IST 263

# Lab 08

## This lab covers:

1. Containers
   1. DIV, SPAN
   2. Semantic Tags
2. Floats, Clearing Floats
3. Flexbox

## SETUP

Create a folder in your Github repository called lab08. Place a copy of your latest skills, bio, contact pages and stylesheet in the new folder. Please make sure to keep the same file names in the new folder or your navigation from the last lab will not work.

## Containers

Semantic tags should be on every web page defining the different areas. They allow us to style different sections of the page, help with accessibility so folks with screen readers understand what the different parts of your content. They also help with SEO (Search Engine Optimization) so your website shows up on the first page in a Google search.

We've already put the nav tag on all the pages around our site navigation in a previous lab. Let's look at the list and apply some of the other semantic tags to sections of our page.

* <article>
* <aside>
* <details>
* <figcaption>
* <figure>
* <footer>
* <header>
* <main>
* <mark>
* <nav>
* <section>
* <summary>
* <time>

Here's my current bio page (index.html) with boxes drawn around the areas we are going to wrap in semantic containers.



1. Let's start with the box in red. This is really a page header image. If your image is a banner style header, wrap it in header tags. This is mine right now:  
     
   <header><img src="../images/bio-banner.jpg"></header>  
     
   Stand by if your image tag is not a header style banner. We'll deal with images that need to flow with text later.
2. Next let's deal with the green box. This is the main content of the page so I'm going to wrap it in the main tag. It's important not to overlap tags. The first paragraph starts with a p tag. The main tag should be before that so:  
     
   <main>  
   <p>Lorem Ipsum</p>  
   …  
   <ul>  
   <li></li>  
   </ul>  
   </main>  
     
   You will have a lot more tags in between the place where you wrap the main tags. Make sure all the tags in between begin and end. You never want this <main><p></main></p>. <- Writing this bad overlapping was painful to my web developer eyes!
3. Last let's define the footer. Wrap footer tags around your horizontal rule (hr tag) and copyright.

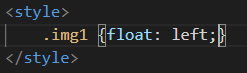
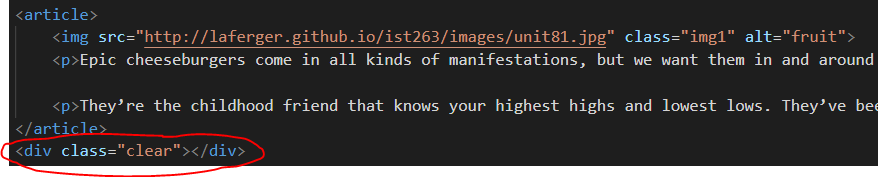
## Floats

The float CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page, though remains a part of the flow.

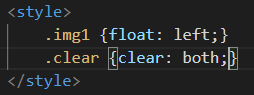
In this section we are going to work on some positioning using floats.

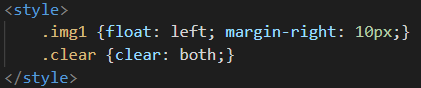
### Practice

To walk you through floats, I've placed a practice file in blackboard. Download it and save it in your lab08 folder. Examine the HTML, and follow the directions below.

1. For the first article we are going to do a simple left float of the image. To do this you must refer to the image's class selector and use the property float with the value left. Use the embedded stylesheet to add the following:  
     
   
2. If you reload your page you will see that the image floated to the left, but so did all the content under it. We need to tell the HTML page when to end the float by clearing it. We add a div tag to the HTML to do this.   
     
   

Then style the div tag with the following in your embedded stylesheet:



1. One thing is still bothering me. The text is squished next to the image. Let's add a margin to fix this.  
   

### Now You Try

Write a short tag line, motto or find a short quote for your website. We're going to position this in the footer to the right of the copyright. Mine is: *“First, solve the problem. Then, write the code.” – John Johnson.*

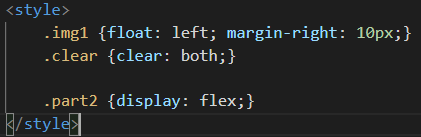
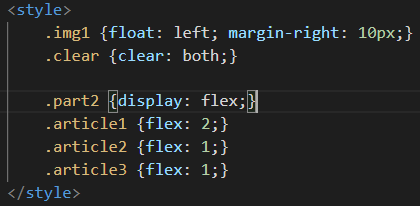
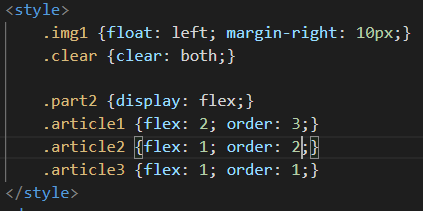
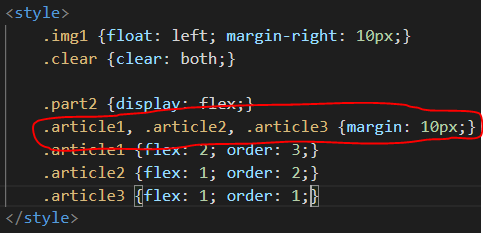
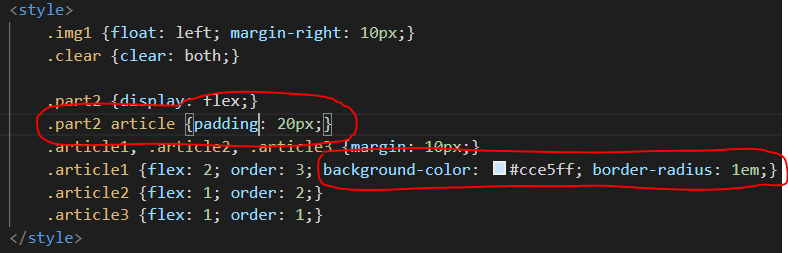
1. Add the text you picked just under the copyright and use a float to position the new text to the right so that the copyright and new text ends up in the footer on opposite sides.  
     
   Hint: Apply the float and everything after wraps around the item you floated. If you apply the float in the wrong place yours will not look like the one below.   
     
   Here's what my footer looks like when done:  
     
   
2. If your image wasn't a banner style header (like above), use floats to wrap the text around the image. You may have to add a margin to the image as well so there is space between the image and the text. You can float the image left or right.

## Flexbox

Flexbox is a one-dimensional layout method for laying out items in rows or columns. Items flex to fill additional space and shrink to fit into smaller spaces. Scroll down in the practice file and you will see an H2 element called "The Flexbox Part".

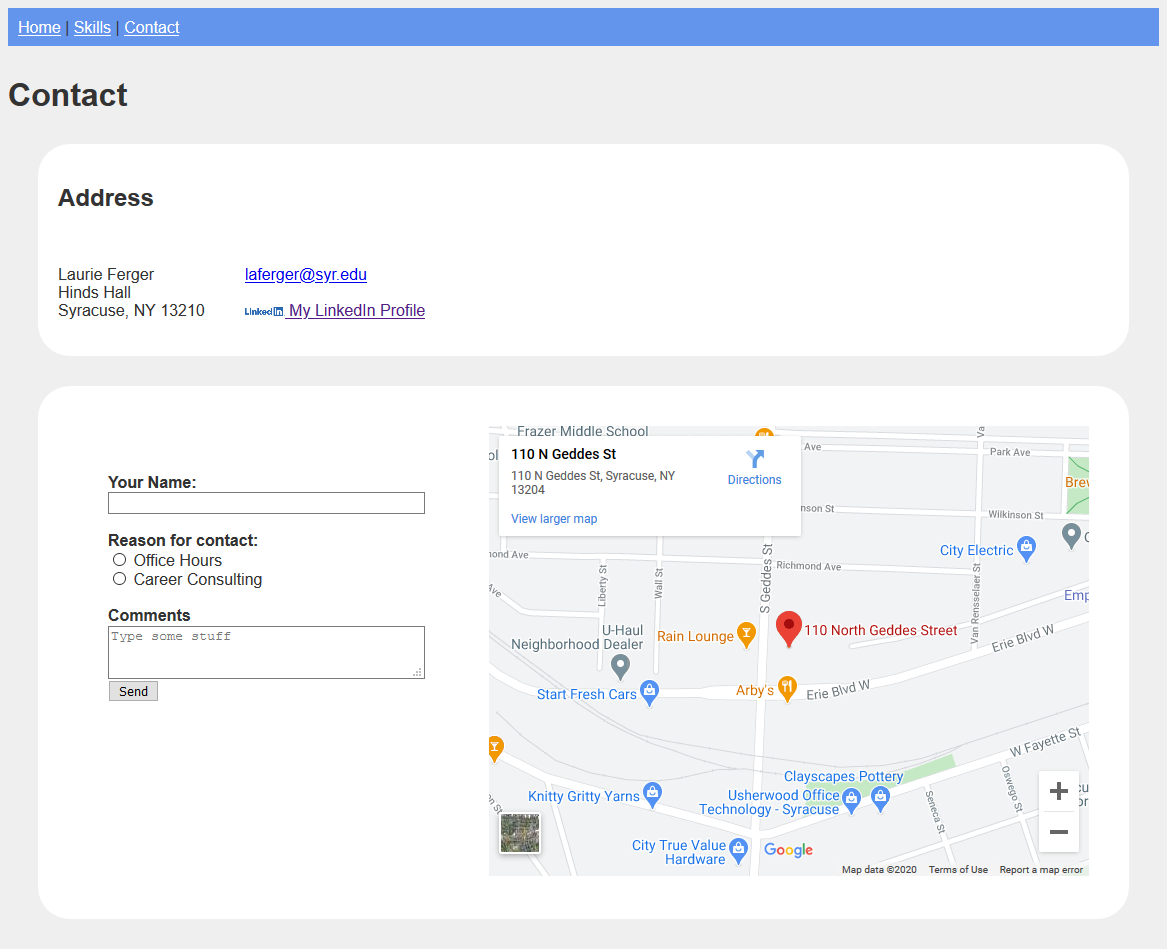
We are going to position those boxes side by side using flex and ratios. Flex uses a parent and children relationship. What does this mean? If you declare an element to be flex in CSS then it's immediate children elements will display side by side. In our case we are going to declare the div container with a class of part2 to be the parent. That means the 3 article tags are that elements immediate children.

### Practice

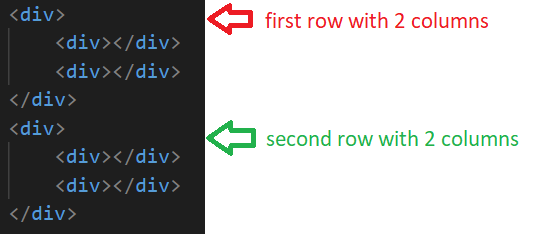
1. Add a style to the embedded style sheet to make the part2 div flex.   
     
   Reload the page and you should see that all the articles are side by side.
2. Let's assume that the first article is the most important and you would like it to take up more space. We are going to assign a ratio for the size of the three boxes. This is done by using class selectors for each box and using the flex property with an integer for the ratio. In this case I'm going to make the first article double the size of the other two. Add the styles below and refresh to see what happens.   
   
3. One of the cool things about flex is that you can change the order of the items that layout without moving the content around. In our case we are going to reverse the order of the boxes in the layout with the order property. The value will be what place in the layout you want your box to occupy.  
     
   
4. All the text is squished together in our layout so let's add some space between the article elements. You can do this with simple margins and a grouped selector.  
     
   
5. This is the perfect time to show you the non-flex layout trick of rounding corners. We're going to apply it to the first article in the Flex part (which is actually displaying as our last because of order). Start by putting a background-color on article1. Rounded corners are accomplished using the border-radius property. The number after describes how rounded you want the edge.   
     
   I'm also adding some padding with a descendant selector so that there is white space around the box with this article.  
     
   

### Now you try it

You are going to do an organized layout of your contact page. This is what mine looks like:



1. We are not changing anything about the nav or page header (contact heading) layout so we’re don’t need to do anything with them for this part.  
     
   If you look at the rest of the page above you can convert it into two rows with two columns. Here’s how I’m thinking about it. I’ve outlined the first row and columns in red and the second row and columns in green.  
     
   Graphical user interface, application

   Description automatically generated
2. We will be using flexbox to lay this out so start by typing all the div tags you need to make these rows and columns. You should have this:  
     
   
3. We have 4 pieces of content that we need to move into the div tags we made above.   
   Row 1 is: address in the first column / email and linked in links in the second column  
   Row 2 is: form in the first column / map in the second column   
     
   Move the content into the div tags in the correct location and be careful. For example, make sure you get the form tags and everything inside them when you move the form into row 2 column 1.
4. Using flexbox position the first row's content (address and links) side by side. If you need to look back at examples of the properties, look at the practice at the beginning of this lab or notes from class.   
     
   Layout for this part will use the CSS properties flex and display.

* What flex properties are needed to make the content appear side by side?
* Make sure you do not have tags overlapping incorrectly.
* Where do you need to add class attributes and values so you can apply styles?
* Where do you need to add padding or margins to allow for whitespace?

1. Using flexbox position the second row's content (form and map) side by side.
2. Your layout must contain the following:
   1. Background colors on both boxes
   2. Space between edge of the page and flexboxes.
   3. Rounded corners
   4. Space between top and bottom containers. (Your two rows shouldn’t look squished together)
   5. Space inside the containers.
   6. Form and map should not look like it's squished to the left. Allow it to fill the space.
   7. Everything must have adequate whitespace.
   8. Work on the appearance of your form. Now is the time to make sure it looks good.

## What will You hand IN?

Create a word document, pdf or use the "write submission" option in blackboard to provide the following:

1. Submit the urls for the index page, practice page and contact page.
2. Validate the contact page and submit the validation link.
3. Submit answers to the following:
   1. What questions did you have about the lab? What didn't you fully understand?
   2. What was the hardest part of the lab?
   3. Rate your comfort level with this week's topics.  
      1 ==> I can do this on my own and explain how to do it.  
      2 ==> I can do this on my own without any help.  
      3 ==> I can do this with help or guidance from others.   
       If you choose this level, please indicate HOW this person helped you.  
      4 ==> I don't understand this at all yet and need extra help.