{ } UX/UI Design

Activity | Practice with Positioning Activity

~20 min | Solo | Laptop

Overview

It's time to try your hand at positioning with all the properties we just discussed in class. Understanding positioning is a fundamental aspect of web development and knowing how to use different display properties. Without an understanding of how to position content, developing layouts is very difficult.

There are four steps to this activity. When you are finished with one step, move on to the next.

Instructions

Step 1: Floats
Floats are primarily used to flow text around images but they have another use case.

Float can also be used to position elements inside containers to the farthest left or farthest right that the element can be positioned.

Let's try out this principle in code and see for ourselves by building a very basic article layout!

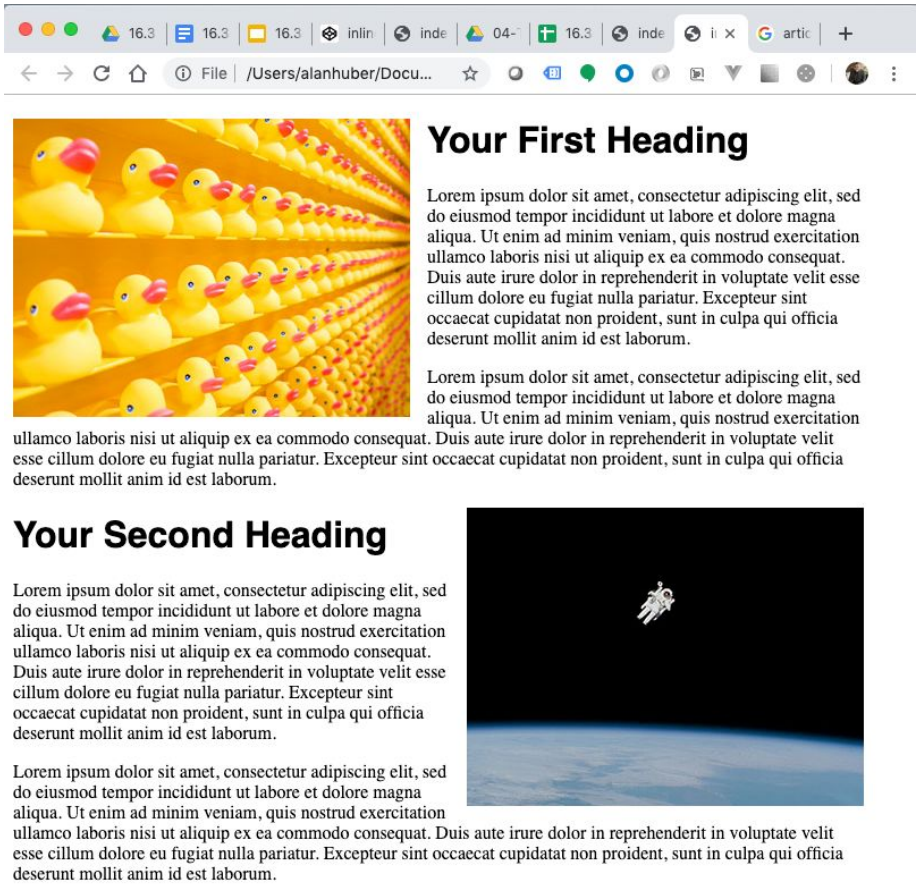
Note: *Floats can also be used to position elements next to each other, but this can cause unexpected behavior for parent containers.*

Instructions:

1. Open index.html contained in the folder named 1_float, located in Activities/positioning_elements/1_floats/index.html.
2. Open this index.html file in your web browser. Notice how the text sits near the bottom of the image? Let's move it to the side of the first image.
3. Open index.css located in /unsolved/css/index.css
4. Create a CSS selector that targets the img tag with the class of .left.
5. Add the property `float: left;` to the CSS selector you just wrote.
6. Notice how our text is too close to our image? Our image needs some padding so that our text doesn't sit so close to it. Add the property `padding-right:15px;` to the class of

.left that you created in the previous step to space our text farther away from the paragraph content.

7. However, we have another image that needs to be positioned. The second image has a CSS selector of .right. Add .right to your stylesheet and give it the property `float: right;`
8. This element also need some padding to move the text away from it. Give .right the CSS property of `padding-left: 15px;`
9. Lastly, our text is spanning the full width of this page because it's a block-level element. We need to give it a defined width so that our text looks better.
10. In our HTML, we have a div with the class of separator that is containing all the contents of our mock article.
11. Write a CSS selector that targets the class of separator.
12. Give this selector the CSS property `width: 750px;`
13. Your page should look like the following image:



This layout could be an informational page, a blog post, or an article about a certain topic, depending on your needs.

Resources:
For more information on how floats can be used to position containers, read the article "How to Clear Floats" by the W3 school.

Step 2: Block and inline elements

Block-level elements always start on a new line and take up the full width available and are used to contain inline content.

Block-level elements can also contain other block-level elements if your container needs more structure.

Block-level elements can be moved around your page using margin and automatically moved across the page using automatic margins.

Inline elements do not start on a new line and only take up as much width as necessary and are contained inside block elements when constructing content.

Inline content contained inside a div can be controlled by adding the text-align property to the div that holds our inline elements.

Instructions:

1. Open index.html contained in the folder named 2_block_and_inline_elements, located in Activities/positioning_elements/2_block_and_inline_elements/index.html.
2. Open this index.html file in your browser so you can see what it looks like before we style it.
3. In the index.css files located in css/index.css, there is already a style targeting an element with the class of block_element. This is where most of our CSS is going to be directed. We have already added styles to this document so that you can see these elements display on your page.

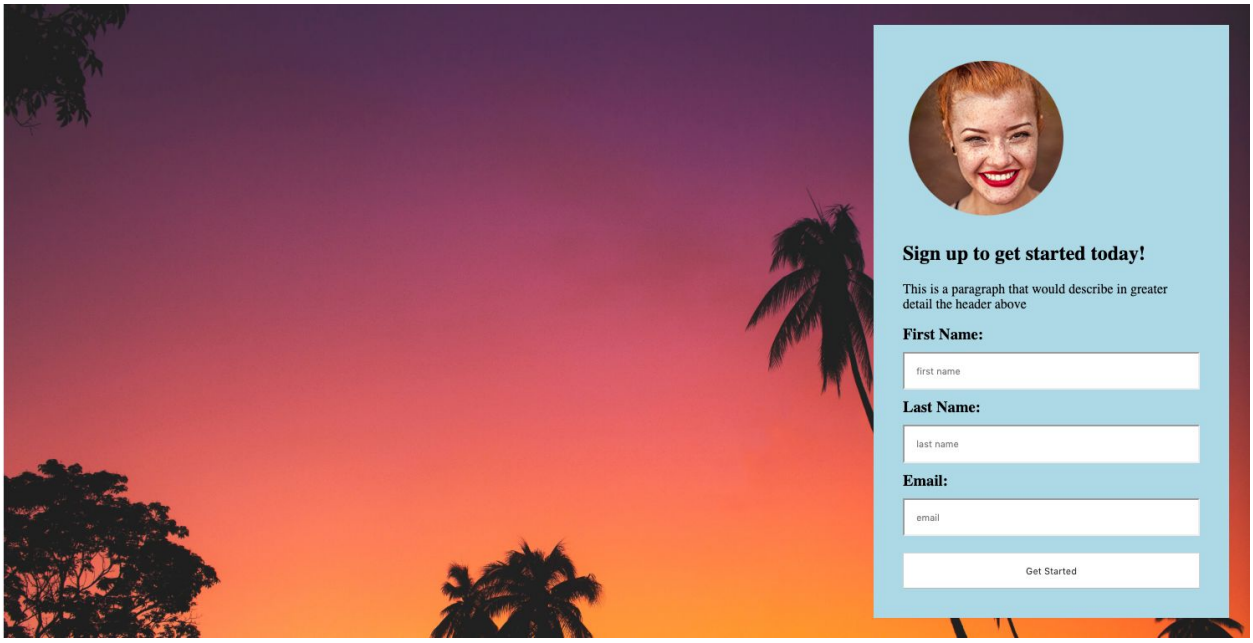
```
.block_element {

}
```

    - *If you don't want to continually refresh the page, feel free to use Chrome's web inspector to preview how your styles affect your HTML in real time!*

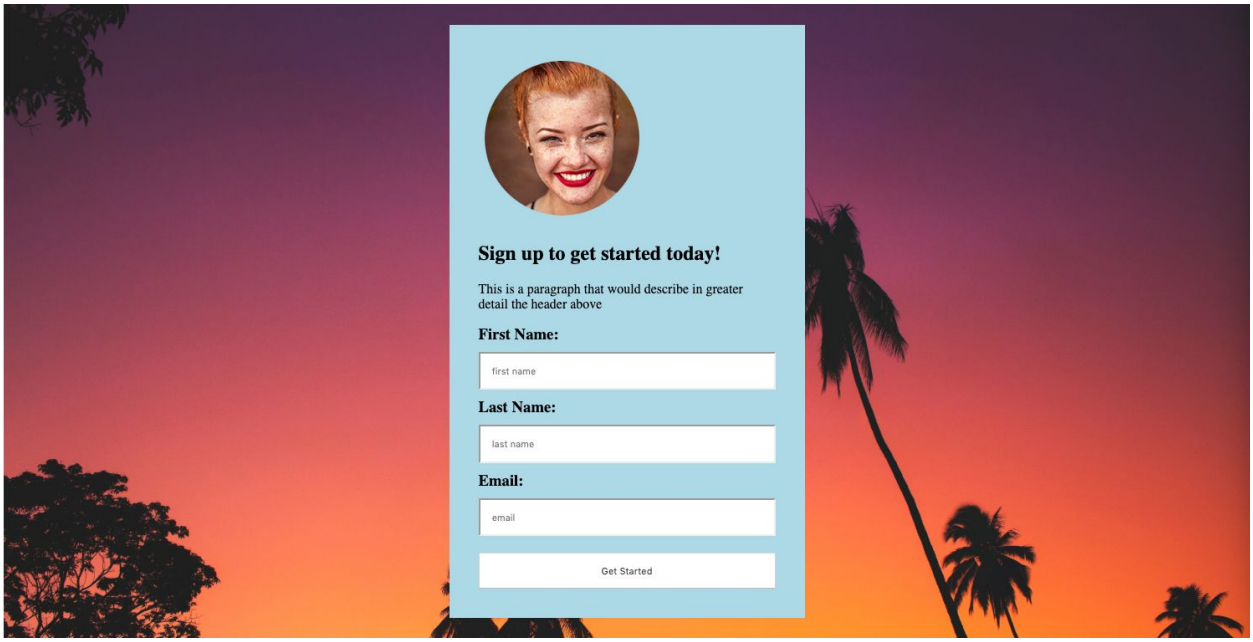
4. Notice how the div (which is a block-level element) that contains our other form already spans the entire width of this document without specifying in our CSS. Block-level elements by default take up 100% width of their parent containers.

5. Add the css property `margin-left: auto;` to the class `.block_element`.
   Note: Any kind of auto margin will not work if your div does not have a defined width, because you can't center something that is already taking up 100% of the width.

6. Reload the page to see how your element is now pushed to the far right. This is an important concept that will really help you build interesting layouts, as it allows you to position block level elements inside a containing element!

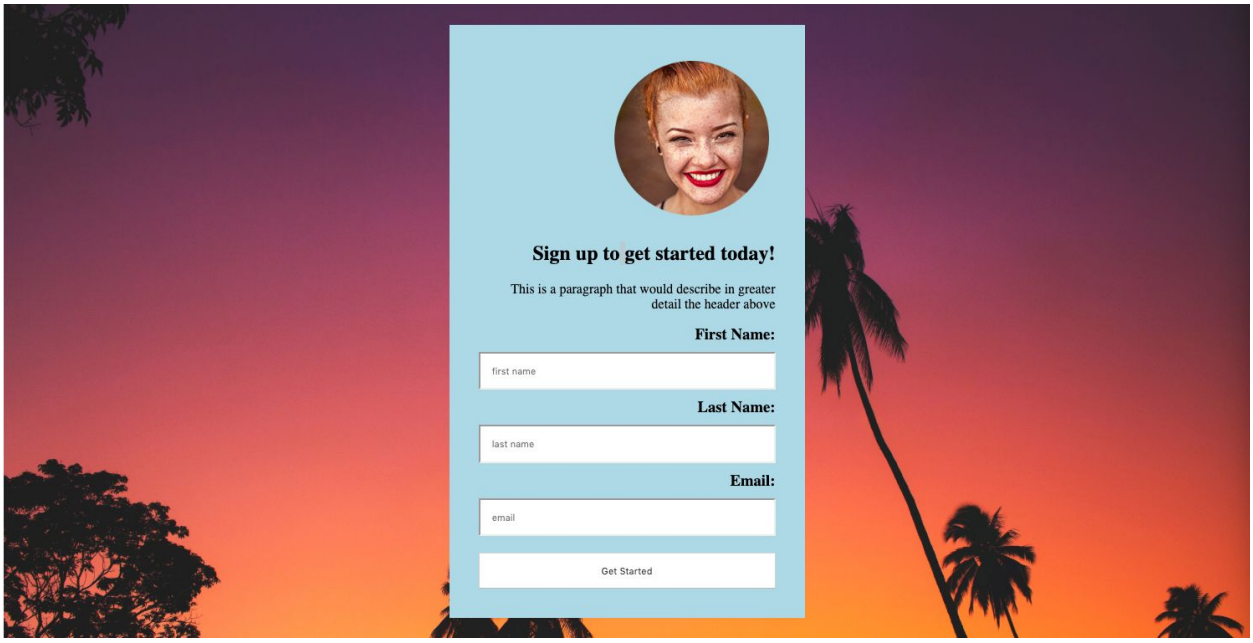


7. Now delete the property `margin-left: auto;` from the class `.block_element` and add `margin: 0 auto;` to it instead. Notice how the element is now centered in its container.

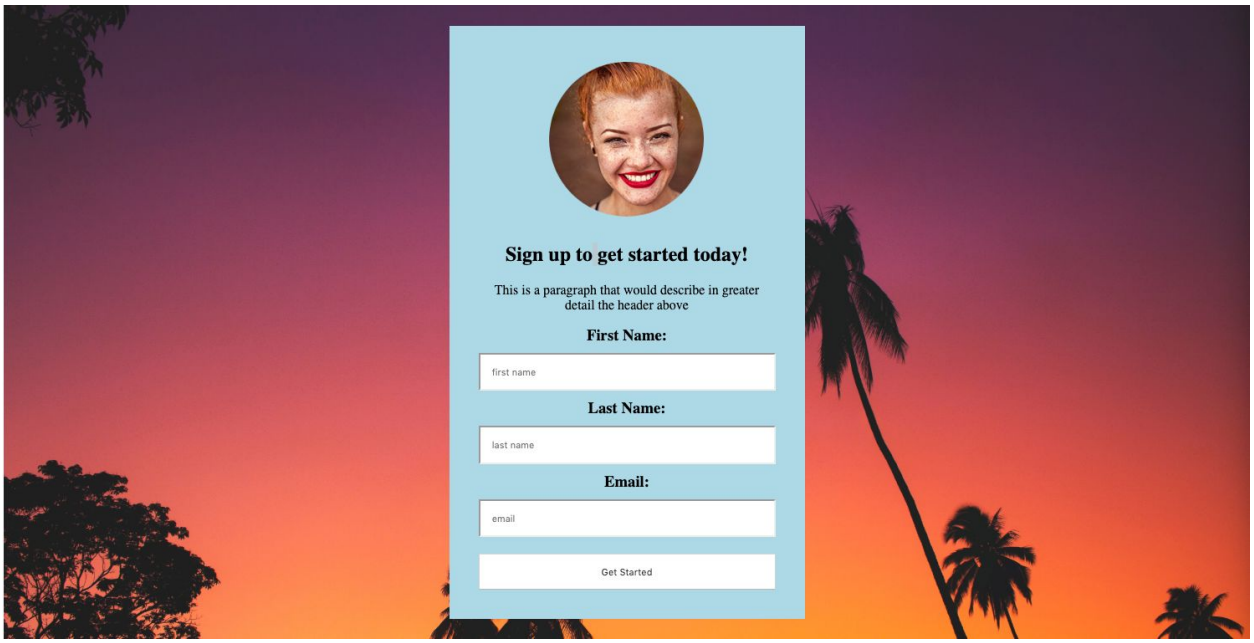


8. Next, let's position our text using the CSS property `text-align`. Text-align is applied to the parent container and affects any inline element nested inside the parent tag (images, etc.).
9. Add the CSS property `text-align: right;` to the class `.block_element` and reload your page (or use the web inspector).

   Notice how your inline elements are now positioned to the right of your div?



10. Now delete `text-align: right;` and add `text-align: center;`
11. Reload the page and see the results.



Resources:

- text-align: https://www.w3schools.com/cssref/pr_text_text-align.asp
- margin: https://www.w3schools.com/css/css_margin.asp

Step 3: Positioning inline-block level elements

Inline-block elements can be used to create elements that line up next to each other such as tabs, navigation items, or social media icons that all sit next to each other.

Inline-block level elements can also be used to create sections that line up next to each other.

Instructions

1. Open index.html contained in the folder named 3_inline_block_elements, located in Activities/positioning_elements/3_inline_block_elements/index.html.
2. Open this index.html file so that you can see how it looks before we style it.
3. Write a CSS selector that targets the HTML element with the class of inline-block.
4. In your selector with the class of `.inline-block`, add the CSS display property of inline-block: `display: inline-block;`
5. Give it a padding of 15px so our text is centered inside the containing div.

```
padding: 15px;
```

6. Reload your page.
7. In your CSS file, write a selector that targets the HTML element with the class of left.

```
.left {
```

```
}
```

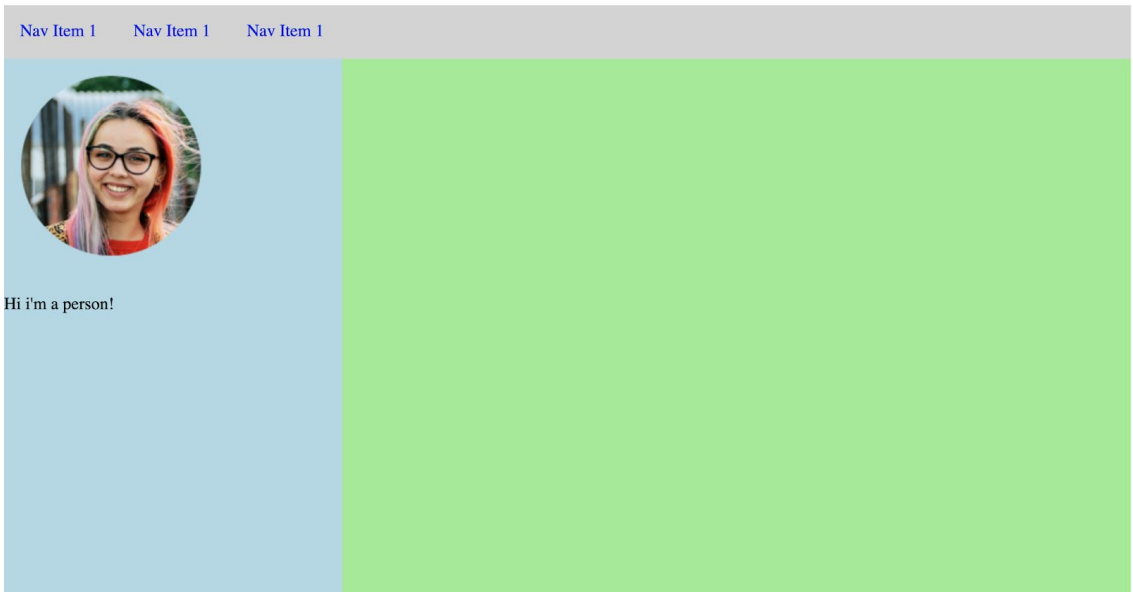
8. In your CSS file, write a selector that targets the HTML element with the class of right.

```
.right {

}
```

9. Give both classes a height of 500px and float left

```
.left {
height: 500px;
float: left;

}
.right {
height: 500px;
float: left;
}
```

10. Give .left the css property of background-color with the value of lightblue.
11. Give .left the css property of width with a value of 30%.
12. Give .right the css property of background-color with a value of lightgreen.
13. Give .right the css property of width with a value of 70%.
14. Reload the page to preview your container.



15. You may have noticed that the container for .left has an image and p tag with text inside it. Let's position these elements.
16. Give .left the property of text-align with a value of center.

    a. Img tags are inline block level elements that can be centered with text align just like the text contained inside our p tag.
17. Lastly we have some cards in the div with the class of .right.
18. Write a css selector that targets the class of card. Give it the following properties:
    a. Give it the css property display with value of inline-block.
    b. Give it the css property background-color with a value of #C6C6C6.
    c. Give it the css property of border-radius with a value of 5px.
    d. Give it the css property of height with a value of 200px.
    e. Give it the css property of width with a value of 150px.
19. Now that we have styled our cards let's position them.
20. In your class of .right add the following properties:
    a. Add the property text-align with a value of center.
    b. These will center all of our cards together.
21. In your class of .card add the following properties:
    a. Add the property margin with a value of 35px.
    b. Add the css property of margin-top with a value of 50px.

Congrats! That's it! Remember, while this activity may seem simple, all the techniques demonstrated today can (and will) be used for creating other layouts. Many of these techniques are responsive and very useful when creating items in flex & grid containers (more on this next week).
We'll be using these properties in our next activity to build a basic website!