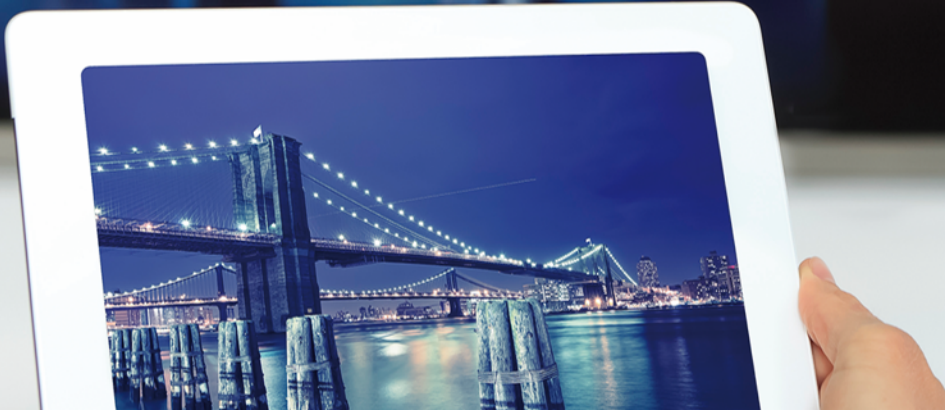


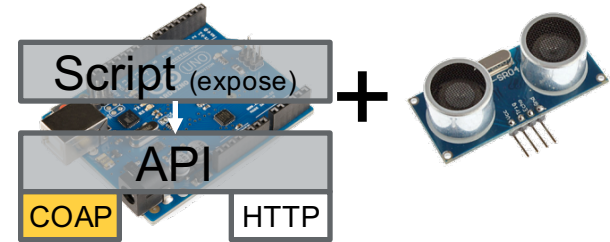
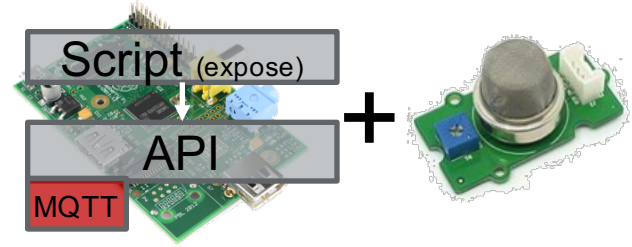
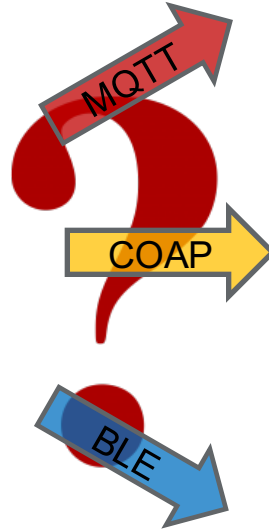
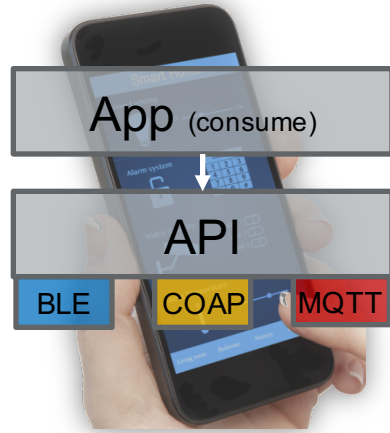
# SCRIPTING APIS FOR THE WEB OF THINGS



Louay Bassbouss | Fraunhofer FOKUS | [louay.bassbouss@fokus.fraunhofer.de](mailto:louay.bassbouss@fokus.fraunhofer.de)

WWW2016 - W3C Track: Building and Designing the Web of Things, Montréal, Canada

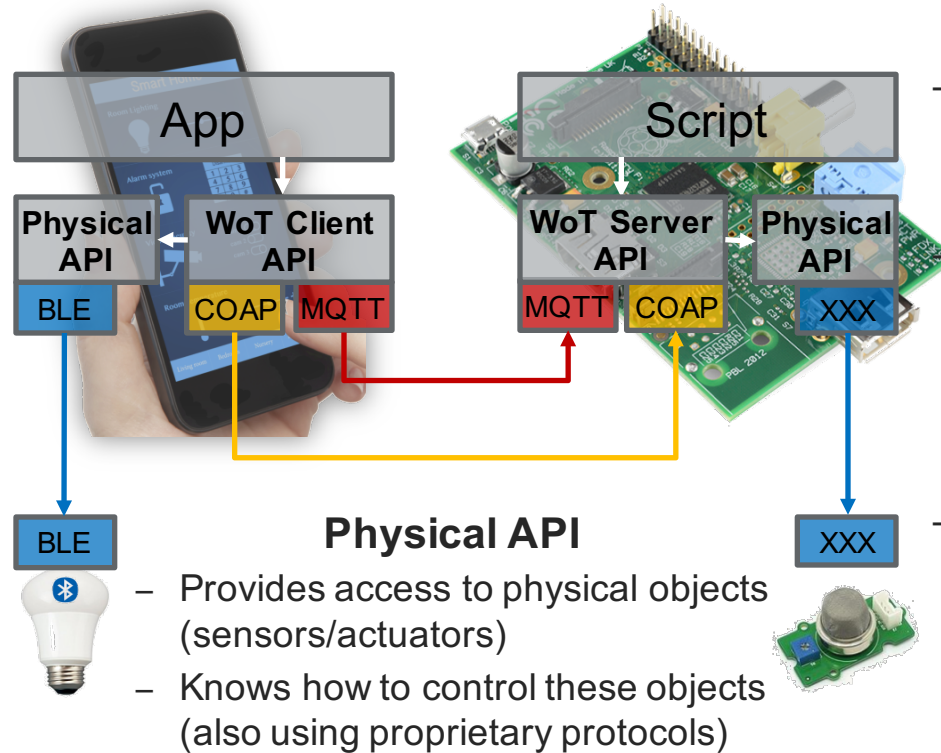
# MOTIVATION




## FEATURES

# WoT Client API

- Discover Things
  - Local
  - Remote
- Access Things
  - Read/Write properties
  - Call actions
  - Observe events
- Consider Thing Descriptions

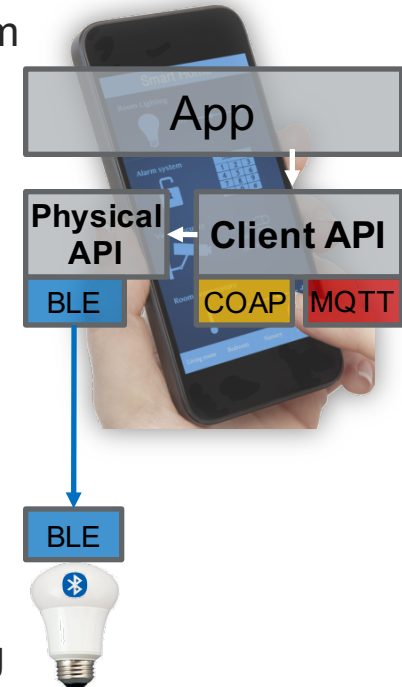


## WoT Server API

- Advertise Things
  - Local
  - Remote
-  Serve Clients
  - Handle read/ write requests
  - Handle action calls
  - Trigger events
- Consider Thing Descriptions

# WOT CLIENT API: REQUIREMENTS

- SHOULD expose interfaces to client applications that abstract from the underlying protocols (BLE, COAP, MQTT, ...)
  - SHOULD offer control interfaces that consider the semantic of Thing Descriptions (name of properties, actions, events, data types, ...)
  - SHOULD offer discovery interfaces with capability of filtering Things based on parameters like thing types, capabilities, ...
  - SHOULD consider Security and Privacy by design
- 
- Providers of the WoT Client API implement bindings to one or more protocols.
  - The API could directly talk to physical objects using corresponding protocols (e.g. BLE)

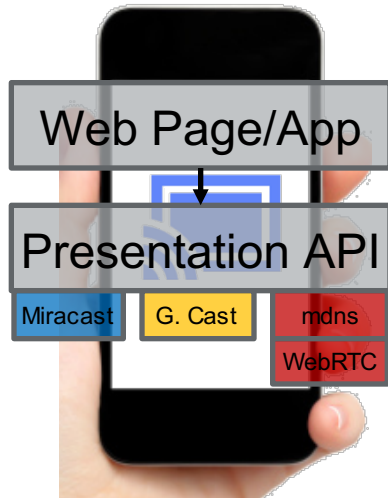


## SIMILAR W3C APIS

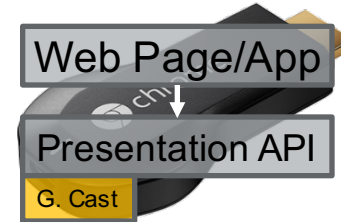
- W3C already offers an API that follow the same principles → **W3C Presentation API**
- The **Presentation API** allows web applications to:
  - ... request display of web content on secondary screens (e.g. TVs, )
  - ... communicate with and control the web content
  - ... identify whether at least one secondary screen is available or not
- The web content may comprise HTML documents, web media types such as images, audio, video, or application-specific media
- Presentation API abstracts from underlying protocols for discovery and communication:
  - Chrome implementation uses Google Cast ()
  - Firefox implementation uses mDNS + WebRTC
- The specification includes security and privacy considerations

# W3C PRESENTATION API

Control Device

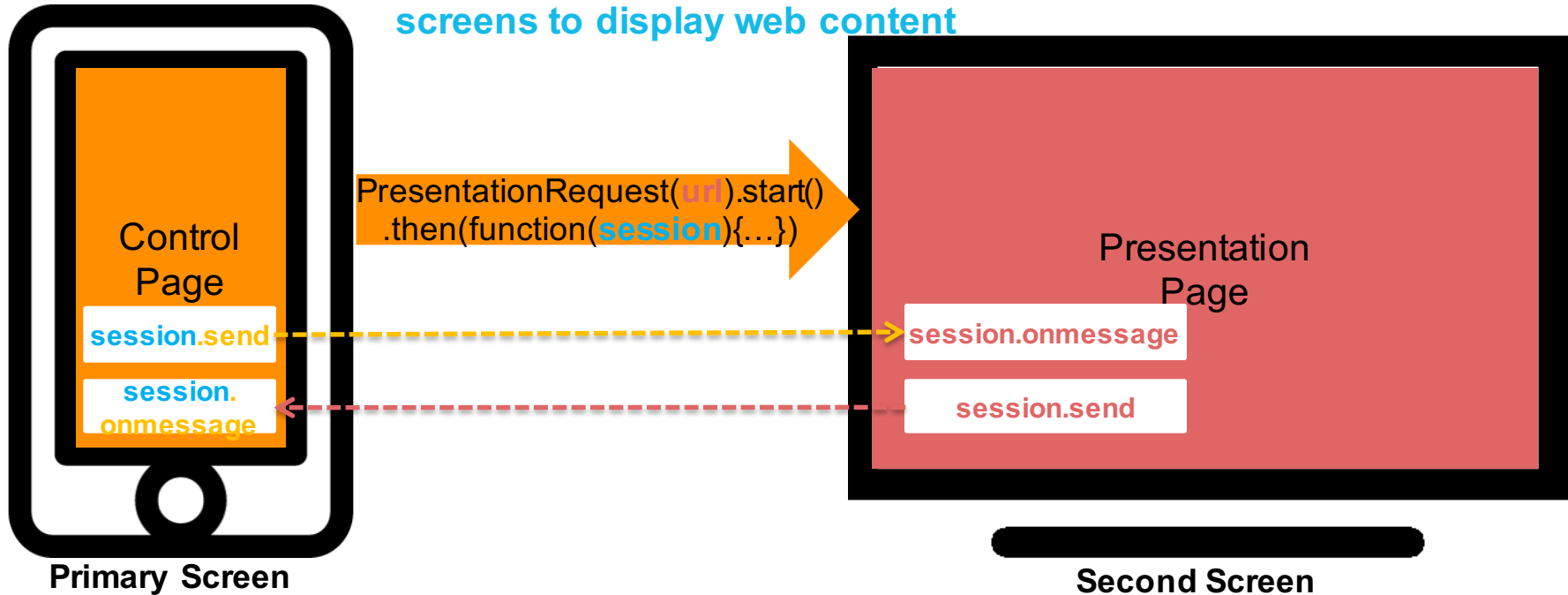


Presentation Devices



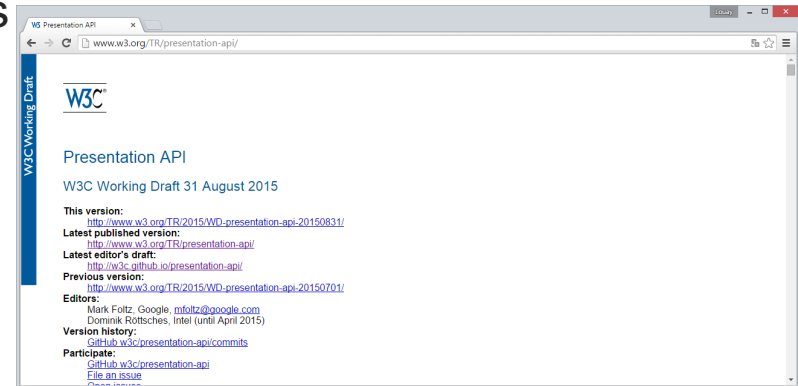
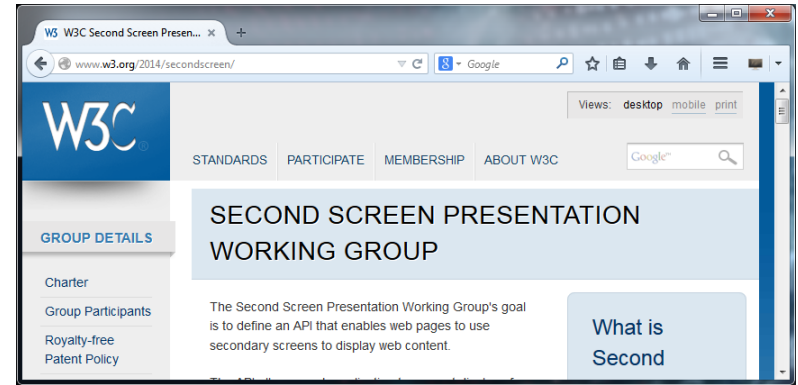
# W3C PRESENTATION API

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



# 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:  
<http://www.w3.org/TR/presentation-api/>





# WOT CLIENT API PROPOSAL

<http://w3c.github.io/wot/current-practices/wot-practices.html#sec-scripting-api>

## WebIDL

```
[Constructor(ThingFilter filter)]  
interface ThingRequest {  
    Promise<sequence<Thing>> start();  
};
```

## WebIDL

```
dictionary ThingFilter {  
    DOMString? type;  
    ThingProximity? proximity;  
    DOMString? id;  
    DOMString? server;  
};
```

# WOT CLIENT API PROPOSAL

## WebIDL

```
[Constructor(ThingDescription td)]
interface ConsumedThing {
    readonly attribute DOMString    id;
    readonly attribute DOMString    type;
    readonly attribute DOMString    name;
    readonly attribute boolean      reachable;
    attribute EventHandler onreachabilitychange;
    Promise<any> callAction(DOMString actionName, any parameter);
    Promise<any> setProperty(DOMString propertyName, any newValue);
    Promise<any> getProperty(DOMString propertyName);
    void addListener(DOMString eventName, ThingEventListener listener);
    void removeListener(DOMString eventName,
                        ThingEventListener listener);
    void removeAllListeners(DOMString eventName);
};

callback ThingEventListener = void (ThingEvent event);
```

# WOT CLIENT API EXAMPLE

<http://w3c.github.io/wot/current-practices/wot-practices.html#scripting-api-examples>

```
var request = new ThingRequest(filter);
request.start().then(function(things){
  var thing = things[0];
  if(thing){
    // get thing basic information
    console.log("id: ", thing.id);
    console.log("name: ", thing.name);
    console.log("type: ", thing.type);
    console.log("manufacturer: ", thing.manufacturer);
    console.log("reachable: ", thing.reachable);
    // store thing id locally e.g. in localStorage
    localStorage && localStorage.setItem("thing.id", thing.id);
    // monitor reachability of the thing
    thing.onreachabilitychange = function(){
      console.log("reachability changed to ", this.reachable);
      // If the thing is not reachable, then the operations callAction
    };
    // Call an action
    var input = ...;
    thing.callAction("myAction", input).then(function(output){
      console.log("Result of myAction()", output);
    }).catch(function(err){
      console.error("Error on call action", err);
    });
  }
});
```

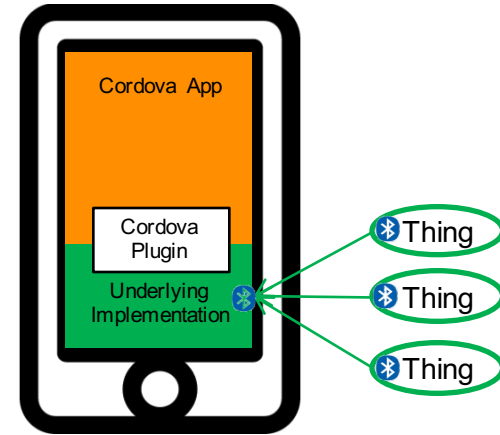
```
var filter = {
  type: "http://example.org#foo",
  proximity: "nearby"
};
```

```
// get and set property
thing.getProperty("myProp").then(function(value){
  console.log("Value of myProp ", value);
  var newValue = ...;
  return thing.setProperty("myProp", newValue);
}).then(function(newValue){
  console.log("Value of myProp is now", newValue);
}).catch(function(err){
  console.error("Error on get or set property myProp", err);
});
// add and remove thing event listener
var myListener;
thing.addListener("myEvent", myListener=function(evt){
  console.log("receive event ", name, "from thing", evt.source.name);
});
thing.removeListener("myEvent", myListener);
thing.removeAllListeners("myEvent");

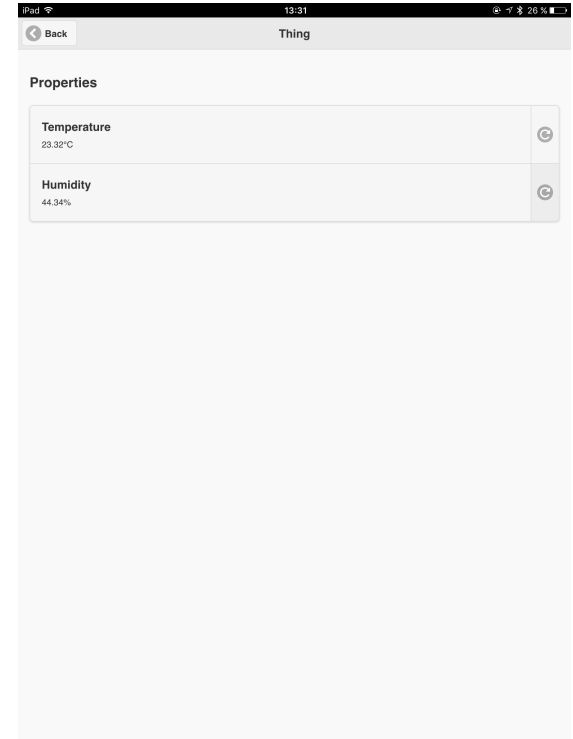
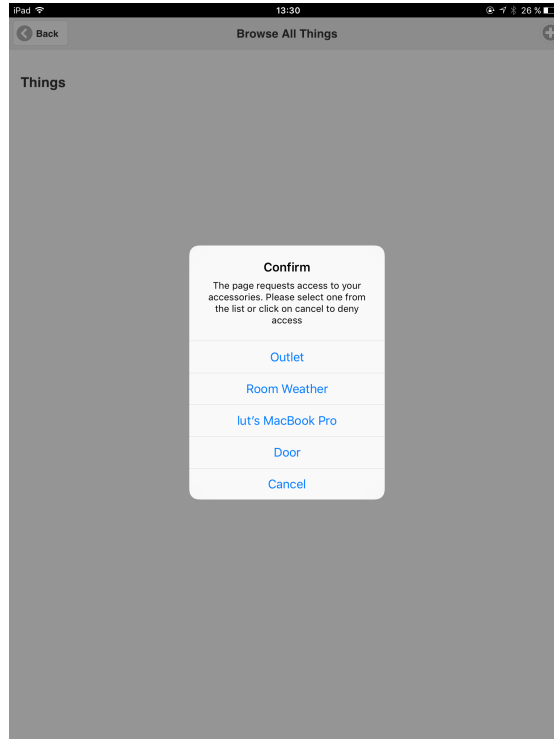
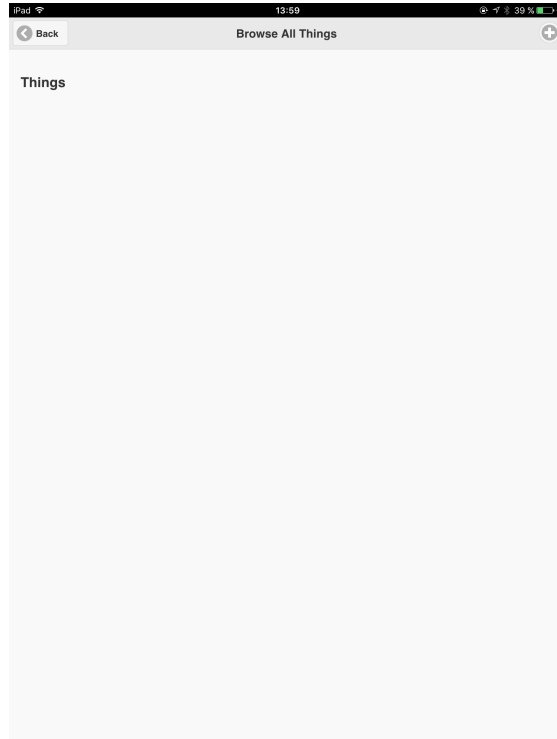
});
}).catch(function(err){
  //TODO: handle error
});
```

# IMPLEMENTATION AS CORDOVA PLUGIN

- The Cordova plugin provides the WoT Client API
- The Cordova plugin implements BLE protocol binding
- Implemented for iOS (Android work in progress)
- Implementation as Node.js module is planned



# DEMO



# THANK YOU



# Contact

## Louay Bassbouss

Senior Project Manager R&D

Future Applications and Media

Tel. +49 (30) 34 63 – 7275

[louay.bassbouss@fokus.fraunhofer.de](mailto: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](http://www.fokus.fraunhofer.de)

