A BOOK APART
Актуальные книги для тех, кто создает сайты
No. 5

Дэн Сидерхолм

CSS3 ДЛЯ ВЕБ-ДИЗАЙНЕРОВ

Предисловие Джеффри Зельдмана

Дэн Сидерхолм

CSS3 для веб-дизайнеров

Серия «Актуальные книги для тех, кто создает сайты», книга 5

Текст предоставлен издательством
http://www.litres.ru/pages/biblio_book/?art=4570394
CSS3 для веб-дизайнеров / Дэн Сидерхолм: Манн, Иванов и Фербер; Москва; 2013
ISBN 978-5-91657-595-8

Аннотация

CSS3 – будущее веб-разработки, новый стандарт оформления документов, расширяющий возможности предыдущего стандарта CSS2. Многие возможности, которые ранее были труднодоступными, в CSS3 могут просто достигаться за счет использования новых свойств оформления.

Абсолютные преимущества технологий нового поколения – на высоте. Прежде всего, простота и легкость – для программистов, а удобство и комфорт – для пользователей.

Книга Дэна Сидерхолма поможет вам использовать CSS3 прямо сейчас, применяя технологии, появившиеся в новых стандартах.

На примере дизайна веб-страницы автор показывает применение всех, изложенных в книге, инструментов.

Содержание

Дэн Сидерхолм
CSS3 для веб-дизайнеров

Предисловие

Сайты – это не то же самое, что изображения сайтов. Когда один человек отрисовывает сайт в Фотошопе, а другой пишет разметку и CSS-код, верстальщик вынужден угадывать и предполагать те или иные намерения дизайнера. Этот процесс интерпретации редко совершенен – только если верстальщика не зовут Дэн Сидерхолм. Когда Дэн верстает макеты других людей, у него все получается так, как нужно, и даже те места, в которых дизайнер ошибся. Например, Дэн неизбежно превращает фиксированные размеры объекта в макете в гибкий, легко читаемый и надежный код.

(И правда, Дэн ввел в обращение фразу «надежный веб-дизайн», когда учил всех нас такому дизайну.)

В случае Дэна гибкий никогда не означает недоделанный. Детали всегда важны, потому что Дэн не только талантливый верстальщик, всегда думающий об интересах пользователя, – он также прирожденный дизайнер. Он живет дизайном, дышит дизайном и даже подарил миру новый способ делиться дизайном: сайт dribbble.com. Дэн еще и прирожденный учитель и веселый человек, и его манера шутить с каменным лицом заставляет самого Стивена Райта выглядеть несерьезным в сравнении. Дэн многому учит, помогая дизайнерам улучшать свои навыки.

Вот почему, друзья, мы попросили его рассказать о CSS3. Можно лишь мечтать об учителе умнее, опытнее; о человеке, более сконцентрированном на дизайне, или о большем ценителе веб-стандартов, чем наш друг Дэн. Приятного чтения!

Джеффри Зельдман

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 Джефри 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 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.

[1] http://www.alistapart.com/articles/tohell/ *Здесь и далее прим. ред.*

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 целом*[2].

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 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.

[2] http://www.w3.org/tR/css3-roadmap/#whymods

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**).

КЛЮЧЕВОЕ	ВТОРОСТЕПЕННОЕ
Брендирование	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

Такие составляющие, как брендирование, 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.

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.

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

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;
}
```

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 экспериментальную 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.

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

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

Как работают браузерные префиксы

Вот как CSS работает на практике с браузерными префиксами. Возьмем свойство `border-radius` в качестве примера. 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 спе-цификация 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 за остальными, намного быстрее достичь интероперабельности. Они могут буквально выкатить реализацию с префиксом в одной публичной бета-версии и опустить префикс в следующей версии.*

[3] http://www.alistapart.com/articles/prefix-or-posthack/

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.

2. Переходы в CSS

Шел 1997 год; я сидел в плохонькой квартирке в красивом Оллстоне, в Массачусетсе. Обычная ночь просмотра исходников и изучения HTML, которой предшествовал день упаковывания компакт-дисков на местной звукозаписывающей студии, – практически бесплатно (потому и плохонькая квартирка). Уверен, вы понимаете.

В одну торжественную ночь я ударил кулаком по столу в восторге от своей победы. Мне удалось написать на JavaScript код, который заменял одну картинку на другую при наведении курсора. Помните такие эффекты?

Я по-прежнему помню свое изумление, когда видел, как наскоро сделанная кнопка сменяется другой, когда я наводил на нее курсор. Тогда я с трудом понимал, что делаю, но заставлять часть веб-страницы меняться динамически – это было, ну… магией.

За последнее десятилетие мы прошли долгий путь в отношении взаимодействия и визуальных эффектов на веб-сайтах. Исторически сложилось так, что анимации, движение и взаимодействие создавались такими технологиями, как JavaScript и Flash. Но в последнее время, когда в браузерах появляется поддержка CSS-переходов и трансформаций, часть анимаций и улучшение взаимодействия могут быть перенесены в таблицы стилей.

На первый скрипт для смены картинок в 1997 году у меня ушло несколько ночей; я написал много строк кода, который тогда мне казался совершенно чуждым, и пришлось использовать несколько картинок. Сейчас CSS3 позволяет строить намного более яркие и гибкие эффекты, создаваемые лишь несколькими строками кода. Такие решения корректно воспринимаются и теми браузерами, которые пока что не поддерживают новые свойства.

Как упоминалось в первой главе, мы можем начать использовать CSS3 прямо сейчас при условии, что мы аккуратно выбираем те ситуации, где применяем новые свойства. То же самое справедливо и для CSS-переходов. Они определенно не заменят существующие технологии – Flash, Javascript или SVG

(особенно без более широкой поддержки браузерами), – но в сочетании с ранее упомянутыми основными свойствами CSS3 (а также трансформациями и анимациями, о которых будет рассказано далее) ими можно пользоваться, чтобы сдвинуть взаимодействие немного вперед. Что самое важное, пользоваться ими сравнительно легко для того, кто уже знаком с CSS. Переход на CSS занимает лишь несколько строк кода.

CSS-переходы описаны во второй главе, они будут применяться во многих примерах книги. Получить начальное представление о синтаксисе переходов и о том, как они работают, будет разумно, прежде чем мы окунемся глубже в изучение CSS3.

Хвост, который размахивает собакой

Изначально разработанные исключительно командой,

работавшей над движком WebKit для Safari, CSS-переходы теперь стали спецификацией в состоянии рабочий черновик в W3C. (У CSS-трансформаций и CSS-анимаций похожее происхождение; о них мы поговорим в главах 4 и 6 соответственно.)

Это хороший пример того, как новшества браузеров становятся частью потенциального стандарта. Потенциального, потому что на сегодняшний день это всего лишь черновик. Однако Opera недавно добавила поддержку CSS-переходов в версии 10.5 и была обещана их поддержка в Firefox 4.0. Иными словами, хоть это и черновая спецификация, которая развивается, она достаточно стабильна, чтобы Opera и Firefox воспринимали ее всерьез и добавляли поддержку для нее. Что важнее всего, CSS-переходы больше не относятся к проприетарным экспериментам Safari.

Давайте посмотрим на то, как работают переходы. Как и свойства CSS3, о которых говорилось в первой главе, я дам лишь определения и покажу основной синтаксис, чтобы у читателя было ясное понимание того, как работают переходы. Позже мы будем делать разнообразные классные штуки, пользуясь переходами, чтобы довести до блеска примеры из следующих глав, и будет ясно, как переходы становятся частью общей композиции.

Что такое CSS-переходы

Мне нравится воспринимать CSS-переходы как масло, сглаживающее изменения значений в стилевых таблицах, вызванные действием пользователя: когда он наводит курсор на объект, нажимает на него или выделяет его. В отличие от настоящего масла переходы не полнят – они представляют собой лишь несколько простых правил, добавляемых в таблицу стилей, которые улучшают определенные события в дизайне сайта.

W3C объясняет CSS-переходы достаточно просто (http:// bkaprt.com/css3/3/)[4]:

> *CSS-переходы позволяют делать так, чтобы изменения значений CSS-свойств происходили плавно в течение указанного интервала времени.*

Это сглаживание анимирует изменение значения CSS, вызванное нажатием мыши, переходом в состояние focus или active или любым изменением элемента (включая изменение классов элемента).

[4] http://www.w3.org/tR/Css3-transitions/

Простой пример

Начнем с простого примера: наложим переход на изменение фона ссылки. Когда пользователь будет наводить на ссылку, цвет ее фона будет меняться, и мы применим переход, чтобы сделать это изменение плавным. Такого эффекта раньше можно было добиться исключительно средствами Flash или JavaScript, но теперь его можно сделать, написав лишь несколько строчек на CSS.

Разметка состоит исключительно из одной ссылки:

```
<a href="#» class=»foo">Transition me!</a>
```

Теперь мы объявим неактивное состояние ссылки с небольшим отступом и светло-зеленым фоном и затем укажем темно-зеленый цвет при наведении (**рис. 2.01**):

```
a.foo {
padding: 5px 10px;
background: #9c3;
}
a.foo: hover {
background: #690;
}
```

Рис. 2.01. Обычное состояние ссылки и: hover

Теперь наложим переход на это изменение. Переход сгладит и анимирует изменение в течение указанного промежутка времени (**рис. 2.02**).

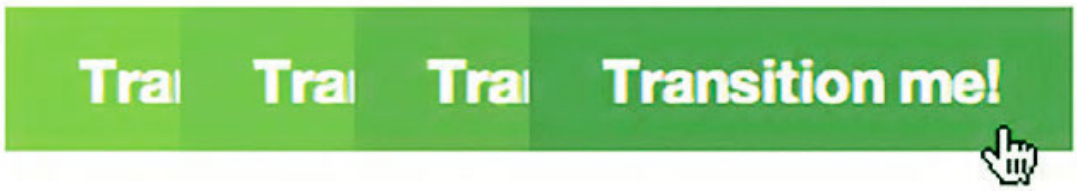

Рис. 2.02. Печатная страница – не лучший способ показать анимированный переход, но на этой картинке попытались сделать именно это: плавный переход от светло– зеленого к темно-зеленому фону

Ради компактности будем использовать только те браузерные префиксы, которые сейчас работают в браузерах на движке WebKit (это Safari и Chrome). Позже добавим префиксы для Firefox и Opera.

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition-property: background;
-webkit-transition-duration: 0.3s;
-webkit-transition-timing-function: ease;
}
a.foo: hover {
```

```
background: #690;
}
```

В этом коде можно увидеть три составляющих перехода:

• `transition-property` – свойство, на которое будет накладываться переход (в этом случае – свойство background);

• `transition-duration` – продолжительность перехода (0,3 с);

• `transition-timing-function` – как быстро переход осуществляется с течением времени (ease).

Временные функции (мне следовало быть внимательнее на уроках математики)

Значение временной функции позволяет менять скорость перехода с течением времени одним из шести способов: `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out` и `cubic-bezier`, который позволяет определить произвольную временную кривую.

Если вы, как и я, проспали все школьные уроки геометрии, не беспокойтесь. Я советую просто подставить каждое значение по очереди и увидеть, чем они отличаются друг от друга.

Продолжительность перехода в этом примере так мала, что сложно различить все шесть способов. Для более длительных анимаций выбранная временная функция становится важным параметром, так как есть время заметить изменение скорости на протяжении анимации.

Если сомневаетесь, знайте: значения `ease` (значение по умолчанию) или `linear` прекрасно подходят для коротких переходов.

Задержка перехода

Можно сделать так, чтобы переход осуществлялся не сразу после того, как срабатывает связанное с ним событие, но с некоторой задержкой. Например, сделаем так, чтобы переход цвета фона происходил через полсекунды после того, как ссылка попала в состояние hover. Такого поведения можно добиться свойством `transition-delay`.

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition-property: background;
-webkit-transition-duration: 0.3s;
-webkit-transition-timing-function: ease;
-webkit-transition-delay: 0.5s;
}
a.foo: hover {
background: #690;
}
```

Краткая форма записи

Можно существенно упростить объявление перехода (в котором нет задержки), пользуясь свойством transition. Такой синтаксис будет использоваться во всех остальных примерах этой книги.

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: background 0.3s ease;
}
a.foo: hover {
background: #690;
}
```

Мы получили намного более компактное правило, которое дает точно такой же результат.

Краткая форма записи перехода с задержкой

Если нужно добавить полусекундную задержку в краткую запись перехода, ее продолжительность ставится в конец правила:

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: background 0.3s ease 0.5s;
}
a.foo: hover {
background: #690;
}
```

Разумеется, эти замечательные переходы прекрасно действуют в браузерах, работающих на движке WebKit. Что насчет остальных?

Поддержка в браузерах

Как упоминалось ранее, переходы были изначально разработаны для движка WebKit и включены в Safari и Chrome начиная с версии 3.2, но Opera также поддерживает их начиная с 10.5 (http://bkaprt.com/css3/4/)[5]. Поддержка заявлена и в Firefox 4.0 (http://bkaprt.com/css3/5/)[6].

Учитывая поддержку переходов на сегодняшний день и в ближайшем будущем, важно перечислять все требуемые браузерные префиксы, чтобы переходы работали в большем количестве браузеров по мере появления поддержки.

[5] http://www.opera.com/docs/specs/presto23/css/transitions/

[6] https://developer.mozilla.org/en/Css/Css_transitions

Полная запись перехода

Ниже приводится дополненное объявление перехода, в которое добавлены префиксы `-moz-` и `-o-`, как и основное свойство `transition`. Как обычно, свойство без префикса ставится в самый конец, чтобы у него был наибольший вес, когда это свойство перейдет из состояния черновика в окончательную версию спецификации.

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: background 0.3s ease;
-moz-transition: background 0.3s ease;
-o-transition: background 0.3s ease;
transition: background 0.3s ease;
}
a.foo: hover {
background: #690;
}
```

Такая запись позволяет получить сглаживание цвета фона в последних версиях Safari, Chrome и Opera, равно как и в более свежих версиях всех остальных браузеров, которые решат поддерживать переходы.

Состояния перехода

Я помню, что слегка запутался, когда в первый раз начал экспериментировать с переходами на CSS. Казалось, что было бы логичнее расположить объявление перехода в тот фрагмент кода, где определяется состояние `:hover`. Оказывается, что элемент может находиться и в других состояниях – не только в `:hover` – и наверняка захочется, чтобы переход происходил в каждом состоянии без дублирования кода.

Например, можно наложить переход на состояния `:focus` и `:active`. Нам не придется добавлять объявление перехода в описание каждого свойства, так как параметры перехода указываются лишь один раз – для основного состояния элемента.

Следующий пример добавляет точно такое же переключение фона для состояния `:focus`.

Таким образом, переход произойдет либо от того, что на ссылку наведут курсор, либо от того, что на нее будет наведен фокус (например, клавиатурой).

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: background 0.3s ease;
-moz-transition: background 0.3s ease;
-o-transition: background 0.3s ease;
transition: background 0.3s ease;
}
a.foo: hover,
a.foo: focus {
background: #690;
}
```

Переход нескольких свойств

Предположим, что кроме цвета фона хочется также менять цвет самой ссылки и накладывать переход на это изменение. Такого эффекта можно достичь, перечисляя одновременно несколько переходов и разделяя их запятой. На каждый переход можно навесить отдельную продолжительность и собственную временную функцию (**рис. 2.03**). (Продолжение строки отмечено символом»).

Рис. 2.03. Обычное состояние ссылки и состояние: hover

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: background.3s ease, color 0.2s linear;
-moz-transition: background.3s ease, color 0.2s linear;
-o-transition: background.3s ease, color 0.2s linear;
transition: background.3s ease, color 0.2s linear;
}
a.foo: hover,
a.foo: focus {
color: #030;
background: #690;
}
```

Переход всех возможных состояний

Вместо того чтобы перечислять несколько свойств, к которым хочется применить переход, можно использовать значение `all`. Тогда переход будет наложен на все возможные свойства.

Заменим перечисление `background` и `color` на значение `all`. Теперь эти переходы получат одинаковую продолжительность и временную функцию.

```
a.foo {
padding: 5px 10px;
background: #9c3;
-webkit-transition: all 0.3s ease;
-moz-transition: all 0.3s ease;
-o-transition: all 0.3s ease;
transition: all 0.3s ease;
}
a.foo: hover,
a.foo: focus {
color: #030;
background: #690;
}
```

Использование `all` – удобный способ отследить все изменения, происходящие в состояниях `:hover`, `:focus`, `:active`, который избавляет от необходимости перечислять каждое свойство, нуждающееся в обозначении перехода.

К каким свойствам применим переход

Мы применили переход к свойствам background и color. Но есть много других свойств, на которые можно наложить переход, включая `width`, `opacity`, `position` и `font-size`. Таблица всех свойств (и их типов значений) опубликована на сайте W3C (http://bkaprt.com/css3/6/)[7].

Возможность создать полностью гибкое взаимодействие ясна. Мы будем использовать некоторые из этих свойств в сочетании с переходами в примерах следующей главы и далее в книге.

[7] http://www.w3.org/tR/css3-transitions/#properties-from-css-

Почему бы не воспользоваться JavaScript?

Читатель может подумать: раз не все браузеры поддерживают CSS-переходы, почему бы не использовать решение на JavaScript, чтобы показывать анимацию? Популярные фреймворки – jQuery, Prototype, script.aculo.us – уже давно предоставляют кросс-браузерные анимации.

Все зависит от того, насколько важны анимации. В книге я делаю акцент на том, что можно извлечь максимум из простоты и гибкости CSS3, выбирая подходящие части интерфейса, к которым можно применять новые свойства и улучшать взаимодействие пользователя с сайтом. Довольно часто анимации, сделанные средствами CSS-перехода, не относятся к ключевым составляющим сайта – таким как бренд, читаемость или расположение информационных блоков. Поэтому использовать несколько простых строчек на CSS, которые порождают простую анимацию естественными средствами браузера вместо того, чтобы привлекать JavaScript-фреймворк, кажется разумным. Очень хорошо, что он есть в нашем распоряжении.

Используйте с умом

Как и все блестящие новые инструменты, переходы стоит использовать там, где они уместны. Очень легко нарушить чувство меры и добавить переходы ко всем элементам на странице, таким образом превратив ее в раздражающего пульсирующего монстра. Очень важно осознавать, где переходы действительно улучшают интерфейс и где они лишь добавляют лишний шум. Кроме того, также важно следить за тем, чтобы переходы не мешали пользователю, когда он очень быстро взаимодействует с сайтом (например, резко переводит фокус с одного элемента меню на другой). Пользуйтесь переходами аккуратно и осторожно.

Больше мыслей о подходящих скоростях переходов и анимаций можно узнать в публикации Трента Уолтона: http:// bkaprt.com/css3/7/[8].

Теперь, когда мы заручились прочными базовыми знаниями о работе CSS-переходов с технической точки зрения, мы можем применять их для сглаживания взаимодействия в следующих примерах начиная со следующей главы. Итак.

[8] http://trentwalton.com/2010/03/22/Css3-in-transition/

3. Hover по-новому

На протяжении последних двух глав мы разбирались в том, какие свойства CSS3 можно использовать сейчас. Мы также говорили о том, в каких частях интерфейса резонно применять эти свойства.

Повторим самое важное, что мы уже успели узнать.

1. Существуют основные свойства CSS3, которые применимы уже сегодня.

2. Кто угодно может применять эти свойства в своих проектах, в особенности в отношении элементов взаимодействия.

3. Браузерные префиксы позволяют нам продвигать технологию прямо сейчас, помогая проверять неустоявшиеся свойства на реальных задачах.

4. CSS-переходы перестали быть проприетарными экспериментами – они стали частью спецификации, которую перенимают другие браузеры. Их стоит использовать!

Теперь самое время применить все эти прекрасные новые инструменты в дизайне настоящей веб-страницы.

Наш пример

В большинстве следующих примеров будет использован выдуманный сайт, который я сделал: шутливый знак уважения ко всем вещам, оставленным на Луне теми космонавтами, которым повезло, чтобы побывать там (**рис. 3.01**). За этим стоит история, которая напрямую связана с тематикой этой книги, – и я расскажу ее.

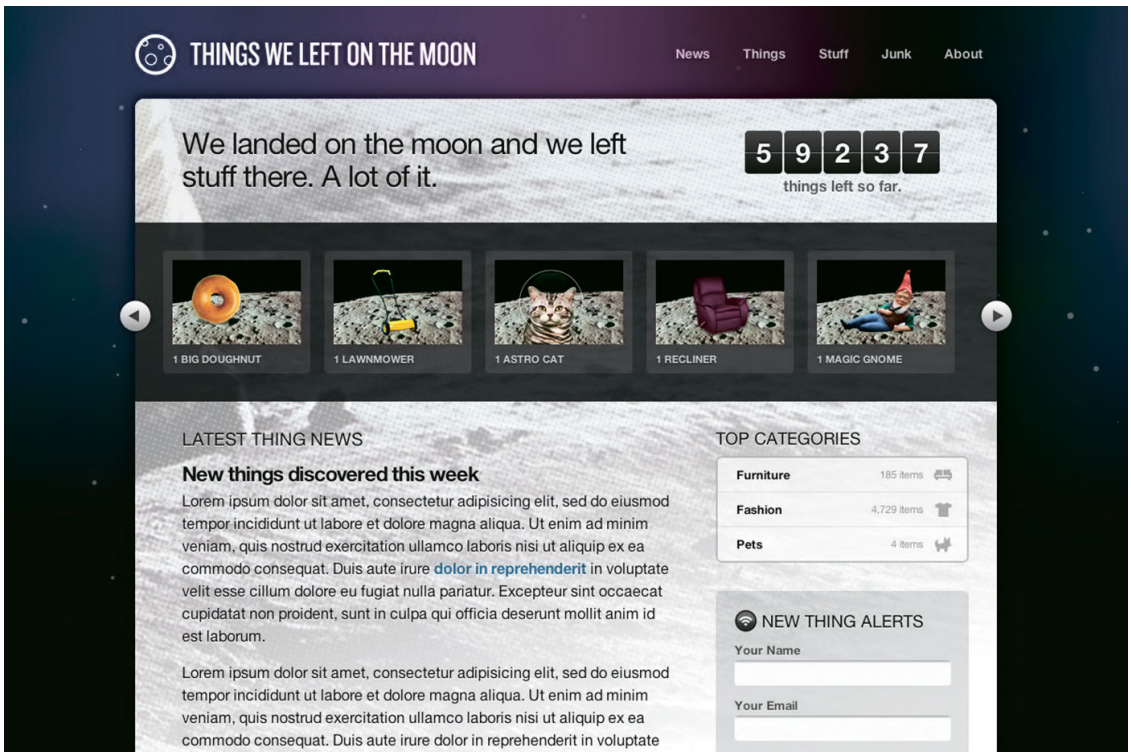

Рис. 3.01. Наш выдуманный пример: Вещи, которые мы оставили на Луне

Сообщения в космосе и в вебе

В 1969 году космонавты Нейл Армстронг и Базз Олдрин стали первыми людьми, ступившими на Луну. Я очень поверхностно интересовался путешествиями в космос и программой NASA (National Aeronautics and Space Administration, Национальное управление по воздухоплаванию и исследованию космического пространства. *Прим. перев.*), но то, что я услышал о миссии Apollo 11 во время ее 40-летнего юбилея, вдохновило меня на то, чтобы прочитать больше об истории и событиях, связанных с этой высадкой. В частности, я был поражен количеством вещей, которые были оставлены на Луне и остаются там по сей день.

Из всех предметов, находящихся там, есть один, который представляет исключительный интерес. Он являет собой прекрасный пример дизайна интерфейсов. Этот предмет – небольшой кремниевый диск размером с полудолларовую монету. На диске выгравированы доброжелательные послания от глав более семидесяти стран со всего мира. Чтобы прочитать послания, нужен микроскоп, но ограничения в отношении того, что космонавты могли взять с собой, определили дизайн памятного предмета, который можно было оставить на Луне для следующих поколений (**рис. 3.02**).

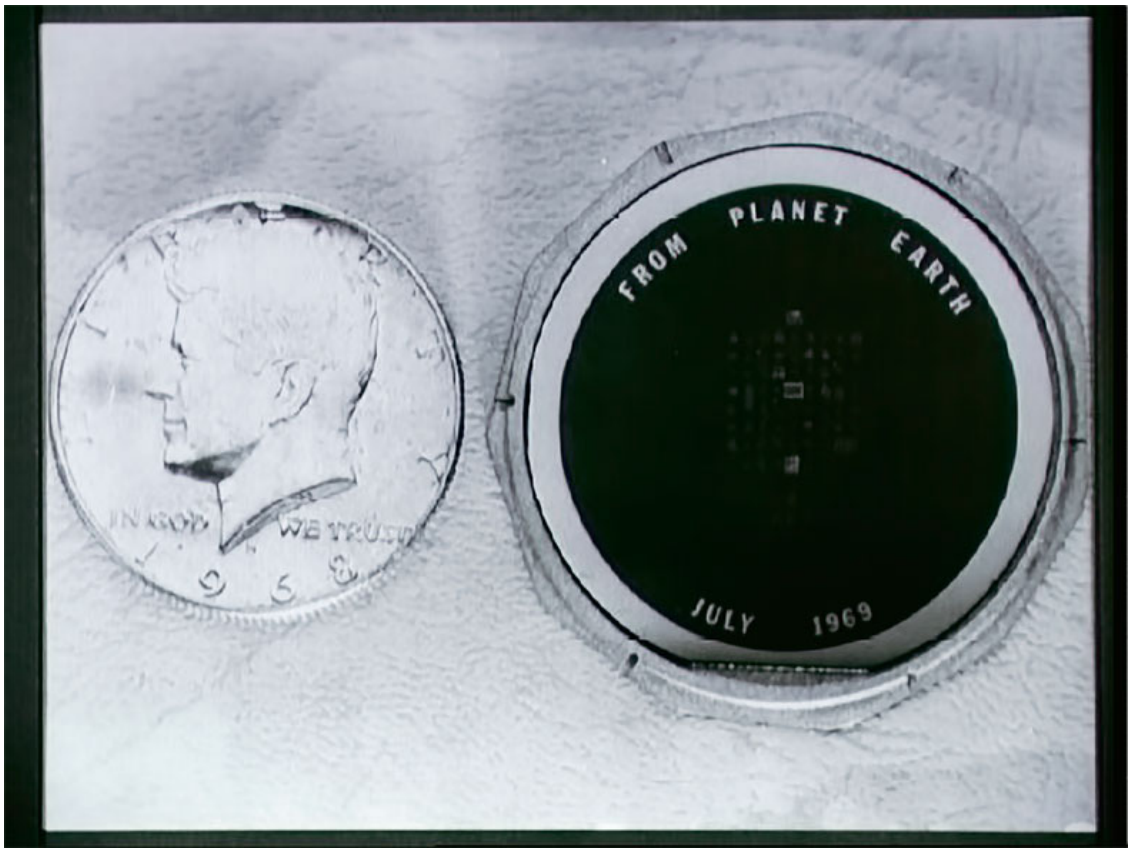

Рис. 3.02. Небольшой (размером с полудолларовую монету) кремниевый диск, оставленный на Луне космонавтами Apollo 11.

В некотором смысле NASA создала предмет, пользуясь самой современной технологией из доступных в то время, предназначенный для неизвестной аудитории из некоторого обозримого будущего. Звучит знакомо?

Позже, в 1977 году, схожая проблема дизайна была решена для летательных аппаратов Voyager 1 и Voyager 2 с помощью золотой пластинки, содержащей звукозаписи, изображения и диаграммы, описывающие жизнь на планете Земля (**рис. 3.03**). В каком-то смысле эта пластинка – письмо в бутылке, адресованное цивилизациям, находящимся за пределами Солнечной системы. На алюминиевом футляре, в который упакована пластинка, даны пояснения: как правильно прослушать запись, из какой галактики прибыл этот предмет и др.

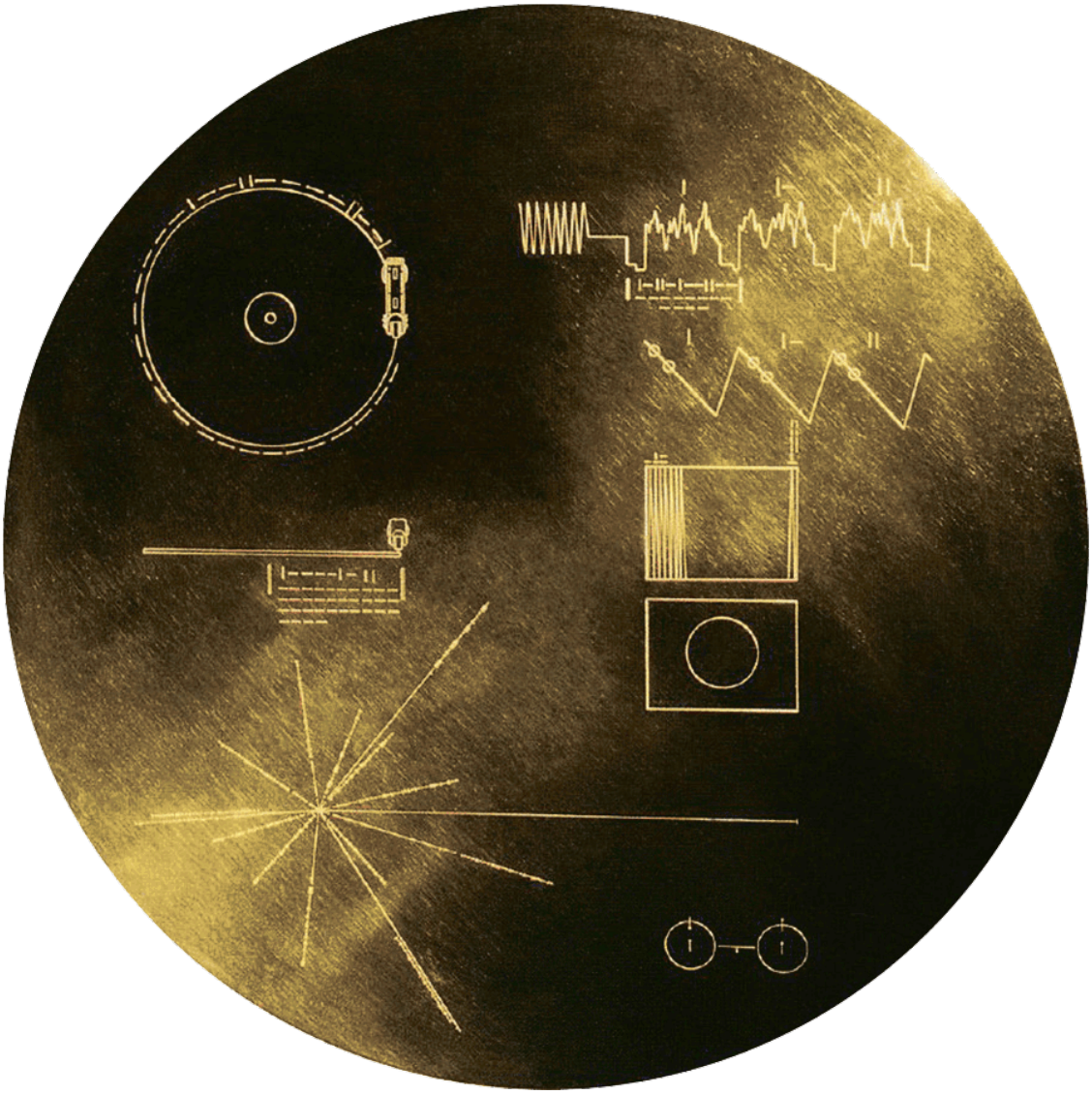

Рис. 3.03. Позолоченный футляр золотой граммофонной пластинки, находившейся на борту космических кораблей Voyager 1 и 2.

Как и кремниевый диск, по-прежнему лежащий в лунной пыли, для изготовления золотой пластинки были применены самые передовые технологии, доступные в то время, для неизвестного контекста использования. Смогут ли инопланетяне, которые найдут пластинку, видеть, ощущать и слышать ее содержимое?

Можно почерпнуть многое из историй о кремниевом диске, оставленном на Луне, и золотой пластинке, летящей сквозь космос: использование наилучшей технологии может помочь донести сообщение, которое отправляется во многом неизвестной аудитории.

Будучи веб-дизайнерами, мы тоже шлем послания в бутылке, создавая сайты для веба. Мы можем делать допущения о том, кто будет читать их, что они способны понять и так далее, – но мы никогда не обладаем полной информацией. Это не должно мешать нам использовать самые лучшие технологии из доступных, чтобы донести мысль и связанное с ней взаимодействие, причем так, чтобы взаимодействие подобающим образом упрощалось для владельцев старых или маломощных устройств.

Наша работа – работа дизайнера – состоит не только в том, чтобы изготовить нарядную бутылку, которая будет красиво выглядеть, но, скорее, в том, чтобы найти способы обогатить содержимое и улучшить подачу. CSS3 уже помогает нам в этом.

Теперь вы знаете, почему наш пример отдает дань уважения тем посланиям, которые оставлены на Луне или летят через космос. Настало время поделить этот сайт на части, выделяя небольшие примеры, относящиеся непосредственно к CSS3. Я считаю разумным собрать в одном месте все приемы, которые мы будем обсуждать. Читатель сможет обращаться к этому шаблону и ко всем примерам когда угодно – все собрано в одной живой, дышащей веб-странице. Код этого примера можно скачать с http://CSS3exp.com/code.

Каждая из оставшихся глав затрагивает отдельный набор примеров, связанных с CSS3. Вместо того чтобы пытаться включить все и рассказать все, что можно знать про CSS3, в этой главе я поступлю наоборот: погружусь в очень конкретные примеры, показывая, как они работают в выдуманном контексте. Эти примеры будут небольшими – такими, которые можно немедленно применять и расширять после прочтения следующих страниц.

Удивление и восторг

Дизайн под веб – такой особенный и интересный в сравнении со статическими носителями отчасти оттого, что веб-страницы интерактивны. В отличие от бумаги элементы могут реагировать на действия пользователя, двигаться и даже удивлять.

Именно интерактивность улучшается средствами CSS3 для тех браузеров, которые поддерживают эту технологию, но и не упускается для тех, которые не поддерживают.

Прекрасный образец того, как удивлять и восхищать с помощью CSS3, можно найти на сайте голландского дизайнера и разработчика Фарука Атеса (http://farukat.es). В боковом меню находится список ссылок на различные социальные сети, которые при наведении мыши раскрываются и оживают с помощью нескольких CSS3-эффектов и мягкого перехода (**рис. 3.04**).

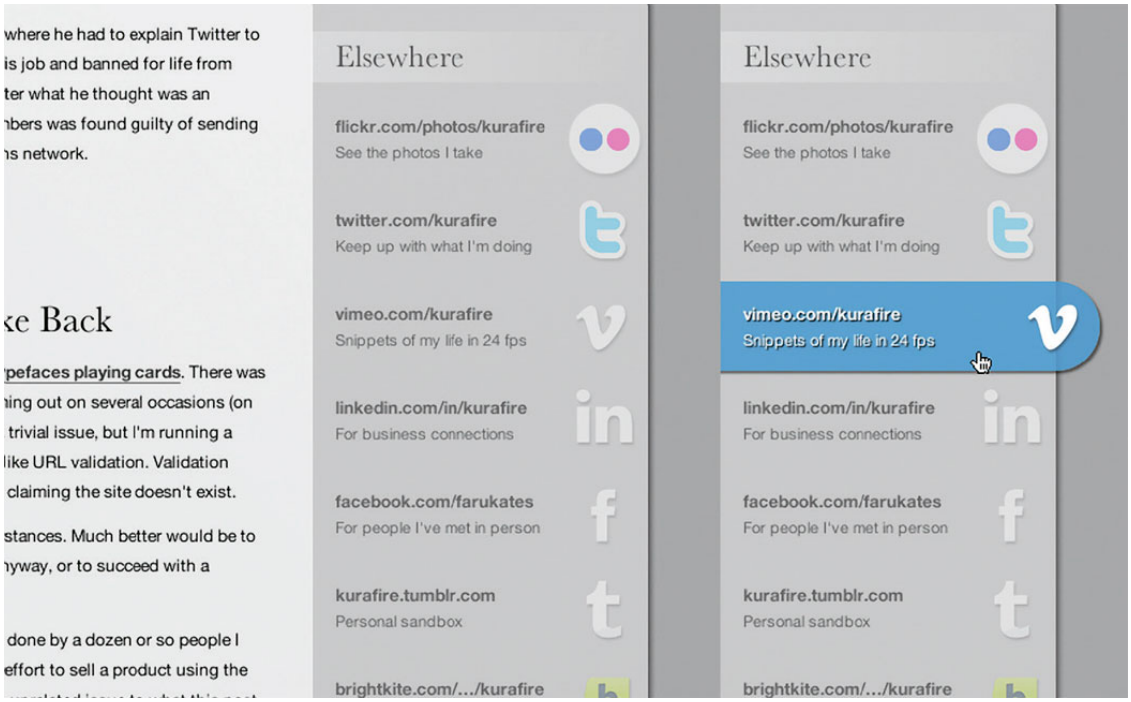

Рис. 3.04. Боковое меню и реакция на наведение мыши – сайт Фарука Атеса

То, что выглядит обычным сдвинутым вправо списком, состоящим из текста и картинок, оказывается намного более интересным объектом, когда начинаешь взаимодействовать с ним. Это важнейший пример обогащенного взаимодействия, и Фарук использует разнообразные свойства CSS3, чтобы достичь этого эффекта.

Рис. 3.05 показывает, как выглядят состояние при наведении и обычное состояние в Internet Explorer 7, который никак не поддерживает CSS3. Можно заметить, что хоть состояние при наведении и небезупречно, сайтом по-прежнему можно пользоваться, информацию можно прочитать – сайт остается функциональным; не говоря о том, что основное состояние выглядит почти неизменно.

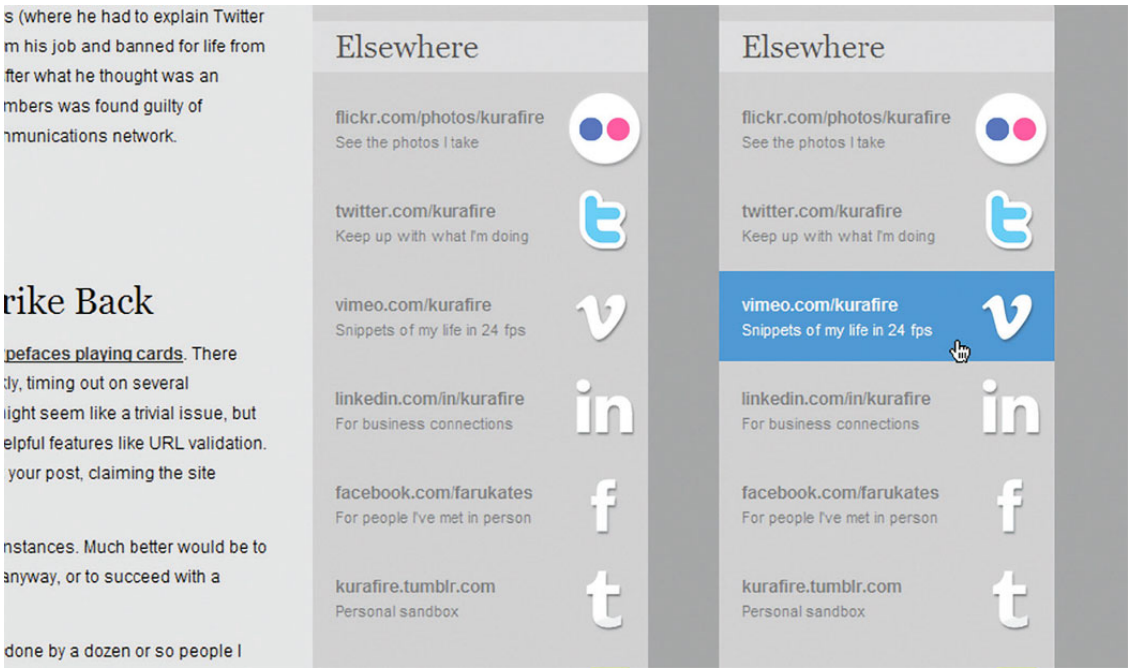

Рис. 3.05. В браузере IE7 сайт Фарука Атеса не демонстрирует тех же визуальных эффектов, но это в порядке вещей

Наведение на элемент (или фокусировка) – прекрасное место сайта, которое можно улучшать средствами CSS3. Пользователи Internet Explorer получат иное взаимодействие (пока в этот браузер не войдет поддержка свойств CSS3). Но это взаимодействие ничем не хуже, и, откровенно говоря, пользователи IE не узнают о том, что они упускают.

То есть – до тех пор, пока не откроют сайт на компьютере друга, где установлен браузер Safari, Chrome, Firefox или Opera (и почувствуют подступающую зависть).

Должны ли сайты выглядеть полностью одинаково в каждом браузере?

Это важный вопрос (и вполне подходящий для того, чтобы задать его сейчас), и я пробую ответить на него на сайте с невероятно длинным доменным именем (**рис. 3.06**): http://dowebsitesneedtobeexperiencedexactlythesameineverybrowser.com.

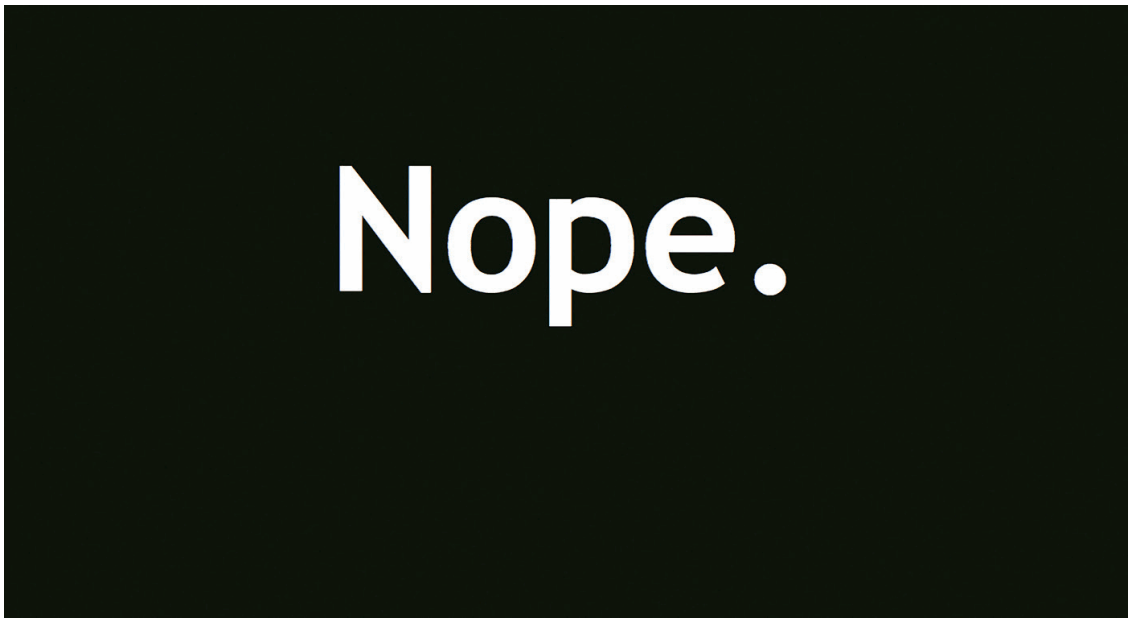

Рис. 3.06. Необычно названный http://dowebsitesneedtobeexperiencedexactlythesameineverybrowser.com

Как и пример Фарука, этот сайт становится интересным лишь тогда, когда начинаешь взаимодействовать с ним. На поверхности он выглядит практически одинаково во всех браузерах, но когда начинаешь двигать мышкой, прикасаясь к тексту и к фону, применяется набор свойств CSS3, переходов и трансформаций, чтобы сделать взаимодействие особенным и запоминающимся.

Опять-таки мы улучшаем дизайн именно внутри слоя взаимодействия. Основное содержимое, читаемость, юзабилити и разметка остаются одинаковыми и неизменными.

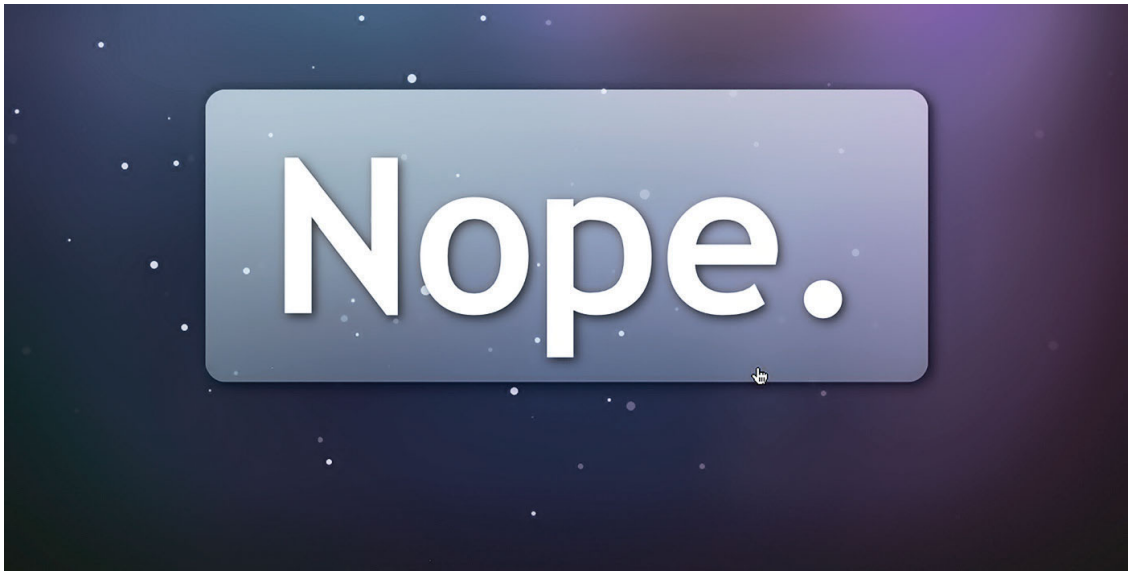

Рис. 3.07. Обогащенное взаимодействие раскрывается, когда начинаешь взаимодействовать с сайтом. Все благодаря CSS3

Навигация на Луне

Применим подход добавления CSS3 к состоянию hover непосредственно в наш пример. Я подробно расскажу каждый шаг на пути создания меню в верхней части сайта (**рис. 3.08**), в котором сочетаются `border-radius`, `text-shadow`, RGBA и CSS-переходы, чтобы построить взаимодействие, которое будет удивлять и восхищать.

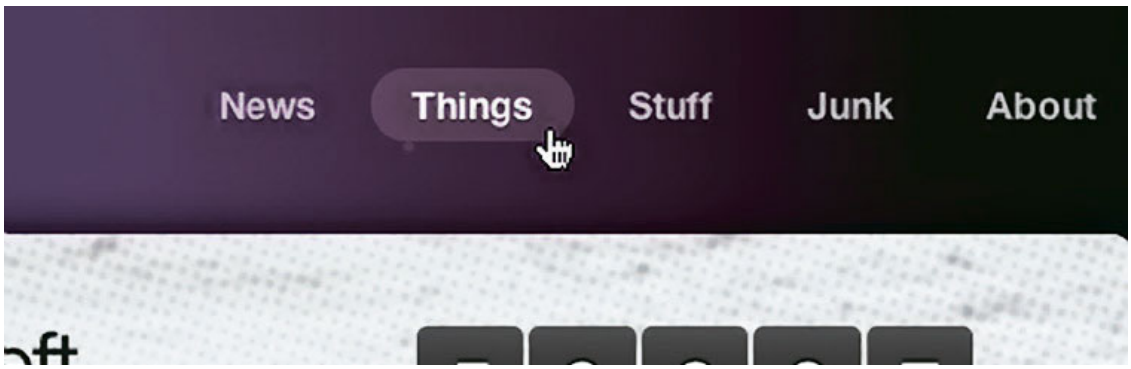

Рис. 3.08. Меню в нашем примере улучшено в состоянии hover за счет CSS3

Сначала разметка

Так как мы приверженцы семантики, то разметка для меню будет состоять из обыкновенного списка.

```
<ul id="nav">
<li><a href="#">News</a></li>
<li><a href="#">Things</a></li>
<li><a href="#">Stuff</a></li>
<li><a href="#">Junk</a></li>
<li><a href="#">About</a></li>
</ul>
```

Разумеется, здесь нет ничего нового – лишь подходящая структура, к которой можно начать применять стили.

Сдвинем элементы

Сначала сдвинем весь список и добавим небольшой отступ, чтобы разместить меню в правой части страницы; затем сдвинем также каждый элемент списка.

```
#nav {
float: right;
padding: 42px 0 0 30px;
}
#nav li {
float: left;
margin: 0 0 0 5px;
}
```

Результат виден на **рис. 3.09**. Теперь список горизонтален.

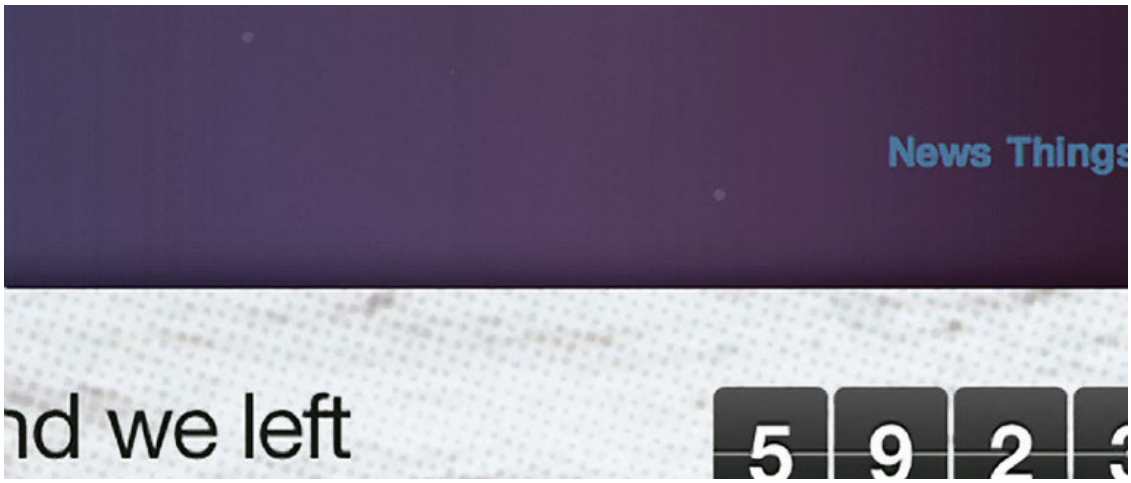

Рис. 3.09. Горизонтальный список ссылок, полученный применением нескольких CSS-правил

Определение цвета ссылки – RGBA

Теперь добавим отступ на каждую ссылку и сменим цвет на полупрозрачный белый. Воспользуемся цветовой моделью RGBA, чтобы указать белый цвет (`255, 255, 255`) и 70% непрозрачности (`0.7`), чтобы текст впитывал в себя часть расположенного за ним фона (**рис. 3.10**).

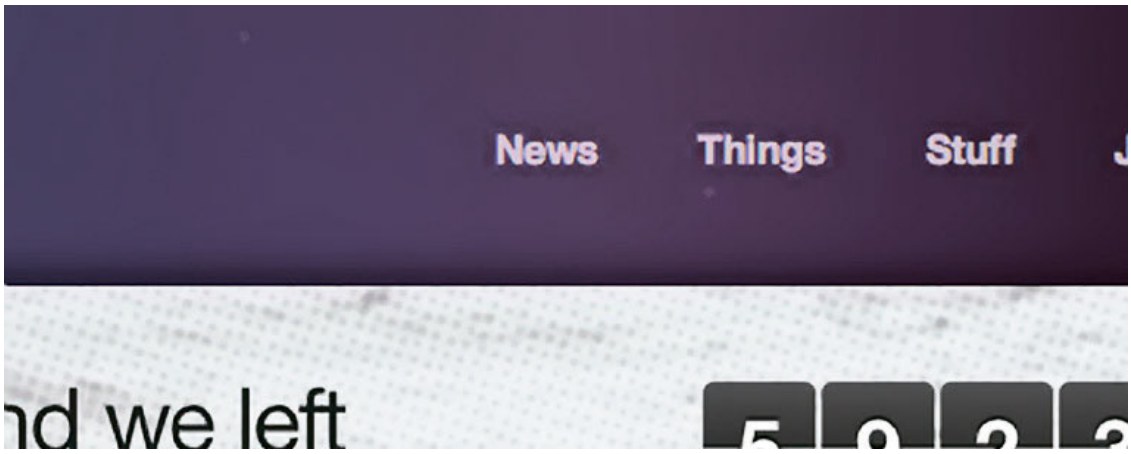

Рис. 3.10. Теперь ссылки стилизованы с помощью RGBA и текст немного смешивается с фоном

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: rgba(255, 255, 255, 0.7);
}
```

Рис. 3.11 показывает ссылки крупным планом. Непрозрачность в 70% делает фон немного просвечивающим.

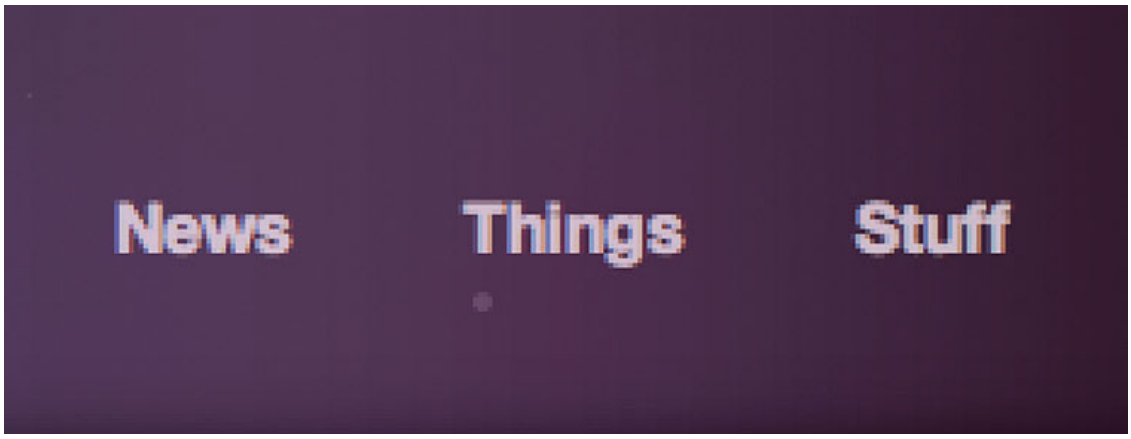

Рис. 3.11. Увеличенный вид полупрозрачных ссылок

Запасной вариант для RGBA

RGBA – удивительно гибкий способ задания цвета и прозрачности, но он не поддерживается всеми браузерами. Safari, Chrome, Firefox, Opera поддерживают его, равно как и Internet Explorer 9, но что насчет IE6-8?

Здесь пригодится запасная цветовая схема. При пользовании RGBA для указания цветов хорошей практикой является сначала указывать только цвет – для тех браузеров, которые не поддерживают RGBA.

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: #ccc;
color: rgba(255, 255, 255, 0.7);
}
```

Браузеры, поддерживающие RGBA, проигнорируют это объявление цвета (светло-серый `#ccc` в этом примере), а браузеры, которые не поддерживают RGBA, проигнорируют RGBA-указание.

Итак, важная вещь, которую следует запомнить: указывайте запасные цвета в RGB для всех цветов, задаваемых в RGBA, в отдельном правиле, которое ставится перед RGBA-объявлением.

Добавим text-shadow

Как последнее добавление к оформлению ссылок, добавим небольшую тень (`text-shadow`). Вновь воспользуемся RGBA, чтобы задать цвет тени: полупрозрачный черный цвет будет смешиваться с цветом фона.

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: #ccc;
color: rgba(255, 255, 255, 0.7);
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.5);
}
```

Рис. 3.12 показывает сравнение ссылок без свойства `text-shadow` (слева) и с ним (справа), как это выглядит в Safari. Эта деталь почти неразличима, и все же крошечная тень приподнимает текст над фоном ровно настолько, насколько нужно.

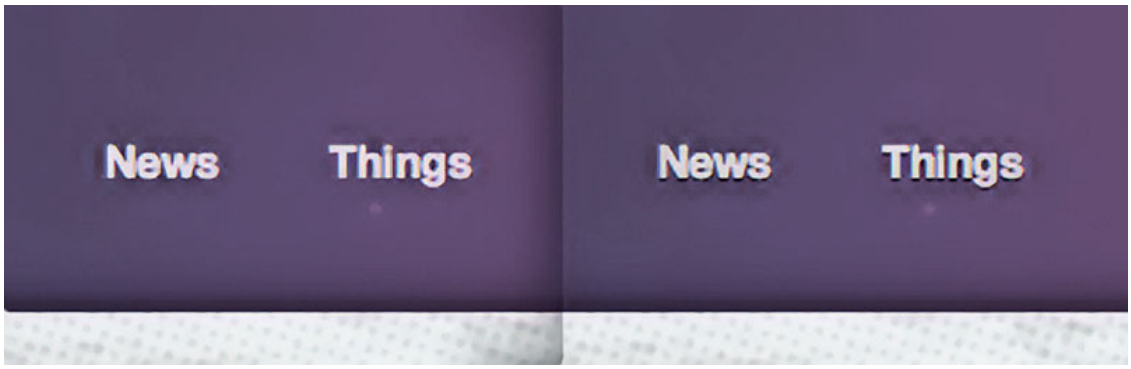

Рис. 3.12. Сравнение ссылок без свойства text-shadow (слева) и с ним (справа)

Помните, что свойство `text-shadow` работает в текущих версиях Safari, Chrome, Firefox и Opera. Браузеры, не поддерживающие `text-shadow` (читай: IE), спокойно проигнорируют это правило. Нет тени – нет проблем.

С готовым `text-shadow` мы можем переходить к оформлению состояния `:hover`. И здесь мы будем в большей мере опираться на CSS3.

Оформление состояний hover и focus

Добавим изменение цвета текста и фона для состояния `:hover` каждой ссылки. Мы вновь применим RGBA, чтобы задать тексту полупрозрачный белый фон.

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: #ccc;
color: rgba(255, 255, 255, 0.7);
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.5);
}
#nav li a: hover,
#nav li a: focus {
color: #fff;
background: rgba(255, 255, 255, 0.15);
}
```

В состоянии `:hover` цвет текста меняется на непрозрачный белый; добавляется фон белый фон с непрозрачностью в 15%. Тот же стиль задается для состояния: focus. Посетители сайта, пользующиеся клавиатурой для перемещения между элементами, увидят изменение цвета текста, когда переводят фокус на ссылку.

Рис. 3.13 показывает ссылки в новом состоянии `:hover` (и `:focus`). Браузеры, поддерживающие RGBA, отобразят полупрозрачный белый фон позади яркого белого текста.

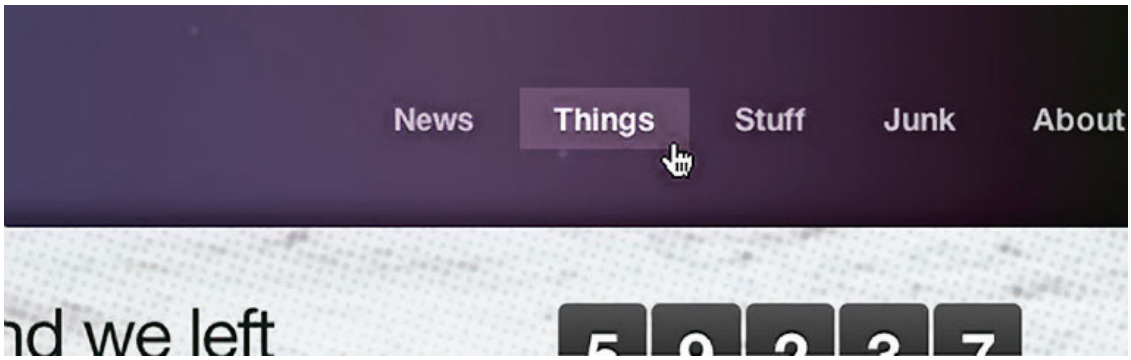

Рис. 3.13. Состояние: hover – теперь с полупрозрачный фоном, полученным с применением RGBA

Скругление углов: border-radius

Следующим шагом мы скруглим углы фона, всплывающего в состоянии: hover, – воспользуемся свойством `border-radius`.

Вспоминая изученное в первой главе о свойстве `border-radius` и о браузерных префиксах, которые позволяют нам использовать это свойство сегодня, мы можем добавить следующие строки к существующим правилам для ссылок:

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: #ccc;
color: rgba(255, 255, 255, 0.7);
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.5);
-webkit-border-radius: 14px;
-moz-border-radius: 14px;
border-radius: 14px;
}
#nav li a: hover,
#nav li a: focus {
color: #fff;
background: rgba(255, 255, 255, 0.15);
}
```

Рис. 3.14 показывает фон ссылки в состоянии `:hover` – теперь со скругленными углами, полученными с использованием свойства `border-radius`. Так страница выглядит в браузерах Safari, Chrome, Firefox, Opera, равно как и в IE9. Помните, что беспрефиксное свойство `border-radius` поставлено в списке последним, чтобы оно получилось приоритет над всеми остальными. Например, Safari 5 поддерживает как беспрефиксное свойство `border-radius`, так и `-webkit-border-radius`, поддерживаемое в Safari 4.

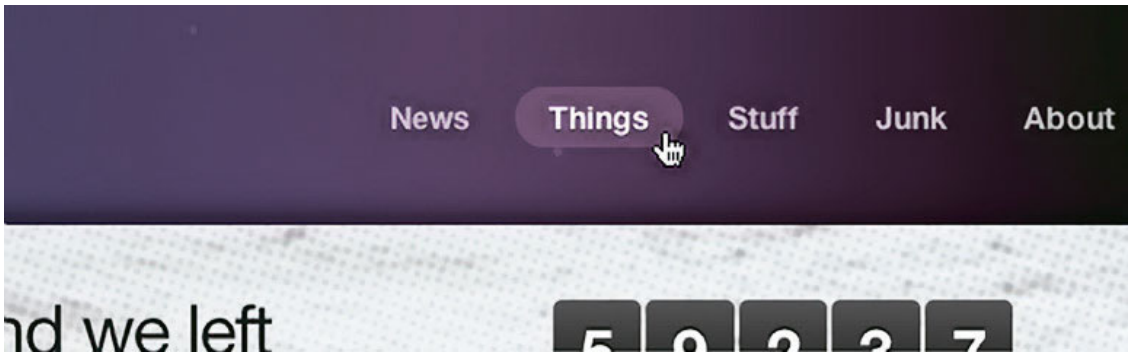

Рис. 3.14. Как скруглить углы, пользуясь свойством border-radius

Возможно, читатель спросит: почему объявление `border-radius` ставится в правило для `#nav li a`, а не `#nav li a: hover` (где скругленные углы появляются на экране)? Ответ кроется в CSS-переходе, который мы сейчас добавим как последний штрих.

Добавим анимацию

Вспомним изученное во второй главе и добавим переход в состояния `:hover` и `:focus` у ссылок в меню. Такой эффект смягчит появление фона: он будет постепенно всплывать позади текста. Переход также сгладит смену цвета текста от полупрозрачного белого до полностью белого (**рис. 3.15**).

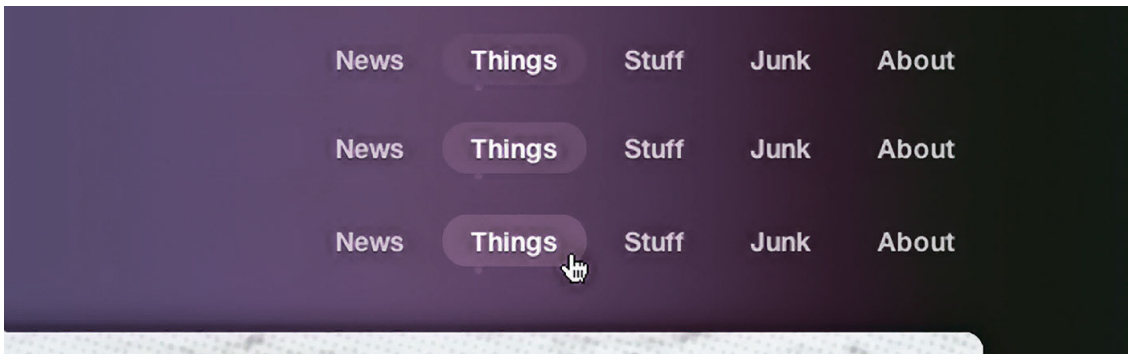

Рис. 3.15. Представьте себе, как постепенно изменяются значения свойств, когда действует переход

В этом примере мы напишем код перехода, работающий в Safari, Chrome, Firefox (4.0) и Opera, и самым последним добавим свойство `transition` без префикса – для свежих версий существующих браузеров и для новых браузеров.

```
#nav li a {
padding: 5px 15px;
font-weight: bold;
color: #ccc;
color: rgba(255, 255, 255, 0.7);
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.5);
-webkit-border-radius: 14px;
-moz-border-radius: 14px;
border-radius: 14px;
-webkit-transition: all 0.3s ease-in-out;
-moz-transition: all 0.3s ease-in-out;
-o-transition: all 0.3s ease-in-out;
transition: all 0.3s ease-in-out;
```

```
}
#nav li a: hover,
#nav li a: focus {
color: #fff;
background: rgba(255, 255, 255, 0.15);
}
```

Помните, что свойства перехода выставляются для обычного состояния элемента, на который накладывается переход. Переходы построены таким способом, чтобы они реагировали не только на состояние `:hover`, но и на `:focus` и `:active`.

В коде анимации используется значение all, чтобы отследить все свойства, которые изменяются в состояниях `:hover` и `:focus`, – в этом примере color и background. Ту же анимацию можно было бы получить другим способом, перечислив все свойства через запятую:

```
-webkit-transition:
color 0.3s ease-in-out,
background 0.3s ease-in-out;
-moz-transition:
color 0.3s ease-in-out,
background 0.3s ease-in-out;
-o-transition:
color 0.3s ease-in-out,
background 0.3s ease-in-out;
transition:
color.3s ease-in-out,
background.3s ease-in-out;
```

Легко заметить, что значение `all` более компактно и эффективно, когда нужно анимировать несколько изменений одного элемента.

Построение взаимодействия

Мы изучили довольно простой пример, добавляя различные свойства CSS3 к элементам нашей страницы, которые относятся исключительно к взаимодействию. Браузеры, поддерживающие эти свойства, будут показывать анимацию полупрозрачного фона со скругленными углами, на котором расположены затененные текстовые ссылки. Браузеры, не поддерживающие CSS3, не отразят улучшенное взаимодействие при наведении на ссылку, но это нормально. Они покажут семантически структурированный список ссылок – и это самое важное в этом примере.

Это маленькое упражнение также показывает, насколько эффективно достигать средствами CSS3 того, для чего раньше требовались Flash и/или JavaScript. Использованные CSS-правила просты, естественны и безвредны для тех браузеров, которые их пока что не поддерживают.

Мы также позаботились о том, что написанный CSS3-код выдержит проверку временем: свойство transition из спецификации включено в список правил последним. Копировать эти правила, помечая их соответствующими префиксами, – это необходимые действия, которые влекут за собой большой выигрыш: мы можем использовать CSS3 прямо сейчас, улучшая взаимодействие для многих пользователей.

Простой и гибкий hover с использованием opacity

Мы постоянно ищем решения, которые экономят время и содержат дополнительную гибкость. CSS3 изобилует такими решениями: эта технология дает возможность написать несколько строчек и получить то, на что раньше уходило больше времени и сил.

В нашем арсенале есть еще одно свойство для улучшения вида элементов в состоянии `:hover: opacity`. Как упоминалось в первой главе, opacity – это свойство CSS3, которым можно явно указать, насколько непрозрачным будет тот или иной элемент. В сочетании с вышеупомянутым RGBA `opacity` предлагает еще один способ добавить прозрачность в дизайны, которые мы создаем для веба.

Один из способов применять `opacity`, который нравится мне, – создавать простое и гибкое состояние: hover для графических ссылок, пользуясь изменением прозрачности для создания нескольких состояний из одного изображения. Прибавим к этому CSS-переход и получим прекрасное, яркое, легко поддерживаемое взаимодействие для графических ссылок на странице.

Посмотрим, как свойство `opacity` применяется в примере с Луной.

Прозрачность на кликабельных картинках

На рис. 3.16 показан футер сайта-примера, в котором – после некоторого юридического текста и шокирующего дисклеймера – расположились три логотипа, на них можно нажать.

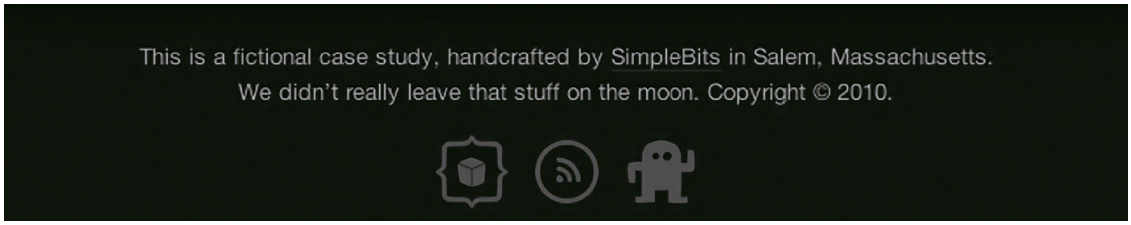

Рис. 3.16. Футер сайта Вещи, оставленные на Луне

Воспользуемся свойством `opacity` не только для того, чтобы определить состояния `:hover` и `:focus`, но и чтобы задать начальный уровень прозрачности. CSS-переход сгладит и анимирует это изменение, чтобы довершить эффект.

Разметка

Как и в предыдущем примере с навигацией, разметка для логотипов в футере проста и семантична – ненумерованный список, состоящий из картинок-ссылок:

```
    <ul id="footer-logos">
    <li><a  href="#"><img  src="img/logo-sb.png"  alt="SimpleBits
logo" /></a>
    </li>
    <li><a href="#"><img src="img/icon-feed.png" alt="RSS feed" /
></a>
    </li>
    <li><a href="#"><img src="img/icon-bitman.png" alt="BitMan" /
></a>
    </li>
```

```
</ul>
```

Прозрачность и эффективность картинки

Я создал иконки полностью белыми PNG-изображениями, зная, что затем применю свойство `opacity`, чтобы отрегулировать прозрачность на уровне CSS. Такой подход изменил то, как я в некоторых ситуациях думаю о графических объектах в веб-проектах.

Вместо того чтобы сохранять полупрозрачные PNG, я сохраняю полностью непрозрачные версии (**рис. 3.17**), которые могу менять в браузере. Кроме того что такой подход экономит время, он также позволяет выставлять разные уровни прозрачности в состояниях `:hover`, `:focus` и `:active`, избавляя от необходимости создавать несколько наборов изображений.

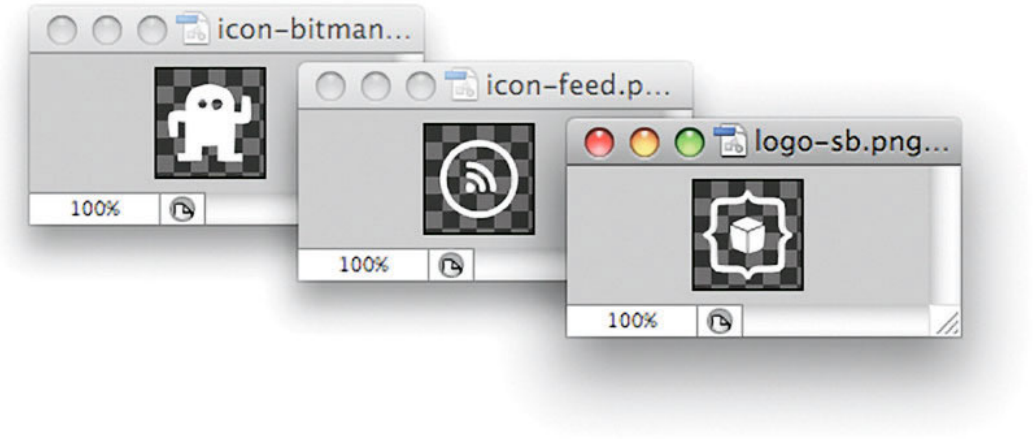

Рис. 3.17. PNG-файлы с логотипами – полностью белые

Оформление списка

Первые фрагменты оформления отцентрируют картинки в футере и поместят их горизонтально (**рис. 3.18**):

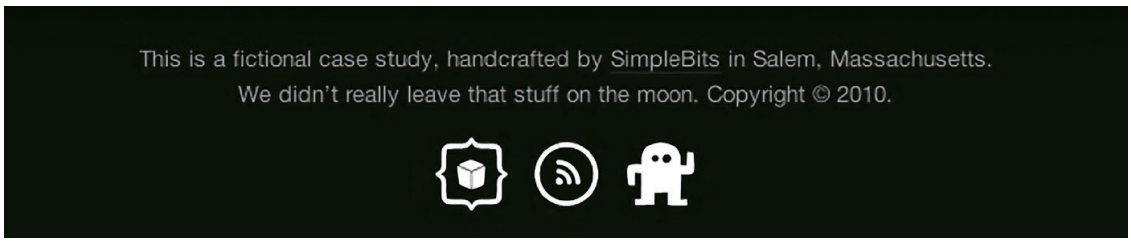

Рис. 3.18. Белые PNG, отцентрированные в футере

```
#footer-logos {
text-align: center;
}
#footer-logos li {
display: inline;
}
```

Теперь выставим такие значения свойства `opacity`, которые затемнят иконки в их обычном состоянии и будут осветлять их в состояниях `:hover` и `:focus`.

```
#footer-logos a img {
```

```
opacity: 0.25;
}
#footer-logos a: hover img,
#footer-logos a: focus img {
opacity: 0.6;
}
```

Мы выставили картинкам непрозрачность в 25%; при наведении или получении фокуса непрозрачность поднимается до 60% (**рис. 3.19**). Крайне просто, не так ли? И для такого эффекта нужен лишь один набор картинок.

Заметим, что свойство `opacity` не требует браузерного префикса и работает в Safari, Chrome, Firefox и Opera. IE8 и ниже не поддерживает `opacity`, но существует хакерское решение для тех, кто не прочь окунуться в дебри проприетарности.

opacity: хак для IE

К счастью, `opacity` поддерживается в Internet Explorer 9 Beta, но мы также можем сымитировать тот же результат в ранних версиях IE, воспользовавшись проприетарным свойством `filter` от Microsoft.

Обычно я бы не стал предлагать использовать свойство `filter`, потому что (в отличие от свойств с браузерными префиксами) оно не входит ни в один предложенный стандарт. Также использование `filter` может повлечь за собой существенные проблемы производительности в зависимости от того, где и как часто оно используется. Это хак – но он решает проблему.

При условии, что вышесказанное понятно, и выделяя это свойство в отдельный стилевой файл или иначе аккуратно комментируя его, `filter` можно воспринимать как приемлемый метод.

Вот как он работает:

```
#footer-logos a img {
border: none;
opacity: 0.25;
-ms-filter:                          "progid:
DXImageTransform.Microsoft.Alpha(Opacity=25)"; /* IE 8 hack */
filter: alpha(opacity = 25); /* IE 5-7 hack */
}
#footer-logos a: hover img,
#footer-logos a: focus img {
opacity: 0.6;
-ms-filter:                          "progid:
DXImageTransform.Microsoft.Alpha(Opacity=60)"; /* IE 8 hack */
filter: alpha(opacity = 60); /* IE 5-7 hack */
}
```

Синтаксис похож: значение opacity вставляется в альфа-фильтр IE. Обратите внимание: IE8 игнорирует свойство `filter` и требует префиксное свойство `-ms-filter` с некоторыми дополнительными некрасивыми значениями.

Воспользовавшись хаками, мы добились такого же результата в IE (**рис. 3.20**). Повторюсь – если решили использовать хаки, пользуйтесь ими с умом. Но реальность заключается в том, что, скорее всего, вам придется использовать хаки, если какой-либо ваш сайт показывает значительный трафик с IE (так обстоит ситуация с большинством сайтов).

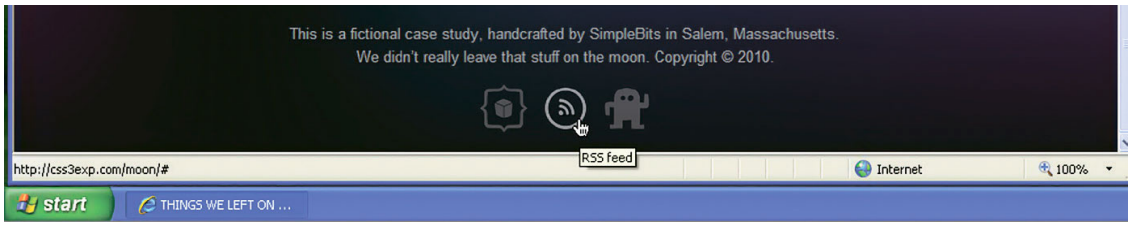

Рис. 3.20. Футер в IE7 показывает псевдосвойство opacity с использованием хака filter

Добавим переход

Наконец, наложим переход на свойство `opacity`, который сгладит изменение свойства и добавит немного приятной глазу анимации, дополняющей эффект.

Добавим знакомые строки `transition`, на этот раз объявляя переход исключительно для свойства `opacity`. Сделаем его быстрым (0,2 секунды) и выставим временную функцию ease-in-out.

```
    #footer-logos a img {
    opacity: 0.25;
    -ms-filter:                                          "progid:
DXImageTransform.Microsoft.Alpha(Opacity=25)"; /* IE 8 hack */
    filter: alpha(opacity = 25); /* IE 5-7 hack */
    -webkit-transition: opacity 0.2s ease-in-out;
    -moz-transition: opacity 0.2s ease-in-out;
    -o-transition: opacity 0.2s ease-in-out;
    transition: opacity 0.2s ease-in-out;
    }
    #footer-logos a: hover img,
    #footer-logos a: focus img {
    opacity: 0.6;
    -ms-filter:                                          "progid:
DXImageTransform.Microsoft.Alpha(Opacity=60)"; /* IE 8 hack */
    filter: alpha(opacity = 60); /* IE 5-7 hack */
    }
```

С использованием перехода мы получили простой метод использования opacity для создания гибкого взаимодействия в состоянии hover на основе лишь одного набора изображений.

Вперед, к новому hover

Ранее я упоминал, что такой подход повлиял на то, как я думаю о подготовке графики для сайтов. Мы можем пользоваться свойством `opacity`, чтобы управлять тем, как графика выглядит в обычном состоянии, смешивая ее с фоном – и затем применять другие значения для состояний `:hover`, `:focus` и `:active`. Мы также можем сочетать это изменение с CSS3-переходом для тех браузеров, в которых они поддерживаются.

Помните о свойстве `opacity` в следующий раз, когда вы будете создавать отдельный вид картинок-ссылок в состоянии hover. Вы выиграете время, трафик и избавитесь от лишней сложности, которая может содержаться в других решениях.

Оформление состояния hover средствами CSS3 заключается прежде всего в добавлении простых стилей, которые расширяют взаимодействие, удивляя и восхищая пользователей – пользователей тех браузеров, в которых поддерживаются эти свойства. Если браузер не поддерживает созданное высококачественное взаимодействие, это нормально – посетитель сайта не узнает, что он упускает.

4. Преобразовывая содержимое

Как и CSS-переходы, CSS-трансформации были изначально разработаны командой WebKit, и затем их включили в два отдельных рабочих черновика W3C:

1. CSS 2D-преобразования (http://www.w3.org/tR/Css3-2d-transforms/);
2. CSS 3D-преобразования (http://www.w3.org/tR/Css3-3d-transforms/).

В этой книге мы поговорим исключительно о 2D-преобразованиях, так как прямо сейчас они больше всего пригодны для использования. Знанием о 3D-преобразованиях можно написать отдельную книгу, и они прекрасны, волшебны. Но у 2D-преобразований (как и у переходов) лучше обстоят дела с поддержкой браузерами: Safari 3.2, Chrome 3.2, Firefox 4.0, Opera 10.5.

Итак, что такое CSS-преобразования? W3C описывает их так:

> CSS 2D-преобразования позволяют преобразовывать на плоскости элементы, отображенные посредством CSS9[9].

Здорово, все понятно. Лучший способ понять, что такое преобразования, – увидеть их в действии.

Давайте сначала посмотрим на простой пример: будем применять различные плоские преобразования к небольшой фотогалерее. Затем (в этой главе) мы воспользуемся теми же приемами на практике на сайте о Луне.

[9] http://www.w3.org/tR/Css3-2d-transforms/#abstract

Масштабирование

Возьмем горизонтальный список из трех фотографий с небольшой рамкой, на которых запечатлена недавняя поездка на Мартас-Виньярд, небольшой остров у побережья Массачусетса (**рис. 4.01**). Сетка из изображений-ссылок – вполне обыкновенный дизайнерский прием.

Рис. 4.01. Сетка из трех фотографий-ссылок

Для разметки мы вновь воспользуемся привычным ненумерованным списком:

```
<ul class="gallery">
<li><a href="#"><img src="photo-1.jpg" /></a></li>
<li><a href="#"><img src="photo-2.jpg" /></a></li>
<li><a href="#"><img src="photo-3.jpg" /></a></li>
</ul>
```

На **рис. 4.02** видно, как выглядит список без применения стилей. Обратите внимание, что изображения существенно крупнее, чем нам нужно. Это сделано намеренно: мы воспользуемся CSS, чтобы уменьшить их.

- 
- 
- 

Рис. 4.02. Список из крупных фотографий – до применения CSS

Добавим стиль

Добавим немного CSS, чтобы превратить этот вертикальный список в горизонтальную сетку; поставим обводку толщиной в один пиксель вокруг каждой картинки (также заметим, что фон страницы – легкий серый `#eee`).

```
ul.gallery li {
float: left;
margin: 0 10px;
padding: 10px;
border: 1px solid #ddd;
list-style: none;
}
ul.gallery li a img {
float: left;
width: 200px;
}
```

Мы сдвинули элементы списка, отключили маркеры `list-style` и обернули каждый элемент `li` в серую однопиксельную линию. Мы также сдвинули сами картинки и уменьшили их до 200 пикселей в ширину.

Эти компактные правила дают нам то, что мы хотели получить (**рис. 4.01**).

Масштабирование в hover

Теперь – преобразования. Добавим преобразование `scale`, чтобы фотография увеличивалась, когда на нее наводят курсор. Помним о том, что исходные изображения больше, чем 200 пикселей в ширину, – значит, фотографии можно увеличивать без потери качества.

Масштабирование поддерживается в Safari, Chrome, Firefox и Opera; везде нужен браузерный префикс. Добавим свойства, которые работают в этих и всех будущих браузерах:

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5);
-moz-transform: scale(1.5);
-o-transform: scale(1.5);
transform: scale(1.5);
}
```

Когда ссылки переходят в состояние hover, мы говорим: «увеличь картинки в полтора раза относительно их изначального размера» (200px в ширину).

Команда `scale(2)` увеличит фотографию вдвое; `scale(0.5)` уменьшит ее вдвое.

Результат показан на **рис. 4.03** (браузер – Safari). Заметим, что `transform` не влияет на остальные части документа и увеличивает фотографию относительно центра, не затрагивая расположения элементов вокруг нее.

Рис. 4.03. Фото в центре увеличено с помощью CSS-трансформации

Также, если нужно, можно задать transform-origin, который объявит, относительно какой части картинки ее нужно масштабировать: верх, низ, центр или положение, обозначенное в процентах (см. http://bkaprt.com/css3/8/)[10].

Например, чтобы фотография увеличивалась относительно нижнего левого края, а не относительно центра, пишется такой код:

```
ul.gallery li a: hover img {
-webkit-transform-origin: bottom left;
-moz-transform-origin: bottom left;
-o-transform-origin: bottom left;
transform-origin: bottom left;
-webkit-transform: scale(1.5);
-moz-transform: scale(1.5);
-o-transform: scale(1.5);
transform: scale(1.5);
}
```

Подходящая тень

Можем пойти дальше и оттенять картинку, когда на нее наводят курсор. Это уместное использование CSS3-свойства `box-shadow`, так как мы заставляем увеличенную фотографию выглядеть так, как будто она приподнимается над страницей.

Заметим: падающая тень – это изысканный эффект, который часто используется сверх меры неразборчивыми дизайнерами. Очень легко увлечься и переусердствовать с получаемым эффектом. Но в этом случае мы пытаемся придать глубины увеличенной фотографии, так что должно получиться вполне хорошо.

Синтаксис box-shadow совпадает с синтаксисом свойства `text-shadow`, которое мы применяли в третьей главе. Однако в отличие от `text-shadow`, чтобы box-shadow заработал в Safari, Chrome и Firefox, ему требуются браузерные префиксы (Opera 10+ и IE9 Beta поддерживают беспрефиксное свойство `box-shadow`). Запишем эти правила.

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5);
-moz-transform: scale(1.5);
-o-transform: scale(1.5);
transform: scale(1.5);
-webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
-moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
```

[10] http://www.w3.org/tR/css3-2d-transforms/#transform-origin

```
box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
}
```

Мы добавили свойство `box-shadow` для браузеров WebKit и Mozilla, дополнив объявление беспрефиксной версией, как и в остальных примерах.

В терминах синтаксиса мы добавляем тень к картинке в состоянии hover. Параметры тени: `4px` сверху, `4px` слева, размытие радиусом `10px`, полупрозрачный черный цвет (чтобы тень смешивалась с любым фоном или элементом, который находится позади нее).

Рис. 4.04 показывает тень в сочетании с масштабированием – эффекты, которые применяются, когда фотография переходит в состояние hover. Получается так, словно увеличенная фотография выскальзывает из страницы.

Рис. 4.04. Увеличенная фотография в состоянии hover, под действием box-shadow

Сгладим масштабирование переходом

Напоследок, добавив переход на фотографию, мы сгладим масштабирование, навесив на состояние `:hover` анимированное увеличение и уменьшение фотографии. Такой эффект раньше можно было воспроизвести только на основе Flash или JavaScript; теперь он достигается несколькими строчками кода на CSS3.

Вот код, задающий переход; он добавлен к полному CSS-коду этой небольшой фотогалереи:

```
ul.gallery li {
float: left;
margin: 0 10px;
padding: 10px;
border: 1px solid #ddd;
list-style: none;
}
ul.gallery li a img {
float: left;
width: 200px;
-webkit-transition: – webkit-transform 0.2s ease-in-out;
-moz-transition: – moz-transform 0.2s ease-in-out;
transition: transform 0.2s ease-in-out;
}
ul.gallery li a: hover img {
-webkit-transform: scale(1.5);
-moz-transform: scale(1.5);
-o-transform: scale(1.5);
transform: scale(1.5);
```

```
-webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
-moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
}
```

Обратим внимание, что на этот раз мы применяем переход к преобразованию `scale`, поэтому подходящие браузерные префиксы записываются для обоих свойств – `transition` и `transform`.

Преобразовывая взаимодействие

Результат получился довольно впечатляющим, учитывая совсем небольшое количество написанного CSS-кода. Бóльшая часть эффекта достигается непосредственно за счет браузеров, которые поддерживают CSS-свойства, – вместо того, чтобы привлекать такие технологии как Flash или JavaScript.

Как и прежде, та часть сайта, которую мы решили полностью реализовывать на CSS3 в этом конкретном примере, – это уровень взаимодействия: когда на фотографию наводят курсор, мы предлагаем улучшенное представление. Отсутствие такого эффекта некритично для тех браузеров, которые не поддерживают эти свойства.

Например, пользователи Internet Explorer увидят лишь фотогалерею, составленную из маленьких изображений, на которые можно нажимать; это нормально. Если бы вид и поведение элементов в состоянии hover были бы критичны, тогда нам потребовалось бы переосмыслить использование CSS3.

Поворот, кручение, сдвиг

Кроме масштабирования, есть еще три преобразования, которыми можно поворачивать, крутить и сдвигать элементы (сдвиг производится по координатам x, y). Добавим каждое преобразование в получившуюся фотогалерею, чтобы быстро разобраться, как они работают.

Добавим поворот

Допустим, нам нужно поворачивать фотографию, когда на нее наводят курсор, одновременно с этим масштабируя ее, как и ранее. Мы можем добавить преобразование rotate к правилу `:hover`:

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5) rotate(-10deg);
-moz-transform: scale(1.5) rotate(-10deg);
-o-transform: scale(1.5) rotate(-10deg);
transform: scale(1.5) rotate(-10deg);
-webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
-moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
}
```

Мы по-прежнему увеличиваем фотографию в состоянии hover, но также поворачиваем ее на 10 градусов влево преобразованием `rotate` (**рис. 4.05**). Оно работает в Safari, Chrome, Firefox и Opera. Отрицательные значения от `-1deg` до `-360deg` поворачивают элемент против часовой стрелки; положительные значения от `1deg` до `360deg` – по часовой стрелке.

Рис. 4.05. Фото в состоянии hover, увеличенное и повернутое влево при помощи преобразования rotate

Кроме того, можно было бы поворачивать фотографии по-разному (задавая каждой фотографии свой угол поворота), чтобы каждая выглядела так, будто бы ее не глядя кинули на стол. Затем ее точно так же можно поворачивать и масштабировать в состоянии `:hover` (**рис. 4.06**).

Рис. 4.06. С помощью rotate можно представить фотографии разбросанными по странице.

В этой небольшой книге я подчеркиваю, что самое подходящее место для использования CSS3 – уровень взаимодействия, но это не означает, что нельзя пользоваться этими приемами в стандартном представлении дизайна. Важно, чтобы используемые приемы не были критичными для пользователя и чтобы сайт выглядел приемлемо в менее современных браузерах.

Например, если браузер не поддерживает преобразование rotate и фотографии выглядят ровными, в этом нет беды. Сайт по-прежнему будет выглядеть работающим.

Нет поворота? Паника ни к чему

Хороший пример использования `rotate` в основном дизайне странице можно найти в блоге компании Panic Software (http://www.panic.com/blog), где применяются крайне малозаметные CSS3-повороты, которые поворачивают записи налево, как будто это листы бумаги, оставленные на столе (**рис. 4.07**). Такой эффект – не критически важная часть дизайна, и без поворота этот сайт также выглядит прекрасно (**рис. 4.08**).

Рис. 4.07. В дизайне блога Panic Software используются небольшие CSS3– повороты, чтобы добавить реалистичности

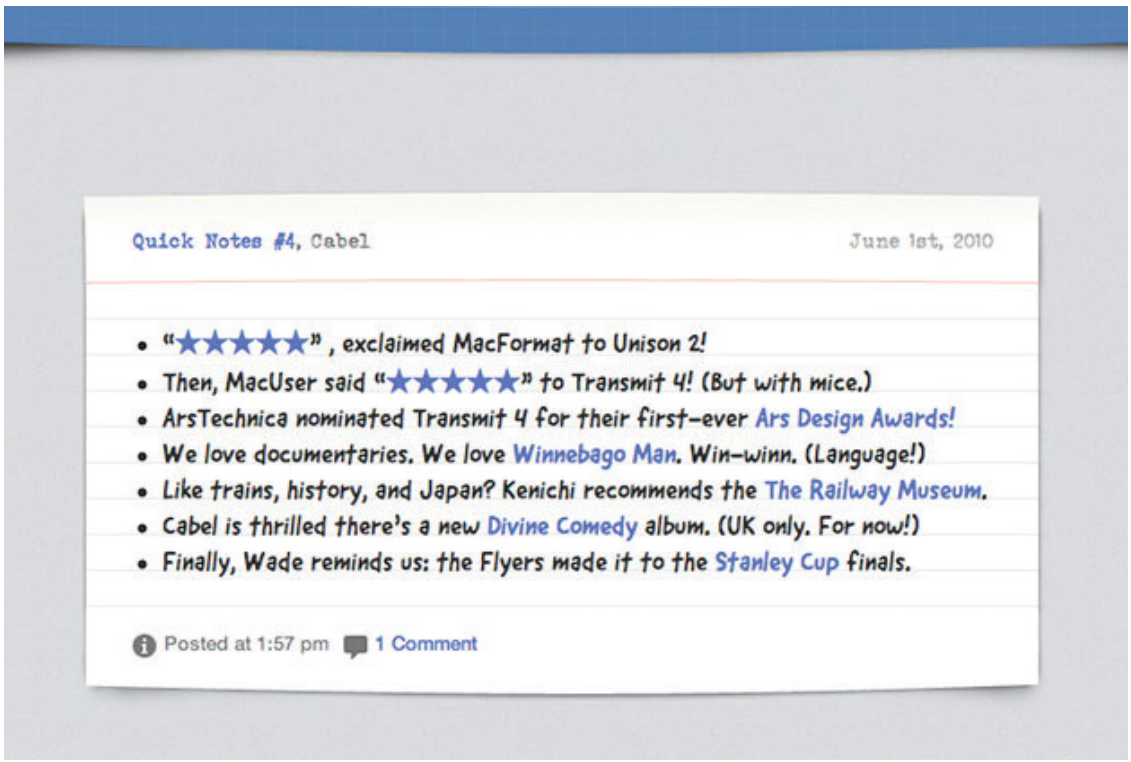

Рис. 4.08. Без поворота блог по-прежнему выглядит замечательно

Повернутое Солнце

Другой хороший пример подходящего использования CSS-преобразований – сайт Outside (http://outsideapp.com), прекрасного приложения о погоде для iPhone (**рис. 4.09**).

Рис. 4.09. Outside, приложение для iPhone, использует rotate, чтобы вращать Солнце

В самом верху страницы показывается изображение Солнца (**рис. 4.10**), которое вращается на 360° с помощью преобразования `rotate`. (В этом случае JavaScript используется для анимации поворота в браузерах, работающих на отличном от WebKit движке, но в шестой главе мы поговорим об анимациях, построенных на чистом CSS.) Это изящное улучшение дизайна просто и уместно, так как оно имитирует ту же анимацию Солнца, которая появляется в самом приложении на iPhone. Солнце не вращается в браузерах, которые не поддерживают CSS-преобразования, и это нормально. Ничто не выглядит неработающим в неанимированной версии сайта.

Преобразования в сочетании с переходами в CSS могут помочь поддержать общее сообщение, которое несет дизайн, создаваемый для веба, и это прекрасный инструмент для нас, веб-дизайнеров.

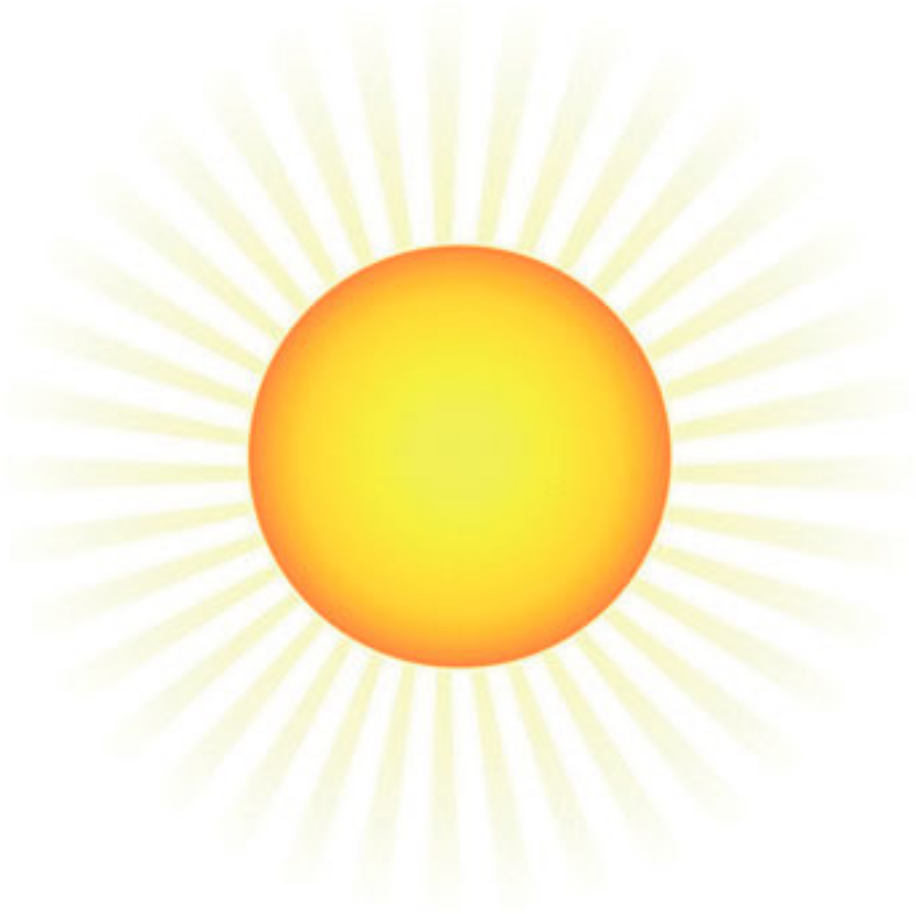

Рис. 4.10. Графика на сайте приложения Outside оживает с помощью позиционирования и вращения на CSS

Кручение (skew)

Преобразование skew берет координаты x, y и прокручивает элемент. Например, если мы хотим применить кручение в нашей фотогалерее, пишется следующий CSS-код. Параметры кручения: –5 градусов по координате x, 30 градусов по координате y (**рис. 4.11**):

Рис. 4.11. Преобразование skew деформирует фотографию

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5) skew(-5deg, 30deg);
```

```
-moz-transform: scale(1.5) skew(-5deg, 30deg);
-o-transform: scale(1.5) skew(-5deg, 30deg);
transform: scale(1.5) skew(-5deg, 30deg);
}
```

Как и `rotate`, преобразование `skew` принимает положительные и отрицательные значения углов. Также можно использовать одно значение и для x, и для y (**рис. 4.12**):

Рис. 4.12. Закручивание фотографии на 30 градусов по обеим осям, x и y

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5) skew(30deg);
-moz-transform: scale(1.5) skew(30deg);
-o-transform: scale(1.5) skew(30deg);
transform: scale(1.5) skew(30deg);
}
```

Разумеется, я осознаю, что полученный результат выглядит не слишком привлекательно, и, признаться, я и сам не использую `skew` слишком часто; однако я убежден, что существуют интересные способы применения этого преобразования.

Например, `skew` может использоваться на текстовых блоках, чтобы создавать трехмерные визуализации на основе семантической разметки и CSS3 (**рис. 4.13 и 4.14**).

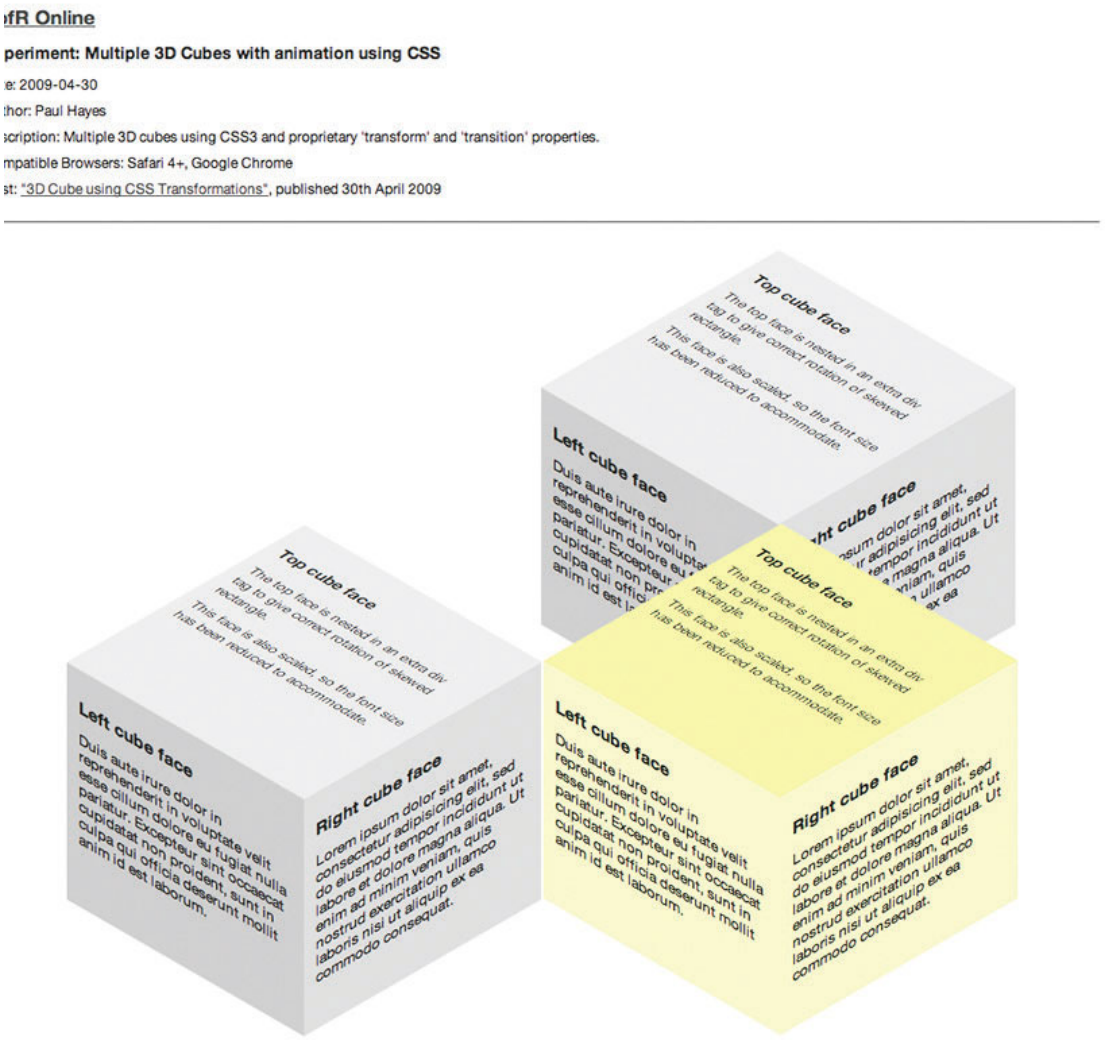

Рис. 4.13. Демо Пола Хэйза использует skew и переходы для создания трехмерных кубов из простых кусков гипертекста (http://www.paulrhayes.com/experiments/cube/multiCubes.html)

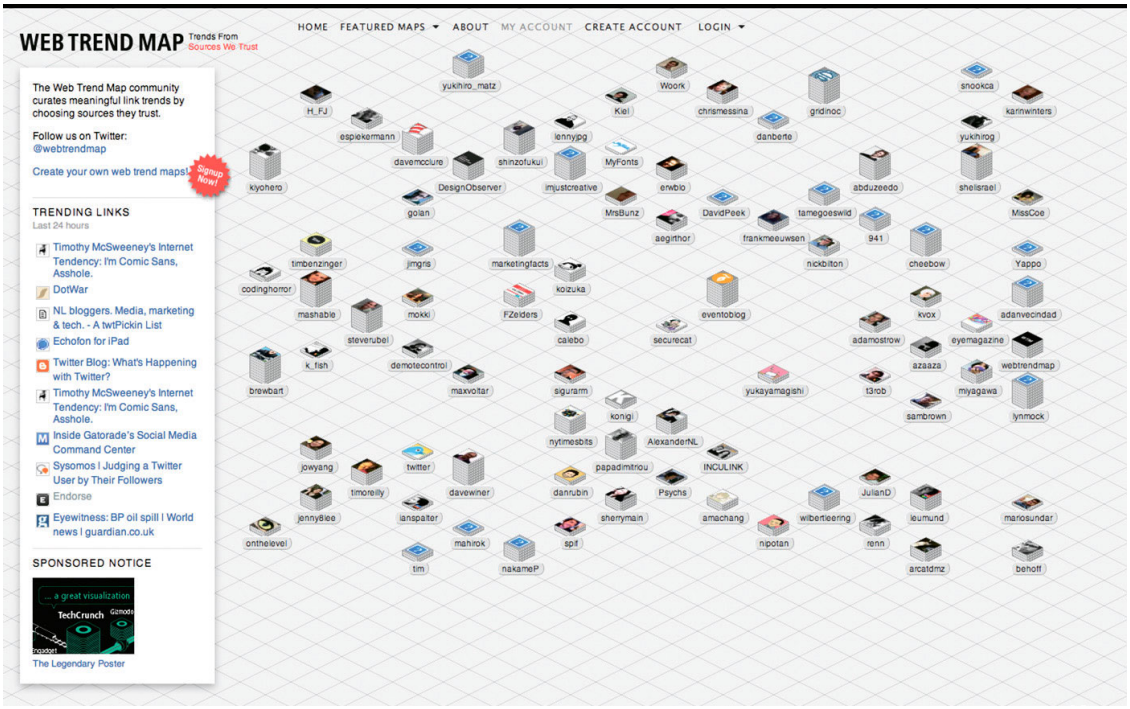

Рис. 4.14. The Web Trend Map использует skew, чтобы разместить аватары на изометрической сетке, таким образом создавая уникальные визуализации данных на основе плоских элементов (http://www.webtrendmap.com/)

Сдвиг (translate)

Наконец, преобразование `translate` позволяет сдвигать элемент относительно его обычного положения на экране, используя координаты x, y.

Например, если в состоянии hover хочется сдвинуть изображение с его начального положения, можно применить преобразование `translate`. Применив переход, такое движение можно плавно анимировать.

Представленный ниже код сдвинет изображение на 20px вправо и на 40px вниз относительно изначального положения (**рис. 4.15**):

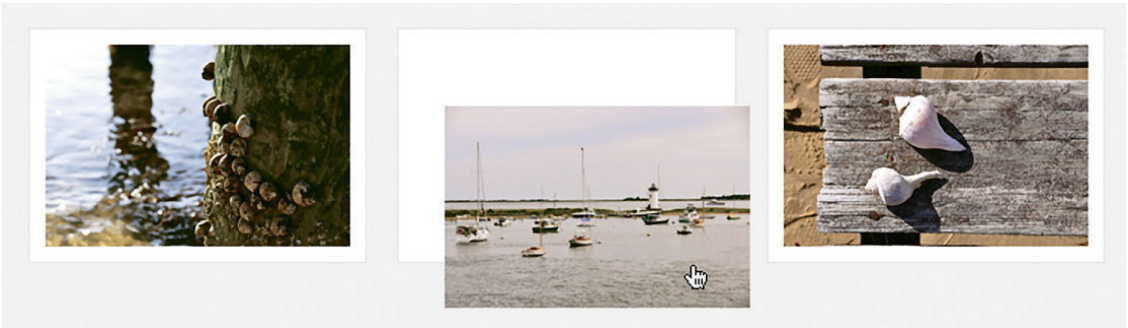

Рис. 4.15. Использование преобразования translate для сдвига изображения в состоянии: hover

```
ul.gallery li a: hover img {
-webkit-transform: scale(1.5) translate(20px, 40px);
-moz-transform: scale(1.5) translate(20px, 40px);
-o-transform: scale(1.5) translate(20px, 40px);
transform: scale(1.5) translate(20px, 40px);
}
```

Если нужно сдвинуть изображение влево и/или вверх, используются отрицательные значения: например, `translate (-20px, -40px)`.

Как и вышеупомянутые преобразования, `translate` не затрагивает остальные элементы в документе и сдвигает указанный элемент именно туда, куда нужно. Это означает, что не нужно думать о полях, отступах, абсолютном позиционировании и об использовании свойства clear для плавающих элементов. Достаточно дать `translate` нужные координаты, и элемент встанет на указанное место.

Разные преобразования, помогающие поддержать рассказ

Пример фотогалереи показал, как преобразования scale, rotate, skew и translate могут сочетаться с переходами, чтобы создавать более яркий дизайн. Ключ к использованию этих преобразований с умом заключается в том, чтобы находить подходящие ситуации, в которых преобразования помогут рассказывать историю того, что показывается на экране.

Опять-таки очень легко увлечься преобразованиями, потому что они классные и их легко использовать. Но крайне важно находить подходящие места для их применения, чтобы получить лучший продукт.

Преобразования Луны

Вернемся к сайту-примеру с Луной, где я использовал различные преобразования и переходы, чтобы оживить взаимодействие с фотогалереей (**рис. 4.16**).

Рис. 4.16. Фотогалерея на сайте Things We Left on the Moon

Когда наводишь на любую оставленную на Луне вещь, изображение реагирует особенным образом, в зависимости от того, что это за предмет: пончик, газонокосилка, кошка и т. д.

Добавлять подходящие преобразования и переходы на каждый элемент не только приятно и легко, но также никак не влияет на браузеры, которые не поддерживают CSS3-механизмы, благодаря которым такое взаимодействие возможно.

Пройдемся по всем элементам и посмотрим, как scale, rotate, позиционирование и opacity сочетаются с переходами, создавая полноценное взаимодействие.

Поддержка сообщения

Если подумать о каждом предмете и в особенности о его значении, можно применить преобразование и/или переход, который поможет раскрыть смысл этого предмета.

Как большой пончик или откидное кресло будут реагировать на взаимодействие? Мы можем выбрать и применить подходящие CSS3-приемы, чтобы улучшить дизайн (**рис. 4.17**).

Рис. 4.17. Предметы, которые будем преобразовывать

Разметка

Семантика разметки этой искусственной карусели странных вещей очень проста: обычный упорядоченный список, составленный из картинок-ссылок, с поясняющим заголовком под каждой картинкой.

```
<ol id="things">
<li id="things-1">
<a href="#"><img src="img/doughnut.png"/></a>
<h2>1 big doughnut</h2>
</li>
<li id="things-2">
<a href="#"><img src="img/mower.png"/></a>
<h2>1 lawnmower</h2>
</li>
<li id="things-3">
<a href="#"><img src="img/cat.png"/></a>
<h2>1 astro cat</h2>
</li>
<li id="things-4">
<a href="#"><img src="img/recliner.png"/></a>
<h2>1 recliner</h2>
</li>
<li id="things-5">
```

```
<a href="#"><img src="img/gnome.png"/></a>
<h2>1 magic gnome</h2>
</li>
</ol>
```

Заметим, что мы назначили идентификатор `#things` самому списку, равно как и отдельный идентификатор каждому элементу списка, так что мы сможем описывать самостоятельные взаимодействия для состояния `:hover` каждого элемента.

Основные стили для каждого предмета

Теперь добавим основной CSS для каждого элемента списка, содержащего изображения. Следующие стили сдвигают элементы, чтобы они расположились горизонтально, задают относительное позиционирование контекста, в котором мы затем расположим изображения при помощи абсолютного позиционирования, и, наконец, добавляют фоновой рамке полупрозрачные скругленные углы.

```
ol#things li {
position: relative;
float: left;
margin: 0 15px 0 0;
padding: 10px;
background: #444; /* backup for non-RGBA */
background: rgba(255, 255, 255, 0.1);
list-style: none;
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
-o-border-radius: 4px;
border-radius: 4px;
}
```

Зададим фоновое изображение Луны, которая будет отображаться позади каждого предмета, и укажем ширину и высоту каждой ссылки (**рис. 4.18**):

Рис. 4.18. Элементы списка, теперь с изображением Луны на фоне

```
ol#things li a {
float: left;
width: 137px;
height: 91px;
background: url(../img/moon-137.jpg)
no-repeat top left;
}
```

Общее правило

Следующий шаг – написать общее правило, которое абсолютно спозиционирует каждое изображение внутри элемента списка и, соответственно, поверх изображения Луны.

Каждый элемент будет спозиционирован по-своему в зависимости от объекта; также будут использоваться различные преобразования. Но мы можем объявить position: absolute; для всех изображений, так что не придется дублировать это правило для каждого элемента. Мы также добавим переход, пользуясь значением `all`. Таким образом каждое преобразование или изменение, которое мы хотим применить на каждый предмет, будет сглажено с помощью перехода – вне зависимости от того, какие CSS-свойства мы решим менять.

```
ol#things li a img {
position: absolute;
-webkit-transition: all 0.2s ease-in;
-moz-transition: all 0.2s ease-in;
-o-transition: all 0.2s ease-in;
transition: all 0.2s ease-in;
}
```

Теперь мы готовы добавить точное позиционирование и ширину для каждой картинки, пользуясь идентификаторами, которые мы ранее назначили каждому элементу списка.

```
ol#things li#things-1 a img {
width: 60px;
top: 23px;
left: 26px;
}
ol#things li#things-2 a img {
width: 50px;
top: 20px;
left: 50px;
}
ol#things li#things-3 a img {
width: 80px;
top: 19px;
left: 30px;
}
ol#things li#things-4 a img {
width: 70px;
top: 25px;
left: 45px;
}
ol#things li#things-5 a img {
width: 80px;
top: 20px;
left: 34px;
}
```

Я создал эти изображения большими, так что если мы хотим масштабировать их, мы можем делать это, не искажая изображения.

Теперь добавим отдельное поведение для состояния `:hover` каждого из элементов, помня о том, что общее правило сгладит и анимирует любые изменения, которые мы придумаем.

Масштабируем большой пончик

Большой пончик становится больше при наведении, так что воспользуемся преобразованием `scale`, чтобы увеличить изображение. Помните, что исходное изображение, заданное в разметке, значительно больше, чем те размеры, которые мы объявили в стилевом файле. Это было сделано намеренно, чтобы изображение можно было увеличить:

```
ol#things li#things-1 a: hover img {
-webkit-transform: scale(1.25);
-moz-transform: scale(1.25);
-o-transform: scale(1.25);
transform: scale(1.25);
}
```

Эти правила увеличат пончик на 25% при наведении курсора. На **рис. 4.19** показаны оба состояния – обычное и: hover; видно, что пончик становится больше.

Рис. 4.19. Большой пончик увеличивается в состоянии: hover с помощью свойства scale

Перспектива: масштабирование и позиционирование

С газонокосилкой мы сделаем две вещи:

1) увеличим ее с помощью преобразования;

2) сдвинем ее вниз и вправо.

Эти два изменения в сочетании с переходом создадут эффект приближения газонокосилки в сторону зрителя (осторожно!). Он совсем незначительный, но простой и эффективный.

Мы будем сдвигаться на 5 пикселей вниз и на 10 пикселей вправо. Также мы добавим преобразование, чтобы увеличить газонокосилку на 20%.

```
ol#things li#things-2 a: hover img {
top: 25px;
left: 60px;
-webkit-transform: scale(1.2);
-moz-transform: scale(1.2);
-o-transform: scale(1.2);
transform: scale(1.2);
}
```

На **рис. 4.20** показаны обычное и активное состояния картинки. Иллюзия приближающейся газонокосилки закончена.

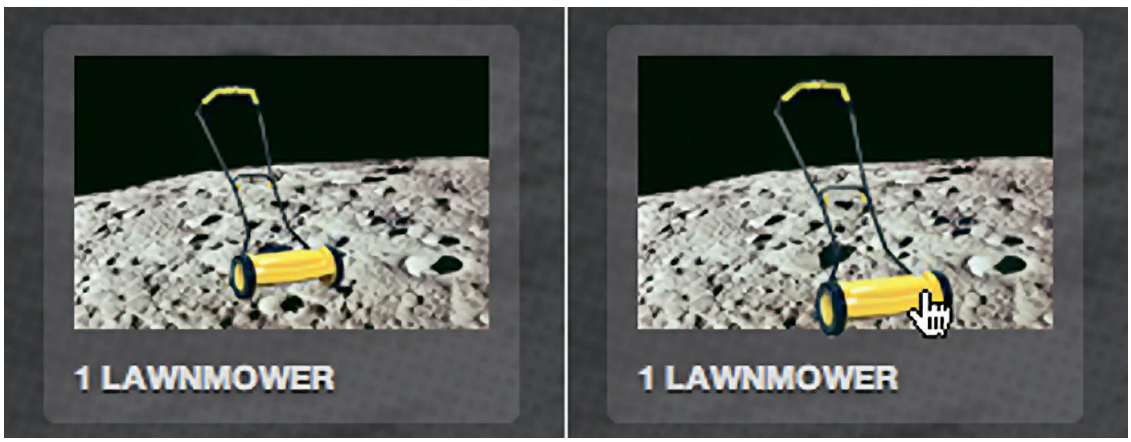

Рис. 4.20. Позиционирование и масштабирование в сочетании дают псевдотрехмерный эффект

Ускользающая космическая кошка

Мы можем добавить CSS-переход на весь набор свойств (не только CSS3). Достаточно сгладить изменение положения, чтобы космическая кошка выглядела так, будто бы она ускользает от мыши.

Когда свойство `left` у изображения в состоянии hover изменяется, общий переход будет сглаживать это изменение и кошка будет выглядеть так, будто бы она скользит из стороны в сторону.

Сдвинем ее на 15px вправо, повышая значение left с `30px` до `45px` (**рис. 4.21**):

Рис. 4.21. Кошка скользит туда и обратно, как она часто делает

```
ol#things li#things-3 a: hover img {
left: 45px;
}
```

Довольно просто. Вся магия здесь осуществляется CSS-переходом (ее сложно изобразить на листе бумаги).

Откидное кресло

Хорошее кресло откидывается назад. Мы можем имитировать это поведение посредством ранее упомянутого преобразования rotate.

Добавим преобразование, которое будет слегка поворачивать кресло влево. Мы будем использовать браузерные префиксы для WebKit, Mozilla и Opera и завершим объявление свойством `transform` – для будущих браузеров.

```
ol#things li#things-4 a: hover img {
-webkit-transform: rotate(-15deg);
-moz-transform: rotate(-15deg);
-o-transform: rotate(-15deg);
transform: rotate(-15deg);
}
```

Мы использовали отрицательное значение, чтобы сдвинуть изображение влево (против часовой стрелки), и, как прежде, переход сгладит этот небольшой поворот, завершая иллюзию комфортного бархатного кресла на Луне (**рис. 4.22**).

Рис. 4.22. Кресло откидывается влево с помощью отрицательного параметра у преобразования rotate

Исчезающий гном

Для последнего элемента списка мы возьмем гнома и сделаем так, чтобы он частично исчезал. Почему-то кажется, что для гнома это чрезвычайно естественный поступок.

Будем использовать свойство `opacity`, чтобы просто и быстро составить стиль для состояния hover этого изображения, делая его существенно более тусклым. Поскольку переход уже определен для изменения всех свойств изображения, изменение значения `opacity` будет анимироваться в браузерах, которые поддерживают переходы, и будет выглядеть как плавное исчезновение нашего маленького друга.

Объявление просто:

```
ol#things li#things-5 a: hover img {
opacity: 0.4;
}
```

На **рис. 4.23** показано, как в состоянии `:hover` гном растворяется в 40% непрозрачности.

Помните: если нужно, чтобы этот эффект точно так же работал в Internet Explorer, можно использовать хак с проприетарным свойством `filter`, описанный в третьей главе.

Рис. 4.23. Гном почти исчезает за счет уменьшения opacity в состоянии: hover

Безопасное упрощение

Как и в примере с фотогалереей, который обсуждался ранее в этой главе, брызги CSS3, которые мы добавляем в этом примере, никак не затрагивают браузеры, которые пока что не поддерживают эти свойства.

В итоге самое важное, что каждый из этих элементов – ссылка, на которую можно нажать. Все остальное – это расширенное взаимодействие, созданное для тех, кто способен увидеть его.

Еще раз: используйте с умом

Потратив немного времени на размышления о смысле содержимого, с которым мы работаем, мы можем выбрать некоторые свойства CSS3, которые функционируют сегодня, вместе с переходами и преобразованиями.

Такие улучшения взаимодействия могут быть знаком по-настоящему искусного веб-мастера: внимание к деталям, которые не все заметят, забота о некритических визуальных событиях и поднятие сообщения на шаг выше привычного уровня. Для браузеров, которые поддерживают эти штуки сейчас, и тех, которые будут поддерживать их в будущем, небольшое количество кода и размышлений вполне стоит того.

Постарайтесь и будьте аккуратны, работая с CSS-преобразованиями. Очень легко увлечься ими, но когда используешь их с умом, они будут играть решающую роль в том, как читатель воспринимает мысль, которую передает сайт.

Побольше «вау-вау», пожалуйста

Говоря о том, что можно увлечься: в следующий раз, когда ваш клиент или босс скажет «Этому дизайну нужно больше “вау”», просто добавьте следующие строки в стилевой файл (и убедитесь, что человек будет просматривать сайт в Safari, Chrome, Firefox или Opera):

```
*:hover {
-webkit-transform: rotate(180deg);
-moz-transform: rotate(180deg);
-o-transform: rotate(180deg);
transform: rotate(180deg);
}
```

Этот маленький кусок CSS3-кода говорит: «Когда наводишь на любой элемент на странице, он развернется на 180 градусов». Попробуйте. Это гарантированный способ произвести сильное впечатление (**рис. 4.24**).

Грустная часть заключается в том, что некоторым клиентам и начальникам это может понравиться.

– Это превосходно! Выкладывайте это!

Эх.

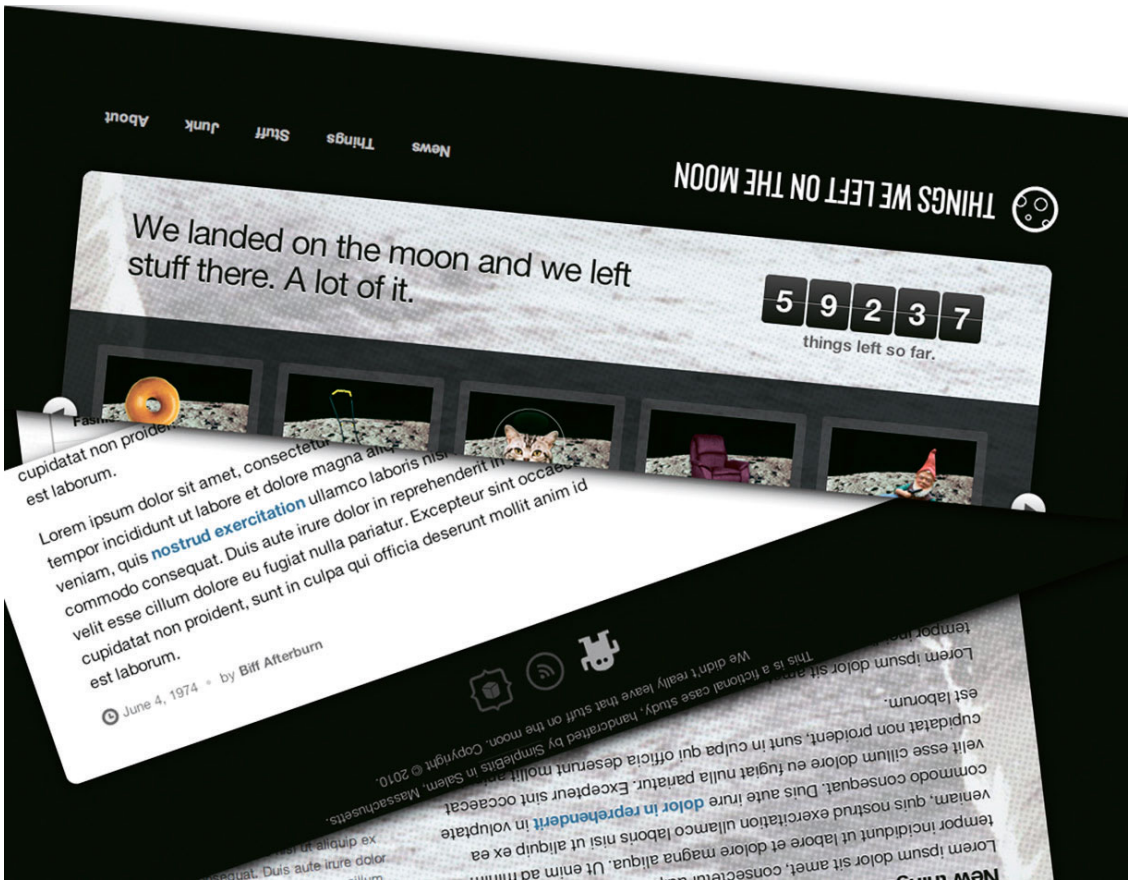

Рис. 4.24. Попытка передать на бумаге тот хаос, который порождается приемом «переворачивать все элементы, которые попадают в состояние hover»

5. Множественные фоны

Если бы два года назад меня спросили «Чего ты больше всего ждешь от CSS3?», я мог бы с энтузиазмом ответить: множественных фоновых изображений! В то время возможность показывать несколько фоновых изображений одного элемента казалась прекрасным лекарством от головной боли, которую испытывали веб-дизайнеры.

Чтобы создавать гибкие и устойчивые решения проблем дизайна, мы должны выяснить, как мы можем обойтись использованием меньшего количества графических элементов или без добавления лишней разметки, чтобы получить несколько фоновых изображений. Мы делали самое лучшее, пользуясь теми средствами, что были под рукой, но обещанная возможность назначать элементу множественные фоновые изображения всегда казалась мне восхитительной – жить станет легче, ведь нужно будет писать меньше кода.

Реальность, впрочем, состоит в том, что с течением времени в браузеры добавили поддержку большинства CSS3-свойств из модуля «Фоны и границы» (http://bkaprt.com/css3/9/)[11]. Многие свойства, которые мы обсуждали ранее в этой книге, хорошо поддерживаются браузерами Safari, Chrome, Firefox, Opera и IE9 Beta. Благодаря свойствам `border-radius`, `box-shadow`, градиентам, RGBA и `opacity` стало возможным в некоторых случаях разрешать часто возникающие задачи, вовсе не используя изображения. Многие приемы, которые ранее требовали использования картинок, стало возможно выполнять исключительно средствами стилевого листа. Все это несет очевидные преимущества.

Итак, несколько лет назад я впадал в восторг при мысли о поддержке множественных фонов; теперь я менее возбужден – потому что в нашем распоряжении сейчас масса других инструментов. При этом существуют изумительные способы использования множественных фонов, и в этой короткой главе мы поговорим об одной такой технике.

[11] http://www.w3.org/tR/Css3-background/

Параллакс

Если вновь обратиться к сайту-примеру с Луной, можно увидеть, как множественные фоновые изображения используются на элементе `body`, чтобы создать составное космическое пространство. Вместо одного плоского изображения используются четыре полупрозрачные PNG, поставленные одна поверх другой. Каждой выставлено свое положение по горизонтали, чтобы создать эффект анимации, когда меняется размер окна браузера (**рис. 5.01**).

Рис. 5.01. Фон на сайте– примере с Луной, где четыре PNG совмещены, чтобы создать эффект космоса

Этот прием быстрой смены слоев обрел название «параллакс», который наши друзья из Википедии определяют так:

> *Особенная техника прокрутки в компьютерной графике, впервые появившаяся в аркаде 1982 года Moon Patrol. В этой псевдотрехмерной технике «камера» двигает фоновые изображения медленнее, чем изображения на переднем плане. Так в двухмерной графике создается иллюзия глубины и погружения. Техника основана на многоплановых камерах, которые используются в традиционной анимации с 1940-х годов.*[12]

В последние годы появилось много великолепных примеров того, как параллакс используется на вебе. Один из моих любимых – сайт Silverback (http://silverbackapp.com), удобное приложение для юзабилити-тестирования от ребят из Clearleft (**рис. 5.02**).

[12] http://en.wikipedia.org/wiki/Parallax_scrolling

Рис. 5.02. Измените размер окна браузера на сайте Silverback и насладитесь трехмерным пользовательским опытом

Изменяя размер окна, можно видеть, как ниспадающие виноградные лозы двигаются туда и обратно с разной скоростью, создавая ощущение объема. (Я просидел на этом сайте час, когда впервые увидел его.)

Разумеется, не все увидят это – но те, кто испытает это, увидят замечательную особенность и улучшенный пользовательский опыт: он не может не сделать зрителя чуточку более счастливым.

Старый способ: дополнительная разметка

Как это сделать? В начале 2008 года в статье для Think Vitamin Пол Аннетт написал о приемах, которые он использовал для создания эффекта параллакса именно на сайте Silverback (http://bkaprt.com/css3/10/)[13].

Чтобы наслоить отдельные PNG одну поверх другой, нужно создать как минимум три доступных элемента блочного уровня. Требуется два дополнительных оберточных слоя, чтобы расположить фоновое изображение на элементах `body`, `#midground`, `#foreground`.

Разметка будет выглядеть приблизительно так:

```
<body>
<div id="midground">
<div id="foreground">
<!- page content here - >
</div>
</div>
</body>
```

CSS-код для размещения трех изображений с различными положениями по горизонтали выглядит так:

```
body {
background: url(vines-back.png) repeat-x 20% 0;
}
#midground {
background: url(vines-mid.png) repeat-x 40% 0;
}
#foreground {
background: url(vines-front.png) repeat-x 150% 0;
}
```

Конечно, это решение работает как следует. Но оно значительно упрощается с использованием синтаксиса множественных фонов, которые приходят вместе с CSS3.

Посмотрим, как множественные фоны применяются к содержимому сайта с Луной и как создается более простой эффект параллакса для тех, кто может испытать его.

[13] http://thinkvitamin.com/design/how-to-recreate-silverbacks-parallax-effect/

Новый способ: множественные фоны на CSS3

Я использую четыре полупрозрачных PNG-изображения, чтобы создать космический фон, используемый на сайте с Луной. Все они накладываются на элемент `body`, одно поверх другого, чтобы создавать ощущение пространства, когда пользователь меняет размеры окна браузера.

На **рис. 5.03** показаны все используемые изображения:

1) облака пыли (clouds.png);
2) сине-розовый градиент (space-bg.png);
3) звездный слой (stars-1.png);
4) еще один слой случайно разбросанных звезд (stars-2.png).

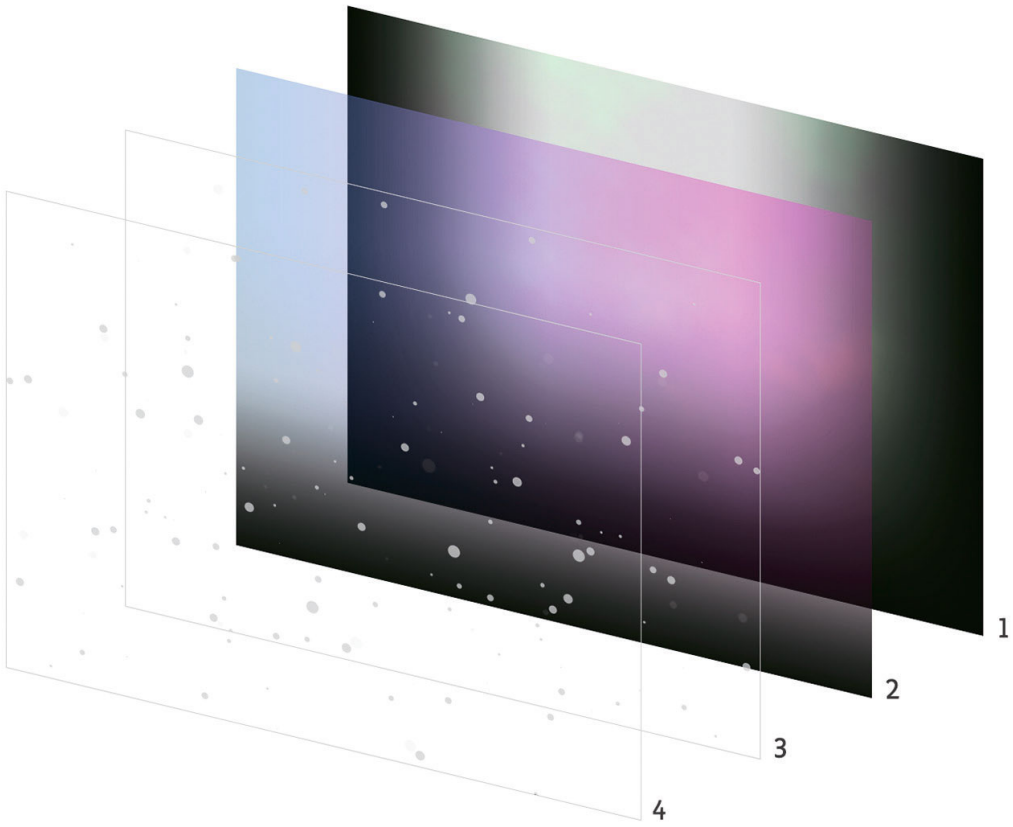

Рис. 5.03. Четыре полупрозрачных фоновых PNG-изображения, которые расположены на фоне сайта с Луной

Синтаксис множественного фона

Поставить эти четыре изображения в качестве фона элемента `body` очень просто с использованием нового синтаксиса CSS3:

```
body {
background:
url(../img/stars-1.png) repeat-x fixed -130% 0,
url(../img/stars-2.png) repeat-x fixed 40% 0,
url(../img/space-bg.png) repeat-x fixed -80% 0,
url(../img/clouds.png) repeat-x fixed 100% 0;
background-color: #1a1a1a;
```

```
}
```

Четыре изображения наслаиваются – облака в самый низ, звезды на самый верх – в виде списка, разделенного запятыми (обратите внимание, что перечисление начинается с того изображения, которое «ближе» к пользователю). Каждое изображение дублируется по горизонтали, и им выставлены различные положения по горизонтали (используя положительные и отрицательные значения), чтобы каждый слой двигался со своей скоростью, когда меняется размер окна браузера. Наконец, их положение на странице зафиксировано с помощью значения `fixed`.

Почти черный цвет `#1a1a1a` добавлен отдельным правилом `background-color` в самом конце.

Это все (**рис. 5.04**). Замечательно, что удалось обойтись без лишней разметки. Все эти изображения выставляются на элемент `body`, так что они будут находиться позади содержимого страницы, но нам не потребовалось обертывать их в лишние вспомогательные слои.

Рис. 5.04. Четыре PNG-изображения наслоены одно поверх другого, равно как и темно– серый цвет фона

Поддержка в браузерах

Как упоминалось в первой главе, множественные фоны поддерживаются в Safari 1.3+, Chrome 2+, Firefox 3.6+, Opera 10.5+ и IE9 Beta. Так что они находятся наравне с многими другими CSS-свойствами, которыми мы пользовались в этой книге.

Мы выбрали использовать эту замечательную возможность CSS3 в некритичной части дизайна из-за несовершенной поддержки: чтобы улучшить фон страницы, чтобы создать новое взаимодействие при изменении размеров окна – эффект параллакса для тех, кто может испытать его.

Запасной вариант для всех браузеров

Браузеры, которые пока что не поддерживают множественные фоны, проигнорируют свойство `background` целиком. Вот почему мы определили свойство `background-color` отдельно.

На **рис. 5.05** показано, как сайт выглядит в IE7: множественные фоны игнорируются, и отображается только темно-серый фон, заданный свойством `background-color`.

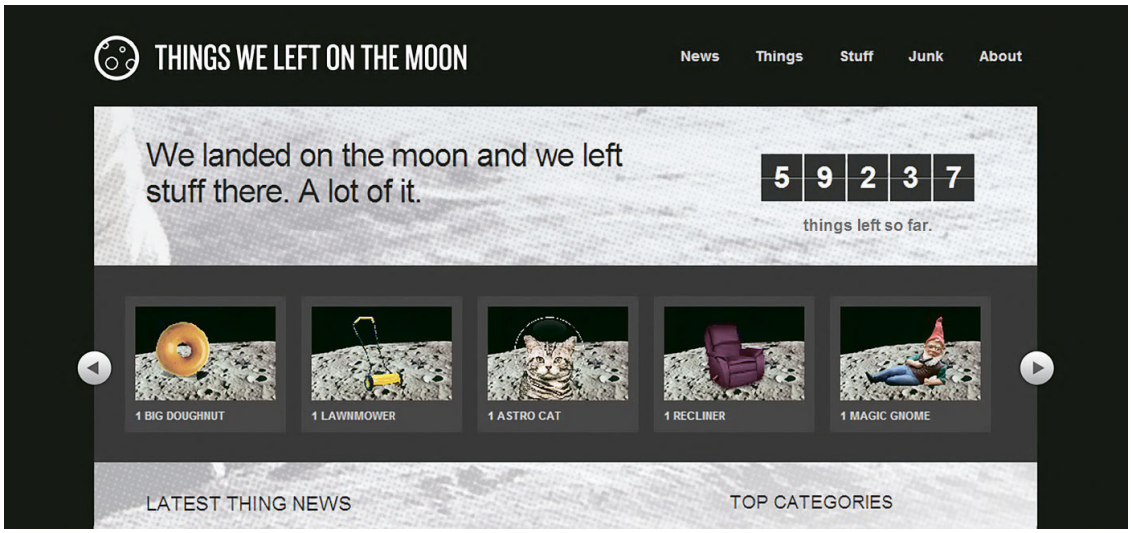

Рис. 5.05. IE7 игнорирует свойство, в котором определены множественные фоны, и показывает только темно-серый фон, заданный свойством background-color

Конечно, все работает должным образом, но то, что объемный фон потерялся, нехорошо. Решение заключается в том, чтобы сначала задать единый запасной фон – для браузеров (таких, как IE7 и 8), которые не поддерживают множественные фоны. Затем можно снова объявить это свойство – на этот раз с множественными фонами (IE проигнорирует его).

```
body {
background: url(../img/space-bg.png) repeat-x fixed -80% 0;
background:
url(../img/stars-1.png) repeat-x fixed -130% 0,
url(../img/stars-2.png) repeat-x fixed 40% 0,
url(../img/space-bg.png) repeat-x fixed -80% 0,
url(../img/clouds.png) repeat-x fixed 100% 0;
background-color: #1a1a1a;
}
```

Для запасного варианта с одной картинкой можно выбрать одно из изображений, которые используются во множественном объявлении, а можно пойти дальше и создать одну картинку, в которой объединялись бы все четыре изображения.

Для сайта с Луной я решил использовать `space-bg.png` – цветной градиент (**рис. 5.06**), таким образом показывая вариант фона без звезд и облаков в тех браузерах, которые пока что не поддерживают множественные фоны. Очень уместно.

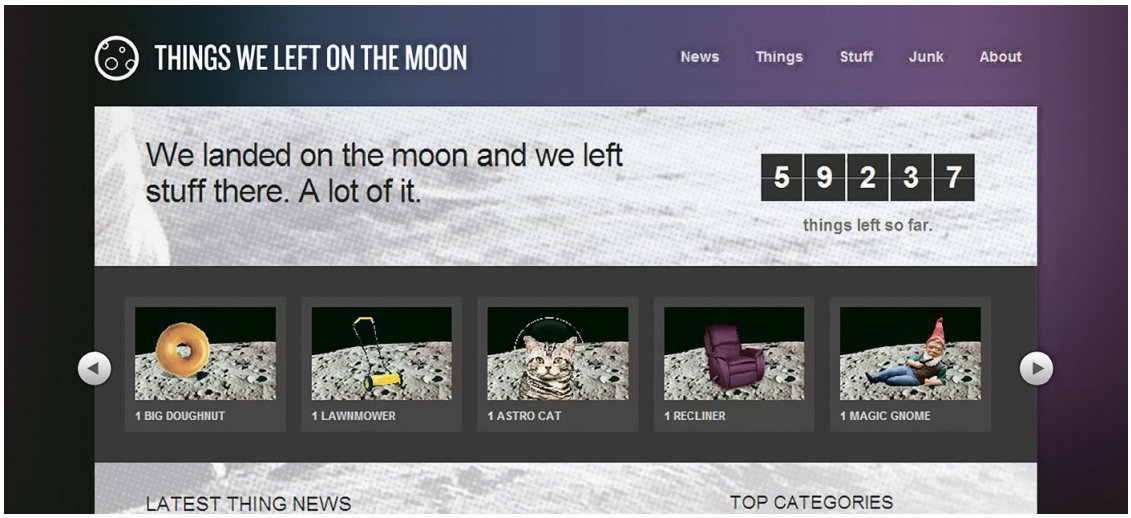

Рис. 5.06. Благодаря запасному варианту картинки на фоне в IE7 частично восстановлено ощущение пространства

Использование множественных фонов

Как и в других примерах этой книги, мы используем множественные фоны уже сегодня. Мы двигаемся вперед, пользуясь CSS3-свойством, которое уверенно поддерживается в Safari, Chrome, Firefox и Opera, равно как и в IE9 Beta. Вместо того чтобы пугаться этой неповсеместной поддержки и дожидаться ее, мы решаем применять это свойство для некритического визуального эффекта (параллакса на фоне).

Мы также знаем, что если браузер не поддерживает множественные фоны, он проигнорирует свойство `background` целиком. Для такой ситуации мы определяем одну картинку в отдельном свойстве, которое следует перед определением множественного фона.

Эти новые знания позволяют более гибко экспериментировать с наслоением, сдвигом и позиционированием фоновых изображений одно поверх другого – без необходимости писать лишнюю разметку. Будет здорово увидеть, какие способы применения будут придуманы для этой техники.

6. Улучшенные формы

Формы – еще одна часть веб-сайтов, которая содержит много интерактивности. У форм есть дополнительные визуальные состояния, и они созрели для того, чтобы улучшить их с помощью CSS3.

По умолчанию сами элементы форм могут очень сильно различаться внешне – в зависимости от операционной системы и браузера, в котором открыт сайт. Почему бы не использовать эту разницу, применяя рабочие фрагменты CSS3, чтобы улучшить пользовательский опыт?

Важно сохранять баланс: немного изменяя элементы форм, нужно сохранять их узнаваемость, чтобы позаботиться о юзабилити. Иными словами, поле ввода должно явно выглядеть как поле ввода. Теперь, когда элементы форм можно стилизовать с помощью CSS (в большинстве браузеров), нужно быть осторожными, чтобы не разрушить самую важную часть: функциональность.

При этом в отношении CSS3 с формами можно сделать очень многое, чтобы расширить пользовательский опыт в тех браузерах, которые поддерживают новые свойства; в остальных браузерах нужно показывать упрощенную версию.

Эта глава также дает возможность поговорить о трех аспектах CSS3, которых мы до сих пор не касались:

1) новые мощные селекторы;
2) CSS-градиенты;
3) CSS-анимации.

Мы вновь обратимся к сайту-примеру с Луной как к отправной точке, чтобы поговорить о том, как формы и CSS3 могут сочетаться новыми и необычными способами. Будем работать с формой регистрации «Оповещение о новых предметах», расположенной справа сбоку (**рис. 6.01**).

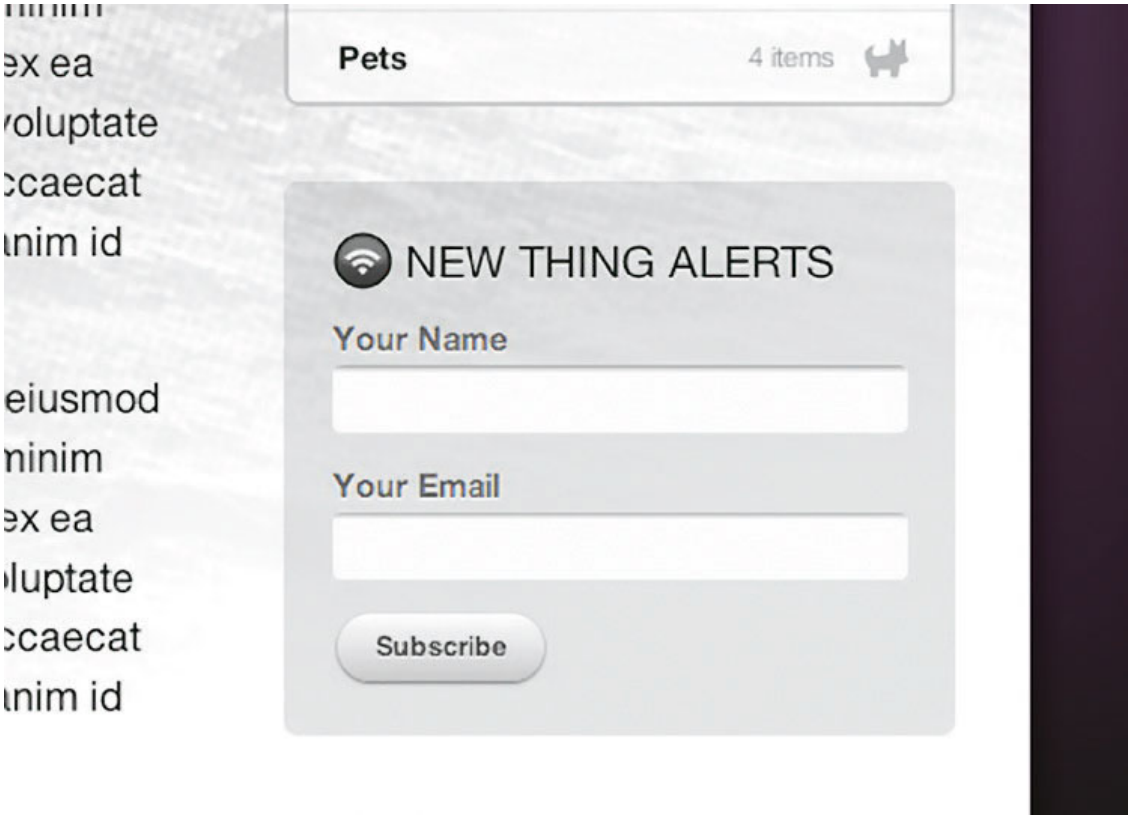

Рис. 6.01. Простая форма, где посетитель может подписаться на обновления о том, что на Луне оставили новые вещи

Разметка для простой формы регистрации

В терминах HTML эта небольшая форма очень проста: всего лишь несколько полей ввода, подписи к ним и кнопка «подписаться».

```
<form action="/" id="thing-alerts">
<fieldset>
<label for="alerts-name">Your Name</label>
<input type="text" id="alerts-name" />
</fieldset>
<fieldset>
<label for="alerts-email">Your Email</label>
<input type="text» id="alerts-email" />
</fieldset>
<fieldset>
<input type="submit" value="Subscribe" />
</fieldset>
</form>
```

На **рис. 6.02** форма показана так, как она выглядит со стилями по умолчанию, которые выставляет браузер (в этом примере – Safari).

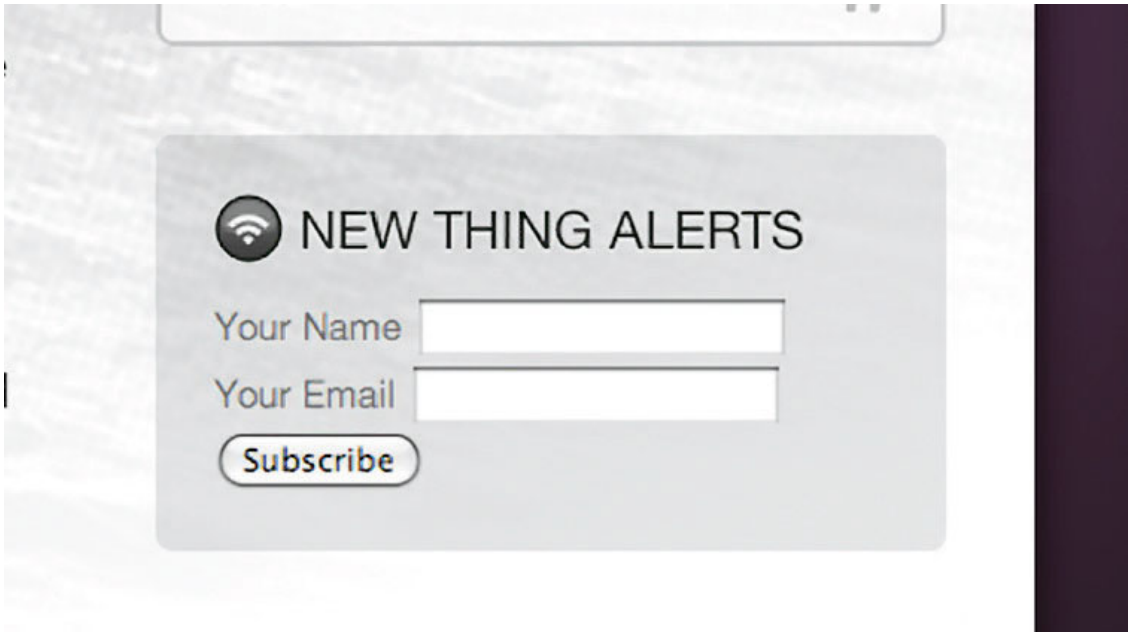

Рис. 6.02. Форма, открытая в Safari; без стилей

Стили для полей и подписей

Первые фрагменты CSS, которые мы добавим, чтобы начать построение этой формы, будут обрабатывать элементы `fieldset` и `label` – лишь немного пространства между строками.

```
#thing-alerts fieldset {
margin: 0 0 10px 0;
}
#thing-alerts label {
display: block;
font-weight: bold;
line-height: 1.4;
color: #666;
color: rgba(0, 0, 0, 0.6);
text-shadow: 0 1px 1px #fff;
}
```

Глядя на **рис. 6.03**, можно увидеть, что мы добавили отступ в `10px` под каждой строкой `fieldset` и задали подписям свойство `display: block`, чтобы они отображались на отдельной строке. Мы также выставили черному тексту непрозрачность в 60% и указали запасной серый цвет для тех браузеров, которые пока не поддерживают RGBA. Мы также добавили небольшую белую подсветку свойством `text-shadow`, чтобы текст выглядел так, будто бы он вставлен на фон.

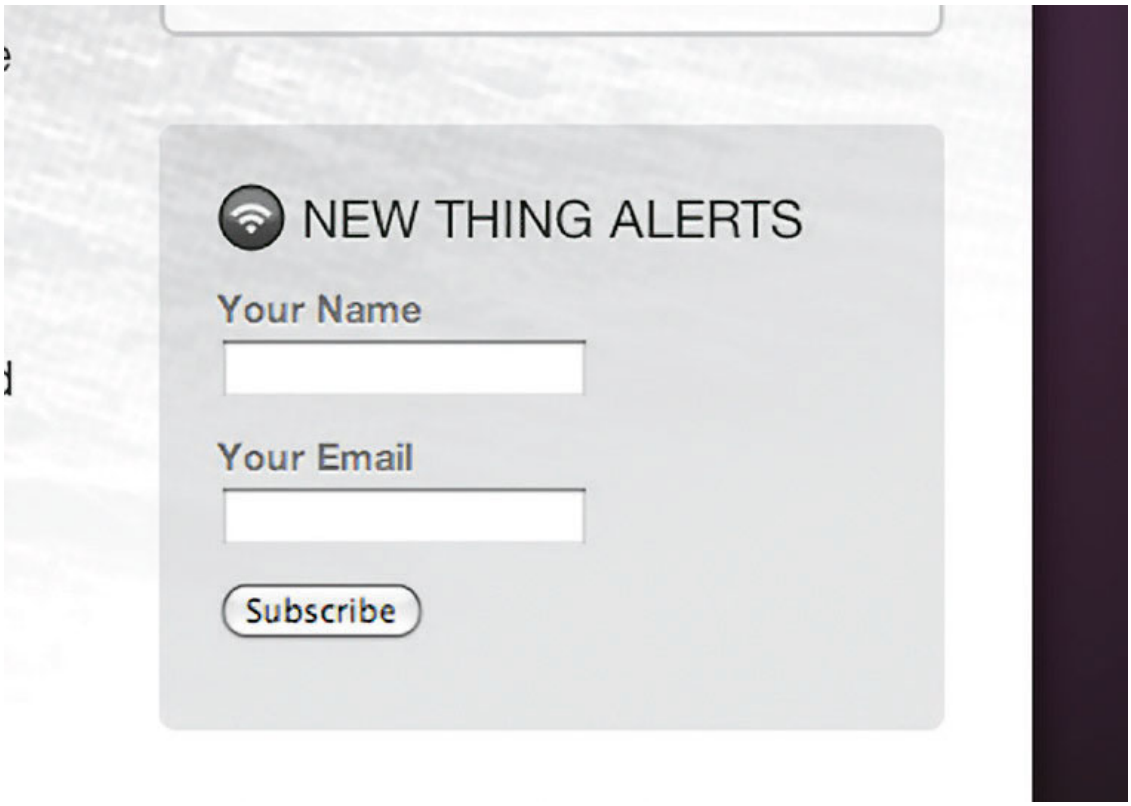

Рис. 6.03. К элементам fieldset и label применены стили

Теперь у нас есть хороший интервал в 10px между элементами `fieldset`, но из-за поля внутри серого блока нам не нужен отступ в 10px под последней строкой (содержащей кнопку «подписаться»).

Это часто встречающаяся ситуация: есть список элементов, к каждому из которых применяется один и тот же стиль, но нужно применить другой стиль к последнему элементу списка.

Вместо добавления `class="last"` к последнему элементу почему бы не воспользоваться CSS3-псевдоклассом: last-child, чтобы убрать отступ снизу, не трогая разметку:

```
#thing-alerts fieldset {
margin: 0 0 10px 0;
}
#thing-alerts fieldset label {
display: block;
font-weight: bold;
line-height: 1.4;
color: #666;
color: rgba(0, 0, 0, 0.6);
text-shadow: 0 1px 1px #fff;
}
#thing-alerts fieldset: last-child {
margin: 0;
}
```

Помните, что `:last-child` не поддерживается в IE8 и ниже, но для небольших визуальных изменений, подобных этому, такое решение намного лучше, чем дополнительный класс в разметке.

На **рис. 6.04** показано, что мы успели сделать: теперь нижний отступ на последнем элементе `fieldset` убран с помощью псевдокласса `:last-child`.

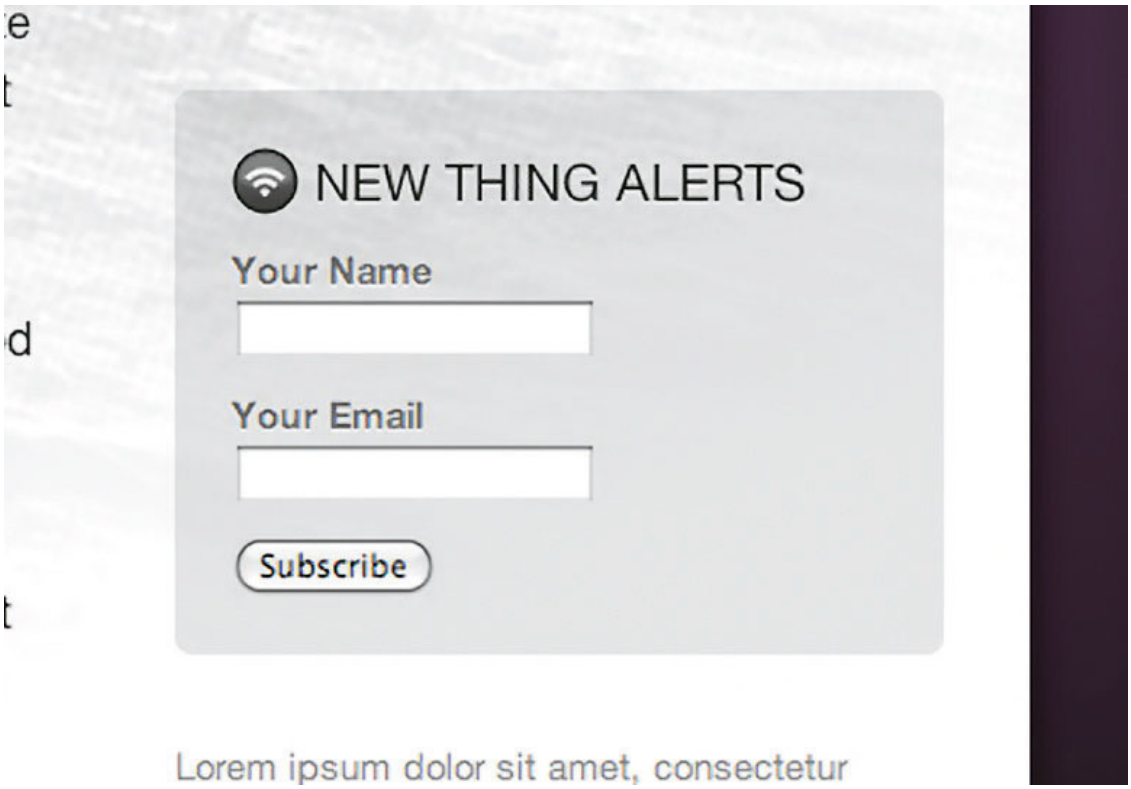

Рис. 6.04. Форма выглядит лучше, когда у последнего элемента fieldset нет отступа снизу

Больше CSS3-селекторов

Теперь, когда мы успешно воспользовались `:last-child`, пора отметить, что в CSS3 есть много других чрезвычайно удобных селекторов.

Я очень рекомендую прочитать статью Роджера Йоханссона на эту тему – CSS selectors explained (http://bkaprt.com/css3/11/)[14], – в которой он показывает, что они представляют собой и как работают. Поддержка этих селекторов различается между браузерами, так что обязательно сверьтесь с доскональными таблицами CSS contents and browser compatibility Питера-Пола Коха (http://bkaprt.com/css3/12/)[15] и CSS Compatibility and Internet Explorer от Microsoft (http:// bkaprt.com/css3/13/)[16], чтобы узнать, что где поддерживается.

[14] http://www.456bereastreet.com/archive/200601/css_3_selectors_explained/

[15] http://www.quirksmode.org/css/contents.html

[16] http://msdn.microsoft.com/en-us/library/cc351024(Vs.85). aspx

Оформление полей ввода

Начнем добавлять стили, которые превращают поля ввода по умолчанию во что-то особенное. На этот раз мы воспользуемся селектором по параметру из CSS2.1, чтобы обратиться исключительно к элементам `input type="text"` (не затрагивая кнопку `input type="submit"`).

Если бы в объявлении мы просто написали:

```
#thing-alerts input
```

то стиль бы применился ко всем объектам типа `input` (и к полям ввода, и к кнопкам). Но если мы исправим это на:

```
#thing-alerts input[type="text"]
```

то будут затронуты исключительно поля ввода.

Повторюсь: грамотное использование селекторов в стилевых файлах позволяет уменьшать количество дополнительных классов в разметке, которые вводятся исключительно для того, чтобы применять какие-то стили лишь к определенным элементам. Такое упрощение прекрасно.

Помните, что селекторы по параметру поддерживаются в IE7 и выше, но не поддерживаются в IE6. Впрочем, это нормально, поскольку мы изменяем некритический слой – вид элементов формы. IE6 проигнорирует эти правила, что вполне допустимо в этом случае.

Следующий код объявляет подходящие `width`, `padding` и `font-size`, отключает выставленный по умолчанию `border`, добавляет `background-color` и скругляет углы свойством `border-radius`.

```
#thing-alerts fieldset input[type="text"] {
width: 215px;
padding: 5px 8px;
font-size: 1.2em;
color: #666;
border: none;
background-color: #fff;
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
-o-border-radius: 4px;
border-radius: 4px;
}
```

На **рис. 6.05** показан результат – так, как он отображается в Safari (с аналогичным видом в Firefox и Opera). Теперь у нас есть плоские поля ввода со скругленными углами, которые выглядят хорошо; но давайте добавим немного объема, чтобы они больше походили на привычные редактируемые поля.

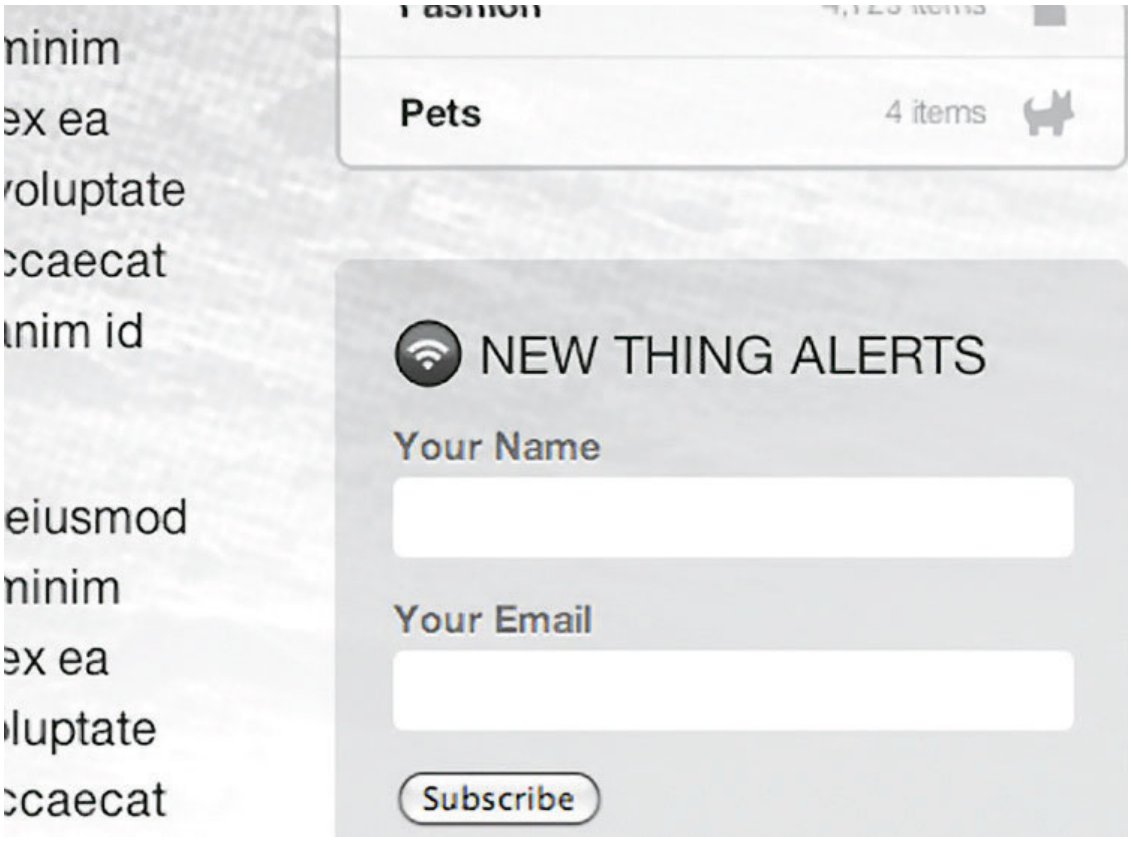

Рис. 6.05. Плоские поля ввода со скругленными углами

Градиенты в CSS3

Искусный способ, которым можно добавить объема, – это градиенты, которые появились в CSS3. То есть можно создавать градиентные переходы между цветами, не используя картинки. Звучит очень привлекательно, не так ли?

Градиенты в CSS сейчас поддерживаются только в Safari 4+, Chrome 2+ и Firefox 3.6+, но опять-таки для некритического использования это может быть гибким решением с хорошим запасным вариантом.

CSS-градиенты могут быть применены везде, где в стилях может быть использовано изображение. Другими словами – в виде значений для `background-image`, `list-style-image` и для сгенерированного содержимого.

Синтаксис CSS-градиентов немного отличается в Safari и Firefox. Спецификация (очень ранняя), впрочем, склоняется к варианту, который предлагает Firefox. Это важный пример того, почему браузерные префиксы – существенная часть процесса проработки спецификации: два различных синтаксиса могут быть должным образом объявлены для каждого браузера, пока официальная спецификация обсуждается.

Я честно признаюсь, что оба варианта могут показаться немного запутанными. Создание градиента включает в себя указание большого количества параметров: цвета, остановки цветов, направление градиента и т. п.

К примеру, вот синтаксис, который задает простой линейный градиент на фон элемента; работает в браузерах на движках WebKit и Mozilla:

```
    #foo {
    background-image: -webkit-gradient(linear, 0% 0%, 0% 100%,
from(#fff), to(#999));
    background-image: -moz-linear-gradient(0% 100% 90deg, #fff,
#999);
    }
```

Он далеко не интуитивно ясный; также сложно запомнить отличия между браузерами.

Лучший способ получить верный код, который я нашел, – использовать прекрасный визуальный редактор, написанный Джоном Оллсоппом (**рис. 6.06**, http://bkaprt.com/css3/14/).[17]

[17] http://www.westciv.com/tools/gradients/index-moz.html

Рис. 6.06. Инструмент создания CSS-градиентов, написанный Джоном Оллсоппом

Пользуйтесь этим редактором, чтобы графически создать нужные градиенты; затем получите нужный код для Safari и Firefox. Инструмент Джона берет на себя всю тяжелую работу. Очень помогает – учитывая, что я сам пока что не запомнил верный синтаксис (и отличия между браузерами).

Джонатан Снук написал хорошую статью о том, как разбираться в синтаксисе градиентов; она также может оказаться полезной: http://bkaprt.com/css3/15/[18]

Градиент, который мы хотим добавить к полям ввода, – очень незаметный переход цвета, который должен придать им объема (**рис. 6.07**). После экспериментов с инструментом Джона Оллсоппа получилось две строки кода:

```
    #thing-alerts fieldset input[type="text"] {
    width: 215px;
    padding: 5px 8px;
    font-size: 1.2em;
    color: #666;
    border: none;
    background-image: -webkit-gradient(linear, 0% 0%, 0% 12%, from(#999), to(#fff));
    background-image: -moz-linear-gradient(0% 12% 90deg, #fff, #999);
    background-color: #fff;
```

[18] http://snook.ca/archives/html_and_css/multiple-bg-css-gradients

```
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
-o-border-radius: 4px;
border-radius: 4px;
}
```

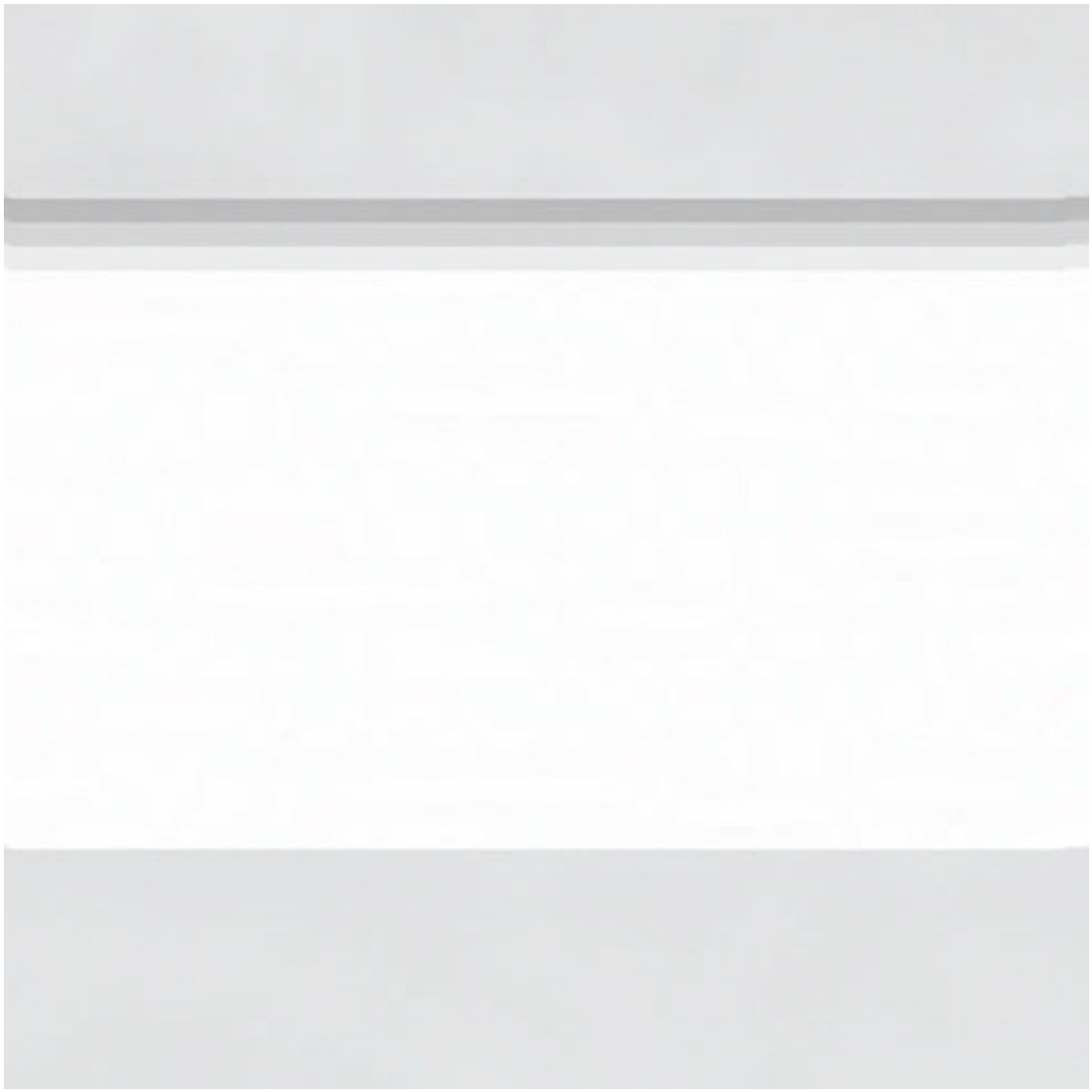

Рис. 6.07. Увеличенный вид небольшого градиента, расположенного над полем ввода. Поле выглядит так, как будто оно вставлено внутрь страницы

Мы применили линейный градиент, но в CSS также доступны круговые градиенты.

Пример выше показывает, как различается синтаксис между вариантами `-webkit` и `-moz`. Мы добавили небольшой линейный градиент, который переходит от светло-серого (`#999`) к белому (`#fff`) и занимает 12% от высоты текстового поля. Мы используем браузерные префиксы на свойстве `background-image`, чтобы градиент отображался в Safari и Firefox.

На **рис. 6.08** показан результат: можно видеть поля ввода со скругленными углами и небольшой внутренней тенью. Изображения не используются.

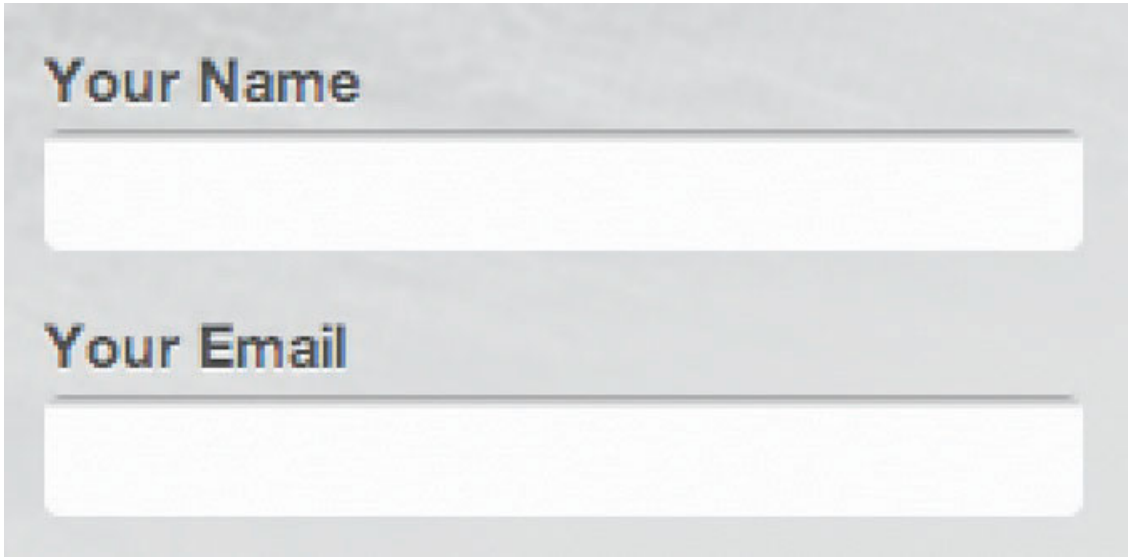

Рис. 6.08. Поля ввода под воздействием CSS-градиентов

Браузеры, которые пока что не поддерживают CSS-градиенты, проигнорируют эти свойства `background-image`; поля ввода будут белыми и плоскими. Это нормально. Но гибкость и управляемость, которые приходят вместе с CSS-градиентами, очень привлекательны.

Мы будем использовать градиенты еще немного в следующем разделе – для кнопки «подписаться».

Настоящая кнопка на CSS3

На примере кнопки очень легко показать, как сильно можно преображать вид элементов с помощью CSS3. Сочетая различные приемы, которые мы обсуждали на протяжении этой книги, превратим обычную кнопку в что-то намного более интересное – используя исключительно CSS (**рис. 6.09**).

Вся прелесть применения CSS3 для оформления кнопки заключается в том, что без использования изображений получается куда более гибкое решение. Если браузер не поддерживает те свойства, которыми мы воспользуемся для оформления кнопки, это нормально. Кнопка будет выглядеть так, как она выглядит по умолчанию в том браузере, который использует посетитель сайта.

Давайте пройдем через все шаги, которые нужно проделать, чтобы обычная кнопка превратилась в прекрасную сияющую кнопку, которая показана справа на **рис. 6.09**.

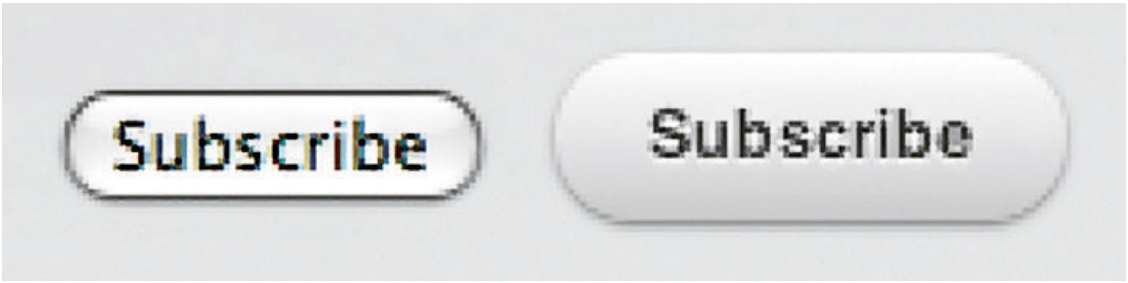

Рис. 6.09. Слева – вид кнопки по умолчанию (в Safari); справа – кнопка, оформленная средствами CSS3. Никаких изображений!

Основные стили для кнопки

Сначала добавим поля, сменим шрифт на Helvetica, чтобы кнопка сочеталась с остальными элементами дизайна, уберем обводку и выставим белый цвет фона.

```
#thing-alerts input[type="submit"] {
padding: 8px 15px;
font-family: Helvetica, Arial, sans-serif;
font-weight: bold;
line-height: 1;
color: #444;
border: none;
background-color: #fff;
}
```

То, что получилось после применения этих простых стилей, показано на **рис. 6.10**. Уже есть что-то сильно отличающееся от кнопки по умолчанию.

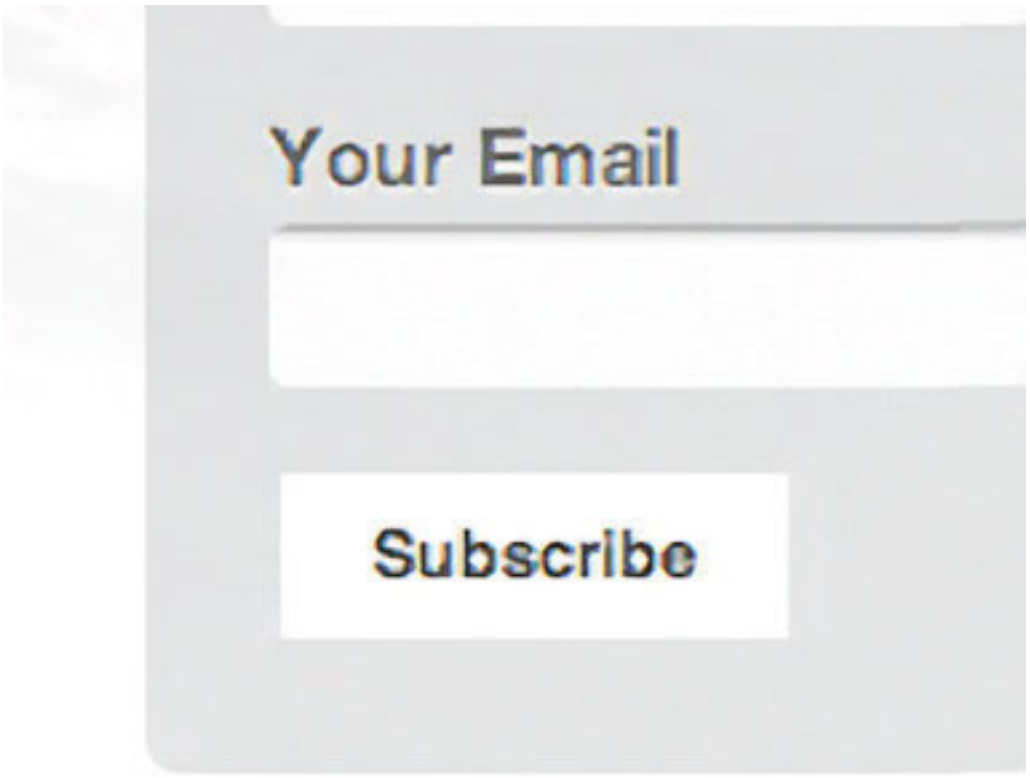

Рис. 6.10. Кнопка, с которой убраны обводка и фон по умолчанию

Скругление углов

Теперь добавим свойство border-radius, чтобы скруглить углы кнопки (**рис. 6.11**).

```
#thing-alerts fieldset input[type="submit"] {
padding: 8px 15px;
font-family: Helvetica, Arial, sans-serif;
font-weight: bold;
line-height: 1;
color: #444;
border: none;
background-color: #fff;
-webkit-border-radius: 23px;
-moz-border-radius: 23px;
-o-border-radius: 23px;
border-radius: 23px;
}
```

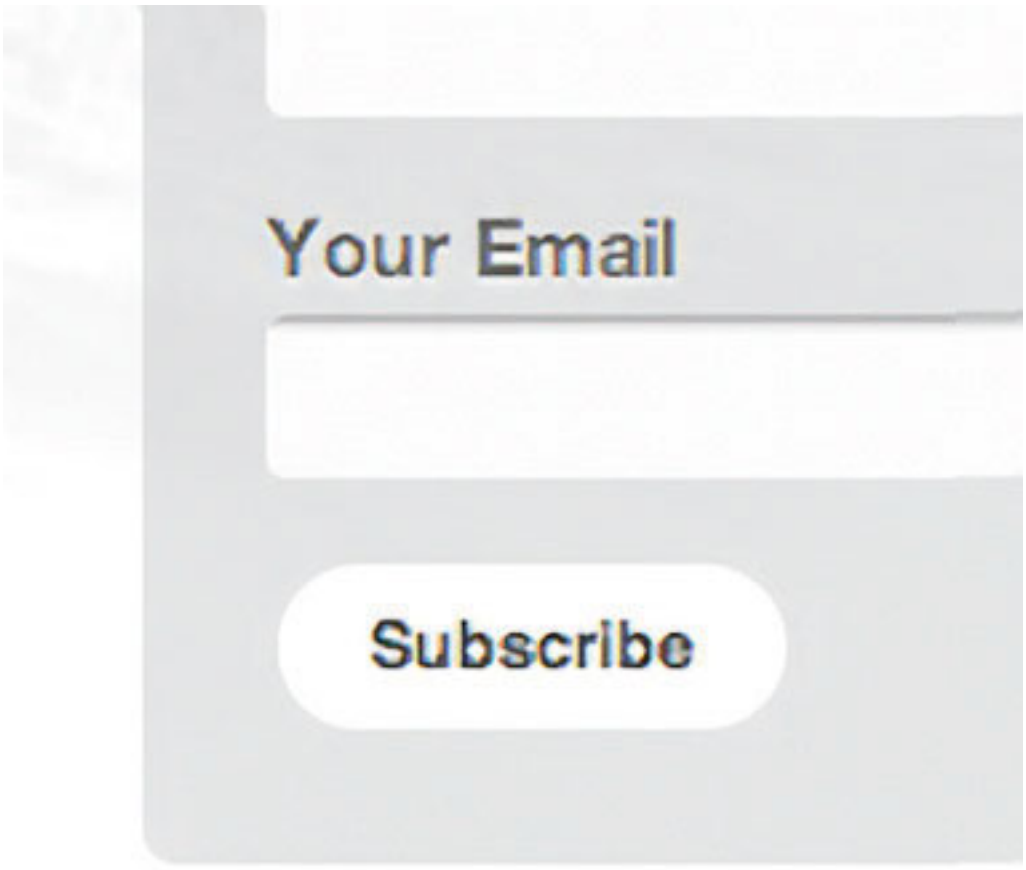

Рис. 6.11. Скругление углов кнопки свойством border-radius

Поэкспериментировав с разными значениями, мы остановились на радиусе в `23px`.

Линейный градиент

Теперь применим линейный градиент с переходом цвета от светло-серого (`#bbb`) к белому (`#fff`), снизу вверх. Мы вновь воспользуемся инструментом Оллсоппа, чтобы получить верный код для Safari, Chrome и Firefox.

```
    #thing-alerts input[type="submit"] {
    padding: 8px 15px;
    font-family: Helvetica, Arial, sans-serif;
    font-weight: bold;
    line-height: 1;
    color: #444;
    border: none;
    background-image: -webkit-gradient(linear,0% 0%, 0% 100%,
from(#fff), to(#bbb));
    background-image: -moz-linear-gradient(0 100% 90deg, #bbb,
#fff);
    background-color: #fff;
    -webkit-border-radius: 23px;
    -moz-border-radius: 23px;
    -o-border-radius: 23px;
    border-radius: 23px;
```

```
}
```

На **рис. 6.12** показан полученный результат – так, как его отображает Safari. Теперь у нас есть кнопка со скругленными углами и градиентом. До сих пор не было использовано ни одного изображения и мы добавили лишь несколько строк кода в стилевой файл.

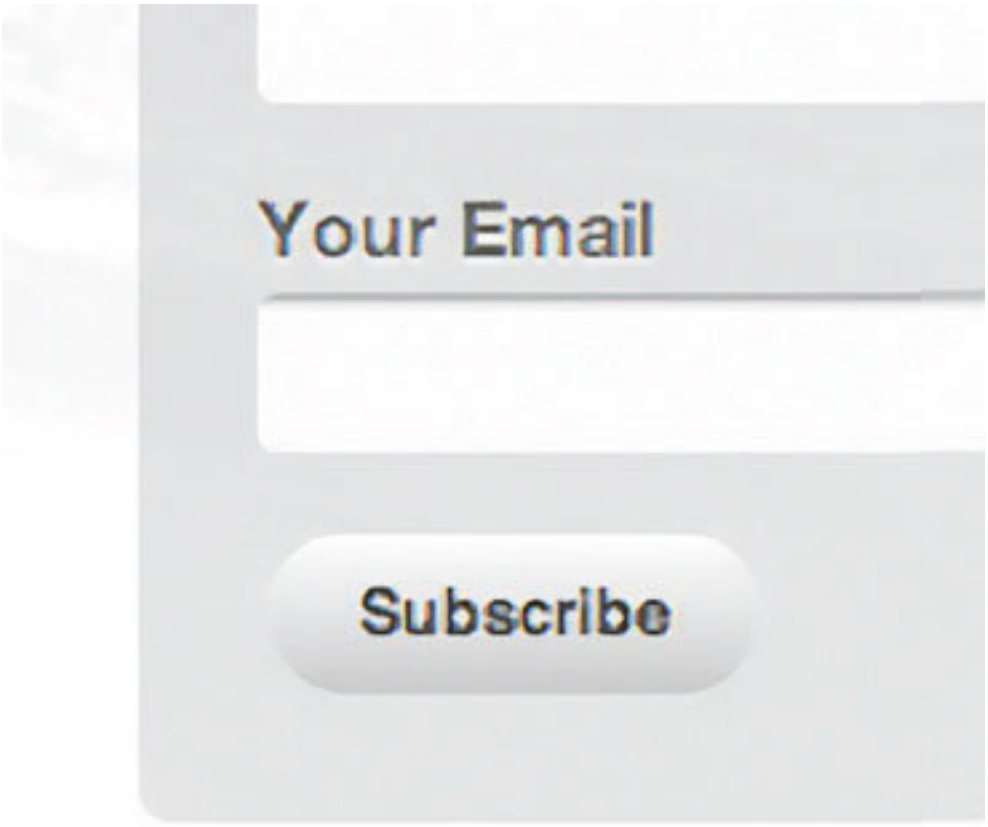

Рис. 6.12. К кнопке применен CSS-градиент

text-shadow

Теперь добавим почти белую тень под текстом, которая создаст эффект того, что текст вдавлен в кнопку.

```
#thing-alerts input[type="submit"] {
padding: 8px 15px;
font-family: Helvetica, Arial, sans-serif;
font-weight: bold;
line-height: 1;
color: #444;
border: none;
text-shadow: 0 1px 1px rgba(255, 255, 255, 0.85);
background-image: -webkit-gradient(linear,0% 0%, 0% 100%,
from(#fff), to(#bbb));
background-image: -moz-linear-gradient(0 100%90deg, #fff,
#bbb);
background-color: #fff;
-webkit-border-radius: 23px;
-moz-border-radius: 23px;
```

```
-o-border-radius: 23px;
border-radius: 23px;
}
```

Мы воспользуемся RGBA, чтобы высветлить белый цвет до 85%, позволяя серому градиенту немного проходить сквозь него. Мы также задаем положение тени – на один пиксель ниже текста – и размывание тени на один пиксель.

На **рис. 6.13** крупным планом показана тень и то, как кнопка выглядит сейчас.

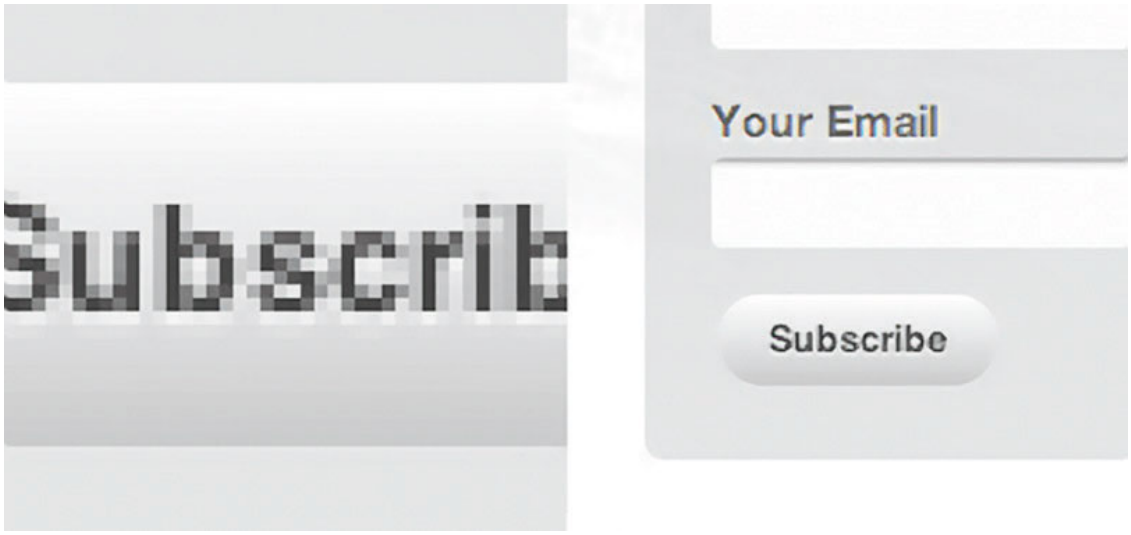

Рис. 6.13. Увеличенное изображение небольшой text-shadow, добавленной, чтобы создать эффект тиснения

Тень на кнопке

Последний фрагмент CSS3, который мы добавим к этой прекрасной кнопке, – небольшая тень (`box-shadow`), которая даст еще немного объема. С такой тенью кнопка будет выглядеть лучше на сером фоне.

Вот код, который добавляет свойство `box-shadow`, работающее в браузерах, в которых оно сейчас поддерживается, как и в будущих браузерах:

```
#thing-alerts input[type="submit"] {
padding: 8px 15px;
font-family: Helvetica, Arial, sans-serif;
font-weight: bold;
line-height: 1;
color: #444;
border: none;
text-shadow: 0 1px 1px rgba(255, 255, 255, 0.85);
background-image: -webkit-gradient(linear, 0% 0%, 0% 100%,
from(#fff), to(#bbb));
background-image: -moz-linear-gradient(0% 100% 90deg, #bbb,
#fff);
background-color: #fff;
-webkit-border-radius: 23px;
-moz-border-radius: 23px;
-o-border-radius: 23px;
border-radius: 23px;
-webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.5);
```

```
-moz-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.5);
box-shadow: 0 1px 2px rgba(0, 0, 0, 0.5);
}
```

На **рис. 6.14** показан результат, отображаемый в Safari, после добавления кнопке `box-shadow`, которая расположена на `1px` относительно верха, и с размытием в `2px`. Мы используем полупрозрачный черный цвет с помощью RGBA, так что фон просвечивает сквозь полупрозрачную тень.

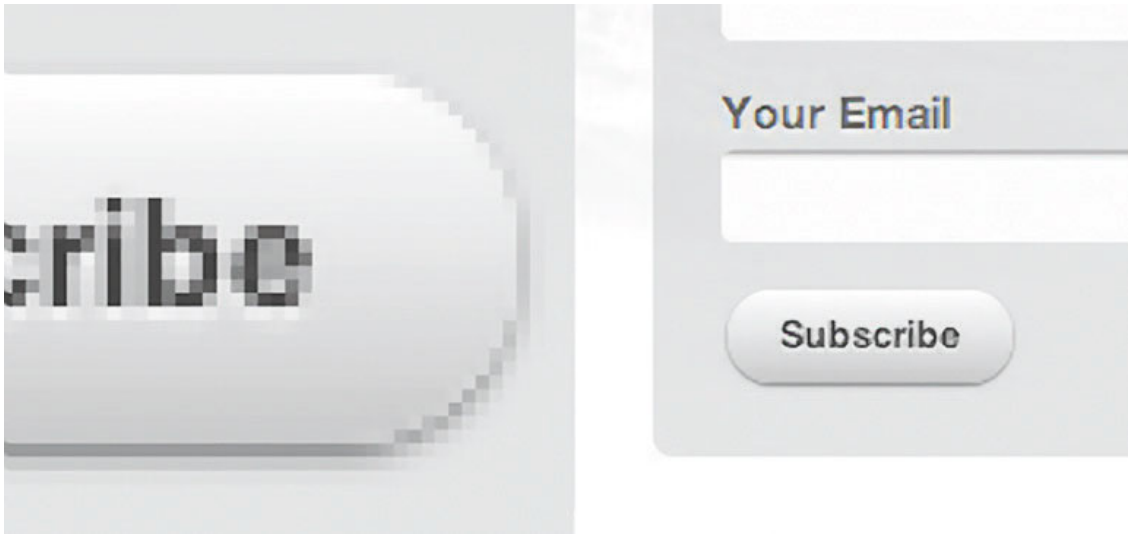

Рис. 6.14. Увеличенное изображение небольшой тени, добавленной к кнопке: благодаря ей кнопка чуть-чуть приподнимается над фоном

На этом мы заканчиваем работу не только с нашей кнопкой, но и со всей формой целиком. Некоторым количеством кода на CSS3 мы преобразили кнопку в том виде, в котором она отображается по умолчанию, в хорошо оформленный элемент, который должным образом сочетается с дизайном всей страницы. Мы предпочли использовать CSS3 вместо изображений, что нормально и безвредно для тех браузеров, которые не поддерживают эти новые свойства. Давайте убедимся в этом самостоятельно.

А как насчет других браузеров?

Открывая форму в Internet Explorer 7 – браузере с нулевой поддержкой CSS3, – мы видим вполне приемлемую рабочую форму (**рис. 6.15**). Это замечательно! Все улучшения, добавленные свойствами CSS3, были проигнорированы; остался скелет формы, работающий так, как нужно. Цель достигнута.

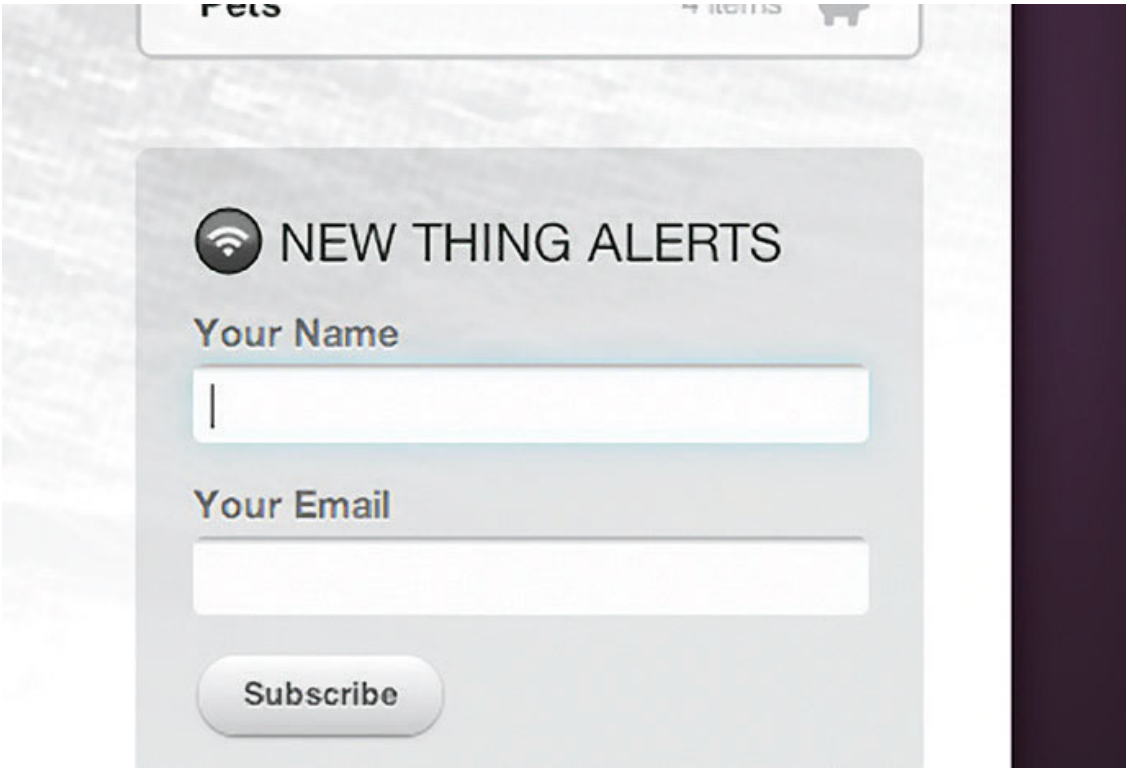

Рис. 6.15. В IE7 форма выглядит и работает как обычная. Это хорошо

Использование box-shadow для создания состояния focus

Мы можем пойти дальше в улучшении взаимодействия с этой формой, используя свойство `box-shadow` на тех элементах, которые находятся в состоянии `:focus`. Это быстро, легко

и, как и прежний CSS3-код, не затрагивает старые браузеры.

Требуется лишь создать новое объявление псевдокласса `:focus` относительно селектора по параметру для текстовых полей ввода.

(Кстати, предыдущий абзац – беспроигрышная фраза для знакомств, если она вам вдруг нужна. Благодарности – позже.)

```
#thing-alerts input[type=»text»]: focus {
-webkit-box-shadow: 0 0 12px rgba(51, 204, 255, 0.5);
-moz-box-shadow: 0 0 12px rgba(51, 204, 255, 0.5);
box-shadow: 0 0 12px rgba(51, 204, 255, 0.5);
}
```

Эти строки добавляют свойство `box-shadow`, которое задает яркую полупрозрачную синюю тень вокруг полей ввода, когда они находятся в состоянии: focus. Результат показан на **рис. 6.16**: мы имитируем поведение операционной системы по умолчанию, но оформление поддается точной настройке благодаря нашему собственному обработчику.

Что насчет браузеров, которые не поддерживают `box-shadow`? Что ж, они будут показывать обыкновенное поле ввода в состоянии: focus. Думаю, можете догадаться, что я сейчас скажу: да, это нормально.

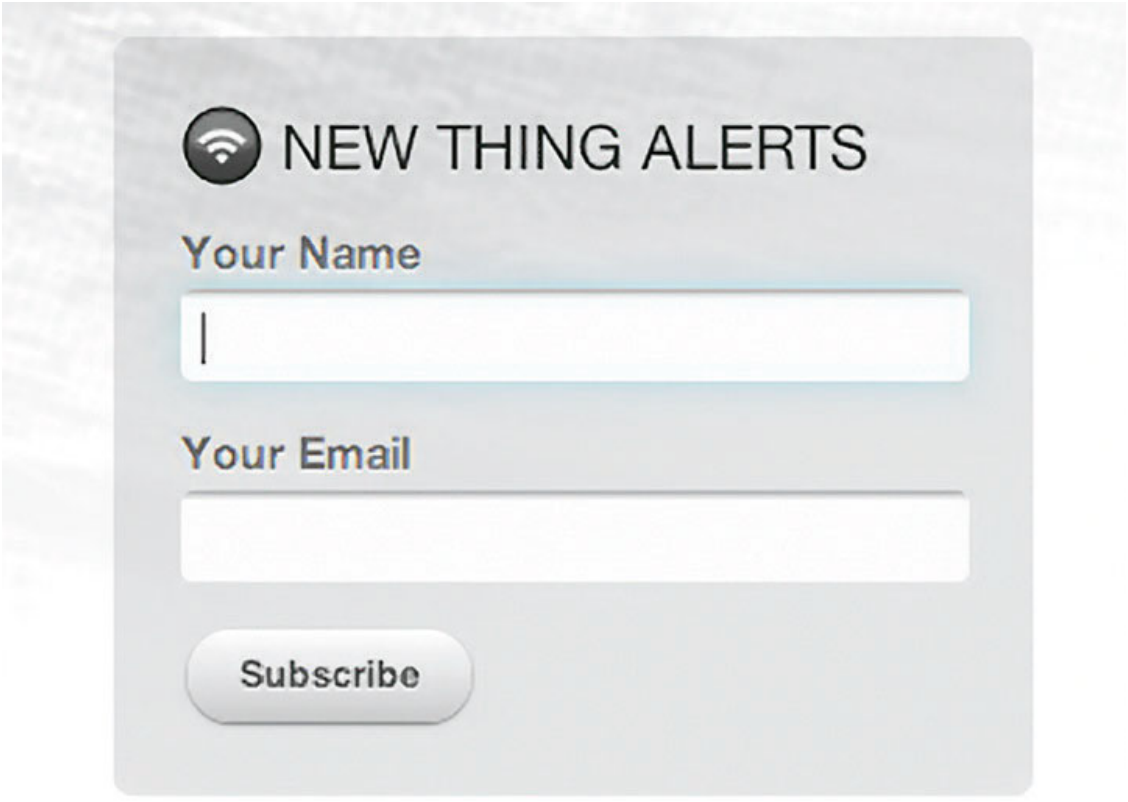

Рис. 6.16. Свойство box-shadow применяется к состоянию: focus текстовых полей

Добавление CSS-анимаций для улучшения взаимодействия с формой

Можно пойти еще дальше со свойством `box-shadow`: что если тень также будет анимированной – например, пульсирующей, показывая, что ожидается ввод. Давайте кратко погрузимся в мир CSS-анимаций, чтобы создать такой эффект в браузерах, основанных на движке WebKit.

Я говорю о браузерах на WebKit в этом случае, потому что CSS-анимации (равно как и CSS-преобразования и переходы) были изначально разработаны командой WebKit и затем включены в предложенный стандарт W3C (http://bkaprt.com/css3/16/)[19]. Однако в отличие от преобразований и переходов анимации пока что не поддерживаются никакими другими браузерами. Сейчас они работают в Safari и Chrome, но не в Firefox или Opera; поддержка в IE9 также не планируется. По этой причине я не уделяю слишком много внимания анимациям (по крайней мере сейчас). Хоть они действительно мощные и захватывающие, еще предстоит увидеть, окажется ли их внедрение таким же исчерпывающим и быстрым, как это произошло с преобразованиями и переходами, у которых уже есть достойная (и растущая) поддержка.

Тем не менее концепция и синтаксис CSS-анимаций – довольно понятные и для некритических улучшений, которые будут видны только в браузерах на WebKit, прекрасно вставлять их в подходящие места. Добавим простую анимацию для состояния: focus полей ввода, чтобы узнать, как все это работает.

Ключевые кадры

Первая часть построения CSS-анимации заключается
в объявлении ключевых кадров. Читатель, знакомый
с программированием, может воспринимать это как создание функции, к которой затем можно обращаться из любого места стилевого файла.

`keyframe` – особенное @-правило CSS. Оно похоже на обычное CSS-объявление, но позволяет назначить ему собственный идентификатор и задать CSS-свойства и изменения их значений вместе со списком значений в процентах (или же использовать ключевые слова «to» и «from»).

Будет разумнее увидеть анимации в действии, так что давайте создадим простую анимацию, которая будет плавно показывать и убирать тень, которую мы раньше создали для полей ввода в состоянии: focus.

Мы назовем его «pulse» и зададим три немного различающихся правила: вначале (0%), посередине (50%) и в конце (100%). Каждое правило задает уровень прозрачности синей тени (`box-shadow`), от 20 до 90% и обратно на 20%. Это изменение, распределенное по времени и зацикленное, создаст эффект того, что поле ввода пульсирует, когда фокус находится на нем (только в браузерах, работающих на движке WebKit).

```
@-webkit-keyframes pulse {
0% {
-webkit-box-shadow: 0 0 12px rgba(51, 204, 255, 0.2);
}
50% {
```

[19] http://www.w3.org/tR/Css3-animations/

```
-webkit-box-shadow: 0 0 12px rgba(51, 204, 255, 0.9);
}
100% {
-webkit-box-shadow: 0 0 12px rgba(51, 204, 255, 0.2);
}
}
```

Здесь задаются свойства только для WebKit – с помощью браузерного префикса. На протяжении всей книги мы аккуратно копировали свойства для всех браузеров и писали беспрефиксную версию. Но в этом случае, когда CSS-анимации поддерживаются лишь в Safari и Chrome и другие производители браузеров еще не определились, стоит ли вообще относить анимации к CSS, я предпочитаю писать правила только для `-webkit-`.

Ссылки на keyframe

Вторая часть CSS-анимации заключается в том, чтобы сослаться на `keyframe` по его имени, пользуясь свойством `animation`.

В этом случае мы хотим, чтобы пульсация `box-shadow` начиналась тогда, когда пользователь переводит фокус на текстовое поле в форме. В этот момент мы можем обратиться к `keyframe` по его имени, задать длительность анимации, зациклить ее и определить временные функции перехода. Можно видеть, что синтаксис анимаций похож на синтаксис переходов.

```
#thing-alerts input[type=»text»]: focus {
-webkit-animation: pulse 1.5s infinite ease-in-out;
}
```

Таким образом мы обеспечиваем, чтобы анимация пульсирования запускалась только тогда, когда пользователь наводит фокус на текстовое поле формы.

Результат довольно впечатляющий. Если бы технология позволяла мне показать его на листе бумаги, я бы сделал это. Вместо этого **рис. 6.17** должен передать ощущение того, что происходит: медленное анимированное затухание и появление `box-shadow`, как будто бы поле ввода ждет, когда с ним начнут взаимодействовать.

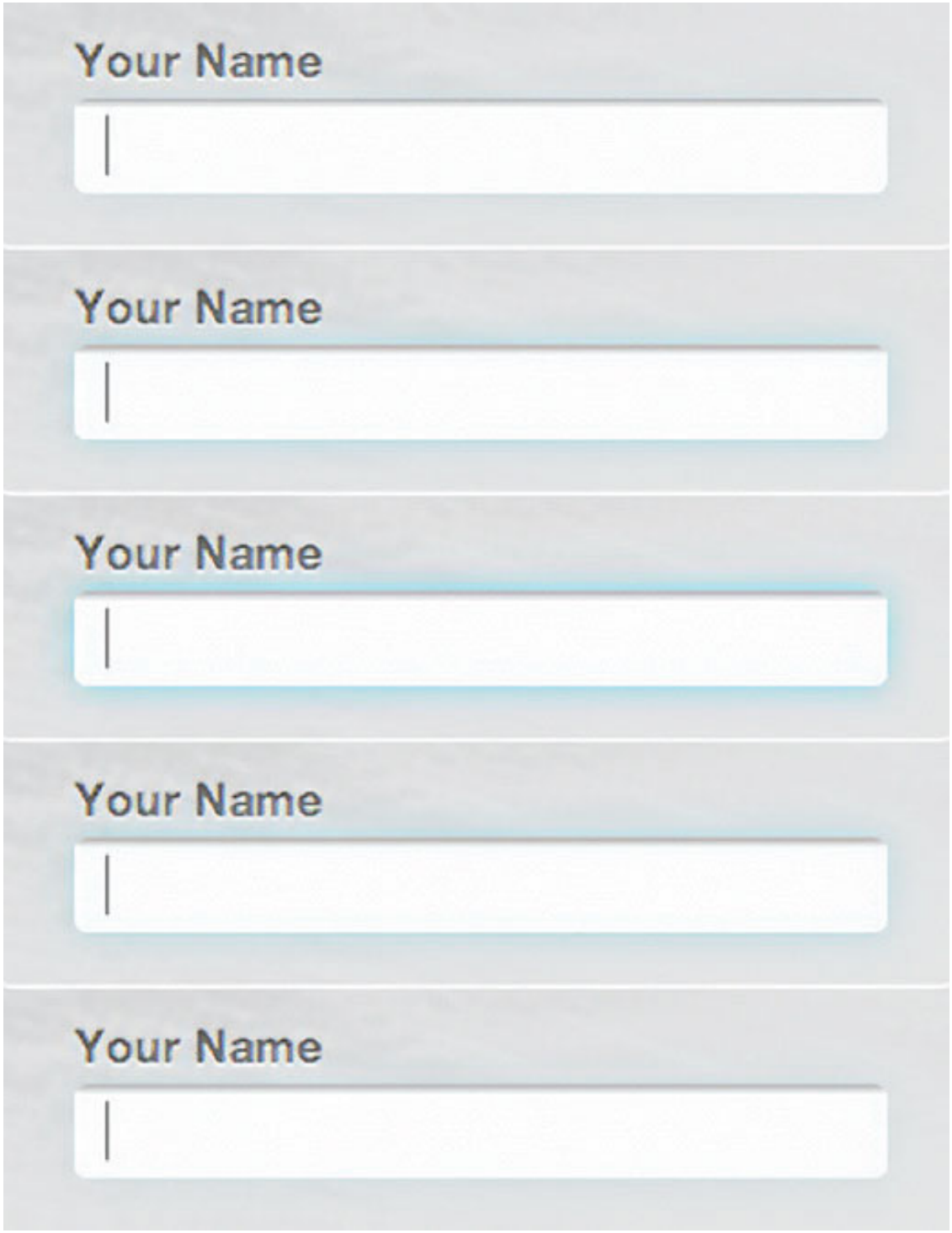

Рис. 6.17. Если быстро перемещать глаза вверх и вниз по этой картинке, можно получить ощущение той анимации, которую мы добавили к полям ввода в состоянии: focus

Использовалось краткое свойство `animation`, чтобы задать значения для обращения к анимации в одном месте. Вместо этого можно задавать каждое значение в отдельном свойстве:

```
#thing-alerts input[type="text"]: focus {
-webkit-animation-name: pulse;
-webkit-animation-duration: 1.5s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-timing-function: ease-in-out;
```

```
}
```

Повторное использование анимации для кнопки в состоянии hover

Одна из приятных особенностей ключевых кадров состоит в том, что их можно повторно использовать внутри нескольких блоков кода в стилевом листе. Например, применить ту же анимацию «pulse» к кнопке в состояниях hover и focus, добавляя Wii-подобное пульсирующее синее свечение.

Это очень просто: к кнопке в состояниях hover и focus добавляется то же самое свойство `animation` – точно так же, как мы проделали с текстовыми полями:

```
#thing-alerts input[type="submit"]: hover,
#thing-alerts input[type="submit"]: focus {
-webkit-animation: pulse 1.5s infinite ease-in-out;
}
```

Благодаря анимации `pulse`, которую мы ранее создали для текстовых полей, синяя тень (`box-shadow`) появляется и затухает. Мы можем заново использовать эту анимацию для кнопки, на которой этот эффект также работает хорошо (**рис. 6.18**), мягко мерцая, когда на нее наводят курсор или переводят фокус, – будто бы ожидая, пока пользователь нажмет на нее.

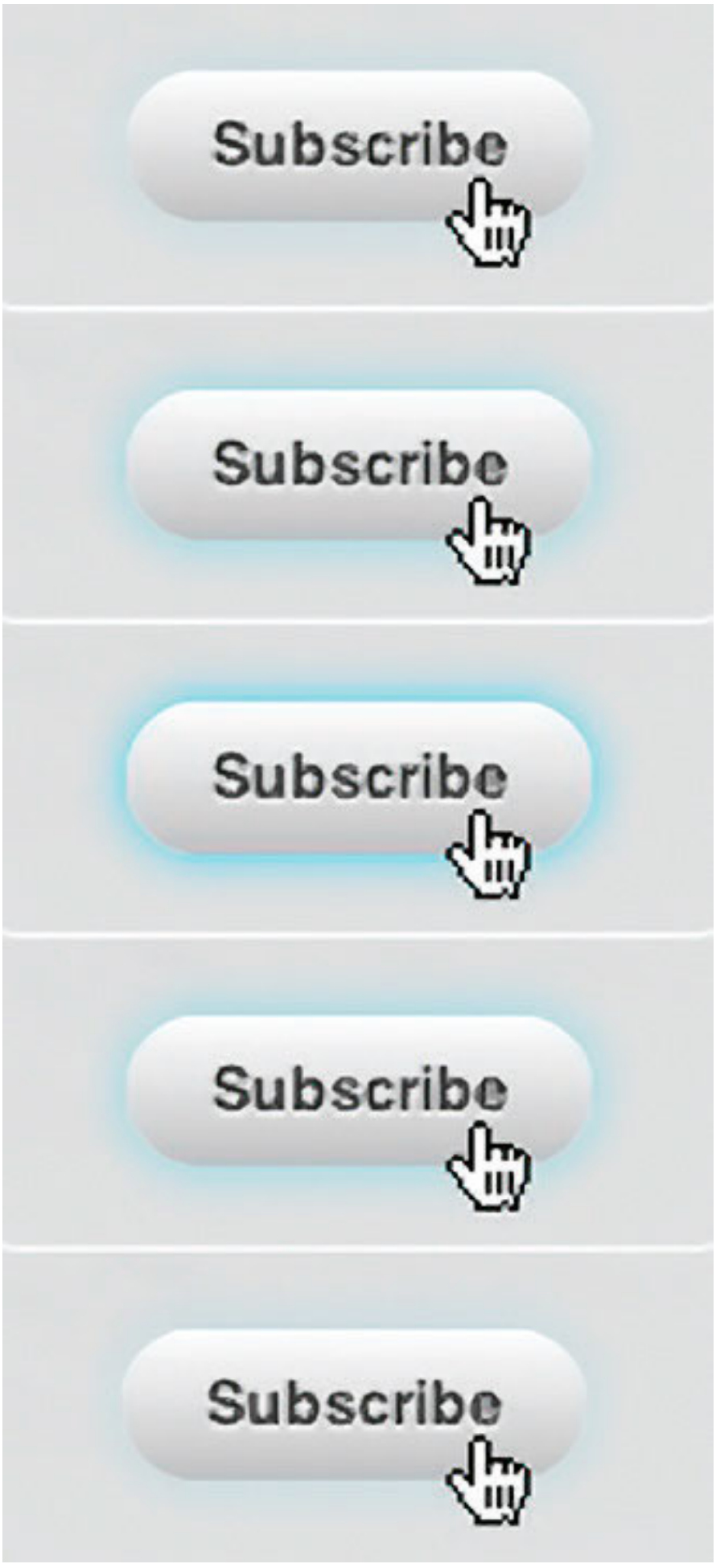
Subscribe
Subscribe
Subscribe
Subscribe
Subscribe

Рис. 6.18. Попытка проиллюстрировать пульсацию тени вокруг кнопки, когда на нее наведен курсор

А как насчет остальных браузеров?

Добавление CSS-анимации – это первый раз в этой книге – когда мы улучшали пользовательский опыт только для одного производителя браузеров: WebKit. Одна из основных причин, по которой CSS3 используется все больше и больше, – новые свойства принимаются браузерами Firefox, Opera и IE9. Вы можете задаваться вопросом, стоит ли добавлять CSS-анимации на сегодняшний день. Это определенно то, что стоит рассматривать.

CSS-анимации либо останутся технологией исключительно для браузеров на движке WebKit, либо пойдут по стопам переходов и преобразований и будут приняты всеми остальными. В любом случае CSS-анимации – простые, не требуют большого труда и спокойно игнорируются браузерами, которые их не понимают. То, что я показал в этом примере, довольно примитивный способ использования анимаций, призванный проиллюстрировать, что возможно создать, пользуясь исключительно разметкой и стилевыми таблицами. Это замечательно, и по этой причине с анимациями стоит экспериментировать.

Сосредоточьтесь на взаимодействии

Элементы форм редко бывают критическими элементами бренда, и поэтому формы – прекрасная область применения CSS3-свойств.

Элементы форм сильно отличаются внешне в зависимости от пользовательского окружения, но мы можем уменьшить эту разницу, оформляя их с помощью расширенного CSS и зная, что в браузерах, которые не поддерживают CSS3, будут отображаться прекрасно работающие, знакомые элементы форм по умолчанию.

Заключение

Теперь спустимся на Землю и посмотрим назад.

Мы обсудили много прекрасных (если позволите мне так говорить) способов использования CSS3 прямо сейчас, в ежедневной работе. Я надеюсь, что, показав, как эти приемы могут улучшать взаимодействие в браузерах, которые поддерживают новые свойства, и как все остается рабочим в остальных браузерах, вы будете вдохновлены на то, чтобы использовать их ежедневно, вне зависимости от проекта.

Подлинный потенциал CSS3 – в том, что оно позволяет нам разрешать распространенные задачи дизайна более эффективно, используя меньше кода и получая решение с большей степенью гибкости. До тех пор пока вы (и ваши заказчики и руководители) признаете, что сайты могут выглядеть и проявлять себя по-разному в разных браузерах и на разных устройствах, возможностям не будет предела.

В первой главе я упоминал, что часто слышу: «Не могу дождаться, хочу начать использовать CSS3… когда его закончат». Цель этой книги – показать, что ждать незачем. Начинайте использовать CSS3 для некритических визуальных

событий в своем дизайне. Теперь, когда вы вооружены тем, что работает и – еще важнее, – как вещи выглядят тогда, когда новые свойства не работают, – вы можете легко добиваться того, на что раньше уходило больше времени и кода, всего лишь несколькими строчками CSS.

А как насчет заказчиков и руководителей, которые не понимают этого?

Еще один вопрос, который мне часто задают о CSS3, – как я использую новые свойства, работая с заказчиками. Как объяснять им преимущества использования CSS3 по сравнению с другими решениями? Обучать своих заказчиков – самое

полезное: покажите им, насколько меньше кода и изображений получается в результате. Продемонстрируйте, как отличается пользовательский опыт в браузерах, которые пока что не поддерживают CSS3. Объясните им, в чем заключается компромисс.

Если это трудоемко, тогда просто используйте CSS3.

Начинайте добавлять фрагменты CSS3 в свою ежедневную работу, и пусть ваши заказчики и руководители с радостью узнают об этом. Правда заключается в том, что многие визуальные эффекты, которые были показаны в примерах в этой книге, можно обнаружить, когда взаимодействуешь с сайтом: наводишь на элементы, переводишь фокус и так далее. Разумеется, такие эффекты создаются намеренно.

В своей работе с заказчиками я часто добавляю эти улучшения взаимодействия в проект, ничего не говоря об этом, удивляя и радуя их, когда они обнаруживает их. И, что еще важнее, удивляя и радуя посетителей сайта заказчика, когда они обнаруживают эти эффекты.

Хотите, чтобы это работало во всех браузерах, о которых только можно подумать? Ну, за это придется еще заплатить. Хм.

Что дальше?

Какое же будет будущее? CSS3 в целом заключает в себе много больше, чем то, о чемговорится в этой маленькой книге. Я хотел честно сосредоточиться на том, что практически применимо сегодня, избегая те места спецификации, которые по-прежнему прорабатываются и для которых нет такой распространенной поддержки.

Но пока что все идет хорошо. Поддержка новых свойств появляется в каждой версии WebKit, Mozilla и Opera. Быстрое принятие новых свойств с помощью браузерных префиксов – это то, что продвигает инновации. Следить за тем, что нового, и отмечать, когда наступает переломная точка в поддержке нового свойства современными браузерами, – такие действия могут дать представление об использовании свойств CSS3 в настоящих проектах.

Нас ждут интересные времена с IE9 Beta. Да, я действительно сказал это. Мы видим, как в Internet Explorer появляется все больше и больше поддержки CSS3, и это замечательно.

В итоге мы сможем полагаться на CSS3 не только для улучшения взаимодействия, но и для критических визуальных элементов (расположение блоков на странице – один из основных примеров). Кажется, что по этому пути мы движемся довольно медленно, но для хорошего результата совершенно необходимо, чтобы все происходило грамотно и продуманно. Пока мы движемся по этому пути, не стесняйтесь брать и использовать то, что хорошо работает сейчас. Вы, ваши заказчики и жители веба будут рады.

Дополнительные материалы и ресурсы

CSS3.info уже давно публикует новости, примеры и информацию о развитии событий:
http://www.CSS3.info

Смотрите также раздел «Предпросмотр», где выложены демки конкретных свойств:
http://www.CSS3.info/preview

Раньше я говорил, что не нужно читать спецификации, но для должного понимания картины в целом, чтобы подготовиться к тому, что будет дальше, и чтобы видеть, в каком состоянии какие модули находятся (рабочий черновик, кандидат в рекомендации и т. д.), смотрите сюда:

http://www.w3.org/style/CSS/current-work

Или сюда, чтобы узнать больше о самих модулях, как они разделены и что в них содержится:

http://www.w3.org/tR/CSS3-roadmap

Блоги разработчиков всех основных браузеров – замечательный источник знаний, который позволит быть в курсе того, какие свойства в каких браузерах поддерживаются. Я советую подписаться на эти блоги, чтобы всегда знать, какие свойства поддерживаются, от каких отказываются и с чем экспериментируют:

http://webkit.org/blog
http://blog.mozilla.com
http://dev.opera.com/articles/css
http://blogs.msdn.com/b/ie

Появилось несколько сайтов, которые могут помочь понять совместимость браузеров и узнать, в каких их версиях поддерживаются конкретные свойства:

http://caniuse.com
http://www.quirksmode.org/css/contents.html
http://html5readiness.com *Пусть URL не смущает вас; на сайте также пишут про CSS3.*

Есть браузерные инструменты, которые дают графический интерфейс для создания рабочего кода во всех браузерах, где есть поддержка конкретного свойства. Эти инструменты могут сильно пригодиться в изучении:

http://CSS3generator.com
http://CSS3please.com
http://gradients.glrzad.com
http://tools.westciv.com
http://border-radius.com

Наконец, решения на JavaScript могут помочь расширить поддержку CSS3 во многих других браузерах. Для критических визуальных событий, которые должны работать везде с помощью современного CSS3, есть несколько вариантов:

http://www.modernizr.com
http://ecsstender.org
http://selectivizr.com/ *Эмулятор псевдоклассов для IE5.5–8.*

Спасибо, что прочитали эту книгу! Отправляйтесь создавать замечательные вещи. Dream big and implement small.

Об издательстве A Book Apart

Веб-дизайн – это междисциплинарное мастерство и высочайшая, лазероподобная точность. Таково направление мыслив наших книгах, издаваемых для людей, которые создают сайты. Мы рассказываем о развивающихся и важнейших темах в мире веб-дизайна и веб-разработки, придерживаясь ясности и прежде всего лаконичности – потому что погруженные в работу дизайнеры и разработчики не могут позволить себе терять время попусту.

Цель каждой книги – разобраться в сложной теме и сделать это понятно и быстро, чтобы читатель мог вскоре вернуться к работе. Спасибо, что поддерживаете нашу миссию: дать всем профессионалам инструменты, которые нужны, чтобы развивать веб.

Об авторе

Дэн Сидерхолм – веб-дизайнер, писатель, муж и отец, живущий в Салеме, штат Массачусетс. Он основал SimpleBits, крохотную дизайн-студию. Будучи признанным специалистом в области веб-дизайна, основанного на стандартах, Дэн работал с YouTube, MTV, Google, Yahoo, ESPN, Fast Company, Blogger и другими компаниями. В своих дизайн-проектах, публикациях и выступлениях он следует принципам гибкого и адаптивного дизайна, использующего веб-стандарты.

Дэн – соосновательи дизайнер Dribbble, сообщества дизайнеров, где они в реальном времени делятся друг с другом проектами, над которыми работают. Ранее он основал Cork’d – первую социальную сеть для страстных любителей вина.

Дэн – автор трех бестселлеров: Handcrafted CSS (New Riders)[20], Bulletproof Web Design[21], Second Edition (New Riders) и Web Standards Solutions, Special Edition (Friends of ED). Также он играет на укулеле (четырехструнный щипковый музыкальный инструмент. *Прим. перев.*) и называет себя любителем космических путешествий.

[20] Издание на русском языке: Седерхольм Д. CSS ручной работы. М.: Питер, 2011.

[21] Издание на русском языке: Седерхольм Д. Пуленепробиваемый веб-дизайн. Библиотека специалиста. М.: Питер, 2012.