Cascading Style Sheets for layout
CSS Positioning Basics

- Clarify your content before you design your page
  - If you design a beautiful page and don't have the content to fill it, you'll have wasted your time.
  - Make sure you know what types of pages and content you'll be building for before you design.
CSS Positioning Basics

- **Put your content in blocks**

  - Even if you don't want your Web page to be "blocky" you should think of your content as blocks or units that can be moved around.

  - Once you know that you have, say, a navigation block, a content block, a footer block, and a header block, you can write your HTML to position them.
CSS Positioning Basics

- Place your content blocks in `<div>` elements

  Use `<div>` elements because they are exactly what they say they are - divisions of your Web page content.
CSS Positioning Basics

- **ID your unique divisions using** `<div id="idname">`

  - If you're only going to have one content division, then that `<div>` element should have an id of "content".

  - Remember that the id attribute must be unique on the page, so only use it for elements where there is only one.
CSS Positioning Basics

- **Classify your non-unique `<div>` elements using `<div class="classname">`**

  - You tend to use IDs for the layout portions of your site, because, generally, you can define layouts with unique names for each section.

  - However, if you have, for instance, two pull-quotes on a page, you can't use the `id="pullquote"` for them, you have to use a `class="pullquote"`. 
Starting to Position with CSS

- Once you have a series of `<div>` tags in your HTML you can start positioning them on your page.

- There are many ways to do CSS positioning, but using the `float` property can be the easiest to manage.
Understanding CSS Float

- The CSS `float` property is a very important property for layout. It allows you to position your Web page designs exactly as you want them to display.

- A CSS `float` property looks like this:

  ```css
  .right {float: right;}
  ```
What Floats?

- You can't float every element on a Web page. To get technical, you can only float **block-level** elements.

- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can). Examples of block-level elements: `<div>`, `<h1>` - `<h6>`, `<p>`, `<form>`

- An inline element does not start on a new line and only takes up as much width as necessary. Examples of inline elements: `<span>`, `<a>`,
Where Do They Float?

- You can float elements to the right or the left.

- Any element that follows the floated element will flow around the floated element on the other side.

- E.g,
  - If I float an image to the left, any text or other elements following it will flow around it to the right.
  - If I float an image to the right, any text or other elements following it will flow around it to the left.
  - An image that is placed in a block of text without any float style applied to it will display as the browser is set to display images. This is usually with the first line of following text displayed at the bottom of the image.
How Far Will They Float?

- An element that has been floated will move as far to the left or right of the container element as it can.

- This results in several different situations depending upon how your code is written.

- Lets look at floating a small `<div>` to the left:
How Far Will They Float?

- If the floated element does not have a pre-defined width, it will take up as much horizontal space as required and available, regardless of the float.

- If the container element is the HTML <body>, the floated div will sit on the left margin of the page.

- If the container element is itself contained by something else, the floated div will sit on the left margin of the container.
How Far Will They Float?

- Floated elements will sit next to each other if there is room in the container. For example, this container has 3 100px wide divs floated in a 400px wide container.
Turning Off the Float

- You turn off the float with the CSS `clear` property. You can clear left floats, right floats or both:

  `clear: left; clear: right; clear: both;`

- Any element that you set the `clear` property for will appear below an element that is floated in that direction.
Turning Off the Float

- You can play with the `clear` value of different elements in your documents to get different layout effects.

- One of the most interesting floated layouts is a series of images down the right or left column next to paragraphs of text.

- Even if the text is not long enough to scroll past the image, you can use the `clear` on all the images to make sure that they appear in the column rather than next to the previous image.
Using Floats for Layout

- Once you understand how the float property works, you can start using it to lay out your Web pages.
  - Design the layout (on paper or in a graphics tool).
  - Determine where the site divisions are going to be.
  - Determine the widths of the various containers and the elements within them.
  - Float everything.

- As long as you know the widths (percentages are fine) of your layout sections, you can use the float property to put them where they belong on the page.
What are Floats used for?

- Aside from the simple example of wrapping text around images, floats can be used to create entire web layouts.
Clearing the Float for Layout

- An element that has the clear property set on it will not move up adjacent to the float like the float desires, but will move itself down past the float.

- Here, the sidebar is floated to the right and is shorter than the main content area. The footer then is required to jump up into that available space as is required by the float.
Clearing the Float for Layout

To fix this problem, the footer can be cleared to ensure it stays beneath both floated columns.

```css
#footer {
    clear: both;
}
```
1. Set a width on your page by creating a container
   - You don't have to do this, but most pages are easier to read if you don't assume that everyone will have their browser set to the same settings as your browser.
   - A lot of people design for 1024x768 resolution, with a typical width of around 960px.
   - However, you can also do fluid widths using percentages.

```css
#container {
  width: 960px;
  background: #FFF;
  margin: 0 auto; /* the auto value on the sides, coupled with the width, centres the layout */
}
```
2. Float everything

- Once you have your maximum width, then you can float everything on the page, and have it line up.

- E.g. if you want your navigation `<div>` to be at the top of the page, you would make it have a width of 100% and float left.
  
  ```css
  #nav { width: 100%; float: left; }
  ```

- But if you wanted it to be on the right side, you'd make it a width of less than 100% and float right.
  
  ```css
  #nav { width: 200px; float: right; }
  ```

- Then, anything that comes after it would be floated left, and as long as those elements had a smaller width than 600px (800 - 200), they would slide right in to the left of the navigation.
3. Use floats to create margins
   
   One of the great things about floats is that you can use them to create margins without using CSS.

   E.g., if my navigation is on the right and 240px wide and my content area is floated left and 700px wide, there will be a 20px margin between the two elements, without any margin tags at all.

   ```css
   #nav { width: 240px; float: right; }
   #content { width: 700px; float: left; }
   ```
4. Get multiple columns by nesting `<div>` elements

   If you want three columns, you create two divs that float left and right, and then in the wider div, you create a second set of two columns that float left and right inside it.

   HTML:
   ```html
   <div id="leftside">
     <div id="leftcolumn">left column</div>
     <div id="centrecolumn">centre column</div>
   </div>
   <div id="rightside">right column</div>
   ```
Starting to Position with CSS

- CSS (note that the inner divs have a width of 50% because they are half of the outer container, which is the "leftside" div:

```css
#leftside { width: 66%; float: left; }
#rightside { width: 33%; float: right; }
#leftcolumn { width: 50%; float: left; }
#centercolumn { width: 50%; float: right; }
```
Starting to Position with CSS

5. Test in multiple browsers
   - While this technique works most of the time, some browsers react strangely to floats.
   - You may have to play with the widths to get your elements to show up correctly.