Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

Занятие 10

CSS-препроцессоры и преимущества их использования

План занятия

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

1. Введение
2. Основные возможности препроцессоров
3. Примеры использования
4. PostCSS
5. Про специфичность

https://sass-scss.ru/

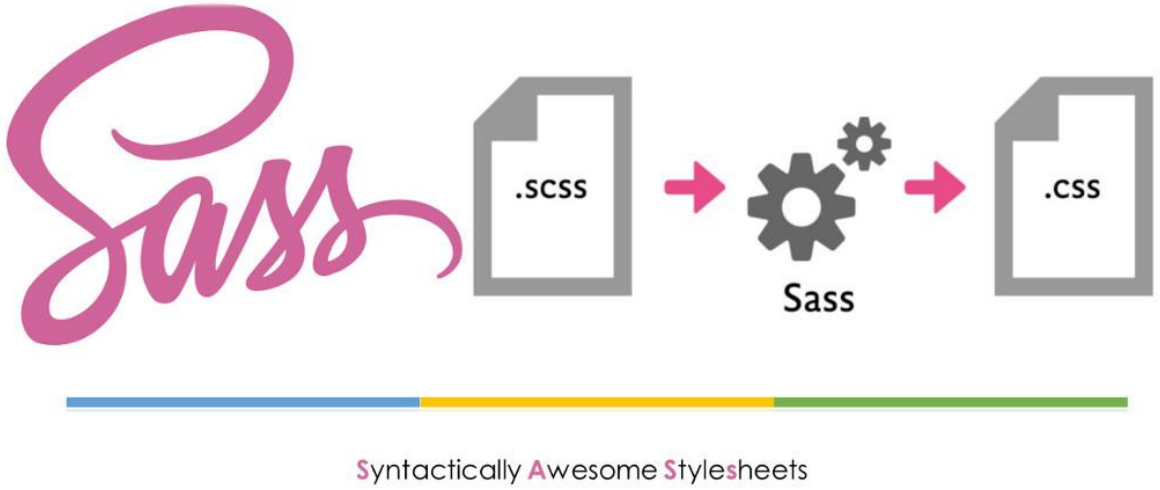
Введение

Препроцессоры CSS были созданы с одной единственной целью - добавить стилевым таблицам CSS мощь и одновременную гибкость без нарушения кросс-браузерности.

Рассмотри преимущество использования любого препроцессора на примере из трех описанных Sass, LESS и Stylus.

Введение

Все препроцессоры компилируют созданный с помощью их синтаксиса код в стандартный CSS-код, который понимает и использует любой браузер, каким бы древним он (*браузер*) не был.



Синтаксис

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Наиболее важной частью при написании кода в любом препроцессоре CSS является его синтаксис.

Все три препроцессора, которые мы будем рассматривать, имеют CSS-подобный вид.

Sass & Less

```
h1{
  color: #0982c1;
}
```

Stylus

```
h1 {
  color: #0982C1;
}
/* опущены фигурные скобки */
h1
  color: #0982C1;
/* опущены фигурные скобки, двоеточия и точки
с запятой */
h1
  color #0982C1
```

Псевдоклассы

Родительская ссылка (&) позволяет сгруппировать ваши правила так, чтобы все, что относится к определенному элементу, можно было красиво сгруппировать.

```
.someElement {
    some-rule: some-value;
    &:hover {
        color: some-color;
    }
}
```

Переменные

В препроцессорах переменные объявляются и используются внутри файлов стилей CSS.

Переменные могут принимать любое значение, допустимое в CSS (цвет, число или текст) и может ссылаться из любого места CSS-документа.

Sass

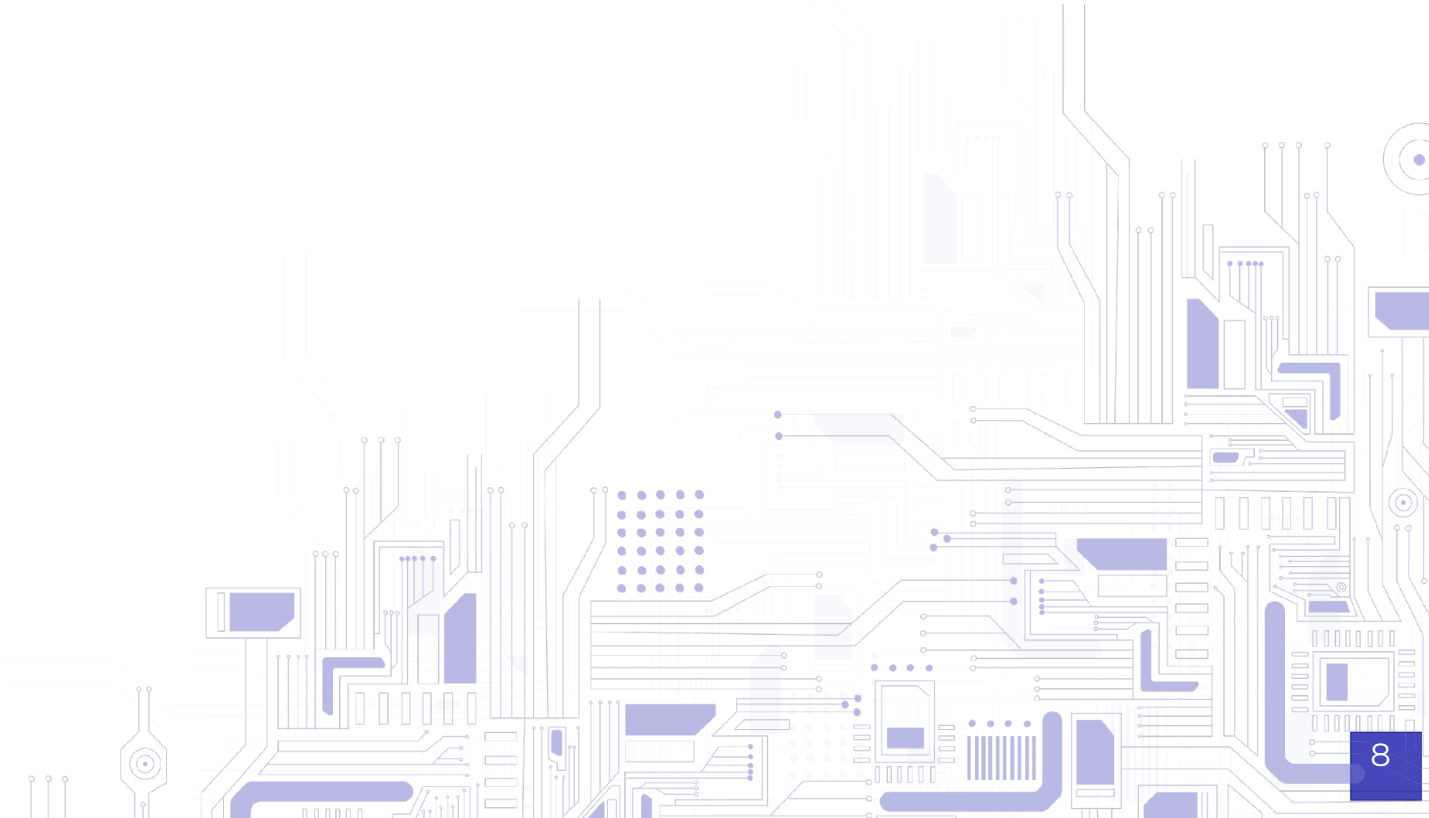
```
$mainColor: #0982c1;
$siteWidth: 1024px;
$borderStyle: dotted;

body {
  color: $mainColor;
  border: 1px $borderStyle $mainColor;
  max-width: $siteWidth;
}
```

Переменные

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Less

```
@mainColor: #0982c1;
@siteWidth: 1024px;
@borderStyle: dotted;

body {
  color: @mainColor;
  border: 1px @borderStyle @mainColor;
  max-width: @siteWidth;
}
```



Переменные

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Stylus

```
mainColor = #0982c1
siteWidth = 1024px
$borderStyle = dotted

body
  color mainColor
  border 1px $borderStyle mainColor
  max-width siteWidth
```

Переменные

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

CSS-переменные

```css
:root {
  --color-primary: #235ad1;
}

.section {
  border: 2px solid var(--color-primary);
}

.section-title {
  color: var(--color-primary);
}

.section-title::before {
  /* Другие стили */
  background-color: var(--color-primary);
}
```

Переменные

CSS-переменные

```css
/* Правильные имена */
:root {
  --primary-color: #222;
  --_primary-color: #222;
  --12-primary-color: #222;
  --primay-color-12: #222;
}

/* Неправильные имена */
:root {
  --primary color: #222; /* Пробелы использовать нельзя */
  --primary$%#% #
}
```

Переменные

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

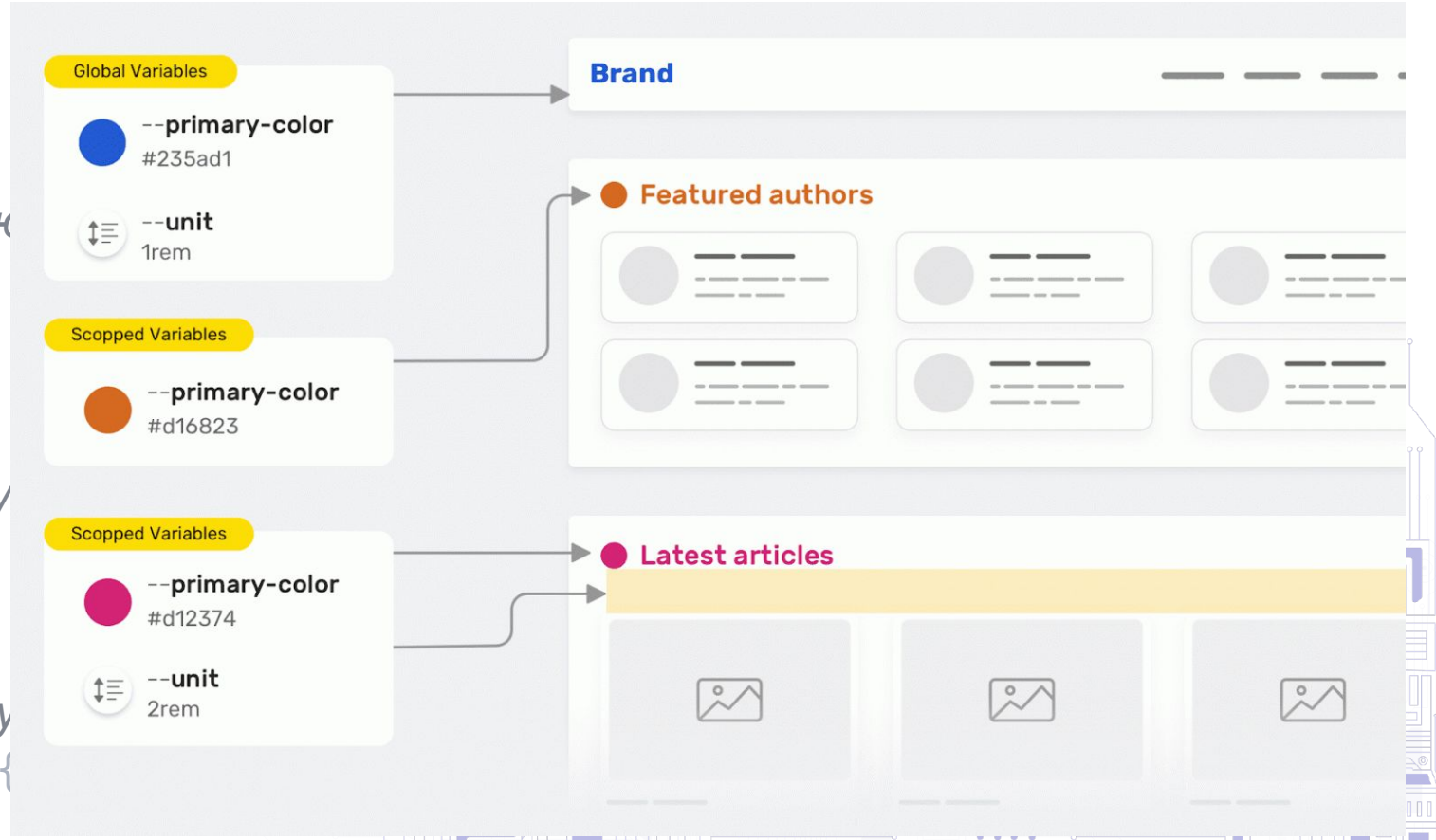
Области видимости

```
:root {
  --primary-color: #235ad1;
}

.section-title {
  --primary-color: d12374;
  color: var(--primary-color);
}
```

Переменные

Области видимости

```
/* Глобальные переменные */
:root {
  --primary-color: #235ad1;
  --unit: 1rem;
}
/* Для цвета и отступа применяк
.section-title {
  color: var(--primary-color);
  margin-bottom: var(--unit);
}
/* Переопределение переменной, у
.featured-authors .section-title
  --primary-color: #d16823;
}
/* Переопределение переменных, у
.latest-articles .section-title {
  --primary-color: #d12374;
  --unit: 2rem;
}
```



Переменные

Резервные значения

```
.section-title {
  color: var(--primary-color, #222);
}
```

```
.section-title {
  color: var(--primary-color, var(--black, #222));
}
```

Этот подход к работе с переменными может оказаться полезным в том случае, если значение переменной зависит от некоего действия.

Если может случиться так, что в переменной не будет значения, важно предусмотреть использование резервного значения.

Вложенность (nesting)

Если в коде CSS поставлена задача обратиться одновременно к нескольким элементам, имеющим одного и того же родителя, то писать снова и снова этого родителя - занятие утомительное.

```
section {
    margin: 10px;
}

section nav {
    height: 25px;
}

section nav a {
    color: #0982C1;
}

section nav a:hover {
    text-decoration: underline;
}
```

Вложенность (nesting)

Sass, Less, Stylus

```
section {
  margin: 10px;

  nav {
    height: 25px;

    a {
      color: #0982C1;

      &#038;:hover {
        text-decoration: underline;
      }
    }
  }
}
```

Подмешивания (mixins)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Mixins - являются функциями, которые позволяют многократно использовать сгруппированные свойства внутри CSS-кода.

Вместо того, чтобы просматривать весь код в поисках нужных строчек для их изменения, теперь можно вносить изменения только один раз, внутри миксина.

Применение подмешиваний особенно оправдывает себя при создании специфичных стилей для элементов или простановке браузерных префиксов.

Когда миксин вызывается внутри CSS-селектора, происходит распознавание аргументов этого миксина, затем его стили применяются к селектору, который его вызвал.

Подмешивания (mixins)

Sass

```
@mixin error($borderWidth: 2px) {
  border: $borderWidth solid #F00;
  color: #F00;
}

.generic-error {
  padding: 20px;
  margin: 4px;
  @include error(); /* Подключается миксин по имени error */
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  @include error(5px); /* Подключается миксин по имени error со
значением аргумента $borderWidth, равным 5px; то есть происходит
переопределение значения аргумента */
}
```

Подмешивания (mixins)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Less

```
.error(@borderWidth: 2px) {
  border: @borderWidth solid #F00;
  color: #F00;
}

.generic-error {
  padding: 20px;
  margin: 4px;
  .error(); /* Подключается миксин по имени error */
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  .error(5px); /* Подключается миксин по имени error со значением
аргумента $borderWidth, равным 5px; то есть происходит
переопределение значения аргумента */
}
```

Подмешивания (mixins)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Stylus

```
error(borderWidth= 2px) {
  border: borderWidth solid #F00;
  color: #F00;
}

.generic-error {
  padding: 20px;
  margin: 4px;
  error(); /* Подключается миксин по имени error */
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  error(5px); /* Подключается миксин по имени error со значением
аргумента $borderWidth, равным 5px; то есть происходит
переопределение значения аргумента */
}
```

Наследование (inheritance)

При написании CSS стилей “классическим” способом, для того чтобы применить одни и те же свойства к нескольким элементам в HTML-документе, нам следовало бы создать такой код:

```
p,
ul,
ol {
  /* какие-то стили здесь */
}
```

Наследование (inheritance)

При написании CSS стилей “классическим” способом, для того чтобы применить одни и те же свойства к нескольким элементам в HTML-документе, нам следовало бы создать такой код:

```
p,
ul,
ol {
  /* какие-то стили здесь */
}
```

Все работает прекрасно. Но если потребуется написать стили отдельно для любого из этих селекторов, нужно будет создавать отдельные правила для каждого из них.

И сразу же код таблиц стилей становиться неряшливым и трудно поддерживаемым.

В противоположность этому применяются наследование. Наследование - это возможность для одних CSS-селекторов наследовать свойства у другого селектора.

Наследование (inheritance)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Sass, Stylus

```
.block {
  margin: 10px 5px;
  padding: 2px;
}

p {
  @extend .block; /* Наследовать свойства у селектора класса .block
*/
  border: 1px solid #EEE;
}

ul, ol {
  @extend .block; /* Наследовать свойства у селектора класса .block
*/
  color: #333;
  text-transform: uppercase;
}
```

Наследование (inheritance)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Less

Препроцессор LESS не поддерживает наследование в полной мере так, как это организовано в Sass или Stylus.

Вместо добавления множественных селекторов в один набор свойств, наследование трактуется как миксин без аргументов.

Импорт стилей выполняется для каждого селектора.

Обратной стороной такого подхода является постоянное повторение строк со свойствами в компилированном CSS-стиле.

Наследование (inheritance)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Less

```less
.block {
  margin: 10px 5px;
  padding: 2px;
}

p {
  .block; /* Наследование свойств у селектора класса .block */
  border: 1px solid #EEE;
}

ul, ol {
  .block; /* Наследование свойств у селектора класса .block */
  color: #333;
  text-transform: uppercase;
}
```

Импортирование

В CSS-сообществе к импортированию стилей с помощью директивы **@import**

Существует стойкое негативное отношение, так как такой подход порождает множественные HTTP-запросы к серверу, что замедляет работу браузера и нагружает сам сервер.

Однако, в препроцессорах технология импортирования работает иначе.

В любом из трех препроцессоров импортирование одного файла внутрь другого фактически приводит к вставке кода одного файла в другой при компиляции, в результате которой получается один CSS-файл.

Обратите внимание, что при компилировании файла со стандартным подключением с помощью директивы `@import "file.css"` *внутри него компиляции последнего не происходит. А вот миксины или переменные импортируются и используются в стилевом файле, как положено.*

Импортирование

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

```
/* file.{type} */

body {
  background: #EEE;
}

@import "reset.css";
@import "file.{type}";

p {
  background: #0982C1;
}
```

```
@import "reset.css";

body {
  background: #EEE;
}

p {
  background: #0982C1;
}
```

Функции работы с цветом

“Цветовые” функции созданы для трансформации цвета при компиляции. Такие функции чрезвычайно полезны при создании градиентов, затемнения цвета при hover и многое другое.

Sass

```
lighten($color, 10%); /* возвращает цвет на 10% светлее чем $color
*/
darken($color, 10%);  /* возвращает цвет на 10% темнее чем $color
*/

saturate($color, 10%);   /* возвращает цвет на 10% более
насыщенный чем $color */
desaturate($color, 10%); /* возвращает цвет на 10% менее
насыщенный чем $color */

grayscale($color);  /* возвращает шкалу полутонов цвета $color */
complement($color); /* returns complement color of $color */
invert($color);     /* возвращает инвертированный цвет от $color
*/

mix($color1, $color2, 50%); /* возвращает результат смешивания
```

Функции работы с цветом

Представленный код является всего лишь кратким списком функций работы с цветом в Sass. Полный список всех доступных функций расположен по адресу Sass Documentation.

“Цветовые” функции могут использоваться везде, где требуется работа с цветов в коде.

Простой пример - объявлена переменная с цветом, к которой дальше в коде применяется функция затемнения цвета darken

```
$color: #0982C1;

h1 {
  background: $color;
  border: 3px solid darken($color, 50%);
}
```

Функции работы с цветом

Less

```
lighten(@color, 10%); /* возвращает цвет на 10% светлее чем $color
*/
darken(@color, 10%);  /* возвращает цвет на 10% темнее чем $color
*/

saturate(@color, 10%);   /* возвращает цвет на 10% более
насыщенный чем $color */
desaturate(@color, 10%); /* возвращает цвет на 10% менее
насыщенный чем $color */

spin(@color, 10);  /* возвращает цвет, смещенный на 10 градусов
вправо относительно цвета @color */
spin(@color, -10); /* возвращает цвет, смещенный на 10 градусов
влево относительно цвета @color */

mix(@color1, @color2); /* возвращает результат смешивания цвета
$color1 с цветом $color2 */
```

Функции работы с цветом

Less

Список функций препроцессора LESS находится на официальном сайте проекта LESS Documentation.

```
@color: #0982C1;

h1 {
  background: @color;
  border: 3px solid darken(@color, 50%);
}
```

Функции работы с цветом

Stylus

```
lighten(@color, 10%); /* возвращает цвет на 10% светлее чем $color
*/
darken(@color, 10%);  /* возвращает цвет на 10% темнее чем $color
*/

saturate(@color, 10%);   /* возвращает цвет на 10% более
насыщенный чем $color */
desaturate(@color, 10%); /* возвращает цвет на 10% менее
насыщенный чем $color */
```

Полный список всех функций работы с цветом препроцессора Stylus представлен на сайте проекта Stylus Documentation.

```
color = #0982C1

h1
  background color
  border 3px solid darken(color, 50%)
```

Арифметические операции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Благодаря препроцессорам выполнение арифметических операций внутри CSS-кода теперь осуществляется просто и легко.

С CSS3 появилась calc() для этих же целей

Sass, Less, Stylus

```
body {
  margin: (14px/2);
  top: 50px + 100px;
  right: 100px - 50px;
  left: 10 * 10;
}
```

Арифметические операции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Благодаря препроцессорам выполнение арифметических операций внутри CSS-кода теперь осуществляется просто и легко.

С CSS3 появилась calc() для этих же целей

Sass, Less, Stylus

```
body {
  margin: (14px/2);
  top: 50px + 100px;
  right: 100px - 50px;
  left: 10 * 10;
}
```

Условия

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Условие вычисляется во время компиляции, а не во время выполнения.

```
$type: line;
p {
  @if $type == line1 {
    color: blue;
  } @else if $type == line2 {
    color: red;
  } @else if $type == line3 {
    color: green;
  } @else {
    color: black;
  }
}
```

Еще примеры: https://www.geeksforgeeks.org/sass-if-and-else/

Управляющие конструкции

Управляющие конструкции препроцессоров — основа, позволяющая сократить исходный код в несколько раз и сделать возможным написание собственного CSS-фреймворка.

Все управляющие конструкции SASS начинаются с символа @: «коммерческое at». В SASS их целых четыре вида.

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Управляющие конструкции препроцессоров — основа, позволяющая сократить исходный код в несколько раз и сделать возможным написание собственного CSS-фреймворка.

Все управляющие конструкции SASS начинаются с символа @: «коммерческое at». В SASS их целых четыре вида.

@for

@if

@each

@while

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

@for

Запись
`@for $var from <start> through <end>`
читается как: для каждого элемента от стартовой точки до конечной.

Представьте, что у вас 100 иконок с разным значением `background-image`.

Разумеется, базовый класс, общий для всех, тоже есть. Что делать?

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

@for

Итерацию с **@for**

```
$img: 'assets/images';  /* физическое расположение иконок */
$class-slug: 'icon';    /* базовое имя класса */
/* модифицировать каждую из ста иконок */
@for $i from 1 through 100 {
  .#{$class-slug}__#{$i} {
    background-image: url("#{$img}/#{$class-slug}-#{$i}.svg");
  }
}
/* получаем в итоге */
.icon__1 {
  background-image: url("assets/images/icon-1.svg");
}
.icon__100 {
  background-image: url("assets/images/icon-100.svg");
}
```

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@if

Обычное условие. Часто используется в подпрограммах. Код ниже описывает подпрограмму, которая принимает аргумент `boolean`. Если условие истинно, значение `display` элемента блочное, в противном случае — `flexbox`

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@if

```
$boolean: true !default;
/* создание подпрограммы */
@mixin display-type($boolean) {
  @if $boolean == true {
    display: block;
  }
  @else {
    display: flex;
  }
}
/* включаем наш mixin с аргументом, отличным от false, в класс .block-type */
.block-type {
  @include display-type(false);
}
/* получаем в итоге */
.block-type {
  display: flex;
}
```

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@each

Выручает, когда вы имеете дело с массивом. SASS будет использовать `@list` для вычленения из него всех указанных элементов. В данном случае задача: менять аватары авторов.

```
$list: (adam, john, wynn, mason, ivan);
.author-images {
  @each $author in $list {
    &--#{$author} {
      background: image-url("#{$img}/#{$author}.png") no-repeat;
    }
  }
}
/* получаем в итоге */
.author-images--adam {
  background: image-url("assets/images/adam.png") no-repeat;
}
.author-images--john {
  background: image-url("assets/images/john.png") no-repeat;
}
```

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

@each

Еще.

Изначально задаём нужные параметры (в примере цвет), а затем для каждой своё название и соответствующий `background`

Обратите внимание на то, что массив увеличился, теперь используются по два значения на итерацию.

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@each

```scss
/* цвета удобнее задать в одном месте */
$twi:   #41b7d8;
$fb:    #3b5997;
$gplus: #d64937;
/* массив $social со значениями: название и цвет */
$social: (twitter, $twi),
         (facebook, $fb),
         (googleplus, $gplus);
/* перебираем */
@each $socialnetwork, $color in $social {
  .social-link--#{$socialnetwork} {
    background-color: $color;
    &:focus,
    &:hover {
      background-color: darken($color, 5%);
    }
  }
}
/* получаем в итоге */
.social-link--twitter {
  background-color: #41b7d8;
}
.social-link--twitter:focus, .social-link--twitter:hover {
  background-color: #2cafd4;
}
```

Управляющие конструкции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@while

Для цикла while нужны два параметра: переменная и шаг. Пока величина шага удовлетворяет условию, происходит преобразование переменной. При каждой итерации шаг увеличивается.

```
$color: #e44;
$step: 1;
@while $step <= 5 {
  .palette-#{$step} {
    background-image: linear-gradient(
      to bottom,
      darken($color, ($step * 2%)) 0%,
      darken($color, ($step * 10%)) 100%
    );
    $step: $step + 1;
  }
}
/* получаем в итоге */
.palette-1 {
  background-image: linear-gradient(to bottom, #ed3b3b 0%, #ea1515 100%);
}
.palette-2 {
  background-image: linear-gradient(to bottom, #ec3131 0%, #bb1111 100%);
}
```

Комментирование

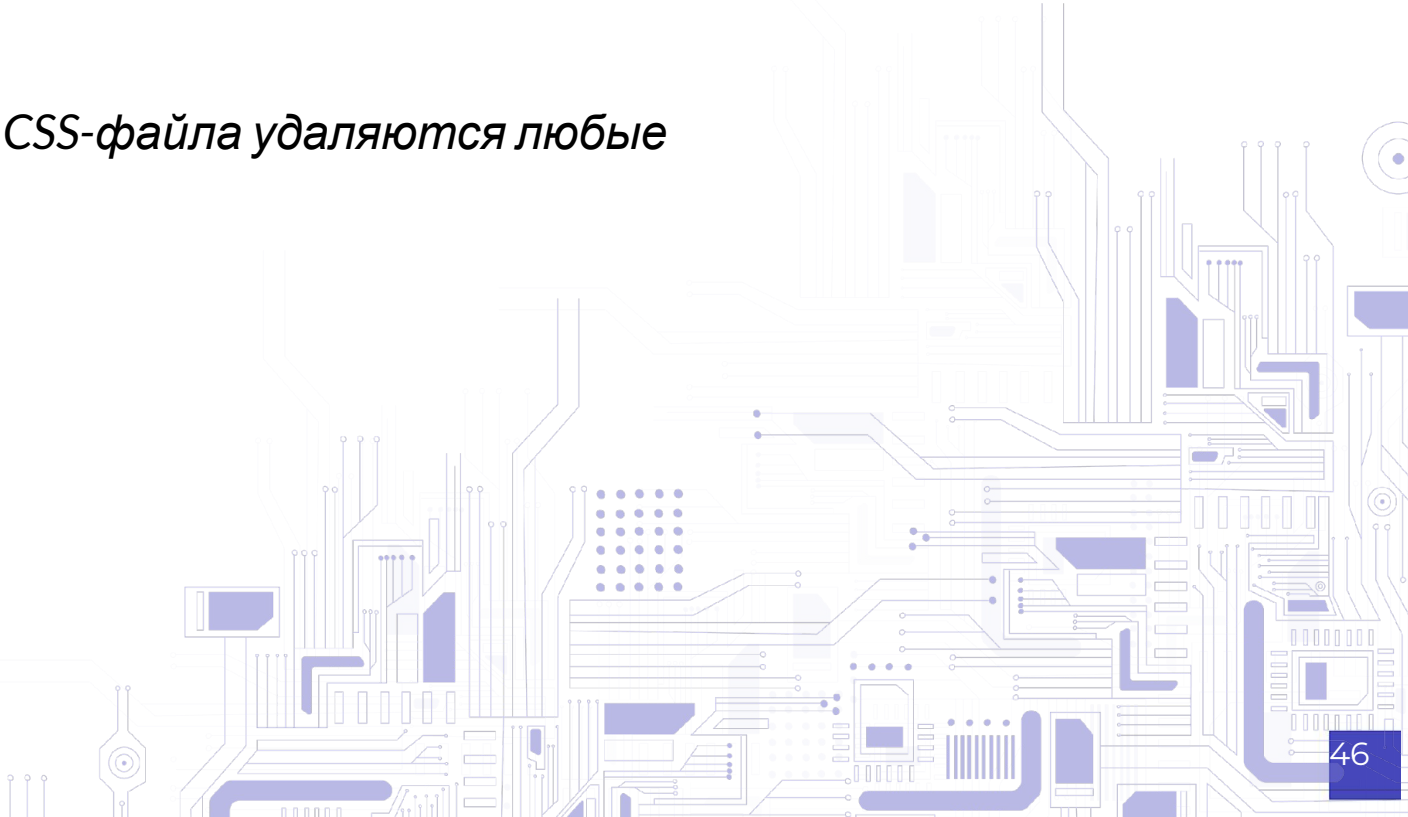
В CSS-препроцессоре при компиляции из кода удаляются любые комментарии в виде двойного слеша

// комментарий

А блочные комментарии остаются

/* комментарий */

При компиляции в минимизированную версию CSS-файла удаляются любые комментарии.



Примеры

Браузерные префиксы

Одним из наиболее ярких примеров преимущества использования препроцессоров является написание с их помощью свойств с браузерными префиксами. Один раз создав миксин с поддержкой браузерных префиксов, мы избавляем себя от рутинной работы.

Создадим для всех трех препроцессоров миксин скругления углов блока:

Sass

```
@mixin border-radius($values) {
  -webkit-border-radius: $values;
     -moz-border-radius: $values;
          border-radius: $values;
}

div {
  @include border-radius(10px);
}
```

Примеры

Браузерные префиксы

Less

```
.border-radius(@values) {
  -webkit-border-radius: @values;
     -moz-border-radius: @values;
          border-radius: @values;
}

div {
  .border-radius(10px);
}
```

Примеры

Браузерные префиксы

Stylus

```
border-radius(values) {
  -webkit-border-radius: values;
     -moz-border-radius: values;
          border-radius: values;
}

div {
  border-radius(10px);
}
```

Примеры

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Трехмерный текст

Создание эффекта трехмерности для текста с помощью CSS-свойства text-shadow

Sass

```
@mixin text3d($color) {
  color: $color;
  text-shadow: 1px 1px 0px darken($color, 5%),
               2px 2px 0px darken($color, 10%),
               3px 3px 0px darken($color, 15%),
               4px 4px 0px darken($color, 20%),
               4px 4px 2px #000;
}

h1 {
  font-size: 32pt;
  @include text3d(#0982c1);
}
```

Примеры

Трехмерный текст

Less

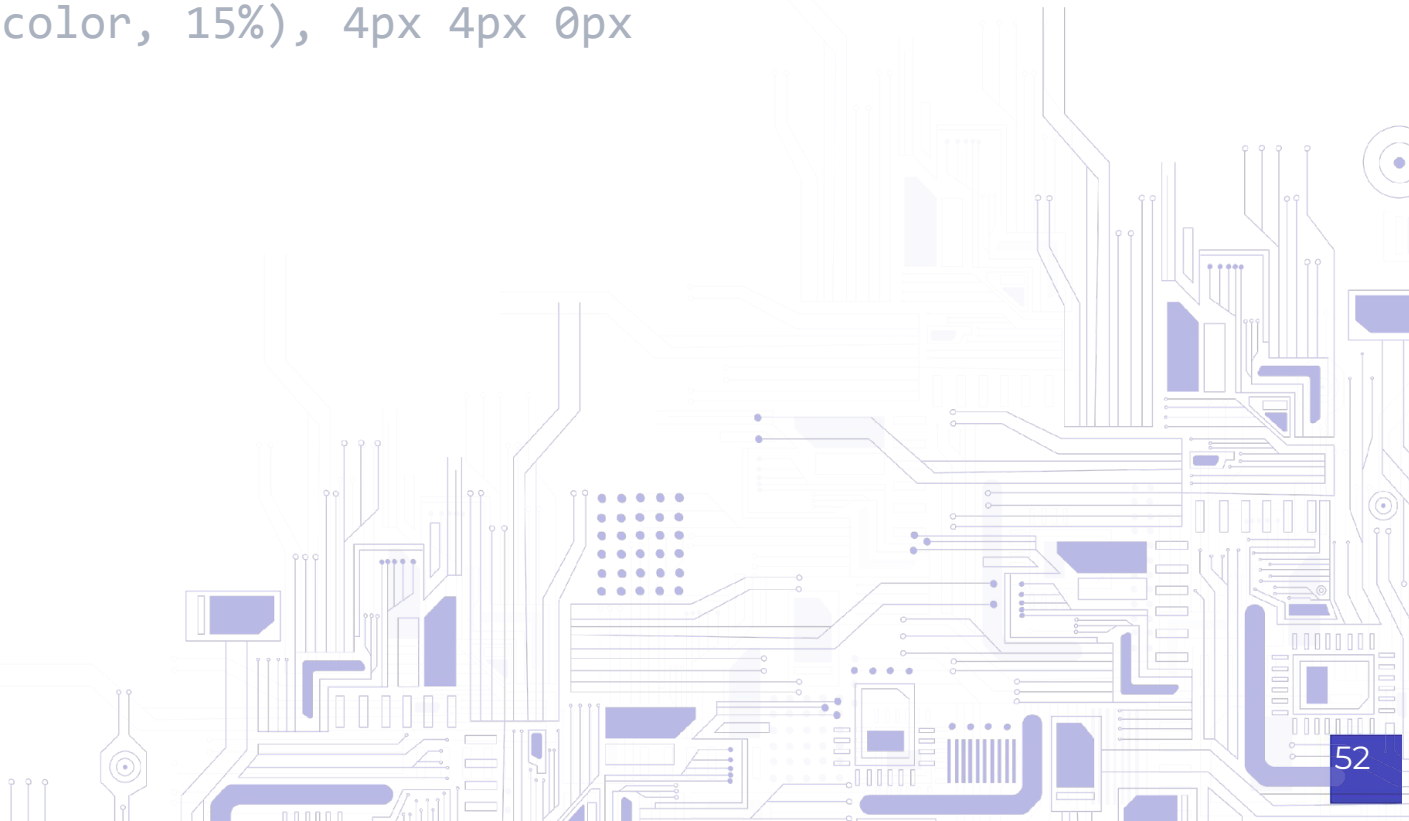
```
.text3d(@color) {
  color: @color;
  text-shadow: 1px 1px 0px darken(@color, 5%),
               2px 2px 0px darken(@color, 10%),
               3px 3px 0px darken(@color, 15%),
               4px 4px 0px darken(@color, 20%),
               4px 4px 2px #000;
}

span {
  font-size: 32pt;
  .text3d(#0982c1);
}
```

Примеры

Трехмерный текст

Stylus

```
text3d(color)
  color: color
  text-shadow: 1px 1px 0px darken(color, 5%), 2px 2px 0px
darken(color, 10%), 3px 3px 0px darken(color, 15%), 4px 4px 0px
darken(color, 20%), 4px 4px 2px #000
span
  font-size: 32pt
  text3d(#0982c1)
```

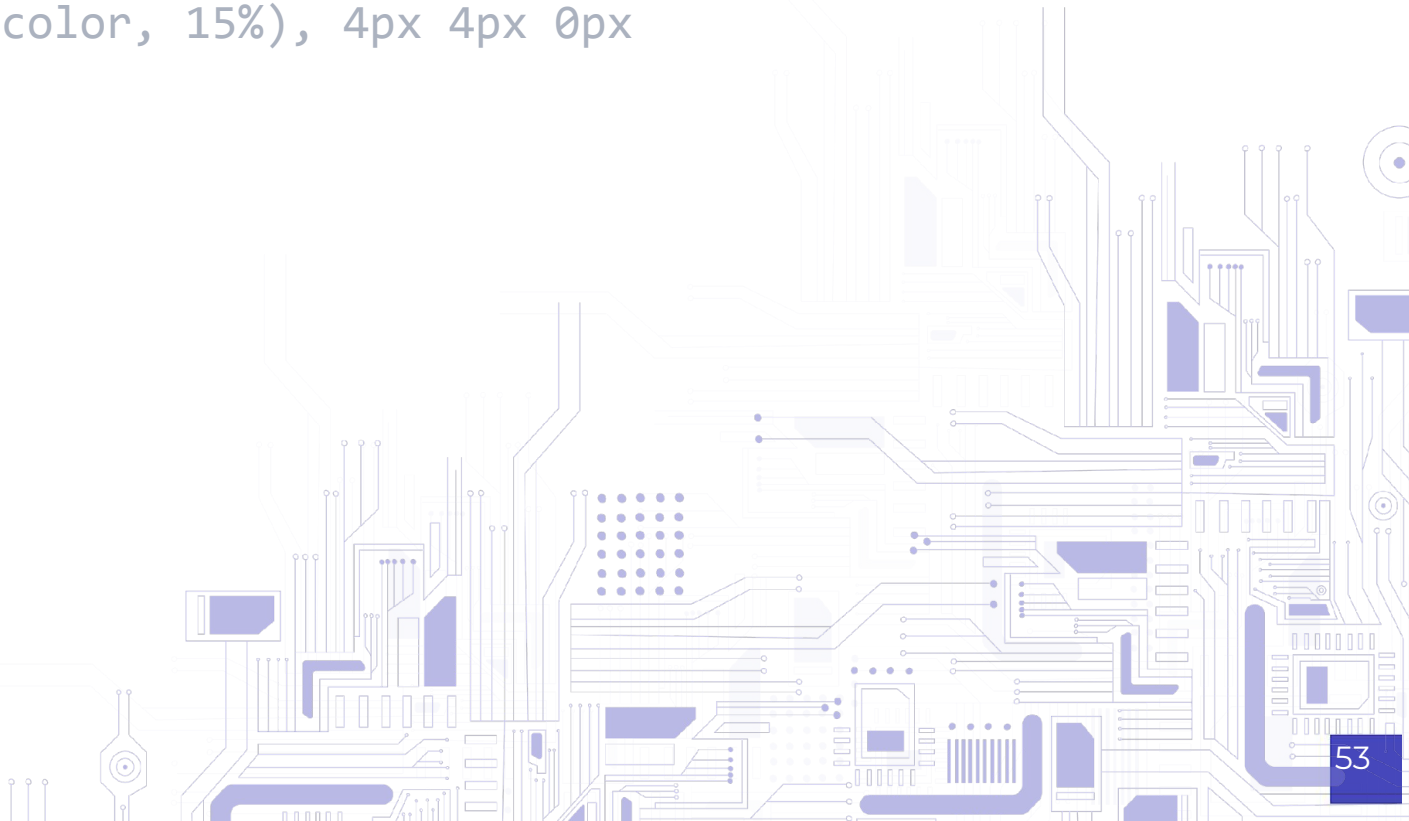


Примеры

Трехмерный **текст**

Stylus

```
text3d(color)
  color: color
  text-shadow: 1px 1px 0px darken(color, 5%), 2px 2px 0px
darken(color, 10%), 3px 3px 0px darken(color, 15%), 4px 4px 0px
darken(color, 20%), 4px 4px 2px #000
span
  font-size: 32pt
  text3d(#0982c1)
```



Постпроцессоры

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

PostCSS — единственный представитель постпроцессоров в контексте css.

PostCSS из коробки на самом деле не делает с CSS ничего. Он просто возвращает файл, который был дан ему на вход. Изменения начинаются, когда к PostCSS подключаются плагины.

Весь цикл работы PostCSS можно описать так:

Исходный файл дается на вход PostCSS и парсится
- Плагин 1 что-то делает
- ...
- Плагин n что-то делает
- Полученный результат преобразовывается в строку и записывается в выходной файл

Постпроцессоры

Плагины

Autoprefixer добавляет браузерные префиксы к вашим правилам. Ничем не заменимый и один из самых важных плагинов, с которого и началась история PostCSS.

```
//in.css
div {
    display: flex
}

//out.css
div {
    display: -webkit-box;
    display: -webkit-flex;
    display: -moz-box;
    display: -ms-flexbox;
    display: flex
}
```

Постпроцессоры

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Плагины

PostCSS Preset Env добавляет возможности, которые только обсуждаются в черновиках разработчиков css. В данном примере была реализована директива @custom-media, а так же функция color-mod.

```
//in.css
@custom-media --med (width <= 50rem);
@media (--med) {
 a:hover {
    color: color-mod(black alpha(54%));
  }
}

//out.css
@media (max-width: 50rem) {
   a:hover  {
     color: rgba(0, 0, 0, 0.54);
   }
 }
```

Постпроцессоры

Плагины

CSS Modules изменяет названия классов по некоторому паттерну (все настраивается). Теперь мы не знаем заранее имя класса, ибо оно определяется динамически. Как же теперь проставлять классы элементам, если мы не знаем их заранее?

```
//in.css
.name {
 color: gray;
}

//out.css
.Logo__name__SVK0g {
 color: gray;
}
```

Объединяя PostCSS, Webpack и ES6
Теперь мы не просто импортируем файл со стилями(например в файле React компонента) и подставляем заранее известные нам значения, а импортируем некий объект. Ключами этого объекта будут изначальные селекторы, а значениями — преобразованные. То есть в данном примере styles['name'] = 'Logo__name__SVK0g'.

Постпроцессоры

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Плагины

PostCSS Short добавляет кучу сокращенных записей для различных правил. Код становится короче, а следовательно в нем меньше места для ошибок. Плюс повышается читаемость.

```
//in.css
.icon {
 size: 48px;
}
.canvas {
 color: #abccfc #212231;
}
//out.css
.icon {
 width: 48px;
 height: 48px;
}
.canvas {
 color: #abccfc;
 background-color: #212231;
}
```

Постпроцессоры

Плагины

PostCSS Auto Reset позволяет нам не создавать отдельный файл со сбросом всех стилей. Плагин создает для всех селекторов один большой селектор, куда помещает правила, сбрасывающее все стили. По умолчанию создается лишь правило all со значением initial. Это полезно в комбинации с плагином postcss-initial, который в свою очередь превращает это правило в портянку правил на 4 экрана.

```
//in.css
div {
 margin: 10px;
}

a {
 color: blue;
}
```

```
//out.css
div, a {
 all: initial;
}

div {
 margin: 10px;
}

a {
 color: blue;
}
```

Специфичность

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Специфичность - это способ, с помощью которого браузеры определяют, какие значения свойств CSS наиболее соответствуют элементу и, следовательно, будут применены. Специфичность основана на правилах соответствия, состоящих из селекторов CSS различных типов.

Специфичность представляет собой вес, придаваемый конкретному правилу CSS.

Вес правила определяется количеством каждого из типов селекторов в данном правиле.

Если у нескольких правил специфичность одинакова, то к элементу применяется последнее по порядку правило CSS.

Специфичность имеет значение только в том случае, если один элемент соответствует нескольким правилам.

Согласно спецификации CSS, правило для непосредственно соответствующего элемента всегда будет иметь больший приоритет, чем правила, унаследованные от предка.

Специфичность

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Типы селекторов

1. селекторы типов элементов (например, h1) и псевдоэлементов (например, ::before).

2. селекторы классов (например, .example), селекторы атрибутов (например, [type="radio"]) и псевдоклассов (например, :hover).

3.селекторы идентификаторов (например, #example).

Универсальный селектор (*), комбинаторы (+, >, ~, ' ') и отрицающий псевдокласс (:not()) не влияют на специфичность. (Однако селекторы, объявленные внутри :not(), влияют)

Стили, объявленные в элементе (например, style="font-weight:bold"), всегда переопределяют любые правила из внешних файлов стилей и, таким образом, их специфичность можно считать наивысшей.

Специфичность

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

!important

Когда при объявлении стиля используется модификатор `!important`, это объявление получает наивысший приоритет среди всех прочих объявлений.

Хотя технически модификатор `!important` не имеет со специфичностью ничего общего, он непосредственно на неё влияет.

Поскольку `!important` усложняет отладку, нарушая естественное каскадирование ваших стилей, он не приветствуется и следует избегать его использования.

Если к элементу применимы два взаимоисключающих стиля с модификатором `!important`, то применён будет стиль с большей специфичностью.

Специфичность

!important

Практические советы:

- Всегда пытайтесь использовать специфичность, а `!important` используйте только в крайних случаях
- Используйте `!important` только в страничных стилях, которые переопределяют стили сайта или внешние стили (стили библиотек, таких как `Bootstrap` или `normalize.css`)
- Никогда не используйте `!important`, если вы пишете плагин или мэшап.
- Никогда не используйте `!important` в общем `CSS` сайта.

Специфичность

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

!important

Вместо `!important` можно:

- Лучше использовать каскадные свойства CSS
- Использовать более специфичные правила. Чтобы сделать правило более специфичным и повысить его приоритет, укажите один элемент или несколько перед нужным вам элементом:

```
<div id="test">
  <span>Text</span>
</div>
```

```
div#test span {
  color: green;
}
div span {
  color: blue;
}
span {
  color: red;
}
```

Вне зависимости от порядка следования правил, текст всегда будет зелёным, поскольку у этого правила наибольшая специфичность (при этом, правило для голубого цвета имеет преимущество перед правилом для красного, несмотря на порядок следования).

Новая плюшка в CSS

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

@layer

Правило CSS используется для объявления каскадного слоя, а также может использоваться для определения порядка приоритета в случае нескольких каскадных слоев.

https://developer.mozilla.org/en-US/docs/Web/CSS/@layer

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

Спасибо за внимание