

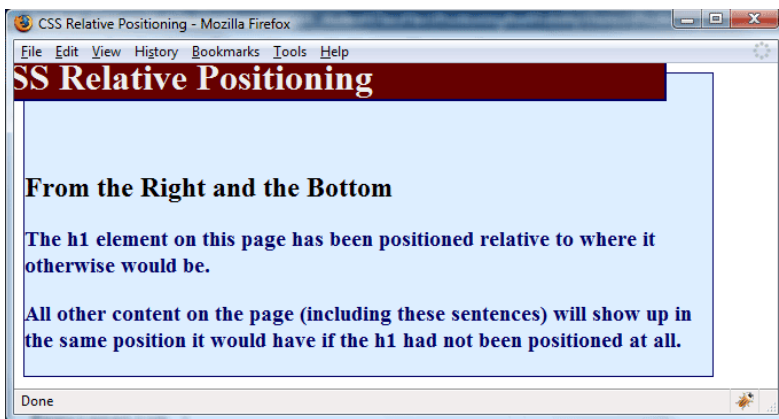


Notice how the paragraph text is positioned as if the h1 element had not been moved from its default position. This is because the h1 element has been relatively positioned.

Let's see what happens if we position the h1 element from the bottom right using relative positioning using the code below:

```
h1 {
  position:relative;
  bottom:40px;
  right:40px;
  border:2px solid #006;
  padding:1px;
  background-color:#600;
  color:#eee;
}
```

The code above will render the following results:



As the h1 element would normally be in the upper-left corner, it is pushed out of the browser window.

### Fixed Positioning

Elements with fixed positioning stay in the same position in the browser window even when the page is scrolled. It can be used to keep an element (e.g. a navigation menu) on the page at all times.

The example below uses fixed positioning to keep a home icon that links to the home page in the upper-left hand corner.

### Code Sample:

[PositioningAndVisibility/Demos/PositioningFixed.html](#)

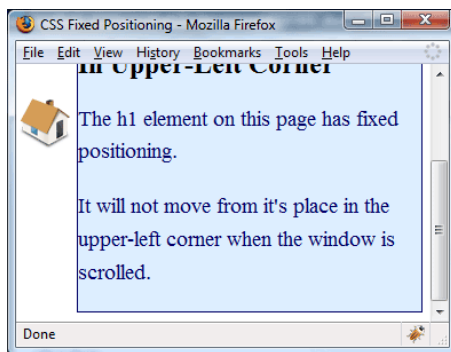
```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#homeLink {
  position:fixed;
  top:20px;
  left:0px;
}
---- C O D E   O M I T T E D ----
</style>
<title>CSS Fixed Positioning</title>
</head>
<body>
<div id="wrapper">
```

```

<a href="index.html" id="homeLink"></a>
<h1>CSS Fixed</h1>
<h2>In Upper-Left Corner</h2>
<div id="explanation">
  <p>The h1 element on this page has fixed positioning.</p>
  <p>It will not move from it's place in the upper-left corner when the window is scrolled.</p>
</div>
</body>
</html>

```

The two screenshots below show how this code is rendered before and after scrolling:



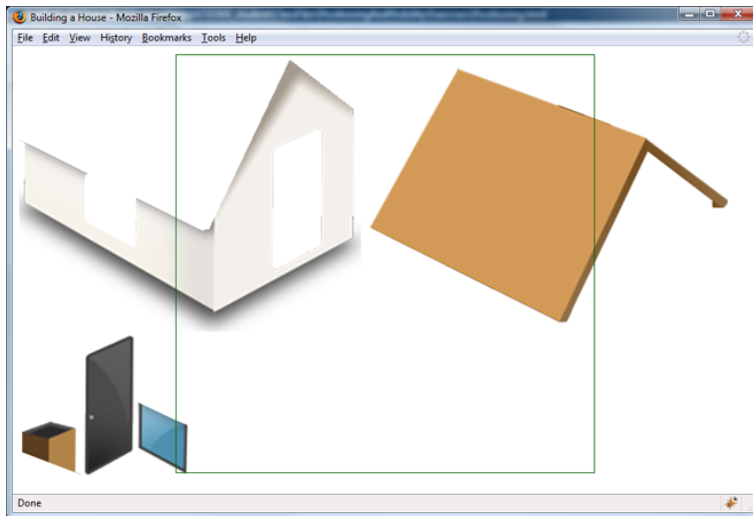
Notice that the home icon remains on the screen.

## Lesson 1, Activity 5: Positioning

Duration: 15 to 25 minutes.

In this exercise you will build a house from its individual parts using CSS positioning.

1. Open [PositioningAndVisibility/Exercises/Positioning.html](#) in your browser. It will look like this:



2. Your job is to build the house in the center of the green box. It will take some trial and error.
3. Open [PositioningAndVisibility/Exercises/Positioning.html](#) in your editor and begin working. The code is shown below:

### Code Sample:

[PositioningAndVisibility/Exercises/Positioning.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#plot {
  /* For centering the plot */
  /* Using relative positioning */
  position: relative;
  margin: 10px auto 0px auto;
  /* For centering the plot */
  /* Using absolute positioning */
  /*position: absolute;
  left: 50%;
  margin-left: -250px;*/
  width: 500px;
  height: 500px;
  border: 1px solid #060;
}
</style>
<title>Building a House</title>
</head>
<body>
<div id="wrapper">
  <div id="plot">
    
    
    
    
    
  </div>
</div>
</body>
</html>
```

### Challenge

1. If you used absolute positioning, try doing it again with relative positioning.
2. If you used relative positioning, try doing it again with absolute positioning.

### Challenge

1. Find [grass.jpg](#) in the [PositioningAndVisibility/Exercises/Images](#) folder.
2. Use it to put your house in a nice new yard.

### Solution:



PositioningAndVisibility/Solutions/PositioningAbsolute.html

```

---- C O D E   O M I T T E D ----
#main {
  position:absolute;
  top:150px;
  left:50px;
}

#roof {
  position:absolute;
  top:60px;
  left:34px;
}

#chimney {
  position:absolute;
  top:65px;
  left:261px;
}

#door {
  position:absolute;
  top:242px;
  left:353px;
}

#window {
  position:absolute;
  top:296px;
  left:129px;
}
---- C O D E   O M I T T E D ----

```

**Solution:**PositioningAndVisibility/Solutions/PositioningRelative.html

```

---- C O D E   O M I T T E D ----
#main {
  position:relative;
  top:150px;
  left:50px;
}

#roof {
  position:relative;
  top:-270px;
  left:34px;
}

#chimney {
  position:relative;
  top:-685px;
  left:261px;
}

#door {
  position:relative;
  top:-415px;
  left:275px;
}

#window {
  position:relative;
  top:-435px;
  left:-15px;
}
---- C O D E   O M I T T E D ----

```

## Lesson 1, Activity 6: Z-index

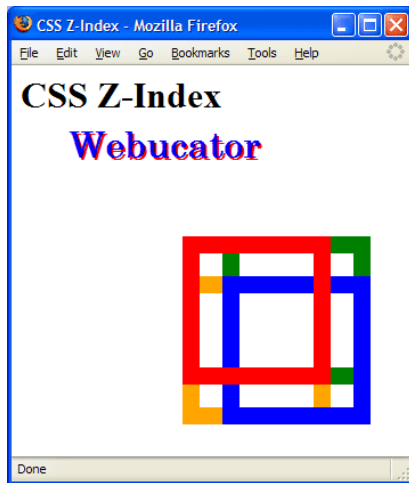
The `z-index` property specifies the stack level of an element on the page compared to other elements in its same flow. It takes a number as a value. The higher an element's `z-index`, the closer to the user it appears.

### Code Sample:

[PositioningAndVisibility/Demos/Zindex.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
.logo {
  position:absolute;
  font-family:Century;
  font-size:2em;
  font-weight:bold;
  z-index:20;
}
#logo1 {
  left:50px;
  top:50px;
  color:#00f;
}
#logo2 {
  left:52px;
  top:52px;
  color:#f00;
  z-index:10;
}
.box {
  position:absolute;
  height:100px;
  width:100px;
  border:15px solid #000;
}
#box1 {
  left:150px;
  top:150px;
  border-color:#f00;
  z-index:40;
}
#box2 {
  left:185px;
  top:185px;
  border-color:#00f;
  z-index:30;
}
#box3 {
  left:185px;
  top:150px;
  border-color:#060;
  z-index:20;
}
#box4 {
  left:150px;
  top:185px;
  border-color:#f60;
  z-index:10;
}
</style>
<title>CSS Z-Index</title>
</head>
<body>
<h1>CSS Z-Index</h1>
<div id="logo1" class="logo">Webucator</div>
<div id="logo2" class="logo">Webucator</div>
<div id="box1" class="box"></div>
<div id="box2" class="box"></div>
<div id="box3" class="box"></div>
<div id="box4" class="box"></div>
</body>
</html>
```

The above code will render the following results:



Notice how some `div` elements sit on top of the others. If you switch their `z-index` values, they will stack the other way.

## Lesson 1, Activity 7: Visibility

The `visibility` property is used to make an element invisible. Possible values are:

- `visible`
- `hidden`

The major difference between setting an element's `visibility` to `hidden` and setting its `display` to `none` is that an `hidden` element still affects the layout of the page; whereas an element that has a `display` of `none` does not.

### Display

The `display` property is used to determine if and how an element appears. The most common values are shown below.

- `block`
- `inline`
- `none`

The most common uses of `display` are:

1. To show and hide elements based on user interaction. A common example is a drop-down menu. This dynamic change of style is done with JavaScript.
2. To hide elements for certain media. For example, the images might be "turned off" by setting the `display` to `none` in a style sheet for print.
3. Converting an inline element such as a link to a block element by setting its `display` property to `block`.

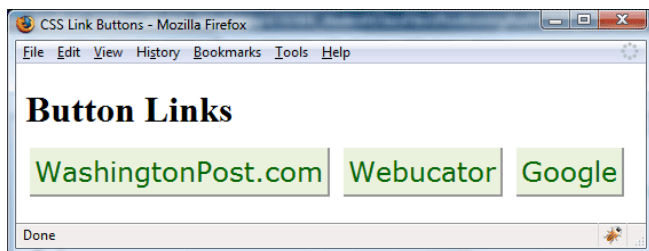
The following example shows how the `display` property can be used to turn links into block elements..

#### Code Sample:

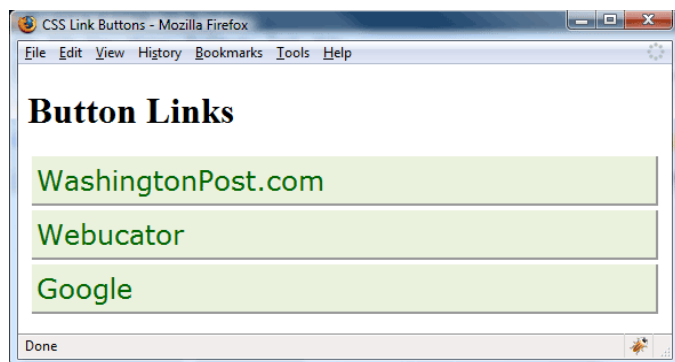
[PositioningAndVisibility/Demos/LinkButtons.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>CSS Link Buttons</title>
<style type="text/css">
a {
display:block;
padding: 6px 4px;
margin: 4px;
border-right: 2px solid #999999;
border-bottom: 2px solid #999999;
border-top-width: 0px;
border-left-width: 0px;
background-color: #eaf1dd;
color:#060;
text-decoration:none;
font-family:Verdana, Geneva, sans-serif;
font-size:1.5em;
}
</style>
</head>
<body>
<h1>Button Links</h1>
<div>
<a href="http://www.washingtonpost.com">WashingtonPost.com</a>
<a href="http://www.webucator.com">Webucator</a>
<a href="http://www.google.com">Google</a>
</div>
</body>
</html>
```

If we didn't set `display` to `block`, the page would look like this:



But with `display` set to `block`, it looks like this:

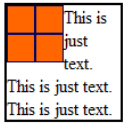


## Lesson 1, Activity 9: Float

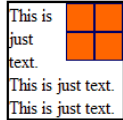
Float can be applied to any element that is not absolutely positioned. It is used to specify whether an element should float to the left, to the right, or not at all. Possible values are listed below.

- left
- right
- none

When an element floats left it will be aligned to the left of the containing element and all subsequent content will align to its right until the bottom of the element is reached:



When an element floats right it will be aligned to the right of the containing element and all subsequent content will align to its left until the bottom of the element is reached:



The code used to create the results above is shown below:

### Code Sample:

[PositioningAndVisibility/Demos/Float.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>CSS Float</title>
<style type="text/css">
  .container
  {
    width: 150px;
    border: 2px solid #000;
  }
  .floatLeft
  {
    float: left;
  }
  .floatRight
  {
    float: right;
  }
</style>
</head>
<body>
<h2>Float Left</h2>
<div class="container">

<p>This is just text. This is just text. This is just text.
  This is just text. This is just text.</p>
</div>
<h2>Float Right</h2>
<div class="container">

<p>This is just text. This is just text. This is just text.
  This is just text. This is just text.</p>
</div>
</body>
</html>
```

## Lesson 1, Activity 10: Clear

The `clear` property is used to specify whether content that is flowing to the side of a floating block should drop down beneath that block. Possible values are shown below.

- `left`
- `right`
- `both`
- `none`

The following files demonstrate how `clear` and `float` can be used in combination with positioning to produce virtually any blocked page layout.

1. [PositioningAndVisibility/Demos/FloatPosition1a.html](#)
2. [PositioningAndVisibility/Demos/FloatPosition1b.html](#)
3. [PositioningAndVisibility/Demos/FloatPosition2a.html](#)
4. [PositioningAndVisibility/Demos/FloatPosition2b.html](#)

### Code Sample:

#### PositioningAndVisibility/Demos/FloatPosition1a.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#wrapper {
  position:absolute;
  left:50%;
  top:30px;
  width: 450px;
  margin-left:-225px;
}

.box {
  border: 2px solid #f00;
  background-color:#000066;
  color:#f90;
  text-align:center;
  font-size:100px;
  font-weight:bold;
  font-family:Arial, Helvetica, sans-serif;
}

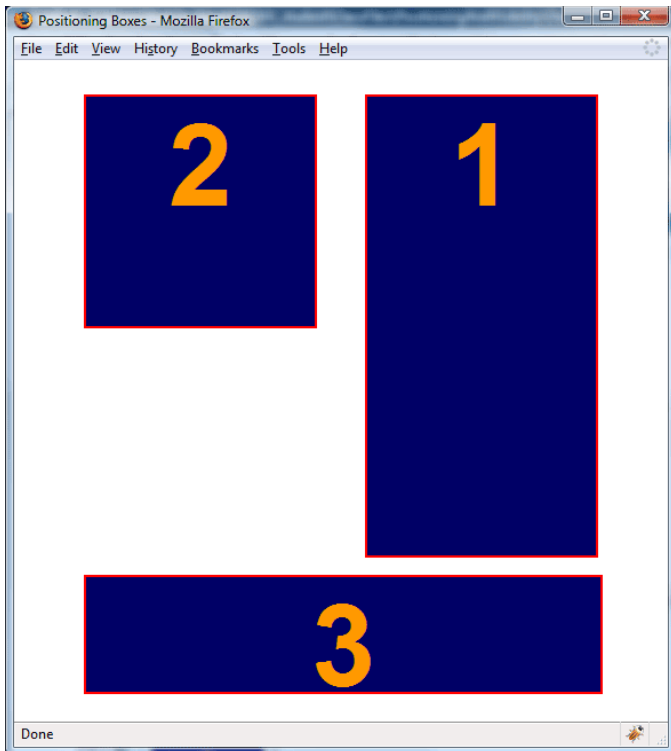
#box2 {
  height:200px;
  width:200px;
  margin-bottom:15px;
}

#box1 {
  float:right;
  height:400px;
  width:200px;
  margin-bottom:15px;
}

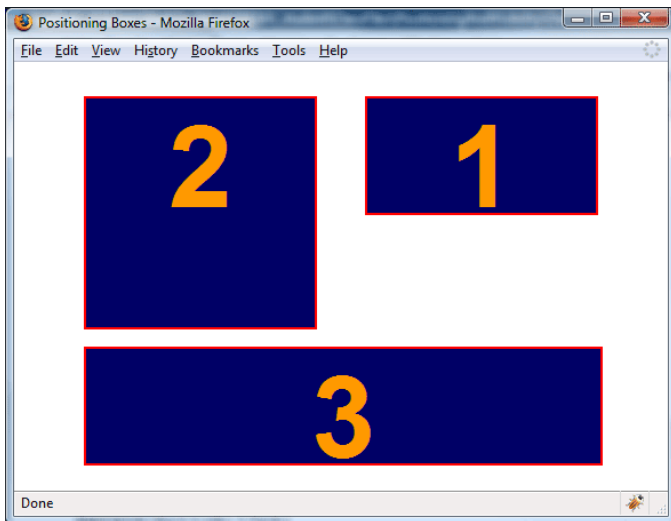
#box3 {
  height:100px;
  width:100%;
  clear:both;
}
</style>
<title>Positioning Boxes</title>
</head>

<body>
<div id="wrapper">
  <div id="box1" class="box">1</div>
  <div id="box2" class="box">2</div>
  <div id="box3" class="box">3</div>
</div>
</body>
</html>
```

The above code will render the following:



Box 3 will always remain below the taller of the two other boxes as we can see by changing the height of box 1 to 100px:



Now let's get a header in there and add a column:

#### Code Sample:

[PositioningAndVisibility/Demos/FloatPosition2a.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#wrapper {
  width:800px;
}

#insideWrapper {
  width:612px;
  position:relative;
  left:50%;
  margin-left:-311px;
}

.box {
  border: 2px solid #f60;
```



```

background-color:#006;
color:#f90;
text-align:center;
font-size:100px;
font-weight:bold;
font-family:Arial, Helvetica, sans-serif;
}

#box1 {
float:left;
height:300px;
width:200px;
}

#box2 {
float:left;
height:400px;
width:200px;
}

#box3 {
float:left;
height:300px;
width:200px;
}

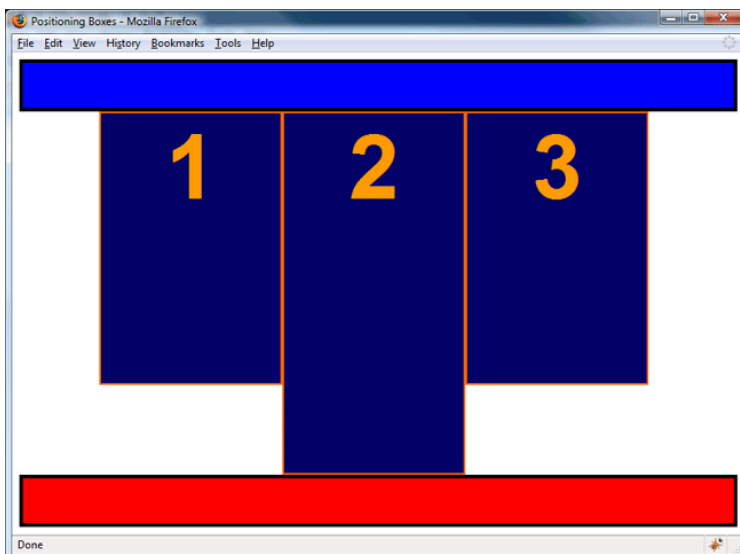
#divClear {
height:50px;
clear:both;
border:4px solid #000;
background-color:#f00;
}

#divTop {
height:50px;
border:4px solid #000;
background-color:#00f;
}
</style>
<title>Positioning Boxes</title>
</head>

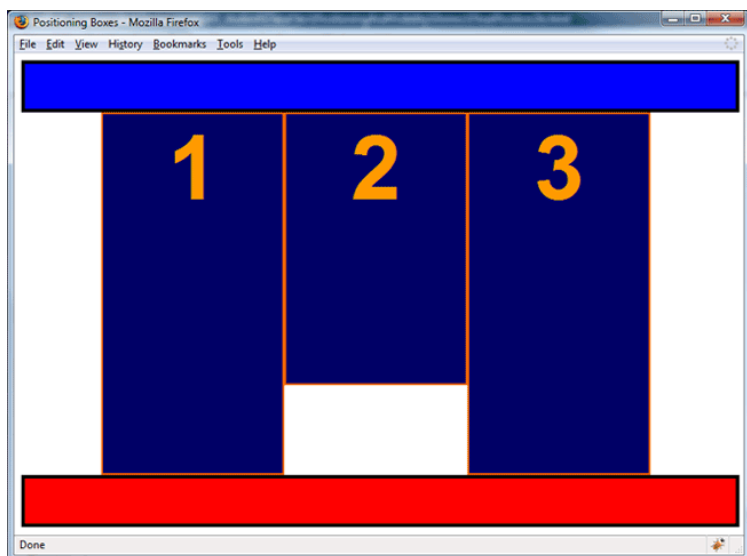
<body>
<div id="wrapper">
<div id="divTop" style=""></div>
<div id="insideWrapper">
<div id="box1" class="box">1</div>
<div id="box2" class="box">2</div>
<div id="box3" class="box">3</div>
</div>
<div id="divClear" style="background:red"></div>
</div>
</body>
</html>

```

The above code will render the following:



Again, the footer will always remain below the taller of the other columns as we can see by changing around the height of the boxes::



Lesson 1, Activity 12: Static vs. Relative vs. Absolute