```
> npm run start-lab --arc=4 --session=S10E09

[Headline] Digital Accessibility: S10E09 — Companion Lab: Accessible
React Components

[Subtitle] Code-Grounded Architecture & Focus Management

[System Message] S10E08 established the foundational operations for
building and maintaining an accessible product.

[Status] SUCCESS

[System Message] Arc 4 initializing...
                 Launching hands-on engineering labs.
```

Gemini Notebook

Diagnosing the Three React Accessibility Killers

Error

Component.js

```
<div onClick={submit}>Submit
</div>
```

Non-Interactive Nodes

Breaks **WCAG 4.1.2 (Name/Role/Value)**. Fails to synthesize native keyboard events (Enter/Space).

Error

List.js

```
<Item key={Math.random()} />
```

Unstable Key Props

Destroys active DOM focus across re-renders, dropping screen reader users back to the top of the document.

Error

Form.js

```
<input id='email' />
<p id='hint'>
```

Hardcoded ID Collisions

Breaks **aria-describedby** mappings in mapping functions, causing inputs to read out the wrong (or multiple) labels.

Gemini Notebook

Anatomy of a WCAG 4.1.2 Failure

The Anti-Pattern

```
<div onClick={save}>Save</div>
```

AX Tree

Node: Text

Save

No role, no keyboard activation, no focus ring.

The Accessible Architecture

```
<button type='button' onClick={save}>Save</button>
```

AX Tree

Node: Button

Name: 'Save'

State: Focusable

Native `<button>` provides free **Enter/Space** activation, focus rings, and the crucial `button` role without manual polyfilling (`tabIndex`, `role`, `onKeyDown`).

Gemini Notebook

Deterministic Hydration-Safe Relationships

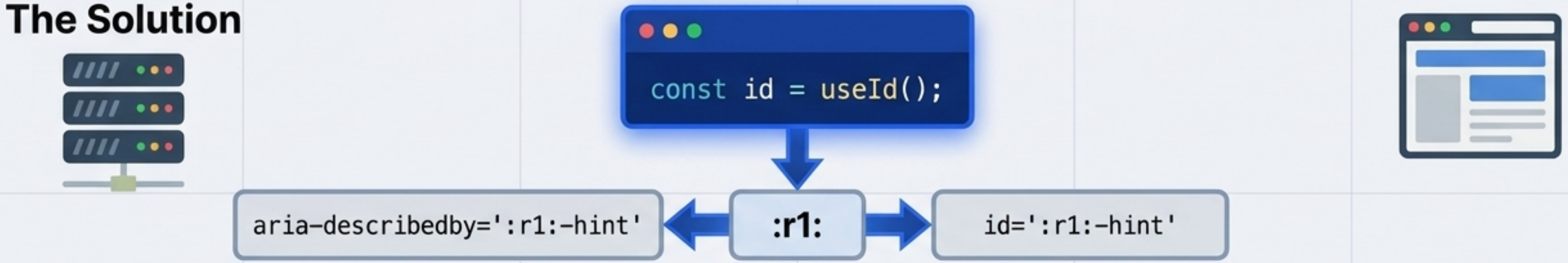
The Problem



Hydration Mismatch

> Error
>
> **Hydration Mismatch**: Server and Client IDs do not match, causing accessibility issues and React warnings.

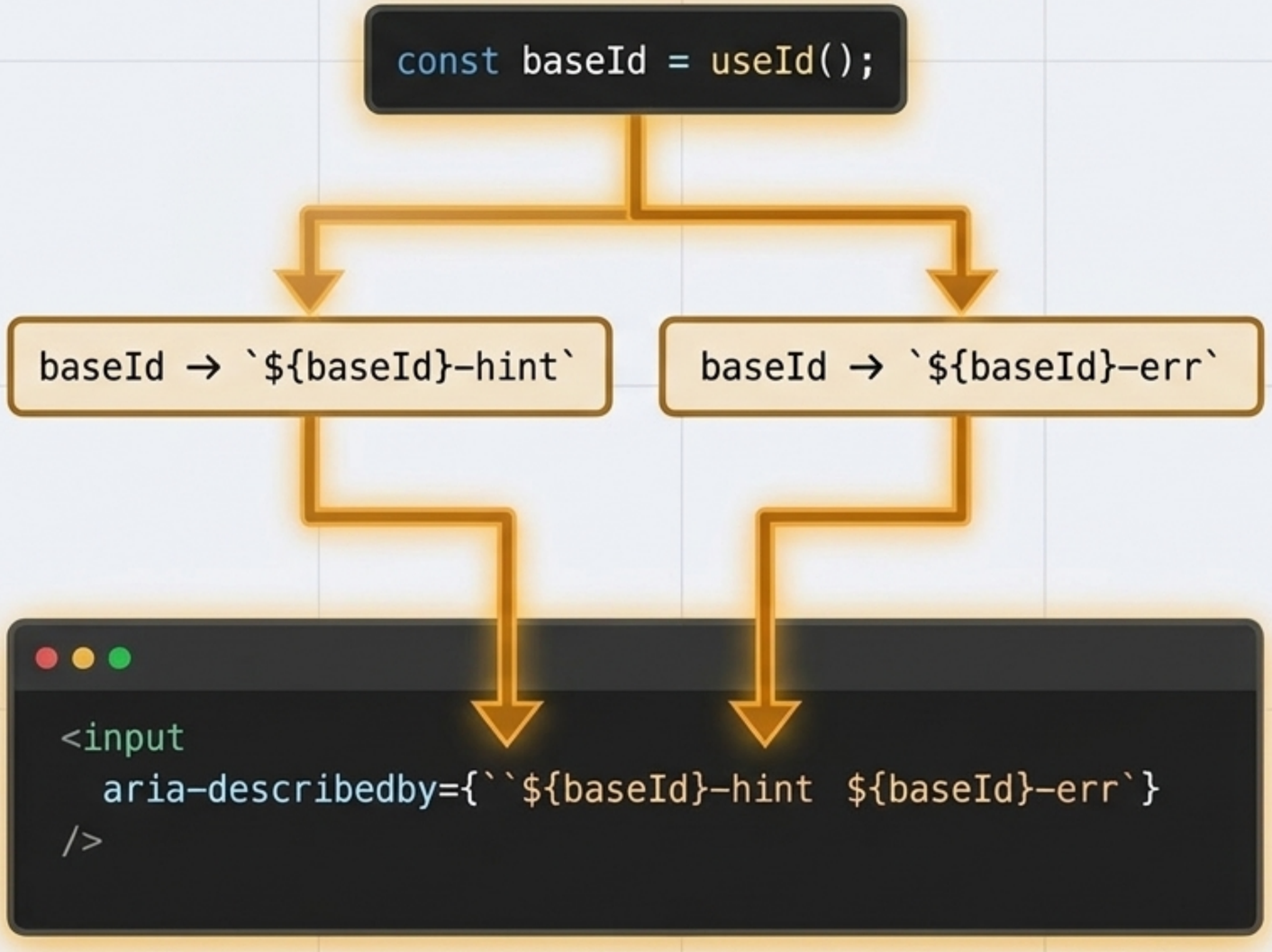
The Solution



`useId()` generates a unique string associated with a specific hook call. It guarantees exact matching between server HTML and hydrated client DOM, safely mapping labels to inputs.

Gemini Notebook

The <AccessibleField /> Pattern & Radix Slot



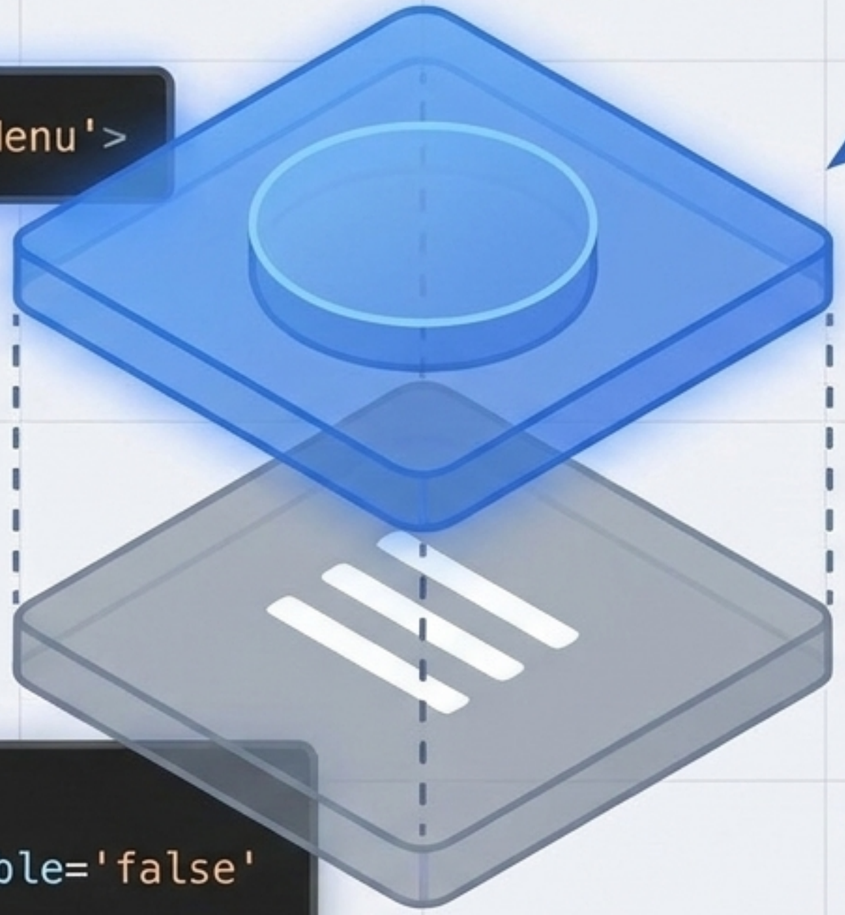

```
<Slot>
  <button type='button'>...</button>
</Slot>
```

Forwarding props via Radix Slot (asChild) merges props onto immediate native children. This eliminates HTML wrapper bloat (** <button>...</button> **) while maintaining strict ARIA relationships.

Gemini Notebook

Structurally Muting the Visuals: Accessible Icon Buttons

```
<button aria-label='Menu'>
```



The layer that speaks to the screen reader.

1

aria-hidden='true' structualy='true' structurally mutes the graphic so it isn't double-announced.

2

focusable='false' prevents severe legacy IE/Edge bugs where the SVG steals tab focus.

```
<svg
  aria-hidden='true' focusable='false'
/>
```

Gemini Notebook

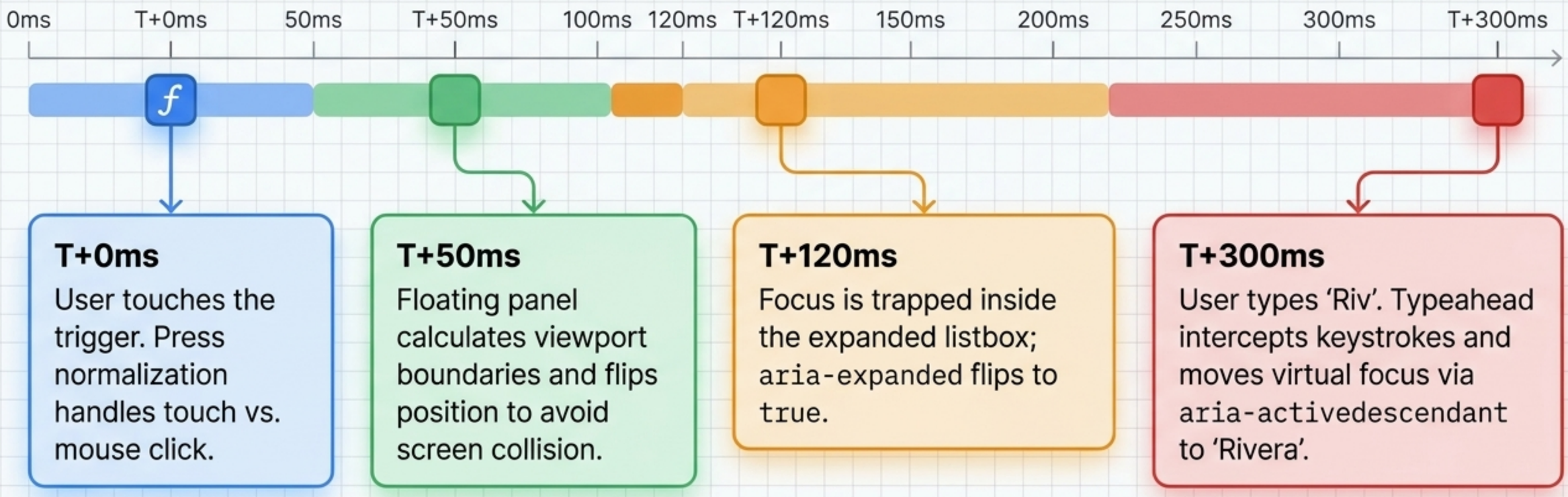
The Engineering Overhead: Headless vs. Hand-Rolled

Feature	Radix Primitives / React Aria	Hand-Rolled WAI-ARIA APG
Pointer Cancellation (Touch vs Click)	✅ Yes	❌ Massive manual JS required
Collision-Aware Positioning	✅ Yes	❌ Requires complex floating-UI math
Typeahead Navigation	✅ Yes	❌ High friction to build
aria-activedescendant Management	✅ Yes	❌ Prone to desync errors

Building disclosures, tabs, and menu buttons from scratch requires reinventing the wheel. Headless UI libraries provide the accessible DOM and keyboard state machine, leaving only the CSS styling to you.

Gemini Notebook

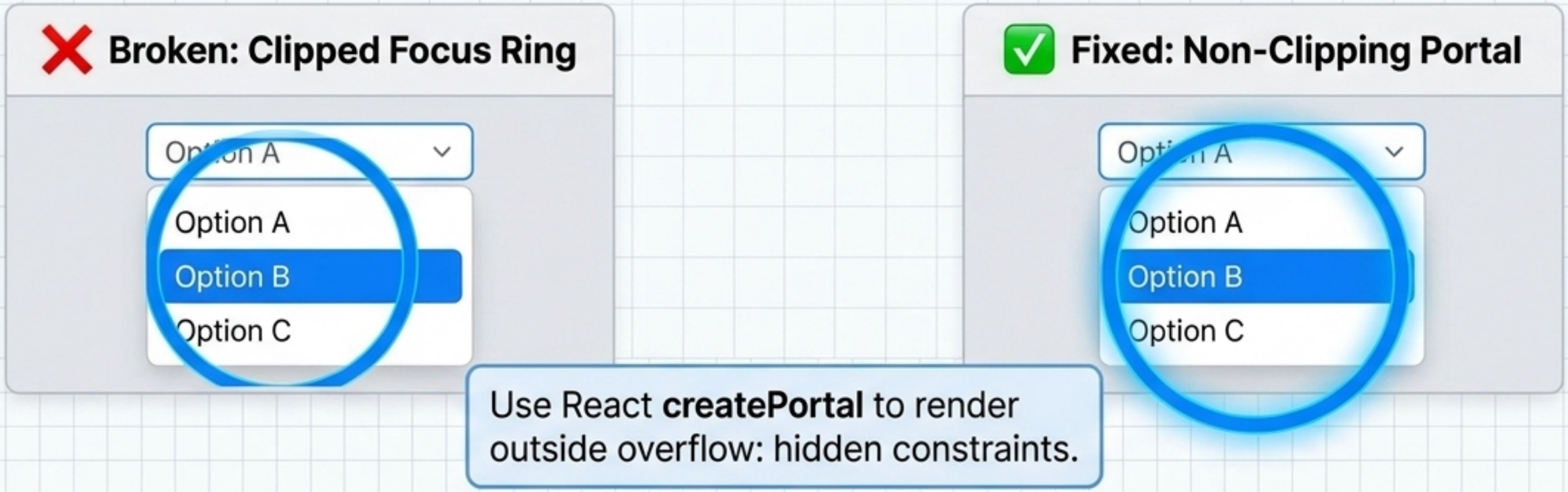
The Hidden Complexities of Interactive Primitives

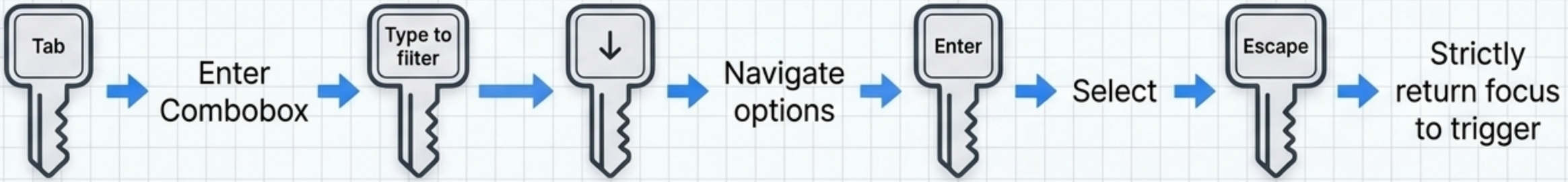


What looks like a simple UI component is actually an incredibly complex state machine. Headless libraries give you this timeline for free.

Gemini Notebook

5-Modality Refactor (1 & 2): Visual and Keyboard





Gemini Notebook

5-Modality Refactor (3 & 4): Screen Reader & 400% Zoom

Screen Reader Speech

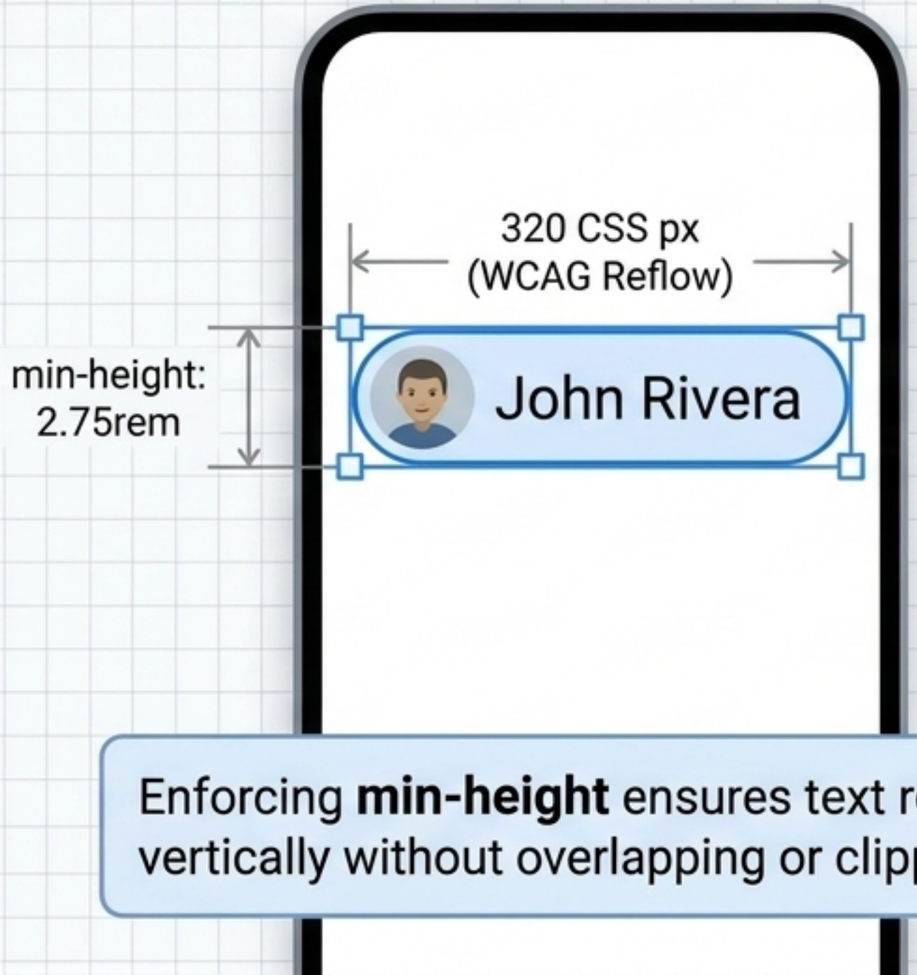
Before (Broken Context)

"Clickable. Assign incident... group. John Rivera. Clickable."

After (Verbatim Output)

Assign incident, combobox, collapsed, has popup listbox. Invalid entry: Assignee required. [Type 'Riv'] 2 results available. John Rivera, 1 of 2, option."

400% Zoom / WCAG Reflow

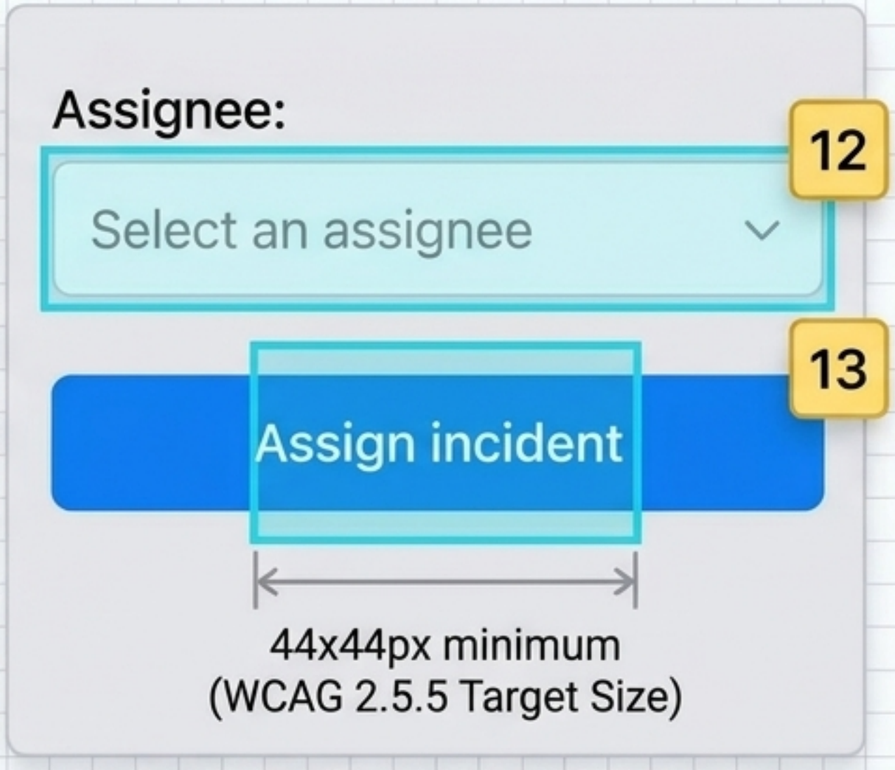


Enforcing **min-height** ensures text reflows vertically without overlapping or clipping.

Gemini Notebook

5-Modality Refactor (5): Mobile, Voice, & Switch

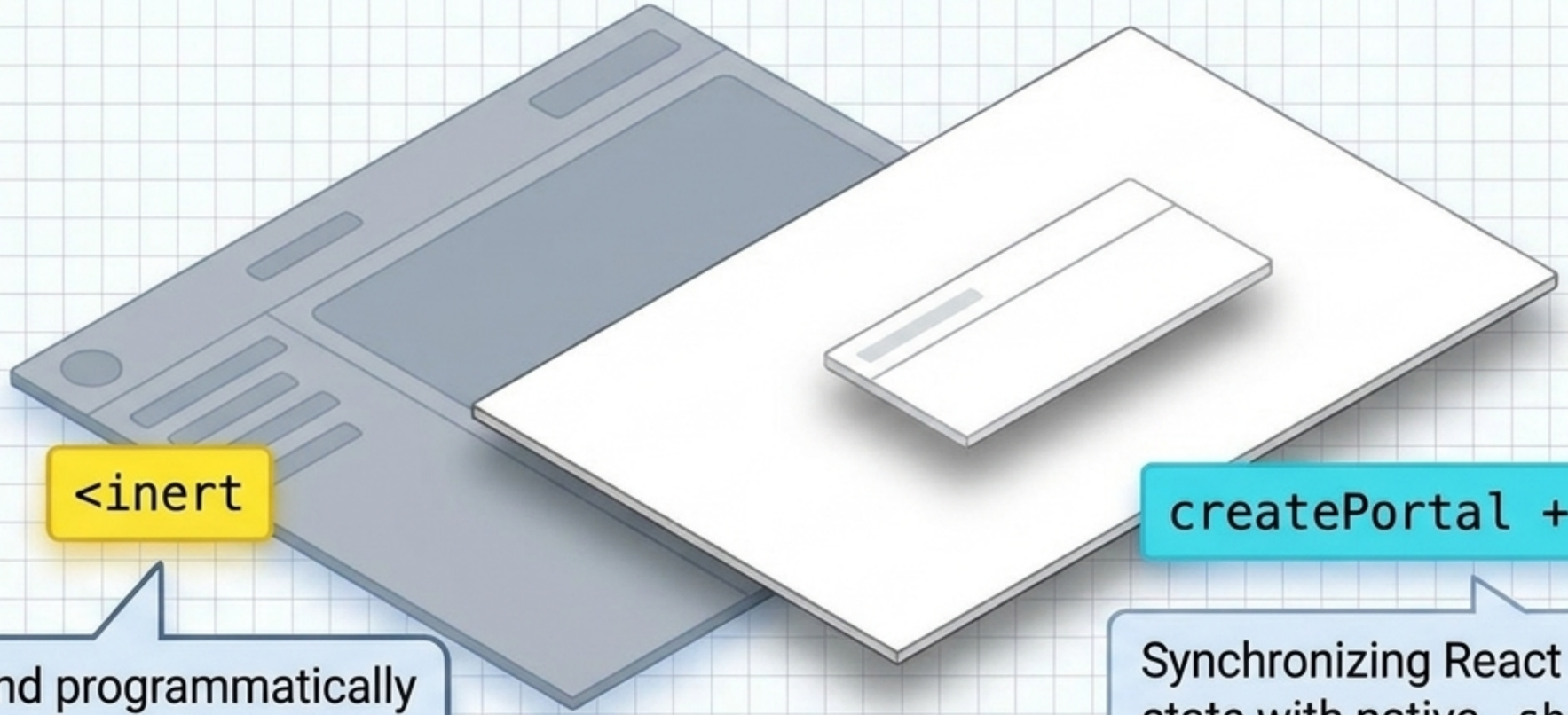
“Tap **Assign incident**”



The accessible name must exactly match the visible text, and the element must have a recognized **interactive role** (button, combobox). Otherwise, Voice Control users **literally cannot click** it.

Gemini Notebook

Modern Modals: Portals, <dialog>, and inert



`<inert`

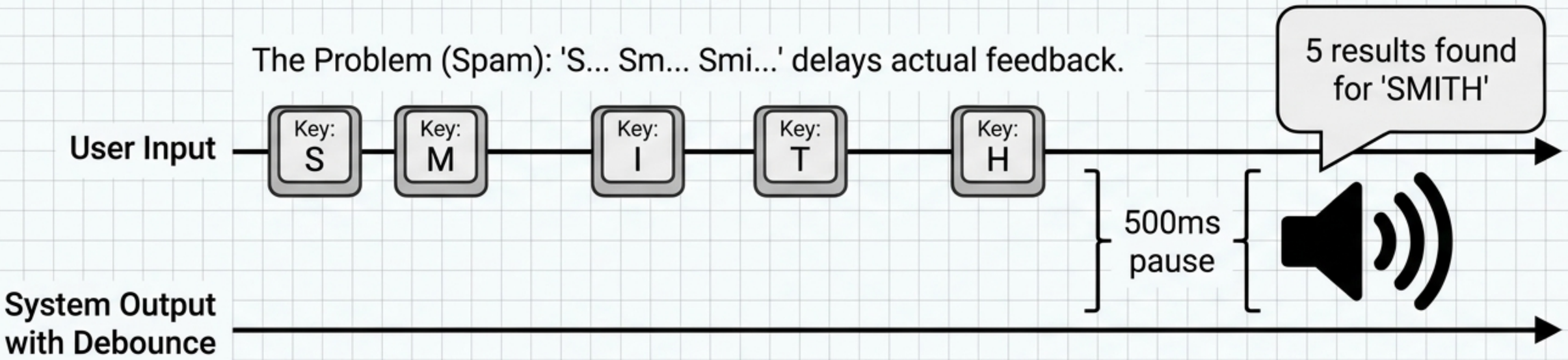
Visually and programmatically hides the root tree from the accessibility tree, **trapping focus** without complex JS.

`createPortal + <dialog>`

Synchronizing React useEffect state with native `.showModal()` and `.close()` methods to .close() methods to leverage native browser top-layer mechanics.

Gemini Notebook

Managing the Speech Queue: Debounced useAnnounce()



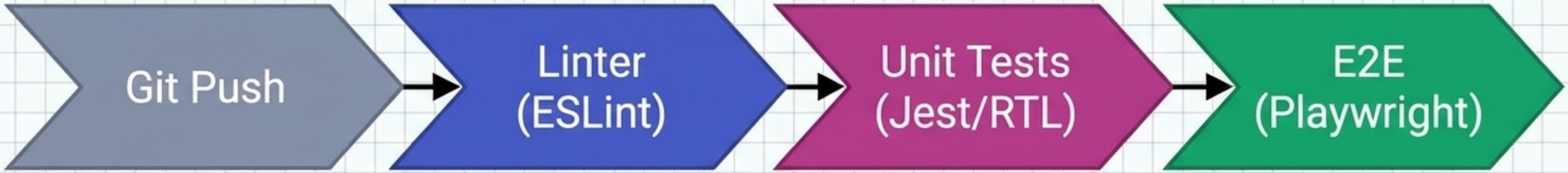
Syntax

```
role='status'
aria-live='polite'
```

Building a permanently mounted singleton **useAnnounce()** hook. Debouncing rapid state updates prevents screen-reader speech-queue spam (WCAG 4.1.3).

Gemini Notebook

Role-First Automated Guardrails



Configure **eslint-plugin-jsx-a11y** in strict mode to catch **no-noninteractive-element-interactions**.

Enforce **screen.getByRole(..., { name })** over brittle **getByTestId**. Use **userEvent.keyboard()** for true event synthesis.

Run **jest-axe** (toHaveNoViolations) across both closed AND open states of custom widgets using @axe-core/playwright.

Gemini Notebook

Constraining AI: AGENTS.md Rules

AGENTS.md

1. Never output <div onClick>. Always use native interactive elements or Radix Slot (asChild).

2. Use useId() for all aria-describedby and label relationships. Never hardcode IDs.

3. Implement strict focus management. Do not drop focus into the document body on unmount.

4. Prioritize WAI-ARIA APG patterns for complex widgets.

Embedding these rules contextually ensures LLMs **default to accessible-first React architecture**.

Gemini Notebook

Lab Exercise: Operations Desk Refactor

Bulk Status Menu & Confirmation Modal

```
// Objective: Strip out the div soup.

// Implement a headless primitive.

// Utilize the <dialog> + inert pattern.

// Ensure it passes jsx-a11y strict rules.

```

Lesson Note Field (Multi-Instance)

```
// Objective: Fix a multi-instance map loop.

// Implement 3-instance useId() checks to guarantee SSR safety and correct label mapping.

// Implement the debounced useAnnounce() for live character count warnings.

```

Gemini Notebook

Verification Checklist & Next Up

1. Zero **<div onClick>** violations (Native elements or Slot used).
2. Multi-instance SSR-safe **useId()** relationships established.
3. Headless primitives utilized for complex state machines.
4. 5-Modalities (Visual, Keyboard, SR, Zoom, Mobile/Touch) verified.
5. Automated tests pass exclusively using **getByRole**.

Next Episode



S10E10 — Companion Lab: Focus Management and Roving Tabindex.

Addressing the deep 2D grid roving-tabindex mechanics deferred from today's session.

Gemini Notebook