Admin Supabase Clients Interview Script

Opening (0:00-0:10)
"On the admin side we run a Next.js app that needs both browser and server Supabase clients, each with different privileges."

Server vs Browser Clients (0:10-0:35)
" `apps/wigg-admin/lib/supabaseAdmin.ts` instantiates a service-role client for API routes—think moderation queues where we need elevated access. Meanwhile `supabaseBrowser.ts` provides a lightweight anon client for client-side hooks, never exposing the service key."

Typed Schema (0:35-0:55)
"We centralize types in `apps/wigg-admin/lib/schema.ts` and `types.ts`, so both clients share the same enums and table definitions. That keeps admin UIs like `ModerationTable.tsx` type-safe when they join moderation requests with profile info."

Route Handlers (0:55-1:15)
"API routes— `app/api/moderation/{pending,approve,reject}/route.ts` —import the server client, verify auth, then call stored procedures. Because they run on the server we can keep the logic minimal yet secure."

Why it Matters (1:15-1:30)
"This split avoids leaking secrets to the browser, enforces RLS-friendly patterns, and keeps our admin dashboards performant."

Closing (1:30-1:35)
"It's a nice demonstration of multi-environment Supabase usage inside a modern Next app."