Week 4_3(en)

Position and Z-index

Position

The default position of an element can be changed more freely by setting its `position` property.
The `position` property can take one of these four values:

- `static` - the default value
- `relative`
- `absolute`
- `fixed`

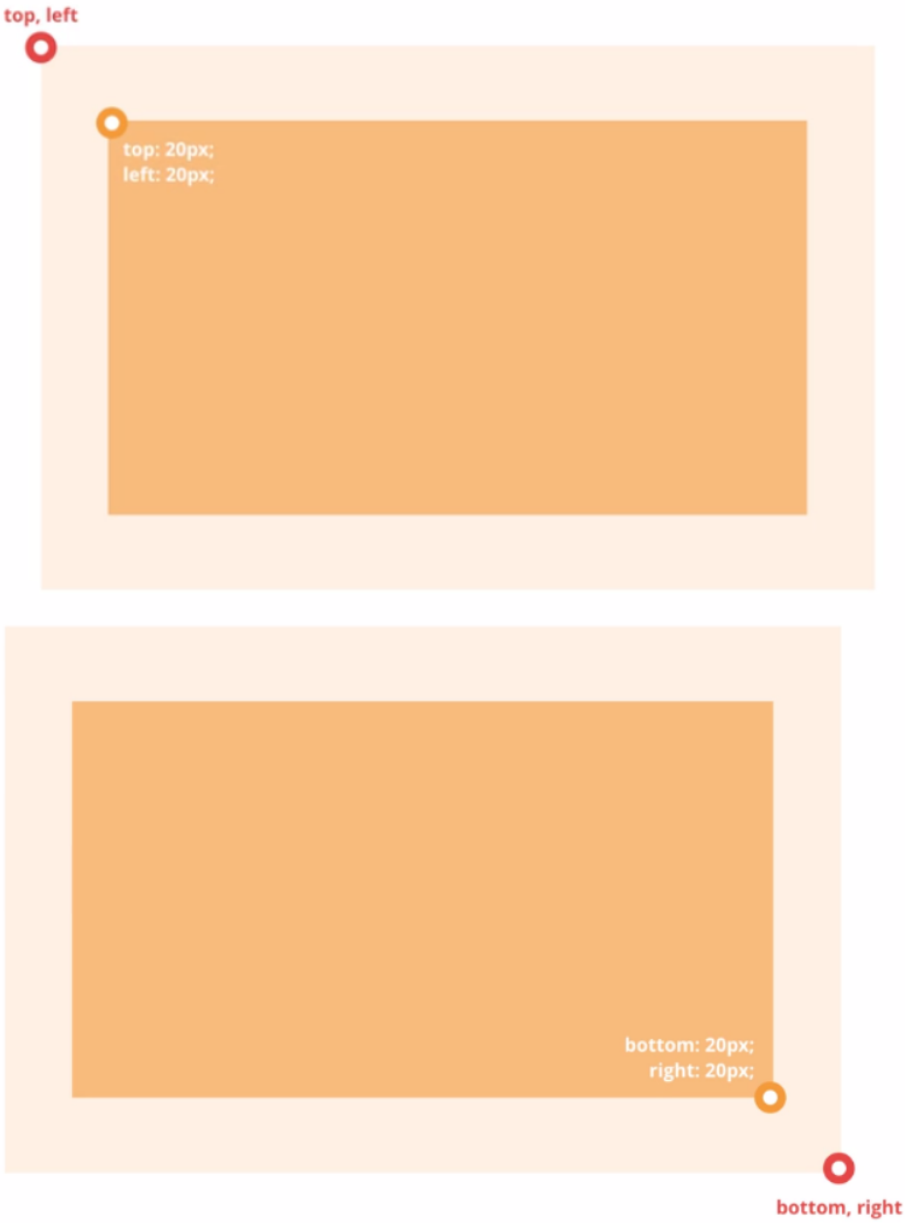
`position: relative;`

This value allows you to position an element *relative* to its default static position on the web page.

Offset Properties

After setting the `position` property, we need to set the offset properties to actually make the element move.
Otherwise, changing the `position` value will not yield any visible results.
The valid offset properties are:

- `top` - moves the element down
- `bottom` - moves the element up
- `left` - moves the element right
- `right` - moves the element left



Only one of `top` and `bottom` should be set, and only one of `left` and `right` should be set.

Let's try adding the following declarations to our `.sidebar` selector.

```
display: relative;
top: 30px;
left: 50px;
```

You can see that the sidebar has moved accordingly.

`position: absolute;`

When an element's position is set to `absolute`, all other elements on the page will *ignore the element* and act like it is not present on the page.
The element will be positioned relative to its closest *positioned* parent element.

> **Note:** A *positioned* element is one whose position is anything except `static`.

Let's add `position: relative;` to our `.content-area` declaration block, and add the following Let's try adding the following CSS code to our `sidebar` declaration block.

```
  position: absolute;
  top: 20px;
  left: 50px;
```

The sidebar will move relative to its closest *positioned* parent element.
Let's add `position: relative;` to our `.content-area` declaration block, so that the sidebar will move relative to its direct parent. (Recall that adding `position: relative;` without any offset specifications does not change anything that we view.)
You will notice that the rest of the elements ignore the sidebar, so the `main-area` is floated to the left side of the `content-area`.

`position: fixed;`

We can *fix* an element to a specific position on the page by setting its position to `fixed`.
Unlike `absolute`, which makes the element **scroll with the rest of the document** when a user scrolls, the element will be fixed in the **same area regardless of user scrolling**.

Let's try adding a search bar that always shows on the top right corner of our page, regardless of scrolling.
We will need to use the `<form>` tag that we learned few weeks ago.
At the beginning of the `<body>` element, add the following code.

```
        <div class="search-bar">
            <form>
                <input type="text">
                <input type=button value="search">
            </form>
        </div>
```

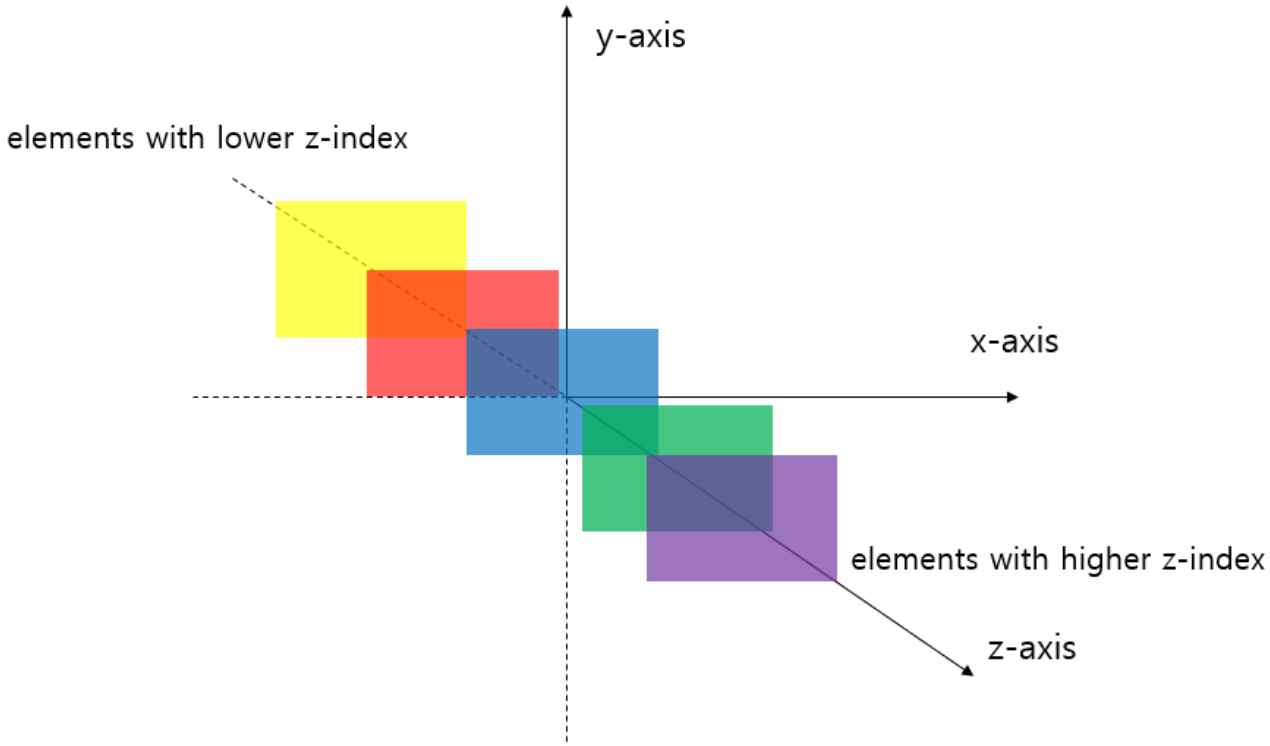
This will make a form with a text input and a button with the text "search" on it.
Now let's add the following CSS code.

```
.search-bar {
  position: fixed;
  right: 20px;
}
```

We fixed the search bar to the top right corner.
If you check the browser, you will notice that as you scroll through the page, the search bar will be above the rest of the content.
This is because when we changed the position of the header to `fixed`, we removed it from the flow of the html document.

z-index

When boxes on a web page have a combination of different positions, the boxes can **overlap** with each other, making the content difficult to read.
The `z-index` property controls how far "back" or how far "forward" an element should appear on the web page when elements overlap.
You can think of the page as a 2-dimensional space with x and y indexes, but if you add a new z axis(that penetrates the page), you think of the browser as a 3-dimensional space.



The `z-index` property accepts integer values.
The order of the integer guides the browser on the order in which elements should be displayed on the web page.
The greater the `z-index` is, the more forward the element comes out.

> **Note:** `z-index` only works for positioned elements (does *not* work for static elements).

For example, we can try changing the `z-index` of the `search-bar` to 1 and the `main-area` to 2 and view the difference.
Since `main-area` is not positioned, you will have to add `position: relative;` to `main-area`.

Afterwards, let's change the `z-index` of the `search-bar` to 3 in order to see the search-bar on top of the `main-area`.