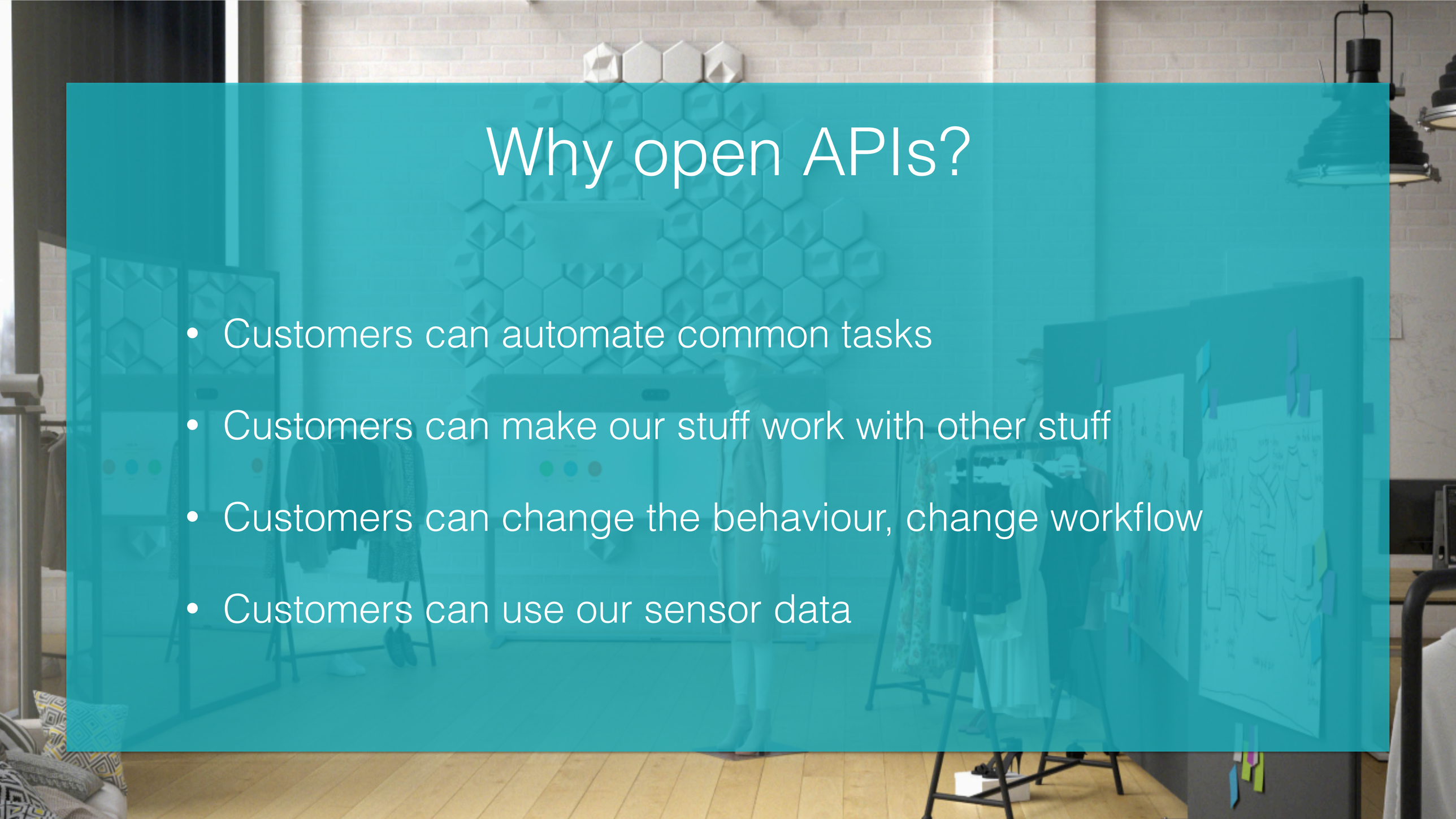
Overview of Webex Device APIs

Tore Bjølseth, SW engineer


Build bridges,
not islands



Why open APIs?

- Customers can automate common tasks
- Customers can make our stuff work with other stuff
- Customers can change the behaviour, change workflow
- Customers can use our sensor data

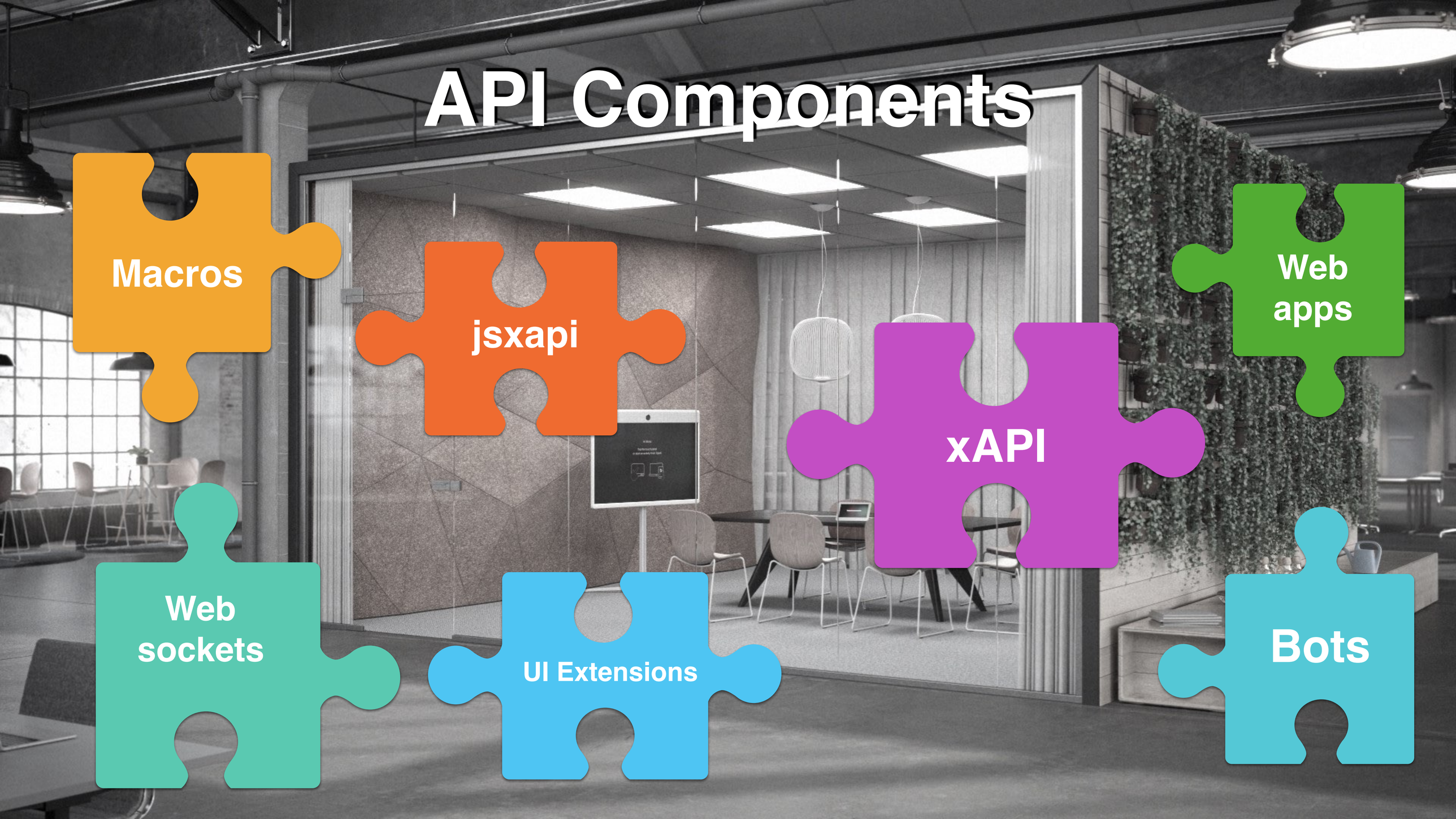
Provided material

Install:

```
git clone https://bitbucket.org/bjolseth/ntnu-hackathon/src/master/
cd ntnu-hackathon
npm install
```

Folder structure

```
README.pdf - Overview of API components
API-overview.pdf - This presentation
examples/ - Runnable examples for each API component
reference-docs/ - Docs for the API components
```


API Components
Macros
jsxapi
xAPI
Web apps
Web sockets
UI Extensions
Bots

xAPI

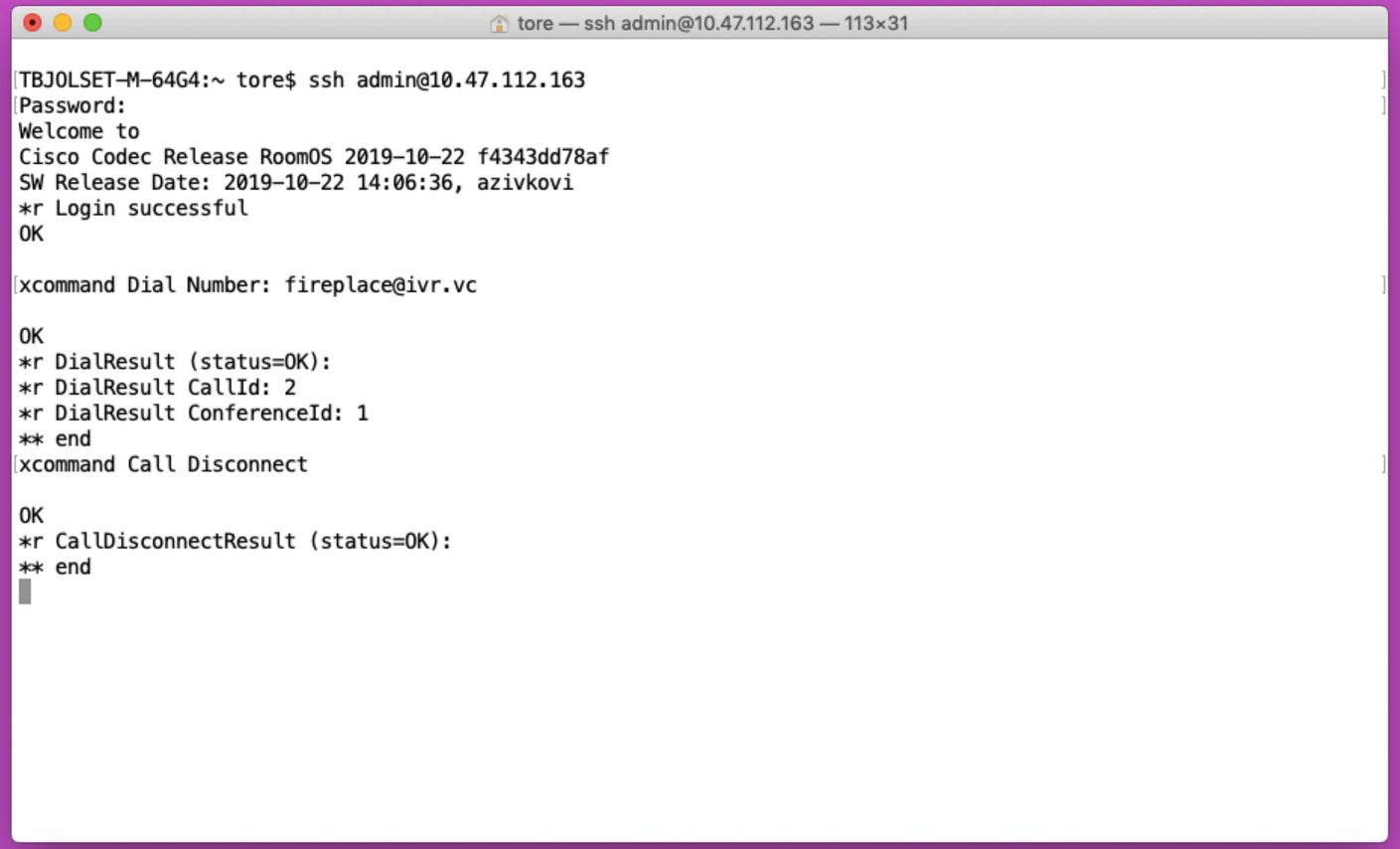
The language that all integrations use to talk to the video system.

Cisco’s user interfaces uses it too.

Easy to test from command line (“TShell”)

Use Tab key to autocomplete, to discover the API

```
> ssh admin@10.47.92.123
```



xCommand

xCommand lets you invoke actions on the system:

```
xCommand Dial Number: fireplace@ivr.vc
OK
*r DialResult (status=OK):
*r DialResult CallId: 2
*r DialResult ConferenceId: 1
** end
```

xStatus

xStatus lets you ask the system about dynamic values:

```
xstatus Audio Volume
*s Audio Volume: 50
** end

xstatus RoomAnalytics PeopleCount Current
*s RoomAnalytics PeopleCount Current: 3
** end
```

xStatus values are **read-only**

xFeedback

xFeedback lets you subscribe to values you want to monitor:

```
xfeedback register status/audio/volume

// when someone changes the volume:
*s Audio Volume: 60
** end
*s Audio Volume: 70
** end

xfeedback deregister all
```

xConfig

xFeedback lets you change system settings permanently

```
xConfiguration Conference AutoAnswer Mode: On
** end

xConfiguration RoomAnalytics PeoplePresenceDetector Mode: On
** end

xConfiguration Standby Signage Url: http://uka.no
** end
```

Configs are persisted on reboot

Try it out

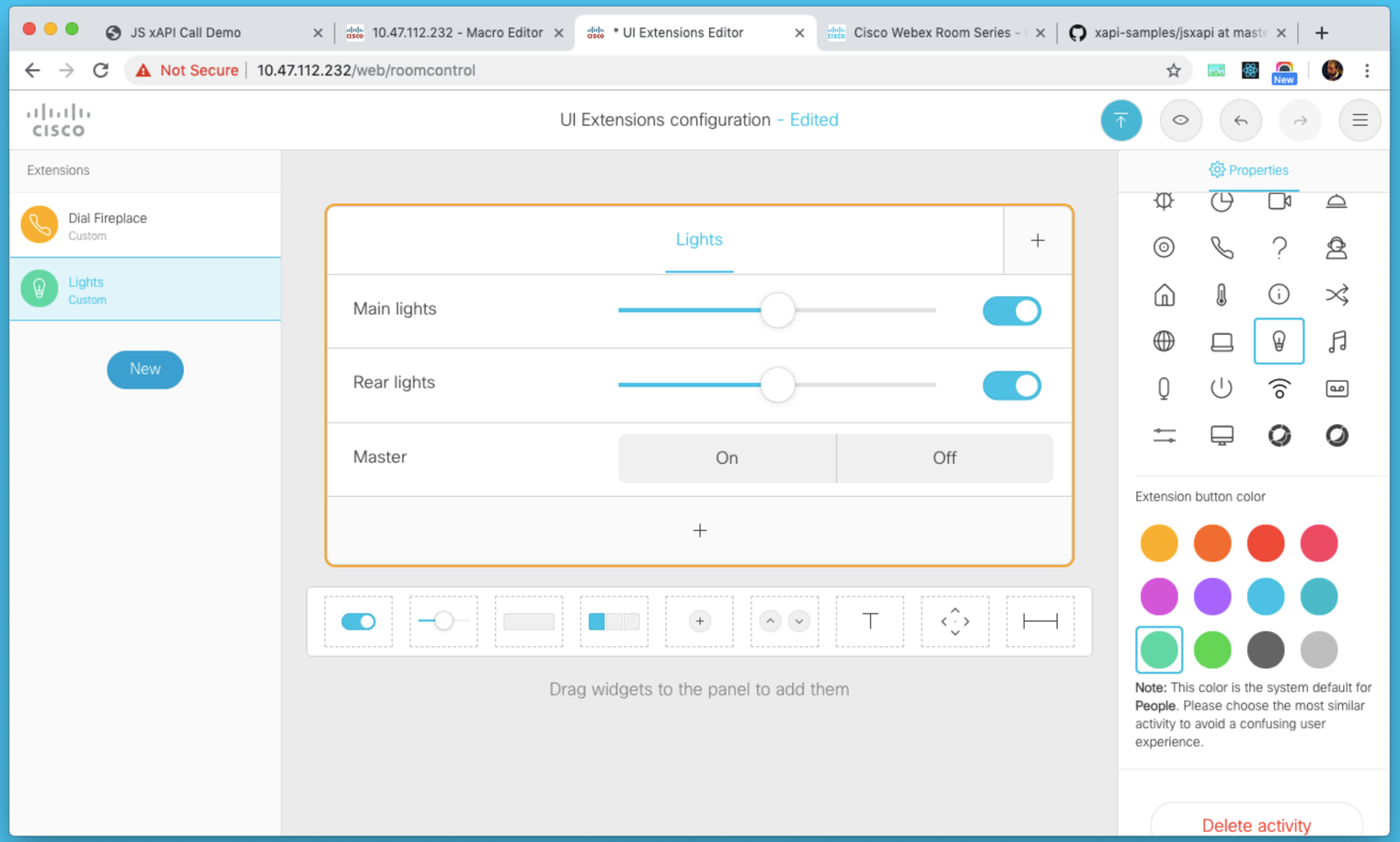
- 💪 Log on to your video device and play with it
- 💪 See more examples in **examples/xapi/README.md**
- 💪 Read tutorial: **reference-docs/macro-tutorial.pdf**
- 💪 Full doc: **reference-docs/customization-guide-ce98.pdf**

UI Extensions

User Interface extensions let you add buttons and panels to the video device.

They can be added with the drag and drop editor available from the web interface of the video device.

The UI Extensions are a configuration / layout, and can be set up without any programming.



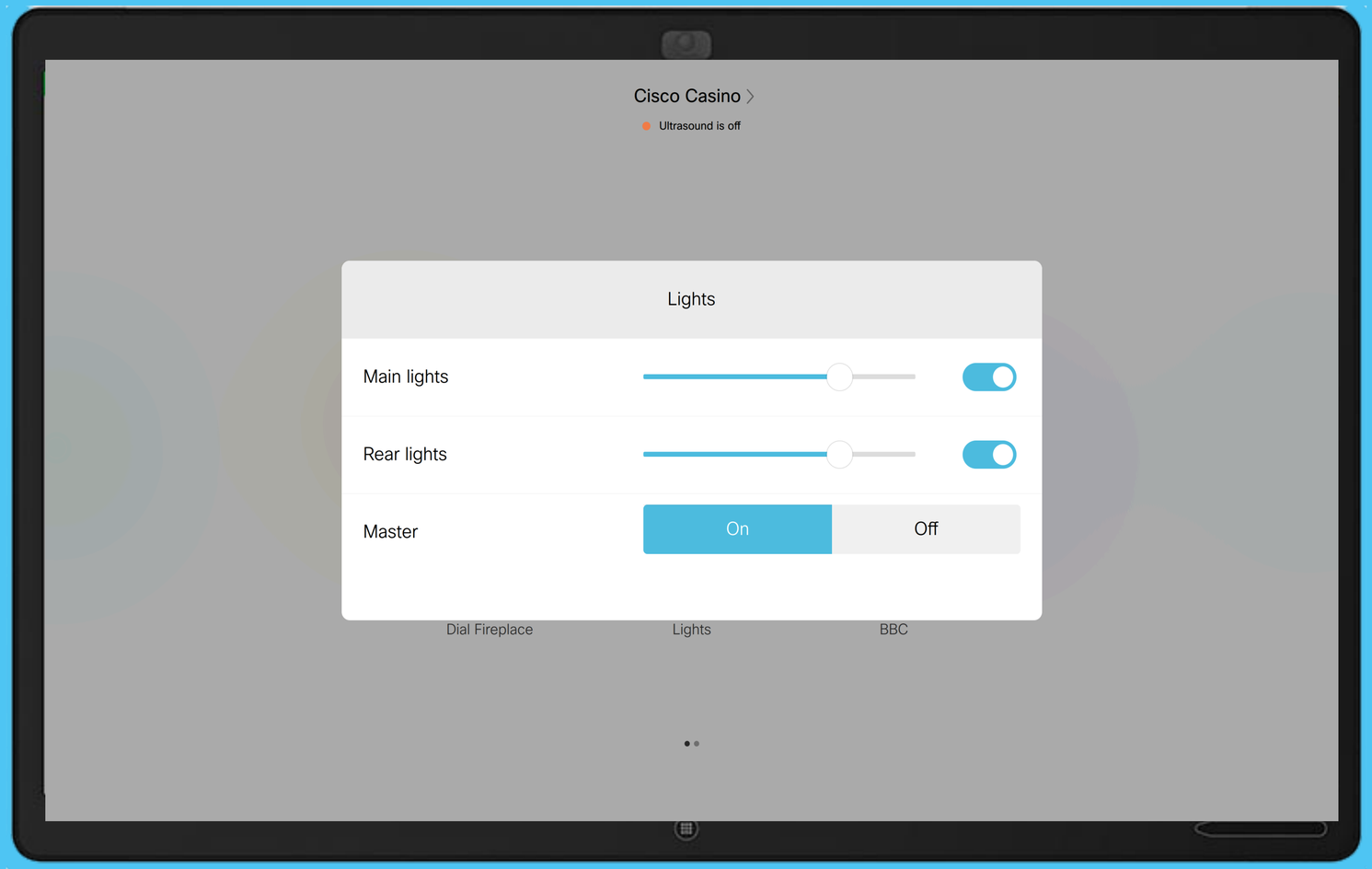
Widgets

Typical widgets are:

- Buttons
- Sliders
- Group buttons
- Toggles

Each widget must be given a unique id.

Then buttons do nothing when pressed other than creating events. You can listen to the events and do everything you want, based on the id and action.



Example use cases

- 🚀 UI to control the lights and blinds
- 🚀 UI to control the climate control
- 🚀 UI to control projector
- 🚀 UI to control Apple TV
- 🚀 UI to control the video layout on the video device

Limitations

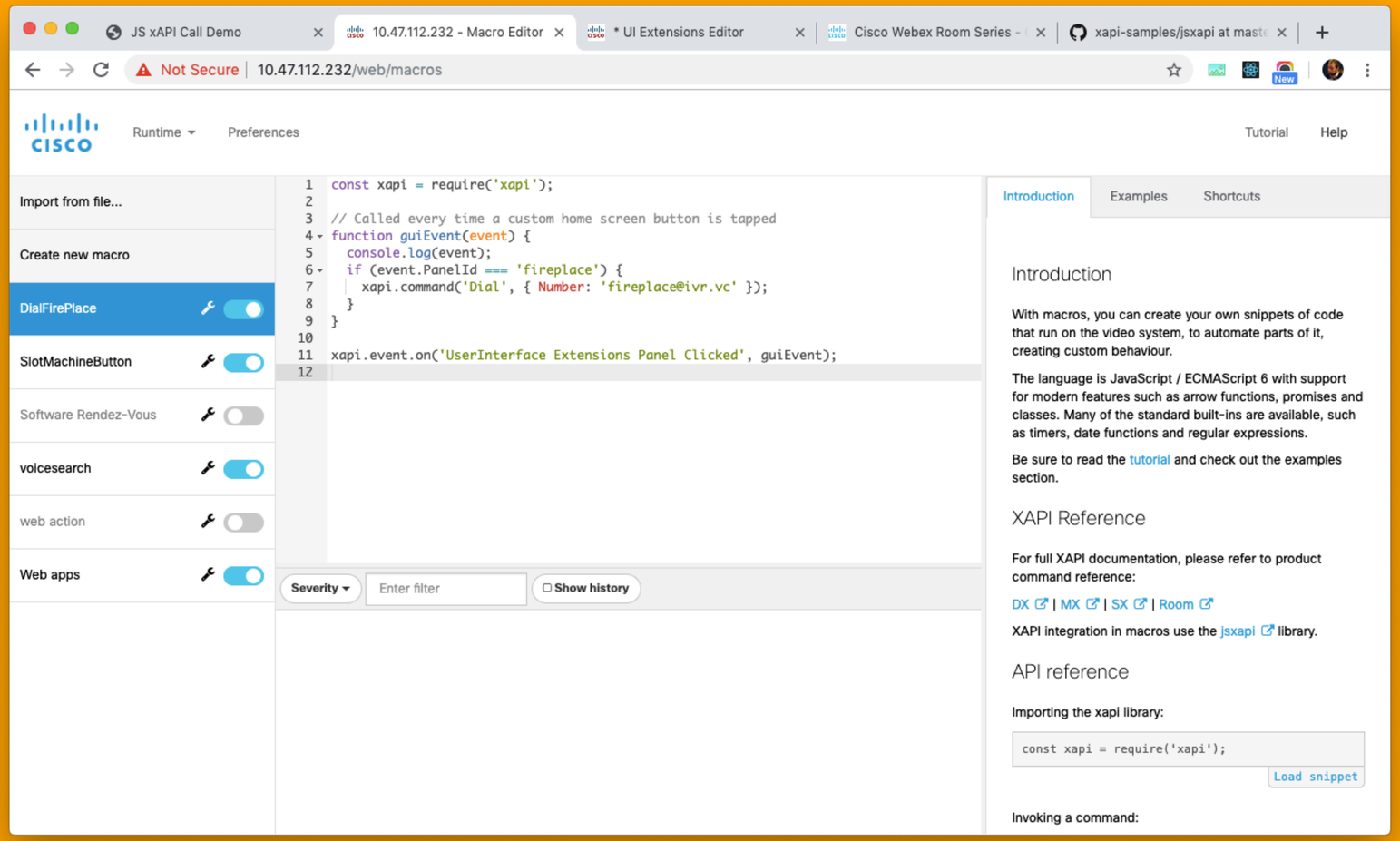
- 🚫 Only simple widgets, not maps, tree structures etc
- 🚫 Can’t add / remove widgets dynamically

Macros

Macros are small snippets of JavaScript code running on the video system itself.

A macro runs continuously and can listen for events, either video system events, user interactions or timers, and perform actions.

Macros can be written in the macro editor, available on the web user interface of the video system itself.

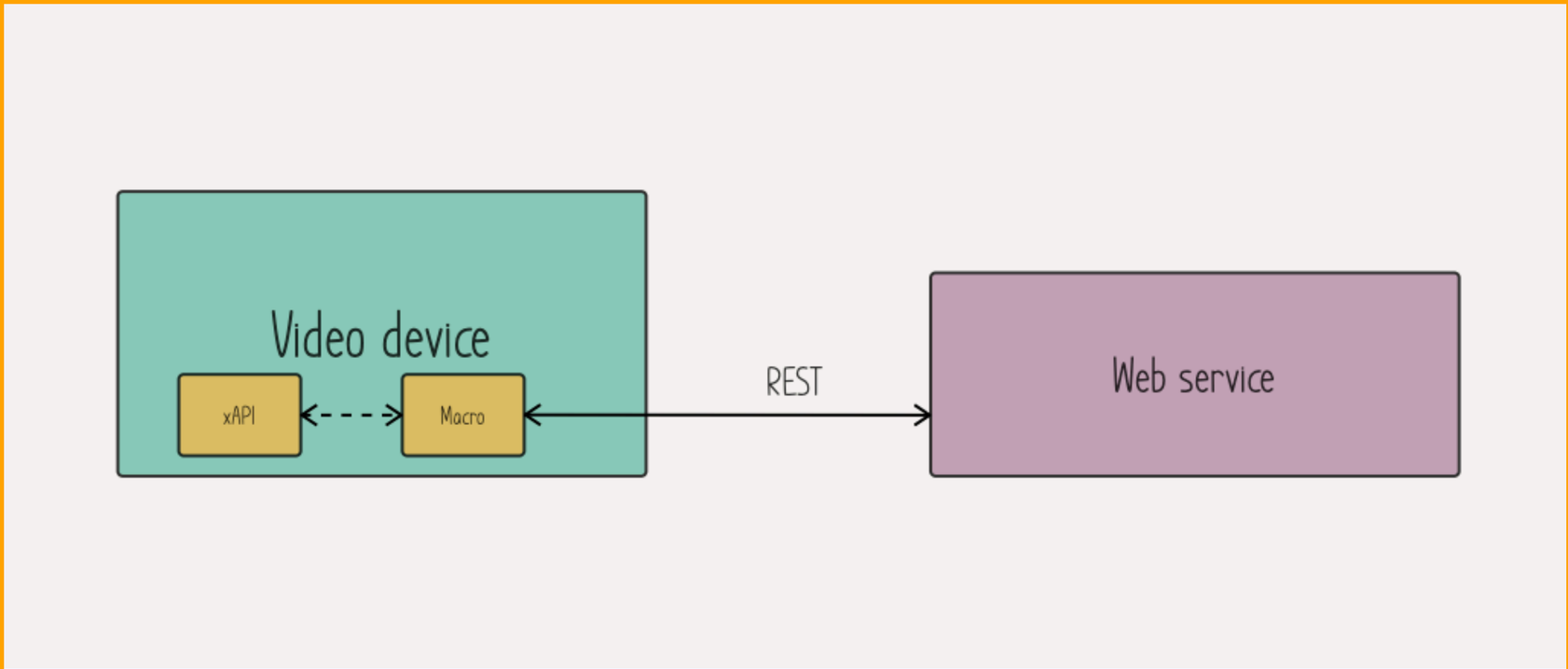


How macros work

Macros talk to the video system using xAPI.

The macros can only use standard JS APIs such as timeout, strings, math etc.

The macros can also talk to external web servers using HTTP GET, POST etc



Example use cases

- 🚀 Control IoT devices in the room (if they support REST)
- 🚀 Quick dials on the home screen to dial a favourite
- 🚀 Shortcuts to settings you change often
- 🚀 Send usage metrics of the room to central server
- 🚀 Automatically start a call at certain points each day
- 🚀 Automatically accept a number if it is from a certain person

Code snippet

```
// The library for talking to the video system
const xapi = require('xapi');

// Called every time a custom home screen button is tapped
function guiEvent(event) {
  if (event.PanelId === 'fireplace') {
    xapi.command('Dial', { Number: 'fireplace@ivr.vc' });
  }
}

// General UI event listener
xapi.event.on('UserInterface Extensions Panel Clicked', guiEvent);
```

Limitations

- 🚫 Macros can only do what the xAPI can do
- 🚫 Macros cannot use external libraries
- 🚫 Macros cannot listen to evens from an external service (no web socket etc). They are self contained
- 🚫 A macro cannot talk to another macro, directly

Try it out

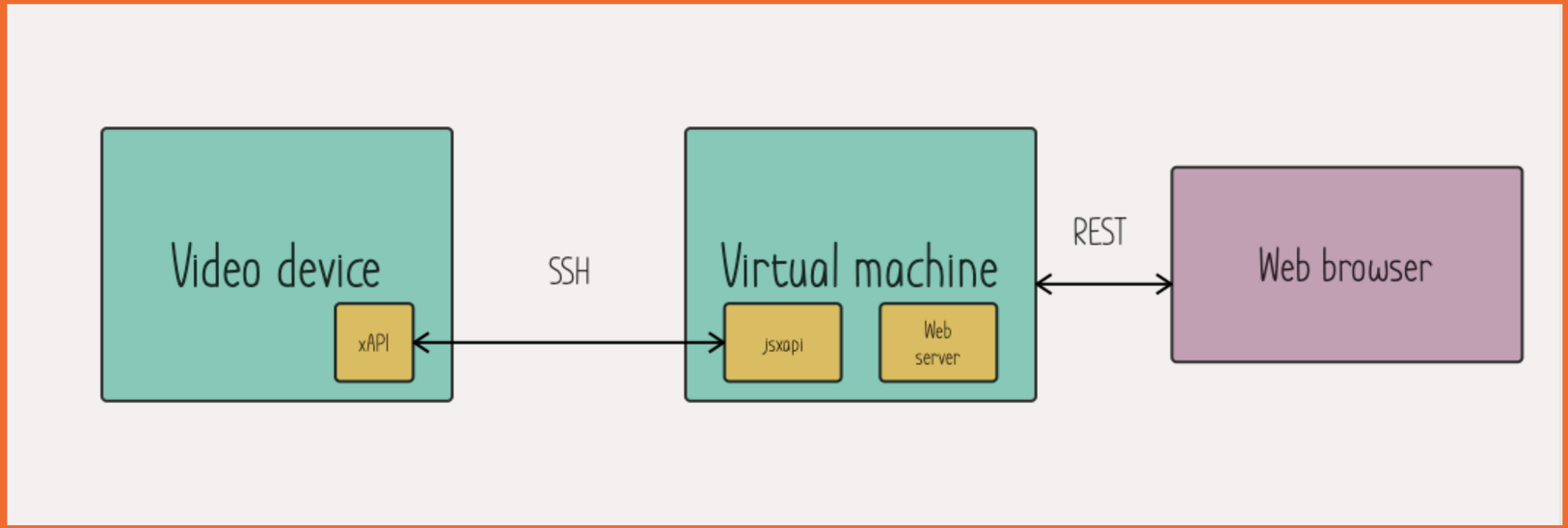
- 💪 Open **examples/macros-user-extensions**
- 💪 Go to the video device’s web interface
- 💪 Launch the Extensions editor, import the xml file
- 💪 Launch the macro editor, import the js file
- 💪 Start the macro
- 💪 Try calling from the custom panel on the video device

jsxapi

jsxapi is a Node.js SDK for talking xAPI to a video device from your own server, eg your laptop or Raspberry PI.

The syntax is the same as the macros, but the code runs on your own server. Therefore, you can do everything that a macro can, but also a lot more.

The jsxapi, unlike the macros, require ip address, hostname and password to connect to the video device.



Example use cases

🚀 Fetch head count number from all meeting rooms for statistics

🚀 Control non-IoT devices in the room that requires proprietary drivers

🚀 Connect a web server with a video device

🚀 Also all the same use cases as the macros

🚀 Show a web page with a lottery wheel of all paired users in the room

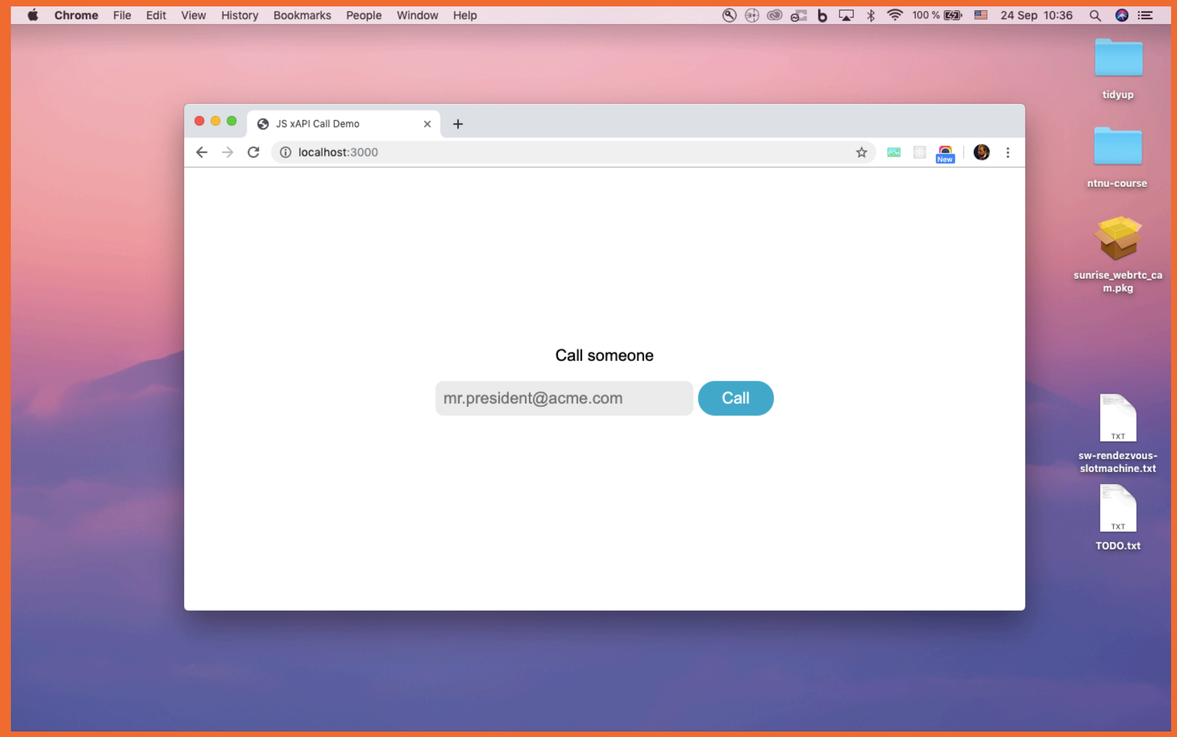
Code snippet

```
// Import the library for talking with the xAPI
const jsxapi = require('jsxapi');

// Replace with your credentials
const host = '10.47.112.123';

// Connect to the video system
const xapi = jsxapi.connect('ssh://' + host, {
  username: 'admin',
  password: 'uka',
});

// Start a call
xapi.command('Dial', { Number: 'fireplace@ivr.vc' });
```

Try it out

🚀 Web server, HTML/JS and jsaxpi to start a call from a web page

🚀 Read **examples/jsxapi/README.md**



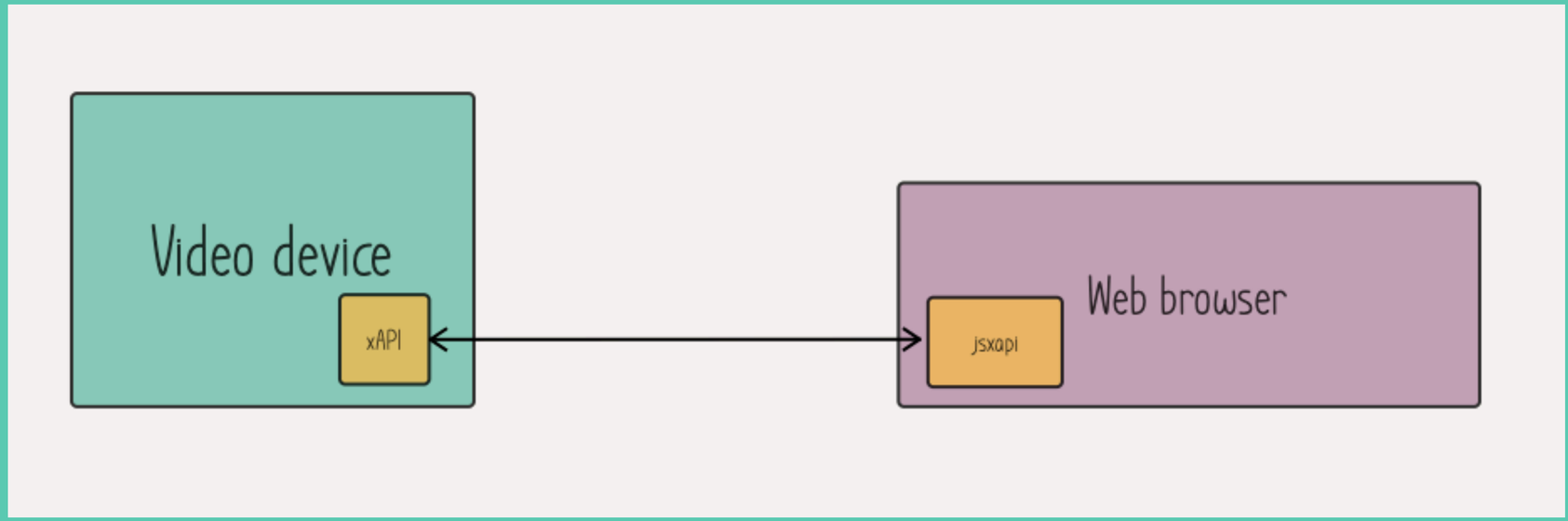
Limitations

🚫 A Node.js server with jsxapi needs to be on the same network as the video device

Web sockets

The same jsxapi SDK that was used by Node.js can also be used in the browser. In this case the browser will talk directly to the video device with web sockets. so you do not need a server at all.

The web page needs to contain the ip address, username and password to the video device, so it should only be available for trusted users.

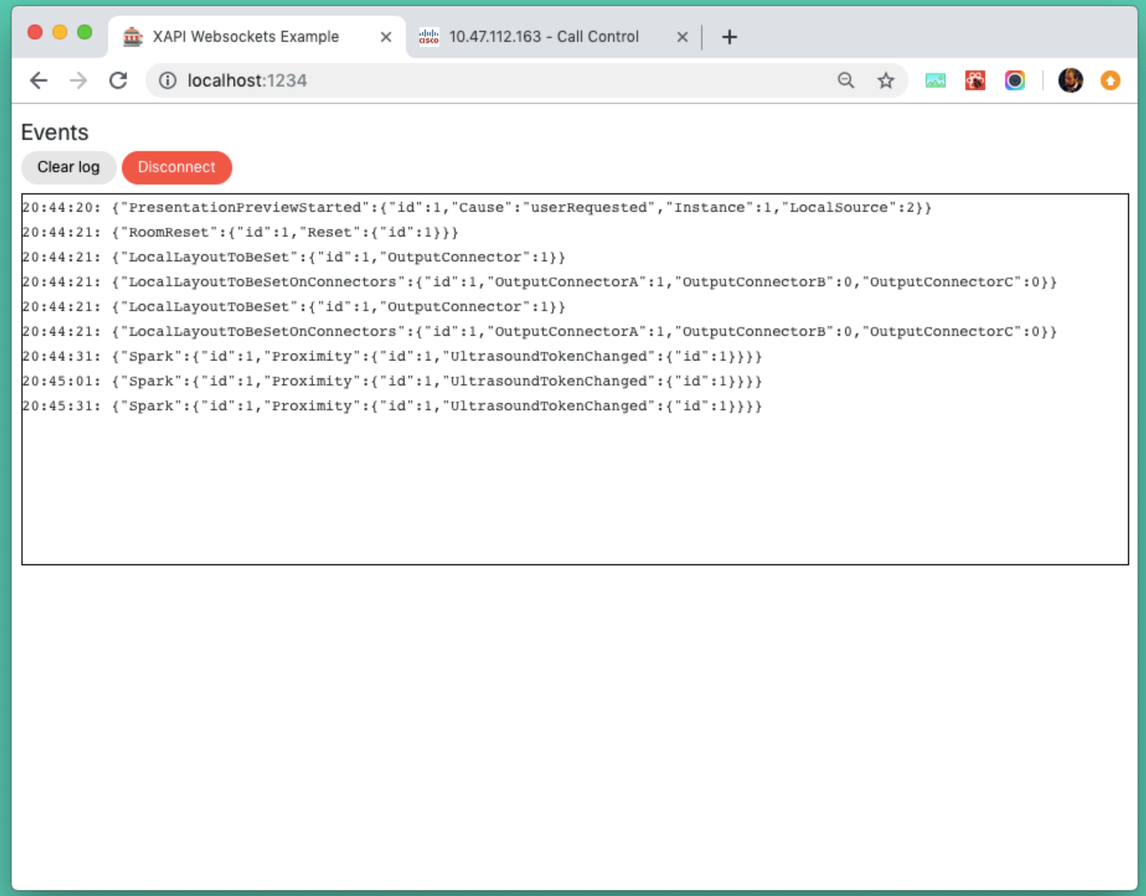


Code snippet

```
import * as jsxapi from 'jsxapi';

// … define hostname, username, password
jsxapi.connect(`wss://${hostname}`, { username, password });

xapi.config.get('SystemUnit Name’).then((hostname) => {
  console.log(‘Hostname: ‘, hostname);
}):
xapi.feedback.on('Event', (event) => {
 console.log(‘Event: ‘, event);
});
```

Try it

- 🚀 The web page connects to a video device and listens to all events
- 🚀 Uses React
- 🚀 Uses the Momentum UI kit
- 🚀 See **examples/websockets/README.md**



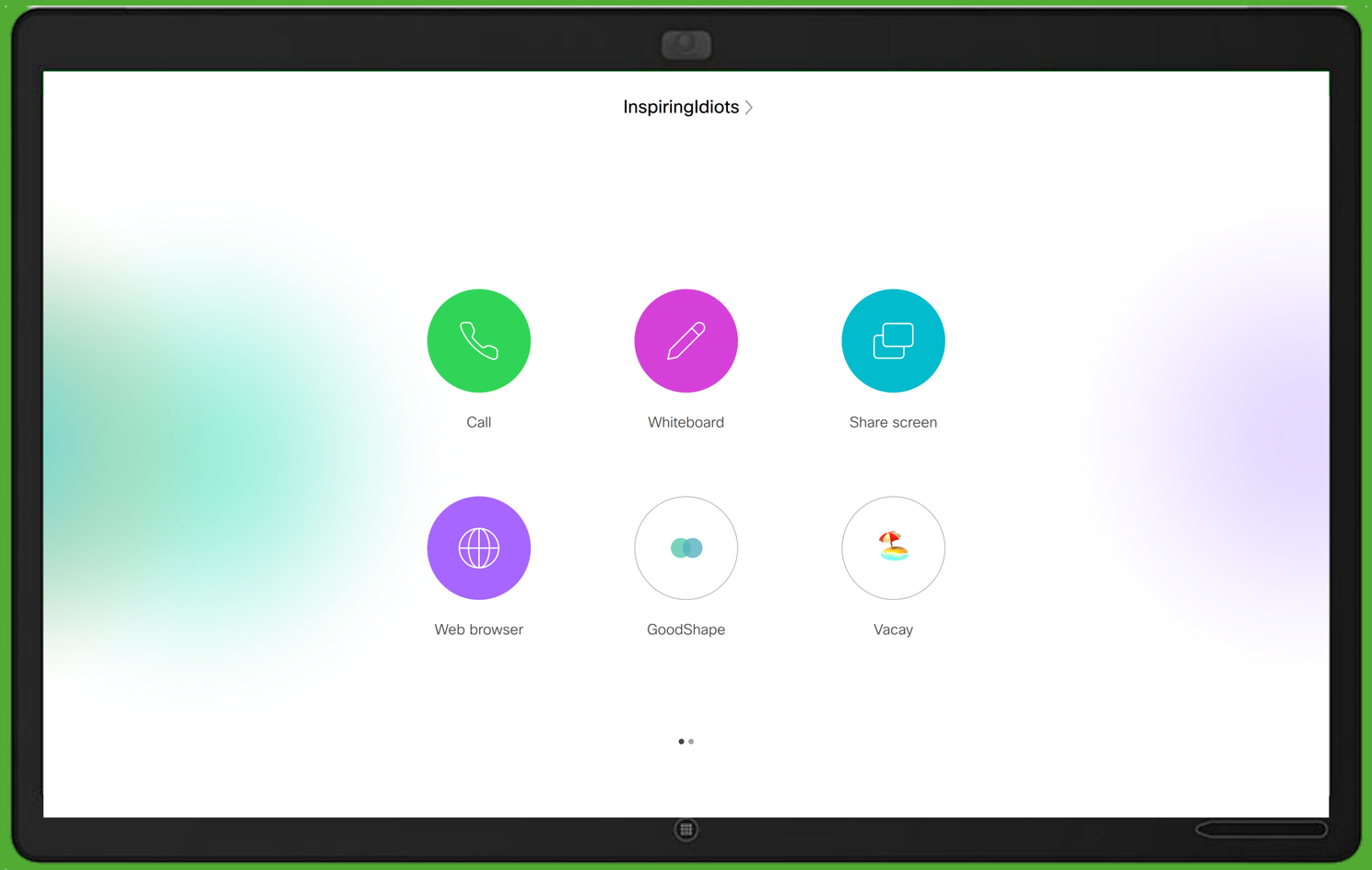
Web applications

Web apps are simply web URLs added to the home screen.

A Chromium browser provides JavaScript parsing, CSS, web sockets etc.

Web apps can be added to the home screen from the UI Extensions editor in the web interface on the video device.

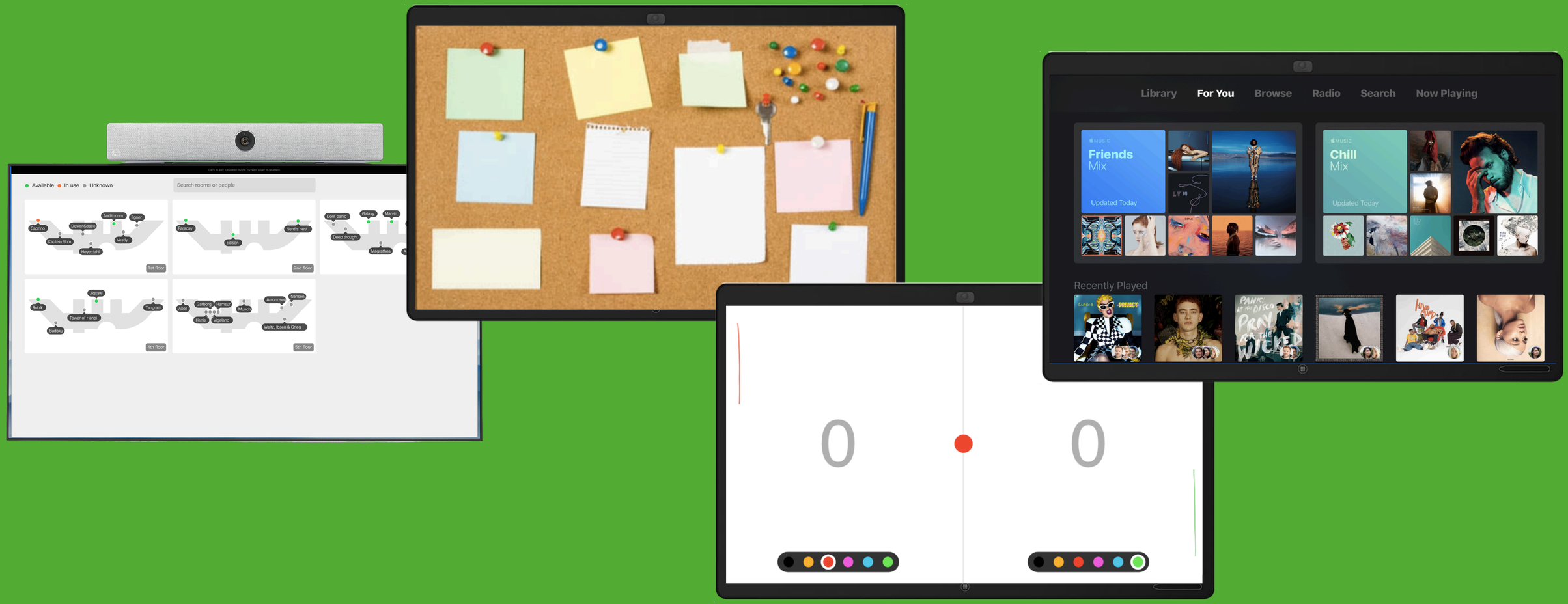
Web views can also be opened programatically with the xapi.



Example use cases

- 🚀 Office apps: Word, Excel, …
- 🚀 Collaboration tools: Trello, …
- 🚀 References: YouTube, Wikipedia, …
- 🚀 Bulletin boards, employee information, …
- 🚀 Games, ice breakers: Kahoot, QuickDraw, …
- 🚀 Help, instructions

Example use cases



Limitations

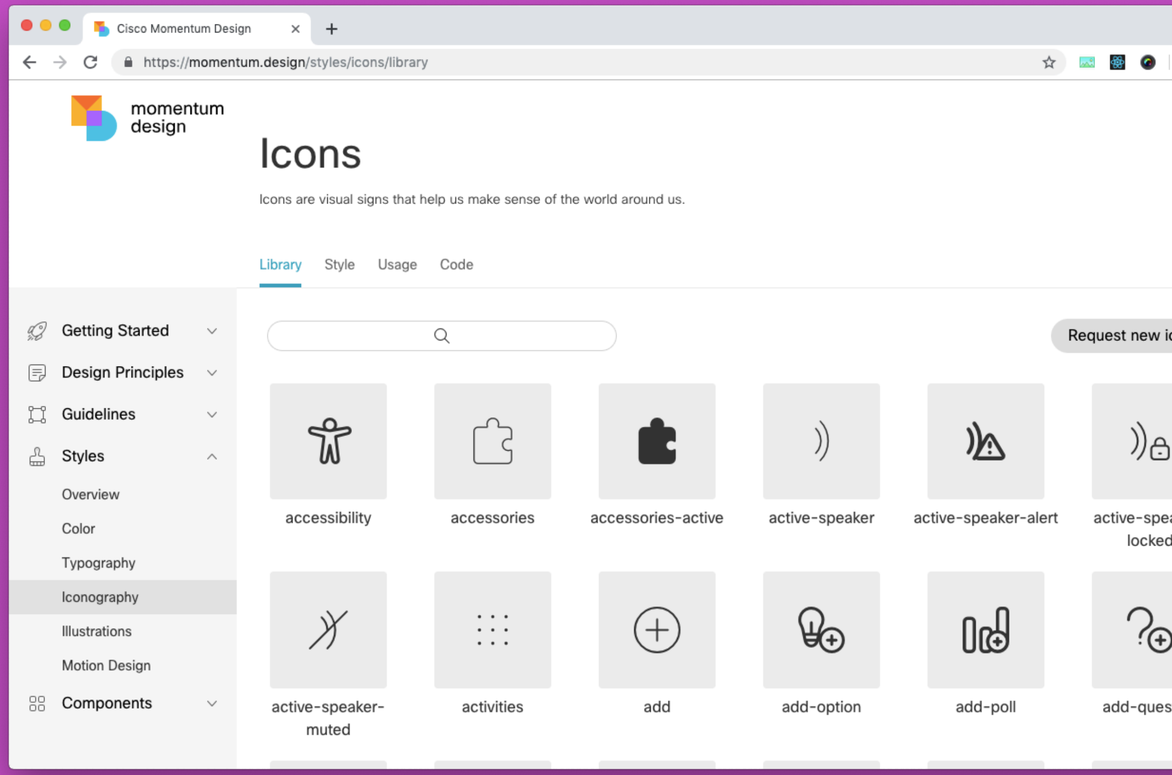
- 🚫 Only a single tab
- 🚫 “Incognito mode” - need to login again for every session
- 🚫 Web pages cannot currently access the xAPI directly (but you can use the web sockets)

Try it

- 🚀 Simple whiteboard web app
- 🚀 Uses paper.js lib
- 🚀 Straightens lines
- 🚀 See **examples/webapp/README.md**
- 🚀 Ref: **reference-docs/roomos-webengine-guide**

momentum.design

- 🚀 momentum.design
- 🚀 Cisco's open source design assets
- 🚀 Support for React and Angular
- 🚀 Use if you want Cisco look and feel

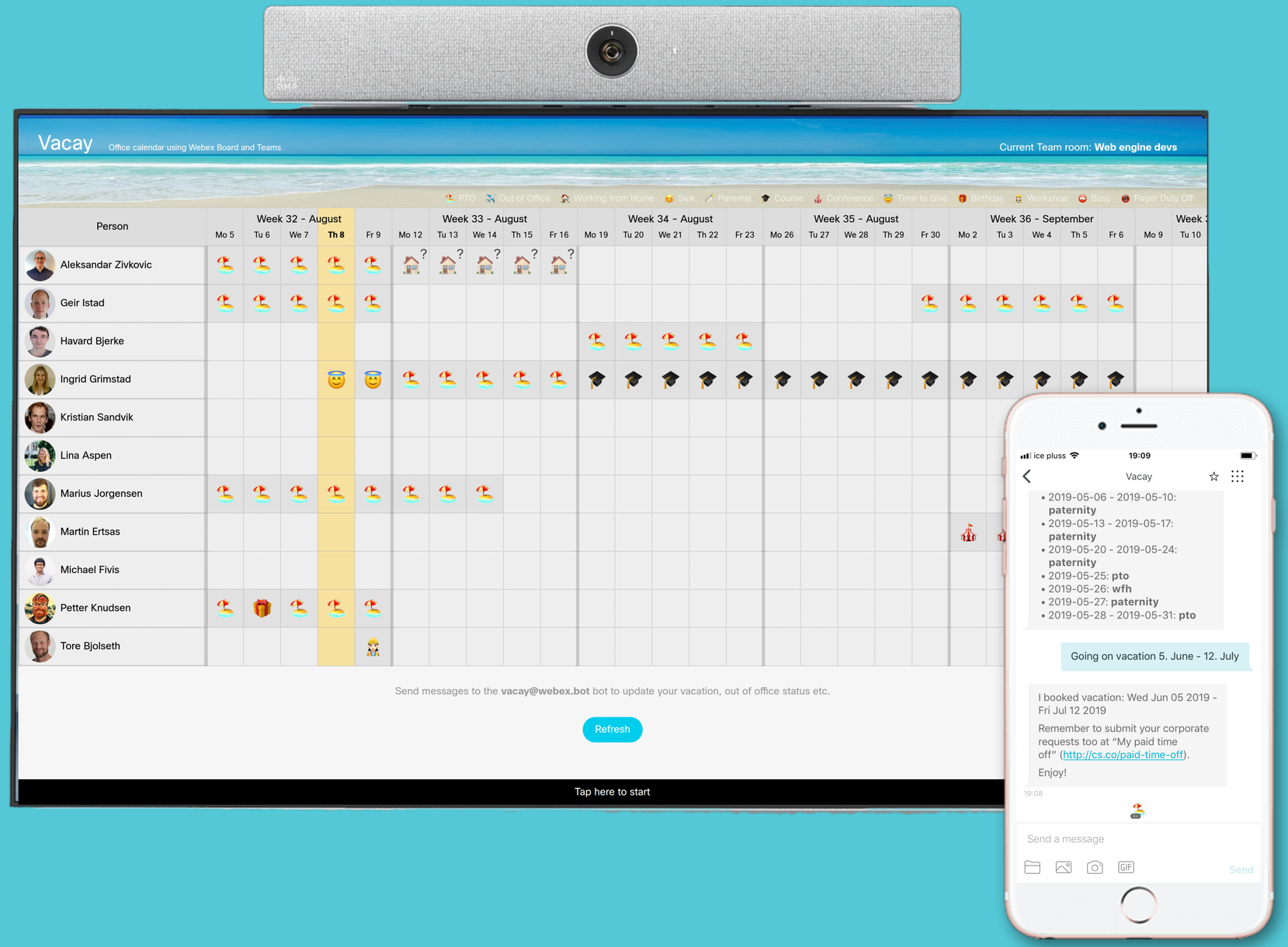


Bots

Bots are automated Webex users.

The bot can answer people on Webex Teams, or send messages automatically when something occurs.

A bot is simply a non-personal Webex Teams user. Create one at developer.webex.com

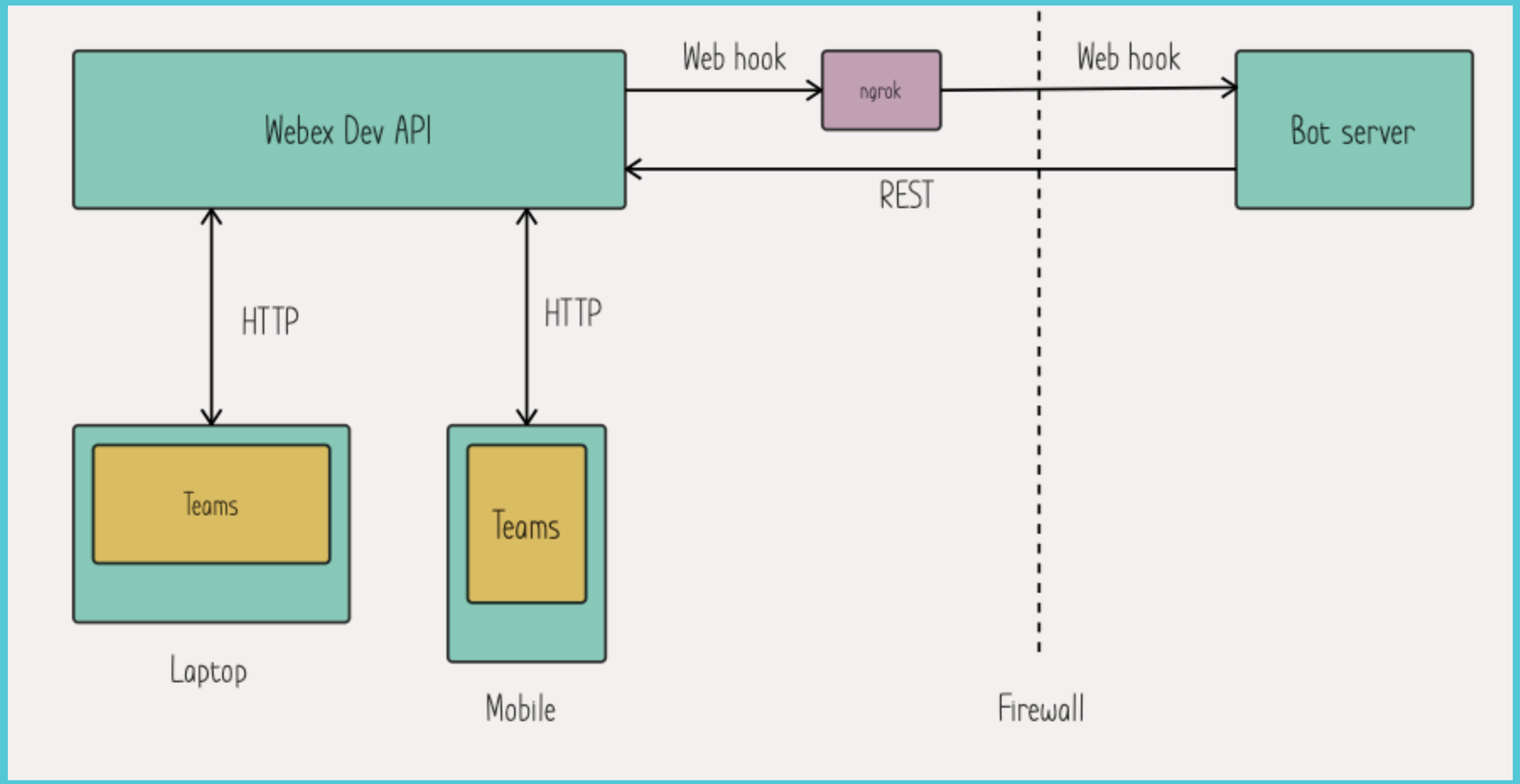


How bots work

A bot usually requires a server. To send a message, the bot just posts a REST message to the Webex cloud.

When someone messages the bot, the Webex cloud sends a HTTP request (“web hook”) to the bot web server.

APIs can easily be tested live at developer.webex.com

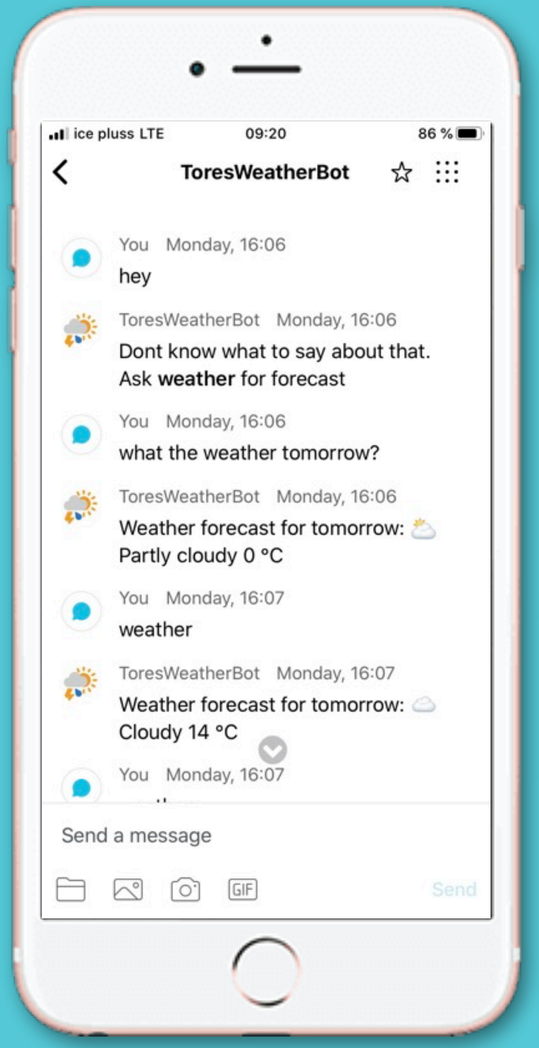


Try it

🚀 Uses existing bot: **ToresWeatherBot**

🚀 Say ‘weather’ to the bot for prediction

🚀 See **examples/bot/README.md**



JUST IN: This example now uses web sockets, so no need for ngrok


3rd party integrations
Philips Hue
Control light
Voice control
Sensor input
Amazon
Disruptive Technology