Mixin

Mixinを簡単に説明すると、プロパティーの集まりを定義しておきそれを他の場所で呼び出すことが可能です。

Extendと似て分かり辛いですが、明確に違う機能があるので少しずつ理解していきましょう！

Mixinの定義と呼び出し

Extendはセレクターを継承先に書くのに対して、Mixinは定義と呼び出しを行う必要があります。

`@mixin ○○`でMixinの定義を行います。ただこれだと定義してだけで呼び出す必要があります。

その時に使用するのが`@include ○○`です。定義で名付けた名称を指定することで呼び出すことができます。

```
@mixin boxContent {
  width: 300px;
  height: 300px;
}

.main-box {
  @include boxContent;
  background: #ff0000;
}

.sub-box {
  @include boxContent;
  background: #0011ff;
}
```

コンパイルして確認して見ましょう。

Extendでは共通部分はまとめてあったのに対してMixinは別々に書かれています。

これならExtendだけて良いのでは？と思いますが次の説明でExtendとの違いを説明します。

引数を使う

Mixinは引数を使用できます。それによりセレクターによって別々の値を持たせることができます。

```
@mixin boxContent($value) {
  width: $value;
  height: $value;
}

.main-box {
  @include boxContent(300px);
  background: #ff0000;
}

.sub-box {
  @include boxContent(100px);
  background: #0011ff;
}
```

コンパイルして確認して見ましょう。

`width`、`height`が引数で渡された値になっています。

これを使用することで場所によって大きさが違う部分にも簡単に対応できます。

さらに定義する側で初期値を持たせることも可能です。

```
@mixin boxContent($value: 300px) {
  width: $value;
  height: $value;
}

.main-box {
  @include boxContent;
  background: #ff0000;
}

.sub-box {
  @include boxContent(100px);
  background: #0011ff;
}
```

ここからわかることは、引数を持っているからといって必ず渡す必要はありません。

引数を複数指定する

引数はカンマ区切りで複数してすることが可能です。

```
@mixin boxContent($value: 300px, $margin: 0 auto, $padding: 0) {
  width: $value;
  height: $value;
  margin: $margin;
  padding: $padding;
}

.main-box {
  @include boxContent;
  background: #ff0000;
}

.sub-box {
  @include boxContent(100px, 0 50px, 20px);
  background: #0011ff;
}
```

スコープの制限

@mixinもスコープの制限が行えます。

```
.main-box {
  @mixin boxContent($value: 300px, $margin: 0 auto, $padding: 0) {
  width: $value;
  height: $value;
  margin: $margin;
  padding: $padding;
}
  .sub-box {
    @include boxContent(100px, 0 50px, 20px);
    background: #0011ff;
  }
}

// エラーを起こす
// .sub-box {
//   @include boxContent(100px, 0 50px, 20px);
//   background: #0011ff;
// }
```

コンテンツブロックを渡す @content

@includeは定義したものを呼び出すのに対して@contentはmixinに渡す機能です。

渡されたCSSは@contentが書かれた位置で展開されます。

```
@mixin media($media-width: 768px) {
  @media screen and (max-width: $media-width) {
    @content
  }
}

.item {
  image {
    float: left;
    // SP版
    @include media {
      float: none;
    }
  }
  .text {
    overflow: hidden;
    margin-left: 15px;
    // SP版
    @include media {
      margin-left: 0;
    }
  }
}
```

コンパイルしたものを見てください。

メディアクエリに書いている分けじゃ何にもかかわらず中身ができ上がっています。

このように書くことによってCSSで別々に書いていたのをまとめてかけるのでここからはレスポンシブ対応と言うのがすぐ分かります。