Find a Location

Description

Find a Location Near You

Enter ZIP or City, State

Search

missing Google Maps API key.'; return; } if (document.getElementById('pflm_gmaps')) { const t = setInterval(()=>{ if (window.google && window.google.maps){ clearInterval(t); cb(); } }, 100); return; } const s = document.createElement('script'); s.id = 'pflm_gmaps'; s.src = 'https://maps.googleapis.com/maps/api/js?key=' + encodeURIComponent(API_KEY) + '&libraries=places'; s.async = true; s.defer = true; s.onload = cb; document.head.appendChild(s); }

function initMap(){ geocoder = new google.maps.Geocoder(); map = new google.maps.Map(document.getElementById('pflm_map'), { zoom: DEFAULT_ZOOM, center: {lat: 33.1959, lng: -117.3795}, mapTypeControl: true }); marker = new google.maps.Marker({ map }); info = new google.maps.InfoWindow(); if (DEFAULT_ADDR) { geocoder.geocode({ address: DEFAULT_ADDR }, (results, status) => { if (status === 'OK' && results && results[0]) { const loc = results[0].geometry.location; map.setCenter(loc); marker.setPosition(loc); } }); } } function normalizePhone(raw){ const digits = String(raw || '').replace(/[^0-9]/g,''); if (!digits) return ''; if (digits.length === 10) return '+1' + digits; if (digits.length === 11 && digits.startsWith('1')) return '+' + digits; return '+' + digits; } function normalizeUrl(raw){ const u = String(raw || '').trim(); if (!u) return ''; if (/^https?:\/\//i.test(u)) return u; return 'https://' + u; } function renderFranchise(fr){ if (!fr) return '

No installer found for that ZIP yet.

'; const name = fr.name || 'Installer'; const zip = (fr.zip_code || '').toString().trim(); let addr = (fr.address || '').toString().trim(); const showAddr = (fr.display_address === undefined || fr.display_address === null) ? true : (parseInt(fr.display_address, 10) === 1 || fr.display_address === true); // Avoid duplicate zip-only address output like "21201, 21201, USA" if (zip && addr === zip) addr = ''; if (zip && addr.toLowerCase().startsWith((zip + ',').toLowerCase())) { // keep as-is, but don't print zip separately } const phoneDigits = normalizePhone(fr.phone); const websiteUrl = normalizeUrl(fr.website); const displayPhone = (fr.phone || '').toString().trim(); const displayEmail = (fr.email || '').toString().trim(); let html = ''; html += '

'; html += '
' + name + '

'; if (addr && showAddr) html += '
' + addr + '

'; // Show phone + email as simple links under the address (no extra CTA buttons) if (phoneDigits && displayPhone) { html += '
' + displayPhone + '

'; } if (displayEmail) { html += '
' + displayEmail + '

'; } if (fr.distance_miles !== undefined && fr.distance_miles !== null && fr.distance_miles !== '') { html += '
Closest location (' + fr.distance_miles + ' mi)

'; } // Keep only the Website button if (websiteUrl) { html += '
Website

'; } html += '

'; return html; } async function doSearch(query){ const resultEl = document.getElementById('pflm_result'); resultEl.textContent = 'Searching...'; const raw = String(query || '').trim(); if(!raw){ resultEl.textContent = 'Please enter a ZIP or City, State.'; return; } const isZip = /^\d{5}(?:-\d{4})?$/.test(raw); const cleanedZip = isZip ? raw.replace(/[^0-9]/g,'').substring(0,5) : ''; const q = isZip ? cleanedZip : raw; // Prefer REST (clean URLs, cacheable), but fall back to admin-ajax if REST is blocked/hidden. const restUrl = REST_URL + '?q=' + encodeURIComponent(q); async function tryRest(){ const res = await fetch(restUrl, { headers: { 'Accept': 'application/json' } }); if(!res.ok)

throw new Error('REST ' + res.status); return await res.json(); } async function tryAjax(){ if(!AJAX_URL) throw new Error('No AJAX URL'); const form = new URLSearchParams(); form.set('action','pflm_locator_search'); form.set('q', q); if(NONCE) form.set('nonce', NONCE); const res = await fetch(AJAX_URL, { method: 'POST', headers: { 'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8' }, body: form.toString() }); if(!res.ok) throw new Error('AJAX ' + res.status); return await res.json(); } let data; try { data = await tryRest(); } catch(e){ try { data = await tryAjax(); } catch(e2){ console.error('Locator search failed', e, e2); resultEl.textContent = 'Search failed. Please try again.'; return; } } if(!data || data.ok === false){ resultEl.textContent = (data && data.message) ? data.message : 'No results.'; return; } // Center map on ZIP geocode if available. Server-side geocoding can be blocked // by API key restrictions, so we also do a client-side fallback. let searchCenter = null; if (data.geo && data.geo.lat && data.geo.lng) { searchCenter = { lat: parseFloat(data.geo.lat), lng: parseFloat(data.geo.lng) }; map.setCenter(searchCenter); map.setZoom(11); } else { // Client-side fallback via the Maps JS Geocoder (usually already allowed). await new Promise((resolve) => { if (!geocoder) { resolve(); return; } geocoder.geocode({ address: (isZip ? ('ZIP ' + cleanedZip + ', USA') : (raw + ', USA')), componentRestrictions: { country: 'US' } }, (results, status) => { if (status === 'OK' && results && results[0] && results[0].geometry && results[0].geometry.location) { const loc = results[0].geometry.location; searchCenter = { lat: loc.lat(), lng: loc.lng() }; map.setCenter(searchCenter); map.setZoom(11); } resolve(); }); }); } // Clear existing installer markers clearResultMarkers(); // Normalize franchises: we may get // - `franchises` as an array // - `franchises` as an object keyed by index (rare PHP/json edge) // - a single `franchise` let franchises = []; if (data) { if (Array.isArray(data.franchises)) { franchises = data.franchises; } else if (data.franchises && typeof data.franchises === 'object') { // If the server returns an object instead of an array, recover safely. franchises = Object.values(data.franchises); } if ((!franchises || franchises.length === 0) && data.franchise) { franchises = [data.franchise]; } } franchises = (franchises || []).filter(Boolean); // Final normalization guard: some environments can JSON-encode a single item in // unexpected shapes. If we have *any* franchise payload, render it. const finalFranchises = (franchises && franchises.length) ? franchises : ((data && data.franchise) ? [data.franchise] : []); if(!data || finalFranchises.length === 0){ resultEl.textContent = (data && data.message) ? data.message : 'No installer found for that ZIP yet.'; return; } // Render installer cards immediately. resultEl.innerHTML = finalFranchises.map(renderFranchise).join(''); // Drop a marker for each franchise. // If we don't have per-franchise lat/lng yet, fall back to the searched ZIP center. const fallbackCenter = searchCenter ? searchCenter : {lat: map.getCenter().lat(), lng: map.getCenter().lng()}; finalFranchises.forEach(fr => { const pos = (fr && fr.lat && fr.lng) ? {lat: parseFloat(fr.lat), lng: parseFloat(fr.lng)} : fallbackCenter; const m = new google.maps.Marker({ position: pos, map, title: fr.name || ('Franchise #' + fr.franchise_id) }); m.addListener('click', () => { info.setContent('**' + (fr.name || '') + '**'); info.open(map, m); }); markers.push(m); }); } function wire(){ const btn = document.getElementById('pflm_btn'); const input = document.getElementById('pflm_zip'); btn.addEventListener('click', () => { const q = (input.value || '').trim(); doSearch(q); }); input.addEventListener('keydown', (e) => { if (e.key === 'Enter') btn.click(); }); } document.addEventListener('DOMContentLoaded', () => { ensureGoogle(() => { initMap(); wire(); }); }); })();

Date Created
October 23, 2025
Author

pfcadmin