

1

CSS3 СЕГОДНЯ

ГЛЯДЯ В ПРОШЛОЕ, на яркую историю CSS, мы видим важные этапы, которые формировали наше видение веб-дизайна. Прорывные приемы, статьи и события научили нас создавать гибкие и доступные сайты, которыми мы можем гордиться — и их обликом, и качеством кода.

Можно утверждать, что все интересное началось в 2001-м, когда Джеффри Зельдман написал статью «К черту плохие браузеры» (<http://bkaprt.com/css3/1/>)¹, обозначив рассвет эпохи CSS. Этот манифест заставил дизайнеров двигаться дальше и использовать CSS не только для задания шрифтов и цветов ссылок, таким образом оставляя позади старые браузеры, не понимавшие CSS1. Да-да, CSS1.

Затем мы провели несколько лет, обнаруживая приемы верстки на CSS, которыми достигали того, что от нас хотели клиенты и начальники, и делились этими приемами. Это было

¹ <http://www.alistapart.com/articles/tohell/> Здесь и далее прим. ред.

прекрасное время: мы экспериментировали, расширяли границы, находили сложные способы борьбы с трудностями, которые несут особенности отображения сайтов в браузерах, — все ради возросшей гибкости, улучшенной доступности, упрощенного кода.

Приблизительно около 2006 года разговоры о CSS стихли. Решения к большей части задач, встающих перед нами, были найдены и хорошо задокументированы. Для известных ошибок браузеров были найдены несколько обходных путей. Были созданы группы поддержки для дизайнеров, страдающих из-за необъяснимых ошибок Internet Explorer. Наши волосы стали седеть (говорю за себя). Впрочем, важнее всего то, что современные браузеры были сравнительно бездвижными. Этот период status quo дал нам время отточить используемые приемы и выработать набор передовых практик, но положение дел стало немного, осмелюсь сказать, скучным для приверженца CSS, который стремился получить лучшие инструменты.

К счастью, изменения наступили. Браузеры стали обновляться чаще (по крайней мере некоторые). Firefox и Safari не только принялись увеличивать доли рынка — они также пожинали плоды более короткого цикла разработки, добавляя поддержку устоявшихся стандартов одновременно с более экспериментальными свойствами. Во многих случаях те технологии, которые внедрялись браузерами, нацеленными на будущее, были включены в черновик спецификации. Другими словами, иногда именно производители браузеров развивали спецификацию CSS3.

НЕ ЧИТАЙТЕ СПЕЦИФИКАЦИИ

Зайдите в комнату, наполненную веб-дизайнерами, и спросите их: «Кто любит читать спецификации?» Возможно, вы увидите одну поднятую руку. (Если этот человек — вы, то я горжусь вами и свободным временем, которое у вас есть,

по всей видимости.) Пусть они и представляют собой важный справочник, я определенно не получаю удовольствия от чтения спецификаций целиком и не рекомендую заниматься этим, чтобы полностью постигнуть CSS3.

Хорошая новость заключается в том, что CSS3 — это на самом деле набор модулей, которые, согласно задумке, должны внедряться изолированно и независимо друг от друга. Это очень хорошо. Такой подход — дробление спецификации — позволяет одним фрагментам спецификации двигаться быстрее, чем другим, и подталкивает производителей браузеров к тому, чтобы они внедряли хорошо проработанные фрагменты до того, как спецификация CSS3 будет считаться целиком законченной.

W3C (World Wide Web Consortium. *Прим. перев.*) объясняет модульный подход так:

Вместо того чтобы пытаться впихнуть десятки обновлений в единую неделимую спецификацию, будет намного проще и эффективнее дать возможность обновлять отдельные куски спецификации. Модульность даст возможность CSS обновляться чаще и точнее, таким образом позволяя более гибкое и своевременное развитие спецификации в целом².

Преимущество для нас, веб-дизайнеров, в том, что одновременно с экспериментированием и ускоренным циклом релизов приходит возможность использовать многие свойства CSS3, не дожидаясь, пока они получат статус кандидат в рекомендации (Candidate Recommendation. *Прим. перев.*) — возможно, это произойдет годы спустя.

Но, разумеется, если вам нравится читать спецификации — вперед! Естественно, из них можно многому научиться — но намного прагматичнее сконцентрироваться на том, что уже внедрено и может быть использовано сегодня, и об этих вещах

² <http://www.w3.org/tR/css3-roadmap/#whymods>

мы поговорим в этой главе. Затем мы будем применять эти вещи на конкретных примерах.

Мне всегда удавалось научиться большему о веб-дизайне, изучая рабочие примеры, нежели читая нормативные документы, и именно такого подхода мы будем придерживаться на страницах этой книги.

CSS3 — для всех

От многих веб-дизайнеров со всего мира я слышу одну и ту же фразу: «Не могу дождаться, хочу начать использовать CSS3... когда его закончат».

Правда состоит в том, что можно начинать использовать CSS3 прямо сейчас. К счастью, не нужно перестраивать свое мышление или радикально менять те способы, на основе которых сейчас разрабатываются сайты. Каким образом можно использовать CSS3 на любом проекте? Мы изучим те ситуации, где пользоваться CSS3 приемлемо, концентрируясь непосредственно на взаимодействии.

Целиться на взаимодействие

Если в течение последних нескольких лет мы все делали правильно, то работали на основе веб-стандартов (семантическая HTML-разметка и CSS для форматирования, шрифтов, цветов и так далее), оставляя большинство интерактивных эффектов — анимацию, обратную связь, движение — технологиям Flash и JavaScript. Свойства CSS3, которые медленно, но верно появляются в браузерах, нацеленных на будущее, позволяют нам переносить часть такого взаимодействия в таблицы стилей.

Как дизайнер интерфейсов, которому намного ближе визуальная сторона веб-дизайна, а не программирование, чем больше я могу сделать для создания наглядного интерфейса, пользуясь привычными инструментами — HTML и CSS, — тем больше я радуюсь.

CSS3 придумали для веб-дизайнеров — таких, как вы и я, — и мы можем начать пользоваться его частями сегодня — до тех пор, пока мы знаем, когда и как его применять.

Когда применять CSS3

В терминах визуального взаимодействия с сайтом можно разбить все на две категории: ключевое и второстепенное (табл. 1.01).

Такие составляющие, как брендинг, юзабилити, и расположение блоков крайне важны для успеха любого сайта, поэтому для этих элементов пользоваться технологией, которая не полностью поддерживается всеми браузерами, — рискованное предприятие.

Например, в развивающейся спецификации CSS3 появилось несколько элементов для управления форматом — это то, что нам всем крайне нужно.

КЛЮЧЕВОЕ	ВТОРОСТЕПЕННОЕ
Брендинг	Взаимодействие
Юзабилити	Визуальные эффекты
Доступность	Обратная связь
Расположение блоков	Движение

Таблица 1.01. Визуальное взаимодействие с сайтом можно разбить на категории ключевое и второстепенное. К второстепенному CSS3 можно применять сегодня

Уже многие годы мы заставляем свойство `float` справляться с раскладкой блоков. Мы разобрались, как справляться с задачами с помощью тех средств, которые у нас

есть, но настоящий механизм создания раскладки совершенно необходим.

Учитывая все вышесказанное, два из трех модулей CSS3, касающихся раскладки, еще не внедрены ни в один браузер. Эти модули по-прежнему находятся в разработке, и они по-прежнему весьма запутанны, сложны для понимания и, скорее всего, отличаются от окончательного решения, которое мы ищем. Что важнее, для такой важной вещи, как форматирование, CSS3 — просто не тот инструмент.

С другой стороны спектра расположены второстепенные события — взаимодействие (эффекты наведения и фокусировки, оформление форм, гибкая адаптация под любой размер окна) и визуальные эффекты, которые сопровождают такое взаимодействие (включая анимации). Менее важно сохранить одинаковое поведение такого рода в разных браузерах, что дает отличную возможность применять определенные фрагменты CSS3 для тех браузеров, которые поддерживают их сейчас.

На протяжении всей книги мы будем применять CSS3 для этих некритических событий, сохраняя более важные характеристики страницы нетронутыми для всех браузеров, вне зависимости от того, поддерживают ли они CSS3.

Когда мы решаем сконцентрироваться на таких некритических частях визуального взаимодействия, становится намного проще использовать CSS3 поверх CSS2.1 и обогащать взаимодействие с сайтом, не беспокоясь о том, что основной смысл, формат и доступность будут искажены.

ГЛАВНЫЕ СВОЙСТВА CSS3, ПРИМЕНИМЫЕ СЕЙЧАС

Теперь, когда мы точно определили область взаимодействия, в которой можно смело использовать CSS3, нам стоит также определиться, какие свойства CSS3 мы можем использовать. Иными словами, какие фрагменты спецификации

достигли того уровня поддержки браузерами, чтобы быть применимыми уже сейчас.

Крупные блоки CSS3 до сих пор не внедрялись ни в один браузер. Какие-то вещи по-прежнему находятся в разработке. Можно любопытствовать о тех блоках, которые находятся в движении, но куда разумнее обратить внимание на то, что на самом деле работает — и, к счастью, такого уже предостаточно.

Давайте рассмотрим сравнительно небольшой набор главных свойств CSS3, которые будут использованы в примерах в этой книги (см. ниже и табл. 1.02). Сейчас приводятся только определения этих свойств; подробное описание синтаксиса и практика применения будут даны позже.

border-radius

Скругляет углы элемента на заданное значение — радиус. Поддерживается в Chrome 3+, Firefox 1+, Opera 10.5+ и IE9 Beta. Пример:

```
.foo {  
  border-radius: 10px;  
}
```

text-shadow

Свойство из CSS2 (выкинутое в версии 2.1, возвращенное в CSS3), которое добавляет тень к тексту; можно указывать направление, количество размытия и цвет тени. Поддерживается в Safari 1.1+, Chrome 2+, Firefox 3.1+ и Opera 9.5+. Пример:

```
p {  
  text-shadow: 1px 1px 2px #999;  
}
```

СВОЙСТВО	ПОДДЕРЖКА
border-radius	 3+  3+  1+  10.5+  9 beta
text-shadow	 1.1+  2+  3.1+  9.5+
box-shadow	 3+  3+  3.5+  10.5+  9 beta
Multiple background images — несколько фоновых изображений	 1.3+  2+  3.6+  10.5+  9 beta
opacity	 1.2+  1+  1.5+  9+  9 beta
RGBA	 3.2+  3+  3+  10+  9 beta

Таблица 1.02. Свойства CSS3 и браузеры, поддерживающие их

box-shadow

Добавляет тень к элементу. Синтаксис тот же, что у свойства `text-shadow`. Поддерживается в Safari 3+, Chrome 3+, Firefox 3.5+, Opera 10.5+ и IE9 Beta. Пример:

```
.foo {
  box-shadow: 1px 1px 2px #999;
}
```

Несколько фоновых изображений

CSS3 дает возможность поставить несколько фоновых изображений на один элемент (разделяя их запятыми) вместо всего лишь одной картинки согласно спецификации CSS2.1. Поддерживается в Safari 1.3+, Chrome 2+, Firefox 3.6+, Opera 10.5+ и IE9 Beta. Пример:

```
body {  
    background: url(image1.png) no-repeat top left,  
    url(image2.png) repeat-x bottom left,  
    url(image3.png) repeat-y top right;  
}
```

opacity

Определяет непрозрачность элемента. Значение 1 соответствует полной непрозрачности; значение 0 соответствует полной прозрачности. Поддерживается в Safari 1.2+, Chrome 1+, Firefox 1.5+, Opera 9+ и IE9 Beta. Пример:

```
.foo {  
    opacity: 0.5; /* .foo will be 50% transparent */  
}
```

RGBA

Не свойство CSS, но, скорее, новая цветовая модель, введенная в CSS3, добавляющая возможность задавать уровень прозрачности элемента вместе с его цветом в формате RGB. Поддерживается в Safari 3.2+, Chrome 3+, Firefox 3+, Opera 10+ и IE9 Beta. Пример:

```
.foo {  
    color: rgba(0, 0, 0, 0.75); /* black at 75% opacity */  
}
```

Разумеется, этот список далеко не полный. CSS3 содержит намного больше свойств и инструментов, многие из которых по-прежнему разрабатываются и пока что не включены ни в один браузер. Но вы заметите, что каждое свойство из списка выше достигло определенного уровня поддержки браузерами: оно работает хотя бы в двух наиболее распространенных браузерах. В некоторых случаях, поддержка обещана в будущих версиях Internet Explorer (и Opera).

Итак, теперь перед нами — краткий список свойств, с которыми можно экспериментировать, построенный на основе их сравнительно качественной поддержки в Safari, Chrome, Firefox и Opera. Пока что они работают не везде одинаково, и далее мы обсудим, почему такое поведение в порядке вещей и как подготовиться к этой неоднородной поддержке.

Какие темы не будут затронуты

Я перечислил несколько свойств, которые будут часто применяться в этой книге, а как же остальные? Я решил не пытаться охватить все в одной книге, а рассказать лишь о том, что применимо прямо сейчас по причине надежной и устойчивой поддержки браузерами.

Есть и другие фрагменты спецификации CSS3, которые уже можно применять, и о которых можно написать отдельную книгу:

1. Медиазапросы (<http://www.w3.org/TR/CSS3-mediaqueries/>)
2. Многоколоночный текст (<http://www.w3.org/TR/CSS3-multicol/>)
3. Веб-шрифты (<http://www.w3.org/TR/CSS3-webfonts/>)

Обязательно посмотрите на эти модули.

ПРЕФИКСЫ КОНКРЕТНЫХ БРАУЗЕРОВ

Ранее упоминалось, что спецификация CSS3 — это набор модулей, которые постепенно интегрируются производителями браузеров. Иногда интеграция включает в себя эксперимен-

тальную поддержку. Это означает, что пока спецификацию пишут, обсуждают и критикуют в W3C, изготовитель браузера может решить добавить поддержку для каких-то свойств, чтобы опробовать их на практике. В последнее время такая практика стала естественной частью процесса, и обратная связь, получаемая во время экспериментального использования, часто применяется, чтобы внести поправки в спецификацию.

С другой стороны, изготовитель браузера может захотеть ввести экспериментальное свойство, которое не входит ни в какой предложенный стандарт, но может получить такой статус когда-либо.

Для такой экспериментальной поддержки свойств CSS часто вводятся браузерные префиксы — например, так:

`-webkit-border-radius`

Отбитое дефисом ключевое слово, стоящее перед названием свойства, помечает его как незавершенное, относящееся исключительно к реализации в конкретном браузере и к интерпретации развивающейся спецификации. Если экспериментальное свойство войдет в законченный модуль CSS3, браузер должен поддерживать беспрефиксное название свойства.

У каждого изготовителя браузеров есть свой префикс, которым он в первую очередь помечает собственные экспериментальные свойства. Все остальные браузеры игнорируют правила, содержащие неизвестные им префиксы.

В табл. 1.03 перечислены самые широко используемые браузеры и связанные с ними префиксы. Мы будем использовать префиксы WebKit, Mozilla и Opera в части, касающейся CSS3 в примерах из следующих глав.

Как работают браузерные префиксы

Вот как CSS работает на практике с браузерными префиксами. Возьмем свойство `border-radius` в качестве примера.

 WebKit	-webkit-
 Mozilla	-moz-
 Opera	-o-
 Konqueror	-khtml-
 Microsoft	-ms-
 Chrome	-chrome-

Таблица 1.03. Наиболее широко используемые браузеры и связанные с ними префиксы

Положим, мы хотим скруглить углы элемента с радиусом 10 пикселей; вот как это делается:

```
.foo {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

Webkit (движок, используемый в браузерах Chrome, Safari, и в Safari для мобильных устройств) и Gecko (движок браузера

Firefox) поддерживают свойство `border-radius` посредством собственных префиксных свойств; Opera поддерживает это свойство без префикса. IE9 также будет поддерживать `border-radius` без браузерного префикса.

На момент подготовки издания (август 2012 года) все упомянутые браузеры поддерживают свойство `border-radius` без префикса, в том числе и IE9. *Прим. ред.*

Оптимальный порядок

Используя браузерные префиксы, важно не забывать о порядке, в котором перечисляются свойства. Можно заметить, что в предыдущем примере сначала написаны префиксные свойства, за которыми следует беспрефиксное свойство.

Зачем ставить подлинное CSS3-свойство последним? Вероятно, в будущем ваши стилевые файлы будут работать в большем количестве браузеров, постепенно улучшая дизайн. Когда браузер, наконец, будет поддерживать то свойство, которое определено в спецификации, применяться будет подлинное свойство, а не экспериментальная версия, так как оно будет стоять последним в списке. Даже если реализация префиксной версии будет отличаться от настоящего свойства из спецификации, вы заботитесь о том, что окончательный стандарт остается первостепенным.

Не пугайтесь браузерных префиксов!

Первоначальная реакция читателя может быть примерно такой: «Ах, это запутанные и проприетарные штуки!» Но я уверяю вас, это не только шаг вперед, а еще и намного менее запутанное решение в сравнении с раздутыми кусками кода и отсутствием гибкости, которыми сопровождаются решения не на CSS3. Кроме того, это важная часть развития спецификации.

Используя эти свойства сейчас с помощью браузерных префиксов, мы можем прозондировать почву и даже дать ценные

комментарии изготовителям браузеров, прежде чем спецификация завершена. Также стоит помнить, что префиксы, как правило, добавляются к предложенным стандартам (а не к утвержденным). В этом заключается большое отличие от разнообразного хакерского CSS, который все мы иногда использовали, чтобы разрешить проблемы с кросс-браузерностью.

Кто-то может сравнить браузерные префиксы с эксплойтами синтаксиса, которые многим из нас приходилось использовать, чтобы дать команду конкретным версиям браузеров (например, синтаксис `w\idth: 200px` или `_width: 200px`, который позволяет обращаться к конкретной версии IE). Но, напротив, браузерные префиксы — это важная часть процесса стандартизации, позволяющая развивать свойство, внедряя его для практического применения.

Эрик Мейер, эксперт по CSS, объясняет разницу в статье «Префикс или постхак» на A List Apart (<http://bkaprt.com/css3/2/>)³:

Префиксы дают нам контроль над нашей хакерской судьбой. В прошлом нам пришлось выдумать кучку ошибок парсера лишь для того, чтобы заставить несовместимые реализации работать одинаково — когда мы обнаружили, что они несовместимы. То был полностью реакционный подход. Префиксы — это подход с прицелом на будущее.

Он продолжает, предполагая, что префиксы — это не только хорошая практика, но они также должны занимать более значимую роль в процессе стандартизации:

...заставить производителей браузеров и рабочую группу трудиться вместе, чтобы разрабатывать тесты, необходимые для проверки интероперабельности. Затем эти тесты могут помочь тем (разработчикам браузеров. Прим. перев.), которые следуют

³ <http://www.alistapart.com/articles/prefix-or-posthack/>

за остальными, намного быстрее достичь интероперабельности. Они могут буквально выкатить реализацию с префиксом в одной публичной бета-версии и опустить префикс в следующей версии.

Так что не беспокойтесь о браузерных префиксах. Пользуйтесь ими, зная, что вы становитесь частью процесса, который позволяет достичь результата сейчас и прокладывает дорогу к будущему, когда от префиксов можно будет отказаться.

А как насчет повторений?

Можно думать, что довольно глупо повторять трижды или четырежды то, что выглядит как одно и то же свойство, и я могу согласиться.

Но реальность такова, что для решений, построенных на CSS3, скорее всего, потребуется писать негибкий и более сложный код, пусть, может быть, и неповторяющийся.

Нам не понадобится повторяться вечно. Сейчас это необходимый, но временный шаг, нужный, чтобы разделять реализации, отличающиеся между браузерами, от реализации окончательной спецификации.

Прежде чем мы начнем делать привлекательные вещи, пользуясь несколькими применимыми свойствами CSS3 и соответствующими браузерными префиксами, давайте познакомимся с переходами в CSS. Понимание переходов и того, как они работают, поможет нам сочетать их с остальными свойствами и создавать замечательные взаимодействия.