




HTML и CSS. Адаптивная вёрстка и автоматизация

Уровень 2, с 21 ноября 2022 по 30 января 2023

Меню курса

[Главная](#) / [8. Погружение в автоматизацию](#) /

8.9. Автопрефиксер

 ~ 12 минут

Браузеры по-разному организуют работу с правилами CSS. У каждого есть свой вариант работы того или иного правила. Все свойства, которые постепенно вводятся в CSS, появляются в каждом браузере постепенно и с особенностями. Только спустя достаточно большой промежуток времени и спустя множество тестов, а также договоренностей между производителями браузеров и разработчиками спецификации правило полноценно начинает работать в браузере.

А пока этого не произошло, в браузере есть механизмы, позволяющие «включить» работу таких экспериментальных свойств. Сейчас таким механизмом являются «флаги» в интерфейсе. А до недавнего времени таким инструментом были «вендорные префиксы» (вендор — производитель браузера).

```
.wrapper {  
  -webkit-transition: 0.3s;  
  -moz-transition: 0.3s;  
  -o-transition: 0.3s;  
  transition: 0.3s;  
}
```

Давайте разбираться с тем, как работает этот механизм. Проще всего понять на примере. Допустим 10 лет назад была X-версия популярного браузера X и начинало свой путь правило Y. Версия X решила включить поддержку правила Y, но конечно же с префиксом. Через 2 года производители браузеров только-только договорились как будет работать правило Y. А в версии X у популярного браузера всё ещё работает старый вариант, и процент пользователей этой версии достаточен, чтобы поддерживать его.

Но как это правило работает правильно в версии X? Ведь то, как оно точно работало стало понятно только тогда, когда уже все договорились и правило введено в об

На самом деле спецификация правила уже была описана, возможно частично, а возможно полностью. На заре рождения правила уже плюс-минус понятно, что оно примерно должно делать и как работать. Дальше браузеры тестируют, пробуют и так далее. Вот как раз поэтому правило с префиксом может работать и даже правильно работать (но есть исключения).

В общем, префикс — устаревший, но всё ещё актуальный, инструмент «включения» неподдерживаемых в браузере правил CSS.

Какие бывают префиксы и к какому правилу писать

У каждого браузера (ну точнее у каждого браузерного движка) свои префиксы:

- Firefox имеет префикс `-moz-`.
- Chrome, Safari и другие браузеры, которые используют движок этих браузеров, имеют префикс `-webkit-`.
- IE имел префикс `-ms-`.
- Opera когда-то имела префикс `-o-`.

Основная масса нужных правил в современных браузерах поддерживается и можно спокойно всеми пользоваться. Однако, когда разговор заходит о достаточно старых версиях браузеров или возникает ситуация, что какое-то правило в каком-то браузере не работает или работает не так, как это предполагается — следует проверить, что с поддержкой этого правила в конкретном браузере. Проверить можно в сервисе caniuse.com.

Как пользоваться этим сервисом? Необходимо ввести в поиск название CSS-правила. Сервис выдаст информацию о том, как и в каком браузере правило поддерживается или не поддерживается.

Давайте посмотрим поддержку grid. В поиск введём `grid`:

Can I use

grid

? ⚙ Settings

☒ Caniuse (2) ☒ MDN (48)

CSS Grid Layout (level 1) - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid*` properties and the `fr` unit.

Usage	% of all users	
Global	92.9% + 0.63% =	93.53%
unprefixed:	92.9%	

Current aligned Usage relative Date relative Filtered All

[illegible]

Notes Test on a real browser Known issues (3) Resources (9) Feedback

See also support for [subgrids](#)

¹ Enabled in Chrome through the "experimental Web Platform features" flag in `chrome://flags`

² Partial support in IE refers to supporting an [older version](#) of the specification.

³ Enabled in Firefox through the `layout.css.grid.enabled` flag

4 There are some bugs with overflow ([1356820](#), [1348857](#), [1350925](#))

Поддержка свойства grid в браузерах

Допустим, нам важно узнать, какая поддержка в IE 11. Наш заказчик хочет, чтобы сайт хорошо отображался в этом браузере. Выбираем соответствующую вкладку:

Home
News
March 18, 2022 - New feature: CSS font-palette
Compare browsers
About

Can I use
grid
?
Settings

CSS Grid Layout (level 1) - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

Usage
% of all users
Global
92.9% + 0.63% = 93.53%
unprefixed:
92.9%

Current aligned
Usage relative
Date relative
Filtered
All

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-9	2-39	4-28	IE 11													
Partial support with prefix: <code>-ms-</code>			Released Oct 17, 2013													
Usage			Global: 0.56%													
Notes			Partial support in IE refers to supporting an older version of the specification.													
Test on IE 11																

Notes
Resources (9)
Feedback

See also support for [subgrids](#)

- Enabled in Chrome through the "experimental Web Platform features" flag in `chrome://flags`
- Partial support in IE refers to supporting an older version of the specification.
- Enabled in Firefox through the `layout.css.grid.enabled` flag
- There are some bugs with overflow ([1356820](#), [1348857](#), [1350925](#))

Проверяем поддержку в IE11

Нам отобразился результат — `Partial support with prefix: -ms-`. Это означает, что `grid` работает только с префиксом `-ms-`.

Работа префиксов в браузере

Возьмем для примера очень простое правило — `box-sizing: border-box;`. Основная масса браузеров поддерживает это правило достаточно давно. Но когда-то его писали с префиксом и выглядело это примерно так:

```
* {
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
```

Сверху пишется правило с префиксом, после него пишется правило без префикса. Почему так? Причина этого заключается в том, что CSS пропускает незнакомые свойства и применяет то, что ниже. То есть если наш браузер поддерживает правило без префикса — он применит именно это правило без префикса. Давайте заглянем в инспектор:

```
* {  
  webkit box sizing: border box;  
  box-sizing: border-box;  
}
```

box-sizing переопределяет -webkit-box-sizing

Правило с префиксом зачеркнуто, применено правило без префикса. Напишем наоборот — будет наоборот. Применится правило с префиксом, а правило без префикса будет зачеркнуто. Но подход «наоборот» плох тем, что это правило с префиксом может работать не совсем так, как правило без префикса (помним, что правило с префиксом — это когда браузер тестирует его, возможно спецификация правила не дописана до конца и есть в конкретно этом браузере свои нюансы). То есть фактически если написать сверху правило без префикса, а под ним правило с префиксом — мы можем столкнуться с каким-нибудь багом даже в новом актуальном браузере, что не очень-то и хорошо. Поэтому пишем всегда правило с префиксом над правилом без префикса.

Отлично, с вариантом, когда префикс на правиле, мы разобрались. Но есть ещё вариант, когда префикс находится прямо в значении. Отличным примером для этого может послужить `linear-gradient`:

```
.wrapper {  
  background: -webkit-gradient(linear, left top, left bottom, from(white),  
to(black));  
  background: -o-linear-gradient(top, white, black);  
  background: linear-gradient(to bottom, white, black);  
}
```

Тут принцип ничем не отличается от того, что описан выше (правило с префиксом пишем выше, чем без префикса), только разве что для поддержки того или иного браузера пишется значение градиента по-своему (конечно, мы не пишем -webkit-background, потому что свойство background работает нормально всегда, а вот градиент уже имеет особенности в каждом браузере).

Теперь, узнав сколько есть нюансов в написании правил с префиксами, можно перейти к тому, что облегчит нам работу.

Автопрефиксер

Нет смысла смотреть поддержку абсолютно всех правил CSS на сайте caniuse. Облегчить работу с префиксами нам поможет программа Автопрефиксер. В ней задать настройки того, какие браузеры нам нужно поддерживать, и она сама расставляет нужные префиксы.

Представим, что у нас есть CSS с кучей правил и среди этих правил есть правила для grid в одном из селекторов:

```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-template-rows: repeat(3, 100px);  
}
```

Предположим, нам нужна та самая поддержка старых браузеров, допустим последних 5 версий. Проштудировав caniuse и ещё половину интернета, мы узнали, какие префиксы нужно писать. Получается такой вот код:

```
.wrapper {  
  display: -ms-grid;  
  display: grid;  
  -ms-grid-columns: (1fr)[3];  
  grid-template-columns: repeat(3, 1fr);  
  -ms-grid-rows: (100px)[3];  
  grid-template-rows: repeat(3, 100px);  
}  
  
.wrapper > *:nth-child(1) {  
  -ms-grid-row: 1;  
  -ms-grid-column: 1;  
}  
  
.wrapper > *:nth-child(2) {  
  -ms-grid-row: 1;  
  -ms-grid-column: 2;  
}  
  
.wrapper > *:nth-child(3) {  
  -ms-grid-row: 1;  
  -ms-grid-column: 3;  
}  
  
.wrapper > *:nth-child(4) {  
  -ms-grid-row: 2;  
  -ms-grid-column: 1;  
}  
  
.wrapper > *:nth-child(5) {  
  -ms-grid-row: 2;  
  -ms-grid-column: 2;  
}  
  
.wrapper > *:nth-child(6) {  
  -ms-grid-row: 2;  
  -ms-grid-column: 3;  
}
```

```
.wrapper > *:nth-child(7) {
  -ms-grid-row: 3;
  -ms-grid-column: 1;
}

.wrapper > *:nth-child(8) {
  -ms-grid-row: 3;
  -ms-grid-column: 2;
}

.wrapper > *:nth-child(9) {
  -ms-grid-row: 3;
  -ms-grid-column: 3;
}
```


Конечно, автопрефиксер сам добавит эти варианты написания правил с префиксами и поставит их в нужное место в нашем CSS-коде. Он не просто расставит префиксы всем необходимым правилам, а проанализирует сколько версий браузера нам нужно поддерживать и какие в этих браузерах должны быть префиксы для каких правил. Если префиксы не нужны, автопрефиксер их не добавит.

Прекрасно. Есть утилита и знаем как работает. А где её взять и как пользоваться?

Есть несколько вариантов. Давайте последовательно рассмотрим каждый.

Сайт autoprefixer

Всё достаточно просто. Переходим на сайт <https://autoprefixer.github.io/> (есть русская версия сайта):



Автопрефиксер CSS онлайн

Автопрефиксер — плагин для PostCSS для добавления вендорных префиксов в CSS

Postcss: v8.3.6, autoprefixer: v10.3.1

```
.example {
  display: grid;
  transition: all .5s;
  user-select: none;
  background: linear-gradient(to bottom, white, black);
}
```

```
/*
 * Prefixed by https://autoprefixer.github.io
 * PostCSS: v8.3.6,
 * Autoprefixer: v10.3.1
 * Browsers: last 5 version
 */

.example {
  display: -ms-grid;
  display: grid;
  -webkit-transition: all .5s;
  -o-transition: all .5s;
  transition: all .5s;
  -webkit-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  background: -webkit-gradient(linear, left top, left bottom, from(white), to(black));
  background: -o-linear-gradient(top, white, black);
  background: linear-gradient(to bottom, white, black);
}
```

Фильтрация браузеров: last 5 version Применить

☒ добавить комментарий с настройками в результат

Вы также можете посмотреть, какие браузеры вы выбираете по строке фильтра на [browserlist](#)

Сайт автопрефиксера

Поле слева — наш CSS-код, поле справа — итоговый результат после работы автопрефиксера, фильтрация браузеров — сколько последних версий браузеров нужно поддерживать (на примере установлено последние 5 версий популярных браузеров). Прокрутив страницу вниз, можно прочитать дополнительную информацию.

Как работать с этим инструментом? Выбираем сколько браузеров надо поддерживать, вставляем слева свой CSS-код и получаем справа нужный нам CSS-код, переносим его в проект и всё.

Это не очень удобно по нескольким причинам:

1. Приходится копировать-вставлять код.
2. Если нужны правки в коде — что делать? Оставлять где-то CSS-файл без префиксов отдельно, править его и потом опять прогонять через сайт и опять копировать-вставлять.
3. Если пишем CSS на препроцессоре (Sass, Less или любом другом) — как быть? Код правим в файлах препроцессора, а результирующий CSS-файл получается только после обработки в Gulp или Webpack.

В общем, есть нюансы по использованию сайта. Он удобен для того, чтобы посмотреть префиксы в правилах у нужных браузеров. Но точно не подходит для разработки. Чтобы закрыть эти вопросы — можно использовать плагин `autoprefixer`.

Работа с плагином `autoprefixer`

Плагин можно найти в [репозитории `autoprefixer`](#). Рассмотрим установку и настройку для сборщика Gulp.

Структура проекта:

- папка `src` находится в корневой папке, в ней лежит рабочий CSS-файл;
- папка `dest` — результат, который получается после работы над проектом, лежит в корневой папке проекта;
- рабочие файлы CSS лежат в папке `src` и на данном этапе мы просто выбираем все CSS-файлы в проекте.

Выполняем в проекте команду `npm install --save-dev postcss gulp-postcss`, которая установит в проект `postcss`.

После неё выполняем команду `npm install --save-dev autoprefixer`.

Отлично, зависимости установлены, можем переходить к настройке. Открываем `gulpfile.js` и указываем следующее:


```
import gulp from 'gulp';
import postcss from 'gulp-postcss';
import autoprefixer from 'autoprefixer';

export const css = () => {
  return gulp.src('./src/*.css')
    .pipe(postcss([
      autoprefixer(),
    ]))
    .pipe(gulp.dest('./dest'));
};
```

Также необходимо добавить в package.json блок `browserslist`, чтобы задать браузеры, которые нам нужно поддерживать (плагин `browserslist` устанавливается вместе с `autoprefixer`). Итоговый файл package.json нашего проекта будет с таким содержимым:

```
{
  "devDependencies": {
    "autoprefixer": "^10.4.4",
    "gulp": "^4.0.2",
    "gulp-postcss": "^9.0.1",
    "postcss": "^8.4.12"
  },
  "type": "module",
  "scripts": {
    "css": "gulp css"
  },
  "browserslist": [
    "last 5 version"
  ]
}
```

Теперь создадим файл `style.css` прямо в папке `src` и запишем в него содержимое:

```
.wrapper {
  appearance: none;
}
```

Выполним команду `npm run gulp css` в терминале и откроем файл `dest/style.css`:

```
.wrapper {
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
}
```

Вот такой код у нас получился. Конечно же, можно настроить работу autoprefixer совместно с Less, Sass или любым другим препроцессором, можно добавить минификацию CSS и ещё кучу дополнительных штук, и конечно же изменится как файл gulpfile.js, так и package.json. Но наша задача состояла в том, чтобы базово настроить autoprefixer и разобраться как он работает на примере обычных CSS-файлов и Gulp.

Итоги

- Браузеры могут добавлять поддержку новых правил CSS с помощью префиксов.
- Правила с префиксом пишутся выше правил без префикса.
- Важно определять, в каких браузерах сайт должен правильно работать. Например, поддержка IE не нужна и тогда нет смысла забивать файл CSS кучей правил с префиксом для IE. Это приведёт только к росту файла.
- Использовать caniuse для проверки работоспособности тех или иных правил в различных браузерах.
- Использовать сайт <http://autoprefixer.github.io/>, чтобы посмотреть, какие правила и как пишутся для тех или иных браузеров.
- Использовать autoprefixer в своей сборке для того, чтобы не писать в код префиксы самостоятельно.

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

⚠ Если вы обнаружили ошибку или неработающую ссылку, выделите ее и нажмите Ctrl + Enter

Поиск по материалам

Git

Все материалы

В самом начале



- ☐ Пройдите опрос
- ☐ Укажите персональные данные
- ☐ Изучите регламент
- ☐ Прочитайте FAQ

- ☐ Добавьте свой Гитхаб
- ☐ Выберите наставника
- ☐ Создайте проект

Мой наставник



[Выбрать наставника](#)

Работа с наставником

У вас осталось **10** из 10 консультаций.

[История](#)



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Информация

Об Академии

О центре карьеры

Услуги

Работа наставником

Для учителей

Стать автором

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

