Week 8: CSS Part III

Agenda

- Overview
  - Previous Participation Questions/Responses
  - Midterm
  - What's Due?
- More Positioning with CSS (Lecture/Class Activity)
  - Box Model Review
  - Positioning with CSS
    - Float
    - Flexbox
- CSS Positioning Exercise 2 (CW/HW)
- Participation Proof
- Next Week

Overview

Last week, we learned about how we can use the box model along with other properties such as float to position items on our page. This week, we will go over those concepts again and learn a little more about how we can position our items and give them plenty of white space. By the end of this week, students will be able to position elements on a page using the float or flexbox properties.

Previous Participation Questions/Responses

These are the participation questions from last week:
https://forms.gle/rMaLpfDhz1gT7UgJ7

This link has the responses to the fun question from last week. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

Midterm

The **Midterm is due on October 28th at midnight**. Please make sure you have started working on it or start soon. Overall, you should be creating 3 pages of HTML that are linked together. You can do this with the <a> tag's href attribute. There should also be one page of externally placed CSS for all three pages of HTML. For more details, please review the "Final Project Description" document available on Blackboard → Final Project.

You don't need to have perfectly placed or finalized content. However, your pages should be mostly put together. Major headings should be used to describe content. If you haven't thought of text to use, feel free to Google "Lorem Ipsum" for place holder text. You can also use place holder images by Googling those as well. The most important thing is that you have some sort of content placed for your Midterm.

The basic-page-example.html file provided earlier in the semester is a good an example of an acceptable page. Feel free to use some of the CSS I've provided (row class, float items to the left, give them a width) to help style your pages. The overall structure should be original and your own work though.

For the Midterm submission, you'll submit 3 links in total. Each link should point to a different page of your midterm. When I check your work, I should be able to go from page to page easily by using links on your pages.

What's Due?

- Participation Proof
- CSS Positioning Exercise 2 (CW/HW)

More Positioning with CSS (Lecture/Class Activity)

Box Model Review

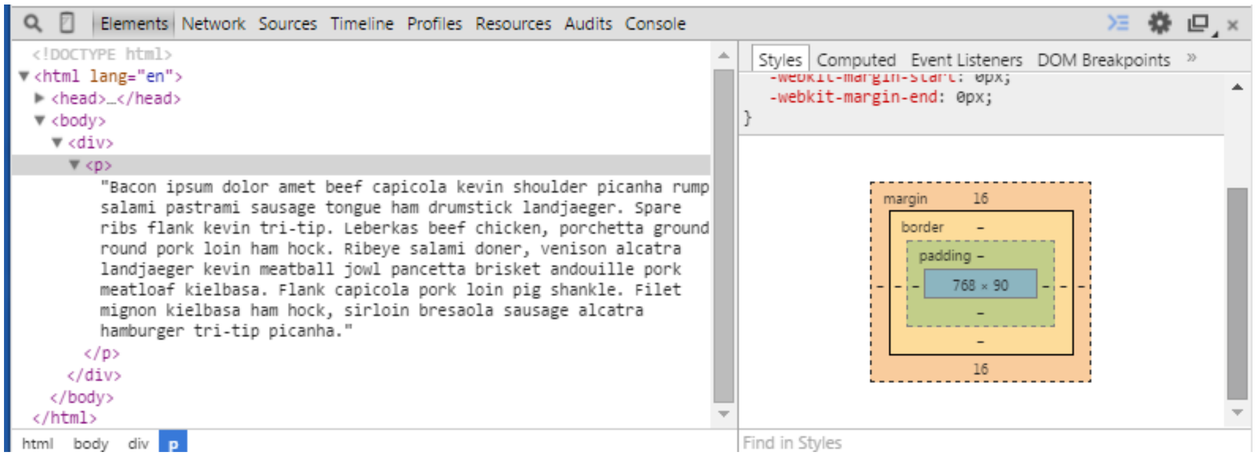
The Box Model in CSS consists of three properties which are padding, border, and margin. The padding and margin properties are the most important of the three since they control the space of the elements on your page. Borders can improve aesthetics but most items on a page don't have one. However, they might have some sort of default margin or padding added to them.

Everything in HTML is a box and so you must learn to think about your content in terms of boxes/containers which hold other boxes.

The padding property handles the space between the content and the border around that content.

The border property handles borders which wraps around the padding and content.

The margin property handles space outside of the border and wraps up the border, padding, and content.



```
<!DOCTYPE html>
▼<html lang="en">
  ▶<head>…</head>
  ▼<body>
    ▼<div>
      ▼<p>
          "Bacon ipsum dolor amet beef capicola kevin shoulder picanha rump
          salami pastrami sausage tongue ham drumstick landjaeger. Spare
          ribs flank kevin tri-tip. Leberkas beef chicken, porchetta ground
          round pork loin ham hock. Ribeye salami doner, venison alcatra
          landjaeger kevin meatball jowl pancetta brisket andouille pork
          meatloaf kielbasa. Flank capicola pork loin pig shankle. Filet
          mignon kielbasa ham hock, sirloin bresaola sausage alcatra
          hamburger tri-tip picanha."
        </p>
      </div>
    </body>
</html>
```

```
  -webkit-margin-start: 0px;
  -webkit-margin-end: 0px;
}
```

In the image above, I have used Google Chrome's "Inspect Element" feature (F12 or right-click and select it) to view the HTML/CSS in a page. On the left side of the inspector, the HTML structure is viewable. On the right side, there is a multi-colored box which depicts the CSS box model in action.

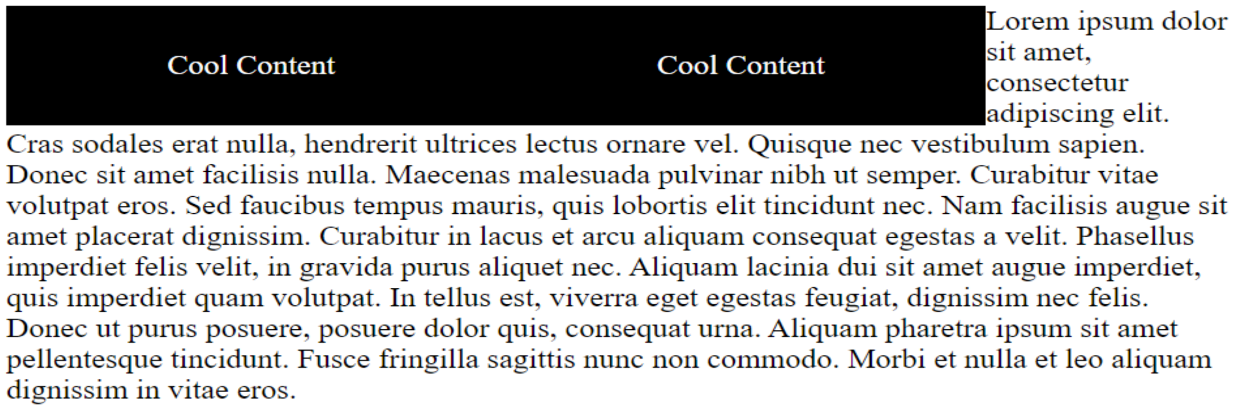
The orange area represents the margin (16px default for paragraphs in Chrome). The border is depicted by the yellow area. Lastly, the green area is the padding which separates the content from the border.

Positioning with CSS
Last week, we covered how to position our elements on the page using CSS. We will review the concepts again and then you will complete a second positioning exercise.

Float Property
This property takes content and aligns it to either the left or right side of a container. The content is removed from the normal document flow, but text content and inline elements will wrap around it. Block level elements will appear on the next available line.

It was intended originally for having text wrap around images but has been used to manipulate the positioning of elements on a page. The problem is that if the total width of items being floated is under 100%, other items below your floated ones may move up in that space and wrap around your content.

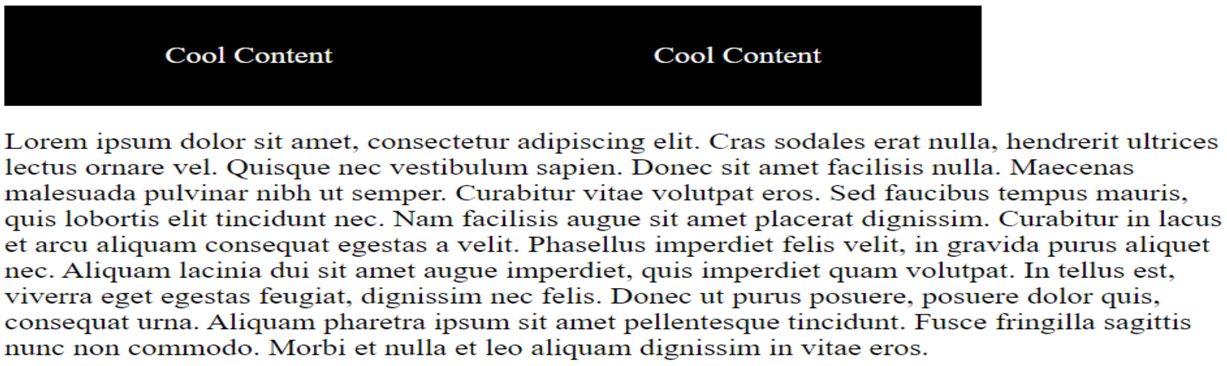
To prevent unwanted wrapping of content, you would have to clear your floats. This means you use the clear property on the element which wraps your floated elements. Unwanted wrapping would look something like the screenshot below.


Cool Content

Cool Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras sodales erat nulla, hendrerit ultrices lectus ornare vel. Quisque nec vestibulum sapien. Donec sit amet facilisis nulla. Maecenas malesuada pulvinar nibh ut semper. Curabitur vitae volutpat eros. Sed faucibus tempus mauris, quis lobortis elit tincidunt nec. Nam facilisis augue sit amet placerat dignissim. Curabitur in lacus et arcu aliquam consequat egestas a velit. Phasellus imperdiet felis velit, in gravida purus aliquet nec. Aliquam lacinia dui sit amet augue imperdiet, quis imperdiet quam volutpat. In tellus est, viverra eget egestas feugiat, dignissim nec felis. Donec ut purus posuere, posuere dolor quis, consequat urna. Aliquam pharetra ipsum sit amet pellentesque tincidunt. Fusce fringilla sagittis nunc non commodo. Morbi et nulla et leo aliquam dignissim in vitae eros.

In the image above, I have 2 divs with 40% width each floated to the left. Currently, they don't have the clearfix code implemented and 0 margin to space them out. Since there is 20% of the width still available, content below these divs will try to fill that space. If you use the clearfix method appropriately, the content can have whatever widths you'd like, and content won't move to its side as shown in the next image.


Cool Content

Cool Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras sodales erat nulla, hendrerit ultrices lectus ornare vel. Quisque nec vestibulum sapien. Donec sit amet facilisis nulla. Maecenas malesuada pulvinar nibh ut semper. Curabitur vitae volutpat eros. Sed faucibus tempus mauris, quis lobortis elit tincidunt nec. Nam facilisis augue sit amet placerat dignissim. Curabitur in lacus et arcu aliquam consequat egestas a velit. Phasellus imperdiet felis velit, in gravida purus aliquet nec. Aliquam lacinia dui sit amet augue imperdiet, quis imperdiet quam volutpat. In tellus est, viverra eget egestas feugiat, dignissim nec felis. Donec ut purus posuere, posuere dolor quis, consequat urna. Aliquam pharetra ipsum sit amet pellentesque tincidunt. Fusce fringilla sagittis nunc non commodo. Morbi et nulla et leo aliquam dignissim in vitae eros.

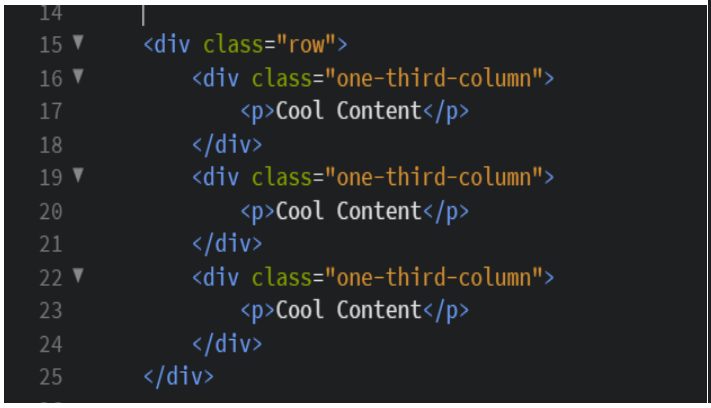
There are many methods to clear floats but the best one I have found is here:

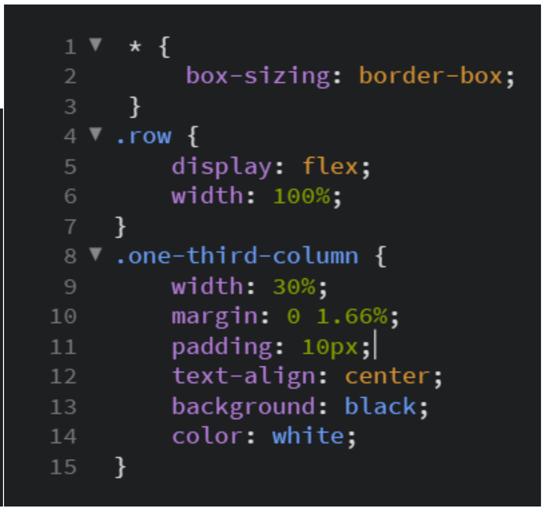
https://css-tricks.com/snippets/css/clear-fix/

To use the clear fix in the link above, you need to wrap all floated elements in a parent container (div usually). Then you target that parent container's before/after pseudo property and apply the styles as depicted. The before/after pseudo selectors allow us to add extra content to our HTML without using anything extra in the HTML markup. However, in our case, we don't display any content (that's why it has empty quotation marks). We display it like a table and clear both sides of it.

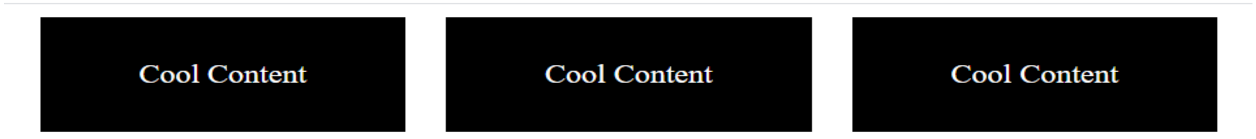
Flexbox

Flexbox is an awesome property which allows you to create layouts without using floats or positioning. Like floats, it requires the usage of a parent container which has other items/containers inside of it. Unlike floats, you don't need to worry about clearing anything and it's easier to implement. If we used the same HTML code as we did previously to create a 3-column layout, the CSS would look like this instead:


```html
<div class="row">
    <div class="one-third-column">
        <p>Cool Content</p>
    </div>
    <div class="one-third-column">
        <p>Cool Content</p>
    </div>
    <div class="one-third-column">
        <p>Cool Content</p>
    </div>
</div>
```


```css
* {
    box-sizing: border-box;
}
.row {
    display: flex;
    width: 100%;
}
.one-third-column {
    width: 30%;
    margin: 0 1.66%;
    padding: 10px;
    text-align: center;
    background: black;
    color: white;
}
```

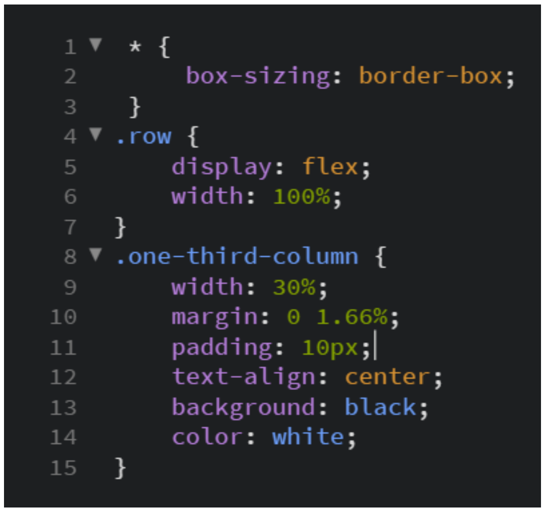
Rather than using the row class for clearing, just simply give it a display of flex and a width of 100%. From there, we apply all the same styles to the children containers (one-third-column) with the exception of floating it. The final result would look pretty much the same, but with less effort as far as clearing floats and choosing float directions:



As a quick recap, for flexbox, you need a flex container and flex items. The flex container is just the parent container which wraps the elements you want to move around on the page. The flex items are the elements inside of the container which will be displayed horizontally or some other way.

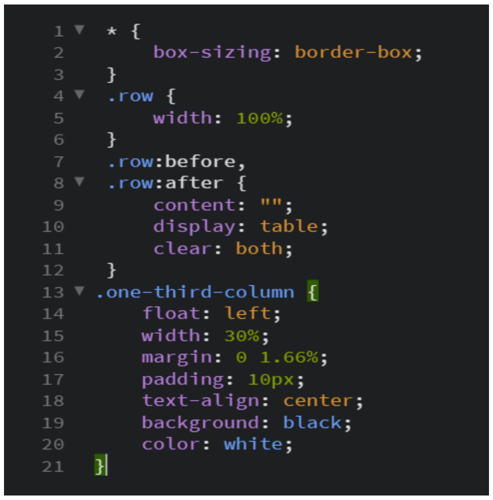
For comparison purposes, I've provided a screenshot of the CSS we would use for the same layout, but by using float instead of flexbox. The underlying HTML is still the same.

Flexbox



```
* {
    box-sizing: border-box;
  }
.row {
    display: flex;
    width: 100%;
}
.one-third-column {
    width: 30%;
    margin: 0 1.66%;
    padding: 10px;
    text-align: center;
    background: black;
    color: white;
}
```

Float



```
* {
    box-sizing: border-box;
}
.row {
    width: 100%;
}
.row:before,
.row:after {
    content: "";
    display: table;
    clear: both;
}
.one-third-column {
    float: left;
    width: 30%;
    margin: 0 1.66%;
    padding: 10px;
    text-align: center;
    background: black;
    color: white;
}
```

The reason you would use float over flexbox is because float is more universally implemented by browsers. Some older browsers don't support the flexbox property, and then your styles wouldn't work as expected. It's important to understand your audience and the types of devices/browsers they are using. For the purposes of this class, flexbox and floats are acceptable for assignments and the final project.

There are many other properties for the flex containers/items, so make sure to review them to fully understand the possibilities with flexbox. My example above is useful for creating quick multi-column layouts.

Online Resources (Not required, but can be helpful for guidance)

- https://css-tricks.com/almanac/properties/p/position/ (Position)
- https://developer.mozilla.org/en-US/docs/Web/CSS/float (Float)
- https://css-tricks.com/all-about-floats/ (Float)
- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox (Flexbox)
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (Flexbox)
- https://css-tricks.com/snippets/css/complete-guide-grid/ (Grid)
- https://www.w3schools.com/css/css_grid.asp (Grid)

CSS Positioning Exercise 2 (CW/HW)

Download the "CSS-Positioning-Exercise-2.zip" folder from Blackboard under today's Lecture Notes or the Assignment folder. Extract everything from inside of this zip folder to some place where you will be able to find them easily. This week, there will be an HTML file and an image with an example of a finished assignment. The HTML file contains <style></style> tags for you to use so you don't have to place your CSS externally.

Based on the image provided, add CSS to the style tags so that the web page looks as close as possible to mine. You can change where it says "Your Name Here" to whatever name you want. In the image provided, it says "Chris Velez" in the header and "Christopher Velez" in the footer, but you can alter that to show your name.

It is also okay to change colors if the text contrast is high enough for the text to be readable. However, the items should be styled similarly in terms of their placement, individual styles (borders, padding, margins, underlines, etc.)

<u>General Tips</u>

- If you use floats, float everything in one direction (I personally always use left)
- To center an image, use "display: block;" and "margin: 0 auto;" (images are always inline so when we display them with block, we can add width/height, margins, etc.)
- Padding/Margin will help with spacing (sometimes needs to be set to 0)
  - Use px or em (em is a relative unit and generally, 1em = one times whatever the default body font size is – sometimes 16px – 2em would then equal 32px)
  - Margin: 0 auto also centers divs and other containers that have widths

<u>Styles Used</u>

I'm not going to list out the styles like I did last week because most of them are the same. However, I will provide some information to get you started on the page. You'll need to add padding besides what I've mentioned below to end up with a similar page.

Floated Items/Flexbox Children (give parent containers display: flex)

- .header h2
- .row .two
- .row .three

Widths/Margins

- Container: 800px maximum width, 100% width
- .row .two: 50% width
- .row .three 33% width
- .three p: 150px width, 10px auto margin
- .blog .two: 45% width, 2.5% margin

For this week's assignment only, you can use an internal style sheet. You may use an external CSS file if you'd like as well. No inline CSS should be used.

But Professor, I Don't Know Where to Start
If you're not sure where to start at all, this section was made for you! I've listed out basic steps that you can take below for your convenience. Follow them carefully and you should be able to get the assignment started.

- Download the "Week-7-Lecture-Notes.zip" folder from Blackboard
- Take the "css-positioning-exercise-2.html" and "css-positioning-exercise-2.png" files and place them in a folder you can find.
    - Create a "Week-7" folder in your "Documents" folder or some other folder you can access easily
    - Place both files in this newly created "Week-7" folder
- Open the "css-positioning-exercise-2.html" file with a text editor of your choice
- Review the structure of the HTML and the image to see where things should be placed
    - Pay attention to which elements are parents vs children
    - Look at class names to see how you can structure your CSS selectors
- From here, you should start adding styles to your page (inside of style tags)
    - Start with styles for the content at the top (header content)
    - Colors for text and backgrounds
        - I use black, white, and #ccc for the gray parts
    - If you're using float, float the elements mentioned above in my styles guide
        - After floating elements, give them widths
    - If you're using flexbox, using the display: flex code on the .row class
        - After giving display: flex to the .row class, give your child elements widths
    - Center the images using margin: 0 auto and display: block
    - Apply padding/margin to space out elements

Your web page is **due October 21st at 6pm.** To submit the work for this exercise, go to Blackboard → Course Materials → Lecture Notes for this week's class. You can also visit the "Assignments" folder and the submission area will be titled "CSS Positioning Exercise 2" **You must submit a live link to your web page**. The work submitted will be evaluated based on the rubric explained below.

CSS Positioning Exercise 2 – 10pts

- Live link was submitted: 1pt
- CSS is organized/nested neatly: 2pt
- Elements on the page are positioned/styled correctly: 7pts

Participation Proof 10/14

There are no participation questions for this week. Instead, you will work on one of the following items:

- Any class assignment (today's assignment preferably)

- Final Project

To receive credit for participation today, you must show me what you worked on during class. I will go around the class after explaining the assignment for today and ask what work you plan on doing. Later in the class, I will come back around to check your progress. Once you have shown me enough work (will depend on what you're working on), I will write your name down, and then you are free to go.

Previous Participation Questions/Responses
These are the participation questions from last week:
https://forms.gle/rMaLpfDhz1gT7UgJ7

This link has the responses to the fun question from last week. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

Next Week

Next week, we will start learning about the third component of client-side web development: JavaScript (JS). The basics of JS syntax and some light programming concepts will be covered. Feel free to read through the links below for a brief introduction to JavaScript.

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.htmldog.com/guides/javascript/
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript