MODULE 1 - 015: SCSS Conditionals (1) IF - ELSE

This guide explains how to use conditional logic in SCSS to dynamically adjust styles based on arguments and conditions.

Why using SCSS Conditionals?

Conditionals in SCSS allow you to:
- Dynamically control styles based on provided arguments.
- Handle different style scenarios with clean, scalable and reusable code.
- Ensure robust styling logic by capturing edge cases. - Prevent undesired effects.

SCSS uses regular conditional expressions, but with its own syntax (basically, adding @):
- `@if` for primary conditions. - `@else if` for additional conditions. - `@else` as a fallback.

IF - ELSE

Cases

- Allows multiple style scenarios.
- Handles unexpected arguments / Prevents for undesired behaviors.

Using `@if`, `@else if`, and `@else`

1 Basic Conditional with `@if`

Define styles based on a condition:

```scss
@mixin featured($bg-color: 'dark') {
  @if $bg-color == 'light' {
    color: Tomato;
  }
}

.featured {
  @include featured('light');
}
```

Here, the logical steps are:

- If `$bg-color` is 'light',then, the text color will be `Tomato`.
- If no value are passed, it defaults to `Dark`.

2 Expanding Conditions with `@else if` and `@else`

This can handle multiple use cases:

```scss
@mixin featured($bg-color: 'dark') {
  @if $bg-color == 'light' {
    color: Tomato;
  } @else if $bg-color == 'dark' {
    color: Blue;
  } @else {
    color: Red;
  }
}

.featured {
```

```
  @include featured('blue');
}
```

Now, the logical steps become:
* `@if` + `$bg-color == 'light'`: Sets color to Tomato.
  - `@else if` + `$bg-color == 'dark'`: Sets color to Blue.
  - Finally, `@else`: Defaults to Red.

Final code with Conditionals included

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

@mixin featured($bg-color: 'dark') {
  @if $bg-color == 'light' {
    color: Tomato;
  } @else if $bg-color == 'dark' {
    color: Blue;
  } @else {
    color: Red;
  }

  .subheading a {
    color: black;
    text-decoration: none;
    &:hover {
      color: black;
      text-decoration: underline;
    }
  }
}

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

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

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

  .featured {
    @include featured('light');
  }

  .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('blue');
      }
    }
  }
}
```

Video lesson Speech

[ENG]
MODULE 01 - 015: SCSS Conditionals This lesson walks through how to implement conditionals in Scss to dynamically change how styles are rendered on the screen.
***

In the last guide, we walk through how to pass arguments into mixins And that's incredibly powerful.
In this guide, we're going to extend that knowledge and we're going to talk about conditionals.

You know if you're coming from a general-purpose programming language like Ruby or javascript then the concept of conditionals may be relatively familiar to you. It gives you the ability to have a situation where you can ask if something happens then I want you to perform one set of actions if something else happens though I want you to change your behavior and do something different.

And Scss allows us to have that same type of flexibility and the same type of power.
Right. In the system. This is a very powerful way of being able to dynamically change how mixins behave.

Now I've taken the code that I put in last time out for the most part. And as you can see right here I've just hardcoded the link colors in and I'm going to leave those there for right now because I don't want to be distracted on what I want to show which is how to change these heading colors and to make them dynamic, based on the background color that they're on.

This is something that is a very common practice where you run into a situation where you have a certain number of styles and say 90 percent of them will work for many different situations and so you want to be able to share those styles. But there are a few things that need to change.

So great example is let's say we have this featured mix in here we want just about all of these things to be the same.

The only thing that we want to be different is we want this color and obviously we have a few other items that can be different but for the sake of this example, we want it to be just the color that we don't want to pass in the color the way we did in the last guide because there may be times where we simply do not want to make something that arbitrary.

Especially say that you're building some type of code library that other people are going to use.

If you just allow them to pass any color and it might break the style guide and there might be some issues. Instead what you can do is you can pass in some other types of values or you can require some other values to be passed in.

So what we're going to do here is I'm going to add a new argument here and it's going to be called `$bg-color: 'dark'` and I'm going to pass it in so it has dark as a default to remember from the last guide.

This is the syntax for setting a default value for an argument in Scss.
And now the syntax for just a basic conditional is sane.

`If` with an AT (@) symbol in front of it.
Now **if you're coming from another language that may look weird but this is a way that the preprocessor with Scss works.**

So we're going to say if `$bg-color` is equal to dark and remember to do two equal signs then I want to change this color so I want to change the color of this tomato right here.
This is going to give us exactly the same behavior.

Now, why is this giving us this behavior as a quick review of how the arguments into arguments work. We scroll down right now.

We're not calling or passing any arguments into this featured mixin call either here or here.
And because of that with the way to fall arguments work that doesn't mean it's always going to be dark.

Now if we say that if the background color is light then right then it's going to just default to whatever the value is. So right now this one changes to light and this one changes to dark and that is just because of how the other types of settings are there.

So we have a subheading with a and we have some other elements on the rest of the page.
So inside of the container right here we have our color of being off-white in this page content wrapper.

So that's a reason why this about us is in white.
So we definitely do not want that to happen.
So what we need to do is make sure we catch each one of the scenarios.
So I say if the background color is light then I want to do tomato coming down to a featured call right here I'm going to pass in light.
And so this one is now going to be tomato. You see how that works it just changed right there because we passed in an argument that matched exactly with what our conditional said.

Now here for `featured`. I'm going to pass it and say this one has a dark background and it goes back to whatever the default is.

And coming up what we can do is there's a number of ways we can implement it conditional.

I'm going to show you one way of doing it. And that is if we want to just capture certain cases so say that we want to say OK if the background color is a light then I want to be tomato.
If it's dark then I just want it to be white just like this or I guess to make it a little bit easier we could go with because this was already the default. So you can see that it is changing.
We could go with one of our blues or something like that.

This is not a good UI/UX, this is more just to show you how to change these colors.
So that's one way of doing it. And that's fine for certain circumstances. And in fact when we go into our more advanced project and we build out an entire mix in the module just for flexbox then I am going to do this because I like to in certain cases organize each one of my scenarios into its own conditional.
That's one way.

But there's also an alternative syntax for more straightforward conditionals and that is by using **ELSE IF**.

ELSE - IF

So here what I can do is actually bring this up top and say else if and it's two words say.
Else if the background color is dark then I want you to change the color to blue. And as you can see everything here remained exactly the same.
So we can combine both the conditionals into one.

Now let's do one more because this is fine but what happens if somebody passes in a completely different color as the background. Well, we can set up another default. We have a full set up here but we could also set up another one that says any other scenario we can pass in. And so here we can change it to red or something like that. And now if I come all the way down to the bottom if someone calls this mix in and they call it and say oh it has a blue background you can see it goes to that final conditional it goes to that final else. This is a smart way of building your system because you don't want to run into a situation where you haven't captured all of the different scenarios that could occur.

So we've we have ourselves covered in two ways.
* One we make it possible to call this feature mix and without any arguments and we set up just a regular default. This would be a scenario where you know in a very large percentage of the situations where this mix is going to be called it's

going to be on a dark background so it's perfectly fine to set this up as the default value.
* But you also want to make sure that somebody who maybe they have a typo maybe they have some type of issue with how they call it. You want to make sure that you have some shrewd defaults so if they call anything besides this dark if they call different than light or dark then we're still capturing that and everything still is working properly.

So this is how you implement a just a regular conditional along with a compound conditional inside of Scss:

HTML Starter 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;

@mixin featured {
  color: Tomato;
  .subheading a {
    color: black;
    text-decoration: none;
    &:hover {
      color: black;
      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;
      }
    }
  }
}
```

Scss Final Code, improved with Conditionals

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

@mixin featured($bg-color: 'dark') {
  @if $bg-color == 'light' {
    color: Tomato;
  } @else if $bg-color == 'dark' {
    color: blue;
  } @else {
    color: red;
  }
```

```
  .subheading a {
    color: black;
    text-decoration: none;
    &:hover {
      color: black;
      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('light');
  }

  .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('blue');
      }
    }
  }
}
```

[SPA]

Usar condicionales en nuestro desarrollo tiene múltiples ventajas: * Eficiencia y escalabilidad, en tiempo de desarrollo y mantenimiento.

* Seguridad y manejo de errores (previene comportamientos inesperados).
* Alcance absoluto (Permite cubrir cualquier caso).

Usamos condicionales IF - ELSE en los siguientes casos:
- `@if` para primeros condicionales. - `@else if` para cada uno de los posibles casos esperados. - `@else` para el resto de situaciones.

Esto no difiere en absoluto respecto al uso de condicionales en cualquier otro tipo de lenguaje, excepto las especificaciones concretas de sintaxis propias en SCSS (básicamente, una arroba antepuesta sobre cada operador condicional).