# CSS Positioning

You can use the `position` CSS property to define where an element is set in the documenbt. Using `position` in conjunction with the `top`, `right`, `bottom`, and `left` properties you can determine the final location of positioned elements. Positioning allows you to take elements out of the normal document layout flow, and make them behave differently, for example sitting on top of one another, or always remaining in the same place inside the browser viewport. This article explains the different position values, and how to use them.

The CSS `position` property has a few different values:

- `relative`: The `top` and `bottom` properties specify the vertical offset from the element's normal position. The `left` and `right` properties specify the horizontal offset.
- `absolute`: The `top`, `right`, `bottom`, and `left` properties specify offsets from the edges of the element's containing block.
- `fixed`: Same as `absolute` only the containing block is the initial containing block (i.e. the `<body>`)
- `static`: This is the normal/default `position`. `top`, `right`, `bottom`, and `left` have no effect on the element.

## Relative Positioning

Consider that a relatively positioned element is positioned according to the normal flow of the document, and then offset "relative" to where it normally would be placed according to the values set for `top`, `right`, `bottom`, and `left`. The following code will illustrate a relatively positioned element:

```html
<div class="box" id="one">One</div>
<div class="box relative" id="two">Two</div>
<div class="box" id="three">Three</div>
<div class="box" id="four">Four</div>
```

```css
.relative {
    position: relative;
}

.box {
    display: inline-block;
    width: 100px;
    height: 100px;
    background: red;
    color: white;
}

#two {
    top: 20px;
    left: 20px;
    background: blue;
}
```

Even without running the code above, it is fairly easy to see that the `<div>` with `class="box relative" id="two"` will be positioned "relative" to where it would normally be positioned. It will be placed `20px` from the top of its normal position, and `20px` from the left of its normal position.

## Absolute Positioning

While `relative` positioned elements remain in the normal document flow, `absolute` positioned elements do not. What this means is that other elements that would normally surround the `absolute` element now act as though the `absolute` element does not exist. The `absolute` element is positioned relative to its nearest ancestor. The following code will illustrate an absolutely positioned element:

```html
<div class="relative">
    <div class="box" id="one">One</div>
    <div class="box absolute" id="two">Two</div>
    <div class="box" id="three">Three</div>
    <div class="box" id="four">Four</div>
</div>
```

```
.relative {
    position: relative;
    top: 50em;
}

.absolute {
    position: absolute;
}

.box {
    display: inline-block;
    width: 100px;
    height: 100px;
    background: red;
    color: white;
}

#two {
    top: 20px;
    left: 20px;
    background: blue;
}
```

Now the `<div>` with `class="box absolute" id="two"` will be plucked from the normal flow of elements and positioned relative to its containing block, which in this case is the `<div>` with `class="relative"`. Note in the css we are setting all elements with class `relative` to `position: relative`. Without that the browser would continue up the ancestor tree to find an element that is `position: relaitve`. The `<body>` is the topmost ancestor in the tree, and is `position: relative`. This means that if you do not set any relative containers in your HTML, all elements with `position: absolute` will be positioned relative to the `<body>`.

## Fixed Positioning

Elements with `position: fixed` are very much like elements with `position: absolute`. The key difference is that `fixed` elements are positioned relative to the initial containing block, which in most cases is the `<body>` element.

```
<div class="relative">
    <div class="box" id="one">One</div>
    <div class="box fixed" id="two">Two</div>
    <div class="box" id="three">Three</div>
    <div class="box" id="four">Four</div>
</div>
```

```
.relative {
    position: relative;
    top: 50em;
}

.fixed {
    position: fixed;
}

.box {
    display: inline-block;
    width: 100px;
    height: 100px;
    background: red;
    color: white;
}

#two {
    top: 20px;
    left: 20px;
    background: blue;
}
```

The code above will look different from the code for the `position: absolute` element. If `#two` is `position: absolute` it will still be placed relative to the `div.relative`. In the above example the `#two` element is `position: fixed`, and it will be placed relative to the `<body>`.

## Z-index

The `z-index` CSS property specifies the z-order of a positioned element. If you think of the document as a coordinate plane, you use `top` and `bottom` to position elements along the y-axis while `left` and `right` are used to position elements along the x-axis. `z-index` is used to position elements along the z-axis.

For any element that is positioned (meaning it is not `position: static`), the `z-index` property specifies the stack level of the element. A higher `z-index` value will mean the element appears in front of elements with lower `z-index` values. The `z-index` value is just a number (positive or negative) or `auto`. `auto` denotes that the element does not establish a new local stacking context.

Consider the following code:

```html
<div class="dashed-box">Dashed box
    <span class="gold-box">Gold box</span>
    <span class="green-box">Green box</span>
</div>
```

```css
.dashed-box {
    position: relative;
    z-index: 1;
    border: dashed;
    height: 8em;
    margin-bottom: 1em;
    margin-top: 2em;
}

.gold-box {
    position: absolute;
    z-index: 3; /* put .gold-box above .green-box and .dashed-box */
    background: gold;
    width: 80%;
    left: 60px;
    top: 3em;
}

.green-box {
    position: absolute;
    z-index: 2; /* put .green-box above .dashed-box */
    background: lightgreen;
    width: 20%;
    left: 65%;
    top: -25px;
    height: 7em;
    opacity: 0.9;
}
```

In the code above, the "gold box" will end up on top, followed by the "green box" and then the "dashed box".