ShadowRealm Host Hooks Summary

Contact: legendecas@gmail.com
Status: Draft | **Accepted** | Done
Bug: [v8:11989]

Original Design Doc: Design Doc: ShadowRealm Host hooks
Reviewers: verwaest@, syg@, yukishiino@

Objective

- Allow hosts to hook into the creation of the ShadowRealm globals, as well as how those globals delegate answers up to their creator realm.
- The shadow realm needs to know its parent realm to get at the settings object for web API.
- To be able to leverage the existing security checks for access globals.

Design

There are one host hooks added and one host hook updated.

Hooks to create ShadowRealm

Hosts are expected to expose some useful APIs like EventTarget, atob, TextEncoder, URL, etc. in the ShadowRealm, and this is in the process of web standardization (tracked here https://github.com/tc39/proposal-shadowrealm/issues/331).

Requirements:

1. The hook should allow hosts to utilize the global object template to generate global properties (Blink)
2. The hook should allow hosts to call javascript functions to bootstrap the global properties (Node.js/Deno)

A new host hook is added to delegate the context creation to the host so that hosts are free to choose which Context::New to create the context with whatever they need (from snapshot or from scratch).

```
using HostCreateShadowRealmContextCallback = MaybeLocal<Context>
(*)(Local<Context> initiator_context);

Isolate::SetHostCreateShadowRealmContextCallback(HostCreateShadowRealmConte
xtCallback);
```

It is the host's choice to call Context::New, and it is the host's choice of what APIs should present in the ShadowRealm global.

Since the ShadowRealm JavaScript API doesn't expose any means to exchange JavaScript objects between realms, host-defined APIs like URL, TextEncoder, etc. must not introduce any means to break the limitation.

Modules

In ShadowRealm, it is also possible to import modules like what we have in the main context. The notable thing is that the referrer parameter is empty (according to the spec, step 10) when the dynamic import is initiated by ShadowRealm.prototype.importValue. For runtimes that resolve specifiers based on the referrer, their algorithms to resolve the referenced module may need to adopt the change.

To reflect the change, HostImportModuleDynamicallyWithImportAssertionsCallback is updated with the referrer parameter to be a MaybeLocal<ScriptOrModule>:

```
using HostImportModuleDynamicallyWithImportAssertionsCallback =
    MaybeLocal<Promise> (*)(
      Local<Context> context,
      MaybeLocal<ScriptOrModule> referrer, // This may be empty
      Local<String> specifier,
      Local<FixedArray> import_assertions
    );
```

Security Checks

On ShadowRealm, the same constraints and security checks on dynamic code generation are applied as same as the main context: through isolate hooks. E.g.

- HostEnsureCanCompileStrings
- AccessCheckCallback and FailedAccessCheckCallback.