

React.js Cheat Sheet

Components & JSX

React is a declarative library for building user interfaces with reusable components.

Functional Components & Props

```
function Greeting({ name = "Guest",
  isOnline }) {
  return (
    <div className="user-card">
      <h1>Hello, {name}!</h1>
      {isOnline ? <p>Active now</p> :
    <p>Offline</p>}
    </div>
  );
}
```

List Rendering & Fragments

```
const items = [{ id: 1, text: "Apple" },
  { id: 2, text: "Banana" }];
```

```
function ItemList() {
  return (
    <
      {items.map((item) => (
        <li key={item.id}>{item.text}</
      li>
    ))}
    </>
  );
}
```

Core React Hooks

useState (State Management)

```
const [count, setCount] = useState(0);
// Functional state updater
setCount((prev) => prev + 1);
```

useEffect (Side Effects)

```
useEffect(() => {
  const timer = setInterval(() =>
    console.log('Tick'), 1000);
  return () => clearInterval(timer); //
Cleanup function
}, [dependency]); // Dependency array
```

useRef (DOM Access & Mutable Values)

```
const inputRef = useRef(null);
const focusInput = () =>
  inputRef.current.focus();
```

useContext (Context Subscription)

```
const ThemeContext =
  createContext('light');
function ThemedButton() {
  const theme =
    useContext(ThemeContext);
  return <button className={theme}>
    Button</button>;
}
```

useReducer (Complex State Logic)

```
function reducer(state, action) {
  switch (action.type) {
    case 'increment': return { count:
    state.count + 1 };
    default: return state;
  }
}
const [state, dispatch] =
  useReducer(reducer, { count: 0 });
// dispatch({ type: 'increment' })
```

Performance Optimization

- **useMemo**: Caches expensive computation results

```
const expensiveResult = useMemo(() =>
  compute(data), [data]);
```

- **useCallback**: Memoizes callback function definitions

```
const handleClick = useCallback(() =>
  {
    doSomething(id);
  }, [id]);
```

- **React.memo**: Skips re-renders when props are unchanged

```
const PureItem = React.memo(function
  Item({ title }) { ... });
```

Modern React Features

- **useTransition**: Marks non-urgent state updates as transitions

```
const [isPending, startTransition] =
  useTransition();
startTransition(() =>
  setSearchQuery(input));
```

- **useDeferredValue**: Defers updating a value to keep UI responsive

```
const deferredQuery =
  useDeferredValue(searchQuery);
```

- **Suspense & React.lazy**: Code splitting and async rendering

```
const HeavyChart = React.lazy(() =>
  import('./HeavyChart'));
<Suspense fallback={<Spinner />}>
  <HeavyChart />
</Suspense>
```

Event Handling & Forms

- **Synthetic Events**: **onClick**, **onChange**, **onSubmit**

- **Form Submission Handling**:

```
const handleSubmit = (e) => {
  e.preventDefault();
  const formData = new
  FormData(e.target);
};
```