`** React Hooks **`

- ✓ State
- ✓ Lifecycle Methods
- ✓ Sharing Non-Visual Logic

```
// Custom Hook -- to share logic between Components
function useRepos(id) {
  const [repos, setRepos] = React.useState([]); // Local State
  const [loading, setLoading] = React.useState(true); // Local State

  React.useEffect(() => { // componentDidMount & componentDidUpdate

    setLoading(true);

    fetchRepos(id).then(repos => {
      setRepos(repos);
      setLoading(false);
    });
  }, [id]);

  return [loading, repos];
}
```

Rules of Hooks

- Only call `Hooks` from the top-level of a `function component` or a `custom Hook`
- can't call them anywhere that's not on the top level like inside of a `loop` , `if statement` , or `event handler`

```
function Counter () {
  // 👍 from the top level function component
  const [count, setCount] = React.useState(0)

  if (count % 2 === 0) {
    // 👎 not from the top level
    React.useEffect(() => {})
  }

  const handleIncrement = () => {
    setCount((c) => c + 1)

    // 👎 not from the top level
    React.useEffect(() => {})
  }

  ...
}

function useAuthed () {
  // 👍 from the top level of a custom Hook
  const [authed, setAuthed] = React.useState(false)
}

class Counter extends React.Component {
  render () {
    // 👎 from inside a Class component
    const [count, setCount] = React.useState(0)
  }
}

function getUser () {
  // 👎 from inside a normal function
  const [user, setUser] = React.useState(null)
}
```