10 Преобразования, переходы и анимация с помощью CSS

За короткую историю Всемирной паутины у разработчиков было несколько вариантов добавления анимации к своим сайтам. Самая простая и примитивная анимация в рамках одного изображения предоставлялась в GIF-формате. Технология Adobe Flash стала инструментом номер один и позволяла создавать сложную анимацию и даже игры и веб-приложения, но обладала рядом недостатков. Изучить ее весьма непросто, и она не позволяет взаимодействовать с другим HTML-кодом на странице, представляющим изображения, заголовки и абзацы, из которых составлен веб-контент, а также недоступна для использования на мобильных устройствах. Эти минусы определили эру заката технологии Flash. JavaScript-сценарии позволяют анимировать все, что имеется на веб-странице, но ценой изучения всех тонкостей этого языка программирования. К счастью, CSS предоставляет способ перемещения, преобразования и анимации любого HTML-элемента, имеющегося на странице без обращения к любой из упомянутых здесь технологий.

Преобразования

Каскадные таблицы стилей включают несколько свойств, связанных с преобразованиями элемента веб-страницы, будь то вращение, масштабирование, перемещение этого элемента или его искажение вдоль горизонтальной или вертикальной оси (которое называется *скашиванием*). Преобразование можно использовать, например, для небольшого наклона (вращения) панели навигации или для увеличения изображения при нахождении поверх него указателя мыши. Можно даже сочетать несколько преобразований для получения весьма впечатляющих визуальных эффектов.

Основным свойством CSS для получения любого из этих изменений является `transform`. Оно используется с предоставлением *типа* желаемого преобразования и добавлением значения, указывающего на *степень* преобразования элемента. Например, для вращения элемента предоставляется ключевое слово `rotate`, за которым следует количество градусов поворота:

```
transform: rotate(10deg);
```

Показанное выше объявление приведет к вращению элемента на 10° по часовой стрелке.

Все популярные браузеры поддерживают преобразования: Internet Explorer 9, Safari, Chrome, Firefox и Opera. Тем не менее, чтобы преобразования работали в браузерах Safari и Internet Explorer 9, вам необходимо использовать вендорные префиксы, как это показано в первых двух строках следующего кода (см. также врезку далее):

```
-webkit-transform: rotate(10deg);
-ms-transform: rotate(10deg);
transform: rotate(10deg);
```

У преобразований в CSS есть одна странность: они не касаются окружающих элементов. То есть, если повернуть элемент на 45°, он может наложиться на те элементы, которые находятся выше, ниже его или по сторонам. Сначала браузеры выделяют элементу то пространство, которое он занимал бы при обычных обстоятельствах (до преобразования), а затем они выполняют преобразование элемента (его вращение, увеличение или скашивание). Этот процесс становится очевидным при увеличении размеров элемента путем использования функции `transform: scale` (см. далее подраздел «Масштабирование»). При увеличении масштаба элемента в два раза браузер увеличивает преобразуемый элемент, но при этом не смещает окружающий его контент, что обычно приводит к частичному перекрытию содержимого страницы (рис. 10.1). Иначе говоря, браузер сохраняет все остальные части страницы так, словно элемент не был масштабирован.

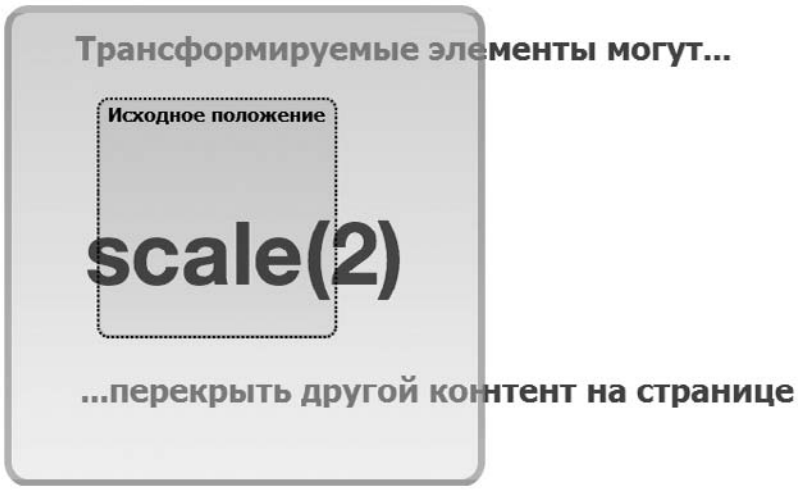

Рис. 10.1. Трансформируемые элементы игнорируются другими, окружающими их, элементами. Показанное здесь увеличение элемента div (большой квадрат) приводит к перекрытию им текста выше и ниже этого элемента. Квадрат посередине, выделенный пунктиром, представляет собой элемент div до его масштабирования

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Вендорные префиксы

Каскадные таблицы стилей — это всегда развивающийся набор правил. Даже сейчас, когда вы создаете страницу, умные люди в Консорциуме W3C работают над новыми свойствами CSS, а другие умные люди добавляют поддержку новых правил в браузеры. На самом деле иногда производители браузеров сами придумывают новые правила CSS, которые, по их мнению, было бы приятно использовать. Периодически члены Консорциума W3C придумывают новые свойства CSS, которые производители браузеров постепенно принимают.

Во время развития новых свойств и стандартов языка CSS производители браузеров настороже. Они не хотят окончательно принимать черновые свойства CSS, которые еще могут измениться. Точно так же в процессе экспериментов со свойствами CSS собственных разработок они не спешат утверждать, что пришли к согласованному стандарту. Чтобы пометить свойство CSS, как экспериментальное или пока еще не полностью утвержденное, производители (вендоры) браузеров используют префиксы перед его именем. Каждый из основных производителей браузеров имеет собственный *вендорный префикс*:

- `-webkit-` используется в Chrome, Safari, Opera и других браузерах, работа которых основана на технологии WebKit;
- `-moz-` применяется браузером Mozilla Firefox;
- `-ms-` используется в Internet Explorer.

Пока свойство CSS находится в разработке, браузер может поддерживать версию с вендорным префиксом. Например, когда было впервые предложено свойство `border-radius`, браузер Firefox поддерживал вариант `-moz-border-radius`, в то время как Safari поддерживал `-webkit-border-radius`.

Если какое-либо свойство требует указать вендорный префикс, вам, как правило, необходимо написать несколько строк кода, чтобы создать такой же эффект: по одной для каждого производителя и, наконец, одну с версией свойства без префикса:

- `::-webkit-input-placeholder { color: red; };`
- `::-moz-placeholder { color: red; };`
- `:-ms-input-placeholder { color: red; };`
- `:placeholder-shown { color: red; }.`

Обычно, если рабочая группа Консорциума W3C принимает свойство и утверждают его как окончательное, разработчики браузеров выпускают обновление своего продукта, поддерживающее изменения. Например, все основные браузеры теперь поддерживают свойство `border-radius` без необходимости использования вендорного префикса. На момент чтения этой книги некоторые свойства CSS по-прежнему будут в разработке, поэтому для них вам может понадобиться использовать вендорные префиксы. Эта книга подскажет вам, когда префиксы необходимы и как применять их с определенными свойствами.

Вращение

Разобраться в работе функции `rotate` свойства `transform` довольно просто: ей предоставляется значение угла в диапазоне от 0 до 360, а браузер вращает указанный элемент по кругу на заданное количество градусов (рис. 10.2). Чтобы указать значение угла, используется число, за которым следует сокращение `deg`. Например, для вращения элемента на 180° добавьте следующее объявление:

```
transform: rotate(180deg);
```

ПРИМЕЧАНИЕ

В приводимые здесь примеры не включаются вендорные префиксы, но при помещении кода в таблицу стилей нужно добавлять к нему свойства -webkit-transform, -moz-transform, -o-transform и -ms-transform.

Для вращения элемента против часовой стрелки можно применять отрицательные числовые значения. Например, средний элемент в верхней части рис. 10.2 повернут на 45° против часовой стрелки благодаря следующему объявлению:

```
transform: rotate(-45deg);
```

Значение `0deg` не выполняет никакого вращения, значение `360deg` выполняет одно вращение на полный оборот, а значение `720deg` — два полных вращения. Разумеется, внешний вид элемента, которому было назначено одинарное, двойное или тройное вращение, останется таким же, как и у элемента, не подвергавшегося никакому вращению (верхнее левое и нижнее правое изображения на рис. 10.2), поэтому использование значения с числом 360 и кратным ему может вызвать удивление. В CSS предоставляется механизм анимации изменений свойств CSS. Так, например, можно заставить кнопку прокрутиться четыре раза, когда указатель мыши пользователя находится поверх нее, путем установки начального значения вращения равным `0deg` и добавления стиля `:hover` для этой кнопки с поворотом на `1440deg`. Как это делается, будет вскоре показано.

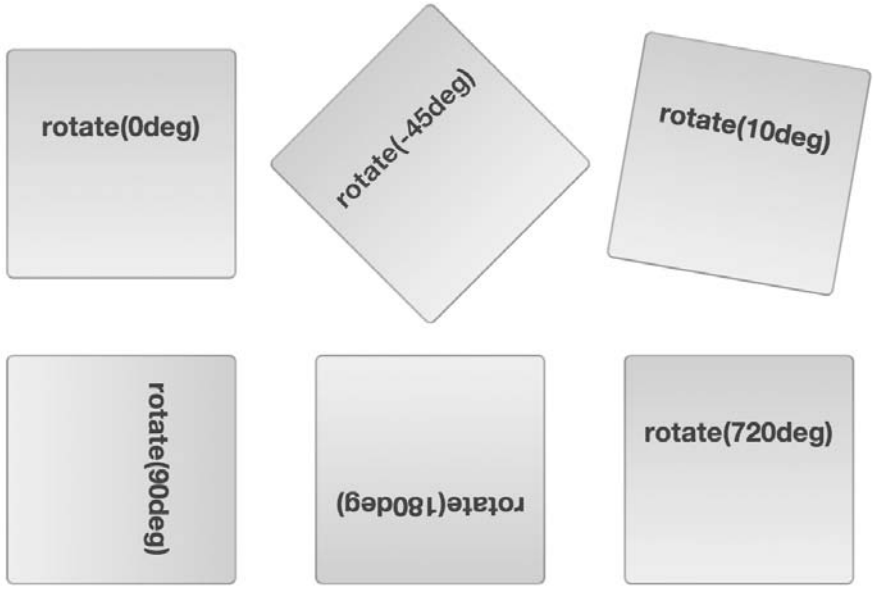

Рис. 10.2. Функция rotate позволяет любой элемент страницы — div, button, banner, photo или footer — повернуть как на небольшой, так и на совершенно немыслимый градус

Масштабирование

Элемент можно увеличить или уменьшить в размерах, воспользовавшись для этого функцией `scale` (рис. 10.3). Например, чтобы увеличить элемент вдвое, нужно добавить следующее объявление:

```
transform: scale(2);
```

Число в скобках является коэффициентом масштабирования — значением, на которое умножаются текущие размеры элемента. Например, `1` указывает на отсутствие масштабирования, `.5` указывает на половину текущего размера, а `4` увеличивает элемент в четыре раза. То есть числа в диапазоне между 0 и 1 приводят к уменьшению, а числа больше 1 — к увеличению элемента (значение `0` фактически делает элемент невидимым на странице).

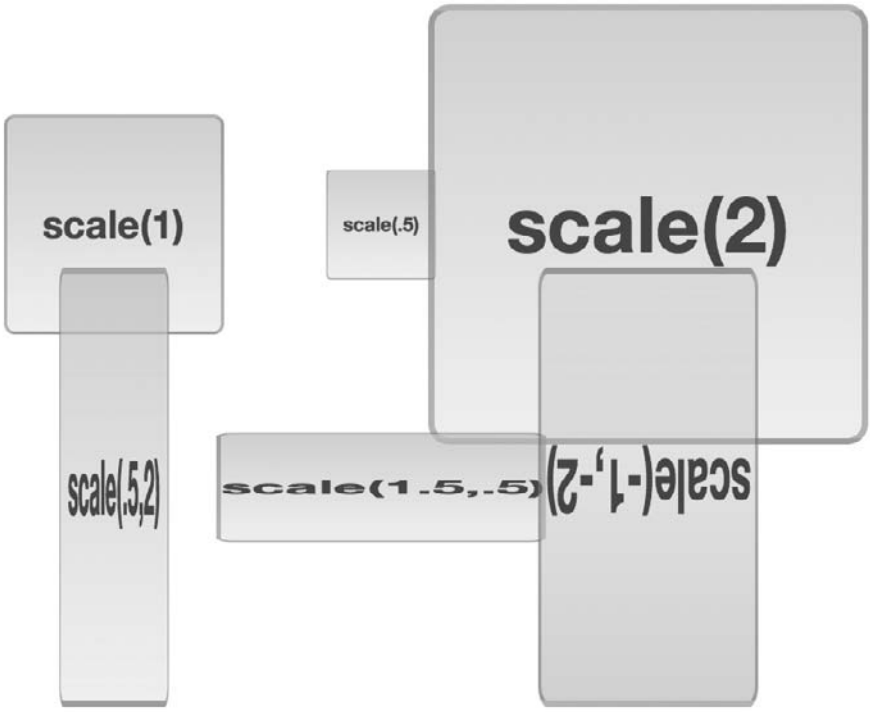

Рис. 10.3. Для увеличения размера любого элемента на странице используется функция scale. Но помните, что при увеличении элемента браузер не убирает другие окружающие элементы

На это число происходит масштабирование элемента и всего, что в нем находится. Например, если масштабировать контейнер `div` с коэффициентом `2`, то вдвое шире и выше станет не только сам контейнер, но и текст внутри него, это же касается и находящихся внутри изображений.

Конечно, можно удивиться, а зачем вообще все это нужно. В конце концов, можно ведь увеличить или уменьшить ширину и высоту элемента, используя свойства `width` и `height`, а изменить размер шрифта текста — с помощью свойства `font-size`. Чаще всего масштабирование используется для визуальных изменений элемента на странице в динамическом режиме. Например, помещение указателя мыши поверх кнопки может тут же привести к увеличению ее размера. Этого эффекта можно добиться в помощью состояния `:hover`.

Предположим, что на странице есть ссылка, к которой применен класс `.button`. Для форматирования этой ссылки в виде кнопки можно создать следующий стиль:

```
.button {
  font: .9em Arial, Helvetica, sans-serif;
  border-radius: .5em;
  background-color: rgb(34,255,23);
  border: 1px solid rgba(0,0,0,.5);
  padding: .5em;
}
```

Чтобы выделить эту кнопку, можно сделать ее немного больше при помещении на нее указателя мыши:

```
.button:hover {
  -webkit-transform: scale(1.2);
  -ms-transform: scale(1.2);
  transform: scale(1.2);
}
```

Когда указатель мыши покидает пределы кнопки, преобразование удаляется и кнопка возвращается к своему обычному размеру. Способ анимации этого изменения размеров с помощью преобразований с применением каскадных таблиц стилей будет рассмотрен в разделе «Переходы».

СОВЕТ

Подобный метод можно применить для изображений. Показать галерею из небольших изображений, а затем добавить к ним стиль :hover, чтобы при помещении на них указателя мыши они становились больше. Чтобы все это выглядело вполне достойно, в HTML-коде указывается финальная, увеличенная версия изображения, но его размер уменьшается либо с помощью каскадных таблиц стилей, либо с помощью свойств width и height элемента img.

Можно даже проводить горизонтальное и вертикальное масштабирование по отдельности. Для этого внутри скобок нужно указать два значения, разделенные запятой: первое число будет относиться к горизонтальному, а второе — к вертикальному масштабированию. Например, чтобы сделать элемент вдвое меньше по ширине, но вдвое выше, используется следующее объявление:

```
transform: scale(.5,2);
```

Результат можно увидеть на левом нижнем изображении на рис. 10.3.

В языке CSS также предоставлены отдельные функции для горизонтального и вертикального масштабирования: `scaleX` проводит масштабирование по горизонтальной оси, а `scaleY` — по вертикальной. Например, чтобы элемент стал вдвое выше без изменения его ширины, можно воспользоваться следующим объявлением:

```
transform: scaleY(2);
```

ПРИМЕЧАНИЕ

Для экономии бумаги в этом примере опять показана только лишь версия свойства transform, в которой вендорные префиксы не используются. Чтобы объявление работало в браузерах Safari и Internet Explorer 9, нужно добавить соответствующие префиксы -webkit- и -moz-.

Однако чтобы элемент стал в три с половиной раза шире, но не выше или ниже, следует воспользоваться таким объявлением:

```
transform: scaleX(3.5);
```

Есть еще один визуальный трюк, предлагаемый функцией масштабирования: возможность перевернуть элемент вокруг его вертикальной или горизонтальной центральной оси. Никто не знает достоверно, какой из разделов математики был использован Консорциумом W3C, чтобы придумать эту систему, но если для масштабирования применить отрицательное значение, то элемент будет перевернут. Например, чтобы перевернуть элемент вокруг его обеих центральных осей, нужно применить следующее объявление:

```
transform: scale(-1);
```

Получится изображение, показанное на рис. 10.4, *слева*. Можно также перевернуть элемент вокруг одной центральной оси. На правом изображении на рис. 10.4 изображение перевернуто только вокруг его горизонтальной центральной оси. Среднее изображение получено за счет переворота изображения вокруг его вертикальной оси:

```
transform: scale(-1,1);
```

Похоже на эффект отражения элемента. Весьма забавно!

Рис. 10.4. Чтобы запутать или обескуражить посетителей вашего сайта, в языке CSS есть масса разных способов. Левый квадрат перевернут вокруг горизонтальной центральной оси, средний — вокруг вертикальной, а правый — вокруг обеих осей

Перемещение

Функция `translate` свойства `transform` перемещает элемент из его текущей позиции на некоторое расстояние вправо/влево и вверх/вниз. Пользы от этой функции как таковой немного. Как выяснится далее, когда браузер перемещает элемент, он не перестраивает контент страницы, а отображает его так, как будто элемент никуда не перемещался. Соответственно, если с использованием функции `translate` перемещается контейнер `div` или другой элемент, браузер оставляет пустое пространство там, где этот элемент отобразился без преобразования, а затем визуализирует элемент в его новой позиции (рис. 10.5). Если нужно лишь разместить элемент на странице, можно воспользоваться абсолютным или относительным позиционированием (см. главу 14).

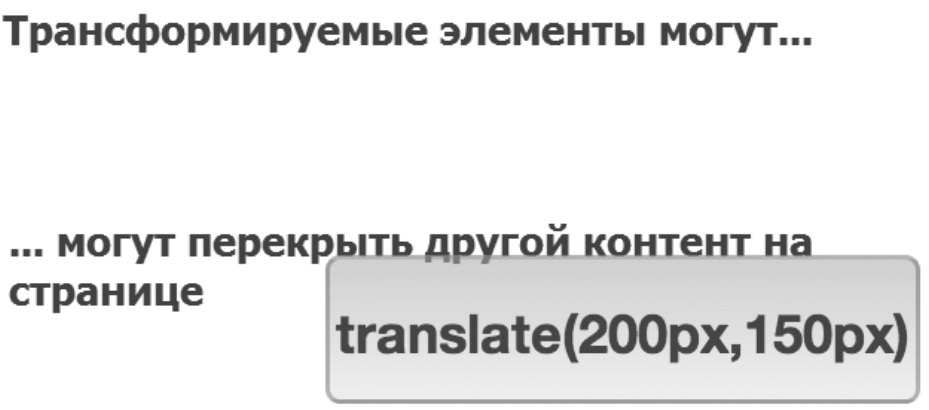

Рис. 10.5. Функция translate перемещает элемент на указанное количество пикселов, единиц em или процентов от его обычного положения на странице

Но перемещение пригодится, когда нужно изобразить небольшое движение в ответ на помещение поверх указателя мыши или на щелчок кнопкой мыши. Например, во многих конструкциях пользовательского интерфейса при нажатии кнопки она смещается немного вниз и влево, имитируя вид реальной объемной кнопки. Применительно к ссылке этот эффект можно сымитировать с помощью функции `translate` и состояния `:active`:

```
.button:active {
  -webkit-translate(1px,2px);
  -ms-translate(1px,2px);
  translate(1px,2px);
}
```

Функции `translate` передаются два значения: первое определяет величину горизонтального, а второе — вертикального перемещения. В данном примере щелчок на элементе с классом `.button` вызывает перемещение этого элемента на один пиксел вправо и на два пиксела вниз. Чтобы элемент переместился влево, нужно для первого значения использовать отрицательное число, применение отрицательного числа в качестве второго значения приведет к перемещению элемента вверх.

Но можно применять не только значения в пикселах. Допустимы любые значения, используемые в CSS для указания длины — px, em, % и т. д.

В CSS предоставляются также две дополнительные функции для перемещения элемента только влево или вправо — `translateX` и только вверх или вниз — `translateY`. Например, для перемещения элемента вверх на `.5em` используется функция `translateY`:

```
transform: translateY(-.5em);
```

Очень интересный эффект от применения функции `translate` можно получить при ее совместном использовании с переходами CSS. Это сочетание позволяет

анимировать перемещение элемента, продемонстрировав его путешествие по экрану. Вскоре вы увидите, как это делается.

Скашивание

Скашивание элемента можно осуществить по его горизонтальной и вертикальной осям: это придает элементу трехмерное представление (рис. 10.6). Например, для скашивания всех вертикальных линий влево на 45° (как на первом изображении на рис. 10.6) нужно написать следующий код:

```
transform: skew(45deg, 0);
```

Чтобы придать элементу такое же скашивание, но по оси *Y* (среднее изображение на рис. 10.6), нужно написать следующий код:

```
transform: skew(0,45deg);
```

Можно скосить элемент по двум осям одновременно. Например, следующий код приводит к результату, демонстрируемому третьим изображением, показанным на рис. 10.6:

```
transform: skew(25deg,10deg);
```

Первое значение задает значение угла в диапазоне от `0deg` до `360deg`, действующее в направлении против часовой стрелки от верхней части элемента. Например, на первом изображении (см. рис. 10.6) 45° представлены линией, нарисованной из центра элемента и повернутой на 45° против часовой стрелки (вниз и влево).

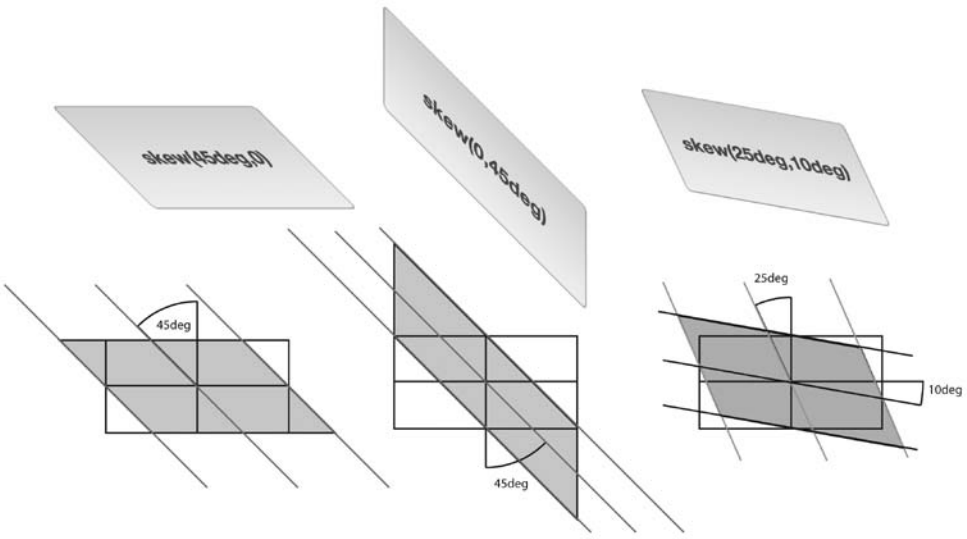

Рис. 10.6. Функция skew — один из способов имитации трехмерного отображения

ПРИМЕЧАНИЕ

Не забудьте в окончательном варианте CSS-кода добавить вендорные префиксы для различных браузеров: -webkit-transform и -ms-transform.

Вторым значением также является значение угла в диапазоне от `0deg` до `360deg`. Но этот угол действует в направлении по часовой стрелке с правой стороны элемента. Среднее изображение на рис. 10.6 является примером 45-градусного скашивания по всем горизонтальным линиям.

ПРИМЕЧАНИЕ

Чтобы получить визуальное представление о преобразованиях с помощью каскадных таблиц стилей, зайдите на сайт с интерактивным средством визуализации по адресу tinyurl.com/ydvlcc2.

Как и в случае с `translate` и `scale`, в CSS предлагаются отдельные функции для осей *X* и *Y*: `skewX` и `skewY`.

ПРИМЕЧАНИЕ

В CSS предлагается еще один метод преобразования под названием matrix. Он представляет собой числовой массив, похожий на тот, что используется в углубленном курсе алгебры. Этот метод не всегда понятен интуитивно и над ним придется поломать голову. Но если изучать принципы его работы не хочется, то наиболее понятное объяснение матриц преобразований можно найти на сайте tinyurl.com/qcj8jzj.

Множественные преобразования

Но вы не ограничены только одним преобразованием. Изображение можно одновременно масштабировать и скашивать, вращать и перемещать или использовать любые из четырех различных преобразований. Для этого нужно добавить через запятую дополнительные функции к свойству `transform`. Например, повернуть элемент на 45° и увеличить его размер вдвое можно с помощью такого объявления:

```
transform: rotate(45deg) scale(2);
```

А вот пример всех четырех преобразований, применяемых одновременно:

```
transform: skew(45deg,0deg) scale(.5) translate(400px,500px) rotate(90deg);
```

Браузер будет применять все эффекты в порядке следования функций. Например, во втором примере элемент сначала будет скошен, затем подвергнется масштабированию, после чего будет перемещен и наконец подвергнется вращению. Порядок не играет роли, если только не используется перемещение. Поскольку при перемещении изменяется местоположение элемента, то, если, к примеру, поместить перемещение перед вращением, браузер сначала переместит элемент, а затем применит к нему вращение. Поскольку сначала элемент перемещается, точка, вокруг которой он будет вращаться, изменится. С другой стороны, если сначала будет применено вращение, то затем элемент, уже подвергшийся вращению, будет перемещен на определенное расстояние относительно его центра (который теперь будет находиться в новой позиции).

Точка преобразования

Обычно, когда к элементу применяется преобразование, в качестве точки начала преобразования браузер использует центр элемента. Например, при вращении элемента браузер поворачивает его вокруг центральной точки (рис. 10.7, *слева*).

Но в CSS разрешается изменять точку преобразования, используя свойство `transform-origin`. Это свойство работает точно так же, как и ранее рассмотренное свойство `background-position`; в качестве значения можно указывать ключевые слова, абсолютные и относительные значения в единицах em и процентах.

Например, чтобы повернуть контейнер `div` вокруг его верхней левой точки (см. рис. 10.7, *посередине*), можно воспользоваться ключевыми словами `left` и `top`:

```
transform-origin: left top;
```

Можно также использовать пиксельные значения:

```
transform-origin: 0 0;
```

или проценты:

```
transform-origin: 0% 0%;
```

Точно так же для вращения элемента вокруг его нижнего правого угла (см. рис. 10.7, *справа*) используются ключевые слова `right` и `bottom`:

```
transform-origin: right bottom;
```

что также эквивалентно следующему объявлению:

```
transform-origin: 100% 100%;
```

При использовании пикселов, единиц em или процентных значений первое число означает горизонтальную, а второе — вертикальную позицию.

ПРИМЕЧАНИЕ

Свойство transform-origin не влияет на элементы, которые подвергаются только перемещению с помощью функции translate.

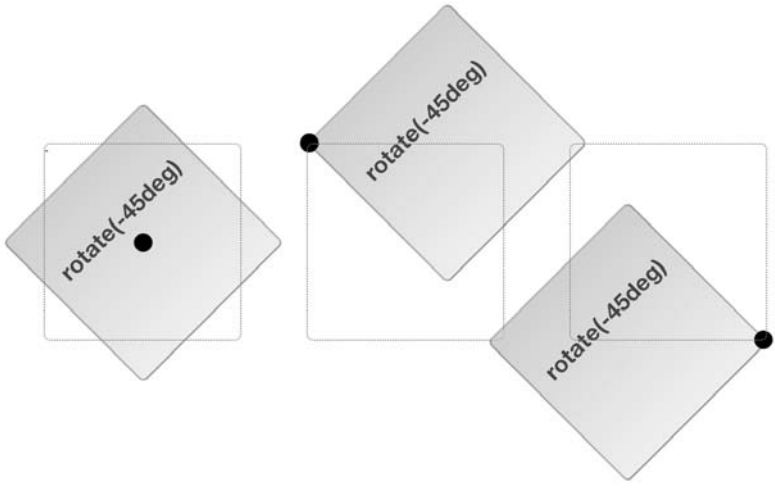

Рис. 10.7. Присваивание значения свойству transform-origin изменяет точку элемента, в которой будет применено преобразование. Пунктирными квадратами обозначен элемент, каким бы он был без вращения (его обычная позиция на странице)

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Трехмерные преобразования

В языке CSS предлагается также более сложный тип преобразования. Трехмерные преобразования позволяют имитировать трехмерное пространство на плоском экране монитора, планшетного компьютера или смартфона.

Краткое введение в трехмерные преобразования дано по адресу tinyurl.com/oqyfcpz, а более детальное описание и несколько примеров — по адресу tinyurl.com/d68z7ud. Посмотреть удачные примеры трехмерных преобразований в действии можно на следующих сайтах:

- один из первых примеров демонстрации возможностей трехмерных преобразований — страница Morphing Power Cubes (tinyurl.com/3n9qr64) — здесь вы увидите вращающийся куб, который может превращаться во вращающийся набор плиток;
- еще один пример эффекта трехмерного транспонирования (эффект выглядит так, будто вы переворачиваете игральную карту, чтобы увидеть ее рубашку) находится по адресу tinyurl.com/oajfynt.

Переходы

Преобразованиями, конечно, можно позабавить посетителей (особенно функцией `rotate`), но по-настоящему они оживят вашу страницу в сочетании с переходами CSS. *Переход* представляет собой анимацию смены одного набора свойств CSS другим за определенный промежуток времени. Например, можно задать вращение баннера на 360° в течение двух секунд. Чтобы выполнить переход, потребуется следующее.

- **Два стиля.** Один стиль должен представлять начальный вид элемента, например красную кнопку, а второй — его конечный вид — синюю кнопку. О процессе анимации изменения между двумя состояниями позаботится браузер (например, об изменении цвета кнопки с красного на синий).
- **Свойство transition.** С помощью CSS добавляется свойство `transition` — секретная приправа, позволяющая осуществить анимацию. Как правило, свойство `transition` применяется к исходному стилю, который определяет внешний вид элемента до начала анимации.
- **Инициатор.** Инициатор представляет собой действие, вызывающее переход от одного стиля к другому. В CSS для запуска анимации можно применять несколько псевдоклассов. Наиболее часто используется псевдокласс `:hover`. С его помощью можно анимировать изменение обычного вида элемента на тот вид, когда посетитель устанавливает поверх элемента указатель мыши. Как это делается, вы увидите совсем скоро. Кроме того, можно использовать такие псевдоклассы, как `:active` (связанный с щелчком кнопкой мыши на элементе), `:target` (связанный с элементом, ставшим целью перехода по ссылке) и `:focus` (связанный с переходом по ссылке с помощью клавиши Tab или щелчком на элементе формы или с переходом на этот элемент с помощью клавиши Tab). Кроме этого, для динамической смены стиля любого элемента можно воспользоваться JavaScript-кодом (см. врезку «Использование JavaScript-сценариев для запуска переходов» далее).

Когда инициатор больше не применяется, то есть когда, к примеру, посетитель убирает указатель мыши с кнопки навигации, браузер возвращает элементу его прежний стиль и анимирует весь этот процесс. Иными словами, вам нужно лишь установить переход для элемента один раз, а анимацию перехода от одного стиля к другому и обратно к исходному стилю браузер возьмет на себя.

Браузер не может анимировать каждое свойство CSS, но у вас в распоряжении остается все же весьма длинный список свойств, на которых можно остановить свой выбор. Кроме таких преобразований, как `rotate`, `scale`, `translate` и `skew`, о которых только что шла речь, можно анимировать такие свойства, как `color`, `background-color`, `border-color`, `border-width`, `font-size`, `height`, `width`, `letter-spacing`, `line-height`, `margin`, `opacity`, `padding`, `word-spacing`, свойства позиционирования — `top`, `left`, `right` и `bottom`, речь о которых пойдет в главе 15, а также многие другие свойства. Их полный список можно найти по адресу tinyurl.com/dxjbdhd.

ПРИМЕЧАНИЕ

Переходы CSS работают в большинстве браузеров. Но когда речь заходит об Internet Explorer, приходится заметить, что переходы этот браузер поддерживает, только начиная с версии 10. Если вы все еще хотите поддерживать более ранние версии браузера Internet Explorer, используйте переходы только как средство придания выразительности. При этом в большинстве случаев все будет выглядеть вполне приемлемо: Internet Explorer 9 и более ранние версии смогут обеспечить переключение между двумя стилями (например, показать стиль :hover) без анимации данного изменения.

Добавление перехода

В основе переходов CSS лежат четыре свойства, который управляют тем, какие свойства анимировать, сколько времени займет анимация, какой тип анимации будет использован и какой будет необязательная отсрочка перед началом анимации. Рассмотрим простой пример. Предположим, что нужно анимировать изменение фонового цвета кнопки навигации с оранжевого на синий при помещении поверх кнопки указателя мыши. Начать нужно с двух стилей, необходимых для переключения между этими двумя цветами. Можно, к примеру, применить к ссылке класс `.navButton`, а затем создать следующие два стиля:

```
.navButton {
  background-color: orange;
}

.navButton:hover {
  background-color: blue;
}
```

Эти стили будут работать в любом браузере. При помещении указателя мыши поверх кнопки навигации фоновый цвет этой кнопки изменится с оранжевого на синий. Но это изменение произойдет мгновенно. Чтобы цвет изменился в режиме анимации за одну секунду, нужно к стилю `.navButton` добавить два новых свойства:

```
.navButton {
  background-color: orange;
  transition-property: background-color;
  transition-duration: 1s;
}

 .navButton:hover {
  background-color: blue;
}
```

Свойство `transition-property` указывает на анимируемое свойство. Можно указать одно свойство (как в показанном выше примере), воспользоваться ключевым словом `all` для анимации всех изменяемых CSS-свойств или воспользоваться списком с запятой в качестве разделителя для указания более чем одного свойства (но не всех свойств). Предположим, что вы создаете стиль `:hover`, чтобы изменялись сразу цвет текста, фоновый цвет и цвет границы. Все эти три свойства указываются в виде следующего списка:

```
transition-property: color, background-color, border-color;
```

или, чтобы упростить код, указывается ключевое слово `all`:

```
transition-property: all;
```

Во многих случаях при анимации каждого изменения средствами каскадных таблиц стилей лучше использовать ключевое слово `all`, в результате чего создается привлекательный визуальный эффект.

Чтобы указать длительность анимации, применяется свойство `transition-duration`. Ему передается значение либо в секундах, либо в миллисекундах (тысячных долях секунды). Например, чтобы переход занимал полсекунды, можно задать либо код:

```
transition-duration: .5s;
```

либо код:

```
transition-duration: 500ms;
```

Можно даже указать длительность отдельно для каждого анимируемого свойства. Например, когда посетитель устанавливает указатель мыши поверх кнопки, может потребоваться, чтобы цвет текста изменялся быстрее, цвет фона изменялся немного медленнее, а цвет границы изменялся заметно медленнее. Для этого нужно перечислить анимируемые свойства, используя `transition`, а затем указать значения длительности в свойстве `transition-duration`:

```
transition-property: color, background-color, border-color;
transition-duration: .25s, .75s, 2s;
```

Порядок перечисления значений длительности должен соответствовать порядку перечисления свойств. Следовательно, в показанном выше примере `.25s` относится к свойству `color`, `.75s` — к свойству `background-color`, а `2s` — к свойству `border-color`. Если поменять свойства местами, их значения длительности переходов изменятся.

Тайминг перехода

Чтобы анимированный переход заработал, нужно только установить значения для свойств `transition-property` и `transition-duration`. Но с помощью свойства `transition-timing-function` можно также контролировать и скорость анимации. В предназначении этого свойства нетрудно и запутаться: оно управляет не длительностью анимации (для этого есть свойство `transition-duration`), а ее *скоростью*.

Свойство `transition-timing-function` может использовать одно из пяти ключевых слов: `linear`, `ease`, `ease-in`, `ease-out` и `ease-in-out`. Если функцию управления скоростью не задавать, браузер будет применять метод `ease`, при котором анимация начинается медленно, ускоряется к середине и замедляется к концу, предоставляя более естественное изменение. Вариант `linear` предоставляет равномерное изменение на протяжении всего периода анимации. Ни один из вариантов не имеет заметных преимуществ перед другими, они лишь предлагают различную общую картину, и для определения своих предпочтений нужно просмотреть все варианты.

Чтобы воспользоваться этим свойством, нужно добавить свойство `transition-timing-function` и требуемый метод:

```
transition-timing-function: ease-in-out;
```

ПРИМЕЧАНИЕ

В случае с различными функциями регулирования скорости перехода лучше один раз увидеть, чем сто раз услышать. Посетите сайт tinyurl.com/36edxvd, чтобы увидеть наглядное сравнение пяти методов тайминга.

Для свойства `transition-timing-function` можно также использовать значение так называемой кубической кривой Безье (`cubic-bezier`). Она определяет график воспроизведения анимации по времени (рис. 10.8). Если вам приходилось работать с такими графическими редакторами, как Adobe Illustrator, то вы, наверное, знакомы с кривыми Безье. Кривизной линии можно управлять путем настройки двух контрольных точек: чем круче линия, тем быстрее анимация, а более пологая линия означает более медленное воспроизведение анимации. Например, кривая Безье, показанная на рис. 10.8, начинается с крутого участка (анимация начинается в быстром темпе), затем переходит к середине в более пологую фазу (анимация замедляется), а потом кривизна снова возрастает (анимация быстро подходит к своей завершающей стадии). Для создания анимации такого профиля нужно добавить следующую строку кода:

```
transition-timing-function: cubic-bezier(.20, .96, .74, .07);
```

Вам совершенно не обязательно забивать себе голову кубическими кривыми Безье. Лучше воспользоваться одним из многочисленных интерактивных средств для создания и тестирования различных функций тайминга. Одним из лучших считается средство Ceasar, разработанное Мэтью Лейном: tinyurl.com/4vebxwg. Интерактивное средство Caesar превращает создание кубических кривых Безье в сущий пустяк: нужно для изменения кривизны перетаскивать нижнюю левую и верхнюю правую контрольные точки. Чем круче линия, тем быстрее воспроизводится анимация на ее участке.

Как и в случае со свойством `transition-duration`, к различным свойствам можно применять разные тайминги.

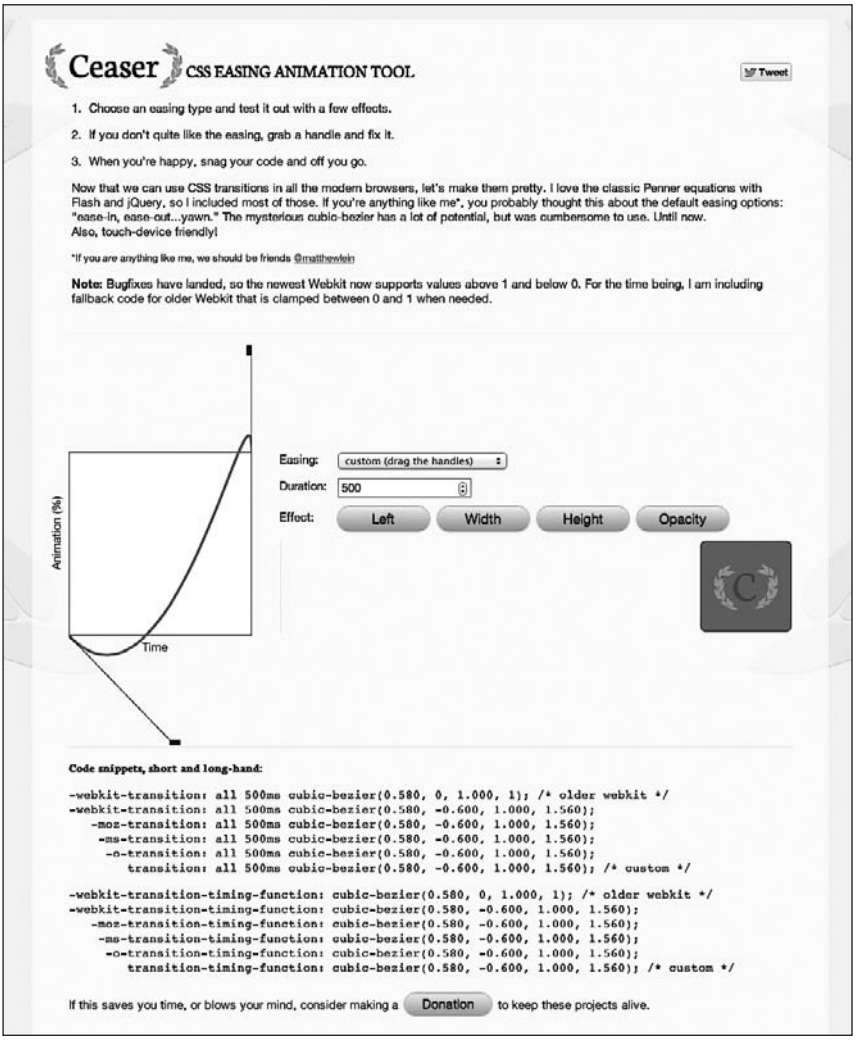

Рис. 10.8. Создание кубической кривой Безье позволяет управлять широким диапазоном интересных функций тайминга анимации. Можно задать очень медленное начало и быстрое завершение или наоборот

Отсрочка перехода

И наконец, можно оттянуть время начала анимации перехода, воспользовавшись свойством `transition-delay`. Например, если следует подождать полсекунды, прежде чем начать анимацию, можно добавить следующий код:

```
transition-delay: .5s;
```

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Использование JavaScript-сценариев для запуска переходов

Переходы CSS — простые в применении средства анимации, встроенные непосредственно в браузеры (по крайней мере в *большинстве* браузеров). Сложные эффекты можно создавать, определяя для этого набор исходных и конечных свойств CSS, возлагая все остальное на браузер. К сожалению, для запуска этих переходов у вас в распоряжении имеется всего лишь несколько селекторов. Чаще всего для изменения элемента при нахождении поверх него указателя мыши используется псевдокласс `:hover`. Для анимации элемента формы при щелчке на нем кнопкой мыши можно также воспользоваться псевдоклассом `:focus` (например, можно изменить высоту текстовой области при получении фокуса, а затем сжиматься до нескольких строк при щелчке за пределами этой области).

Но в языке CSS нет, к примеру, псевдокласса `:click`, поэтому при желании запустить переход по щелчку на элементе сделать это средствами каскадных таблиц стилей не получится. Не удастся также запустить анимацию какого-нибудь элемента при нахождении указателя мыши поверх другого элемента… …если использовать только CSS. Но переходы CSS работают при любом изменении свойства CSS, даже если это делается с помощью JavaScript.

JavaScript — язык программирования, позволяющий добавлять интерактивность веб-страницам, создавать динамические пользовательские интерфейсы и осуществлять множество других полезных действий. Но JavaScript-код можно также использовать для простого добавления или удаления класса элемента или изменения какого угодно свойства CSS. Благодаря JavaScript можно добавить класс к изображению, когда посетитель щелкает на ссылке Показать увеличенный вариант изображения. Этот новый класс просто увеличивает масштаб изображения (с помощью свойства `scale`). Добавляя к изображению переход, вы тут же получите анимацию.

Изучение JavaScript — довольно обширная тема, заслуживающая отдельной книги, которой как раз может стать «JavaScript и jQuery. Исчерпывающее руководство» этого же автора. Но если для начала вы захотите освоить простейший способ использования переходов CSS, запускаемых с помощью JavaScript, прочитайте краткое руководство на сайте tinyurl.com/qb9mnv7.

Чаще всего отсрочка перехода для всех свойств вам не понадобится, поскольку в этом случае снижается эффект интерактивности. В конце концов, преднамеренно заставлять посетителей ждать целую секунду, пока они увидят изменения в кнопке при наведении на нее указателя мыши, не очень хорошая идея. Но если анимируется сразу несколько свойств, может понадобиться выждать с изменением одного свойства, пока не закончится анимация перехода других свойств. Предположим, что у вас есть кнопка, чей цвет фона и текста нужно изменить, а затем неожиданно поменять цвет ее границы после завершения изменения первых двух свойств. Это можно сделать с помощью следующего кода:

```
.navButton {
  color: black;
  background-color: #FF6603;
  border: 5px solid #660034;
  transition-property: color, background-color, border-color;
  transition-duration: 1s, 1s, .5s;
  transition-delay: 0, 0, 1s;
}
.navButton:hover {
  color: white;
```

```
  background-color: #660034;
  border-color: #FF6603;
}
```

Как и в случае применения свойства `transition-duration`, для каждого свойства можно указать свое время отсрочки. Порядок перечисления показателей времени должен соответствовать порядку перечисления этих свойств для `transition-property`. Например, в предыдущем коде отсрочки для переходов цвета текста и цвета фона отсутствуют, но зато есть односекундная отсрочка до начала изменения цвета границы.

СОВЕТ

Обычно свойства перехода помещаются в начальный (.navButtonon в предыдущем примере), а не в конечный стиль (.navButton:hover). Но при использовании раскрывающегося меню CSS возникают некоторые сложности. Проблема в том, что при задании таких меню с помощью CSS они слишком быстро пропадают при случайном выходе указателя мыши за их пределы. Но с помощью свойства transition-delay можно заставить меню быстро появляться и медленно исчезать. Для этого добавьте в исходный стиль следующий код:

transition-delay: 1s;

Затем добавьте отсрочку к стилю :hover:

transition-delay: 0;

Каким бы странным этот код ни показался, он заставляет переход :hover происходить немедленно, без отсрочки. А вот возвращение к обычному стилю (при котором меню исчезает) занимает одну секунду. За это время посетитель успеет вернуть указатель мыши обратно на меню, пока оно не исчезло.

Сокращенная запись свойства transition

Запись всех свойств по отдельности — `transition-property`, `transition-duration`, `transition-timing-function` и `transition-delay` — может утомить. Особенно если потребуется создать и версию каждого из них с вендорным префиксом. На этот случай есть более быстрый способ задания переходов — свойство `transition`. Оно связывает все другие свойства. Нужно лишь перечислить через запятые свойство, длительность, тайминг и отсрочку. Например, для анимации всех свойств CSS на одну секунду, используя функцию `ease-in` и полусекундную отсрочку, нужно написать следующий код:

```
transition: all 1s ease-in .5s;
```

Список должен состоять из ключевого слова `all` или какого-нибудь определенного свойства CSS, а также из длительности, а тайминг и отсрочку можно не указывать. По умолчанию в качестве функции тайминга указывается `ease-in`, а отсрочка не используется. То есть, если нужно просто анимировать переход всех свойств CSS на одну секунду, нужно написать следующий код:

```
transition: all 1s;
```

Если нужно анимировать изменение фонового цвета, включите в список именно это свойство:

```
transition: background-color 1s;
```

Здесь можно включить в список только одно свойство CSS, поэтому, если требуется анимировать сразу несколько свойств CSS (но не все), можно воспользоваться их списком с запятой в качестве разделителя, где элементами будут разделенные пробелами свойства переходов. Возьмем один из предыдущих примеров, где свойство `border-color` анимировалось отдельно от цвета текста и цвета фона. Его код можно переписать следующим образом:

```
transition: color 1s, background-color 1s, border-color .5s 1s;
```

Чтобы код было легче читать, многие веб-разработчики помещают каждый переход в отдельной строке:

```
transition: color 1s,
background-color 1s,
border-color .5s 1s;
```

Нужно только не забывать разделять переходы запятыми, а всю связку завершать точкой с запятой.

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

Создание плавной анимации

Добавление анимации к различным свойствам может замедлить работу браузера. Используете ли вы переходы или анимацию средствами каскадных таблиц стилей, браузерам необходимо выполнить много вычислений, чтобы анимировать изменения в свойствах CSS. Слишком много анимации и переходов, использующихся одновременно, могут существенно снизить скорость работы браузера или даже привести к его зависанию. Особенно это касается мобильных устройств и планшетов, процессоры которых гораздо менее производительны, чем в настольных компьютерах и ноутбуках.

Тем не менее существует четыре параметра, которые браузеры могут анимировать, не прибегая к значительным вычислительным затратам процессора: непрозрачность (см. приложение 1), а также параметры `translate`, `scale` и `rotate` свойства `transform` (см. предыдущий раздел). Эти четыре свойства обрабатываются более эффективно, чем другие, так что любые переходы или анимация, созданные с их помощью, будут осуществляться плавно (чтобы узнать технические подробности, перейдите по ссылке tinyurl.com/pjdx8le).

Кроме того, для ускорения расчетов можно переложить функции обработки анимации на *графический процессор (GPU)* (иными словами, видеокарту). Графические процессоры — высокопроизводительные устройства, которые могут выполнять определенные типы вычислений гораздо быстрее, чем центральный процессор компьютера. Вы можете инструктировать браузер перенаправить расчет стиля в графический процессор, добавив в него свойство трехмерного преобразования. Например, если вы планируете анимировать цвет фона элемента, когда посетитель сайта установит поверх него указатель мыши, можно создать стиль с оригинальным цветом фона, который будет выглядеть следующим образом:

```
. highlight {
  background-color: rgb(231,0,23);
  transition: background-color 1s;
  transform: translateZ(0);
}
.highlight:hover {
  background-color: rgb(0,0,255);
}
```

Строка `transform: translateZ(0)` не вносит никаких визуальных изменений. Она инструктирует браузер переместить элемент на 0 пикселов вдоль оси *Z*; иными словами, элемент не двигается вообще. Однако, поскольку он использует трехмерное преобразование, браузер обрабатывает этот стиль с помощью графического процессора. Суть заключается в том, что анимация

ИНФОРМАЦИЯ ДЛЯ ОПЫТНЫХ ПОЛЬЗОВАТЕЛЕЙ

может воспроизводиться более плавно, потому что графический процессор более производительный.

Но прежде, чем вы начнете применять свойство `transform: translateZ(0)` к каждому стилю, необходимо учесть, что графический процессор может обрабатывать много команд, однако передача большого количества визуальных эффектов может замедлить браузер.

Кроме того, большое количество анимации и переходов может оказывать негативное влияние на производительность страницы, особенно на мобильных устройствах пользователей. Более тщательно проверьте все переходы и эффекты анимации в различных браузерах и на мобильных устройствах, чтобы убедиться, что ваш сайт работает должным образом.

Анимация

В CSS предоставляется еще один, более богатый на свойства, механизм создания анимации. С помощью CSS-переходов можно анимировать только переход от одного набора свойств CSS к другому. Технология *анимации*, предусмотренная в CSS, позволяет анимировать переходы от одного набора свойств к другому, затем к третьему и т. д. Кроме того, можно задать повторяющуюся анимацию, приостановить ее выполнение при помещении указателя мыши поверх элемента и даже повернуть ее вспять, когда анимация завершится.

Анимация CSS сложнее переходов, но у нее есть дополнительное преимущество в том, что не нужен обязательный инициатор. Наряду с тем, что анимацию можно добавить к состоянию `:hover` для ее проигрывания при нахождении указателя мыши поверх элемента, запустить анимацию можно и при загрузке страницы. Тем самым можно привлечь внимание к баннеру или логотипу, анимируемому, когда посетитель заходит на сайт.

ПРИМЕЧАНИЕ

Анимация CSS, как и переходы, не поддерживается в Internet Explorer 9 и более ранних версиях. Но эти эффекты поддерживаются большинством других современных браузеров.

При создании анимации первым делом создаются ключевые кадры. *Ключевой кадр* в анимации — это отдельный кадр анимации, определяющий внешний вид сцены. Предположим, первый ключевой кадр содержит изображение мяча на одной половине футбольного поля. Добавив второй ключевой кадр, можно определить конечную точку анимации, например мяч в воротах на другой половине поля. После этого браузер создаст анимацию между этими двумя ключевыми кадрами, прорисовывая все промежуточные фазы, отображающие перемещение мяча по полю на его пути к голу в воротах.

Ели вы подумали, что в переходы заложена такая же идея, то так оно и есть. В переходе определяются два стиля, а на браузер возлагается задача анимировать изменения от одного стиля к другому. В этом смысле каждый из этих стилей можно представить в качестве ключевых кадров. Но анимация CSS позволяет определять *множество* ключевых кадров и создавать более сложные эффекты: например, футбольный мяч, перемещающийся с одной стороны поля к игроку, к другому игроку, а затем в ворота.

Создание анимации проходит в два приема.

1. Определение анимации.

   Включает настройку ключевых кадров со списком анимируемых CSS-свойств.

2. Применение анимации к элементу.

   После определения анимации ее можно применить к любому количеству элементов страницы. Можно даже задать для каждого элемента разные тайминги, отсрочки и другие свойства анимации. То есть одну и ту же анимацию на странице можно использовать с разными настройками несколько раз.

Определение ключевых кадров

Первым делом нужно настроить ключевые кадры. Реализующий их синтаксис может показаться немного странным, но его основная структура имеет следующий вид:

```
@keyframes имяАнимации {
  from {
    /* здесь перечисляются свойства CSS */
  }

  to {
    /* здесь перечисляются свойства CSS */
  }
}
```

Сначала указывается ключевое слово `@keyframes`, за которым следует имя — название анимации. В дальнейшем это имя будет использоваться при применении анимации к элементу страницы, поэтому оно должно быть описательным, например `fadeOut` или `fadeIn`.

ПРИМЕЧАНИЕ

Само понятие @keyframes является не свойством CSS, а так называемым правилом. В CSS есть и другие правила, например инструкция @import для загрузки внешней таблицы стилей из другой таблицы стилей и @media — для определения стилей для различных типов аппаратуры представления информации, к примеру принтера или устройств с экранами различного размера и разрешением.

Затем добавляются не менее двух ключевых кадров. В данном примере слова `from` и `to` используются для создания начального ключевого кадра (`from`) и конечного ключевого кадра (`to`). Внутри каждого ключевого кадра находится одно или несколько свойств CSS, словно создается стиль. Фактически каждый ключевой кадр можно представить в виде простого стиля, содержащего одно или несколько свойств каскадных таблиц стилей. Предположим, что нужно создать анимацию постепенного появления элемента. Можно начать со значения `opacity`, равного `0` (невидимый), и закончить значением этого свойства, равным `1` (полностью видимый):

```
@keyframes fadeIn{
  from {
    opacity: 0;
  }
```

```
  to {
    opacity: 1;
  }
}
```

Но вы не ограничены использованием только двух ключевых кадров. С помощью процентных значений можно определить несколько ключевых кадров. Процентное значение определяет момент общей длительности анимации, в который должно произойти изменение. Предположим, что нужно создать эффект изменения фона элемента с желтого на синий, а затем на красный. Для этого можно написать следующий код:

```
@keyframes backgroundGlow {
  from {
    background-color: yellow;
  }
  50% {
    background-color: blue;
  }
  to {
    background-color: red;
  }
}
```

В данном случае синий фон появится в середине анимации. Если нужно, чтобы желтый цвет присутствовал в фоне дольше, а синим он становился после трех четвертей времени анимации, используйте значение не `50%`, а `75%`. Таким же образом можно продолжать добавлять дополнительные ключевые кадры (например, на `25%`, `66%` и т. д.).

ПРИМЕЧАНИЕ

Ключевое слово from можно заменить значением 0%, а ключевое слово to — значением 100%.

Но одним свойством каскадных таблиц стилей можно не ограничиваться. Внутри ключевого кадра может находиться любое количество пригодных к анимации свойств — `background-color`, `opacity`, `width`, `height` и т. д.:

```
@keyframes growAndGlow {
  from {
    background-color: yellow;
  }

  50% {
    transform: scale(1.5);
    background-color: blue;
  }

  to {
    transform: scale(3);
    background-color: red;
  }
}
```

В этом примере наряду с троекратным изменением цвета фона происходит увеличение масштаба элемента, размеры которого в итоге становятся втрое больше первоначальных.

Можно также усложнить использование процентных значений, добавив несколько таких значений к одному набору свойств CSS. Это может пригодиться в двух случаях: во-первых, при необходимости довести анимацию до некоторого момента, сделать паузу, а затем продолжить анимацию. Предположим, что вам нужно, чтобы сначала у элемента `div` был желтый цвет фона. Затем нужно, чтобы цвет сменился на синий, некоторое время оставался синим, а затем сменился на красный. То есть нужна пауза в середине анимации, при которой цвет не меняется, а затем нужна новая смена цвета. Эта задача решается с помощью следующего кода:

```
@keyframes glow {
  from {
    background-color: yellow;
  }
  25%, 75% {
    background-color: blue;
  }
  to {
    background-color: red;
  }
}
```

Обратите внимание на процентные значения `25%`, `75%` в строке 5. Они означают, что на 25 % от длительности всей анимации фоновый цвет элемента станет синим. Но он будет оставаться синим до момента 75 % длительности анимации. То есть от отметки 25 % до отметки 75 % фон будет сохранять синий цвет, перед тем как к окончанию анимации превратиться в красный. Если длительность этой анимации будет четыре секунды, то в средние две секунды у элемента будет синий фон.

Процентное значение можно также указывать при необходимости использования одного и того же набора свойств CSS для различных частей анимации. Предположим, нужно выполнить еще одну анимацию фонового цвета, но на этот раз менять цвет от желтого к синему, затем к оранжевому, после чего к синему, к оранжевому и к красному. Синий и оранжевый цвета фигурируют в этом перечне дважды, поэтому вместо многократного упоминания об этих свойствах фонового цвета можно воспользоваться следующим кодом:

```
@keyframes glow {
  from {
    background-color: yellow;
  }
  20%, 60% {
    background-color: blue;
  }
  40%, 80% {
    background-color: orange;
  }
  to {
```

```
    background-color: red;
  }
}
```

В данном случае фоновый цвет будет синим на отметке 20 %, превратится в оранжевый на отметке 40 %, затем опять в синий на отметке 60 % и, перед тем как стать в конце анимации красным, еще раз на отметке 80 % превратится в оранжевый.

К сожалению, как было сказано ранее, для использования анимации в браузерах Chrome, Safari и Opera вы должны указывать вендорные префиксы производителей. Другими словами, вам нужно создать одну анимацию для этих браузеров и другую (без вендорных префиксов) — для браузеров Firefox и Internet Explorer 10, для чего придется написать следующий код:

```
@-webkit-keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
@keyframes fadeIn{
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

Обратите внимание на два дефиса: один между символом `@` и вендорным префиксом и второй — между вендорным префиксом и словом `keyframes`.

Применение анимации

Когда набор ключевых кадров определится, анимация будет готова. Чтобы заставить ее работать, нужно применить ее к какому-нибудь элементу страницы. Анимацию можно добавить к любому стилю любого элемента страницы. Если добавить анимацию к стилю, который сразу применяется к элементу, например к стилю элемента `h1`, то она будет активизирована при загрузке страницы. Этот прием можно применить для добавления на страницу вводной анимации, которая увеличивает масштаб логотипа, помещая его в верхний левый угол страницы, или подсвечивает некий блок информации, привлекая к нему внимание.

Кроме того, анимацию можно применить к одному из псевдоклассов, включая `:hover`, `:active`, `:target` или `:focus`, чтобы, к примеру, запустить анимацию при установке указателя мыши над ссылкой или при щелчке на элементе формы. И наконец, анимацию можно применить к классу и воспользоваться JavaScript-кодом для динамического применения этого класса в ответ на нажатие посетителем кнопки или щелчок на каком-нибудь другом элементе страницы.

В CSS предоставляется несколько свойств, связанных с анимацией, позволяющих управлять способом и временем проигрывания анимации (а также сокращенная запись, охватывающая все отдельные свойства). Как минимум, чтобы заставить анимацию выполняться, нужно указать имя, которое было присвоено исходной анимации (в правиле `@keyframes`), и длительность анимации.

Рассмотрим простой пример. Предположим, что нужно, чтобы при загрузке страницы плавно появился контейнер `div` с важным объявлением. Этому контейнеру был назначен класс с именем `announcement`:

```
<div class="announcement">
```

1. Создайте анимацию постепенного появления, задав правило `@keyframes`:

```
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

СОВЕТ

Когда анимируется всего лишь одно свойство, его будет проще прочитать, если поместить код в одной строке:

```
from { opacity: 0; }
```

Но если анимируется множество свойств, то читать (и набирать) проще будет при распределении кода по нескольким отдельным строкам:

```
from {
  opacity: 0;
  color: red;
  width: 50%;
}
```

2. Примените эту анимацию к стилю, предназначенному для элемента `div`:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
}
```

Свойство `animation-name` сообщает браузеру, какую анимацию нужно применить. Ему указывается то имя, которое предоставлялось анимации при выполнении шага 1. Свойство `animation-duration` устанавливает время, которое займет анимация от старта до финиша. В данном примере перечисляются лишь два свойства анимации, но вы можете (и, наверное, захотите) поместить в стиль и другие, не связанные с анимацией свойства. Например, в реальности к нашему стилю `.announcement` могут быть добавлены такие свойства, как `width`, `background-color`, `border` и т. д.

ПРИМЕЧАНИЕ

С технической точки зрения помещать имя анимации в кавычки ('fadeIn') не требуется, и в этом примере так не делается, но добавление кавычек поможет избежать конфликтов, возникающих при использовании в качестве имени анимации какого-нибудь ключевого слова CSS.

Как и в случае использования правила `@keyframes`, каждое из свойств анимации требует указания вендорных префиксов, поэтому показанный выше стиль `.announcement` для работы в максимально возможном количестве браузеров нужно переписать следующим образом:

```
.announcement {
  -webkit-animation-name: fadeIn;
  -webkit-animation-duration: 1s;
  animation-name: fadeIn;
  animation-duration: 1s;
}
```

Возможно, покажется несколько неудобным определять анимацию с помощью правила `@keyframes` в одном месте, а затем применять ее в другом месте (в стиле), но после определения анимация может применяться несколько раз в любом количестве стилей. Например, можно создать общий тип анимации появления элемента на экране и применить этот тип к различным элементам. Более того, можно управлять анимацией для каждого стиля совершенно независимо, например, заголовок может быть виден в течение половины секунды, а другие элементы страницы — в течение пяти секунд.

Кроме того, к одному и тому же элементу можно применить более одной анимации. Предположим, вы создаете одну анимацию с именем `fadeIn`, чтобы анимировать появление элемента, и еще одну анимацию, `blink`, чтобы получить мигающий цвет фона. Чтобы применить к элементу обе анимации, нужно предоставить список имен с запятой в качестве разделителя:

```
animation-name: fadeIn, blink;
```

Чтобы задать этим анимациям разные длительности воспроизведения, нужно предоставить список данных параметров с запятой в качестве разделителя:

```
animation-name: fadeIn, blink;
animation-duration: 1s, 3s;
```

Порядок применения параметров времени совпадает с порядком указания имен анимаций. Например, первая анимация получает время, указанное в списке первым. В показанном выше примере длительность `fadeIn` составляет одну секунду, а длительность `blink` — три секунды.

Можно также применить ряд других полезных свойств анимации, рассматриваемых далее.

Тайминг анимации

Как уже говорилось, свойство `animation-duration` позволяет управлять длительностью анимации. Как и в случае с переходами, для задания длительности можно использовать миллисекунды (например, `750ms`) или секунды (например, `.75s`).

Как и в случае с переходами, можно указывать функцию тайминга, которая управляет этой скоростью в заданном моменте анимации. Например, анимацию можно начинать медленно и завершать быстро, используя для этого кубическую кривую Безье или одно из предустановленных ключевых слов, определяющих ме-

тод: `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`. Все это работает точно так же, как и с переходами.

Свойство `animation-timing-function` можно использовать для управления всей анимацией или только для конкретных ключевых кадров. Например, чтобы применить функцию `ease-out` для представленной ранее анимации `fadeIn` (шаг 1 в предыдущем разделе), добавьте функцию тайминга к стилю `.announcement` (шаг 2 в предыдущем разделе):

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
  animation-timing-function: ease-out;
}
```

Но можно также управлять функцией тайминга для анимации между ключевыми кадрами. Предположим, что создается анимация с тремя ключевыми кадрами с тремя различными цветами фона. Браузер будет осуществлять ее при превращении одного цвета в другой, а затем в третий. Возможно, вам захочется замедлить превращение первого цвета во второй с помощью кубической кривой Безье, а затем равномерно продолжить анимацию от середины и до ее окончания. Это можно сделать, добавив две функции тайминга: одну для первого ключевого кадра (она будет управлять анимацией от ключевого кадра 1 к ключевому кадру 2) и вторую — для второго ключевого кадра (для управления анимацией от второго ключевого кадра к третьему):

```
@keyframes growAndGlow {
  from {
    background-color: yellow;
    animation-timing-function: cubic-bezier(1, .03, 1, .115);
  }
  50% {
    transform: scale(1.5);
    background-color: blue;
    animation-timing-function: linear;
  }
  to {
    transform: scale(3);
    background-color: red;
  }
}
```

С помощью свойства `animation-delay` можно задержать начало анимации. Оно работает точно так же, как и `transition-delay`, использующееся для отсрочки переходов, и задает до начала анимации период ожидания на указанное количество миллисекунд или секунд. Например, если нужно подождать одну секунду, прежде чем инициировать постепенное появление на экране контейнера `div` с классом `announcement`, можно переписать стиль `.announcement` следующим образом:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: 1s;
```

```
  animation-delay: 1s;
}
```

Добавление отсрочки к анимации — неплохой способ привлечения внимания и добавления сюрприза к странице.

ПРИМЕЧАНИЕ

Свойства animation-timing-function и animation-delay для браузеров Chrome, Safari и Opera требуют указывать вендорный префикс -webkit-, поэтому не забудьте добавить его: -webkit-animation-timing-function и т. д.

Завершение анимации

С помощью каскадных таблиц стилей можно контролировать еще несколько аспектов анимации, включая ее повторение, направление, если она выполняется более одного раза, а также порядок форматирования браузером элемента по завершении анимации.

Переходы являются анимациями, выполняемыми однократно, например, когда указатель мыши находится поверх кнопки, она увеличивается в размерах. А вот анимации благодаря свойству `animation-iteration-count` могут запускаться сколько угодно раз или непрерывно. Если нужно запустить анимацию 10 раз (возможно, для десятикратного появления и исчезновения элемента), добавьте к анимируемому стилю следующий код:

```
animation-iteration-count: 10;
```

Обычно браузеры проигрывают анимацию только один раз, и если это все, что вам нужно, то свойство подсчета итераций можно оставить в покое. Если же нужно, чтобы анимация повторялась непрерывно, свойству `animation-iteration-count` передается ключевое слово `infinite`. Следовательно, для запуска анимации `fadeIn` для контейнера `div` с классом `announcement` бесконечное количество раз, следует создать такой стиль:

```
.announcement {
  animation-name: fadeIn;
  animation-duration: .25s;
  animation-iteration-count: infinite;
}
```

Но подобное решение, несомненно, утомит ваших пользователей, поэтому лучше от него отказаться. Тем не менее таким стилем можно воспользоваться для «пульсирующего» эффекта, при котором вызывается мягкое свечение кнопки Подпишитесь на рассылку (путем анимации ранее рассмотренного свойства `box-shadow`).

Обычно при многократном запуске анимации браузер запускает ее с самого начала. Поэтому, если анимируется превращение желтого фонового цвета в синий, и это повторяется дважды, браузер покажет, как желтый блок превращается в синий, а затем внезапно появляется желтый блок, который превращается в синий еще раз. Этот эффект может быть слишком резким. В таком случае он будет выглядеть лучше, если через некоторое время после анимации браузер воспроизведет эффект в обратном порядке. Именно так работают переходы: например, когда указатель

мыши находится поверх элемента, браузер анимирует переход из обычного состояния к состоянию, заданному при нахождении поверх элемента. Когда указатель мыши выходит за пределы элемента, браузер воспроизводит анимацию в обратном порядке, возвращая внешний вид элемента в обычное состояние. Чтобы анимация при нечетных воспроизведениях шла в прямом порядке, а при четных — в обратном, используйте свойство `animation-direction` и ключевое слово `alternate`. Например, чтобы элемент постепенно исчезал, а затем снова появлялся, можно создать анимацию `fadeOut`:

```
@keyframes fadeOut {
  from { opacity: 1; }
  to { opacity: 0; }
}
```

А затем воспроизвести ее дважды, меняя при втором воспроизведении направление на противоположное:

```
.fade {
  animation-name: fadeOut;
  animation-duration: 2s;
  animation-iteration-count: 2;
  animation-direction: alternate;
}
```

Этот код запускает анимацию `fadeOut` в отношении любого элемента, имеющего класс `fade`. Анимация должна продолжаться две секунды, а затем повторяться. Поскольку свойство `animation-direction` имеет значение `alternate`, анимация в первый раз приведет к исчезновению элемента (его свойство непрозрачности от полностью непрозрачного (значение `opacity` равно `1`) постепенно сменится на `0` — полностью прозрачное, а затем, при втором воспроизведении, все пойдет вспять (значение `opacity` будет изменяться от `0` до `1`) и элемент снова появится на экране.

СОВЕТ

Чтобы анимация воспроизводилась несколько раз, но в конечном итоге возвращала элементу его первоначальный вид, используйте четное количество итераций и присвойте свойству animation-direction значение alternate.

Неважно, сколько раз вы будет воспроизводиться анимация, но как только она завершится, браузер отобразит элемент в его первоначальном виде. Предположим, что изображение анимируется таким образом, чтобы его размер медленно увеличивался до достижения двойного размера. Как только анимация будет завершена, браузер тут же вернет изображению его первоначальный размер, создавая слишком резкий визуальный эффект. В этом случае можно сохранить элементу тот вид, который был у него по завершении анимации. Для этого нужно присвоить свойству `animation-fill-mode` значение `forwards`.

```
animation-fill-mode: forwards;
```

Это свойство применяется к анимируемому элементу вместе с такими свойствами, как `animation-name`, `animation-duration`, и другими, имеющими отношение к анимации.

Сокращенная запись свойства animation

Как видите, существует множество свойств анимации, и набирать их вручную, да еще и со всеми версиями, включающими вендорные префиксы, может быть затруднительно. Поскольку вам по-прежнему нужно пользоваться версиями с вендорными префиксами, путь к упрощению ситуации лежит через использование сокращенной записи свойства `animation`. В одном этом свойстве сочетаются такие свойства, как `animation-name`, `animation-duration`, `animation-timing-function`, `animation-iteration-count`, `animation-direction`, `animation-delay` и `animation-fill-mode`. Можно, к примеру, взять следующий код:

```
.fade {
  animation-name: fadeOut;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: 5s;
  animation-iteration-count: 2;
  animation-direction: alternate;
  animation-fill-mode: forwards;
}
```

И придать ему такой вид:

```
.fade {
  animation: fadeOut 2s ease-in-out 5s 2 alternate forwards;
}
```

Всего одна строка кода вместо семи! Значения свойств нужно перечислять в ранее упомянутом порядке: имя, длительность, тайминг, количество повторов, направление, отсрочка и режим заполнения. Кроме того, важно указать между значениями запятые. Обязательны только имя и длительность. Все остальные значения опциональны.

Если к элементу нужно применить несколько анимаций, следует воспользоваться списком их свойств с использованием запятой в качестве разделителя. Например, чтобы применить две анимации (скажем, `fadeOut` и `glow`) к элементам, выбираемым стилем `.fade`, напишите следующий код:

```
.fade {
  animation: fadeOut 2s ease-in-out 5s 2 alternate forwards,
             glow 5s;
}
```

Разумеется, при реальном использовании этого свойства нужны еще и версии с вендорными префиксами:

```
.fade {
  -webkit-animation: fadeOut 2s ease-in-out 5s 2 alternate forwards,
                     glow 5s;
  animation: fadeOut 2s ease-in-out 5s 2 alternate forwards,
             glow 5s;
}
```

Ваш выбор должен быть за краткой формой записи: она намного лаконичнее и удобнее.

Приостановка анимации

В CSS доступно еще одно свойство анимации — `animation-play-state`. Оно предназначено для управления воспроизведением анимации. Этому свойству передается одно из двух ключевых слов — `running` или `paused`. Чтобы приостановить анимацию, нужно применить к стилю следующее объявление:

```
animation-play-state: paused;
```

Но фактически применить это свойство к CSS можно только с помощью псевдокласса. Как и в случае с переходами, для приостановки анимации нужен инициатор. Один из вариантов предусматривает приостановку анимации при помещении поверх анимируемого элемента указателя мыши. В следующем примере используется стиль для класса `fade`:

```
.fade {
  animation: fadeOut 2s ease-in-out 2 alternate 5s forwards,
             glow 5s;
}
```

Этот код запускает две анимации — `fadeOut` и `glow` — для любого элемента, к которому применен класс `fade`. Предположим, что нужно позволить посетителям приостанавливать эту анимацию при помещении поверх элемента указателя мыши. Для этого нужно только добавить еще один стиль:

```
.fade:hover {
  animation-play-state: paused;
}
```

Разумеется, вам понадобится вендорный префикс `-webkit-`, то есть свойство примет вид `-webkit-animation-play-state`.

Более эффективным способом приостановки анимации будет динамическое применение свойства `animation-play-state` к элементу с помощью JavaScript-сценария. Можно будет создать сложную анимацию и добавить кнопку паузы, при нажатии которой анимация будет приостановлена. Дополнительные сведения о JavaScript и анимациях CSS представлены во врезке «Использование JavaScript-сценариев для запуска переходов».

Анимация при наведении указателя мыши

Все ранее показанные анимации запускаются при загрузках страницы. Но у вас есть еще несколько вариантов запуска анимации CSS, включая несколько псевдоклассов и использование JavaScript-сценариев. Чаще всего для анимации применяется псевдокласс `:hover`. С его помощью можно запустить анимацию при установке указателя мыши поверх некоего элемента, например, можно придать логотипу элемент динамичности, скрывая его со страницы и отображая снова. Чтобы применить анимацию к элементу при помещении поверх него указателя мыши, нужно сначала создать анимацию с помощью правила `@keyframes` (шаг 1 в предыдущем разделе). Затем для анимируемого элемента следует создать псевдокласс `:hover`. В стиль для этого псевдокласса добавляются свойства анимации

(шаг 2 в предыдущем разделе). Теперь анимация запускается, только когда посетитель устанавливает указатель мыши поверх элемента.

Практикум

В этом уроке к баннеру будут добавлены преобразования, анимация и переходы.

Чтобы начать обучение, вы должны иметь в распоряжении файлы с учебным материалом. Для этого нужно загрузить файлы для выполнения заданий практикума, расположенные по адресу github.com/mrightman/css_4e. Перейдите по ссылке и загрузите ZIP-архив с файлами (нажав кнопку Download ZIP в правом нижнем углу страницы). Файлы текущего практикума находятся в папке 10.

1. Откройте в редакторе HTML-кода файл `styles.css`, который находится в папке 10.

   Файл `banner.html` включает баннер с изображением логотипа, заголовок и набор навигационных кнопок (рис. 10.9). Файл `styles.css` представляет собой таблицу стилей, форматирующую страницу `banner.html`. Сначала будет добавлено преобразование, увеличивающее масштаб изображения кнопки при помещении поверх нее указателя мыши.

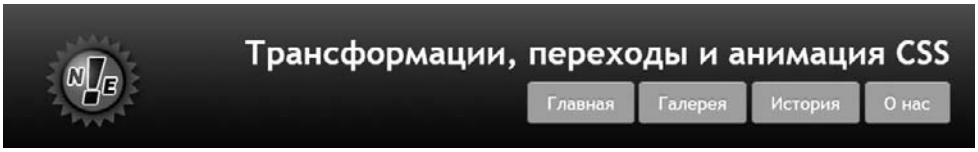

Рис. 10.9. Обычный статический баннер, ожидающий оживления с помощью анимаций, преобразований и переходов

2. В верхней части файла `styles.css` добавьте следующее правило:

```
nav a:hover {
  -webkit-transform: scale(1.5);
  -ms-transform: scale(1.5);
  transform: scale(1.5);
}
```

   Кнопки на странице находятся в HTML5-элементе `nav`. Этот селектор потомков нацелен на содержимое элемента `nav` в состоянии помещения поверх него указателя мыши (то есть на момент нахождения указателя мыши посетителя поверх ссылки). Стиль применяет функцию `scale`, которая немного увеличивает кнопку. Стоит учесть, что для того, чтобы код работал в браузерах Safari и Internet Explorer 9, придется использовать вендорные префиксы `-webkit-` или `-ms-`.

3. Сохраните файл `styles.css` и просмотрите страницу `banner.html` в браузере. Установите указатель мыши поверх одной из ссылок-кнопок под заголовком.

Когда указатель будет находиться поверх кнопки, она станет больше, выделяясь на странице (рис. 10.10). Увеличенная кнопка никак не влияет на окружающий ее контент (например, она не расталкивает соседние кнопки). В этом заключается уникальность преобразований CSS. Неплохо, но было бы еще лучше, если бы ко всему этому добавить анимацию.

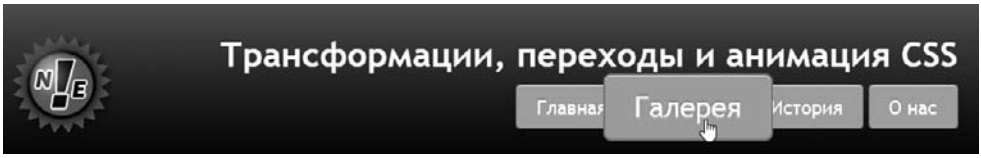

Рис. 10.10. Добавляя кнопке эффект масштабирования при нахождении поверх нее указателя мыши, можно выделить ее на странице

4. Добавьте перед только что созданным стилем `nav a:hover` еще один стиль:

```
nav a {
  transition: all .5s;
}
```

Этот код предписывает браузеру анимировать все изменения свойств элемента `a`, находящегося внутри элемента `nav`, и задает длительность анимации равной половине секунды. Чтобы еще больше оживить ситуацию, добавим к состоянию `hover` фоновый цвет.

5. Найдите стиль `nav a:hover` и добавьте к нему объявление `background-color: red`, чтобы стиль приобрел следующий окончательный вид:

```
nav a:hover {
  background-color: red;
  -webkit-transform: scale(1.5);
  -ms-transform: scale(1.5);
  transform: scale(1.5);
}
```

Если сейчас сохранить страницу и просмотреть ее в браузере, можно увидеть, что кнопки не только становятся крупнее при установке поверх них указателя мыши, но и приобретают красный фон. Но теперь нужно добавить для переходов разные тайминги, чтобы кнопки сначала становились крупнее, а затем приобретали красный фон.

6. Удалите код внутри стиля `nav a` и добавьте два других свойства:

```
nav a {
  -webkit-transition: -webkit-transform .5s,
                      background-color 1s ease-in .5s;
  transition: transform .5s,
              background-color 1s ease-in .5s;
}
```

Вам необходимо добавить два свойства: одно для браузера Safari, а другое — для остальных браузеров. Даже если Safari поддерживает свойство `transition`

(без вендорного префикса), нужно добавить префикс `-webkit-`, поскольку вы используете свойство `-webkit-transform`, которое бы не имело смысла для других браузеров.

Второе свойство — `transition` — предназначено для других браузеров.

Первая часть — `transform .5s` — сообщает браузеру о том, что нужно анимировать любые изменения в свойстве `transform` за полсекунды. Вторая часть — `background-color 1s ease-in .5s` — показывает, что следует анимировать изменение цвета фона за одну секунду, используя для тайминга метод `ease-in`, но до начала перехода взять паузу на полсекунды. Параметр `.5s` в конце играет важную роль, поскольку он соответствует параметру `.5s` в анимации `transform`. То есть браузер будет ждать, пока не завершится переход `transform`, и только потом приступит к изменению цвета фона.

7. Сохраните файл `styles.css` и просмотрите страницу `banner.html` в браузере.

   Установите указатель мыши поверх кнопки. Сначала кнопка станет крупнее. Оставьте указатель поверх кнопки, и через некоторое время она станет красной. Можно, конечно, перенастроить порядок перехода, например удалить отсрочку для перехода `background-color`, и изменение цвета начнется одновременно с изменением размера. Но так как изменение цвета длится в течение одной секунды, оно еще будет продолжаться, когда кнопка уже достигнет своего окончательного размера.

Добавление анимации

Теперь настало время испытать в деле анимацию средствами каскадных таблиц стилей. Начнем с вращения и масштабирования изображения логотипа. Первым шагом в создании анимации CSS станет использование правила `@keyframes` для настройки ключевых кадров анимации. Этот пример начинается с анимации, состоящей всего лишь из двух ключевых кадров.

СОВЕТ

Поскольку анимация CSS требует большого объема кода, лучше начать с использования только одного вендорного префикса для наиболее применяемого браузера. Протестируйте страницу в этом браузере и после доведения ее до совершенства добавьте код для других браузеров.

В этом примере используется код, который будет работать в браузерах Chrome, Safari и Opera. Если вы предпочитаете Firefox, Internet Explorer 10 или его более ранние версии, пропустите часть с префиксом -webkit-.

1. Откройте файл `styles.css` и добавьте следующий код, указав его после стиля `nav a:hover` в нижней части файла:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
  }
  to {
    -webkit-transform: rotate(-720deg) scale(1);
  }
}
```

В примере используется синтаксис `-webkit-`, следовательно, он будет работать в браузерах Chrome, Safari и Opera. Если в качестве основного браузера вы пользуетесь программой Firefox или Internet Explorer 10 (или более поздней версии), замените код `@-webkit-keyframes` фрагментом `@keyframes`.

Код сообщает браузеру, что он должен сначала показать элемент без вращения — `rotate(0)` — и в половинном размере — `scale(.5)`. Затем будут анимированы изменения от состояния `from` до состояния `to` — в данном случае это вращение на –720°, что означает три оборота против часовой стрелки, и возвращение элемента к его нормальному размеру. То есть эта анимация будет вращать элемент и сделает его крупнее.

Теперь нужно применить анимацию к какому-нибудь элементу страницы.

2. Создайте после только что добавленного стиля еще один:

```
.logo {
  -webkit-animation: logo 1s;
}
```

К логотипу баннера применен класс `logo`, следовательно, этот селектор класса применит анимацию к нему. Свойству `animation` можно передать множество значений, но обязательными являются только имя и длительность. Здесь указывается анимация, созданная в последнем действии — `logo`, и браузеру сообщается ее длительность, равная одной секунде.

3. Сохраните файл `styles.css` и просмотрите файл `banner.html` в браузере.

Логотип должен прокрутиться, увеличиться в размерах и остановиться. Если этого не произойдет, перепроверьте код и убедитесь, что просматриваете страницу в соответствующем браузере. Если все идет по плану, эта анимация должна выглядеть весьма привлекательно, но было бы еще лучше, если бы логотип появлялся из-за пределов страницы, а затем останавливался в предназначенной для него позиции баннера. Для этого нужно анимировать позицию элемента на странице.

4. Отредактируйте правило `@keyframe`, придав ему следующий вид:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
    left: 120%;
  }
  to {
    -webkit-transform: rotate(-720deg) scale(1);
    left: 0;
  }
}
```

Здесь начальная позиция логотипа находится правее баннера и кнопок навигации у края экрана (вообще-то, левый край логотипа помещается на 1,2-кратную ширину баннера). Чтобы этот код сработал, нужно воспользоваться свойством `position`. Более подробно мы рассмотрим его в главе 15, а сейчас нужно только понять, что каскадные таблицы стилей дают вам возможность позиционировать любой элемент в любом месте окна браузера и даже за его пределами.

Завершающий ключевой кадр помещает логотип в позицию 0, где он обычно должен появляться на странице, то есть в левой части баннера.

5. Сохраните файл styles.css и просмотрите страницу banner.html в браузере.

   Выглядит неплохо. Но можно сделать еще лучше. Добавьте еще один ключевой кадр, чтобы логотип перекатывался на свое место, а затем вращался в обратном направлении по мере приобретения более крупного размера.

6. Измените код, добавленный в шаге 4, придав ему следующий вид:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
    left: 120%
  }
  50% {
    -webkit-transform: rotate(-720deg) scale(.5);
    left: 0;
  }
  to {
    -webkit-transform: rotate(0) scale(1);
  }
}
```

   Теперь у вас три ключевых кадра: один в начале, второй точно посредине анимации и третий — в ее конце. Браузер будет анимировать свойства по мере их изменения от кадра 1 к кадру 2 и к кадру 3.

   Первый ключевой кадр такой же, как и созданный в шаге 4: логотип не прокручен, имеет половинный размер и помещен за пределами правой стороны. Второй ключевой кард сохраняет логотип в том же размере, три раза его прокручивает и перемещает в левую сторону баннера. И наконец, когда логотип встанет на свое место, браузер увеличит его масштаб, вращая назад с позиции –720° в позицию 0°, то есть логотип теперь будет вращаться три раза по часовой стрелке.

7. Сохраните файл styles.css и просмотрите страницу banner.html в браузере.

   Логотип должен прокатиться с правой стороны баннера в левую, остановиться, увеличиться в размерах и провернуться по часовой стрелке. Если этот код не сработает, перепроверьте его на соответствие коду, добавленному на шаге 6. На данном этапе анимация проигрывается немного быстрее желаемого. Но это легко исправить.

8. Отредактируйте стиль .logo так, чтобы анимация продолжалась не одну, а три секунды:

```
.logo {
  -webkit-animation: logo 3s;
}
```

   Теперь необходимо продублировать код и удалить префикс -webkit-, чтобы код работал в браузерах Firefox и Internet Explorer 10 (и более поздних версиях). Сначала нужно дополнить правило @keyframes.

9. Скопируйте правило `@keyframes` и вставьте копию сразу же после оригинала. Удалите префикс `-webkit-`. Ваш код должен выглядеть следующим образом:

```
@-webkit-keyframes logo {
  from {
    -webkit-transform: rotate(0) scale(.5);
    left: 120%
  }
  50% {
    -webkit-transform: rotate(-720deg) scale(.5);
    left: 0;
  }
  to {
    -webkit-transform: rotate(0) scale(1);
  }
}

@keyframes logo {
  from {
    transform: rotate(0) scale(.5);
    left: 120%
  }
  50% {
    transform: rotate(-720deg) scale(.5);
    left: 0;
  }
  to {
    transform: rotate(0) scale(1);
  }
}
```

Когда-нибудь в прекрасном будущем все браузеры станут использовать единое правило `@keyframes` без всяких префиксов и такое же, не имеющее префиксов, свойство `transform`. Но пока придется набирать весь этот код.

К счастью, самая тяжелая часть — добавление правил `@keyframes`. Добавление анимации к стилю `.logo` потребует меньшего объема работы.

10. Отредактируйте стиль `.logo`, добавив к нему дополнительную строку кода:

```
.logo {
  -webkit-animation: logo 3s;
  animation: logo 3s;
}
```

11. Сохраните файл `styles.css` и просмотрите страницу `banner.html` в браузере Firefox или Internet Explorer (рис. 10.11).

Логотип должен прокатываться по странице и увеличиваться во всех браузерах. Разумеется, в Internet Explorer 9 и более ранних версиях никакой анимации не будет. Но логотип все же будет помещен в нужное место на странице, не нарушая ее внешнего вида. Окончательную версию урока можно найти в папке 10_finished.

Чтобы продолжить усовершенствование страницы, добавьте анимацию, подсвечивающую одну из кнопок, поверх которой установлен указатель мыши. (Подсказка: создайте анимацию, осуществляющую циклический переход через различные значения тени блока. Для добавления тени, размещающейся внутри блока, воспользуйтесь значением `inset`. Затем добавьте анимацию к стилю `nav a:hover`, чтобы она проигрывалась только при помещении указателя мыши поверх кнопки.)

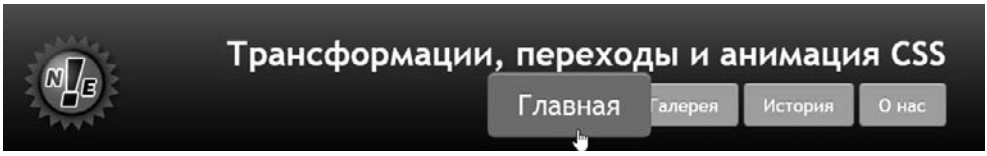

Рис. 10.11. Книги способны на многое, но, к сожалению, они не могут показать ту впечатляющую анимацию, которая только что была создана. Так должна выглядеть страница в ходе воспроизведения анимации и при нахождении указателя мыши над кнопкой