4 Встроенное аудио

Эмили Льюис

4.0. Введение

Почти каждый день мы имеем дело со *встроенным контентом* — информацией, которая импортируется или вставляется на веб-страницу (http://www.w3.org/TR/html5/content-models.html#embedded-content-0). Вспомните хотя бы элемент **img**, который с помощью атрибута **src** добавляет на веб-страницу изображения.

В HTML5 появилось гораздо больше вариантов встроенного контента, в том числе аудио, добавляемого с помощью нового элемента **audio** (http://dev.w3.org/html5/markup/audio.html).

Встроенные объекты? Да. Больше не будет неудобных элементов **object** и **embed**. Не придется вставлять аудио с помощью сторонних плагинов и не будет головной боли, связанной с разработкой динамических макетов или раскрывающихся меню.

Имея в арсенале элемент **audio**, вы сможете не только выкладывать аудиофайлы непосредственно в браузере, но и управлять стилем и атрибутами этого элемента с помощью CSS и JavaScript.

4.1. Добавление HTML5-аудио

Проблема

Необходимо воспроизводить на веб-странице встроенное аудио.

Решение

Добавьте элемент **audio** с атрибутом **src**, ссылающимся на аудиофайл, а также укажите резервный контент для старых браузеров:

```
<audio src="audio.ogg" controls>
    Скачать <a href="audio.ogg">42 эпизод "Учимся любить HTML5"</a>
</audio>
```

Если вы хотите, чтобы браузеры по умолчанию отображали интерфейс управления аудио, не забудьте добавить атрибут **controls** (рис. 4.1):

```
<audio src="audio.ogg" controls>
```

Аудиофайл в примере использует Ogg Vorbis (формат OGG) — бесплатный кодек с открытым исходным кодом (http://www.vorbis.com). Тем не менее существуют и другие аудиоформаты для Сети (табл. 4.1), и это одна из самых больших проблем реализации HTML5-аудио.

Кодек — это технология сжатия и распаковки данных. Аудиокодеки, используемые для сжатия и/или распаковки цифровых аудиоданных в различные форматы, позволяют сохранить высокое качество звука при минимальной скорости передачи информации.

Аудиокодеки

Спецификация HTML5 не дает четких рекомендаций по поводу использования конкретных аудиокодеков. Создателям браузеров было бы слишком трудно учесть все (http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2009-June/020620.html). Как видно из табл. 4.1, нет ни одного формата, который работает во всех браузерах.

Таблица 4.1. Текущая поддержка браузерами HTML5-аудиоформатов

Браузер	AAC (.aac)	MP3 (.mp3)	Ogg Vorbis (.ogg)	WAV (.wav)	WebM (.webm)
Chrome 6+		×	×		×
Firefox 3.6+			×	×	×
IE9+	×	×		×	
Opera 10.5+			×	×	
Safari 5+	×	×		×	

Рис. 4.1. По умолчанию в Chrome 9 панель управления аудио включает кнопку Воспроизведение/Пауза, индикатор выполнения и кнопку Регулировка/Отключение звука

Познакомимся с этими форматами поближе:

- AAC — формат сжатия с потерями, разработанный лучше, чем MP3, имеющий ту же скорость, но более высокое качество звука;
- MP3 — популярный и уже запатентованный формат, использующий сжатие с потерями и позволяющий уменьшить размер файла в десять раз;
- OGG — открытый альтернативный ресурс; как и MP3, использует формат сжатия с потерями;
- WAV — проприетарный аудиоформат без какого-либо сжатия;
- WEBM — открытый бесплатный медиаформат Google, основанный на аудиокодеке Vorbis.

Объединение нескольких источников

При выборе между форматами реальность такова, что, если вам необходимо обеспечить доступность контента для максимально широкой аудитории, придется сжать и включить в элемент **audio** *несколько* звуковых файлов. К счастью, HTML5 предоставляет такую возможность.

При использовании основного элемента audio атрибут src не нужен. Он необходим, только если вы ссылаетесь на один аудиоформат.

Оптимальный вариант — как минимум добавить файл в бесплатном формате OGG, *а также* в формате MP3 или WAV. Такой подход должен обеспечить поддержку аудио последними браузерами:

```
<audio controls>
    <source src="audio.ogg">
    <source src="audio.mp3">
    Скачать <a href="audio.ogg">42 эпизод "Учимся любить HTML5"</a>
</audio>
```

Предварительная загрузка аудио

Элемент **audio** имеет несколько атрибутов, позволяющих настроить воспроизведение звука.

Полное описание атрибутов, доступных для медиаэлементов HTML5, вы найдете в стандарте WHATWG: http://www.whatwg.org/specs/web-apps/current-work/multipage/video.html#media-element-attributes.

Атрибут **preload** сообщает браузеру, когда необходимо начать буферизацию аудио:

```
<audio controls preload>
```

Несмотря на то что **preload** частично поддерживается браузерами, его использование может пригодиться для оптимизации процесса загрузки. Достаточно указать этот атрибут и предоставить браузеру возможность самому выполнить соответствующие действия или указать один из трех вариантов.

- **preload="auto"** — браузер должен начать загрузку файла, но выбор остается за ним. При использовании мобильного устройства или при медленном соединении браузер может принять решение не начинать предварительную загрузку, чтобы сэкономить трафик.
- **preload="metadata"** — браузер самостоятельно не выполняет буферизацию аудио, пока пользователь не активизирует элементы управления. Но программа-обозреватель предварительно загружает метаданные, например треки и их продолжительность.
- **preload="none"** — аудио не будет загружаться, пока пользователь не активизирует элементы управления.

Обсуждение

Вдобавок к несоответствию форматов для разных браузеров сама поддержка аудио осуществляется несколько непоследовательно. У каждого браузера встречаются ошибки, причуды и странности, которые в ближайшем будущем, надеюсь, будут устранены производителями, но пока не стоит о них забывать.

На сайте 24Ways приводится список некоторых проблем браузеров: http://24ways.org/2010/the-state-of-html5-audio.

Создание резервного контента

Как видно из первого примера этого рецепта, элемент **audio** может включать *резервный контент*. Иными словами, если браузер не поддерживает HTML5-элемент **audio**, то пользователь увидит несколько видоизмененную информацию (рис. 4.2).

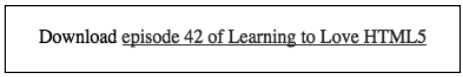

Рис. 4.2. Так отображается резервный контент в IE8, где отсутствует поддержка HTML5-элемента audio

Спецификация HTML5 говорит, что все дочерние элементы **audio**, кроме **source**, должны быть проигнорированы. Это означает, что для пользователей HTML5-совместимых браузеров предоставление резервного контента не приведет к негативным последствиям.

К примеру, можно добавить резервный Flash-проигрыватель:

```
<audio controls>
    <source src="audio.ogg">
    <source src="audio.mp3">
    <object data="player.swf?audio=audio.mp3">
        <param name="movie" value="player.swf?audio=audio.mp3">
        Видео и Flash не поддерживаются вашим браузером.
    </object>
</audio>
```

Или просто сообщить пользователю, что аудиофайл доступен для скачивания и может быть воспроизведен с помощью медиапроигрывателя (и наряду с этим ненавязчиво предложить перейти на новейший браузер):

```
<audio controls>
    <source src="audio.ogg">
    <source src="audio.mp3">
    Ваш браузер не поддерживает HTML5-аудио. Вам следует обновить его.
    Пока вы можете скачать <a href="audio.ogg">42 эпизод "Учимся любить
    HTML5"</a>.
</audio>
```

Доступные альтернативы

Еще одна проблема HTML5-аудио состоит в том, что для мультимедиа пока нет практической реализации альтернативного контента. *Теоретически*, доступ может открываться в два этапа: сначала авторы мультимедиа добавляют файл субтитров в контейнер формата OGV или MP3, а затем браузеры предоставляют пользователям интерфейс для доступа к этим субтитрам и подписям.

Использование атрибута alt для img нельзя назвать удачным решением. Этот вариант не предполагается в спецификации HTML5, и, что более важно, вспомогательные технологии не обрабатывают резервный контент audio подобным образом.

На данный момент можно рассмотреть несколько экспериментальных подходов:

- HTML5-видео с подписями, созданными в JavaScript: http://dev.opera.com/articles/view/accessible-html5-video-with-javascripted-captions/;
- HTML5 и временное медиа: http://www.bbc.co.uk/blogs/rad/2009/08/html5.html;
- демонстрация HTML5-аудио и видео: http://www.annodex.net/~silvia/itext/.

Как вы заметили, некоторые из этих справочных ресурсов посвящены видео. Это произошло потому, что HTML5-аудио и видео настолько похожи, что и применяемые к ним подходы могут быть одинаково описаны (подробнее см. в главе 5).

У **audio** есть проблемы не только с субтитрами. Некоторые программы чтения с экрана вообще не признают этот элемент и просто пропускают его.

Кроме того, у браузеров нет единой поддержки работы **audio** через клавиатуру. В этом случае для HTML5 рекомендуется выполнить полное обновление: http://html5accessibility.com.

Право интеллектуальной собственности

Возможно, вы уже поняли, что HTML5-элемент **audio** — не универсальное решение. Но это связано не только с необходимостью хранения аудио в различных форматах или с отсутствием поддержки браузерами — проблема еще в том, что HTML5 не обеспечивает защиту от копирования.

Содержимое элемента **audio** (и **video**, см. в главе 5) можно легко сохранить на жестком диске пользователя подобно **img**, и, следовательно, такое решение иногда неприемлемо. Если необходимо предусмотреть технические средства защиты авторских прав, то лучшим решением сегодня будет использование плагинов, а не **audio**.

Дополнительная информация

Для получения более подробной информации о доступности HTML5-мультимедиа W3C предлагает глубже рассмотреть известные проблемы и возможные решения: http://www.w3.org/html/wg/wiki/MultimediaAccessibilty.

4.2. Управление аудиопотоком

Проблема

Необходимо иметь возможность управлять воспроизведением HTML5-аудио в браузере.

Решение

В спецификации для элемента **audio** предусмотрено несколько атрибутов, позволяющих просто и быстро контролировать воспроизведение звука.

- **autoplay** — сообщает браузеру о том, что необходимо начать воспроизведение аудио, как только будет загружена страница.

Я не решаюсь даже упоминать этот атрибут, поскольку он делает именно то, что больше всего раздражает на сайтах (http://www.punkchip.com/2009/04/autoplay-is-bad-for-all-users). Таким образом, пока autoplay все еще существует, не используйте его. Серьезно. Не надо.

- **loop** — еще один описательный атрибут. Он указывает браузеру циклично воспроизводить аудио.

Он не так ужасен, как атрибут autoplay, но все же применяйте его с осторожностью.

Как и **controls**, атрибуты **autoplay** и **loop** — логические, поэтому при необходимости достаточно включить их в открывающий тег **audio**:

```
<audio controls loop>
```

Обсуждение

Что делать, если нужно получить больше возможностей управления, чем могут предоставить основные атрибуты? К счастью, у **audio** и **video** есть атрибуты, события и методы для работы с JavaScript, позволяющие создавать пользовательские элементы управления, в том числе:

- **canplaytype(*mun*)** — возвращает строку, указывающую, может ли браузер воспроизводить данный тип медиафайла;
- **currentTime** — определяет текущую позицию воспроизведения, заданную в секундах;
- **duration** — содержит длину аудиофайла в секундах;
- **play();** — начинает воспроизведение с текущей позиции;
- **pause();** — приостанавливает воспроизведение аудио, если оно проигрывается.

Предположим, вам необходимо добавить элементы управления, позволяющие пользователю перейти к определенному моменту аудиофайла. Вы можете добавить эту функцию, разместив кнопку и написав JavaScript-код для управления методом **play()** на основе чтения/записи свойства **currentTime**:

```
<audio>
    <source src="audio.ogg">
    <source src="audio.mp3">
</audio>
<button title="Воспроизвести с 30 секунды" onclick="playAt(30);">
30 секунда</button>
<script>
    function playAt(seconds){
        var audio = document.getElementsByTagName("audio")[0];
        audio.currentTime = seconds;
        audio.play();
    }
</script>
```

Метода **stop** не существует, но можно имитировать его функциональность, применив подход из предыдущего примера. Для этого введем метод **pause()**, чтобы вернуться к началу аудиофайла, используя **currentTime**:

```
<audio>
    <source src="audio.ogg">
    <source src="audio.mp3">
</audio>
<button title="Воспроизвести с 30 секунды" onclick="playAt(30);">
30 секунда</button>
<button title="Остановить воспроизведение" onclick="stopAudio();">
Стоп</button>
<script>
    function playAt(seconds){
        var audio = document.getElementsByTagName("audio")[0];
        audio.currentTime = seconds;
        audio.play();
    }
    function stopAudio(){
        var audio = document.getElementsByTagName("audio")[0];
        audio.currentTime = 0;
        audio.pause();
    }
</script>
```

При создании собственных пользовательских элементов управления не используйте для audio логический атрибут controls.

Для получения дополнительной информации о создании пользовательских элементов управления обратитесь к рецепту 4.5.

Дополнительная информация

Для того чтобы иметь минимальное представление о воспроизведении звука с пользовательскими элементами управления для Opera, прочитайте статью

«Все, что нужно знать о HTML5-видео и аудио»: http://dev.opera.com/articles/view/everything-you-need-to-know-about-html5-video-and-audio/.

4.3. Создание <audio> с помощью JavaScript

Проблема

Необходимо в реальном режиме времени создавать аудио на веб-странице.

Решение

С помощью методов, определенных Mozilla Audio Data API, можно создавать в браузере аудио без атрибутов **src** или элементов **source** (https://wiki.mozilla.org/Audio_Data_API#Writing_Audio):

- **mozSetup(***каналы***,** *частоты_дискретизации***)** — определяет каналы и частоты дискретизации для созданного звукового потока;
- **mozWriteAudio(***буфер***)** — записывает фрагменты из созданного для аудио массива;
- **mozCurrentSampleOffset()** — возвращает текущую позицию воспроизведения аудио, заданную во фрагментах.

Обсуждение

Такая реализация **audio** возможна далеко не во всех браузерах. В настоящее время этот элемент поддерживают только Firefox 4+ и Chrome Beta. Это скорее экспериментальный подход, нежели решение, предназначенное для основного использования.

Если вам хочется ознакомиться с экспериментальными типами, посмотрите короткую видеопрезентацию о возможностях Mozilla Audio Data API: http://www.youtube.com/watch?v=1Uw0CrQdYYg.

Дополнительная информация

На Barcamp London 8 приводится статья «jasmid — MIDI, соединенное с JavaScript и HTML5-аудио», рассматриваются проблемы и практические последствия динамического создания аудио в браузере: http://matt.west.co.tt/music/jasmid-midi-synthesis-with-javascript-and-html5-audio/.