High Level Interoperability Testing

For the Web of Things

Dave Raggett <dsr@w3.org>

F-Interop is an international project supported by the European Union's Horizon 2020 programme under Grant Agreement number: 687884

F-Interop work on the Web of Things

- To create a test framework for high level interoperability
  - Above the level of protocols
  - Complementing F-Interop work on protocol level interoperability
- To make this framework available as an open source project
- In addition, an open source WoT Web Hub for the NodeJS community
- Presentation at W3C's all group meetings TPAC 2018 in Oct 2018
- Agreement on use as part of W3C WoT WG test plan for their RECs

Testing and the Web of Things

Two talks and demos at TPAC 2018:

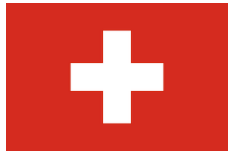
- **Dave Raggett** on high level interoperability testing
  - W3C staff activity lead for Data and a champion for Web of Things
  - Long history of work on web standards (HTML, HTTP, …)
- **Ege Korkan** on testing things exposed by particular devices

This work has been supported by F-Interop

F-Interop, see www.f-interop.eu

- European project supported by the European Commission and the Swiss Government with the following partners
  - Sorbonne Université, Mandat International, ETSI, imec, EANTC, Digital Catapult, Université du Luxembourg, INRIA, UDG and W3C/ERCIM
- Interoperability testing at the protocol level, e.g. HTTP, CoAP
  - Using AMQP for test control messaging
  - And a VPN for intercepting packets
  - Available as Docker images for easy installation
- W3C/ERCIM has added support for high level testing for the Web of Things
  - Moving beyond protocol level testing to ensure application interoperability

Horizon 2020
European Union
Funding for Research
and Innovation



Swiss State
Secretariat for
Education, Research
and Innovation

The Web of Things

The Web of Things

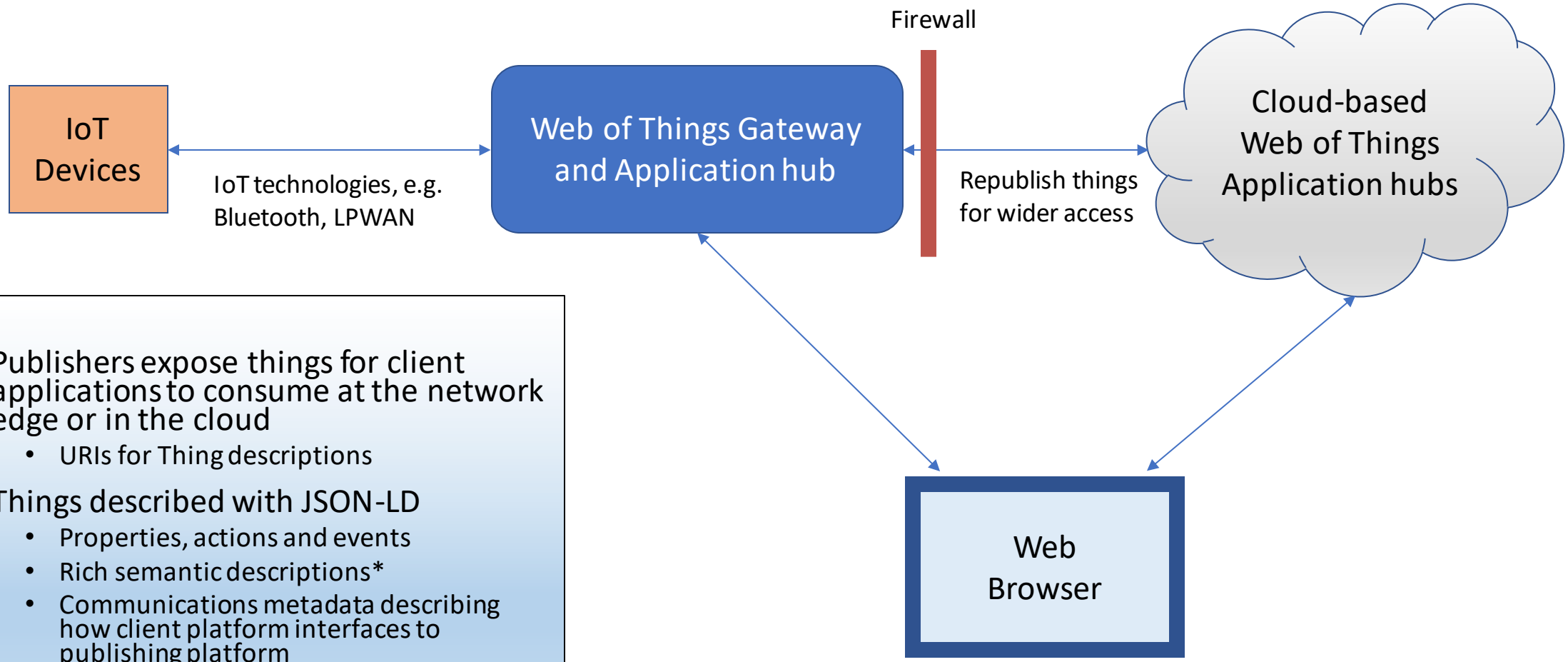
- **The IoT is fragmented**
  - Lots of incompatible technologies, protocols and standards
  - This is holding back the potential by increasing costs and risks
- **Web of Things as a simplifying abstraction layer**
  - Things as software objects with properties, actions and events for sensors, actuators and related information services
  - Descriptions of the kinds of things, their capabilities and the context in which they reside
- **Web of Things for open standards based application platforms that insulate developers from the myriad IoT technologies and standards**
  - Loosely analogous to Web browsers and how HTML & HTTP revolutionised Internet services for consumers
  - This will stimulate rapid growth in services as happened with the Web

Web Hubs as open Web platforms

- Each thing has a URI
  - Dereference to obtain JSON-LD describing its properties, actions and events
    - With datatypes expressed using JSON schema and physical units such as “celsius”
    - Metadata defining how client libraries remotely access the thing
    - Or request HTML description via HTTPS GET with Accept: text/html
  - This URI allows things to be described with Linked Data, including the kinds of things, their capabilities, interrelationships and the context in which they reside
    - e.g. a smart light with controllable hue and brightness in a room in your home
- Open source implementations
  - Eclipse *ThingWeb*, Mozilla *Things Gateway*, and my *Arena Web Hub**

** Named after my web browser which was the first to support tables and CSS*

Web Hubs host web apps that expose and consume things



- Publishers expose things for client applications to consume at the network edge or in the cloud
  - URIs for Thing descriptions
- Things described with JSON-LD
  - Properties, actions and events
  - Rich semantic descriptions*
  - Communications metadata describing how client platform interfaces to publishing platform

* e.g. as defined by iot.schema.org

Arena Web Hub

- Node module installable with npm
  - *npm install arena-webhub*
  - Installable on Raspberry Pi etc. for home gateway
  - Github project at https://github.com/draggett/arena-webhub (release: November 2018)
- Used by server-side application to expose things for use by clients
  - Server-side application hides IoT protocols, e.g. Bluetooth, ZigBee, …
- Security based on use of TLS and JWT
- Supports HTTPS, Server-Sent Events and WebSockets*
  - Applications responsible for generating and validating JWT bearer tokens
    - As well as management of user accounts and notifications over SMTP and SMS
    - Examples for how to do this securely are provided in the GitHub project
  - JavaScript client library for use in web pages
    - With examples for some virtual things
- Looking for help with providing examples for Bluetooth, ZigBee etc.

* Protocols you can use from web page scripts using the fetch, EventSource and WebSocket APIs

Simple server app example

```
let webhub = require('arena-webhub').({
        port: 8888,
        validateJWT: (token, url) => { // some code to validate the JWT token }
});

let thing = webhub.produce({
        name: "test",
        properties: {
                power: {
                        type: "boolean",
                        value: false
                }
        }
});

let state = thing.properties.on.value;
setInterval( function() { thing.properties.on.write(state); state = !state; }, 5000 );

thing.expose();  // publish thing to clients
```

Simple client app example

```
<!DOCTYPE html>
<html lang="en">
<head>
        <title>Web Hub demo</title>
        <meta charset="utf-8">
        <script type="text/javascript" src="wot.js"></script>
        <script>
                let thingURI = "https://localhost:8888/things/test";

                wot.consume(thingURI).then (thing => {
                        thing.properties.power.subscribe(state => {
                                console.log("power set to " + state)
                        }
                });
        </script>
…
</html>
```


Note that Arena Web Hub integrates a Web server for Web pages

Web of Things & Standardisation

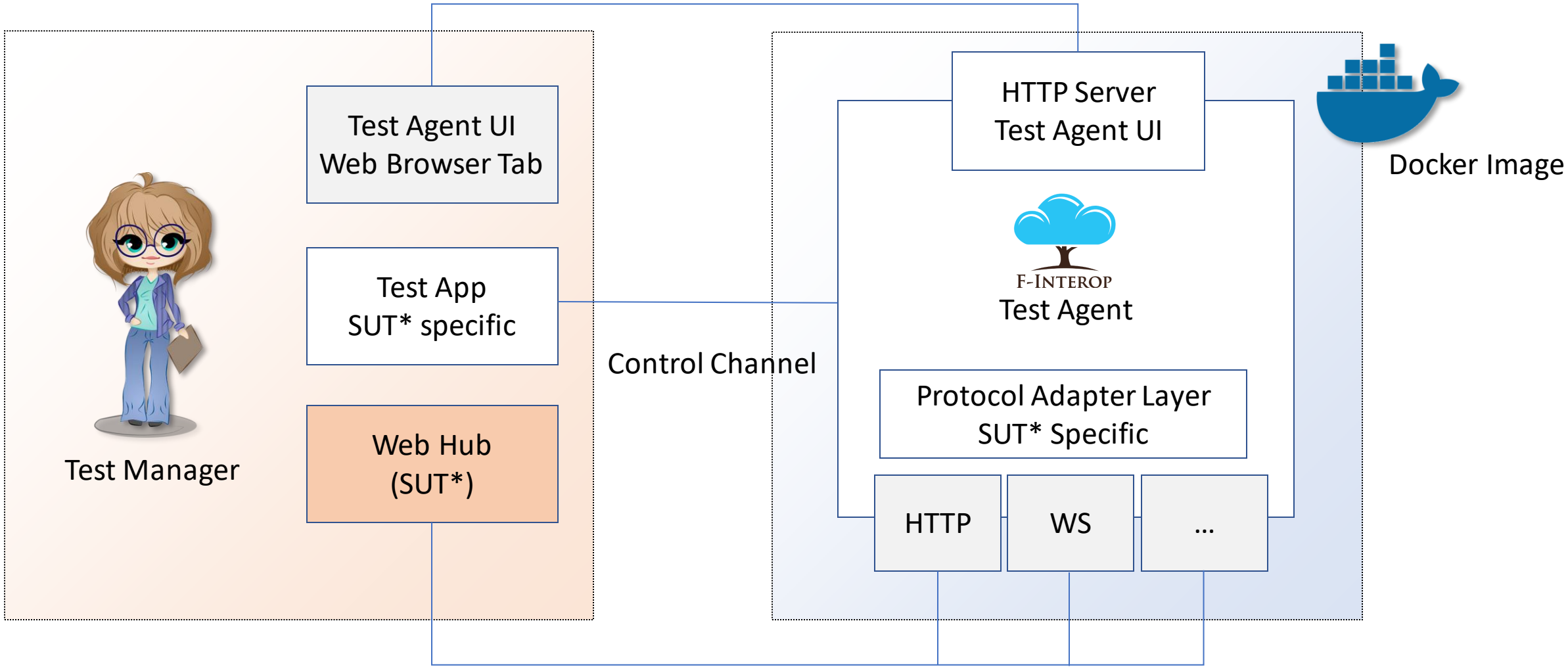
- W3C Workshop in Berlin in 2014
- W3C Web of Things Interest Group launched 2015
- W3C Web of Things Working Group launched 2017
- Aiming for W3C Recommendations in mid 2019
  - Thing Descriptions using JSON-LD + JSON Schema
  - JavaScript Scripting API
- Plus Working Group Notes for
  - Declarative protocol binding templates
    - Describing how to access existing devices
  - Guidance on security
- Future work on incubation
  - Simplified scripting API
  - Defining sub-protocol for Web Sockets

High Level Interoperability Testing

High Level Interoperability Testing

- Web Hubs may vary in respect to scripting languages and APIs
- Web Hubs may vary in how they use Web protocols
  - Many different ways to use HTTP and WebSockets
    - HTTP for all things, all properties and all events vs individual HTTP requests
    - HTTP long poll for next event vs text/event-stream with Server-Sent Events
    - Things Gateway shares WebSocket for a thing's properties, actions and events
    - ThingWeb requires separate WebSockets for each property, action and event
    - Further work is now needed to drive **convergence** on use of protocols with the involvement of stakeholders from different application sectors
      - Including work on WebSocket subprotocol
- High level interoperability testing is possible based upon the abstract contract implied by the thing descriptions
  - Properties, actions and events, and associated data types
  - Independent of scripting language and protocols

Interoperability Testing Architecture



Framework supports different kinds of tests

- Client side functional tests
  - The client-side contract implied by by TD and Scripting specs
  - Test agent invokes client-side tests directly
  - Relies on appropriately written client-side test app
- Server side functional tests
  - The server-side contract implied by by TD and Scripting specs
  - Test agent invokes server-side tests via TD actions
  - Relies on appropriately written server-side test app
- Non-functional tests
  - e.g. performance at pushing stream of 10,000 property updates

Some examples*

- should succeed on writing a valid array
- should fail on writing an array that is too short
- should fail on writing an array that is too long

Property definition in the testing TD

```
"list": {
        "type": "array",
        "minItems": 2,
        "maxItems": 3,
        "items": {
                "type": "number”
        }
}
```

```
await test_write(10, thing, 'list', [0, 1], "should succeed on writing an array", true);
await test_write(11, thing, 'list', [0], "should fail on writing an array that is too short", false);
await test_write(12, thing, 'list', [0, 1, 2, 3, 4], "should fail on writing an array that is too long", false);
```

Run each test to completion before starting next one

Test number

Test property

Test data

Test report

Test expectation:
a) should succeed or
b) should fail

** Handful of testing lego blocks, e.g.* ***test_action****, …*

Further examples …

- Test that the property of a consumed thing is the same as for the exposed thing
  - Query property value of exposed thing (request/response)
  - Query property value of consumed thing (request/response)
  - Test that the values are the same (after rendezvous for asynch responses)
- Test that a property update on an exposed thing signals the corresponding event
  - Set property value of exposed thing
  - Listen for corresponding update event from exposed thing
- Test that a property update on a consumed thing signals the corresponding event
  - Set property value of consumed thing
  - Listen for corresponding update event from consumed thing
- Test that setting unknown property triggers exception
  - Set value for unknown property of consumed thing
  - Listen for corresponding exception from consumed thing
- Test trying to set an invalid property value for some example data types
  - Set invalid property value of consumed thing
  - Listen for corresponding exception from consumed thing

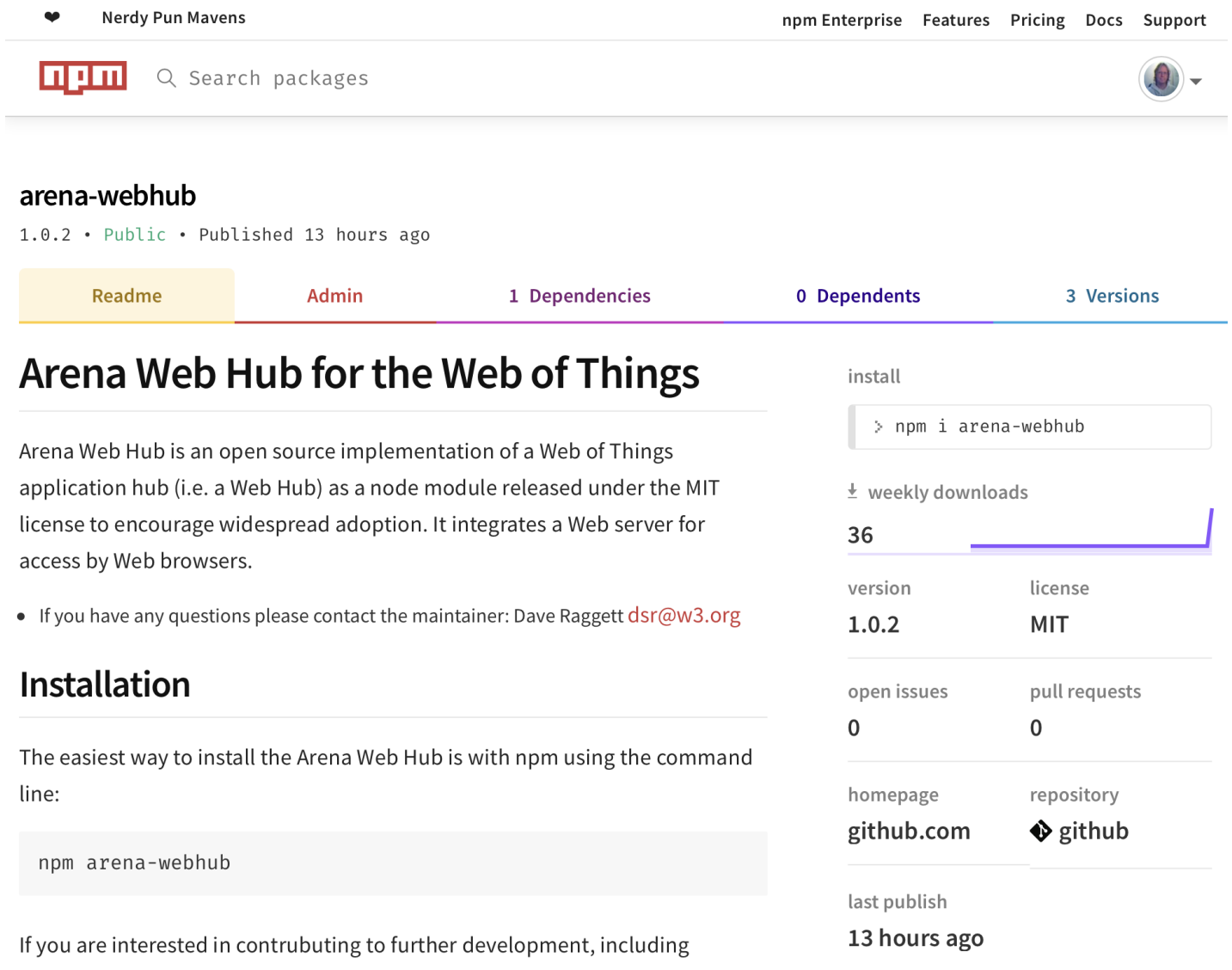
And similarly for actions and events …

Test suites and your help

- Separate test suites
  - Thing Descriptions
  - Scripting API
- Each test needs to be derived from normative statements in the specification
- Positive and negative tests
- We need additional volunteers to draft tests and independently review tests

Request for help from W3C Web of Things Working Group members

Arena Web Hub & Interoperability Testing Tool



- Arena Web Hub is a new NPM module for a secure Web of Things application server
- Open source implementation in JavaScript under MIT license
  - https://github.com/draggett/arena-webhub
- Testing tool in the examples folder
  - See “Interop Testing” sub-folder
- Client-side library with support for multiple platforms: currently Thing Web, Things Gateway and Arena Web Hub
- Interop tests with Arena Web Hub, Thing Web and soon Mozilla Things Gateway

Demos & Questions

Source code is available at:

https://github.com/draggett/arena-webhub

See “examples/Interop\ Testing” folder for testing tool

Many thanks for listening: please contact Dave Raggett dsr@w3.org for further details