Media Source API: Automatically Ensure Seamless Playback of Media Segments in Append Order

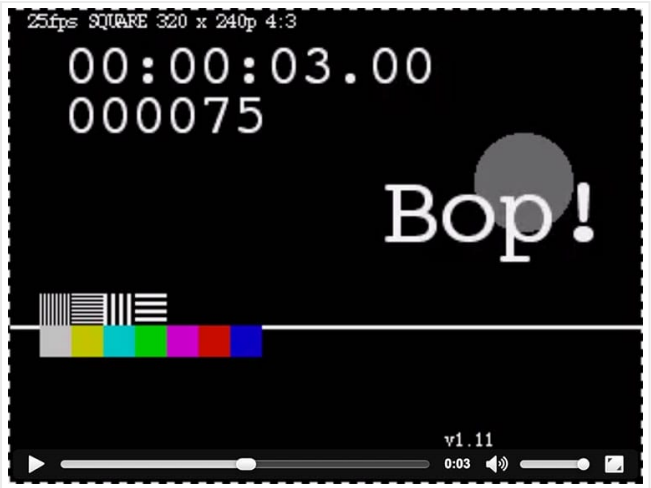
By Sam Dutton
Sam is a Developer Advocate

The HTML audio and video elements enable you to load, decode and play media, simply by providing a src URL:

```
<video src='foo.webm'></video>
```

That works well in simple use cases, but for techniques such as adaptive streaming, the Media Source Extensions API (MSE) provides more control. MSE enables streams to be built in JavaScript from segments of audio or video.

You can try out MSE at simpl.info/mse:



The code below is from that example.

A **MediaSource** represents a source of media for an audio or video element. Once a **MediaSource** object is instantiated and its **open** event has fired, **SourceBuffers** can be added to it. These act as buffers for media segments:

```
var mediaSource = new MediaSource();
video.src = window.URL.createObjectURL(mediaSource);
mediaSource.addEventListener('sourceopen', function() {
```

```
  var sourceBuffer =
      mediaSource.addSourceBuffer('video/webm; codecs="vorbis,vp8"');
  // Get video segments and append them to sourceBuffer.
}
```

Media segments are 'streamed' to an audio or video element by adding each segment to a `SourceBuffer` with `appendBuffer()`. In this example, video is fetched from the server then stored using the File APIs:

```
reader.onload = function (e) {
  sourceBuffer.appendBuffer(new Uint8Array(e.target.result));
  if (i === NUM_CHUNKS - 1) {
    mediaSource.endOfStream();
  } else {
    if (video.paused) {
      // start playing after first chunk is appended
      video.play();
    }
    readChunk(++i);
  }
};
```

Setting playback order

Chrome 50 adds additional support to the `SourceBuffer` `mode` attribute, allowing you to specify that media segments are played back continuously, in the order that they were appended, no matter whether the media segments initially had discontinuous timestamps.

Use the `mode` attribute to specify playback order for media segments. It has one of two values:

- *segments*: The timestamp of each segment (which may have been modified by `timestampOffset`) determines playback order, no matter the order in which segments are appended.
- *sequence*: The order of segments buffered in the media timeline is determined by the order in which segments are appended to the `SourceBuffer`.

If the media segments have timestamps parsed from byte stream data when they are appended to the `SourceBuffer`, the `SourceBuffer`'s `mode` property will be set to *segments*. Otherwise `mode` will be set to *sequence*. Note that timestamps are not optional: they *must* be there for most stream types, and *cannot* be there for others: inband timestamps are innate to stream types that contain them.

Setting the **mode** attribute is optional. For streams that don't contain timestamps (audio/mpeg and audio/aac) **mode** can only be changed from *segments* to *sequence*: an error will be thrown if you try to change **mode** from *sequence* to *segments*. For streams that have timestamps, it is possible to switch between *segments* and *sequence*, though in practice that would probably produce behaviour that was undesirable, hard to understand or difficult to predict.

For all stream types, you can change the value from *segments* to *sequence*. This means segments will be played back in the order they were appended, and new timestamps generated accordingly:

```
sourceBuffer.mode = 'sequence';
```

Being able to set the **mode** value to *sequence* ensures continuous media playback, no matter if the media segment timestamps were discontinuous — for example, if there were problems with video muxing, or if (for whatever reason) discontinuous segments are appended. It is possible for an app to polyfill with **timestampOffset** to ensure continuous playback, if correct stream metadata is available, but *sequence* mode makes the process simpler and less error prone.

MSE apps and demos

These show MSE in action, though without **SourceBuffer.mode** manipulation:

- Media Source API
- Shaka Player: video player demo that uses MSE to implement DASH with the Shaka JavaScript library

Browser support

- Chrome 50 and above by default
- For Firefox, see MDN for details

Specification

- Media Source Extensions **appendMode**() method

API information

- MDN: SourceBuffer.mode

Last updated July 2, 2018.