Листинг 3.28. Поворот блока

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;

  -webkit-transform: rotate(30deg);
  -moz-transform: rotate(30deg);
}
```

Отрицательное значение угла поворота меняет направление вращения элемента.

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.28 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

Функция transform: skew

Функция `skew` влияет на симметрию элемента, наклоняя его в обоих измерениях на указанный угол.

Листинг 3.29. Горизонтальный наклон

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  text-align: center;
  border: 1px solid #999999;
  background: #DDDDDD;

  -moz-transform: skew(20deg);
  -webkit-transform: skew(20deg);
}
```

Функция `skew` принимает два параметра, но здесь, в отличие от других функций, каждый из них влияет только на одно измерение, следовательно, эти параметры независимы друг от друга. В листинге 3.29 мы выполнили операцию трансформации поля заголовка, задав наклон на 20° (рис. 3.25). Использован только первый параметр, поэтому искажение происходит лишь в горизонтальном измерении. Если бы мы задали оба параметра, то могли бы исказить объект в обоих

измерениях. В качестве альтернативы можно применять независимые функции `skewX` и `skewY` для каждого из измерений.

Рис. 3.25. Результат использования функции skew()

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.29 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

Функция transform: translate

Аналогично старым свойствам `top` и `left`, функция `translate` перемещает элемент по экрану.

Листинг 3.30. Перемещение блока заголовка вправо

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;

  -webkit-transform: translate(100px);
  -moz-transform: translate(100px);
}
```

Функция `translate` воспринимает экран как сетку пикселов, а точкой отсчета считает исходное положение элемента. Координаты верхнего левого угла элемента равны (0, 0), поэтому отрицательные значения параметра задают перемещение объекта левее или выше исходного местоположения, а положительные — правее или ниже.

В листинге 3.30 мы переместили поле заголовка на 100 пикселов вправо относительно исходного положения. Этой функции можно передавать два значения, чтобы передвинуть элемент как по горизонтали, так и по вертикали. Также можно воспользоваться функциями `translateX` и `translateY` для независимого перемещения в обоих измерениях.

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.30 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

Одновременное использование всех видов трансформации

Иногда возникает необходимость одновременно применить к элементу несколько трансформаций. Для того чтобы получить составное свойство `transform`, нужно всего лишь разделить функции пробелом.

Листинг 3.31. Перемещение, масштабирование и поворот элемента — трансформации определяются в одной строке

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;

  -webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
  -moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);
}
```

Следует помнить о том, что порядок перечисления функций весьма важен, потому что некоторые функции передвигают точку отсчета и центр объекта, изменяя, таким образом, параметры, с которыми будут работать последующие функции.

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.31 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

Динамические трансформации

Возможности, которые мы изучали до сих пор, в значительной степени изменили Сеть, но никак не повлияли на ее статичность. Однако мы можем объединить преобразования и псевдоклассы и превратить нашу страницу в динамическое приложение.

Листинг 3.32. Отклик на действия пользователя

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;
}
#mainbox:hover{
  -webkit-transform: rotate(5deg);
  -moz-transform: rotate(5deg);
}
```

В листинге 3.32 мы сохранили в исходном виде правило из листинга 3.2, описывающее поле заголовка, но добавили новое правило, применяющее эффект трансформирования через псевдокласс `:hover`. В результате каждый раз, когда указатель мыши будет подведен к полю заголовка, свойство `transform` повернет заголовок на 5°, а когда указатель мыши выводится за пределы блока, заголовок возвращается в исходное положение. Таким образом, мы создали простейшую, но довольно полезную анимацию, не применяя ничего, кроме свойств CSS.

ПОВТОРЯЕМ ОСНОВЫ

Псевдокласс :hover, как и другие псевдоклассы, с которыми мы познакомились в главе 2, добавляет специальный эффект к правилу. В этом случае дополнительные стили, описанные в псевдоклассе, применяются, лишь когда указатель мыши расположен над элементом, для которого создано правило. Чтобы получить полный список псевдоклассов, посетите наш веб-сайт и воспользуйтесь ссылками для этой главы.

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.32 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

3D-трансформации

Элементы HTML можно подвергать не только двумерным, но и трехмерным преобразованиям. CSS3 располагает функциями и свойствами, которые позволяют с помощью нескольких строк кода создать потрясающие 3D-эффекты на веб-страницах.

Листинг 3.33. Трехмерное преобразование заголовка

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;

  -webkit-transform: perspective(500px) rotate3d(0, 1, 0, 45deg);
  -moz-transform: perspective(500px) rotate3d(0, 1, 0, 45deg);
}
```

Как видно из листинга 3.33, для трехмерных преобразований используются новые функции. В первом примере задаем перспективу для заголовка документа из листинга 3.1, а также выполняем трехмерное вращение (рис. 3.26).

Рис. 3.26. 3D-эффект

За исключением `perspective()`, остальные функции свойства `transform` похожи на те, что использовались для двумерных эффектов:

- `perspective()`. Создает глубину сцены, увеличивая элементы, расположенные ближе к зрителю. Эту функцию нужно применять первой, чтобы увидеть трехмерные эффекты на экране;
- `translate3d(x, y, z)`. Перемещает элемент в другую точку трехмерного пространства. Ее параметры — три значения в пикселах для осей *X*, *Y* и *Z*;
- `scale3d(x, y, z)`. Масштабирует элемент в трехмерном пространстве. В качестве параметров используются три десятичных значения для задания масштаба по осям *X*, *Y* и *Z*. Как и в двумерном случае, значение 1 сохранит прежний масштаб;
- `rotate3d(x, y, z, angle)`. Вращает элемент в соответствии со значениями, заданными для каждых оси и угла. Значения для осей должны быть десятичными числами, а угол указывается в градусах (например, 30 deg). Значения осей задают вектор для поворота, поэтому важны не абсолютные значения, а их соотношения. Например, `rotate3d(5, 2, 6, 30deg)` осуществит тот же поворот,

что и `rotate3d(50, 20, 60, 30deg)`, поскольку вектор, заданный первыми тремя параметрами, в обоих случаях один и тот же.

Как и раньше, свойство `transform` может содержать столько преобразований, сколько необходимо (листинг 3.34).

Листинг 3.34. Перенос и вращение в 3D

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;
  -webkit-transform: perspective(500px) translate3d(0, 0, -300px)
                     rotate3d(0, 1, 0, 50deg);
  -moz-transform: perspective(500px) translate3d(0, 0, -300px)
                  rotate3d(0, 1, 0, 50deg);
}
```

Кроме функций, есть еще ряд свойств, благодаря которым преобразования выглядят более реалистично:

- `perspective`. Работает аналогично функции `perspective()`, но в родительском блоке. Эффекты перспективы будут применены к потомкам этого блока-контейнера;
- `perspective-origin`. Меняет координаты *X* и *Y* наблюдателя. В качестве параметров использует два значения в процентах, пикселах или ключевые слова `center`, `left`, `right`, `top` и `bottom`. Значения по умолчанию равны 50 %, 50 %;
- `backface-visibility`. Определяет, будет ли видна задняя часть элемента. Допустимы два значения: `visible` или `hidden`. Значение по умолчанию `visible`.

Листинг 3.35. Задание других координат наблюдателя

```
body {
  text-align: center;

  -webkit-perspective: 500px;
  -moz-perspective: 500px;

  -webkit-perspective-origin: 50% 90%;
  -moz-perspective-origin: 50% 90%;
}
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
```

```
    padding: 15px;
    border: 1px solid #999999;
    background: #FFFFFF;

    -webkit-transform: rotate3d(0, 1, 0, 135deg);
    -moz-transform: rotate3d(0, 1, 0, 135deg);
  }
  #title {
    font: bold 36px verdana, sans-serif;
  }
```

Свойство `perspective` должно быть определено для родителя того элемента, на который мы хотим воздействовать. В документе из листинга 3.1 родительский блок для заголовка — это тело документа. Результат применения свойства `perspective` точно такой же, как применение функции `perspective()` для потомка (рис. 3.27). Однако, задавая перспективу таким образом, мы смогли одновременно использовать свойство `perspective-origin` для изменения точки зрения наблюдателя.

Рис. 3.27. 3D-эффект, заданный с помощью свойства perspective

САМОСТОЯТЕЛЬНО

Замените код листинга 3.2 соответствующим кодом из листинга 3.35 и проверьте результат в своем браузере. Попробуйте добавить в #mainbox свойство backface-visibility со значением hidden, чтобы сделать заголовок невидимым, когда он повернут к нам тыльной стороной.

Переходы

Теперь благодаря CSS3 красивые эффекты с динамической трансформацией доступны и просты в реализации. Однако для создания настоящей анимации необходимо определять переходы между двумя стадиями процесса.

Свойство `transition` было создано специально для упрощения этой нетривиальной задачи. Оно волшебным образом создает недостающие шаги, подразумевающиеся в нужном направлении движения. Добавляя всего одно это свойство, мы приказываем браузеру взять на себя заботу об анимации, реализовать вместо нас все эти невидимые шаги и сгенерировать плавный переход из одного состояния в другое.

Листинг 3.36. Красивое вращение с использованием перехода

```
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  background: #FFFFFF;

  -webkit-transition: -webkit-transform 1s ease-in-out 0.5s;
  -moz-transition: -moz-transform 1s ease-in-out 0.5s;
}
#mainbox:hover{
  -webkit-transform: perspective(500px) rotate3d(0, 1, 0, 360deg);
  -moz-transform: perspective(500px) rotate3d(0, 1, 0, 360deg);
}
```

Как видно из листинга 3.36, свойство `transition` принимает до четырех параметров, разделенных пробелами. Первое значение представляет свойство, на базе которого будет создаваться переход (в нашем случае это `transform`). Его необходимо указывать, так как одновременно могут меняться несколько свойств, а нам, вероятно, нужно создать шаги только для одного из них. Второй параметр устанавливает время, за которое должен произойти переход из начального положения в конечное (в нашем примере — 1 секунда). Третьим параметром может выступать любое из пяти ключевых слов: `ease`, `linear`, `ease-in`, `ease-out` и `ease-in-out`. Эти ключевые слова определяют, на основе какой кривой Безье будет выполняться процесс перехода. Каждое ключевое слово представляет отдельную кривую Безье, и единственный способ узнать, какая из них лучше всего подходит именно для вашего перехода, — протестировать все варианты на экране. Последний параметр свойства `transition` — это задержка. Он определяет, какой будет пауза перед началом перехода.

Для того чтобы переход охватывал все меняющиеся свойства данного элемента, можно указать ключевое слово `all`. Также можно явно объявить все участвующие свойства, просто перечислив их через запятую.

САМОСТОЯТЕЛЬНО

Замените кодом из листинга 3.36 соответствующий код в листинге 3.2 и проверьте результат в своем браузере.

ВНИМАНИЕ

В листинге 3.36 мы создали переход для изменений, определяемых свойством transform. Следует иметь в виду, что не всякое свойство CSS поддерживается сейчас свойством transition и всеми браузерами, но со временем положение будет меняться. Вам придется самостоятельно тестировать их или изучать веб-сайты производителей соответствующих браузеров для получения дополнительной информации о поддержке.

Анимация

Анимация, возможно, — самый сложный эффект, достижимый в CSS. В предыдущем разделе мы научились создавать простейшую анимацию с помощью свойства `transition`, но в ней было всего два состояния: начальное и конечное. Однако для настоящей анимации нужно задать несколько шагов, как в кино. CSS3 позволит нам сделать это с помощью функции `@keyframes` и свойства `animation`. Функция создает саму анимацию, а свойство определяет все необходимые параметры, как показано в листинге 3.37.

ВНИМАНИЕ

В настоящее время производители браузеров убирают префиксы для свойств transform, transition и animation, но все еще вероятно, что префиксы необходимы для корректной работы наших примеров. Функция @keyframes использует префикс между символом @ и именем: @-webkit-keyframes.

Листинг 3.37. Первая CSS-анимация

```
body {
  text-align: center;
}
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;
  animation: myanimation 1s ease-in-out 0s infinite normal none;
}
@keyframes myanimation {
  0% {
    background: #FFFFFF;
  }
  50% {
    background: #FF0000;
  }
  100% {
    background: #FFFFFF;
  }
}
#title {
  font: bold 36px verdana, sans-serif;
}
```

Кадры анимации задаются функцией `@keyframes` с помощью процентов от всего времени анимации и стилей в фигурных скобках. Значение 0 % определяет

первый кадр, а 100 % — последний, завершающий кадр анимации. Эти значения необязательны, их можно заменить любыми цифрами от 0 до 100 %, потому что анимация может начаться в любой момент и содержать сколько угодно кадров.

Листинг 3.38. Задание большего количества кадров для анимации

```
body {
  text-align: center;
}
#mainbox {
  display: block;
  width: 500px;
  margin: 50px auto;
  padding: 15px;
  border: 1px solid #999999;

  animation: myanimation 1s ease-in-out 0s infinite normal none;
}
@keyframes myanimation {
  20% {
    background: #FFFFFF;
  }
  35% {
    transform: scale(0.5);
    background: #FFFF00;
  }
  50% {
    transform: scale(1.5);
    background: #FF0000;
  }
  65% {
    transform: scale(0.5);
    background: #FFFF00;
  }
  80% {
    background: #FFFFFF;
  }
}
#title {
  font: bold 36px verdana, sans-serif;
}
```

В листинге 3.38 анимация начинается с 20 %, заканчивается при 80 % и содержит 5 кадров. Свойства, указанные для каждого кадра, показывают, как будет выглядеть элемент на этом этапе процесса анимации, но не определяют, как будет выглядеть сама анимация. За это отвечает свойство `animation`, с помощью которого можно одновременно установить значения нескольких свойств, управляющих процессом анимации:

- `animation-name`. Указывает имя анимации, используемое для создания кадров функцией `@keyframe`, чтобы облегчить связь между кадром и параметрами анимации. Это свойство можно использовать для задания нескольких анимаций, в этом случае имена разделяются запятыми;
- `animation-duration`. Задает длительность одного цикла анимации. Значение должно указываться в секундах (например, 1 s);
- `animation-timing-function`. Работает аналогично знакомому нам свойству `transition` и определяет кривую Безье для процесса перехода. В качестве значений используются ключевые слова `ease`, `linear`, `ease-in`, `ease-out` и `ease-in-out`;
- `animation-delay`. Определяет задержку старта анимации. Значение должно указываться в секундах (например, 1 s), значение по умолчанию равно 0 s;
- `animation-iteration-count`. Определяет, сколько раз будет запущена анимация. В качестве параметра используется целое число или значение `infinite` для бесконечного числа повторений. Значение по умолчанию равно 1;
- `animation-direction`. Определяет направление анимации и задается четырьмя значениями: `normal`, `reverse`, `alternate` и `alternate-reverse`. Значение `normal` устанавливается по умолчанию и не вносит никаких изменений в порядок проигрывания кадров. Значение `reverse` обращает анимацию, показывая кадры в порядке, обратном описанному. Значение `alternate` перемешивает оригинальную анимацию, проигрывая нечетные кадры в нормальном порядке, а четные — в противоположном. Наконец, значение `alternate-reverse` работает аналогично `alternate`, но в нормальном порядке будут проигрываться четные кадры, а в обратном — нечетные;
- `animation-fill-mode`. Определяет стили элемента до и после запуска анимации. Может принимать четыре значения: `none`, `forwards`, `backwards` и `both`. Значение `none` устанавливается по умолчанию и не влияет на стили элемента. Значение `forwards` применяет к элементу стили, заданные в последнем кадре анимации, определенном как `100%` или ключевым словом `to`. Значение `backwards` применит к элементу стили первого кадра, определенного как `0%` или ключевым словом `from`. Это случится еще до того, как будет запущена анимация. Наконец, значение `both` приведет к тому, что будут реализованы оба эффекта.

Как мы уже видели в примерах, можно задать все эти свойства одновременно, используя свойства `animation` и следующий синтаксис (порядок свойств имеет значение): `animation: name duration timing-function delay iteration-count direction fill-mode`.

ВНИМАНИЕ

Не забудьте добавить соответствующие префиксы к свойству animation и функции @keyframes (например, -webkit-animation, @-webkit-keyframes), когда будете тестировать примеры кода в браузере.