Web Components Integrity

Caridy Patiño

March, 2020

How to Preserve Web Components Integrity

Web Components provide a certain degree of encapsulation to preserve integrity

1. Closed shadow roots, element internals, etc.
2. Event retargeting
3. Style encapsulation

How to Preserve Web Components Integrity (2)

Need more features for large-scale applications and sharing components across orgs

1. Manage custom element registration at scale: e.g., scoped registries, tag names restrictions, etc.
2. Prevent shared object manipulations: e.g., DOM APIs, globals, component prototypes, etc.
3. Sandboxing shared resources: cookies, local storage, etc.

Example of DOM Virtualization Today: “Locker”

- Developed at Salesforce for web components running on its platform
- Focused on very large scale multi-authors applications (+1K components per page)
- Virtualize DOM APIs for integrity
- Manage communication between multiple JavaScript realms using Proxy Membranes
- Communication is synchronous to match semantics of existing DOM APIs
- Compilation and runtime overhead

Locker implementation strategy

Window
- Document
- Platform System Code
- Platform Base WC

Namespace “foo”
- Sandboxed DOM
- Yellow Components
- Yellow Resources:
  - cookie, registry, etc.

Namespace “bar”
- Sandboxed DOM
- Green Components
- Green Polyfills
- Green Resources:
  - cookie, registry, etc.

Demo

Limitations of using iframes to sandbox

Developers can technically already create a new Realm by creating new same-domain iframe, but there are few impediments to use this as a reliable mechanism:

- We either use the DOM apis of the iframe (which requires to keep the iframe attached), and deal with *unforgeables* that can be used to affect the integrity of the app (e.g.: `window.top`).
- or we can detach the iframe while using the realm associated to it and lose some features that require host behavior (e.g. `import()` statements).

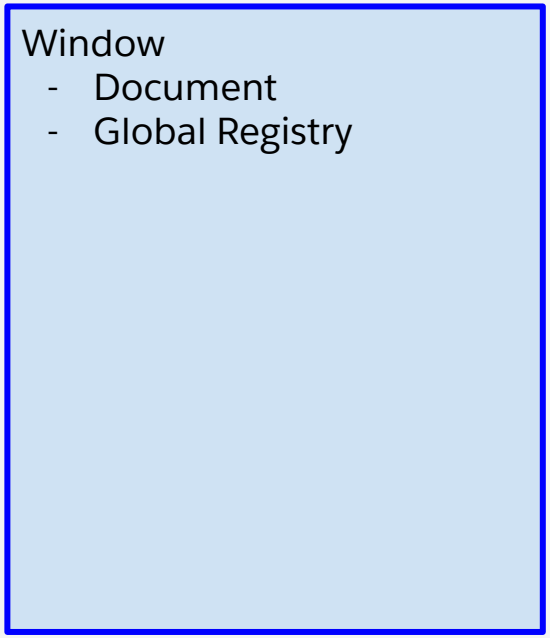
In the future we might be able to create new Realms:

- https://github.com/tc39/proposal-realms (currently at stage 2)

Takeaways

- We should continue developing our integrity preserving story for WC, scoped registry is a good next step.
- Virtualization using async or sync iframes is very far from ideal, Realms proposal seems to help a lot to do user-land virtualization for integrity.

How Locker will use scoped registries



Namespace “foo”
- Create yellow scoped registry
- Distort ce.define() to add elements to yellow registry
- Disort E.p.attachShadow to attach the yellow registry

Namespace “bar”
- Create green scoped registry
- Distort ce.define() to add elements to green registry
- Disort E.p.attachShadow to attach the green registry