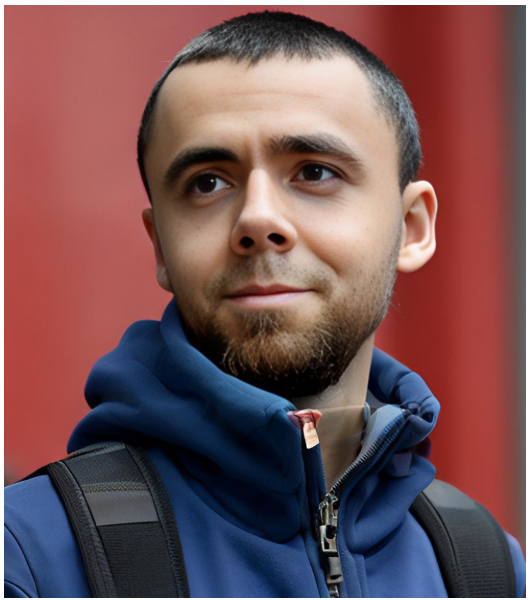
Atomic CSS Deep Dive

Валентин Ульянов

Обо мне



- В IT больше 8 лет
- Делаю бэкенд на Node.js и тулинг
- Разрабатываю open source проект
- Выступаю на конференциях и веду IT-сообщество в Питере

Я и Atomic CSS

- В теме с 2018

Я и Atomic CSS

- В теме с 2018
- Смотрел все инструменты, у которых >20 звезд на гитхабе

Я и Atomic CSS

- В теме с 2018
- Смотрел все инструменты, у которых >20 звезд на гитхабе
- 3 года карьеры много верстал

Я и Atomic CSS

- В теме с 2018
- Смотрел все инструменты, у которых >20 звезд на гитхабе
- 3 года карьеры много верстал
- В разработку своего инструмента вложил >1000 часов

Содержание

- База
- State of Atomic CSS
- Сравнение инструментов и технические делати
  - Как устроены утилиты: нейминг, синтаксис и значения
  - Конфигурирование
  - JIT engine
- Заключение

База

```html
<button class="D-ib Bgc-blue C#fff P10 Bgc-red_h Cs-p">
  I am a button
</button>
```

```css
.D-ib {
  display: inline-block;
}

.Bgc-blue {
  background-color: blue;
}

.C\#fff {
  color: #fff;
}

.P10 {
  padding: 10px;
}

.Bgc-red_h:hover {
  background-color: red;
}

.Cs-p {
  cursor: pointer;
}
```

Почему Atomic CSS

В сравнении с рукописным CSS

- **Тратим меньше мыслетоплива**: названия сущностей, структура каталогов

Почему Atomic CSS

В сравнении с рукописным CSS

- **Тратим меньше мыслетоплива**: названия сущностей, структура каталогов
- **Меньше CSS на клиенте**: стили перестают добавляться

Почему Atomic CSS

В сравнении с рукописным CSS

- **Тратим меньше мыслетоплива**: названия сущностей, структура каталогов
- **Меньше CSS на клиенте**: стили перестают добавляться
- **Быстрее пишем стили**: короткие классы, переключение файлов

Для тех, кто
плохо знает CSS

Нечитаемая
разметка

Сложно
поддерживать

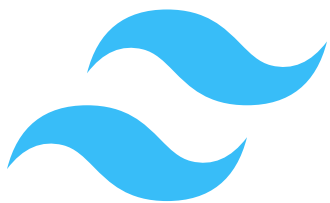
То же самое,
что инлайн-стили

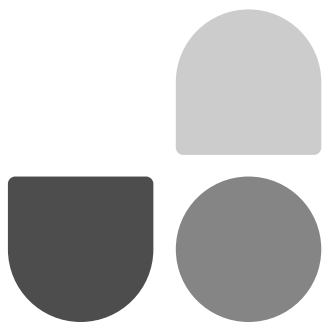
imgflip.com

State of Atomic CSS

Актуальные и популярные инструменты

1. Tailwindcss
2. UnoCSS
3. Atomizer





Актуальные проблемы

- 👎 Неконсистентный нейминг

Актуальные проблемы

- 👎 Неконсистентный нейминг
- 😢 Неудобно писать сложные утилиты

Актуальные проблемы

- 👎 Неконсистентный нейминг
- 😢 Неудобно писать сложные утилиты
- 🤡 Связь с рукописным CSS

Актуальные проблемы

- 👎 Неконсистентный нейминг
- 😢 Неудобно писать сложные утилиты
- 🤡 Связь с рукописным CSS
- 🥴 Неудобно расширять

Неконсистентный нейминг 👎

- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`

Неконсистентный нейминг 👎

- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`
- `tracking-wide`?

Неконсистентный нейминг 👎

- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`
- `tracking-wide` => `letter-spacing: 0.025em;`

Неконсистентный нейминг 👎

- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`
- `tracking-wide` => `letter-spacing: 0.025em;`
- `normal`?

Неконсистентный нейминг 👎

- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`
- `tracking-wide` => `letter-spacing: 0.025em;`
- `normal`: line-height, font-weight, letter-spacing?

Сложные утилиты 😢

`[@media(any-hover:hover){&:hover}]:opacity-100`

```
@media(any-hover:hover) {
  .\[\@media\(any-hover\:hover\)\{\&\:hover\}\]\:opacity-100:hover {
    opacity: 1
  }
}
```

Сложные утилиты 😢

`[&:not(:first-child)]:rounded-full`

```
.\[\&\:not\(\:first-child\)\]\:rounded-full:not(:first-child) {
  border-radius: 9999px
}
```

Сложные утилиты 😢

`supports-[margin:1svw]:ml-[1svw]`

```
@supports (margin: 1svw) {
  .supports-\[margin\:1svw\]\:ml-\[1svw\] {
    margin-left: 1svw
  }
}
```

Страшная тайна Atomic CSS 😱

> *В большинстве проектов, небольшую часть CSS вам придется написать руками!*

Связь с рукописным CSS 🤡

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .card {
    background-color: theme(colors.white);
    border-radius: 5px / theme(borderRadius.lg);
    padding: 1rem theme(spacing[2.5]);
  }
}
```

Связь с рукописным CSS 🤡

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .card {
    background-color: theme(colors.white);
    border-radius: 5px / theme(borderRadius.lg);
    padding: 1rem theme(spacing[2.5]);
  }
}
```

- Конфликты с CSS

Связь с рукописным CSS 🤡

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .card {
    background-color: theme(colors.white);
    border-radius: 5px / theme(borderRadius.lg);
    padding: 1rem theme(spacing[2.5]);
  }
}
```

- Конфликты с CSS
- Структура файлов / каталогов

Связь с рукописным CSS 🤡

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .card {
    background-color: theme(colors.white);
    border-radius: 5px / theme(borderRadius.lg);
    padding: 1rem theme(spacing[2.5]);
  }
}
```

- Конфликты с CSS
- Структура файлов / каталогов
- Получение значений утилит

Связь с рукописным CSS 🤡

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .card {
    background-color: theme(colors.white);
    border-radius: 5px / theme(borderRadius.lg);
    padding: 1rem theme(spacing[2.5]);
  }
}
```

- Конфликты с CSS
- Структура файлов / каталогов
- Получение значений утилит
- Нет фич препроцессоров*

*Да, часть можно подключить с PostCSS, и они не всем нужны

Связь с рукописным CSS 🤡

Windi Lang Draft

```
@var width = 1px;
@var baseColor = #c6538c;
@var borderDark = rgba(base-color, 0.88);
@var borderWidth = width + 1px;
@var prop = 'color';

.alert {
  border: ${borderWidth + 4px} solid ${borderDark};
  ${prop}: red;
}
```

Неудобно расширять 🥲

```
module.exports = {
  theme: {
    tabSize: {
      // map with values
    }
  },
  plugins: [
    plugin(function({ matchUtilities, theme }) {
      matchUtilities(
        {
          tab: (value) => ({
            tabSize: value
          }),
        },
        { values: theme('tabSize') }
      )
    })
```

Неудобно расширять 🥲

```
variants: [
  // hover:
  (matcher) => {
    if (!matcher.startsWith('hover:'))
      return matcher
    return {
      matcher: matcher.slice(6),
      selector: s => `${s}:hover`,
    }
  },
],
```

Актуальное решение 😎

> *Atomic CSS toolkit with **Sass** and ergonomics for creating styles of any complexity*


Mlut
ЭТО TAILWIND НА SASS?
imgflip.com

Не стоит хоронить Sass

Version History

☐ show deprecated versions

Version	Downloads (Last 7 Days)	Published
1.71.1	2 594 119	22 days ago
1.71.0	202 579	a month ago
1.70.0	1 182 948	2 months ago
1.69.7	533 801	2 months ago
1.69.6	36 478	2 months ago
1.69.5	1 120 535	5 months ago
1.69.4	128 428	5 months ago
1.69.3	70 281	5 months ago
1.69.2	11 901	5 months ago
1.69.1	13 071	5 months ago
1.69.0	30 883	5 months ago
1.68.0	156 663	6 months ago

Install

```
> npm i sass
```

Repository

github.com/sass/dart-sass

Homepage

github.com/sass/dart-sass

Weekly Downloads

13 783 143


ЩА БУИТ МЯСО))0)

Как устроены утилиты

```
xl_^:h_D-f_af
```

Нейминг

```
xl_^:h_D-f_af
```

Tailwind 😢

Opinionated названия, созвучные с CSS свойствами/значениями

Tailwind 😢

Opinionated названия, созвучные с CSS свойствами/значениями

- `justify-*`: content, items, self?

Tailwind 😢

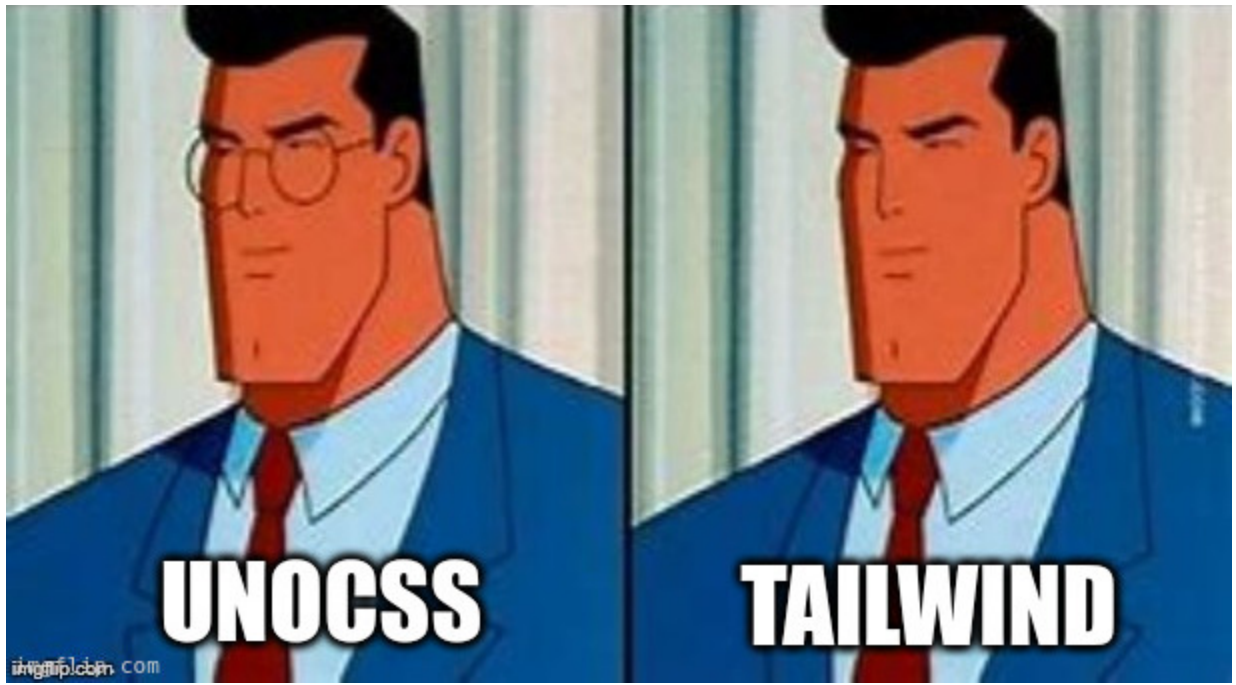
Opinionated названия, созвучные с CSS свойствами/значениями

- `justify-*`: content, items, self?
- `bg-none` - убрать весь background? **Нет**, только `background-image`

Tailwind 😢

Opinionated названия, созвучные с CSS свойствами/значениями

- `justify-*`: content, items, self?
- `bg-none` - убрать весь background? **Нет**, только `background-image`
- `flex` => `display: flex`, но `flex-auto` => `flex: 1 1 auto`


UNOCSS
TAILWIND

Unocss (Tachyons) 😭

- `br-0` => `border-right-width: 0`, но

  `br1` => `border-radius:.125rem`

Unocss (Tachyons) 😭

- `br-0` => `border-right-width: 0`, но

  `br1` => `border-radius:.125rem`
- `b` : bottom, border, `display: block` ? **Нет**, это `font-weight:bold` !

Unocss (Tachyons) 😭

- `br-0` => `border-right-width: 0`, но

  `br1` => `border-radius:.125rem`
- `b` : bottom, border, `display: block` ? **Нет**, это `font-weight:bold` !
- `normal` : line-height, font-weight, letter-spacing?

Atomizer 🤦

Сокращения Emmet + додуманные

Atomizer 🤦

Сокращения Emmet + додуманные

- `Js(c)` => `justify-self: center`

Atomizer 🤦

Сокращения Emmet + додуманные

- `Js(c)` => `justify-self: center`
- `Bg(n)` => `background: none`

Atomizer 🤦

Сокращения Emmet + додуманные

- `Js(c)` => `justify-self: center`
- `Bg(n)` => `background: none`
- `Bgbm(c)` => `background-blend-mode: color`

mlut 😎

Единный алгоритм для всех сокращений

mlut 😎

Единный алгоритм для всех сокращений

- `Js-c` => `justify-self: center`

mlut 😎

Единный алгоритм для всех сокращений

- `Js-c` => `justify-self: center`
- `Bdr` => `border-right: 1px solid`

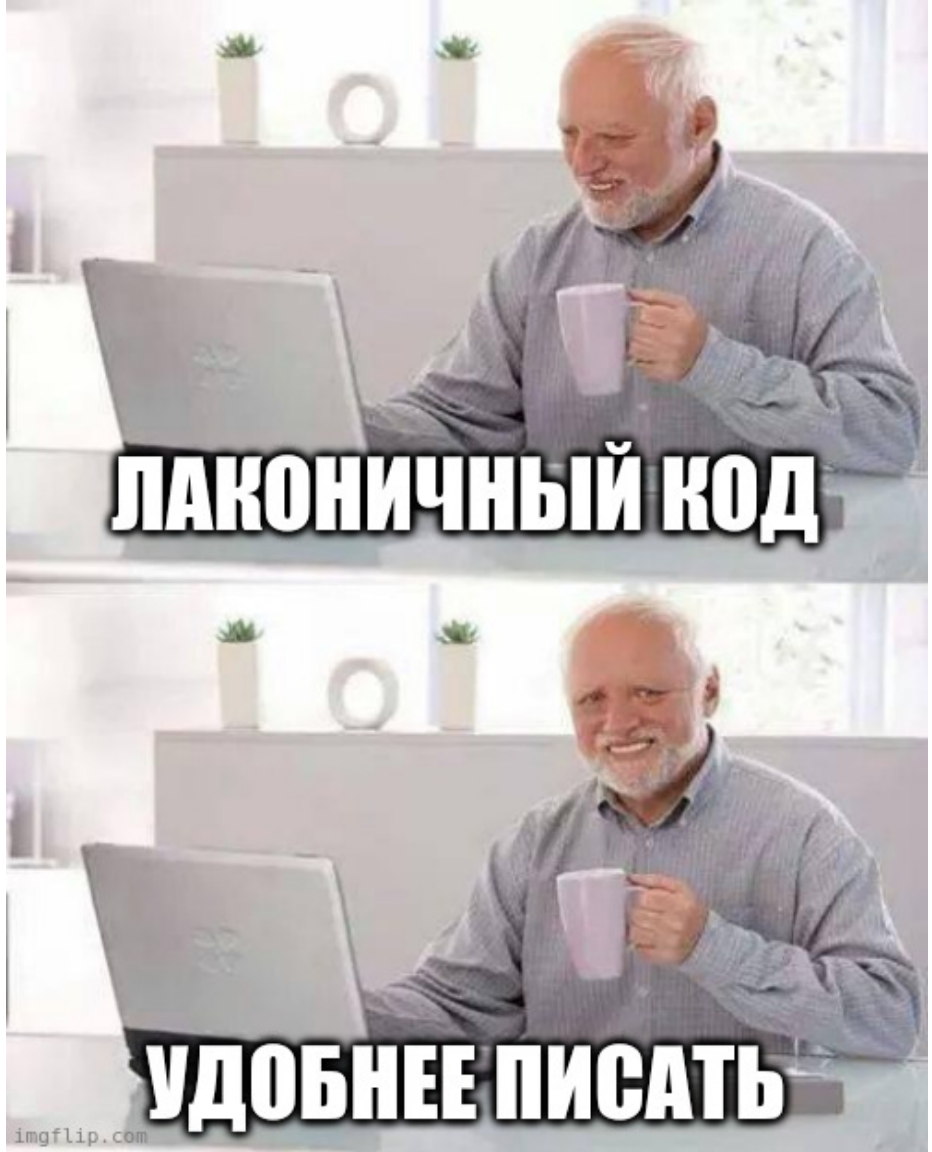
mlut 😎

Единный алгоритм для всех сокращений

- `Js-c` => `justify-self: center`
- `Bdr` => `border-right: 1px solid`
- `Bdrd1` => `border-radius: 1px`

Почему сокращения?

- \+ Лаконичный код
- \+ Удобнее писать
- \+ Чуть меньше вес кода


ЛАКОНИЧНЫЙ КОД
УДОБНЕЕ ПИСАТЬ
imgflip.com

Почему сокращения?

+ Лаконичный код
+ Удобнее писать
+ Чуть меньше вес кода

− Есть порог входа
− Подойдут не всем

Сокращения везде

- `const`, `int`, `char`
- `cd`, `pwd`, `ls`
- `Ldar`, `Star`, `SubSmi`

Зачем алгоритм сокращений?

– Избежать коллизий с новыми свойствами/значениями в CSS

Зачем алгоритм сокращений?

- Избежать коллизий с новыми свойствами/значениями в CSS
- Возможность выводить свойства "в голове", а не запоминать их

Зачем алгоритм сокращений?

- Избежать коллизий с новыми свойствами/значениями в CSS
- Возможность выводить свойства "в голове", а не запоминать их

Сокращения Emmet неплохи, но в них нет четких правил

(confirmed by @sergeche)

Как это было

- mdn-data

Как это было

- mdn-data
- Спеки, черновики спек

Как это было

- mdn-data
- Спеки, черновики спек
- Chrome platform status

Как это было

- mdn-data
- Спеки, черновики спек
- Chrome platform status
- Недели размышлений

data / css / properties.json

Code Blame 10095 lines (10095 loc) · 285 KB

```
  },
  "border-collapse": {
    "syntax": "collapse | separate",
    "media": "visual",
    "inherited": true,
    "animationType": "discrete",
    "percentages": "no",
    "groups": [
      "CSS Table"
    ],
    "initial": "separate",
    "appliesto": "tableElements",
    "computed": "asSpecified",
    "order": "uniqueOrder",
    "status": "standard",
    "mdn_url": "https://developer.mozilla.org/docs/Web/CSS/border-collapse"
  },
  "border-color": {
    "syntax": "<color>{1,4}",
    "media": "visual",
    "inherited": false,
    "animationType": [
      "border-bottom-color",
      "border-left-color",
      "border-right-color",
      "border-top-color"
    ],
    "percentages": "no",
    "groups": [
      "CSS Backgrounds and Borders"
    ],
    "initial": [
      "border-top-color",
      "border-right-color",
      "border-bottom-color",
      "border-left-color"
    ],
    "appliesto": "allElements",
    "computed": [
      "border-bottom-color",
      "border-left-color",
```

W3C Candidate Recommendation Draft

TABLE OF CONTENTS

CSS Text Module Level 3

W3C Candidate Recommendation Draft, 3 September 2023

▾ **More details about this document**

This version:
https://www.w3.org/TR/2023/CRD-css-text-3-20230903/

Latest published version:
https://www.w3.org/TR/css-text-3/

Editor's Draft:
https://drafts.csswg.org/css-text-3/

Previous Versions:

W3C Candidate Recommendation Draft

TABLE OF CONTENTS

CSS Backgrounds and Borders Module Level 3

W3C Candidate Recommendation Draft, 11 March 2024

▾ **More details about this document**

This version:
https://www.w3.org/TR/2024/CRD-css-backgrounds-3-20240311/

Latest published version:
https://www.w3.org/TR/css-backgrounds-3/

Editor's Draft:

W3C Editor's Draft

TABLE OF CONTENTS

CSS Text Decoration Module Level 3

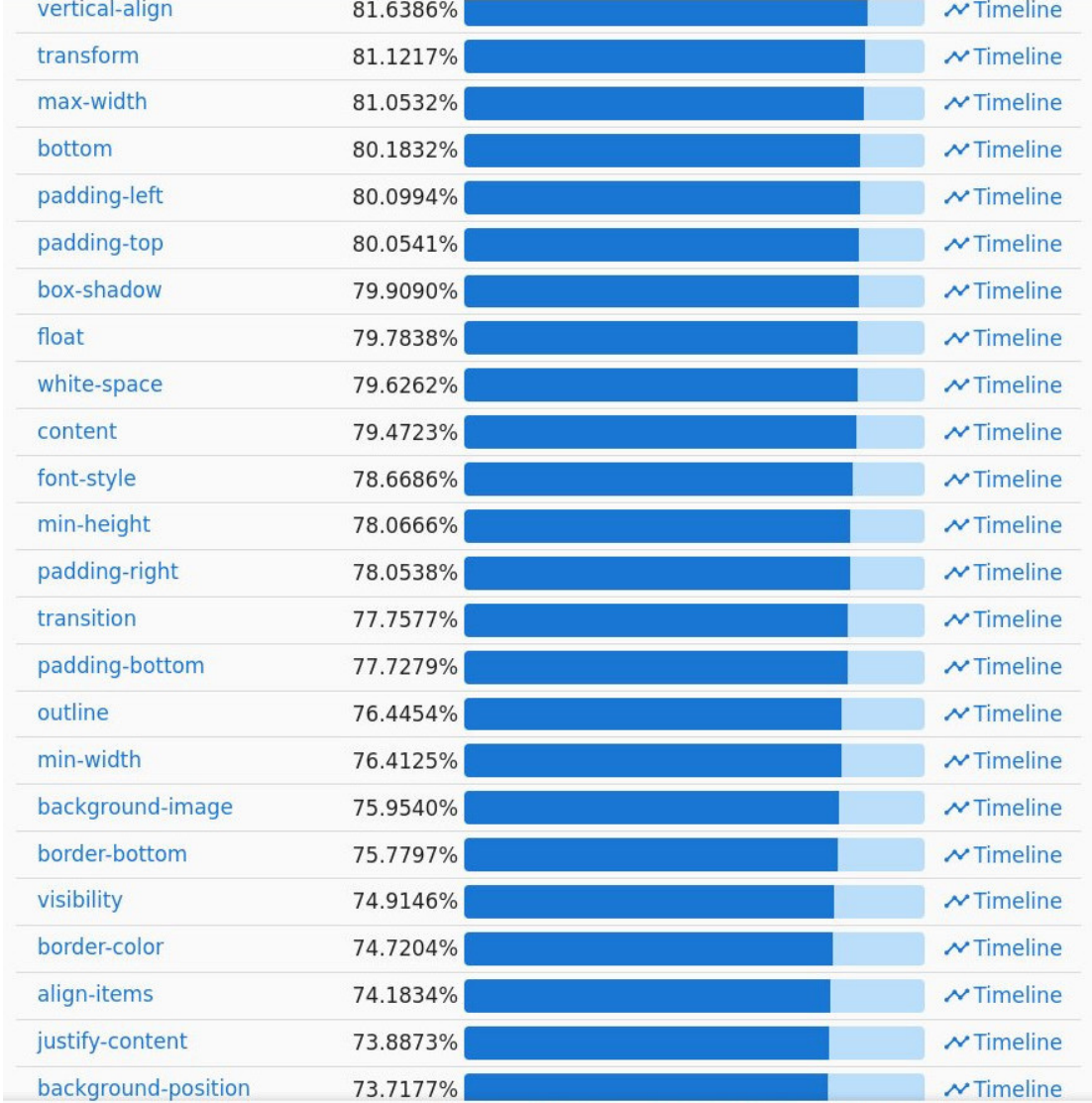
Editor's Draft, 24 January 2024

▾ **More details about this document**

This version:
https://drafts.csswg.org/css-text-decor-3/

Latest published version:
https://www.w3.org/TR/css-text-decor-3/

Previous Versions:
https://www.w3.org/TR/2018/CR-css-text-decor-3-20180703/


vertical-align 81.6386% Timeline
transform 81.1217% Timeline
max-width 81.0532% Timeline
bottom 80.1832% Timeline
padding-left 80.0994% Timeline
padding-top 80.0541% Timeline
box-shadow 79.9090% Timeline
float 79.7838% Timeline
white-space 79.6262% Timeline
content 79.4723% Timeline
font-style 78.6686% Timeline
min-height 78.0666% Timeline
padding-right 78.0538% Timeline
transition 77.7577% Timeline
padding-bottom 77.7279% Timeline
outline 76.4454% Timeline
min-width 76.4125% Timeline
background-image 75.9540% Timeline
border-bottom 75.7797% Timeline
visibility 74.9146% Timeline
border-color 74.7204% Timeline
align-items 74.1834% Timeline
justify-content 73.8873% Timeline
background-position 73.7177% Timeline
Help Discuss File an issue Privacy

font-variation-settings	0.2	Fnvs
gap	3.0	Gap
grid	0.06	G
grid-template	0.4	Gt
grid-template-columns	15.3	Gtc
grid-template-rows	7	Gtr
grid-template-areas	2.9	Gta
grid-column	5.2	Gc
grid-column-start	2.3	Gcs
grid-column-end	2.1	Gce
grid-area	4	Ga
grid-auto-flow	3.3	Gatf
grid-auto-rows	2.2	Gatr
grid-auto-columns	0.9	Gatc
grid-row	2.6	Gr
grid-row-start	1.6	Grs
grid-row-end	1.3	Gre
-gradient-linear		-Gdl
-gradient-repeating-linear		-Gdrl
margin	92	
margin-top	90	
margin-left	84	
margin-bottom	83	
margin-right	82	

Общий алгоритм сокращений кратко

1. Находим свойства, которые начинаются с одинаковой буквы

Общий алгоритм сокращений кратко

1. Находим свойства, которые начинаются с одинаковой буквы
2. Составляем их рейтинг

Общий алгоритм сокращений кратко

1. Находим свойства, которые начинаются с одинаковой буквы
2. Составляем их рейтинг
3. Выделяем группы

Общий алгоритм сокращений кратко

1. Находим свойства, которые начинаются с одинаковой буквы
2. Составляем их рейтинг
3. Выделяем группы
4. Составляем сокращения внутри групп

I. Алгоритм сокращения одной сущности

Название сокращаем до первой буквы свойства/значения

`color` => `C`

II. Алгоритм сокращения одной сущности

Если название из N слов, то берется первая буква из каждого слова

`color-adjust` => `Ca`

III. Алгоритм сокращения одной сущности

Если два названия имеют одну и ту же начальную букву, то в следующем названии, при сортировке их по рейтингу, добавляется буква

1. `color` => `C`
2. `cursor` => `Cs`

IV. Алгоритм сокращения одной сущности

Если название из N слов, то буква добавляется в соответствующем по порядку слове

1. `color` => `C`
2. `cursor` => `Cs`
3. `color-scheme` => `Csc`

I. Порядок добавления буквы

Согласная следующего слога

`cursor` => `Cs`

I. Порядок добавления буквы

Согласная следующего слога

`cursor` => `Cs`

Если следующий слог начинается на гласную, то берется ближайшая

предыдущая согласная от нее

II. Порядок добавления буквы

Следующая согласная

1. `content` => `Ct`
2. `contain` => `Cn`

III. Порядок добавления буквы

Следующая гласная (без перескока через согласную)

1. `content` => `Ct`
2. `counter-increment` => `Coi`

Ps

Ps => position

Fnw

Fnw => font-weight

Tf

Tf => transform

Flg

Flg => flex-grow

Слабые места 😦

- Популярность свойств меняется

Слабые места 😦

- Популярность свойств меняется
- Редкие длинные свойства может быть сложно вспомнить

Слабые места 😦

- Популярность свойств меняется
- Редкие длинные свойства может быть сложно вспомнить
- Возможны спорные ситуации, по мере развития CSS

Синтаксис

```
xl_^:h_D-f_af
```

Tailwind 😢

- Нет хотя бы подобия спецификации

Tailwind 😢

- Нет хотя бы подобия спецификации
- Ad-hoc с arbitrary частями

Tailwind: утилита и значение 😢

- `util-value`

Tailwind: утилита и значение 😢

- `util-value`
- `-util-2` - отрицательные значения

Tailwind: утилита и значение 😢

- `util-value`
- `-util-2` - отрицательные значения
- `util-[42px]` - произвольные значения

Tailwind: утилита и значение 😢

- `util-value`
- `-util-2` - отрицательные значения
- `util-[42px]` - произвольные значения
- `[css-prop:value]` - произвольное CSS свойство и значение

Tailwind: variants 😢

- `variant:util-value` - селекторы и некоторые at-rules

Tailwind: variants 😢

- `variant:util-value` - селекторы и некоторые at-rules
- `group/name:util-value` - группы с именами

Tailwind: variants 😢

- `variant:util-value` - селекторы и некоторые at-rules
- `group/name:util-value` - группы с именами
- `@md:util-value` - container queries

Tailwind: arbitrary variants 😢

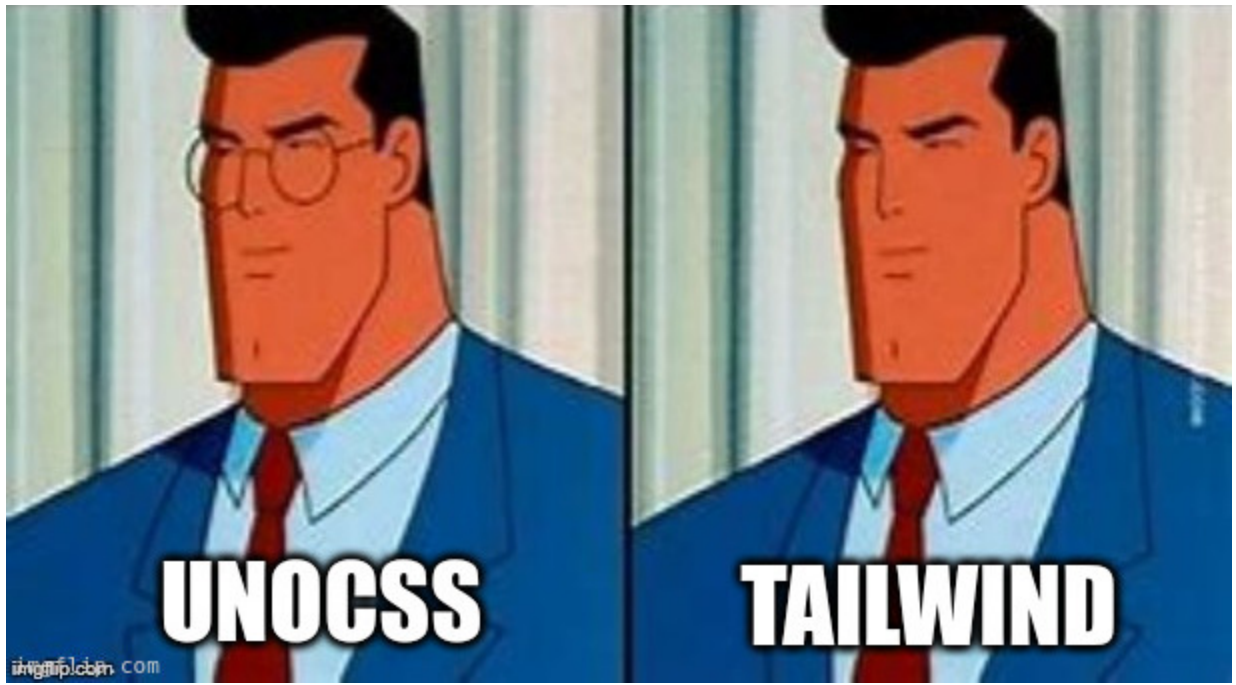
- `variant-[.class]:util` - произвольные значения variant

Tailwind: arbitrary variants 😢

- `variant-[.class]:util` - произвольные значения variant
- `[&:nth-child(3)]:util` - произвольный variant

Tailwind: arbitrary variants 😢

- `variant-[.class]:util` - произвольные значения variant
- `[&:nth-child(3)]:util` - произвольный variant
- `@[17.5rem]:util` - container queries


UNOCSS
TAILWIND

Atomizer 🤦

- Есть спецификация!
- Покрывает мало фич CSS

Atomizer 🤦

```
[<context>[:<pseudo-class>]<combinator>]<Style>
[(<value>,<value>?,...)][<!>][:<pseudo-class>]
[::<pseudo-element>][--<breakpoint_identifier>]
```

mlut: components syntax 😎

```
@:ah_01_h
```

mlut: components syntax 😎

`@:ah_01_h` =>

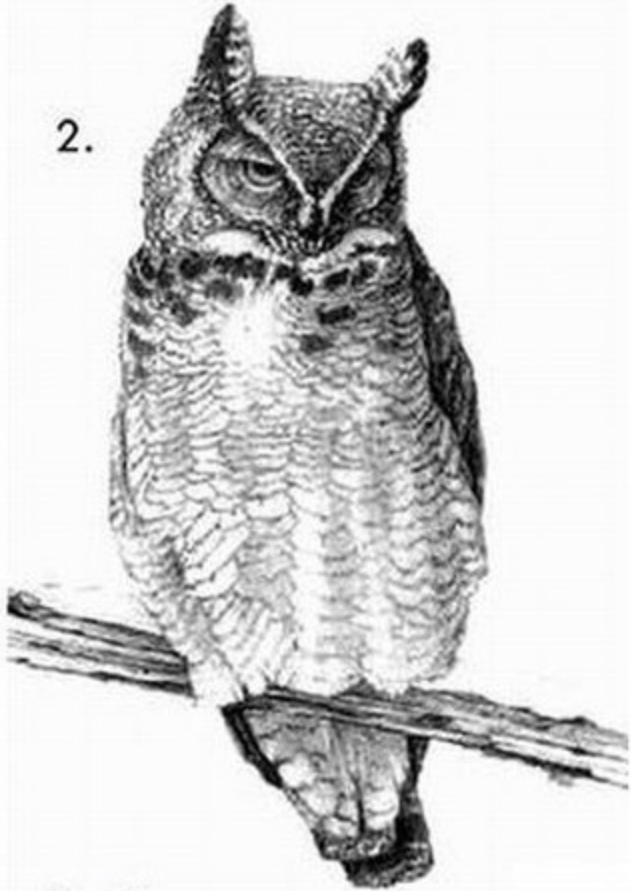
```
@media (any-hover) {
  .\@\:ah_01_h:hover {
    opacity: 1
  }
}
```

Как нарисовать сову

1.

1. Рисуем кружочки

2.



2. Рисуем остаток совы

Зачем проектировать синтаксис?

Основная цель - концептуальная близость с CSS для органичного развития вместе с ним

Minimum opinions - maximum standards!

Зачем проектировать синтаксис?

- Меньше учить "фантазийных" сущностей

Зачем проектировать синтаксис?

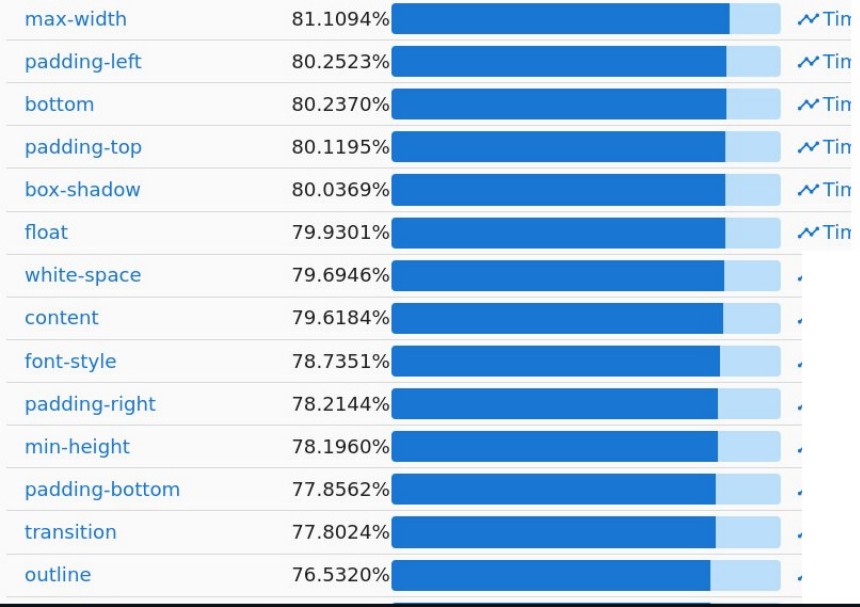
- Меньше учить "фантазийных" сущностей
- Избежать (минимизировать) конфликты с CSS

Зачем проектировать синтаксис?

- Меньше учить "фантазийных" сущностей
- Избежать (минимизировать) конфликты с CSS
- Удобство написания

Зачем проектировать синтаксис?

- Меньше учить "фантазийных" сущностей
- Избежать (минимизировать) конфликты с CSS
- Удобство написания
- Получить высокую выразительность для реализации наибольшего количества фич CSS



CSS Pseudo-Elements Module Level 4

Editor's Draft, 13 September 2023

▼ More details about this document

This version:
https://drafts.csswg.org/css-pseudo-4/

CSS Values and Units Module 4

W3C Working Draft, 12 March 2024

ment

Media Queries Level 5

Editor's Draft, 14 March 2024

▼ More details about this document

This version:
https://drafts.csswg.org/mediaqueries-5/

Latest published version:
https://www.w3.org/TR/mediaqueries-5/

Previous Versions:
https://www.w3.org/TR/2021/WD-mediaqueries

https://www.w3.org/TR/2020/WD-mediaqueries

w3c / csswg-drafts Public

Code | Issues 3k | Pull requests 73 | Actions | Projects 17 | Security | Insights

is:issue is:open

Labels 252

3,046 Open ✓ 4,962 Closed Author Label Projects Mile

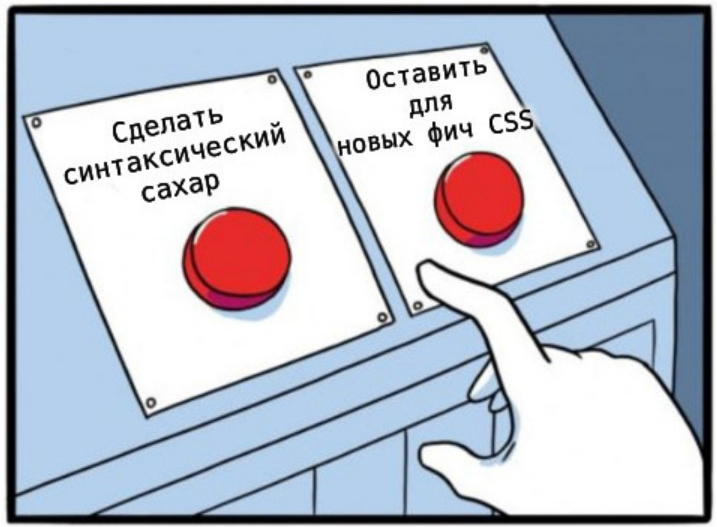
[mediaqueries-5] Privacy concerns about the prefers-reduced-data media feature
mediaqueries-5
#10076 opened 10 hours ago by frivoal

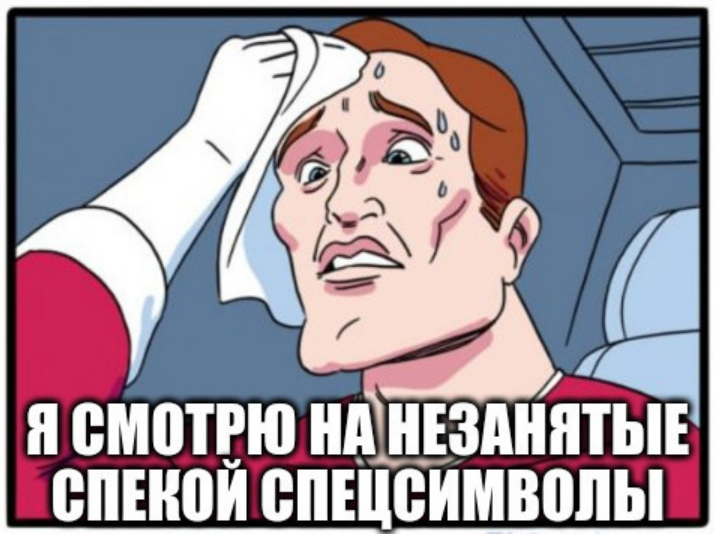
[css-images-4] Inconsistent but non-normative advice regarding image metadata and streamability/progressive display css-images-4
#10075 opened 10 hours ago by svgeesus

data / css / properties.json

Code Blame 10095 lines (10095 loc) · 285 KB


```
6351       "appliesto": "listItems",
6352       "computed": "asSpecified",
6353       "order": "uniqueOrder",
6354       "status": "standard",
6355       "mdn_url": "https://developer.mozilla.org/docs/Web/CSS
6356     },
6357     "list-style-type": {
6358       "syntax": "<counter-style> | <string> | none",
6359       "media": "visual",
6360       "inherited": true,
6361       "animationType": "discrete",
6362       "percentages": "no",
6363       "groups": [
6364         "CSS Lists and Counters"
6365       ],
6366       "initial": "disc",
6367       "appliesto": "listItems",
6368       "computed": "asSpecified",
6369       "order": "uniqueOrder",
6370       "status": "standard",
6371       "mdn_url": "https://developer.mozilla.org/docs/Web/CSS
6372     },
6373     "margin": {
6374       "syntax": "[ <length> | <percentage> | auto ]{1,4}",
6375       "media": "visual",
```


Сделать синтаксический сахар
Оставить для новых фич CSS


Я СМОТРЮ НА НЕЗАНЯТЫЕ СПЕКОЙ СПЕЦСИМВОЛЫ

Utility components syntax

Синтаксис, в котором утилита разделяется на *компоненты*, каждый из которых, соответствует части CSS-правила

Utility components syntax

```
@:ah_01_h
```

Utility components syntax

`@:ah_01_h` =>

```
01. @media (any-hover) {
02.
03.
04.
05. }
```

Utility components syntax

`@:ah_01_h` =>

```
@media (any-hover) {
   .\@\:ah_01_h {

   }
}
```

Utility components syntax

`@:ah_01_h` =>

```
01. @media (any-hover) {
02.   .\@\:ah_01_h {
03.     opacity: 1
04.   }
05. }
```

Utility components syntax

`@:ah_01_h` =>

```
01. @media (any-hover) {
02.   .\@\:ah_01_h:hover {
03.     opacity: 1
04.   }
05. }
```

1. **CSS at-rule:** брейкпоинты, @supports, etc

1. **CSS at-rule:** брейкпоинты, @supports, etc
2. **pre-states** - часть селектора перед классом утилиты

1. **CSS at-rule:** брейкпоинты, @supports, etc
2. **pre-states** - часть селектора перед классом утилиты
3. **Имя**
4. **Значение**

1. **CSS at-rule:** брейкпоинты, @supports, etc
2. **pre-states** - часть селектора перед классом утилиты
3. **Имя**
4. **Значение**
5. **post-states** - часть селектора после класса утилиты

Конвертация в mlut

Конвертация - превращение сокращения из названия класса в реальную CSS-сущность

Конвертация в mlut

Конвертация - превращение сокращения из названия класса в реальную CSS-сущность

- Значение
- Часть селектора
- Часть at-rule: название, CSS-свойство внутри, etc

Селекторы в CSS

- **Simple selector** - 1 условие: `.class`, `#id`, `element`

Селекторы в CSS

- **Simple selector** - 1 условие: `.class`, `#id`, `element`
- **(Pseudo-)Compound selector** - N простых без комбинаторов:

  `.class[attr]`, `element.class`

Селекторы в CSS

- **Simple selector** - 1 условие: `.class`, `#id`, `element`
- **(Pseudo-)Compound selector** - N простых без комбинаторов:

  `.class[attr]`, `element.class`
- **Complex selector** - N simple/compound с комбинаторами:

  `.class:hover + .item`

Селекторы в CSS

- **Simple selector** - 1 условие: `.class`, `#id`, `element`
- **(Pseudo-)Compound selector** - N простых без комбинаторов:

  `.class[attr]`, `element.class`
- **Complex selector** - N simple/compound с комбинаторами:

  `.class:hover + .item`
- **Selector list** - comma-separated список из simple/compound/complex:

  `.class, .item + .item, a.active`

States в mlut

States - это упрощенный *selector list*

```
xl_^:h_D-f_af
```

States в mlut

States - это упрощенный *selector list*

- `:` - объединение стейтов
- `,` - разделение на список
- `<empty>:` - пробел

`Bgc-red_h,f`

`Bgc-red_h,f` =>

```
.Bgc-red_h\,f {
  background-color: red
}
```

`Bgc-red_h,f` =>

```
.Bgc-red_h\,f:hover,
.Bgc-red_h\,f:focus {
  background-color: red
}
```

```
^:lc:>_D-f
```

`^:lc:>_D-f` =>

```
.-Ctx:last-child > .\^one\:lc\>_D-f {

}
```

`^:lc:>_D-f` =>

```
.-Ctx:last-child > .\^one\:lc\>_D-f {
  display: flex
}
```

Conditional at-rules в CSS

- **Conditions** - операторы, скобки, features / queries:

```
(hover) and (min-width: 20rem)
```

Conditional at-rules в CSS

- **Conditions** - операторы, скобки, features / queries:

  ```
  (hover) and (min-width: 20rem)
  ```

- **Операторы** - логические: `and`, `or`, `not`

Conditional at-rules в CSS

- **Conditions** - операторы, скобки, features / queries:

  `(hover) and (min-width: 20rem)`

- **Операторы** - логические: `and`, `or`, `not`
- **Features** - выражения, функции, etc:

  `(pointer: fine)`, `style(color: green)`

```
<supports-condition> = not <supports-in-parens>
                     | <supports-in-parens> [ and <supports-in-parens> ]*
                     | <supports-in-parens> [ or <supports-in-parens> ]*
<supports-in-parens> = ( <supports-condition> ) | <supports-feature> | <general-enclosed>
<supports-feature> = <supports-decl>
<supports-decl> = ( <declaration> )
```

Conditional at-rules в CSS

- Состав очень разный

Conditional at-rules в CSS

- Состав очень разный
- Можно строить сложные выражения, используя операторы

Conditional at-rules в CSS

- Состав очень разный
- Можно строить сложные выражения, используя операторы
- Можно вкладывать друг в друга

```
<media-feature> = ( [ <mf-plain> | <mf-boolean> | <mf-range> ] )
<mf-plain> = <mf-name> : <mf-value>
<mf-boolean> = <mf-name>
<mf-range> = <mf-name> <mf-comparison> <mf-value>
           | <mf-value> <mf-comparison> <mf-name>
           | <mf-value> <mf-lt> <mf-name> <mf-lt> <mf-value>
           | <mf-value> <mf-gt> <mf-name> <mf-gt> <mf-value>
<mf-name> = <ident>
<mf-value> = <number> | <dimension> | <ident> | <ratio>
<mf-lt> = '<' '='?
<mf-gt> = '>' '='?
<mf-eq> = '='
<mf-comparison> = <mf-lt> | <mf-gt> | <mf-eq>
```

At-rules в mlut

- Breakpoints: `lg`, `sm:md` - отдельный подсинтаксис
- Rules: `@media`, `@supports`, etc

```
xl_^:h_D-f_af
```

At-rules в mlut

- Breakpoints: `lg`, `sm:md` - отдельный подсинтаксис
- Rules: `@media`, `@supports`, etc

Для составления сложных выражений:

- `:` => `and`
- `,` => `,` (or)

Rules

У каждого правила свое:

- Сокращение: `m`, `s`, `c`

Rules

У каждого правила свое:

- Сокращение: `m`, `s`, `c`
- Конвертер - превращает сокращения в цепочку CSS-выражений

Rules

У каждого правила свое:

- Сокращение: `m`, `s`, `c`
- Конвертер - превращает сокращения в цепочку CSS-выражений
- Кастомные значения - алиасы для часто используемых цепочек

Rules

У каждого правила свое:

- Сокращение: `m`, `s`, `c`
- Конвертер - превращает сокращения в цепочку CSS-выражений
- Кастомные значения - алиасы для часто используемых цепочек

Синтаксис закрывает *весь класс* фич

```
@:dm-s:h>=20r_Mxw35r
```

`@:dm-s:h>=20r_Mxw35r` =>

```
@media (display-mode: standalone) {


}
```

`@:dm-s:h>=20r_Mxw35r` =>

```
01. @media (display-mode: standalone) and (min-height: 20rem) {
02.
03.
04.
05. }
```

`@:dm-s:h>=20r_Mxw35r` =>

```
@media (display-mode: standalone) and (min-height: 20rem) {
  .\@\:dm-s\:h\>\=20r_Mxw35r {

  }
}
```

`@:dm-s:h>=20r_Mxw35r` =>

```
@media (display-mode: standalone) and (min-height: 20rem) {
  .\@\:dm-s\:h\>\=20r_Mxw35r {
    max-width: 35rem
  }
}
```

At-rules можно комбинировать!

```
@s:apcr4/3@:dm_Mxw40u
```

At-rules можно комбинировать!

`@s:apcr4/3@:dm_Mxw40u` =>

```
01. @supports (aspect-ratio: 4/3) {
02.
03.
04.
05.
06.
07. }
```

At-rules можно комбинировать!

`@s:apcr4/3` `@:dm` `_Mxw40u` =>

```
@supports (aspect-ratio: 4/3) {
  @media (display-mode: fullscreen) {


  }
}
```

At-rules можно комбинировать!

`@s:apcr4/3@:dm_Mxw40u` =>

```
@supports (aspect-ratio: 4/3) {
  @media (display-mode: fullscreen) {
    .\@s\:apcr4\/3\@\:dm_Mxw40u {

    }
  }
}
```

At-rules можно комбинировать!

`@s:apcr4/3@:dm_Mxw40u` =>

```
01. @supports (aspect-ratio: 4/3) {
02.   @media (display-mode: fullscreen) {
03.     .\@s\:apcr4\/3\@\:dm_Mxw40u {
04.       max-width: 10rem
05.     }
06.   }
07. }
```

Слабые места 😦

- Нельзя (пока) написать arbitrary псевдоселектор:

  `D-f_:pseudo` => `.D-f pseudo {...}`

Слабые места 😦

- Нельзя (пока) написать arbitrary псевдоселектор:

  `D-f_:pseudo` => `.D-f pseudo {...}`

- Возможны конфликты пользовательских алиасов (`-myQuery`) с CSS custom media или cutsom selector

Конвертация значений

xl_^:h_D-f_af

Конвертация значений

Превращение сокращения из названия класса в реальное CSS-значение

- `ml-1` => `margin-left: 0.5rem`
- `D-f` => `display: flex;`

Tailwind 😢

- Подстановка значения из словаря в теме (theme)

Tailwind 😢

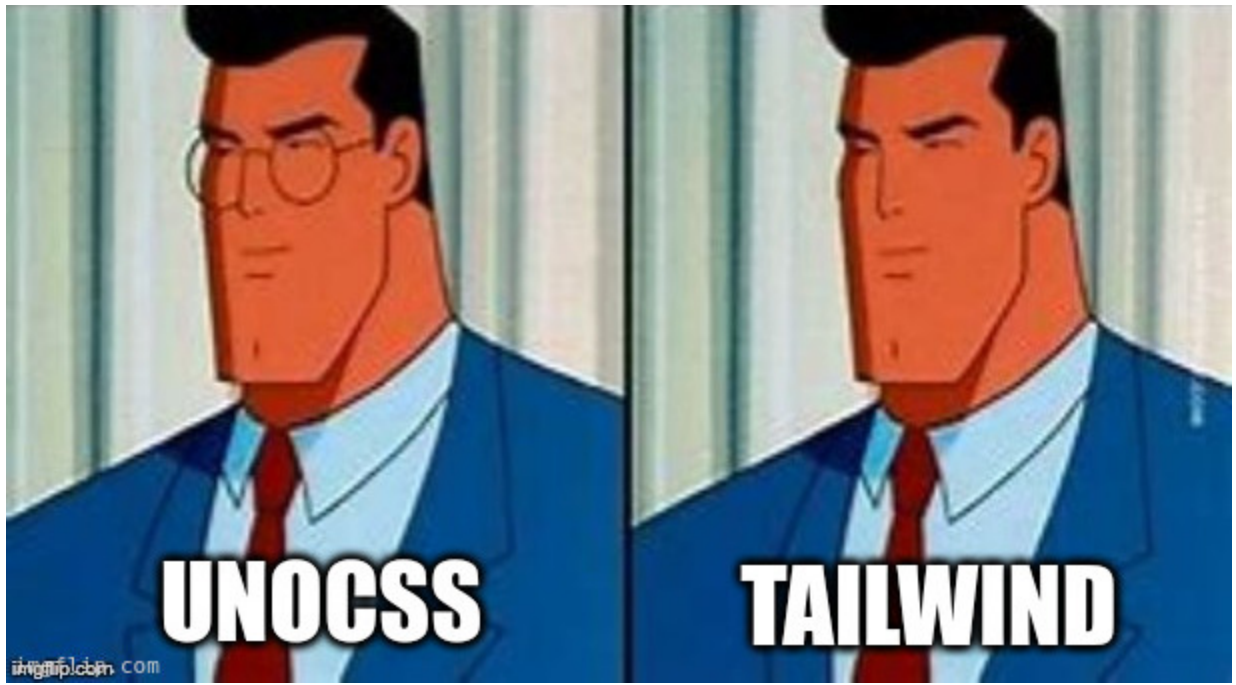
- Подстановка значения из словаря в теме (theme)
- Прозрачность цветов: `bg-sky-500/75`

Tailwind 😢

- Подстановка значения из словаря в теме (theme)
- Прозрачность цветов: `bg-sky-500/75`
- Императивная конвертация, при добавлении утилиты через плагин

Tailwind 😢

- Подстановка значения из словаря в теме (theme)
- Прозрачность цветов: `bg-sky-500/75`
- Императивная конвертация, при добавлении утилиты через плагин
- Части arbitrary значений: custom properties, пробелы, etc


UNOCSS
TAILWIND

Atomizer 🤦

- Ключевые слова + RTL by design

Atomizer 🤦

- Ключевые слова + RTL by design
- Прозрачность цветов: `C(#fff.5)`

Atomizer 🤦

- Ключевые слова + RTL by design
- Прозрачность цветов: `C(#fff.5)`
- Custom properties

Atomizer 🤦

- Ключевые слова + RTL by design
- Прозрачность цветов: `C(#fff.5)`
- Custom properties
- Множественные значения: `Bgp(20px,50px)`

Atomizer 🤦

- Ключевые слова + RTL by design
- Прозрачность цветов: `C(#fff.5)`
- Custom properties
- Множественные значения: `Bgp(20px,50px)`
- Подстановка пользовательских значения из конфига

mlut 😎

*Система конвертации для **почти произвольных** значений*

mlut 😎

*Система конвертации для **почти произвольных** значений*

- `Ml-1/7` => `margin-left: -14.3%`

mlut 😎

*Система конвертации для **почти произвольных** значений*

- `Ml-1/7` => `margin-left: -14.3%`
- `Bdrd1r;2/5p` => `border-radius: 1rem 2px / 5%`

Зачем система конвертации?

- Значения свойств CSS - это сложно

Зачем система конвертации?

- Значения свойств CSS - это сложно
- Оставаться ближе к платформе

Зачем система конвертации?

- Значения свойств CSS - это сложно
- Оставаться ближе к платформе
- Чтобы было удобно

Значения в CSS

- Value Definition Syntax
- Типы данных
- Единицы измерения
- Функции

W3C Working Draft

TABLE OF CONTENTS

CSS Values and Units 4

W3C Working Draft, 12 March 2024

▾ **More details about this document**

This version:

https://www.w3.org/TR/2024/WD-css-values-4-202

CSS Definition Syntax

Роман Дворнов

Москва, ноябрь 2019

```
3255      "border-left": {
3256        "syntax": "<line-width> || <line-style> || <color>",
```

```
3255      "border-left": {
3256        "syntax": "<line-width> || <line-style> || <color>",
```

```
4821      "font-family": {
4822        "syntax": "[ <family-name> | <generic-family> ]#",
```

```
3255      "border-left": {
3256        "syntax": "<line-width> || <line-style> || <color>",
```

```
4821      "font-family": {
4822        "syntax": "[ <family-name> | <generic-family> ]#",
```

```
5529      "grid-column": {
5530        "syntax": "<grid-line> [ / <grid-line> ]?",
```

data / css / syntaxes.json

Code Blame 881 lines (881 loc) · 29.7 KB

```
434     "line-name-list": {
435       "syntax": "[ <line-names> | <name-repeat> ]+"
436     },
437     "line-style": {
438       "syntax": "none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset"
439     },
440     "line-width": {
441       "syntax": "<length> | thin | medium | thick"
442     },
```

Основные понятия конвертации

- **Конвертер** - функция, которая преобразует значение из сокращенного класса в реальное CSS значение

Основные понятия конвертации

- **Конвертер** - функция, которая преобразует значение из сокращенного класса в реальное CSS значение
- **Трансформер** - функция, которая может еще как-то изменить готовое CSS-значение. Так же указывается в опциях

Основные понятия конвертации

Тип конвертации - список конвертеров, которые применяются к значению утилиты. Есть у каждой утилиты и если не указан явно в опциях утилиты, то применяется дефолтный

Опции утилиты

```
'Apcr': (
  'properties': aspect-ratio,
  'conversion': 'num-length', // тип конвертации
),
```

Общий конфиг для утилит

```
'conversion-types': (
  //...
  'num-length': ('num-length', 'global-kw', 'cust-prop')
                // ^цепочка конвертеров
),
```

Общая схема работы конвертации

1. Полное значение утилиты разбивается (по пробелам или разделителю) на простые значения

Общая схема работы конвертации

1. Полное значение утилиты разбивается (по пробелам или разделителю) на простые значения
2. Каждое простое значение проходит по цепочки конвертеров до тех пор, пока 1 из них не сработает

Общая схема работы конвертации

1. Полное значение утилиты разбивается (по пробелам или разделителю) на простые значения
2. Каждое простое значение проходит по цепочки конвертеров до тех пор, пока 1 из них не сработает
3. К CSS значению применяется трансформер

Общая схема работы конвертации

1. Полное значение утилиты разбивается (по пробелам или разделителю) на простые значения
2. Каждое простое значение проходит по цепочки конвертеров до тех пор, пока 1 из них не сработает
3. К CSS значению применяется трансформер
4. Итоговое значение подставляется в CSS правило

Конвертеры

```
@function convert-uv-number($value, $data: ())
```

- `$value` - исходное значение
- `$data` - map с дополнительными данными

Конвертеры

```
@function convert-uv-number($value, $data: ())
```

- Выделенное название используется в типах конвертации

Конвертеры

```
@function convert-uv-number($value, $data: ())
```

- Выделенное название используется в типах конвертации
- Применяются друг за другом

Конвертеры

```
@function convert-uv-number($value, $data: ())
```

- Выделенное название используется в типах конвертации
- Применяются друг за другом
- Внутри может использовать другие конвертеры

Конвертеры

```
@function convert-uv-number($value, $data: ())
```

- Выделенное название используется в типах конвертации
- Применяются друг за другом
- Внутри может использовать другие конвертеры
- Можно написать свои

Трансформеры

```
@function gradient($value, $data: ())
```

- Применяются едоножды ко всему итоговому CSS значению
- Типичный кейс: оборот значения в CSS-функцию

Кейс: утилита для градиентов

```
'-Gdl': (
  'properties': background-image,
  'transformer': 'gradient',
  'css-function': 'linear-gradient',
  'conversion': 'gradient',
  'multi-list-separator': ',',
  'keywords': ('position', 'gradient'),
),
```

Кейс: утилита для градиентов

```
'conversion-types': (
  //...
  'gradient': (
    'keyword', 'color', 'cust-prop', 'Pl',
    'number', 'angle', 'global-kw'
  ),
),
```

Кейс: утилита для градиентов

```
-Gdl-r,$action;30p,red;80p,tp
```

Кейс: утилита для градиентов

`-Gdl-r,$action;30p,red;80p,tp` =>

```
.-Gdl-r\,\$action\;30p\,\red\;80p\,tp {

}
```

Кейс: утилита для градиентов

`-Gdl-r,$action;30p,red;80p,tp` =>

```
.-Gdl-r\,\$action\;30p\,\red\;80p\,tp {
  background-image: linear-gradient(

  );
}
```

Кейс: утилита для градиентов

`-Gdl -r,$action;30p,red;80p,tp` =>

```
.-Gdl-r\,\$action\;30p\,\red\;80p\,tp {
  background-image: linear-gradient(
    to right, var(--ml-action) 30%, red 80%, transparent
  );
}
```


ЧТО
ТЫ
ТАКОЕ?

Слабые места 😦

- (пока) Нет first-class поддержки CSS-функций: `calc()`, `clamp()`

Слабые места 😦

- (пока) Нет first-class поддержки CSS-функций: `calc()`, `clamp()`
- CSS значения могут занять используемые спецсимволы: `;`, `$`, `?`

Конфигурация

Настройка инструмента

- Добавление значений: цвета, шрифты, ключевые слова

Настройка инструмента

- Добавление значений: цвета, шрифты, ключевые слова
- Создание утилит

Настройка инструмента

- Добавление значений: цвета, шрифты, ключевые слова
- Создание утилит
- Изменение настроек: брейкпоинты, новые states

Добавить значения в Tailwind 🙂

```
module.exports = {
  theme: {
    extend: {
      fontFamily: {
        display: 'Oswald, ui-serif',
      }
    }
  }
}
```

Добавить утилиту в Tailwind 😢

Надо написать плагин!

- Статические
- Динамические

Статическая утилита в Tailwind

- Вручную пишиту CSS(-in-JS)-правило
- Будут доступны variants

Статическая утилита в Tailwind

```
module.exports = {
  plugins: [
    plugin(function({ addUtilities }) {
      addUtilities({
        '.content-auto': {
          'content-visibility': 'auto',
        },
        '.content-hidden': {
          'content-visibility': 'hidden',
        },
      })
    })
  ]
}
```

Динамическая утилита в Tailwind

- Можно добавить словарь со значениями

Динамическая утилита в Tailwind

- Можно добавить словарь со значениями
- Будет доступен arbitrary синтаксис

Динамическая утилита в Tailwind

- Можно добавить словарь со значениями
- Будет доступен arbitrary синтаксис
- Будут доступны variants

Динамическая утилита в Tailwind

```
module.exports = {
  theme: {
    tabSize: {
      // map with values
    }
  },
  plugins: [
    plugin(function({ matchUtilities, theme }) {
      matchUtilities(
        {
          tab: (value) => ({
            tabSize: value
          }),
        },
        { values: theme('tabSize') }
      )
    })
```

Добавить значения в Unocss 🙂

```
theme: {
  // ...
  colors: {
    'veryCool': '#0000ff', // class="text-very-cool"
  },
}
```

Добавить утилиту в Unocss 🥲

- Лаконичный API

Добавить утилиту в Unocss 🥲

- Лаконичный API
- Простые утилиты - в 1 строку

Добавить утилиту в Unocss 🥲

- Лаконичный API
- Простые утилиты - в 1 строку
- Для сложных - пишем регулярки и императивную конвертацию

Добавить утилиту в Unocss 🥲

```
export default defineConfig({
  rules: [
    ['m-1', { margin: '1px' }],
    [/^p-([\.\d]+)$/, ([_, num]) => ({ padding: `${num}px` })],
  ],
})
```

mlut 😎

Все расширения в одном конфиге за пару строк кода

mlut 😎

Все расширения в одном конфиге за пару строк кода

- Утилиты и их значения
- States
- At-rules
- Breakpoints и общие настройки

Добавить утилиту в mlut

```
@use 'mlut' with (
  $utils-data: (
    'utils': (
      'registry': (
          'Mm': margin-magick,
      ),
    ),
  ),
);
```

Простая утилита в mlut из коробки

- Числовые значения:

```
Mm1r => margin-magick: 1rem
```

Простая утилита в mlut из коробки

- Числовые значения:

```
Mm1r => margin-magick: 1rem
```

- Глобальные ключевые слова:

```
Mm-ih => inherit
```

Простая утилита в mlut из коробки

- Числовые значения:

  `Mm1r => margin-magick: 1rem`

- Глобальные ключевые слова:

  `Mm-ih => inherit`

- Custom properties:

  `Mm-$myCard?200 => var(--ml-myCard, 200px)`

Простая утилита в mlut из коробки

- Числовые значения:

  `Mm1r => margin-magick: 1rem`

- Глобальные ключевые слова:

  `Mm-ih => inherit`

- Custom properties:

  `Mm-$myCard?200 => var(--ml-myCard, 200px)`

- Несколько значений: `Mm10p;1/3 => 10% 33.3333%`

Диспетчеризация

Поиск и выбор, какая функция будет вызываться для типа данных

Диспетчеризация

Поиск и выбор, какая функция будет вызываться для типа данных

- Статическая - на этапе компиляции
- Динамическая - в рантайме

Статическая диспетчеризация

```c
struct Calculator {
    int (*operation)(int, int);
};

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}

int main() {
    Calculator calc;

    calc.operation = add;
    printf("5 + 3 = %d\n", calc.operation(5, 3));

    calc.operation = subtract;
    printf("5 - 3 = %d\n", calc.operation(5, 3));

    return 0;
}
```

Динамическая диспетчеризация (ДД)


```
 1 class Toad {
 2   sleep() {
 3   }
 4 }
 5
 6 class Lizard {
 7   sleep() {
 8   }
 9 }
10
11 function lull(animal) {
12   animal.sleep();
13 }
14
15 const toad = new Toad();
16 lull(toad);
```

ДД в mlut

```scss
// _at-rules.scss
$at-rules-db: (
  'media': (
    'alias': 'm',
    'default': true,
  ),
  'supports': (
    'alias': 's',
  ),
);

// _mk-ar.scss
// ...
$converter: map.get(ml.$at-rules-db, $ar-name, 'converter');

@#{$ar-name} #{meta.call($converter, $ar-str, $this-util)} {
  //..
}
```

Open

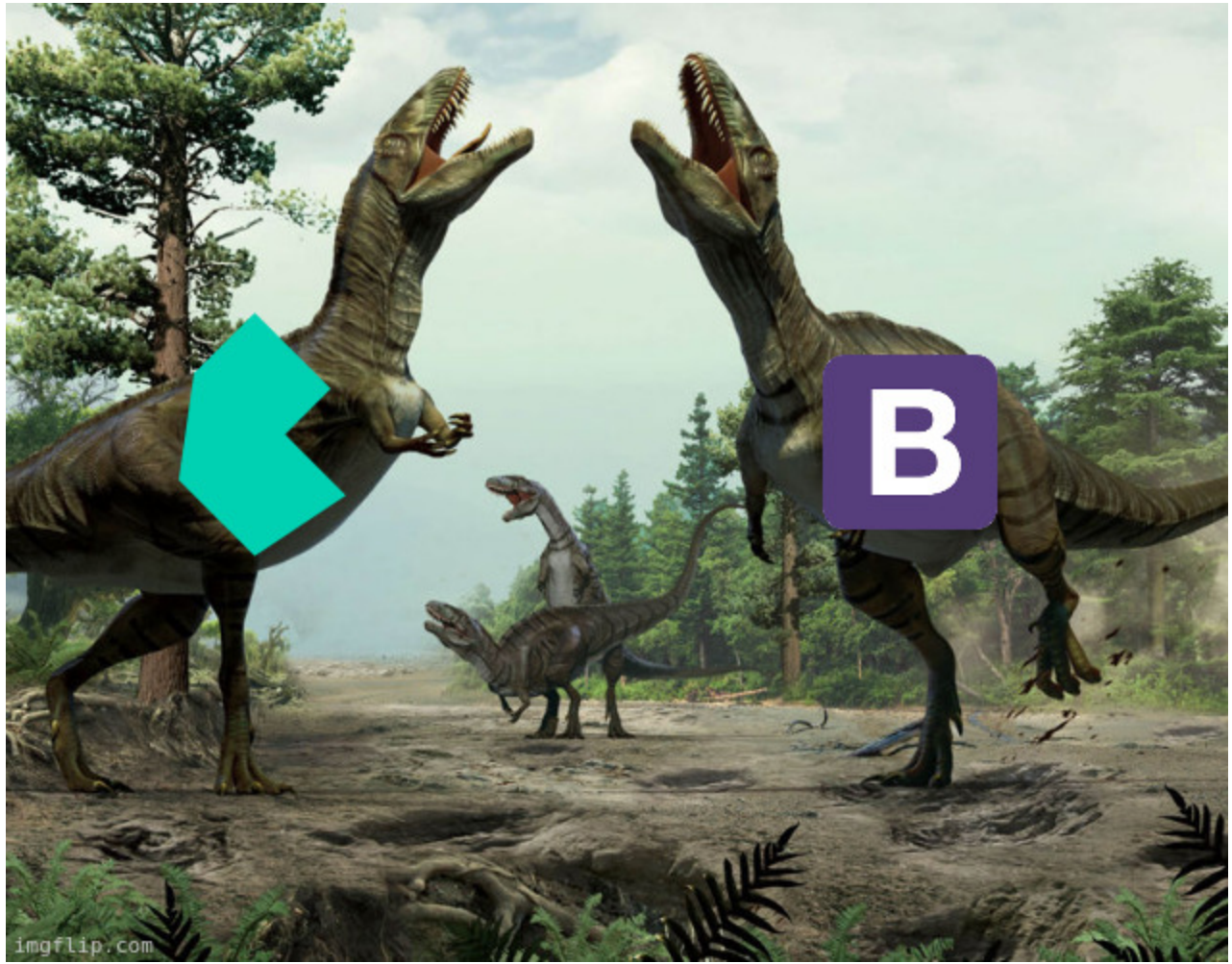
Support for container queries #51

Dgcim1 opened this issue on Mar 7 · 2 comments

```
@use 'mlut/tools' as ml with (
  $utils-data: (
    'utils': (
      'registry': (
        'Ctnt': (
          'properties': container-type,
          'keywords': (
            'is': inline-size,
          ),
        ),
      ),
    ),
  ),
  $at-rules-data: (
    'container': (
      'alias': 'c',
    ),
  ),
);

// using at-rule converter from @supports
ml.$at-rules-db: map.set(
  ml.$at-rules-db,
  'container',
  'converter',
  meta.get-function('convert-ar-supports', false, ml),
);
```

JIT engine


B
imgflip.com

Инструменты старого поколения: AOT

1. Генерируем over9000 утилит

Инструменты старого поколения: AOT

1. Генерируем over9000 утилит
2. Смотрим, какие используются

Инструменты старого поколения: AOT

1. Генерируем over9000 утилит
2. Смотрим, какие используются
3. Удаляем лишние

Проблемы старых инструментов

- Произвольные значения

Проблемы старых инструментов

- Произвольные значения
- Постоянно надо редактировать конфиг

Проблемы старых инструментов

- Произвольные значения
- Постоянно надо редактировать конфиг
- Большой бандл CSS в режиме разработки


Mlut
imgflip.com

Инструменты нового поколения: JIT

1. Смотрим, какие утилиты используются
2. Генерируем только их

Общая схема работы JIT движка

1. Находим файлы с контентом

Общая схема работы JIT движка

1. Находим файлы с контентом
2. Сканируем их и достаем утилиты

Общая схема работы JIT движка

1. Находим файлы с контентом
2. Сканируем их и достаем утилиты
3. Генерируем CSS для найденных утилит

Tailwind 🥲

- PostCSS плагин: AST, интеграции, экосистема

Tailwind 🥲

- PostCSS плагин: AST, интеграции, экосистема
- Oxide - новый движок в v4

Unocss 😎

— Самописный генератор утилит

Unocss 😎

- Самописный генератор утилит
- Самый быстрый

Unocss 😎

- Самописный генератор утилит
- Самый быстрый
- Тесная интеграция с небольшим количеством сборщиков

Unocss 😎

- Самописный генератор утилит
- Самый быстрый
- Тесная интеграция с небольшим количеством сборщиков
- Много дополнительных фич: attributify, shortcuts, compilation, etc

Atomizer 🙂

— Самописный генератор утилит

Atomizer 🙂

- Самописный генератор утилит
- Относительно простой, но с legacy-зависимостями

Atomizer 🙂

- Самописный генератор утилит
- Относительно простой, но с legacy-зависимостями
- Хорошая интеграция с большинством сборщиков: unplugin

mlut 🥴

mlut 🥴

Фронт: TypeScript

- CLI / плагин
- JIT движок

mlut 🥴

Фронт: TypeScript

- CLI / плагин
- JIT движок

Бэк: Sass

- Генератор утилит и конфиги
- CSS библиотека
- Компилятор Sass

Связь Sass и JS

- Получение данных из Sass-конфига

Связь Sass и JS

- Получение данных из Sass-конфига
- Передача утилит в генератор


SASS IN JS
imgflip.com

Sass in JS

1. Загружаем код нужного Sass-модуля

Sass in JS

1. Загружаем код нужного Sass-модуля
2. Дописываем в него код

Sass in JS

1. Загружаем код нужного Sass-модуля
2. Дописываем в него код
3. Компилируем итоговый скрипт в CSS

Sass in JS

1. Загружаем код нужного Sass-модуля
2. Дописываем в него код
3. Компилируем итоговый скрипт в CSS
4. (при необходимости) Достаем данные из вывода

```
1  // default userConfig
2  @use "sass:map";
3  @use "../sass/tools/settings" as ml;
```

```
// default userConfig
@use "sass:map";
@use "../sass/tools/settings" as ml;
```

```
const { css } = (await sass.compileStringAsync(
  userConfig + '\n a{ all: map.keys(map.get(ml.$utils-db, "utils", "registry")); }',
  {
    style: 'compressed',
    loadPaths: [ __dirname ],
  }
));
```

```
// default userConfig
@use "sass:map";
@use "../sass/tools/settings" as ml;
```

```
const { css } = (await sass.compileStringAsync(
  userConfig + '\n a{ all: map.keys(map.get(ml.$utils-db, "utils", "registry")); }',
  {
    style: 'compressed',
    loadPaths: [ __dirname ],
  }
));
```

```
a {
  all: "Ps", "T", "R", "B", "-X", "-Y", "-I", "Zi", "D", // etc
}
```

Особенности работы с Sass

- **Нет рантайма**. Выручает нативный компилятор на Dart: sass-embedded

Особенности работы с Sass

- **Нет рантайма**. Выручает нативный компилятор на Dart: sass-embedded
- **Мало фич в языке**. Нет классов, regexp и т.д. Но есть кое-что из ФП

Особенности работы с Sass

- **Нет рантайма**. Выручает нативный компилятор на Dart: sass-embedded
- **Мало фич в языке**. Нет классов, regexp и т.д. Но есть кое-что из ФП
- **Максимальная интеграция с CSS**. Работа с селекторами, типы данных, перевод структур данных в значения

Заключение

Инсайты

- Не бойтесь безумных идей

Инсайты

- Не бойтесь безумных идей
- Попробуйте пойти до конца, докапаться до сути проблемы

Инсайты

- Не бойтесь безумных идей
- Попробуйте пойти до конца, докапаться до сути проблемы
- Неудача - тоже результат

Зачем?

- Решение проблем

Зачем?

- Решение проблем
- State Of CSS: показать сообществу новые идеи

Зачем?

- Решение проблем
- State Of CSS: показать сообществу новые идеи
- Мечта - full time open source

Вопросы?

Валентин Ульянов

👋 150.lv

📢 t.me/blog150

⭐ github.com/mr150/mlut

𝕏 x.com/mlutcss

Презентация: