Введение

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

Три самых известных препроцессора — это Less, SASS и Stylus. Они во многом похожи между собой, но имеют и ключевые различия. В этом и последующих частях цикла мы рассмотрим препроцессор Less.

в тренажёре по Less мы будем шаг за шагом строить небольшой ~~велосипед~~ фреймворк компонентов. Когда он будет готов, мы сможем собирать из компонентов цельные интерфейсы. Компонентный подход позволяет структурировать большие объёмы кода и легко масштабировать проекты. Препроцессор в этом деле — хорошее подспорье.

В этом тренажёре редактор CSS заменён на редактор Less. Чтобы увидеть скомпилированный из Less кода CSS код, можете использовать кнопку `CSS` в правом верхнем углу редактора.

Итак, теперь вы знаете всё, чтобы начать!

Переменные, шаг 1

Цветовая схема — основа любого дизайна в вебе. Применяя цветовое кодирование, можно сделать интерфейс более понятным. Поэтому первым делом при создании нашего мини-фреймворка давайте займёмся цветовой схемой основных элементов интерфейса. В этой задаче Less нам очень поможет.

В прошлом задании в коде встречалась подобная запись:

```
@navy: #1d365d;
```

Так в Less описываются переменные. Синтаксис переменных такой:

```
@название_переменной: значение_переменной;
```

Создав переменную один раз, можно использовать её в любом месте кода. Например:

```
background-color: @navy;

color: @navy;

border-color: @navy;
```

Во всех местах, где указана переменная, Less заменит строку `@navy` на `#1d365d`. Теперь, если понадобится изменить цвет, не нужно искать все его объявления в файле, а достаточно просто изменить значение переменной в одном месте.

Попробуем использовать переменные в коде: зададим с помощью переменной основной цвет нашей цветовой схемы.

. Переменные, шаг 2

Переменные можно объявлять как «снаружи» правил, так и «внутри». В случае «внутреннего» объявления переменная будет доступна только внутри правила, в котором она объявлена:

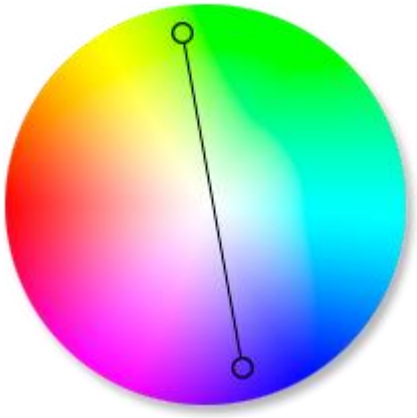
```
.rule-1 {
  @align: right;
  text-align: @align; // text-align задаётся значение right
}
.rule-2 {
  text-align: @align; // в этом месте произойдёт ошибка
}
```

Если переменная объявлена и «внутри» правила, и «снаружи» — Less применит «внутреннее» значение.

```
@align: left;

.rule-1 {
  @align: right;
  text-align: @align; // text-align задаётся значение right
}

.rule-2 {
  text-align: @align; // text-align задаётся значение left
}
```

Таким образом можно «переопределять» глобальные переменные в локальном контексте.

Цветовые функции, шаг 1



Итак, мы задали базовый цвет для нашей схемы, от него мы будем отталкиваться при выборе других цветов. Давайте узнаем, как это сделать.

Все цвета модели `RGB` можно расположить на цветовом колесе, где они плавно переходят друг в друга.

С помощью Less-функции `spin` можно повернуть цветовое колесо на определённый угол относительно заданного цвета и получить новый цвет. Функция принимает два параметра, синтаксис её такой:

```
spin(цвет, угол_поворота)
```

Цвет можно задавать в любом цветовом формате. Значение угла может быть как положительным, так и отрицательным. При положительном угле функция повернёт колесо по часовой стрелке, при отрицательном — против. Примеры:

```
color: spin(red, 90); // цвет повернётся от красного на 90° по часовой

border-color: spin(#f0f, -45); // цвет на 45° от #f0f против часовой
```

Противоположный цвет на колесе называется *комплементарным*. Он находится под углом `180°` к заданному цвету. Комплементарные цвета используют для создания контраста.

Вложенные правила, шаг 1

Отвлечёмся ненадолго от цвета и рассмотрим ещё одну замечательную особенность Less — вложенные правила. Они позволяют избавиться от дублирования одинаковых названий в коде и делают его более структурированным. Например, вот такой код:

```
.super-class-name {

  color: #ffffff;

}
```

```
.super-class-name a {
  text-decoration: none;
}

.super-class-name a span {
  font-size: 1em;
}
```

можно более кратко и без повторов написать, используя вложенность:

```
.super-class-name {
  color: #ffffff;

  a {
    text-decoration: none;
    span {
      font-size: 1em;
    }
  }
}
```

То есть вложенные правила просто пишутся внутри других правил. Из цепочек вложенных правил Less сам составляет итоговые селекторы.

Вложенные правила, шаг 2

С помощью вложенных правил можно не только обращаться к дочерним элементам, но и составлять по частям комплексные названия классов. Например, следующий код:

```
.super-button-red {
  color: red;
}

.super-button-blue {
  color: blue;
```

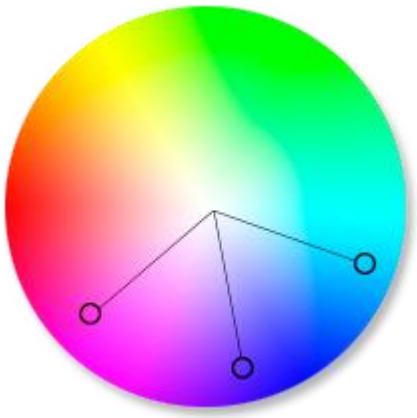
```
}
```

можно записать проще с помощью вложенных правил:

```
.super-button {

  &-red { color: red; }



  &-blue { color: blue; }

}
```

То есть, если перед вложенным правилом поставить амперсанд `&`, то оно станет *родственным* родительскому, и Less подставит родительский селектор вместо амперсанда.

Математические операции



Теперь давайте создадим третий цвет. Его «сдвинем» по кругу на `60°` в другую сторону от основного. Он светлее и будет использоваться для выделения второстепенной информации и элементов.

Чтобы «сдвинуть» цвет против часовой стрелки, нужно в функцию `spin` передать отрицательное значение переменной `@distance`.

Над любыми численными значениями в Less-коде можно произвести математические операции сложения, вычитания, умножения или деления:

```
padding-top: 10px + 20; // = 30px

padding-bottom: 100px - 50; // = 50px

font-size: 2em * 2; // = 4em

left: 50% / 2; // 25%
```

Less выполнит математическую операцию и вернёт в CSS уже вычисленное значение. Единицы измерения всегда берутся от первого параметра в выражении.

Чтобы вычислить отрицательное значение `@distance` просто умножим переменную на `-1`.

Математические операции, шаг 2

И теперь, когда палитра цветов построена, можно немного «поиграть» ~~со шрифтами~~ с базовым цветом и посмотреть, как вместе с ним будут меняться остальные.

Для этого давайте просто поменяем параметры в `RGB`-записи цвета в переменной `@base-color` математическими операциями. Кстати, к значению цвета «целиком» тоже можно применять операции. В случае суммы числа и `RGB`-записи цвета слагаемое будет прибавляться к каждому цветовому каналу одновременно:

```
rgb(10, 10, 10) + 10


// то же самое, что


rgb(20, 20, 20)
```

Цветовые функции, шаг 2

Less-функции `lighten` и `darken`. Их синтаксис одинаковый:

```
color: lighten(red, 50%); // светлее red на 50%

color: darken(blue, 25%); // темнее blue на 25%
```

Второе значение задаётся в процентах от `0%` до `100%`. При задании `100%` в `lighten` функция возвращает полностью белый цвет, а `100%` для `darken` — полностью чёрный. То есть эти функции «смешивают» заданный цвет с белым или чёрным.

Цветовые функции, шаг 3

Ещё две Less-функции для работы с цветом: `saturate` и `desaturate`. Они увеличивают и уменьшают насыщенность заданного цвета. Их синтаксис такой же как и у `lighten/darken`:

```
color: saturate(green, 20%); // green насыщеннее на 20%

color: desaturate(blue, 50%); // blue бледнее на 50%
```

Одновременно функции цвета можно использовать так:

```
// цвет светлее красного на 50% и насыщеннее на 20%
```

```
color: saturate(lighten(red, 50%), 20%);



// цвет темнее синего на 20% и бледнее на 50%

color: desaturate(darken(blue, 20%), 50%);
```

То есть функции можно «вкладывать» друг в друга, используя их в качестве аргументов.