Now, that we have grasped the grid concept and coded an example, it's time to convert it in to a mixin. Converting it in to a mixin will save us time and effort in the task to generate different grids for different responsive breakpoints. We combine this with our media query mixin to generate five variations of our grid.

base/_helper_classes.scss:

```
//
//-- Helper Classes
//

.clearfix {
    &::before, &::after {
        content: "";
        display: table;
    }

    &::after {
        clear: both;
    }
}
```

mixins/_mixins_grids.scss:

```
//
//-- Grid Mixins
//

@mixin generate_grid($type, $gutter) {
    %grid-#{$type}--base {
        float: left;
        padding-left: $grid-gutter;
        padding-right: $grid-gutter;
    }

    @for $i from 1 through 12 {
        .grid-#{$type}--#{$i} {
            @extend %grid-#{$type}--base;
            width: (100% / 12) * $i;
        }
    }
```

```
}
```

This `for` loop iterates a temporary variable `i` from 1 to 12, each time, outputting a grid cell class with the desired CSS. This `.grid-#{$type}--#{$i}` is called string interpolation. This is used when we want to put a variable like the type of degree, enter class name like this.

modules/_grids.scss:

```
//
//- Grid Systems
//

$grid-gutter: 16px;

.grid {
    margin-right: ($grid-gutter * -1);
    margin-left: ($grid-gutter * -1);
}

// smallest grid
@include generate_grid(xs, $grid-gutter);

// small grid
@include phone-land {
    @include generate_grid(sm, $grid-gutter);
}

// medium grid
@include tablet-port {
    @include generate_grid(md, $grid-gutter);
}

// large grid
@include tablet-land {
    @include generate_grid(lg, $grid-gutter);
}
```

We have setup an responsive grid system that will be invaluable, when styling the various sections of the page.