SASS / SCSS

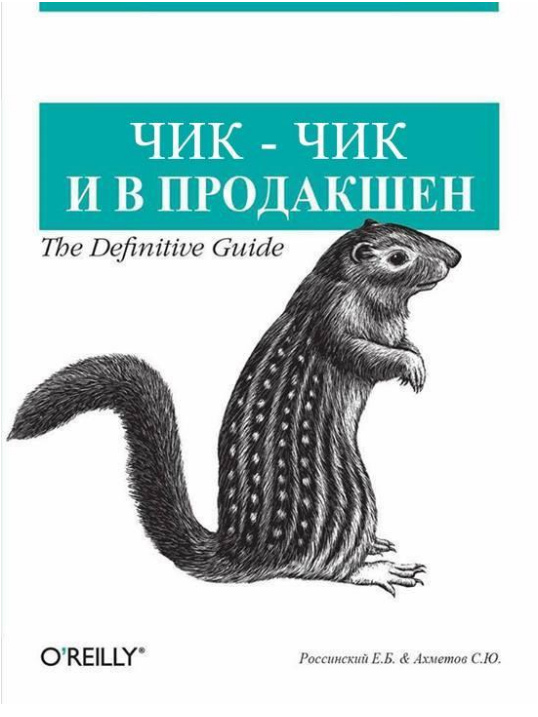
LoftSchool
от мыслителя к создателю

Препроцессоры

Препроцессоры в терминах веба - это программа которая транслирует входящий в нее код (как правило расширяющий возможности и упрощающий написание на стандартных языках) в код другого языка читаемый браузерами.

Ускорение процесса разработки

- вложенности селекторов
- составные селекторы
- арифметические операции
- переменные
- функции
- и много прочих вкусностей...



LoftSchool
от мыслителя к создателю

SASS или SCSS ?

SASS (Syntactically Awesome Style Sheets) - это язык который при помощи компилятора написанного на Ruby, транислируется в CSS.

sass

```
.class
  color: red
  border: 1px solid blue
```

SCSS - css-подобный диалект SASS (любой CSS код - это валидный код SCSS)

scss

```
.class {
  color: red;
  border: 1px solid blue;
}
```

LoftSchool
от мыслителя к создателю

Вложенности селекторов

Если написать дополнительный селектор внутри основных фигурных скобок, то это селектор становится **зависимым!**

scss

```scss
.class {
  color: red;

  .dependent-class {
    background: blue;
  }
}
```

css

```css
.class {
 color: red;
}

.class .dependent-class {
   background: blue;
}
```

LoftSchool
от мыслителя к создателю

Составные селекторы

Оператор - **&** - конкатенирует строку написанную после него с первоначальным селектором.

scss

```scss
.class {
  font-size: 12px;

  &__inner{
    color: red;
  }
  &:hover{
    font-size: 16px;
  }
  & + .class {
    font-size: 20px;
  }
}
```

css

```css
.class {
 font-size: 12px;
}
.class__inner {
 color: red;
}
.class:hover {
 font-size: 16px;
}
.class + .class {
 font-size: 20px;
}
```

LoftSchool
от мыслителя к создателю

Арифметические операции

SASS умеет производить операции над числами, но если будут складываться **разные** единицы измерения то будет ошибка.

scss

```scss
.class {
  padding: 0 20px;
  width: 400px - (20 * 2);
}


// будет ошибка
.error {
  padding: 20px + 30em;
}
```

css

```css
.class {
 padding: 0 20px;
 width: 360px;
}
```

LoftSchool
от мыслителя к создателю

Переменные

Переменные в **SASS** могут хранить что угодно - строки, числа, цвета, списки (массивы) и т.д. Имя переменной может начинаться либо с латиницы либо с подчеркивания.

scss

```scss
$color: #fff;
$width: 200px;
.class {
  color:  $color;
  width: $width - 50;
}

$content-text: 'контент';
.class {
  &:after {
     content: $content-text;
  }
}
```

css

```css
.class {
 color: #fff;
 width: 150px;
}

.class:after {
 content: "контент";
}
```

LoftSchool
от мыслителя к создателю

Используем функции

SASS содержит в себе огромное количество встроенных функций (работы с цветами, с массивами, со строками и пр.)

scss

```scss
.class {
  background: lighten(#000, 10%);
}
```

css

```css
.class {
 background: #1a1a1a;
}
```

LoftSchool
от мыслителя к создателю

Так же функции можно создавать

Для создания функций есть специальные конструкции **@function** и **@return** для возвращения результата.

scss

```scss
@function funcName($param) {
  $value : $param;

  @return $value;
}
```

LoftSchool
от мыслителя к создателю

Повторное использование кода

- примеси
- циклы
- наследование
- “плейсхолдеры”
- условия

Примеси (mixins)

Примеси созданы для генерации свойств в зависимости от входных параметров

scss

```scss
@mixin transition ($value){
  -webkit-transition: $value;
  -moz-transition: $value;
  -ms-transition: $value;
  -o-transition: $value;
  transition: $value;
}
.class{
  opacity : .8;
  @include transition(opacity  .3s);
}
```

css

```css
.class {
 opacity: .8;
 -webkit-transition: opacity 0.3s;
 -moz-transition: opacity 0.3s;
 -ms-transition: opacity 0.3s;
 -o-transition: opacity 0.3s;
 transition: opacity 0.3s;
}
```

LoftSchool
от мыслителя к создателю

Бесконечное число параметров

Иногда в свойство CSS можно передавать несколько параметров что бы в примесях это не вызывало ошибок, нужно применить оператор (...)

scss

```scss
@mixin transition ($value...) {
  -webkit-transition: $value;
  -moz-transition: $value;
  -ms-transition: $value;
  -o-transition: $value;
  transition: $value;
}

.class {
  @include transition(opacity  .3s, color .3s);
}
```

LoftSchool
от мыслителя к создателю

Циклы

Перебирать массивы можно при помощи конструкции **@each in**

scss

```scss
$drinks: beer, rum, absinthe;

@each $drink in $drinks {
  .my__lovely-#{$drink} {
    background: url('/img/for-#{$drink}');
  }
};
```

css

```css
.my__lovely-beer {
 background: url('/img/for-beer');
}

.my__lovely-rum {
 background: url('/img/for-rum');
}

.my__lovely-absinthe {
 background: url('/img/for-absinthe');
}
```

LoftSchool
от мыслителя к создателю

Условия

Конечно же есть **@if, @else** и **@else if**

scss

```scss
$bg: backgrounded, colored;

@each $type in $bg {
  .colored-block__#{$type} {
    @if $type == colored {
      background: red;
    } @else {
      background: url('/img/pic.jpg');
    }
  }
};
```

css

```css
.colored-block__backgrounded {
  background: url("/img/pic.jpg");
}

.colored-block__colored {
  background: red;
}
```

LoftSchool
от мыслителя к создателю

Наследование

Если нам нужно расширить или унаследовать свойства класса, мы можем применить директиву **@extend**

scss

```scss
.class {
  color: red;
  width: 200px;
  height: 300px;
}

.same-class {
  @extend .class;
  margin-left: 200px;
}
```

css

```css
.class, .same-class {
 color: red;
 width: 200px;
 height: 300px;
}

.same-class {
 margin-left: 200px;
}
```

LoftSchool
от мыслителя к создателю

Плейсхолдеры

Что бы не дублировать функциональные классы, можно использовать специальную конструкцию

scss

```scss
%inlineblock {
  display: inline-block;
  vertical-align: top;
}

.one {
  @extend %inlineblock;
}

.two {
  @extend %inlineblock;
}
```

scss

```scss
.one, .two {
  display: inline-block;
  vertical-align: top;
}
```

LoftSchool
от мыслителя к создателю

Импортирование стилей

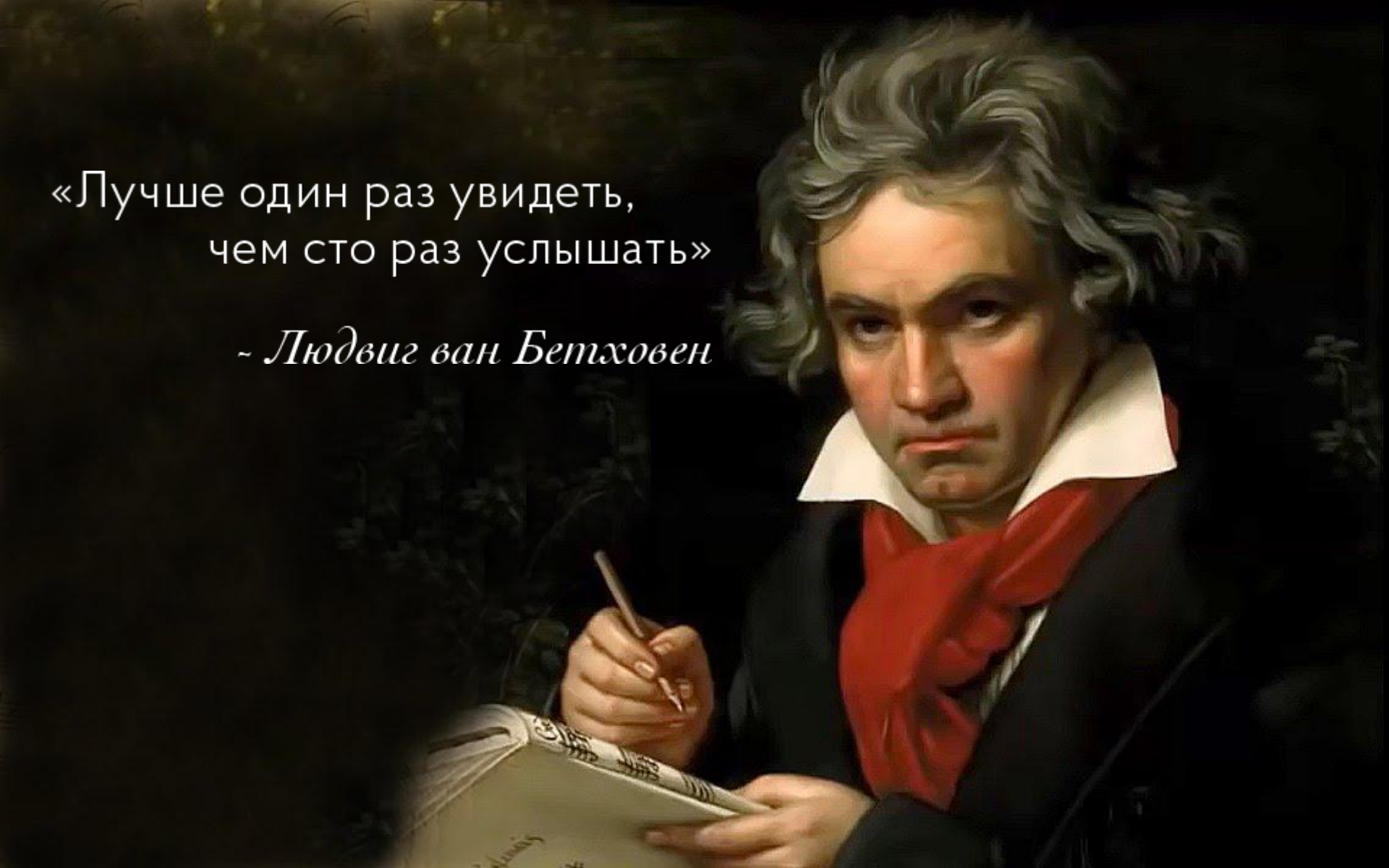
Импорт стилей

Одним из факторов влияющих на скорость загрузки страницы - это количество файлов подключеных в неё. Поэтому в документе должен быть подключен только один файл со стилями. Такие файлы называют **входной точкой (entry file).**

Но, что бы файл стилей не разрастался до огомных размеров, и весь код не сливался в одну “кучу” принято **разбивать CSS** на логические части. Это помогает реализовать конструкция **@import.**

sass

```
@import “путь/до/файла”
```


«Лучше один раз увидеть,
чем сто раз услышать»
- Людвиг ван Бетховен