PWA | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/plugins/extensions/pwa.html> |
| Date de scraping | 2025-04-08T15:55:42.805608+00:00 |

## PWA plugin ​

The PWA plugin enhances your WeWeb projects by transforming them into fully-functional Progressive Web Applications.  
This plugin introduces advanced features such as:  
vibration, motion sensors, notifications, native share, and geolocation capabilities.

## Pre-requisites ​

Before using the PWA plugin, ensure you've enabled mobile app features in WeWeb. Refer to our mobile app features guide guide for instructions.

## Installation ​

Navigate to Plugins > Extensions > PWA in your WeWeb project. Add the PWA plugin. No additional setup is required!  
  
Once installed, you can use PWA actions in your workflows. Each action is tagged with supported mobile platforms (Android and/or iOS).

### Images

<https://docs.weweb.io/assets/pwa1.nhLOyZEL.png>

<https://docs.weweb.io/assets/pwa2.B-sTEUZh.png>

## Add to Home Screen ​

This action allows users to install your PWA on Android devices, making it accessible from the home screen like a native app.  
In the example below, we added a button and triggered a workflow that installs the app on click:  
Add a button that triggers a workflow on click  
  
Add the Add to Home Screen action  
  
If anyone clicks on the button now, they will be able to add the application to the home screen.

### Images

<https://docs.weweb.io/assets/pwa4.DsxbFZ_h.png>

<https://docs.weweb.io/assets/pwa3.BbzZJc1t.png>

## How it works ​

Websites meeting PWA criteria (manifest file, service worker) can offer installation. Learn how to enable PWA in a WeWeb app . The action prompts Android users to add the PWA to their device's home screen. The PWA opens in a standalone window without browser UI.

### Images

<https://docs.weweb.io/assets/pwa-add-to-home-screen.DXFRVR-f.png>

## Geolocate ​

The Geolocate action accesses the user's geographical location.

## Usage ​

Triggers a browser permission prompt for location access. Accesses device location services (GPS, Wi-Fi, cellular networks).

## Use cases: ​

Location-based services or content Customized user experiences Mapping and navigation features Location tagging for posts or photos

## Implementation ​

Add the Geolocate action to a workflow (e.g. on app load or page load or on click of a button)  
  
If we test the action, we will get a timestamp and the user’s current coordinates:  
  
Use a Change variable value action to store the result of the Geolocate action, which includes: a coords object: the device's geographical position, and a timestamp : the date and time of when the location was obtained.

### Images

<https://docs.weweb.io/assets/pwa5.DncL58ss.png>

<https://docs.weweb.io/assets/pwa6.Eg0mne9w.png>

<https://docs.weweb.io/assets/pwa7.BcrA0_Qa.png>

<https://docs.weweb.io/assets/pwa-geolocate-error.B9_ZMUp-.png>

## Share ​

Enables users to share text and URLs using the device's native sharing mechanism.

## Functionality: ​

Opens the device's built-in share menu. Users can choose from available sharing options (messaging apps, email, social media).

## Implementation: ​

Add a Share button to your page.  
  
Create a workflow triggered by the button click. Include the Share action with the following fields: Title : Main headline for social media shares Text : Description under the title URL : Web address to be shared

### Images

<https://docs.weweb.io/assets/pwa8.DylfJi3e.png>

<https://docs.weweb.io/assets/pwa9.MYzV7P4O.png>

## Vibrate ​

Causes Android devices to vibrate based on a specified pattern.

## Use cases ​

Notification systems (e.g. vibrate when user receives new message) Error management (e.g. vibrate if user login failed) Mobile gaming (e.g. vibrate when user attemps forbidden move)

## Implementation: ​

Add the Vibrate action to a workflow (e.g. triggered by a realtime event like a database change or a user attempting a forbidden move in a chess game) Define a vibration pattern as an array of numbers: Odd-indexed elements (including the first) specify vibration periods in milliseconds. Even-indexed elements specify pause periods in milliseconds.  
  
Example: [200, 100, 300, 200]  
200ms vibration 100ms pause 300ms vibration 200ms pause

### Images

<https://docs.weweb.io/assets/pwa10.D2GFZ_lq.png>

## Show Notification ​

Presents alerts and updates to users.

## When will it work: ​

app closed ❌ app in use ✅ app opened but screen locked ✅ app opened in the background ✅

## Implementation: ​

Add the Show notification action to your workflow. Configure the following fields: Title : Main headline of the notification Body : Detailed content Icon URL : URL of a small icon (optional) Image URL : URL of a larger image (optional) Tag : Identifier for grouping or replacing notifications Data : Additional JSON-formatted data Vibration Pattern : Array of numbers defining vibration behavior

### Images

<https://docs.weweb.io/assets/pwa11.DLle2GNT.png>

<https://docs.weweb.io/assets/pwa12.ns0mVXqm.png>

## Recommended dimensions for icon and image URLs: ​

width around 300px height around 200px (this idea is to have a landscape format) Avoid images wider than 1500-2000px The image size should not exceed 1-2MB

## Device Motion ​

There are two actions for web-apps to access your phone's movement data:  
Request Motion Permission : Asks for the user's consent to access motion data, ensuring privacy and security. Listen Device Motion : If permission is granted, this action captures real-time data from the user's phone sensors (mainly accelerometer and gyroscope) to respond to the device's movements.

## Use cases ​

Step counter Mobile gaming Augmented reality apps

## Implementation: ​

For the Start button of the step counter app, we could trigger a workflow with those two actions:  
  
As an example, we might add a Change a variable action to update a stepCount variable based on motion data available in the From PWA section of variables.  
  
In the context of a fitness app, these properties can be used to create features such as step counting, distance estimation, and basic fitness tracking. While not as accurate as dedicated devices, they can add functional elements to your app. The possibilities of what you can do with device motion are endless!  
Learn more about device motion properties .

### Images

<https://docs.weweb.io/assets/pwa13.CJmG1Smk.png>

<https://docs.weweb.io/assets/pwa14.C-c7xpTx.png>

## Additional Device Information ​

The PWA plugin also provides access to various device information:  
  
battery : This object provides information about the battery status of the device on which the PWA is running. It includes details such as the battery - level and whether the device is charging. deviceInfo : Provides general information about the device itself, such as the model, operating system, and other hardware details, including brand/model. network : Offers details about the network status of the device, such as whether the device is connected to the internet, the type of connection (Wi-Fi, cellular). pageVisibility : This variable shows if the PWA page is visible to the user. If it's true, the page is open and visible in the foreground. If it's false, the page is in the background or the device screen is off.

### Images

<https://docs.weweb.io/assets/pwa15.REDwYZK-.png>

## Conclusion ​

The PWA plugin significantly enhances your WeWeb projects, bringing them closer to native app functionality. By leveraging these features, you can create more engaging and responsive web applications that take full advantage of mobile device capabilities.