Week 4_1(en)

Float and Clear

> Last time we learned about the CSS box model. This time, let's learn ways to modify the positions of HTML elements on a page in more diverse ways!

Before starting, please create a directory named `date-your name` in your computer or your USB.
Then copy & paste or download below files.
index.html
seoul.jpg

Additionally, make a file named `blog.css` in your directory.

HTML Flow

As we've already seen while dealing with pure HTML code, an HTML document already has its own rules for *displaying and positioning* elements even without any CSS applied.

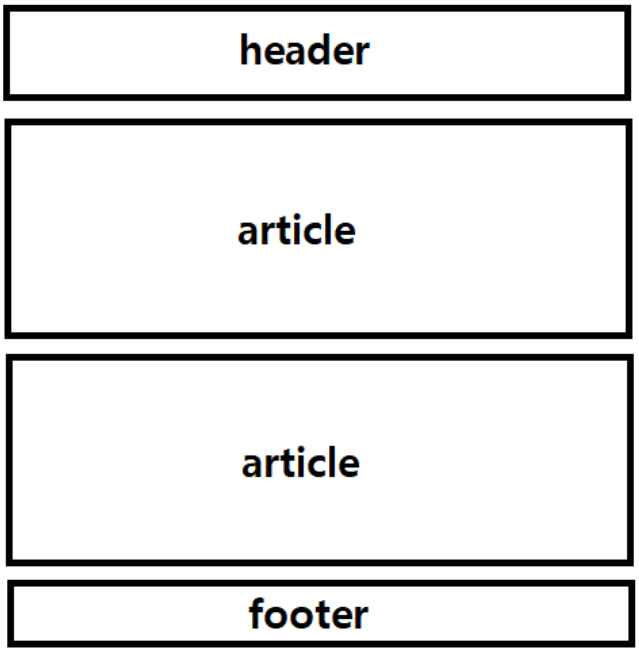
- **fluidity**: how the content adapts to browser dimensions
- **ordering**: in which order elements appear
- **stacking**: how elements appear on top of each other

This default behavior is called the *flow* of elements in HTML.

We can use several CSS properties like `height`, `width`, `float`, `position`, `z-index`, `display`, and `clear` to **disrupt** the flow.

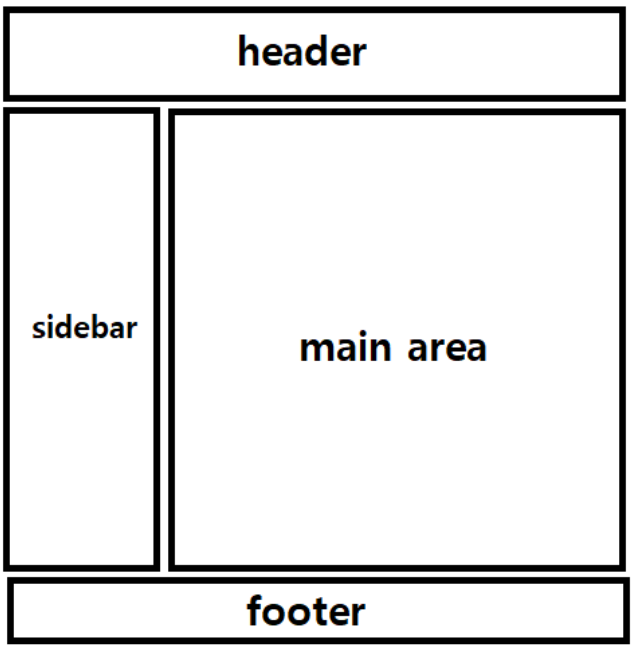
Float

For now, the overall structure of our page looks like this.



All of the elements sit below and above one another, each taking up the entire horizontal space.

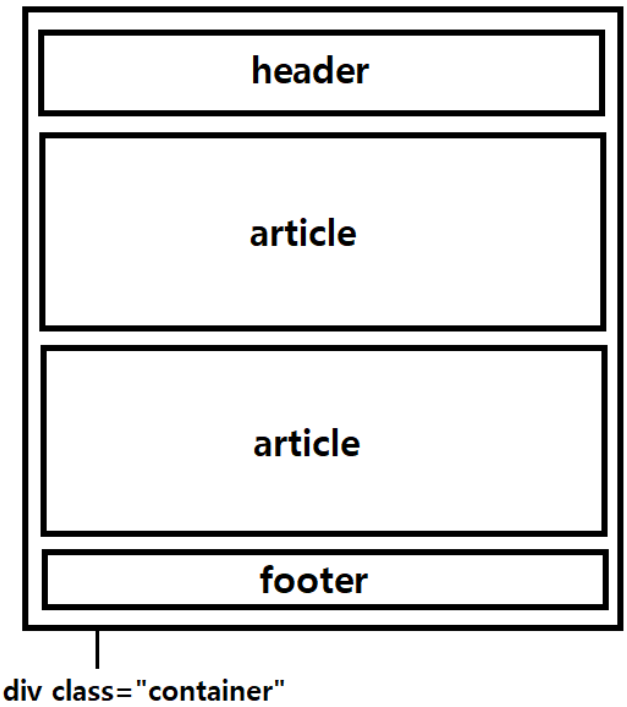
Let's try adding a **sidebar** which enables navigating to other category pages like below.



Here, the sidebar and the main area sit side by side one another.

Before creating a sidebar, let's make the content of the whole page have a maximum width.
This is important because it is hard for the human eye to read text when it is spread throughout the horizontal space of the entire browser window.
To do this, let's create a `<div>` with `class="container"` and put the whole content in the `<body>` element

inside the `<div>`.



Then, let's add the following CSS code.

```
.container {
  max-width: 800px;
  margin-left: auto;
  margin-right: auto;
}
```

If you set the `max-width` property for an element, the element will have a maximum width of 800 pixels, and no bigger.
If you set `margin-left` and `margin-right` properties to `auto`, the content will be centered horizontally.

`width` vs. `max-width`

As mentioned previously, block-level elements by default take up the full width available.
To change this(*break the flow*), we can use the `width` or `max-width` property.

If you use `width` (with pixels), the element will have a fixed width that *does not change* according to the browser size.
However, if you use `max-width`, the element will be *resized* when the browser width is smaller than the `max-width` value.
`max-width` is useful when making a site usable on small devices.
Try resizing the browser window to less than 800 pixels wide, and see how the container resizes its width!

> **Note**: If you set only the `max-width`, the width of the child elements need to be specified by %, not by pixels. This is because setting them by pixels might lead to overflows in small devices.

Now let's add a element for sidebar above the first `<article>` element.
We will use the `<nav>` element since the sidebar will be a navigation bar to navigate through different categories of our blog.

```
        <nav class="sidebar">
             <ul>
                <li><a href="cities.html">Cities</a></li>
                <li><a href="movies.html">Movies</a></li>
                <li><a href="books.html">Books</a></li>
            </ul>
        </nav>
```

We don't have any of the html files indicated as `href`, so it will not function when we click on each of the `<a>` elements, but do not worry about this!
We just want to make a sidebar appear on the page for now.

Then, let's make a `<div>` element with `class="main-area"` and put in the two `<article>` elements inside it.
Now, your sidebar appears above the main area.
In order to make our sidebar and main area appear side by side, we need to use the `float` property.

The `float` property moves(or *floats*) the element as far left or as far right as possible.
It can be set to one of two values, `left` or `right`.

Let's add the following CSS code.

```
.sidebar {
  width: 20%;
  float: left;
  background-color: #466995;
}

.main-area {
  width: 80%;
  float: left;
  background-color: #DBE9EE;
}
```

Notice that the `background-color` is added to clearly see the amount of area each element takes.
If you check the browser, you will notice that we got the aside and main-area sit side by side.

> **Note**: Floated elements must have a width specified. Otherwise, the element will assume the full width of its containing element, and changing the float value will not yield any visible results.

Clear

When multiple floated elements on one page have different heights, it can affect their layout on the page.
The clear property specifies on which sides of an element floating elements are not allowed to float.
It can have one of the following values:

- `none` - Allows floating elements on both sides (default)

- `left` - No floating elements allowed on the left side
- `right` - No floating elements allowed on the right side
- `both` - No floating elements allowed on either the left or the right side
- `inherit` - The element inherits the clear value of its parent

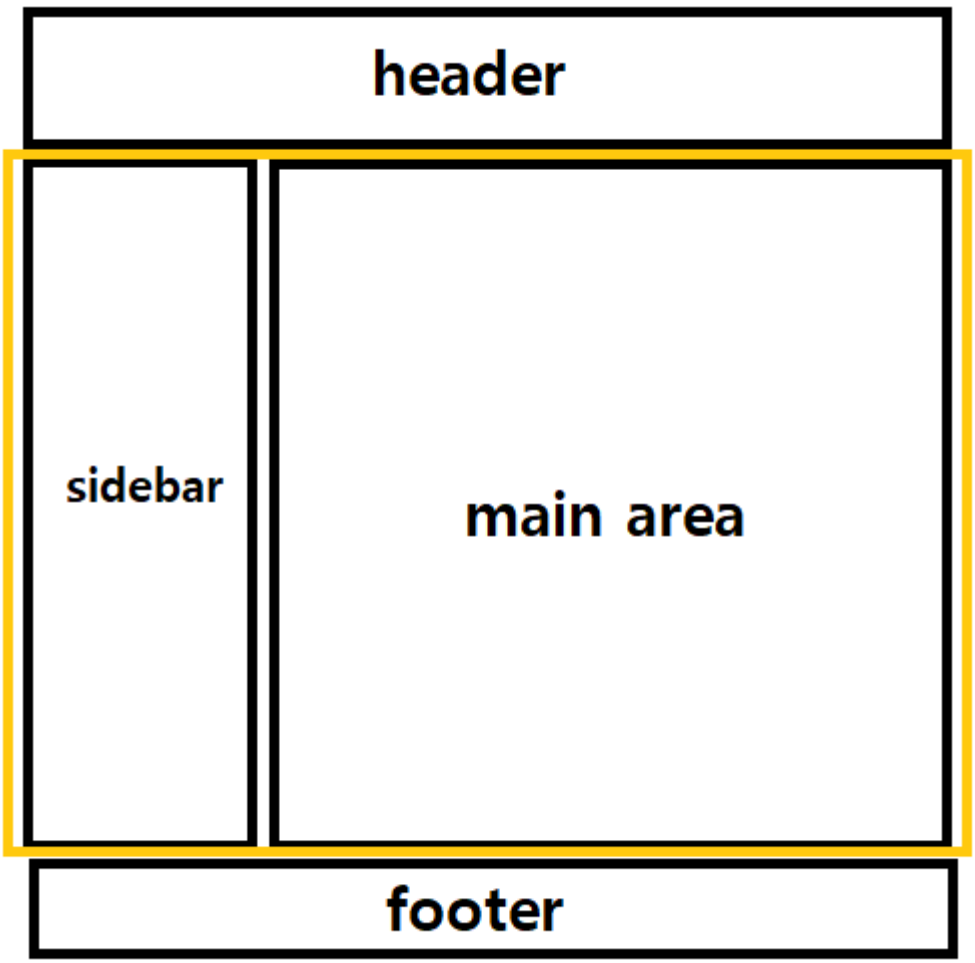
To test the `clear` property, let's temporarily change the width of `main-area` to `"70%"`.
Then, the width of the sidebar and main area totals to 90%, which leaves 10% of space.
If you check the browser, you will see that the footer has been moved next to the main area to fill this space.
To prevent this, we can use the `clear` property on the `<footer>` element.
Let's add `class="blog-footer"` to our footer element with corporation name.
Then, let's add the following CSS code.

```
footer {
  clear: left;
}
```

You will see that the footer is down at the bottom like before.
Now let's change the width of main area to 80% again.

The Clearfix Method

Now let's add a border to the sidebar and main-area like below.



To do this, let's make a `<div class="content-area">` element and put the `<nav class="sidebar">` and `<div class="main-area">` element inside it.
Then, let's add the following CSS code.

```
.content-area {
  border: 3px solid black;
  border-radius: 5px;
}
```

If you check the browser, you will notice that we can only see the top of the border.
Why does this happen?

This is because the parent element(`content-area`) is *not aware of child elements that are floated*(`sidebar` and `main-area`).
You can think of the floated elements as floating **up** from its parent and siblings so that they do not recognize it.

There are several methods to fix this. One of them is called the *clearfix method*.
Let's first add a class `clearfix` to our `content-area` div element.
Then, let's add the following CSS code.

```
.clearfix:before,
.clearfix:after {
  content: "";
  display: table;
}

.clearfix:after {
  clear: both;
}

.clearfix {
  zoom: 1;
}
```

This is not to memorize, but to copy and paste every time you have this problem when using `float`.
Now you will see that the border appears properly.

We can reuse this `clearfix` class anywhere on our page where we have a parent or a container element with floated children, and we need that element to be able to recognize its child elements.