



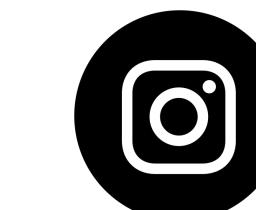
IS
SOFT

CSS ещё не торт

Пишем код, который поддерживается уже сегодня

Вероника Новикова
[@niktariy](https://twitter.com/niktariy)

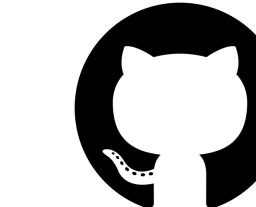




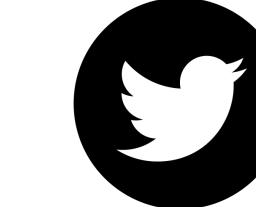
[@niktariy_dev](#)



[@niktariy](#)



[@niktariy](#)



[@niktariy](#)

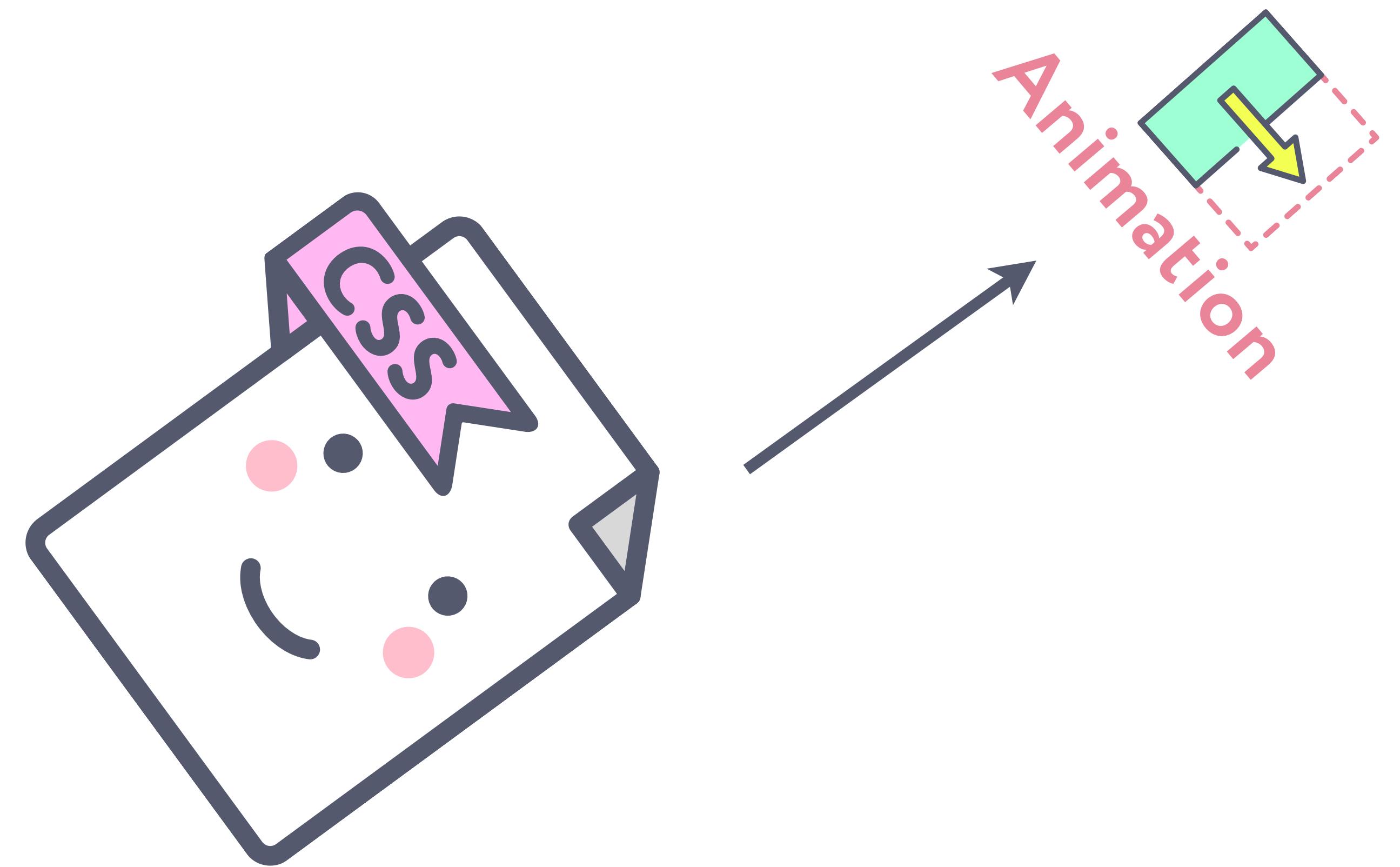


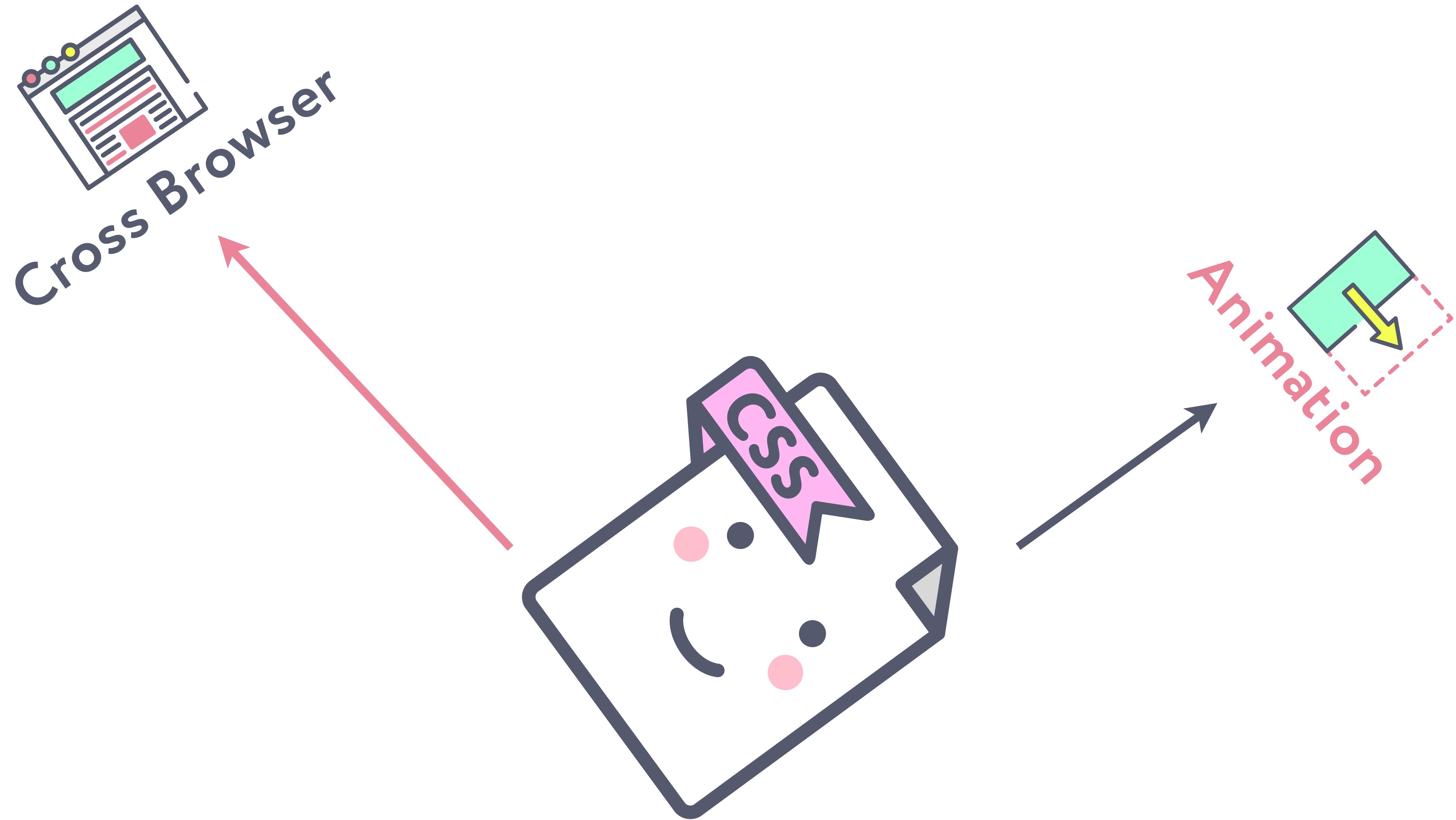
[@niktariy](#)

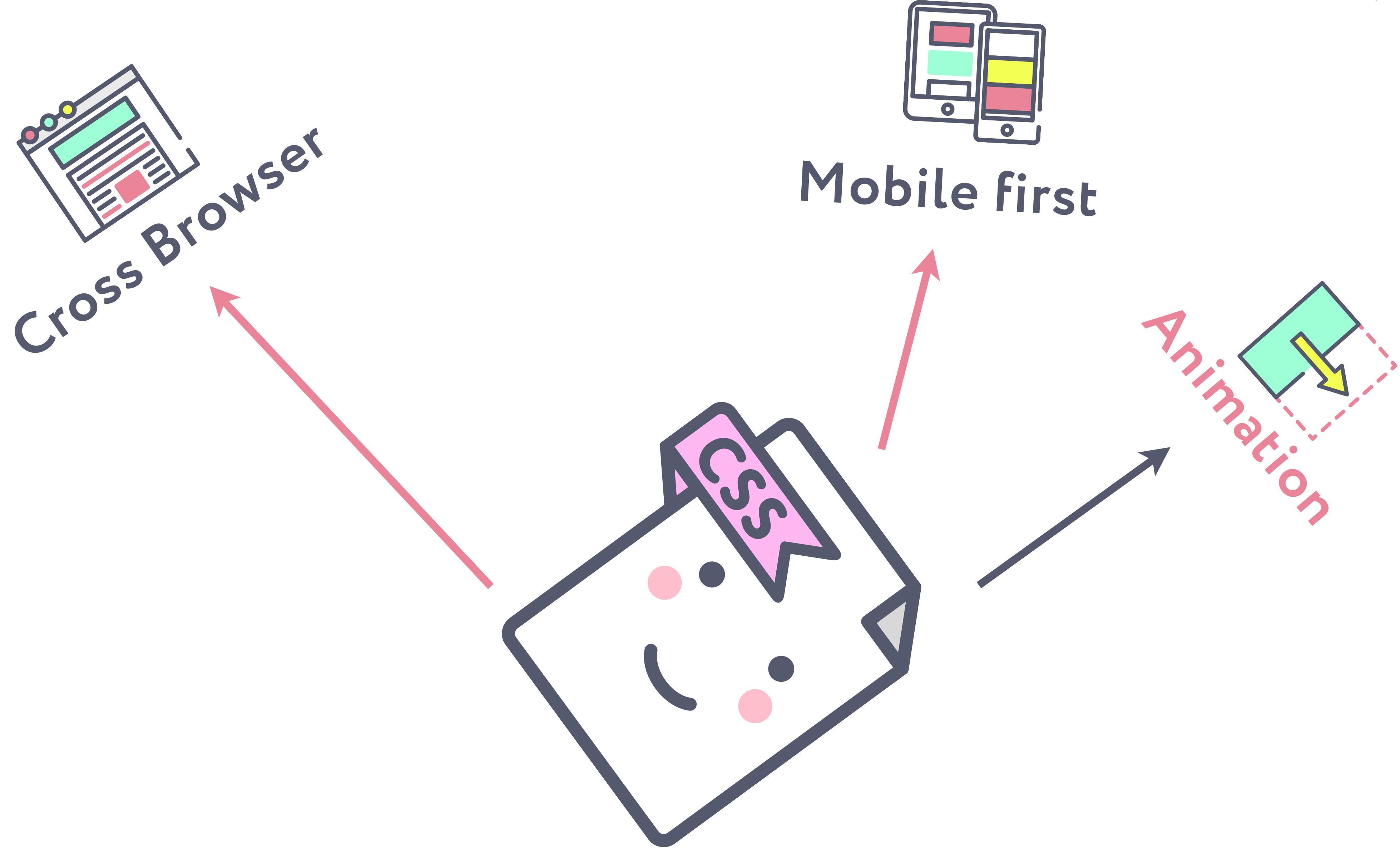
Вероника Новикова

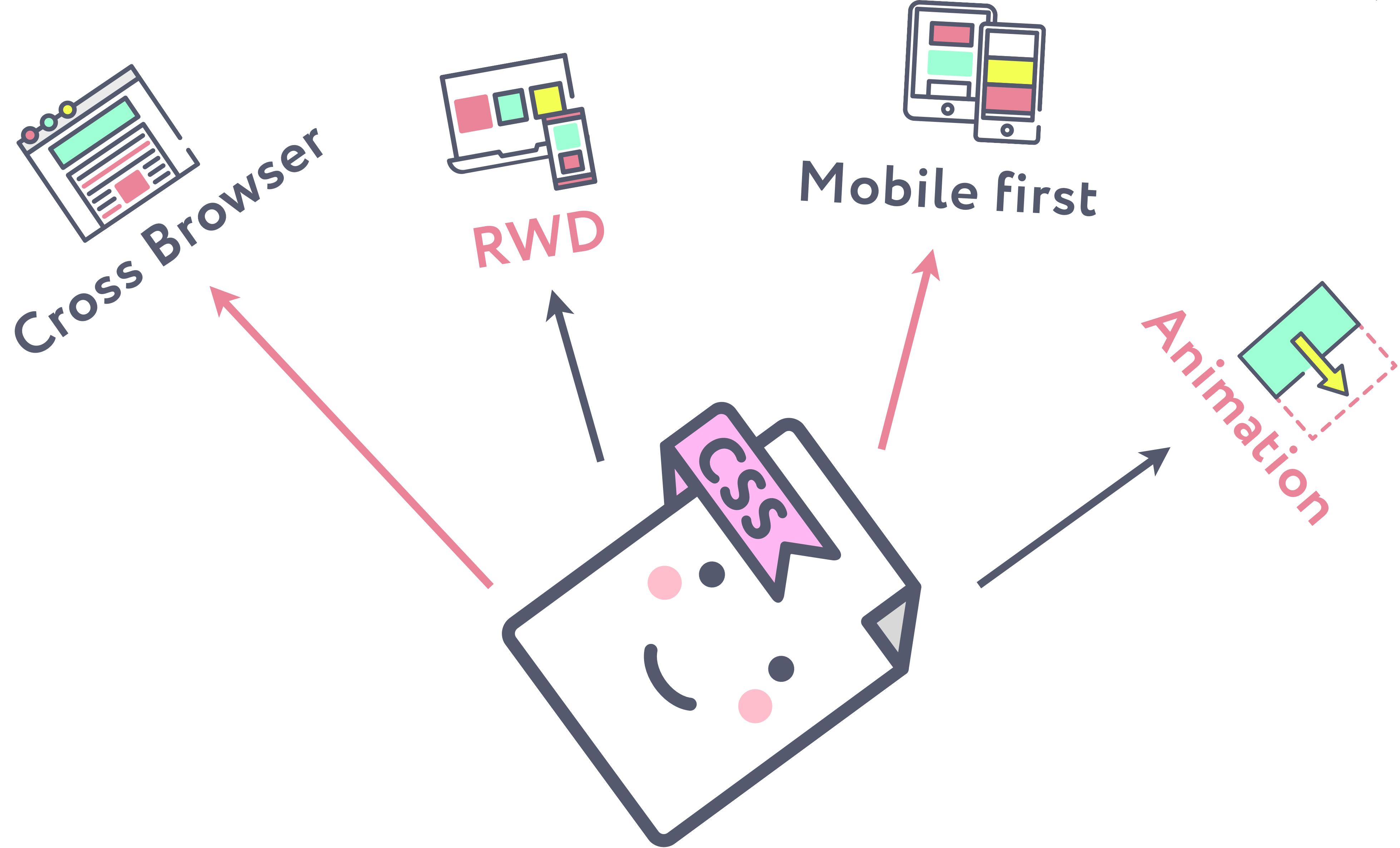
Разработчик в ISsoft и просто хороший человек



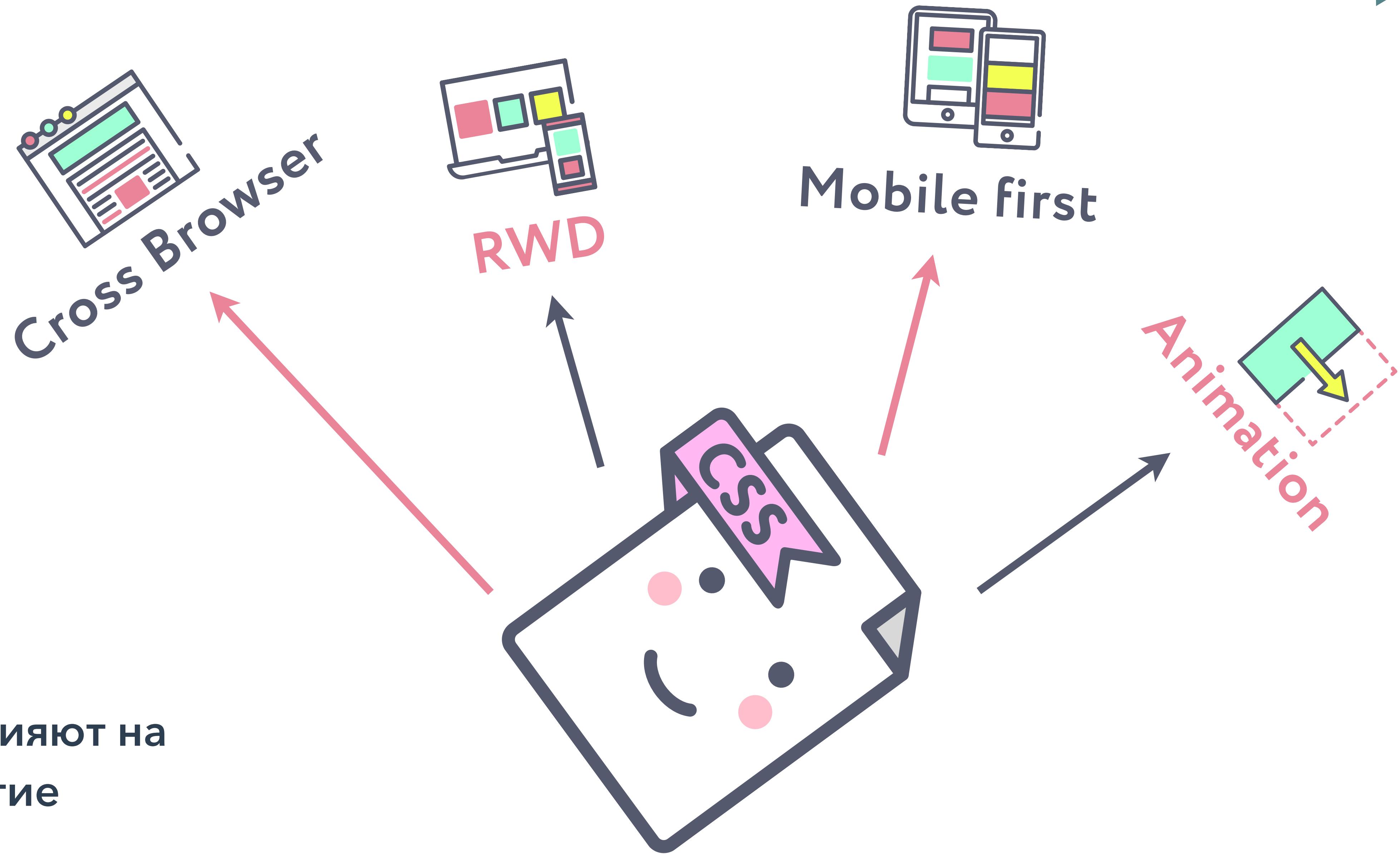








Стили влияют на восприятие



Preprocessors

Существующие CSS-препроцессоры:

Существующие CSS-препроцессоры:

1. Sass (2006 г.) – самый популярный препроцессор с огромным сообществом (sassmeister.com и sass-scss.ru)

Существующие CSS-препроцессоры:

1. Sass (2006 г.) – самый популярный препроцессор с огромным сообществом (sassmeister.com и sass-scss.ru)
2. Less (2009 г.) – еще один лидирующий CSS препроцессор (lesstester.com и less-lang.info)

Существующие CSS-препроцессоры:

1. Sass (2006 г.) – самый популярный препроцессор с огромным сообществом (sassmeister.com и sass-scss.ru)
2. Less (2009 г.) – еще один лидирующий CSS препроцессор (lesstester.com и less-lang.info)
3. Stylus (2010 г.) – не разрабатывается уже больше года

Существующие CSS-препроцессоры:

1. Sass (2006 г.) – самый популярный препроцессор с огромным сообществом (sassmeister.com и sass-scss.ru)
2. Less (2009 г.) – еще один лидирующий CSS препроцессор (lesstester.com и less-lang.info)
3. Stylus (2010 г.) – не разрабатывается уже больше года

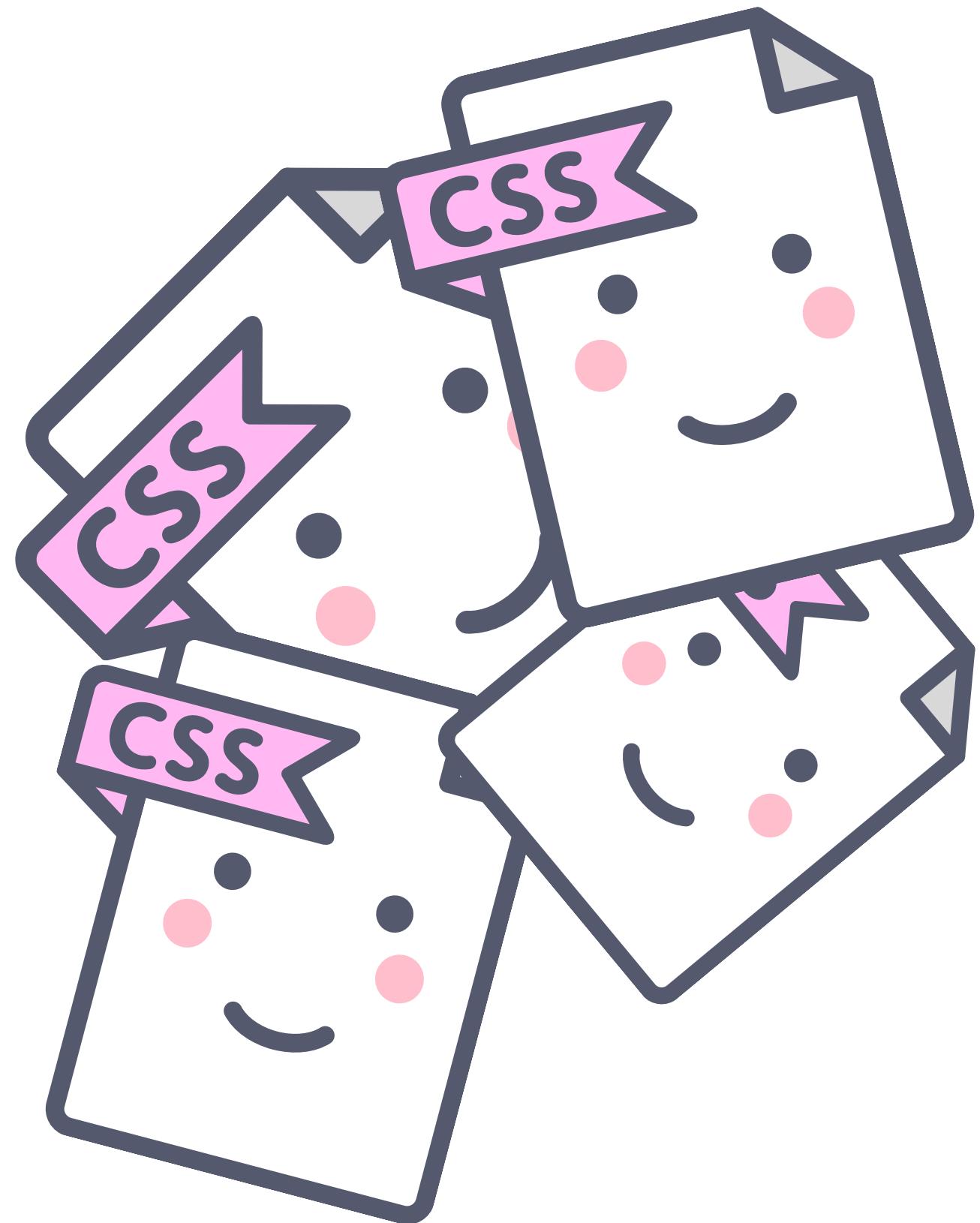
PostCSS (2013 г.) – с ним можно повторить почти всю функциональность настоящих препроцессоров.

Sass syntax

```
// SCSS // Sass  
01. button {  
02.   font: {  
03.     size: 16px;  
04.     weight: bold;  
05.   }  
06.   color: #333;  
07.   &:hover {  
08.     color: #222;  
09.   }  
10. }
```

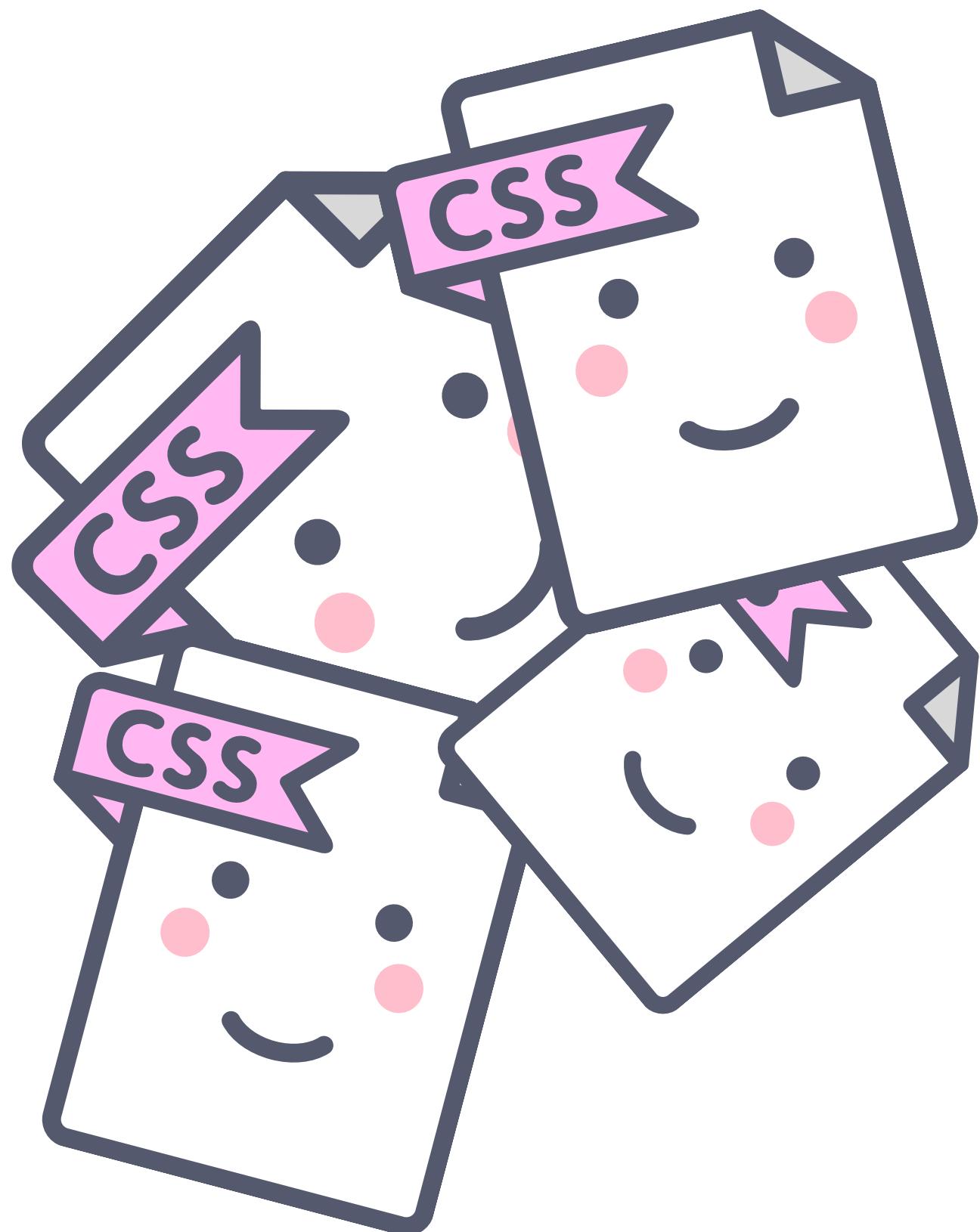
```
01. button  
02.   font:  
03.     size: 16px  
04.     weight: bold  
05.   color: #333  
06.   &:hover  
07.     color: #222
```

Беда с организацией файлов



Свалка стилей

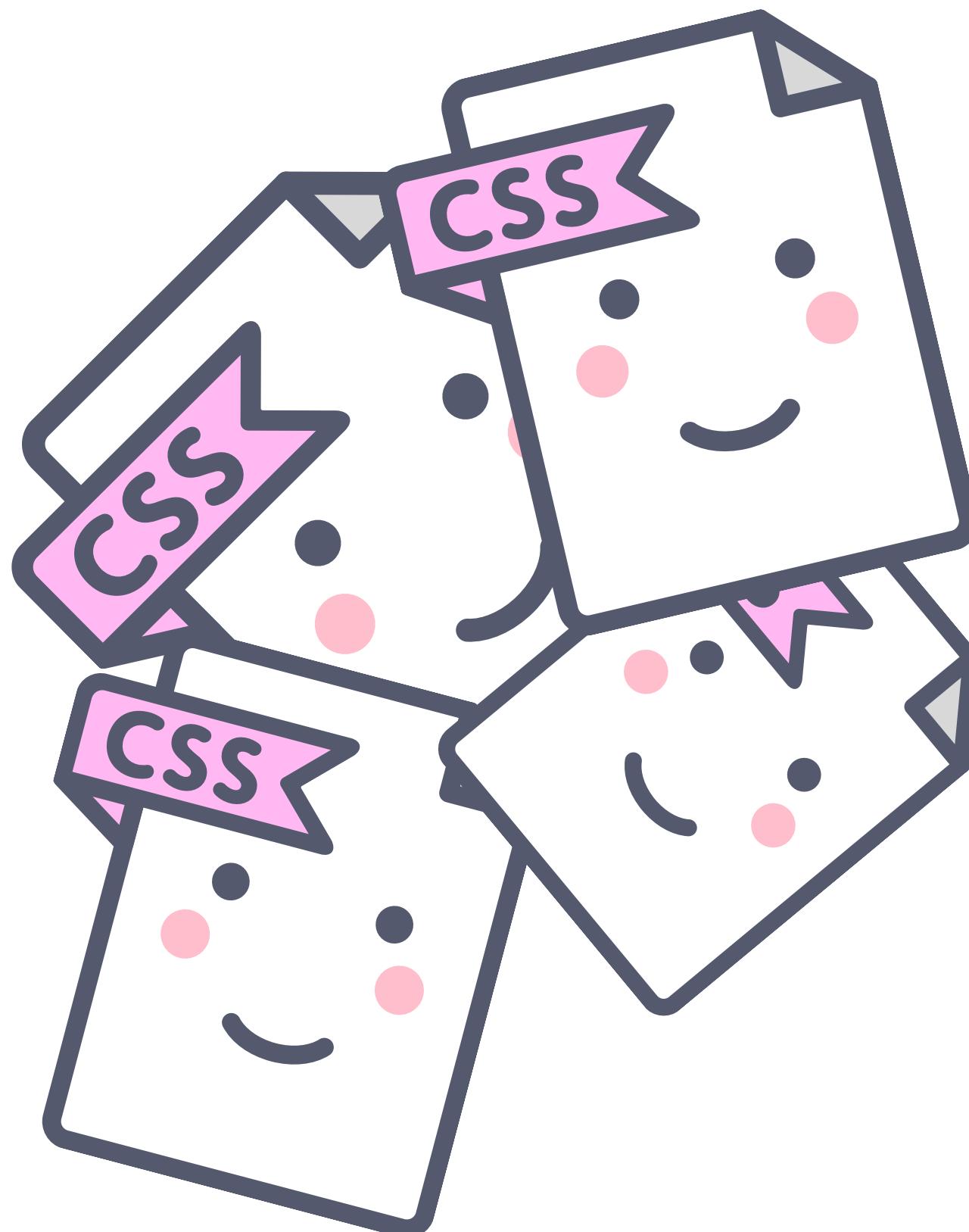
Беда с организацией файлов



Свалка стилей

- Сложная иерархия файлов

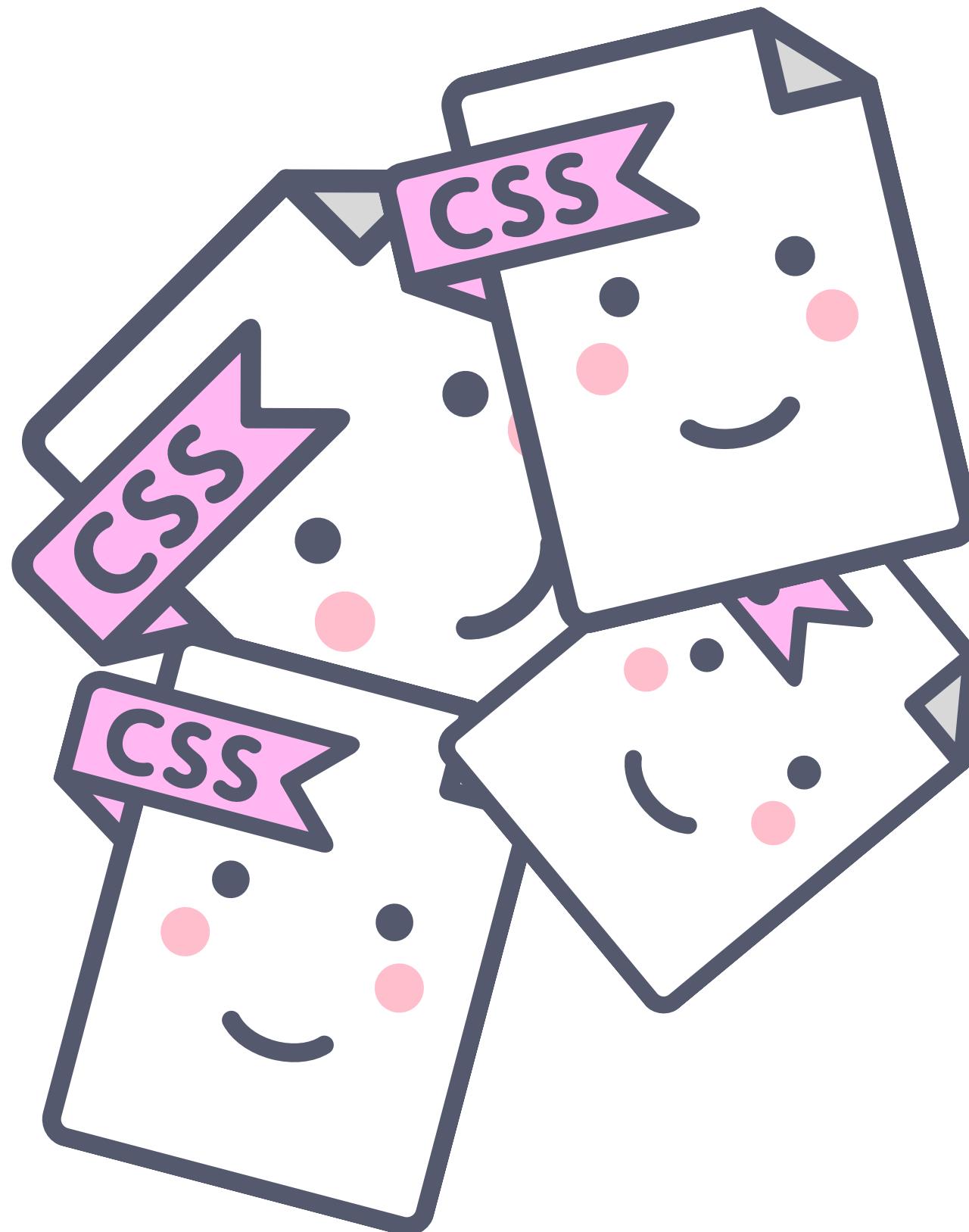
Беда с организацией файлов



Свалка стилей

- Сложная иерархия файлов
- Трудно найти нужный селектор или набор свойств

Беда с организацией файлов



- Сложная иерархия файлов
- Трудно найти нужный селектор или набор свойств
- МНОГАСТРОЧЕК

Свалка стилей

@import

@import

+ BEM
+ SMACSS

CSS @import

@import позволяет разделить CSS-файл и импортировать одни таблицы стилей в другие
([MDN @import](#))

Проверь, есть ли на твоём сайте CSS @import giftofspeed.com/css-delivery

CSS @import

@import позволяет разделить CSS-файл и импортировать одни таблицы стилей в другие
([MDN @import](#))

- поддерживает медиа-запросы:

```
@import url('landscape.css') screen and (orientation: landscape);
```

Проверь, есть ли на твоём сайте CSS @import giftofspeed.com/css-delivery

CSS @import

@import позволяет разделить CSS-файл и импортировать одни таблицы стилей в другие
([MDN @import](#))

- поддерживает медиа-запросы:

```
@import url('landscape.css') screen and (orientation: landscape);
```

- **@import** всегда должен быть в верху документа
(но после любой @charset декларации)

CSS @import

`@import` позволяет разделить CSS-файл и импортировать одни таблицы стилей в другие
([MDN @import](#))

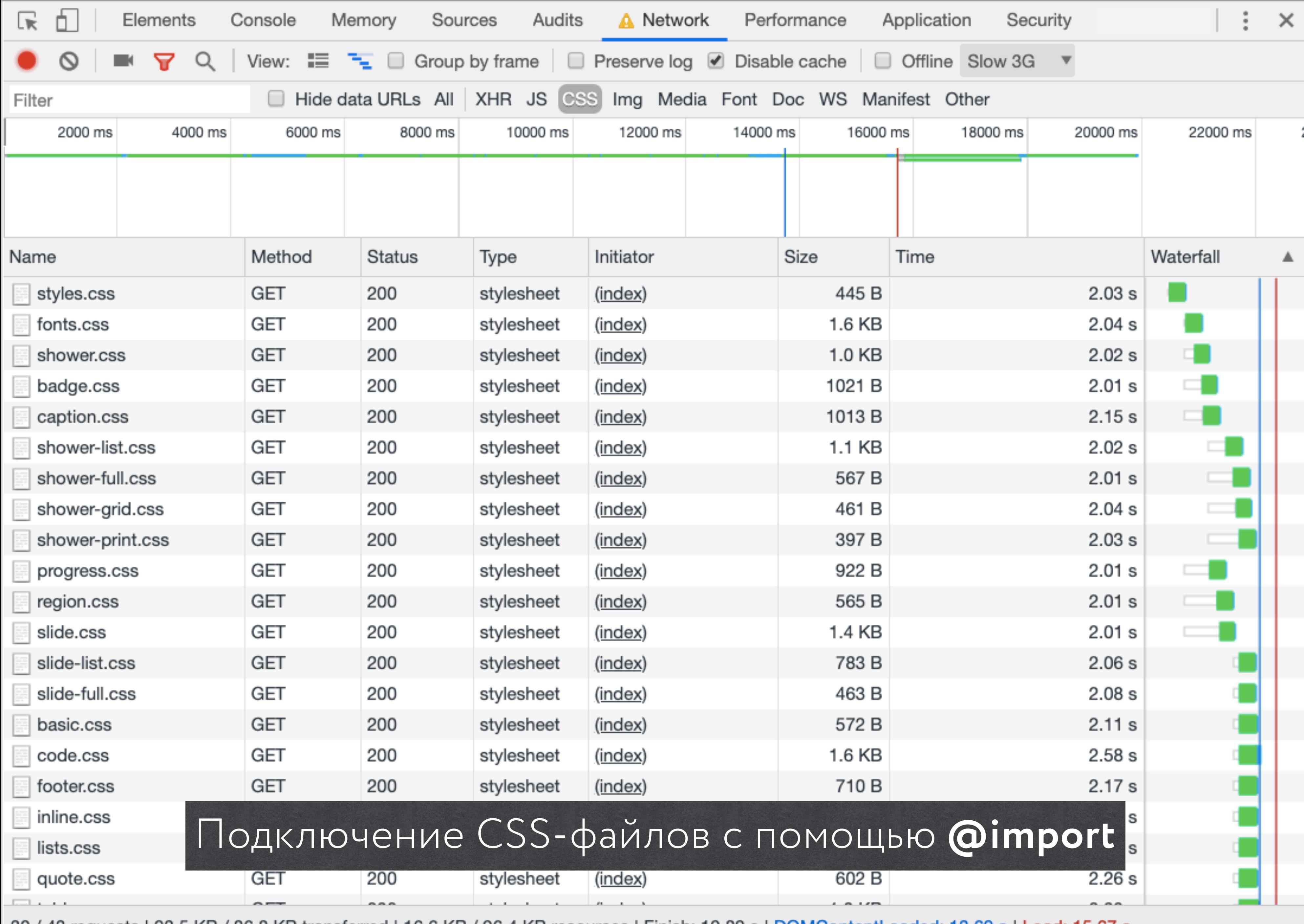
- поддерживает медиа-запросы:

```
@import url('landscape.css') screen and (orientation: landscape);
```

- `@import` всегда должен быть в верху документа

(но после любой `@charset` декларации)

- создаёт **дополнительные HTTP-запросы**



Elements Console Memory Sources Audits Network Performance Application Security

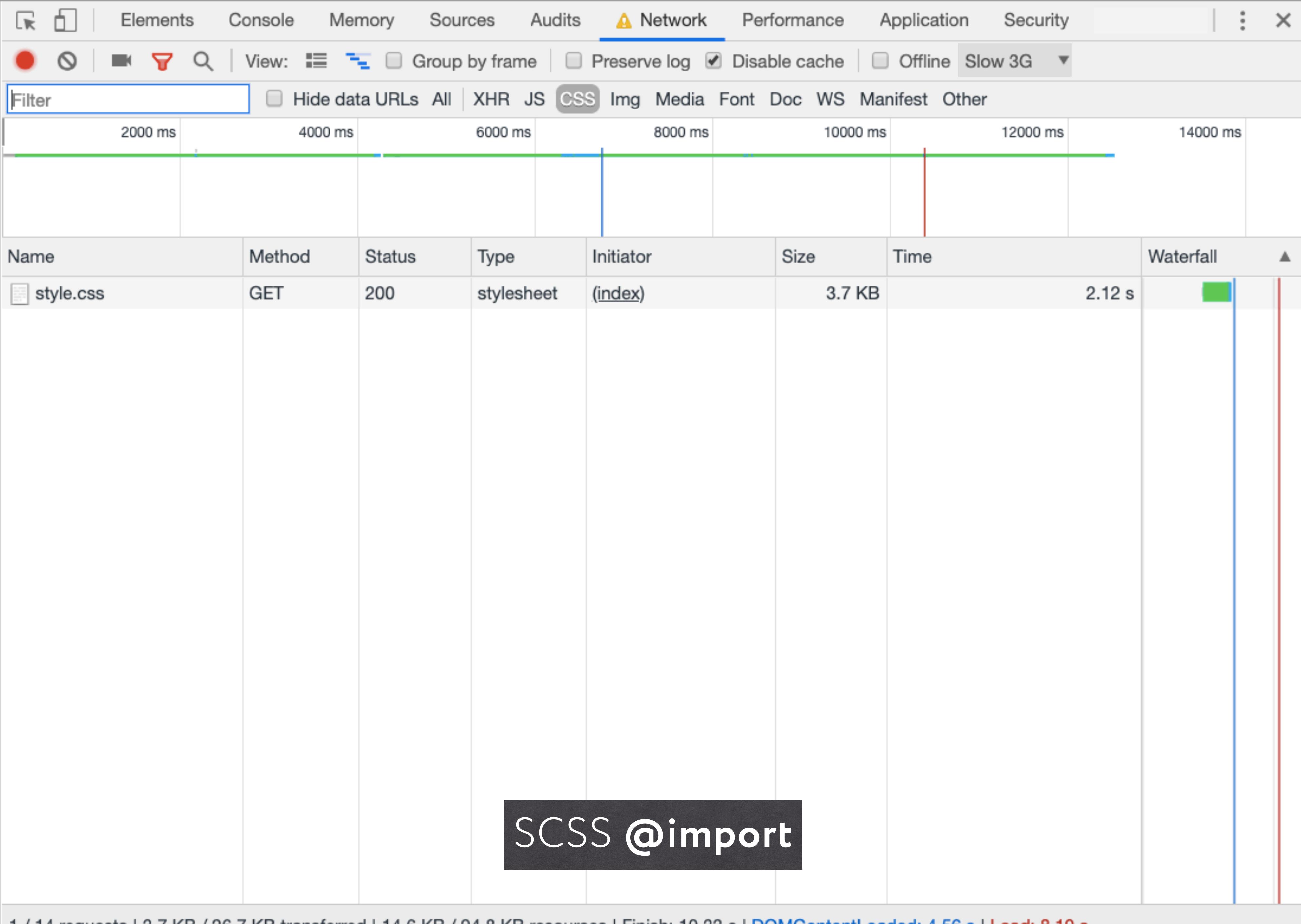
View: Group by frame Preserve log Disable cache Offline Slow 3G

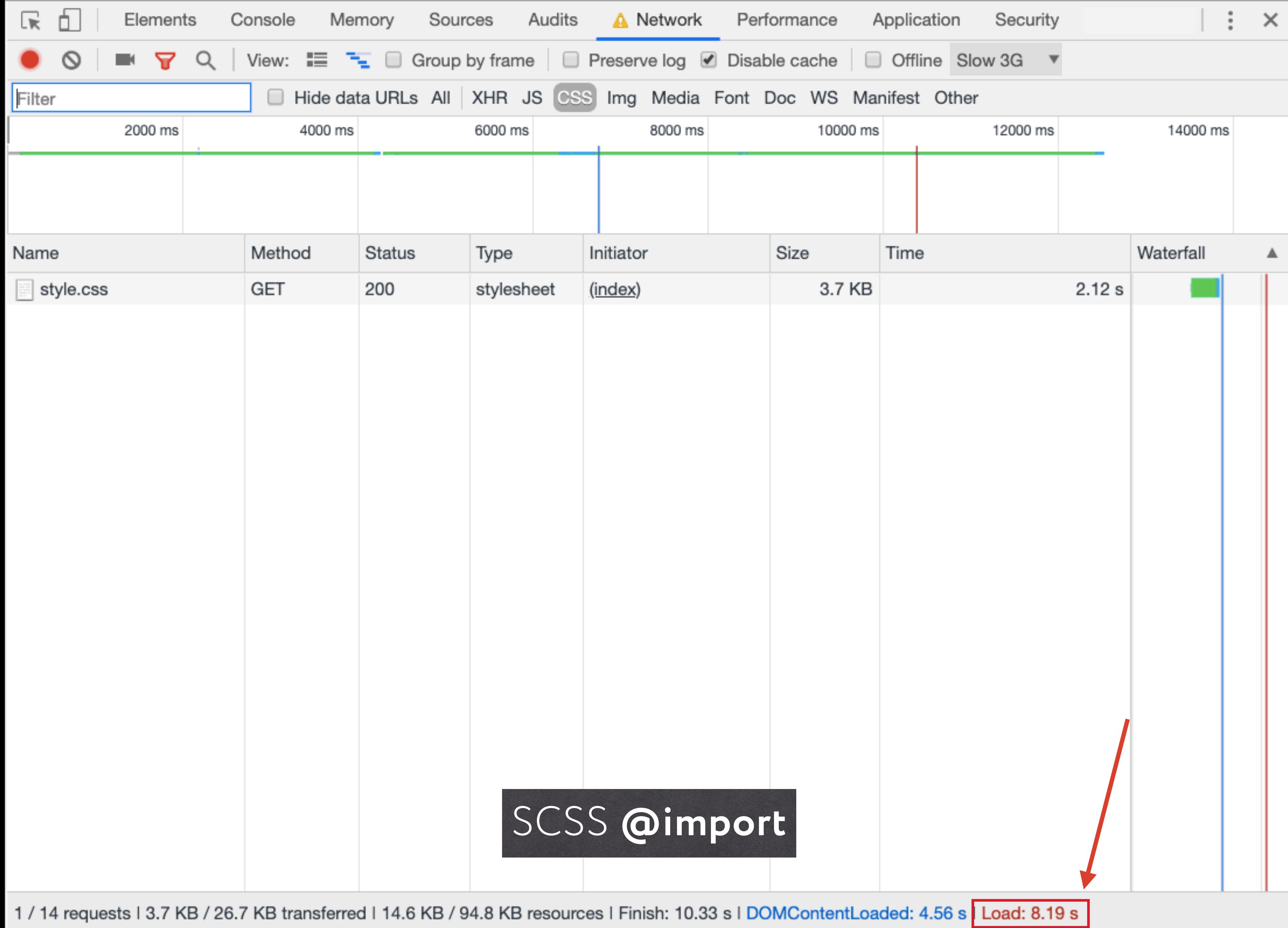
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Method	Status	Type	Initiator	Size	Time	Waterfall
styles.css	GET	200	stylesheet	(index)	445 B	2.03 s	
fonts.css	GET	200	stylesheet	(index)	1.6 KB	2.04 s	
shower.css	GET	200	stylesheet	(index)	1.0 KB	2.02 s	
badge.css	GET	200	stylesheet	(index)	1021 B	2.01 s	
caption.css	GET	200	stylesheet	(index)	1013 B	2.15 s	
shower-list.css	GET	200	stylesheet	(index)	1.1 KB	2.02 s	
shower-full.css	GET	200	stylesheet	(index)	567 B	2.01 s	
shower-grid.css	GET	200	stylesheet	(index)	461 B	2.04 s	
shower-print.css	GET	200	stylesheet	(index)	397 B	2.03 s	
progress.css	GET	200	stylesheet	(index)	922 B	2.01 s	
region.css	GET	200	stylesheet	(index)	565 B	2.01 s	
slide.css	GET	200	stylesheet	(index)	1.4 KB	2.01 s	
slide-list.css	GET	200	stylesheet	(index)	783 B	2.06 s	
slide-full.css	GET	200	stylesheet	(index)	463 B	2.08 s	
basic.css	GET	200	stylesheet	(index)	572 B	2.11 s	
code.css	GET	200	stylesheet	(index)	1.6 KB	2.58 s	
footer.css	GET	200	stylesheet	(index)	710 B	2.17 s	
inline.css							
lists.css							
quote.css	GET	200	stylesheet	(index)	602 B	2.26 s	
...							

Подключение CSS-файлов с помощью `@import`

30 / 43 requests | 23.5 KB / 86.8 KB transferred | 16.6 KB / 96.4 KB resources | Finish: 19.89 s | DOMContentLoaded: 13.69 s | Load: 15.67 s





Sass @import

Sass импортирует файл напрямую в место вызова, т.е. на выходе получится один CSS-файл

Sass @import

Sass импортирует файл напрямую в место вызова, т.е. на выходе получится один CSS-файл

- можно не указывать расширение `@import 'reset';`

Sass @import

Sass импортирует файл напрямую в место вызова, т.е. на выходе получится один CSS-файл

- можно не указывать расширение `@import 'reset';`
- можно вызывать в любом месте

Sass @import

Sass импортирует файл напрямую в место вызова, т.е. на выходе получится один CSS-файл

- можно не указывать расширение `@import 'reset';`

- можно вызывать в любом месте

```
body { ...properties... }

...
@import 'reset';

...
.button { @import 'small'; }
```

Sass @import

Sass импортирует файл напрямую в место вызова, т.е. на выходе получится один CSS-файл

- можно не указывать расширение `@import 'reset';`

- можно вызывать в любом месте `body { ...properties... }`

...

```
...  
@import 'reset';
```

...

```
...  
.button { @import 'small'; }
```

- можно и в медиа-выражениях (но не нужно)

Sass @import

```
// _reset.scss

html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}
```

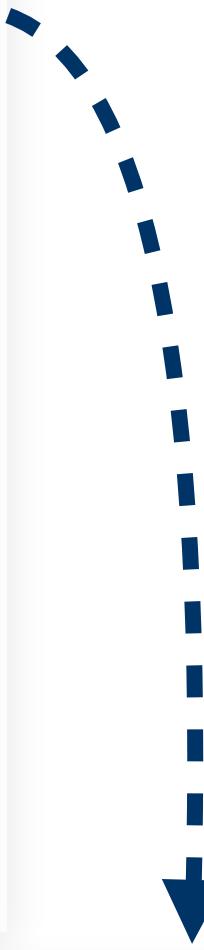
```
// main.scss

@import 'base/reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

Sass @import

```
// _reset.scss  
  
html,  
body,  
ul,  
ol {  
  margin: 0;  
  padding: 0;  
}
```



```
// main.scss  
  
@import 'base/reset';  
  
body {  
  font: 100% Helvetica, sans-serif;  
  background-color: #efefef;  
}
```

Sass @import

```
// _reset.scss  
  
html,  
body,  
ul,  
ol {  
    margin: 0;  
    padding: 0;  
}
```

```
// main.scss  
  
@import 'base/reset';  
  
body {  
    font: 100% Helvetica, sans-serif;  
    background-color: #efefef;  
}
```

// CSS output

```
html,  
body,  
ul,  
ol {  
    margin: 0;  
    padding: 0;  
}  
  
body {  
    font: 100% Helvetica, sans-serif;  
    background-color: #efefef;  
}
```

Sass @import

```
sass/
  - base/
    |- _reset.scss      # Reset/normalize
    |- _typography.scss # Typography rules
    ...
  - components/
    |- _buttons.scss    # Buttons
    |- _dropdown.scss   # Dropdown
    |- _navigation.scss # Navigation
    ...
  - helpers/
    |- _variables.scss  # Sass Variables
    |- _functions.scss   # Sass Functions
    |- _mixins.scss      # Sass Mixins
    |- _helpers.scss     # Class & placeholders helpers
    ...
  - layout/
    |- _grid.scss        # Grid system
    |- _header.scss       # Header
    |- _footer.scss       # Footer
    ...
  - pages/
    |- _home.scss         # Home specific styles
    |- _contacts.scss     # Contacts specific styles
    ...
  - themes/
    |- _theme.scss        # Default theme
    |- _admin.scss        # Admin theme
    ...
  - main.scss            # primary Sass file
```

CSS Custom Properties

CSS Custom properties

Плюсы

- изменяются динамически

CSS Custom properties

Плюсы

- изменяются динамически
- не требуют компиляции

CSS Custom properties

Плюсы

- изменяются динамически
- не требуют компиляции
- могут эмулировать CSS-свойства (эмуляция - [пример](#))

CSS Custom properties

Плюсы

- изменяются динамически
- не требуют компиляции
- могут эмулировать CSS-свойства (эмуляция - [пример](#))
- знают о DOM-структуре (наследование - [пример](#))

CSS Custom properties

Плюсы

- изменяются динамически
- не требуют компиляции
- могут эмулировать CSS-свойства (эмуляция - [пример](#))
- знают о DOM-структуре (наследование - [пример](#))
- доступны внутри HTML (использование в SVG - [пример](#))

CSS Custom properties

Плюсы

- изменяются динамически
- не требуют компиляции
- могут эмулировать CSS-свойства (эмуляция - [пример](#))
- знают о DOM-структуре (наследование - [пример](#))
- доступны внутри HTML (использование в SVG - [пример](#))
- доступны из JavaScript ([пример](#))



A photograph showing a man and a woman working on a large-scale wooden model of the Leaning Tower of Pisa. They are in a workshop or studio setting, surrounded by tools and materials. The man, wearing a yellow shirt, is focused on the tower's base, while the woman, wearing a dark top, stands behind him. The text "Frontend Developer" is overlaid on the upper left of the image.

Frontend Developer



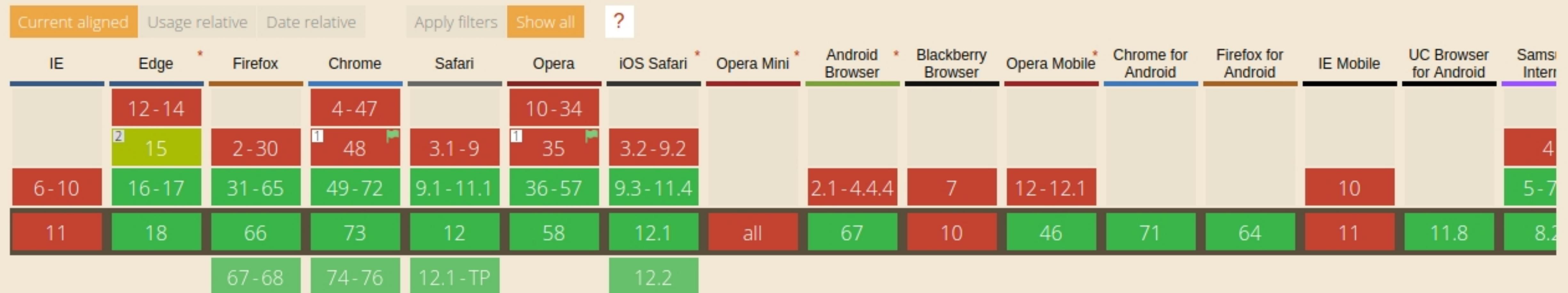
A photograph showing a man and a woman working on a large-scale wooden model of the Leaning Tower of Pisa. They are in a workshop or studio setting, surrounded by tools and materials. The man, wearing a yellow shirt, is focused on the tower's base, while the woman, wearing a dark top, stands behind him. The text "Frontend Developer" is overlaid on the upper left of the image.

Frontend Developer

CSS Variables (Custom Properties) - CR

Permits the declaration and usage of cascading variables in stylesheets.

Usage	% of all users	 	
Global	90.54%	+ 0.06%	= 90.6%
Belarus	89.95%	+ 0.05%	= 90%



Notes Known issues (3) Resources (8) Feedback

- In Edge 15 is not possible to use css variables in pseudo elements [see bug](#)
- In Edge 15 animations with css variables may cause the webpage to crash [see bug](#)
- In Edge 15, nested calculations with css variables are not computed and are ignored [see bug](#)

CSS Custom properties

Минусы

- не работают в Internet Explorer

CSS Custom properties

Минусы

- не работают в Internet Explorer
- не типизированы

CSS Custom properties

Минусы

- не работают в Internet Explorer
- не типизированы
- не анимируются ([пример](#))

Что почитать?

- Спецификация [CSS Custom Properties for Cascading Variables Module Level 1.](#)
- Доклад Павла Ловцевича ["Пользовательские свойства как основа архитектуры CSS"](#) с FrontTalks, где он подробно рассказывает о том, почему пользовательские свойства крутые.
- Отличные подробные статьи на английском языке:
 - [It's Time To Start Using CSS Custom Properties](#), Serg H ospodarets
 - [What is the difference between CSS variables and preprocessor variables?](#), Chris Coyier

Переменные в препроцессорах

В препроцессорах можно объявлять и использовать переменные практически где угодно, включая внешние блоки объявлений, правила мультимедиа или даже как часть селектора.

Чтобы создать переменную в Sass используется символ «\$»

Переменные в препроцессорах

В препроцессорах можно объявлять и использовать переменные практически где угодно, включая внешние блоки объявлений, правила мультимедиа или даже как часть селектора.

Чтобы создать переменную в Sass используется символ «\$»

```
// SCSS

$breakpoint: 800px;
$cute-color: red;
$cute-things: ".cute, .cats";

@media screen and (min-width: $breakpoint) {
  #{$cute-things} {
    color: $cute-color;
  }
}
```

HTML

```
1<h3 class="cute">cute</h3>
2<h4 class="cats">cats</h4>
```

CSS (SCSS)

```
6
7/* SCSS variables */
8
9$breakpoint: 800px;
10$cute-color: red;
11$cute-things: ".cute, .cats";
12
13@media screen and (min-width: $breakpoint) {
14 #${$cute-things} {
15    color: $cute-color;
16 }
17 }
18
```

cute
cats

Переменные в препроцессорах

```
/* CSS */

:root {
    --breakpoint: 800px;
    --cute-color: red;
}

@media screen and (min-width: var(--breakpoint)) {
    .cute, .cats {
        color: var(--cute-color);
    }
}
```

HTML

```
1<h3 class="cute">cute</h3>
2<h4 class="cats">cats</h4>
```

CSS

```
7/* CSS Custom properties */
8
9:root {
10  --breakpoint: 800px;
11  --cute-color: #d24799;
12 }
13
14@media screen and (min-width: var(--breakpoint)) {
15  .smashing-text, .cats {
16    color: var(--cute-color);
17  }
18}
19
```

cute
cats

HTML

```
1<h3 class="cute">cute</h3>
2<h4 class="cats">cats</h4>
```

CSS

```
7/* CSS Custom properties */
8
9:root {
10  --breakpoint: 800px;
11  --cute-color: #d24799;
12 }
13
14@media screen and (min-width: var(--breakpoint)) {
15  .smashing-text, .cats {
16    color: var(--cute-color);
17  }
18}
19
```

cute
cats

Упс..

[CSS native variables not working in media queries](#)

Переменные в препроцессорах

Плюсы

- подддерживаются всеми браузерами

Переменные в препроцессорах

Плюсы

- подддерживаются всеми браузерами
- вставляются в медиа-запросы

Переменные в препроцессорах

Плюсы

- подддерживаются всеми браузерами
- вставляются в медиа-запросы
- различные типы данных ([Sass Data Types](#))

Переменные в препроцессорах

Плюсы

- поддерживаются всеми браузерами
- вставляются в медиа-запросы
- различные типы данных ([Sass Data Types](#))
- можно использовать вместе с Custom properties

Переменные в препроцессорах

```
// SCSS

$vars: (
  primary: #7F583F,
);

:root {
  --primary: #{map-get($vars, primary)};
}

@mixin var($property, $varName) {
  #{$property}: map-get($vars, $varName);
  #{$property}: var(--#{$varName}, map-get($vars, $varName));
}

a {
  @include var(color, primary);
}
```

Переменные в препроцессорах

```
// SCSS

$vars: (
  primary: #7F583F,
);

:root {
  --primary: #{map-get($vars, primary)};
}

@mixin var($property, $varName) {
  #{$property}: map-get($vars, $varName);
  #{$property}: var(--#$varName, map-get($vars, $varName));
}

a {
  @include var(color, primary);
}

/* CSS output */

a {
  color: #7F583F;
  color: var(--primary, #7F583F);
}
```

Nesting

Matches, Reference Parent selector



Nesting

Matches, Reference Parent selector



CSS nesting drafts

- «Совпадает с любым» :matches() / :is()

Позволяет задать стили для элемента, вложенного в любой из передаваемых селекторов

Черновик рабочей группы – www.w3.org/TR/selectors-4/#matches

CSS nesting drafts

- «Совпадает с любым» :matches() / :is()

Позволяет задать стили для элемента, вложенного в любой из передаваемых селекторов

Черновик рабочей группы – www.w3.org/TR/selectors-4/#matches

```
h2 { color: indigo; }

:is(section, article, aside) h2 {
  color: tomato;
}
:-moz-any(section, article, aside) h2 {
  color: tomato;
}
:-webkit-any(section, article, aside) h2 {
  color: tomato;
}
```

```
<section>
  <h2>Section: Hello World!</h2>
</section>

<article>
  <h2>Article: Hello World!</h2>
</article>

<header>
  <h2>Header: Hello World!</h2>
</header>

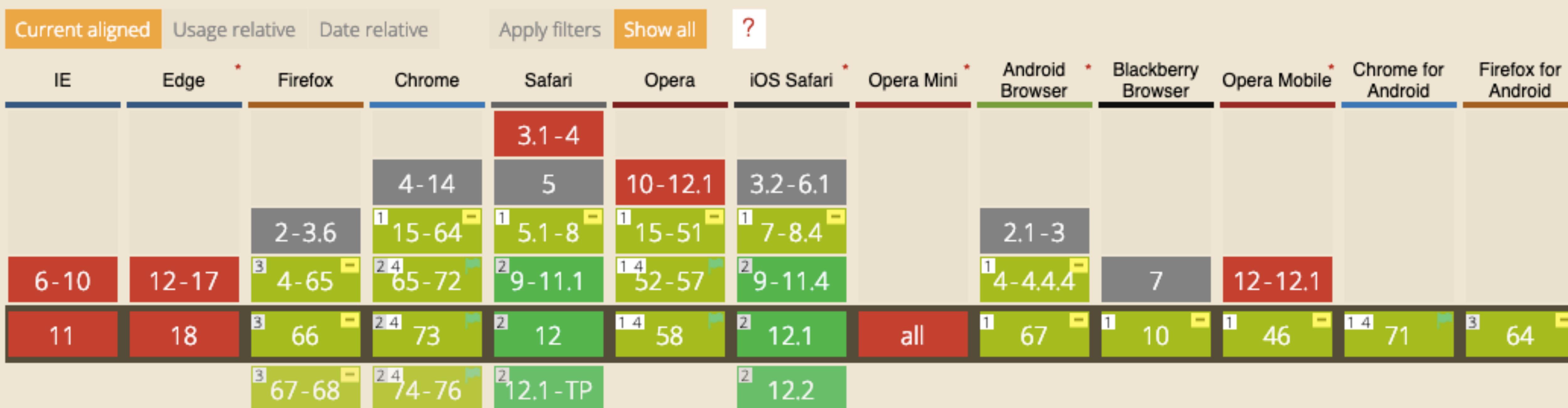
<aside>
  <h2>Aside: Hello World!</h2>
</aside>
```

Section: Hello World!
Article: Hello World!
Header: Hello World!
Aside: Hello World!

:matches() CSS pseudo-class

The `:matches()` (formerly `:any()`) pseudo-class checks whether the element at its position in the outer selector matches any of the selectors in its selector list. It's useful syntactic sugar that allows you to avoid writing out all the combinations manually as separate selectors. The effect is similar to nesting in Sass and most other CSS preprocessors.

Usage	% of all users	30	▼	?
Global	13.14% + 78.39%	= 91.53%		
unprefixed:	13.14% + 62.48%	= 75.62%		



CSS nesting drafts

- Селектор отношений :has()

Задаёт стили родительскому селектору, который имеет перечисленных потомков

Черновик рабочей группы – www.w3.org/TR/selectors-4/#relational

```
/* Все ссылки, в которых есть <span> */  
a:has(> span)
```

```
/* Все секции, в которых нет заголовков */  
section:not(:has(h1, h2, h3, h4, h5, h6))
```

CSS nesting drafts

- Селектор отношений :has()

Задаёт стили родительскому селектору, который имеет перечисленных потомков

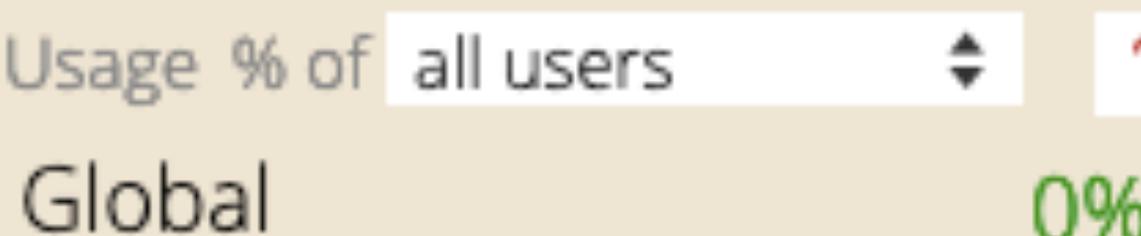
Черновик рабочей группы – www.w3.org/TR/selectors-4/#relational

```
/* Все ссылки, в которых есть <span> */  
a:has(> span)
```

```
/* Все секции, в которых нет заголовков */  
section:not(:has(h1, h2, h3, h4, h5, h6))
```

:has() CSS relational pseudo-class

Only select elements containing specified content. For example,
`a:has(>img)` selects all `<a>` elements that contain an `` child.



Current aligned	Usage relative	Date relative	Apply filters	Show all	?						
IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrom And
6-10	12-17	2-65	4-72	3.1-11.1	10-57	3.2-11.4		2.1-4.4.4	7	12-12.1	
11	18	66	73	12	58	12.1	all	67	10	46	7
	67-68	74-76	12.1-TP			12.2					

CSS nesting drafts

- Селектор вложенности '&'

Сылается на родительский селектор

Черновик – drafts.csswg.org/css-nesting-1/#nest-selector

```
/* direct nesting */

.foo, .parent &
  .child { color: red; }
}

/* CSS output
   :matches(.foo, .parent) .child { color: red; }
*/
```

CSS nesting drafts

- Вложение в правила '@nest'
Идентичен, но снимает некоторые ограничения

Черновик – drafts.csswg.org/css-nesting-1/#at-nest

```
/* at-nest rule */

.child {
  @nest .parent & { color: blue; }
}

/* CSS output
   .parent .child { color: blue; }
*/
```

SCSS

```
.some-class {  
  &.another-class { }  
}
```

SCSS

```
.some-class {  
  &.another-class { }  
}
```

Preprocessors nesting

- Родительский селектор '&' ([пример на sassmeister](#))

LibSass v3.5.2	CSS
<pre> 1 .header { 2 background-color: white; 3 4 &__logo { 5 color: darkblue; 6 7 &:hover { 8 color: blue; 9 } 10} 11 12.parent & { 13 background-color: whitesmoke; 14} 15 16.is-fixed { 17 position: fixed; 18} 19 </pre>	<pre> 1 .header { 2 background-color: white; 3 } 4 5.header__logo { 6 color: darkblue; 7 } 8 9.header__logo:hover { 10 color: blue; 11 } 12 13.parent .header { 14 background-color: whitesmoke; 15 } 16 17.header.is-fixed { 18 position: fixed; 19 } </pre>

[Подробнее про &](#)

Preprocessors nesting

- Вложенность селекторов и CSS-свойств ([пример на sassmeister](#))

LibSass v3.5.2	CSS
<pre>1 .heading { 2 font: { 3 size: 24px; 4 family: "Proxima Nova", sans-serif; 5 weight: 600; 6 } 7 8 span { 9 border: { 10 width: 2px; 11 style: solid; 12 radius: 10px; 13 } 14 } 15 }</pre>	<pre>1 .heading { 2 font-size: 24px; 3 font-family: "Proxima Nova", sans-serif; 4 font-weight: 600; 5 } 6 7 .heading span { 8 border-width: 2px; 9 border-style: solid; 10 border-radius: 10px; 11 } 12 13 14 15 }</pre>

LibSass v3.5.2

```
1 .socials {  
2   position: relative;  
3   width: 100%;  
4   z-index: 4;  
5  
6 @media screen and (min-width: 600px) {  
7   position: absolute;  
8   width: auto;  
9 }  
10  
11 &_link {  
12   padding: 12px;  
13   margin: 16px 12px 0;  
14  
15 @media screen and (min-width: 600px) {  
16   padding: 16px;  
17   margin: 0 5px;  
18 }  
19 }  
20 }  
21  
22 |
```

CSS

```
1 .socials {  
2   position: relative;  
3   width: 100%;  
4   z-index: 4;  
5 }  
6  
7 @media screen and (min-width: 600px) {  
8   .socials {  
9     position: absolute;  
10    width: auto;  
11  }  
12 }  
13  
14 .socials__link {  
15   padding: 12px;  
16   margin: 16px 12px 0;  
17 }  
18  
19 @media screen and (min-width: 600px) {  
20   .socials__link {  
21     padding: 16px;  
22     margin: 0 5px;  
23   }  
24 }
```

[CSS output](#)

Советы

Советы

- Чем меньше уровней вложенности, тем лучше (поиск вложенных селекторов идёт с самого последнего, что в больших количествах может замедлить рендер)

Советы

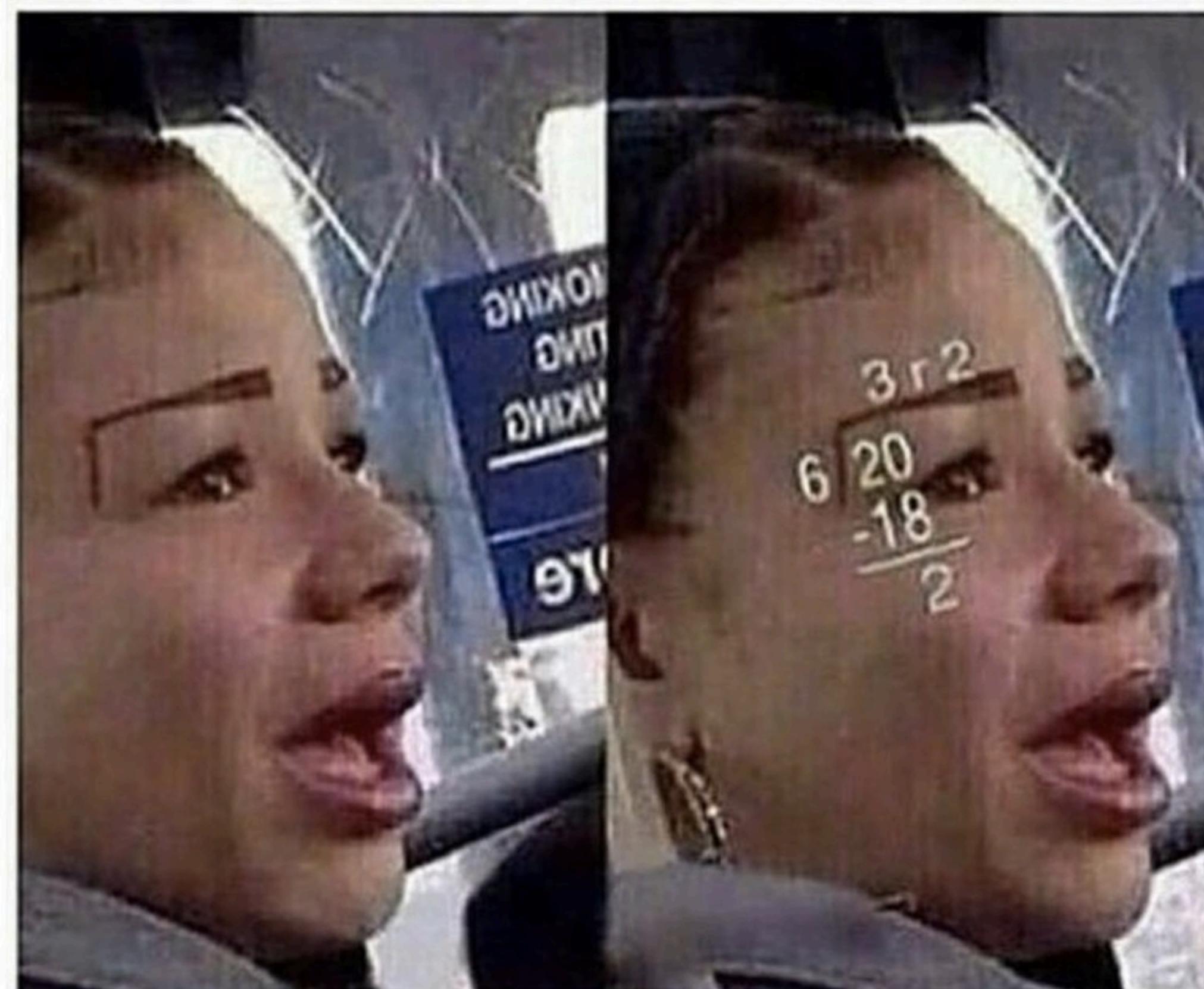
- Чем меньше уровней вложенности, тем лучше (поиск вложенных селекторов идёт с самого последнего, что в больших количествах может замедлить рендер)
- Осторожно используйте жесткое наследование (`.parent > .children`)

Советы

- Чем меньше уровней вложенности, тем лучше (поиск вложенных селекторов идёт с самого последнего, что в больших количествах может замедлить рендер)
- Осторожно используйте жесткое наследование (`.parent > .children`)
- Не ленитесь форматировать ваши стили (строки, отступы и т.д)

Math Functions

They said math wasn't used in everyday life





Benjamin De Cock
@bdc

Читать

The CSS Working Group agreed this morning on adding many math functions. We now have:

- calc()
- min()
- max()
- clamp()
- sin()
- cos()
- tan()
- acos()
- asin()
- atan()
- atan2()
- hypot()
- sqrt()
- pow()



Benjamin De Cock

@bdc

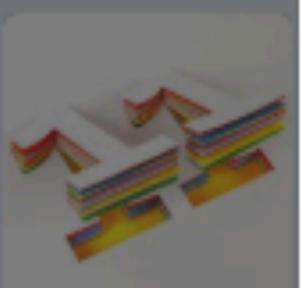
Designer [@stripe](#), lurker [@csswg](#).

⌚ San Francisco, CA

↗ [dribbble.com/bdc](#)

📅 Дата регистрации: ноябрь 2007 г.

🖼 386 фото или видео



The face of CSS is rapidly changing.

16:50 - 27 февр. 2019 г. [San Francisco, CA](#)



Читать

Не пользуетесь
Твиттером?

Зарегистрируйтесь и получите свою
собственную персонализированную
ленту!

Регистрация

Зам также может
понравиться

Обновить

Max "Tim Van Damme" Vo

Math Expressions

- Базовая арифметика: `calc()`

Math Expressions

- Базовая арифметика: `calc()`
- Функции сравнения: `min()`, `max()` и `clamp()` ([черновик](#))

Math Expressions

- Базовая арифметика: `calc()`
- Функции сравнения: `min()`, `max()` и `clamp()` ([черновик](#))
- Тригонометрические функции: `sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()` и `atan2()` ([черновик](#))

[Hardcore CSS calc\(\)](#)

[Пример использования min\(\) max\(\) clamp\(\)](#)

CSS math functions min(), max() and clamp()

[WD](#)

Usage

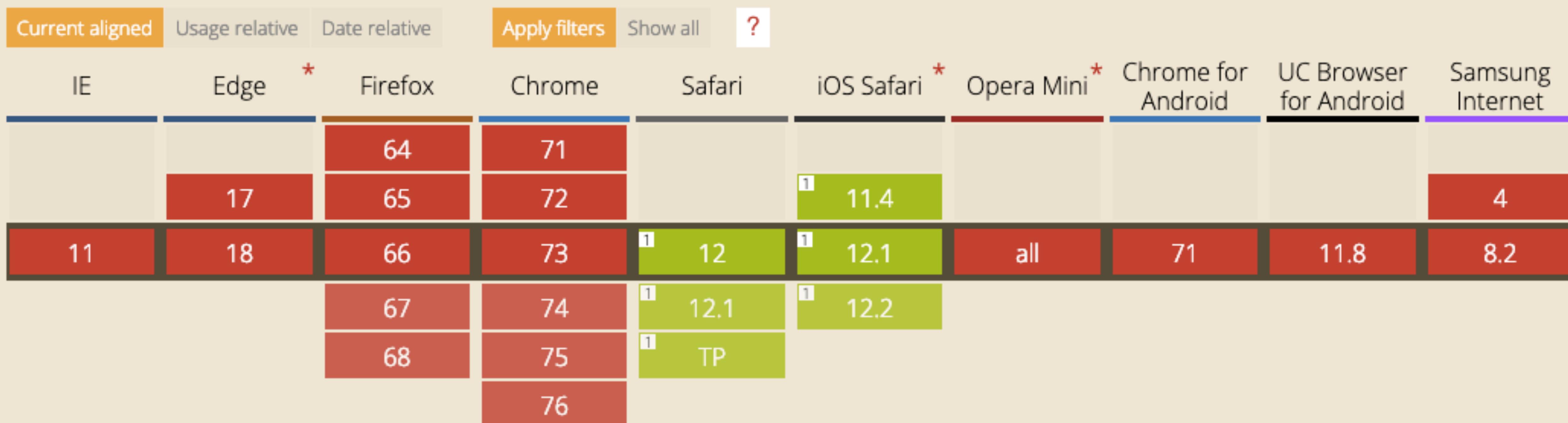
Global

% of all users



0% + 11.61% = 11.61%

More advanced mathematical expressions in addition to calc()



Notes

Known issues (0)

Resources (8)

Feedback

¹ Does not support clamp()

Я могу уже делать всё это ..используя препроцессоры

1. В Sass есть встроенные функции для математических операций:

Документация SassScript – sass-lang.com/documentation/Sass/Script/Functions.html

Я могу уже делать всё это ..используя препроцессоры

1. В Sass есть встроенные функции для математических операций:

[percentage\(\\$number\)](#),

[round\(\\$number\)](#),

[ceil\(\\$number\)](#),

[floor\(\\$number\)](#),

[abs\(\\$number\)](#),

[min\(\\$numbers...\)](#),

[max\(\\$numbers...\)](#),

[random\(\)](#)

Документация SassScript – sass-lang.com/documentation/Sass/Script/Functions.html

Я могу уже делать всё это ..используя препроцессоры

2. Написать свои функции, либо использовать библиотеку Compass:

Я могу уже делать всё это ..используя препроцессоры

2. Написать свои функции, либо использовать библиотеку Compass:

pi(),

e(),

sin(\$number),

cos(\$number),

tan(\$number),

asin(\$number),

acos(\$number),

atan(\$number),

logarithm(\$number, \$base),

sqrt(\$number),

pow(\$number, \$exponent)

Math Expressions

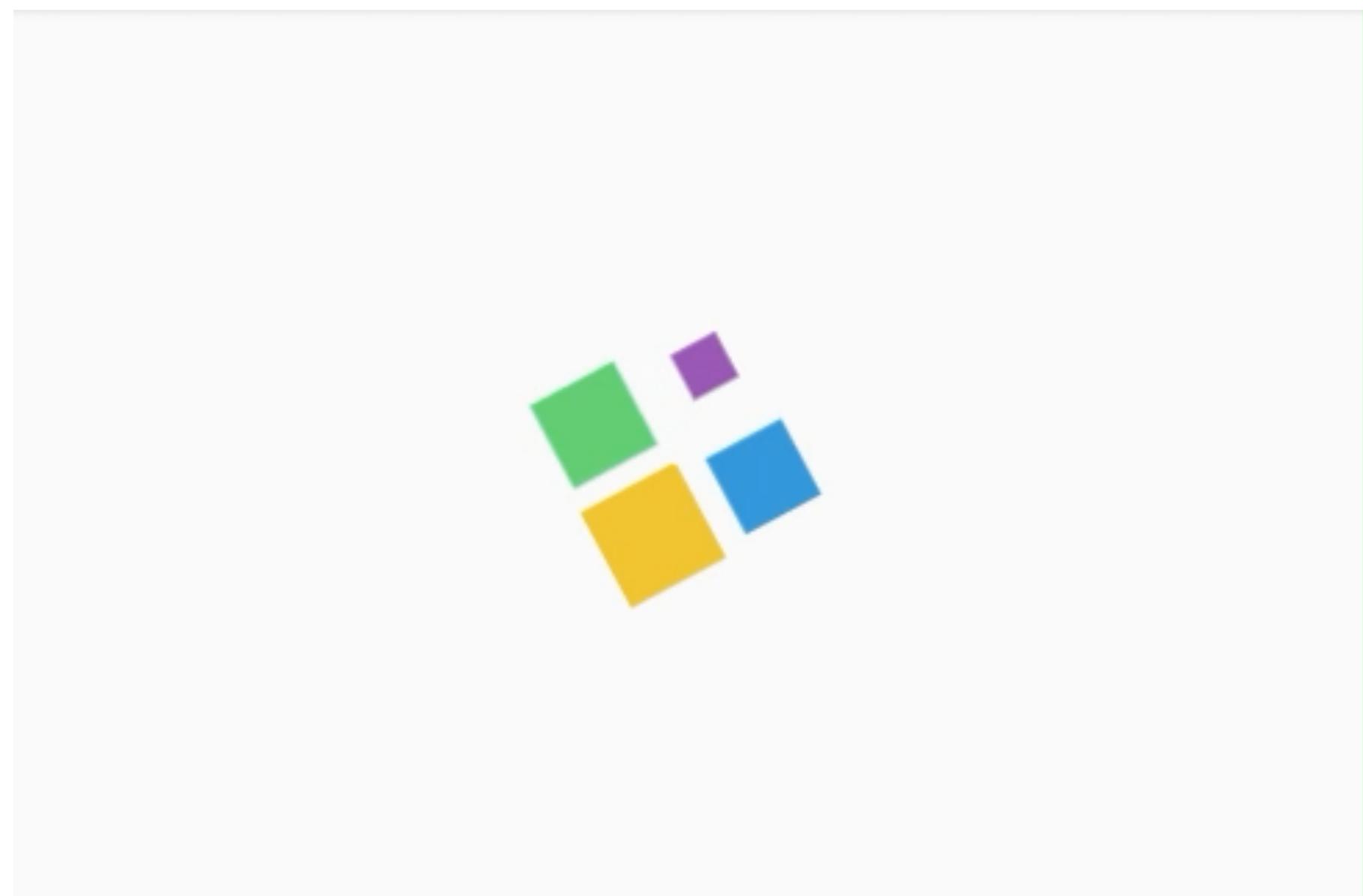
```
@keyframes gw {
  @for $j from 0 through $n {
    #{$j*100%/$n} {
      $sh-list: ();

      @for $i from 0 to $n {
        $curr-angle: ($i + .5)*$base-angle;

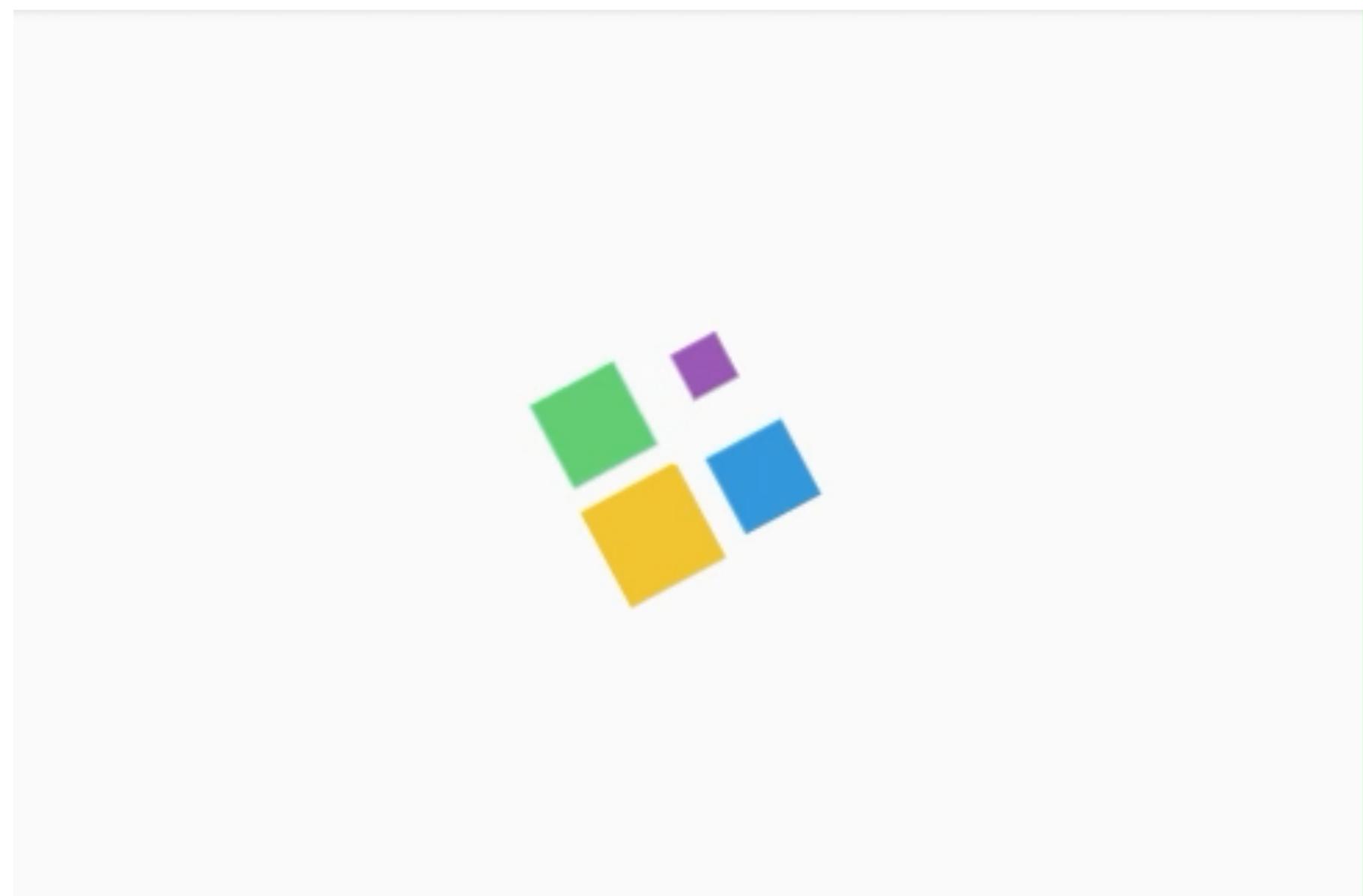
        $sh-list: $sh-list,
        $k*$d*cos($curr-angle) $k*$d*sin($curr-angle)
        0 $d/2 + abs(3*$d/2*sin((($j + $i)*$base-angle/2))
        nth($c-list, $i + 1);
      }

      box-shadow: $sh-list;
    }
  }
}
```

Math Expressions

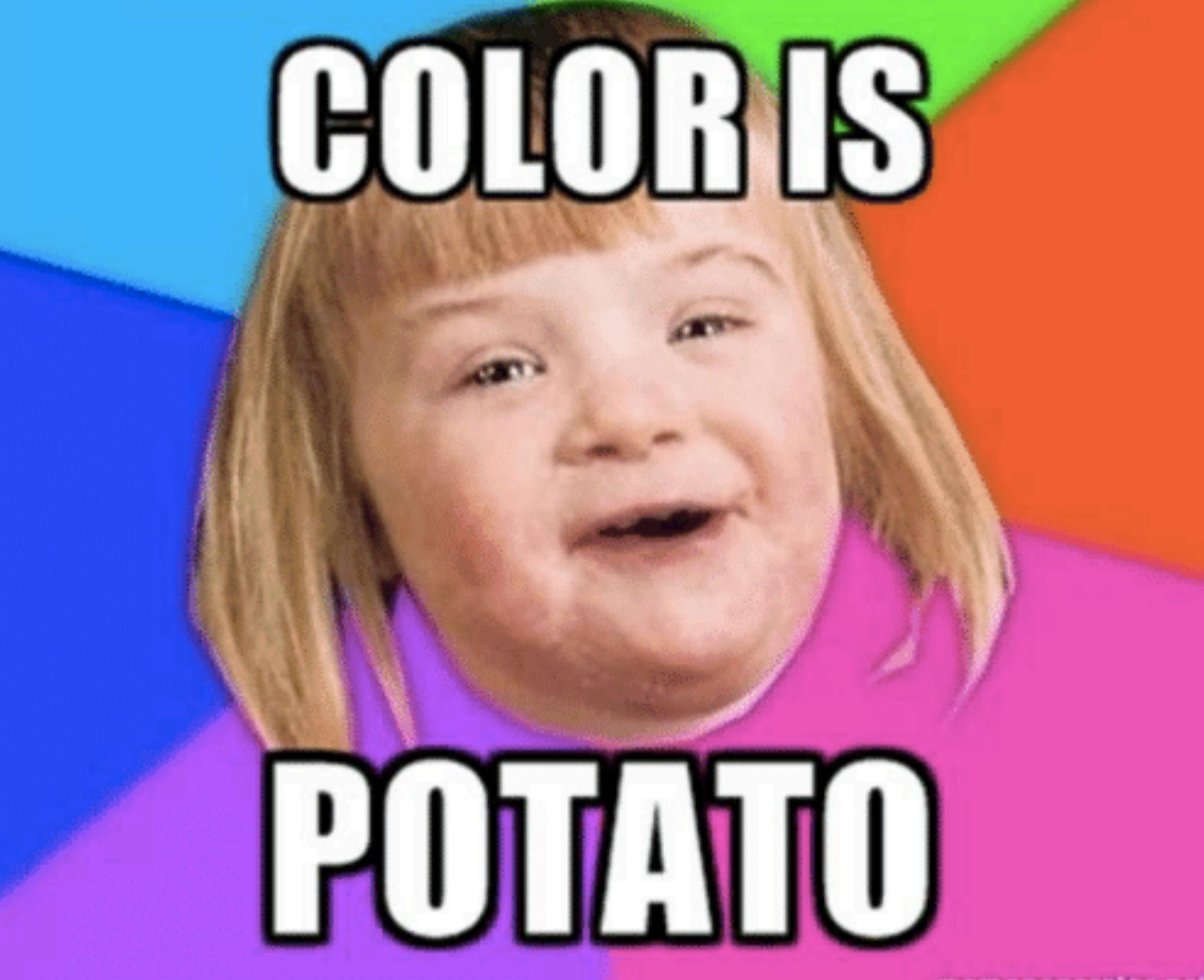


Math Expressions



Color Functions

MY FAVORITE
COLOR IS



POTATO

CSS Color Functions

- Функция color-mod() в старой спецификации [CSS Color Module Level 4](#) (05 июля 2016 г.)

CSS Color Functions

- Функция color-mod() в старой спецификации [CSS Color Module Level 4](#) (05 июля 2016 г.)
- Плагин [PostCSS color-mod\(\) Function](#)

The `color-mod()` function accepts `red()`, `green()`, `blue()`, `a() / alpha()`, `rgb()`, `h() / hue()`, `s() / saturation()`, `l() / lightness()`, `w() / whiteness()`, `b() / blackness()`, `tint()`, `shade()`, `blend()`, `blendalpha()`, and `contrast()` color adjusters.

§ Changes

§ Changes since Working Draft of 05 July 2016

- Initial value of the "color" property is now black
- Clarify hue in LCH is modulo 360deg
- Clarify allowed range of L in LCH and Lab, and meaning of L=100
- Update references for colorspaces used in video
- Add ProPhotoRGB predefined colorspace
- Correct black and white luminance levels for image-p3
- Clarify image-p3 transfer function
- Add a98rgb colorspace
- Clarify that currentColor's computed value is not the resolved color
- Update syntax is examples to conform to latest specification
- Remove the color-mod() function
- Drop the "media" from propdef tables

Sass Color Functions

- red(\$color), green(\$color), blue(\$color)
- hue(\$color), saturation(\$color), lightness(\$color)
- lighten(\$color, \$amount), darken(\$color, \$amount)
- saturate(\$color, \$amount), desaturate(\$color, \$amount)
- grayscale(\$color)
- alpha(\$color) / opacity(\$color)
- transparentize(\$color, \$amount) / fade-out(\$color, \$amount)
- opacify(\$color, \$amount) / fade-in(\$color, \$amount)

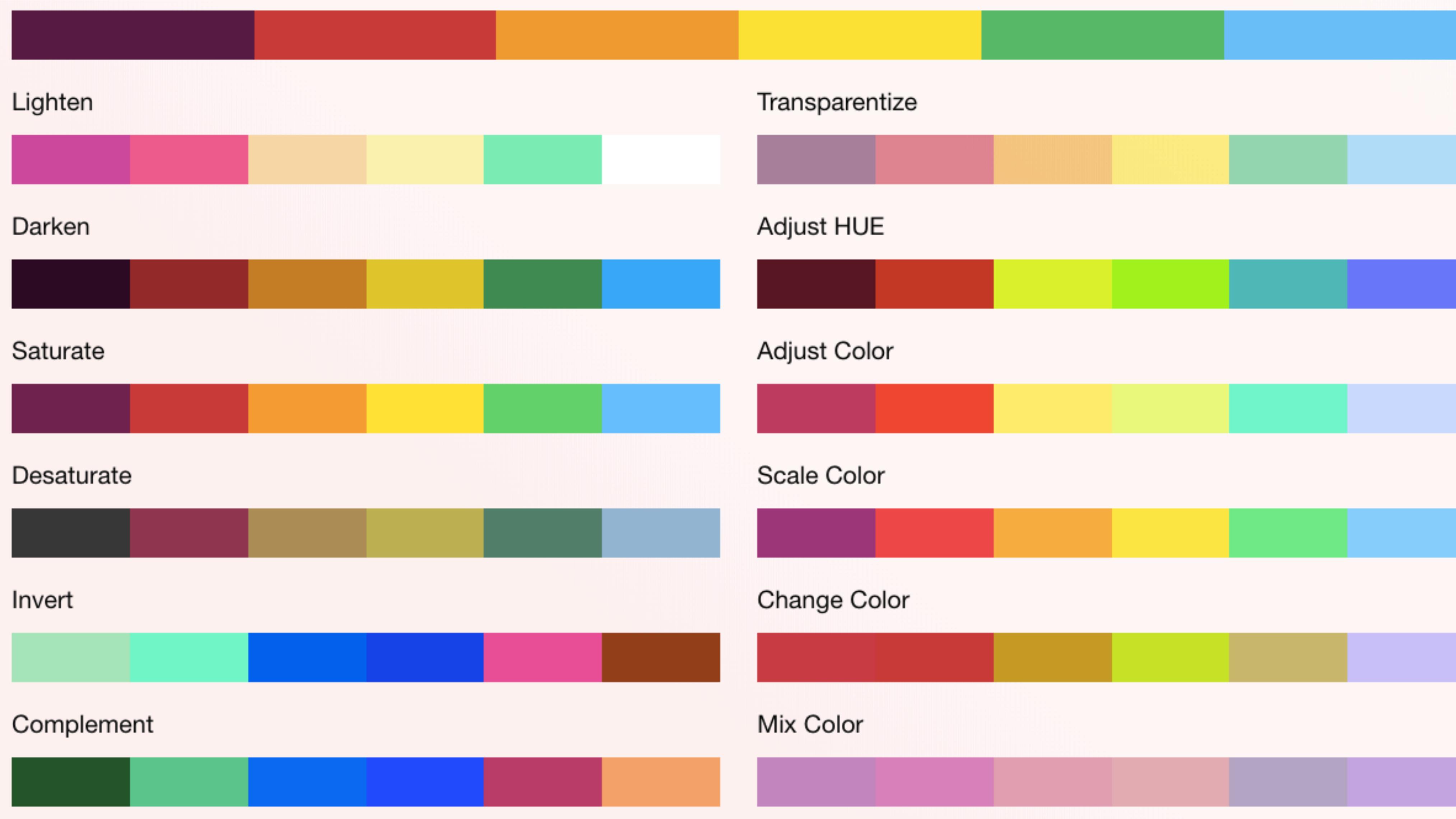
- mix(\$color1, \$color2, [\$weight])
- adjust-hue(\$color, \$degrees)

- adjust-color(\$color, [\$red], [\$green], [\$blue], [\$hue], [\$saturation], [\$lightness], [\$alpha])
- change-color(\$color, [\$red], [\$green], [\$blue], [\$hue], [\$saturation], [\$lightness], [\$alpha])
- scale-color(\$color, [\$red], [\$green], [\$blue], [\$saturation], [\$lightness], [\$alpha])

- complement(\$color), invert(\$color, [\$weight])

- ie-hex-str(\$color)

Sass Color Functions



Sass Color Functions

```
// SCSS

$button-color: hsl(242, 29%, 36%);

.button {
    background-color: $button-color;

    &:hover {
        background-color: lighten($button-color, 16%);
    }

    &:focus {
        background-color: darken($button-color, 8%);
    }

    &:disabled {
        // change-color – тоже самое что hsl(hue($button-color), 20%, 80%);
        background-color: change-color($button-color, $saturation: 20%, $lightness: 80%);
        color: invert($button-color, 10%);
    }
}
```

Sass Color Functions

```
// SCSS

$button-color: hsl(242, 29%, 36%);

.button {
    background-color: $button-color;

    &:hover {
        background-color: lighten($button-color, 16%);
    }

    &:focus {
        background-color: darken($button-color, 8%);
    }

    &:disabled {
        // change-color – тоже самое что hsl(hue($button-color), 20%, 80%);
        background-color: change-color($button-color, $saturation: 20%, $lightness: 80%);
        color: invert($button-color, 10%);
    }
}
```

Sass Color Functions

```
// SCSS

$button-color: hsl(242, 29%, 36%);

.button {
    background-color: $button-color;

    &:hover {
        background-color: lighten($button-color, 16%);
    }

    &:focus {
        background-color: darken($button-color, 8%);
    }

    &:disabled {
        // change-color – тоже самое что hsl(hue($button-color), 20%, 80%);
        background-color: change-color($button-color, $saturation: 20%, $lightness: 80%);
        color: invert($button-color, 10%);
    }
}
```

Sass Color Functions

```
// SCSS

$button-color: hsl(242, 29%, 36%);

.button {
    background-color: $button-color;

    &:hover {
        background-color: lighten($button-color, 16%);
    }

    &:focus {
        background-color: darken($button-color, 8%);
    }

    &:disabled {
        // change-color - тоже самое что hsl(hue($button-color), 20%, 80%);
        background-color: change-color($button-color, $saturation: 20%, $lightness: 80%);
        color: invert($button-color, 10%);
    }
}
```

Sass Color Functions

```
<!-- HTML -->
```

```
<button class="button">  
  Hover or Focus on me  
</button>
```

```
<button class="button" disabled>  
  I'm disabled  
</button>
```

Hover or Focus on me

I'm disabled

Sass Color Functions

```
<!-- HTML -->
```

```
<button class="button">  
  Hover or Focus on me  
</button>
```

```
<button class="button" disabled>  
  I'm disabled  
</button>
```

Hover or Focus on me

I'm disabled

Sass Colour Function Calculator

A tool for calculating the SASS colour function required to get from one colour to another.

ORIGINAL COLOUR

#00d1b4

TARGET COLOUR

#b1f5e8

```
lighten(desaturate(adjust-hue(#00d1b4, -3), 22.73), 41.76)
```

Created by

RAZOR

The Technology Works

Using [tinycolor.js](#) and work by [Hugo Giraudel](#)

[Сылочка на сервис тут](#)

@Mixin

Примеси

@Mixin

Миксины позволяют создавать группы CSS-свойств, которые можно повторно использовать

[Подробнее о миксинах](#)

Миксины позволяют создавать группы CSS-свойств, которые можно повторно использовать

- принимают параметры

Миксины позволяют создавать группы CSS-свойств, которые можно повторно использовать

- принимают параметры
- в разы ускоряют время написания стилей

Миксины позволяют создавать группы CSS-свойств, которые можно повторно использовать

- принимают параметры
- в разы ускоряют время написания стилей
- могут привести к дублированию

/* Как могло бы быть в CSS */

```
:root {  
  --toolbar-theme: {  
  
    background-color: hsl(120, 70%, 95%);  
    border-radius: 4px;  
    border: 1px solid var(--theme-color late);  
  };  
  --toolbar-title-theme: {  
    color: green;  
  };  
}  
  
.toolbar {  
  @apply --toolbar-theme;  
}  
.toolbar > .title {  
  @apply --toolbar-title-theme;  
}
```

Неофициальный черновик @apply

<https://tabatkins.github.io/specs/css-apply-rule/>



Я ТЕБЯ НАПИСАЛ

Я ТЕБЯ И СЛОМАЮ

troll-pikabu.ru
troll-face.ru

Tab Completion

I'm [Tab Atkins Jr.](#), and I wear many hats. I work for Google on the Chrome browser as a Web Standards Hacker. I'm also a member of the CSS Working Group, and am either a member or contributor to several other working groups in the W3C. You can contact me [here](#).

[Listing of All Posts](#)

[Сылочка тут](#)

Why I Abandoned @apply

Apr 24

Last updated: Monday, May 8 2017

Some time ago, I created the modern spec for CSS Variables, which lets you store uninterpreted values into "custom properties", then substitute those values into real properties later on.

This worked great, and was particularly convenient for Shadow DOM, which wanted a simple, controllable way to let the outside page twiddle the styling of a component, but without giving them full access to the internals of the component. The component author could just use a couple of `var()` functions, with default values so that

@mixin

```
// SCSS Multi line clamp

@mixin line-clamp($lines: 1, $font-size: 1rem, $line-height: 1.3) {
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: $lines;

  font-size: $font-size;
  line-height: $line-height;
  max-height: $font-size * $line-height * $lines;

  overflow: hidden;
  text-overflow: ellipsis;
}
```

@mixin

```
// SCSS Multi line clamp

@mixin line-clamp($lines: 1, $font-size: 1rem, $line-height: 1.3) {
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: $lines;

  font-size: $font-size;
  line-height: $line-height;
  max-height: $font-size * $line-height * $lines;

  overflow: hidden;
  text-overflow: ellipsis;
}
```

@mixin

```
// SCSS Multi line clamp

@mixin line-clamp($lines: 1, $font-size: 1rem, $line-height: 1.3) {
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: $lines;

  font-size: $font-size;
  line-height: $line-height;
  max-height: $font-size * $line-height * $lines;

  overflow: hidden;
  text-overflow: ellipsis;
}
```

@mixin

```
// Usage Multi line clamp mixin

.profile-name {
  @include line-clamp;
  max-width: 100px;
}

.card_title {
  @include line-clamp(2, 3rem, 1.2);
}

.card_description {
  @include line-clamp($lines: 3, $font-size: 2rem);
}
```

@mixin

```
// Usage Multi line clamp mixin

.profile-name {
  @include line-clamp;
  max-width: 100px;
}

.card_title {
  @include line-clamp(2, 3rem, 1.2);
}

.card_description {
  @include line-clamp($lines: 3, $font-size: 2rem);
}
```

@mixin

```
// Usage Multi line clamp mixin

.profile-name {
  @include line-clamp;
  max-width: 100px;
}

.card_title {
  @include line-clamp(2, 3rem, 1.2);
}

.card_description {
  @include line-clamp($lines: 3, $font-size: 2rem);
}
```

@mixin

```
// Usage Multi line clamp mixin

.profile-name {
  @include line-clamp;
  max-width: 100px;
}

.card_title {
  @include line-clamp(2, 3rem, 1.2);
}

.card_description {
  @include line-clamp($lines: 3, $font-size: 2rem);
}
```

Без использования миксина

Этот заголовок самый длинный, много строк ничего не понятно

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строки в каждой карточке, поэтому мы используем миксин, который обрежет необходимое количество строк, а остальное скроет

С использованием миксина

Этот заголовок самый длинный, много стро...

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строк...

Без использования миксина

Этот заголовок самый длинный, много строк ничего не понятно

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строки в каждой карточке, поэтому мы используем миксин, который обрежет необходимое количество строк, а остальное скроет

С использованием миксина

Этот заголовок самый длинный, много стро...

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строк...

Без использования миксина

Этот заголовок самый длинный, много строк ничего не понятно

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строки в каждой карточке, поэтому мы используем миксин, который обрежет необходимое количество строк, а остальное скроет

С использованием миксина

Этот заголовок самый длинный, много стро...

Очень длинное описание, которое занимает много строк, но нам надо отображать только 3 строк...

@mixin

```
// SCSS Media query mixin

$small: 640px;
$medium: 1024px;

@mixin sm-only {
  @media screen and (min-width: $small) and (max-width: #{ $medium - 1 }) {
    @content;
  }
}
```

@mixin

```
// SCSS Media query mixin

$small: 640px;
$medium: 1024px;

@mixin sm-only {
  @media screen and (min-width: $small) and (max-width: ${$medium - 1}) {
    @content;
  }
}

// SCSS Media query mixin usage

.class {
  @include sm-only() {
    background-color: pink;
  }
}
```

@mixin

```
// SCSS Media query mixin

$small: 640px;
$medium: 1024px;

@mixin sm-only {
  @media screen and (min-width: $small) and (max-width: #{($medium - 1)}) {
    @content;
  }
}
```

```
// SCSS Media query mixin usage

.class {
  @include sm-only() {
    background-color: pink;
  }
}
```

@mixin

```
// SCSS Media query mixin

$small: 640px;
$medium: 1024px;

@mixin sm-only {
  @media screen and (min-width: $small) and (max-width: #{($medium - 1)}) {
    @content;
  }
}

// SCSS Media query mixin usage

.class {
  @include sm-only() {
    background-color: pink;
  }
}
```

@mixin

```
1 <div class="class"></div>

2
3 @mixin sm-only {
4   @media screen and (min-width: #{map-
5     get($breakpoints, sm)}) and (max-width:
6     #{map-get($breakpoints, md) - 1}) {
7     @content;
8   }
9
10 // SCSS Media query mixins usage
11 .class {
12   margin: 10rem 2rem;
13   width: calc(100% - 4rem);
14   height: 72px;
15   background-color: lightblue;
16   @include sm-only() {
17     background-color: pink;
18   }
}
```



@mixin

```
1 <div class="class"></div>

2
3 @mixin sm-only {
4   @media screen and (min-width: #{map-
5     get($breakpoints, sm)}) and (max-width:
6     #{map-get($breakpoints, md) - 1}) {
7     @content;
8   }
9
10 // SCSS Media query mixins usage
11 .class {
12   margin: 10rem 2rem;
13   width: calc(100% - 4rem);
14   height: 72px;
15   background-color: lightblue;
16   @include sm-only() {
17     background-color: pink;
18   }
}
```



Больше миксинов

- библиотеки готовых примесей: [Compass](#) и [Bourbon](#)
- стандартные [миксины для медиа-выражений](#)
- пишешь одну строчку и добавляешь треугольник в нужном месте
codepen.io/eduardoboucas/pen/JomROG
- весь animate.css в одном миксине
codepen.io/jakob-e/pen/bENvGj
- линейный градиент sitepoint.com/building-linear-gradient-mixin-sass/
- интересная библиотека примесей для обращения к дочерним элементам lukyvj.github.io/family.scss
- анимация глитч codepen.io/ayamflow/pen/DCifh

@extend

Расширения

@extend

```
// SCSS

.error {
    border: 1px red;
    background-color: red;
}

.fatal-error {
    @extend .error;

    border-width: 5px;
}

.serious-error {
    @extend .error;
}
```

@extend

```
// SCSS                                     /* CSS output */  
  
.error {  
  border: 1px red;  
  background-color: red;  
}  
  
.fatal-error {  
  @extend .error;  
  border-width: 5px;  
}  
  
.serious-error {  
  @extend .error;  
}
```

```
.error, .fatal-error, .serious-error {  
  border: 1px red;  
  background-color: red;  
}  
  
.fatal-error {  
  border-width: 5px;  
}
```

@extend

```
// SCSS

%error {
    border: 1px red;
    background-color: red;
}

.fatal-error {
    @extend %error;

    border-width: 5px;
}

.serious-error {
    @extend %error;
}
```

@extend

```
// SCSS

%error {
    border: 1px red;
    background-color: red;
}

.fatal-error {
    @extend %error;

    border-width: 5px;
}

.serious-error {
    @extend %error;
}
```

@extend

```
// SCSS                                     /* CSS output */  
  
%error {  
    border: 1px red;  
    background-color: red;  
}  
  
.fatal-error {  
    @extend %error;  
    border-width: 5px;  
}  
  
.serious-error {  
    @extend %error;  
}
```

.fatal-error, .serious-error {
 border: 1px red;
 background-color: red;
}

.fatal-error {
 border-width: 5px;
}

Что почитать?

- Как правильно использовать @extend

<https://www.smashingmagazine.com/2015/05/extending-in-sass-without-mess/>

- Почему вы должны избегать Sass @extend

<https://www.sitepoint.com/avoid-sass-extend/>

- Нет, серьезно: не используйте @extend

<https://webinista.com/updates/dont-use-extend-sass/>

Lists & Maps

Lists & Maps

- Списки

```
$list: value1 value2 value3;
```

```
// Обращение по индексу  
nth($list, 1);
```

[Подробнее](#)

- Ассоциативные массивы (карты)

```
$map: (key1: value1, key2: value2, key3: value3);
```

```
// Обращение по ключу  
map-get($map, key1);
```

[Подробнее](#)

Lists & Maps

- Списки

```
$list: value1 value2 value3;
```

```
// Обращение по индексу
```

```
nth($list, 1);
```

[Подробнее](#)

- Ассоциативные массивы (карты)

```
$map: (key1: value1, key2: value2, key3: value3);
```

```
// Обращение по ключу
```

```
map-get($map, key1);
```

[Подробнее](#)

Lists & Maps

- Списки

```
$list: value1 value2 value3;
```

```
// Обращение по индексу
```

```
nth($list, 1);
```

[Подробнее](#)

- Ассоциативные массивы (карты)

```
$map: (key1: value1, key2: value2, key3: value3);
```

```
// Обращение по ключу
```

```
map-get($map, key1);
```

[Подробнее](#)

Control Directives & Expressions

@if / @else, @for / @each

Условные операторы

- `@if / @else`

Принимает выражение SassScript и использует вложенные стили.

```
// SCSS input
p {
  @if 1 + 1 == 2 { border: 1px solid; }
  @else if 5 < 2 { border: 2px dotted; }
  @else           { border: 3px double; }
}
```

```
// CSS output
p {
  border: 1px solid;
}
```

[Подробнее](#)

```
@mixin css-triangle($color, $direction, $size: 6px) {  
    // base styles  
    border: $size solid transparent;  
  
    @if $direction == 'down' {  
        border-top-color: $color;  
        bottom: -$size * 2;  
        left: 50%;  
        transform: translateX(-50%);  
    }  
    @else if $direction == 'up' {  
        border-bottom-color: $color;  
        top: -$size * 2;  
        left: 50%;  
        transform: translateX(-50%);  
    }  
    @else if $direction == 'right' {  
        border-left-color: $color;  
        right: -$size * 2;  
        top: 50%;  
        transform: translateY(-50%);  
    }  
    @else if $direction == 'left' {  
        border-right-color: $color;  
        left: -$size * 2;  
        top: 50%;  
        transform: translateY(-50%);  
    }  
}
```

```
// Mixin usage

.tooltip {
  $tooltip-color: hsl(0, 0%, 100%);

  position: relative;
  padding: 16px;
  background-color: $tooltip-color;
  box-shadow: 0 2px 10px 0 rgba(gainsboro, 0.3);
  border-radius: 8px;

  &--down::after {
    @include css-triangle($tooltip-color, 'up');
  }

  &--up::after {
    @include css-triangle($tooltip-color, 'down', 10px);
  }

  &--right::after {
    @include css-triangle($tooltip-color, 'left', 10px);
  }

  &--left::after {
    @include css-triangle($tooltip-color, 'right');
  }
}
```

triangle up

triangle down

triangle left

triangle right

Циклы

- **@for**

Выводит набор стилей заданное число раз. Используйте ключевое слово **through** вместо **to**

```
@for $i from 1 through 3 {  
  .item-#${$i} { width: 2em * $i; }  
}
```

- **@each**

Считывает элементы из указанного списка и генерирует стили с использованием значений

```
@each $value in (sass, less, stylus) {  
  .#${$value} {  
    content: '#{$value} is a preprocessor';  
  }  
}
```

[Подробнее](#)

Циклы

```
// SCSS @each example

$status-colors: (
  'primary': hsl(245, 24%, 24%) ,
  'success': hsl(160, 56%, 56%) ,
  'info':    hsl(200, 96%, 64%) ,
  'warning': hsl(25, 96%, 56%) ,
  'danger':  hsl(355, 88%, 56%)
);

.message {
  // message base style

  @each $status, $bg-color in $status-colors {
    &--{$status} {
      background: $bg-color;
    }
  }
}
```

Циклы

```
// CSS output

.message {/*base styles*/

.message--primary { background: #312f4c; }

.message--success { background: #50cea4; }

.message--info { background: #4bc1fb; }

.message--warning { background: #fb7d23; }

.message--danger { background: #f22c3d; }}
```

```

$primaryLight: hsla(212, 48%, 80%, 1)
$primary: hsla(212, 48%, 64%, 1)
$primaryDark: hsla(212, 48%, 32%, 1)

$secondaryLight: hsla(330, 48%, 64%, 1)
$secondary: hsla(330, 44%, 40%, 1)
$secondaryDark: hsla(330, 56%, 24%, 1)

$primary-colors: (base: $primary, light: $primaryLight, dark: $primaryDark)
$secondary-colors: (base: $secondary, light: $secondaryLight, dark: $secondaryDark)
$component-colors: (primary: $primary-colors, secondary: $secondary-colors)

@each $status, $color in $component-colors
  .component--#{$status}

    .btn-raised, .fab--raised
      background-color: map_get($color, "dark")
      color: rgb(255, 255, 255)

      &:hover
        box-shadow: 0 2px 16px 0 rgba(map_get($color, "dark"), 0.48)
      &:focus
        background-color: map_get($color, "base")
        box-shadow: 0 1px 8px 0 rgba(map_get($color, "dark"), 0.6)
      &:active
        box-shadow: 0 0 0 0 transparent

    .btn--flat, .fab--flat
      background-color: rgba(map_get($color, "base"), 0.08)
      color: map_get($color, "dark")

      &:hover
        background-color: rgba(map_get($color, "base"), 0.18)
        box-shadow: 0 2px 16px 0 rgba(map_get($color, "base"), 0.48)
      &:focus
        background-color: rgba(map_get($color, "light"), 0.3)
        box-shadow: 0 1px 8px 0 rgba(map_get($color, "dark"), 0.6)
      &:active
        box-shadow: 0 0 0 0 transparent

  button:disabled
    $disabled-btn: mix(map_get($color, "dark"), rgb(240, 240, 240), 10%)
    background: desaturate($disabled-btn, 55%) !important
    pointer-events: none

```

```
1 .component--primary .btn--raised,  
2 .component--primary .fab--raised {  
3   background-color: #2a4f79;  
4   color: rgb(255, 255, 255);  
5 }  
6 .component--primary .btn--raised:hover,  
7 .component--primary .fab--raised:hover {  
8   box-shadow: 0 2px 16px 0 rgba(42, 79, 121, 0.48);  
9 }  
10 .component--primary .btn--raised:focus,  
11 .component--primary .fab--raised:focus {  
12   background-color: #77a0cf;  
13   box-shadow: 0 1px 8px 0 rgba(42, 79, 121, 0.6);  
14 }  
15 .component--primary .btn--raised:active,  
16 .component--primary .fab--raised:active {  
17   box-shadow: 0 0 0 0 transparent;  
18 }  
19 .component--primary .btn--flat,  
20 .component--primary .fab--flat {  
21   background-color: rgba(119, 160, 207, 0.08);  
22   color: #2a4f79;  
23 }  
24 .component--primary .btn--flat:hover,  
25 .component--primary .fab--flat:hover {  
26   background-color: rgba(119, 160, 207, 0.18);  
27   box-shadow: 0 2px 16px 0 rgba(119, 160, 207, 0.48);  
28 }  
29 .component--primary .btn--flat:focus,  
30 .component--primary .fab--flat:focus {  
31   background-color: rgba(180, 202, 228, 0.3);  
32   box-shadow: 0 1px 8px 0 rgba(42, 79, 121, 0.6);  
33 }  
34 .component--primary .btn--flat:active,  
35 .component--primary .fab--flat:active {  
36   box-shadow: 0 0 0 0 transparent;  
37 }
```

```
1 .component--primary .btn--raised,  
2 .component--primary .fab--raised {  
3   background-color: #2a4f79;  
4   color: rgb(255, 255, 255);  
5 }  
6 .component--primary .btn--raised:hover,  
7 .component--primary .fab--raised:hover {  
8   box-shadow: 0 2px 16px 0 rgba(42, 79, 121, 0.48);  
9 }  
10 .component--primary .btn--raised:focus,  
11 .component--primary .fab--raised:focus {  
12   background-color: #77a0cf;  
13   box-shadow: 0 1px 8px 0 rgba(42, 79, 121, 0.6);  
14 }  
15 .component--primary .btn--raised:active,  
16 .component--primary .fab--raised:active {  
17   box-shadow: 0 0 0 0 transparent;  
18 }  
19 .component--primary .btn--flat,  
20 .component--primary .fab--flat {  
21   background-color: rgba(119, 160, 207, 0.08);  
22   color: #2a4f79;  
23 }  
24 .component--primary .btn--flat:hover,  
25 .component--primary .fab--flat:hover {  
26   background-color: rgba(119, 160, 207, 0.18);  
27   box-shadow: 0 2px 16px 0 rgba(119, 160, 207, 0.48);  
28 }  
29 .component--primary .btn--flat:focus,  
30 .component--primary .fab--flat:focus {  
31   background-color: rgba(180, 202, 228, 0.3);  
32   box-shadow: 0 1px 8px 0 rgba(42, 79, 121, 0.6);  
33 }  
34 .component--primary .btn--flat:active,  
35 .component--primary .fab--flat:active {  
36   box-shadow: 0 0 0 0 transparent;  
37 }
```

```

$size: 3vmin;
$n: 14;
$light: #eddede;
$dark: #232323;

body {
  margin: 0;
  min-height: 100vh;
  background-color: $light;
}

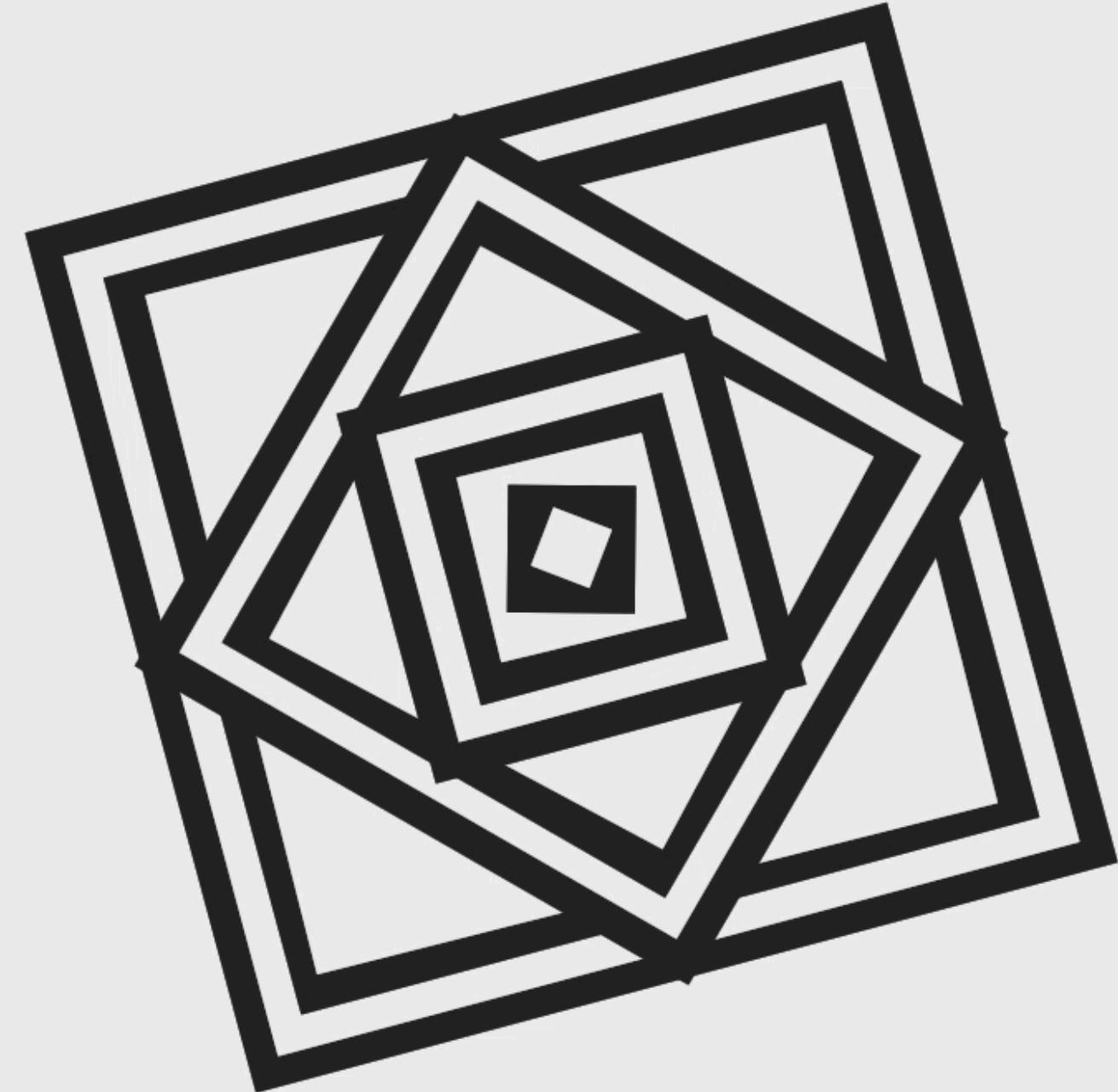
.square {
  position: absolute;
  top: 50%;
  left: 50%;
  background-color: $light;
  transform: translate(-50%, -50%);
}

.square:nth-child(2n+1) {
  background-color: $dark;
}

@for $i from 1 through $n {
  $c: $n + 1 - $i;
  .square:nth-child(#{$c}) {
    width: $size * $i;
    height: $size * $i;
    animation: rot#{$i} infinite ease #{1.5 * $n / $c}s;
  }
}

@keyframes rot#{$i} {
  0%, #{$50 - 150 / ($i+3)}%, 100% {
    transform: translate(-50%, -50%) rotate(30deg);
  }
  50%, #{$100 - 150 / ($i+3)}% {
    transform: translate(-50%, -50%) rotate(-15deg);
  }
}
}

```



```

$size: 3vmin;
$n: 14;
$light: #eddede;
$dark: #232323;

body {
  margin: 0;
  min-height: 100vh;
  background-color: $light;
}

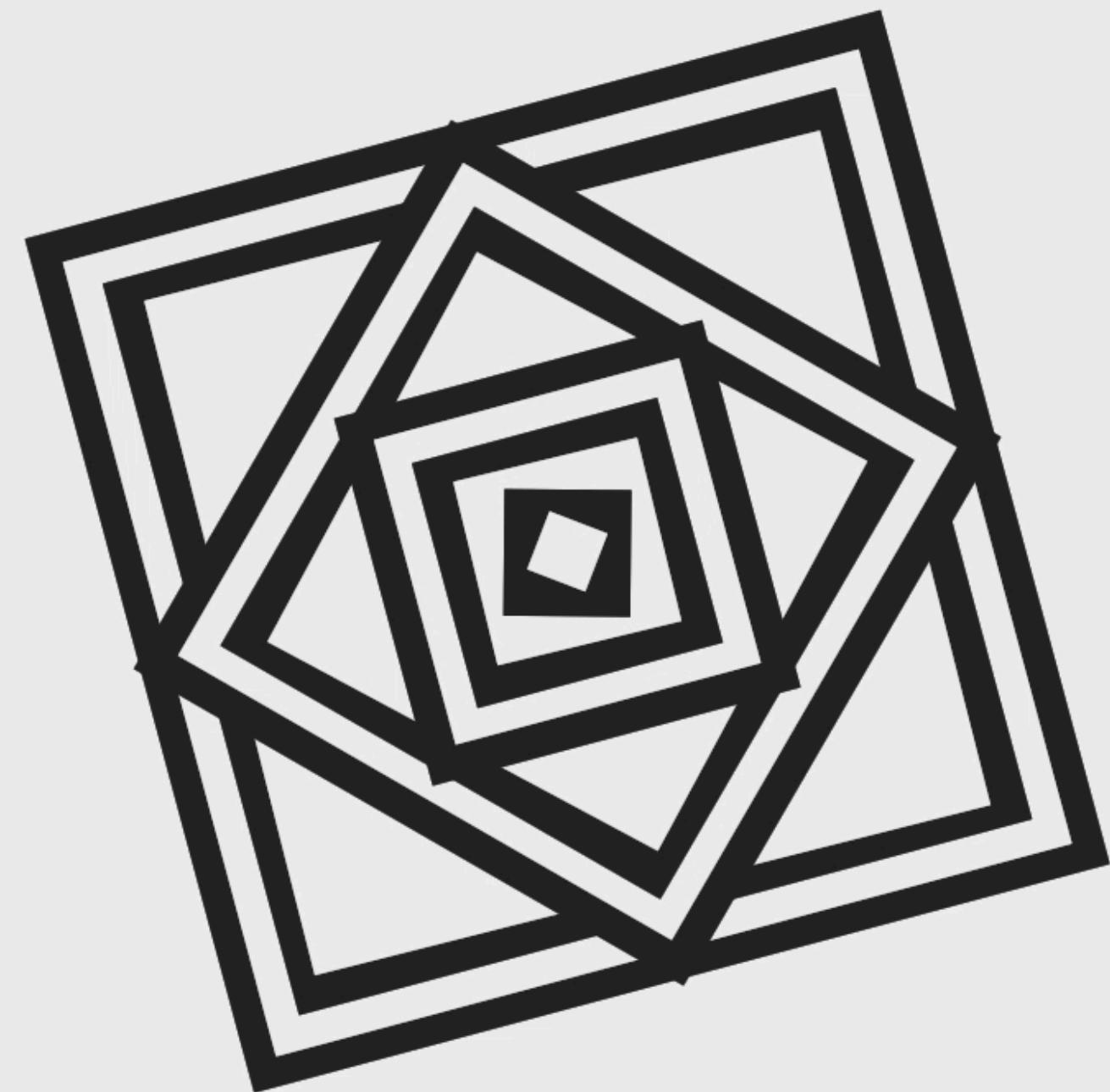
.square {
  position: absolute;
  top: 50%;
  left: 50%;
  background-color: $light;
  transform: translate(-50%, -50%);
}

.square:nth-child(2n+1) {
  background-color: $dark;
}

@for $i from 1 through $n {
  $c: $n + 1 - $i;
  .square:nth-child(#{$c}) {
    width: $size * $i;
    height: $size * $i;
    animation: rot#{$i} infinite ease #{1.5 * $n / $c}s;
  }
}

@keyframes rot#{$i} {
  0%, #{$50 - 150 / ($i+3)}%, 100% {
    transform: translate(-50%, -50%) rotate(30deg);
  }
  50%, #{$100 - 150 / ($i+3)}% {
    transform: translate(-50%, -50%) rotate(-15deg);
  }
}
}

```



```

1 $size: 3vmin;
2 $n: 14;
3 $light: #ededed;
4 $dark: #232323;
5
6 body {
7   margin: 0;
8   min-height: 100vh;
9   background-color: $light;
10 }
11
12 .square {
13   position: absolute;
14   top: 50%;
15   left: 50%;
16   background-color: $light;
17   transform: translate(-50%, -50%);
18 }
19 .square:nth-child(2n+1) {
20   background-color: $dark;
21 }
22
23 @for $i from 1 through $n {
24   $c: $n + 1 - $i;
25
26   .square:nth-child(#{$c}) {
27     width: $size * $i;
28     height: $size * $i;
29     animation: rot#{$i} infinite ease #{1.5 * $n / $c}s;
30   }
31
32 @keyframes rot#{$i} {
33   0%, #{50 - 150 / ($i+3)}%, 100% {
34     transform: translate(-50%, -50%) rotate(30deg);
35   }
36   50%, #{100 - 150 / ($i+3)}% {
37     transform: translate(-50%, -50%) rotate(-15deg);
38   }
39 }
40

```

```

1 body {
2   margin: 0;
3   min-height: 100vh;
4   background-color: #ededed;
5 }
6
7 .square {
8   position: absolute;
9   top: 50%;
10  left: 50%;
11  background-color: #ededed;
12  -webkit-transform: translate(-50%, -50%);
13  | -ms-transform: translate(-50%, -50%);
14  | | | transform: translate(-50%, -50%);
15 }
16
17 .square:nth-child(2n+1) {
18   background-color: #232323;
19 }
20
21 .square:nth-child(14) {
22   width: 3vmin;
23   height: 3vmin;
24   -webkit-animation: rot1 infinite ease 1.5s;
25   | | | | animation: rot1 infinite ease 1.5s;
26 }
27
28 @-webkit-keyframes rot1 {
29   0%, 12.5%, 100% {
30     -webkit-transform: translate(-50%, -50%) rotate(30deg);
31     | | | | transform: translate(-50%, -50%) rotate(30deg);
32   }
33   50%, 62.5% {
34     -webkit-transform: translate(-50%, -50%) rotate(-15deg);
35     | | | | transform: translate(-50%, -50%) rotate(-15deg);
36   }
37 }
38
39
40

```

```

1 $size: 3vmin;
2 $n: 14;
3 $light: #ededed;
4 $dark: #232323;
5
6 body {
7   margin: 0;
8   min-height: 100vh;
9   background-color: $light;
10 }
11
12 .square {
13   position: absolute;
14   top: 50%;
15   left: 50%;
16   background-color: $light;
17   transform: translate(-50%, -50%);
18 }
19 .square:nth-child(2n+1) {
20   background-color: $dark;
21 }
22
23 @for $i from 1 through $n {
24   $c: $n + 1 - $i;
25
26   .square:nth-child(#{$c}) {
27     width: $size * $i;
28     height: $size * $i;
29     animation: rot#{$i} infinite ease #{1.5 * $n / $c}s;
30   }
31
32 @keyframes rot#{$i} {
33   0%, #{50 - 150 / ($i+3)}%, 100% {
34     transform: translate(-50%, -50%) rotate(30deg);
35   }
36   50%, #{100 - 150 / ($i+3)}% {
37     transform: translate(-50%, -50%) rotate(-15deg);
38   }
39 }
40

```

```

1 body {
2   margin: 0;
3   min-height: 100vh;
4   background-color: #ededed;
5 }
6
7 .square {
8   position: absolute;
9   top: 50%;
10  left: 50%;
11  background-color: #ededed;
12  -webkit-transform: translate(-50%, -50%);
13  | -ms-transform: translate(-50%, -50%);
14  | | | transform: translate(-50%, -50%);
15 }
16
17 .square:nth-child(2n+1) {
18   background-color: #232323;
19 }
20
21 .square:nth-child(14) {
22   width: 3vmin;
23   height: 3vmin;
24   -webkit-animation: rot1 infinite ease 1.5s;
25   | | | | animation: rot1 infinite ease 1.5s;
26 }
27
28 @-webkit-keyframes rot1 {
29   0%, 12.5%, 100% {
30     -webkit-transform: translate(-50%, -50%) rotate(30deg);
31     | | | | transform: translate(-50%, -50%) rotate(30deg);
32   }
33   50%, 62.5% {
34     -webkit-transform: translate(-50%, -50%) rotate(-15deg);
35     | | | | transform: translate(-50%, -50%) rotate(-15deg);
36   }
37 }
38
39
40

```

@function

@function

Собственные функции можно использовать в любом значении и контексте

- имеют доступ к глобальным переменным
- принимают параметры
- возвращают значение

@function

```
$z-indexes: (
  'modal': 5000,
  'dropdown': 4000,
  'default': 1,
  'below': -1,
);

@function z($layer) {
  @return map-get($z-indexes, $layer);
}

.modal {
  z-index: z('modal');
}

.dropdown {
  z-index: z('dropdown');
}
```

@function

```
// SCSS

$base-pixel: 16;

@function unit($value, $is-pixel: false) {

@if $is-pixel == true {
  $value: #{$value}px;
} @else {
  $value: #{$value/$base-pixel}rem;
}
@return $value;
}
```

@function

```
// Usage unit function
```

```
.box {  
  width: unit(48);  
  height: unit(24);  
}
```

```
/* CSS output */
```

```
.box {  
  width: 3rem;  
  height: 1.5rem;  
}
```

Итого

Итого

Плюсы

- Расширяют стандартные возможности языка

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку
- Облегчают модификацию проекта и навигацию в коде

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку
- Облегчают модификацию проекта и навигацию в коде
- Упрощают компонентный подход

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку
- Облегчают модификацию проекта и навигацию в коде
- Упрощают компонентный подход
- Контрибьютинг

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку
- Облегчают модификацию проекта и навигацию в коде
- Упрощают компонентный подход
- Контрибьютинг

Минусы

- Требуют компиляции (дополнительного ПО)

Итого

Плюсы

- Расширяют стандартные возможности языка
- Ускоряют разработку
- Облегчают модификацию проекта и навигацию в коде
- Упрощают компонентный подход
- Контрибьютинг

Минусы

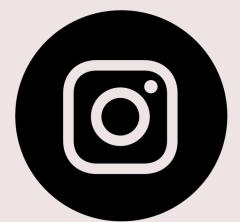
- Требуют компиляции (дополнительного ПО)
- Требуют знания языка и дисциплины

“

Спасибо за внимание!

Вопросы?

”

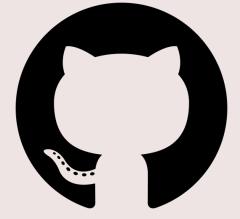


[instagram.com/@niktariy_dev](https://www.instagram.com/@niktariy_dev)

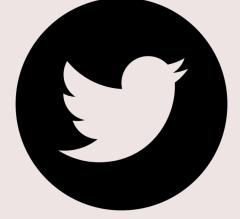
– Вероника Новикова



codepen.io/@niktariy



github.com/@niktariy



twitter.com/@niktariy



medium.com/@niktariy