URLs for Bundle Contents

Jeffrey Yasskin

WPACK — IETF 108

2020-07-31

WICG/webpackage/explainers/bundle-urls-and-origins.md

URL design

David Hawgood / Bike shed, Kew Gardens

Roberto Uderio / Atomic power plant, Cofrentes, Spain

Bundle Contents (simplified)

```
fetch(bundle-URL) =>
    claimed-URL-1 => Resource 1
    claimed-URL-2 => Resource 2
    claimed-URL-3 => Resource 3
    ⋮
```

How do we name things inside bundles?

e.g. `<img src=...>` or `<a href=...>`

draft-soilandreyes-arcp or draft-shur-pack-uri-scheme?

Fragments!

- https://site.example/page/subresources.wbn
  #url=https://site.example/images/tree.jpg
- https://archive.example/https/news.example/story.wbn
  #url=https://news.example/story.html
- https://el.paquete.example/recommended.wbn
  #url=https://camera.example/edit.html

What about origins?

Matters when using bundle contents as HTML pages.

What storage partition ("shelf") does `https://archive.example/2020-04-01.wbn#url=https://camera.example/edit.html` use?

- Same as https://archive.example/login?
- Same as https://archive.example/**2019-01-01**.wbn#url=https://camera.example/edit.html?
- Same as https://archive.example/2020-04-01.wbn#url=https://**bonk.example**/uses_camera_storage_keys.html?
- Same as https://archive.example/2020-04-01.wbn#url=https://**evil.example**/attack_camera.html?

Origin should depend on:

- The fact that it's a bundle.
- The origin of the bundle's URL.
- The **path** of the bundle's URL.
- The origin of the contained resource. (*)

(*) Origin might not depend on subresource origin

Martin Thomson suggested nesting bundles instead, and defining a way to address relative to the outer bundle.

For archived cross-site links, this requires rewriting the resources, but that's a common practice today.

These slides do not pursue this variant.

New URL scheme (1)

a. **pkg+https://archive.example/2020-04-01.wbn**#url=**https://camera.example/**edit.html
   - No worse to read than other URLs.
   - Makes origin depend on parts of the fragment. How scary is that?

b. **package:https://archive.example/2020-04-01.wbn$https://camera.example**/edit.html
   - Only slightly worse to read.
   - Still hard to see where origin stops

New URL scheme (2)

https://archive.example/2020-04-01.wbn#url=https://camera.example/edit.html redirects to **arcp:ni,hash;*hash(fetch(https://archive.example/2020-04-01.wbn))*$https://camera.example**/edit.html

- Based on draft-soilandreyes-arcp.
- Storage stays consistent across transfer, but not update.
- Completely unreadable authority.
- I may be misinterpreting some ideas from the wpack@ list.

New URL scheme (3)

a. **package:https:,,archive.example,2020-04-01.wbn**$**https:,,camera.example**/edit.html
    - Ew.
    - The origin is everything before the "/", like for `https:`.

b. **package:https%3a%2f%2farchive.example%2f2020-04-01.wbn**$**https%3a%2f%2fcamera.example**/edit.html
    - Ew. Ew.
    - The origin is everything before the "/", like for `https:`.
    - No special encoding rules.

Scheme should apply to more than our bundles

- ZIP
- .tar.gz
- PDF
- BagIt (RFC 8493)?

Probably based on media type.

Use `file:///path` or just `/path` as the claimed URL for containers of paths.

Other behavior

URL rendering

Governed by https://url.spec.whatwg.org/#url-rendering.

Browsers emphasize the registrable domain of a URL.

Should also emphasize the bundle portion of a package: URL.

Programmatic exposure

- Within a single bundle, only expose claimed URLs.
- Outside of a subresource's home bundle,
  expose its package: URL instead.
- `fetch`("package:bundle-url$claimed-url") causes a
  `fetch`(bundle-url) + a lookup of claimed-url in the bundle.

`Referer` and `Origin` headers

1. Apply the bundle's referrer policy to the Bundle URL.
2. If that retains the path, apply the subresource's referrer policy to the claimed URL.
3. Merge the results back into a package: URL.
4. Correct for being in the same bundle on read.

`Referer` examples

package:https:,,foo.example,package1.wbn$https:,,bar.example/page.html navigates to or has a subresource of https://foo.example/image.jpg

		Subresource's Referrer Policy	
		strict-origin-when-cross-origin	strict-origin
Bundle's Referrer Policy	strict-origin-when-cross-origin	package:https:,,foo.example,package1.wbn$https:,,bar.example	
	strict-origin	package:https:,,foo.example	

`Referer` examples

package:file:,,,package2.wbn$https:,,foo.example/page.html
navigates to or has a subresource of
package:file:,,,package2.wbn$https:,,foo.example/page2.html

		Subresource's Referrer Policy	
		strict-origin-when-cross-origin	strict-origin
Bundle's Referrer Policy	strict-origin-when-cross-origin	package:file:,,,package2.wbn$ https:,,foo.example/page.html	package:file:,,,package2.wbn$ https:,,foo.example
	strict-origin		

Discussion

(see next slide)

Questions

- Which **semantics**? (slides 5–7)
- Which overall **syntax**? (slides 8–10)
- What details? Depending on overall syntax:
    - `package:` or `package://`?
    - What fragment params?
    - How are characters encoded?
    - For ZIP, etc., use a claimed URL of `file:///path` or `/path`?