بسم الله الرحمن الرحیم

Sass

Sass یک Css-Pre-processor است و باید برای استفاده از آن چیزی را نصب کنیم و نمی توانیم همانند css بدون نصب چیزی استفاده کرد. Sass فایده ی زیادی در استایل دهی دارد و ان هم این است که می توانیم از تکرار جلوگیری کنیم به این معنا که اگر از رنگ مثلا برای یک پس زمینه استفاده کردیم و می خواهیم جاهای دیگر هم استفاده کنیم می توانیم یک متغیری داشته باشیم و هنگام تغییر فقط ان متعیر را تغییر بدهیم.

$bgcolor : #1c78f2;

Body{

Background-color: $bgcolor;

}

Browser ها توانایی فهمیدن sass را ندارند و باید ان ها تبدیل به css شود تا بتوانند متوجه شوند پس از sass-preprocessor ها نیاز داریم تا بتواند ان ها را تبدیل به کد css کند به این عمل تبدیل transoiling می گویند. پسوند فایل sass برابر با .scss است.

برای نصب می توانیم از دستور npm install –g sass استفاده کرد و با حتی می توان از نرم افزار های متنوعی هم برای استفاده از sass استفاده کرد که ما در اینجا از npm برای نصب استفاده می کنیم.

حال در vscode باید از sass استفاده کنیم به صورت زیر

ابتدا باید یک اکستنشن را نصب کنیم و ان هم live Sass compiler است بعد از نصب کردن باید فولدری برای sass بسازیم و در داخل ان مثلا فایل main.scss ایجاد کنیم و در داخل ان باید مثلا یک استایل دهی انجام دهیم. در حالت عادی نمی توانیم استفاده کنیم چرا که ابتدا باید کامپایل کنیم تا یک فایل css به وجود بیاید و ان فایل css به وجود امده را در تگ head موجود در index.html اضافه کنیم.

برای کامپایل کردن بعد از نصب ان اکستنشن در پایین صفحه یک ایتم جدید اضافه شده به نام watch sass که روی ان کلیک کنیم شروع به کامپایل می کند و یک فایل main.css را برای ما می سازد و ان فایل css را باید به تگ head اضافه کنیم.

می توانیم متغیر ها را هم به صورت محلی تغریف کنیم به این صورت که مثلا درداخل تگ div یک متفیر بسازیم مثلا.

$mcolor : green;

div{

$mcolor : red;

Color : $mcolor;

}

P{

Color:$mcolor;

}

در این مثال طبیعی است که رنگ متن div باید قرمز باشد و رنگ p باید سبز باشد حالا اگر از !global استفاده کنیم کار فرق می کند.

$mcolor : green;

div{

$mcolor : red !global;

Color : $mcolor;

}

P{

Color:$mcolor;

}

این به این معناست که بعد از تگ div دیگه هر المنتی رل سلکت کنیم و از $mcolor استفاده کنیم قرمز خواهد بود.

نکته ی دیگر sass این است که می توانیم در sass به صورت تو در تو استایل دهی کنیم مثلا.

<div>

<p></p>

<h1></h1>

</div>

Main .scss =>

div {

p{

color:red;

}

H1{

Color :black;

}

}

در داخل یک nested css style اگر از & استفاده کنیم به این معنی است که اسم خود کلاس مدنظر ماست به این معنا که مثلا.

.container{

&:hover{

}

}

آن جایی که از & استفاده شده به این معناست که همانجا کلاس container حضور دارد یعنی گویا به شکل زیر است.

.container{

.container:hover{

}

}

هر جا که با % تعریف کنیم یک فیلدی را به این معنا است که placeholder selector است. یعنی جایی وجود ندارد اما ما می توانیم مثلا چند جا که مشترک داریم یک % name بسازیم و بعدش در جاهایی که مورد نیاز است extend کنیم.مثلا

% example{

Width:100px;

Height : 100px;

Border:1px solid red;

}

.firstDiv{

@extend %example;

Color : red;

}

.secondDiv{

@extend %example;

Color : green;

}

در داخل scss هم میتوان از flow control استفاده کرد به این معنا که می توان از if هم استفاده کرد که در جلو تر بهتر مشاهده می کنیم.

Partial ها کد هایی هستند که با \_ (اندرلاین) مشخص می شوند مثلا \_colors.scss و به این معنا است که این فایل ها فقط import شوند و نیازی نیست به این که کامپایل شوند و فقط کد های \_colors را به فایلی که در ان import کردیم اضافه کن و اصلا نیازی به کامپایل کردن نیست.

ما می توانیم در دل کلاسی که تعریف می کنیم هم چیزی را import کنیم به این کار nested import می گویند مثل

Div{

@Import "fontLibrary"

}

H1{

@Import "colors"

}

Mixin ها یک تکه کدی هستند که می توانیم از ان ها زمانی استفاده کنیم که می خواهیم از non-semantic کلاس ها در css استفاده کنیم که با @mixin تعریف می شود و با @include فراخوانی می شود. Mixin ها می توانند ارگومان به عنوان ورودی بگیرند که باید زمانی که با include فراخوانی می شوند به عنوان پارامتر پاس بدهیم به این صورت:

@mixin test{

Margin: 0px;

Padding: 0px;

}

Div {

@include test;

}

@mixin rtl ($property, $ltr-value)

{

#{$propperty} : $ltr-value;

}

.sidebar {

@include rtl("margin","30px")

}

می توانیم با کلمه ی کلیدی @content خودمان تکه کدی را به mixin پاس بدهیم. یعنی به فرم زیر مثلا

@mixin hover{

&:hover{

@content

}

}

.button{

Border: 1px solid red;

@include hover{

Border-radius : 50%

}

}

این تکه کد بالا به این معنا است که زمانی که روی button هاور شد فیلد border-radius آن تغییر کند و به 50% برسد. به این فرایند ها content Block می گویند

**Function**

فانکشن ها با @function معرفی می شوند.

@function pow ($base,$exponent){

$result: 1;

@for $counter from 1 though $exponent {

$result : $result \* $base;

}

@return $result;

}

.slidbar{

Margin-left : pow (4,3) \* 1px;

}

فانکشن ها هم می توانند همانند mixin ها مقدار اولیه داشته باشند مثلا در پارامتر های ورودی تابع pow که $base و $ exponent داریم می توانیم مقدار اولیه بدهیم مثلا بگیم $base : 10 , $exponent : 2 تا زمانی که فراموش کردیم ارگومانی را پاس بدهیم این مقادیر به صورت دیفالت ذخیره شود.

**Interpolation**

زمانی که بخواهیم بعضی پارامتر ها را به صورت متغیر داشته باشیم باید از interpolation استفاده کنیم مثلا برای نام فایلی که می خواهیم به عنوان background image داشته باشیم می توانیم نام های گوناگونی را انتخاب کنیم به همین دلیل فیلد نام به صورت متغیر است پس از interpolation استفاده می کنیم.

مثلا

**@mixin** **corner-icon**($name**,** $top-or-bottom**,** $left-or-right) {

**.icon-**#{$name} {

**background-image**: url("/icons/#{$name}.svg");

**position**: absolute;

#{$top-or-bottom}**:** 0;

#{$left-or-right}**:** 0;

}

}

**@include** **corner-icon**("mail"**,** top**,** left);

یعنی در css می خواهیم متغیر قرار دهیم.

Errors

ارور ها هم که از اسمشان معلوم است برای زمانی است که به مشکل می خوریم و می خواهیم به کاربر اطلاع بدهیم که به مشکل خوردیم.

**@mixin** **reflexive-position**($property**,** $value) {

**@if** $property **!=** left **and** $property **!=** right {

**@error** "Property #{$property} must be either left or right.";

}

$left-value: **if**($property **==** right**,** initial**,** $value);

$right-value: **if**($property **==** right**,** $value**,** initial);

**left**: $left-value;

**right**: $right-value;

**[**dir**=**rtl**]** **&** {

**left**: $right-value;

**right**: $left-value;

}

}

**.sidebar** {

**@include** **reflexive-position**(top**,** 12px);

*// ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*

*// Error: Property top must be either left or right.*

}

در مثال بالا یک property و یک value پاس میدهیم و در داخل mixin می بینیم که شرط گذاشته ایم که اگر left یا right نبود ارور بدهد و با توجه به ارگومان پاس داده شده باید ارور داشته باشیم و زمانی که به ارور می خوریم کامپایل stop میشود.

**Flow Controls**

Flow control ها در پیاده سازی الگوریتم ها به ما کمک می کنند که ما در اینجا 4 تا flow control خواهیم داشت که عبارتند از if – each – for – while که برای استفاده از این flow-control ها باید از @ استفاده کنیم مثلا : @if-@else- @else if - @for - @while -@ each

@each همانند for each در زبان های برنامه نویسی است.