2 ПЕРЕХОДЫ В CSS

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

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

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

За последнее десятилетие мы прошли долгий путь в отношении взаимодействия и визуальных эффектов на веб-сайтах.

Исторически сложилось так, что анимации, движение и взаимодействие создавались такими технологиями, как JavaScript и Flash. Но в последнее время, когда в браузерах появляется поддержка CSS-переходов и трансформаций, часть анимаций и улучшение взаимодействия могут быть перенесены в таблицы стилей.

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

Как упоминалось в первой главе, мы можем начать использовать CSS3 прямо сейчас при условии, что мы аккуратно выбираем те ситуации, где применяем новые свойства. То же самое справедливо и для CSS-переходов. Они определенно не заменят существующие технологии — Flash, Javascript или SVG (особенно без более широкой поддержки браузерами), — но в сочетании с ранее упомянутыми основными свойствами CSS3 (а также трансформациями и анимациями, о которых будет рассказано далее) ими можно пользоваться, чтобы сдвинуть взаимодействие немного вперед. Что самое важное, пользоваться ими сравнительно легко для того, кто уже знаком с CSS. Переход на CSS занимает лишь несколько строк кода.

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

ХВОСТ, КОТОРЫЙ РАЗМАХИВАЕТ СОБАКОЙ

Изначально разработанные исключительно командой, работавшей над движком WebKit для Safari, CSS-переходы

теперь стали спецификацией в состоянии рабочий черновик в W3C. (У CSS-трансформаций и CSS-анимаций похожее происхождение; о них мы поговорим в главах 4 и 6 соответственно.)

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

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

ЧТО ТАКОЕ CSS-ПЕРЕХОДЫ

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

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

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

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

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

ПРОСТОЙ ПРИМЕР

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

Разметка состоит исключительно из одной ссылки:

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

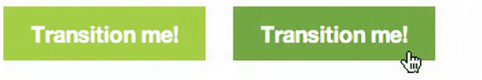

Рис. 2.01. Обычное состояние ссылки и :hover

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

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

```
a.foo:hover {
  background: #690;
  }
```

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

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

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

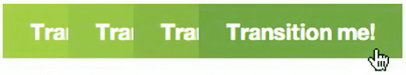

Рис. 2.02. Печатная страница — не лучший способ показать анимированный переход, но на этой картинке попытались сделать именно это: плавный переход от светло-зеленого к темно-зеленому фону

```
a.foo:hover {
  background: #690;
  }
```

В этом коде можно увидеть три составляющих перехода:

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

- `transition-duration` — продолжительность перехода (0,3 с);
- `transition-timing-function` — как быстро переход осуществляется с течением времени (ease).

ВРЕМЕННЫЕ ФУНКЦИИ (МНЕ СЛЕДОВАЛО БЫТЬ ВНИМАТЕЛЬНЕЕ НА УРОКАХ МАТЕМАТИКИ)

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

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

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

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

ЗАДЕРЖКА ПЕРЕХОДА

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

```
a.foo {
  padding: 5px 10px;
```

```
  background: #9c3;
  -webkit-transition-property: background;
  -webkit-transition-duration: 0.3s;
  -webkit-transition-timing-function: ease;
  -webkit-transition-delay: 0.5s;
  }

a.foo:hover {
  background: #690;
  }
```

КРАТКАЯ ФОРМА ЗАПИСИ

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

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

a.foo:hover {
  background: #690;
  }
```

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

Краткая форма записи перехода с задержкой

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

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

a.foo:hover {
  background: #690;
  }
```

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

ПОДДЕРЖКА В БРАУЗЕРАХ

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

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

ПОЛНАЯ ЗАПИСЬ ПЕРЕХОДА

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

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

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

свойство перейдет из состояния черновика в окончательную версию спецификации.

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

a.foo:hover {
  background: #690;
  }
```

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

СОСТОЯНИЯ ПЕРЕХОДА

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

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

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

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

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

a.foo:hover,
a.foo:focus {
  background: #690;
  }
```

ПЕРЕХОД НЕСКОЛЬКИХ СВОЙСТВ

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

```
a.foo {
  padding: 5px 10px;
  background: #9c3;
  -webkit-transition: background .3s ease, »
    color 0.2s linear;
  -moz-transition: background .3s ease, »
```

```
    color 0.2s linear;
  -o-transition: background .3s ease, color 0.2s linear;
  transition: background .3s ease, color 0.2s linear;
  }

a.foo:hover,
a.foo:focus {
  color: #030;
  background: #690;
  }
```

ПЕРЕХОД ВСЕХ ВОЗМОЖНЫХ СОСТОЯНИЙ

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

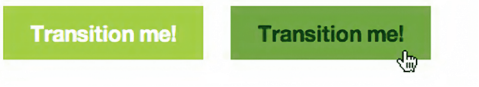

Рис. 2.03. Обычное состояние ссылки и состояние :hover

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

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

```
a.foo:hover,
a.foo:focus {
  color: #030;
  background: #690;
  }
```

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

К КАКИМ СВОЙСТВАМ ПРИМЕНИМ ПЕРЕХОД

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

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

ПОЧЕМУ БЫ НЕ ВОСПОЛЬЗОВАТЬСЯ JAVASCRIPT?

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

Все зависит от того, насколько важны анимации. В книге я делаю акцент на том, что можно извлечь максимум из простоты и гибкости CSS3, выбирая подходящие части интерфейса,

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

к которым можно применять новые свойства и улучшать взаимодействие пользователя с сайтом. Довольно часто анимации, сделанные средствами CSS-перехода, не относятся к ключевым составляющим сайта — таким как бренд, читаемость или расположение информационных блоков. Поэтому использовать несколько простых строчек на CSS, которые порождают простую анимацию естественными средствами браузера вместо того, чтобы привлекать JavaScript-фреймворк, кажется разумным. Очень хорошо, что он есть в нашем распоряжении.

ИСПОЛЬЗУЙТЕ С УМОМ

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

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

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

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