

Adding `display_override` Manifest Property (draft)

This Document is Public

Authors: dmurph@chromium.org, mjackson@microsoft.com, <your_email_here@>

Contributors/Reviewers: <your_email_here@>

July 2020

One-page overview

Background

For a WebApp to have its own window (for a WebApp to be a WebApp) it needs to request a supported display mode, usually `standalone` or the newly created `minimal-ui`, which provides the WebApp with its own window.

Summary

The current way to set the display mode for a website (e.g. in a standalone window, a standalone window with browser controls, just a regular browser tab, etc) is inflexible for developers & prevents adding new display modes without significant problems. See the problem summary in [this explainer](#) to learn more.

This new field allows developers to specify their own display fallback ordering, which solves the most major 2 problems of the current field - inability to change the fallback behavior, and inability to add new display modes without forcing fallbacks on developers that they don't want.

Platforms

Mac, Windows, Linux, Chrome OS, Android.

Team

pwa-dev@chromium.org

Bug

<https://crbug.com/1092667>, Intent to Prototype [Email](#)

Code affected

WebAppProvider (chrome/browser/web_applications), and WebManifest (third_party/blink/renderer/modules/manifest)

Design

We can probably assume a [BMO](#) launch here, so we don't have to implement an extensions version.

Adding the `display_override` to the manifest

The first step is adding the `display_override` property to the manifest. This would involve adding:

- Adding the field to [manifest.mojom](#)
 - Probably can re-use the enum we use for `display`.
- Parsing of the field in [manifest_parser.cc](#)
 - Ignoring unknown entries
 - Duplicate entries are preserved.

The manifest should store the ordered list of display modes (without unsupported/unknown ones).

Completed: 2274170: dpwas: Support display_override in webapp manifest | <https://chromium-review.googlesource.com/c/chromium/src/+2274170>

Integration with WebAppsProvider

- Storing the field in [web_app.h](#) (and probably making the virtual function for `app_registrar.h` to get it)
- Adding the field to the `web_app_database.cc` (populating into `WebAppProto`)
- Populating the field from the manifest (following from [here](#)) and the database (following back from [here](#) and [here](#)).

Completed: 2299192: dpwas: display_override integration with WebAppsProvider | <https://chromium-review.googlesource.com/c/chromium/src/+2299192>

Using it to compute the 'effective' display mode

Updating the function [here](#) to take the `display_override` field into consideration.

Completed: 2314977: dpwas: compute effective display mode with display_override | <https://chromium-review.googlesource.com/c/chromium/src/+2314977>

Using it to compute if the manifest is valid

We should update the logic here:

https://source.chromium.org/chromium/chromium/src/+master:chrome/browser/installable_manager.cc;l=652;drc=f56b5e1e538025dbd4f1a99ab53e09e2152501ac?q=installable_manager.cc&ss=chromium

To also use `display_override`.

We basically want to make sure that we don't end up with `kBrowser`. If the resolution of the `display_override` is 'browser', then that should return false with the given error.

Then we should have a test in `installable_manager_unittest.cc` or `installable_manager_browser_test.cc` to verify that a webapp with a manifest that resolves in 'browser' should NOT be installable.

Completed: 2333541: dpwas: update installability to account for `display_override` | <https://chromium-review.googlesource.com/c/chromium/src/+2333541>

Supporting the 'update' of this value in the manifest

The [manifest update task](#) is used to update web apps to a new manifest (so a website can update the configuration of their web app). The `display` property is updated here, so the `display_override` property will also need to be updated. Tests should be added [here](#) (and possibly in the `web_app_browser_test.cc`, as an integration test)

Completed: 2330338: dpwas: support updating `display_override` in the manifest | <https://chromium-review.googlesource.com/c/chromium/src/+2330338>

Update DevTools to show the installability check failure

Update `getInstallabilityErrorMessage`s in `third_party/devtools-frontend/src/front_end/resources/AppManifestView.js` to account for the new installability string failure.

Completed: 2355440: Update installability to account for `display_override` | <https://chromium-review.googlesource.com/c/devtools/devtools-frontend/+2355440>

Update WebAppBrowserController to account for display_override

Update `WebAppBrowserController::HasMinimalUiButtons` which uses the display mode to determine if minimal ui should be shown.

chrome/browser/ui/web_applications/web_app_browser_controller.cc
`WebAppBrowserController::HasMinimalUiButtons`

Completed: 2344525: dpwas: Update WebAppBrowserController to account for display_override | <https://chromium-review.googlesource.com/c/chromium/src/+2344525>

Update RecordAppWindowLaunch to account for display_override

This function logs the display value used to launch in telemetry.

chrome/browser/ui/web_applications/web_app_launch_manager.cc

RecordAppWindowLaunch

Complete: 2354531: dpwas: Update RecordAppWindowLaunch to get winning display mode for telemetry | <https://chromium-review.googlesource.com/c/chromium/src/+2354531>

Update AppBannerManagerDesktop to account for display_override

This function confirms that the user preference is set such that it launches into a new window, and it uses GetAppDisplayMode to determine the mode to launch in. We need to update this to read from the manifest.

chrome/browser/banners/app_banner_manager_desktop.cc

AppBannerManagerDesktop::OnWebAppInstalled

Completed: 2354665: pwlas: Update AppBannerManagerDesktop to get winning display mode for telemetry |

<https://chromium-review.googlesource.com/c/chromium/src/+2354665>

Not updated references to GetAppDisplayMode:

There are two references that are not being updated.

- 1) chrome/browser/web_applications/web_app_migration_manager.cc

WebAppMigrationManager::MigrateBookmarkApp

This is concerned with migrating from Bookmark Apps to BMO - and since this feature only exists in BMO - no further code changes are necessary.

- 2) chrome/browser/extensions/api/management/chrome_management_api_delegate.cc

CreateExtensionInfoFromWebApp

You can generate a web app through <chrome.management.generateAppForLink>

extension API. It uses the display_mode to compute the launchType. I think the correct change is to update the call from GetAppDisplayMode to

GetEffectiveDisplayModeFromManifest, which I've done here:

2359374: dpwas: Update CreateExtensionInfoFromWebApp to get winning display mode |
<https://chromium-review.googlesource.com/c/chromium/src/+/2359374>

However, it doesn't seem like this is a configurable value and its always set to 'kBrowser'. [chrome_management_api_delegate.cc - Chromium Code Search](#)

Security Implications

There are no security implications here -- all of the display modes that are going to be triggered by this new feature as already available, it just adds new logic around choosing this mode.

Testing plan

The following tests should be created:

- Test the manifest is read correctly, and incorrect/unsupported values are ignored
- Test that the web app registrar stores the correct values
- Test that the web app database stores & retrieves the correct values
- Test an actual webapp w/ a manifest that contains this is treated correctly / launches w/ the correct display mode.

Followup work

If the specification changes, then the implementation needs to change.

One change that could happen is a [name change and behavior change](#), but that shouldn't change much about this work & can be done at any time.