2.2 Optimizing Performance by Skipping Effects

In some cases, executing the effect after every render creates a performance problem, It can be optimized by **Skipping** the **effect** when it is not required

The

`useEffect` accepts another argument called **Dependency Array**, using which we can control the execution of effect

Syntax:

```jsx
useEffect(effect, [var1, var2, ...])
```

- The values we pass to the dependency array are called dependencies, which can be state variables or props
- Effect will be executed whenever the dependencies are changed

Without Dependencies

The Effect executes after every render

```jsx
useEffect(effect);
```

Passing Empty Dependency Array

The Effect executes only once after the first render

```jsx
useEffect(effect, []);
```

Passing Dependencies

The Effect executes when dependencies are changed

```jsx
useEffect(effect, [var1, var2,...])
```

File: src/components/BmiCalculator/index.js

```jsx
...
const BmiCalculator = () => {
  ...
  useEffect(() => {
    document.title = `Your BMI: ${getBmi(weight, height)}`
  }, [height, weight])

  useEffect(() => {
    localStorage.setItem("height", JSON.stringify(height))
  }, [height])

  useEffect(() => {
   localStorage.setItem("weight", JSON.stringify(weight))
  }, [weight])
  ...
}

export default BmiCalculator
```

Collapse

3. BmiCalculator Application Code

Use the below command to get the Final Code for BmiCalculator Application

SHELL

```
ccbp start RHSIVTWEU5
```

S