Positioning.

Objectives:

By the end of this chapter, you should be able to:

- Compare and contrast `static`, `relative`, `absolute`, and `fixed` positioning
- List the properties that positioning gives an element
- Use positioning to build more complex layouts

Positioning

So far in this unit we've examined the `display` and `float` properties, and looked at some examples of how to use these properties to lay out a page. In this chapter, we'll explore another important property for layouts: `position`.

Positioning allows you to take an element on the page and control where and how it's positioned relative to things such as its original starting position, other elements, or even the window itself. Depending on the type of positioning you use, the element will either remain in the document flow (relative) or be removed from the document flow (absolute, fixed), just like with floats.

Once `position` is set, offset values can be set for `top`, `right`, `bottom`, `left`, and `z-index` (a 3D axis that we can use to stack elements on top of each other).

By default, the position of an element is set to `static`. `static` positioning and `relative` positioning are basically the same, but with one important difference: a statically positioned element won't respond to the offset properties listed above. If you set `top: 10px` on an element that is statically positioned, this style rule will simply be ignored. This is analogous to how `inline` elements don't respond to `height` or `width`; elements that are positioned statically (which is the default positioning for elements) will not respond to `top`, `left`, `right`, `bottom`, or `z-index`.

Here's a quick example. The HTML page below has a single `div`, which is relatively positioned and has `top` and `left` values set. See what happens when you remove the `position: relative;` line. With this line removed, the `div` will revert to its default positioning (`static`), and it should ignore the positioning set by `top` and `left`:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
```

```
      width: 300px;
      height: 300px;
      background: orange;
      position: relative;
      /* move divs 100 pixels from the top, relative to where they'd normally
sit */
      top: 100px;
      /* move divs 200 pixels from the left, relative to where they'd
normally sit */
      left: 200px;
    }
  </style>
</head>
<body>
  <div>hello.</div>
</body>
</html>
```

Absolute Positioning vs. Relative Positioning

With `relative` positioning, elements are not removed from the document flow, and any offsets you place on the element will be *relative* to its default position in the document flow. In the above example, for instance, `top` is set to `100px`; this means that the `div` will be 100 pixels below where it would otherwise be (in other words, you're offsetting the `div` 100 pixels from the top).

With `absolute` positioning, the situation is a little different. In this case, the element is removed from the document flow, and any offsets you place on the element will be relative to its parent, *provided its parent is not statically positioned!* If the parent element *is* statically positioned, then the offsets will be relative to the grandparent, provided the grandparent is not statically positioned. If the grandparent is statically positioned, we keep going up the chain until we find an element that is not statically positioned. If no such element exists, the offsets are relative to the `body`.

Here is an example with relative and absolute positioning:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
  body {
    margin: 0; /*REMOVE BROWSER MARGIN*/
  }

  .red {
```

```
    background-color: red;
    position:  absolute;
    width: 200px;
    height: 200px;
    bottom: 0;
    right: 0;
  }
  .wrapper {
    width: 500px;
    height: 400px;
    background: #e3e3e3;
    position: relative;
    margin-bottom: 10px;
  }
  </style>
</head>
<body>
  <div class="wrapper">
    <div class="red"></div>
  </div>
</body>
</html>
```

When you open up this page, you should see that the red `div` is in the bottom-right corner of its parent. This makes sense: the parent has relative positioning, and the red div is set to be offset `0` pixels from the bottom and the right.

Now, what happens if you remove the `position: relative` rule from the `wrapper` class? In this case, the parent of the red `div` will have static positioning, which means that the position of the red `div` will be relative to the body! Refresh the page and you should see that the `red` div has moved to the bottom-right corner of the `body`, not of its parent!

Fixed positioning

The fourth type of positioning is `fixed`. This behaves similar to absolute positioning, but elements with fixed positioning are **ALWAYS** positioned relative to the active viewport. Now, what's that mean? If you position, for example, a fixed element with a top offset of 50 pixels and a right offset of 100 pixels, the top right corner of the element will be positioned 50 pixels over to the left and 100 pixels down. Unlike with absolute positioning, scrolling the page content would not affect this element's position at all. It will remain in that position, no matter how the screen was resized or scrolled. It truly is fixed.

Let's modify our previous example to see what fixed positioning looks like. We'll need to add some extra `div`s to the page so that it's long enough to scroll through. Let's also play around with the `z-index` a bit:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
  body {
    margin: 0; /*REMOVE BROWSER MARGIN*/
  }
  .red {
    background-color: red;
    position: fixed;
    width: 200px;
    height: 200px;
    bottom: 0;
    left: 0;
    z-index: 1;
  }
  .wrapper {
    width: 500px;
    height: 400px;
    background: #e3e3e3;
    position: relative;
    margin-bottom: 10px;
  }
  #front {
    z-index: 2;
  }
  </style>
</head>
<body>
  <div class="wrapper">
    <div class="red"></div>
  </div>
  <div class="wrapper"></div>
  <div class="wrapper"></div>
  <div class="wrapper"></div>
  <div class="wrapper" id="front"></div>
  <div class="wrapper"></div>
</body>
</html>
```

In this example, notice that the red `div` is *always* in the bottom-left corner of the screen, no matter where on the page you scroll. If you change the style so that the red `div` has absolute positioning, you'll see the difference.

Also note that with fixed positioning, the red `div` is in front of all the wrapper `div`s except for the one with an id of `front`. This is because of the `z-index`! The larger the `z-index`, the higher the element should be stacked. If two divs are positioned on top of one another, the one with the lower `z-index` will appear to be behind the one with the higher `z-index`.