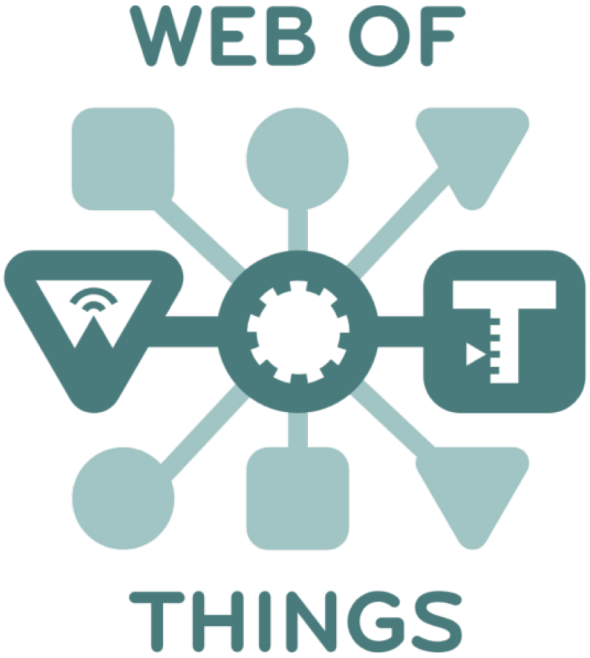



# W3C WoT Standardization

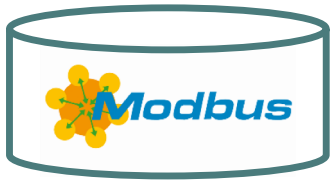
2nd W3C WoT Workshop, Munich, Germany, 4/5 June 2019

# The Internet of SILOS


CoAP


CoAP


MQTT



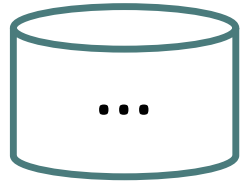

Modbus TCP

CoAP

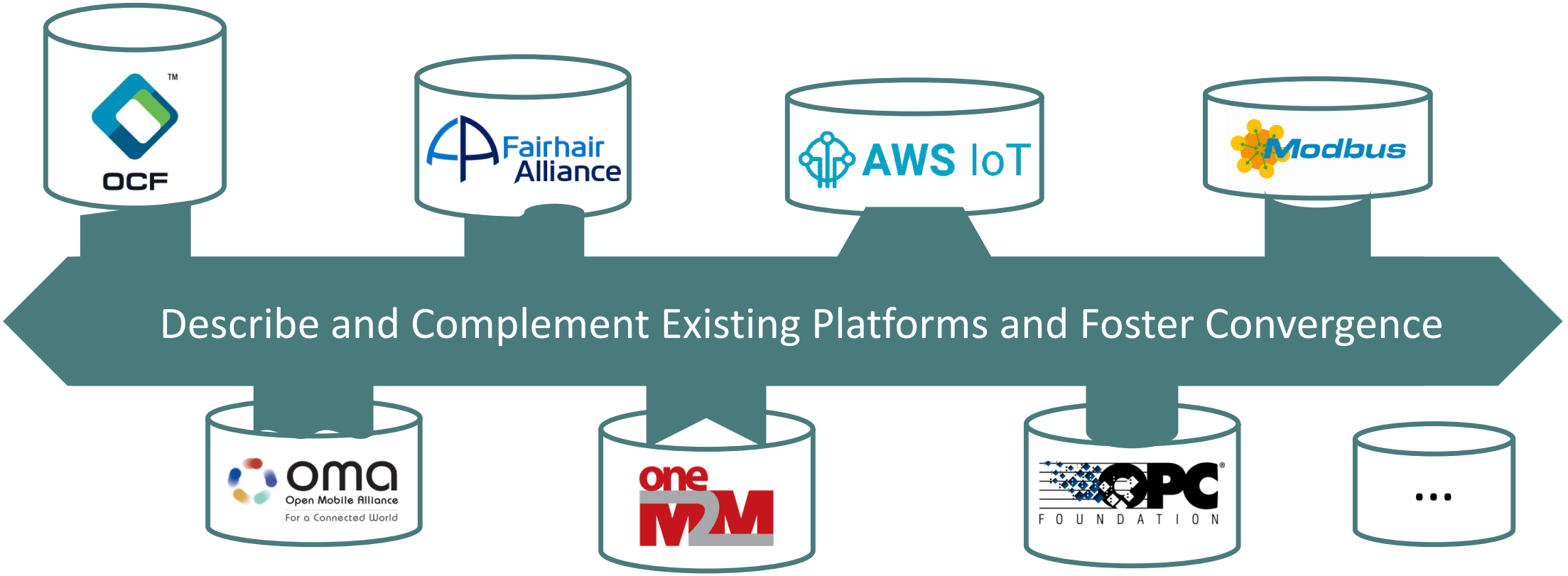

HTTP / CoAP / MQTT


UA Binary


HTTP / MQTT



# Counter the Fragmentation in the IoT



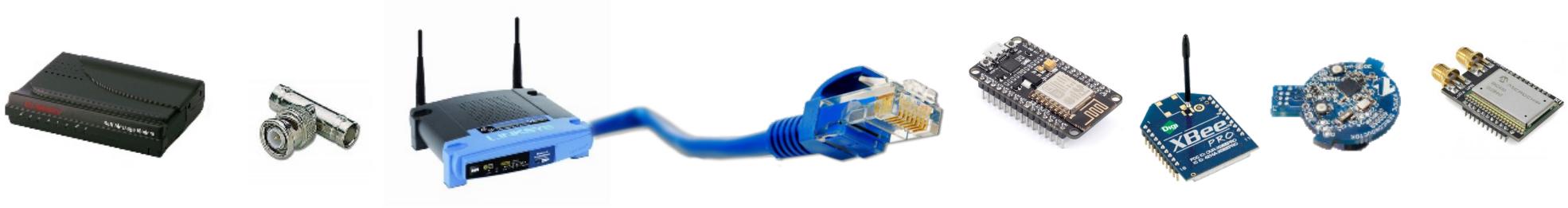
# From the IoT to the Web of Things

World Wide Web (WWW)

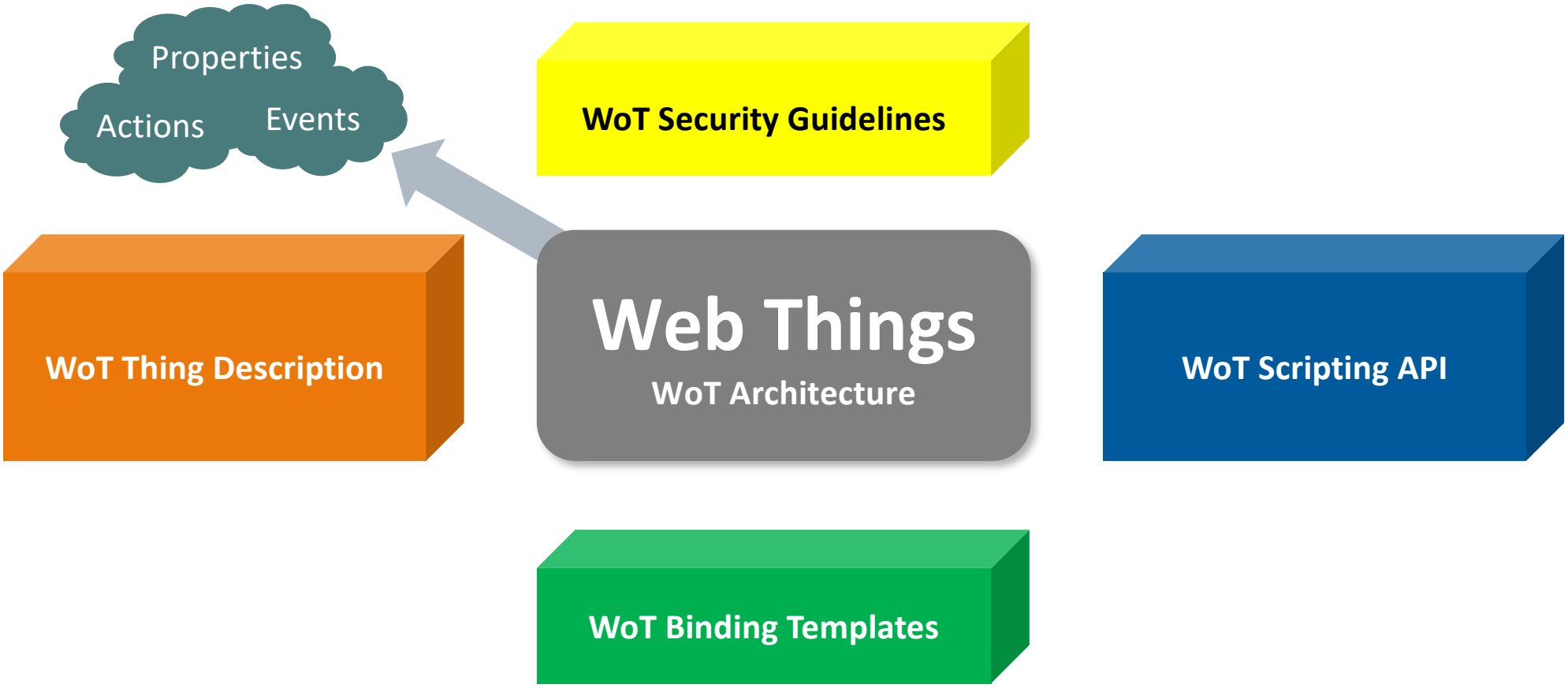
**Web of Things (WoT)**

Internet

Internet of Things (IoT)

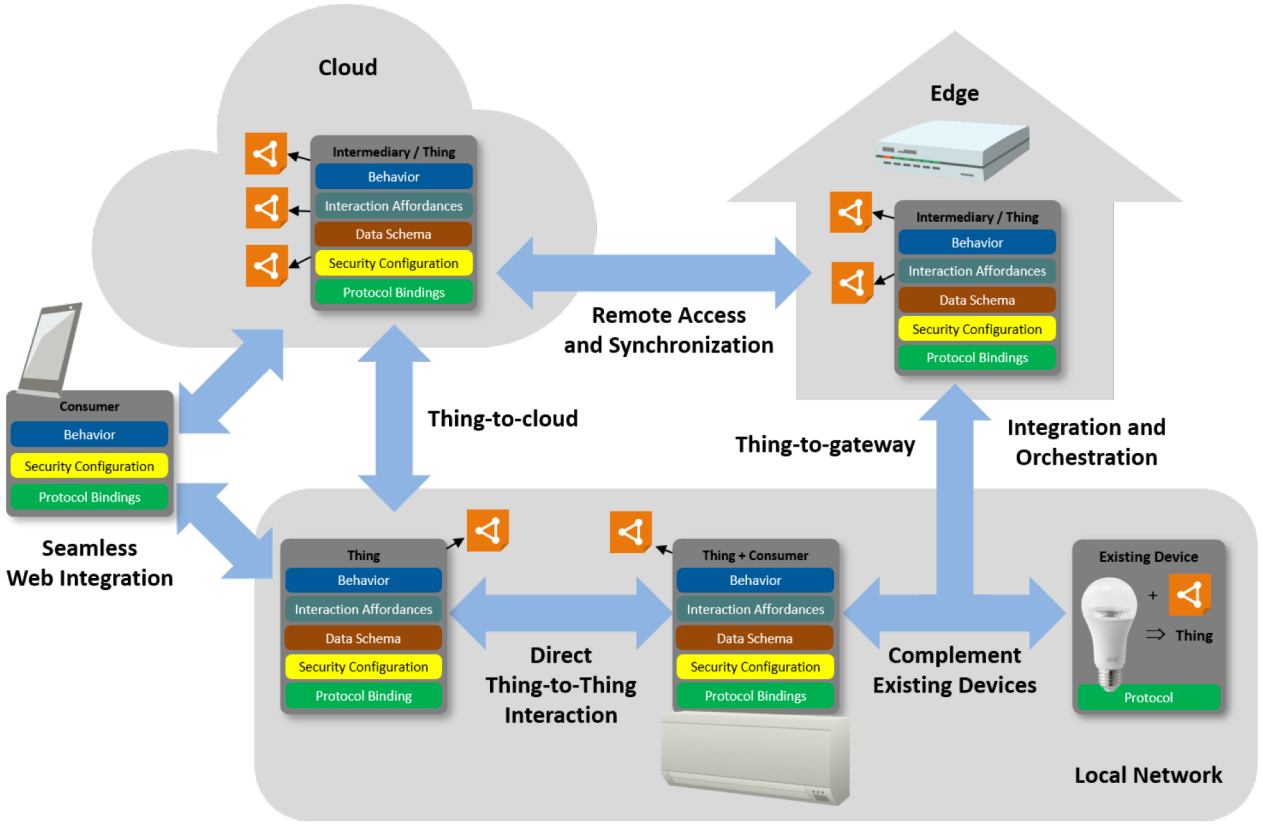


# W3C WoT Building Blocks



# WOT ARCHITECTURE

# Abstract WoT Architecture



# Interaction Model

- Properties
  - Describe the state of a thing
- Actions
  - Describe how to use a thing
- Events
  - Enable a thing to communicate state changes

# Hypermedia Controls

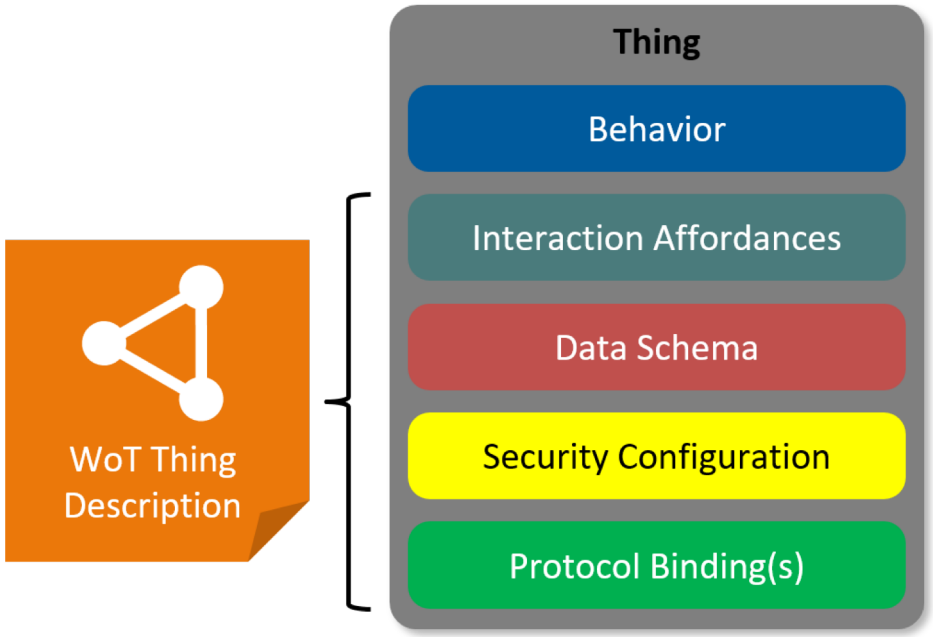
- Links
  - Enable modeling relationships between things
  - Context
  - Relationship type
  - Link target and optional target attributes

- Forms
  - Context
  - Operation type
  - Submission target
  - A request method

# Building Blocks

- Thing Description
  - Information model, semantic vocabulary, serialized representation JSON LD
- Binding Templates
  - Blueprints for communication metadata
- Scripting API
  - ECMA Script based API
- Security and Privacy Guidelines
  - Cross-cutting security guidelines for each building block

# WoT Thing Description

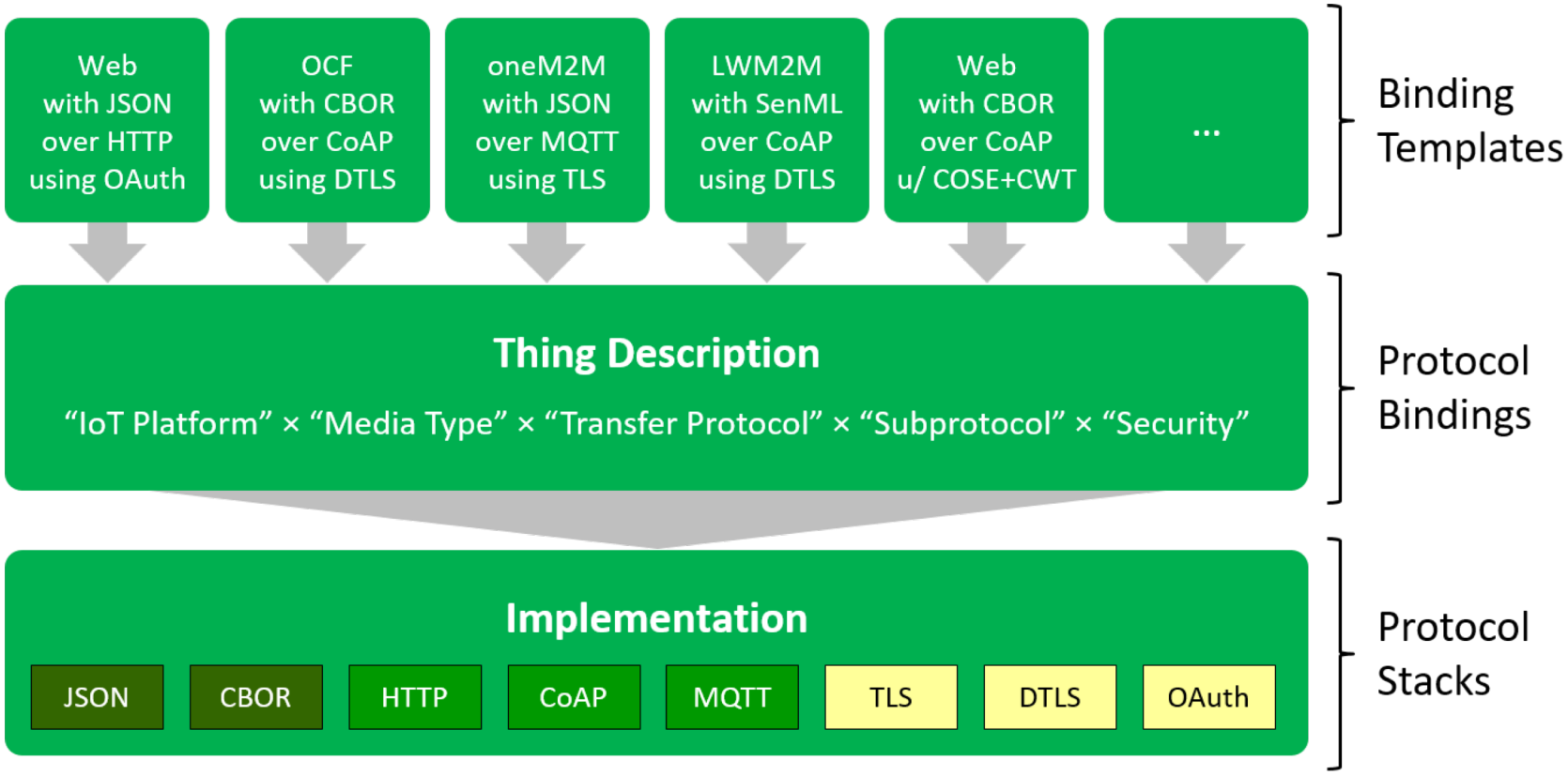



WoT Thing Description describes several architectural aspects of a thing.

Thing description are used by consumers, who can interact with the thing, based on information in the TD.

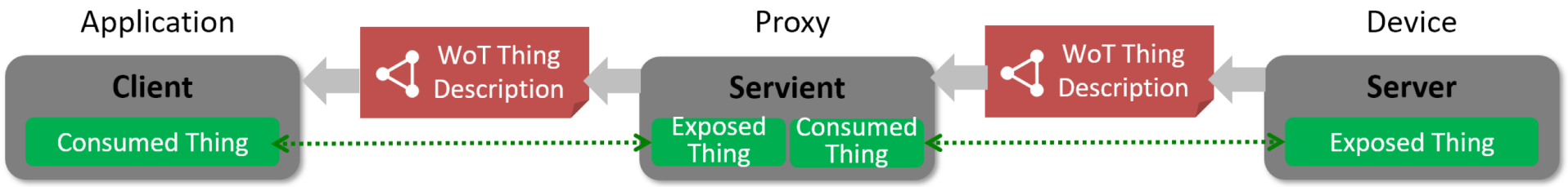
A JSON-LD based serialisation format is defined in the Thing Description specification.

# Protocol Bindings



# Direct Communication

# Indirect Communication



# WOT THING DESCRIPTION

# The WoT Thing Description
# The *index.html* for Things

What kind of data do you serve?

Who are you?

How does the payload structure look like?


Are there some context information (e.g., unit)?

How can I access the data/function?

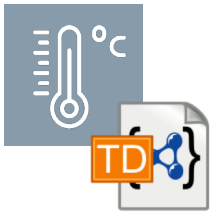
What kind of functions do you have?

What kind of protocols & serializations do you support?

Are there some security constrains?

Do you have other relations to other Things?

# Describe Things with TDs





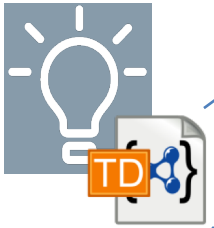
```
{
 "@context": [...],
 "@type": "Thing",
 "id": "urn:dev:ops:13473-temp-12",
 "title": "Temperature",
 "security": { "scheme": "basic" },
 ....
 "properties" : {
  "value": {
     "type": "number",
     "minimum": "-40.2",
     "maximum": "48.4",
     "unit": "Celsius",
     "forms": [{
       "href": "http://192.168.0.1/temp",
       "contenttype": "application/json"
  }]}}}
```




```
{
 "@context": [...],
 "@type": "Thing",
 "id": "urn:dev:ops:32473-robot-1",
 " title": "Robotic Arm",
 "security": { "scheme": "basic" },
 ....
 "actions": {
  "openGrip": {...},
  "closeGrip": {...},
  "moveGripTo": {...}
 }
}
```



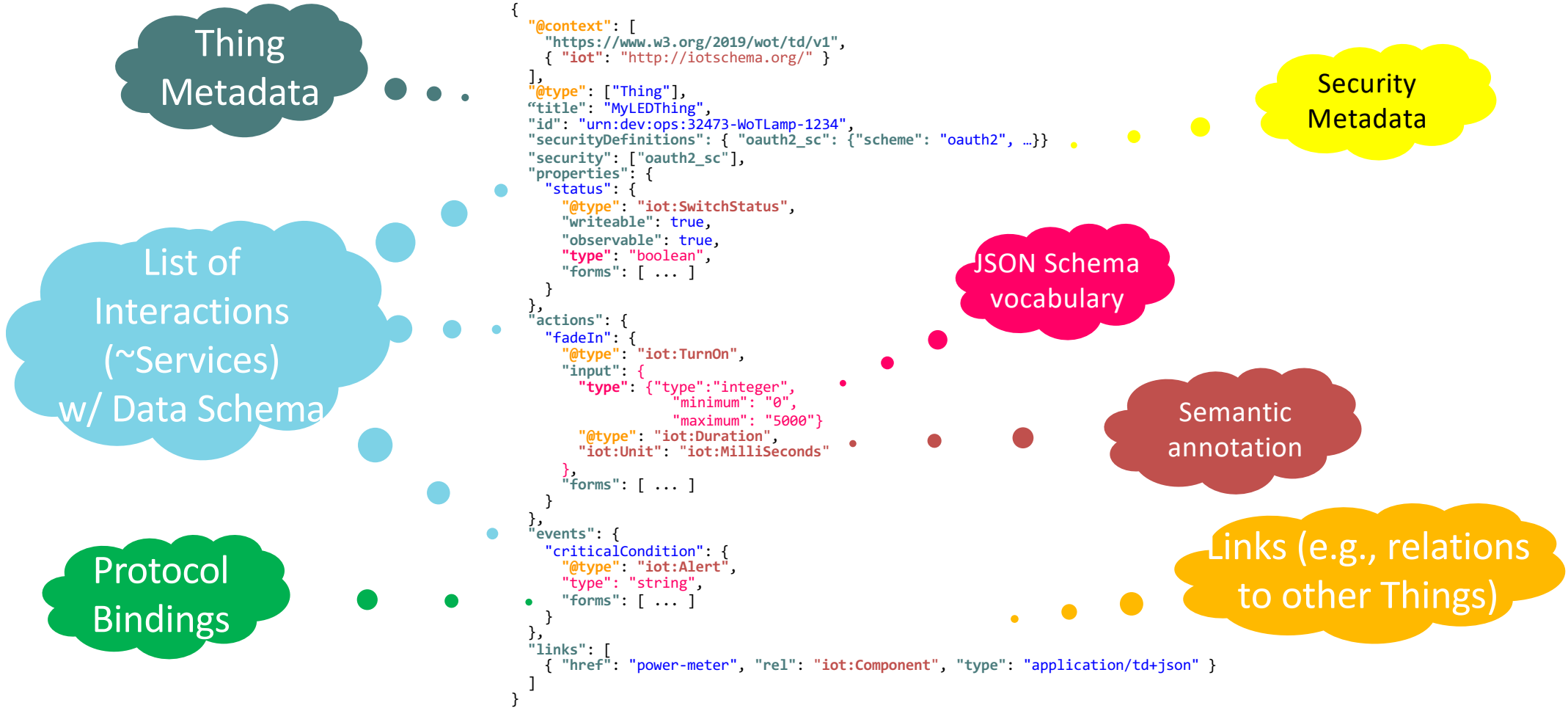

```
{
 "@context": [...],
 "@type": "Thing",
 "id": "urn:dev:ops:42473-engine-12",
 "title": "Engine",
 "security": { "scheme": "oatuh2" },
 ....
 "properties" : {
  "status": {...},
  "speed": {...}
 },
 "actions": {
  "onOff": {...}
  }
}
```



```
{
 "@context": [...],
 "@type": "Thing",
 "id": "urn:dev:ops:53473-lamp-12",
 "title": "Lamp",
 "security": { "scheme": "basic" },
 ...
 "properties" : {
  "status": {...}
 },
 "actions" : {
    "toggle" : {...}
 },
 "events" : {
    "overheating" : {...}
 }
}
```

# WoT Thing Description – JSON-LD based Document Format



# WOT BINDING TEMPLATES

# WoT Binding Templates – Instantiated in TDs

Basics to build the request

Deviation from defaults

```
...
"properties": {
  "brightness": {
    ...
    "forms": [
      {                                                  // Default: GET to read, PUT to write
        "href": "https://myled.example.com:8080/pwr",
        "contentType": "application/json"
      }
    ]
  }
},
"actions": {
  "fadeIn": {
    ...
    "forms": [
      {
        "href": "coaps://myled.example.com:5684/pwr",
        "contentType": "application/ocf+cbor",
        "coap:methodCode": 3,                            // PUT instead of POST to invoke
        "coap:options": [ {
          "coap:optionNumber": 2053,                     // OCF-Content-Format-Version
          "coap:optionValue": "1.1.0"
        } ]
      }
```

# WOT SCRIPTING API

# Scripting API standardization

- 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

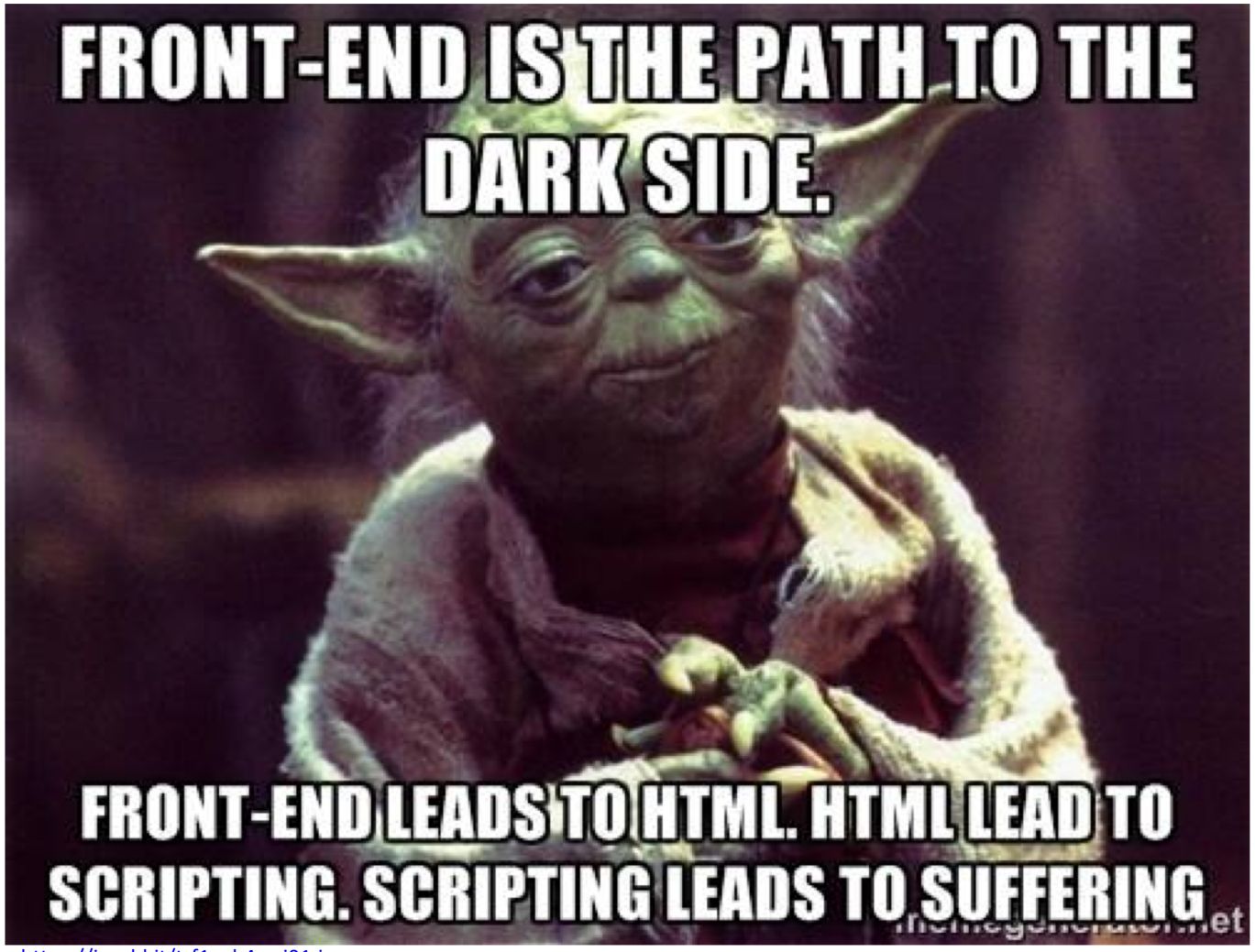
WD1: 05.04.2018

WD2: 29.11.2018

WG Note: June 2019 (work can continue)

Reference implementation: node-wot

# Why a Scripting API?




https://i.redd.it/trf1qch4ywi01.jpg

# Why a Scripting API

- Scripting has transformed the Web
  - Marc Andreessen, the founder Netscape, “believed that HTML needed a ‘glue language’ that was easy to use by Web designers and part-time programmers to assemble components such as images and plugins, where the code could be written directly in the Web page markup.”
  - Brendan Eich wrote Java-inspired Mocha in 10 days in May 1995
  - Later called LiveScript, then JavaScript, then standardized as ECMAScript
  - 10.7 million JavaScript developers in 2018 (out of 23 million)

- WoT describes and integrates IoT platforms through Web technologies
  - addressing, discovery, access control, data transfer, and
  - **scripting**.

# Scripting API

- Web page → Thing
- URL → URI
- HTTP → HTTP, CoAP, BLE, WS
- HTML → Thing Description
- **ECMAScript → WoT Script**
- Web search → Discovery
- Served page → Exposed Thing
- Rendered page → Consumed Thing

# Scripting API place in WoT architecture

## Thing Description (TD)

Metadata describing the data model, security & interactions.

## Scripting API

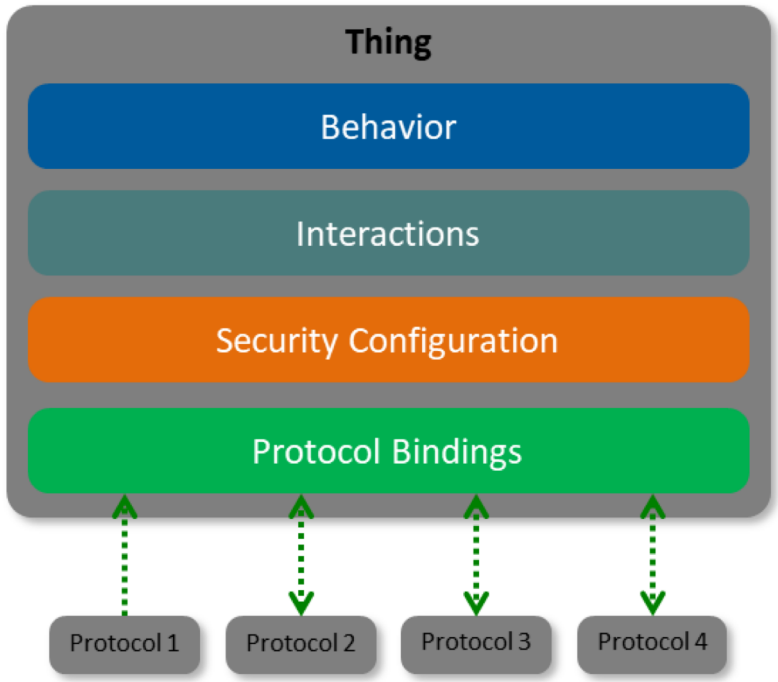
A standardized API to control Thing interactions and implement behaviour.

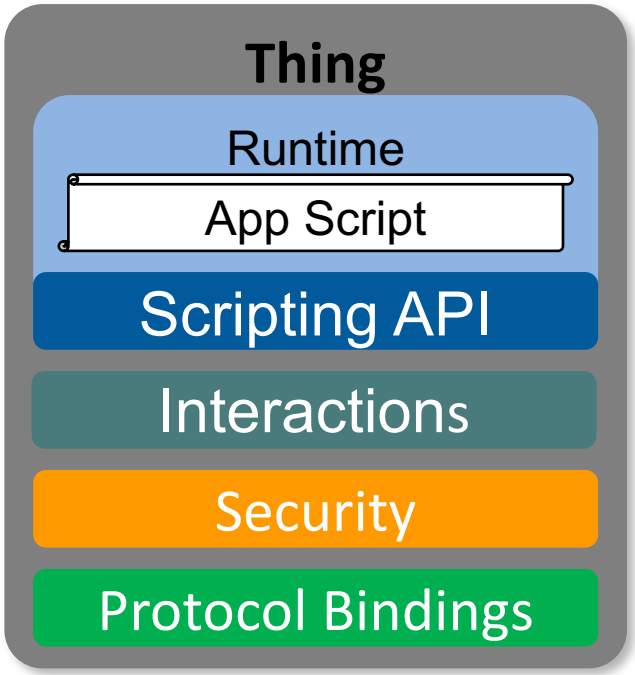
## Protocol Bindings

Describes how to translate WoT interactions to the underlying protocols.

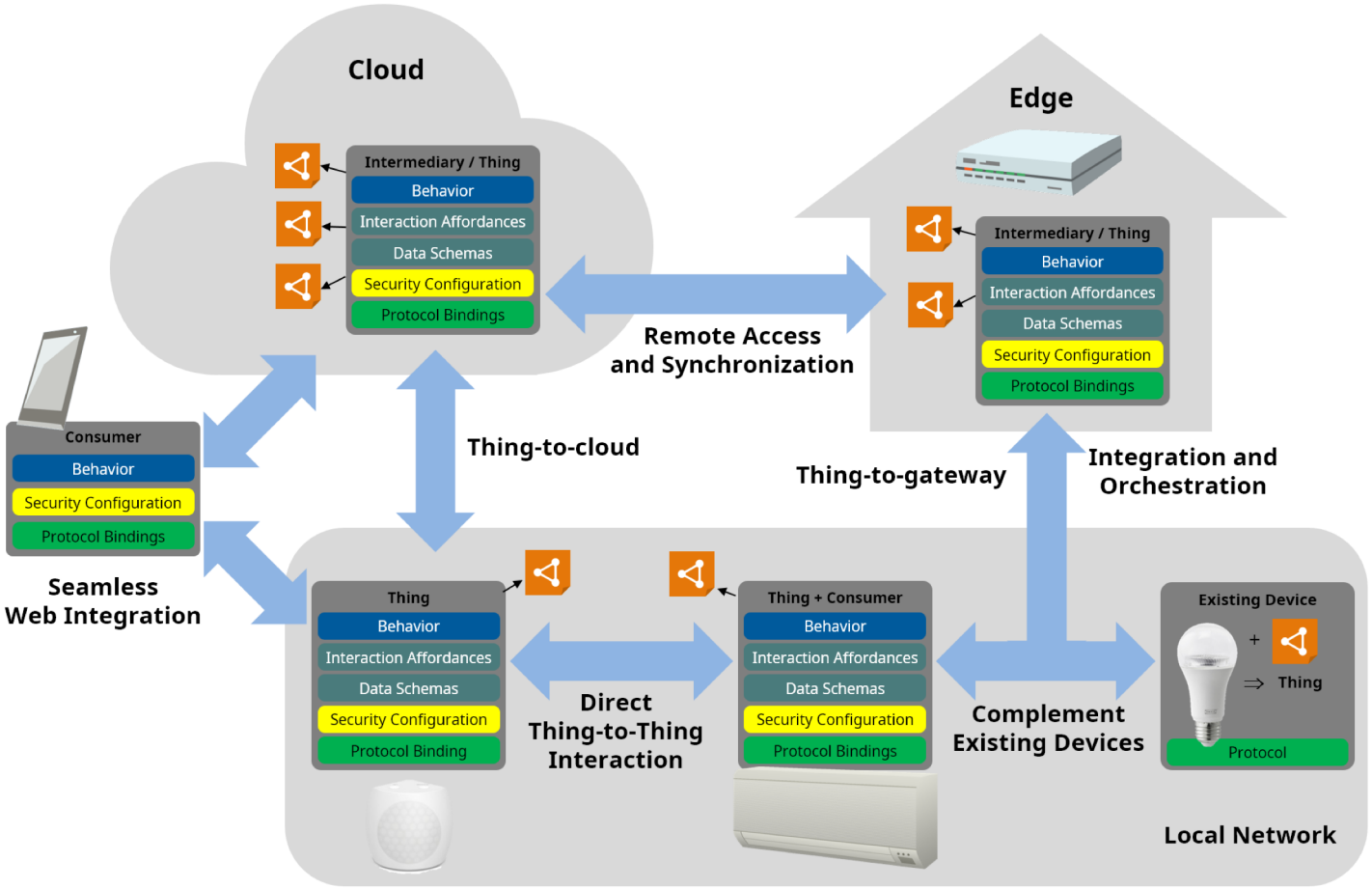
## Security & Privacy

Ensures that all building blocks provide means to describe the security and privacy mechanisms used in underlying platforms.



# Scripting API use cases



# Approaches to the Scripting API

| | |
|---|---|
| No externally exposed API<br>(only WoT network interface) | A WoT gateway can encapsulate other IoT deployments:<br>- presents a REST-ful API towards clients<br>- implements IoT protocols towards IoT deployments |
| Simple API<br>`lock = WoT.consume('https://td.my.com/lock');`<br>`print(lock.status);`<br>`lock.open();` | `Thing = object`<br>`Thing Property = object property`<br>`Thing Action = object method`<br>`Thing Event = event`<br>`WoT API object = lifecycle methods` |
| Current API (based on the TD spec)<br>`lock = WoT.consume('https://td.my.com/lock');`<br>`print(lock.readProperty('status'));`<br>`lock.invokeAction('open');` | `Thing Description = data object`<br>`Thing = TD instance + API methods`<br>`WoT API object = lifecycle methods` |

# WOT SECURITY AND PRIVACY GUIDELINES

# Security and Privacy Guidelines

- *Security and Privacy Considerations* sections in each of the Architecture and Thing Description documents
- Metadata supporting security mechanisms in TD
  - Can be easily extended with vocabulary extensions
- Delivered separate document: *Security and Privacy Guidelines* Note
  - Covers threat model, risks, and mitigations
  - Testing plan including adversarial testing
  - Previously *Security and Privacy Considerations* + *Security Testing Plan*, content to be merged and published as a single Note
- Work in Progress: *Security and Privacy Best Practices*

# Contact

**Kazuyuki Ashimura**

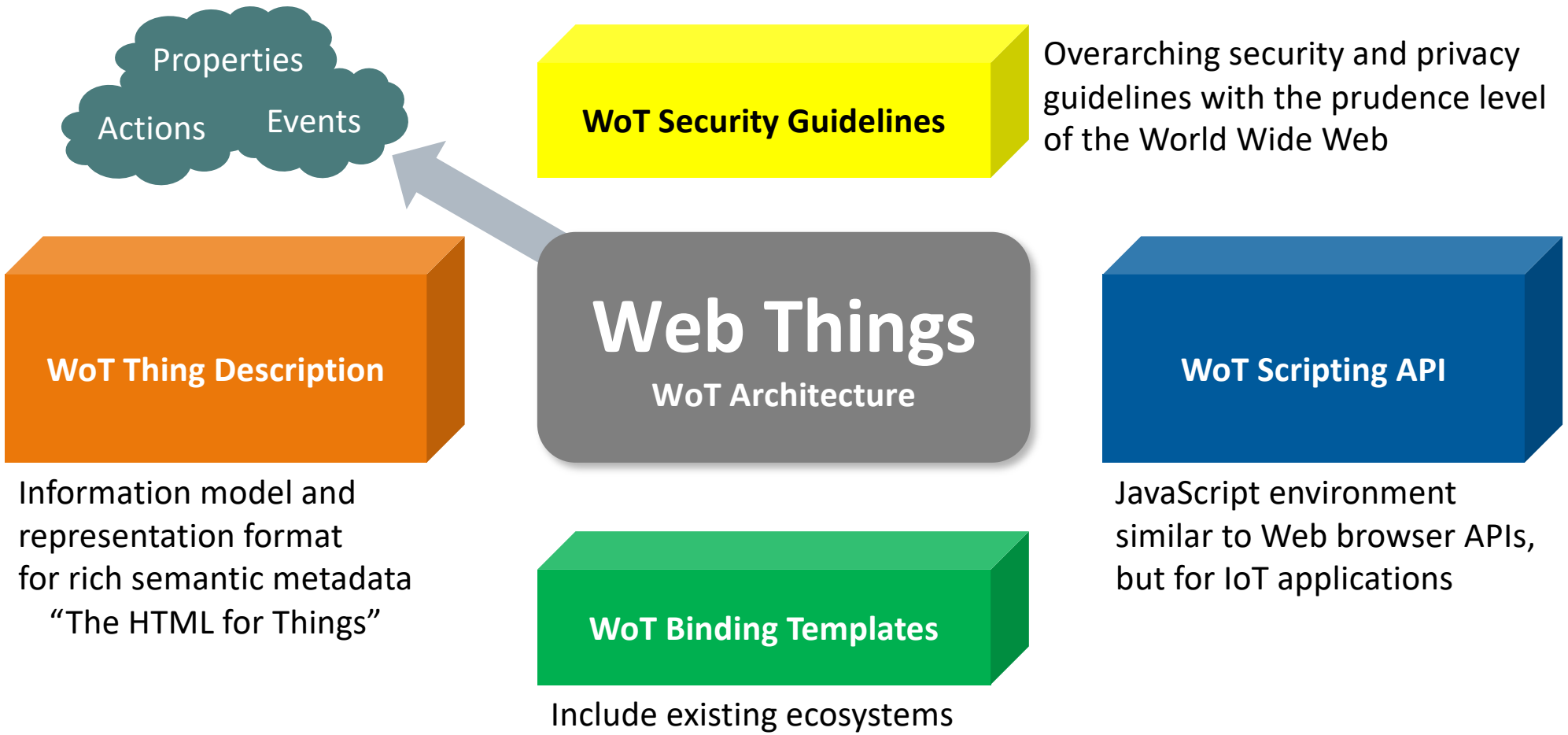
W3C Staff

Web of Things Team Contact

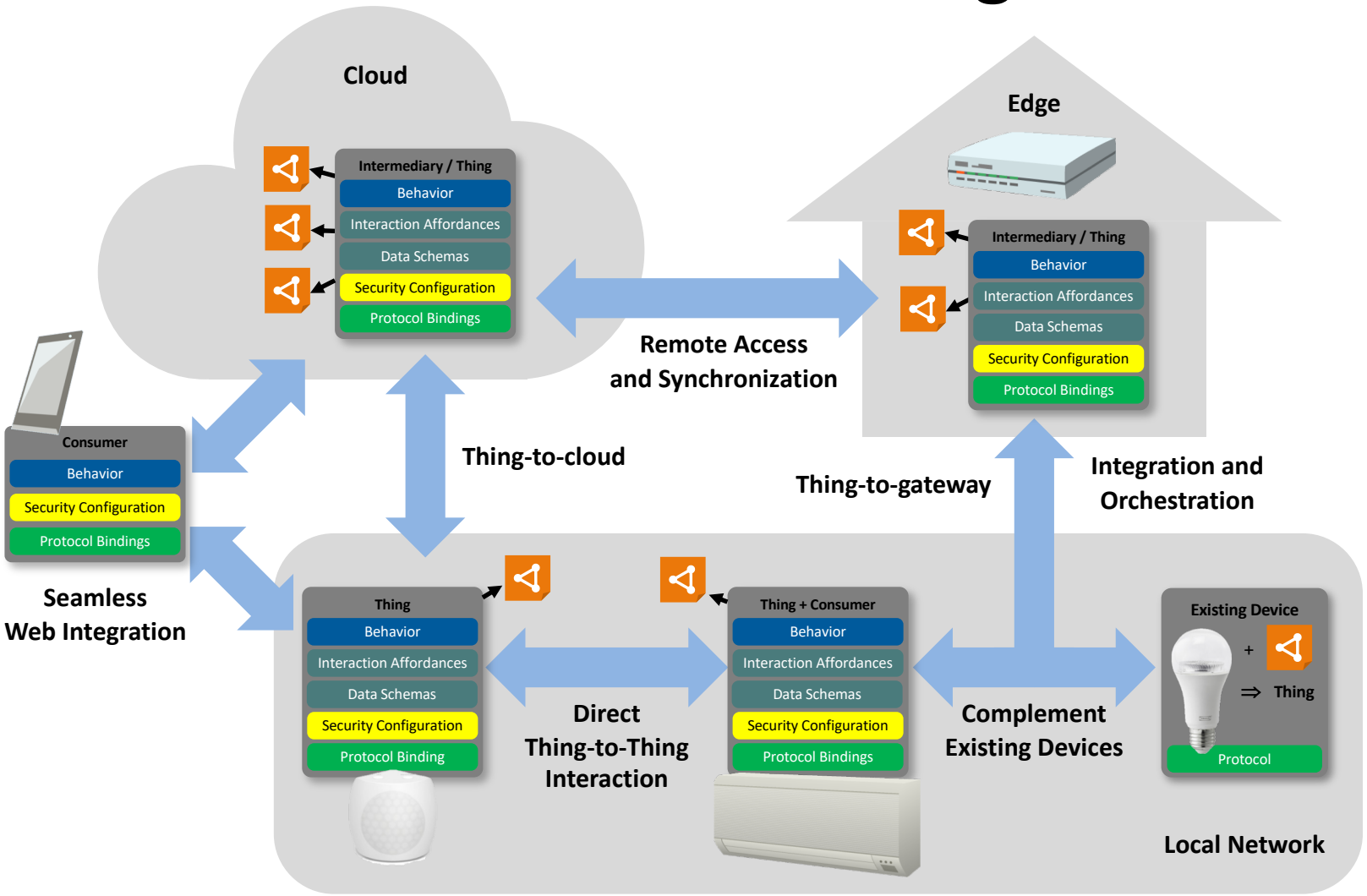
ashimura@w3.org

team-wot@w3.org

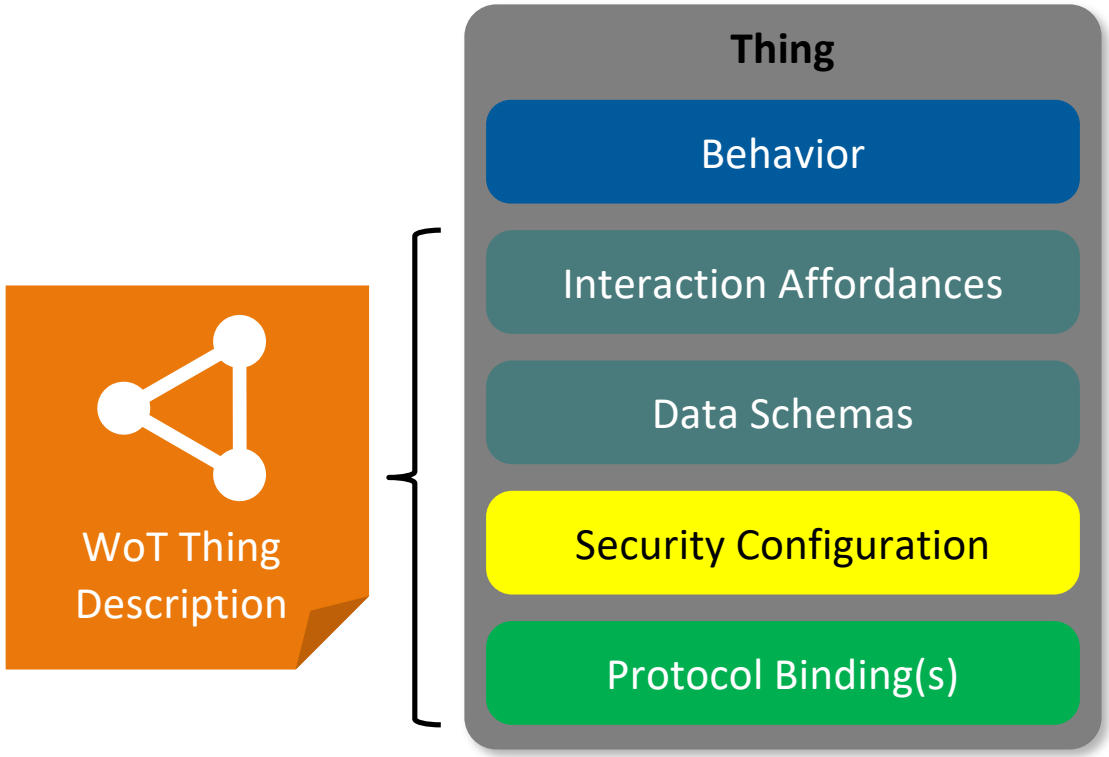
# <For Color Picking>



# <For Color Picking>



# <For Color Picking>



# <For Color Picking>