

Bottom sheet (in-game web view)

The swipe-down sheet that shows every Baziness page inside your game. It slides up, closes with a swipe, a tap outside or the page's own button, and passes the page's messages to your game.

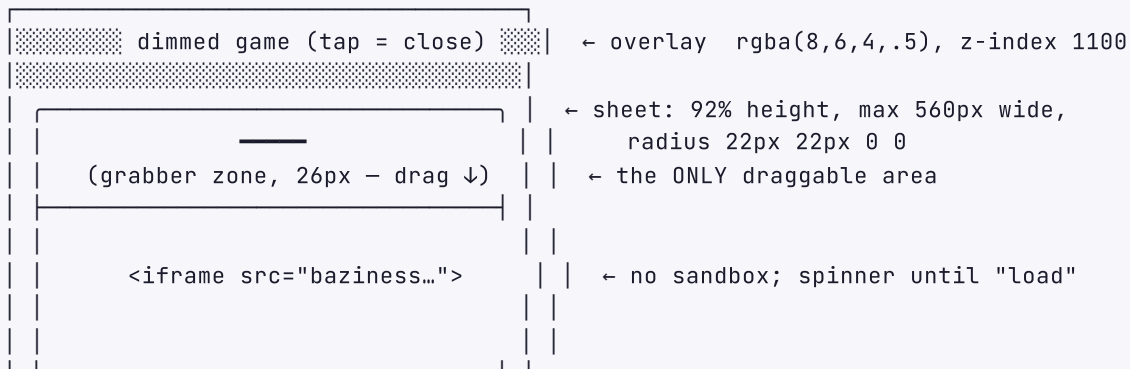
Why a sheet

Every Baziness page ([sign-in](#), [profile](#), [reward shop](#), [game offer](#)) opens **inside your game**, not in a new tab. The recommended container is a **bottom sheet**. It slides up over the dimmed game, stops a little short of the top, and has a rounded top edge with a drag handle. The player can close it in three ways, and the page can close it too.

Try it:

Anatomy

Text



How it must behave

1. **Opening:** draw the sheet below the screen, then animate it up on the next frame (about 320 ms, ease-out).
2. **Three ways to close:**
 - drag the **handle** down further than `min(150px, 28% of the sheet height)`; a shorter drag springs back

- tap the **dimmed area** above the sheet
 - the page sends `{ "type": "close" }`
3. **Only the handle is draggable.** The page swallows its own touches, so a drag that starts inside the page can't move the sheet.
 4. **No `sandbox` on the iframe.** The sign-in page needs its own cookies, pop-ups (for the code) and redirects. Allow `clipboard-read; clipboard-write; payment`.
 5. **Show a spinner** until the page has loaded.
 6. **Accept messages only from the page's own origin.** Read both string JSON and plain objects, and require a string `type`.
 7. **Close cleanly:** animate down, then tear down **once**. Point the frame at `about:blank`, stop listening for messages, and only then report "closed". If the sheet was opened again in the meantime, do nothing.
 8. **Stay above everything**, the splash screen included: the sign-in page can open from the splash.
 9. **Size it like your game's column:** full width on phones, at most about 560 px wide and centred on wide screens.

Web: TypeScript, React and CSS

TypeScript (web)

```
// bazinesswebview.ts - the sheet, framework-free.
export interface BazinessMessage { type: string; data?: unknown }

export class BazinessSheet {
  readonly el: HTMLElement;
  private sheet: HTMLElement;
  private frame: HTMLIFrameElement;
  private spinner: HTMLElement;
  private allowedOrigin: string | null = null;
  private isOpen = false;
  private dragging = false;
  private dragStartY = 0;
  private dragDelta = 0;
  private sheetHeight = 1;

  onMessage: ((m: BazinessMessage) => void) | null = null;
  onClosed: (() => void) | null = null;

  constructor() {
    this.frame = Object.assign(document.createElement("iframe"), { className: "baziness-frame" });
    this.frame.setAttribute("allow", "clipboard-read; clipboard-write; payment"); // no sandbox!
    this.spinner = Object.assign(document.createElement("div"), { className: "baziness-spinner" });
    const grabberZone = document.createElement("div");
```

```

grabberZone.className = "baziness-grabber-zone";
grabberZone.innerHTML = '<div class="baziness-grabber"></div>';
this.sheet = document.createElement("div");
this.sheet.className = "baziness-sheet";
this.sheet.append(grabberZone, this.spinner, this.frame);
this.el = document.createElement("div");
this.el.className = "overlay baziness-webview hidden";
this.el.append(this.sheet);
document.body.append(this.el);

this.frame.addEventListener("load", () => this.spinner.classList.add("hidden"));
grabberZone.addEventListener("pointerdown", (e) => this.dragStart(e));
grabberZone.addEventListener("pointermove", (e) => this.dragMove(e));
grabberZone.addEventListener("pointerup", () => this.dragEnd());
grabberZone.addEventListener("pointercancel", () => this.dragEnd());
this.el.addEventListener("pointerdown", (e) => { if (e.target === this.el) this.close();
}); // tap outside
}

open(url: string, cropBottomPx = 0): void {
    if (this.isOpen) return;
    this.isOpen = true;
    this.frame.style.marginBottom = cropBottomPx > 0 ? `${-cropBottomPx}px` : "";
    this.allowedOrigin = new URL(url).origin;
    this.spinner.classList.remove("hidden");
    this.el.classList.remove("hidden");
    this.sheet.style.transition = "none";
    this.sheet.style.transform = "translateY(100%)";
    this.frame.src = url;
    window.addEventListener("message", this.onWindowMessage);
    requestAnimationFrame(() => {
        this.sheet.style.transition = "";
        this.sheet.style.transform = "translateY(0)";
    });
}

close(): void {
    if (!this.isOpen) return;
    this.isOpen = false;
    this.dragging = false;
    window.removeEventListener("message", this.onWindowMessage);
    this.sheet.style.transition = "";
    this.sheet.style.transform = "translateY(100%)";
    let torn = false;
    const teardown = () => {
        if (torn || this.isOpen) return; // already torn down, or reopened meanwhile
        torn = true;
        this.sheet.removeEventListener("transitionend", teardown);
        this.el.classList.add("hidden");
        this.frame.src = "about:blank";
        this.frame.style.marginBottom = "";
        this.sheet.style.transform = "";
        this.allowedOrigin = null;
        this.onClosed?.();
    };
    this.sheet.addEventListener("transitionend", teardown);
    window.setTimeout(teardown, 400);
}

private dragStart(e: PointerEvent) {

```

```

    this.dragging = true;
    this.dragStartY = e.clientY;
    this.dragDelta = 0;
    this.sheetHeight = this.sheet.getBoundingClientRect().height || 1;
    this.sheet.style.transition = "none";
    (e.currentTarget as HTMLElement).setPointerCapture(e.pointerId);
  }
  private dragMove(e: PointerEvent) {
    if (!this.dragging) return;
    this.dragDelta = Math.max(0, e.clientY - this.dragStartY); // downward only
    this.sheet.style.transform = `translateY(${this.dragDelta}px)`;
  }
  private dragEnd() {
    if (!this.dragging) return;
    this.dragging = false;
    this.sheet.style.transition = "";
    if (this.dragDelta > Math.min(150, this.sheetHeight * 0.28)) this.close();
    else this.sheet.style.transform = "translateY(0)"; // spring back
  }

  private onWindowMessage = (event: MessageEvent) => {
    if (this.allowedOrigin && event.origin !== this.allowedOrigin) return;
    let value: unknown = event.data;
    if (typeof value === "string") { try { value = JSON.parse(value); } catch { return; } }
    if (value && typeof value === "object" && typeof (value as BazinessMessage).type ===
"string") {
      this.onMessage?.(value as BazinessMessage);
    }
  };
}

```

React

```

// BazinessSheet.tsx – the React version. Mount once at the app root; baziness.host drives
it.
import { useEffect, useRef, useState } from "react";
import { createPortal } from "react-dom";
import { baziness } from "../platform/baziness";

export function BazinessSheet() {
  const [url, setUrl] = useState<string | null>(null);
  const [shown, setShown] = useState(false);
  const [loading, setLoading] = useState(true);
  const [drag, setDrag] = useState(0);
  const dragStart = useRef<number | null>(null);

  useEffect(() => {
    baziness.host = {
      open: (next) => {
        setLoading(true);
        setUrl(next);
        requestAnimationFrame(() => requestAnimationFrame(() => setShown(true))); // slide up
      },
      close: () => setShown(false),
    };
    return () => { baziness.host = null; };
  }, []);

  useEffect(() => {
    // messages, from the page's origin

```

```

only
  if (!url) return;
  const origin = new URL(url).origin;
  const onMessage = (e: MessageEvent) => {
    if (e.origin !== origin) return;
    const msg = parse(e.data); // see "Handling page messages"
  }
  (Client code)
    if (msg) baziness.handleMessage(msg);
  };
  window.addEventListener("message", onMessage);
  return () => window.removeEventListener("message", onMessage);
}, [url]);

const onTransitionEnd = () => {
  if (!shown && url) { // finished sliding down → tear down
once
  setUrl(null);
  setDrag(0);
  baziness.handleClosed();
  measureViewport(); // the phone may have moved its nav
bar
  }
};

if (!url) return null;
return createPortal(
  <div className={`overlay baziness-overlay ${shown ? "is-open" : ""}` `
    onPointerDown={(e) => e.target === e.currentTarget && setShown(false)}>
    <div className="baziness-sheet"
      style={{ transform: shown ? `translateY(${drag}px)` : "translateY(100%)",
        transition: dragStart.current !== null ? "none" : undefined }}
      onTransitionEnd={onTransitionEnd}>
      <div className="baziness-grabber-zone"
        onPointerDown={(e) => { dragStart.current = e.clientY;
e.currentTarget.setPointerCapture(e.pointerId); }}
        onPointerMove={(e) => dragStart.current !== null && setDrag(Math.max(0,
e.clientY - dragStart.current))}
        onPointerUp={() => { dragStart.current = null; if (drag > 140) setShown(false);
else setDrag(0); }}>
        <span className="baziness-grabber" />
      </div>
      {loading && <div className="baziness-loading"><Spinner /></div>}
      <iframe className="baziness-frame" src={url} title="بارينس"
        allow="clipboard-read; clipboard-write; payment" onLoad={() =>
setLoading(false)} />
      </div>
    </div>,
    document.getElementById("app-frame") ?? document.body,
  );
}

```

CSS

```

/* Above the splash (z 1000) so it shows during the first-time login choice. */
.overlay.baziness-webview {
  position: fixed;
  inset: 0;
  z-index: 1100;
  display: flex;

```

```

    justify-content: center;
    align-items: flex-end;          /* dock the sheet to the bottom */
    background: rgba(8, 6, 4, 0.5); /* the dimmed, tap-to-close area */
}
.overlay.baziness-webview.hidden { display: none; }

.baziness-sheet {
    position: relative;
    width: min(560px, 100%);        /* full width on phones, capped on wide screens */
    height: 92%;                   /* stops short of the top */
    display: flex;
    flex-direction: column;
    background: #fff;
    border-radius: 22px 22px 0 0;
    overflow: hidden;              /* also clips the cropped bottom bar */
    box-shadow: 0 -10px 34px rgba(0, 0, 0, 0.42);
    transform: translateY(100%);
    transition: transform 320ms cubic-bezier(0.22, 0.61, 0.36, 1);
    will-change: transform;
}

/* The only draggable area – the iframe captures its own pointer events. */
.baziness-grabber-zone {
    flex: 0 0 auto;
    height: 26px;
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: grab;
    touch-action: none;
    background: #fff;
}
.baziness-grabber {
    width: 44px;
    height: 5px;
    border-radius: 999px;
    background: rgba(0, 0, 0, 0.22);
}

.baziness-frame {
    flex: 1 1 auto;
    width: 100%;
    min-height: 0;
    border: 0;
    background: #fff;
}

.baziness-spinner {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 42px;
    height: 42px;
    margin: -21px 0 0 -21px;
    border-radius: 50%;
    border: 4px solid rgba(120, 90, 46, 0.25);
    border-top-color: #b4824a;
    animation: spin 0.9s linear infinite;
}

```

```
.baziness-spinner.hidden { display: none; }  
@keyframes spin { to { transform: rotate(360deg); } }
```

Cocos Creator

The `BazinessWebView` component in [Client code → Cocos Creator](#) does the same job with a `WebView` prefab: a loading overlay until the page loads, a close button that appears after a few seconds, and `Open` / `Close` slide animations.

Unity

Unity has no built-in web view, so you need a plugin. This sketch uses [gree/unity-webview](#). The page starts 8% below the top of the screen, the drag handle is a Unity UI element above it, and the page's messages reach C# through a small JavaScript bridge.

C# (Unity)

```
// BazinessSheet.cs - a bottom sheet around a WebView plugin (gree/unity-webview).  
using System;  
using UnityEngine;  
using UnityEngine.EventSystems;  
  
public class BazinessSheet : MonoBehaviour, IDragHandler, IEndDragHandler // put this on  
the grabber UI  
{  
    [SerializeField] RectTransform grabber; // a 26px bar drawn by Unity above the web  
view  
    [SerializeField] GameObject dimmer; // full-screen button behind; onClick →  
Close()  
    WebViewObject web;  
    Action<string> onMessage;  
    Action onClosed;  
    float topMargin, drag;  
  
    public void Open(string url, Action<string> onMessage, Action onClosed = null)  
    {  
        this.onMessage = onMessage;  
        this.onClosed = onClosed;  
        topMargin = Screen.height * 0.08f; // the sheet stops 8% short of the top  
        drag = 0;  
        web = new GameObject("BazinessWebView").AddComponent<WebViewObject>();  
        web.Init(  
            cb: msg => this.onMessage?.Invoke(msg), // messages forwarded by the bridge  
below  
            ld: _ => web.EvaluateJS(  
                "window.addEventListener('message',function(e){" +  
                "Unity.call(typeof e.data==='string'?e.data:JSON.stringify(e.data));});");  
        Layout();  
        web.LoadURL(url);  
    }  
}
```

```

        web.SetVisibility(true);
        dimmer.SetActive(true);
        grabber.gameObject.SetActive(true);
    }

    public void OnDrag(PointerEventData e) { drag = Mathf.Max(0, drag - e.delta.y); Layout();
}

    public void OnEndDrag(PointerEventData e)
    {
        if (drag > Mathf.Min(150, (Screen.height - topMargin) * 0.28f)) Close();
        else { drag = 0; Layout(); } // spring back
    }

    public void Close()
    {
        if (web == null) return;
        Destroy(web.gameObject);
        web = null;
        dimmer.SetActive(false);
        grabber.gameObject.SetActive(false);
        onClose?.Invoke();
    }

    void Layout()
    {
        int top = (int)(topMargin + drag + grabber.rect.height);
        web.SetMargins(0, top, 0, 0);
        grabber.anchoredPosition = new Vector2(0, -(topMargin + drag));
    }
}

```

Common mistakes

Mistake	What happens
<code>sandbox</code> on the iframe	sign-in gets stuck: pop-ups and cookies are blocked
The whole sheet is draggable	drags inside the page scroll it and move the sheet, or nothing moves
No origin check on messages	a message from somewhere else is taken for a page message
"Closed" reported twice (animation end + timeout)	a waiting sign-in finishes twice; the starter pack opens twice
The sheet sits below the splash screen	the sign-in page opens invisibly behind it
The frame isn't reset to <code>about:blank</code>	the old page keeps running in the background