

Different layout properties available in CSS: `position:static`, `position:relative`, `position:absolute`, and `float`.

1. `position:static`

The default positioning for all elements is *position:static*, which means the element is not positioned and occurs where it normally would in the document.

Normally you wouldn't specify this unless you needed to override a positioning that had been previously set.

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

2. `position:relative`

If you specify *position:relative*, then you can use *top* or *bottom*, and *left* or *right* to move the element relative to where it would normally occur in the document.

Let's move `div-1` down 20 pixels, and to the left 40 pixels:

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

3. `position:absolute`

When you specify *position:absolute*, the element is removed from the document and placed exactly where you tell it to go.

Let's move `div-1a` to the top right of the page:

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

4. `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;
}
```

6. two column absolute height

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;
}
```

7. float

For variable height columns, absolute positioning does not work, so let's come up with another solution.

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).

```
#div-1a {
  float: left;
  width: 200px;
}
```

8. float columns

If we float one column to the left, then also float the second column to the left, they will push up against each other.

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

9. float columns with clear

Then after the floating elements 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;  
}
```