HTMLMediaElement.play() Returns a Promise

By Jeff Posnick
Web DevRel @ Google

Automatically playing audio and video on the web is a powerful capability, and one that's subject to different restrictions on different platforms. Today, most desktop browsers will always allow web pages to begin `<video>` or `<audio>` playback via JavaScript without user interaction. Most mobile browsers, however, require an explicit user gesture before JavaScript-initiated playback can occur. This helps ensure that mobile users, many of whom pay for bandwidth or who might be in a public environment, don't accidentally start downloading and playing media without explicitly interacting with the page.

It's historically been difficult to determine whether user interaction is required to start playback, and to detect the failures that happen when (automatic) playback is attempted and fails. Various workarounds exist, but are less than ideal. An improvement to the underlying `play()` method to address this uncertainty is long overdue, and this has now made it to the web platform, with an initial implementation in Chrome 50.

A `play()` call on an a `<video>` or `<audio>` element now returns a Promise. If playback succeeds, the Promise is fulfilled, and if playback fails, the Promise is rejected along with an error message explaining the failure. This lets you write intuitive code like the following:

```
var playPromise = document.querySelector('video').play();

// In browsers that don’t yet support this functionality,
// playPromise won’t be defined.
if (playPromise !== undefined) {
  playPromise.then(function() {
    // Automatic playback started!
  }).catch(function(error) {
    // Automatic playback failed.
    // Show a UI element to let the user manually start playback.
  });
}
```

In addition to detecting whether the `play()` method was successful, the new Promise-based interface allows you to determine when the `play()` method succeeded. There are contexts in which a web browser may decide to delay the start of playback—for instance, desktop Chrome will not begin playback of a `<video>` until the tab is visible. The Promise won't fulfill until playback has actually started, meaning the code inside the `then()` will not execute until the media is playing. Previous methods of determining if `play()` is successful, such as waiting a set amount of time for a `playing` event and assuming failure if it doesn't fire, are susceptible to false negatives in delayed-playback scenarios.

We've published a live example of this new functionality. View it in a browser such as Chrome 50 that supports this Promise-based interface. Be forewarned: the page will automatically play music when you visit it. (Unless, of course, it doesn't!)

Danger zone

`<source>` within `<video>` makes `play()` promise never rejects

For `<video src="not-existing-video.mp4"\>`, the `play()` promise rejects as expected as the video doesn't exist. For `<video><source src="not-existing-video.mp4" type='video/mp4'></video>`, the `play()` promise never rejects. It only happens if there are no valid sources.

Chromium Bug

Last updated July 2, 2018.