In a Frontend System Design interview, the distinction between **Progressive Hydration** and **Selective Hydration** is subtle but important.

While **Progressive Hydration** is the general *concept* of hydrating the page in pieces over time, **Selective Hydration** is the specific *implementation* introduced in **React 18** that allows React to prioritize parts of the page based on **user interaction**.

---

1. What makes Selective Hydration different?

The "magic" of Selective Hydration is that it is **interruptible** and **interaction-aware**.

In older patterns, even if you hydrated in pieces, if React started hydrating a large "Comments" section, the main thread was busy until that section was finished. If you clicked a "Login" button during that time, the click would be ignored until the comments were done.

Selective Hydration solves this with two key features:

A. Non-Blocking Hydration

React no longer has to hydrate the whole tree in one go. It can work on small chunks. If the browser needs to paint a frame or handle an animation, React can pause and let the browser work.

B. Interaction Prioritization (The "Click" Trigger)

This is the most important part for interviews. If you have multiple sections wrapped in <Suspense>, React will start hydrating them in the order they appear in the HTML.

However, if a user **clicks** on a component that hasn't been hydrated yet, React will:

1. **Stop** hydrating the current component.
2. **Jump** to the component the user clicked on.
3. **Hydrate** it immediately.
4. **Replay** the user's click (so the button actually works).
5. **Resume** hydrating the rest of the page once the interaction is handled.

---

2. How it works with Streaming SSR

Selective Hydration is built on top of **Streaming SSR**.

1. **Streaming:** The server sends the HTML for the "Header" and "Sidebar" immediately, but sends a <template> placeholder for a slow "Data Grid."
2. **Hydration:** React starts hydrating the Header and Sidebar.
3. **Late Arrival:** The HTML for the "Data Grid" arrives later through the stream.
4. **Selection:** React starts hydrating the Data Grid as soon as it arrives, without waiting for the entire page's JS to be ready.

---

3. Implementation Example

You don't need a special "SelectiveHydration" tag; it happens automatically when you use **React 18's hydrateRoot** and **<Suspense>**.

```
// App.js
<Layout>
  <Suspense fallback={<NavbarSkeleton />}>
    <Navbar /> {/* React starts here */}
  </Suspense>

  <Suspense fallback={<MainContentSkeleton />}>
    <MainContent /> {/* If user clicks here, React jumps here immediately */}
  </Suspense>

  <Suspense fallback={<FooterSkeleton />}>
    <Footer />
  </Suspense>
</Layout>
```

---

4. Comparison Table

Feature	Progressive Hydration (General)	Selective Hydration (React 18)
Concept	Hydrating the page in stages.	Prioritizing hydration based on user input.
Trigger	Usually scroll or idle time.	**User interactions (clicks/taps).**
Implementation	Custom (Intersection Observer, etc).	Native to React 18 + Suspense.
Main Thread	Can still be blocked by large chunks.	**Interruptible**; doesn't block interactions.

---

5. Interview Q&A

Q: Why is Selective Hydration better than just "Lazy Loading" components?
A: Lazy loading only delays when the code is downloaded. Selective Hydration manages when the code is executed and attached to the DOM. It ensures that even if all the code is downloaded, the most important part (what the user is touching) gets the CPU's attention first.

Q: Does Selective Hydration work with the App Router?
A: Yes, it is a core part of the App Router's performance. By wrapping 'use client' components in <Suspense>, you allow Next.js to stream the HTML and React to selectively hydrate the client-side parts.

Q: What is "Event Replay"?
A: It's a key part of Selective Hydration. Since the user might click a button before the JS is ready, React captures that event, finishes the hydration of that specific button, and then "replays" the click so the user's action isn't lost.

.