Hooks

Hooks are essential when working with Functional Components
Most frequentely used hooks: **useState, useEffect.**

useState:
const [name, setName] = useState('Tom')

Is used to as a hook to state, this state will be tracked and your page re-render whenever the state is changed.

useEffect(() => {
//do stuff
}, [**dependacyArray**])

- Most commonly used with a dependacyArray
- Replaces the need to use LifeCycle hooks
- It is designed to fire effects, what are effects? Some kind of event that will change your state, API calls, click events

Hooks - useEffect dependacy array rules:

useEffect(() => {}, [dependencyArray])

1. **An empty dependency** array will execute the code inside the useEffect on the FIRST render of the component, so only once!
2. **No dependency array** will execute the code whenever ANY state is changed – this will likely cause problems and most likely lead to INFINITE LOOPS 💀 💀, at least in my experience this has happened
3. **A dependency array with a dependency inside** will fire useEffect ONLY AFTER THAT DEPENDENCY HAS BEEN CHANGED, this is the most common use case for **useEffect**. Eg, something has changed and we need to re-render the component to show that change ✓✓✓✓.