React Refs CheatSheet

by Ndeye Fatou Diop

Ref mistakes

- Using *ref.current* to render the JSX tree
- Accidentally running the ref callback function on every render (hint: wrap in useCallback)

What Are Refs?

- **Refs** (short for "references") are a way to:
  - access and interact with DOM nodes directly.
  - store mutable values (e.g., timers, IDs) without triggering re-renders.
- Refs are created using the **useRef()** hook (functional components) or you can pass a function to the ref attribute (i.e: **ref callback function**).
- Refs are retained by React between re-renders and changing the ref value doesn't re-renders the component.

Using Refs to Store Mutable Values Not Needed for Rendering.

```
function Timer() {
  const timerRef = useRef(0);

  const startTimer = () => {
    timerRef.current = setInterval(()
=> {
      console.log("Timer running");
    }, 1000);
  };

  const stopTimer = () => {
    clearInterval(timerRef.current);
  };

  return (
    <>
      <button onClick={startTimer}
>Start</button>
      <button onClick={stopTimer}
>Stop</button>
    </>
  );
}
```

Using Refs for DOM manipulation

- We use a ref to store the div DOM node and scroll to it when requested.

```
function ScrollToSection() {
  const sectionRef = useRef();

  const scrollToSection = () => {
    sectionRef.current.scrollIntoView({ behavior: "smooth" });
  };

  return (
    <div>
      <button onClick={scrollToSection}>Scroll to Section</
button>
      <div style={{ height: "100vh", backgroundColor: "#f0f0f0"
}}>
        <p>Scroll down</p>
      </div>
      <div
        ref={sectionRef}
        style={{ height: "100vh", backgroundColor: "#add8e6" }}
      >
        <h2>This is the section to scroll to!</h2>
      </div>
    </div>
  );
}
```

Integrating Refs with Third-Party Libraries

```
import { Chart, registerables } from "chart.js";

function ChartComponent() {
  const ref = useCallback((canvasNode) => {
    // Register Chart.js components
    Chart.register(...registerables);

    const ctx = canvasNode.getContext("2d");

    const myChart = new Chart(ctx, {
      type: "bar",
      data: {
        labels: ["Red", "Blue", "Yellow"],
        datasets: [
          {
            label: "# of Votes",
            data: [12, 19, 30],
            borderWidth: 1,
          },
        ],
      },
      options: {
        scales: {
          y: {
            beginAtZero: true,
          },
        },
      },
    });

    // Cleanup
    return () => {
      myChart.destroy();
      Chart.unregister(...registerables);
    };
  }, []);

  return <canvas ref={ref}></canvas>;
}
```