Types around the generator and generated tests

This document describes types and concepts used across JavaScript and Python parts of this test framework. Please refer to the JSDoc in `common.sub.js` or docstrings in Python scripts (if any).

Scenario

Properties

- All keys of `test_expansion_schema` in `spec.src.json`, except for `expansion`, `delivery_type`, `delivery_value`, and `source_context_list`. Their values are **string**s specified in `test_expansion_schema`.
- `source_context_list`
- `subresource_policy_deliveries`

Types

- Generator (`spec.src.json`): JSON object
- Generator (Python): `dict`
- Runtime (JS): JSON object
- Runtime (Python): N/A

`PolicyDelivery`

Types

- Generator (`spec.src.json`): JSON object
- Generator (Python): `util.PolicyDelivery`
- Runtime (JS): JSON object (`@typedef PolicyDelivery` in `common.sub.js`)
- Runtime (Python): N/A

`SourceContext`

Subresource requests can be possibly sent from various kinds of fetch client's environment settings objects. For example:

- top-level windows,
- `<iframe>`s, or
- `WorkerGlobalScope`s.

A `SourceContext` object specifies one environment settings object, and an Array of `SourceContext` specifies a possibly nested context, from the outer-most to inner-most environment settings objects.

Note: The top-level document is processed and trimmed by the generator, and is not included in the `sourceContextList` field of `Scenario` in the generated output.

For example, `[{sourceContextType: "srcdoc"}, {sourceContextType: "worker-classic"}]` means that a subresource request is to be sent from a classic dedicated worker created from `<iframe srcdoc>` inside the top-level HTML Document.

Note: A `SourceContext` (or an array of `SourceContext`) is set based on the fetch client's settings object that is used for the subresource fetch, NOT on the module map settings object nor on the inner-most settings object that appears in the test. For example, the `sourceContextList` field of `Scenario` is `[]` (indicating the top-level Window):

- When testing top-level worker script fetch, e.g. `new Worker('worker.js')`. There is `WorkerGlobalScope` created from `worker.js`, but it isn't the fetch client's settings object used for fetching `worker.js` itself.
- When testing worker script imported from the root worker script, e.g. `new Worker('top.js', {type: 'module'})` where `top.js` has `import 'worker.js'`. Again, the fetch client's settings object used for `worker.js` is the top-level Window, not `WorkerGlobalScope` created by `top.js`.

Properties

- `sourceContextType`: A string specifying the kind of the source context to be used. Valid values are the keys of `sourceContextMap` in `common.sub.js`, or `"top"` indicating the top-level Document (`"top"` is valid/used only in the generator).
- `policyDeliveries`: A list of `PolicyDelivery` applied to the source context.

Types

- Generator (`spec.src.json`): JSON object
- Generator (Python): `util.SourceContext`
- Runtime (JS): JSON object (`@typedef SourceContext` in `common.sub.js`)
- Runtime (Python): N/A