Mastering CSS with Mark Lassoff

Section 6: Understanding CSS Positioning

CSS Positioning needlessly confuses a lot of developers, but once you get the hang of it, it can be reasonably straightforward.

HTML

Here is a simple HTML document with two `div`. Both boxes are 200px width by 200px height. One has an orange background, and the other has a blue background.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CSS Positioning</title>
    <style>
        .orange-box {
            height: 200px;
            width: 200px;
            background: orange;
        }
        .blue-box {
            height: 200px;
            width: 200px;
            background: blue;
            position: static;
        }
    </style>
</head>
<body>
    <div class="orange-box"></div>
    <div class="blue-box"></div>
</body>
</html>
```

Static Positioning

You'll notice that the `blue-box` has `position` of static. `Static` is the default value for the CSS `position`

rule. When you use static positioning, each element is lined up one on top of each other in the order in which they appear in the HTML.

Relative Positioning

We'll start with changing the CSS for `orange-box` to `position: relative`. This will make its position relative to its normal position in the page element stack. By itself, setting the `position` rule to `relative` doesn't do anything. However, we can see the effect of `relative` positioning by writing some helper rules.

Please note: with Relative Positioning, the space that would have ordinarily occupied is still being held open. Other elements will not move to fill in the space created.

```
.orange-box {
    height: 200px;
    width: 200px;
    background: orange;

    position: relative;
    top: 200px;
    left: 200px;
}
```

In the code above we've added the following helper rules:

- `top: 200px` will move the element down 200px from it's static position.
- `left: 200px` will move the element left 200px from it's static position.

We can also try changing the position values of the orange box to a 100px each. With this change, the orange box will overlap the blue box.

```
.orange-box {
    height: 200px;
    width: 200px;
    background: orange;

    position: relative;
    top: 100px;
    left: 100px;
}
```

z-index

Right now the orange is overlapping the blue box. If you want the blue box on top, you can change the `z-index` rule.

By default, all elements get stacked with new elements in front of older elements. By giving the orange box

`z-index: -1` or the blue box `z-index: 2` the blue box will appear on top.

Absolute Positioning

Absolute positioning allows you to select an exact position for the element within the browser window. Unlike relative positioning, the element will be taken out of the element flow, and other elements will fill the empty space created.

Let's try this:

```
.orange-box {
    height: 200px;
    width: 200px;
    background: orange;

    position: absolute;
    bottom: 0px;
    right: 0px;
}
```

We'll use these rules to select an *absolute* position for our element:

- `bottom: 0px` - It'll be 0px away from the bottom edge.
- `right: 0px` - It'll be 0px away from the right edge.

These rules will put the orange box in the lower right-hand corner of the browser window. If we add a large amount of content, the orange box will overlay the content, no matter how much content we add.

We can add a few paragraphs of Lorem Ipsum text to show this.

Dev Tools Tips:

> Lorem Ipsum text is used as a placeholder. It's prevalent in publishing and graphic design for putting in place of content that hasn't been written yet. You can collect your own at lipsum.com.

Visit lipsum.com and collect enough text to make the screen scrollable. You'll now notice that the orange box will remain in the same position based on the CSS rules as we scroll the page.

Absolute Positioning based on another element

Absolute positioning adjusts the positioning based the closest positioned ancestor. So far, we've been positioning based on the body element. To demonstrate nested positioning, we're going to add a purple box on top of the orange box.

First a small change to the HTML:

```
<div class="orange-box">
    <div class="purple-box"></div>
</div>
```

Using this CSS to place the purple-box as an absolute positioned element, it'll remain on top of the orange box no matter where you put the orange box on the page.

```
.purple-box {
    height: 20px;
    width: 20px;
    background: purple;
    position: absolute;
    left: 10px;
    top: 10px;
}
```

Try moving the orange box around by changing values in the CSS. What happens when you give the purple box a negative number for the top?

Fixed Positioning

Fixed positioning is very much like absolute positioning with one crucial difference: Elements assigned fixed positioning will not scroll with the rest of the content on the page. Like absolute positioning, the element will be removed from the flow, and the next element will take up its space.

```
.orange-box {
    height: 200px;
    width: 200px;
    background: orange;

    position: fixed;
    bottom: 0px;
    right: 0px;
}
```

- `bottom: 0px` - The element will be placed 0px away from the bottom edge.
- `right: 0px` - The element will be placed 0px away from the right edge.

Now if we scroll, you'll notice that unlike the absolute positioned element, the fixed doesn't scroll with the content. Fixed positioning is often used when the design requires an item always be visible on the screen-- Like a footer

Submit this

1. Make a face.
   Using the following HTML, create a face design.

```
<div id="face">
    <div id="rightEye"></div>
    <div id="leftEye"></div>
    <div id="nose"></div>
    <div id="month"></div>
</div>
```

Use any and all CSS rules you need to create the face out of the HTML provided.

Added points for using border-radius for a more rounded face. :)

2. Add a sticky footer: Create a footer that will stick to the bottom of the page that has your contact information inside it.