СТИЛИ (CSS)

CSS (Cascading Style Sheets - каскадные таблицы стилей) — язык, который позволяет прикреплять стиль к структурированным документам (HTML, XML, SVG). Отделяя стиль документов от содержимого документов, CSS упрощает создание веб-страниц. обслуживание сайтов и web- приложений.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. CSS3 по сути есть тот же CSS, но с набором новых аргументов, которые дают дополнительные возможности в плане различных эффектов.
Спецификация CSS3 не является частью спецификации HTML5. Эти два стандарта были разработаны отдельно друг от друга. Но W3C настоятельно рекомендует использовать HTML5 и CSS3 вместе, как часть современного веб-дизайна.

Внедрять CSS3 в веб-сайт можно, используя три подхода (три типа рекомендаций)):

1. Используйте то, что можно. Логично использовать возможности с уровнем поддержки на всех основных браузерах. Но не все возможности CSS3 входят в эту категорию.

2. Рассматривайте возможности CSS3 как усовершенствования.
Идея такого подхода – доработки и настройка страниц, причем эта доработка не повлияет на возможность просмотра основного содержимого и форматирования страницы в предыдущих версиях браузеров.

Стили, специфичные для конкретных браузеров. Префиксы

Для каждого браузера существует собственный префикс разработчика.

Префикс	*Браузер*
-moz-	Firefox
-webkit-	Chrome и Safari (оба браузера используют один движок визуализации)
-ms-	Internet Explorer
-o-	Opera

Если перед названием свойства стоит префикс, это означает, что данное свойство реализовано и будет применяться исключительно в указанном браузере. Все остальные браузеры данное свойство будут игнорировать. Префиксы браузеров усложняют разработку стилей для приложения, но их использование объясняют следующими причинами:

- включение в браузер экспериментальных свойств CSS, которые стандартом ещё не утверждены (так производители браузеров производят тестирование перед утверждением свойств CSS в стандарте);
- решения проблем кроссбраузерности приложений;
- для создания собственных свойств.

Пример использования префикса

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset='utf-8'>
```

```
<style>
    form {    width: 200px;        border: 2px solid green;   }
    form input,  form  select,    form  p
            {    display: block;    padding-left: 50%;
                 -moz-box-sizing: border-box;          /* только для  Firefox */
                  width: 100%;
            }
</style>
 </head>
<body>
<form>  <br> <input  type='text'  value ='text_1' >
                 <input  type='text'  value ='text_2' >
                 <select>  <option>опция 1</option> <option>опция 2</option>
                 </select>
            <p>  Демонстрация префикса  <b>-moz-box-sizing: border-box  </b> </p>
</form>
<body>
</html>
```

Первый набор стилевых параметров для элемента form задает ширину элемента пикселах и границу (толщина линии 2 пиксела, линия сплошная, цвет зеленый).

Второй набор стилевых параметров задан для элементов input, select, p – дочерних для элемента form.
Многоцелевое свойство **display** определяет, как элемент должен быть показан в документе.

Обычно используют три значения данного свойства display:
block – элемент отображается отдельным блоком с новой строки, стремится занять всю ширину родительского элемента;
inline – элемент отображается как встроенный, не располагается с новой строки;
none – элемент не отображается.

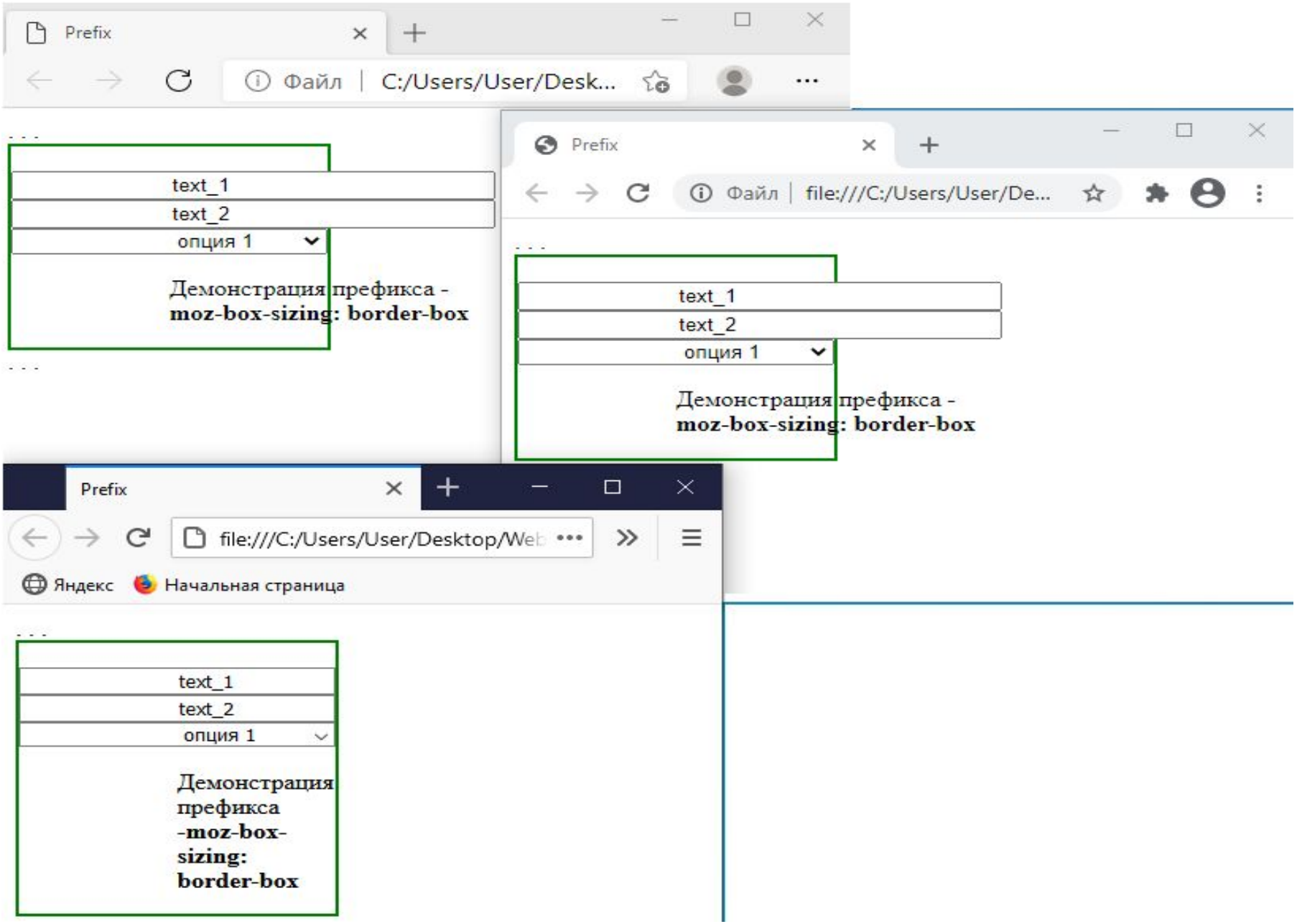
Свойство **padding** устанавливает внутренние отступы элемента, т.е. пространство между содержанием элемента и его границей. Отрицательные значения не допускаются. padding-left – отступ содержимого от левой границы элемента.

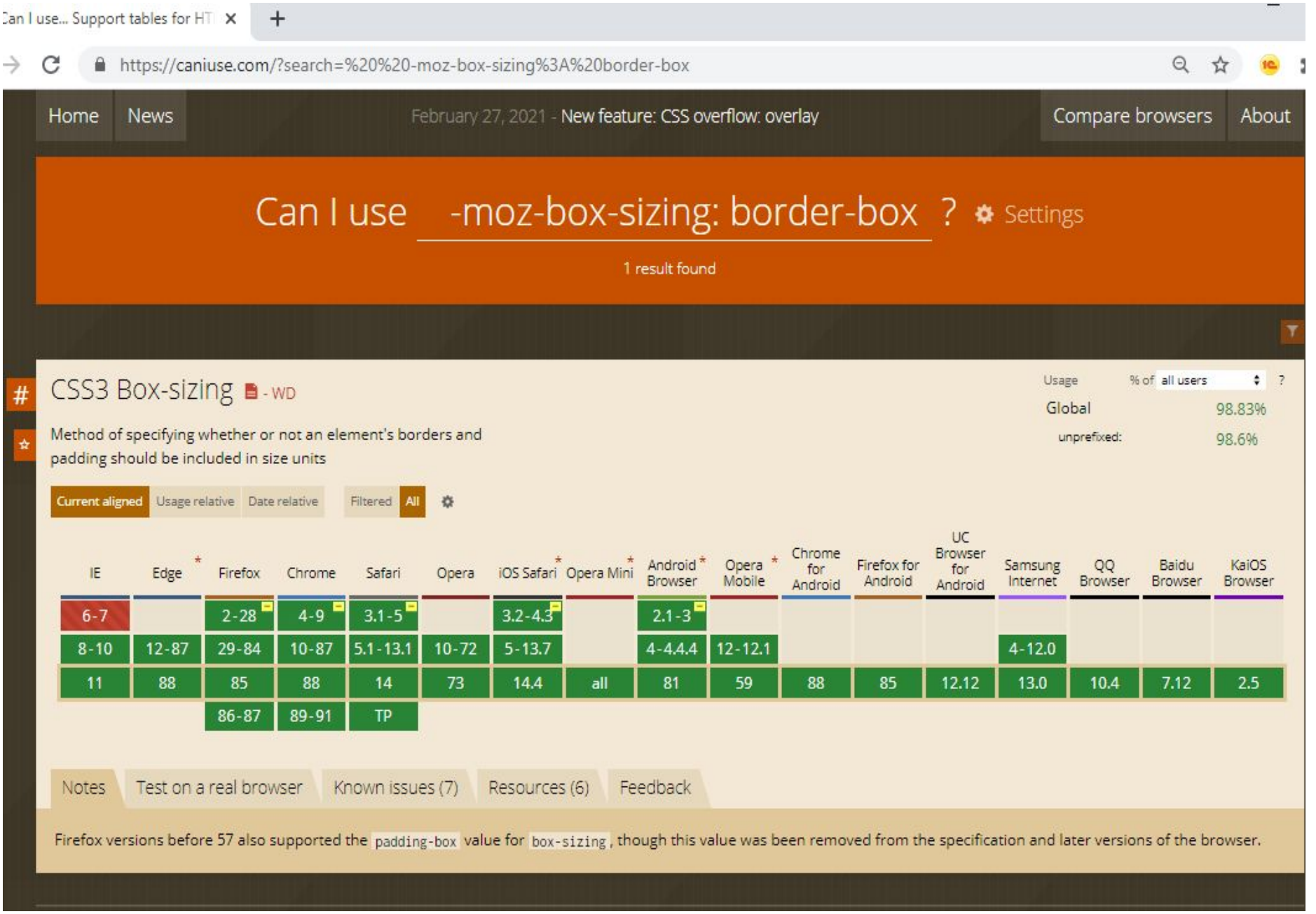
Свойство **box-sizing** принимает одно из двух значений – border-box или content-box (значение по умолчанию). В зависимости от выбранного значения браузер по-разному трактует значение свойств width / height. В первом случае браузер включают в размеры элемента width, height отступы содержимого от границ элемента, во втором случае – браузер интерпретирует width / height как размеры внутреннего содержимого.

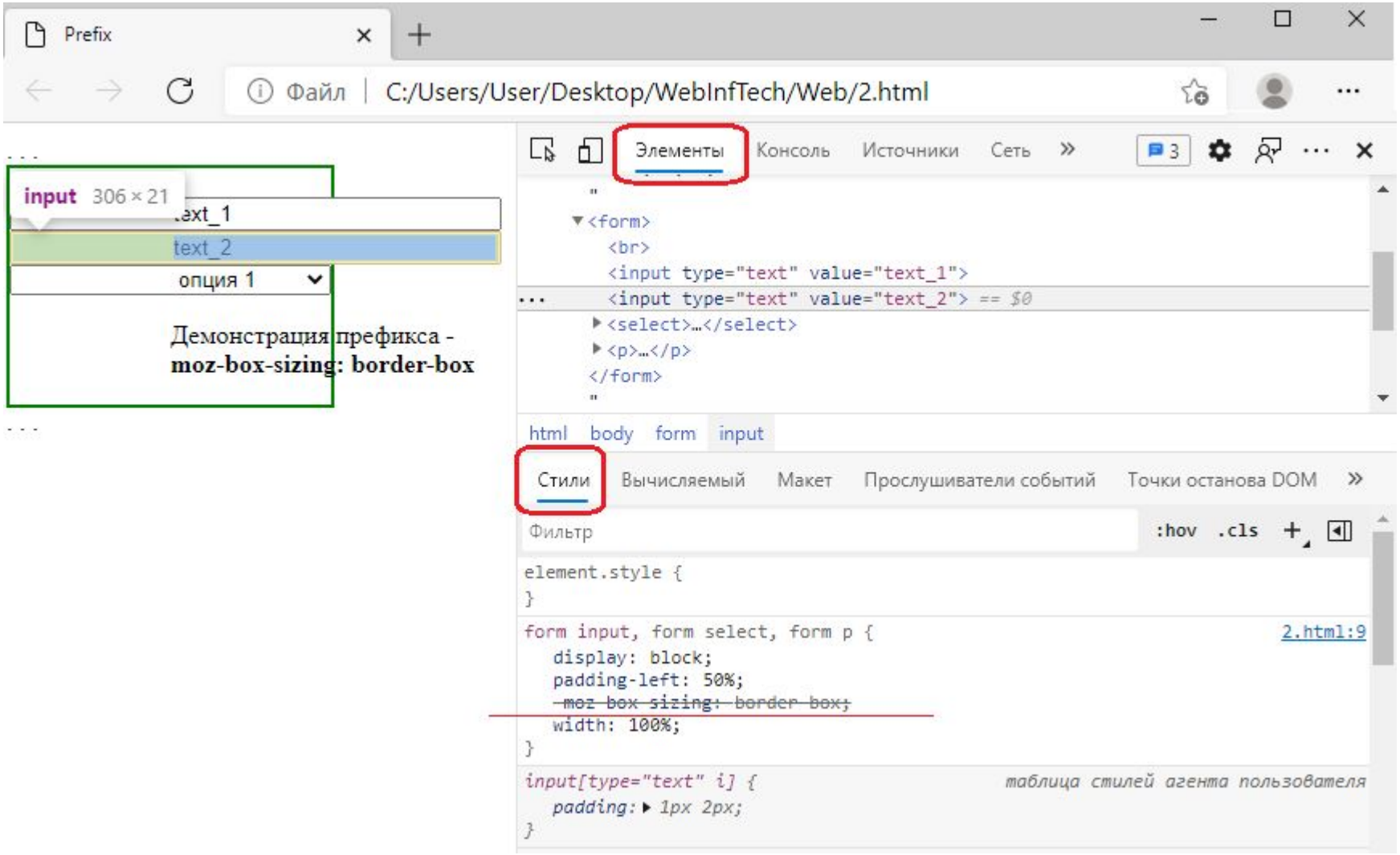
Свойство с префиксами следует располагать **до** этого свойства CSS без префикса. Потому, что, если в браузере уже реализовано оригинальное свойство (без префикса), то будет использоваться именно оно (т.к. располагается первым), а не экспериментальная версия с префиксом (занимает второе место).

Проверить поддержку того или иного стилевого свойства можно на сайте https://caniuse.com/

Отображение элемента формы в браузерах Edge, Google Chrome, Forefox,




Can I use... Support tables for HT
https://caniuse.com/?search=%20%20-moz-box-sizing%3A%20border-box
Home
News
February 27, 2021 - New feature: CSS overflow: overlay
Compare browsers
About
Can I use -moz-box-sizing: border-box ?
Settings
1 result found
CSS3 Box-sizing - WD
Method of specifying whether or not an element's borders and padding should be included in size units
Usage
% of all users
Global 98.83%
unprefixed: 98.6%
Current aligned
Usage relative
Date relative
Filtered
All
IE
Edge
Firefox
Chrome
Safari
Opera
iOS Safari
Opera Mini
Android Browser
Opera Mobile
Chrome for Android
Firefox for Android
UC Browser for Android
Samsung Internet
QQ Browser
Baidu Browser
KaiOS Browser
6-7
2-28
4-9
3.1-5
3.2-4.3
2.1-3
8-10
12-87
29-84
10-87
5.1-13.1
10-72
5-13.7
4-4.4.4
12-12.1
4-12.0
11
88
85
88
14
73
14.4
all
81
59
88
85
12.12
13.0
10.4
7.12
2.5
86-87
89-91
TP
Notes
Test on a real browser
Known issues (7)
Resources (6)
Feedback
Firefox versions before 57 also supported the padding-box value for box-sizing, though this value was been removed from the specification and later versions of the browser.



Браузер Edge отменяет версию -moz-.
Также действует и браузер Google Chrome

CSS-хаки

Это специальное использование CSS-свойств, понимаемое только определенными браузерами. Например, если определить класс (набор некоторый стилей)

```
@-moz-document url-prefix()
        {  .hackBlock  {  /* какие-то стили */.
                        }
        }
```

и некоторому элементу назначить этот класс, то **только браузер Firefox** будет применять данные стили, другие браузеры будут эти стили игнорировать.

Конкретного ответа, что использовать, какие средства и какие CSS-фрейморки, нет.

В основном для кроссбраузерной верстки используют префиксы, это увеличивает объем кода, но код остается быть валидным и понятным.

Применение CSS-хаков являются нежелательным способом. Использование их приводит к плохой читабельности кода, невалидности, возможно некачественной поддержке в будущем.

Задание стилей

1. Внешние страницы стилей, это отдельные css-файлы, в том числе интернет-ресурсы, их может быть несколько.

```
<head>
   <link rel="stylesheet"  href="css/style.css">
   <link rel="stylesheet"
         href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
```

2. Внутренние стили в элементе <style>.

Правило **@import** позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
 @import url (mobile.css);
                              /* стили  отдельных элементов*/
  селектор: { параметр : значение;  параметр : значение;   . . .  }
  селектор: { параметр : значение;  параметр : значение;   . . .  }
. . .
  </style>
```

3. Стили, задаваемые атрибутом style в теге элемента.

<*имя_тега* style = "*параметр=значение; параметр=значение;* . . ." . . . >

Селекторы

Представляют структуру веб-страницы. Селекторами могут быть:

– элементы (имена элементов),
– классы;
– идентификаторы,
– псевдоклассы,
– псевдоэлементы.

Универсальный селектор соответствует любому HTML-элементу.

```
* { margin: 0;              /*  обнулит внешние отступы для всех элементов */
  }
```

Селектор элемента позволяют форматировать все элементы данного типа на всех страницах сайта/ приложения, например,

```
h1 { font-family: Lobster, cursive;          }
```

Это общий стиль всех заголовков h1, если этот стиль включен в страницу css, доступную все страницам приложения.

Класс - поименованная группа стилевых параметров, применяемая к различным элементам страницы или на разных страницах приложения.

Пример: любые элементы, помеченные атрибутом class="p1", будут иметь стиль

```
.p1 {  text-transform: uppercase;
       color: lightblue;
    }
```

Селектор идентификатора позволяет форматировать конкретный элемент с заданным уникальным идентификатором. Пример стиля для элемента с id="id1"

```
#id1     {  width: 300px;          float: left;
         }
```

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера.

Применение стилей к элементам li, которые являются потомками элементов ul,

```
ul  li   { text-transform: uppercase;
         }
```

Применение стилей к потомкам-ссылкам внутри любого элемента с классом first, этот элемент в свою очередь должен быть потомком элемента <p>;

```
p   .first  a   {color: green;
                }
```

Применение стилей к потомкам-ссылкам внутри любого элемента с классом first

```
.first  a  { color: green;
           }
```

Выражение *селектор[атрибут]* или *селектор[атрибут="значение"]* означает выбор элементов, определяемых селектором и содержащих указанный атрибут или указанный атрибут с конкретным значением. Пример: выбора картинок, название которых содержит слово flower

```
Img [title="flower"]
```

Возможны варианты, когда стиль следует применить к тегу с определённым атрибутом, при этом частью его значения является некоторый текст.
а) устанавка стиля для элемента, если значение атрибута тега начинается с определенного текста: [*атрибут*^="*значение*"]
б) устанавка стиля для элемента, если значение атрибута заканчинается определенным текстом [*атрибут*$="*значение*"]
b) устанавка стиля для элемента, если значение атрибута содержит определенный текст в любом месте: [*атрибут**="*значение*"]

Пример выбор всех ссылок, название которых содержит "web".

```
a [ href * = "web" ]
```

Псевдоклассы

Это классы, фактически не прикрепленные к HTML-тегам. Псевдоклассы характеризуют элементы со следующими свойствами:

:link — не посещенная ссылка;
:visited — посещенная ссылка;
:hover — любой элемент, по которому проводят курсором мыши;
:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;
:active — элемент, который был активизирован пользователем;
:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;
:invalid — поля формы, содержимое которых не соответствует указанному типу

:disabled — заблокированные поля форм, т.е. находящиеся в неактивном состоянии;
:in-range — поля формы, значения которых находятся в заданном диапазоне;
:out-of-range — поля формы, значения которых не входят в установленный диапазон;
:lang() — элементы с текстом на указанном языке;
:target — элемент с символом #, на который ссылаются в документе;
:checked — выделенные (выбранные пользователем) элементы формы.

Селекторы структурных псевдоклассов выбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

:nth-child (odd) — нечетные дочерние элементы;
:nth-child (even) — четные дочерние элементы;
:nth-child (3n) — каждый третий элемент среди дочерних;
:nth-child (3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);
:nth-child (n+2) — выбирает все элементы, начиная со второго;
:nth-child (3) — выбирает третий дочерний элемент;
:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, но начиная с последнего, в обратную сторону;
:first-child — выбирает только самый первый дочерний элемент;
:last-child — последний дочерний элемент;
:only-child — выбирает элемент, являющийся единственным дочерним элементом;
:empty — выбирает элементы, у которых нет дочерних элементов;
:root — выбирает элемент, являющийся корневым в документе(элемент

```
<!DOCTYPE html>
<html>
 <head>  <meta charset='utf-8'>  <title>Color</title>
<style>
   .r    { margin-left:3vw;   display: block;  width: 300px }
   .r1   { margin-top:3vh;   }
   .e    {  width:60px; height:60px;  text-align:center; line-height: 50px; display: inline-block; }
   .c1   {   background: radial-gradient( #fff700, #d8b62b)}
   .c2   {   background: radial-gradient( #fefeff,  #e0e0e0)}
                              /*  задание стилей с использованием  псевдоклассов */
 body  div:nth-child(4)      { margin-left:3vw;   display: block;  width: 300px;    margin-top:3vh   }
 body  div:nth-child(n+5)    { margin-left:3vw;   display: block;  width: 300px;}
 body  div:nth-child(n+4)   div { width:60px; height:60px;    text-align:center;  line-height: 50px;
                                        display: inline-block;  }
                        /* заданы стили для :
                           для  4-го дочернего элемента div элемента body,
                           для  всех дочерних элементов div, начиная с 5-го,
                           для всех элементов div, являющихся дочерними  для  элементов div в
body ,
                             начиная с 4-го,
                        */
</style>
 </head>
<body>
```

```
<div class='r  r1'> <div class="e  c1">11</div>
                    <div class="e  c2">12</div>
                    <div class="e  c1">13 </div>  </div>
<div class='r'>     <div class="e  c2">21</div>
                    <div class="e  c1">22</div>
                    <div class="e  c2">23</div>   </div>
<div class='r'>     <div class="e  c1">31</div>
                    <div class="e  c2">32</div>
                    <div class="e  c1">33</div>   </div>

<div >      <div class="c1">11</div>
            <div class="c2">12</div>
            <div class="c1">13 </div>    </div>
<div >      <div class="c2">21</div>
            <div class="c1">22</div>
            <div class="c2">23 </div>    </div>
<div >      <div class="c1">31</div>
            <div class="c2">32</div>
            <div class="c1">33 </div>     </div>
<body>
</html>
```

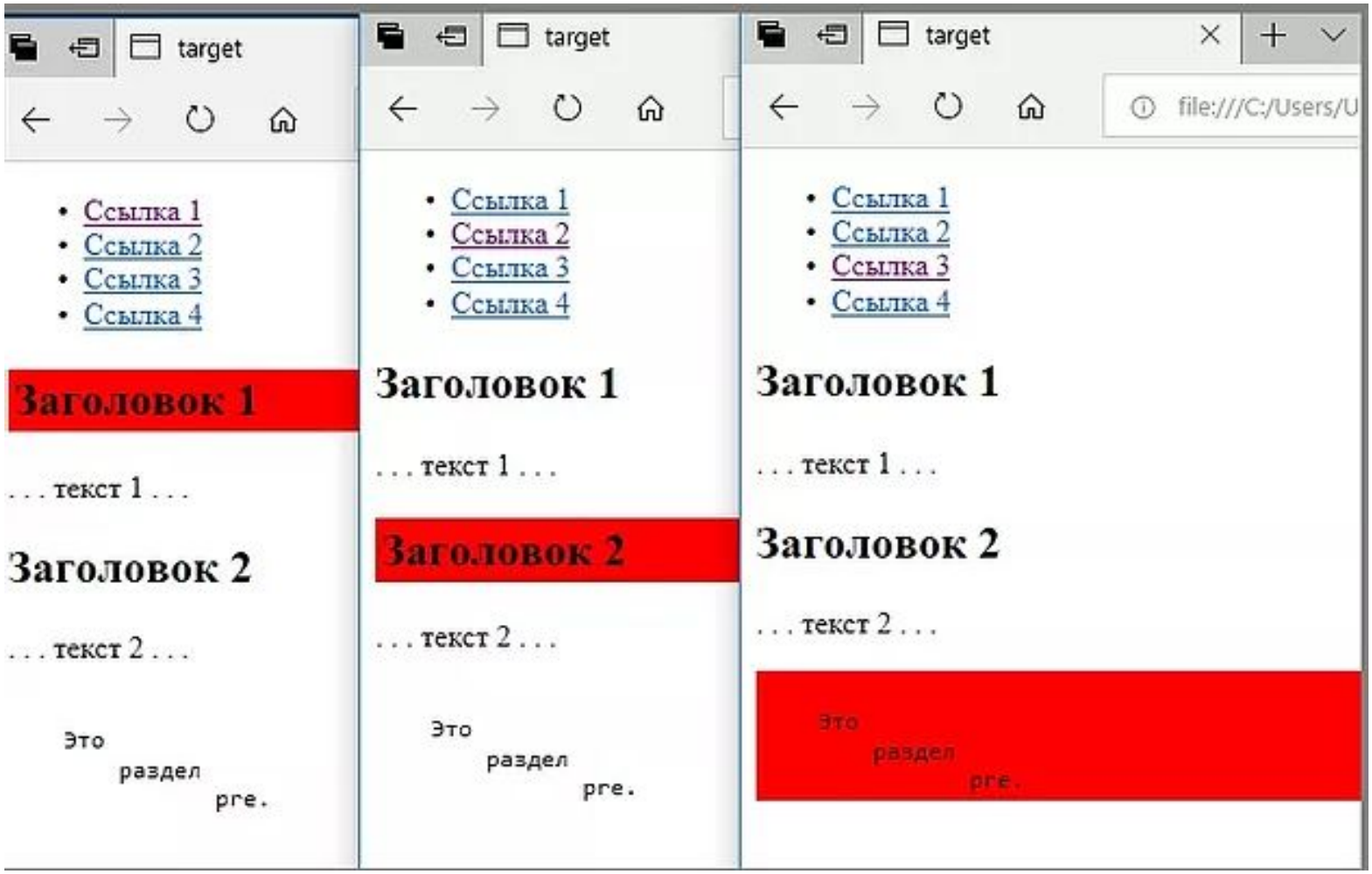
Пример 2. Использования псевдокласса target

```
<!DOCTYPE html>
<html>
 <head> <meta charset="utf-8">      <title>target</title>

<style>  pre:target,  h2:target   { background: #f00;
                                   padding: 3px; }
    /* если будет выбрать (активизирован) элемент pre или
h2, то этот  элемент  получит указанные стилевые
параметры   */
</style>
 </head>
 <body>

  <ul>     <li><a href="#id1"> Ссылка 1</a></li>
           <li><a  href="#id2"> Ссылка 2</a></li>
           <li><a  href="#id3"> Ссылка 3</a></li>
           <li><a  href="#id4"> Ссылка 4</a></li>    </ul>

  <h2 id="id1">Заголовок 1</h2>    <p>. . .  текст  1 . . . </p>
  <h2 id="id2">Заголовок 2</h2>    <p>. . .  текст  2 . . . </p>
  <pre id="id3">
  Это
      раздел
           pre. </pre>
   . . .
</body>
```

Выбор ссылки – активизация соответствующего элемента. Суть активизации при наличии псевдокласса target – примене-ние стилей к элементу.

Ссылки могут быть внешние (на другой документ), внутренние (на элемент в текущем до-кументе). В данном примере параметры href ссылок указывают на элементы h2, pre и несуществующий элемент.

При активизации нового элемента или при обращении по несуществующей ссылке теку-щий элемент деактивизируется, т.е. теряет стили с псевдо-классом target.


target
Ссылка 1
Ссылка 2
Ссылка 3
Ссылка 4
Заголовок 1
. . . текст 1 . . .
Заголовок 2
. . . текст 2 . . .
Это
раздел
pre.
target
Ссылка 1
Ссылка 2
Ссылка 3
Ссылка 4
Заголовок 1
. . . текст 1 . . .
Заголовок 2
. . . текст 2 . . .
Это
раздел
pre.
target
file:///C:/Users/U
Ссылка 1
Ссылка 2
Ссылка 3
Ссылка 4
Заголовок 1
. . . текст 1 . . .
Заголовок 2
. . . текст 2 . . .
Это
раздел
pre.

Универсальные атрибуты

Универсальные (глобальные) атрибуты применяются практически ко всем элементам, поэтому собираются в отдельную группу, чтобы не повторять их для всех тегов.

accesskey —позволяет получить доступ к элементу с помощью заданного сочетания клавиш. Атрибут зависит от платформы (ОС), браузера и версии браузера.
class —задает имя класса, которое позволяет связать тег со стилевым оформлением.
dir/direction —задает расположение текста - слева(dir="ltr") или справа (dir="rtl") , по умолчанию dir="ltr".
hidden —скрывает содержимое элемента от просмотра.
id —указывает уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты..
lang —браузер использует значение параметра для правильного отображения некоторых национальных символов.
style —применяется для определения стиля элемента с помощью правил CSS.
tabindex —устанавливает порядок получения фокуса при переходе между элементами с помощью клавиши Tab.
title —описывает содержимое элемента в виде всплывающей подсказки.
contenteditable — сообщает, что элемент доступен для редактирования (изменения) пользователем при отображении в браузере.

Пример использования атрибута accesskey (фрагмент html-кода)

```
<br> 1-й элемент ввода
 <input type="text"      accesskey="1"  onfocus="alert ('1-й элемент ввода получил фокус');
                                                this.blur(); console.log('-1-'); " >
<br> 2-й элемент ввода
<input type="text"      accesskey="2"  onfocus="alert ('2-й элемент ввода получил фокус');
                                               this.blur(); console.log('-2-'); " >
<br> 3-й элемент ввода
 <input type="text"      accesskey="3"  onfocus="alert ('1-й элемент ввода получил фокус');
                                                this.blur(); console.log('-3-'); " >
```

Три элемента ввода имеют атрибут accesskey. Элементы получают фокус ввода либо щелчком левой клавишей мыши по элементу (стандартный способ) либо нажатием комбинаций клавиш **Alt+1, Alt+2, Alt+3** в браузерах Edge, Google Chrome. Для браузера Mozilla следует использовать **Alt+Shift+1, Alt+ Shift+2, Alt+ Shift+3.**

onfocus – событие получения фокуса ввода элементом.

Вызов this.blur(); удаляет фокус с элемента, ключевое слово **this** представляет текущий объект, к которому применяется метод blur(). Метод **console.log(. . .)** выводит в окно отладки текстовое сообщение-аргумент.

Пример использования атрибутов spellcheck, contenteditable, title, contextmenu

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">   <title>Универсальные атрибуты</title>
```

```
<style>
 body { margin-left: 3vw; font-family: Calibri; font-size: 1.2em; text-align: center       }
 .rotateleft { transform: rotate(-90deg); }
 .rotateright { transform: rotate(45deg); }
   div {  display: inline-block;  border: thin solid gray;
          width: 35%; min-width: 250px;  max-width: 36%;
          padding: 2vh 3vh;   margin: 1vw 2vw 2vw 1vw;          border: thin solid gray      }
   .image { width: 100% }
    textarea {  background: #e0e0e0; width: 90%;  height: 100px;
                font-size: 1.2em;  padding: 1vw;  text-align: left          }
</style>
</head>
<body>
   <h2 title="Текст заголовка можно изменять в процессе выполнения"
     spellcheck="true" contenteditable="true"> Универсальные атрибуты </h2>
   <div>  <img src="3.jpg" alt=""  contextmenu="myMenu" id="pic" class="image">
   </div>
   <div spellcheck="true" contenteditable="true" >
      Повторн. выбор пункта меню отменяет преобразование
     <menu type="context" id="myMenu">
        <menuitem onclick="pic.classList.toggle('rotateright') ">   a) Повернуть на 45º ПоЧС
        </menuitem>
       <br>
        <menuitem onclick="pic.classList.toggle('rotateleft')">     b) Повернуть на 90º ПрЧС
        </menuitem>
     </menu>
     <textarea>Здесь можно вводить произольный текст, для  которого будет выполняться проверка
орфографии
     </textarea>
   </div>
</body>
</html>
```

Анализ html- текста

Указывать длину в CSS можно в разных единицах: пункts (pt), дюйм (in), пикселы (px) и другие **абсолютные** единицы.
Но размеры шрифтов лучше задавать в em, точнееиспользовать размер шрифта по умолчанию для устройства, поскольку это наиболее удобный для читателя размер. font-size: 1.2em означает, что шрифт в 1.2 раза крупнее основного размера шрифта по умолчанию.

Универсальные атрибуты **spellcheck="true"** **contenteditable="true"** позволяют редактировать любое текстовое содержимое (крсной тонкой линией подчеркнуты орфографические ошибки).
Универсальный атрибут **id="pic"** позволяет обращаться в скриптах к данному элементу идентификатор pic.

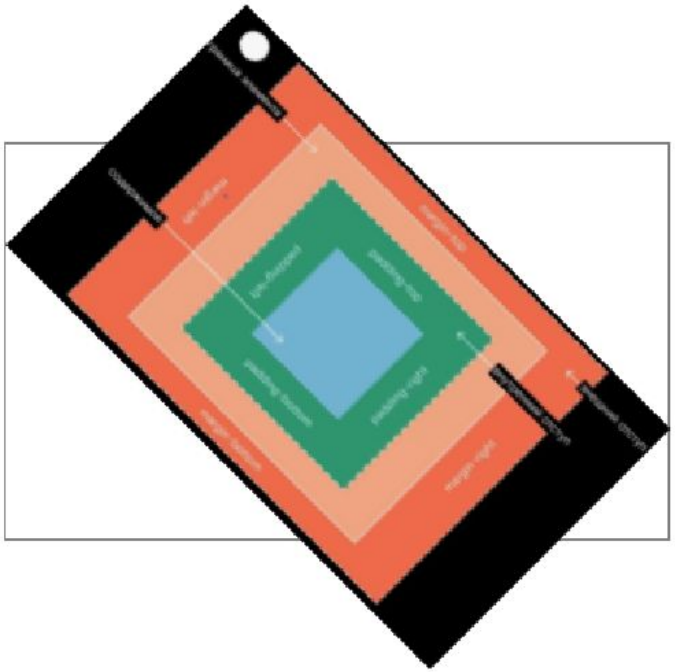
onclick="**pic.classList.toggle('rotateright')**" - задание обработчика события "клик по элементу". Тело обработчика события представлено текстовой строкой, содержащей javascript- код. В данном случае этот код получает список классов, примененных к лементу pic. Если в списке есть класс **rotateright,** то он удаляется из списка, в противном случае добавляется, что вызывает поворот элемента. Таким образом, каждый нечетный клик добавляет класс, вызывающий поворот элемента, каждый четный клик убирает этот класс, возвращая элемент в исходное состояние.

Универсальные атрибуты **menu, menuitem** позволяет создавать меню из нескольких пунктов. В данном случае для пунктов меню задан обработчик onclick.

Универсальные атрибуты

localhost:64583/3.html

Универсальные атрибуты



Повторн. выбор пункта меню отменяет преобразование

*Повернуть на 45º ПоЧС
*Повернуть на 90º ПрЧС

Здесь можно вводить произольный текст, для которого будет выполняться проверка орфографии.

Проверка правописания в браузерах обычно отключена по умолчанию или отсутствует.

Включение проверки правописания в Edge (аналогично в Google, в настройках надо