|  |  |  |  |
| --- | --- | --- | --- |
| **Переменные** | | | |
| Sass/SCSS позволяет работать с переменными. В CSS они обозначаются двойным тире (--), а в препроцессорах знаком доллара ($).  $number: 1;  $color: #ff0000;  $text: "tproger forever.";  $text: "IT forever." !default;  $nothing: null;  Вы можете присваивать значение по умолчанию переменным, у которых ещё нет значения, добавив метку !default в конце значения. В таком случае, если переменной уже было присвоено значение, оно не изменится; если же переменная пуста, ей будет присвоено новое указанное значение.  #container {  content: $text;  } | Область видимости переменных Sass ограничивается вложенными элементами — это позволяет предотвратить влияние стилей друг на друга. Зададим различные цвета для основного цвета и для заголовка:  *$myColor: red; h1 { $myColor: green; color: $myColor; } p { color: $myColor; }*  **!global** – переопределять область видимости переменной | | |
| **Вложенные правила** | | | |
| nav {  ul {  margin: 0;  padding: 0;  list-style: none;  }  li { display: inline-block; }  a {  display: block;  padding: 6px 12px;  text-decoration: none;  }  } | Среди CSS-свойств существует разные пространства имён, например, font или margin. Чтобы каждый раз не вводить пространство имён при указании свойства, с помощью Sass можно указать пространство один раз, и затем добавлять нужные свойства:  .selector {  margin: {  top: 10px  bottom: 15px  left: 20px  right: 30px  }  } | | |
| **Амперсан** | | | |
| header {  color: black;  &\_\_link {  font-weight: bold;  &:hover {  color: red;  }  }  }  С помощью символа & вы можете явно указать, где должен быть вставлен родительский селектор. | | | |
| **Миксины (они же примеси)** | | | |
| @mixin theme($theme: DarkGray) – по умолчанию {  background: $theme;  box-shadow: 0 0 1px rgba($theme, .25);  color: #fff;  }  .info {  @include theme;  }  .alert {  @include theme($theme: DarkRed);  } | | | @mixin border($color)  border: $color 1px solid  @include border(#333) |
| **@import**,  Вы можете импортировать в ваш Sass файл **sass**, **scss** и **css** файлы с помощью директивы **@import**, при этом все миксины и переменные будут работать в основном файле, в который происходит импорт.  @import сработает как обычный CSS @import, если:  - в пути к файлу присутствует **https://**;  - файл вызывается через **url()**;  - или в импорте присутствуют медиапараметры.  @import "foo.css";  @import "media.sass";  @import "media.scss";  @import "media"; // Если импортируем media.css или \_media.sass / \_media.scss  Внимание! В новых версиях **gulp-sass** для импорта CSS файлов в Sass необходимо указывать расширение .css  Возможен импорт нескольких файлов, через запятую: **@import "header", "media"** | | | |
| **Директива @if @else**  @mixin spacing($padding, $margin) {  @if ($padding > $margin) {  padding: $padding;  } @else {  padding: $margin;  }  }  .container {  @include spacing(10px, 20px);  }  **После компиляции в CSS:**  .container { padding: 20px; } | | | |
| **Арифметические операции** | | | |
| Как и в реальной жизни, вы не можете работать с числами, у которых несовместимы типы данных (например, сложение рх и em).  Сложение и вычитание  p {  font-size: 10px + 2em; // ОШИБКА!  font-size: 10px + 6px; // 16px  font-size: 10px + 2; // 12px  }  **Пример вычитания:**  div {  height: 12% - 2%;  margin: 4rem - 1;  }  **Умножение**  Выполняется точно так же, как в CSS, с помощью calc(a \* b), но без calc и круглых скобок. Кроме того, можно ещё отделять знак умножения пробелами от чисел (5\*6 == 5 \* 6).  Исключение Нельзя умножать пиксели между собой. То есть, 10px \* 10px != 100px. 10px \* 10 == 100px.  p {  width: 10px \* 10px; // ОШИБКА!  width: 10px \* 10; // 100px  width: 1px \* 5 + 5px; // 10px  width: 5 \* (5px + 5px); // 50px  width: 5px + (10px / 2) \* 3; // 20px  } | | **Деление**  С делением дела обстоят немного сложнее, но разобраться можно, ведь в стандартном CSS косая линия (слэш) зарезервирована для использования краткой формы записи свойств. Пример ниже.  /\* краткая форма записи свойств \*/  font: italic bold .8em/1.2 Arial, sans-serif;  Есть три помощника, которые намекнут на возможность деления:  - Значение (или любая его часть) хранится в переменной или возвращается функцией.  - Значения заключены в круглые скобки.  - Значение используется как часть другого арифметического выражения.  **Пример:**  $var1: 20;  $var2: 4;  p {  top: 16px / 24px; // Отображается без изменений в стандартном CSS  top: (20px / 5px); // Производится деление (но только при использовании скобок)  top: #{$var1} / #{$var2}; // Выводится как обычный CSS-код, деление не выполняется  top: $var1 / $var2; // Деление выполняется  top: random(4) / 5; // Деление выполняется (если использовать в паре с функцией)  top: 2px / 4px + 3px; // Деление выполняется, если добавлена ещё одно арифметическое действие  } | |
| **Логические операторы (and, or, not)**  @mixin button-color ($height, $width) {  @if (($height < $width) and ($width >= 35px)) {  background-color: blue;  } @else {  background-color: green;  }  }  .button {  @include button-color(20px, 30px)  } | | | |
| **Строки**  В CSS определено 2 типа строк: с кавычками и без. Sass распознаёт и то, и другое. В итоге вы получите в CSS тот тип строк, который использовали в Sass.  В некоторых случаях можно добавить строки в допустимые значения CSS без кавычек, но только если добавленная строка является завершающим элементом.  p {  font: 50px Ari + "al"; // Компилируется в 50px Arial  }  p {  font: "50px" + Arial; // ОШИБКА!  }  p:after {  content: "Верните Линуса " + Торвальдса!; // ОШИБКА!  }  p:after {  content: "Верните Линуса " + "Торвальдса!"; // обратите внимание на "Торвальдса!"  } | | | |