Web USB and Filesystem APIs

A brief overview of how they work and where they are used

This presentation is an interactive demo, running in a web browser

Overview

- What are these APIs are trying to solve
- Interacting with the file system and USB devices in a browser
- Real world demo

There have always been limitation of the Web

- Sandboxed
- Browsers were not gaurenteed to be up to date

Electron

- Provides access to Node libraries
- Let us access outside the sandbox
- Guarantee an environment your code executes in

It's 2022 - Who needs electron?

- Javascript is more standardised than ever
- Browsers are evergreen
- APIs which let us break out of the sandbox have been made available

Accessing files in the browser (the old way)

```
<input
  type="file"
  multiple
  onChange={(event) => {
    ...
  }}
/>
```

Web File System Access API

https://developer.mozilla.org/en-US/docs/Web/API/File_System_Access_API

Currently supported on most Chromium browsers on Windows, macOS, Chrome OS, and Linux

Working Draft - 24 September 2021

- Read files from a folder
- Write files into a folder
- Create folders in a folder
- Update the contents of a file
- Remove files from a folder

Seems problematic?

Select folder

Accessing only comes through a tree

```
nst rootHandle = await window.showDirectoryPicker({ id: "my-test" }

r await (const entry of rootHandle.values()) {
const entryHandle = await entry.getHandle();
await rootHandle.removeEntry(entry.name);
```

(Same thing in node)

```
import fs from 'fs/promises';

const files = await fs.readDir("directoryName");

for (const file of files) {
  await fs.unlink('directoryName/' + file)
}
```

In the wild

https://vscode.dev

https://excalidraw.com

Take aways

- Access is explicit (requires user interaction)
- Operations are completed against a tree like api
- Only available on chrome (for now)

Web USB

https://developer.mozilla.org/en-US/docs/Web/API/USB

Supported in all chromium based browsers for a few years (including Android!)

- Request access to a USB device
- Access a list of previously accessed USB devices
- Open a connection to a USB device
- Read Write arbitrary USB packets

Connecting to a device

```
const device = await navigator.usb.requestDevice({ filters: [] });
```

Select device

Currently connected devices (remembers existing devices)

```
const device = await navigator.usb.getDevices();
```

In the wild

https://flash.android.com

https://buddy.edgetx.org/#/flash?version=v2.6.0&target=nv14

Take aways

- Access is explicit (requires user interaction)
- Devices are "paired" with a session after selection
- Some devices will still require drivers for the OS to interact with the device (windows)

Conclusion

“Any application that can be written in JavaScript, will eventually be written in JavaScript.” - Jeff Atwood (2007)

Further reading

- https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API

Useful resources

Types

- https://www.npmjs.com/package/@types/wicg-file-system-access
- https://www.npmjs.com/package/@types/w3c-web-usb

Libraries

- https://github.com/node-usb/node-usb
- https://github.com/jimmywarting/native-file-system-adapter
- https://github.com/flipperdevices/webdfu