Типы данных в CSS

София Валитова

FC Frontend Conf 2019

Профессиональная конференция фронтенд-разработчиков

Предыстория

CSS иногда работает не так, как я ожидаю

```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)
03 height: calc(25% - 0)
```

Предыстория

CSS иногда работает не так, как я ожидаю

```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)        // => 0px
03 height: calc(25% - 0)
```

Предыстория

Посмотрим в спецификацию и пойдем гуглить статьи с примерами

§ **8.1.1. Syntax**

The syntax of a 'calc()' function is:

```
<calc()> = calc( <calc-sum> )
<calc-sum> = <calc-product> [ [ '+' | '-' ] <calc-product> ]*
<calc-product> = <calc-value> [ '*' <calc-value> | '/' <calc-number-value> ]*
<calc-value> = <number> | <dimension> | <percentage> | ( <calc-sum> )
<calc-number-sum> = <calc-number-product> [ [ '+' | '-' ] <calc-number-product> ]*
<calc-number-product> = <calc-number-value> [ '*' <calc-number-value> | '/' <calc-number-valu
<calc-number-value> = <number> | ( <calc-number-sum> )
```

In addition, white space is required on both sides of the '+' and '-' operators. (The '*' and '/' operaters can be used without white space around them.)

Спецификация CSS Values

W3C Candidate Recommendation

CSS Values and Units Module Level 3

W3C Candidate Recommendation, 6 June 2019

108 Tests

This version:

https://www.w3.org/TR/2019/CR-css-values-3-20190606/

Latest published version:

https://www.w3.org/TR/css-values-3/

Editor's Draft:

https://drafts.csswg.org/css-values-3/

Previous Versions:

https://www.w3.org/TR/2019/CR-css-values-3-20190131/
https://www.w3.org/TR/2018/CR-css-values-3-20180814/
https://www.w3.org/TR/2016/CR-css-values-3-20160929/

Где в CSS данные?

Если определять интуитивно, то вот так:

Но давайте попробуем более формально.

Спецификация CSS Syntax

W3C Candidate Recommendation

CSS Syntax Module Level 3

W3C Candidate Recommendation, 16 July 2019

No Tests

This version:

https://www.w3.org/TR/2019/CR-css-syntax-3-20190716/

Latest published version:

https://www.w3.org/TR/css-syntax-3/

Editor's Draft:

https://drafts.csswg.org/css-syntax/

Previous Versions:

https://www.w3.org/TR/2014/CR-css-syntax-3-20140220/

https://www.w3.org/TR/2013/WD-css-syntax-3-20131105/

https://www.w3.org/TR/2013/WD-css-syntax-3-20130919/

Test Suite:

http://test.csswg.org/suites/css-syntax-3_dev/nightly-unstable/

Editors:

```
<style-rule> ::=
  <selectors-list> { <properties-list> }
<selectors-list> ::=
  <selector>[':' <pseudo-class>] ['::' <pseudo-element>]
  [',' <selectors-list>]
<properties-list> ::=
  [<property> ':' <value>] [';' <properties-list>]
```

Немного формальной грамматики

Простой пример

Попробуем описать грамматику отрицательных чисел от -1 до -9:

```
01 <neg_num> ::= "–" <num>
02 <num> ::= "1" | "2" | "3" | "4" | "5" |
03       | "6" | "7" | "8" | "9"
```

Сложение отрицательных чисел:

```
01 <add_expr> ::= <neg_num> "+" "(" <neg_num> ")"
```

Простой пример

Попробуем описать грамматику отрицательных чисел от -1 до -9:

```
01 <neg_num> ::= "-" <num>
02 <num> ::= "1" | "2" | "3" | "4" | "5" |
03         | "6" | "7" | "8" | "9"
```

Терминалы

Нетерминалы

Сложение отрицательных чисел:

```
01 <add_expr> ::= <neg_num> "+" "(" <neg_num> ")"
```

Форма Бэкуса-Наура

"text" или 'text'	терминал
<A>	нетерминал
<A> ::= **<B>**	**<A>** заменимо на **<B>**
<A> <B>	конкатенация
<A> | **<B>**	один из элементов
integer	имя класса нетерминалов, описывается не через BNF

```
01 <expression> ::= <add_expr>
02 <add_expr> ::= <mul_expr> '+' <add_expr>
03     | <mul_expr> '-' <add_expr>
04     | <mul_expr>
05 <mul_expr> ::= integer '*' <mul_expr>
06     | integer '/' <mul_expr>
07     | integer '%' <mul_expr>
08     | integer
09 integer ::= [0-9]+
```

```
01 <expression> ::= <add_expr>
02 <add_expr> ::= <mul_expr> [ ('+' | '-') <add_expr> ]
03 <mul_expr> ::= integer [ ('*' | '/' | '%') <mul_expr> ]
04 integer ::= [0-9]+
```

```
01 <mul_expr> ::= integer '*' <mul_expr>
02     | integer '/' <mul_expr>
03     | integer '%' <mul_expr>
04     | integer
```

```
01 <add_expr> ::= <mul_expr> '+' <add_expr>
02     | <mul_expr> '-' <add_expr>
03     | <mul_expr>
```

Расширенная форма Бэкуса-Наура

[<A>]	условное вхождение
(<A> | <B>) <C>	группировка
{<A>}	повторение элемента (0 или более раз)

```
<style-rule> ::=
  <selectors-list> '{' <properties-list> '}'
<selectors-list> ::=
  <selector>[':' <pseudo-class>] ['::' <pseudo-element>]
  [',' <selectors-list>]
<properties-list> ::=
  [<property> ':' <value>] [';' <properties-list>]
```

```
<style-rule> ::=
  <selectors-list> '{' <properties-list> '}'
```

```
.class#id {
    height: initial;
    width: 100%;
}
```

```
01 <properties-list> ::=
02   [<property> ':' <value>] [';' <properties-list>]
```

```
01 .class#id {
02     height: initial;
03     width: 100%
04 }
```

```
.class {
    height: initial;
    width: 100%;
    border-width: 2px medium 4px;
    box-shadow: rgba(0,0,0,0.4) 10px 10px inset;
    margin: 0 -7px 0 2px;
    transition: opacity .2s ease, visibility .2s ease;
}
```

```
margin: 0 -7px 0 2px;
```

`0 -7px 0 2px` - 4 токена `<length>`

```
01 <margin> ::= "auto"
02     | <length>
03     | <length> {" "} <length>
04     | <length> {" "} <length> {" "} <length>
05     | <length> {" "} <length> {" "} <length> {" "} <length>
```

Value Definition Syntax

1. ключевые слова

   `initial`

2. базовые типы данных

   `<length>`

3. сложные типы данных

   `<'background-attachment'>`

[<A> | **<B>] <C>**	группировка
<A> || **<B>** || **<C>**	один или более из группы в заданном порядке
<A> && **<B>** && **<C>**	один или более из группы в любом порядке
<A>*	ноль или более раз
<A>+	один или более раз
<A>?	необязательный
<A> {n}	A повторяется n раз
<A> {n, m}	A повторяется не меньше n раз и не больше m раз
<A>#	один или более раз, разделенные запятыми

```
<'margin'> ::= "auto" | <length>{1, 4}

<'line-height'> ::=
    normal | <number> | <length> | <percentage>

<'font-family'> =
   [ <family-name> | <generic-family> ]#
```

```
<'border-width'> =
  [ <length> | thick | medium | thin ]{1,4}
```

```
border-width: 3px;
border-width: thick;
border-width: medium thick;
border-width: 8em thick;
```

```
<'text-shadow'> =
  [ inset? [ <length>{2,4} <color>? ]# ] | none
```

```
text-shadow: none;
text-shadow: 1px 1px 2px black;
text-shadow: inset 1px 1px 2px black;
text-shadow: inset 1px 1px 2px black,
    0 0 1em red, 0 3vw green;
```

Посмотрим на проблему еще раз

Посмотрим на проблему еще раз

CSS иногда работает не так, как я ожидаю

```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)        // => 0px
03 height: calc(25% - 0)
```

§ 8.1.1. Syntax

The syntax of a 'calc()' function is:

```
<calc()> = calc( <calc-sum> )
<calc-sum> = <calc-product> [ [ '+' | '-' ] <calc-product> ]*
<calc-product> = <calc-value> [ '*' <calc-value> | '/' <calc-number-value> ]*
<calc-value> = <number> | <dimension> | <percentage> | ( <calc-sum> )
<calc-number-sum> = <calc-number-product> [ [ '+' | '-' ] <calc-number-product> ]*
<calc-number-product> = <calc-number-value> [ '*' <calc-number-value> | '/' <calc-number-valu
<calc-number-value> = <number> | ( <calc-number-sum> )
```

In addition, white space is required on both sides of the '+' and '-' operators. (The '*' and '/' operaters can be used without white space around them.)

UAs must support 'calc()' expressions of at least 20 terms, where each `NUMBER`, `DIMENSION`, or `PERCENTAGE` is a term. If a 'calc()' expression contains more than the supported number of terms, it must be treated as if it were invalid.

Вокруг + и - должны быть пробелы

```
calc(50%-2em);
```

```
01 calc(50% - 2em);  // => <length> '-' <length>
02 calc(50% -2em);   // => <length> <length>
03 calc(50% - -2em);
04 calc(50% - (-2em));
```

Посмотрим на проблему еще раз

CSS иногда работает не так, как я ожидаю

```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)        // => 0px
03 height: calc(25% - 0)
```

Calc(). Type Checking

1. Нельзя использовать в одном выражении <length> и <time>
2. Нельзя использовать в одном выражении <angle> и <length>
3. Нельзя складывать или вычитать <length> и <number-token>
4. ...
5. ...

Calc(). Type Checking

Нельзя использовать в одном выражении несовместимые типы

```
01 calc(5px + 10s)  // => <length> '+' <time>
02 calc(5deg / 5em)  // => <angle> '/' <length>
03 calc(5px - 5)  // => <length> '-' <number-token>
04 calc(5px - 0)  // => <length> '-' <number-token>
```

Посмотрим на проблему еще раз

CSS иногда работает не так, как я ожидаю

```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)        // => 0px
03 height: calc(25% - 0)
```

Похожая проблема

```
01 height: -12px
02 height: calc(-12px)  // => 0px
```

Проверка соответствия значения свойства его грамматике

CSS Type Checking

Проверка соответствия значения свойства его грамматике

10.5 Content height: the 'height' property

Name:	***height***
Value:	<length> | <percentage> | auto | inherit
Initial:	auto
Applies to:	all elements but non-replaced inline elements, table columns, and column groups
Inherited:	no
Percentages:	see prose

CSS Type Checking

Проверка соответствия значения свойства его грамматике

W3C Working Draft

the element itself, and thus a percentage height on such an element can always be resolved. However, it may be that the height is not known until elements that come later in the document have been processed.

Negative values for 'height' are illegal.

For example, the following rule sets the content height of paragraphs to 100 pixels:

```
p { height: 100px }
```

Paragraphs of which the height of the contents exceeds 100 pixels will overflow according to the 'overflow'

§ 5.1. Range Restrictions and Range Definition Notation

Properties can restrict numeric values to some range. If the value is outside the allowed range, then un
otherwise specified, the declaration is invalid and must be ignored. Range restrictions can be annotated in the numeric type notation using ***CSS bracketed range notation***— `[min,max]` —within the angle brackets, after the identifying keyword, indicating a closed range between (and including) *min* and *max*. For example, <integer [0,10]> indicates an integer between '0' and '10', inclusive.

> Note: CSS values generally do not allow open ranges; thus only square-bracket notation is used.

CSS theoretically supports infinite precision and infinite ranges for all value types; however in reality implementations have finite capacity. UAs should support reasonably useful ranges and precisions. Range extremes that are ideally unlimited are indicated using ∞ or –∞ as appropriate.

If no range is indicated, either by using the bracketed range notation or in the property description, then `[–∞,∞]` is assumed.

> Note: At the time of writing, the bracketed range notation is new; thus in most CSS specifications any range limitations are described only in prose. (For example, "Negative values are not allowed" or "Negative values are invalid" indicate a `[0,∞]` range.) This does not make them any less binding.

CSS Type Checking

Проверка соответствия значения свойства его грамматике

Грамматика свойства:

```
01 height : <length>[0,∞] | auto | <percantage>
```

* Negative values for 'height' are illegal.

CSS Type Checking

```
01 height: -12px          // => auto
02 height: calc(-12px)    // => 0px
```

Проводится перед каскадом

CSS Type Checking

```
01 height: -12px          // => auto
02 height: calc(-12px)    // => 0px

01 height: -12s           // => auto
02 height: calc(-12s)     // => auto
```

Проверка финального типа, но не диапазона

Calc(). Range Checking

```
01 height: calc(-12px);
02     // => 0px при обращении
```

Calc(). Range Checking

```
01 height: calc(-12px);
02     // => 0px при обращении
```

Разные ситуации:

```
01 height: -12px;
02     // property syntax check
```

```
01 height: calc(-12px);
02     // calc range check
```

Calc(). Range Checking

```
height: 100px; // => 100px
```

```
height: 50%; // => 50px
```

```
height: inherit; // 25px
```

Calc(). Range Checking

```
height: 100px; // => 100px
```

```
height: calc(50% - 25px); // => 25px
```

```
height: inherit; // => calc(50% - 25px) => 0px
```

Посмотрим на проблему еще раз

CSS иногда работает не так, как я ожидаю

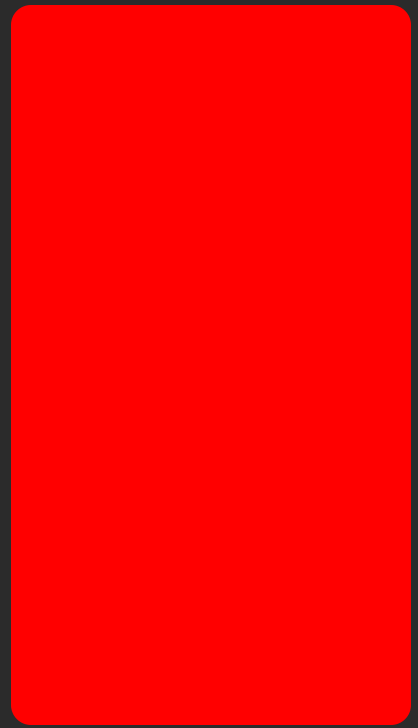
```
01 height: calc(3px+4px)
02 height: calc(3px - 4px)          // => 0px
03 height: calc(25% - 0)
```

Итого:

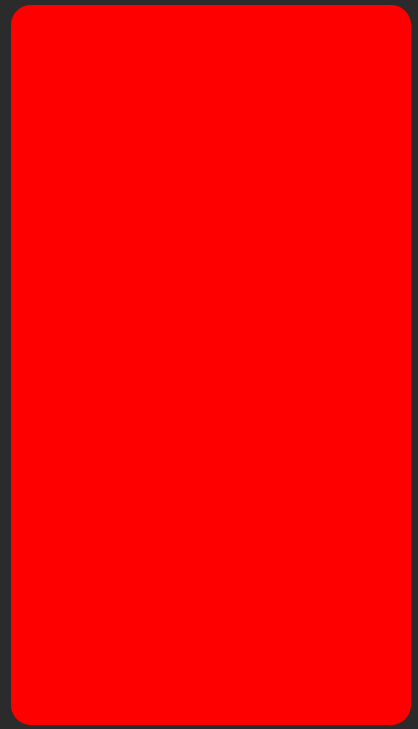
1. В CSS есть проверка синтаксиса и типов
    1. Невалидный синтаксис => значение выше из каскада
    2. Не подходит тип => значение выше из каскада
2. Если используется calc(), то есть проверка синтаксиса, типов и диапазона
    1. Невалидный синтаксис => значение выше из каскада
    2. Не подходит тип => значение выше из каскада
    3. Не подходит диапазон => значение приводится

Взгляд в будущее

```
#el {
  background-color: red;
  animation: anim infinite 3s;
}
@keyframes anim {
  50% { background-color: blue; }
}
```



```
#el {
  --color-stop: red;
  background: var(--color-stop);
  animation: anim infinite 3s;
}
@keyframes anim {
  50% { --color-stop: blue; }
}
```



Почему?

Потому что изначально CSS-переменные не типизированы

```
01 div {
02   --name: red;
03   --name: 11px;
04   --name: calc(14deg * 3);
05 }
```

```
#el {
  --color-stop: red;
  margin: 50px;
  animation: anim infinite 3s;
}
@keyframes anim {
  50% { --color-stop: 10px;
        margin: var(--color-stop); }
}
```

Почему?

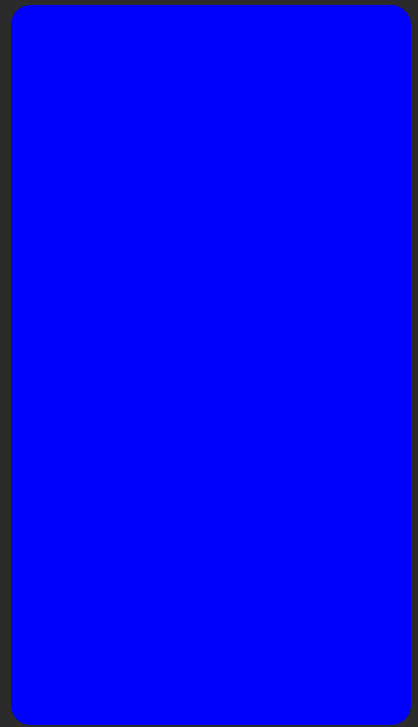
Браузер не знает, как построить переход от одного значения к другому

```
01 red  =>  ???  =>  blue
02 red  =>  ???  =>  10px
```

Типизация переменной

```
01 CSS.registerProperty({
02   name: '--color-stop',
03   syntax: '<color>',
04   inherits: false,
05   initialValue: 'transparent'
06 })
```

```
01 #el {
02   --color-stop: red;
03   background: var(--color-stop);
04   animation: anim infinite 3s;
05 }
06 @keyframes anim {
07   50% { --color-stop: blue; }
08 }
```



CSS Typed OM

W3C Editor's Draft

CSS Typed OM Level 1

Editor's Draft, 1 October 2019

This version:

https://drafts.css-houdini.org/css-typed-om-1/

Latest published version:

https://www.w3.org/TR/css-typed-om-1/

Previous Versions:

https://www.w3.org/TR/2018/WD-css-typed-om-1-20180410/
https://www.w3.org/TR/2017/WD-css-typed-om-1-20170801/
https://www.w3.org/TR/2016/WD-css-typed-om-1-20160607/

Feedback:

public-houdini@w3.org with subject line "`[css-typed-om] … message topic …`" (archives)

Issue Tracking:

Inline In Spec

JS =>

elem.style.height = "1px"

=> CSS

CSS Typed OM

```
01 const height = CSS.px(2);
02 elem.attributeStyleMap.set("height", height);
03
04 elem.attributeStyleMap.get("height"); // =>
05   // CSSUnitValue {value: 2, unit: "px"}
```

CSS Typed OM

1. Лёгкое и наглядное взаимодействие с CSS OM
2. Использование Value Definition Syntax
3. Очень гибкая типизация переменных
4. Можно посмотреть внутрь CSS

Итого

- В CSS есть данные, типы данных, проверка типов и все связанное
- Если CSS работает не так, как вы ожидаете, проверьте типы.
- Для описания типов данных в CSS есть свой Value Definition Syntax.
- Его знание облегчает чтение спецификаций.
- Если спека сложная, понять ее помогут эксперименты в браузере
- Благодаря Houdini скоро это ждет нас всех

Источники

- Спецификация css-values-3
- Урезанная версия на MDN по-русски
- Хорошая статья о BNF на английском
- Рабочие черновики Houdini
- Доклад про Houdini от Никиты Дубко

Почтовые рассылки W3C

- Расслыка public-houdini
- Расслыка www-style

София Валитова из ВКонтакте

- ariarzer@gmail.com
- Twitter – @ariarzer
- ВКонтакте – @ariarzer
- Telegram – @ariarzer

Презентация сделана с помощью Shower