# launch_handler in manifest of PWA

The

`launch_handler` member in a Progressive Web App (PWA) manifest is part of the **Launch Handler API**, which allows you to control how the PWA is launched—specifically, whether it opens in a new window or reuses an existing one, and how it handles the target URL.

It is an experimental feature and may not be supported in all browsers, so the `window.launchQueue` API must be used in conjunction for robust handling.

Usage

Add the `launch_handler` member to your `manifest.json` file with a `client_mode` subfield:

json

```
{
  "name": "My PWA App",
  "start_url": "/",
  "display": "standalone",
  "launch_handler": {
    "client_mode": "focus-existing"
  }
}
```

`client_mode` Values

The `client_mode` field determines the launch behavior. The available values are:

- **`auto`**: The default behavior. The user agent (browser) decides the best context for the platform (e.g., `navigate-existing` on mobile, `navigate-new` on desktop).
- **`focus-existing`**: If an app instance is already running, it is brought into focus. The target URL is made available to the app via `window.launchQueue` but the navigation does not

happen automatically, allowing you to handle it with custom JavaScript.
- **`navigate-existing`**: If an app instance is running, it is brought into focus and navigated to the launch target URL. The URL is still available via `window.launchQueue` for additional handling.
- **`navigate-new`**: The PWA always opens in a new web app window (or a new browsing context within the app) to load the target URL.

Custom Handling with JavaScript

To handle the launch parameters within your PWA, especially when using `focus-existing` or handling files via the `file_handlers` API, you use the `window.launchQueue.setConsumer()` method in your application's JavaScript code:

javascript

```
if ('launchQueue' in window) {
  window.launchQueue.setConsumer(launchParams => {
    // Check if there's a target URL or files to handle
    if (launchParams.targetURL) {
      const url = new URL(launchParams.targetURL);
      // Implement custom routing or logic based on the URL
      console.log('Launched with URL:', url.pathname);
    }
    if (launchParams.files) {
      // Handle file system handles
      for (const fileHandle of launchParams.files) {
        console.log('Handling file:', fileHandle.name);
      }
    }
  });
}
```

This allows developers to create a more integrated, native-app-like experience by managing windows and navigation behavior according to user expectations

---

Revision #1
Created 14 December 2025 04:49:13 by AI Channel
Updated 14 December 2025 04:50:22 by AI Channel