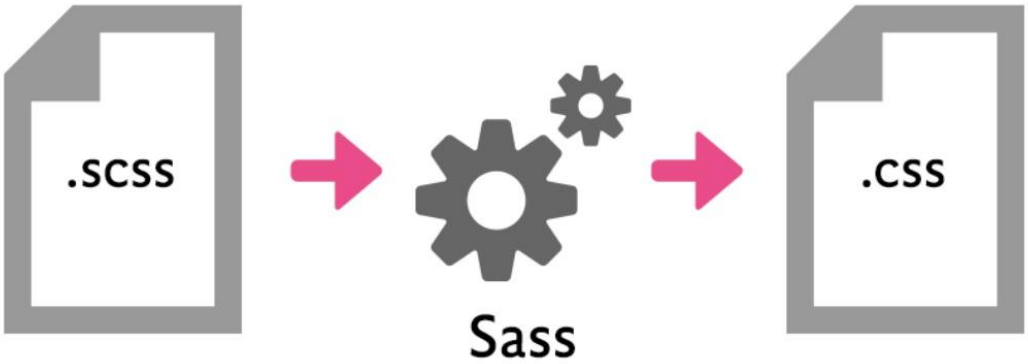
SCSS/SASS

What are Sass and SCSS?

Sass - это препроцессор для CSS, расширяющий его функциональность.

Изначально файлы, написанные на Sass имели расширение **.SASS**, но потом появился более новый формат **.SCSS**(Sassy CSS), который сейчас использовать предпочтительнее.

(Файлы **Sass** конвертируются в обычный CSS)



Sass vs pure CSS

Главные преимущества использования препроцессора Sass по сравнению с чистым CSS:

- *Наличие переменных, функций и циклов*. В больших сайтах можно менять настройки стилей, изменив пару переменных. (Например, можно изменить переменную цвета темы сайта и цвет изменится во всех местах)
- *Вложенность вкладывать селекторов*. В CSS для обозначения вложенности используются селекторы потомков. Это не передаёт иерархию элементов и заставляет дублировать родительские селекторы для каждого дочернего элемента.
- *Миксины* (функции, которые могут вставить любые стили в любую часть файла)
- *Расширенный импорт*, позволяющий объединить несколько файлов стилей в один и автоматически минифицировать его. (В css же для импорта производятся отдельные HTTP запросы)

Installing Sass

```
npm install -g sass
```

Установка Sass

```
sass input.scss output.css

sass input.sass output.css
```

Преобразование файла Sass в CSS файл

```
sass --watch input.scss:output.css
```

Автоматически преобразовывать Sass файл в CSS файл при наличии изменений

```
sass --watch app/input:app/output
```

Автоматически преобразовывать все Sass файлы в определенной папке и помещать результирующие CSS файлы в другой папке

Nested rules

Sass позволяет вкладывать правила CSS друг в друга.

Таким образом, визуальная структура элементов разметки сохраняется и в стилях.

SCSS

```scss
#header {
    width: 100%;
    #logo {
        width: 30%;
        img {
            width: 100px;
            height: 30px;
        }
    }
    #menu {
        display: flex;
        margin-left: 10px;
    }
}
```

CSS

```css
#header {
  width: 100%;
}
#header #logo {
  width: 30%;
}
#header #logo img {
  width: 100px;
  height: 30px;
}
#header #menu {
  display: flex;
  margin-left: 10px;
}
```

.SCSS vs .SASS

Синтаксис SASS отличается от SCSS тем, что он использует **отступы вместо фигурных скобок**, указывающие на вложение селекторов, и **новые строки, заменяющие точки с запятой**, для разделения свойств.

SCSS (Далее будем использовать только его)

```scss
#header {
    width: 100%;
    #logo {
        width: 30%;
        img {
            width: 100px;
            height: 30px;
        }
    }
    #menu {
        display: flex;
        margin-left: 10px;
    }
}
```

SASS

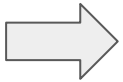
```sass
#header
    width: 100%
    #logo
        width: 30%
        img
            width: 100px
            height: 30px
    #menu
        display: flex
        margin-left: 10px
```

Nested properties

Помимо вложенных правил можно использовать вложенные свойства (свойства должны находиться в одном пространстве имён)

SCSS

```scss
#header {
    margin: {
        left: 10px;
        right: 9px;
        top: 8px;
        bottom: 7px;
    }
}
```



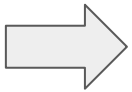
```css
#header {
  margin-left: 10px;
  margin-right: 9px;
  margin-top: 8px;
  margin-bottom: 7px;
}
```

CSS

Само пространство имен также может иметь значение

SCSS

```scss
#header {
    margin: 5px {
        left: 10px;
        right: 9px;
    }
}
```



```css
#header {
  margin: 5px;
  margin-left: 10px;
  margin-right: 9px;
}
```

CSS

Link to selector parent

Вложенные правила могут указать ссылку на родительский селектор с помощью символа &

Extension/Inheritance

Мы можем наследовать наборы свойств от одного селектора к другому с помощью директивы **@extend**.

Набор объявляется как: **%**<имя набора> **{** <свойства набора> **}**

SCSS

```scss
%error {
    color: red;
    background-color: orange;
    text-transform: uppercase;
}

.message1 {
    font-size: 30px;
    @extend %error;
}
.message2 {
    font-size: 40px;
    @extend %error;
}
```

Наследуем набор свойств у нужных нам элементов

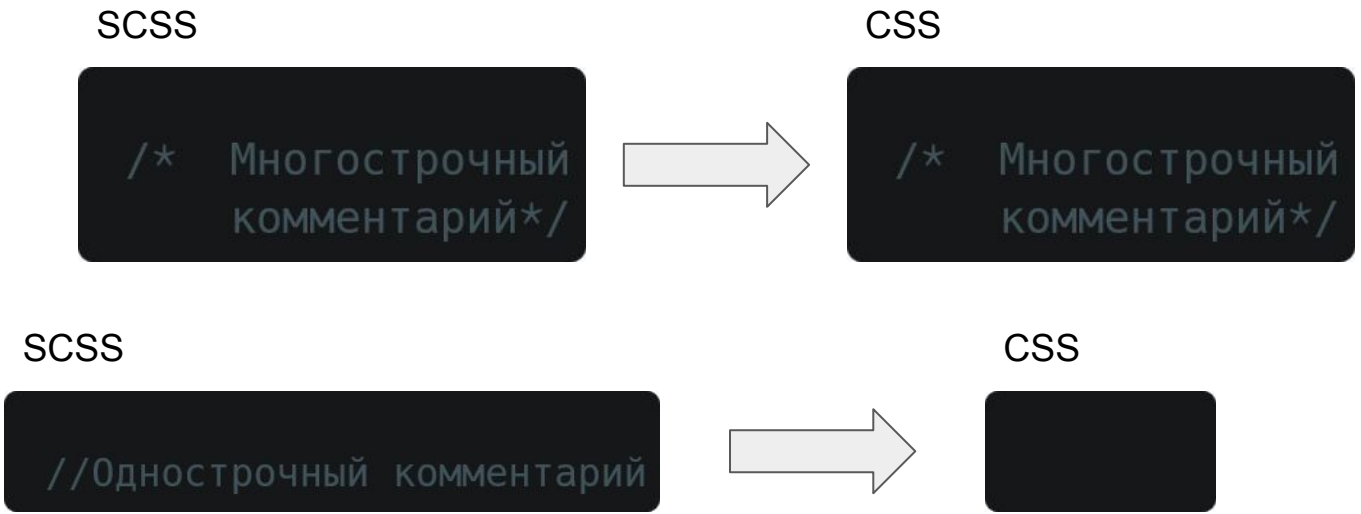
CSS

```css
.message2, .message1 {
  color: red;
  background-color: orange;
  text-transform: uppercase;
}

.message1 {
  font-size: 30px;
}

.message2 {
  font-size: 40px;
}
```

Comments

Многострочные комментарии добавляются в CSS.

Однострочные комментарии не добавляются в CSS, так как он их не поддерживает.

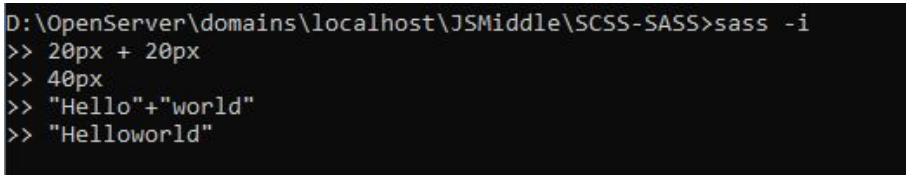


SassScript

Sass REPL

Для работы с SassScript в интерактивном режиме нужно выполнить команду: **sass -i**

В командной строке можно ввести любое поддерживаемое SassScript выражение и результат выведется на экран.



(Для выхода из этого режима можете нажать Ctrl+C)

Variables

Переменные начинаются со знака доллара **$** и задаются как свойства CSS. Дальше их можно использовать в нужных местах:

SCSS

```scss
$myColor: #ccc;
$height: 50px;
$border: 1px solid black;

.card {
    color: $myColor;
    height: $height;
    border: $border;
}
```

```css
.card {
  color: #ccc;
  height: 50px;
  border: 1px solid black;
}
```

CSS

Препроцессор довольно глуп и не будет проверять правильность присвоенных свойствам значений:

SCSS

```scss
$myColor: #ccc;
$height: 50px;
$border: 1px solid black;

.card {
    color: $height;
    height: $border;
    border: $myColor;
}
```

```css
.card {
  color: 50px;
  height: 1px solid black;
  border: #ccc;
}
```

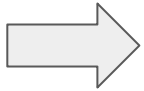
CSS

Variables

Переменные доступны только в пределах того уровня вложенности селекторов, на котором они определены.

SCSS

```scss
.card {
    $color: #aaa;
    width: 300px;
}

body {
    background-color: $color;
}
```



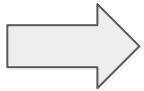
```
/* Error: Undefined variable.
 *   ,
 * 7 |     background-color: $color;
 *   |                       ^^^^^^
 *   '
 *   sass\main.scss 7:20  root stylesheet */
```

CSS

Переменную можно сделать глобальной, если после ее значения указать метку **!global**

SCSS

```scss
.card {
    $color: #aaa !global;
    width: 300px;
}

body {
    background-color: $color;
}
```



```css
.card {
  width: 300px;
}

body {
  background-color: #aaa;
}
```

CSS

Default variables

Метка **!default** в конце присваемого значения говорит Sass не выполнять присваивание, если переменная уже имеет значение

```
>> $newVar: "HelloWord" !default
>> "HelloWord"
>> $newVar: "NewText" !default
>> "HelloWord"
```

null тоже рассматривается как отсутствие значения

```
>> $newVar2: null !default
>> null
>> $newVar2: "NewText" !default
>> "NewText"
```

Data types

В Sass есть 7 основных типов данных:

1) **числа** (1.5, 17, 20px)
2) **текстовые строки**, с кавычками и без них ("tuk", 'buz', cum)
3) **цвета** (red, #112233, rgba(255, 0, 0, 0.7))
4) **булевы значения** (true, false)
5) **null**
6) **списки значений**, с разделительными пробелами или запятыми (15px 10px 20px; Verdana, Arial, sans-serif)
7) **массивы(мапы)** (key1: value1, key2: value2, key3: value3)

Все другие виды данных CSS (например метка !important) также поддерживаются и используются как строки без кавычек.

Operations with numbers

Арифметические операции

```
>> 10px + 10px
>> 20px

>> 10px - 10px
>> 0px

>> 10 * 10px
>> 100px

>> calc(100px / 10)
>> 10px
```

Операцию деления необходимо обернуть в функцию calc
(так как символ “/” используется в CSS для обособления численных данных,
а не деления)

Операции сравнения

```
>> 100px > 10px
>> true

>> 100px < 10px
>> false

>> 100px >= 10px
>> true

>> 100px <= 10px
>> false

>> 100px == 10px
>> false

>> 100px != 10px
>> true
```

Операция == и != работают с любыми типами данных

Division operation

Символ “/” используется в CSS для обособления численных данных.

Но в Sass есть 4 случая, когда “/” будет интерпретироваться как деление:

1) Если выражение заключено в функцию calc
2) Если один из операндов является переменной или возвращается функцией ***<Deprecated!>***
3) Если операнды(или всё выражение) заключены в круглые скобки ***<Deprecated!>***
4) Если всё выражение используется как часть другого арифметического выражения ***<Deprecated!>***

На данный момент **2, 3 и 4 способы являются устаревшими**.

Вот, что выведет Sass при попытке использовать деление вне calc:

```
>> (100px / 10)
>> Deprecation Warning: Using / for division outside of calc() is
deprecated and will be removed in Dart Sass 2.0.0.
```

Чтобы использовать “/” как разделитель CSS, нужно обернуть операнды в #{}

Operations with colors

Арифметические операции с цветами были признаны “**deprecated**” в 2016 году и **были удалены из новых версий Sass**.

Вместо них нужно использовать специальные функции для работы с цветами:

Смешивание двух цветов (функция **mix**)

```
>> mix(#112233, #aabbcc, 50%)
>> #5e6f80
```

Непрозрачность (opacity) первого цвета при наложении поверх второго (по умолчанию 50%).
Полученный цвет и будет возвращен

Изменение яркости цвета (функции **darken** и **lighten**)

```
>> darken(#123456, 10%)
>> #091b2c
>>
>> lighten(#123456, 10%)
>> #1b4e80
```

Процент применения эффекта (параметр обязателен)

Добавление альфа канала к цвету (функция rgba)

```
>> rgba(#101010, 0.5)
>> rgba(16, 16, 16, 0.5)
```

Note: данные функции можно комбинировать.

Также передавать в них можно цвета не только в виде 16-чных кодов, но и в виде: rgb, rgba, hsl, именных цветов

Operations with strings

Конкатенация строк (оператор +)

```
>> "Hello" + "world"
>> "Helloworld"
```

Интерполяция (конструкиця #{})

```
>> "У вас на счёте лежит #{30+70} долларов"
>> "У вас на счёте лежит 100 долларов"

>> $width: 10px
>> 10px
>> #{$width} solid black
>> 10px solid black
```

Наличие или отсутствие кавычек в итоговом выражении зависит от первого операнда

```
>> Hello + "world"
>> Helloworld

>> "Hello" + world
>> "Helloworld"
```

Два значения, находящиеся рядом друг с другом, объединяются через пробел

```
>> 10px + 10px world
>> 20px world
```

Operations on boolean values

Логические операторы (and, or, not)

```
>> true and false
>> false

>> true or false
>> true

>> not true
>> false
```

Операнды могут также представлять из себя выражения

```
>> 5 > 4 and 5 > 3
>> true
```

```
>> 5 > 4 or 100 < 200
>> true
```

Operations with lists and maps

Список - это серия из нескольких однотипных значений, разделенных пробелами или запятыми.

Индивидуальное значение = список из 1-ого элемента.

```
>> $list: one two three four five
>> one two three four five
```

Получение элемента по номеру

```
>> nth($list, 2)
>> two
```

Объединение списков

```
>> join($list, six seven eight)
>> one two three four five six seven eight
```

```
>> append(a b c d, six)
>> a b c d six
>> append(a b c d, six seven eight)
>> a b c d (six seven eight)
```

Добавление элемента в список

Добавление списка в список

Мапы представляют собой связь между ключами и значениями.

Ключами мапа может быть любой тип Sass данных (даже другой массив)

```
>> $codes: (Россия: RU, США: US, Великобритания: GB, Украина: UA)
>> (Россия: RU, США: US, Великобритания: GB, Украина: UA)
```

Получение элемента по ключу

```
>> map-get($codes, США)
>> US
```

Объединение мапов

```
>> map-merge($codes, (Белоруссия: BY, Китай: CN))
>> (Россия: RU, США: US, Великобритания: GB, Украина: UA, Белоруссия: BY, Китай: CN)
```

Для работы со списками и мапами в основном используется директива **@each**, которая будет рассмотрена позже.

Parentheses

Скобки позволяют менять порядок действий операторов

```
>> (2 + 3) * 5
>> 25
```

```
>> not (4 > 5)
>> true
```

```
>> $width: 100px
>> 100px
>> ($width + 20px) * 2
>> 240px
```

Functions

Собственные функции определяются с помощью директивы **@function.** Для возврата значений используется директива **@return**.

SCSS

```scss
$offsetX: 50px;

@function getObjectCenterX($width) {
    @return $offsetX + calc($width / 2);
}

.card {
    position: absolute;
    left: getObjectCenterX($width: 200px)
}
```

```css
.card {
  position: absolute;
  left: 150px;
}
```

CSS

Функция имеет доступ к входным аргументам и глобальным переменным

При вызове любых функций (в том числе встроенных) можно использовать именованные аргументы.

Функцию из предыдущего примера можно было вызывать так:

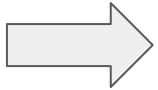
```scss
getObjectCenterX($width: 200px)
```

Interpolation

Интерполяция #{} позволяет использовать переменные в названии свойств и селекторов

SCSS

```scss
$variable1: info;
$variable2: color;

div.#{$variable1} {
    background-#{$variable2}: green;
}
```



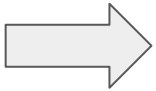
```css
div.info {
  background-color: green;
}
```

CSS

Также интерполяция помогает вставить значения переменных, не выполняя над ними операции (в данном случае деление).

SCSS

```scss
div {
  $font-size: 24px;
  $line-height: 60px;
  font: #{$font-size}/#{$line-height};
}
```



```css
div {
  font: 24px/60px;
}
```

CSS

Rules and Directives

Directive @import

Sass расширяет CSS правило **@import**, позволяя импортировать scss и sass файлы.

SCSS (main.scss)

```scss
@import "anotherStyle.scss";

p {
    width: $width;
}
```

Неуказанное расширение по умолчанию считается .scss и .sass

SCSS (anotherStyle.scss)

```scss
$width: 50px;
body {
    width: $width;
}
```

(Препроцессор просто вставляет содержимое одного файла в другой)

CSS

```css
body {
  width: 50px;
}

p {
  width: 50px;
}
```

Если *расширение файла .css*, или имя файла начинается с *http://* или вызывается через *url()*, или @import включает в себя любые медиа-запросы - то @import отрабатывает как обычное CSS-правило.

Directive @import

@import можно использовать и в css-свойствах и в медиа-условиях

SCSS (main.scss)

```scss
.header {
    @import "anotherStyle.scss";
    width: 50px;
}
```

CSS

```css
.header {
  width: 50px;
}
.header p {
  color: blue;
}
```

SCSS (_anotherStyle.scss)

```scss
p {
    color: blue;
}
```

Нижнее подчёркивание перед именем файла, говорит о том, что это **фрагмент.** При сканировании папки он не будет преобразован в отдельный CSS файл.

Фрагмент предназначен для включения в другой файл (при этом “_” в импорте указывать не нужно)

Directive @media

Директива **@media** также расширяет стандартное правило CSS:

1) Директива может вкладываться в правила CSS. Тогда при компиляции она будет поднята наверх таблицы стилей, а все селекторы в которых была директива переместятся внутрь **@media.**
2) Несколько @media запросов также могут вкладываться друг в друга. После компиляции эти @media будут объединены оператором and.
3) @media запросы могут содержать в себе все возможности SassScript (включая переменные, функции и операторы)

SCSS

```scss
$myWidth: 960px;
.header {
    width: $myWidth;

    @media screen {
        @media (max-width: $myWidth) {
            width: 100%;
        }
    }
}
```

CSS

```css
.header {
  width: 960px;
}
@media screen and (max-width: 960px) {
  .header {
    width: 100%;
  }
}
```

Directive @extend

Применение директивы **@extend** с наборами свойств (**%**<имя набора> **{** <свойства набора> **}**) уже рассматривалось.

Также с помощью **@extend** можно наследовать элементы не только от наборов свойств, но и от других селекторов.

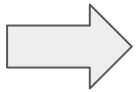
Причем селекторы, от которых мы наследуемся, могут быть и составные
(например div.p:hover)

SCSS

```scss
.red {
    color: red;
    background-color: red;
}
.big {
    font-size: 50px;
}
.error {
    @extend .red;
    @extend .big;
    text-transform: uppercase;
}

.message1 {
    @extend .error;
    font-family: Georgia, serif;
}
```

Множественные расширения

Цепное расширение
(наследуемся от .error, который наследуется от .red и .big)



CSS

```css
.red, .error, .message1 {
  color: red;
  background-color: red;
}

.big, .error, .message1 {
  font-size: 50px;
}

.error, .message1 {
  text-transform: uppercase;
}

.message1 {
  font-family: Georgia, serif;
}
```

Directive @extend

Если последовательность селекторов расширяет другой селектор, который также является частью последовательности, то эти две последовательности селекторов должны быть объединены.

SCSS

```scss
.tabbar p {
  font-weight: bold;
}

.header .logo {
  @extend p;
}
```

CSS

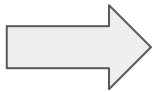
```css
.tabbar p, .tabbar .header .logo, .header .tabbar .logo {
  font-weight: bold;
}
```

Directive @extend

Нельзя наследоваться от селекторов в @media директиве

SCSS

```scss
@media print {
    .red {
        color: red;
        background-color: red;
    }
    .error {
        @extend .red;
        text-transform: uppercase;
    }
}
```



CSS

```
/* Error: From line 7, column 3 of sass\main.scss:
 *   |
 * 7 |         @extend .red;
 *   |         ^^^^^^^^^^^^
 *   '
 * You may not @extend selectors across media queries.
 *   ,
 * 7 |         @extend .red;
 *   |         ^^^^^^^^^^^^
 *   '
 *   sass\main.scss 7:3  root stylesheet */
```

(В @media можно наследоваться только от набора свойств (который должен находиться в том же блоке директивы)

Directive @at-root

@at-root перемещает одно или несколько правил из родительского селектора в корневой уровень документа.

SCSS

```scss
.container {
    width: 800px;
    @at-root {
        .header {
            width: 500px;
        }
        .aside {
            width: 300px;
        }
    }
}
```

CSS

```css
.container {
  width: 800px;
}

.header {
  width: 500px;
}
.aside {
  width: 300px;
}
```

Directive @at-root

Данная директива имеет и расширенный синтаксис **@at-root(without: ...)** или **@at-root(with: ...)**

На месте “...” указываются через пробел директивы, из которых нужно изъять свойство (в случае without) или в которых его нужно оставить (в случае with). Среди списка директив может быть “rule”, который отвечает за исключение(или оставление) из обычных свойств.

SCSS

```scss
@media print {
    .container {
        width: 800px;
        @at-root(without: media rule) {
            .header {
                width: 500px;
            }
            .aside {
                width: 300px;
            }
        }
    }
}
```

исключаем блок из директивы @media и из обычных правил CSS

CSS

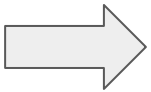
```css
@media print {
  .container {
    width: 800px;
  }
}
.header {
  width: 500px;
}

.aside {
  width: 300px;
}
```

The @debug, @warn and @error directives

@debug, **@warn** и **@error** выводит значения выражений Sass средствами стандартного вывода ошибок.

SCSS

```scss
@function toPercent($value) {
    @if $value < 0 {
        @error "$value не может быть отрицательным"
    }
    @return calc($value / 100)
}

p {
    width: toPercent(-5px);
}
```



CSS

```
/* Error: "$value не может быть отрицательным"
 *    ,
 * 9 |     width: toPercent(-5px);
 *   |            ^^^^^^^^^^^^^^^
 *   '
 *   sass\main.scss 9:9  root stylesheet */
```

@warn (в отличие от **@debug)** будет выведен в консоль вместе с отрывком таблицы стилей, чтобы можно было понять, где в коде требуется внимание.
@error, помимо этого, выведется в итоговый CSS, и **ничего, кроме этой ошибки, в CSS файле содержаться не будет**.

Control directives and expressions

Function if()

Функция if возвращает одно из двух значений на основе условия (выражения, возвращающего boolean):

if(<условие>, <значение при истинном условии>, <значение при ложном условии>)

SCSS

```scss
$deviceWidth: 1920px;
body {
    width: if($deviceWidth > 960, 960px, 100%)
}
```

```scss
$deviceWidth: 600px;
body {
    width: if($deviceWidth > 960, 960px, 100%)
}
```

CSS

```css
body {
  width: 960px;
}
```

```css
body {
  width: 100%;
}
```

Directive @if

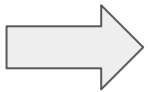
Директива **@if** служит для применения определённых стилей, если заданное условие истинно.

В противном случае будут проверяться блоки **@else if,** пока один из них не вернёт истину или интерпретатор не достигнет **@else.**

(Условная конструкция может и не иметь **@else if** или **@else**)

SCSS

```scss
$deviceWidth: 1100px;
.card {
    @if $deviceWidth < 600px {
        width: 30px;
    } @else if $deviceWidth >= 600px and $deviceWidth < 960px {
        width: 45px;
    } @else if $deviceWidth >= 960px and $deviceWidth < 1280px {
        width: 60px;
    } @else {
        width: 80px;
    }

}
```



CSS

```css
.card {
  width: 60px;
}
```

Directive @for

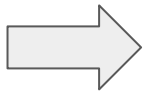
Директива **@for** выводит набор стилей заданное число раз:

@for $i **from** <начало> **through** <конец> (с включением конца)

@for $i **from** <начало> **to** <конец> (без включения конца)

SCSS

```scss
@for $i from 1 through 5 {
    .card#{$i} {
        width: 100px * $i;
    }
}
```



CSS

```css
.card1 {
  width: 100px;
}

.card2 {
  width: 200px;
}

.card3 {
  width: 300px;
}

.card4 {
  width: 400px;
}

.card5 {
  width: 500px;
}
```

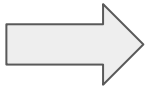
Directive @each

@each $var **in** <список или мап значений>

Директива @each последовательно устанавливает $var в каждое из значений списка или мапы и выводит содержащиеся в ней стили

SCSS

```scss
$list: Msu Mipt Hse Mephi Bmstu;
@each $vuz in $list {
    .#{$vuz} {
        background-image: url("#{$vuz}.png")
    }
}
```



CSS

```css
.Msu {
  background-image: url("Msu.png");
}

.Mipt {
  background-image: url("Mipt.png");
}

.Hse {
  background-image: url("Hse.png");
}

.Mephi {
  background-image: url("Mephi.png");
}

.Bmstu {
  background-image: url("Bmstu.png");
}
```

Directive @each

@each $var, $var2, $var3,... **in** <список или мап значений>

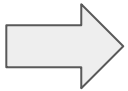
Конструкция **@each** с несколькими переменными позволяет удобно выводить стили на основе списков списков и мапов

SCSS

```scss
$list: (Msu "msu.ru") (Mipt "mipt.ru") (Hse "hse.ru");
@each $vuz, $site in $list {
    .#{$vuz} {
        background-image: url("#{$site}")
    }
}
```

<или>

```scss
$map: (Msu: "msu.ru", Mipt: "mipt.ru", Hse:"hse.ru");
@each $vuz, $site in $map {
    .#{$vuz} {
        background-image: url("#{$site}")
    }
}
```



CSS

```css
.Msu {
  background-image: url("msu.ru");
}

.Mipt {
  background-image: url("mipt.ru");
}

.Hse {
  background-image: url("hse.ru");
}
```

Directive @while

Директива **@while** принимает выражение SassScript и циклично выводит вложенные в неё стили, пока выражение равно true.

SCSS

```scss
$i: 0;
@while $i < 10 {
    p#{$i} {
        color: blue
    }
    $i: $i + 2;
}
```

CSS

```css
p0 {
  color: blue;
}

p2 {
  color: blue;
}

p4 {
  color: blue;
}

p6 {
  color: blue;
}

p8 {
  color: blue;
}
```

Declaring a Mixin (@mixin directive)

Миксин - это по сути функция, вставляющая стили в любые нужные места scss файла (обычная функция возвращает лишь SassScript значения).

Для объявления миксина служит директива **@mixin**

Миксины могут содержать только свойства

```scss
@mixin mixinName {
    margin: {
        left: 10px;
        right: 5px;
        top: 3px;
        bottom: 1px;
    }
}
```

SCSS

Миксины могут содержать селекторы

```scss
@mixin mixinName {
    .header {
        margin: {
            left: 10px;
            right: 5px;
            top: 3px;
            bottom: 1px;
        }
    }
}
```

SCSS

Миксины могут принимать аргументы

Установка значений переменных по умолчанию

```scss
@mixin mixinName($left, $right, $top: 3px, $bottom: 1px) {
    .header {
        margin: {
            left: $left;
            right: $right;
            top: $top;
            bottom: $bottom;
        }
    }
}
```

SCSS

Using a Mixin (@include directive)

Чтобы включить содержимое миксина в определённое место стилей используется директива **@include.** После неё идёт имя миксина и передаваемые аргументы (если они есть).

SCSS

Все оставшиеся аргументы будут упакованы в список*

```scss
@mixin gridMixin($width, $height, $columns...) {
    .container {
        display: grid;
        width: $width;
        height: $height;
        grid-template-columns: $columns;
    }
}

@include gridMixin(100px, 100px, 10px, 20px, 30px);
```

Вызов миксина

CSS

```css
.container {
  display: grid;
  width: 100px;
  height: 100px;
  grid-template-columns: 10px, 20px, 30px;
}
```

*работает и в обратную сторону. Мы могли бы вызвать миксин так:

```scss
$list: 100px 100px 10px 20px 30px;

@include gridMixin($list...);
```

Content blocks in mixins

В миксин можно передавать не только аргументы, но и целый блок ситлей.

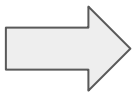
В самом миксине мы можем обращаться к этому блоку с помощью директивы **@content**

SCSS

Ссылаемся на переданный блок

Передаем блок стилей в миксин

```scss
@mixin duplicateBlock($num) {
    @for $i from 1 through $num {
        @content;
    }
}

@include duplicateBlock(3) {
    .header {
        color: green;
    }
}
```



CSS

```css
.header {
  color: green;
}

.header {
  color: green;
}

.header {
  color: green;
}
```

Блоки контента, переданные в миксин, вычисляются в той же области видимости, где определен этот блок, а не миксин.

Compressed style

Sass позволяет указать минифицированный стиль выходного CSS файла.

Для этого нужно задать при компиляции параметр **--style**=compressed

CSS

```css
.Msu {
  background-color: #112233;
}

.Mipt {
  background-color: #112233;
}

.Hse {
  background-color: #112233;
}
```

--style=expanded (по умолчанию)

--style=compressed

CSS

```css
.Msu{background-color:#123}.Mipt{background-color:#123}.Hse{background-color:#123}
```

Спасибо за внимание

Более подробно о Sass можете почитать в документации