🚀React Internals — Complete Senior-Level Notes

WHAT IS REACT?

React is a JavaScript library for building component-based, declarative UIs.

- **Declarative** → You describe what the UI should look like.
- **React handles** how and when to update it efficiently.
- **Component-based** → Encourages reusable, modular UI code.

WHY REACT EXISTS / WHY USE REACT

Problems React Solves:

- Manual DOM manipulation is slow and error-prone (jQuery era).
- Complex UI updates freeze the browser.
- Hard to reuse logic across components.
- Long, complex lifecycle methods in classes.

Benefits of React:

- Component reusability.
- Declarative UI.
- Virtual DOM for efficient updates.
- One-way data flow (props).
- Performance optimization with Fiber.

DECLARATIVE VS IMPERATIVE

Imperative Example:

```
document.getElementById("count").innerText = count;
```

Declarative (React) Example:

```
<h1>{count}</h1>
```

React handles **how the DOM changes**, you declare **what UI should look like**.

WHAT IS THE DOM?

- Browser represents HTML as a **tree of nodes**.
- Node types: element, text, comment.
- DOM operations are expensive: **reflow + repaint**.

WHAT IS VIRTUAL DOM?

- Lightweight JavaScript object representing UI.
- Immutable, cheap to create.
- Used for **diffing and batching updates**.
- Does **not manipulate real DOM**.

Example:

```
<h1>Hello</h1>
```

Compiles to:

```
{
  type: "h1",
  props: { children: "Hello" }
}
```

HOW REACT UPDATES UI

```
State / Props change
        ↓
Component re-renders → New React Elements (VDOM)
        ↓
Diffing / Reconciliation
        ↓
Fiber schedules updates
        ↓
Commit Phase → DOM updated
```

WHAT IS A TREE AND TREE NODE?

- **Tree** → Hierarchical data structure.
- **Node** → Single unit in the tree.

Example:

```
App
 ├─ Header
 └─ Content
     ├─ Card
     └─ Button
```

- Each element/component → tree node.

WHAT IS DIFFING?

Diffing = Comparing current UI with new UI to determine minimal changes.

Rules: - Different type → replace (`<div />` → `<span />`). - Same type → update props. - Children without keys → compared by position. - Keys define identity in lists.

WHAT IS RECONCILIATION?

Reconciliation = Diffing + determining changes + preparing for DOM. - Diffing is a subset of reconciliation. - Decides what to insert, delete, or update.

🚙OLD REACT RECONCILER (PRE-FIBER)

- Recursive traversal of component tree.
- Blocks main thread until complete.
- Cannot pause or prioritize.
- Leads to janky UI for large trees.

WHY FIBER WAS INTRODUCED

Problems Fiber Solves: - Rendering blocking → UI freezes. - Cannot pause or resume. - Cannot prioritize updates. - No support for concurrent rendering.

Fiber = solution to make reconciliation **interruptible, schedulable, and efficient**.

WHAT IS FIBER?

Fiber = React's internal data structure representing a **unit of work**.

Each component → one Fiber node.

Fiber tracks: - DOM node (`stateNode`). - Component state. - Effects (`useEffect`, etc.). - Pointers (`child`, `sibling`, `return`). - Priority. - Work to perform. - Flags (Placement, Update, Deletion).

FIBER TREE STRUCTURE

JSX:

```
<App>
  <Header />
  <Content>
    <Card />
    <Button />
  </Content>
</App>
```

Fiber Tree:

```
App (fiber)
 └─ child → Header (fiber)
 └─ sibling → Content (fiber)
       └─ child → Card (fiber)
       └─ sibling → Button (fiber)
```

Pointers: - `child` → first child. - `sibling` → next sibling. - `return` → parent.

RELATIONSHIP BETWEEN DOM, VIRTUAL DOM, AND FIBER

Concept	Represents	Notes
Virtual DOM	UI description (React Elements)	Immutable, created every render
Fiber	Internal structure for scheduling, state, effects	Mutable, holds DOM refs (`stateNode`)
DOM	Actual rendered UI	Updated in commit phase

Important: - Fiber is not a copy of Virtual DOM. - Fiber is created from Virtual DOM. - Fiber tracks updates, effects, scheduling, state, and DOM refs.

UPDATE / STATE CHANGE FLOW

1. `setState` → component re-runs.
2. New React Elements (VDOM) created.

3. Compare **Current Fiber ↔ New React Elements**.
4. Work-in-progress Fiber tree created.
5. Fiber marks flags (Placement / Update / Deletion).
6. Fiber schedules commit phase (may be paused/resumed).
7. Commit phase updates DOM.
8. WIP Fiber → becomes Current Fiber.

COMPARISON (DIFFING) IN DETAIL

- ❌Not: VDOM vs VDOM
- ❌Not: Current Fiber vs WIP Fiber
- ✅Correct: Current Fiber node ↔ New React Element → Build WIP Fiber

Flags: - Placement → insert new DOM node. - Update → update existing DOM node. - Deletion → remove DOM node.

RENDER PHASE VS COMMIT PHASE

Phase	Purpose	Characteristics
Render	Compare, build WIP Fiber	Interruptible, can pause, async
Commit	Apply DOM changes	Synchronous, must finish, triggers effects

---

DOUBLE BUFFERING

- Current Fiber Tree → represents on-screen UI.
- Work-in-progress Fiber Tree → being built.
- After commit: WIP Fiber → Current Fiber.
- Allows interruption and rollback.

PRIORITY & SCHEDULING

- High-priority updates (typing, clicks) → handled first.
- Low-priority updates (data fetch, logging) → scheduled later.

Example:

```
startTransition(() => setSearchResults(data));
```

- React marks it low-priority → smooth UI.

OLD REACT VS FIBER

Aspect	Old React	Fiber
Traversal	Recursive	Iterative
Interruptible	❌No	✅Yes
Scheduling	❌No	✅Yes
UI Blocking	Yes	No
Priority	❌No	✅Yes
Pausing	❌No	✅Yes

KEY POINTS TO REMEMBER

- Virtual DOM → describes UI.
- Fiber → tracks state, DOM nodes, scheduling, effects.
- Diffing → Current Fiber vs New React Elements.
- Work-in-progress Fiber → stores planned updates.
- Commit Phase → updates DOM synchronously.
- Double Buffering → ensures safe interruption.
- Priority & Scheduling → keeps UI responsive.

ONE-LINE SENIOR-LEVEL EXPLANATION

“React creates new React Elements on every render, compares them with the current Fiber tree to build a work-in-progress Fiber tree that marks what changes to make, and schedules updates efficiently, committing them to the DOM.”

FINAL VISUAL FLOW

```
JSX / Component Function
        ↓
New React Elements (Virtual DOM)
        ↓
Diff with Current Fiber
        ↓
Work-in-progress Fiber built (flags set)
        ↓
Fiber schedules commit
        ↓
Commit Phase → update DOM
```

```
        ↓
WIP Fiber → becomes Current Fiber
```