

Переменные и примеси

Использование переменных в CSS

CSS переменные (пользовательские CSS-свойства) - сущности, хранящие конкретные значения, которые можно повторно использовать в документе

Устанавливаются с использованием custom property нотации (например, `--main-color: black;`) и доступны через функцию `var()` (например, `color: var(--main-color);`);

CSS переменные позволяют сохранять значение в одном месте, а затем многократно использовать его в любом другом месте.

CSS переменные подчиняются каскаду и наследуют значения от своих родителей.

Использование переменных в CSS

Пример использования переменных:

Объявление переменной:

```
:root {  
    --main-bg-color: blue;  
}
```

Использование переменной:

```
.element {  
    background-color: var(--main-bg-color);  
}
```

Переменные не поддерживаются в браузере Internet Explorer

Использование переменных в CSS

Область видимости переменных:

Глобальные переменные - к ним

есть доступ в любой части

программы

```
:root {  
  --primary-color: blue;  
}  
  
.section {  
  color: var(--primary-color);  
}
```

Локальные переменные - доступны

только внутри блока или функции, в

которой они определены

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

Использование переменных в CSS

Резервное значение (на случай, когда переменная не определена):

```
.section {  
  color: var(--primary-color, blue);  
}
```



резервное значение

Можно вкладывать одну переменную в другую:

```
.section {  
  color: var(--primary-color, var(--my-color, #fff));  
}
```

Интерполяция переменных

Интерполяция переменных — это получение значения одной переменной в зависимости от другой или других переменных.

Интерполяция позволяет использовать произвольную строку, которая хранится в переменной в качестве строковых значений, или части строковых значений CSS правил, свойств, значений этих свойств, использовать это значение в наименовании селекторов и даже внутри селекторов.

Интерполяция переменных

Использование URL как значение.

Сохранение значения URL-адреса ссылки в переменной
CSS:

```
:root {  
  --main-background: url('example.com/image.png')  
}  
  
.section {  
  background: var(--main-background);  
}
```

Интерполяция переменных CSS с помощью url():

```
:root {  
  --main-background: 'example.com/image.png'  
}  
  
.section {  
  background: url(var(--main-background));  
}
```



Не сработает, т.к. `var(--main-background)` рассматривается сама как URL, который недопустим. К тому моменту, когда браузер вычислит это значение, оно уже не будет действительным и не будет работать, как ожидалось

Что такое примеси

Примесь (mixin) — набор свойств и селекторов, расширяющий поведение другой сущности (селектора). Это фрагмент кода, который можно переиспользовать.

Рассмотрим примеси на примере синтаксиса Sass:

- Создание примеси:

Используется команда `@mixin` после которой указывается название примеси.

Внутри в фигурных скобках пишутся стилевые правила. В том месте, куда нужно вставить примесь, добавляется команда `@include` с её именем.

```
@mixin my_opacity {  
  -webkit-transition: opacity 1s ease-in;  
  -moz-transition: opacity 1s ease-in;  
  transition: opacity 1s ease-in;  
}  
.block {  
  height: 100px;  
  background: #fc0;  
}  
.block:hover {  
  opacity: 0.5;  
  @include my_opacity;  
}
```

Компиляция в CSS-код



```
.block {  
  height: 100px;  
  background: #fc0;  
}  
.block:hover {  
  opacity: 0.5;  
  -webkit-transition: opacity 1s ease-in;  
  -moz-transition: opacity 1s ease-in;  
  transition: opacity 1s ease-in;  
}
```


Что такое примеси

- Примеси с параметрами (для переиспользования примеси в различных контекстах):

Параметры указываются в скобках после имени примеси и представляют собой обычные локальные переменные. Параметров может быть несколько, и они разделяются либо запятой, либо точкой с запятой.

```
.bordered(@_color) {  
  border-top: dotted 1px @_color;  
  border-bottom: solid 2px @_color;  
}  
  
.article {  
  .bordered(#ccc);  
  color: #443d3d;  
}
```

Компиляция в CSS-код



```
.article {  
  border-top: dotted 1px #cccccc;  
  border-bottom: solid 2px #cccccc;  
  color: #443d3d;  
}
```

Что такое примеси

- Значения параметров по умолчанию:

Если примесь вызвана, а значения для параметров не были переданы или переданы частично, то ошибки компилятор не выдаст, а возьмёт значение, указанное по умолчанию.

```
.bordered(@_color: #ccc) {  
  border-top: dotted 1px @_color;  
  border-bottom: solid 2px @_color;  
}  
  
.article {  
  .bordered();  
  color: #443d3d;  
}
```

Компиляция в CSS-код



```
.article {  
  border-top: dotted 1px #cccccc;  
  border-bottom: solid 2px #cccccc;  
  color: #443d3d;  
}
```

Дополнительные возможности примесей

- Примеси как функции:

Идеология функции заключается в том, что она должна возвращать значение, а её вызов можно использовать как выражение. С помощью примесей можно сделать настоящие математические функции для вычисления чего-либо.

Примеры:

Перевод единиц измерения

```
.pxToEm(@value, @base: 16px) {  
  @calcEm: (@value / @base) + 0em;  
}  
  
.class {  
  .pxToEm(20px);  
  content: @calcEm;  
}
```

Вычисление площади треугольника по основанию и высоте

```
.areaTriangle(@a, @h) {  
  @calcAreaTriangle: (0.5 * @a * @h);  
}  
  
.class {  
  .areaTriangle(20, 50);  
  content: @calcAreaTriangle;  
}
```

Дополнительные возможности примесей

- Пространство имен:

Пространства имён позволяют сортировать примеси по условным категориям и избегать конфликта их имён. Такой подход позволяет создавать примеси, которые не будут конфликтовать с именами других примесей в проекте или используемых библиотеках

Примеры:

Объявление примеси внутри селектора

```
.selector {  
  .mixin(@color: #333) {  
    color: @color;  
  }  
}
```

Использовании примеси, объявленной внутри селектора
(указывается полный путь)

```
// не работает / ошибка (Error: .mixin is undefined)  
.class {  
  .mixin();  
}  
  
// работает (разные варианты)  
.class {  
  .selector > .mixin();  
  .selector .mixin();  
  .selector.mixin();  
}
```