14 Позиционирование элементов на странице

Когда Консорциум W3C представил концепцию *CSS-позиционирования*, некоторые разработчики подумали, что они смогут делать так, чтобы веб-страницы выглядели как печатные документы, созданные в компьютерных текстовых редакторах. Благодаря лишь нескольким свойствам CSS-позиционирование позволяет поместить элемент в определенной позиции на странице, скажем в 100 пикселах от верхнего края страницы и 200 пикселах от левого края. Казалось, что точное пиксельное размещение обещало, что вы наконец-то сможете проектировать страницу, просто помещая фотографию в одной позиции, заголовок — в другой и т. д.

К сожалению, те возможности контроля, которые разработчики ожидали от CSS-позиционирования, так и не были реализованы. Разные браузеры всегда по-разному отображали элементы, позиционированные с помощью каскадных таблиц стилей. Но, что еще более существенно, Всемирная паутина функционирует не так, как печатная брошюра, журнал или книга. Веб-страницы намного более изменчивы, чем печатные. Как только журнал тиражируется в издательстве, читатели не могут изменить размер страницы или шрифта.

Посетители сайтов, с другой стороны, могут переделать вашу работу. Они могут увеличить размер шрифта в своем браузере, что, возможно, приведет к тому, что текст выйдет за пределы точно позиционированных элементов макета, которым заданы определенные размеры. Кроме того, поскольку в наше время для веб-серфинга люди используют смартфоны, планшеты и даже телевизоры, нельзя заранее предсказать размеры экранов, чтобы точно определить позиции *каждого* элемента на странице. Но все не так плохо: пока вы не пытаетесь навязать определенные ширину, высоту и позицию каждого элемента, свойства CSS-позиционирования будут для вас полезны. Вы можете использовать их, чтобы поместить подпись над фотографией, логотип в верхней части страницы и т. д.

Принципы работы свойств позиционирования

Свойство `position` каскадных таблиц стилей позволяет управлять, *где* и *как* браузер отобразит определенные элементы. Используя его, вы можете, например, поме-

стить боковую панель на странице там, где вы этого желаете, или убедиться, что навигационная панель наверху страницы останется на своем месте, даже если пользователи прокручивают страницу вниз. Каскадные таблицы стилей предлагают четыре типа позиционирования.

- **Абсолютное.** Такое позиционирование позволяет устанавливать расположение элемента, задавая позиции `left`, `right`, `top` или `bottom` в пикселах, единицах em или процентах (для получения дополнительной информации о выборе между различными единицами измерения см. главу 6). Вы можете поместить блок на расстоянии 20 пикселов от верхнего и 200 пикселов от левого края страницы, как показано на рис. 14.1, *посередине* (далее вы узнаете, как писать код с этими инструкциями).

  Кроме того, абсолютно позиционированные элементы полностью отделены от потока страницы, определенного HTML-кодом. Другими словами, остальные элементы на странице не подозревают, что существует абсолютно позиционированный элемент. По вашей невнимательности они могут даже полностью исчезнуть, попав под него.

ПРИМЕЧАНИЕ

Не пытайтесь применять одновременно свойство float и любой тип позиционирования, кроме статичного (рассмотрен ниже). Выравниваемые объекты и позиционирование (абсолютное или фиксированное) не могут применяться к одному и тому же элементу.

- **Относительное.** Элемент с таким позиционированием позиционируется относительно его текущего положения в HTML-потоке. Так, например, присваивая свойству `top` значение 20 пикселов и `left` — значение 200 пикселов для относительно позиционированного заголовка, вы переместите его на 20 пикселов вниз и 200 пикселов влево *от той позиции, где он появился бы на странице*.

  В отличие от абсолютного позиционирования, здесь остальные элементы страницы регулируют старое HTML-размещение относительно позиционированного объекта. Соответственно, перемещение объекта с относительным позиционированием оставляет «дыру», на месте которой он должен был находиться. Посмотрите на темную полосу на рис. 14.1, *внизу*. Это то место, где *появился бы* относительно позиционированный блок, если бы ему не было дано указание на перемещение. Основная польза относительного позиционирования не в том, чтобы переместить элемент, а в установке новой точки привязки для абсолютно позиционированных элементов, которые вложены в него (больше об этой концепции вы узнаете в подразделе «Когда абсолютное позиционирование относительно» далее).
- **Фиксированное.** Фиксированный элемент блокируется в определенной позиции на экране. Определение такого позиционирования играет ту же роль, что и присвоение значения `fixed` свойству `background-attachment` (см. раздел «Позиционирование фоновых изображений» главы 8). Когда посетитель прокручивает страницу, фиксированные элементы остаются на экране, например абзацы и заголовки, в то время как фотографии прокручиваются вместе со страницей.

  Использование фиксированных элементов — отличный способ создать неподвижную панель навигации в верхней или нижней части окна браузера. Вы скоро узнаете, как создается такой эффект.

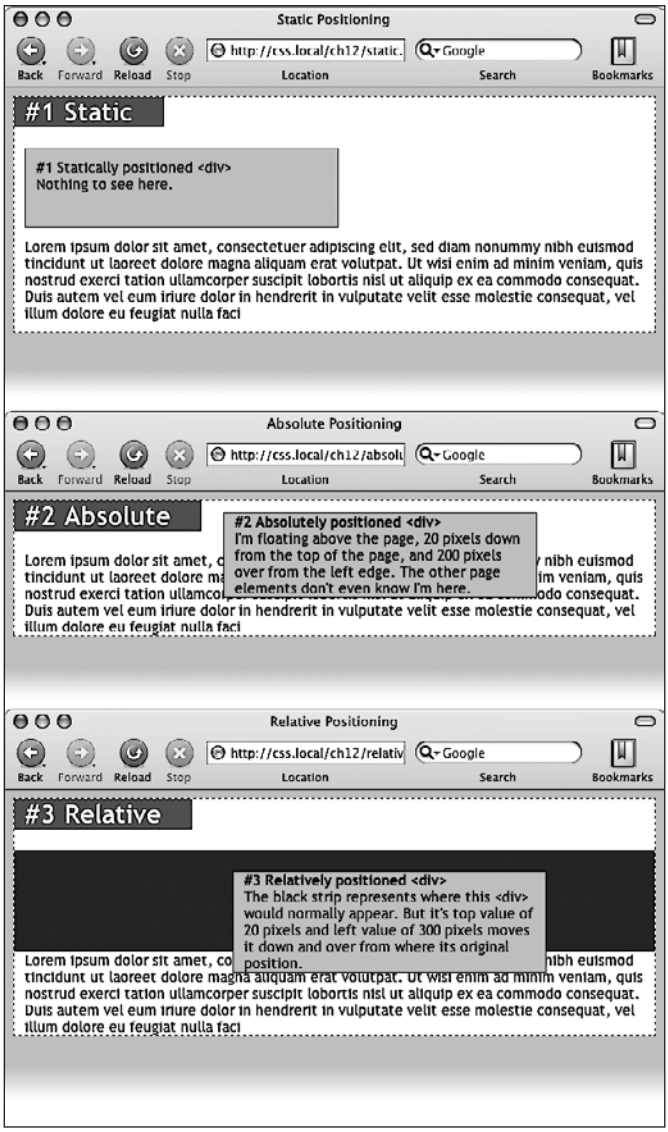

Рис. 14.1. Каскадные таблицы стилей предлагают несколько путей позиционирования на веб-странице

- **Статичное позиционирование** просто означает, что контент следует нормальному нисходящему потоку HTML-элементов (см. рис. 14.1, *вверху*). Но зачем тогда назначать элементу статичное позиционирование? Скорее всего, вы никогда не будете делать этого.

Итак, можно сделать выводы. Статичное позиционирование — это то, как браузеры представляли содержимое с начала существования Всемирной паутины. Они просто отображали HTML-элементы в нисходящем порядке (сверху вниз). Абсо-

лютное позиционирование удаляет элемент из потока страницы, перемещая его в верхнюю часть страницы, иногда перекрывая какой-нибудь другой контент. Относительное позиционирование помещает элемент относительно той позиции, где он появился бы на странице, и оставляет «дыру» там, где этот элемент находился бы без относительного позиционирования.

Чтобы изменить позиционирование любого элемента, используйте свойство `position`, которому присваивается одно из этих четырех значений: `static`, `absolute`, `relative` или `fixed`. Чтобы создать абсолютно позиционированный элемент, добавьте следующее свойство в стиль:

```
position: absolute;
```

Статичное позиционирование — это обычный метод расположения элементов, так что, если только вы не отменяете ранее созданный стиль, которым уже назначается абсолютное, относительное или фиксированное позиционирование, вам не нужно указывать значение `static`. Кроме того, статичные элементы не подчиняются ни одному из значений позиционирования, как обсуждается далее.

Присвоение значений позиционирования — это обычно только часть битвы. Чтобы на самом деле расположить элемент где-нибудь на странице, вы должны разобраться с различными свойствами позиционирования.

Настройка позиций

Область отображения браузера, также называемая областью просмотра, имеет верхний, нижний, правый и левый края. Для каждого из этих четырех краев существует соответствующее CSS-свойство: `top`, `bottom`, `left` и `right`. Вам не нужно задавать значения для всех четырех сторон. Обычно достаточно двух, чтобы расположить элемент на странице. Вы можете, к примеру, поместить элемент на расстоянии 10 em от левого края страницы и 20 em от верхнего.

Чтобы указать расстояние от края страницы до соответствующей стороны элемента, используйте любую из действующих в CSS единиц измерений: пикселы, единицы em, проценты и т. д. При позиционировании можно также использовать отрицательные значения, например `left: -10px;`, чтобы поместить элемент частично за пределы страницы (или другого элемента). Это приведет к появлению особого визуального эффекта, который будет описан в разделе «Эффективные стратегии позиционирования» этой главы.

После свойства `position` вы указываете одно свойство или более для сторон (`top`, `bottom`, `left` или `right`). Если вы хотите, чтобы элемент принял ширину меньше допустимой (например, чтобы сделать тонкую боковую панель), то можете также использовать свойство `width`. Чтобы поместить баннер страницы в определенной позиции относительно верхнего и левого краев окна, создайте следующий стиль:

```
.banner {
  position: absolute;
  left: 100px;
  top: 50px;
  width: 760px;
}
```

Этот стиль расположит баннер так, как показано на рис. 14.2, *вверху*.

ПРИМЕЧАНИЕ

Если не указать значение позиции по вертикали (top или bottom), браузер помещает элемент в ту же позицию на странице по вертикали, где бы он располагался при отсутствии позиционирования. То же самое справедливо для настройки размещения по горизонтали (left или right). То есть, если установить для элемента абсолютное позиционирование, но не предоставить значений позиционирования top, right, bottom или left, браузер оставит элемент в той же позиции, но поверх другого контента.

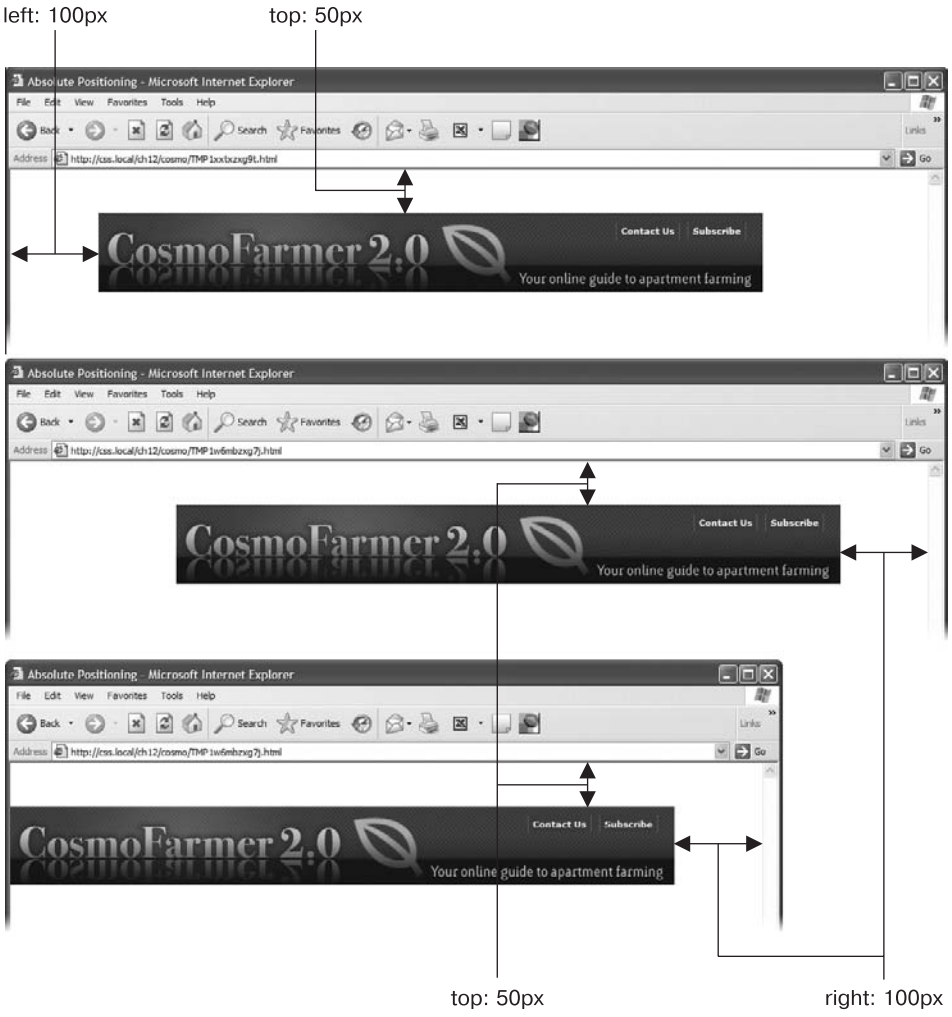

Рис. 14.2. Польза абсолютного позиционирования состоит в возможности поместить элемент относительно правого края окна браузера *(посередине)*. Если ширина окна изменится, расстояние между правым краем окна и элемента останется неизменным *(внизу)*

Рассмотрим другой пример: поместим элемент таким образом, чтобы он всегда оставался на определенном расстоянии от правой стороны браузера. Если вы ис-

пользуете свойство `right`, браузер измеряет расстояние от правого края окна до правого края элемента (см. рис. 14.2, *посередине*). Чтобы поместить баннер на расстоянии 100 пикселов от правого края окна, вы создадите тот же самый стиль, что и ранее, но только используя свойство `right` вместо `left`:

```
.banner {
  position: absolute;
  right: 100px;
  top: 50px;
  width: 760px;
}
```

Поскольку рассчитываемая позиция основывается на правом крае окна браузера, регулирование его размера автоматически меняет расположение баннера, что вы можете видеть на рис. 14.2, *внизу*. Хоть баннер и смещается, расстояние между правым краем элемента и правым краем окна браузера остается неизменным.

Можно даже указать свойства для левой и правой позиций, а также для верхней и нижней и позволить браузеру определить ширину и высоту элемента. Скажем, вы хотите, чтобы центральный блок текста размещался на расстоянии 10 % от верхнего края окна браузера и 10 пикселов от левого и правого края. Чтобы позиционировать блок, вы можете использовать стиль с абсолютным позиционированием, с присвоением свойствам `top`, `left` и `right` значения 10 %. В окне браузера левый край блока начинается от левого края окна на расстоянии 10 % от ширины окна, а правый край находится на расстоянии 10 % от правого края (рис. 14.3, *вверху*). Точная ширина блока при этом зависит от ширины окна браузера. Более широкое окно увеличит блок; чем у́же окно, тем меньше будет блок. Левая и правая позиции, однако, по-прежнему составляют 10 % от ширины окна браузера.

Свойства `width` и `height`, о которых вы узнали в главе 7, работают аналогично и для позиционированных элементов. Чтобы поместить серый блок размером 50 × 50 пикселов в верхний правый угол окна браузера, создайте следующий стиль:

```
.box {
  position: absolute;
  right: 0;
  top: 0;
  width: 50px;
  height: 50px;
  background-color: #333;
}
```

Здесь следует вспомнить предостережение, приведенное в разделе «Изменение высоты и ширины» главы 7: будьте внимательны с указанием высоты элементов. Если вы не создаете какие-либо изображения с уже заданной высотой, то не можете знать, насколько высоким будет на странице конкретный элемент. Вы могли бы определить для боковой панели высоту 200 пикселов, но если вы добавите текст или изображения, которые увеличат боковую панель свыше 200 пикселов в высоту, то в конечном счете содержимое панели выйдет за ее пределы. Даже если вы уверены, что контент поместится, посетитель в любой момент может увеличить размер шрифта в своем браузере, сделав текст достаточно крупным, чтобы он мог «выпасть» из блока. К тому же, если вы ограничиваете ширину и высоту в стиле,

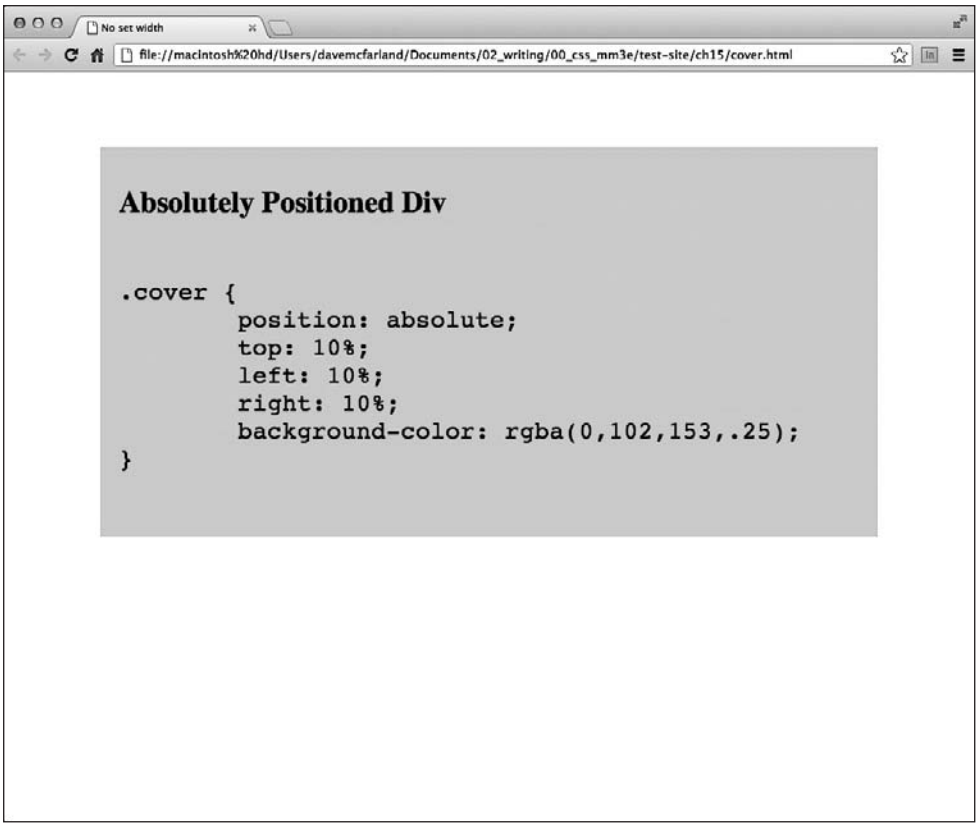

Рис. 14.3. Можно позволить браузеру определить ширину элемента, указав для элемента с абсолютным позиционированием значения свойств как left, так и right

а контент внутри разрабатываемого элемента оказывается шире или выше, могут быть достигнуты непредсказуемые результаты.

Когда абсолютное позиционирование относительно

Ранее в этой главе мы говорили о позиционировании элемента в конкретной позиции в окне браузера. Однако абсолютное позиционирование не всегда работает таким образом.

На самом деле абсолютно позиционированный элемент помещается *относительно* пределов его ближайшего позиционированного предка. Проще говоря, если вы уже создали элемент с абсолютным позиционированием (скажем, контейнер `div`, который появится на расстоянии 100 пикселов от верхнего края браузера), то любые абсолютно позиционированные HTML-элементы *внутри* этого контейнера `div` размещаются относительно верхнего, нижнего, левого и правого краев контейнера.

ПРИМЕЧАНИЕ

Если вы не помните, что такое родительские элементы и предки, см. раздел «Форматирование вложенных элементов» главы 3.

На верхнем изображении на рис. 14.4 светло-серый блок абсолютно позиционирован на расстоянии 5 em от верхнего и левого краев окна браузера.

Есть также элемент `div`, вложенный в этот блок. Абсолютное позиционирование этого элемента позволит поместить его *относительно абсолютно позиционированного родительского элемента*. Присвоение свойству `bottom` значения `0` переместит блок не к основанию экрана, а к основанию его предка. Аналогично свойство `right` этого вложенного элемента `div` относится к правому краю его родителя (см. рис. 14.4, *внизу*).

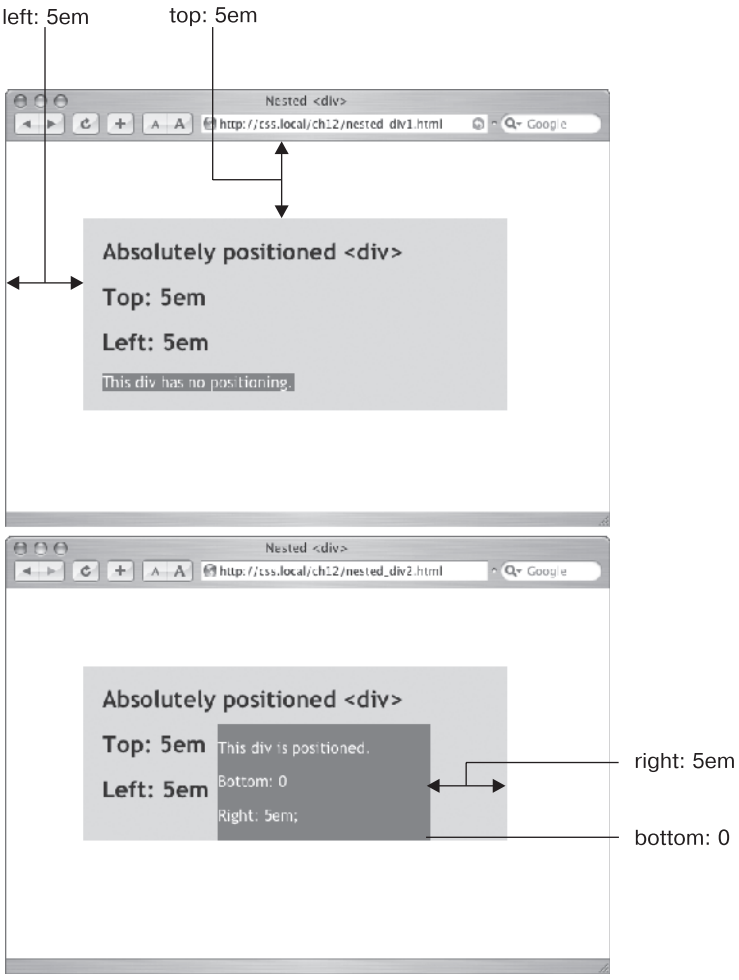

Рис. 14.4. Вы можете расположить элемент относительно окна браузера *(вверху)* или относительно позиционированного предка *(внизу)*

Каждый раз, когда вы используете абсолютное позиционирование, чтобы расположить элемент на странице, его точная позиция зависит от местонахождения любых других элементов, в которые вложен форматируемый элемент. Позиционирование подчиняется следующим правилам.

- **Элемент позиционирован относительно окна браузера,** если у него абсолютное позиционирование *и* он не находится внутри любого другого элемента, к которому применено абсолютное, относительное или фиксированное позиционирование.
- **Элемент позиционирован относительно сторон другого элемента,** если он находится внутри другого элемента с абсолютным, относительным или фиксированным позиционированием.

Когда (и где) использовать относительное позиционирование

Вы получаете существенную выгоду, помещая элемент относительно другого элемента: если он смещается, позиционированный элемент перемещается вместе с ним. Скажем, вы поместили изображение внутрь элемента `h1` и хотите, чтобы оно появилось с правого края заголовка. Если вы просто поместите изображение в конкретной позиции в окне браузера, то рискуете потерять его из виду. Если заголовок будет смещен, абсолютно позиционированное изображение останется «приклеенным» к назначенной ему позиции. Вместо этого лучше определить изображение относительно элемента `h1` так, что, если заголовок будет перемещен, изображение переместится вместе с ним (два нижних изображения в нижней части рис. 14.5).

ПРИМЕЧАНИЕ

Используйте свойство background-image (см. раздел «Добавление фоновых изображений» главы 8), чтобы поместить изображение в качестве фона элемента h1. Однако если изображение будет выше элемента h1 или вы хотите, чтобы оно появилось за пределами заголовка (см. третье изображение на рис. 14.5), встройте изображение с помощью элемента img и воспользуйтесь относительным позиционированием, рассматриваемым в данном разделе.

Чтобы поместить изображение на страницу, вы могли использовать значение `relative` свойства `position`, но здесь также есть недостатки. Если вы устанавливаете относительное позиционирование для элемента, а затем размещаете его (к примеру, используя свойства `left` и `top`), элемент перемещается на определенное расстояние от той позиции, где он появился бы в нормальном потоке HTML. Другими словами, он перемещается относительно своего текущего положения. В результате на его исходном месте остается пустое пространство, которое элемент занимал бы без настройки позиционирования (см. рис. 14.1, *внизу*).

Лучший способ использовать относительное позиционирование состоит в том, чтобы создать новую связь позиционирования для вложенных элементов. Например, элемент `h1` в примере в начале этого раздела является предком элемента `img`, который находится внутри него. При установке относительного позиционирования элемента `h1` любое абсолютное позиционирование, которое вы примените к элементу `img`, будет определено относительно четырех сторон заголовка `h1`, а не окна браузера. CSS-код выглядит следующим образом:

```
h1 { position: relative; }
h1 img {
  position: absolute;
  top: 0;
  right: 0;
}
```

Присвоение свойствам `top` и `right` изображения значения `0` перемещает его в верхний правый угол заголовка, а не окна браузера.

В CSS термин *относительный* означает не совсем то же самое, что в реальном мире. Все же, если вы хотите поместить элемент `img` относительно элемента `h1`, вашей первой мыслью может быть определение для изображения относительной позиции. На самом деле элемент, который вы хотите поместить, — изображение — получает абсолютное позиционирование, в то время как элемент, *относительно которого* вы хотите определить изображение, — заголовок — получает значение `relative`. Расшифруйте его как «относительно меня». Когда вы применяете относительное позиционирование к элементу, это означает, что «все элементы, заданные внутри него, должны размещаться относительно его местоположения».

ПРИМЕЧАНИЕ

Поскольку вы будете часто применять относительное позиционирование для задания новой связи расположения вложенных элементов, вам даже не нужно использовать значения top, bottom, left и right. У элемента h1 есть свойство position: relative, но нет значений top, bottom, left и right.

Наложение элементов

Как вы можете видеть на рис. 14.6, абсолютно позиционированные элементы расположены на переднем плане веб-страницы и могут даже находиться поверх (или позади) других позиционированных элементов. Такое наложение определяется индексом позиционного уровня (`z-index`). Если вы знакомы с понятием стопок слоев в программе Photoshop, Sketch или Adobe Illustrator, то знаете, как работает индекс позиционного уровня: он представляет порядок, в котором элементы накладываются друг поверх друга на странице.

Чтобы сказать это другими словами, представьте веб-страницу как лист бумаги, а абсолютно позиционируемый элемент — как стикер, приклеиваемый поверх. Каждый раз, когда вы добавляете на страницу абсолютно позиционированный элемент, вы «приклеиваете» на нее стикер. Конечно, если вы делаете это, то рискуете перекрыть какой-либо контент на странице.

Обычно порядок наложения элементов следует их порядку в HTML-коде страницы. На странице с двумя абсолютно позиционированными элементами `div` второй HTML-элемент появится *поверх* другого элемента `div`. Однако вы можете управлять порядком, в котором накладываются элементы, используя свойство `z-index` каскадных таблиц стилей. В свойстве указывается числовое значение:

```
z-index: 3;
```

Чем больше значение, тем выше в стопке появится элемент. Скажем, у вас есть три абсолютно позиционированных изображения и все они частично перекрываются. Изображение, имеющее больший индекс позиционного уровня, появится

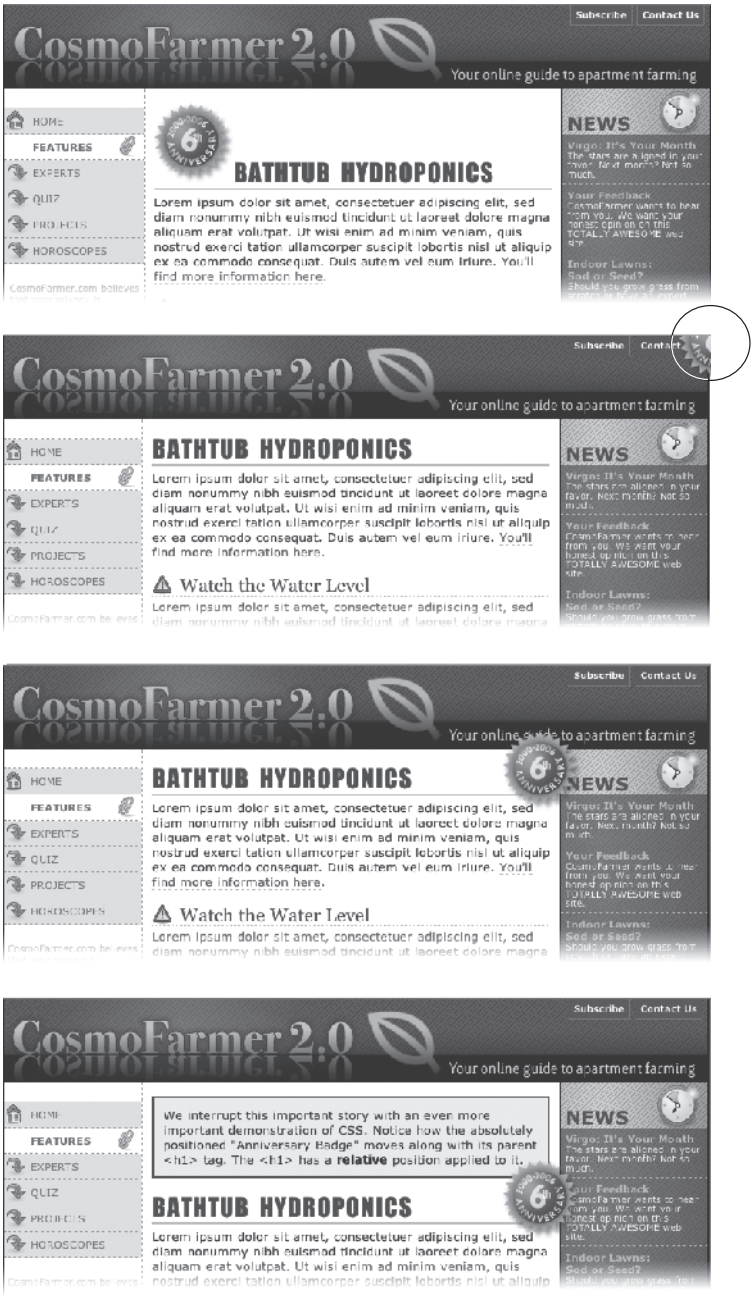

Рис. 14.5. *Вверху*: круглая графическая кнопка помещена внутрь элемента h1. *Второе сверху*: на втором изображении страницы к кнопке применено абсолютное позиционирование, которое смещает ее за пределы области элемента h1 и устанавливает в верхнем правом углу окна браузера. *Третье сверху*: в коде третьей веб-страницы к элементу h1 добавлено свойство position: relative, которое создает новую связь позиционирования для элемента img. *Внизу*: когда вы смещаете заголовок ниже, изображение также перемещается

поверх других (см. рис. 14.6, *вверху*). Когда вы меняете индекс позиционного уровня одного или нескольких изображений, вы изменяете порядок их наложения (см. рис. 14.6, *посередине*).

Свойству `z-index` можно присваивать и отрицательные значения, которые могут пригодиться, если нужно позиционировать элемент позади его родительского элемента или любого из его предков. Например, на рис. 14.6, *вверху*, элемент `div` имеет относительное позиционирование. Если нужно поместить одно из изображений позади `div`-контейнера, можно воспользоваться отрицательным значением свойства `z-index`:

```
z-index: -1;
```

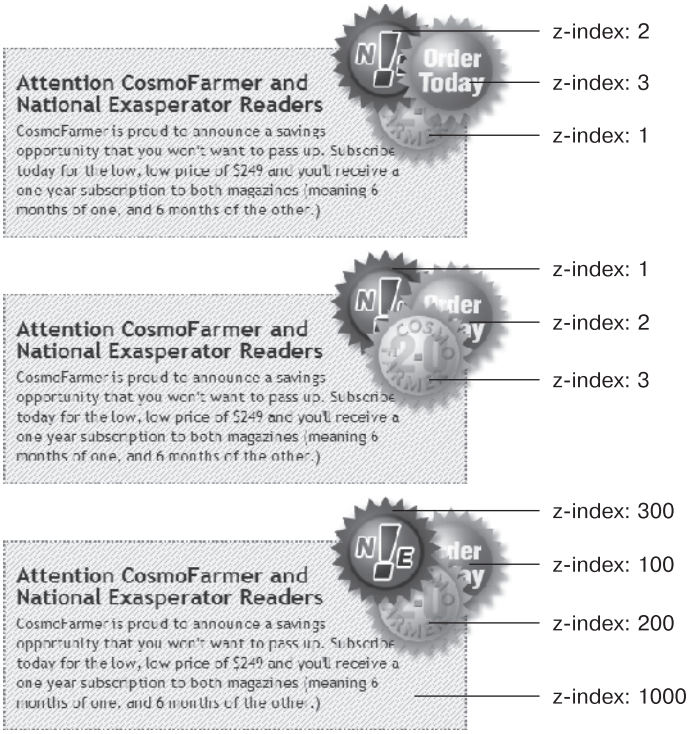

Рис. 14.6. Если вы используете индекс позиционного уровня, чтобы управлять порядком наложения элементов, родительские элементы определяют порядок наложения для своих детей

На рис. 14.6 в двух верхних примерах текстовый блок не позиционирован: он разделяет индекс позиционного уровня самой страницы непосредственно, для которой значение свойства `z-index` равно `0`. Таким образом, кнопки в этих двух текстовых блоках накладываются поверх страницы. Однако для текстового блока в нижнем изображении задано абсолютное позиционирование со значением `1000` свойства `z-index`. Элемент `div`, содержащий текстовый блок, становится отправной точкой для укладки в него изображений. Поэтому, хотя свойству `z-index` контейнера `div` присвоено значение `1000`, его вложенные дети (с более низкими значениями свойства `z-index`) появятся поверх, в то время как сам текстовый блок находится позади.

СОВЕТ

Промежутки в значениях индекса позиционного уровня вполне допустимы. Другими словами, значения 10, 20, 30 определяют то же самое, что и 1, 2, 3. В действительности, промежуток в числовых значениях допускает в дальнейшем добавление новых элементов. Если нужно обеспечить, чтобы поверх того или иного элемента больше ничего не появлялось, задайте ему очень большой индекс позиционного уровня, например z-index: 10000;. Но не переусердствуйте, поскольку максимальное значение, которое обрабатывают некоторые браузеры, равно 2147483647.

Сокрытие фрагментов страницы

Другое свойство каскадных таблиц стилей, часто используемое с абсолютно позиционированными элементами, — `visibility`. Оно позволяет скрыть часть страницы (или отобразить ее). Скажем, вы хотите, чтобы поверх изображения внезапно появлялось всплывающее сообщение, когда пользователь помещает поверх него указатель мыши. Или сделать подрисуночную подпись невидимой, когда страница загружается (`visibility: hidden`), и переходящей в режим видимости (`visibility: visible`), когда указатель находится поверх изображения.

Значение `hidden` свойства `visibility` подобно значению `none` свойства `display`, но между ними есть существенное различие. Если вы присваиваете свойству `display` элемента значение `none`, он буквально бесследно исчезает со страницы. А присвоение свойству `visibility` значения `hidden` предотвращает отображение браузером содержимого элемента, но оставляет пустое пространство в той позиции, где должен был быть элемент. При использовании с абсолютно позиционированными элементами, которые уже удаляются из потока страницы, свойства `visibility: hidden` и `display: none` ведут себя одинаково. Большинство разработчиков используют метод `display: none` и вообще не обращаются к свойству `visibility`.

Существует и другой способ сокрытия элемента — присвоение его свойству непрозрачности `opacity` нулевого значения:

```
opacity: 0;
```

Чтобы элемент снова появился на экране, его свойству `opacity` можно вернуть значение `1`:

```
opacity: 1;
```

Преимущество использования свойства `opacity` заключается в том, что браузер его может анимировать. Это означает, что вы можете использовать CSS-переходы, рассмотренные в главе 10, для анимации изменений уровня непрозрачности. Например, элемент можно сделать постепенно появляющимся, изменяя уровень его непрозрачности от `0` до `1` и добавляя переход.

Самый распространенный способ переключать режим элемента от скрытого к видимому и наоборот — через JavaScript-сценарий. Однако вам не нужно изучать программирование на JavaScript, чтобы использовать свойство `visibility` (или `display`). Вы можете добавить псевдокласс `:hover` (см. раздел «Псевдоклассы и псевдоэлементы» главы 3), чтобы сделать невидимый элемент видимым. Предположим, вам требуется поместить подпись поверх изображения, но важно, чтобы она появлялась только при нахождении указателя мыши над изображением (рис. 14.7). Для достижения результата выполните следующие действия.

1. Создайте HTML-код для изображения и подписи.

   Один из способов предполагает использование HTML5-элементов `figure` и `figcaption`:

   ```
   <figure class="hat">
   <img src="hat.jpg" width="100" height="100">
     <figcaption>Изображение шляпы</figcaption>
   </figure>
   ```

   Здесь к элементу `figure` применяется класс `hat`, поэтому стиль можно применить только к данному изображению.

2. Позиционируйте подпись.

   Для помещения подписи поверх изображения используется абсолютное позиционирование. Чтобы поместить подпись относительно элемента `img`, нужно установить для его родительского элемента `figure` относительное позиционирование, а также задать параметры ширины и высоты, соответствующие размерам фотографии.

   ```
   .hat {
     position: relative;
     width: 100px;
     height: 100px;
   }
   .hat figcaption {
     position: absolute;
     bottom: 0
     left: 0;
     right: 0;
     background-color: white;
   }
   ```

   Подпись помещается в нижнюю часть изображения (`bottom: 0`). За счет присвоения нулевых значений его свойствам `left` и `right` подпись займет всю ширину рисунка.

3. Скройте подпись.

   При использовании кода выше браузер отобразит подпись поверх изображения, но вам нужно, чтобы это происходило только при нахождении указателя мыши над изображением. Чтобы скрыть подпись, добавьте к стилю `.hat figcaption` строку кода либо `visibility: hidden`, либо `display: none`.

   ```
   .hat figcaption {
     display: none;
     position: absolute;
     bottom: 0
     left: 0;
     right: 0;
     background-color: white;
   }
   ```

4. Сделайте так, чтобы подпись появлялась, когда посетитель помещает указатель мыши поверх изображения.

Для этого воспользуйтесь псевдоклассом :hover. Нужно сделать подпись видимой при нахождении поверх изображения указателя мыши. К сожалению, такого стиля не существует, но, так как подпись находится внутри элемента figure, можно создать селектор потомков, влияющий на подпись при расположении указателя мыши над изображением:

```
.hat:hover figcaption {
  display: block;
}
```

Этот селектор потомков предписывает следующее: «Нужно нацелиться на любой элемент figcaption, который находится внутри элемента с классом hat, но только когда указатель мыши находится поверх элемента». Он работает только потому, что элемент figcaption является потомком элемента, поверх которого находится указатель мыши.

Такую идею можно использовать для создания основанных на CSS-коде всплывающих подсказок — дополнительной информации, появляющейся при помещении указателя мыши поверх ссылки. Для подробностей обратитесь к сайту tinyurl.com/nnmcjlo. Существует также множество JavaScript-сценариев: например, полноценным и простым в использовании JavaScript-сценарием для создания всплывающих подсказок является jQuery-модуль aTip2 (tinyurl.com/pf2myco).

Как уже упоминалось, для сокрытия/отображения элемента можно также воспользоваться свойством opacity, а для анимации эффекта — свойством transition. Для этого можно изменить код ранее созданных стилей следующим образом:

```
.hat figcaption {
  opacity: 0;
  transition: opacity .5s ease-in;
  position: absolute;
  bottom: 0
  left: 0;
  right: 0;
  background-color: white;
}
.hat:hover figcaption {
  opacity: 1;
}
```

Вы рассмотрите этот пример в действии в практикуме этой главы. Но следует помнить, что свойство transition не поддерживается браузером Internet Explorer 9 и более ранними версиями. Однако в данном случае ничего страшного не произойдет. Ваши посетители все равно увидят подпись, но только без ее плавного появления и исчезновения.

Эффективные стратегии позиционирования

Как говорилось в начале этой главы, если вы попытаетесь использовать CSS-позиционирование для размещения *каждого* элемента страницы, то можете столкнуться с проблемой. Поскольку просто невозможно предсказать все мыслимые комби-

Рис. 14.7. Подпись то видна, то не видна. Элемент с абсолютным позиционированием можно поместить поверх другого элемента, но скрыть его *(слева)*, пока посетитель не наведет указатель мыши на родительский элемент *(справа)*

нации браузеров и параметров настройки, используемые вашими посетителями, позиционирование под управлением CSS работает лучше всего в качестве «тактического оружия». Применяйте его аккуратно, чтобы обеспечить точное размещение для определенных элементов.

В этом разделе вы узнаете, как использовать абсолютное позиционирование, чтобы добавлять маленькие, но важные детали к дизайну своей страницы, как абсолютно позиционировать определенные компоненты макета и закреплять важные элементы страницы в одном месте, в то время как остальной контент будет прокручиваться.

Позиционирование внутри элемента

Один из самых эффективных способов использования позиционирования состоит в том, чтобы размещать маленькие элементы относительно других объектов на странице. Абсолютное позиционирование похоже на выравнивание по правому краю, которое вы задаете обтекаемым элементам. На рис. 14.8, *1*, дата на верхнем заголовке кажется высоко расположенной, но с помощью каскадных таблиц стилей вы можете переместить ее к правому краю нижнего заголовка.

Чтобы поместить дату отдельно от остальной части заголовка, вам нужно заключить ее в HTML-элемент. Добавление элемента `span` — распространенный способ применения класса к строке текста, форматирующий ее отдельно от остальной части абзаца:

```
<h1><span class="date">Nov. 10, 2006</span> CosmoFarmer Bought By Google</h1>
```

Теперь возникает вопрос о создании стилей. Во-первых, вы должны присвоить элементу-контейнеру (здесь — `h1`) значение `relative` (относительное позиционирование). Затем применить абсолютное позиционирование (значение `absolute`) к элементу, который желаете разместить, — дате. Далее приведен CSS-код для нижнего изображения на рис. 14.8, *1*:

```
h1 {
  position: relative;
  border-bottom: 1px dashed #999999;
}
h1 .date {
  position: absolute;
  bottom: 0;
  right: 0;
  font-size: .5em;
  background-color: #E9E9E9;
  color: black;
  padding: 2px 7px 0 7px;
}
```

Некоторые из приведенных выше свойств, например `border-bottom`, представлены для наглядности. Ключевые свойства выделены полужирным шрифтом: `position`, `bottom` и `right`. Как только вы задаете заголовку относительное позиционирование, можете поместить элемент `span`, содержащий дату, в нижний правый угол заголовка, присвоив свойствам `bottom` и `right` значение `0`.

Исключение элемента за пределы блока

Вы можете также использовать позиционирование для размещения элемента таким образом, чтобы он «выглядывал» из-под другого элемента. На рис. 14.8, *2*, верхнее изображение демонстрирует заголовок с графикой. Так, элемент `img` помещен внутрь элемента `h1` в качестве части заголовка. Использование абсолютного позиционирования и отрицательных значений свойств `top` и `left` позволяет переместить изображение к левому краю заголовка и вытеснить его за верхний и левый края. Рассмотрим CSS-код, который применяется в этом примере:

```
h1 {
  position: relative;
  margin-top: 35px;
  padding-left: 55px;
  border-bottom: 1px dashed #999999;
}
h1 img {
  position: absolute;
  top: -30px;
  left: -30px;
}
```

Основная концепция этого кода такая же, что и в предыдущем примере, но с некоторыми дополнениями. Во-первых, значения свойств `top` и `left` изображения отрицательные, поэтому изображение фактически появляется на расстоянии 30 пикселов над верхним краем заголовка и 30 пикселов левее от левого края заголовка. Будьте внимательны, когда используете отрицательные значения. В результате может получиться так, что элемент будет частично (или полностью) выпадать за пределы страницы или будет перекрывать содержимое другого элемента. Для предотвращения того, чтобы «отрицательно» позиционированный элемент выпадал за

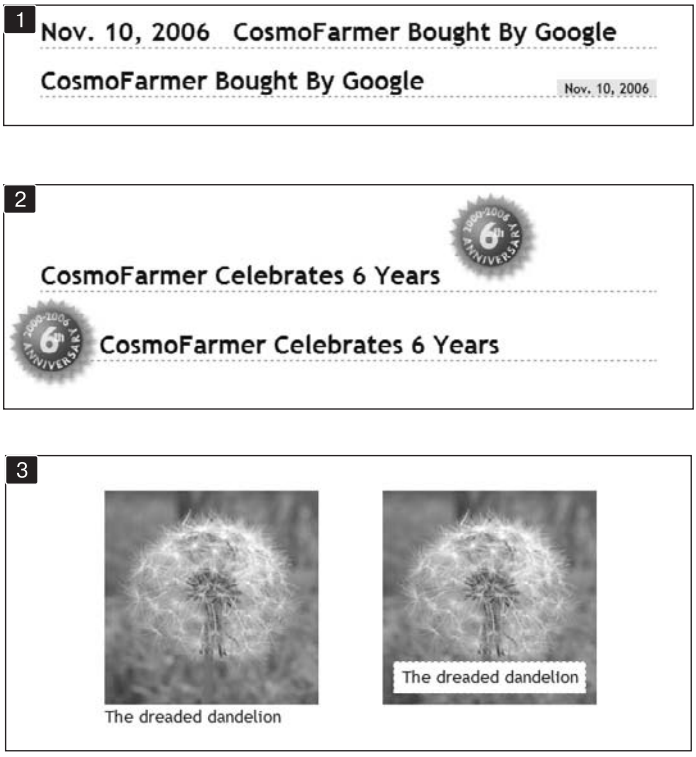

Рис. 14.8. Абсолютное позиционирование хорошо подходит для простых элементов дизайна, например для размещения даты в нижнем правом углу заголовка *(вверху)*, вытеснения изображения из содержащего его блока *(посередине)* или наложения подписи поверх фотографии *(внизу)*

пределы окна браузера, добавьте поля или отступы необходимого размера либо к самому элементу, либо к содержащему его относительно позиционированному элементу — в данном примере `h1`. Дополнительные поля обеспечат достаточно пространства для размещения выпадающего изображения. В данном случае, чтобы предотвратить перекрытие изображением любого контента над заголовком, добавьте верхнее поле. Отступ слева шириной 55 пикселов также сместит текст заголовка из-под абсолютно позиционированного изображения.

Создание CSS-фреймов с фиксированным позиционированием

Поскольку многие веб-страницы длиннее одного экрана (прокрутки), может возникнуть необходимость сохранения на виду некоторых элементов страницы, например навигационной панели, поля поиска или логотипа сайта. HTML-фреймы были когда-то единственным способом «удержать» важные элементы, в то время как другое содержимое прокручивалось и исчезало из области видимости. Однако у HTML-фреймов есть существенные недостатки. Поскольку каждый фрейм

описывается в отдельном файле веб-страницы, приходилось создавать несколько HTML-файлов, чтобы сделать одну полноценную веб-страницу (называемую страницей с *набором фреймов*). Это не только отнимало много времени у разработчиков, но и усложняло поиск по сайту для поисковых машин. HTML-страницы, содержащие набор фреймов, также были неудобны посетителям с недостатками зрения и людям, использующим программы экранного доступа и желающим распечатать страницы сайта.

Тем не менее концепция фреймов все еще жизнеспособна, поэтому каскадные таблицы стилей предлагают вариант позиционирования, который позволит получить вид фреймов с меньшим количеством работы. Страница, созданная с использованием *фиксированного* позиционирования, показана на рис. 14.9. Присваивая значение `fixed` свойству `position`, можно имитировать HTML-фреймы, фиксируя некоторые элементы в желаемых позициях, допуская возможность прокрутки контента длинной веб-страницы. Полоса прокрутки (выделена на рисунке) позволяет прокручивать только большую текстовую область; верхний и нижний баннеры и боковая панель остаются неподвижными.

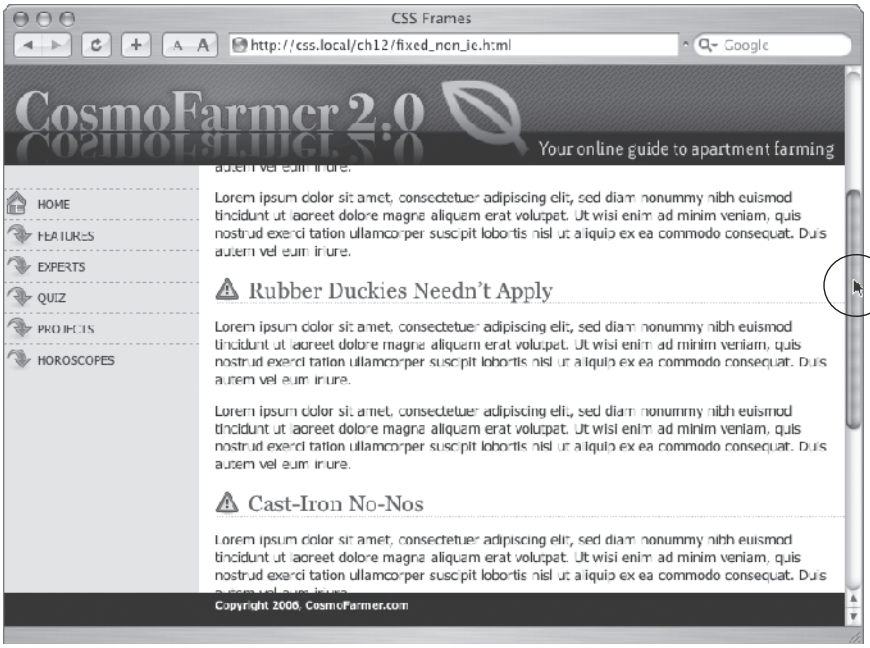

Рис. 14.9. Эффект старых страниц с фреймами, но с намного меньшим объемом кода

Фиксированное позиционирование действует во многом подобно абсолютному — вы точно так же можете использовать свойства `top`, `bottom`, `left` или `right` для размещения элемента. Как и абсолютное, фиксированное позиционирование удаляет элемент из потока HTML. Он «плавает» поверх прочего контента страницы, который игнорирует его.

Рассмотрим, как создать страницу, похожую на изображенную на рис. 14.9, у которой есть фиксированный баннер, боковая панель и нижний колонтитул, а также прокручиваемая область с основным контентом.

1. Добавьте элементы `div` с классами (или идентификаторами) к каждому разделу страницы.

   У вас может быть четыре основных элемента `div` с такими классами (или идентификаторами), как `banner`, `sidebar`, `main` и `footer` (рис. 14.10). Порядок, в котором вы указываете эти элементы в HTML-коде, не имеет значения. Как и абсолютное, фиксированное позиционирование позволяет размещать элементы на странице независимо от их расположения в HTML-коде.

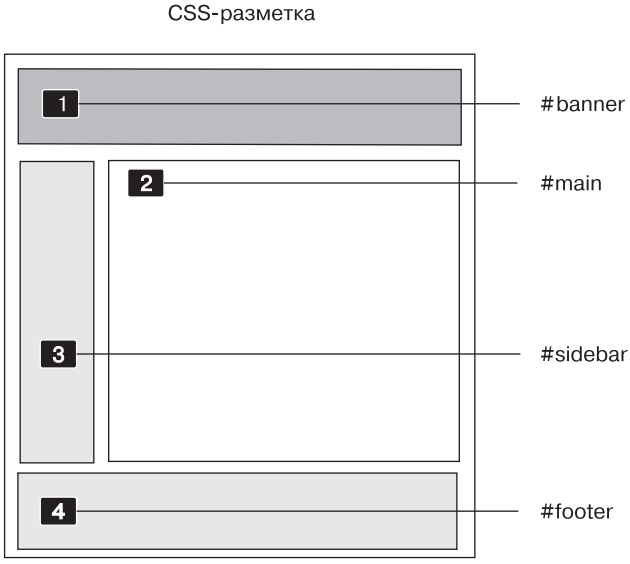

Рис. 14.10. С фиксированным позиционированием можно закрепить все элементы страницы, чтобы они всегда были в поле зрения при прокрутке. В этом примере заголовок *(1)*, боковая панель *(3)* и нижний колонтитул *(4)* зафиксированы, а область с основным контентом *(2)* прокручивается

2. Добавьте контент к каждому контейнеру `div`.

   В основном используйте фиксированные `div`-контейнеры для контента с необходимостью постоянного доступа в областях, которые вы желаете закрепить в конкретных позициях. В этом примере баннер, боковая панель и нижний колонтитул содержат логотип, глобальную навигацию по сайту и сведения об авторском праве. Основной контент помещается в оставшийся элемент `div`.

   Важно отметить: не добавляйте слишком много контента к фиксированному элементу `div`. Если фиксированная боковая панель длиннее, чем окно браузера пользователя, посетитель не сможет увидеть боковую панель целиком. Поскольку фиксированные элементы не прокручиваются, у пользователя не будет никакой возможности (за исключением покупки монитора с большей

диагональю) увидеть содержимое боковой панели, которое не соответствует окну его браузера.

3. Создайте стили для всех фиксированных элементов.

   Значения `top`, `bottom`, `left` и `right` задаются относительно окна браузера, таким образом, определите, где какие элементы вы хотите видеть на экране, и добавьте значения. Кроме того, определите ширину для элементов.

ПРИМЕЧАНИЕ

В отличие от абсолютного, фиксированное позиционирование всегда задается относительно окна браузера, даже если фиксирующийся элемент помещается внутрь другого элемента с относительным или абсолютным позиционированием.

Код стилей для позиционирования элементов *1*, *3* и *4* (на рис. 14.10) выглядит следующим образом:

```
.banner {
  position: fixed;
  left: 0;
  right: 0;
  top: 0;
}
.sidebar {
  position: fixed;
  left: 0;
  top: 110px;
  width: 175px;
}
.footer {
  position: fixed;
  bottom: 0;
  left: 0;
  right: 0;
}
```

4. Создайте стиль для прокручиваемой области контента.

   Поскольку фиксированные элементы удаляются из потока HTML, прочие элементы на странице не учитывают данное обстоятельство. Так, элемент `div` с основным контентом страницы, к примеру, появляется под фиксированными элементами. Основная задача следующего стиля состоит в применении полей для свободного перемещения контента.

```
.main {
  margin-left: 190px;
  margin-top: 110px;
}
```

Фиксированное позиционирование прекрасно поддерживается в Internet Explorer 8 и последующих версиях, а также во всех других основных браузерах (включая самые актуальные версии для мобильных устройств под управлением операционных систем iOS и Android).

Практикум: позиционирование элементов страницы

Этот практикум позволит вам исследовать несколько различных способов применения абсолютного позиционирования, таких как создание трехколоночного макета, позиционирование элементов внутри баннера и добавление подписей поверх фотографий. В отличие от предыдущей главы, где вы заключали фрагменты HTML-кода в элементы `div` и добавляли к ним имена классов, в этих уроках большая часть работы с HTML-кодом уже была выполнена. Вы можете сосредоточиться на оттачивании ваших новых навыков с каскадными таблицами стилей.

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 14.

Улучшение баннера страницы

Во-первых, сделаем несколько маленьких, но визуально важных изменений в баннере страницы. Создадим стили для HTML-элементов с примененными к ним классами (классы уже применены).

1. Запустите браузер и откройте файл `index.html` из папки 14.

   На этой веб-странице (рис. 14.11) вы измените позиции некоторых частей баннера.

2. Откройте файл `styles.css` в редакторе HTML-кода.

   Файл уже содержит основные стили форматирования. Далее вы добавите собственные стили в нижнюю часть файла.

   Начните с перемещения небольшого изображения шляпы в левую сторону баннера. Чтобы избавиться от блочного вида, типичного для CSS-верстки, вытесните изображение за пределы баннера, таким образом сделав его похожим на стикер-арт.

3. В нижней части файла `styles.css` добавьте новый стиль:

```
header .badge {
  position: absolute;
  top: -20px;
  left: -90px;
}
```

   Изображение находится внутри HTML5-элемента `header` с классом `badge`. Только что созданный стиль приводит к тому, что левый верхний угол изображения помещается на 90 пикселов влево и 20 пикселов над верхним краем страницы.

   Теперь просмотрите страницу, и вы увидите, что на ней есть несколько проблем. Во-первых, изображение «нависает» над краем страницы, а вы хотите, чтобы она располагалась над краем области баннера. Займемся этой проблемой.

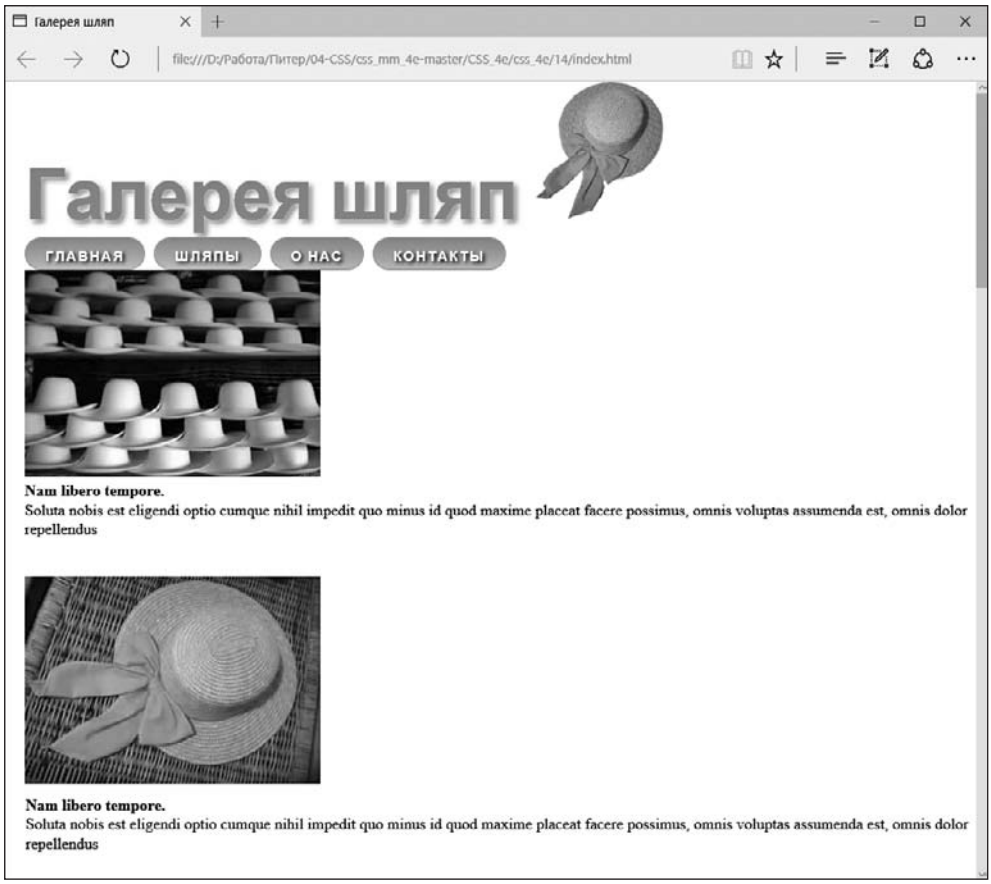

Рис. 14.11. На странице находится много элементов — логотип, баннер, панель навигации и галерея изображений с подписями, — при этом визуальная структура отсутствует. Это обычная статичная HTML-страница с потоком контента сверху вниз. Вы можете сделать ее более удобной для чтения и привлекательной, применив абсолютное позиционирование

4. Добавьте перед только что созданным стилем следующий код:

```
header {
  position: relative;
}
```

Код стилей, которые управляют основным разделом страницы (как этот стиль `header`), принято помещать выше кода стилей, форматирующих фрагменты этого раздела (как стиль, который мы создали в шаге 3). Кроме того, группирование стилей для связанных разделов упрощает их поиск, когда нужно проанализировать или отредактировать CSS-код страницы. В этом случае стиль `header` указывается первым во внутренней таблице стилей, потому что применяется к большему фрагменту HTML-кода. Но вы должны указать стиль `header .badge` рядом с ним, так как добавляете дополнительные стили на страницу (подробнее

о способах организации CSS-кода на странице можно прочитать в разделе «Организация стилей» главы 18).

Стиль header создает новую связь позиционирования для любых вложенных элементов. Другими словами, значение relative функционирует так, что любые другие элементы внутри этого элемента позиционируются относительно краев баннера. Это изменение в позиционировании меняет размещение элемента со стилем, который вы создали в шаге 3. Теперь он смещен на 20 пикселов вверх и на 90 пикселов влево от области баннера. Изображение все еще немного «нависает» над страницей, поэтому нужно добавить небольшие поля для заголовка, чтобы немного его опустить.

5. Отредактируйте стиль header, добавив в его нижнюю часть две строки кода, выделенные полужирным шрифтом:

```
header {
  position: relative;
  margin-top: 20px;
  padding: 20px 0 0 10px;
}
```

Свойство margin-top добавляет пространство над элементом header, достаточное, чтобы он и изображение переместились вниз. Кроме того, отступ добавляет пространство внутри элемента заголовка, чтобы заголовок (и расположенная рядом навигационная панель) не смотрелись слишком скученно. Но теперь появилась другая проблема — заголовок частично скрыт под значком. Перекрытие элементов — один из недостатков абсолютного позиционирования. В данном случае проблему можно решить путем добавления свойства z-index изображения и помещения его позади текста.

6. Добавьте в стиль header .badge свойство z-index: -1;:

```
header .badge {
  position: absolute;
  top: -20px;
  left: -90px;
  z-index: -1;
}
```

Значение -1 приводит к тому, что элемент с абсолютным позиционированием помещается позади своего родительского элемента, в данном случае позади текста (рис. 14.12). Затем абсолютным позиционированием нужно воспользоваться для смещения панели навигации в правую часть элемента header.

7. Добавьте после стиля header .badge следующий код:

```
header nav {
  position: absolute;
  right: 0;
  top: 45px;
}
```

Хотя задать позицию панели навигации можно за счет выравнивания элемента h1, в данном случае намного проще будет воспользоваться абсолютным

позиционированием. Здесь создается стиль для HTML-элемента `nav`, когда он находится в элементе `header`. Следует помнить, что в шаге 4 вы задали для элемента `header` относительную позицию, следовательно, любые элементы внутри, например `nav`, позиционируются относительно него. Поэтому нулевое значение, присвоенное свойству `right`, в этом стиле приводит к тому, что правый край панели навигации помещается по правому краю баннера (см. рис. 14.12).

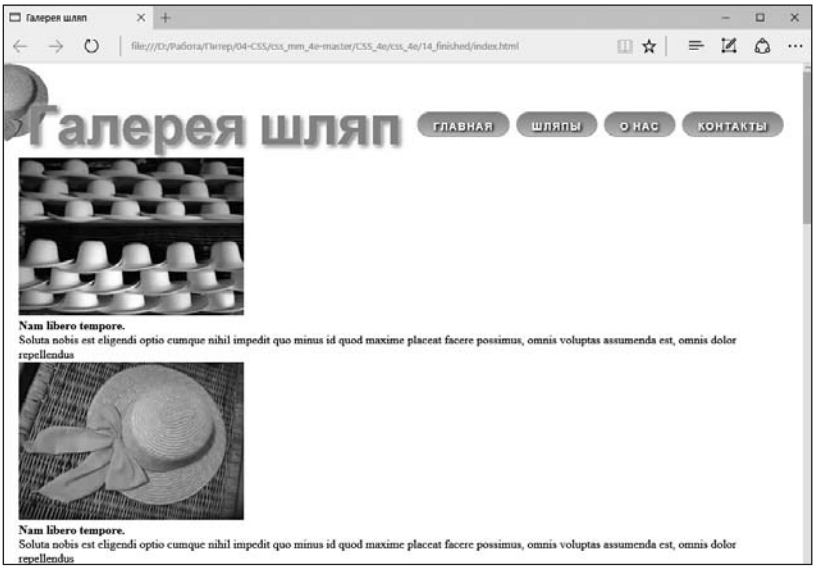

Рис. 14.12. Абсолютное позиционирование — хорошее подспорье в размещении небольших элементов вроде изображения шляпы и панели навигации. В отличие от обтекаемых элементов, точно указанная позиция изображения и навигационной панели в HTML-коде не имеет значения, предоставляя большую степень гибкости

Добавление подписей к иллюстрациям

В главе 8 мы рассмотрели один из способов добавления подписи к фотографии (см. практикум главы 8). В примерах той главы подписи располагались под фотографиями, чего мы и добиваемся в большинстве случаев. Но может понадобиться добавить подпись непосредственно *поверх* фотографии, как, например, субтитры в телевизионных новостях, которые отображаются в нижней части экрана.

1. Откройте файл `index.html` в редакторе HTML-кода.

   На этой странице находится галерея фотографий. HTML-код одного из изображений имеет следующий вид:

```
<figure>
  <img src="images/file00079963469.jpg" width="300" height="210" alt="Панама">
  <figcaption><strong>Nam libero tempore.</strong> Soluta nobis
    est eligendi optio cumque nihil impedit quo minus id quod
    maxime placeat facere possimus, omnis voluptas assumenda est,
```

```
    omnis dolor repellendus</figcaption>
  </figcaption>
<figure>
```

В этом примере используются HTML-элементы `figure` и `figcaption`. Элемент `figure` по умолчанию блочный, но, поскольку нужно, чтобы изображения располагались друг рядом с другом, начните с преобразования их в строчные элементы.

2. Ниже ранее добавленного стиля `header nav` укажите еще один стиль:

```
.gallery figure {
  display: inline-block;
  width: 300px;
  height: 210px;
  margin: 15px;
  position: relative;
}
```

Код создает селектор потомков, который применяется ко всем элементам `figure`, сгруппированным внутри контейнера `div` с классом `gallery`. Селектор потомков используется потому, что на эту страницу могут добавляться другие элементы `figure`, которые не являются частью галереи и должны быть отформатированы по-другому. Этот селектор потомков предназначен только для тех элементов `figure`, которые нас интересуют в данный момент.

После преобразования блочного элемента `figure` в строчный (`inline-block`) все изображения смогут находиться друг рядом с другом. Значения свойств `width` и `height` соответствуют ширине и высоте изображений. То есть элементы `figure` должны быть достаточного размера для вмещения изображений. Свойство `margin` добавляет небольшое пространство по периметру изображений, чтобы они не сталкивались друг с другом. И наконец, объявление `position: relative` устанавливает новую связь позиционирования, чтобы каждую подпись можно было позиционировать относительно связанного с ней изображения.

Теперь настал черед позиционирования подписей.

3. Ниже только что добавленного стиля разместите следующий код:

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgba(153,153,153,.9);
}
```

Элементы `figcaption` получают абсолютное позиционирование с использованием всех четырех квадрантов позиционирования: `top`, `bottom`, `left` и `right`. По существу, подписи будут распространяться на все изображение, но помещаться немного ниже его верхнего края и немного выше нижнего края (фактически на 15 % ниже и выше). Использование всех четырех настроек означает, что вам не

нужно переживать за настройку ширины или высоты подписей: вместо этого вы оставляете ее на усмотрение браузера.

И наконец, указывается объявление фонового цвета `background-color` с установкой полупрозрачного фона поверх каждого изображения, что означает, что изображения можно видеть сквозь фон подписей.

Теперь займемся форматированием текста.

4. Отредактируйте только что созданный стиль, добавив код, выделенный полужирным шрифтом:

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgba(153,153,153,.9);
  padding: 20px;
  font-family: Titillium, Arial, sans-serif;
  font-weight: 400;
  font-size: .9em;
  color: white;
}
```

Отступы создают небольшую разрядку для текста, а другие свойства задают шрифт, размер и цвет.

Если теперь просмотреть страницу, можно увидеть, что подписи отображаются поверх всех изображений. Далее нужно будет изменить стиль, чтобы подписи появлялись только при нахождении указателя мыши поверх изображения. Начнем с сокрытия подписей.

5. Добавьте в стиль код `opacity: 0;`:

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgba(153,153,153,.9);
  padding: 20px;
  font-family: Titillium, Arial, sans-serif;
  font-weight: 400;
  font-size: .9em;
  color: white;
  opacity: 0;
}
```

Присвоение свойству `opacity` значения `0` полностью скрывает подпись. Для сокрытия подписей можно также воспользоваться объявлением `display: none;` или `visibility: hidden;`, но выбранный способ позволяет анимировать значение не-

прозрачности с помощью CSS-перехода, и этот эффект будет вскоре добавлен. Но сначала нужно добавить состояние :hover, чтобы при помещении указателя мыши над изображением появлялась соответствующая подпись.

6. Добавьте в таблицу стилей следующий код:

```
.gallery figure:hover figcaption {
  opacity: 1;
}
```

Этот хитрый фрагмент CSS-кода можно расшифровать как «при установке указателя мыши поверх элемента figure (figure:hover) внутри элемента с классом gallery (.gallery) установить уровень непрозрачности подписи равным 1». То есть при установке указателя мыши поверх элемента figure его элемент-потомок figcaption становится видимым.

Сохраните страницу и посмотрите, что получилось. Когда указатель мыши помещается поверх изображения, должна появляться подпись. Мы можем анимировать этот эффект, добавив в стиль .gallery figcaption переход.

7. Отредактируйте стиль .gallery figcaption, чтобы он приобрел следующий вид (изменения выделены полужирным шрифтом):

```
.gallery figcaption {
  position: absolute;
  top: 15%;
  bottom: 15%;
  left: 0;
  right: 0;
  background-color: rgba(153,153,153,.9);
  padding: 20px;
  font-family: Titillium, Arial, sans-serif;
  font-weight: 400;
  font-size: .9em;
  color: white;
  opacity: 0;
  transition: opacity .75s ease-out;
}
```

Вы добавили свойство transition. Стоит отметить, что Internet Explorer 9 и более ранние версии этого браузера не поддерживают переходы, но здесь это не вызовет проблем, подписи все равно будут появляться в этом браузере. Они будут моментально возникать и исчезать, а не постепенно становиться видимыми и затухать.

И наконец, нужно привязать сведения об авторских правах к нижней части окна браузера, используя фиксированное позиционирование.

8. Добавьте в таблицу стилей следующий код:

```
footer {
  position: fixed;
  bottom: 0;
  left: 0;
```

```
  right: 0;
  padding: 10px;
  background-color: black;
  color: white;
}
```

Как уже говорилось, фиксированное позиционирование позволяет «привязать» элемент к определенной позиции в окне браузера. В данном случае элемент привязывается к нижней части страницы (`bottom: 0`) и расширяется на всю страницу (благодаря объявлениям `left: 0` и `right: 0`). Последние три объявления добавляют пространство по периметру нижнего колонтитула, а также устанавливают ему черный фон и белый текст.

9. Сохраните файл `styles.css` и просмотрите страницу `index.html` в браузере.

   В окончательном виде страница должна выглядеть так, как показано на рис. 14.13. Полную версию этого урока можно найти в папке 14_finished.

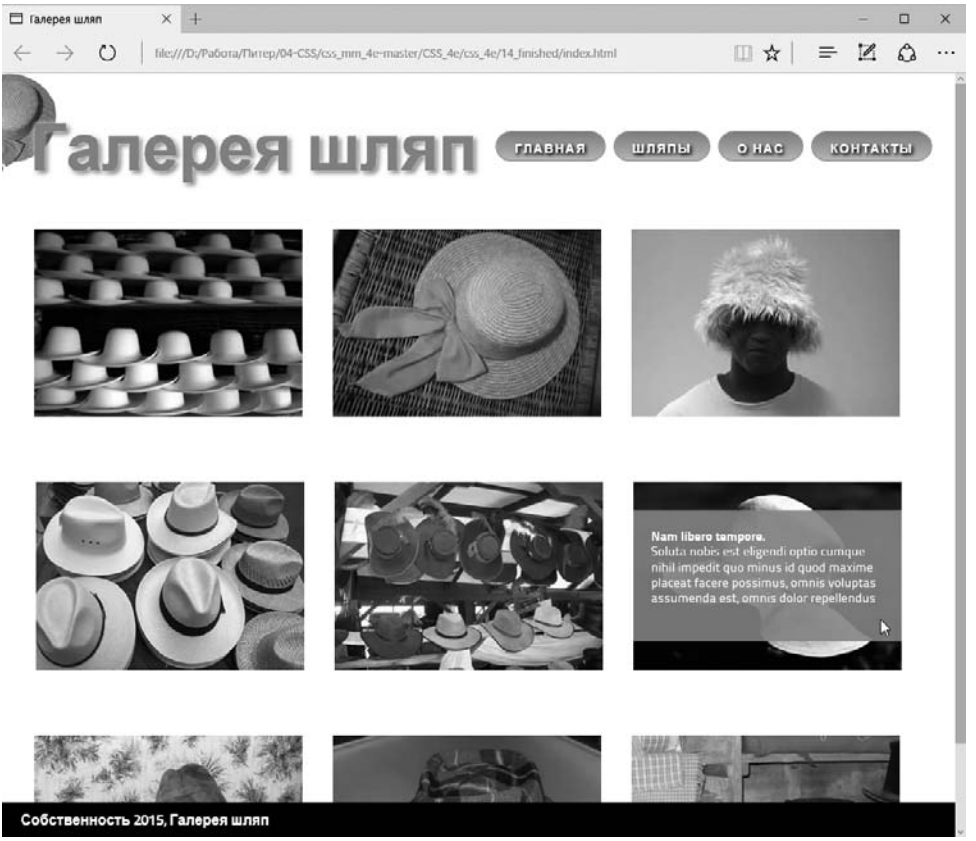

Рис. 14.13. Используя искусный CSS-код, можно без особого труда заставить подписи плавно появляться при наведении указателя на изображение