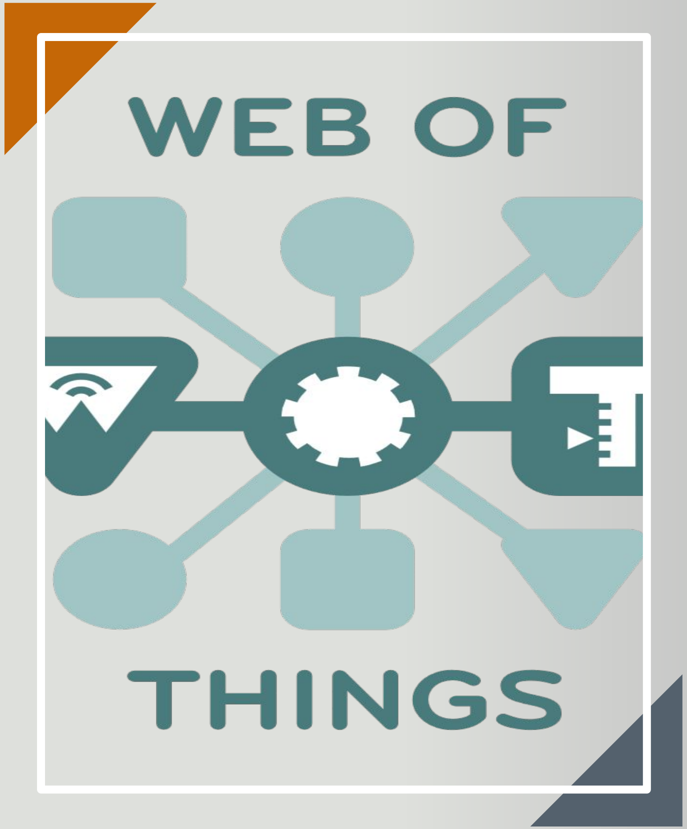

Web of Things Scripting API Status & Demo

Zoltan Kis, Intel
Daniel Peintner, Siemens

September 2019, TPAC Fukuoka

WoT Scripting API standardization status

- In the WoT IG
  - Proposals
  - Discussed in weekly calls
  - Tested on plug-fests
- In the WoT WG
  - GitHub repository
  - Proposals in GitHub issues
  - Several versions:
    - Editor's Draft (ED)
    - First Public Working Draft (FPWD)
    - Working Draft (WD)
    - WG Note

Initial ED:	February 2017
FPWD:	14.09.2017
WD1:	05.04.2018
WD2:	29.11.2018
WD3:	11.09.2019

Reference implementation: node-wot

The WoT API object

```
interface WOT {
  Promise<ConsumedThing> consume(ThingDescription td);
  Promise<ExposedThing> produce(ThingDescription td);
  ThingDiscovery discover(optional ThingFilter filter);
};
```

New: *conformance classes* for implementations.

- WoT Consumer conformance class
- WoT Producer conformance class
- WoT Discovery conformance class

To create and expose a Thing, we need a TD.

Client API: ConsumedThing

```
interface ConsumedThing {
  constructor(ThingDescription td);
  Promise<any> readProperty(DOMString propertyName, optional InteractionOptions options = null);
  Promise<PropertyMap> readAllProperties(optional InteractionOptions options = null);
  Promise<PropertyMap> readMultipleProperties(sequence<DOMString> propertyNames, optional InteractionOptions options = null);
  Promise<void> writeProperty(DOMString propertyName, any value, optional InteractionOptions options = null);
  Promise<void> writeMultipleProperties(PropertyMap valueMap, optional InteractionOptions options = null);
  Promise<any> invokeAction(DOMString actionName, optional any params = null, optional InteractionOptions options = null);
  Promise<void> observeProperty(DOMString name, WotListener listener, optional InteractionOptions options = null);
  Promise<void> unobserveProperty(DOMString name);
  Promise<void> subscribeEvent(DOMString name, WotListener listener, optional InteractionOptions options = null);
  Promise<void> unsubscribeEvent(DOMString name);
  ThingDescription getThingDescription();
};
dictionary InteractionOptions {
  object uriVariables;
};
typedef object PropertyMap;
callback WotListener = void(any data);
```

Once a Thing is found, scripts can

- observe properties and events
- change it using properties and actions.

The client needs access rights (provisioning is out of scope).

Server API: ExposedThing

```
interface ExposedThing: ConsumedThing {
  ExposedThing setPropertyReadHandler(DOMString name, PropertyReadHandler readHandler);
  ExposedThing setPropertyWriteHandler(DOMString name, PropertyWriteHandler writeHandler);
  ExposedThing setActionHandler(DOMString name, ActionHandler action);
  void emitEvent(DOMString name, any data);
  Promise<void> expose();
  Promise<void> destroy();
};
callback PropertyReadHandler = Promise<any>(optional InteractionOptions options = null);
callback PropertyWriteHandler = Promise<void>(any value, optional InteractionOptions options = null);
callback ActionHandler = Promise<any>(any params, optional InteractionOptions options = null);
```

A server Thing can

- programmatically create a TD
- define behavior for client requests:
  - get/set Property
  - invoke Action
  - observe Events.

Discovery API

```
interface ThingDiscovery {
  constructor(optional ThingFilter filter = null);
  readonly attribute ThingFilter? filter;
  readonly attribute boolean active;
  readonly attribute boolean done;
  readonly attribute Error? error;
  void start();
  Promise<ThingDescription> next();
  void stop();
  };


typedef DOMString DiscoveryMethod; // "any", "local", "directory", "multicast"

dictionary ThingFilter {
  (DiscoveryMethod or DOMString) method = "any";
  USVString? url;
  USVString? query;
  object? fragment;
};
```

Discovery provides TDs (as JSON objects):

- Things exposed in the local WoT Runtime
- Things listed in a directory service
- Things exposed in a local network.

node-wot

One implementation of the Scripting API

Dual W3C and Eclipse license

The *de-facto* reference implementation

node-wot: *a* Scripting API implementation

- node-wot is an open-source implementation of the WoT Scripting API
  http://www.thingweb.io

- The project can be fully customized using various packages
  - td-tools
  - core
  - bindings (HTTP, CoAP, MQTT, WebSockets, …)
    - Other binding protocols can be added by fulfilling a given API
    - Content codecs (besides JSON, text, and octet-stream) can be added
  - Miscellaneous: demos, command-line interface

- Facts
  - NodeJS implementation in TypeScript
  - Development on GitHub: https://github.com/eclipse/thingweb.node-wot/
  - Dual-licensed: Eclipse Public License v. 2.0 and W3C Software Notice and Document License
  - Available through NPM (packages such as core, td-tools, …)

node-wot - Demos and Tools

- Web UI
  - node-wot can be used as a browser-side JavaScript library (~160kB JS code)
  - http://plugfest.thingweb.io/webui/

- TD Playground
  - Tool to check the validity of a TD
  - Performs both syntactic checks and semantic checks
  - http://plugfest.thingweb.io/playground/

- TD Directory
  - REST interface to add, update and query TDs for discovery purposes
  - http://plugfest.thingweb.io

Demo

- Server script (`ExposedThing`) example (`counter.js`)

```
$ node packages\cli\dist\cli.js examples\scripts\counter.js
```

- Client script (`ConsumedThing`) example (`counter-client.js`)

```
$ node packages\cli\dist\cli.js --clientonly examples\scripts\counter-client.js
```

- Browser Client example
  - Pointing to Property, Action, Event
  - Listening to events
  - Changing with different binding (e.g., CoAP) values