Знакомство с препроцессорами

06 DECEMBER 2017

Привет, дорогой друг! Ранее мы познакомились с БЭМ и написали HTML и CSS для его воплощения. Но крутые frontend-разработчики не использует CSS. Они пишут на **препроцессорах** вместо этого.

CSS-препроцессор — это надстройка над CSS, которая добавляет ранее недоступные возможности для CSS. Т.е. мы можем использовать все то, что еще не заложено в стандарте. И это МЕГА-круто :) .

CSS-препроцессоры различны, но суть их от этого не меняется. Существует **Less, Sass (SCSS), Stylus** и др. Отличаются друг от друга они синтаксисом и несущественными дополнительными возможностями. Подробнее о препроцессорах можно узнать здесь. В дальнейшем мы рассмотрим примеры на SCSS.

Что позволяют делать препроцессоры?

1.Задавать переменные. Например, для цвета. Это удобно, т.к. убирает необходимость каждый раз вспоминать rgba/hex код.

Например, вместо многочисленных повторений HEX-кода цвета:

```
.main-class{
   color: #333;
 }

 .text{
   color: #333;
 }
```

```
  h1{
    color: #333;
  }
```

мы можем задать его один раз:

```
$black-light: #333;

 .main-class{
    color: $black-light;
  }

  .text{
    color: $black-light3;
  }

  h1{
    color: $black-light;
  }
```

Применимо для всевозможных свойств.

2.Вложенные селекторы. Если бы мы использовали только CSS, наш код выглядел бы так:

```
.new-message {
  font-family: Arial, Helvetica, sans-serif;
  border: 1px solid #000;
}

.new-message .date {
  color: #000;
}

.new-message .subtitle {
  color: #333;
  background: red;
}

.new-message .header {
  font-size: 32px;
  font-weight: 500;
}
```

Но препроцессор избавляет от написания повторяющейся фразы `new-message`, давая возможность воспользоваться **вложенностью**.

Код на препроцессоре:

```
.new-message {
  font-family: Arial, Helvetica, sans-serif;
  border: 1px solid #000;
  .date {
    color: #000;
  }

  .subtitle {
    color: #333;
    background: red;
  }

  .header {
    font-size: 32px;
    font-weight: 500;
  }
}
```

3.Возможность разбить код на разные части. Мы можем создать `partials` - короткий кусок кода, стилизующий отдельный элемент/выносящий все цвета и др. В общем, имеющий вполне конкретное назначение. А затем этот `partials` подключить в необходимую часть проекта с помощью `@import`.

Сделали отдельный файл для заголовка:

```
// _header.scss

header{
  background: red;
  color: #000;
  font-size: 16px;
}
```

```
// _main.scss

@import 'header.scss';

body {
```

```
  background: gray;
  font-family: Arial, Helvetica, sans-serif;
}
```

4.Конечно же, одна из самых важных возможностей, это - **миксины**. Миксины представляют собой кусок кода, который может быть переиспользован с параметрами.

```
// Это - миксин
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}
```

```
// А здесь мы его используем
.container_round { @include border-radius(5px); }
```

Результат не заставит себя долго ждать:

```
.container_round {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  -ms-border-radius: 5px;
  border-radius: 5px;
}
```

5.**Наследование и расширение**. Мы можем создать класс единожды и применить его для разных ситуаций.

```
// Создаем класс
.article{
   text-align: center;
   font-size: 12px;
   color: #333;
}

.article__left{
   @extend .article; // Применяем в рамках класса
   width: 50%;
   left: 0;
}
```

Результат получается таким же, как и в случае использования одного CSS:

```
// Создаем класс
.article__left{
   text-align: center;
   font-size: 12px;
   color: #333;
   width: 50%;
   left: 0;
}
```

6.С помощью препроцессоров мы оказываемся в силах использовать математические **операторы**. Например, `+, -, *, /, и %`. Особенно полезно для адаптивной верстки.

```
 .left-block{
    padding-right: 2px / 4px + 3px;
    opacity: random(4);
    font-size: 5em + 2; // 7em
  }
```

Но математикой дело не ограничивается. Также возможны операторы равенства, условные и сравнения.

```
$list: "tomato", "lime", "lightblue";

@mixin fg-color($property) {
    @each $item in $list {
        $color-length: str-length($item);
        @if( $color-length % 2 != 0 ) {
            #{$property}: unquote($item);
        }
    }
}

h2 {
    @include fg-color(color);
}
```

Также возможны строчные операторы

```
content: "В этот текст будет добавлена переменная #{$value}";
```

и операторы для отображения цветов:

```
color: rgba(255, 190, 0, 1) + rgba(206, 20, 171, 1);
```

Мы изучили азы. Более подробную информацию всегда можно найти в официальной документации (SASS, LESS, Stylus).

@Afelua **Share this post**

Read more posts by this author.

Moscow