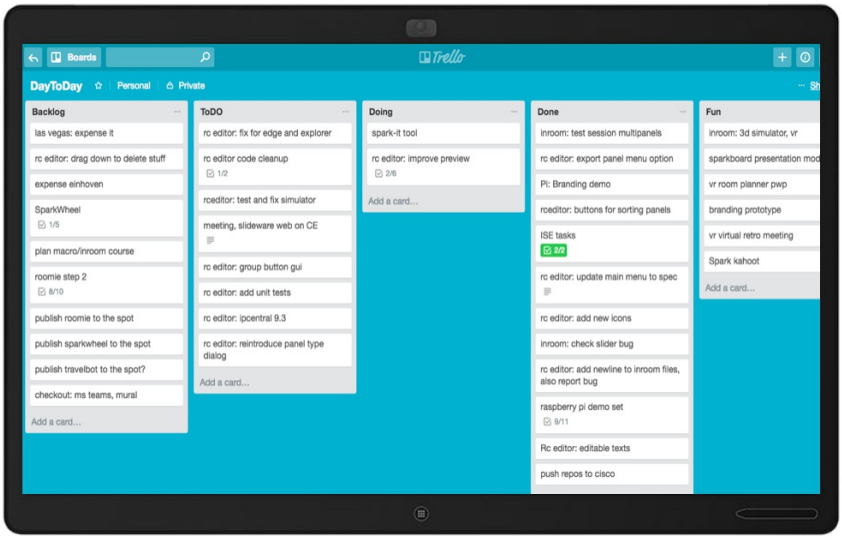
RoomOS WebEngine Developer Guide



Version: 0.3 / September 2019

Purpose and audience

This guide describes the feature set of RoomOS web engine for Cisco's Webex Devices. The target audience is web developers or stake holders that need a technical overview of the capabilities and limitations of the web view.

Separate documentation details how to enable the settings in Control Hub and configure the web engine, and will not be covered here.

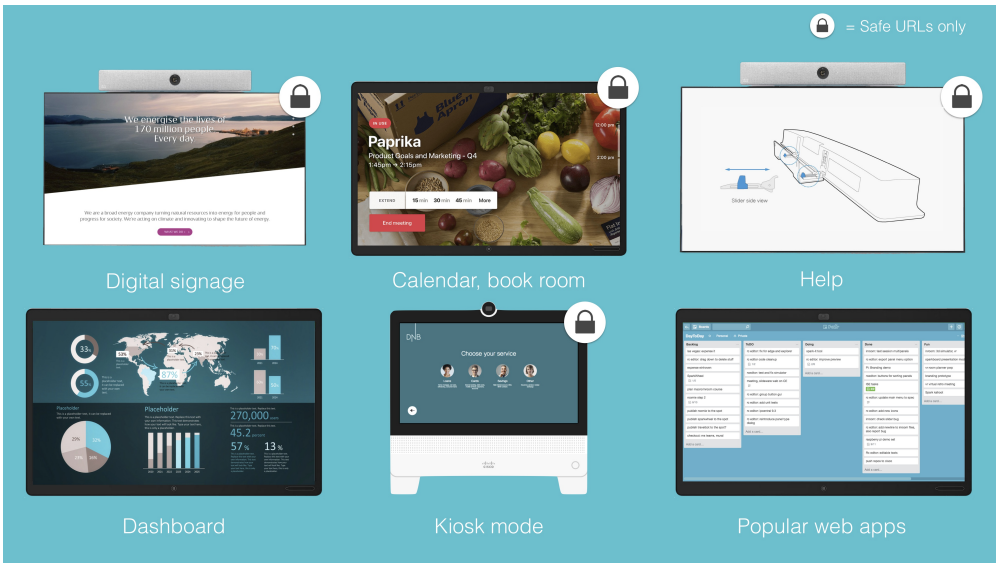
Short summary

Don't have time to read all the details? Summary: It is a single-tab Chromium browser with most modern web technologies available, running on an embedded device. Use web standards and develop with a slightly older iPad web browser in mind, and you should be fine.

What are the typical use cases?

- Digital signage
- Dashboards
- Digital collaboration tools such such as Trello, Realtime Board
- Facility services such as building maps, room bookings

- Help, instructions
- Games, ice breakers



Which browser engine is used?

The web engine is based on Chromium / Qt WebEngine with V8 JavaScript. The Chromium version is upgraded every time Cisco updates the Qt version in RoomOS, meaning it will not be as up-to-date as your Chrome laptop version, but will be updated periodically.

The version at the time of writing is Chrome 65. You can inspect the version at any time by looking at the user agent, eg by visiting http://whatsmyuseragent.org/ on your device.

Which Cisco devices are web compatible?

All 'newer' devices, such as:

- Webex Board 55, 70 and 85 (touch enabled)
- RoomKit, RoomKit Pro, RoomKit Plus, RoomKit Mini
- Room 55 and Dual, Room 70 and Dual

As of CE 9.9, the web engine is available on both cloud and on-premise deployments.

Digital signage

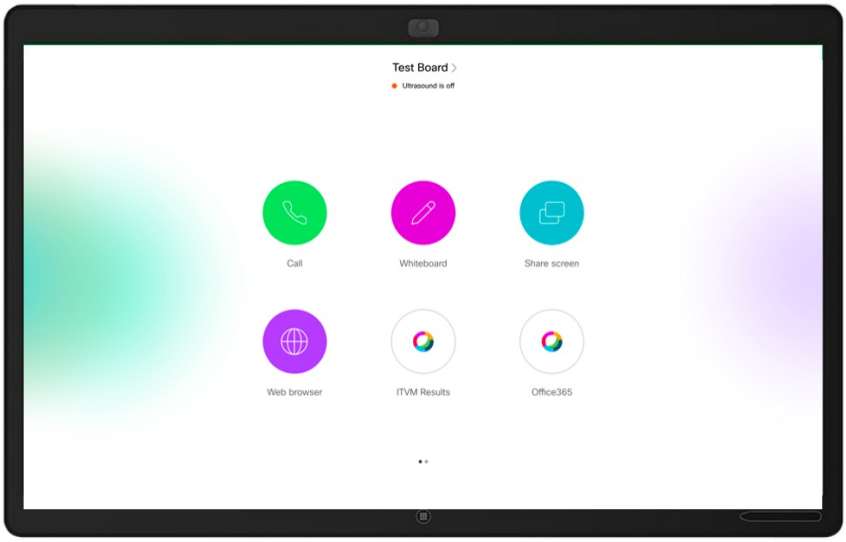
This web view is loaded whenever the system is not used for a while. The admin can configure which URL will be loaded, and whether the web supports user interactions or not. The signage web view shows a footer at the bottom to inform the user about how to exit the mode.

In this mode, data is persisted (cache, cookies, localstorage etc).

2. Web Apps

Web apps are available as activity icons on the home screen. The admin can configure a list of URLs and names. The app launches in full screen, and will time out after 15 minutes of not being in use.

Typical apps can be Office 365, Trello, Wikipedia, YouTube, or company internal web pages and tools.



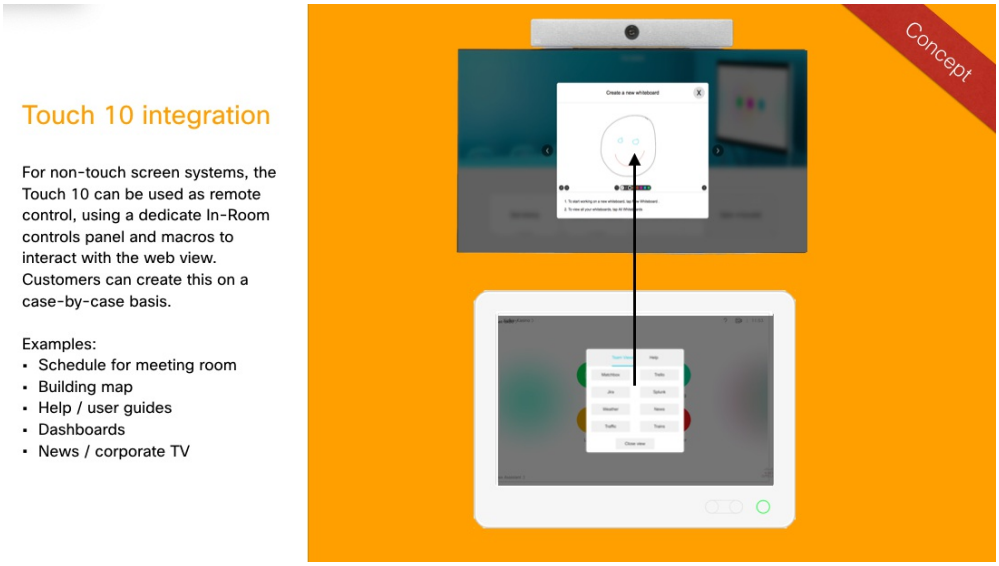
Currently, all data (cache, cookies, localstorage etc) is cleared when the RoomOS session ends.

3. API-driven views

Embedded web views can be opened from API such as third party integrations or macros. The integrator decides which URL to load based on external events. Examples can be to show important company alerts. The views are fullscreen and will time out after 15 minutes, or by calling the API command to close the view.

```
xCommand UserInterface WebView Display Url: <url>
```

For systems without touch screens, this also allows basic integrations with the custom In-Room Control buttons and macros on Touch 10, for example to open and browse basic help pages or show instructional videos.

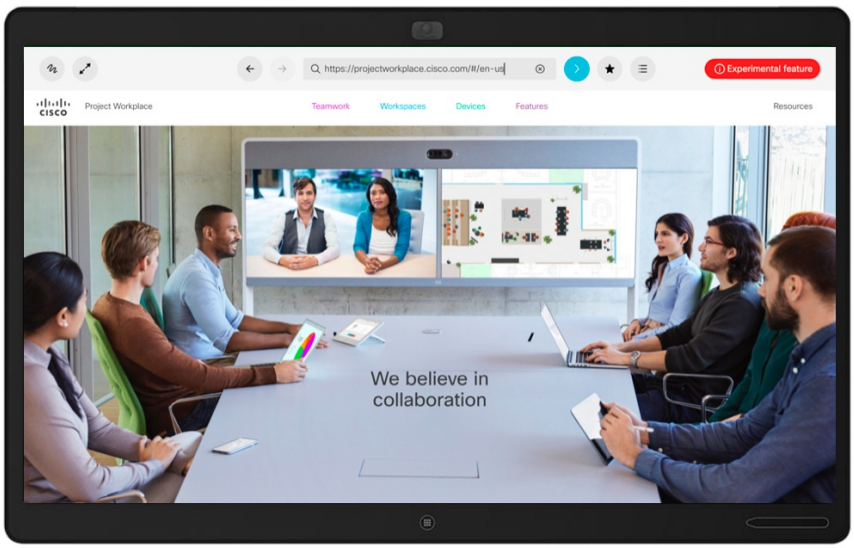


<--

4. Web browser

This is the ultimate in freedom and power for the end user, letting them enter URLs, save bookmarks and navigate in history.

The web browser is currently mostly for convenience and in-house testing, and may only be made generally available if user experience, performance and security aspects are satisfactory.



-->

Developer console

The remote developer console is by far the most important and powerful tool available to third party developers. This is the standard Chrome dev console, that you can run remotely on your own laptop to inspect what is happening in the web engine, see the log console and manipulate the content live, just like with local web development.

The developer console need to be turned on from the Control Hub, with the setting

```
xConfiguration WebEngine RemoteDebugging: On
```

When enabled, all web views will display a prominent warning, to warn the user that they may be monitored, for privacy reasons. The info bar also shows the URL that you can type into your local Chrome browser to open the console. Be sure to disable the console again after use. Note that the developer console only works with the Chrome / Chromium browser. Also note that you have to specify ip address, not hostname, to access it.

What hardware resources are available?

Most of the currently supported video systems, including the Webex Board, is based on the Nvidia Tegra chip set, with 4 GB of total memory.

However, the web engine has lower priority than the main video features and is restricted. Currently it will be limited to roughly 650 MB of memory (excluding GPU usage). If the web page requires more memory than allowed, Chromium will try to optimise usage with the memory pressure handler. Failing this, the web view will eventually be terminated and show an error page. The web view may also be terminated the if the video system is running low on memory in general.

For the Room Kit Pro, there are 6 cores and 8 GB of memory, but the restrictions are currently the same as the other devices.

These restrictions will be monitored in the field, and possibly adjusted over time to balance the

need of custom web content versus the traditional video system features.

What browser features are supported / not supported?

Since the browser is based on a standard Chromium browser, most of the features you expect from a modern desktop browser is available.

Features such as HTML5 tags, EcmaScript 6, CSS3, web fonts, multi-touch, SVG, canvas, iframes, web sockets, web assembly, web workers and more are available.

Notably, these features are not currently supported:

- PDF
- WebGL (disabled for now)
- WebRTC
- Password manager
- Plugins (Flash, etc)
- Downloading and uploading files
- Notifications

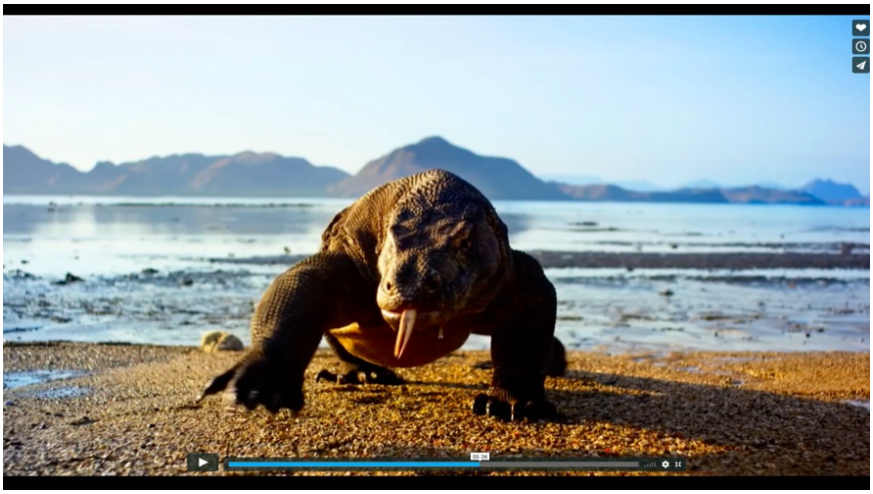
Some of these features may be added later based on external demand.

How many windows / tabs are supported?

Only one web tab / window is supported. If a web page tries to open a page in a new window or tab, it will replace the existing page.

Is multimedia supported?

Standard video codes are supported, such as WebM and mpeg4. Decoding is done through software. It is not recommended to go beyond 1080p resolution, as this will lead to choppy performance.



What about volume?

The volume follows the volume setting of the video system. For digital signage, the audio is off by default to avoid unwanted noise by accident, but it can be enabled with an xAPI config:

```
xConfiguration Standby Signage Audio: <on/off>
```

You can of course to enforce volume and mute control on top of this by using the JavaScript audio frameworks.

What does the user agent look like?

The web view's user agent follows the same pattern as the popular browsers, and adds RoomOs and the device type. Example:

```
Mozilla/5.0 (Linux; RoomOS; Cisco Webex Board (70) AppleWebKit/537.36(KHTML, like Gecko) QtWebEngine/5.11.2 Chrome/65.0.3325.230 Safari/537.36
```

This makes it possible to adapt your web page specifically for the Cisco devices, if needed, and also to track how much of your web traffic that comes from Cisco devices.

What's the security story?

In short, the security model is similar on Chromium. The renderer process runs in a sandbox, and all system calls are filtered. Even if a malicious site is able to exploits vulnerabilities, it should never be able to access anything outside the renderer process itself.

For root CAs, we use the same bundle as RoomOS. Work is in progress to allow customers to

add and remove root CAs specifically for the web engine.

Chromium also uses the proxy settings of RoomOS, if the admin has enforced it.

We apply security patches regularly in addition to the full Qt upgrades.

For more details, see the separate security FAQ.

Is user data persisted?

For digital signage, web engine is running in persistent ("desktop browser") mode, meaning that cookies, local storages etc are saved to disk and persisted across reboots and even upgrades. This means its possible to have persisted logins for team dashboards etc.

Currently the cache size is restricted to 250 MB.

Web apps and the browser currently runs in "incognito" mode, but we may offer to configure this later.

Are touch events supported?

Yes, the Webex board supports up to 10 simultaneous touch events, using `ontouchstart`, `ontouchmove` and `ontouchend`.

Note that the ordering of touch events is not stable, so use the touch event `identifier` to keep track of simultaneous touches.

The traditional `onclick` event is also supported. Be aware that the web engine, like mobile browsers, is "slow" to react to onclick events, since they are fired after a 100 ms wait where the browser looks for gestures. It is therefore recommended to use ontouchstart or ontouchend, as the onclick behavior make the app appear sluggish.

How to prevent the standard zoom gestures

Two finger zoom is default for accessibility and convenience, but not always wanted, for example in immersive apps such as Google Maps, whiteboards and games, where you need more control. You can override this with `preventDefault`:

```
document.querySelector('.myCanvas').ontouchstart = (e) => {
  e.preventDefault();
```

```
  // ... my action
}
// similar for ontouchmove, ontouchend
```

Be careful doing this on the whole document though, as prevent default on for example form elements such as input fields and textareas might cause them to behave poorly, eg not getting focus.

What's the viewport / screen size?

The logical viewport is 1920 pixels wide, and 1080 pixels high minus whatever is used for the toolbar (typically less than 200 pixels).

The actual rendering is done on a 4k canvas (3840 x 1260), similar to Apple's *Retina* mechanism, so text and images will be crisp. Be sure to provide high resolution images and assets for optimal user experience.

Developer can also modify the viewport using the viewport meta tag, eg:

```
<meta name="viewport" content="width=960, initial-scale=1">
```

Are the JavaScript dialogs available?

Yes, most JavaScript dialogs such as alert, prompt and confirm work like on a desktop and is implemented using the native dialogs of the video system.

Upload and download dialogs are not supported.

What fonts are available?

Currently only the system font of RoomOS (sans serif). Web fonts are supported though, so third party app developers can add their own fonts as needed with CSS, eg:

```
@font-face {
  font-family: handwriting;
  src: url(handwriting.woff);
}

div {
  font-family: handwriting;
}
```

Login / authorisation

One of the most challenging aspects for a web app on a shared video device is the user login. We cannot just store the users credentials like on a personal device, since it can be accessed by anyone, and entering the username and password on a large-screen, soft keyboard device is both annoying and unsafe.

We will strive towards a solution that makes this easy and safe in the future. In the meantime, the model solution is to use a second, personal device for authentication and authorisation.

User flow:

- The user launches the web app on the video device
- The user is prompted and enters their email
- The web app sends a notification to the user's phone
- The user accepts the login on the phone
- The web app is automatically logged in on the video device

The Microsoft Authenticator for Office 365 an example of how this can be implemented in a way that is convenient, fast and safe for the user.

What animations are supported

Both CSS transitions and web animations are supported and hardware accelerated whenever possible. This typically means the `transform` and `opacity` CSS property. Try to avoid doing animations that requires DOM or layout operations.

Can I create offline applications?

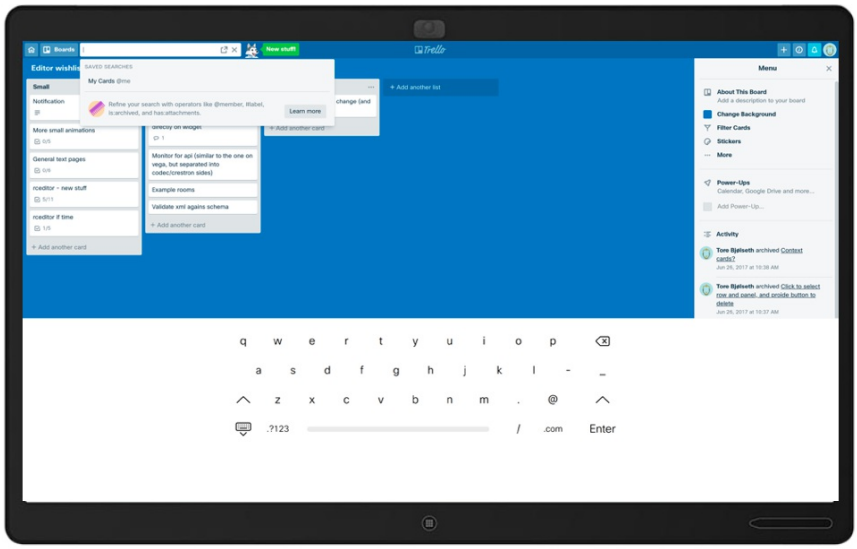
Yes. Web workers are supported, and can be used to speed up initial loading of heavy web apps. Note that currently the web views are not available if the device does not have network, so real offline apps are not possible.

Keyboard input

The RoomOS soft keyboard behaves similarly to the touch keyboards on Android and iOs, and pops up any time an input field gets focus. The content also scrolls up if needed so both the input field and keyboard are visible.

It does not support specialised formats such as numeric, calendar and colour picker at the moment.

A vertical soft keyboard does not encourage a lot of text input and provides little privacy, so keep that in mind.



Internationalisation

The web engine sends the language header of the current language in each HTTP requests, eg `Accept-Language: fr` for French. A web app can then translate the content to the current language, either server side by looking at the header, or in JavaScript by querying `useragent.language` in the browser.

Non-fullscreen windows?

Currently only fullscreen web views are supported. In the future we may support different types such as web dialogs and side-bars.

What about multiple screens?

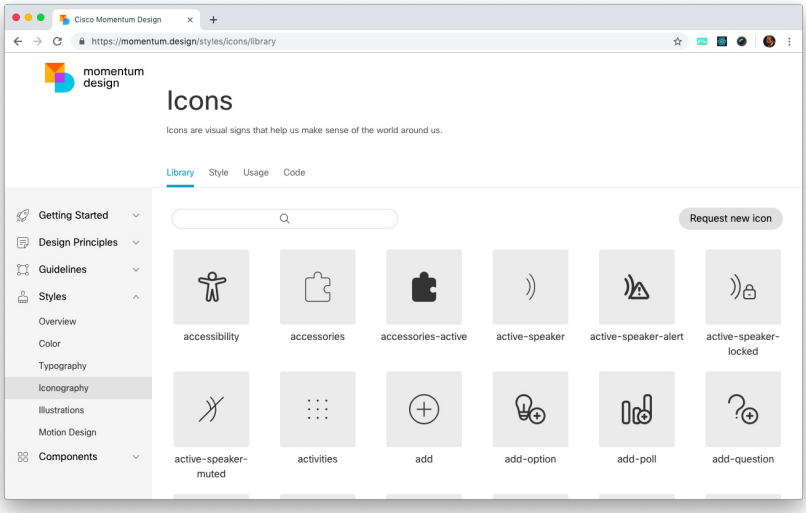
For digital signage, content is cloned to the extra screens. A web view cannot be logically stretched across multiple screens. In other cases than signage, only the primary monitor will display the web view.

Best practises

Writing web apps for a shared device with a huge touch screen comes with its own set of challenges. A dedicated best practise manual may be offered later, but here are a couple of pointers:

- The screens are large and the user can be very close to it - choose your content based on wether you prefer interactive use (smaller font and content) or passive audience (larger font and content).
- Provide high resolution assets for crisp graphics
- Only use hardware accelerated CSS animations (transform and opacity)
- If possible, provide a second, personal device for log in such as Microsoft Authenticator
- Use local storage to store temporary user data, to prevent data loss if the web view is closed by accident
- Don't rely on multiple tabs

Style guide



As a third party developer you do of course have the tools and freedom to design the web content exactly as you like or to match your company's visual profile.

If you prefer something that feels native and in line with the Cisco design language, you can find everything you need at https://momentum.design/.

As well as styles and guide lines, this web site contains CSS, assets, fonts and icons and component support for popular web frameworks.

Can the web content access the xAPI?

At the moment, not directly.

The Cisco video systems provide rich and powerful API integrations through the xAPI, such as making calls, starting presentations, counting the number of people in the room etc. Currently a web view does not have direct access to the xAPI, but it can access it the same way as other integrations, provided that credentials to the video systems are available. Alternatives include local REST APIs (on premise deployments), HTTP feedback (web hooks), cloud APIs or direct web socket communication.

For more info about these, see separate RoomOS developer documentation. Be careful about exposing credentials in your client side code, though, and consider going through your web app's server.

What about non-touch devices?

At the moment, only the Webex Board has touch screen capability, but all the new Cisco video devices such as Room Kit, Room Kit Mini, Room 55 etc also support web view. The most obvious use case is digital signage, but soon you will also be able to open web views programatically through the xAPI. See the section on API-driven web views above.

Trouble shooting

Are there logs available?

At the moments, there are no logs for the web engine. In many cases the developer console will give you all the info you need.

How to improve performance

The web engine has the difficult task of running modern, full scale web content on a 4K screen while having lower priority than the main video features. This means it can sometimes struggle with heavy web sites such as Google Maps 3D and similar.

But there are many things a developer can do to improve performance. This is worthy of its own guide, but here are a few things you can try:

- Start testing on the device early in development, if possible, or use a mobile device
- Avoid using images that are larger than needed, resize them on the server

- Reduce the number of layers with opacity
- Avoid drop shadows
- Avoid huge canvases, load content dynamically as needed instead
- Use only hardware accelerated CSS / web animations
- Skip certain features, eg animations, based on user agent
- Avoid doing much work in event handlers, delegate to async tasks
- Use `requestAnimationFrame` instead of `setInterval` if you are doing low level animations/simulations
- Move heavy calculations / algorithms to WebAssembly

Best but also most challenging: Learn the performance monitor of Chromium and use it actively.

Why isn't the web page refreshing?

If you are doing changes to your web page, and don't see the changes in the web view (especially in signage mode), you can force a refresh by adding a random parameter eg `?dummy=123` to your signage URL.

How to contact us or report issues?

Go through the normal Cisco channels, such as EFT program and partners.