# WoT Scripting

Virtual F2F, June 2020

# Agenda

- Progress
- Opens
- Next steps
- Implementation update: node-wot

# WoT Scripting API specification updates

Work done since the March vF2F:

- Use Web Platform cross referencing and more modern ReSpec:
  https://github.com/w3c/wot-scripting-api/pull/207
- Add *formIndex* in order to select *Form*:
  https://github.com/w3c/wot-scripting-api/pull/203
- Handling *DataSchema* vs *contentType* (add *InteractionData*):
  https://github.com/w3c/wot-scripting-api/pull/209
- Add support for *ExposedThing* property observe, unobserve and event handlers:
  https://github.com/w3c/wot-scripting-api/pull/218
- Stub text on error mapping: https://github.com/w3c/wot-scripting-api/pull/221
- Make WOT a namespace: https://github.com/w3c/wot-scripting-api/pull/220
- *InteractionInput* and *InteractionOutput*, support streams
  https://w3c.github.io/wot-scripting-api/#handling-interaction-data

Current API at https://w3c.github.io/wot-scripting-api/#idl-index .

# Opens

- Error handling, https://github.com/w3c/wot-scripting-api/issues/200
  PR (partial solution): https://github.com/w3c/wot-scripting-api/pull/221
- Reading/writing multiple properties (TD):
  https://github.com/w3c/wot-scripting-api/issues/219
  https://github.com/w3c/wot-thing-description/issues/848
- Further clarifications for *DataSchema* vs *contentType*:
  https://github.com/w3c/wot-thing-description/issues/912
- Improving clarity on URI variables vs actions (TD):
  https://github.com/w3c/wot-thing-description/issues/910
  https://github.com/w3c/wot-thing-description/issues/913
- Discovery API, https://github.com/w3c/wot-scripting-api/issues/206
  - agreed to substitute the idea of “Fetch convenience API” to “direct discovery”
- Semantic API, https://github.com/w3c/wot-scripting-api/issues/204
  - no work done on this
- Script deployment and management
  - node-wot
  - WASM/WASI not explored yet.

# Deployment scenarios

1. Manufacturer implements WoT server that exposes TD and Thing services. Flashes image on boards. Code can be reused between different devices (same API) when the JS runtime + bindings are supported. It's convenient to use a standard API for multiple solutions.

2. It would be nice to be able to distribute WoT scripts to devices working in a given management realm.
    a. On powerful devices that can run Node.js and npm, scripts can be distributed as Node.js packages. Local system API can be exposed via libraries and scripts can be run from console or via service.
    b. Managed by the WoT runtime in the language of choice (currently JavaScript). Exposes its own script management interface that can be e.g. a Manager Thing.
    c. Managed by a native container runtime to run a single WoT script + runtime in a sandbox. Exposes its own script management API, perhaps through another special container that runs WoT and exposes a manager Thing. Could be integrated with orchestrators (e.g. Kubernetes).
    d. A variant of the previous, a WASM+WASI container runtime that runs a single script in a sandbox. It can consume WASM intermediary binary format generated from any supported language. Module management is tied to WASM. WASI needs to be *standardized*.
    e. Run WoT **inside browsers**
        i. as a PWA, with or without WASM, isolating from WoT dependencies
        ii. As node-wot is doing with Web UI: http://plugfest.thingweb.io/webui/

# Next

- Error mapping (Bindings TF)
- Discovery API update (Discovery TF)
  - Direct discovery (as convenience API instead of Fetch)
  - Directory
  - Filtering, querying
- Reading/writing multiple properties (TD TF)
- More *DataSchema*/*contentType* clarifications (TD TF)
- More URI variables vs actions clarifications/examples (TD TF)
- Actions progress handling (TD TF)

# Status node-wot

- Protocol Support: HTTP/HTTPS, CoAP/CoAPs, Websocket, MQTT, OPC-UA, NETCONF, Modbus (WIP)
- MediaType Support: JSON, Text, Base64, OctetStream
- Running in browsers: see http://plugfest.thingweb.io/webui/
- Recent Updates:
  - Updates of Hands-on material and videos/tutorials
  - OAuth2 client-side (credential flow) implementation
    See also Smart coffee machine and oAuth 2.0 example
  - OAuth2 server-side implementation (WIP PR based on RFC7662)
  - Improved logging
  - Note: The current Scripting API is not yet in master (WIP).