CSS value processing

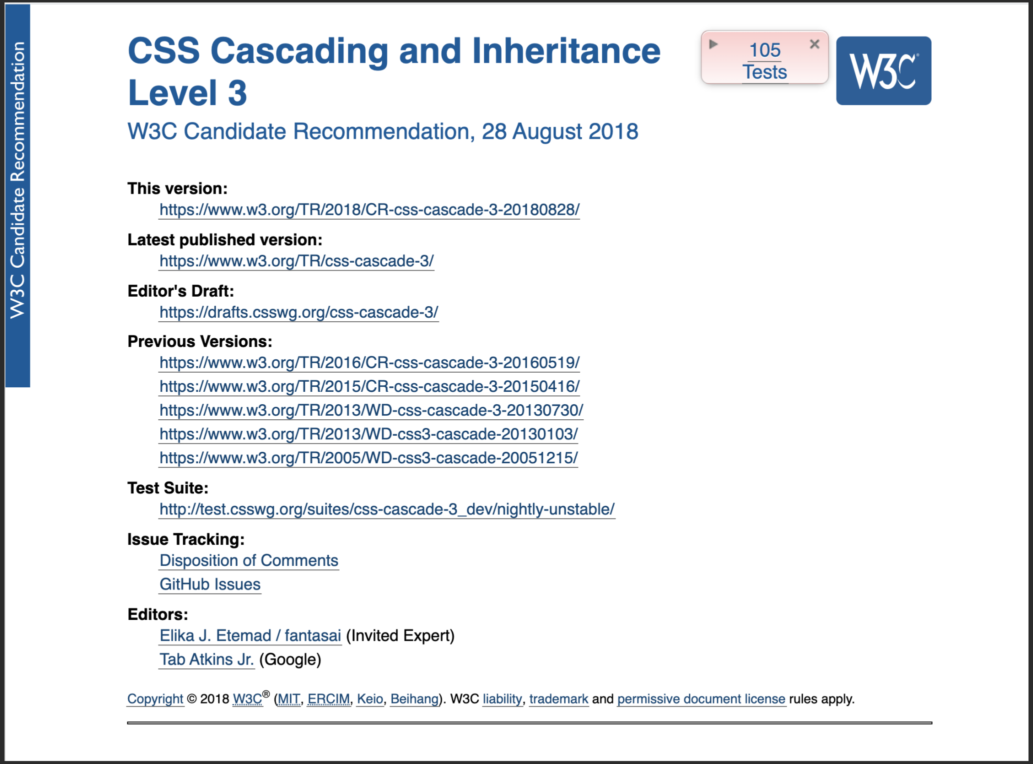
CSS иногда работает не так, как я ожидаю

```
<div class="blue"> CSS </div>
<style>
  .blue {
    color: blue !important;
    animation: anim 2s infinite;
  }
  .@keyframes anim {
    50% {color: chartreuse;}
  }
</style>
```

CSS иногда работает не так, как я ожидаю

CSS

Спецификация



W3C Candidate Recommendation

105 Tests

W3C®

CSS Cascading and Inheritance Level 3

W3C Candidate Recommendation, 28 August 2018

This version:
https://www.w3.org/TR/2018/CR-css-cascade-3-20180828/

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

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

Previous Versions:
https://www.w3.org/TR/2016/CR-css-cascade-3-20160519/
https://www.w3.org/TR/2015/CR-css-cascade-3-20150416/
https://www.w3.org/TR/2013/WD-css-cascade-3-20130730/
https://www.w3.org/TR/2013/WD-css3-cascade-20130103/
https://www.w3.org/TR/2005/WD-css3-cascade-20051215/

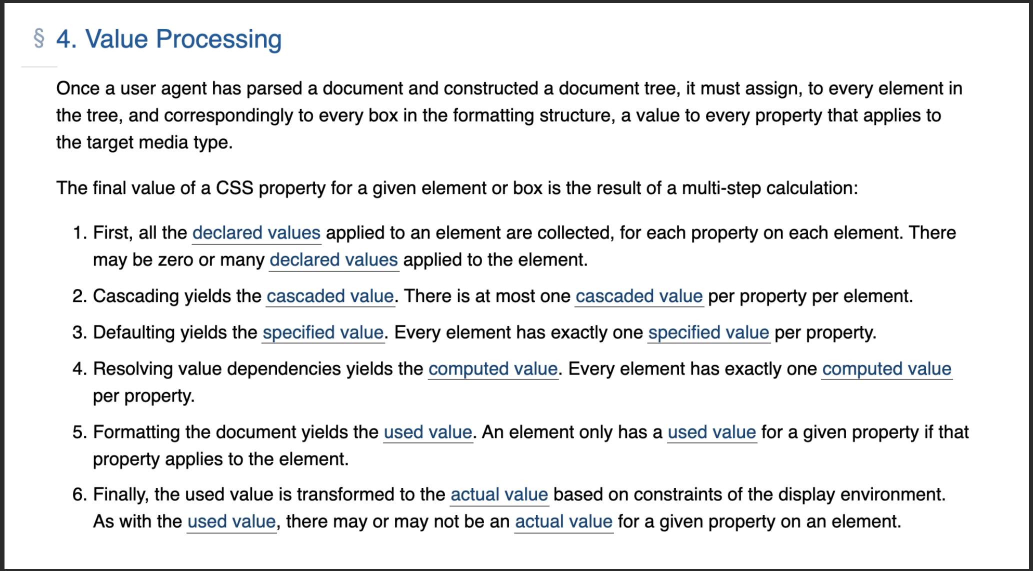
Test Suite:
http://test.csswg.org/suites/css-cascade-3_dev/nightly-unstable/

Issue Tracking:
Disposition of Comments
GitHub Issues

Editors:
Elika J. Etemad / fantasai (Invited Expert)
Tab Atkins Jr. (Google)

Кратко

- Выбрать объявления свойства, которые теоретически подходят для заданного элемента
- Выбрать из них то одно, которое подходит, по заданным правилам
- Сделать его удобным для наследования
- Сделать его удобным для отрисовки



§ 4. Value Processing

Once a user agent has parsed a document and constructed a document tree, it must assign, to every element in the tree, and correspondingly to every box in the formatting structure, a value to every property that applies to the target media type.

The final value of a CSS property for a given element or box is the result of a multi-step calculation:

1. First, all the declared values applied to an element are collected, for each property on each element. There may be zero or many declared values applied to the element.
2. Cascading yields the cascaded value. There is at most one cascaded value per property per element.
3. Defaulting yields the specified value. Every element has exactly one specified value per property.
4. Resolving value dependencies yields the computed value. Every element has exactly one computed value per property.
5. Formatting the document yields the used value. An element only has a used value for a given property if that property applies to the element.
6. Finally, the used value is transformed to the actual value based on constraints of the display environment. As with the used value, there may or may not be an actual value for a given property on an element.

По спецификации

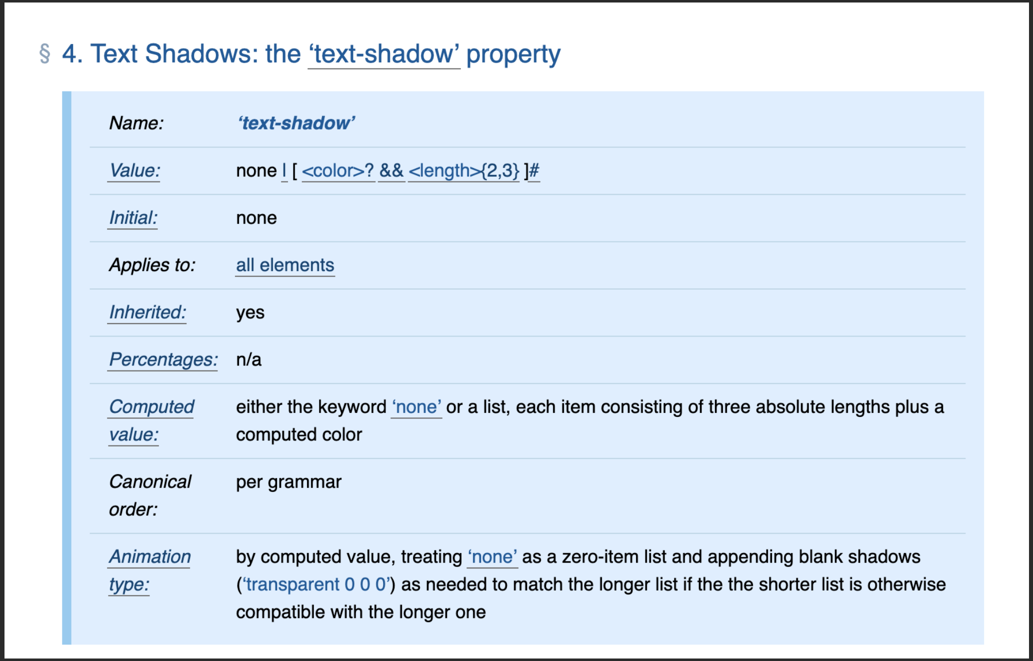
1. Declared value
2. Cascaded value
3. Specified value
4. Computed value
5. Used Value
6. Actual Value

Declared value

Сбор всех деклараций свойства, применимых к элементу

- @media, @supports
- Подходящий селектор
- Type checking

Type checking



§ 4. Text Shadows: the 'text-shadow' property

Name:	***'text-shadow'***
Value:	none | [<color>? && <length>{2,3}]#
Initial:	none
Applies to:	all elements
Inherited:	yes
Percentages:	n/a
Computed value:	either the keyword 'none' or a list, each item consisting of three absolute lengths plus a computed color
Canonical order:	per grammar
Animation type:	by computed value, treating 'none' as a zero-item list and appending blank shadows ('transparent 0 0 0') as needed to match the longer list if the the shorter list is otherwise compatible with the longer one

Type checking

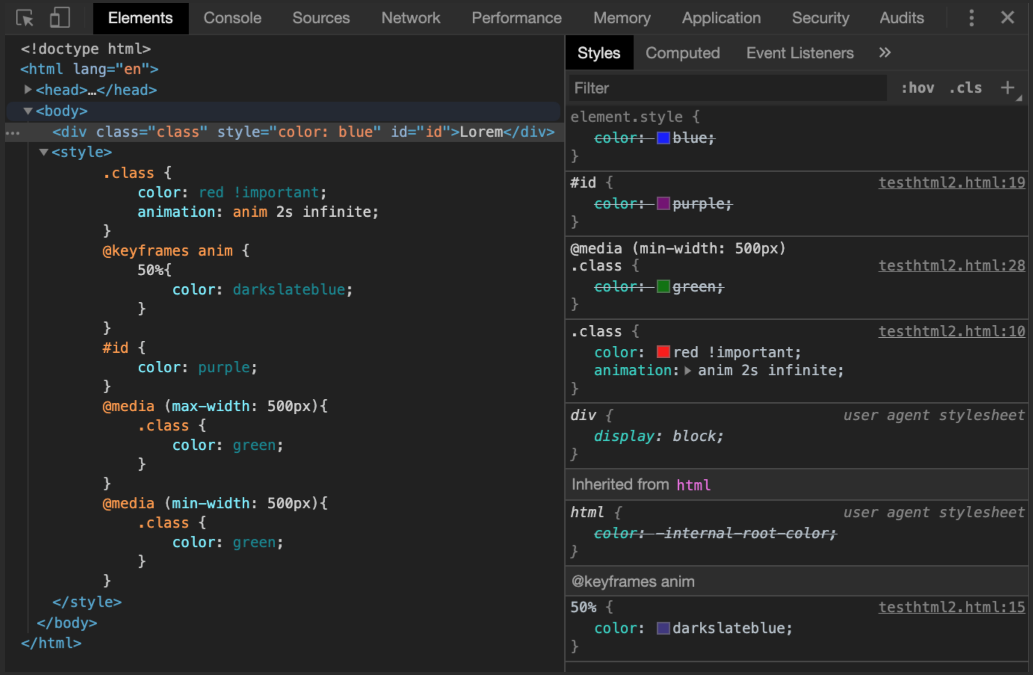
```
<'text-shadow'> =
  none | [ <color>? && <length>{2,3} ]#
```

```
text-shadow: black 0.1em 0.1em 0.2em;
text-shadow: 0.1em 0.1em black;
text-shadow: 0.1em
```

подробнее про типы данных в CSS

Declared value

```
<div class="a"> CSS </div>
<style>
  .a { text-shadow: 0.1em 0.1em black; }
  .a { text-shadow: 0.1em }
</style>
```



```
<!doctype html>
<html lang="en">
▶<head>…</head>
▼<body>
    <div class="class" style="color: blue" id="id">Lorem</div>
  ▼<style>
        .class {
            color: red !important;
            animation: anim 2s infinite;
        }
        @keyframes anim {
            50%{
                color: darkslateblue;
            }
        }
        #id {
            color: purple;
        }
        @media (max-width: 500px){
            .class {
                color: green;
            }
        }
        @media (min-width: 500px){
            .class {
                color: green;
            }
        }
    </style>
  </body>
</html>
```

element.style {
  ~~color: 🟦 blue;~~
}

#id { testhtml2.html:19
  ~~color: 🟪 purple;~~
}

@media (min-width: 500px)
.class { testhtml2.html:28
  ~~color: 🟩 green;~~
}

.class { testhtml2.html:10
  color: 🟥 red !important;
  animation: ▶ anim 2s infinite;
}

div { user agent stylesheet
  display: block;
}

Inherited from html

html { user agent stylesheet
  ~~*color: -internal-root-color;*~~
}

@keyframes anim

50% { testhtml2.html:15
  color: 🟪 darkslateblue;
}

Cascaded value

Каскад принимает неупорядоченный список объявленных значений, сортирует их по приоритету их объявления, и выводит одно значение.

```
<div style="color: green">
  <span style="color: purple"> CSS </span>
</div>
```

Cascaded value

```
<div class="blue" id="red"> CSS </div>
<style>
  .blue { color: blue; }
  #red { color: red; }
</style>
```

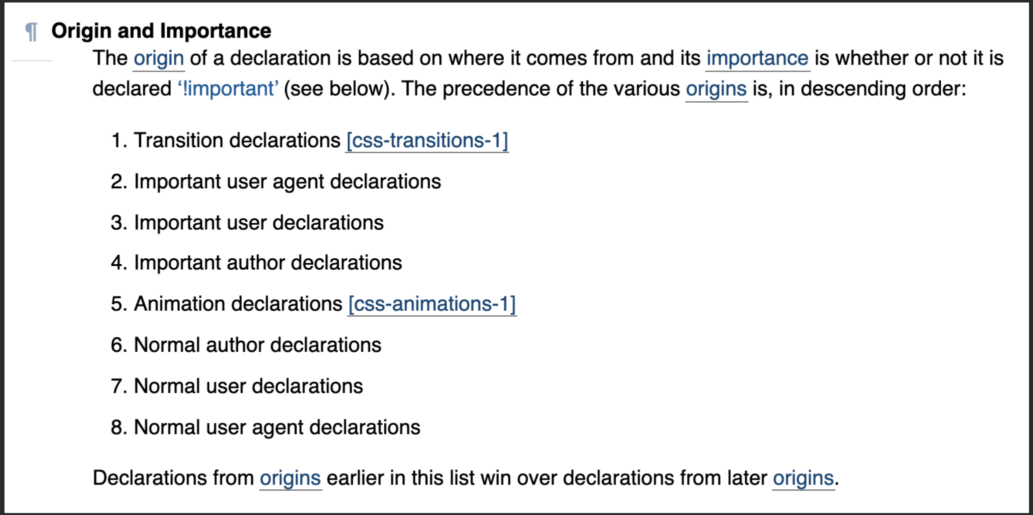
Cascaded value

```
<div class="blue" id="red"> CSS </div>
<style>
  .blue { color: blue !important; }
  #red { color: red; }
</style>
```

Cascaded value

```
<div class="blue" id="red"> CSS </div>
<style>
  .blue { color: blue !important; }
  #red { color: red !important; }
</style>
```

Cascaded value



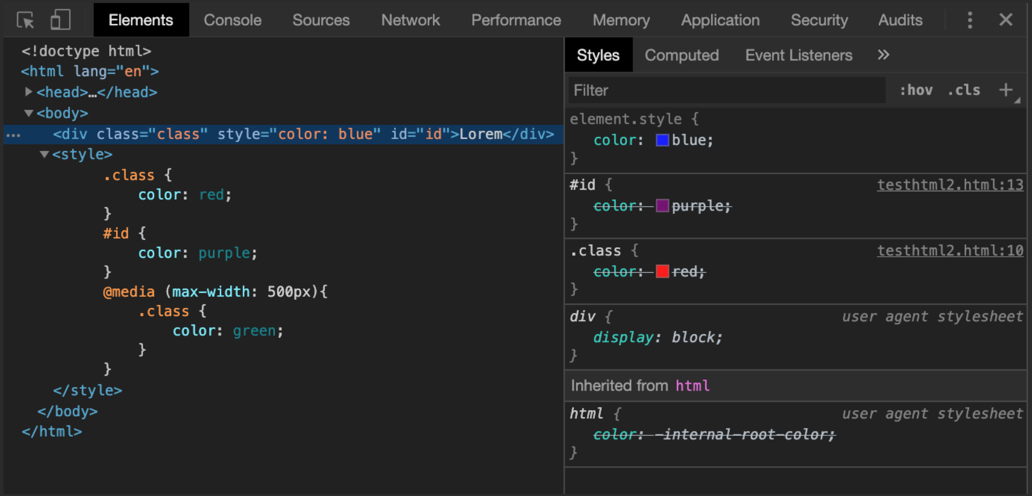
¶ **Origin and Importance**

The origin of a declaration is based on where it comes from and its importance is whether or not it is declared ‘!important’ (see below). The precedence of the various origins is, in descending order:

1. Transition declarations [css-transitions-1]
2. Important user agent declarations
3. Important user declarations
4. Important author declarations
5. Animation declarations [css-animations-1]
6. Normal author declarations
7. Normal user declarations
8. Normal user agent declarations

Declarations from origins earlier in this list win over declarations from later origins.

Cascaded value



```
<!doctype html>
<html lang="en">
▶<head>…</head>
▼<body>
    <div class="class" style="color: blue" id="id">Lorem</div>
  ▼<style>
        .class {
            color: red;
        }
        #id {
            color: purple;
        }
        @media (max-width: 500px){
            .class {
                color: green;
            }
        }
    </style>
  </body>
</html>
```

```
element.style {
   color: ■blue;
}
#id {                                testhtml2.html:13
   ~~color: ■purple;~~
}
.class {                             testhtml2.html:10
   ~~color: ■red;~~
}
div {                            user agent stylesheet
   display: block;
}
Inherited from html
html {                           user agent stylesheet
   ~~color: -internal-root-color;~~
}
```

Коэффициент специфичности

Селектора деляться на 3 уровня:

#id	1 0 0
.class, :hover, [name="value"]	0 1 0
#id [name="value"]	0 0 1

Коэффициент специфичности

span	0	0	1
.class #id	1	1	0
.class #id::before:hover	1	2	1
#id [name="value"]	1	1	0

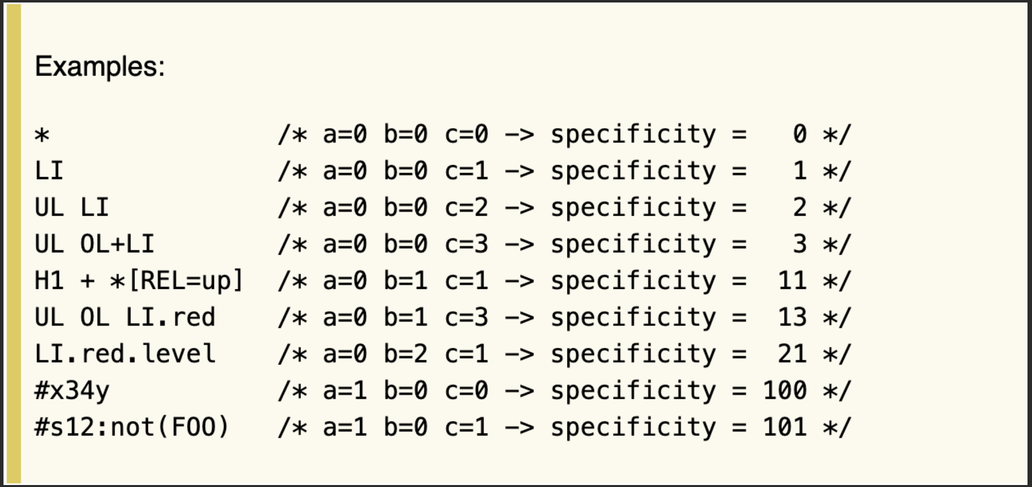
Коэффициент специфичности

span	0 * 100	0 * 10	1 * 1
.class #id	1 * 100	1 * 10	0 * 1
.class #id::before:hover	1 * 100	2 * 10	1 * 1
#id [name="value"]	1 * 100	1 * 10	0 * 1

Коэффициент специфичности

span	0 * 100 +	0 * 10 +	1 * 1 =	001
.class #id	1 * 100 +	1 * 10 +	0 * 1 =	110
.class #id::before:hover	1 * 100 +	2 * 10 +	1 * 1 =	121
#id [name="value"]	1 * 100 +	1 * 10 +	0 * 1 =	110

Коэффициент специфичности



Examples:

```
*               /* a=0 b=0 c=0 -> specificity =   0 */
LI              /* a=0 b=0 c=1 -> specificity =   1 */
UL LI           /* a=0 b=0 c=2 -> specificity =   2 */
UL OL+LI        /* a=0 b=0 c=3 -> specificity =   3 */
H1 + *[REL=up]  /* a=0 b=1 c=1 -> specificity =  11 */
UL OL LI.red    /* a=0 b=1 c=3 -> specificity =  13 */
LI.red.level    /* a=0 b=2 c=1 -> specificity =  21 */
#x34y           /* a=1 b=0 c=0 -> specificity = 100 */
#s12:not(FOO)   /* a=1 b=0 c=1 -> specificity = 101 */
```

Коэффициент специфичности

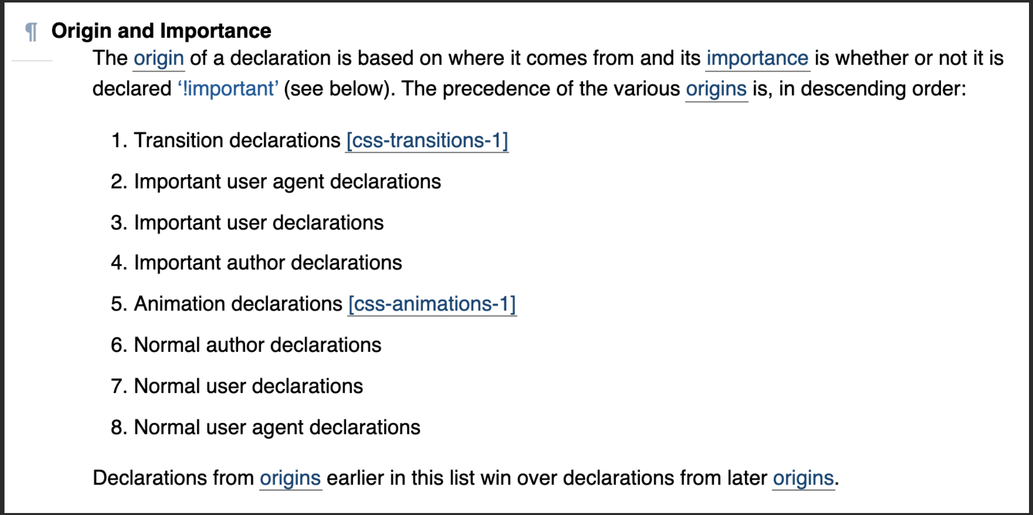
span	0 * 2^32 +	0 * 2^16 +	1 * 2^8
.class #id	1 * 2^32 +	1 * 2^16 +	0 * 2^8
.class #id::before:hover	1 * 2^32 +	2 * 2^16 +	1 * 2^8
#id [name="value"]	1 * 2^32 +	1 * 2^16 +	0 * 2^8

Коэффициент специфичности

CSS иногда работает не так, как я ожидаю

```
<div class="blue"> CSS </div>
<style>
  .blue {
    color: blue !important;
    animation: anim 3s infinite;
  }
  .@keyframes anim {
    50% {color: chartreuse;}
  }
</style>
```

Cascaded value



¶ **Origin and Importance**

The origin of a declaration is based on where it comes from and its importance is whether or not it is declared ‘!important’ (see below). The precedence of the various origins is, in descending order:

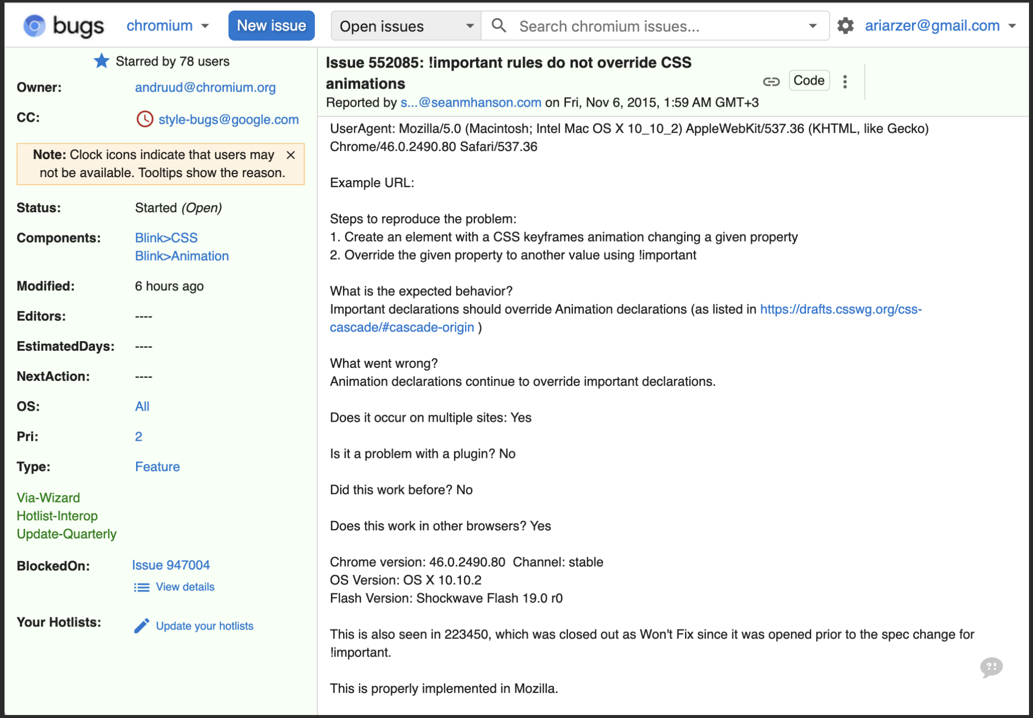
1. Transition declarations [css-transitions-1]
2. Important user agent declarations
3. Important user declarations
4. Important author declarations
5. Animation declarations [css-animations-1]
6. Normal author declarations
7. Normal user declarations
8. Normal user agent declarations

Declarations from origins earlier in this list win over declarations from later origins.

CSS иногда работает не так, как я ожидаю

CSS

Cascaded value



Starred by 78 users

Owner:	andruud@chromium.org
CC:	style-bugs@google.com

Note: Clock icons indicate that users may not be available. Tooltips show the reason.

Status:	Started *(Open)*
Components:	Blink>CSS Blink>Animation
Modified:	6 hours ago
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	All
Pri:	2
Type:	Feature

Via-Wizard
Hotlist-Interop
Update-Quarterly

BlockedOn:	Issue 947004 View details
Your Hotlists:	Update your hotlists

Issue 552085: !important rules do not override CSS animations

Reported by s...@seanmhanson.com on Fri, Nov 6, 2015, 1:59 AM GMT+3

UserAgent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36

Example URL:

Steps to reproduce the problem:
1. Create an element with a CSS keyframes animation changing a given property
2. Override the given property to another value using !important

What is the expected behavior?
Important declarations should override Animation declarations (as listed in https://drafts.csswg.org/css-cascade/#cascade-origin)

What went wrong?
Animation declarations continue to override important declarations.

Does it occur on multiple sites: Yes

Is it a problem with a plugin? No

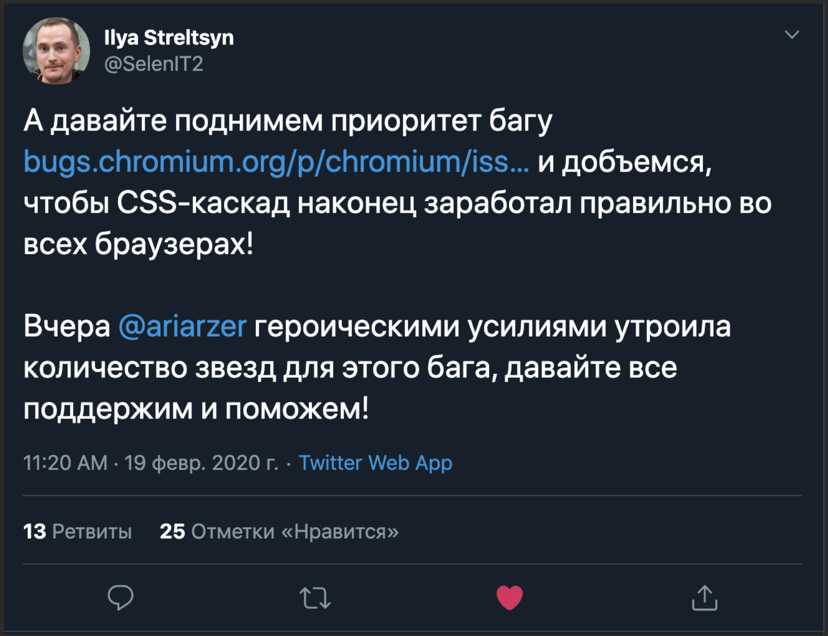
Did this work before? No

Does this work in other browsers? Yes

Chrome version: 46.0.2490.80 Channel: stable
OS Version: OS X 10.10.2
Flash Version: Shockwave Flash 19.0 r0

This is also seen in 223450, which was closed out as Won't Fix since it was opened prior to the spec change for !important.

This is properly implemented in Mozilla.



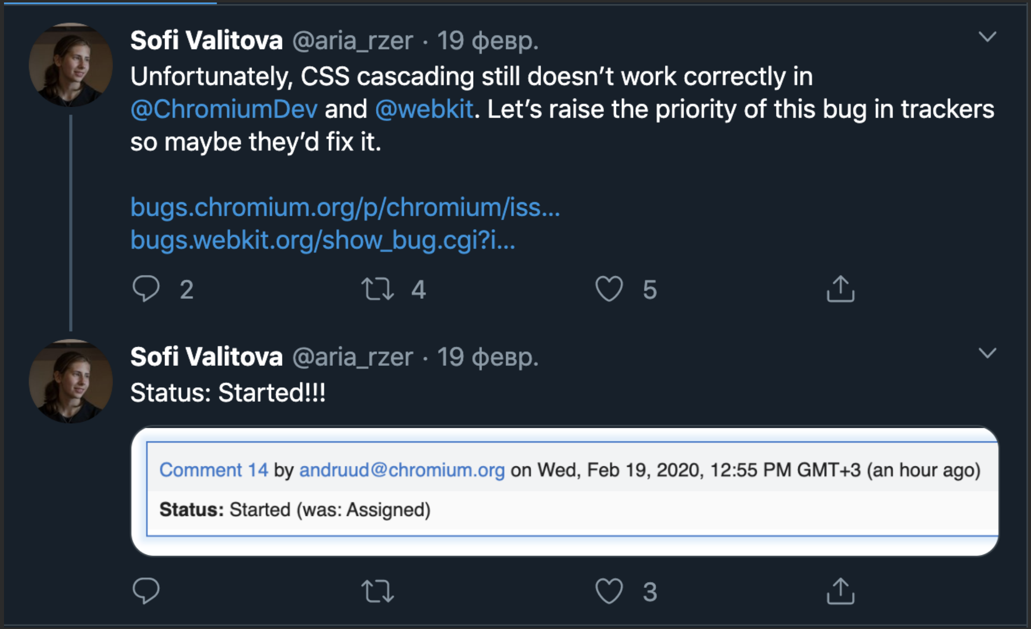
Ilya Streltsyn
@SelenIT2

А давайте поднимем приоритет багу bugs.chromium.org/p/chromium/iss… и добъемся, чтобы CSS-каскад наконец заработал правильно во всех браузерах!

Вчера @ariarzer героическими усилиями утроила количество звезд для этого бага, давайте все поддержим и поможем!

11:20 AM · 19 февр. 2020 г. · Twitter Web App

13 Ретвиты **25** Отметки «Нравится»



Sofi Valitova @aria_rzer · 19 февр.

Unfortunately, CSS cascading still doesn't work correctly in @ChromiumDev and @webkit. Let's raise the priority of this bug in trackers so maybe they'd fix it.

bugs.chromium.org/p/chromium/iss...
bugs.webkit.org/show_bug.cgi?i...

2 4 5

Sofi Valitova @aria_rzer · 19 февр.

Status: Started!!!

Comment 14 by andruud@chromium.org on Wed, Feb 19, 2020, 12:55 PM GMT+3 (an hour ago)

Status: Started (was: Assigned)

3

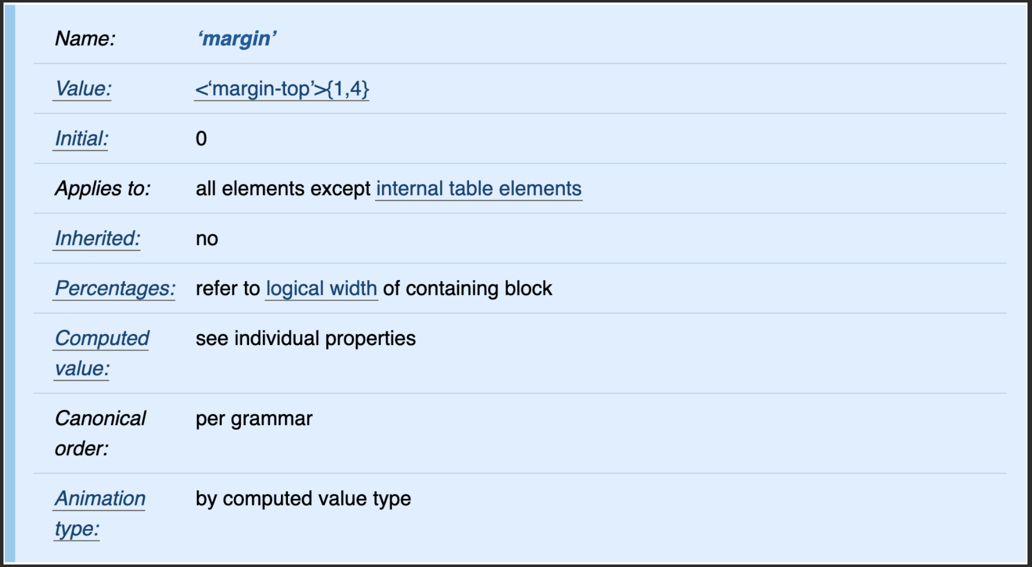
Поставьте звёдочку багу

Specified value

```
<div> CSS </div>
```

```
color: ???
margin: ???
padding: ???
```

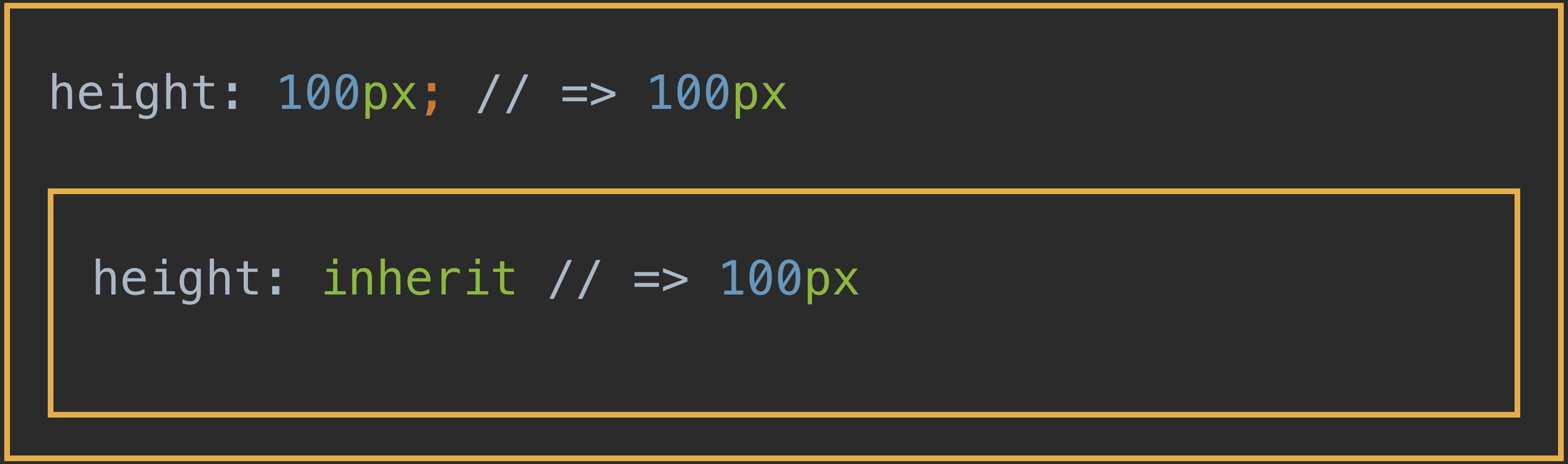
Initial value



Name:	***'margin'***
Value:	<'margin-top'>{1,4}
Initial:	0
Applies to:	all elements except internal table elements
Inherited:	no
Percentages:	refer to logical width of containing block
Computed value:	see individual properties
Canonical order:	per grammar
Animation type:	by computed value type

Computed value

Значение, которое наследуется



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

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

Computed value

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

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

        height: inherit; // => 25px CV = 50%
```

Computed value

```
font-size: 10px;

  height: 5em; // => 50px

    font-size: 20px;
    height: inherit; // => 50px
```

Computed value

```
font-size: 10px;

  height: 5em; // => 50px CV=50px

    font-size: 20px;
    height: inherit; // => 50px CV=50px
```

Used Values

- результат взятия computed value и завершения любых оставшихся вычислений, чтобы сделать его абсолютным теоретическим значением, используемым в макете документа.

- вычисление %
- вычисление calc
- ...

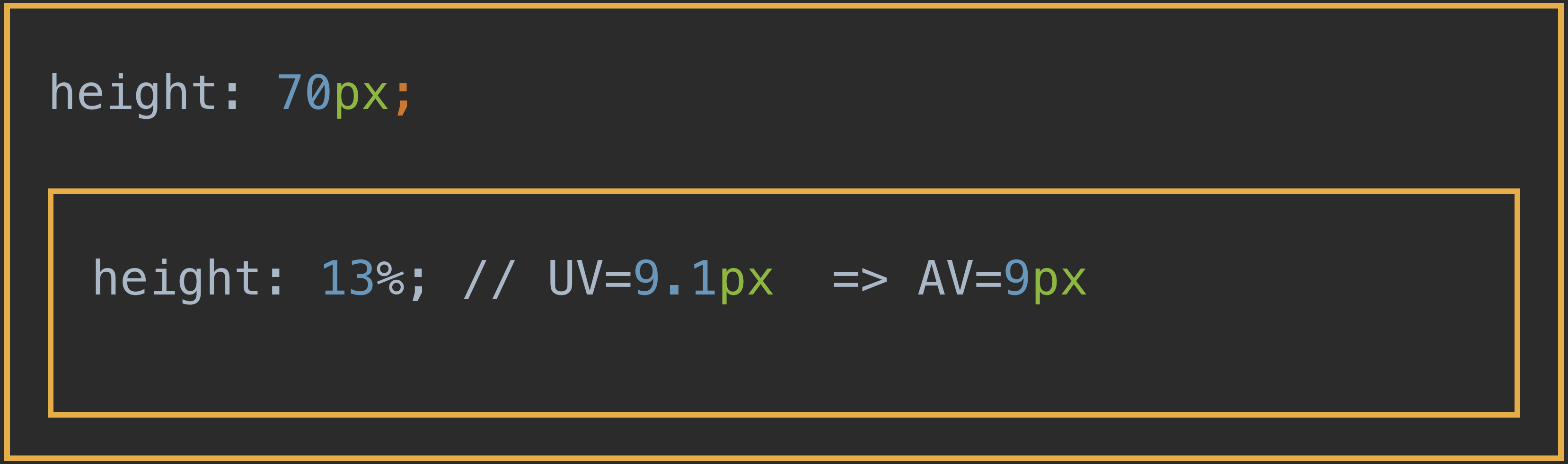
Used Values

```
height: 70px;
```

```
height: 13%; // CV=13% => UV=9.1px
```

Actual Values

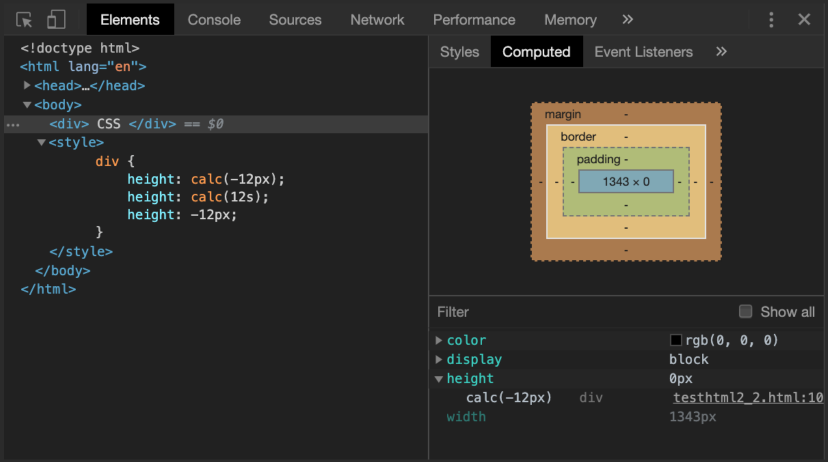
Значение преобразуется на основе ограничений среды отображения.



```
height: 70px;
```

```
height: 13%; // UV=9.1px  => AV=9px
```

Несколько слов про calc

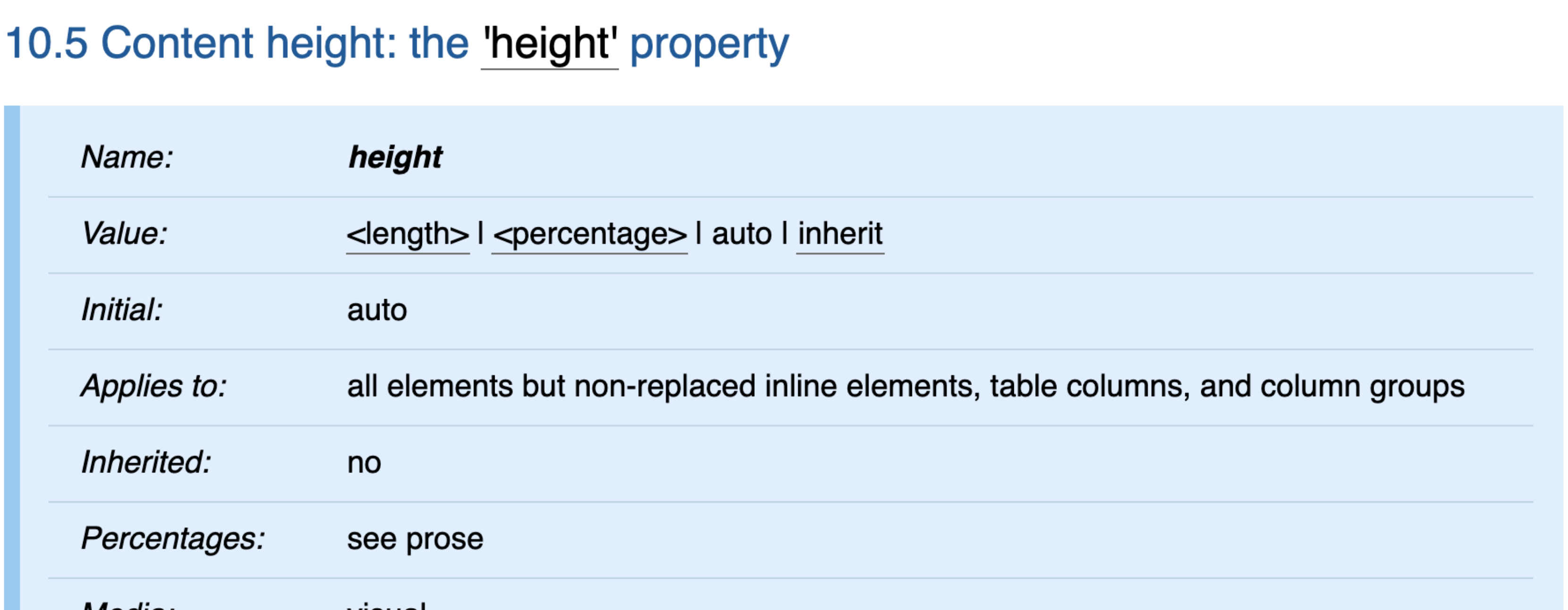


```
<!doctype html>
<html lang="en">
  ▶<head>…</head>
  ▼<body>
      <div> CSS </div> == $0
    ▼<style>
          div {
              height: calc(-12px);
              height: calc(12s);
              height: -12px;
          }
      </style>
    </body>
</html>
```

```
▶ color      rgb(0, 0, 0)
▶ display    block
▼ height     0px
    calc(-12px)   div   testhtml2_2.html:10
  width      1343px
```

CSS Type Checking

Проверка соответствия значения свойства его грамматике

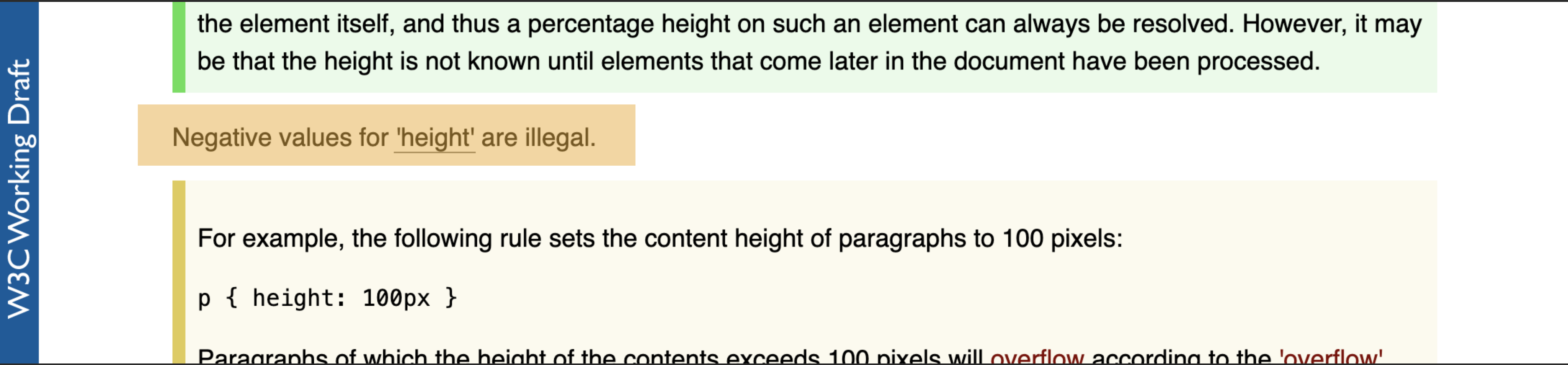


10.5 Content height: the 'height' property

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
Media:	visual

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.

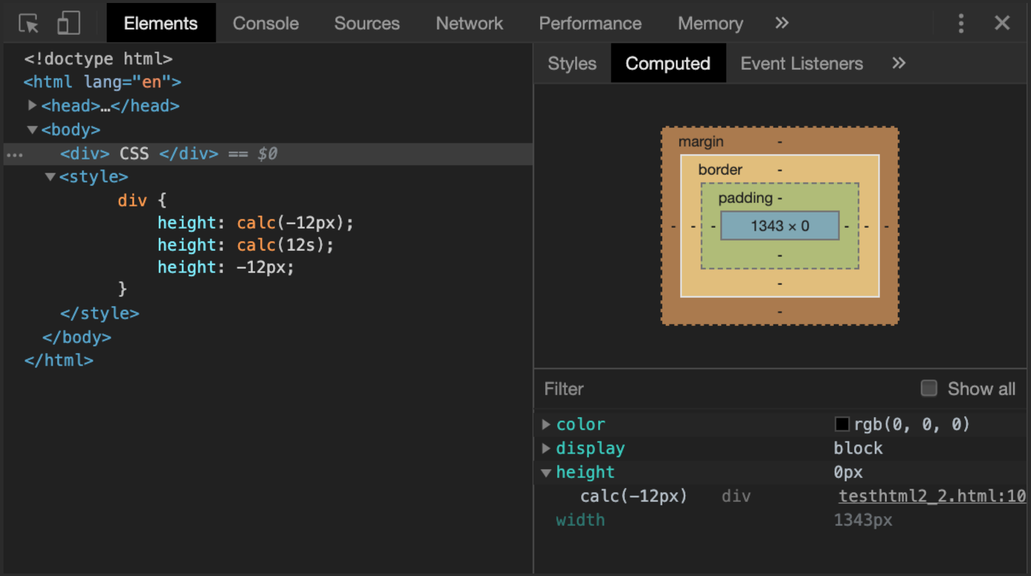
Negative values for 'height' are illegal.

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'

Несколько слов про calc



```
<!doctype html>
<html lang="en">
  <head>…</head>
  <body>
    <div> CSS </div> == $0
    <style>
          div {
              height: calc(-12px);
              height: calc(12s);
              height: -12px;
          }
    </style>
  </body>
</html>
```

```
color          rgb(0, 0, 0)
display        block
height         0px
  calc(-12px)   div     testhtml2_2.html:10
width          1343px
```

CSS Type Checking

```
~~height: -12px~~        // => Initial Value
height: calc(-12px)  // => 0px

~~height: -12s~~         // => Initial Value
~~height: calc(-12s)~~   // => Initial Value
```

Проверка финального типа, но не диапазона

Несколько слов про calc

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

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

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

Несколько слов про calc

```
font-size: 10px; // => CV=100px

    height: calc(100px - 5em); // => CV=calc(50px)

        height: inherit; // => CV=calc(50px)
```

Итого

- Есть алгоритм вычисления значения CSS-свойства, он описан в спеке.
- Он включает в себя каскад и не только
- Каскадные таблицы стилей не везде следуют спецификации каскада
- Спецификации читать вообще полезно
- Если спека сложная, понять ее помогут эксперименты в браузере
- Мы действительно можем повлиять на браузеры

Источники

- Спецификация css-cascade-4
- Специфичность не каскад
- Правильная шпаргалка по CSS-каскаду
- Почтовая рассылка www-style

София Валитова из ВКонтакте

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

Презентация сделана с помощью Shower