JavaScript 를 사용하여 HTML5 동영상 플레이어 제어

HTML5 video 개체는 JavaScript 에서 재생을 제어하는 데 사용할 수 있는 메서드, 속성 및 이벤트를 제공합니다.
웹 페이지에 HTML5 동영상 컨트롤 추가에 설명된 대로 HTML5 video 요소를 통해 웹 페이지에서 동영상 요소 사용을 시작합니다.
JavaScript 를 추가하면 프로그래밍 방식으로 사용자 지정 재생 컨트롤을 만들고 현재 재생 위치를 가져오고 설정하며 현재 동영상을 변경할 수 있습니다.

여기서는 video 개체를 제어하는 데 사용할 수 있는 몇 가지 기본 메서드 및 속성에 대해 살펴봅니다.

컨트롤러의 버튼 기능을 만들려면?

HTML5 video 요소에는 동영상 재생을 재생, 일시 중지 및 검색하는 고유한 기본 제공 컨트롤이 있습니다.
다음 예제에서는 play 및 pause 메서드를 사용하여 동영상을 시작하고 중지하며, currentTime 속성을 사용하여 동영상을 검색하고, paused 속성을 사용하여 동영상 플레이어의 현재 재생 상태를 가져옵니다.

HTML

```
<script type="text/javascript">

    function vidplay() {
       var video = document.getElementById("Video1");
       var button = document.getElementById("play");
       if (video.paused) {
          video.play();
          button.textContent = "||";
       } else {
          video.pause();
          button.textContent = ">";
       }
    }

    function restart() {
        var video = document.getElementById("Video1");
        video.currentTime = 0;
    }

    function skip(value) {
        var video = document.getElementById("Video1");
        video.currentTime += value;
    }
</script>

</head>
<body>

<video id="Video1" >
//  Replace these with your own video files.
     <source src="demo.mp4" type="video/mp4" />
     <source src="demo.ogv" type="video/ogg" />
     HTML5 Video is required for this example.
     <a href="demo.mp4">Download the video</a> file.
</video>

<div id="buttonbar">
    <button id="restart" onclick="restart();">[]</button>
    <button id="rew" onclick="skip(-10)">&lt;&lt;</button>
    <button id="play" onclick="vidplay()">&gt;</button>
    <button id="fastFwd" onclick="skip(10)">&gt;&gt;</button>
</div>
```

이 예제에서는 HTML5 video 요소를 source 요소와 함께 사용하여 웹 페이지에서 플레이어 및 동영상 콘텐츠를 설정합니다.
동영상 요소에는 id 특성만 있어 단추 및 JavaScript 기능에 대한 재생 컨트롤을 유지합니다.
 단추는 동영상의 재생/일시 중지, 앞뒤로 건너뛰기 및 다시 시작 컨트롤을 제공합니다.
재생은 일시 중지 단추이기도 합니다.
별도의 단추가 있는 것이 아니라 재생 단추의 텍스트가 기능을 반영하도록 변경됩니다.

재생 단추를 클릭하면 "vidplay()" 함수를 호출하여 paused 속성을 확인합니다.
paused 속성은 동영상이 일시 중지되거나 방금 로드된 경우 true 를 반환하고 동영상이 재생 중이면 false 를 반환합니다.
동영상이 일시 중지되면 "vidplay()"에서 play 메서드를 호출하고 재생 단추의 innerHTML 을 "일시 중지"에 대한 기호인 "||"로 변경합니다.
재생 단추를 클릭할 때 동영상이 재생 중이면 pause 메서드가 호출되고 innerHTML 이 "재생"에 대한 기호인 ">"로 변경됩니다.
다시 시작("[]"), 뒤로("<<") 및 앞으로(>>) 단추는 모두 currentTime 속성을 사용합니다.
currentTime 속성은 현재 재생 위치를 초 단위로 나타냅니다.
재생 및 앞으로 단추는 "skip()" 함수를 호출합니다.
"skip()" 함수는 현재 시간에서 10 초를 더하거나 빼는 매개 변수를 사용합니다.
currentTime 을 0 보다 작거나 동영상 기간보다 큰 값으로 설정하면 동영상 플레이어에서 해당 값을 동영상의 시작이나 끝으로 설정합니다.
다시 시작("[]") 단추는 "restart()" 함수를 호출하여 currentTime 을 0 으로 설정합니다.

원하는 파일 형식을 재생하려면 ?

JavaScript 에서 사용할 형식을 확인하는 작업은 웹 페이지에 HTML5 동영상 컨트롤 추가에 표시된 대로 source 요소를 사용하는 경우보다 약간 더 복잡합니다.
그러나 동영상의 요소의 지원은 변경되지 않으므로 사용 가능한 지원을 확인한 후 나머지 검색 세션에 대해 형식을 가정할 수 있습니다.

브라우저의 형식 기능을 찾으려면 video 개체의 canPlayType 메서드를 사용합니다.
canPlayType 메서드는 동영상 MIME 형식과 선택적인 코덱 매개 변수를 사용하고 "empty", "maybe" 또는 "probably"의 세 문자열 중 하나를 반환합니다.

HTML

```
if (myvideo.canPlayType) {
  // CanPlayType returns maybe, probably, or an empty string.
  var playMsg = myvideo.canPlayType('video/mp4; codecs="avc1.42E01E"');
  if ("" != playMsg) {
    msg.innerHTML += "mp4/H.264 is " + playMsg + " supported <br/>";
  }
  playMsg = myvideo.canPlayType('video/ogg; codecs="theora"');
  if ("" != playMsg) {
    msg.innerHTML += "ogg is " + playMsg + " supported<br/>";
  }
  playMsg = myvideo.canPlayType('video/webm; codecs="vp8, vorbis"');
  if ("" != playMsg) {
    msg.innerHTML += "webm is " + playMsg + " supported<br/>";
  }
}
else {
  msg.innerHTML += "no video support";
}
}
```

"maybe"와 "probably"의 차이점은 canPlayType 메서드에 정보가 충분하지 않다는 것입니다.
예를 들어 코덱 매개 변수가 누락된 경우 메서드에서 mp4 지원에 대해 "maybe"를 반환할 수 있습니다.
이는 지원되지 않는 mp4 코덱이 있을 수 있기 때문입니다.
코덱 매개 변수는 mp4 파일의 보다 구체적인 버전으로 지원의 범위를 좁힙니다.

canPlayType 메서드가 반환하는 세 상태의 모호함으로 인해 브라우저에서 해당 파일 형식을 지원하는지 여부를 결정하기가 어려울 수 있습니다.
규칙은 아니지만 브라우저에서 "maybe"를 반환하는 경우 해당 형식을 지원할 수도 있습니다.

다음 예문에서는 반환된 문자열이 "maybe" 또는 "probably"일 경우 부울 true 를 반환하고 문자열이 비어 있으면 false 를 반환합니다.

HTML

```
var playMsg = myvideo.canPlayType('video/mp4; codecs="avc1.42E01E"');
if ("" != playMsg) {
  msg.innerHTML += "mp4/H.264 is " + playMsg + " supported <br/>";
}
```

이전 예제의 첫 번째 줄에서는 "video/mp4; codecs="avc1.42E01E"" 매개 변수를 사용하여 canPlayType 메서드를 호출합니다.
그런 다음 "if" 문에서 결과가 빈 문자열이 아닌 다른 값인지를 확인합니다. 그럴 경우 메시지와 해당 결과를 표시합니다.

다음 예제에서는 mp4, ogv 및 webm 의 세 가지 동영상 형식을 테스트합니다.

HTML

```
<!doctype html>
<html>
<head>
    <title>Using multiple file formats in JavaScript</title>
    <!-- Uncomment the following meta tag if you have issues rendering this page on an intranet site. -->
    <!--  <meta http-equiv="X-UA-Compatible" content="IE=edge"/> -->
</head>
<body>
  <h1>CanPlayType test for multiple files</h1>
    <div>The canPlayType method tests for specific video formats and codecs. <br />It retur
    <br />
    <div>
        <button onclick="checkVideoCompat();">
            Test for video format type
        </button>
    </div>
    <div id="display"> </div>

    <script type= "text/javascript">
      function checkVideoCompat() {
        var myvideo = document.createElement('video');
        var msg = document.getElementById("display");
        msg.innerHTML = "";
        if (myvideo.canPlayType) {
          // CanPlayType returns maybe, probably, or an empty string.
          var playMsg = myvideo.canPlayType('video/mp4; codecs="avc1.42E01E"');
          if ("" != playMsg) {
            msg.innerHTML += "mp4/H.264 is " + playMsg + " supported <br/>";
          }
          playMsg = myvideo.canPlayType('video/ogg; codecs="theora"');
          if ("" != playMsg) {
            msg.innerHTML += "ogg is " + playMsg + " supported<br/>";
          }
          playMsg = myvideo.canPlayType('video/webm; codecs="vp8, vorbis"');
          if ("" != playMsg) {
            msg.innerHTML += "webm is " + playMsg + " supported<br/>";
          }
        }
        else {
          msg.innerHTML += "no video support";
        }
      }
    </script>


</body>
</html>
```

Windows Internet Explorer 에서 HTML5 요소 및 기능을 사용하는 경우 사용하는 기능을 테스트하는 것이 가장 좋습니다.

이 예제에서는 createElement 메서드를 사용하여 동영상 개체를 만듭니다.
video 개체가 성공적으로 만들어진 경우 "if (myvideo.canPlayType)" 문에서 true 가 반환되고 특정 파일 형식을 테스트하는 실행이 계속됩니다.

파일을 변경하려면?

이전 예제에서는 코드의 HTML 부분에서 source 태그를 사용하여 동영상 원본 파일을 지정합니다. 둘 이상의 파일을 재생하려면 src 속성을 사용하여 동영상 파일의 URL 을 JavaScript 로 지정할 수 있습니다.

이 예제에서 src 속성은 텍스트 입력 필드의 값으로 설정됩니다. 호환되는 동영상 파일의 URL 을 필드에 입력하거나 붙여 넣은 다음 로드 단추를 클릭합니다.

HTML

```
<div id= "inputField" style="display:none;" >
    <input type="text" id="videoFile" value="http://ie.microsoft.com/testdrive/ieblog/2011/nov/pp4_blog_c
    <button id="loadVideo" >Load</button>
</div>
```

로드 단추를 클릭하면 "getVideo()" 함수가 입력 필드(URL)에서 값을 가져와 src 속성에 할당합니다.
그런 다음 "getVideo()" 함수에서 "재생" 단추에 대한 click 메서드를 호출하여 파일이 재생을 시작합니다.

HTML

```
//  load video file from input field
function getVideo() {
    var fileURL = document.getElementById("videoFile").value;  // get input
    if (fileURL != ""){
       video.src = fileURL;
       video.load();  // if HTML source element is used
       document.getElementById("play").click();  // start play
     } else {
        errMessage("Enter a valid video URL");  // fail silently
     }
}
```

재생 단추를 클릭하면 "vidplay()" 함수를 호출하여 재생하거나 일시 중지합니다.
재생할 단일 파일만 원본 요소에 의해 설정된 이전 예제와 달리 "vidplay()"에서 먼저 src 속성을 확인합니다.

페이지가 처음 로드되는 경우처럼 비어 있으면 "getVideo()" 함수가 호출되어 텍스트 필드에서 URL 을 로드합니다.
"getVideo()"는 입력 필드에서 URL 을 가져오고, load 동영상 메서드를 호출한 다음 파일을 재생하는 재생 단추의 클릭 이벤트를 발생시킵니다.
페이지에서 이미 동영상을 재생 중인 경우 새 파일을 입력 필드에 붙여 넣으면 로드 단추를 클릭하여 파일을 변경합니다.

HTML

```
//  play video
function vidplay(evt) {
   if (video.src == "") {  // on first run, src is empty, go get file
       getVideo();
   }
   button = evt.target; //  get the button id to swap the text based on th
   if (video.paused) {   // play the file, and display pause symbol
      video.play();
      button.textContent = "||";
   } else {              // pause the file, and display play symbol
      video.pause();
      button.textContent = ">";
   }
}
```

```
// load video file from input field
function getVideo() {
    var fileURL = document.getElementById("videoFile").value;  // get input field
    if (fileURL != ""){
       video.src = fileURL;
       video.load();  // if HTML source element is used
       document.getElementById("play").click();  // start play
     } else {
        errMessage("Enter a valid video URL");  // fail silently
     }
}
```

재생 단추는 "vidplay()" 함수를 호출하여 파일을 재생합니다.
이상으로 "JavaScript 를 사용하여 HTML5 동영상 플레이어 제어" 에 관하여 살펴 보았습니다.