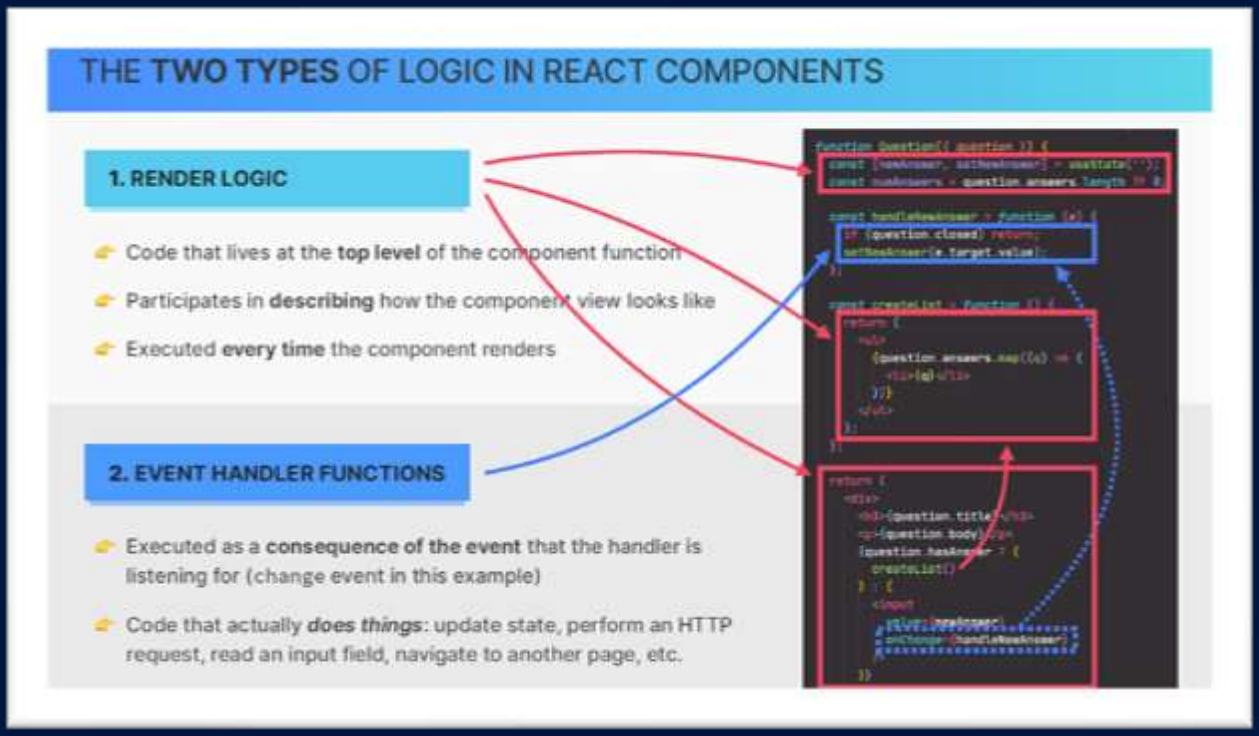
React

PURE COMPONENTS



REFRESHER: FUNCTIONAL PROGRAMMING PRINCIPLES

- **Side effect:** dependency on or modification of any data outside the function scope. *"Interaction with the outside world"*. **Examples:** mutating external variables, HTTP requests, writing to DOM.

> 👋 **Side effects are not bad!** A program can only be useful if it has some interaction with the outside world

- **Pure function:** a function that has **no** side effects.
  - Does **not** change any variables outside its scope
  - Given the **same input**, a pure function always returns the **same output**

✅ Pure function

```
function circleArea(r) {
  return 3.14 * r * r;
}
```

☝️ Impure function

```
const areas = {};

function circleArea(r) {
  areas.circle = 3.14 * r * r;
}
```

Side effect: Outside variable mutation

☝️ Impure function

```
function circleArea(r) {
  const date = Date.now();
  const area = 3.14 * r * r;
  return `${date}: ${area}`;
}
```

Unpredictable output (date changes)

RULES FOR RENDER LOGIC

- ☝️ **Components must be pure when it comes to render logic:** given the same props (input), a component instance should always return the same JSX (output)
- ☝️ **Render logic must produce no side effects:** no interaction with the "outside world" is allowed. So, in render logic:
  - Do NOT perform **network requests** (API calls)
  - Do NOT start **timers**
  - Do NOT directly **use the DOM API**
  - Do NOT **mutate objects or variables** outside of the function scope
  - Do NOT **update state (or refs)**: this will create an infinite loop!

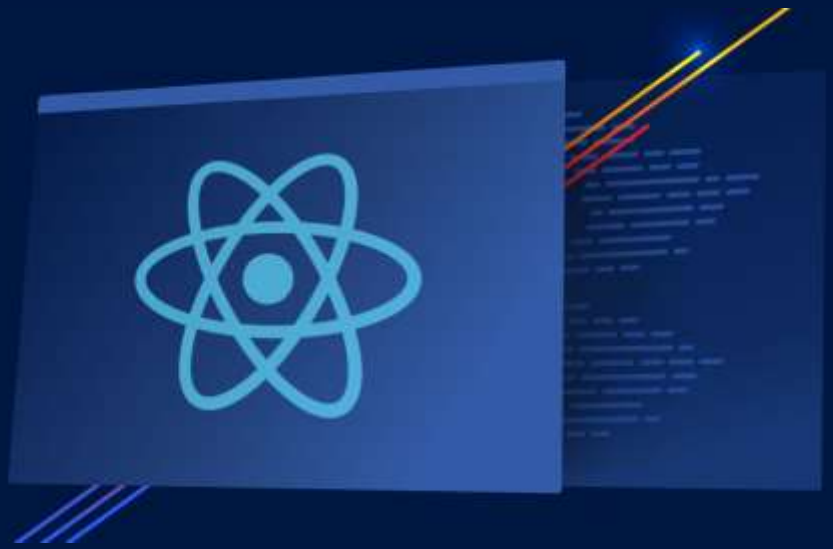
This is why we can't mutate props!

> 👋 Side effects are allowed (and encouraged) in **event handler functions!** There is also a special hook to **register side effects** (`useEffect`)

React

Do you find it helpful?

Let me know down in the comments



Click To Follow For More On LinkedIN