

MODULE 01 - 016 + 017: SCSS Task 1 - FlexBox Hater

This is a real use where conditionals and mixins makes the great difference when coding.
Try the challenge using CodePen, or your HTML/SCSS favourite IDE. CodePen Try

HTML Project Starter Code

```
<div class='container'>
  <div class='item'>
    <div class='content'>
      <img
        src='https://d32xj74kbqkoqn.cloudfront.net/uploads/campsite/campsite_image/129/dissecting-rails-icon.'
        width='200px'
        height='120px'
      >
      <div class='metadata'>
        <h1 class='title'>My amazing title</h1>
        <h3>And a subtitle</h3>
      </div>
    </div>
    <div class='btn-group'>
      <div class='button'>
        <a href="#">+</a>
      </div>
      <div class='button'>
        <a href="#">x</a>
      </div>
    </div>
  </div>
  <div class='item'>
    <div class='content'>
      <img
        src='https://d32xj74kbqkoqn.cloudfront.net/uploads/guide/video_image/43/foundations-video-thumb.jpg'
        width='200px'
        height='120px'
      >
      <div class='metadata'>
        <h1 class='title'>Another amazing post</h1>
      </div>
    </div>
    <div class='btn-group'>
      <div class='button'>
        <a href="#">+</a>
      </div>
      <div class='button'>
        <a href="#">x</a>
      </div>
    </div>
  </div>
</div>
```

SCSS Project Starter Code

```
.container {
  display: flex;
  .item {
    flex: 1;
    display: flex;
    justify-content: space-between;
```

}

Now, how would you implement mixin's and conditionals in order to cleanise / make more efficiente code?

Are this tip so useful for those FlexBox haters? Ha, ha!

are going to describe:

```
@mixin flex-config($justify-content: false, $flex: false, $flex-direction: false, $align-items: false) {
  display: flex;

```

```
justify-content: $justify-content;
}
```

```
flex: $flex;
```

```

}

@if $flex-direction != false {
  flex-direction: $flex-direction;
}

@if $align-items != false {
  align-items: $align-items;
}
}

.container {
  @include flex-config;
  .item {
    @include flex-config($flex: 1, $justify-content: space-between);
    border: 1px solid grey;
    border-radius: 5px;
    margin-bottom: 10px;
    .content {
      @include flex-config($flex-direction: column, $justify-content: center);
      .metadata {
        margin-left: 20px;
        .title {
          margin: 0px;
        }
      }
    }
  }
  .btn-group {
    @include flex-config($align-items: center);
    .button {
      @include flex-config($flex-direction: row, $align-items: center, $justify-content: center);
      height: 100%;
      width: 42px;
      font-size: 2em;
      a {
        color: green;
        text-decoration: none;
      }
      &:hover {
        background-color: maroon;
        cursor: pointer;
        a {
          color: white;
        }
      }
    }
  }
}
}

```

1. Define the Mixin

```

@mixi flex-config($justify-content: false, $flex: false, $flex-direction: false, $align-items: false) {
  display: flex;

  @if $justify-content != false {
    justify-content: $justify-content;
  }
}

```

```

@if $flex != false {
  flex: $flex;
}

@if $flex-direction != false {
  flex-direction: $flex-direction;
}

@if $align-items != false {
  align-items: $align-items;
}
}

```

- Always applies display: flex.
- Conditionally applies justify-content, flex, flex-direction, and align-items (the items usually used, reused and repeated ever and ever) @if their corresponding variables are not false. ##### Basic @mixin usage

```

.container {
  @include flex-config;
}

```

Custom usage Sets flex: 1 and justify-content: space-between.

```

.item {
  @include flex-config($flex: 1, $justify-content: space-between);
}

```

Nested @mixin usage Encapsulates specific flexbox layouts for child elements, improving modularity and code reusability.

```

.content {
  @include flex-config;
  .metadata {
    @include flex-config($flex-direction: column, $justify-content: center);
  }
}

```