Service Worker Caching, PlaybackRate and Blob URLs for Audio and Video on Chrome for Android

By Sam Dutton

Sam is a Developer Advocate

Sometimes good things have boring names.

Case in point: the Unified Media Pipeline, **UMP** for short.

This may sound like a sinister Soviet era directive, but in fact it's an important step towards consistent cross-platform audio and video delivery. Chrome on Android will now use the same media stack as desktop Chrome, rather than relying on the underlying platform implementation.

UMP enables you to do a lot:

- Cache audio and video with service workers, since media delivery is now implemented directly within Chrome rather than being passed off to the Android media stack.
- Use blob URLs for audio and video elements.
- Set `playbackRate` for audio and video.
- Pass MediaStreams between Web Audio and MediaRecorder.
- Develop and maintain media apps more easily across devices — media works the same on desktop and Android.

UMP took some hard engineering work to implement:

- A new caching layer for improved power performance.
- Updating a new MediaCodec based video decoder hosted in Chrome's GPU process.
- Lots of testing and iteration on different devices.

Here's a demo of video caching with a service worker:

Caching the video file and the video poster image is as simple as adding their paths to the list of URLs to prefetch:

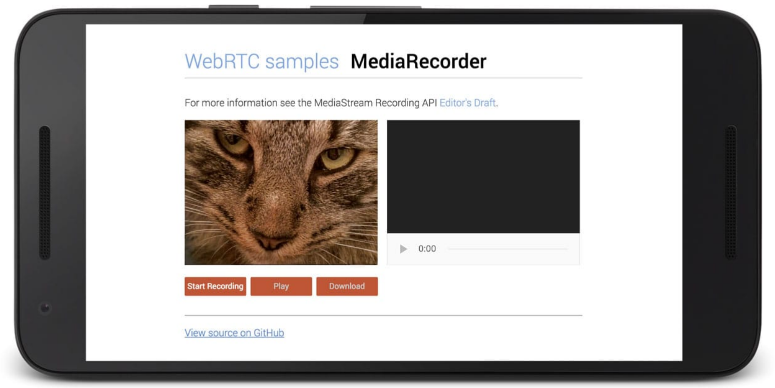
```
<video controls  poster="static/poster.jpg">
  <source src="static/video.webm" type="video/webm" />
  <p>This browser does not support the video element.</p>
</video>

 var urlsToPrefetch = [
    'static/video.webm', 'static/poster.jpg',
  ];
```

There's some shim code in this demo to handle range requests, which are not yet implemented by service worker.

The inability to change **`playbackRate`** on Android has been a long-standing bug. UMP fixes this. For the demo at simpl.info/video/playbackrate, **`playbackRate`** is set to 2. Try it out!

UMP enables blob URLs for media elements — which means that, for example, you can now play back a video recorded using the MediaRecorder API in a video element on Android:



WebRTC samples MediaRecorder

For more information see the MediaStream Recording API Editor's Draft.

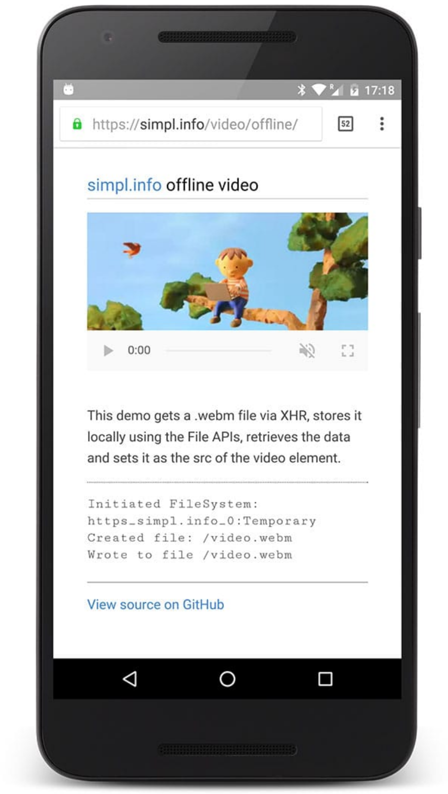
Here's the relevant code:

```
var recordedBlobs = [];

mediaRecorder.ondataavailable = function(event) {
  if (event.data && event.data.size > 0) {
    recordedBlobs.push(event.data);
  }
};

function play() {
  var superBuffer = new Blob(recordedBlobs, {type: 'video/webm'});
  recordedVideo.src = window.URL.createObjectURL(superBuffer);
}
```

For the demo at simpl.info/video/offline, video is stored using the File APIs, then played back using a Blob URL:



simpl.info offline video

This demo gets a .webm file via XHR, stores it locally using the File APIs, retrieves the data and sets it as the src of the video element.

```
Initiated FileSystem:
https_simpl.info_0:Temporary
Created file: /video.webm
Wrote to file /video.webm
```

View source on GitHub

```
function writeToFile(fileEntry, blob) {
  fileEntry.createWriter(function(fileWriter) {
    fileWriter.onwriteend = function() {
      readFromFile(fileEntry.fullPath);
    };
    fileWriter.onerror = function(e) {
      log('Write failed: ' + e.toString());
    };
    fileWriter.write(blob);
  }, handleError);
}

function readFromFile(fullPath) {
  window.fileSystem.root.getFile(fullPath, {}, function(fileEntry) {
    fileEntry.file(function(file) {
      var reader = new FileReader();
      reader.onloadend = function() {
        video.src = URL.createObjectURL(new Blob([this.result]));
      };
      reader.readAsArrayBuffer(file);
    }, handleError);
  }, handleError);
}
```

The Unified Media Pipeline has also been enabled for Media Source Extensions (MSE) and Encrypted Media Extensions (EME).

This is another step towards unifying mobile and desktop Chrome. You don't need to change your code, but building a consistent media experience across desktop and mobile should now be easier, since the media stack is the same across platforms. Debugging with Chrome DevTools? Mobile emulation now uses the 'real' audio and video stack.

If you experience problems as a result of the Unified Media Pipeline, please file issues on the implementation bug or via new.crbug.com.

Demos

- Cache video with a service worker
- Media `playbackRate`
- MediaRecorder: playback using a blob URL
- Offline video implemented with the File APIs

Relevant bugs

- Tracking issues for the Unified Media Pipeline
- UMP Field Trial
- MSE, EME and the Unified Media Pipeline

There are a couple of bugs affecting <video>, service workers and the Cache Storage API:

- <video> triggers a mode: cors request and won't accept an opaque service worker response
- Seeking doesn't work in videos served by service worker cache

Browser support

- Enabled by default in Chrome 52 and above.

Last updated July 2, 2018.