Here’s what you’ll have after completing the steps (and passing each checkpoint):

1. **Inline game, no pop-up tab**
   * FlickWord opens inside your app in a modal (iframe) instead of window.open.
   * Close button clears the frame and returns to the home view.
   * Fallback to window.open remains if the modal host isn’t found.
2. **Clean, aligned layout**
   * The FlickWord card and the left column share the same height.
   * No more bottom-edge misalignment on the row.
3. **Working message wiring**
   * postMessage from the game page still updates the home card.
   * Results flow through your existing handlers; no broken events.
4. **Local streaks & best streak**
   * Streak increments on consecutive winning days.
   * Best streak is tracked and displayed.
   * Stored in localStorage (no backend change).
5. **Visible stats/badge on the home card**
   * Today’s guesses, games played, best streak update live.
   * Simple badge label (e.g., “Binge Watcher”, “Marathoner”) appears when thresholds are met.
6. **Jeopardy-style meta game (starter)**
   * A lightweight meta state stored locally (current set ID, clues array, progress).
   * Each win unlocks a daily clue; on completion, you trigger a reveal/notification hook.
7. **Pro gating hooks (no paywall yet)**
   * A simple isPro flag controls visibility of Pro-only UI (e.g., leaderboard placeholder).
   * Ready to wire to your real account system later.
8. **Controlled rollout with guardrails**
   * Each step has a pass/fail test (UI/console/storage) before moving on.
   * Minimal file touch points; easy rollback if a step fails.
9. **Zero server changes required**
   * Everything runs with your current static stack and localStorage.
   * Future server/account integration can be added without reworking this flow.

If you want, I’ll package the exact code edits as patch blocks (by file/line) so you can paste, run the test, and proceed step-by-step without drift.

**📦 FlickWord Inline Integration — Update Pack**

**Files touched**

* index.html
* scripts/inline-script-01.js
* styles/components.css *(or)* styles/inline-style-01.css (whichever currently styles the row)
* *(Optional tiny markup tweak)* index.html → ensure stat IDs exist

**0) Safety: make a quick backup**

REM from project root

copy index.html index.bak.html

copy scripts\inline-script-01.js scripts\inline-script-01.bak.js

copy styles\components.css styles\components.bak.css

copy styles\inline-style-01.css styles\inline-style-01.bak.css

**1) Add a modal host so FlickWord runs inline (no new tab)**

**Edit:** index.html — place this **once**, near the FlickWord card (anywhere inside <body>, typically after the card).

<!-- FlickWord Inline Modal Host -->

<div id="flickwordModal" class="modal" style="display:none; position:fixed; inset:0; z-index:9999; background:rgba(0,0,0,0.6);">

<div class="modal-content" style="max-width:900px; width:92%; margin:5vh auto; background:var(--card,#fff); border-radius:12px; box-shadow:0 10px 30px rgba(0,0,0,.25); padding:12px; display:flex; flex-direction:column; gap:8px;">

<div style="display:flex; align-items:center; justify-content:flex-end;">

<button id="closeFlickwordModal" class="btn" aria-label="Close" title="Close" style="font-size:20px; line-height:1; padding:6px 10px;">×</button>

</div>

<iframe id="flickwordFrame" src="" title="FlickWord"

style="width:100%; height:70vh; border:0; border-radius:8px; background:#fff;"></iframe>

</div>

</div>

**Test 1 (UI):**

* Reload the app. The modal is **not visible**.
* No console errors.

**2) Switch from window.open() to inline modal**

**Edit:** scripts/inline-script-01.js

Find the openFlickWord() method and replace the window.open(...) path with this:

openFlickWord() {

console.log('🎯 Opening FlickWord');

const now = new Date();

const iso = now.toISOString().split('T')[0];

const flickwordUrl = `/features/flickword.html?date=${iso}`;

// Inline modal path

const modal = document.getElementById('flickwordModal');

const frame = document.getElementById('flickwordFrame');

if (modal && frame) {

frame.src = flickwordUrl;

modal.style.display = 'block';

} else {

// Fallback (safety)

console.warn('FlickWord modal not found; falling back to new tab');

window.open(flickwordUrl, '\_blank');

}

}

In the same file, ensure your event listeners set a close handler (inside the same init where you wire the Play button):

const closeBtn = document.getElementById('closeFlickwordModal');

if (closeBtn) {

closeBtn.addEventListener('click', () => {

const modal = document.getElementById('flickwordModal');

const frame = document.getElementById('flickwordFrame');

if (frame) frame.src = 'about:blank';

if (modal) modal.style.display = 'none';

// Maintain existing semantics

window.postMessage({ type: 'flickword:close' }, '\*');

});

}

**Test 2 (UI):**

* Click **Play Today’s Word** → modal appears, iframe loads /features/flickword.html?....
* Click **×** → modal hides and iframe clears.
* No new console errors.

**3) Fix the row misalignment**

Add these safe overrides (pick **one** stylesheet that actually governs this row—if unsure, add to both styles/components.css and styles/inline-style-01.css):

/\* Align the two columns to share the same height \*/

.quote-flickword-container {

align-items: stretch !important;

}

/\* Let the left column stretch (remove height lock) \*/

.left-column {

height: auto !important;

}

/\* Ensure the right card stretches to the row height \*/

.flickword-card {

align-self: stretch !important;

height: auto;

display: flex;

flex-direction: column;

}

/\* Keep the two left boxes equal height if you use a stacked design \*/

.left-column > \* {

flex: 1 1 0;

min-height: 0;

display: flex;

flex-direction: column;

}

**Test 3 (Visual):**

* Reload → the right card’s bottom edge should align with the combined height of the left stack.
* Toggle content density (e.g., change window width) → edges remain aligned.

**4) Add streak + best streak logic (localStorage only)**

**Edit:** scripts/inline-script-01.js — extend your existing result handler:

handleFlickWordResult(result) {

console.log('🎯 FlickWord result received:', result);

const results = JSON.parse(localStorage.getItem('flickword:results') || '{}');

const today = new Date().toISOString().slice(0,10);

const lastPlayed = localStorage.getItem('flickword:lastPlayed');

// Record today's result

results[today] = {

guesses: result?.guesses ?? null,

won: !!result?.won,

answer: result?.answer ?? null,

ts: Date.now()

};

// Streak math

let streak = parseInt(localStorage.getItem('flickword:streak') || '0', 10);

let best = parseInt(localStorage.getItem('flickword:bestStreak') || '0', 10);

if (lastPlayed !== today) {

const yday = new Date(Date.now() - 86400000).toISOString().slice(0,10);

const consecutive = (lastPlayed === yday);

if (result?.won) {

streak = consecutive ? (streak + 1) : 1;

if (streak > best) best = streak;

} else {

streak = 0;

}

}

localStorage.setItem('flickword:lastPlayed', today);

localStorage.setItem('flickword:streak', String(streak));

localStorage.setItem('flickword:bestStreak', String(best));

localStorage.setItem('flickword:results', JSON.stringify(results));

this.updateFlickWordStats?.();

}

**Test 4 (Console/Storage):**

* In DevTools → Application → Local Storage, simulate two consecutive “wins” (you can manually call handleFlickWordResult({won:true, guesses:4, answer:'LOKI'}) twice on different “days” by temporarily editing today or localStorage).
* Verify flickword:streak, flickword:bestStreak, and flickword:results update as expected.

**5) Surface streaks and games played in the home card**

**5a) Ensure stat placeholders exist in the card markup**

**Edit:** index.html — find the FlickWord stats area and ensure you have these spans:

<div class="flickword-stats">

<div><strong>Today:</strong> <span id="todayScore">0</span></div>

<div><strong>Best Streak:</strong> <span id="bestStreak">–</span></div>

<div><strong>Games Played:</strong> <span id="gamesPlayed">0</span></div>

<div><strong>Badge:</strong> <span id="flickwordBadge">—</span></div>

</div>

**5b) Update the stat writer**

**Edit:** scripts/inline-script-01.js — in your updateFlickWordStats():

updateFlickWordStats() {

const results = JSON.parse(localStorage.getItem('flickword:results') || '{}');

const bestStreak = parseInt(localStorage.getItem('flickword:bestStreak') || '0', 10);

const gamesPlayed = Object.keys(results).length;

const today = new Date().toISOString().slice(0,10);

const todayScoreEl = document.getElementById('todayScore');

const bestStreakEl = document.getElementById('bestStreak');

const gamesPlayedEl = document.getElementById('gamesPlayed');

const badgeEl = document.getElementById('flickwordBadge');

const todays = results[today];

if (todayScoreEl) todayScoreEl.textContent = todays?.guesses ?? 0;

if (bestStreakEl) bestStreakEl.textContent = bestStreak;

if (gamesPlayedEl) gamesPlayedEl.textContent = gamesPlayed;

if (badgeEl) {

badgeEl.textContent = bestStreak >= 7 ? '🏆 Marathoner'

: bestStreak >= 3 ? '🎬 Binge Watcher'

: '—';

}

}

**Test 5 (UI):**

* Reload → stats populate with stored values (or defaults).
* Trigger a “win” and call updateFlickWordStats() → numbers/badge update.

**6) Add Jeopardy-style meta-game scaffold**

**Edit:** scripts/inline-script-01.js — at the end of handleFlickWordResult, after saving streaks:

// Meta-game: unlock a clue for each win

try {

if (result?.won) {

const metaId = localStorage.getItem('flickword:meta:currentId') || 'week-001';

const clues = JSON.parse(localStorage.getItem('flickword:meta:clues') || '[]');

const progress = JSON.parse(localStorage.getItem('flickword:meta:progress') || '{"required":5,"unlocked":0}');

clues.push({

date: new Date().toISOString().slice(0,10),

clue: `Clue from word: ${result.answer || 'unknown'}`

});

progress.unlocked = Math.min(progress.unlocked + 1, progress.required);

localStorage.setItem('flickword:meta:currentId', metaId);

localStorage.setItem('flickword:meta:clues', JSON.stringify(clues));

localStorage.setItem('flickword:meta:progress', JSON.stringify(progress));

if (progress.unlocked >= progress.required) {

showNotification?.('🎉 Meta-puzzle complete! Claim your reward.', 'success');

// TODO: open a lightweight reveal modal

}

}

} catch (e) {

console.warn('Meta-game update failed:', e);

}

**Test 6 (Storage/Console):**

* Manually call handleFlickWordResult({won:true, guesses:4, answer:'LOKI'}) across 5 “days”.
* Verify:
  + flickword:meta:clues grows.
  + flickword:meta:progress.unlocked reaches required.
  + Notification appears once complete.

**7) Add Pro gating hooks (placeholder only)**

**Edit:** scripts/inline-script-01.js — inside your FlickWord initializer:

const isPro = !!(window.appData?.settings?.pro); // adapt to your real flag

// Example gate (if you later add a leaderboard element by id)

const leaderboardEl = document.getElementById('flickwordLeaderboard');

if (leaderboardEl) leaderboardEl.style.display = isPro ? 'block' : 'none';

**Test 7 (UI):**

* In console, toggle appData.settings.pro = true/false; then re-run your initializer.
* Verify gated elements show/hide accordingly.

**README (concise)**

**Goal**

Move FlickWord from a pop-up tab to an **inline modal**, fix row alignment, add **streaks**, simple **badges**, and a **meta-game scaffold**—with strict pass/fail checks after each change.

**Steps & Tests**

1. **Modal host** in index.html → **Test:** modal hidden by default; no console errors.
2. **Inline open** in scripts/inline-script-01.js → **Test:** Play opens modal; × closes; no new errors.
3. **Alignment CSS** in styles/\*.css → **Test:** right card bottom aligns with left stack across widths.
4. **Streak logic** → **Test:** storage shows flickword:streak, bestStreak, results updating correctly.
5. **Home card stats** → **Test:** todayScore, bestStreak, gamesPlayed, badge update live.
6. **Meta-game scaffold** → **Test:** after N wins, progress completes and shows notification.
7. **Pro gating hooks** → **Test:** toggle appData.settings.pro to verify gated UI.

**Rollback**

Restore the \*.bak.\* files you created in step 0.

**Notes**

* All new logic is **additive** and guarded; no server changes required.
* Keep the CSS overrides close to existing row styles to avoid specificity fights.
* If your stats markup already exists, keep it; only add the IDs if missing