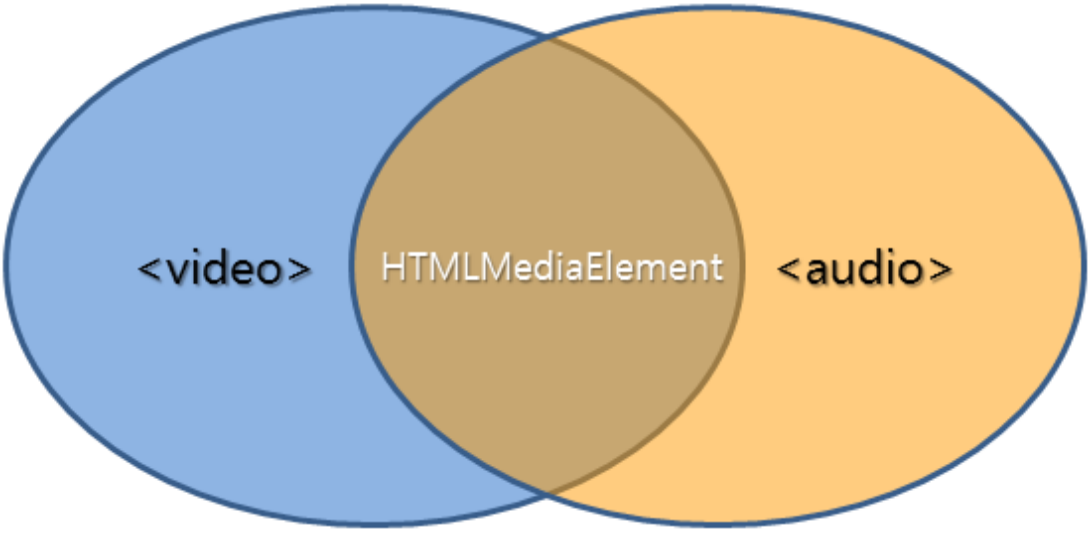
웹 브라우저에서 네이티브(native)로 제공된다는 것은 웹 브라우저가 가진 다른 기술, 즉, 자바스크립트나 CSS 등과도 자연스럽게 융화된다는 뜻이기도 합니다. Adobe Flash 또는 Microsoft Silverlight 와 같은 플러그인 기술들 역시 자바스크립트와 연동되기는 했으나 웹 브라우저나 실행 시점에 따라 종종 오류가 발생했으며 때로는 HTML 문서의 레이아웃을 망치는 원인이 되기도 했었습니다. 하지만, HTML5 <video>에는 플러그인 기술에서 보아왔던 이러한 단점이 거의 해결되었습니다. 자바스크립트와 문제없이 잘 동작하는 것은 물론이고, 별도의 외부 파일을 준비하지 않아도 동영상 플레이어를 구현할 수 있습니다.

HTML5 <video>를 사용한 동영상 플레이어는 이 연재물의 최종 목표이기도 한데, 그를 위해 이번 글에서는 <video> 태그의 속성에 대해서 알아보도록 하겠습니다.

HTMLMediaElement

HTML5에는 미디어(media)를 표시할 수 있는 두 개의 요소가 있습니다. 바로 동영상 데이터를 다루는 <video>와 음성 데이터를 다루는 <audio> 입니다. 두 개의 요소는 '미디어를 다룬다'라는 공통의 성질 덕분에 공통적인 속성을 가집니다. 이러한 공통의 성질은 HTML5 스펙에서 HTMLMediaElement라는 이름으로 정의되어 있고, <video> 태그 역시 여기서 정의된 속성과 메소드를 그대로 따릅니다(조금 더 개발자스러운 언어로 표현하면 HTMLMediaElement 인터페이스를 상속한다고 표현합니다).

앞의 설명을 간단한 그림으로 풀어보면 다음과 같이 표현할 수 있습니다.



HTML5 스펙에서는 미디어 요소들의 공통 속성인 HTMLMediaElement를 <video> 태그와 따로 기술하고 있지만 이 글에서는 <audio> 태그를 다루지 않으므로 통합하여 기술하도록 하겠습니다. 단, 모든 HTML 요소가 가진 공통 속성/메소드는 포함하지 않습니다.

속성(Attributes)

HTML5 <video> 태그에서 사용할 수 있는 속성입니다. HTML에서 표현할 수 있음은 물론, 자바스크립트에서 접근할 수도 있습니다.

표1. HTML5 video 태그의 속성

속성	값 (형식)	설명
audio	*muted*	음성의 기본 상태를 정의합니다. 현재는 `muted`(음소거)만 가능합니다.
autoplay	*	이 속성이 설정되어 있으면 사용자 입력이 없어도 동영상을 자동으로 재생합니다.
controls	*	이 속성이 설정되어 있으면 재생 버튼과 같은 제어도구가 표시됩니다.
height	픽셀 (숫자)	동영상 재생기의 높이
loop	*	이 속성이 설정되어 있으면 동영상을 계속 반복합니다.
poster	URL (문자열)	동영상 대표 이미지의 URL
preload	*	이 속성이 설정되어 있으면 페이지를 읽을 때 미디어도 같이 읽어들여 재생을 준비합니다. `autoplay` 속성이 설정되어있으면 무시됩니다.
src	URL (문자열)	재생할 미디어의 URL
width	픽셀 (숫자)	동영상 재생기의 너비

표1 에서 *로 표시한 값은 값을 입력하지 않거나 아무런 값을 입력해도 된다는 의미입니다. HTML5에서는 값을 가지지 않는 속성도 유효하므로 속성 값을 입력하지 않아도 됩니다. 값을 입력한 형태로는 autoplay="autoplay"처럼 속성 이름을 값에 입력해서 사용하는 방법이 일반적입니다.

src 속성

웹 페이지에서 이미지 데이터를 표현할 때는

태그를 사용하고 src 속성의 값으로 이미지 파일의 URL을 설정합니다. 마찬가지로 웹 페이지에서 동영상 데이터를 표현할 때는 <video> 태그를 사용하고 src 속성의 값으로 미디어 파일의 URL을 설정합니다...처럼 일이 간단했으면 좋았을 것입니다. 지난 글에서 말했던 것과 같이 '코덱 전쟁'이 현재 진행형이기 때문에 src 속성 하나만으로 모든 브라우저를 다 지원할 수는 없습니다.

따라서, 모든 브라우저에서 동영상을 지원하려면 다음과 같이태그를 사용해 각각의 포맷을 정해줘야 합니다.태그에 대해서는 이 글 후반부에 다룰 예정이니 지금은 '미디어를 여러개 설정할 때 사용하는 요소'쯤으로 이해해두셔도 됩니다.

```
<video>
    <source src="pics/video/gizmo.mp4" type="video/mp4" />
    <source src="pics/video/gizmo.webm" type="video/webm" />
    <source src="pics/video/gizmo.ogv" type="video/ogg" />
</video>
```

<video> 태그를 지원하지 않는 웹 브라우저 사용자들을 위한 대비책(fallback)은 다음과 같이 간단히 제공할 수 있습니다.

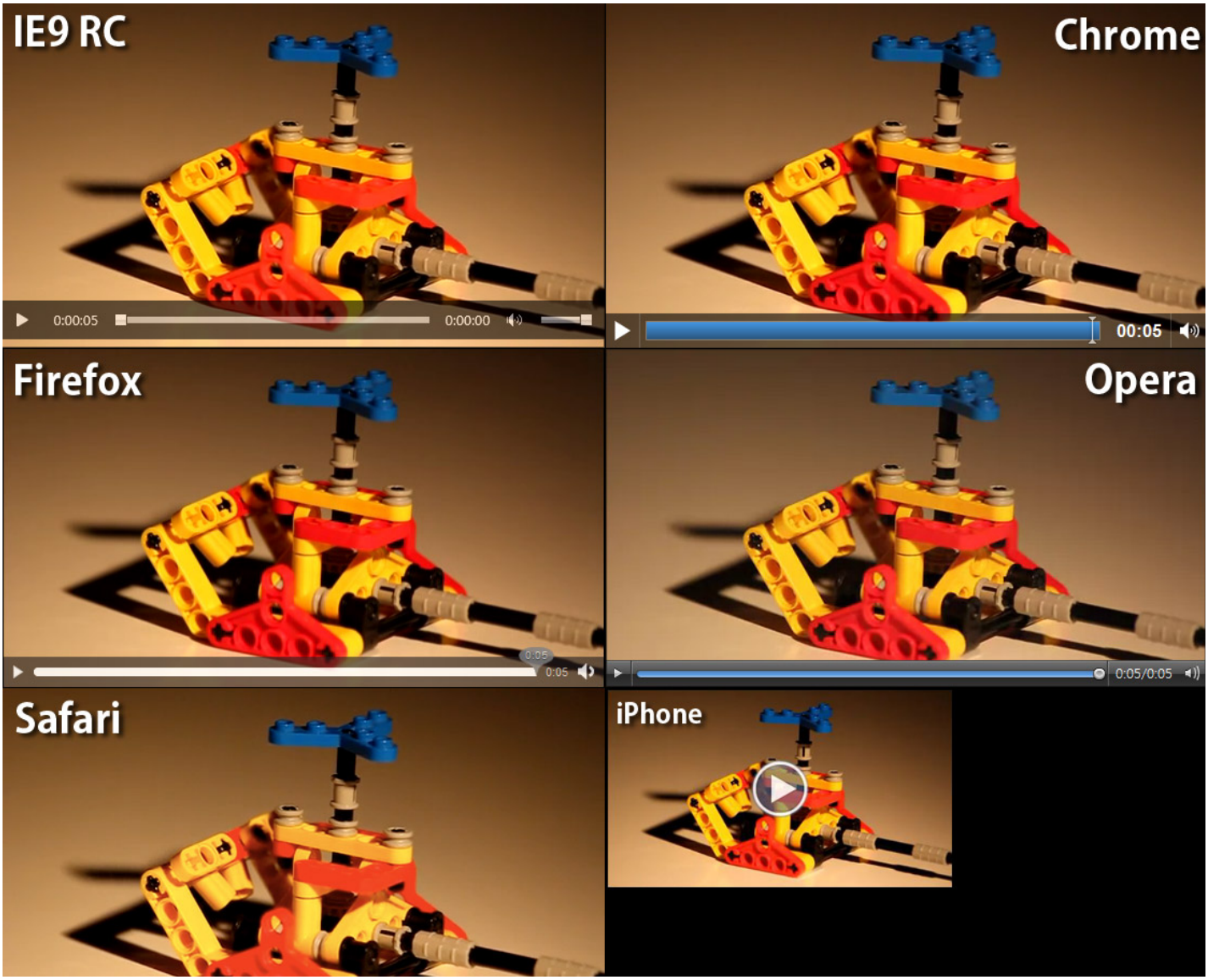
```
<video>
    <source src="pics/video/gizmo.mp4" type="video/mp4" />
    <source src="pics/video/gizmo.webm" type="video/webm" />
    <source src="pics/video/gizmo.ogv" type="video/ogg" />
    video 요소를 지원하지 않는 브라우저입니다. 동영상은 <a href="pics/video/gizmo.mp4">다운로드</a> 후
</video>
```

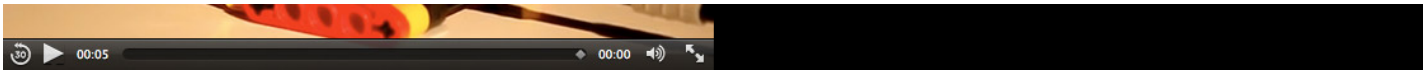
controls 속성

이 속성이 설정되었을 때 각 브라우저마다 나타나는 컨트롤의 기능은 거의 같습니다. 다만, 대부분의 브라우저에서는 <video> 에 마우스를 오버했을 경우에만 컨트롤이 나타나는데 반해 구글 크롬 브라우저에서는 항상 컨트롤이 표시됩니다.

```
<video controls="controls">
    <source src="pics/video/gizmo.mp4" type="video/mp4" />
    <source src="pics/video/gizmo.webm" type="video/webm" />
    <source src="pics/video/gizmo.ogv" type="video/ogg" />
</video>
```

각 브라우저에서 controls 속성을 설정하면 나타나는 컨트롤의 외양은 다음 그림을 참고하세요.





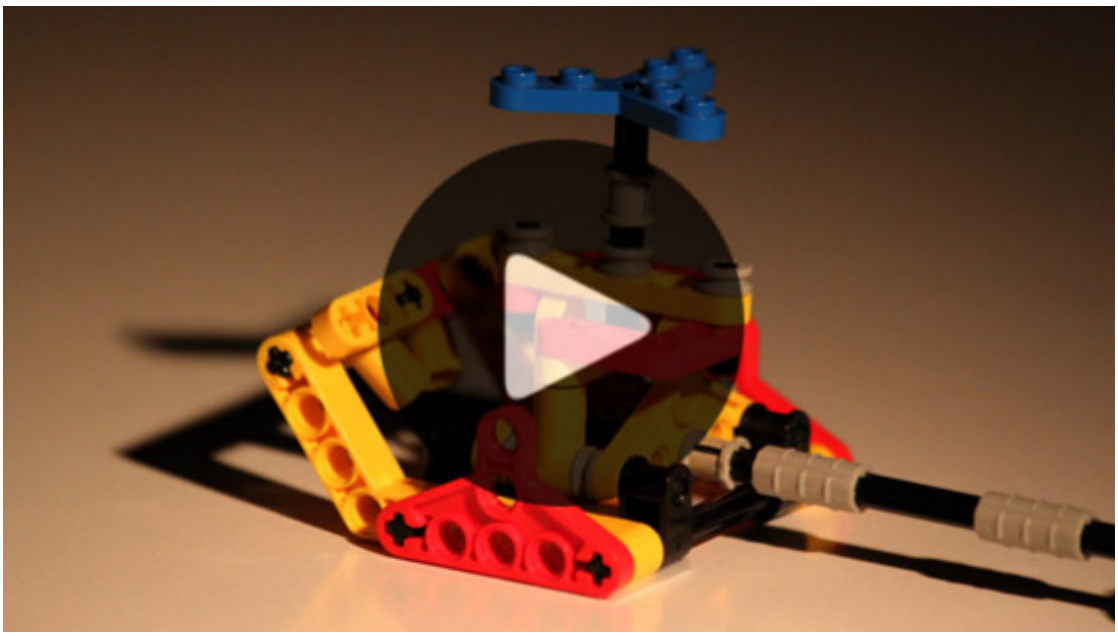
참고로 아이폰/아이패드는 controls 속성 설정 여부에 상관없이 아이폰에서 제공하는 반투명한 재생 버튼이 <video> 영역 중앙에 나타나고 이를 클릭하면 아이폰에서는 전체 화면으로 재생이 되고, 아이패드에서는 웹 브라우저처럼 그 자리에서 재생이 됩니다. 안드로이드의 경우(프로요 기준), 컨트롤이 전혀 나타나지 않고 비디오 영역임을 알리는 작은 아이콘만 나타났으며, 클릭해도 동영상이 재생되지는 않았습니다. 간단한 자바스크립트를 통해 클릭할 때 실행하도록 하자 그제서야 동영상이 나타났습니다.[1]이 글에서 사용된 기기는 아이폰 3GS+iOS 4.2, HTC Desire+안드로이드 프로요(2.2)입니다.

poster 속성

<video> 태그는 재생 대기 상태의 첫 화면을 표시할 때 미디어 파일의 첫 프레임을 대표 이미지로 가정합니다. 하지만, 모든 미디어의 첫 프레임이 그 동영상을 대표하는 것은 아닙니다. 예컨대, 어떤 동영상은 페이드-인(Fade-in)으로 시작하기 때문에 첫 장면이 그냥 새까만 화면일 수도 있습니다. 이런 문제때문에 HTML5 <video> 에서는 poster 속성을 통해 아직 재생 대기중인 동영상의 대표 이미지를 설정할 수 있도록 했습니다.

```
<video poster="pics/image/gizmo.jpg">
    <source src="pics/video/gizmo.mp4" type="video/mp4" />
    <source src="pics/video/gizmo.webm" type="video/webm" />
    <source src="pics/video/gizmo.ogv" type="video/ogg" />
</video>
```

이 코드는 다음과 같이 표현됩니다.



단, 웹 브라우저 별로 약간의 차이를 보였는데,

- IE9 RC 버전과 Safari 브라우저는 최초에는 포스터 이미지를 표시했다가 미디어를 읽으면 미디어의 첫 프레임으로 화면을 전환했습니다.

- 아이폰의 모바일 Safari는 포스터 이미지를 그대로 표시하지만, 그 위에 아이폰 고유의 재생 버튼을 겹쳐서 표시합니다(controls 속성 설명의 그림 참조).
- 안드로이드의 기본 웹 브라우저도 포스터 이미지는 그대로 잘 표현해주므로, 아이콘만 간단히 표시해버리는 안드로이드를 위해서는 poster 속성을 적절히 사용하는 것이 필요합니다.

프로퍼티(Properties)

프로퍼티[2]일반적으로는 프로퍼티(Properties)도 '속성'이라고 번역하는 용어이지만, 이 글에서는 '자바스크립트에서 접근하고 사용할 수 있는 값'을 구분하기 위해 일부러 프로퍼티를 속성(attributes)과 구분해서 사용했습니다.는 자바스크립트에서 접근할 수 있는 속성을 말합니다. 밑줄 친 속성 이름은 읽기 전용이라는 뜻입니다.

표2. HTML5 video 태그의 프로퍼티

프로퍼티	형식	설명
buffered	TimeRanges[3]시간 범위를 나타내는 객체로 length, start, end 를 속성으로 가집니다.	현재 재생중인 미디어의 URL
currentSrc	문자열	현재 재생중인 미디어의 URL
currentTime	숫자	재생 중인 미디어의 현재 위치
defaultPlaybackRate	숫자	기본 미디어 재생 속도 배속. 기본값은 1.0으로 원래의 속도대로 재생합니다.
duration	숫자	미디어의 재생 길이. 미디어 데이터가 없다면 NaN을 반환합니다.
ended	Boolean	재생이 종료됐는지의 여부
initialTime	숫자	초기 재생 위치. 초 단위 숫자로 표현
loop	Boolean	미디어의 반복 재생 여부
muted	Boolean	음소거 여부
networkState	숫자	네트워크 상태를 나타내는 0부터 3까지의 정수.
paused	Boolean	미디어 재생 일시 정지 여부
poster	문자열	video 요소의 poster 속성
playbackRate	숫자	미디어 재생 속도 배속. 기본값은 1.0으로 원래의 속도대로 재생합니다. 이 속성을 조절하면 빨리 감기나 슬로우 모션 기능을 구현할 수 있습니다.
played	TimeRanges	현재 재생중인 미디어의 URL
preload	문자열	`video` 요소의 preload 속성
readyState	숫자	video 요소의 준비 상태. 0~3까지의 숫자값을 가집니다.
seekable	TimeRanges	탐색 가능한 범위
seeking	Boolean	미디어의 특정 위치로 탐색 중일 때. 탐색 전이거나 탐색을 완료한 후에는 항상 `false`입니다.
src	문자열	`video` 요소의 src 속성
startOffsetTime	Date	시작 위치를 Date 객체로 반환
tracks	TextTrack[]	text track 배열
videoHeight	숫자	동영상 원본의 높이

프로퍼티	형식	설명
videoWidth	숫자	동영상 원본의 너비
volume	숫자	0.0(음소거)부터 1.0(최대)까지의 실수로 표현하는 볼륨 크기
width	숫자	`video` 요소의 width 속성

readyState

미디어의 준비 상태를 나타내며, 0부터 4까지의 정수를 값으로 가집니다. 정수값은 상수4)미디어의 로딩 상태, 네트웍의 상태 등은 일반적으로 숫자로 표현됩니다. 예를 들어, Ajax에서 사용하는 XmlHttpRequest 객체(이하 XHR)는 네트웍 요청 후 수신 상태는 readyState 속성을 통해 알 수 있는데 0(XHR이 생성되었으나 초기화 안된 상태)부터 4(모든 데이터 수신 완료)까지 숫자로 표현합니다. 그러나, 숫자에는 의미가 포함되지 않기에 2가 어떤 상태였는지 3이 어떤 상태였는지 기억하기 어렵습니다. 그래서 실제로는 숫자로 취급되지만 사람이 조금 더 읽기 편하게 의미있는 문자 집합을 따로 사용하기도 하는데, 이를 **상수**라고 부릅니다.로도 표현되는데, 각 상수는 다음과 같은 의미를 가집니다.

- HAVE_NOTHING = 0 : 정보 없음
  미디어에 대한 정보가 없는 상태. 이 상태에서는 현재 재생 위치를 알 수 없으며, networkState는 항상 NETWORK_EMPTY가 됩니다.
- HAVE_METADATA = 1 : 메타데이터 있음
  미디어의 재생 길이를 알 수 있습니다. 지금 재생할 미디어 데이터는 아직 없지만, 탐색을 시도해도 API에서 오류를 일으키지는 않습니다.
- HAVE_CURRENT_DATA = 2 : 현재의 미디어 데이터 있음
  지금 바로 재생할 미디어 데이터는 준비되었으나 향후 진행할 데이터는 아직 없는 상태입니다.
- HAVE_FUTURE_DATA = 3 : 현재는 물론 앞으로의 데이터도 있음
  지금 바로 재생할 미디어 데이터는 물론, 앞으로 진행할 데이터까지 여분으로 준비된 상태입니다.
- HAVE_ENOUGH_DATA = 4 : 모든 데이터 충분
  모든 조건이 HAVE_FUTURE_DATA 상태를 만족시키는 것은 물론, `defaultPlaybackRate`에서 설정한 속도대로 진행한다면 미디어 끝까지 재생해도 문제없이 진행할 수 있게 준비된 상태입니다.

networkState

미디어의 준비 상태를 나타내는 readyState와 마찬가지로 네트워크의 상태를 나타내는 networkState 역시 0부터 3까지의 정수로 표현되며, <video> 객체는 이에 해당하는 상수를 가지고 있습니다. 각 상수는 다음과 같습니다.

- NETWORK_EMPTY = 0
  요소가 아직 초기화 되지 않은 상태
- NETWORK_IDLE = 1
  어떤 미디어를 사용할지 선택했으나 아직 네트워크는 사용하지 않은 상태
- NETWORK_LOADING = 2 :
  웹 브라우저에서 미디어를 읽어 오는 중
- NETWORK_NO_SOURCE = 3 :
  사용할 미디어를 선택했으나 설정한 경로에 미디어가 없을 때

상수에는 readyState와 동일한 방법으로 접근할 수 있습니다.

메소드(Methods)

MutableTextTrack addTrack(String kind[, String label, String language])

새로운 텍스트 트랙을 추가합니다.

- kind는 subtitles, captions, descriptions, chapters, metadata 중 하나의 값을 가질 수 있으며 이 외의 값을 입력하면 에러가 발생합니다.
- label, language의 기본값은 빈 문자열입니다.
- 이 메소드가 정상적으로 실행됐다면 TextTracks의 크기가 1만큼 증가합니다.

String canPlayType(String type)

이 메소드에서 반환되는 문자열은 *빈 문자열*, "probably", "maybe" 중 하나의 값입니다.

- *빈 문자열* : 전달받은 type이 웹 브라우저에서 알 수 없는 유형이거나 type이 "application/octet-steam"인 경우. **전혀 재생할 수 없다는 뜻입니다.**
- "probably" : 아마 재생할 수 있을 것 같기는 한데, <video> 요소를 통해서 재생할 수 있을지는 잘 모르겠다라는 의미입니다. **<embed> 등의 오래된 태그를 통해 재생할 수 있습니다.**
- "maybe" : 가장 희망적인 메시지로 **미디어를 재생할 수 있을 것 같다는 의미입니다.**

가장 희망적인 메시지조차 "maybe"처럼 애매한 값을 반환하는 이유는 미디어 파일은 재생해 보기 전에는 결과를 알 수 없기 때문입니다. 다음은 웹 브라우저가 fictional 코덱을 지원하는지 확인 후 미디어 재생 요소를 동적으로 변경시키는 코드입니다(출처 : W3C HTML5 스펙 문서).

```
<section id="video">
  <p><a href="playing-cats.nfv">Download video</a></p>
</section>
<script>
 var videoSection = document.getElementById('video');
 var videoElement = document.createElement('video');
 var support = videoElement.canPlayType('video/x-new-fictional-format;codecs="kittens,bunnies"');
 if (support != "probably" && "New Fictional Video Plug-in" in navigator.plugins) {
   // 브라우저 지원 여부가 확실하지 않을 때,
   //  플러그인이 있다면 플러그인을 사용합니다
   videoElement = document.createElement("embed");
 } else if (support == "") {
   // 브라우저에서 지원도 안되고, 플러그인도 없으면 아무 것도 안합니다
   videoElement = null;
 }
 if (videoElement) {
   while (videoSection.hasChildNodes())
     videoSection.removeChild(videoSection.firstChild);
   videoElement.setAttribute("src", "playing-cats.nfv");
   videoSection.appendChild(videoElement);
 }
</script>
```

void load()

미디어를 읽어 들입니다.

void play()

미디어를 재생합니다.

void pause()

미디어 재생을 일시 중지합니다.