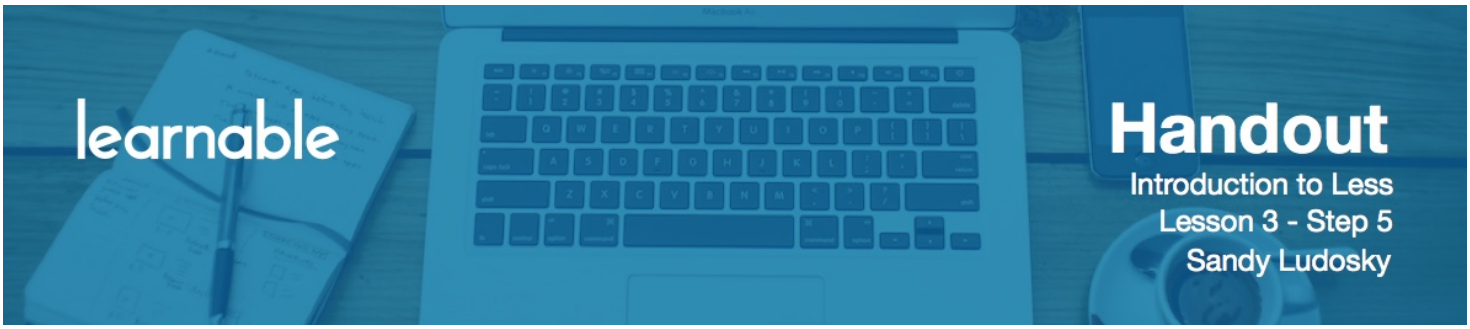

Getting Started

At this step we are going to discuss **parametrized mixins**.

We're going to work with the main section of the page. We have an image and some content, a paragraph. What we're going to do is increase the font size of the content, then float content around the image. And finally we're going to add some rounded corners to the image.

Add some margin to the `container` and increase the size of the paragraph:

```
.container {
    margin: @margin;
    p {
        font-size: @fontSize;
    }
}
```

Now we're going to float the text around the image and add some margin:

```
.container {
    margin: @margin;
    p {
        font-size: @fontSize;
        img {
            float: left;
            margin: @margin;
        }
    }
}
```

Now let's talk about rounded corners that I'd like to apply to this image. In order to achieve that I'm going to be using the property `border-radius` which became available with CSS3. In order for these features to work across browsers, we need to use what we call the **CSS Vendor prefixes**, that are browser specific version of the properties to make sure they work across browsers.

Parameterized Mixins : How it works?

For example we're going to write a mixin that we're going to name `radius` which will take a parameter:

```
.radius(@radius) {
    -webkit-border-radius: @radius;
    -moz-border-radius: @radius;
    -o-border-radius: @radius;
    -ms-border-radius: @radius;
    border-radius: @radius;
}
```

This mixin includes browser specific properties. The variable that has been passed in as a parameter will be applied as a value for each of the browser specific properties.

Now use this mixin:

```
img {
    .radius(10px);
    float: left;
    margin: @margin;
}
```

We're passing a value as an argument, and as a result the value `10px` will be applied to each of the browser-specific property. And this is one good example for which you'd use the parametized mixins, because everytime you'd want to use the CSS3 features you need to make sure that you apply the browser specific properties. It really saves you the trouble to have to write multiple lines of code every time you want to work with the CSS3 features.

Parameterized Mixins with default

Mixin's arguments may take default values. Let's use exactly same example of `border-radius`:

```
.rounded(@rounded:50%) {
    -webkit-border-radius: @rounded;
    -moz-border-radius: @rounded;
    -o-border-radius: @rounded;
    -ms-border-radius: @rounded;
    border-radius: @rounded;
}
```

We specify a default value next to the variable that we're using as a parameter. In that case you don't have to specify any value when using mixin:

```
img {
    .rounded;
    float: left;
    margin: @margin;
}
```

So in order to save time and to write LESS code, you can define a mixin with parameters, then use it multiple times as an object in your LESS code. Then the LESS compiler will take care of generating the required browser specific version for each of the property, in order for your design to work across browsers.