REACT INTERVIEW QUESTIONS & SOLUTIONS

1. What is React and what are its key features?

Answer: React is a JavaScript library for building user interfaces, particularly web applications.

Key Features:

- **Virtual DOM** - Efficient updates and rendering
- **Component-Based** - Reusable UI components
- **Unidirectional Data Flow** - Predictable state management
- **JSX** - JavaScript XML syntax
- **Hooks** - State and lifecycle in functional components

Example:

jsx

```jsx
import React, { useState, useEffect } from 'react';

function UserProfile({ userId }) {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchUser(userId)
      .then(userData => {
        setUser(userData);
        setLoading(false);
      });
  }, [userId]);

  if (loading) return <div>Loading...</div>;

  return (
    <div className="user-profile">
      <h2>{user.name}</h2>
      <p>Email: {user.email}</p>
      <p>Age: {user.age}</p>
    </div>
  );
}
```

2. Explain React Hooks (useState, useEffect, useContext).

Answer: Hooks allow using state and other React features in functional components.

useState:

jsx

```jsx
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);
  const [name, setName] = useState('');

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count +
1)}>Increment</button>
      <button onClick={() => setCount(prev => prev -
1)}>Decrement</button>

      <input
        value={name}
        onChange={(e) => setName(e.target.value)}
        placeholder="Enter name"
      />
    </div>
  );
}
```

useEffect:

jsx

```jsx
import React, { useState, useEffect } from 'react';

function UserList() {
  const [users, setUsers] = useState([]);
  const [filter, setFilter] = useState('');

  // Mount effect
  useEffect(() => {
    fetchUsers().then(setUsers);
  }, []);

  // Effect with dependency
  useEffect(() => {
    if (filter) {
      const filtered = users.filter(user =>
        user.name.toLowerCase().includes(filter.toLowerCase())
      );
      setUsers(filtered);
    }
```

```
  }, [filter]);

  // Cleanup effect
  useEffect(() => {
    const timer = setInterval(() => {
      console.log('Timer tick');
    }, 1000);

    return () => clearInterval(timer); // Cleanup
  }, []);

  return (
    <div>
      <input
        value={filter}
        onChange={(e) => setFilter(e.target.value)}
        placeholder="Filter users"
      />
      {users.map(user => (
        <div key={user.id}>{user.name}</div>
      ))}
    </div>
  );
}
```

useContext:

jsx

```
import React, { createContext, useContext, useState } from
'react';

// Create Context
const ThemeContext = createContext();

// Provider Component
function ThemeProvider({ children }) {
  const [theme, setTheme] = useState('light');

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}

// Consumer Component
function Header() {
  const { theme, setTheme } = useContext(ThemeContext);
```

```
  return (
    <header className={`header ${theme}`}>
      <h1>My App</h1>
      <button onClick={() => setTheme(theme === 'light' ? 'dark' :
'light')}>
        Toggle Theme
      </button>
    </header>
  );
}

// App Component
function App() {
  return (
    <ThemeProvider>
      <Header />
    </ThemeProvider>
  );
}
```

3. What is the difference between props and state?

Answer:

Feature	Props	State
Definition	Data passed from parent	Component's internal data
Mutability	Immutable	Mutable
Ownership	Owned by parent	Owned by component
Updates	Cause re-render	Cause re-render

Example:

jsx

```
// Parent Component
function App() {
  const [users, setUsers] = useState([]);

  return (
    <div>
      <UserList users={users} title="User Directory" />
      <AddUserForm onAddUser={(user) => setUsers([...users,
user])} />
    </div>
```

```
  );
}

// Child Component - receives props
function UserList({ users, title }) {
  const [searchTerm, setSearchTerm] = useState(''); // Internal
state

  const filteredUsers = users.filter(user =>
    user.name.toLowerCase().includes(searchTerm.toLowerCase())
  );

  return (
    <div>
      <h2>{title}</h2> {/* Using props */}
      <input
        value={searchTerm} {/* Using state */}
        onChange={(e) => setSearchTerm(e.target.value)}
        placeholder="Search users..."
      />
      {filteredUsers.map(user => (
        <UserCard key={user.id} user={user} />
      ))}
    </div>
  );
}
```

4. Explain React lifecycle methods and their Hook equivalents.

Answer:

Class Component Lifecycle:

jsx

```
class UserProfile extends React.Component {
  constructor(props) {
    super(props);
    this.state = { user: null, loading: true };
  }

  componentDidMount() {
    this.fetchUser();
  }

  componentDidUpdate(prevProps) {
    if (prevProps.userId !== this.props.userId) {
      this.fetchUser();
    }
```

```
  }

  componentWillUnmount() {
    // Cleanup
  }

  fetchUser = async () => {
    const user = await fetchUserData(this.props.userId);
    this.setState({ user, loading: false });
  }

  render() {
    const { user, loading } = this.state;

    if (loading) return <div>Loading...</div>;
    return <div>{user.name}</div>;
  }
}
```

Functional Component with Hooks:

jsx

```
function UserProfile({ userId }) {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  // componentDidMount + componentDidUpdate
  useEffect(() => {
    const fetchUser = async () => {
      setLoading(true);
      const userData = await fetchUserData(userId);
      setUser(userData);
      setLoading(false);
    };

    fetchUser();
  }, [userId]); // Dependencies array

  // componentWillUnmount
  useEffect(() => {
    return () => {
      // Cleanup code
      console.log('Component unmounting');
    };
  }, []);

  if (loading) return <div>Loading...</div>;
  return <div>{user.name}</div>;
```

```
}
```

5. What is React Router and how do you implement routing?

Answer: React Router enables navigation between different components/pages in a React application.

Example:

jsx

```jsx
import React from 'react';
import {
  BrowserRouter as Router,
  Routes,
  Route,
  Link,
  useParams,
  useNavigate
} from 'react-router-dom';

// Components
function Home() {
  return <h2>Home Page</h2>;
}

function About() {
  return <h2>About Page</h2>;
}

function UserProfile() {
  const { userId } = useParams();
  const navigate = useNavigate();

  return (
    <div>
      <h2>User Profile: {userId}</h2>
      <button onClick={() => navigate('/')}>Go Home</button>
    </div>
  );
}

function Users() {
  const users = ['john', 'jane', 'bob'];

  return (
    <div>
      <h2>Users</h2>
      {users.map(user => (
```

```jsx
        <div key={user}>
          <Link to={`/users/${user}`}>{user}</Link>
        </div>
      ))}
    </div>
  );
}

// Navigation Component
function Navigation() {
  return (
    <nav>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/about">About</Link></li>
        <li><Link to="/users">Users</Link></li>
      </ul>
    </nav>
  );
}

// Main App
function App() {
  return (
    <Router>
      <div>
        <Navigation />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/users" element={<Users />} />
          <Route path="/users/:userId" element={<UserProfile />}
/>
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

6. Explain state management in React (useState, useReducer, Context).

Answer:

useState for Simple State:

jsx

```
function TodoApp() {
  const [todos, setTodos] = useState([]);
  const [input, setInput] = useState('');

  const addTodo = () => {
    if (input.trim()) {
      setTodos([...todos, { id: Date.now(), text: input,
completed: false }]);
      setInput('');
    }
  };

  const toggleTodo = (id) => {
    setTodos(todos.map(todo =>
      todo.id === id ? { ...todo, completed: !todo.completed } :
todo
    ));
  };

  return (
    <div>
      <input value={input} onChange={(e) =>
setInput(e.target.value)} />
      <button onClick={addTodo}>Add Todo</button>

      {todos.map(todo => (
        <div key={todo.id}>
          <span
            style={{ textDecoration: todo.completed ?
'line-through' : 'none' }}
            onClick={() => toggleTodo(todo.id)}
          >
            {todo.text}
          </span>
        </div>
      ))}
    </div>
  );
}
```

useReducer for Complex State:

jsx

```
import React, { useReducer } from 'react';

// Reducer function
function todoReducer(state, action) {
  switch (action.type) {
```

```
    case 'ADD_TODO':
      return {
        ...state,
        todos: [...state.todos, {
          id: Date.now(),
          text: action.payload,
          completed: false
        }]
      };
    case 'TOGGLE_TODO':
      return {
        ...state,
        todos: state.todos.map(todo =>
          todo.id === action.payload
            ? { ...todo, completed: !todo.completed }
            : todo
        )
      };
    case 'DELETE_TODO':
      return {
        ...state,
        todos: state.todos.filter(todo => todo.id !==
action.payload)
      };
    case 'SET_FILTER':
      return {
        ...state,
        filter: action.payload
      };
    default:
      return state;
  }
}

function TodoAppWithReducer() {
  const [state, dispatch] = useReducer(todoReducer, {
    todos: [],
    filter: 'all'
  });

  const addTodo = (text) => {
    dispatch({ type: 'ADD_TODO', payload: text });
  };

  const toggleTodo = (id) => {
    dispatch({ type: 'TOGGLE_TODO', payload: id });
  };
```

```
  return (
    <div>
      <AddTodoForm onAdd={addTodo} />
      <TodoList todos={state.todos} onToggle={toggleTodo} />
      <FilterButtons filter={state.filter}
onFilterChange={(filter) =>
        dispatch({ type: 'SET_FILTER', payload: filter })
      } />
    </div>
  );
}
```

Context for Global State:

jsx

```
import React, { createContext, useContext, useReducer } from
'react';

// Create Context
const AppContext = createContext();

// Provider Component
function AppProvider({ children }) {
  const [state, dispatch] = useReducer(todoReducer, {
    todos: [],
    user: null,
    theme: 'light'
  });

  return (
    <AppContext.Provider value={{ state, dispatch }}>
      {children}
    </AppContext.Provider>
  );
}

// Custom Hook
function useAppState() {
  const context = useContext(AppContext);
  if (!context) {
    throw new Error('useAppState must be used within
AppProvider');
  }
  return context;
}

// Component using global state
function TodoList() {
```

```
  const { state, dispatch } = useAppState();

  return (
    <div>
      {state.todos.map(todo => (
        <TodoItem
          key={todo.id}
          todo={todo}
          onToggle={(id) => dispatch({ type: 'TOGGLE_TODO',
payload: id })}
        />
      ))}
    </div>
  );
}
```