W3C THING API PROPOSAL

Louay Bassbouss | Fraunhofer FOKUS | louay.bassbouss@fokus.fraunhofer.de

Fraunhofer FOKUS

making cities smart

AGENDA

- Idea and Requirements
- W3C Presentation API
- Proposal for W3C Thing API
- Implementation as cordova plugin

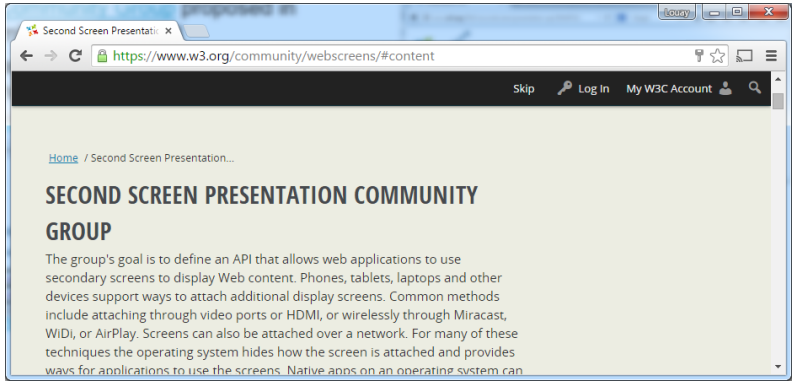
IDEA AND REQUIREMENTS

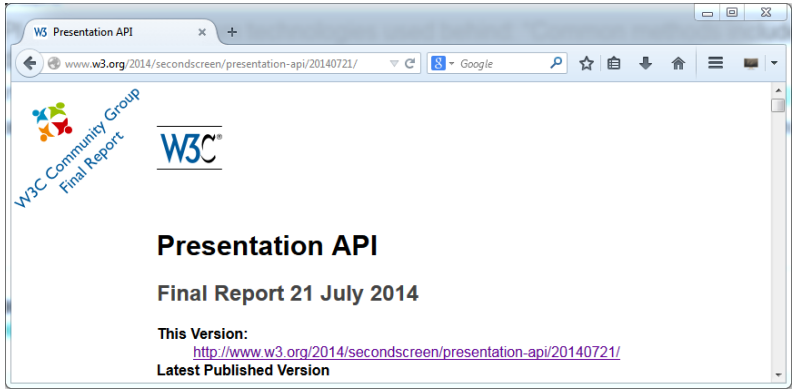
- W3C Thing API
- JavaScript API that allows Web pages to discover and interact with things
- The API should consider security and privacy by design
- The API should abstract from underlying protocols for discovery and communication
- The API should consider the concept of Thing Description
- Some concepts are taken from W3C Presentation API
  - Presentation API considers displays (or presentation devices like TVs, projectors, …)
  - Thing API considers Things as Tag
- The API could be implemented on top of existing Frameworks like:
  - Apple HomeKit
  - Google Brillo??

W3C Presentation API

W3C SECOND SCREEN PRESENTATION CG

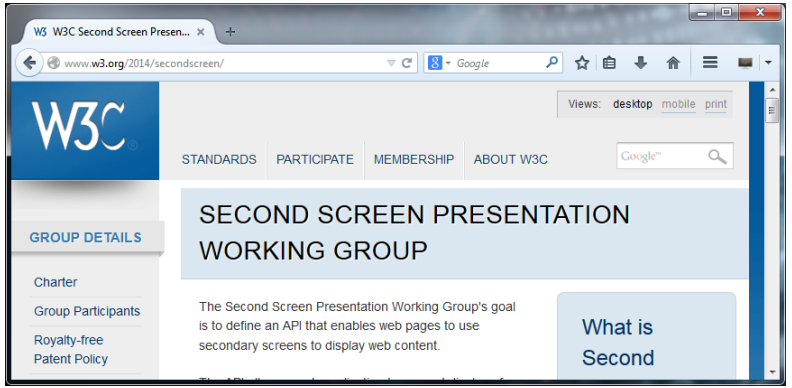
- W3C Community Group proposed in September 2013 by Intel
- Key partners: Intel, Google, Mozilla, Fraunhofer FOKUS, Netflix, LGE, etc.
- Goal: “Is to define an API that allows web applications to use secondary screens to display Web content”
- Final Report of the CG published in July 2014.

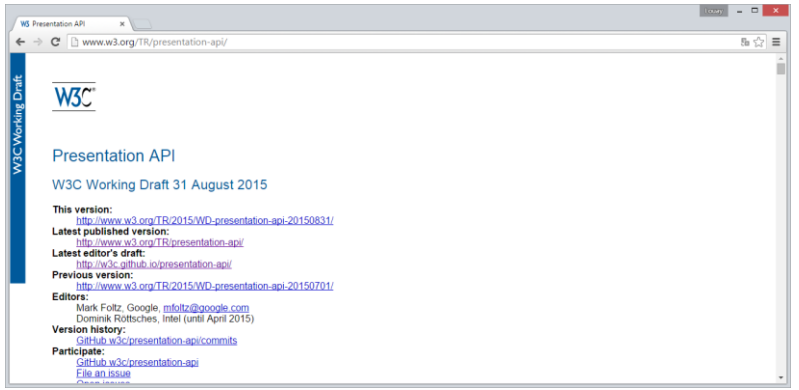




W3C SECOND SCREEN PRESENTATION WG

- The work of the Second Screen Presentation API is continued in a Working Group
- The Working Group was created in October 2014 → End date: 31 October 2016
- The WG took the final report of the CG as initial working draft for the Presentation API
- Working Draft 31 August 2015: http://www.w3.org/TR/presentation-api/

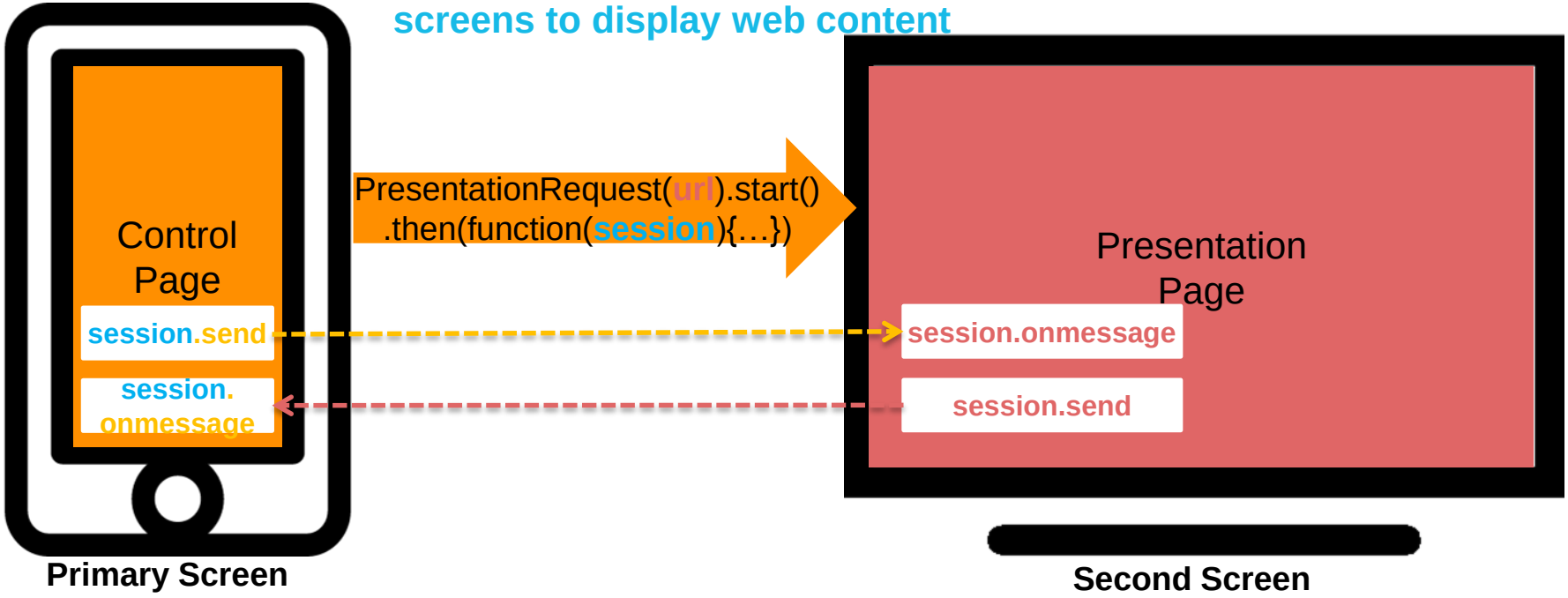




Fraunhofer FOKUS

W3C SECOND SCREEN PRESENTATION API

Goal is to define an API that enables web pages to use secondary screens to display web content

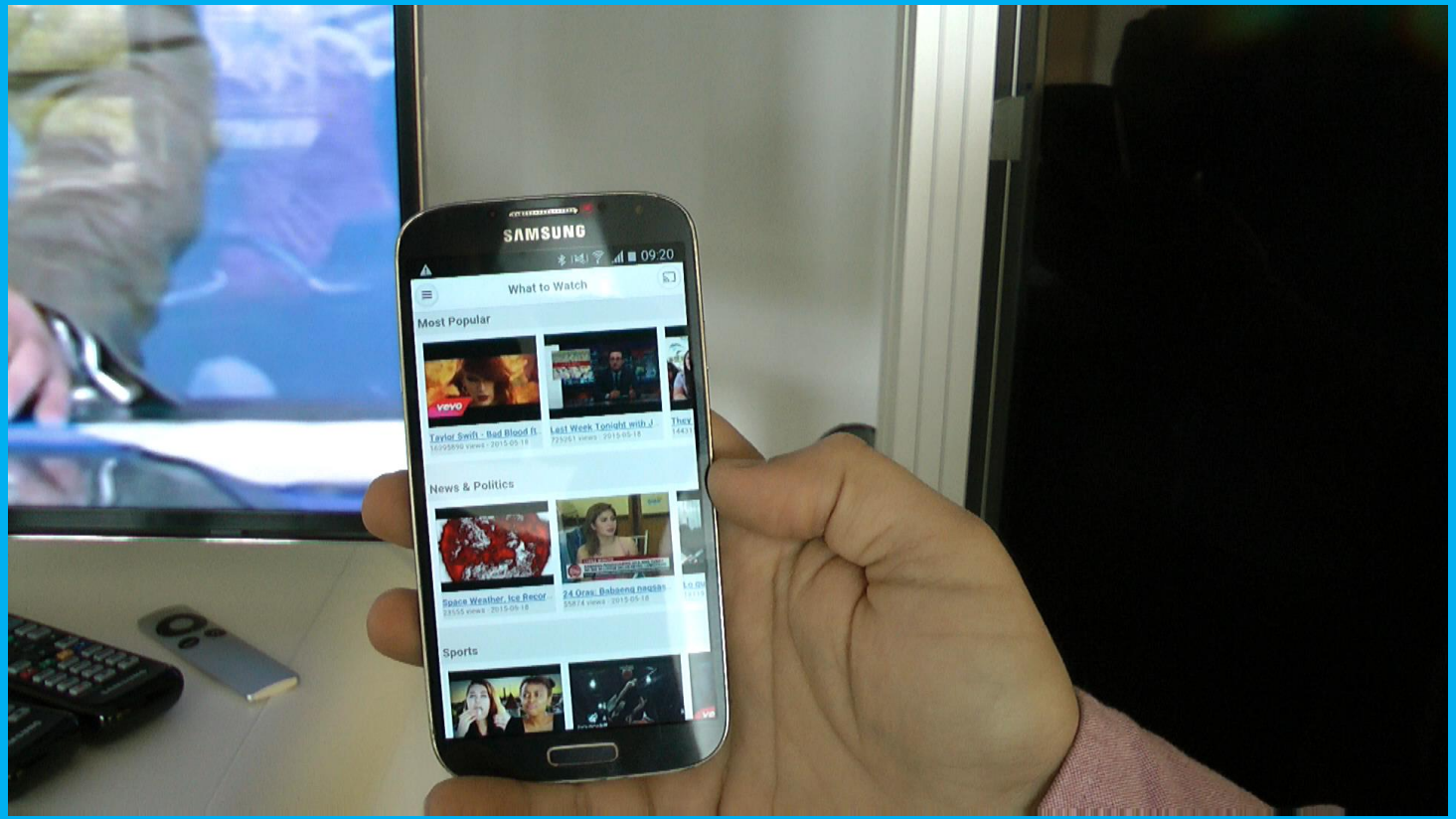


W3C SECOND SCREEN PRESENTATION API

Scope

- Define an API that allows a web application to:
  - ... request display of web content on a connected display
  - ... communicate with and control the web content
  - ... identify whether at least one secondary screen is available for display
- The web content may comprise HTML documents, web media types such as images, audio, video, or application-specific media
- The specification includes security and privacy considerations

Fraunhofer FOKUS implementation of Presentation API as Cordova Plugin



FIRST PRESENTATION API WG F2F MEETING IN BERLIN – MAY 2015

13:30 - 14:30

- **Presentation API – Intro and Recent Improvements**
  - François Daoust – Web and TV specialist at W3C
- **Presentation API in Chromium**
  - Mark Foltz – Google, Senior Staff Software Engineer
- **Presentation API / DIAL integration**
  - Mark Watson – Netflix, Director Streaming Standards
- **Companion Screens and HbbTV 2.0**
  - Matt Hammond – BBC, Senior R&D Engineer

- **14:30 - 15:00**
  - Coffee Break, Demos & Exhibition

15:00 - 16:00

- **Presentation API on Firefox OS**
  - Shih-Chiang Chien – Mozilla Foundation, Senior Software Engineer
- **Presentation API on Smart Watches**
  - Soonbo Han – LG Electronics, Senior Research Engineer
- **Multiscreen on Cloud Browsers**
  - Oliver Friedrich – Deutsche Telekom, Senior Expert New Media
- **Digital Signage Provides Information of Games and Disasters**
  - Masayuki Ihara – NTT Japan, Senior Research Engineer
- **Extending Video for Multiscreen**
  - Jean-Claude Dufourd – Télécom ParisTech, Research Director

Thing API proposal

THING API PROPOSAL

- W3C Thing API
  - Potential namespace: *navigator.thing* or *navigator.things*
- JavaScript API that allows Web pages to discover and interact with things
  - ThingRequest(filter).start().then(function(thing){…}).catch(function(err){…});
  - filter is a JSON that contains filter properties like type, proximity, etc.
- The API should consider security and privacy by design
  - In order to obtain access to a Thing, the browser may show (after ThingRequest.start() is called) a dialog (like <input type="file"> dialog) that displays a list of available Things. Once the user selects a thing, then it will be available for the Web page after the promise is resolved. Otherwise the promise will be rejected. This step may be not needed for non-browser JavaScript environments like Node.js
  - After the user approved access to a thing, the web page can access it (e.g. when the page is reloaded or opened again at a later time) by using:
    - navigator.things.getById(thingId).then(function(thing){…}).catch(function(err){…});

THING API PROPOSAL

- The API should abstract from underlying protocols for discovery and communication
  - Once the web page get access to a thing, the following API can be used to ready/write properties, call actions or subscribe to events by using the information (name of properties, actions, events, etc.) form the corresponding Thing Description:
  - thing.property.set("colorTemperature", 123456).then(success).catch(error);
  - thing.property.get("colorTemperature").then(success).catch(error);
  - thing.action.call("ledOnOff", true).then(success).catch(error);
  - thing.event.on("colorTemperatureChanged", callback).then(success).catch(error);
- Check and Watch reachability of a thing:
  - thing.getReachability().then(function(reachability) {
    handleReachabilityChange(reachability.value);
    reachability.onchange = function() { handleReachabilityChange(this.value);}
    });

EXAMPLE (1/2)

```
// filter of things to discover. Additional parameters can be added.
// in this example we want to discover LEDs nearby (Discovery and Communication technologies is not important)
// The value of the type element is just an example here for LEDs. Ontology for Thing types needs to be defined (or reused from somewhere else).
var filter = {
        type: "http://www.w3c.org/wot/thing/led"
        proximity: "nearby"
};
var req = new ThingRequest(filter);
// onSuccess will be called only when the user selects a Thing from the Thing Selection Dialog.
// The Thing Selection Dialog is a native UI provided by the User Agent and not accessible to the Web App.
// The Thing Selection Dialog will be displayed after the Web App calls "req.start()". The user may select
// a Thing from the Dialog or may cancel the Dialog.
var onSuccess = function(thing){
        // onSetPropertySuccess is called when the property is set successfully.
        // onSetPropertyError is called for example when a thing is not reachable, the property is not writable or when the property doesn't exist.
        thing.property.set("colorTemperature", 123456).then(onSetPropertySuccess).catch(onSetPropertyError);
        // onGetPropertySuccess is called when the property is retrived successfully.
        // onGetPropertyError is called for example when a thing is not reachable or when the property doesn't exist.
        thing.property.get("colorTemperature").then(onGetPropertySuccess).catch(onGetPropertyError);
        // onActionCallSuccess is called when the action is successfully executed. Results are passed as input.
        // onActionCallError is called for example when a thing is not reachable, when the action doesn't exist or when an error is raised during execution
        thing.action.call("ledOnOff", true).then(onActionCallSuccess).catch(onActionCallError);
        // colorTemperatureChangedCallback is executed each time the LED reports a new value.
        // onSubscribeSuccess is called when subscribtion was successfull.
        // onSubscribeError is called when the thing is not reachable or when the event doesn't exist
        thing.event.on("colorTemperatureChanged", colorTemperatureChangedCallback).then(onSubscribeSuccess).catch(onSubscribeError);
        // Get reachability of the Thing. Reachability may change during runtime.
        thing.getReachability().then(function(reachability) {
                // reachability.value may be kept up-to-date by the UA as long as the reachability
                // object is alive. It is advised for the web developers to discard the object
                // as soon as it's not needed.
                handleReachabilityChange(reachability.value);
                // reachability.onchange is executed each time the reachability is changed.
                // For example when the device in not the range of the LED or the LED is not available anymore.
                reachability.onchange = function() { handleReachabilityChange(this.value);}
        });
```

EXAMPLE (2/2)

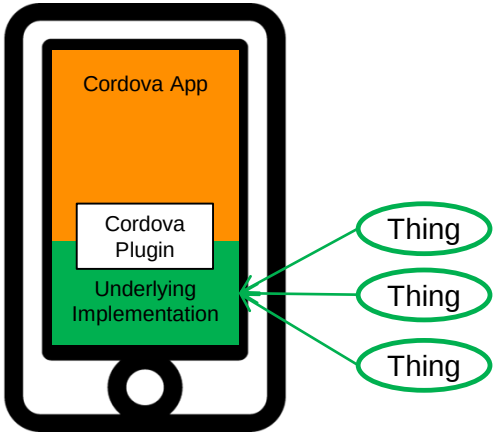
```
        // the Web App may store the thing.id in localStorage or somewhere else
        // and requests access to the Thing after reload the Web App using
        var thingId = thing.id;
        localStorage.setItem("thingId", thingId);
        var thingType = thing.type;
        var thingName = thing.name;
};
// onError will be called when the user cancels the selection dialog.
var onError = function(err){
        console.error("Unexpected Error", err);
};
// start the request will display the Thing Selection Dialog.
req.start().then(onSuccess).catch(onError);

// This call is relevant when the page is reloaded, but the app already accessed the thing before and stored its Id in the Storage.
var thingId = localStorage.getItem("thingId");
thingId && navigator.things.getById(thingId).then(function(thing){
        thing.getReachability().then(function(reachability) {
                if(reachability.value){
                        // access thing in the same way as described above
                }
        });
}).catch(function(err){
        console.error("Error on get thing by Id", err);
});
```

Implementation as cordova plugin

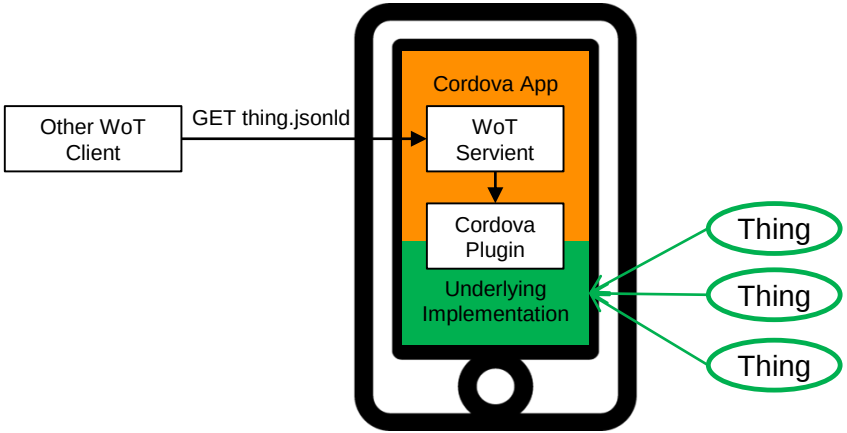
IMPLEMENTATION AS CORDOVA PLUGIN

- The Implementation is work in progress
- It is a cordova plugin for android and iOS
- The iOS implementation supports HomeKit
  - HomeKit Accessory ←→ W3C Thing
- Implementation will be showcased during TPAC



PLUGFEST IMPLEMENTATION: CORDOVA APP AS WOT SERVIENT

- Cordova App implements a WoT servient over WebSockets and HTTP
- A TD is available for each available Thing



FEEDBACK

- Feedback on the API proposal is welcome
- Feedback from Francois:
  - *“Would supporting the ability to select more than one Thing at a time be useful?”*
  - *“There may be more Things to choose from, which might mean that the list could grow out of control”*
  - *“Requiring the user to select a light in a list just to be able to switch it on or off may not lead to the best user experience”*

CONTACT

Louay Bassbouss
Senior Project Manager R&D
Future Applications and Media
Tel. +49 (30) 34 63 – 7275
louay.bassbouss@fokus.fraunhofer.de

Fraunhofer Institute for Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

Tel: +49 (30) 34 63 – 7000
Fax: +49 (30) 34 63 – 8000
www.fokus.fraunhofer.de

Dr. Stephan Steglich
Director of Competence Center
Future Applications and Media
Tel. +49 (30) 34 63 – 7373
stephan.steglich@fokus.fraunhofer.de

Fraunhofer Institute for Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

Tel: +49 (30) 34 63 – 7000
Fax: +49 (30) 34 63 – 8000
www.fokus.fraunhofer.de