# Week 5: CSS Part II

**Agenda**

* More Fun With CSS
  + Box Model
  + Positioning with CSS
  + CSS Positioning Exercises
  + CSS Evaluation Initial Post & Response
* Next Week

## More Fun With CSS

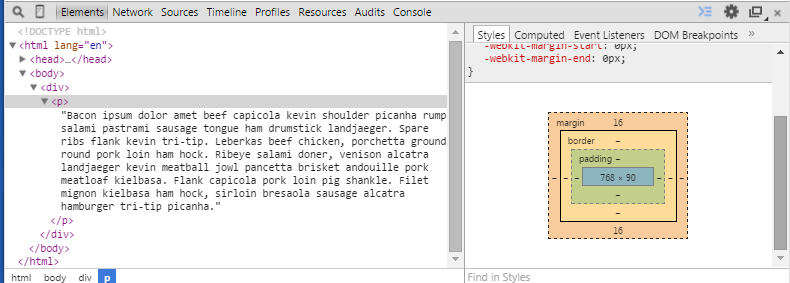
**Box Model**

Every element in HTML can be considered as a box. Each of these boxes takes up a certain amount of space on our page relative to the other boxes. Also, the spacing inside of these boxes can vary depending on the content inside of them. The box model in CSS is a way to understand how HTML is laid out and uses certain properties to help achieve this goal. It is comprised of **three** main properties. The properties are padding, border, and margin which all impact how elements are spaced out on a page:

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.



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. The links below cover the CSS Box Model in greater detail and explain things like inline vs block level elements as well. The CSS slides on Blackboard also provide a quick glimpse into these topics and should be reviewed.

Additional Reading

* <https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model>
* <https://css-tricks.com/the-css-box-model/>

**Positioning with CSS**

There are numerous ways to position HTML elements using CSS. Position, float, flexbox, grid, and other properties allow us to place our elements, but each one has its strengths and weaknesses. I will explain the position and float properties and provide links to some of the other ones as well. For this course, most of you will probably use the float property since position is harder to implement for a site with more content. Flexbox and grid are also great methods and were more recently implemented in the CSS specs. They can accomplish more things but can be complicated in some instances.

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. 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. 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 and apply the styles as depicted. The example link provided below demonstrates how to do this with multiple rows of content.

Position Property

This property allows us to alter the location of an element in a web page. The six values to consider are static, relative, absolute, fixed, sticky, and inherit. By combining these values with the top, right, bottom, left, and z-index properties, we can move items around and layer them as needed.

By default, every element has a static position and so top, left, etc. will not work on them. When you give an element a position value of relative, those other properties can be manipulated. Absolute positioning will remove an element from the flow of the document. The positioning of these elements would be relative to other containers that have a “relative” position value. Unlike floats, the other content will behave as if the absolutely positioned content doesn’t exist. Fixed does the same thing as absolute except the content is always relative to the viewport or screen. Sticky is experimental, and I wouldn’t recommend using it for the time being. It behaves similarly to fixed positioning though.

Additional Reading

* <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) - I’m normally against using w3schools since there are inaccuracies sometimes but this tutorial is solid. Exercise caution when using this website.

**Class Examples:**

<https://iinf362.000webhostapp.com/examples/box-model-and-floats/>

The link above points to the directory which has the html file and the css folder. The link below points to the html page so you can see it in action.

<https://iinf362.000webhostapp.com/examples/box-model-and-floats/box-model-float-example.html>

**If you haven’t done so already, please read the “Creating and Viewing our Web Pages.docx” file on Blackboard. You will not be able to submit anything for the assignment without completing that portion first.**

**CSS Positioning Exercises**

Download the “CSS-Positioning-Exercises.zip” folder from Blackboard under the Lecture Notes for this week or in the Assignments folder. Inside of this zip folder, there will be two folders. One will be titled “black-white-page” and the other will be “rainbow-page.” In each of these folders, you’ll find HTML/CSS files and an associated image. Using the HTML/CSS provided (or your own CSS file), try to recreate the styles depicted in the image for each page. Your page doesn’t have to look **exactly** like my image but try to get as close as reasonably possible. I’ve listed some tips below for each page to help you with styles that may not be obvious. Please note that these are not exactly requirements but will help your get your page to look like the image provided. Each page is worth 10pts for a total of 20pts for this assignment.

General Tips

* 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 my CSS file
  + It has code so you don’t have to perform calculations for width (box-sizing: border-box which is mentioned in the reading)
  + The “.row” class declarations will clear the floats for you

Black/White Page

* The width of the left column of content is 30%
* The width of the right column (main content) is 65%
* The “product” divs have a width of 33%

Rainbow Page

* The divs are equally divided in each section (halves, one-third, and one-quarter but in percentages)
* The boxes inside of the red containers have a height of 100px
* The height of the blue boxes is 150px
* The width/height of the white box inside of the blue boxes are 50px
* The colors used specifically are yellow, red, pink, purple, green, blue, black, and white. I would recommend using the color names for this assignment instead of hexadecimal or RGB values (keep it simple for this week – best practice is to use hexadecimal)
* Pay attention to borders (1px solid black mostly)
* You can float and give borders to most divs in one declaration (isn’t necessary but might be a nice exercise in reducing your total code)

To receive credit for this assignment, please submit **two** links to your pages as a comment in the assignment submission area. You can find the assignment under the Lecture Notes for this week or in the Assignments folder. One link should direct me to your version of the black-white page and the other should point to your rainbow page. Please make sure your CSS is externally placed. The links will be due Wednesday, February 26th at midnight.

**CSS Evaluation Initial Post & Response**

Read the following links about common mistakes made with CSS and general CSS best practices. Your task is to evaluate two websites based on the info provided. Do the websites make any of those mistakes? Do you think the mistakes matter given the context of the website? What best practices, if any, are followed? How would you improve the CSS?

Discussion Links

* <https://www.webfx.com/blog/web-design/12-common-css-mistakes-web-developers-make/>
* <https://1stwebdesigner.com/css-best-practices/> (scroll down a bit first)
* <http://www.webhostingsearch.com/articles/the-7-most-common-css-mistakes.php>

In your **initial post**, evaluate two websites based on the questions and links above. You may also add any other commentary as you’d like but try to support your assessment with things mentioned in the links or class notes. Please include links to both websites you decide to evaluate. This initial post is due Sunday, February 23rd at midnight.

In your **response post**, visit one website another student has mentioned in their initial post and look at the CSS (must be different than the websites you initially posted). Do you agree with the assessment they made? Would you make the same changes or different ones? Try to back up your assessment based on practices mentioned in the readings. In your response post, make sure to mention the specific website you decided to check. This response post is due Wednesday, February 26th at midnight.

## Next Week

Next week, I will release a quiz that will be due by the end of the week. It will consist of multiple choice and/or short answer questions. Also, we will look at more CSS, but it will be regarding responsive design. If you’d like your website to work across different devices, this is how you can do it.

1. <https://www.pewresearch.org/internet/fact-sheet/mobile/>
2. <https://developers.google.com/web/fundamentals/design-and-ux/responsive>
3. <https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries>