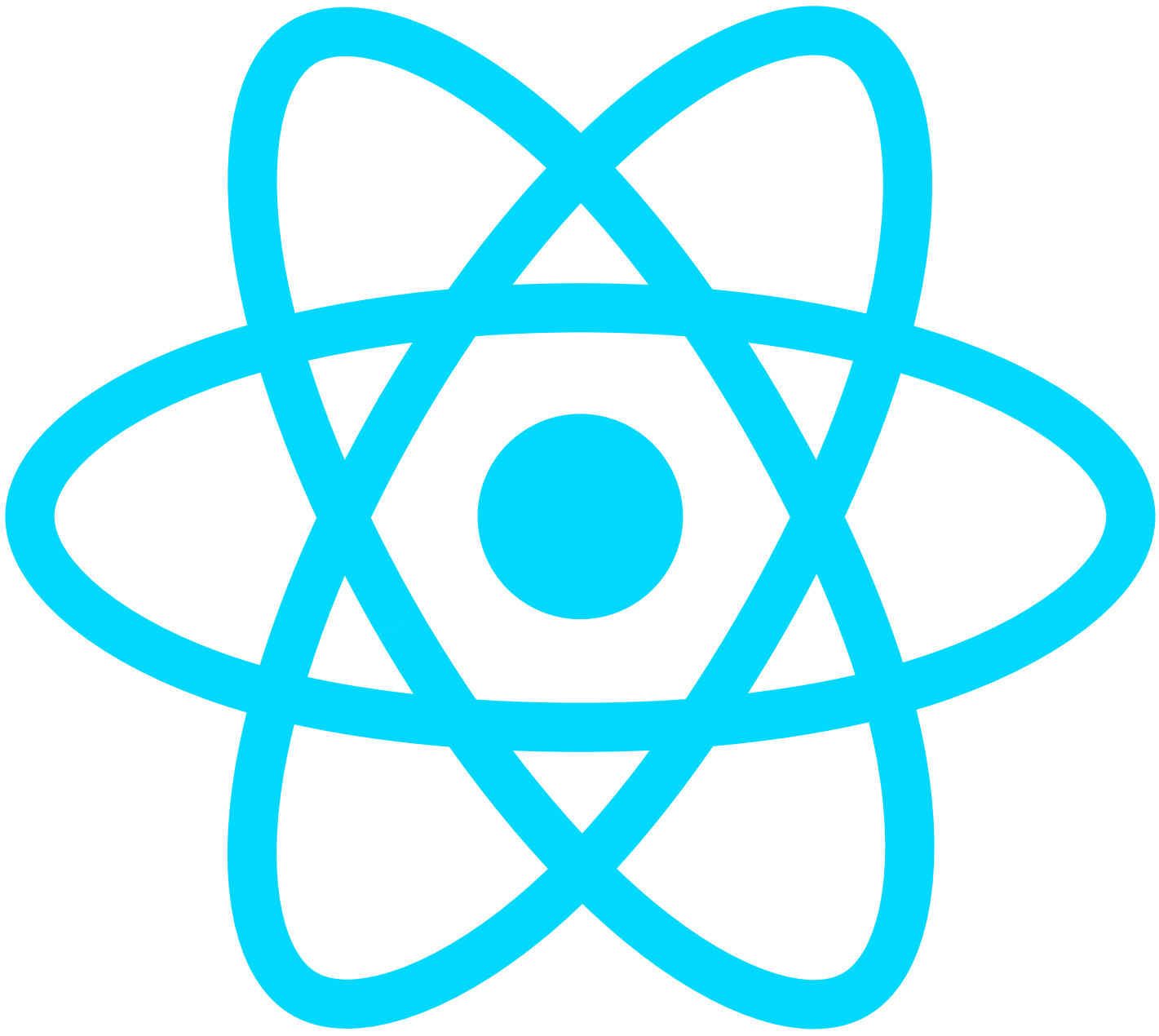

COMPLETE REACT

REACT

CORE CONCEPTS

- ***Components*: Functional and class components.**
- ***JSX*: JavaScript XML, a syntax extension for writing HTML in JavaScript.**
- ***Props*: Data passed from parent to child components.**
- ***State*: Internal data managed within a component.**
- ***Virtual DOM*: A lightweight copy of the actual DOM for efficient updates.**

REACT HOOKS

***useState*: Manage state in functional components.**

```javascript
const [count, setCount] = useState(0);
```

***useEffect*: Perform side effects (e.g., data fetching, subscriptions).**

```javascript
useEffect(() => {
  console.log("Component mounted or updated");
}, []);
```

- ***useContext*: Access context in functional components.**
- ***useReducer*: Manage complex state logic.**
- ***Custom Hooks*: Reusable logic across components.**

REACT

STATE MANAGEMENT

- ***Lifting State Up*: Share state between components by moving it to a common ancestor.**
- ***Context API:* Share state across the component tree without prop drilling.**
- ***Redux*: A predictable state container for managing global state.**

ROUTING

React Router: Declarative routing for SPAs

javascript Copy

```
<Route path="/about" component={About} />
```

PERFORMANCE OPTIMIZATION

- ***Memoization*: React.memo for functional components, PureComponent for class components.**
- ***Lazy Loading*: Load components on demand using React.lazy and Suspense.**
- ***Code Splitting*: Split code into smaller bundles for faster loading.**

ADVANCED TOPICS

- ***Error Boundaries*: Catch JavaScript errors in the component tree.**
- ***Portals*: Render children outside the DOM hierarchy.**
- ***Refs*: Access DOM nodes or React elements directly.**
- ***Higher-Order Components (HOCs)*: Reuse component logic.**

TESTING

- ***Jest*: A JavaScript testing framework.**
- ***React* Testing Library: Test React components in a user-centric way.**

STYLING

- ***CSS Modules*: Scoped CSS for components.**
- ***Styled Components*: CSS-in-JS library for styling React components.**
- ***SASS/SCSS*: Preprocessor for writing CSS.**

INTERVIEW QUESTIONS

BEGINNER-LEVEL QUESTIONS

1. **What is React, and why is it used?**
2. **What is the difference between functional and class components?**
3. **What is JSX, and how is it different from HTML?**
4. **What are props, and how are they used?**
5. **What is state, and how is it managed in React?**
6. **What is the Virtual DOM, and how does it improve performance?**
7. **How do you handle events in React?**
8. **What is the difference between state and props?**
9. **How do you conditionally render components in React?**
10. **What are keys in React, and why are they important?**

INTERMEDIATE LEVEL QUESTIONS

1. **What are React hooks, and how do you use them?**
2. **What is the purpose of useEffect?**
3. **How do you share state between components?**
4. **What is the Context API, and how is it used?**
5. **How do you implement routing in React?**
6. **What is Redux, and how does it work?**
7. **How do you optimize performance in React?**
8. **What is the difference between React.memo and PureComponent?**
9. **How do you handle forms in React?**
10. **What are error boundaries, and how do you use them?**

ADVANCE LEVEL QUESTIONS

1. **How do you create custom hooks in React?**
2. **What is the difference between useState and useReducer?**
3. **How do you implement lazy loading in React?**
4. **What are portals, and how are they used?**
5. **How do you test React components?**
6. **What are higher-order components (HOCs), and how are they used?**
7. **How do you integrate React with a backend API?**
8. **How do you handle authentication in a React application?**
9. **How do you implement internationalization (i18n) in React?**
10. **How do you use React with TypeScript?**

PRACTICAL/CODING QUESTIONS

1. **Write a React component to display a list of items.**
2. **Write a React component to handle a form with validation.**
3. **Write a React component to fetch and display data from an API.**
4. **Write a React component to implement a counter using hooks.**
5. **Write a React component to implement a toggle button.**
6. **Write a React component to implement a modal dialog.**
7. **Write a React component to implement a search filter.**
8. **Write a React component to implement a pagination control.**
9. **Write a React component to implement a drag-and-drop feature.**
10. **Write a React component to implement a dark mode toggle.**

BEHAVIORAL/SCENARIO-BASED QUESTIONS

1. **How do you debug a React application?**
2. **How do you handle performance issues in a React application?**
3. **How do you ensure code quality in a React project?**
4. **How do you handle state management in a large React application?**
5. **How do you collaborate with designers to implement React components?**