Управление графикой, аудио, видео и мультимедиа с помощью JavaScript.

HTML5 предоставляет конкурирующий, открытый стандарт для доставки мультимедиа на Web с его родными видео и аудио элементами и API.

Очевидным компаньоном для элемента <video> является элемент <audio>; они имеют много похожих функций, поэтому мы обсуждаем их вместе и отметим только различия.

Раньше, когда разработчики хотели включить видео на веб-страницу, им приходилось использовать элемент <object>, который является общим контейнером для «посторонних объектов». Из-за несоответствий браузера им также необходимо было использовать ранее недействительные <embed> и дублировать многие параметры. Это привело к тому, что код выглядел так:

```
1 <object width="425" height="344">
2 <param name="movie" value="http://www.youtube.com/
Âv/9sEI1AUFJKw&hl= en_GB&fs=1&" />
3 <param name="allowFullScreen" value="true" />
4 <param name="allowscriptaccess" value="always" />
5 <embed src="http://www.youtube.com/ Âv/9sEI1AUFJKw&hl=
en_GB&fs=1&" type="application/x-shockwave-flash"
allowscriptaccess="always"
allowfullscreen="true" width="425" height="344">
6 </embed>
7 </object>
```

В этом коде браузер должен передать видео в сторонний плагин; при условии, что у пользователя есть правильная версия этого плагина (или имеет права на его загрузку и установку, или соответствующие знания); а затем плагин должен быть доступен для клавиатуры - наряду со всеми другими неизвестными, связанными с передачей содержимого стороннему приложению.

Плагины также могут быть важной причиной нестабильности браузера и могут создавать проблемы у менее технических пользователей, когда им будет предложено загрузить и установить более новые версии.

Всякий раз, когда вы включаете плагин на своих страницах, вы резервируете определенную область рисования, которую браузер делегирует плагину. Что касается браузера, область плагина остается черным ящиком - браузер не обрабатывает и не интерпретирует все, что там происходит.

Проблемы и причуды также могут возникать, если ваша страница имеет динамические изменения макета. Если размеры области рисования плагина изменяются, это может иногда иметь непредвиденные эффекты - фильм, воспроизводимый в плагине, может не изменяться, а вместо этого просто обрезается или отображается лишнее свободное пространство. HTML5 предоставляет стандартизованный способ воспроизведения видео непосредственно в браузере без необходимости использования плагинов.

Одним из основных преимуществ элемента видео HTML5 является то, что, наконец, видео является полноправным гражданином в Интернете. Он больше не отключается от внутренних областей <object> или неаудирующего элемента <embed>.

Итак, элементы ***<video>*** могут быть стилизованы с помощью CSS; Например, они могут быть изменены при наведении с использованием переходов CSS. Их можно настроить и перерисовать на <canvas> с помощью JavaScript. Лучше всего, открыта врожденная взломанность, открывающая веб-стандарты. Раньше все ваши видеоданные были заблокированы; ваши биты оказались в ловушке в коробке. Благодаря мультимедийным средствам HTML5 ваши биты можно свободно манипулировать, как вы хотите.

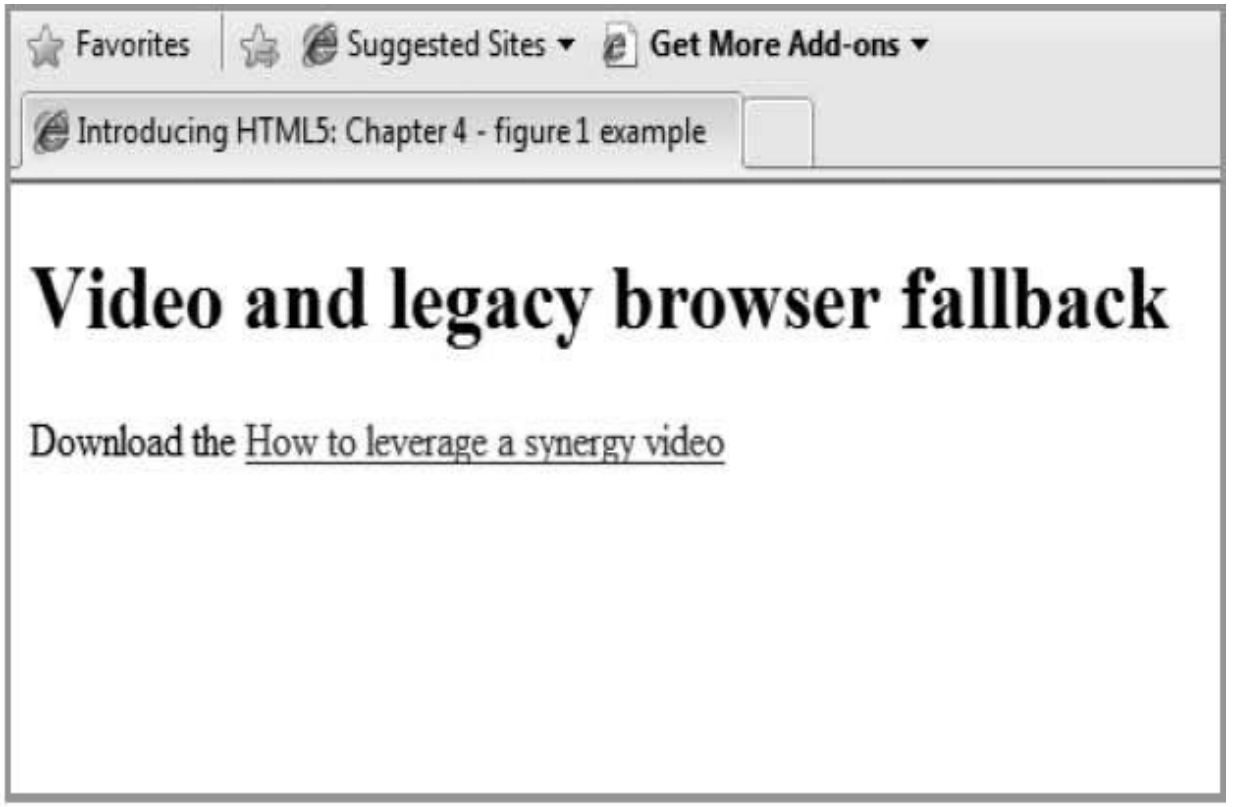
Анатомия видеоэлемента

В своем простейшем виде, включая видео на странице в HTML5, просто требуется этот код:

```
<video src=turkish.ogv></video>
```

Расширение файла .ogv используется здесь, чтобы указать на видео Ogg Theora. Подобно <object>, вы можете поместить резервную разметку между тегами, для более старых веб-браузеров, которые не поддерживают собственное видео. Вы должны по крайней мере предоставить ссылку на видео, чтобы пользователи могли загружать их на свои жесткие диски и смотреть позже на медиаплеере операционной системы.

```
1 <h1>Video and legacy browser fallback</h1>
2 <video src=leverage-a-synergy.ogv>
3 Download the <a href=leverage-a-synergy.ogv>How to leverage a synergy video</a>
4 </video>
```



autoplay

Вы можете сказать браузеру автоматически воспроизводить видео или аудио, но вы почти наверняка не должны этого делать, так как многие пользователи (и особенно те, кто использует вспомогательные технологии, такие как программа чтения с экрана) найдут это очень навязчивым. Пользователи на мобильных устройствах, вероятно, не хотят, чтобы вы использовали свою полосу пропускания, без необходимости явно запрашивать видео. Тем не менее, вот как вы это делаете:

```
<video src=leverage-a-synergy.ogv autoplay> </video>
```

controls

Обеспечение контроля намного лучше, чем автоматическое воспроизведение вашего видео. Вы можете использовать простой JavaScript для написания своих собственных, или вы можете сообщить браузеру о том, чтобы предоставить их автоматически:

```
<video src=leverage-a-synergy.ogv controls> </video>
```

Естественно, они отличаются между браузерами, так же, как, например, с элементами управления формами, но вы не найдете ничего удивительного. Существует переключатель воспроизведения/паузы, панель поиска и регулятор громкости.

Браузеры имеют разный уровень доступности клавиатуры. Собственные элементы управления Firefox не отображаются, когда JavaScript отключен (контекстное меню позволяет пользователю останавливать и запускать фильм, но есть проблема с возможностью обнаружения, и не представляется возможным выбрать эти параметры без JS.) Доступные встроенные элементы управления Opera всегда присутствуют, когда JavaScript отключен, независимо от того, указан ли атрибут управления.

У Chrome и Safari есть проблемы с доступностью клавиатуры. Мы ожидаем повышенную доступность клавиатуры, поскольку производители устраняют проблемы с прорезями.

Обратите внимание, что эти элементы управления отображаются, когда пользователь наводил курсор на видео или когда он записывал видео. Также можно выполнить вкладку с помощью различных элементов управления. Эта привычная доступность на клавиатуре уже улучшает плагины, что может быть сложным для включения в окружающий HTML-контент

Если элемент <audio> имеет атрибут элементов управления, вы увидите их на странице. Без атрибута ничего не визуализируется на странице вообще, но, конечно же, есть в DOM и полностью управляется с помощью JavaScript и новых API.

poster

Атрибут плаката указывает на изображение, которое браузер будет использовать во время загрузки видео, или пока пользователь не покажет видео. (Этот атрибут не применим к <audio>.) Он устраняет необходимость в дополнительных трюках, таких как отображение изображения, а затем удаление его с помощью JavaScript при запуске видео.

Если вы не используете атрибут плаката, в браузере отображается первый кадр фильма, который не может быть изображением представителя, которое вы хотите показать.

height, width

Эти атрибуты сообщают браузеру размер в пикселях видео. (Они не применимы к <audio>.) Если вы их не используете, браузер использует внутреннюю ширину видеоресурса, если таковая имеется. В противном случае это внутренняя ширина рамки плаката, если таковая имеется. В противном случае это 300 пикселей.

Если вы укажете одно значение, но не другое, браузер изменит размер неопределенного измерения, чтобы сохранить соотношение сторон видео.

<video src=leverage-a-synergy.ogv autoplay> </video>

Если вы установите ширину и высоту в пропорции, которая не соответствует таковой для видео, видео не растягивается до этих размеров, а отображается в виде букв внутри элемента видео определенного размера, сохраняя при этом соотношение сторон.

loop

Атрибут loop - это еще один логический атрибут. Как вы могли себе представить, он воспроизводит воспроизведение мультимедиа.

preload

Возможно, вы уверены, что пользователь хочет активировать носитель (например, он просверливается от него, например, или это единственная причина быть на странице), но вы не хотите использовать автозапуск. Если это так, вы можете предположить, что браузер предварительно загрузит видео, чтобы он начал буферизацию, когда страница загружается в ожидании того, что пользователь активирует элементы управления.

<video src=leverage-a-synergy.ogv controls preload> </video>

Существует три специфицированных состояния атрибута предварительной загрузки. Если вы просто укажете preload, пользовательский агент может решить, что делать. Мобильный браузер может, например, по умолчанию не предварительно загружать до тех пор, пока пользователь явно не скажет об этом.

1. preload=auto (or just preload)

Предложение браузеру, что он должен начать загрузку всего файла. Обратите внимание, что мы говорим «предложение». Браузер может игнорировать это - возможно, потому, что обнаружил очень медленное соединение или настройку в мобильном браузере «Никогда не загружайте носитель», чтобы сохранить полосу пропускания пользователя.

2. preload=none

Это состояние предлагает браузеру, что он не должен предварительно загружать ресурс, пока пользователь не активирует элементы управления.

3. preload=meta

Это состояние предлагает браузеру, что он должен просто предварять метаданные (размеры, первый кадр, список дорожек, продолжительность и т. д.), Но что он не должен загружать ничего дальше, пока пользователь не активирует элементы управления.

src

Как и в <img>, этот атрибут указывает на файл, который будет отображаться. Однако, поскольку не все браузеры могут воспроизводить одни и те же форматы, в производственных средах вам необходимо иметь более одного исходного файла. Использование одного исходного файла с атрибутом src действительно полезно только для быстрого прототипирования или для сайтов интрасети, где вы знаете браузер пользователя и какие кодеки он поддерживает.

codecs—the horror, the horror

Ранние черновики спецификации HTML5 предусматривали, что во всех браузерах должна быть хотя бы встроенная поддержка мультимедиа в двух кодеках: Ogg Vorbis для аудио и Ogg Theora для фильмов. Vorbis - это кодек, используемый такими сервисами, как Spotify, среди прочих, а также для образцов аудио в таких играх, как Microsoft Halo, он часто используется с Theora для видео и объединяется вместе в формате контейнера Ogg.

Многократные <source> элементы

Для этого вам необходимо дважды закодировать мультимедиа: один раз в качестве Theora и один раз в качестве H.264 в случае видео, а также в Vorbis и MP3 для аудио.

Затем вы привязываете эти отдельные версии файла к элементу мультимедиа. Вместо использования единственного атрибута src вы вставляете отдельные элементы <source> для каждой кодировки с соответствующими атрибутами типа внутри элемента <audio> или <video> и позволяете браузеру загружать формат, который он может отображать.

Обратите внимание, что в этом случае мы не предоставляем атрибут src в самом медиа-элементе:

```
1 <video controls>
2  <source src=leverage-a-synergy.ogv type=’video/ogg; codecs="theora, vorbis"'>
3 <source src=leverage-a-synergy.mp4 type=’video/mp4; codecs="avc1.42E01E, mp4a.40">
4  <p>Your browser doesn’t support video.
5 Please download the video in <a href=leverage-a-synergy.ogv>Ogg</a> or <a href=leverage-a-synergy.mp4>mp4</a> format.</p>
6 </video>
```