

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

Preprocessors. LESS



Set it up

Несколько способов установить:

- подключить скрипт (локальный или с сервера, подробнее: <http://lesscss.org/>)
- Расширение в IDE (vscode: EasyLESS)
- через npm (node package manager): `npm install -g less`. Предварительно нужно скачать node.js и npm на рабочее устройство, чтобы команды npm были доступны.

После чего можно создать файл с расширением `.less`, в котором можно писать стили.

Если устанавливать less первым способом - указываем link на less файл, иначе на css файл, который будет автоматически создаваться.



Variables

В less существуют переменные. Решают проблему изменения одинаковых значений в стилях по всему проекту (например нужно все оранжевые цвета поменять на черные).

Синтаксис:

```
@base-color: rgba(80,80,80,0.3);
```

```
body{
```

```
background: @base-color;
```

```
}
```

```
.container {
```

```
background: @base-color;
```

```
}
```



Nesting

Еще одна проблема - перегруженные селекторы и дублирование селекторов:

```
.my-link, my-link:hover, my-link:hover:not(.disabled)
```

```
.my-link {
```

```
  &:hover {
```

```
    &:not(.disabled) {
```

```
    }
```

```
  }
```

```
}
```



Nesting. BEM and etc.

Nesting хорошо работает с BEM

```
.my-block {
```

```
  &__element {
```

```
    &--modifier {
```

```
    }
```

```
  }
```

```
}
```



Operations

- Смешение цветов:

```
@color: green;
```

```
h1 {  
  color: @color + #aa0010;  
}
```

- математические операции:

```
@base-border-width: 1px;  
h1 {  
  border: @base-border-width + 10 solid red;  
}
```



Built-in functions

```
@color: red;
```

```
.lighter {
```

```
background: lighten(@color, 30%);
```

```
}
```

```
.saturated {
```

```
background: saturate(@color, 40%);
```

```
}
```

```
.darker {
```

```
background: darken(@color, 25%);
```

```
}
```



Mixins

Миксины - возможность переиспользовать набор свойств и значений:

```
.triangle{
```

```
width: 1px;
```

```
height: 1px;
```

```
border: 30px solid transparent;
```

```
border-top: 30px solid red;
```

```
}
```




Mixins

Переиспользовать свойства класса `.triangle` можно, например, так:

```
.with-triangle {  
  
  &:after {  
  
    content: " ";  
  
    .triangle;  
  
  }  
  
}
```



Mixins with parameters

Что, если мы хотим переиспользовать треугольник, но цвета каждый раз нужны разные?

Можно создать миксин с параметрами того, что может меняться:

```
.triangle(@color) {  
  
width: 1px;  
  
height: 1px;  
  
border: 30px solid transparent;  
  
border-top: 30px solid @color;  
  
}
```



Namespacing

Чтобы убедиться, что наши селекторы не пересекаются с селекторами сторонних библиотек, можно задать пространство имен. Тогда мы можем избежать потенциального конфликта селекторов:

```
#project {  
  
  .btn {  
  
    color: red;  
  
  }  
  
}
```



Namespacing

После чего можно использовать эти стили следующим образом:

```
my-button {  
  
  #project > .btn;  
  
}
```

Т.е. мы переиспользовали стили из пространства имен: project