Interactive module diagram component

Overview & goals

A component is needed to create interactive visualizations of ShellyX modules & dev boards, as well as exporting the generated images into SVG graphics for inclusion in printed documentation.

Use cases

Visualization of manufacturer configurations on the ShellyX portal

TODO

Generation of images for printed documentation and user manuals

TODO

Data model

```
io
  id
  role: enum - input / switch / cct cool / cct hot / ...


"ios": [
  {
    "id": 0,
    "role": "input" // enum: input / switch / cct cool / cct hot / ...
  },
  {
    "id": 1,
    "role": null
  },
  {
    "id": 2,
    "role": null,
    "rolesAvailable": ["input", "output"] // if provided - limit roles; if missing - all roles are allowed
  },
]

SVG element ids:

  io-2
  io-label-2
  io-arrow-2

SVG role CSS classes:

  role-input
  role-switch
  role-cct-cool
  role-cct-hot
  ...


```

Technical requirements

Technology: **pure/vanilla JavaScript; standard WebComponent**

Runtime: **browser** & **server side**

Component must be agnostic to roles and number of pins

Component initialization :

- **base SVG image URL**, with active elements marked with `id` and `class` attributes, as defined in this document
- **current configuration** - initial state for rendering and updatable throughout the lifetime of the component using methods on the JS interface of the component.
- **selector** for DOM component to render into
- io roles available on the diagram

Output :

- visualization in a DOM component (if running in browser)
- image output to a file specified in the input (if running server side)
- error messages with details, if the image provided does not comply with the requirements - e.g. missing SVG elements with certain `id` and `class` attributes set.

Notes

When loading SVG, fixed viewport & sizes must be removed - the SVG must fit into the container.

```
svg?.removeAttribute('width');
svg?.removeAttribute('height');
```

Open questions

How to provide the CSS to the diagram component?

Usage

Using the web component on a web page should look something like this:

```
. . .

<div id="module-diagram"></div>

. . .

<script src="path/to/module-diagram-component.js" type="text/javascript" defer></script>
<script type="text/javascript">
(function() {
  // when instantiated, the component displays the diagram in the DOM element specified in the constructur
  let diagram = new ModuleDiagram(
      "#module-diagram",             // selector for target element
      "/path/to/svg-base-image.svg", // URL to fetch the SVG image from
      {                              // initial state
         . . . as defined above . . .
      }
  );

  . . .

  // when something in the configuration UI changes, the component must support updating
  // the diagram by providing a partial state change
  function handleConfigurationChange() {
    diagram.update({
         . . . as defined above . . .
    });
  }
})();
</script>
```

Methods:

- get/set base image url
- [LOW] get/set roles available on the diagram : list of string
- get io details -   getIoDetails(id) → { “id“: 0, “role”: “input” }
- set io details -   setIoDetails(id 2, { “id“: 2 “role“: “switch” })   // id in the details object is optional, but must be checked if provided
- getIoDetailsAll() ?? / serialize() → [{ “id“: 2 “role“: “switch” }, { “id“: 1 “role“: “switch” }]
- [LOW] setIoDetailsAll(….)
- addEventListener() : EventTarget
  - “click” => event(.. io state obj, what (“pin”, “arrow”, “label”)
- setActiveIO / setHighlightedIO - implemented with css

Current implementation

The current version of the portal currently supports similar functionality on a very rudimentary level.