**CSS Basics #5: Some Advanced Topics -** [Eric Hu](https://huericnan.medium.com/?source=post_page-----d7281aa61714--------------------------------) Feb 8, 2022

**Float**

The float property is another CSS property used for positioning HTML elements. It specifies how an element should "float". For example, an image could float left or right to the text in a container.

*You can download the source code for this tutorial here:*

[*Download the Source Code*](https://www.ericsdevblog.com/index.php/beginners-roadmap-to-web-development/)

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

**No Float**

<div>  
 <img src="image.png" />  
 ...  
</div>img {  
 float: none;  
}

Text, letter

Description automatically generated

**Float Left**

img {  
 float: left;  
}

Text

Description automatically generated

**Float Right**

img {  
 float: right;  
}

Text

Description automatically generated

**Float Next to Each Other**

By default, two <div> elements will be displayed on top of each other. However, it is possible to make them float next to each other using float: left.

<div class="div1">Div #1</div>  
<div class="div2">Div #2</div>.div1 {  
 float: left;  
 background-color: red;  
 padding: 15px;  
}.div2 {  
 float: left;  
 background-color: green;  
 padding: 15px;  
}



**Transform**

The Transform property allows you to move, rotate, scale, and skew elements. It allows us to use the following transformation methods:

**The translate() Method**

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

div {  
 transform: translate(50px, 100px);  
}

For example, this code will move the <div> element 50px to the right, and 100px down from its current position.

**The rotate() Method**

The rotate() method rotates an element clockwise (positive) or counter-clockwise (negative) according to a given degree.

div {  
 transform: rotate(20deg);  
}section {  
 transform: rotate(-20deg);  
}

The <div> element will rotate 20 degrees clockwise, and the <section> element will rotate 20 degrees counter-clockwise.

**The scale() Method**

The scale() method increases or decreases the size of an element.

div {  
 transform: scale(2, 3);  
}

The first parameter corresponds to the width and the second parameter corresponds to the height. The above code increases the <div> element to be two times its original width, and three times its original height.

**The skew() Method**

The skew() method skews an element along the X and Y-axis by the given angles.

div {  
 transform: skew(20deg, 10deg);  
}

A close-up of a note

Description automatically generated with low confidence

**The matrix() Method**

The matrix() method is the combination of all of the above methods. It takes six parameters, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY()), where X means around the horizontal axis and Y means around the vertical axis.

div {  
 transform: matrix(1, -0.3, 0, 1, 0, 0);  
}

**Flexbox**

The flexbox is another layout module that makes it easier for us to design a responsive layout without using float or positioning. Just like the grid system, each flexbox has a flex container and several flex items.

<div class="flex-container">  
 <div>1</div>  
 <div>2</div>  
 <div>3</div>  
</div>

This <div> block becomes a flexbox (becomes flexible) by setting the display property to flex:

.flex-container {  
 display: flex;  
}

A picture containing shape

Description automatically generated

**The Flex Container**

**The flex-direction Property**

The flex-direction property defines in which direction the container wants to stack the flex items. The default value is row, which makes the items stack from left to right.

We can also stake them vertically:

.flex-container {  
 display: flex;  
 flex-direction: column;  
}

Chart

Description automatically generated with medium confidence

Or in the reversed order:

.flex-container {  
 display: flex;  
 flex-direction: column-reverse;  
}

Chart, treemap chart

Description automatically generated with medium confidence

**The flex-wrap Property**

The flex-wrap property specifies whether or not the flex items should wrap (automatically put items in a new row when there is not enough space).

When flex-wrap is set to wrap:

Calendar

Description automatically generated

When flex-wrap is set to nowrap:

A screenshot of a game

Description automatically generated with low confidence

**The Alignment Properties**

Just like the grid system, there are also alignment properties in the flexbox, and they work exactly the same. The justify-content property is used to vertically align the flex items and the align-items property is used to horizontally align the flex items.

**The Flex Item**

**The order Property**

The order property specifies the order of the flex items.

<div class="flex-container">  
 <div style="order: 3">1</div>  
 <div style="order: 2">2</div>  
 <div style="order: 4">3</div>  
 <div style="order: 1">4</div>  
</div>

Icon

Description automatically generated

**The flex-grow Property**

When the flex-grow property is set, the flex items will always stretch to the edge of the viewport as you resize your browser, and the number indicates how much faster the item grows relative to the other items in the flexbox.

<div class="flex-container">  
 <div style="flex-grow: 1">1</div>  
 <div style="flex-grow: 1">2</div>  
 <div style="flex-grow: 8">3</div>  
</div>

Icon

Description automatically generated

**The flex-basis Property**

The flex-basis property specifies the initial length of a flex item.

<div class="flex-container">  
 <div>1</div>  
 <div>2</div>  
 <div style="flex-basis: 200px">3</div>  
 <div>4</div>  
</div>

Shape

Description automatically generated with medium confidence

**The object-fit & object-position Property**

The object-fit property defines how an image will be displayed inside a container.

For example, you are designing a website, you want the users to upload images as their profile pictures. The profile pictures will be displayed as small squares, but the uploaded images come in different sizes. Simply setting the width and height of the image will result in a stretched and strange-looking profile picture. How can we solve this problem?

Here we have an image of a cat, and we can resize it into a 200px by 200px image using CSS:

img {  
 width: 200px;  
 height: 200px;  
}

And as you can see, the image has been stretched and it doesn’t look very good.

Here is where the object-fit property comes in. The object-fit property can take one of the following values:

* fill - This is default. The image is resized to fill the given dimension. If necessary, the image will be stretched or squished to fit
* contain - The image keeps its aspect ratio, but is resized to fit within the given dimension
* cover - The image keeps its aspect ratio and fills the given dimension. The image will be clipped to fit
* none - The image is not resized
* scale-down - the image is scaled down to the smallest version of none or contain

**Using object-fit: cover;**

**Using object-fit: contain;**

**Using object-fit: none;**

**Using object-fit: scale-down;**

If we want to use this image as a profile picture, the best option will be using object-fit: cover;.

However, now we face another problem, this option zooms in on the centre of the image to fill the container, and it clips a part of the cat. What if we want to zoom in on the cat?

The object-position property can help us with this problem. It is used to specify how an <img> or <video> should be positioned within its container.

img {  
 width: 200px;  
 height: 200px;  
 object-fit: cover;  
 object-position: 100% 100%;  
}

The first parameter determines the position on the x-axis, and the second parameter determines the position on the y-axis.

**Units**

Finally, let’s talk about units in CSS. There are two types of length units: absolute and relative.

**Absolute Lengths**

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

**Relative Lengths**

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering mediums.