Pickling for Clipboard API

huangdarwin@chromium.org
Darwin Huang (he/him)
Agenda

- Current Spec Status
- Problems
- Pickling design
- Questions?
Current Spec Status

- **DataTransfer**
  - Synchronous. Support for text, html, and images.
  - Blink and Webkit both implement an unstandardized and minimally documented custom format implementation (based on Pickling)...

- **Clipboard API:**
  - Asynchronous, promise-based API
  - Early 2019: Specified navigator.clipboard.{read,write}()
    - new ClipboardItem object instead of the previous synchronous DataTransfer concept.
  - Blink and Webkit both implement support for text, html, and images.
  - No support for custom formats
Problems - niche formats

CAD design web apps (SketchUp)
Niche, custom formats (CAD formats)

Image editor web apps (Photopea)
Niche, custom formats (images with layers)
Problems - proprietary formats

Document editors (Google Docs)
Proprietary formats (document formats)
Problems - metadata

UX design web apps (Figma)
Formats untouched by sanitization (Images with all metadata preserved)
Problems - interop with native

Figma (or other sites) ...with native app counterparts

Native apps interested in interop (Figma)
Interop with custom formats
Problems - interop with native

Office Online (or other sites) ...with native app counterparts

Native apps interested in interop (Microsoft Word)

Interop with custom formats
Pickling Design

● Provide web applications with access to custom formats.
  ○ Web-originated, so no interop (or RCE concerns) with legacy native apps.
  ○ Unsanitized, because formats are custom.
  ○ Native apps may opt in to using custom formats.

● Still recommend existing sanitized APIs for general use, for wide compatibility and safety.

● Context: Last year, Pickling was suggested when discussing raw clipboard.
const image = await fetch('myImage.png');
const text = new Blob(['this is an image'], {type: 'text/plain'});
const clipboard_item = new ClipboardItem({"text/plain": text, 'image/png': image});
await navigator.clipboard.write([clipboard_item]);
// Pickling write example.
// This format 'text/plain' is recognized by the Clipboard API, so will be written as usual.
const text = new Blob(['text'], {type: 'text/plain'});
// This format 'text/custom' is not sanitized by the Clipboard API. It will be pickled if the format is specified in the {direct: []} formats list.
const customText = new Blob(['<custom_markup>pickled_text</custom_markup>'], {type: 'text/custom'});

// Clipboard format ordering: Pickled formats will be written before sanitized formats by the browser, since they're more "custom" and likely more targeted towards this use case.
const clipboardItem = new ClipboardItem({
  'text/plain': text,  /* Sanitized format. */
  'text/custom': customText /* Pickled format. This new format will be accepted and written without rejection, as long as the new direct list contains this format. */
},
{direct: ['text/custom']} /* This new list specifies the pickled format 'text/custom'. */
);
navigator.clipboard.write([clipboard_item]);
const clipboardItems = await navigator.clipboard.read();
const clipboardItem = clipboardItems[0];
const text = await clipboardItem.getType('text/plain');
const image = await clipboardItem.getType('image/png');
// Pickling read example. ClipboardItems returned by clipboard.read() may contain pickled formats.
const clipboardItems = await navigator.clipboard.read({direct: ['text/custom']}) /* This new list specifies the pickled format 'text/custom' for all read ClipboardItems. */;
const clipboardItem = clipboardItems[0];

const textBlob = await clipboardItem.getType('text/plain');  // This format reads as a pickled format, only if it is included in the direct format list.
const customTextBlob = await clipboardItem.getType('text/custom');
Basic Protections

- Secure Context
- Active Frame
- Permission
- Consistent with many other new, “powerful” APIs
- User gesture, and optionally feature policy (recently renamed to permissions policy).
Proposed mapping

- Open to alternative mappings.
- Example: “custom/foo-format”
  - Windows:
    - Capital words separated by spaces.
    - Web Custom Foo-format
  - MacOS:
    - UTI reverse-DNS naming
      - com.web.custom.foo-format.
  - All other platforms (Linux, Android, ChromeOS, etc):
    - MIME types
      - application/web;type=”custom/foo-format”
Why map formats to native

- Sites should consistently use MIME types across platforms, to avoid platform-dependent code.
- Browsers should be able to interop with one another.
- Use consistent native formats between browsers.
- Different per platform, to follow platform conventions.
- Native applications may also opt in by implementing support for these formats.
Privacy Concerns (navigator.clipboard.read)

- Some sites may provide excess metadata or PII on the clipboard. This will be less transparent with pickled formats, which aren’t pasted everywhere.
- Ex. Some image formats provide metadata with GPS and camera information.
Security Concerns (navigator.clipboard.write)

- Previous related raw clipboard proposal’s arbitrary code execution concern alleviated.
- Some sites may place scripts on the clipboard. This will be less transparent with pickled formats, which aren’t pasted everywhere.
- Ex. Pickled SVG or HTML might contain scripts.
Other

- Sorry, no public signals yet...
Questions?

- Open-ended overall, but please separate some concerns, such as read/privacy and write/security, into separate topics.
- Please also provide feedback on API shape.