Глава 3

Гибкая работа с цветом при помощи RGBA

Использование незатейливых форм и простых, удобных инструментов и средств позволяет достигнуть очень высокого уровня мастерства. Так, с помощью цвета и текстуры мы придаем индивидуальность каждому автомобилю.

Ларри Эриксон, бывший дизайнер компании Ford

Как-то у меня был заказчик (я буду называть его «Стэнли»), который, ознакомившись с идеей дизайна сайта, выдал такой комментарий: «Мне совершенно не нравится этот зеленый цвет. У моей бывшей девушки было такого же гадкого цвета одеяло. Нельзя использовать этот цвет».

Стэнли (как он сам себя называет, «не дизайнер») был определенно недоволен цветом или по крайней мере этим конкретным оттенком. Ни для кого не секрет, что *цвет воздействует на эмоции*. У каждого есть любимый и/или нелюбимый цвет. И нет сомнений в том, что в мире веб-дизайна цвет является важным и мощным (и, конечно, доступным и распространенным) ресурсом.

Ларри Эриксон, бывший дизайнер автомобилей компании Ford, говорит об этом в эпиграфе. Цвета и текстуры достаточно, чтобы подчеркнуть индивидуальность, если их применить к продуманной структуре. Хотя в эпиграфе речь идет о различных моделях автомобилей, я думаю, вы согласитесь, что к веб-дизайну это, в общем-то, тоже относится.

Продолжая переоценивать старые методы и самые лучшие технические приемы, особенно интересно будет поговорить о цвете с точки зрения альфа-прозрачности, а также о гибкости и текстуре, в основе которых она лежит (рис. 3.1).

ПРИМЕЧАНИЕ

Цветной вариант этой главы вы найдете на сайте издательства по адресу *www.piter.com*, на странице, посвященной этой книге (ссылка «*Отрывок*»).

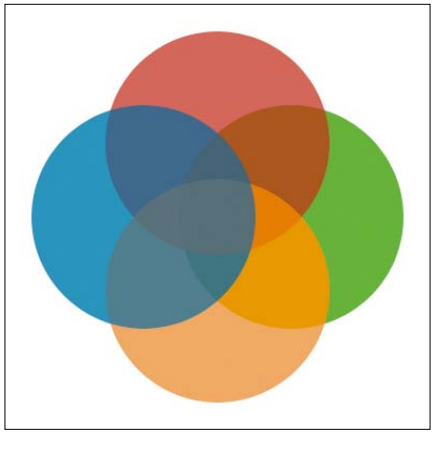

Рис. 3.1. Пример полупрозрачных цветов, накладывающихся друг на друга

Когда мы говорим об альфа-прозрачности, мы имеем в виду способность цвета иметь разную степень *непрозрачности*. Благодаря этой дополнительной характеристике средством выражения индивидуальности может быть такое простое свойство как цвет. Наложение полупрозрачных цветов друг на друга открывает нам целый мир визуальной неповторимости, что особенно ценится в веб-интерфейсах.

Раньше мы бы использовали для этого PNG-изображения (которые могут быть полупрозрачными) или, возможно, свойство opacity (свойство CSS3, которое неплохо поддерживается в Firefox 1.5, Safari 1.2 и более поздних версиях, а также в Opera 9 и более поздних версиях).

Сейчас существует новая интересная цветовая модель, добавленная в CSS3 Color Module, которая позволяет задавать одновременно цвет и прозрачность: RGBA. Эта модель дает нам особые преимущества, которые будут рассматриваться в этой главе. Мы также обсудим плюсы и минусы других методов. В процессе мы будем постепенно улучшать наш шаблон Tugboat с помощью RGBA, а также возьмемся за создание раздела This Week's Specials, состоящего из картинок и описаний (рис. 3.2). Но сначала я объясню, как работает RGBA.

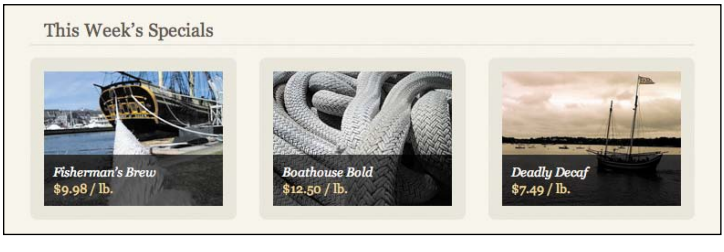

Рис. 3.2

Что такое RGBA?

Сначала давайте поговорим о том, что такое RGBA. RGB (что означает Red, Green, Blue — красный, зеленый, синий) — цветовая модель, позволяющая задавать цвет с помощью численных значений трех составляющих. Комбинируя их, можно получить множество различных оттенков.

На рисунке 3.3 показана палитра цветов Photoshop. Обратите внимание, что выбранный нами голубой цвет может обозначаться несколькими способами, включая знакомую шестнадцатеричную запись, которую мы бы использовали в CSS.

```
body {
   background: #3792b3;
   }
```

Можно задать тот же цвет в RGB, используя три десятичных значения (для красного, зеленого и синего).

```
body {
   background: rgb(55,146,179);
   }
```

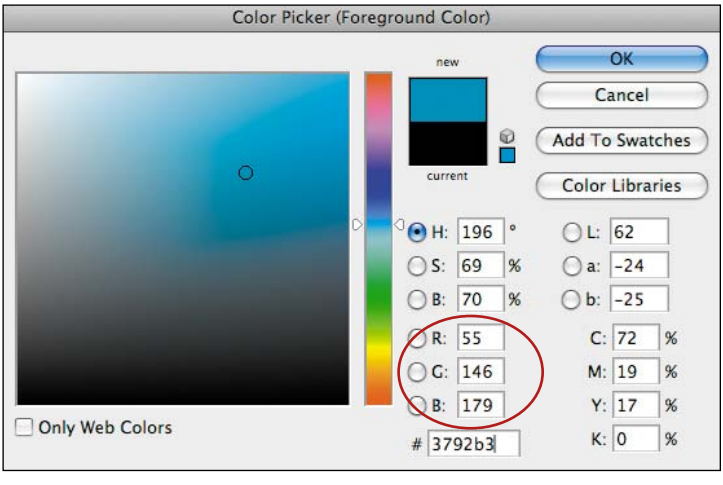

Рис. 3.3

В обоих случаях получается один и тот же оттенок, но используется разный синтаксис.

RGBA расшифровывается как Red Green Blue Alpha. На W3C объясняется: «В этой спецификации цветовая модель RGB расширена и включает составляющую альфа, позволяющую задать непрозрачность цвета» (*http://www.w3.org/TR/css3-color/#rgba-color*).

Это значит, что можно добавить четвертое значение (от 1 до 0), чтобы задать уровень непрозрачности данного RGB-цвета. Для полной непрозрачности используется значение 1, для полной прозрачности — 0.

Например, мы можем сделать наш голубой цвет полупрозрачным, добавив .5 в качестве четвертого значения после значений RGB.

```
body {
   background: rgba(55,146,179,.5);
   }
```

Рис. 3.4

На рисунке 3.4 показано, как один и тот же оттенок голубого может иметь различную степень непрозрачности, если он задан в RGBA. Ниже приведен соответствующий код; в нем выделены фрагменты, в которых задается цвет (простите, что я использую внутренний стиль, но так проще показать сразу все).

```
<div style="float: left; width: 50px; height:
     50px; background: rgba(55,146,179,1);">
     </div>
<div style="float: left; width: 50px; height:
     50px; background: rgba(55,146,179,.75);">
     </div>
<div style="float: left; width: 50px; height:
     50px; background: rgba(55,146,179,.5);">
     </div>
<div style="float: left; width: 50px; height:
     50px; background: rgba(55,146,179,.25);">
     </div>
<div style="float: left; width: 50px; height:
     50px; background: rgba(55,146,179,.1);">
     </div>
```

Для одного и того же значения RGB можно добиться разной степени непрозрачности: `1` задает непрозрачность 100%, `.75` дает тот же цвет, но с непрозрачностью 75%, `.5` означает 50% и т. д.

Возможность задавать прозрачность цвета легко и быстро прямо в таблице стилей — это прекрасно. Но как насчет собственно свойства `opacity` и чем оно отличается от RGBA?

Свойство opacity против RGBA

В CSS3 можно добавлять прозрачность с помощью свойства `opacity`. Просто задайте значение от `1` до `0`, чтобы определить степень прозрачности любого элемента.

Например, если вы хотите добиться непрозрачности 65% для всех абзацев на странице, можно написать такой код.

```
p {
   opacity:.65;
   }
```

ПРИМЕЧАНИЕ

О различиях между opacity и RGBA можно узнать на http://www.css3.info/*introduction-opacity-rgba/*.

Значительная разница между `opacity` и RGBA заключается в том, что `opacity` задает прозрачность элемента *и всего, что в нем содержится*, тогда как RGBA задает прозрачность *только фона или цвета элемента*.

КОГДА ПЕРЕСТАЕТ РАБОТАТЬ СВОЙСТВО OPACITY

Пусть, например, у нас есть полупрозрачный голубой блок с текстом, который мы хотим наложить на фон с узором (рис. 3.5).

Рис. 3.5

Разметка для этого маленького модуля будет выглядеть примерно так.

```
<div class="coupon">
   <p>Save 50% off all coffee, this week only!
     </p>
</div>
```

Чтобы блок выглядел так, как мы хотим, но без применения прозрачности, можно написать такие основные стили.

```
.coupon {
   padding: 1em;
   background: #3792b3;
   border-radius: 1em;
   -webkit-border-radius: 1em;
   -moz-border-radius: 1em;
   }

.coupon p {
   font-family: Helvetica, sans-serif;
   font-size: 2em;
   font-weight: bold;
   color: #a14141;
   }
```

Обратите внимание, что здесь мы используем закругленные углы, применяя свойство `border-radius`, описанное в главе 2. Вот такие мы молодцы.

Использование свойства opacity

Если бы мы использовали свойство `opacity`, чтобы сделать блок прозрачным, то нам бы понадобилось добавить соответствующее правило в объявление `.coupon`, которое отвечает за создание голубого блока.

```
.coupon {
   padding: 1em;
   background: #3792b3;
   border-radius: 1em;
   -webkit-border-radius: 1em;
   -moz-border-radius: 1em;
   opacity:.25;
   }
```

На рисунке 3.6 показан результат. Как вы видите, непрозрачным на 25% стал не только голубой блок, но и текст внутри блока. А это уже не то, чего мы хотели.

Рис. 3.6

Использование цветовой модели RGBA

Вместо свойства `opacity` давайте попробуем использовать RGBA. Так мы сможем задать цвет фона и степень прозрачности блока с помощью одного правила.

```
.coupon {
   padding: 1em;
   background: rgba(55,146,179,.25);
   border-radius: 1em;
   -webkit-border-radius: 1em;
   -moz-border-radius: 1em;
   opacity:.25;
   }
```

Рис. 3.7

Здесь мы изменили способ задания цвета фона: вместо шестнадцатеричной записи (`#3792b6`) мы использовали ее эквивалент в модели RGB (`55,146,179`). Затем, задав альфа-значение `.25`, мы сделали этот голубой цвет непрозрачным на 25%.

Существенное отличие здесь вот в чем: используя RGBA, мы меняем прозрачность только фона, а не всего блока и его содержимого (как это произошло со свойством `opacity`). Теперь мы выполнили нашу задачу успешно (рис. 3.7).

Совместимость

Как и свойство `border-radius`, рассмотренное в предыдущей главе, RGBA является удивительно гибким инструментом. Но так как эту модель не поддерживает Internet Explorer, ее однозначно можно считать средством для *прогрессивного оформления*. Сейчас поддержка RGBA есть в движке WebKit (Safari, Chrome, iPhone и т. д.), Firefox 3 и Opera 10. Если вас устраивает тот факт, что эта цветовая модель будет работать только в прогрессивных браузерах, то все в порядке.

RGBA	
WebKit (Safari)	✓
Firefox 3	✓
Opera 10	✓

Похожим образом поддерживается свойство `opacity`, хотя, по большей части, оно было принято немного раньше, чем RGBA. Но и оно не поддерживается браузером Internet Explorer.

Свойство opacity	
WebKit (Safari 1.2+)	✓
Firefox 1.5+	✓
Opera 9+	✓

Более подробно об этом говорится на *http://dev.opera.com/articles/view/css-and-opacity-methods-for-creating-tr/*.

А как насчет других браузеров?

RGBA — необычайно гибкое средство, позволяющее задавать прозрачность цвета с помощью одной простой строки кода. И здесь вы наверняка вспомните тезис из главы 2: возможность управлять элементами дизайна через таблицы стилей вместо того, чтобы редактировать изображения, не только облегчает жизнь, но и позволяет работать быстрее и эффективнее — браузер берет на себя всю тяжелую работу. И, опять же, это перспективное направление в CSS, поэтому мы можем уже сейчас начать экспериментировать с новыми возможностями в браузерах, поддерживающих продвинутые стили (такие как RGBA).

К счастью, использование синтаксиса цветовой модели RGBA безопасно для браузеров, которые ее не поддерживают. Например, Internet Explorer просто проигнорирует такое правило.

Создание резервных правил для неполноценных браузеров

Поскольку такие браузеры, как Internet Explorer, игнорируют правила с RGBA, неплохо было бы задать «резервный» цвет до объявления цвета RGBA.

Так, если в нашем примере мы хотим добавить чистый голубой цвет для браузеров, не поддерживающих RGBA, мы должны расположить стили в таком порядке.

```
.coupon {
   padding: 1em;
   background: #c4dada; /* для IE */
   background: rgba(55,146,179,.25); /* для
     браузеров, поддерживающих RGBA */
   border-radius: 1em;
   -webkit-border-radius: 1em;
   -moz-border-radius: 1em;
   }
```

На рисунке 3.8 показано, как будет выглядеть наш блок в Internet Explorer или другом браузере, не поддерживающем RGBA (а также `border-radius`). Чистый голубой цвет,

который мы используем в качестве резервного варианта, задан первым; далее мы переопределяем его с помощью цвета RGBA с непрозрачностью 25% для браузеров, которые поддерживают эту модель. Вот это и есть настоящее прогрессивное оформление!

Рис. 3.8. Как может выглядеть наш пример в браузерах, не поддерживающих RGBA и border-radius

Хотя элементы вашего дизайна потеряют прозрачность в браузерах, не поддерживающих RGBA, вы все же будете полностью управлять цветами внутри таблицы стилей. Причудливые прозрачные элементы — для прогрессивных браузеров, непрозрачный запасной вариант (но функциональный и читаемый) — для остальных.

ЗАПОЛНЕНИЕ PNG-ИЗОБРАЖЕНИЯМИ

Еще один вариант (который будет работать в большинстве браузеров) — это заполнение PNG-изображениями. Этот формат поддерживает альфа-канал, и поэтому PNG-изображение можно было бы использовать в качестве фона при создании такого полупрозрачного блока, какой мы получили с помощью RGBA.

Например, если мы создадим небольшое PNG-изображение, добавив слой нашего голубого цвета с непрозрачностью 25% (рис. 3.9), мы сможем заполнить им фон в блоке. Эффект будет таким же.

```
.coupon {
   padding: 1em;
   background: url(../img/bg.png);
   border-radius: 1em;
   -webkit-border-radius: 1em;
   -moz-border-radius: 1em;
   }
```

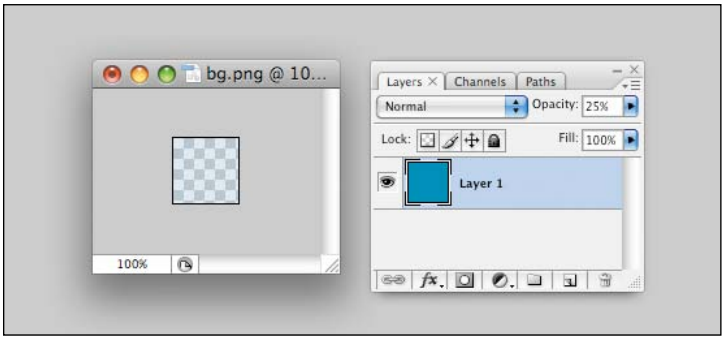

Рис. 3.9. Создание полупрозрачного PNG-изображения, предназначенного для заполнения фона, в Photoshop

PNG-изображения с альфа-прозрачностью отображаются правильно в Safari, Firefox и Opera (так же, как и RGBA). Но, кроме того, они работают в Internet Explorer версий 7 и 8. В версии 6 есть поддержка PNG-формата, но нет поддержки альфа-канала, так что ваши полупрозрачные PNG-изображения будут абсолютно непрозрачными в IE6.

Существуют различные приемы, позволяющие добиться того, чтобы альфа-канал поддерживался в IE6. В некоторых используется JavaScript, в некоторых — специфическое для Microsoft CSS-свойство `filter` (если PNG-изображение задано в качестве фона).

Проблемы с цветом и степенью непрозрачности

И снова, как и в случае с закругленными углами, использование PNG-изображений для создания полупрозрачного фона связано с тем недостатком, что цвет и степень прозрачности задаются при создании изображения. А если в таких ситуациях использовать RGBA, то можно быстро изменять оттенок и степень прозрачности прямо в таблице стилей. И не нужно будет создавать новое изображение каждый раз, когда понадобится задать новый цвет или отрегулировать прозрачность. Этот факт важно учитывать, когда вы решаете, какой метод выбрать. Если вы не против того, что для браузеров, не поддерживающих RGBA, придется задавать непрозрачный цвет (или какой-либо другой запасной вариант), обязательно начинайте экспериментировать.

Превосходный инструмент для создания прототипа

Я уже приводил этот аргумент в главе 2, когда мы говорили о свойстве `border-radius`, и сейчас, при обсуждении RGBA, он тоже уместен. Независимо от того, используете ли вы RGBA при создании проекта, эта модель может быть ценным инструментом для создания прототипов интерфейсов. То есть если при первоначальной разработке проекта вы пользуетесь Safari, Firefox или даже Opera, модель RGBA очень удобна для выполнения однотипных операций с цветом и прозрачностью. Изменения и корректировки можно вносить в код быстро и безболезненно.

А уже потом вы сможете выбрать подходящий метод окончательной реализации: возможно, это будет RGBA и резервные правила для других браузеров или же PNG-изображения, работающие также в IE7+.

И, опять же, из этой главы стоит сделать такой вывод: уже сейчас можно экспериментировать с RGBA. И можно даже использовать эту модель при создании проекта, если вы являетесь сторонником идеи прогрессивного оформления.

Не только фон

Еще одно исключительное свойство RGBA заключается в том, что эту модель можно применять не только к фоновым изображениям. С ее помощью можно задавать цвет и прозрачность *гипертекста*. Это открывает нам массу новых возможностей, а также элегантных и гибких решений.

Я, к примеру, использовал RGBA, чтобы уменьшить непрозрачность фрагмента гиперссылки на сайте моей студии (*http://simplebits.com/*).

Уменьшение насыщенности шрифта с помощью RGBA

На рисунке 3.10 показано несколько ссылок, и у каждой ссылки есть заголовок (написанный полужирным шрифтом) и подзаголовок (написанный обычным шрифтом). Я могу выбрать в качестве основного цвета для ссылок зеленый, а потом немного уменьшить насыщенность шрифта, задав непрозрачность для этого же цвета с помощью RGBA.

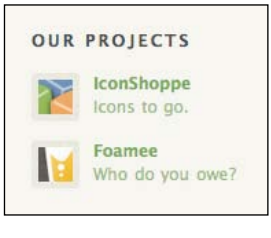

Рис. 3.10

Иными словами, вместо двух цветов для ссылки я выберу один и с помощью RGBA сделаю его светлее, чтобы он стал ближе к цвету фона и отличался от более насыщенного цвета.

Для этих ссылок в разметке используется неупорядоченный список. При этом в тег `<a>` заключены изображение, заголовок (выделенный в отдельный элемент `<strong>`) и подзаголовок.

```
<ul class="lst group">
   <li>
      <a href="http://iconshoppe.com/">
         <img src="/img/ishoppe-25.gif"
     alt="icon" />
         <strong>IconShoppe</strong> Icons to go.
      </a>
   </li>
   <li>
      <a href="http://foamee.com/">
         <img src="/img/foamee-25.gif" alt="icon"
         />
         <strong>Foamee</strong> Who do you owe?
      </a>
   </li>
</ul>
```

Таблицы стилей, в которых задается цвет ссылки, будут выглядеть примерно так.

```
ul. lst li a {
   color: #76a65c; /* для всех браузеров */
   color: rgba(118,166,92,.75); /* для браузеров,
     поддерживающих RGBA */
   }

ul. lst li a strong {
   display: block;
   color: #76a65c; /* переопределение для браузеров,
     поддерживающих RGBA */
   }
```

Как видите, для всей ссылки мы задаем значение RGBA с непрозрачностью 75%, а затем переопределяем его значение (в шестнадцатеричной записи) для элемента `<strong>`, используя тот же цвет, но повышая непрозрачность до 100%. Кроме того, мы добавили строку `display: block;`, чтобы заголовок располагался на отдельной строке.

Этот метод освобождает нас от необходимости подбирать разные оттенки одного и того же цвета. Остается выбрать один основной цвет, а потом с помощью RGBA менять значения его непрозрачности.

Браузеры, не поддерживающие RGBA, проигнорируют одноименное правило, и в результате для ссылок будет использоваться основной цвет, заданный правилом `color: #76a65c;` (рис. 3.11).

Рис. 3.11

WILSONMINER.COM

Похожим образом RGBA использует на своем сайте Вилсон Майнер (рис. 3.12): он уменьшает непрозрачность черного текста, приближая его оттенок к цвету фона (*http://www.wilsonminer.com/*).

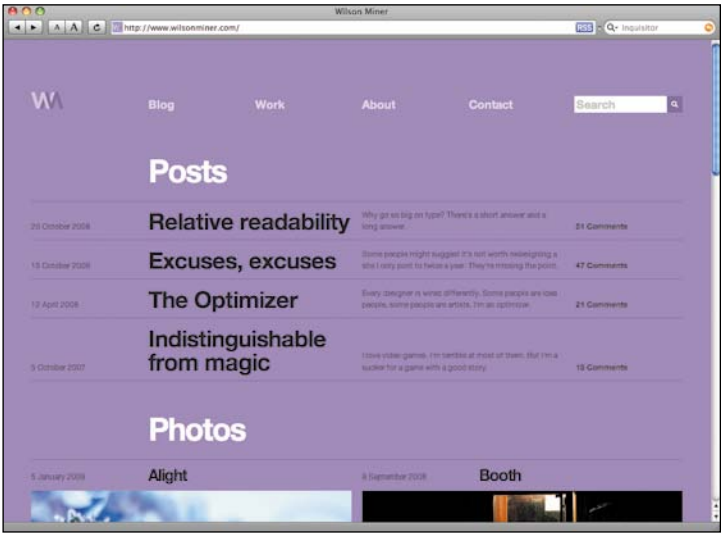

Рис. 3.12. Вилсон Майнер искусно применяет RGBA для задания цвета текста на своем сайте

В одной из записей блога (*http://www.wilsonminer.com/posts/2008/oct/15/excuses-excuses/*) он объясняет:

> *Я хотел немного расширить проект, но при этом добиться гибкости дизайна. Поэтому я сделал так, чтобы можно было легко изменять цвета и фоновые изображения для фрейма сайта и домашней страницы. Я начал с классического зеленого цвета, но буду время от времени обновлять оформление.*
>
> *На домашней странице, в частности, значения RGBA используются для придания прозрачности тексту. Благодаря*

этому, к цвету текста добавляется оттенок фона, каким бы он ни был (чистым цветом или изображением). Даже «черный» текст на странице сделан полупрозрачным, и потому содержит частичку фонового цвета. Поэтому контраст становится более мягким (в полиграфии похожий эффект получается при последовательном наложении красок).

Так что, хотя на этом сайте почти весь текст черный, Вилсон с помощью RGBA смягчает этот цвет, и текст немного сливается с фоном (независимо от того, какого цвета фон) (рис. 3.13). Можно задать другой цвет фона, не меняя цвета текста и степеней прозрачности. В полупрозрачном тексте в любом случае будет содержаться оттенок фона. Вот это и есть гибкий цвет.

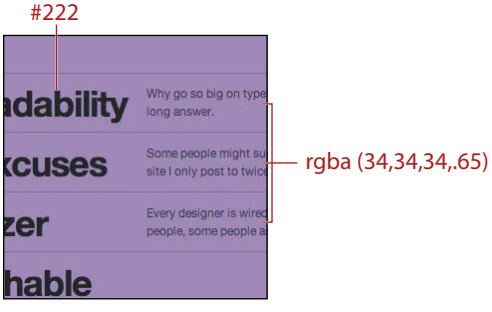

Рис. 3.13

КАК ЗАПРОСТО ДОБАВИТЬ HOVER-ЭФФЕКТ

У RGBA есть еще одно преимущество: можно легко задавать изменение цвета ссылки при наведении курсора мыши. Добиться hover-эффекта можно, всего лишь уменьшая непрозрачность ссылки, и не нужно выбирать никакой другой цвет. При этом дизайн будет гибким: можно без проблем изменить цвет фона страницы или основной цвет ссылки.

Я решил реализовать эту идею в шаблоне Tugboat. Сделаем так, чтобы цвет при наведении становился полупрозрачным вариантом основного цвета ссылки (рис. 3.14).

```
a: link, a: visited {
   font-weight: bold;
   text-decoration: none;
   outline: none;
   color: #3792b3;
   }
```

```
a: hover {
   color: rgba (55,146,179,.65);
   }
```

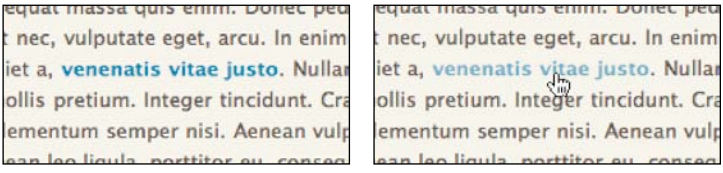

Рис. 3.14. Обычный цвет ссылки и цвет при наведении (в увеличенном масштабе)

РАСКРАСКА ПО НОМЕРАМ

Такое смешивание цветов напомнило мне об одном интереснейшем интервью с Артом Лоцци, который был художником студии «Хана-Барбера» в далекие 60 годы. Джон Крисфалуси (создатель скандально известного мультипликационного сериала «Рен и Стимпи») рассуждает в своем блоге о работе Арта (пример приведен на рисунке 3.15), анализируя фоновые рисунки к мультфильму «Мишка Йоги» (полностью интервью можно прочитать на *http://johnkstuff.blogspot.com/2006/12/color-theory-art-lozzi-explains-some.html*).

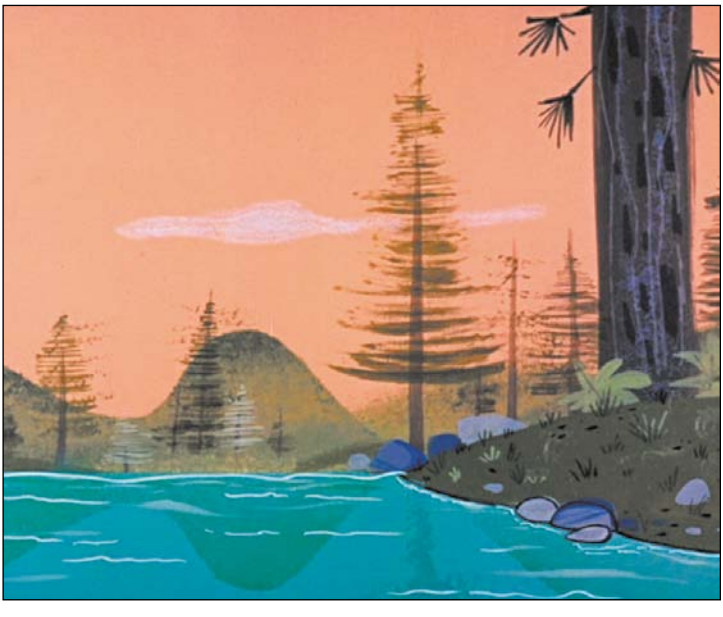

Рис. 3.15. Один из великолепных фоновых рисунков Арта Лоцци

Лоцци рассказывает, как они вначале раскрашивали весь фон одним цветом:

> *Этот цвет наносится на толстенный бристольский картон (не обычный картон и не холст), а уже поверх него рисуется все остальное; и если появляются какие-то «дыры» или промежутки, то цвет неба не позволяет целостности рисунка нарушиться.*

В ответ Крисфалуси излагает свой взгляд на метод Лоцци:

> *Через маленькие просветы в холмах, не до конца прорисованных губкой, проглядывает персиковое небо. Создается эффект смешивания основного фонового цвета с остальными цветами фона, который и объединяет все это в гармоничную цветовую схему.*
>
> *Точно так же цвет холма виден сквозь деревья, и так далее.*
>
> *В результате такого смешивания все элементы рисунка становятся частью цветовой системы, а не группой отдельных объектов (каждый из которых состоит из совершенно разных цветов), нарушающих целостность изображения и конкурирующих между собой.*

Смешивать цвета для создания гармоничной палитры. Это то, что Арт делал десятилетия назад с помощью краски и кистей, и то, для чего Вилсон Майнер использовал RGBA, когда он хотел добиться удачного сочетания цвета текста и фона в Сети. Мастер использует все средства!

Создание раздела This Week's Specials

Теперь, когда мы хорошо знаем, что такое RGBA и как работать с этой моделью, а также имеем представление о других способах задания прозрачности в Сети, давайте вернемся к шаблону Tugboat и проанализируем раздел This Week's Specials. Потом мы обсудим способ его реализации, где будет использоваться RGBA для создания изображений, накладываемых на каждую из фотографий (рис. 3.16).

Рис. 3.16

На каждую фотографию накладывается полупрозрачное изображение, на котором расположено название и цена. Это обычное дизайнерское решение, гибкий способ добавлять метаинформацию к фотографиям, не зная заранее их цвета/яркости/контрастности. Чтобы не нарушалась удобочитаемость текста, используется полупрозрачное фоновое изображение, позволяющее создать удачный дизайн вне зависимости

от фотографии, на которую оно накладывается. Идеальное место, чтобы поиграть с RGBA.

СОЗДАНИЕ РАЗМЕТКИ

Начнем с разметки. Будем использовать для трех наших напитков недели упорядоченный список, где для каждого элемента мы хотим, чтобы само изображение, название и цена были активными. Поэтому мы поместим каждый элемент списка в тег `<a>`, используя строковые элементы для выделения названия и цены (так как блоковые элементы нельзя располагать внутри гиперссылки).

Я также добавил несколько дополнительных элементов. Для чего они нужны, станет понятно на завершающей стадии, после создания стилей.

```
<ol class="specials">
   <li>
      <div class="special">
         <div class="special-img">
            <a href="/specials/fb">
               <img src="img/boat.jpg" alt="ship"
               />
               <span>
                  <strong>Fisherman&#8217;s
                     Brew</strong>
                  <em>$9.98 / lb.</em>
               </span>
            </a>
         </div>
      </div>
   </li>
   <li>
      <div class="special">
         <div class="special-img">
            <a href="/specials/bb">
              <img src="img/ropes.jpg" alt="ropes" />
               <span>
                  <strong>Boathouse Bold</strong>
                  <em>$12.50 / lb.</em>
               </span>
            </a>
         </div>
```

```
        </div>
    </li>
    <li>
        <div class="special">
            <div class="special-img">
                <a href="/specials/dd">
                    <img src="img/fame.jpg"
                        alt="sailboat" />
                    <span>
                        <strong>Deadly Decaf</strong>
                        <em>$7.49 / lb.</em>
                    </span>
                </a>
            </div>
        </div>
    </li>
</ol>
```

КАК СДЕЛАТЬ СПИСОК ГОРИЗОНТАЛЬНЫМ

Первые несколько правил таблицы стилей задают ширину каждого элемента списка и его расположение вдоль левого края, чтобы превратить вертикальный список в горизонтальный.

```
ol.specials li {
    width: 210px;
    float: left;
    margin: 0 15px 1.5em 0;
    }
```

ПРИМЕЧАНИЕ

Как мы уже говорили во введении, применение таких стилей требует добавления таблицы стилей *reset. css*, которая обнуляет стандартные умолчания браузера.

На рисунке 3.17 показан результат добавления этого стиля: каждое изображение является ссылкой, как и расположенные ниже название и цена. Ширина каждой картинки — 210 пикселов; для каждого элемента мы также добавили небольшие поля справа и снизу.

Рис. 3.17

ДОБАВЛЕНИЕ ГРАНИЦЫ С ЗАКРУГЛЕННЫМИ УГЛАМИ

Применяя на деле то, что мы узнали из главы 2, давайте добавим к каждому элементу границу шириной 15 пикселов и сделаем ее углы закругленными с помощью свойств `border-radius`, которые мы знаем и любим.

Для этого нам нужно увеличить ширину каждого элемента списка на 30 пикселов.

```
ol.specials li {
   width: 240px;
   float: left;
   margin: 0 15px 1.5em 0;
   }

ol.specials li div. special {
   position: relative;
   border: 15px solid #e2e1d4;
   border-radius: 8px;
   -webkit-border-radius: 8px;
   -moz-border-radius: 8px;
   }
```

ПРИМЕЧАНИЕ

Я раньше этого не говорил, но в шаблоне Tugboat используется «резиновая» верстка. Однако в этой конкретной ситуации я для простоты использую значения ширины в пикселах. Оставайтесь с нами, и в главе 6 я объясню, как эти стили можно применить в случае «резиновой» верстки.

Также к каждому элементу списка мы добавили правило `position: relative;`, чтобы в будущем можно было поместить название и цену на фотографию. На рисунке 3.18 показан наш список с добавленной закругленной границей, и теперь мы готовы приступить к наложению изображения.

Рис. 3.18

ДОБАВЛЕНИЕ НАКЛАДЫВАЕМОГО ИЗОБРАЖЕНИЯ

Теперь давайте добавим накладываемое изображение с названием и ценой. Так как мы хотим, чтобы вся картинка тоже была активной, воспользуемся элементом `<span>`, который содержит цену и название. Сделаем его блоковым и поместим поверх фотографии, закрепив у нижнего края.

Следующее объявление берет на себя кучу всего, включая позиционирование и задание основных стилей шрифта для названия и цены. Сейчас мы также зададим темно-серый фон для накладываемого изображения.

```
ol.specials li div.special a span {
   display: block;
   position: absolute;
   width: 100%;
   bottom: 0;
   left: 0;
   font-family: Georgia, serif;
   font-size: 1.1em;
   font-weight: normal;
   line-height: 1.3em;
   color: #ccc;
   background: #333;
   }
```

На рисунке 3.19 показано текущее состояние дел. Вы наверняка заметите, что нужно еще добавить отступы и подправить оформление текста названия и цены, чтобы все выглядело так, как надо.

Рис. 3.19

СТИЛИ НАЗВАНИЯ И ЦЕНЫ

Теперь перейдем к шрифтовому оформлению и расположению названия и цены. Для этого будем использовать два новых объявления, в которых зададим отступ вокруг текста и изменим цвет. Также добавим `display: block;` (так как по умолчанию `<strong>` и `<em>` являются строковыми элементами), после чего название и цена будут располагаться на отдельных строках.

```
ol.specials li div.special a span strong {
   display: block;
   padding: 10px 10px 0 10px;
   font-weight: normal;
   font-style: italic;
   color: #fff;
   }

ol.specials li div.special a span em {
   display: block;
   padding: 0 10px 10px 10px;
   font-style: normal;
   color: #e3c887;
   }
```

На рисунке 3.20 показан результат добавления этих двух объявлений. Ура, мы почти закончили!

Рис. 3.20

ДОБАВЛЕНИЕ RGBA К НАКЛАДЫВАЕМОМУ ИЗОБРАЖЕНИЮ

На завершающем этапе добавим значение RGBA, чтобы сделать накладываемое изображение прозрачным: пусть оно будет черным с непрозрачностью 70%.

```
ol. specials li div. special a span {
   display: block;
   position: absolute;
   width: 100%;
   bottom: 0;
   left: 0;
   font-family: Georgia, serif;
   font-size: 1.1em;
   font-weight: normal;
   line-height: 1.3em;
   color: #ccc;
   background: #333;
   background: rgba(0,0,0,.7);
   }
```

Мы сохраним предыдущее значение `#333` для браузеров, не поддерживающих RGBA, и при этом текст останется разборчивым независимо от фотографии. Но мы все же переопределим этот цвет, задав значение RGBA: черный с непрозрачностью 70%. На рисунке 3.21 показан законченный список, в котором мы имеем дело с приятным гибким стилем накладываемого изображения для каждого напитка.

Рис. 3.21

Внесение изменений — быстро и просто

Регулировать цвет и степень прозрачности накладываемого изображения теперь очень просто: для этого нужно исправить соответствующие правила в CSS. К примеру, если мы хотим

сделать это изображение красным, можно быстро и просто изменить значение RGB (рис. 3.22).

```
ol.specials li div.special a span {
   display: block;
   position: absolute;
   width: 100%;
   bottom: 0;
   left: 0;
   font-family: Georgia, serif;
   font-size: 1.1em;
   font-weight: normal;
   line-height: 1.3em;
   color: #ccc;
   background: #600;
   background: rgba(102,0,0,.7);
   }
```

Рис. 3.22

В ЗАКЛЮЧЕНИЕ

Сейчас мы лишь поверхностно изучили возможности RGBA. Однако я надеюсь, что примеры из этой главы станут той искрой, из которой разгорится пламя, и вы будете и дальше пробовать добавлять прозрачность к элементам вашего дизайна. Возможность задавать непрозрачность одновременно с цветом — гибкий и мощный инструмент, а RGBA — всего лишь один из способов добиться этого. Но модель RGBA также позволяет сделать все быстро и легко, и за это мы ее любим. Как и в случае со свойством `border-radius`, в настоящее время ее поддерживает ограниченное число браузеров. Однако поскольку WebKit, Mozilla и Opera обязательно поведут за собой остальные браузеры, этот метод можно проверять и использовать уже сейчас, и не важно, делаете вы это для создания прототипа и итеративной разработки или же хотите внести что-то новое в создание массовых проектов.

После двух обычных глав о свойствах CSS3 и новых технических приемах, которые неодинаково отображаются в разных браузерах, пришло время остановиться и кое-что выяснить. А нужно ли вообще, чтобы веб-сайт выглядел одинаково во всех браузерах? В следующей главе мы поговорим как раз об этом, а также рассмотрим несколько забавных приемов прогрессивного оформления.