Препроцессор Sass

Если вы пишете очень много кода на CSS, препроцессор может значительно сократить объём вашей работы и ваш уровень стресса. Использование таких инструментов, как Sass, Less, Stylus или PostCSS делает ваши CSS-файлы более читабельными и облегчает последующую их поддержку. Благодаря переменным и функциям код становится более организованным и позволяет разработчикам быстрее работать и совершать меньше ошибок.

Браузер не распознает файлы Sass, так что сначала их нужно скомпилировать в обычный CSS. Поэтому вам необходимо иметь что-нибудь для конвертации .scss файлов в .css. На этот счёт есть несколько возможных решений:

* Самое простое решение - браузерный конвертер для компиляции Sass на месте - SassMeister.
* Использование приложений: существуют как платные, так и бесплатные приложения. Можете узнать больше, перейдя по этой ссылке.
* Если вы любитель командной строки, как мы, то можете установить Sass на ваш компьютер и скомпилировать файлы вручную.

Вот так можно скомпилировать .scss файлы, используя командную строку (необходимо установить node.js):

node-sass input.scss output.css

Переменные

Переменные работают по такому же принципу, как и в любом языке программирования. Объявляя переменную, мы храним в ней какое-либо значение, которое обычно встречается в CSS в виде цвета, шрифта или целого набора свойств, например для box-shadow.

Ниже приведены примеры переменных как в SCSS

Изображение выглядит как текст

Автоматически созданное описание

Идея в том, что данный подход упрощает процесс повторного использования переменных, а также мы можем быстро изменить значение конкретной переменной там, где мы её объявляем, вместо повсеместного перепечатывания кода.

Mixin

Mixin можно также представить как класс-конструктор в языке программирования: вы используете ряд свойств из CSS, создавая отдельный объект, который потом используете где хотите, задавая разные значения его свойствам.

Ниже приведён пример использования Mixin в коде:

Изображение выглядит как текст

Автоматически созданное описание

Ещё один способ упростить себе работу с помощью Mixin - использование его в местах, где требуются префиксы для адаптации под разные браузеры.

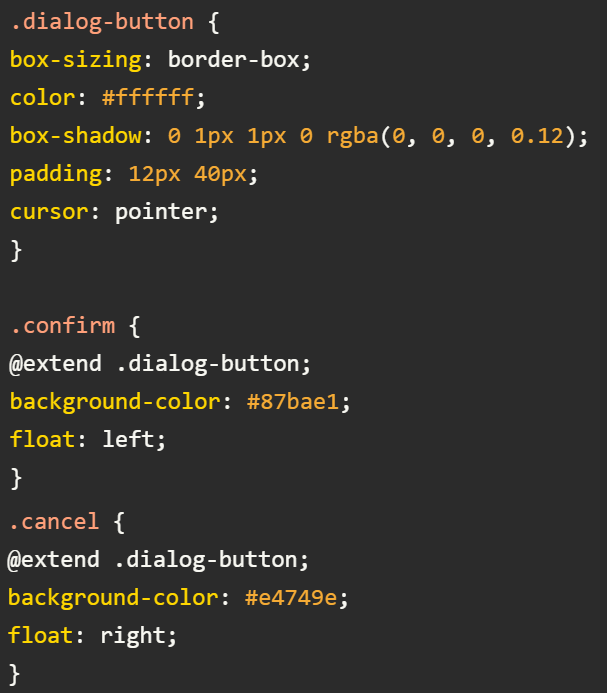
Изображение выглядит как текст

Автоматически созданное описание

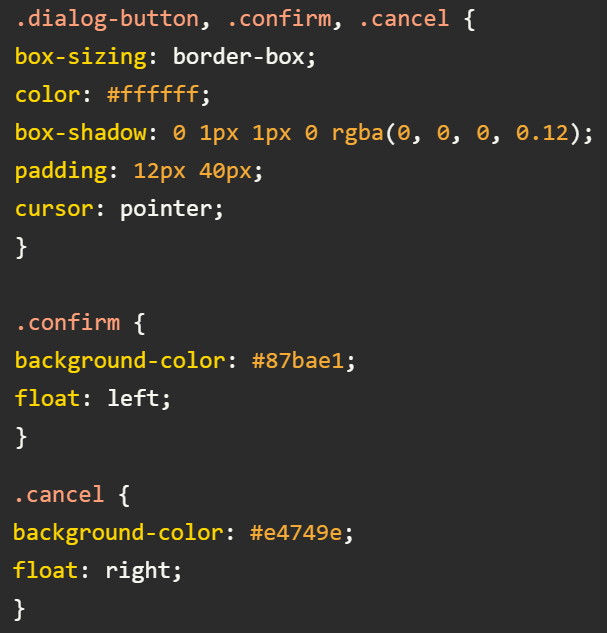
Extend

Следующая особенность, на которую мы взглянем, будет @extend, она позволяет вам наследовать CSS-свойства одного селектора от другого. Принцип работы напоминает Mixin, но Extend, как правило, используется для того, чтобы создать логическую связь между элементами страницы.

Extend используется, когда нам, к примеру, нужно два похожих элемента, которые имеют некоторые отличия. Например, давайте возьмём две кнопки: согласие и отмена.



Итог компиляции в css. Если вы взгляните на CSS код, то заметите, что Sass скомбинировал селекторы вместо повторения одних и тех же строк несколько раз в коде.



Вложенные конструкции

Как известно, в HTML, как правило, программист пишет код по принципу "гнездования". Иными словами, блоки кода находятся в других блоках кода и содержат вложенные блоки кода. CSS же в этом плане представляет собой полнейший хаос. Если для вас это проблема, Sass может помочь вам в организации кода.

Пример кода на SCSS

Изображение выглядит как текст

Автоматически созданное описание

И он же на CSS:

Изображение выглядит как текст

Автоматически созданное описание

Операции

Вы можете выполнять различные математические операции прямо в коде, что значительно упрощает работу в некоторых случаях.

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Функции

В Sass имеется целый ряд встроенных функций разного рода. К примеру, функции для операций со строками, цветами или выполняющие различные математические операции вроде random() или round().

Чтобы было нагляднее, представим функцию darken($color, $amount), которая, как понятно из названия, затемняет или применяет hover.

Фрагмент кода на SCSS

Изображение выглядит как текст

Автоматически созданное описание

Он же на css

Изображение выглядит как текст

Автоматически созданное описание

Также существует набор собственных функций SCSS: <https://sass-lang.com/documentation/modules>

ЗАДАЧИ:

1. Реализовать примеры, описанные в обучающей части и отобразить их в html файле (создать html файл, который будет использовать скомпилированный css)
2. Реализовать макет из файла семинара 5-6 с использованием SCSS, используя его возможности по максимуму