

Препроцессоры

Что такое препроцессоры

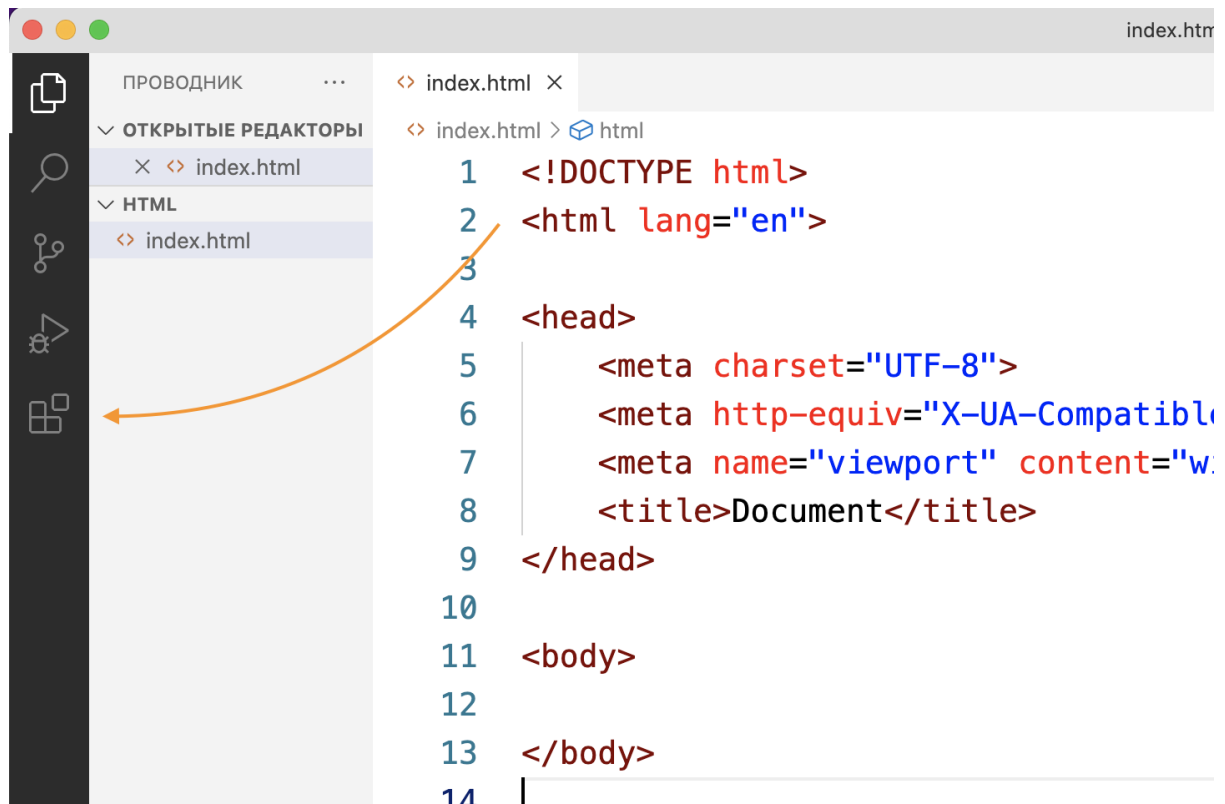
CSS-препроцессоры — это «программистский» подход к CSS. Они позволяют использовать при написании стилей собственные языкам программирования приемы и конструкции: переменные, вложенность, наследуемость, циклы, функции и математические операции. Синтаксис препроцессоров похож на обычный CSS. Код, написанный на языке препроцессора, не используется прямо в браузере, а преобразуется в чистый CSS-код с помощью специальных библиотек.

Установка препроцессора

Перед тем как начать использовать их нужно установить и запустить. Первое что вам нужно понять, что препроцессорный код это для программиста, а css для браузера, в этом и есть огромное преимущество, так как начать использовать препроцессоры вы ничего не потеряете, только получаете дополнительный функционал. Получается начать использовать можно и уже с существующим проектом, но лучше всего подключить препроцессор перед началом работы с проектом. Мы рассмотрим 2 способа подключения, на начальном этапе и когда часть проекта уже создана.

Подключение препроцессора Sass к новому проекту

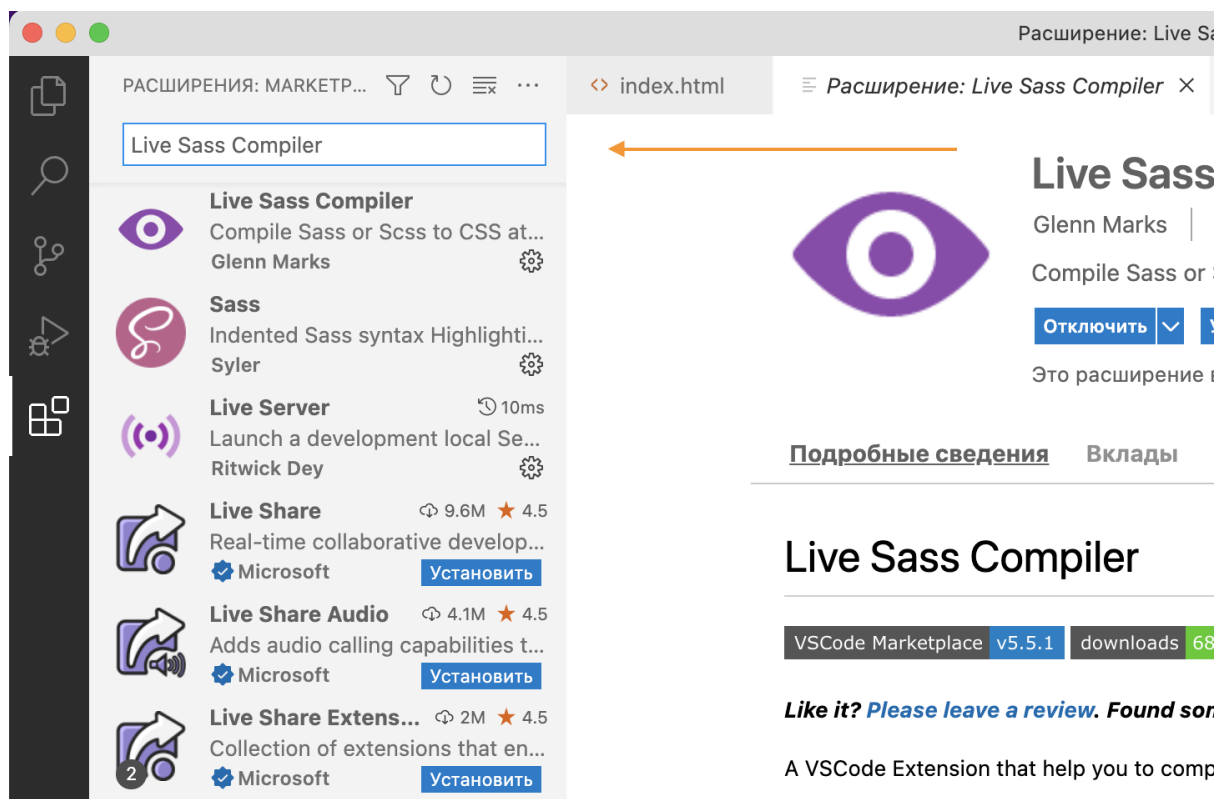
Установка расширений для редактора кода Visual Studio Code



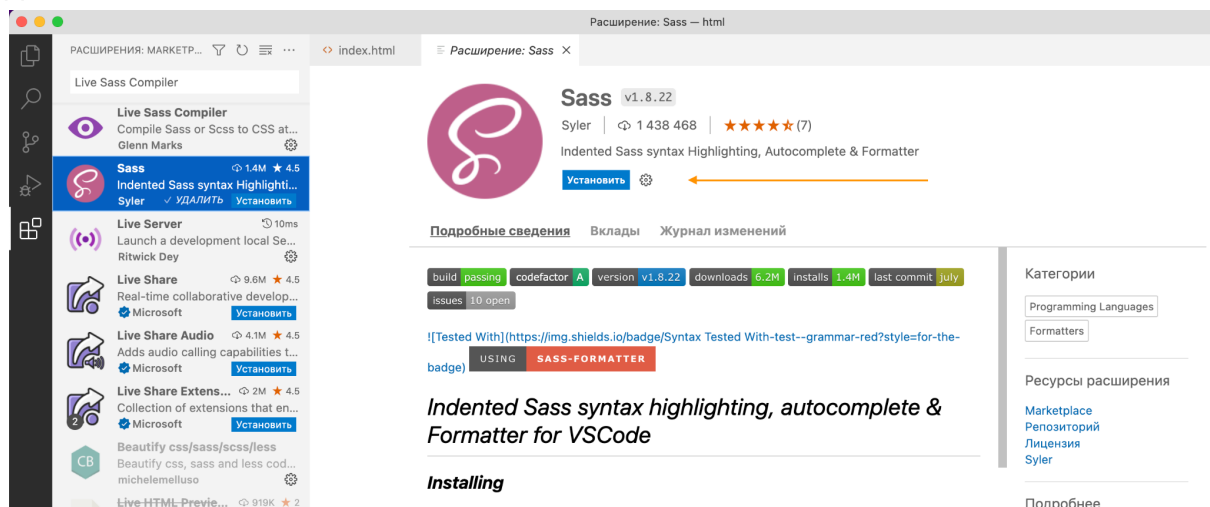
Нажимаем в левой панели “Расширения”

Устанавливаем 3 расширения

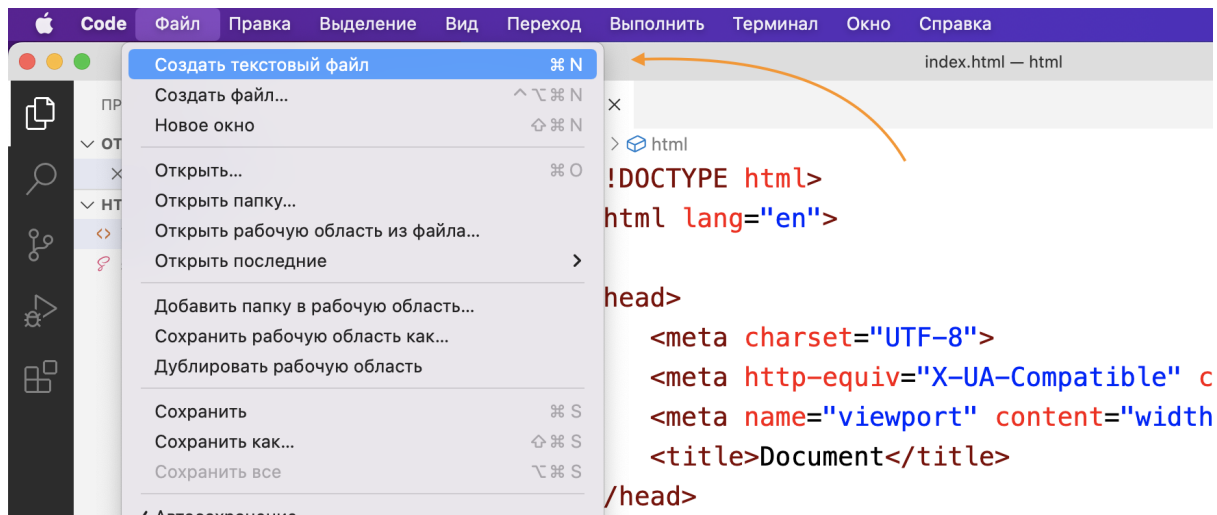
1. Live Sass Compiler
2. Sass
3. Live Server



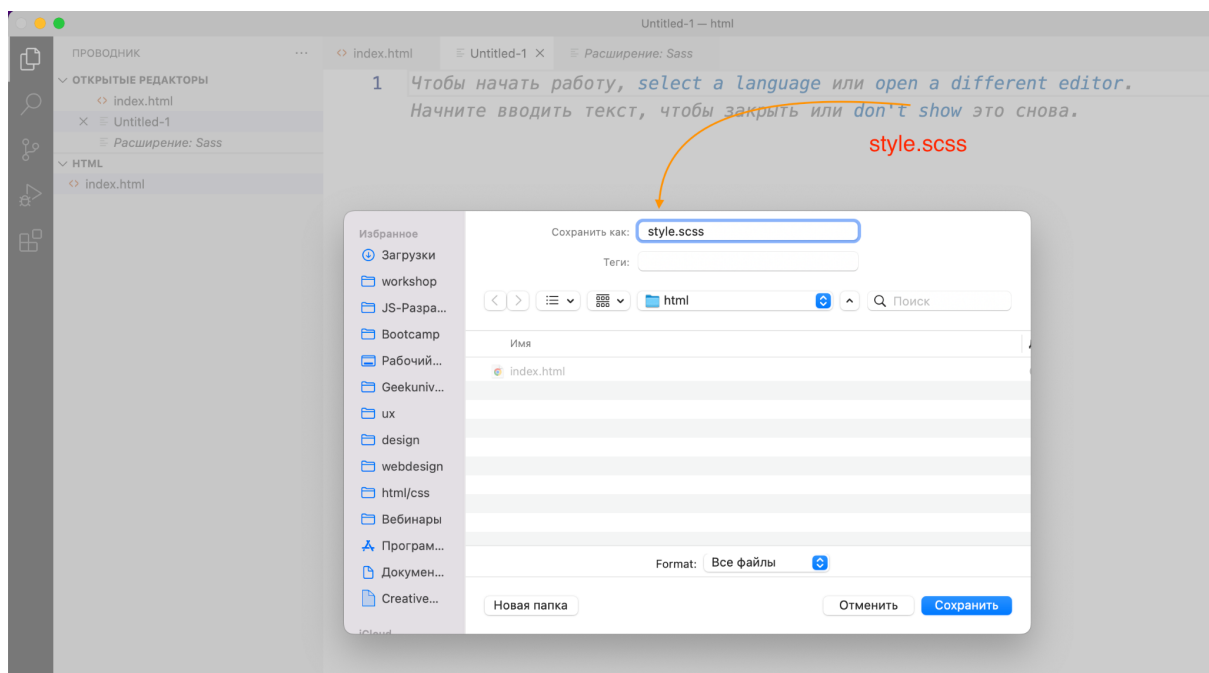
Далее нажимаем “Установить”



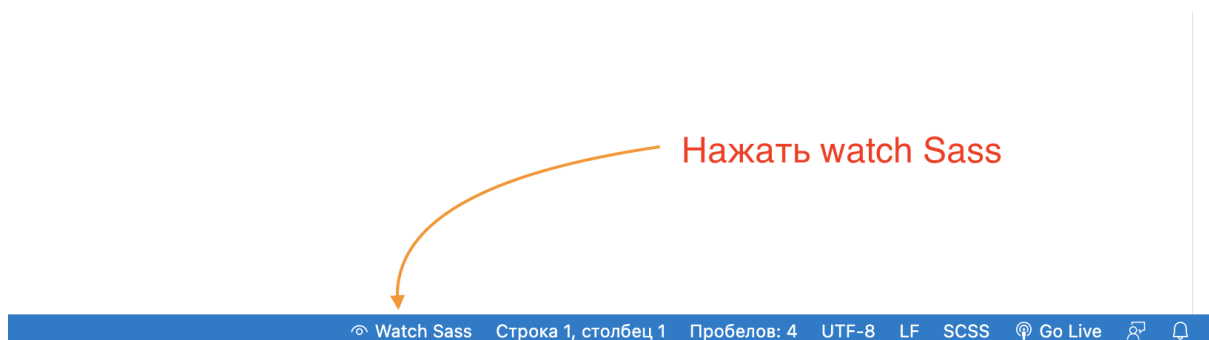
Создаём новый файл



Важно Расширение файла scss

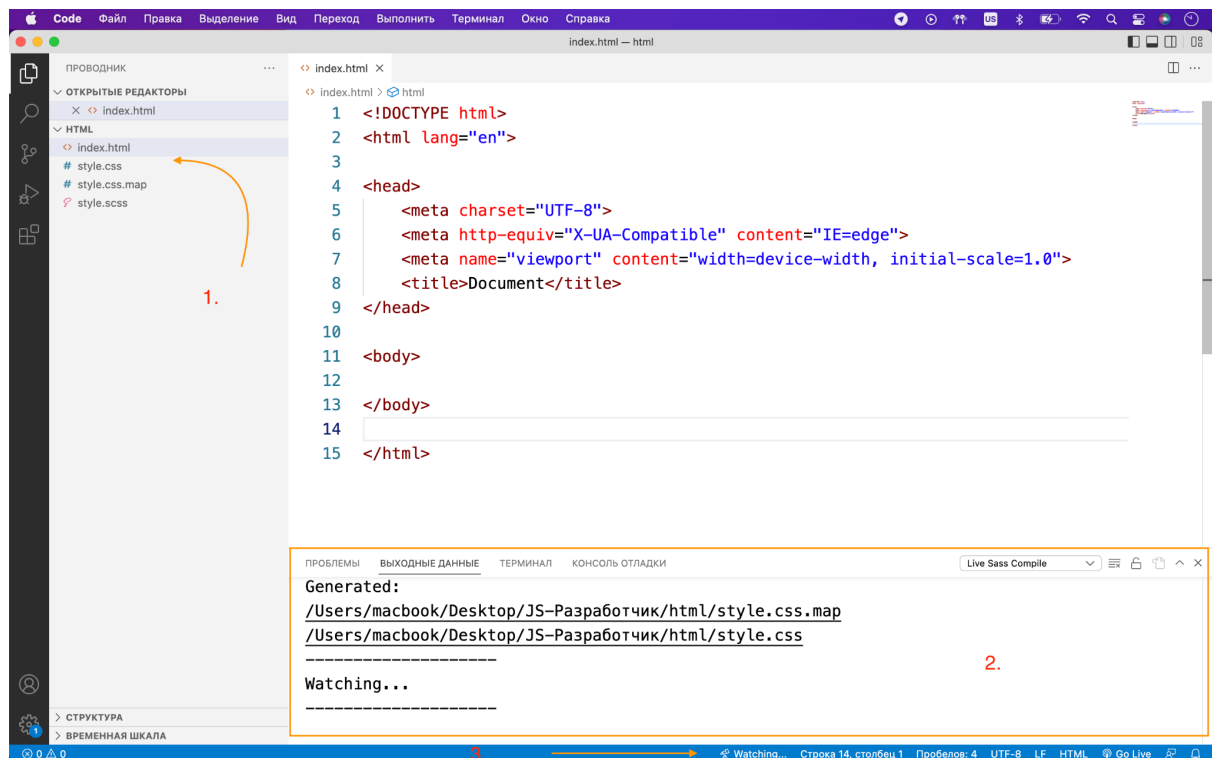


Запускаем препроцессор



Важно: Запуск препроцессора нужно будет производить каждый раз когда вы открываете проект, допустим сегодня вы работали 5 часов, закрыли проект, открываете его утром, вам нужно будет еще раз нажать клавишу **Watch Sass**.

Итоги:



1. Появился файл style.css (Мы его никогда не открываем и ничего в нем не делаем)
2. Ошибок в консоли нет, если они есть, их нужно обязательно исправить
3. Кнопка watch sass поменялась на watching

Теперь вы можете открыть файл style.scss и записывать ваши первые стили.

Подключение препроцессора к уже существующему проекту

Раздел установка не будет отличаться, всё точно так же устанавливаем расширения

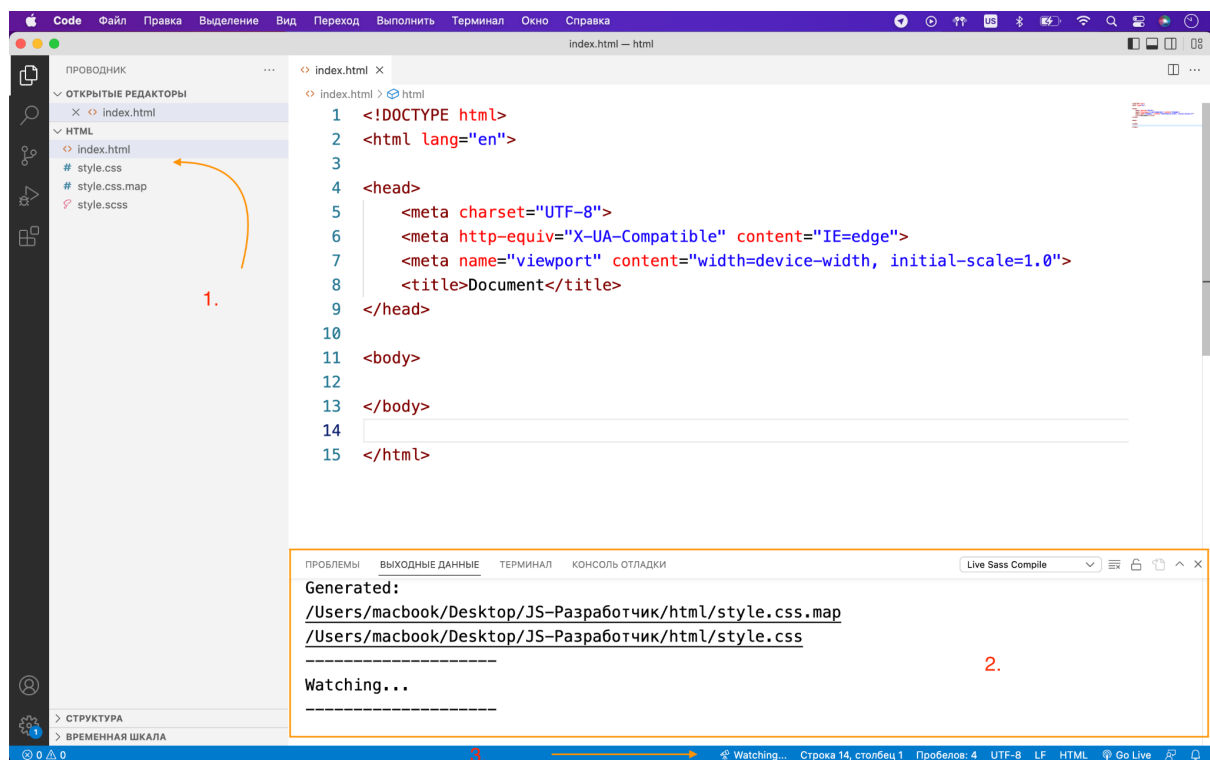
1. Live Sass Compiler
2. Sass
3. Live Server

Создаём новый файл

1. Скопируйте всё содержимое уже существующего файла style.css
2. Создайте новый файл style.scss
3. Добавьте все скопированные стили из пункта 1.
4. Нажмите на клавишу watch sass



Итоги:



1. Появился файл style.css (Мы его никогда не открываем и ничего в нем не делаем)
2. Ошибок в консоли нет, если они есть, их нужно обязательно исправить
3. Кнопка watch sass поменялась на watching

Теперь вы можете открыть файл style.scss и писать уже стили с использованием препроцессора Sass

Sass

Переходим к знакомству с самим препроцессором, первое что хочется отметить что данный препроцессор является одним из самых популярных, используется в большинстве проектов и активно развивается, именно по этой причине мы его и будем использовать.

Важно: Вам не нужно использовать весь функционал препроцессора сразу, вам достаточно будет начать использовать переменные и вложенность, а уже позже вы сможете добавить и другой функционал.


Препроцессинг

Когда таблица стилей CSS становится огромной, ее сложно обслуживать. В таком случае нам поможет препроцессор. **Sass** позволяет использовать функции, недоступные в самом CSS, например переменные, вложенности, миксины, наследование и другие приятные вещи, возвращающие удобство написания CSS.

Как только вы начинаете пользоваться **Sass**, препроцессор обрабатывает ваш Sass-файл и сохраняет его как простой CSS-файл, который вы сможете использовать на любом сайте.

Переменные

Переменные – способ хранения информации, которую вы хотите использовать при написании всех стилей проекта. Вы можете хранить в переменных цвета, стеки шрифтов или любые другие значения CSS. Чтобы создать переменную в Sass, нужно использовать символ \$.

```
 _vars.scss > ...  
1 $colorSite: #F16D7F;  
2 $widthSite: 1140px;  
3 $colorContent: #222222;  
4 $gapGrid: 30px;  
5 $colorTextContent: #FBFBFB;
```

Несколько важных моментов

1. Создавайте переменные в верхней части стилей, так как если вы не объявили переменную то использовать ее не получится.
2. Придумывайте логичные названия для переменных, так как через пол года понять что означает переменная \$a будет очень сложно

Математические операторы

Использовать математику в CSS очень полезно. Sass имеет несколько стандартных математических операторов, таких как +, -, *, / и %.

Давайте подумаем где будет логично использовать математику?

Первый хороший способ это добавить в переменную значение ширины контентной части сайта и дальше вы можете разделить ее на 3 равные части.

```
$widthSite / 3
```

В итоге если ширина сайта изменится, вам не нужно переживать, ширина блока так и останется $\frac{1}{3}$ от общей ширины

Давайте рассмотрим еще один пример

```
width: ($widthSite - 2 * $gapSite) / 3;
```

мы берём ширину сайта и вычитаем из нее удвоенный отступ (стандартный отступ из макета) и делим значение на 3, в итоге получаем значение ширины элемента, без учета отступов

Вложенности

HTML имеет четкую вложенную и визуальную иерархию, которой нет у **CSS**.

Sass позволит вкладывать CSS-селекторы таким же образом, как и в визуальной иерархии HTML. Но помните, что чрезмерное количество вложенностей затрудняет чтение и восприятие документа, а потому считается плохой практикой. Как понять что вложенность достаточная? ответ очень прост, не используйте вложенность более трех элементов

```
.product {  
  
    position: relative;  
  
    width: 370px;  
  
    .product__img {  
  
        width: 100%;  
  
        border-radius: 10px;  
  
    }  
  
}
```

Что мы можем увидеть в данном примере? блок **product__img** находится внутри **product** и располагается с табуляцией, это важно для вложенности

В данном примере мы видим что используется методология БЭМ и в итоге данный код можно сделать наглядней

```
.product {  
  
    position: relative;  
  
    width: 370px;  
  
    &__img {  
  
        width: 100%;
```



```

border-radius: 10px;

}

}

```

В итоге мы заменили **.product** на значок **&** и в итоге мы получаем максимальную наглядность и вы всегда можете свернуть блок нажав на стрелку рядом с номером строки

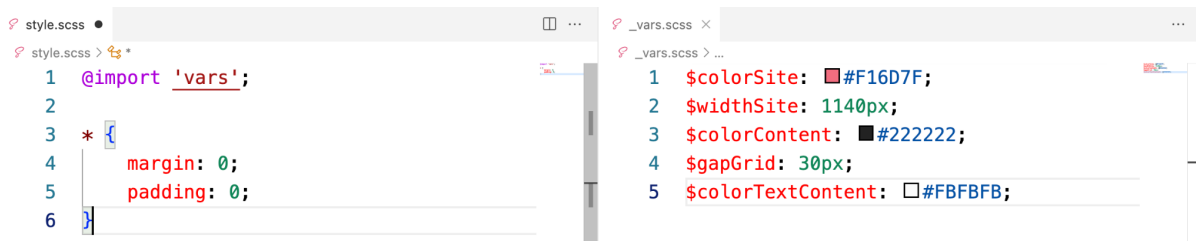
```

7 > .product { ...
15 }

```

Фрагментирование

Вы можете создавать фрагменты Sass-файла, содержащие в себе небольшие отрывки CSS, которые можно использовать в других Sass-файлах. Это отличный способ сделать CSS модульным и облегчить его обслуживание. Фрагмент – это простой Sass-файл, имя которого начинается с нижнего подчеркивания, например **_partial.scss**. Нижнее подчеркивание в имени Sass-файла говорит компилятору о том, что это только фрагмент и он не должен компилироваться в CSS. Фрагменты Sass подключаются при помощи директивы **@import**.



Несколько простых правил

1. При создании нового файла обязательно используйте нижнее подчеркивание **_vars.scss**
2. **@import** лучше всего размещать в верхней части стилей
3. Значение внутри кавычек пишется без расширения файла **@import 'vars'**
4. Фрагментов может быть сколько угодно, но тут главное не создавать их очень много, ведь в них потом легко будет запутаться

Примеси

Некоторые вещи в CSS весьма утомительно писать, особенно в CSS3, где зачастую требуется использовать много вендорных префиксов. Миксины позволяют создавать группы деклараций

CSS, которые вам придется использовать по несколько раз на сайте. Хорошо использовать миксины для вендорных префиксов.

Для создания миксина используйте директиву **@mixin + название этого миксина**. Мы назвали наш миксин **font**. В нем мы используем переменную **\$color** внутри скобок, позволяя себе передавать в переменную все, что захотим. После создания миксина вы можете использовать его в качестве параметра CSS, начиная вызов с **@include** и имени миксина.

```
style.scss ●
style.scss > ...
1  @mixin font($color) {
2      font-family: sans-serif;
3      font-weight: normal;
4      text-align-last: left;
5      color: $color
6  }
7
8  .text {
9      @include font(■red);
10     line-height: 16px;
11 }
12
13 .heading {
14     @include font(■green);
15     background-color: ■#ccc;
16 }
```

Наследование

Это одна из самых полезных функций Sass. Используя директиву **@extend**, можно наследовать наборы свойств CSS от одного селектора другому. Это позволяет держать Sass-файл в «чистоте».

Давайте рассмотрим пример

Код scss

```
%flex-center {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap
```

```
}

.product {
  @extend %flex-center;
  border: 1px solid #000;
}

.header {
  background-color: #ccc;
  @extend %flex-center;
}
```

Результат в CSS

```
.header, .product {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
}

.product {
  border: 1px solid #000;
}

.header {
  background-color: #ccc;
}
```

Что мы можем заметить, вы пере используете стиль, который в CSS будет записан через запятую, тем самым вам не нужно его дублировать.

Less

- Динамический язык стилей.
- Продукт с открытым исходным кодом.
- Может работать на стороне клиента или на стороне сервера под управлением Node.js или Rhino.

Переменные

Синтаксис переменных:

```
@название_переменной: значение_переменной;
```

Создав переменную один раз, можно использовать ее в любом месте кода.

Например:

```
@color_red: #f00;  
background-color: @color_red;  
color: @color_red;  
border-color: @color_red;
```

Во всех местах, где указана переменная, Less заменит строку `@color_red` на `#f00`. Теперь, если понадобится изменить цвет, не нужно искать все его объявления в файле, достаточно просто изменить значение переменной в одном месте.

Область видимости переменных:

Переменные можно объявлять как «снаружи» правил, так и «внутри». В случае «внутреннего» объявления переменная будет доступна только внутри правила, в котором она объявлена. Если переменная объявлена и «внутри» правила, и «снаружи», Less применит «внутреннее» значение. Таким образом можно «переопределять» глобальные переменные в локальном контексте.

Важно отметить, что переменные в Less больше похожи на константы – в отличие от переменных, они могут быть определены только один раз.

Операции

Можно использовать операции умножения, деления, сложения и вычитания.

```
@width: 100px;  
.class_1 {  
    width: @width;  
}  
.class_2 {  
    width: @width / 3;  
}  
.class_3 {  
    height: 100px + 20px;  
}
```

Цветовые операции

На практике есть немало случаев, когда мы начинаем с базового цвета и нуждаемся в слегка затемненном или осветленном его варианте.

```
@color-button: #ccc;
.button {
width: 150px;
height: 75px;
background:@color-button;
border:5px solid @color-button - #222;
}
```

Этот код создает кнопку с немного затемненной рамкой. Это частая ситуация, и определение лишь одного цвета – большая помощь.

Less позволяет манипулировать цветами на канальном уровне:

- осветлять lighten.
- затемнять darken.
- насыщать saturate.
- обесцвечивать desaturate.
- вращать цвета spin.

Примеси (mixins)

Примеси в Less избавят от набора излишнего кода. Вам когда-нибудь приходилось создавать закругленную рамку, в которой только верхние углы скруглены?

```
.border_top {
-webkit-border-top-left-radius: 6px;
-webkit-border-top-right-radius: 6px;
-moz-border-radius-topleft: 6px;
-moz-border-radius-topright: 6px;
border-top-left-radius: 6px;
border-top-right-radius: 6px;
}
.block {
background: #333;
.border_top;
}
```

Благодаря такому синтаксису мы можем использовать любой элемент в качестве примеси.

Заключение

Препроцессоры – очень удобное изобретение, рекомендуемое всем разработчикам. Они дают сокращение кода, возможности создания большого функционала сайтов на простом CSS.

Использовать препроцессоры стало намного проще, чем раньше. Нужно лишь установить программу, которая будет следить за файлами, предназначенными для препроцессора, и при

их изменении будет компилировать содержимое этих файлов в чистый CSS-код. Для более продвинутых пользователей есть специальные сборщики проектов.

Не думайте, что если вы используете программу для препроцессоров, а не сборщик проектов, то вы недостаточно круты. Такие сборщики всего лишь предлагают полный контроль и расширенные настройки.

Разумеется, как и в любой другой области, всегда есть конкуренция, и на рынке препроцессоров есть несколько предложений. Все они очень похожи, с минимальными отличиями. Какой препроцессор использовать, какой синтаксис лучше подходит – выбирать вам!