(Public) App Home intro

owner:phillis@chromium.org

Product Summary

Problem statement/Situational Overview

Currently, the only home for web apps in Chrome is chrome://apps. It serves as a launcher and a management surface, but it is outdated, available only on Desktop, and affected by the pending Chrome Apps deprecation.

In this situation Chrome as a web app platform relies on underlying OSs to provide solutions for developer and user needs - from new app discovery, to app launching and app management.

This is a three fold issue:

1. **Strategy** - it creates a gap that might be filled by a competitor. Consequently, that puts the competitor in a position to define what the web experience is - and that experience may or may not be based on the best practices of the open web (e.g. embrace and extend).
2. **Experience** - there is no coherent web app experience and a consistent object model we can count on to address current and future user journey moments. Different OSs implement different levels of web apps integration which can leave key moments in user journeys either unaddressed or partially addressed. For example, on Windows and macOS, if the user deletes a shortcut on desktop there is no way to uninstall the app (or recreate the shortcut) from the OS.
3. **Opportunity** - we are not playing to our strengths and leaning into what Chrome and the web can uniquely do. We can do more, innovate faster without OS limitations, and create a virtuous cycle that will close the web ecosystem loop.

For Web Platform to succeed, we need to build a coherent web app experience that works great across OSs (including integrations with OSs where possible), and utilizes the best of Chrome and the best of the web (by doing things we can do that native can't). This will help make users love the web and Chrome by making it easier to discover, launch and manage their apps. And this will make it easier for developers to succeed by helping with key challenges such as user acquisition and re-engagement. These differentiated capabilities for discovery and re-engagement can then be an additional incentive for them to invest in the web.

App launching and app management are the highest priority foundational features that cover the most critical user journeys, and where chrome://apps plays a critical role. For example, with chrome://apps deprecated, there would be no place in Chrome where the user can find all the web apps installed from Chrome. MVP that starts simple and meets those most common user

journeys can act as a foundation for more complex app discovery and re-engagement journeys in the future.

Target user

- Web apps users
- Web platform developers / Partners

Goals

- Build a home for web apps in Chrome, a consistent object metaphor we can always count on to address current and future user journey moments.
    - CUJ #1: I want a place to find all the web apps I have installed on the device.
    - CUJ #2: I want a place to find all the web apps that are synced for my profile.
- Make users love the web and Chrome by making it easier to launch and manage their web apps.
    - CUJ #3: I want to remove/uninstall a web app.
    - CUJ #4: I want to quickly launch a web app I use frequently.
    - CUJ #5: I want to manage settings for web apps I have installed.
    - CUJ #6: I want to create a Desktop OS shortcut for a web app.
    - CUJ #7: I want to navigate to a frequently visited location in a web app.
- Make it easier for developers/partners to succeed and meet their business objectives by helping with re-engagement/retention. These unique re-engagement capabilities can then be a differentiator and an additional incentive for them to invest in the web.

Non-goals

- Solving new app discovery (for users) and user acquisition (for developers) by making it easier to find and install new apps is out of scope for this MVP and will be explored separately in Phase 2.

Solution

The proposal is to use a two pronged approach:

1. **App Home** - a canonical surface that provides access to all the web apps the user has installed (CUJ #1, CUJ #2) and implements easy ways to manage them (CUJ #3, CUJ #5, CUJ #6). As home for apps, this surface will become the foundational surface where additional features (like new app discovery) could be implemented in future iterations.

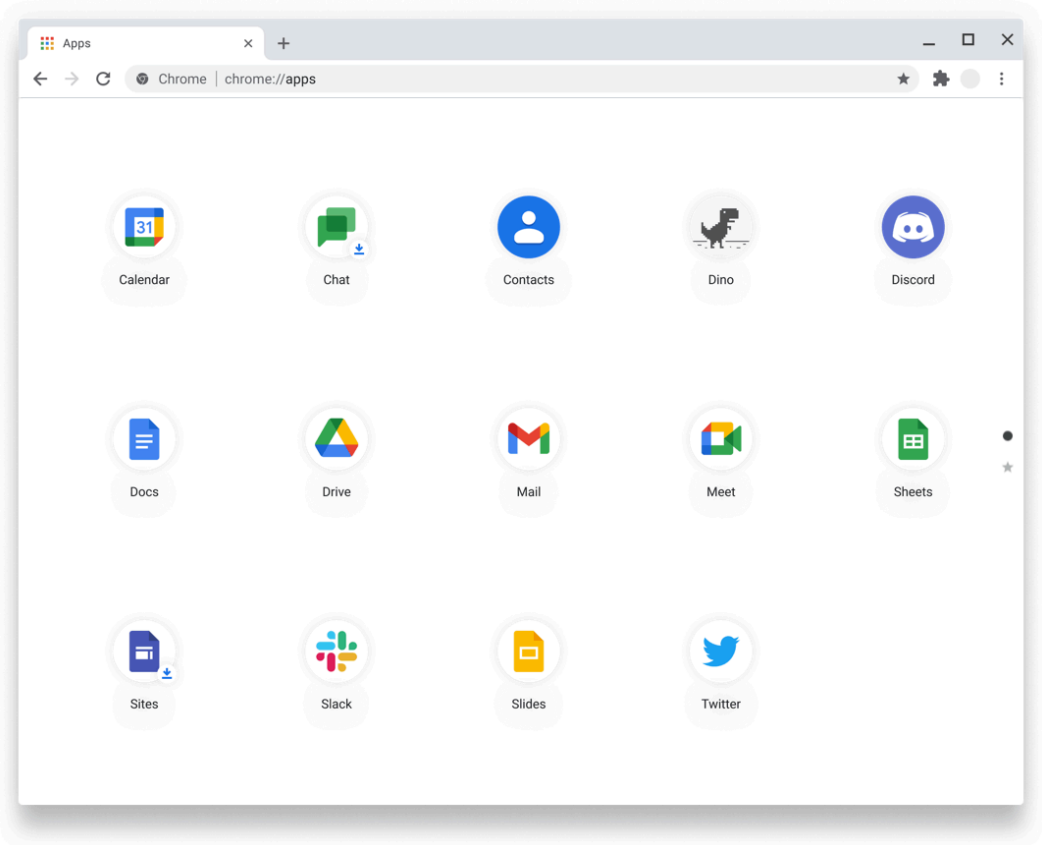
2. **App Home Module** - a module that will make it easy to quickly launch frequently used apps (CUJ #4), and serve as a new entry point to the full App Home (containing all the

web apps). This module will be surfaced along natural user journey points in the browser app.

For App Home, pending Chrome Apps deprecation is an opportunity to:

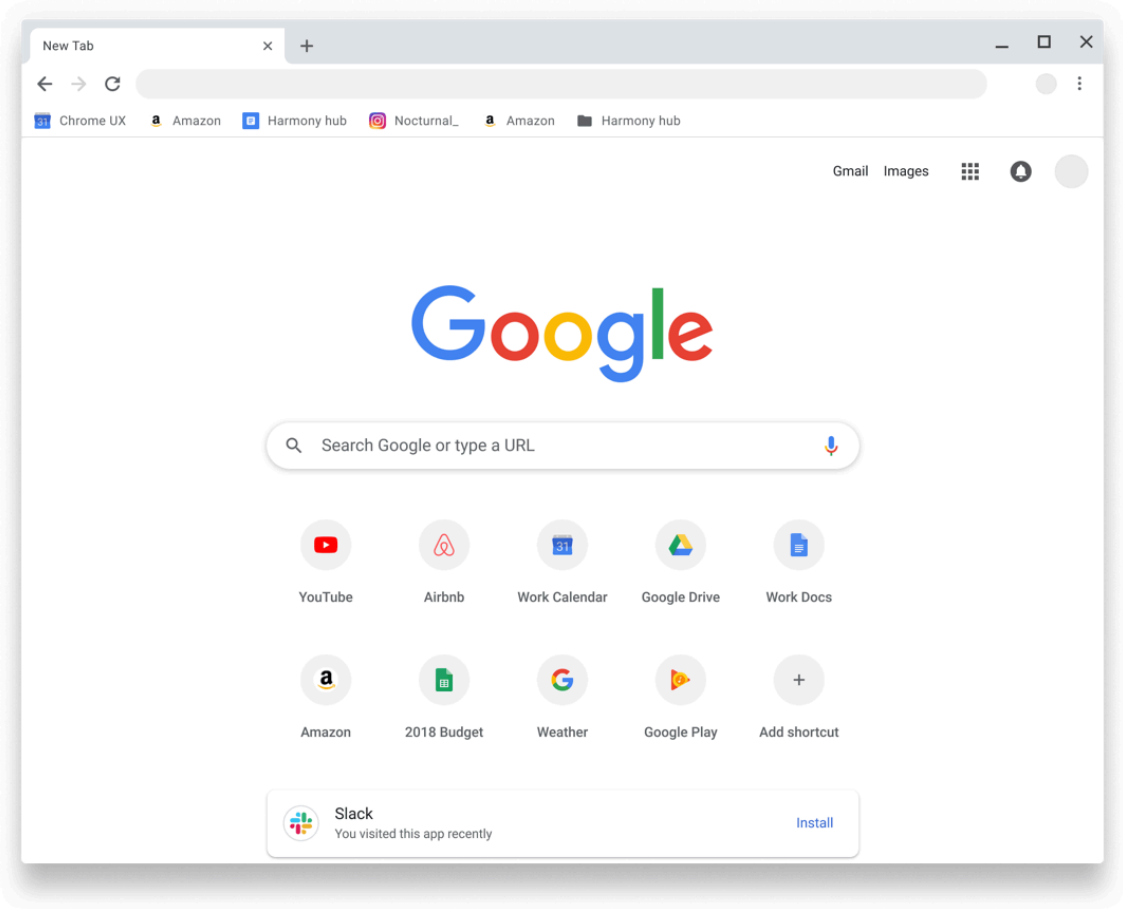
1. Rebuild chrome://apps as App Home for **Windows/macOS/Linux**. Chrome OS implementation can be explored later as Chrome OS implements a higher degree of web apps integration and chrome://apps doesn't exist there.

2. Extend the surface to **Clank** (as a separate implementation). The proposal also effectively opens a path to unlocking web apps on iOS - this could be a big opportunity and will be explored in the future.



Vision - canonical App Home surface with app launching and app management (MVP) + new app discovery (to be scoped separately in Phase 2)

For App Home Module, we propose an experiment based on NTP Modules (Cards). This is a natural fit given that NTP Modules aim to make the NTP a more compelling launchpad to common user journeys.

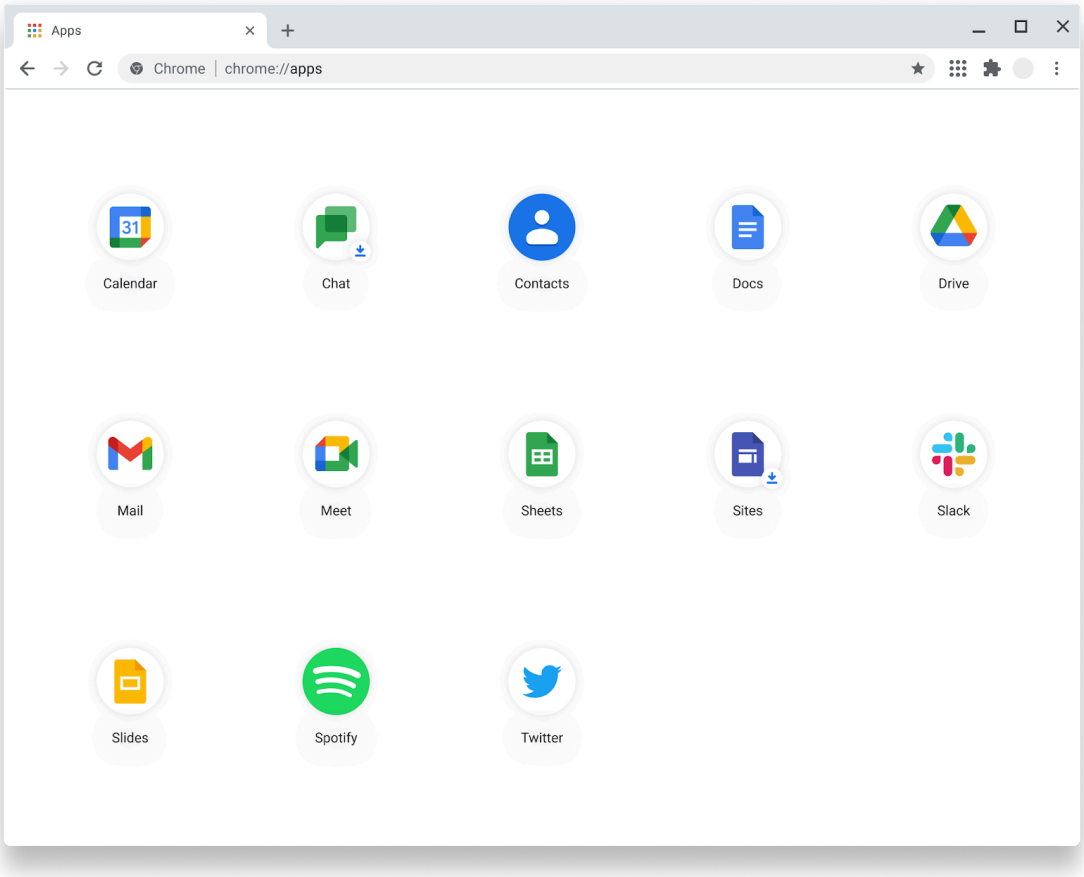


App Home Module on NTP

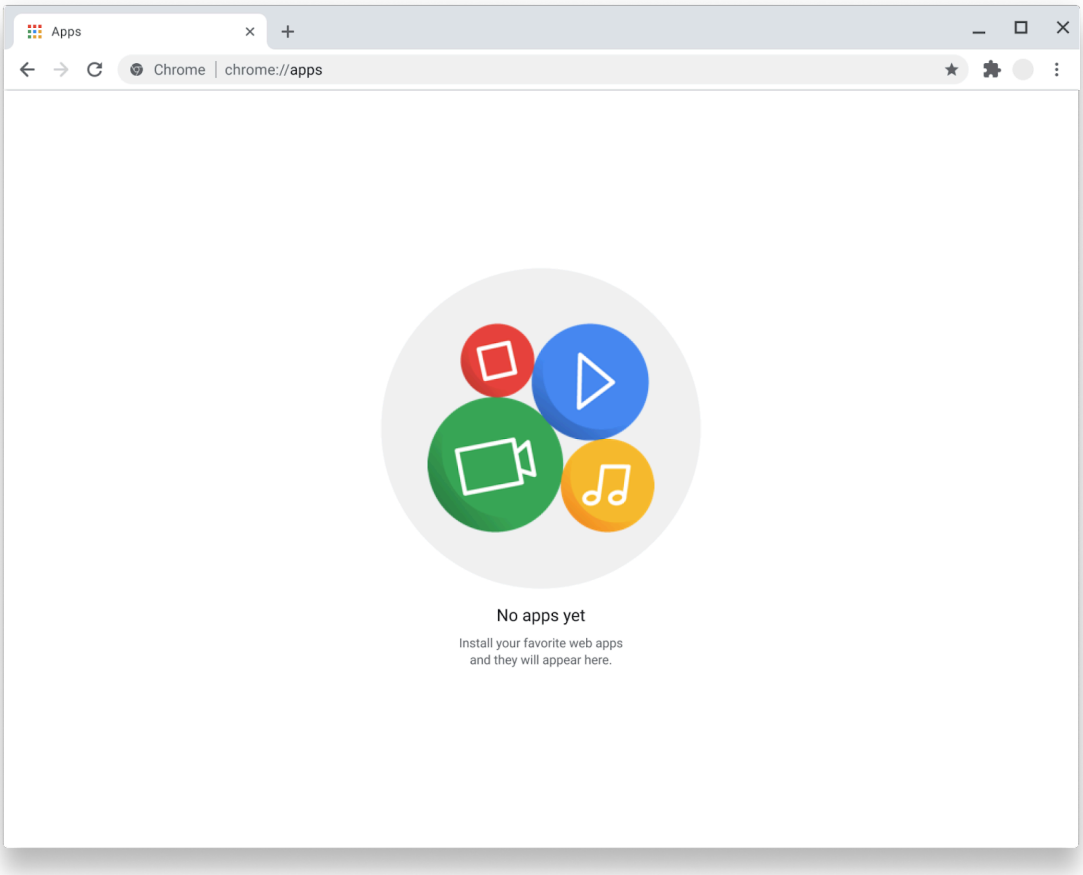
UX detail

App Home MVP scoping

App Home surface



When no apps are available, an empty app page should be displayed.



UX/UI difference with existing chrome://apps:

- App icon will get a new white circle background.
- Not locally installed apps won't be grayed out, but will have a download icon in the bottom-right corner instead.
- Pagination will be done vertically instead.
- We'll **stop supporting** the drag-and-drop install feature.
- More responsive and sophisticated layout styling/
- More management actions.

App ordering:
Ordering is done the same as the existing UX and it's very similar to mobile native apps home screen experience.

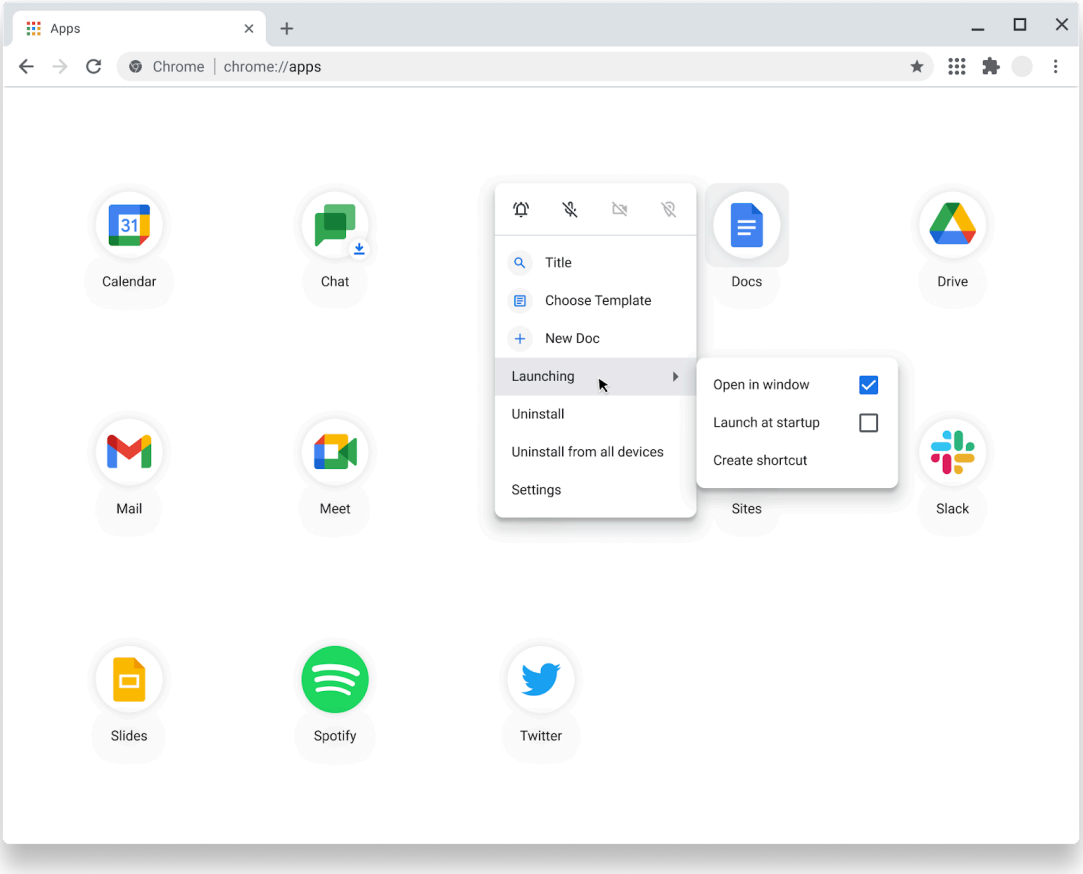
- New apps are put to the end of the apps in the first page that have space.
- Users can drag to reorder the app.
- When users uninstall an app, apps from the next page won't be collapsed to this page, they will stay where they are.

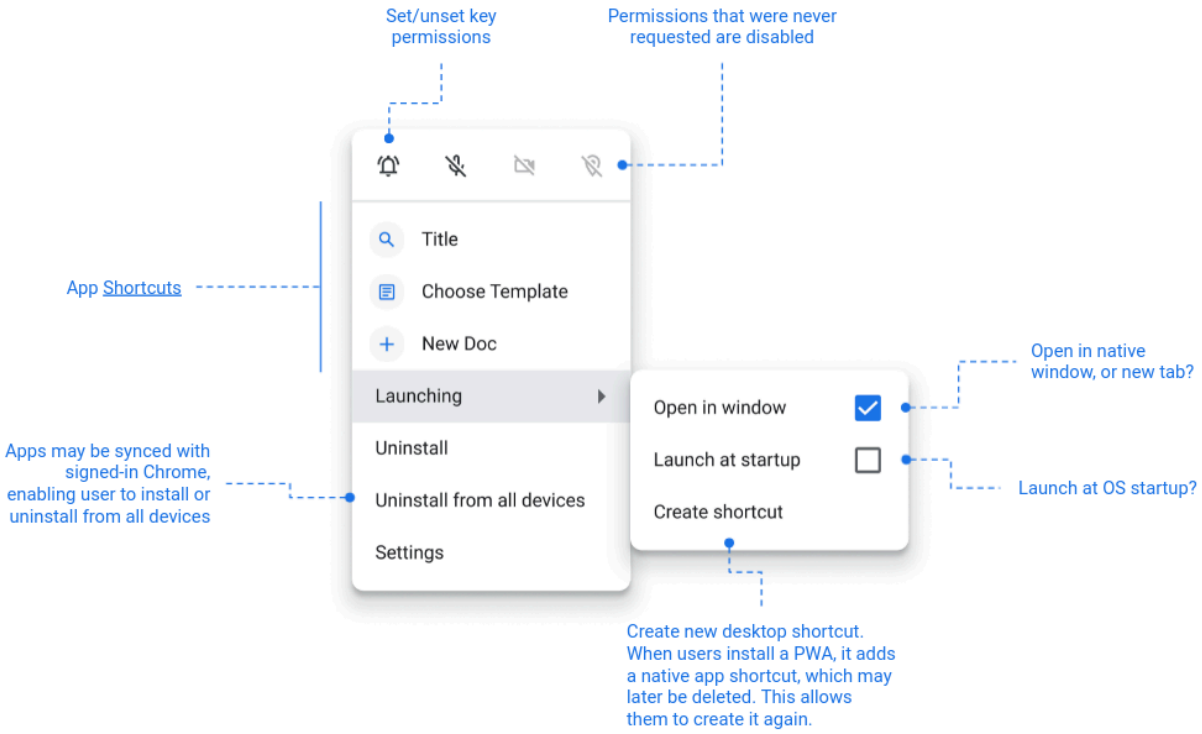
Apps that show up in app home will include PWAs and extension-based chrome apps that `ShouldDisplayInNewTabPage`.

These extension based apps are legacy chrome apps that will be deprecated, but we will keep supporting them until they are deprecated. They include the following categories of apps if they are not blocked by policy:

- Legacy packaged app
- Hosted app
- Platform app

Management actions





Management actions menu shows up when the user right clicks the app icon.

1. Permissions: Grant or revoke access to 4 pre-set site permissions: notifications, mic, camera, location.
2. Shortcut Items: Any shortcuts defined by the app.
3. Launch configuration.
4. Uninstall: uninstall the web app on this device or all devices.
5. Link to site settings page.

Technologies

The chrome://apps page will be built using WebUI framework that supports handling actions and data changes in the html page.

Message handling

There are two ways for the html-based frontend to communicate with c++ code.

Mojo WebUI handlers

This is a new way developed by the WebUI team. Not much documentation exists for this but there are lots of WebUI pages using it, e.g. chrome://downloads page.

Pros:

- Using Mojo is more secure and auditable.

    - Type annotations generated from your Mojo interface definition.
    - Message validation.
    - More structured data i.e. unions, enums, structs, interfaces, etc. in C++ and JS.
    - Interface is more easily auditable by security.
- It's the direction the WebUI is moving forward with, so it will be better maintained.

Cons:

- Need to port existing code to this.
- Less documentation
- Less examples to learn from / more overhead to ramp up on this approach

Alternative - WebUIMessageHandlers + chrome.send/cr.sendWithPromise

This is the old way to register message handlers and is used by the current chrome://apps page. Frontend will call chrome.send(message_name, data) in Javascript which is hooked with the specific handler registered to WebUI. Data is passed as nested dictionary to JS callback.
This is still supported and maintained by the webUI framework and documented as the recommended way in WebUI explainer.
Pros:

- Used by the current chrome://apps page so that we can re-use existing code.
- Well documented.

Cons:

- Not as secure as using the Mojo WebUI handlers.

Since this approach is still supported, there is no strong reason to re-write the backend. So we can go with still using the WebUIMessageHandlers.

Changes needed

- Update `CreateWebAppInfo` to include shortcut menu info and icons and site settings permission status.
- Update `CreateExtensionInfo` to include site settings permission status.
- RegisterMessageCallback for `UninstallAppLocally`
- RegisterMessageCallback for `SetNotificationsPermission`
- RegisterMessageCallback for `SetCameraPermission`
- RegisterMessageCallback for `SetMicrophonePermission`
- RegisterMessageCallback for `SetLocationPermission`

-

Frontend

Existing chrome://apps page uses a custom pre-WebComponents framework to define UI components (see cr.ui.define, decorate etc). The WebUI team is moving away from it and highly recommends us to use **JS modules + Polymer + Typescript**. See style guide.

Docs:
https://source.chromium.org/chromium/chromium/src/+/main:docs/webui_in_chrome.md
grit instructions

Components

- Apps laid out vertically, with pagination. Apps can be dragged and reordered.
- Pagination dots
- Actions menu
- Empty page