MODULE 01 - 013: Creating a @mixin

Simplify your SCSS code with **mixins**, reusable blocks of styles that can be called anywhere in your stylesheet.

Here's how to define a mixin:

```
@mixin theMixinName {
  // Styles go here
}
```

And this is how to `@include` it where needed:

```
.iAmAClass {
  @include theMixinName;
  font-family: Verdana;
  ....
}
```

Step-by-step example

Styling a `featured` section with links and :hoover

```
@mixin featured {
  color: Tomato;

  .subheading a {
    color: LightSteelBlue;
    text-decoration: none;

    &:hover {
      color: LightSkyBlue;
      text-decoration: underline;
    }
  }
}
```

Including a @mixin A @mixin can be included in any selector by using `@include`:

```
@include mixin-name;
```

And the, applying the `@mixin feature` to specific sections:

```
.page-wrapper {
  .featured {
    @include featured;
  }

  .sidebar {
    @include featured;
  }
}
```

Combining `@mixin`'s with Unique styles **Mixins can serve as a base; then, unique styles for each section are added:

```
.sidebar {
  @include featured;
  font-family: Verdana;
  text-align: right;
  float: right;
  width: 25%;
}
```

Final SCSS code Finally, the complete example combining the `@mixin`with the other styles, here:

```
$off-white: #f6f6f6;
$featured-color: DarkRed;

@mixin featured {
  color: Tomato;

  .subheading a {
    color: LightSteelBlue;
    text-decoration: none;

    &:hover {
      color: LightSkyBlue;
      text-decoration: underline;
    }
  }
}

body {
  background-color: $off-white;
  height: 100vh;
  height: 100vw;
}

.container {
  font-family: Verdana;
  font-size: 0.8rem;
}

.page-wrapper {
  padding: 21px;
  $featured-color: RoyalBlue;

  .featured {
    @include featured;
  }

  .page-content {
    background-color: $featured-color;
    padding: 42px;
    color: $off-white;

    .container {
      height: 60px !important;
      font-family: courier;

      .description {
        float: left;
        width: 75%;
      }

      .sidebar {
        font-family: Verdana;
        text-align: right;
        float: right;
        width: 25%;
        @include featured;
      }
    }
  }
}
```

Video lesson Speech

[ENG] # Creating Your First Mixin This lesson examines how you can build a basic Mixin that can be called from anywhere in the application's Scss files.

I've updated the starter code that we have here and as you can see we now have a few different components.
Starter HTML Code

```
<div class="container">
  <div class="page-wrapper">
    <div class="featured">
      <h1>About us</h1>

      <div class="subheading">
        <h4>
          <a href="#">My subheading</a>
        </h4>
      </div>
    </div>

    <div class="page-content">
      <div class="container">
        <div class="description">
          <p>
            Cras mattis consectetur purus sit amet fermentum. Aenean eu leo quam. Pellentesque ornare sem lac
          </p>
        </div>

        <div class="sidebar">
          <h1>About us</h1>

          <div class="subheading">
            <h4>
              <a href="#">My subheading</a>
            </h4>
          </div>
        </div>

      </div>
    </div>
  </div>
</div>
```

Scss Starter Code

```
$off-white: #f6f6f6;
$featured-color: DarkRed;

body {
  background-color: $off-white;
  height: 100vh;
  height: 100vw;
}

.container {
  font-family: Verdana;
  font-size: 0.8rem;
}

.page-wrapper {
  padding: 21px;
  $featured-color: RoyalBlue;

  .featured {
    color: Tomato;
    .subheading a {
      color: LightSteelBlue;
      text-decoration: none;
      &:hover {
        color: LightSkyBlue;
        text-decoration: underline;
      }
    }
  }

  .page-content {
    background-color: $featured-color;
    padding: 42px;
    color: $off-white;

    .container {
      height: 60px !important;
      font-family: courier;

      .description {
        float: left;
        width: 75%;
      }

      .sidebar {
        font-family: Verdana;
        text-align: right;
        float: right;
        width: 25%;
        color: Tomato;
        .subheading a {
          color: LightSteelBlue;
          text-decoration: none;
          &:hover {
            color: LightSkyBlue;
            text-decoration: underline;
          }
        }
      }
    }
  }
}
```

If you look on the HTML side I've added a description, also I've added a sidebar div and class. And as you may have noticed it is literally identical to what we have right here. So something that you'll come across quite a bit as you're developing applications, is when you have identical styles or very very close to the same style that needs to be placed throughout an entire application and that is where we get into how we can leverage mixins.

We've talked about how variables are excellent at helping in cleaning up duplication but variables can only go so far when you need to be able to store a large amount of items or when you need to have dynamic behavior. That is where mixins really start to show how helpful they are. It's a huge reason why SCSS has become so popular through the years because it lets you wrap up functionality that you use quite a bit and call it from anywhere in the application. If you're used to working with methods or functions in a programming language you can think about SCSS mixins very much in the same way.

So let's look at the code we have here. So we have two components that are completely identical and one is in the sidebar. The other is there in the featured section. If you come inside of the nested feature class you can see we have a color of tomato and a nested subheading link with its own respective hover state. Now if you come down all the way down to the bottom you can see we have a sidebar class and inside of the sidebar class we're calling things such as the font family, text-align right float, and all of these kinds of components. So what we have here is identical but we do have a few things that we've added on. So this is going to show how mixins can be called. So what we're going to do is come into the featured section and I'm going to cut all of it out. And as you can see this will change just to the basic defaults. At the very top of the file, we are going to create a mixin and I'm going to call it featured and just leave it at that. And inside a set of curly braces, I'm going to paste in all of those styles.

Now in order to code this all I have to do is call it like this.

So now we can make changes in our mixin and they are reflected where ever we have used them. Let's see how we can do the same thing down below for our Sidebar. So even though our Sidebar has a few items that are different and unique we can still just call these mixins and the system will do the remainder of the work for us.

I personally like to think about mixins like a method where they can be called from anywhere in the application. I also liken them to very powerful variables, kind of like the way where we made a change to just one value of a variable at the very beginning and it populated to the rest of the application. Same thing with mixins. Now, this is just one part of mix-ins and this is very helpful to create reusable code components that can be called from anywhere else in any other SCSS file and you can just reuse those without having to copy and paste a full set of style definitions.

Code with Mixins

```scss
$off-white: #f6f6f6;
$featured-color: DarkRed;

@mixin featured {
  color: Tomato;
  .subheading a {
    color: LightSteelBlue;
    text-decoration: none;
    &:hover {
      color: LightSkyBlue;
      text-decoration: underline;
    }
  }
}

body {
  background-color: $off-white;
  height: 100vh;
  height: 100vw;
}

.container {
  font-family: Verdana;
  font-size: 0.8rem;
}

.page-wrapper {
  padding: 21px;
  $featured-color: RoyalBlue;

  .featured {
    @include featured;
  }

  .page-content {
    background-color: $featured-color;
    padding: 42px;
    color: $off-white;

    .container {
      height: 60px !important;
      font-family: courier;

      .description {
        float: left;
        width: 75%;
      }

      .sidebar {
        font-family: Verdana;
        text-align: right;
        float: right;
        width: 25%;
        @include featured;
      }
    }
  }
}
```

[SPA]

La idea general de los mixins, y cómo incluirlos, es brillante.

Simplemente, creamos y nombramos nuestro mixin, en primer lugar, con el método “@mixin”.

Después, lo vamos añadiendo allí donde sea necesario, a través del método “@include”. Un mixin puede añadirse en cualquier selector y servir de base, ahorrando mucho código y tiempo.