Google Cast for Chrome on Android

By Sam Dutton

Sam is a Developer Advocate

Imagine being able to use a web app from your phone to present a slide deck to a conference projector — or share images, play games or watch videos on a TV screen — using the mobile web app as a controller.

The latest release of Chrome on Android allows sites to present to Google Cast devices using the Cast Web SDK. This means you can now create Cast sender apps using the Web SDK with Chrome on Android or iOS (or on desktop with the extension) as well as creating apps that use the native Cast SDK for Android and iOS. (Previously, a Google Cast sender application needed the Google Cast Chrome extension, so on Android it was only possible to interact with Cast devices from native apps.)

Below is a brief introduction to building a Cast sender app using the Web SDK. More comprehensive information is available from the Chrome Sender App Development Guide.

All pages using Cast must include the Cast library:

```
<script type="text/javascript"
  src="https://www.gstatic.com/cv/js/sender/v1/cast_sender.js"></script>
```

Add a callback to handle API availability and initialize the Cast session (make sure to add the handler before the API is loaded!):

```
window['__onGCastApiAvailable'] = function(isLoaded, error) {
  if (isLoaded) {
    initializeCastApi();
  } else {
    console.log(error);
  }
}

function initializeCastApi() {
  var sessionRequest = new chrome.cast.SessionRequest(applicationID);
  var apiConfig = new chrome.cast.ApiConfig(sessionRequest,
      sessionListener, receiverListener);
  chrome.cast.initialize(apiConfig, onInitSuccess, onError);
};
```

If you're using the default Styled Media Receiver application and not a roll-your-own, registered Custom Receiver application, you can create a **`SessionRequest`** like this:

```
var sessionRequest = new chrome.cast.SessionRequest(chrome.cast.media.
  DEFAULT_MEDIA_RECEIVER_APP_ID);
```

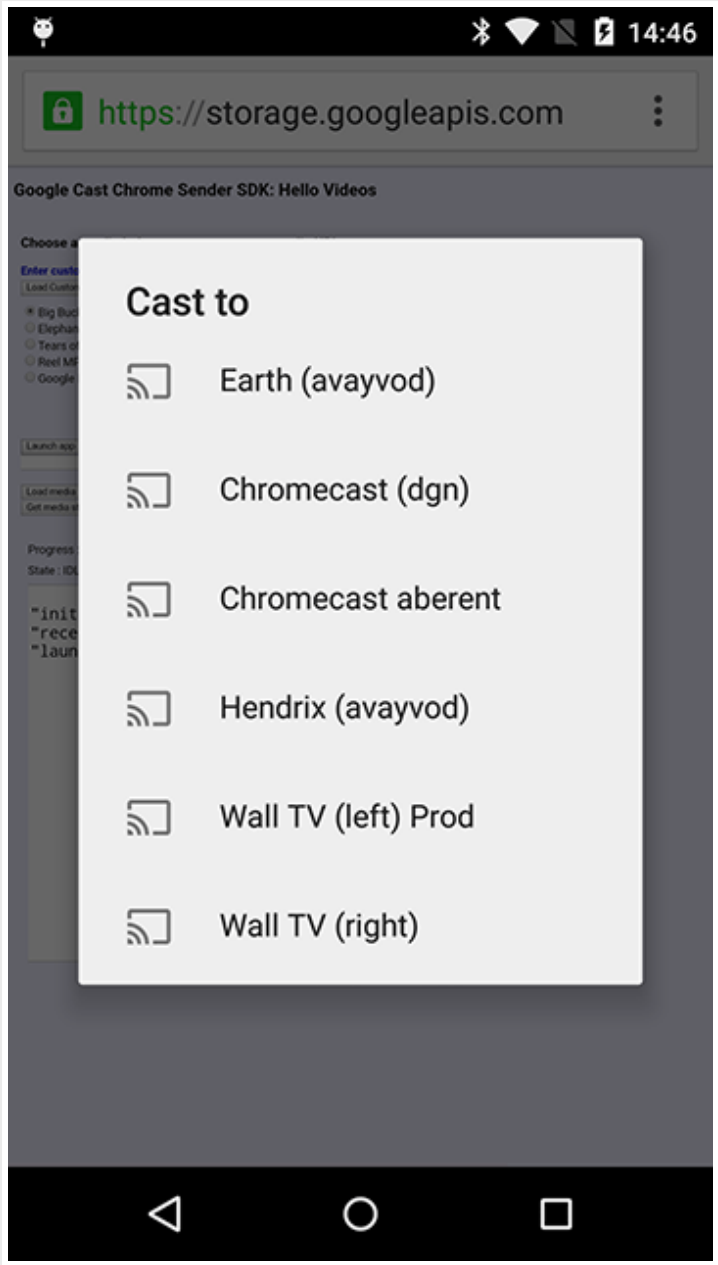
The **`receiverListener`** callback above is executed when one or more devices becomes available:

```
function receiverListener(e) {
  if (e === chrome.cast.ReceiverAvailability.AVAILABLE) {
    // update UI
  }
}
```

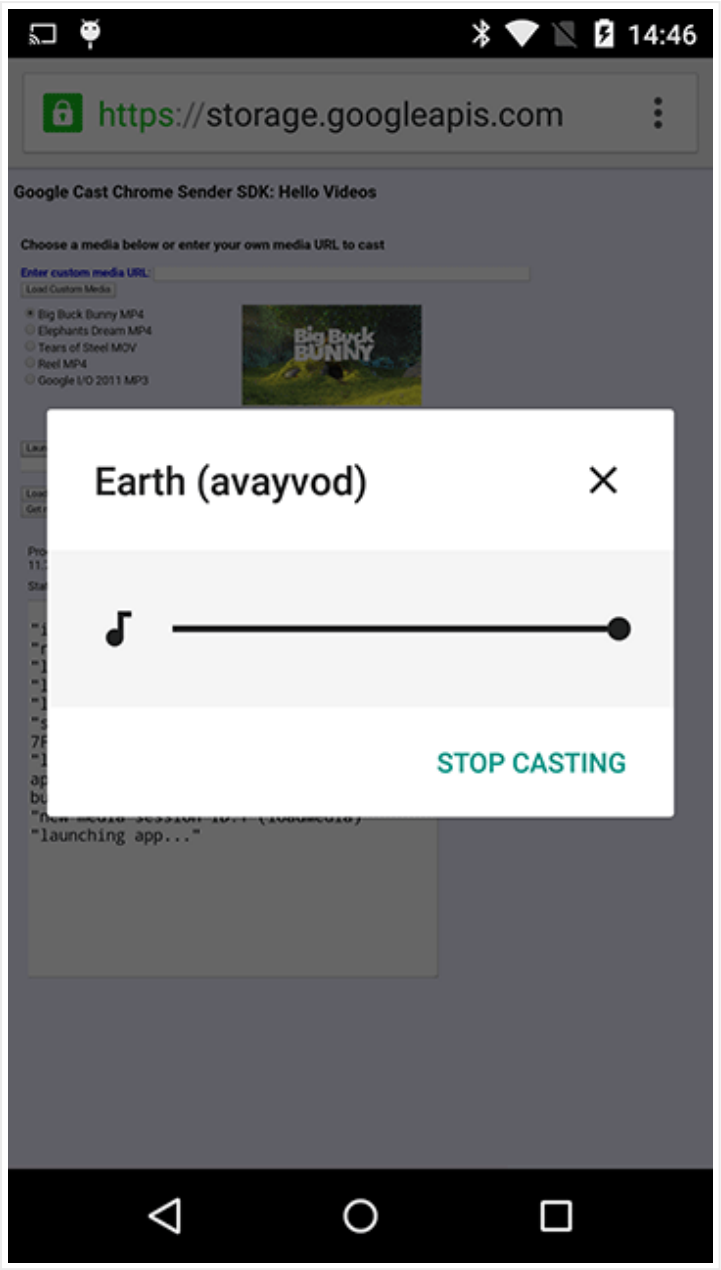
Launch a Cast session when your user clicks the Cast icon, as mandated by the User Experience Guidelines:

```
chrome.cast.requestSession(onRequestSessionSuccess,
    onRequestSessionError);

function onRequestSessionSuccess(e) {
  session = e;
}
```

The user will be presented with a device picker:



The **route details** dialog is shown when the page is already connected and calls `requestSession()`:



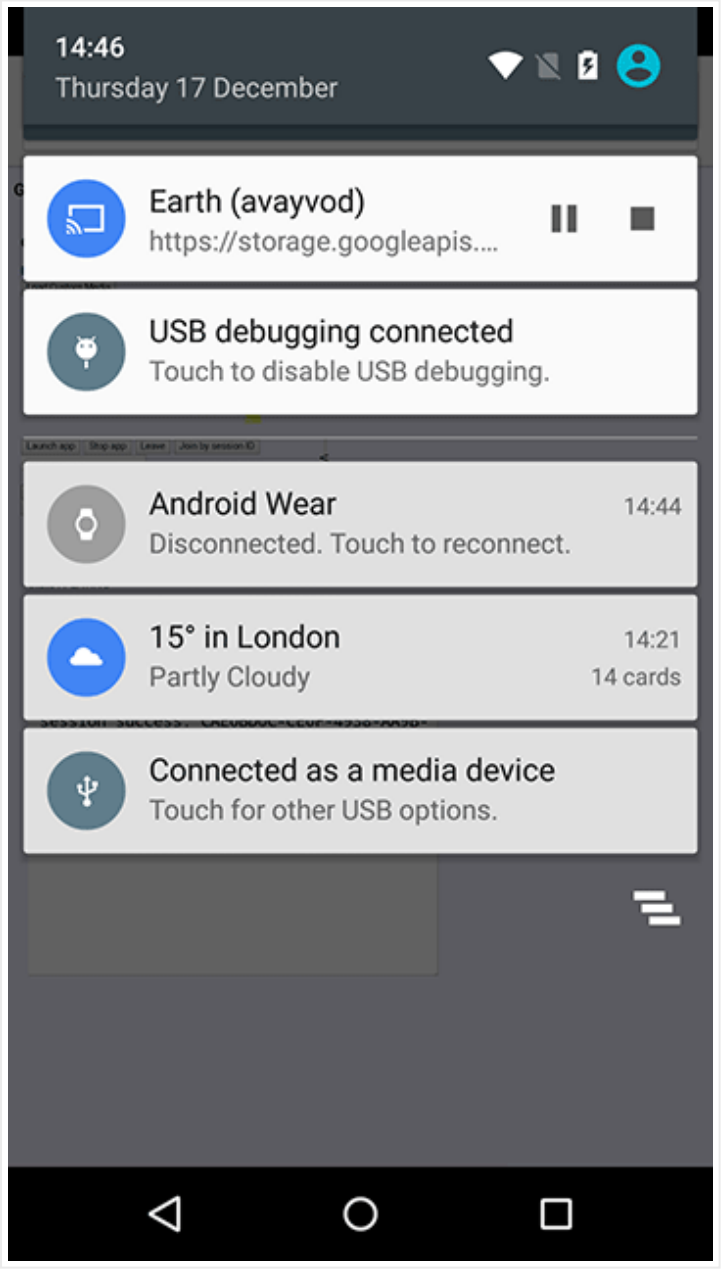
Once you have a Cast session, you can load media for the selected Cast device, and add a listener for media playback events:

```
var mediaInfo = new chrome.cast.media.MediaInfo(mediaURL);
var request = new chrome.cast.media.LoadRequest(mediaInfo);
session.loadMedia(request,
    onMediaDiscovered.bind(this, 'loadMedia'),
    onMediaError);

function onMediaDiscovered(how, media) {
  currentMedia = media;
  media.addUpdateListener(onMediaStatusUpdate);
}
```

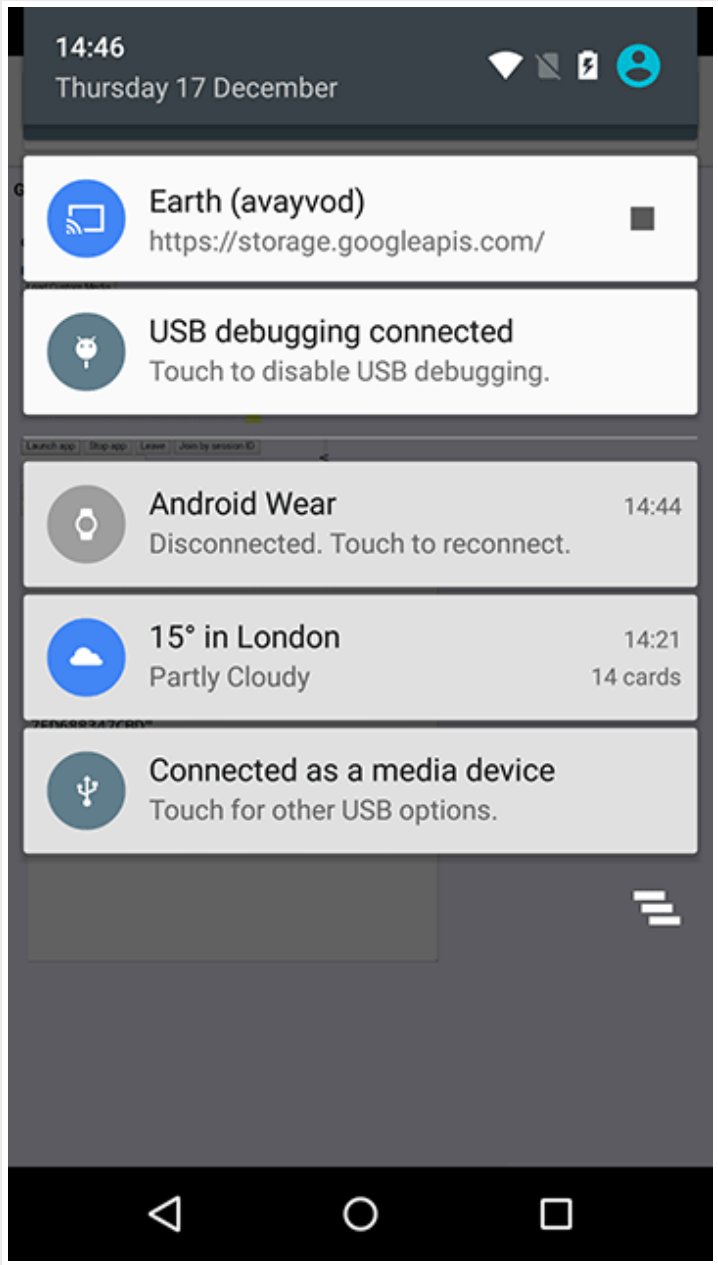
The `currentMedia` variable here is a `chrome.cast.media.Media` object, which can be used for controlling playback:

```
function playMedia() {
  currentMedia.play(null, success, error)
}

// ...
```

A play/pause notification is shown when media is playing:



If no media is playing, the notification only has a stop button, to stop casting:

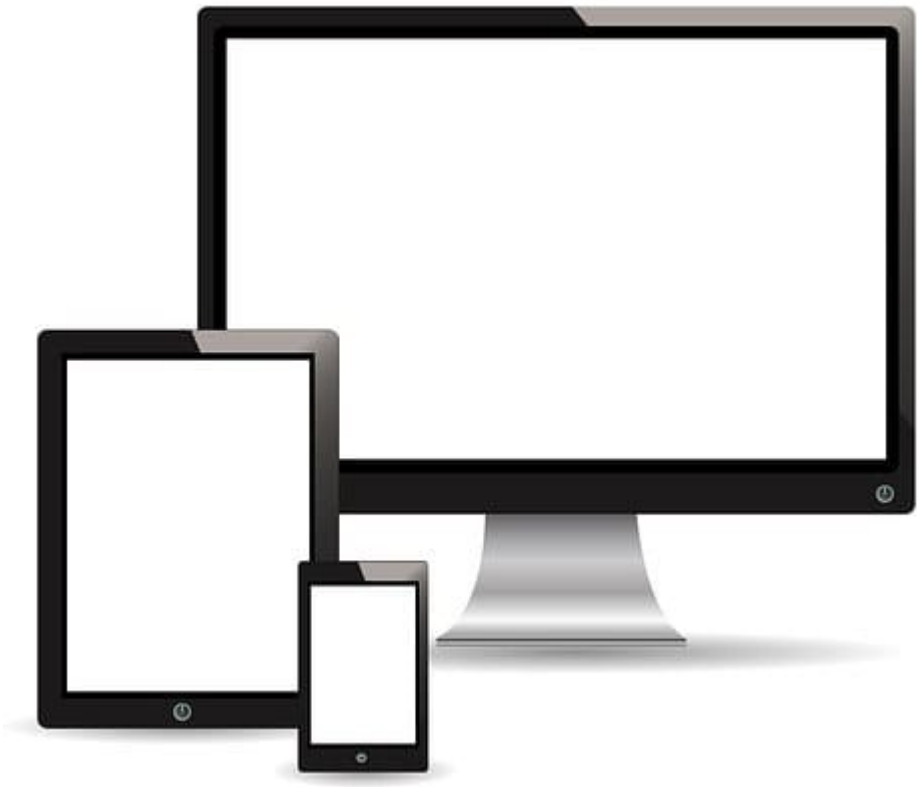


The `sessionListener` callback for `chrome.cast.ApiConfig()` (see above) enables your app to join or manage an existing Cast session:

```
function sessionListener(e) {
  session = e;
  if (session.media.length !== 0) {
    onMediaDiscovered('onRequestSessionSuccess', session.media[0]);
  }
}
```

If Chrome on Android allows casting media from your website but you want to disable this feature so the default casting UI doesn't interfere with your own, use the disableRemotePlayback attribute, available in Chrome 49 and above:

```
<video disableRemotePlayback src="...">
```



The Cast Web SDK guide has links to sample apps, and information about Cast features such as session management, text tracks (for subtitles and captions) and status updates.

At present, you can only present to a Cast Receiver Application using the Cast Web SDK, but there is work underway to enable the Presentation API to be used without the Cast SDK (on desktop and Android) to present any web page to a Cast device without registration with Google. Unlike the Chrome-only Cast SDK, using the standard API will allow the page work with other user agents and devices that support the API.

The Presentation API, along with the Remote Playback API, is part of the Second Screen Working Group effort to enable web pages to use second screens to display web content.

These APIs take advantage of the range of devices coming online — including connected displays that run a user agent — to enable a rich variety of applications with a 'control' device and a 'display' device.

We'll keep you posted on progress with implementation.

In the meantime, please let us know if you find bugs or have feature requests: crbug.com/new.

Find out more

- Get Started with Google Cast SDK
- Presentation API spec
- Use cases and requirements
- The Second Screen Working Group
- Remote Playback API

Last updated July 2, 2018.