GeekBrains

Профессиональная вёрстка

Урок 3

Препроцессоры: применение на практике

Эффекты перехода. Применение трансформации.

Препроцессоры

Что такое препроцессоры

CSS-препроцессоры — это «программистский» подход к CSS. Они позволяют использовать при написании стилей свойственные 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.

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.

Sass

Первый шаг для начала работы – установка gem'a Sass:

gem install sass

Если для работы вы используете Windows, сначала необходимо установить Ruby. Для запуска Sass из командной строки используйте команду 1:

sass input.scss output.css

Также вы можете указать Sass следить за файлом и автоматически его компилировать в CSS при любом изменении:

sass --watch input.scss:output.css

Если у вас в папке имеется несколько файлов Sass, вы можете указать Sass следить за всей папкой:

sass --watch app/sass:public/stylesheets

Используйте команду sass --help для получения полной документации.

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 $.

Математические операторы

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

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.

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.

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 нескольку раз на сайте. Хорошо использовать миксины для вендорных префиксов.

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

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».

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!

Практическое задание

1. Сверстать страницу Shopping Cart.psd.
2. Написать стили с использованием любого препроцессора на ваш выбор.
3. * Перевести все стили на препроцессорный язык.

Дополнительные материалы

1. Sass vs. Less vs. Stylus – http://forwebdev.ru/css/sass-vs-less-vs-stylus/.
2. С чего начать изучение SCSS/SASS? https://toster.ru/q/101697.
3. Основы Sass: http://sass-scss.ru/guide/ , https://youtu.be/H4cG4tbc-xQ.
4. Подключение: http://sass-scss.ru/install/.
5. Почему Sass? https://web-standards.ru/articles/why-sass/.
6. CSS с суперсилой: http://sass-scss.ru/.
7. Компилятор Sass – http://koala-app.com/.
8. SCSS – немного практики, часть I: https://habrahabr.ru/post/140612/.
9. SCSS: пара полезных техник – https://habrahabr.ru/post/151679/.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. http://www.wisdomweb.ru/.
2. http://html5book.ru/css3-flexbox/.
3. http://sass-scss.ru/guide/.
4. Гоше Х. HTML5. Для профессионалов. – СПб.: Питер, 2013. – 496 с.: ил.
5. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. – СПб.: Питер, 2012. – 272 с.
6. Макфарланд Д. Большая книга CSS3. – СПб.: Питер, 2016. – 608 с.