React Essentials Cheatsheet

■ COMPONENT BASICS
- React components are just functions that return JSX.
- Example: function Navbar() { return <nav>Bloom</nav> }
- Must start with a capital letter.

■ JSX
- Looks like HTML but is actually JavaScript.
- Can embed JS inside {}.

```
const name = 'Bloom';
return <h1>Hello {name}</h1>
```

■ PROPS
- Props = inputs to components (like function parameters).
- Example: function Button({ label }) { return <button>{label}</button> }
- Use object destructuring to pull out props.

■ CHILDREN
- Special prop for nested content.

```
function Card({ children }) { return <div>{children}</div> }
<Card><h2>Title</h2><p>Body</p></Card>
```

■ STATE
- useState stores local state inside a component.

```
import { useState } from 'react'
const [count, setCount] = useState(0)
<button onClick={() => setCount(count + 1)}>Count: {count}</button>
```

- Changing state re-renders the component.

■ EVENTS
- Event handlers like onClick, onChange.

```
function SearchBar() {
  const handleChange = (e) => console.log(e.target.value)
  return <input onChange={handleChange} />
}
```

■ CONDITIONAL RENDERING
- Show elements only if conditions are true.

```
{isLoggedIn ? <p>Welcome</p> : <p>Please log in</p>}
```

■ RENDERING LISTS
- Use map() to render arrays as elements.

```
const items = ['apple','banana']
<ul>{items.map((f, i) => <li key={i}>{f}</li>)}</ul>
```

- Always add a unique key to each item.

■ useEffect (SIDE EFFECTS)

- Runs code after render (e.g. fetch data).
  import { useEffect } from 'react'
  useEffect(() => { console.log('Component mounted') }, [])

■ COMPONENT LIFECYCLE WITH useEffect

- [] = run once on mount.
- [dep] = run when dependency changes.
- No array = run after every render.

■ NEXT.JS ROUTING

- File-based: app/about/page.tsx => /about
- Layouts wrap all pages with shared UI (Navbar, Footer).

■ BEST PRACTICES

- Keep components small and focused.
- Use props for flexibility, state for dynamic behavior.
- Break UI into reusable building blocks.
- Start simple, add complexity later.