# Updating the State Cheat Sheet

## Rules for Updating state

1. Never update the state directly, always use **setState()**
2. For any code that needs to run after the state update, put it in a setState **callback function.**
3. Use a callback function in setState with **prevState** if you need to update state based on previous state values

## Part 1: Updating the state using setState()

1. Add the onClick attribute to the button to listen for the click event

```
{/* Step 1: Set the button onClick action */}
<button onClick = {() =>
this.handleClick()}>{this.state.buttonText}</button>
```

2. Write an event handler to run for every click event:

```
handleClick(){
       // Step 2: Create the handleClick function
   }
```

3. Use setState() in the event handler to update the state

```
handleClick(){
       // Step 3: Use the this.setState() Method to set the state
       this.setState({
           introduction : "Goodbye",
           buttonText: "Enter",
       });
       console.log(this.state.introduction);
       console.log(this.state.buttonText);
   }
```

**Note:** A call to setState is asynchronous, or in other words the console.log statement in the code above will run before the setState method has completed. Therefore you may also need to add a callback function to make it synchronous (make console logs run after setState).

## Part 2: Updating the state with Callback function

1. Add an arrow function as a second parameter to your handleClick method, or in other words, use a callback function:

```
handleClick(){
       this.setState({
           introduction: "Goodbye",
           buttonText: "Enter",
        }, () => { // Use a call back function to ensure the code is
run synchronously with setState() method
           console.log(this.state.introduction);
           console.log(this.state.buttonText);
       });
   }
```

## Part 3: Updating the state with ternary conditionals

1. Use a ternary conditional statement to check for previous state in the setState method.

```
handleClick() {
   this.setState({
   // Use a ternary conditional statement to add a Previous State
     introduction: this.state.introduction === "Hello" ? "Goodbye" :
"Hello",
     buttonText: this.state.buttonText === "Exit" ? "Enter" : "Exit",
   }, () => {
     console.log(this.state.introduction);
     console.log(this.state.buttonText);
   });
 }
```

**Note:** This will create a simple check back-and-forth switch between the 2 values above. Problems occur, however, if you need to call this function multiple times. React groups such calls and omits the extra functions being called.

## Part 4: Updating the state with prevState

1. Add an arrow function to the this.setState method which takes prevState and prevProps.

```
handleClick() {
 // Add an arrow function with prevState and prevProps as parameters
 this.setState((prevState, prevProps) => {
   return {

   }
 })
}
```

2. Call on prevState or prevProps when returning a value

```
handleClick() {
 this.setState((prevState, prevProps) => {
   console.log("Previous State: ", prevState);
   console.log("Previous Props: ", prevProps);
   return {
     // Use prevState or prevProps to call on state
     introduction: prevState.introduction === "Hello" ? "Goodbye" :
"Hello",
     buttonText: prevState.buttonText === "Exit" ? "Enter" : "Exit",
   }
 })
}
```