ВЕНДОРНЫЕ ПРЕФИКСЫ.

Введение. Виды. Порядок записи. Варианты использования. Сервисы для работы.

Введение

Над развитием CSS работает Рабочая группа CSS, или коротко CSSWG (CSS Working Group). Они собирают потребности разработчиков сайтов и описывают возможности CSS в новых модулях. Получившийся документ называется спецификацией. В ней содержится описание того, как новое свойство должно работать.

Рабочим группам для подготовки стандартов нужна обратная связь от разработчиков. А разработчикам в свою очередь нужен доступ к новым возможностям, чтобы их протестировать.

Браузерные (вендорные) префиксы были придуманы как раз для того, чтобы дать возможность разработчикам свободно экспериментировать с новыми возможностями.

До момента, пока не стабилизируется спецификация или пока не будут написаны все тесты, можно использовать эту новую возможность в тестовом режиме, с вендорным префиксом.

Каждый браузер — это отдельный вендор (от англ. vendor — продавец) услуг просмотра сайтов, интернета. Отсюда и слово «вендорный». Буквально это означает, что существуют некие отдельные префиксы — они же приставки — которые работают в конкретном браузере — вендоре.

Итак, **вендорные префиксы** — это приставки перед свойствами, селекторами, функциями или другими сущностями в CSS, позволяющие браузерам внедрять

экспериментальные возможности до того, как они полностью стандартизированы и готовы к использованию.

Но, если экспериментальные технологии начинают широко использоваться в производстве, у рабочей группы не остается иного выхода, кроме как продолжать придерживаться ранней, экспериментальной версии технологии, чтобы её изменение не сломало уже существующие продукты. Очевидно, это полностью сводит на нет преимущества, которые способно принести тестирование ранних версий стандартов реальными разработчиками.

При этом, используя только префиксы, к уже имеющемуся коду придется возвращаться и обновлять его каждый раз, когда в новом браузере появляется поддержка той самой возможности CSS.

Решением стало использовать все возможные префиксы, в конце заодно добавляя версию без префикса, для того чтобы гарантировать правильную обработку кода в будущем.

Очевидно, что повторять каждое объявление до пяти раз неудобно, а итоговый код получается не приспособлен для нормальной поддержки.

Поскольку разработчики использовали версии без префиксов в качестве гарантии работоспособности своего кода в будущем, изменять их стало невозможно. По сути, процесс раннего тестирования зашел в тупик с полуготовыми ранними спецификациями, допускающими лишь незначительные изменения.

Это привело к тому, что сегодня браузерные префиксы применяются для новых экспериментальных реализаций достаточно редко. Вместо этого экспериментальные возможности включаются с помощью конфигурационных

флагов в настройках браузера, что предотвращает использование их разработчиками в производственном окружении, — ведь вы не можете заставлять пользователей менять локальные параметры для того, чтобы сайт на их машинах отображался правильно.

При этом в сети огромное количество примеров кода с префиксами, в том числе не актуальными. Поэтому в своей лекции поговорим о том, как же использовать префиксы, если это действительно необходимо. Как понять, актуальные ли они в том фрагменте кода, который вы изучаете и нужно ли их копировать в свой проект.

Виды

Основные браузеры используют следующие префиксы:

-webkit- — Safari, Chrome, Opera 15+, почти все браузеры iOS, включая Firefox для iOS; практически любой браузер на основе WebKit или Chromium.

-moz- — Firefox и браузеры на движке Gecko.

-o- — Opera 12 и раньше, на движке Presto.

-ms- — Internet Explorer и старый Microsoft Edge 12–18.

Полный список префиксов можно посмотреть в википедии - https://en.wikipedia.org/wiki/CSS_hack#Browser_prefixes

Порядок записи

Очень важно указывать сущности (свойства, селекторы, директивы и так далее) с вендорными префиксами выше, чем без префиксов.

Это нужно для того, чтобы браузер использовал самую последнюю стабильную реализацию. Если браузер уже поддерживает фичу без префиксов, то применится последнее

из перечисленных. А если нет, то сработает подходящая запись из кода выше.

Варианты использования

В CSS существует много разных сущностей: селекторы и псевдоэлементы, свойства и их значения, функции, директивы. Для использования нововведений в любой их этих сущностей можно применять вендорные префиксы.

Директивы

Например, вендорный префикс для директивы — @keyframes при настройке анимации.

```
@-webkit-keyframes animation {
    0% { right: 0; }
    100% { right: 100%; }
  }

  @keyframes animation {
    0% { right: 0; }
    100% { right: 100%; }
  }
```

Написать директивы @-webkit-keyframes и @keyframes через запятую, чтобы не дублировать их содержимое, не получится.

Псевдоклассы

В CSS достаточно большой набор интересных псевдоклассов, которые не всегда имеют полную поддержку браузеров. Например, стилизовать **placeholder** в поле ввода можно при помощи такого кода:

```
input::-webkit-input-placeholder { color: #350454; }
input::-moz-placeholder { color: #350454; }
input:-ms-input-placeholder { color: #350454; }
input::-ms-input-placeholder { color: #350454; }
```

```
input::placeholder { color: #350454; }
```

Как и в случае с директивами, префиксы в псевдоэлементах тоже приводят к дублированию кода.

Свойства

Например, новое свойство **background-clip** со значением **text**. Позволяет обрезать фоновое изображение по форме текста на переднем плане. Без префикса поддерживается только в браузерах Mozilla и Safari

```
.text-gradient {
    color: transparent;
    -webkit-background-clip: text;
    background-clip: text;
    background-image: linear-gradient(...);
}
```

Значения свойств

Бывает и так, что свойство старое, а вот значение для него новое, экспериментальное. Тогда к нему будет добавлен префикс.

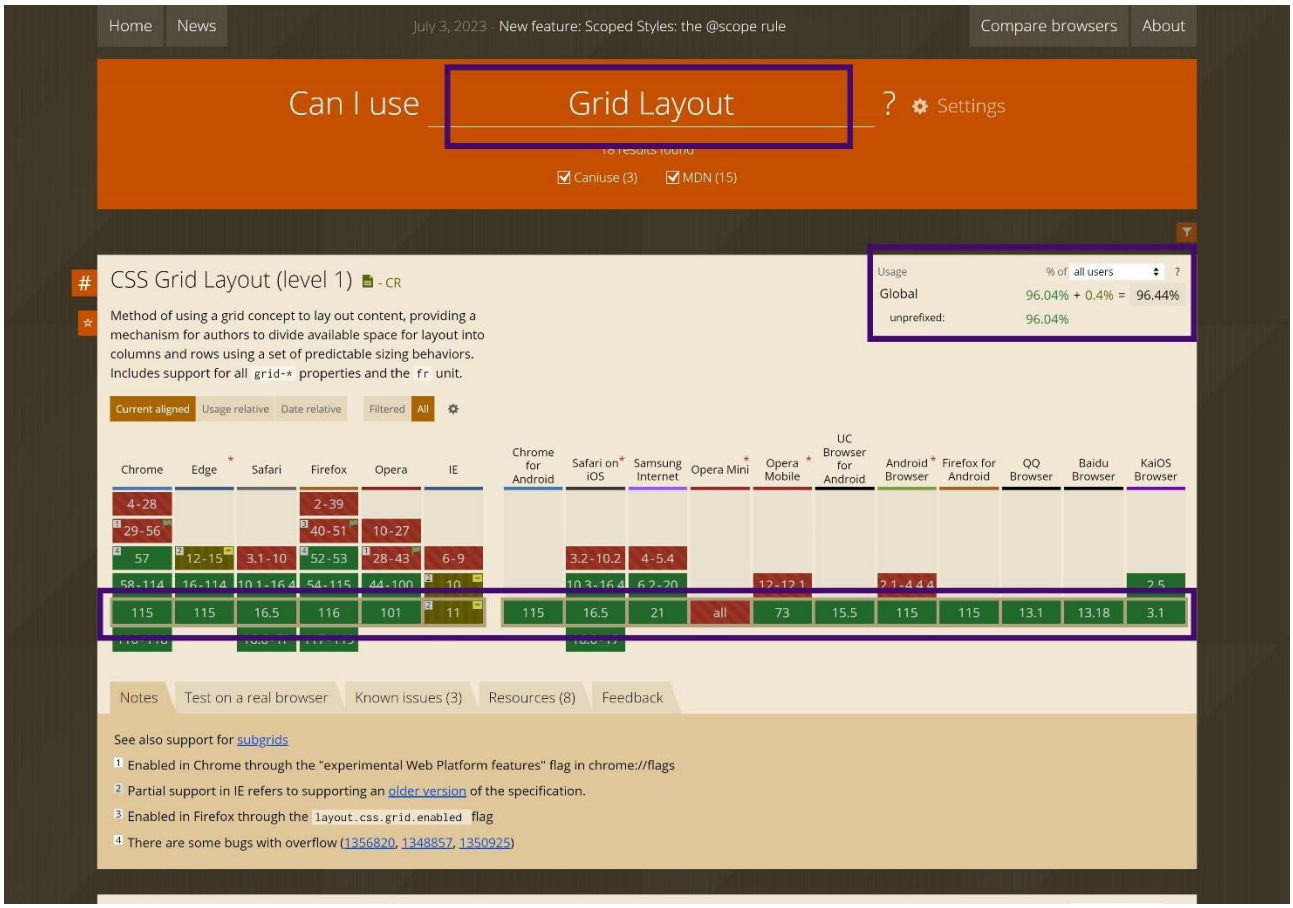
Селекторы

Случается так, что в процессе внедрения возможность меняется. Изначально планировалось, что функция будет называться в одном варианте, потом во втором, а в итоге в третьем. Чтобы селектор сработал везде, даже в старых браузерах, где функция называлась иначе, нужна будет запись с вендорными префиксами на каждый из вариантов.

Сервисы для работы

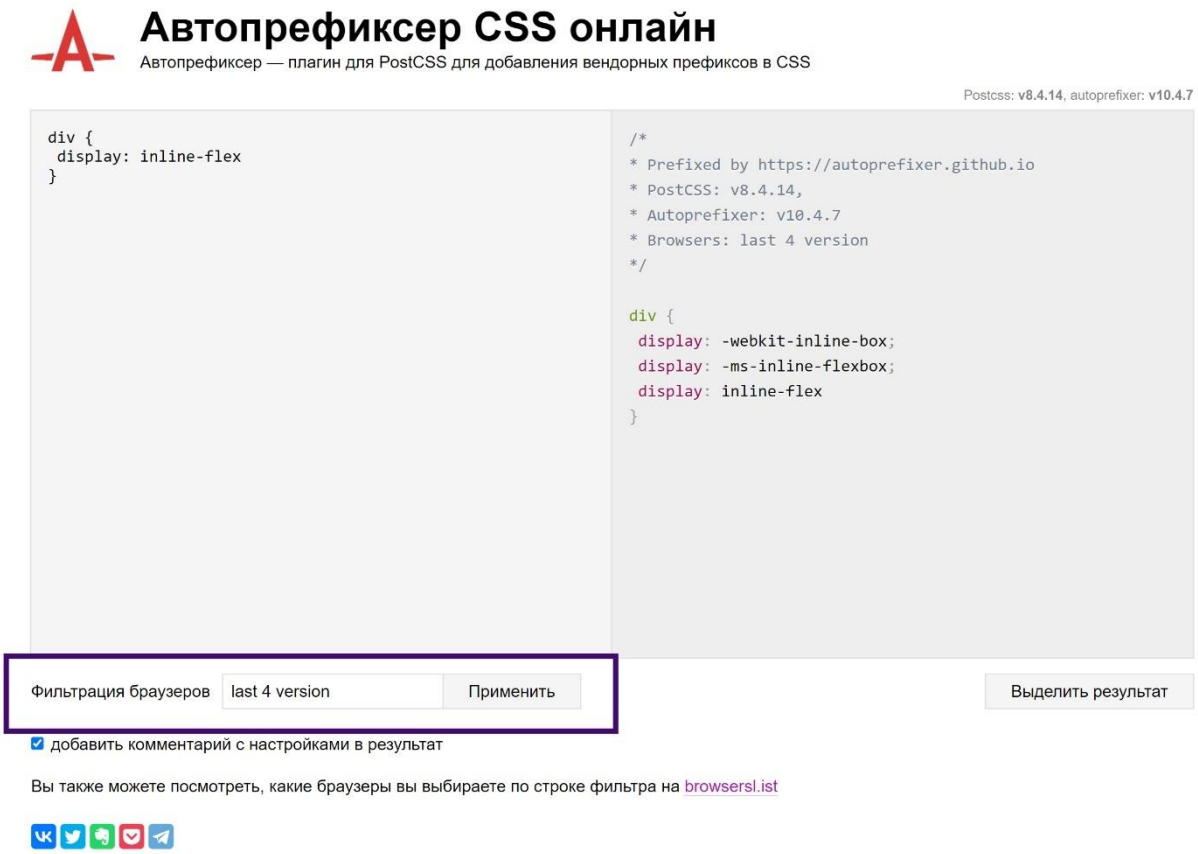
Перед использованием префиксов важно проверить поддержку свойства браузерами.

Самый простой способ проверить поддержку свойства — найти его на сайте Can I use.



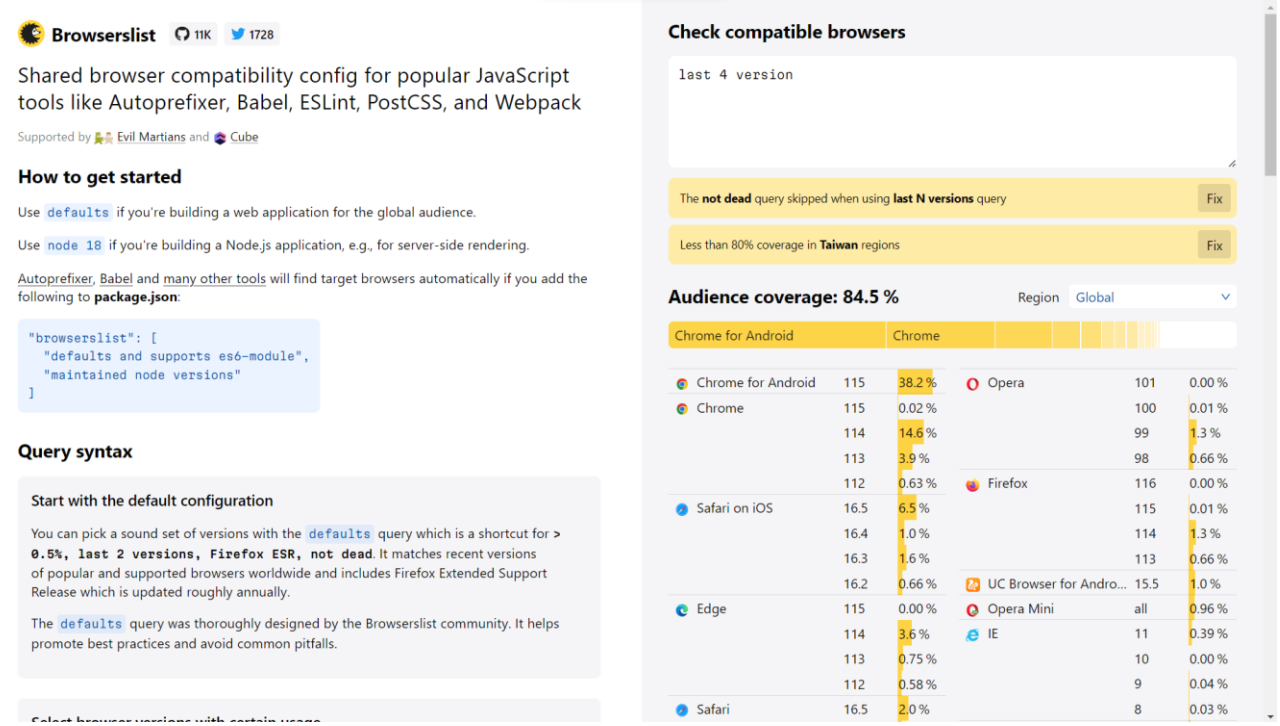
Зеленые версии – полная поддержка, коричневые – частичная поддержка через префиксы, красные – свойство не поддерживается. На вкладках ниже дополнительная информация, далее могут быть описаны варианты свойства и их поддержка.

Для ускорения процесса написания кода можно использовать инструменты автоматизации. Самым популярным из них на сегодня является Автопрефиксер. Он использует базу данных из Can I Use… для определения, какие префиксы необходимо добавить к коду без браузерных префиксов, и компилирует его локально, как препроцессор;



Вы можете попробовать его онлайн: вставляете ваш CSS, указываете, какие браузеры должны поддерживаться, и получаете код с проставленными префиксами там, где это нужно.

Подробная информация о выборе версий браузеров по ссылке https://browsersl.ist/ Сервис подскажет вам какой указать фильтр для максимального покрытия браузеров.



Еще один вариант автоматизации – использование плагина **Autoprefixer** для VSCode.

Установите плагин, выделите свой код и на командной панели (Ctrl+Shift+P) введите команду **Autoprefixer: Run**

Настроить отдельные параметры для списка браузеров можно в файле **settings.json.**

```
  "autoprefixer.options": {"браузеры":  ["последние 4 версии", "ie> = 9", "> 5%"]}
```

Плюс ко всему, системы сборки проектов, которыми пользуются серьезные разработчики, могут добавлять префиксы на этапе сборки без вашего участия, таким образом исходный код будет чист от повторений.

ВАЖНО! Не добавляйте вендорные префиксы без веской на то причины. Просто проверьте новое для вас свойство на предмет поддержки браузеров и подумайте так ли важна вам поддержка, например старых версий IE.

ЗАДАНИЕ № 33 ПРИМЕР.

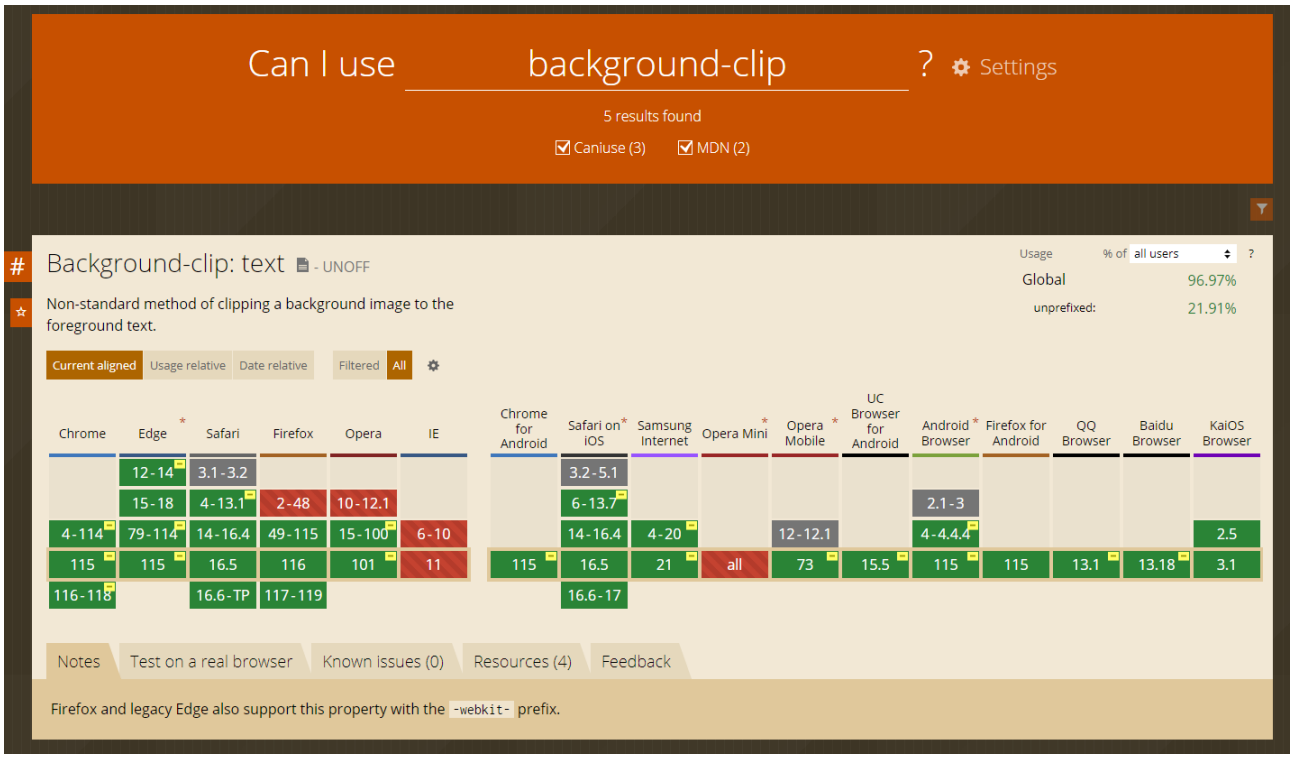
В качестве примера представляем фрагмент сайта LiveTV.

С помощью префиксов оформим часть текста градиентной заливкой.

Технически все чуть сложнее. Цвет текста на самом деле прозрачный, а блок имеет градиентную заливку.

С помощью свойства **background-clip** обрезаем заливку по форме текста.

Проверяем поддержку свойства через ресурс caniuse.com



Видим, что поддержка свойства браузерами кроме Safari и Firefox осуществляется через префикс -webkit-.

Можно уточнить список необходимых префиксов через ресурс Autoprefixer

Автопрефиксер CSS онлайн

Автопрефиксер — плагин для PostCSS для добавления вендорных префиксов в CSS

Postcss: **v8.4.14**, autoprefixer: **v10.4.7**

```
background-clip: text;
```

```
-webkit-background-clip: text; background-clip: text;
```

А значит код наших стилей будет выглядеть следующим образом

```
.text-gradient {
    font-weight: 700;
    font-size: 40px;
    color: transparent;
    -webkit-background-clip: text;
    background-clip: text;
    background-image: linear-gradient(90deg, #0052d4, #fff);
}
```

Закомментируйте префикс и проверьте как будет выглядеть страница. Получается, если браузер не поддерживает background-clip через префикс, то текст пользователь не увидит.

Чтобы исправить этот недостаток можно написать два набора стилей: для устройств, поддерживающих background-clip и для всех остальных.

Сделаем это с помощью @supports.

Без подробностей, правило позволяет вам указывать объявления CSS , которые зависят от поддержки браузером

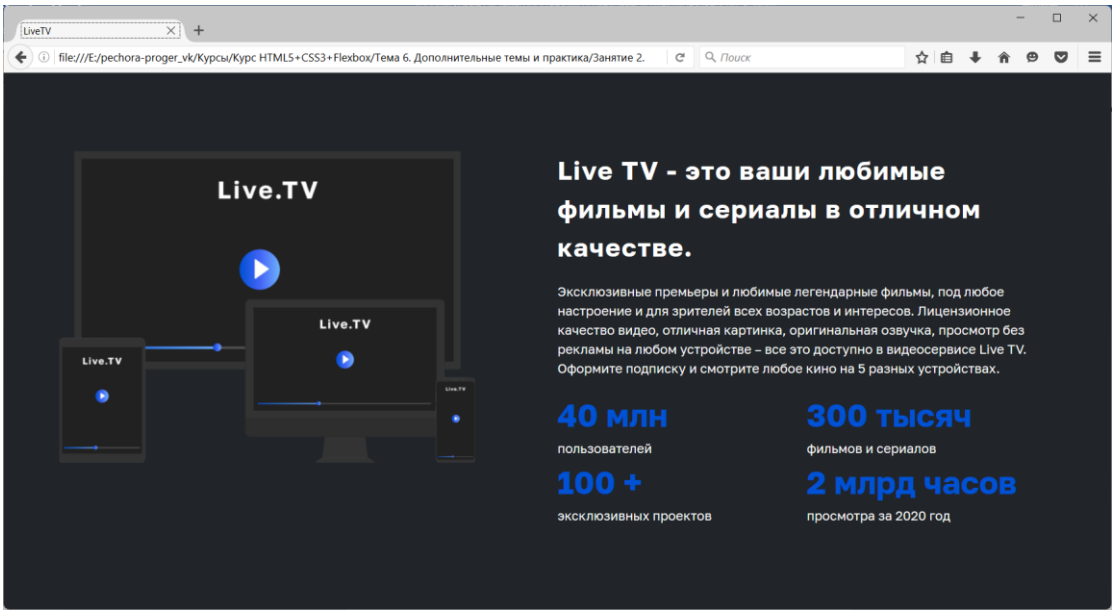
функций CSS. Использование этого правила обычно называют *запросом признаков*.

Для проверки

```
.text-gradient {
    font-weight: 700;
    font-size: 40px;
    color: #0052d4;
}

@supports (-webkit-background-clip: text){
    .text-gradient {
        color: transparent;
        -webkit-background-clip: text;
        background-clip: text;
        background-image: linear-gradient(90deg, #0052d4,
#fff);
    }
}
```

Для проверки я скачал версию Firefox 48, который свойство через префикс не поддерживает. Вот так выглядит наша страница. Да, менее привлекательно, но вполне читаемо.



Можете убрать префиксы и проверить поведение страницы в Chrome и последних версиях Firefox.

```
@supports (background-clip: text){
    .text-gradient {
        color: transparent;
        background-clip: text;
        background-image: linear-gradient(90deg, #0052d4,
#fff);
    }
}
```