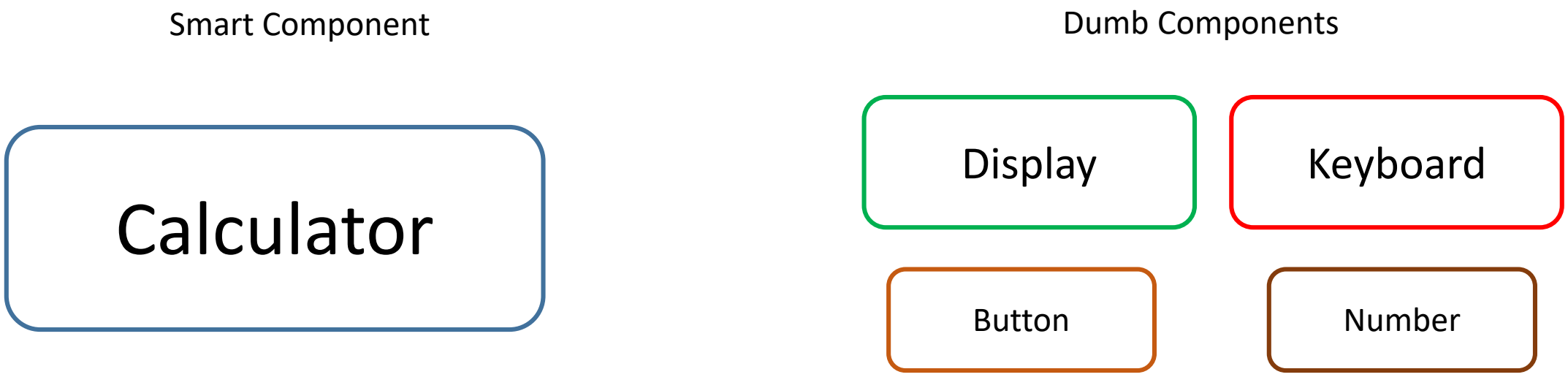
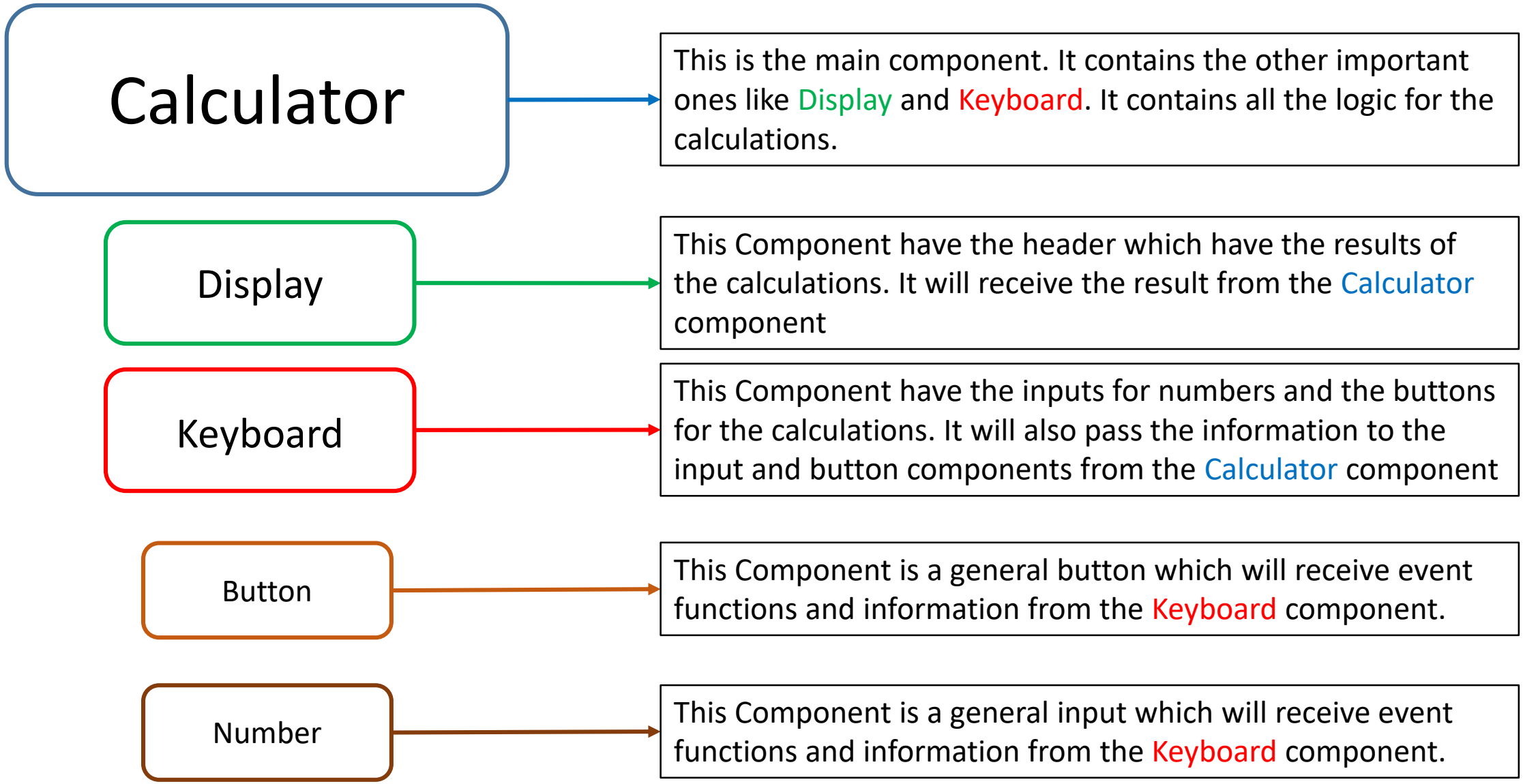
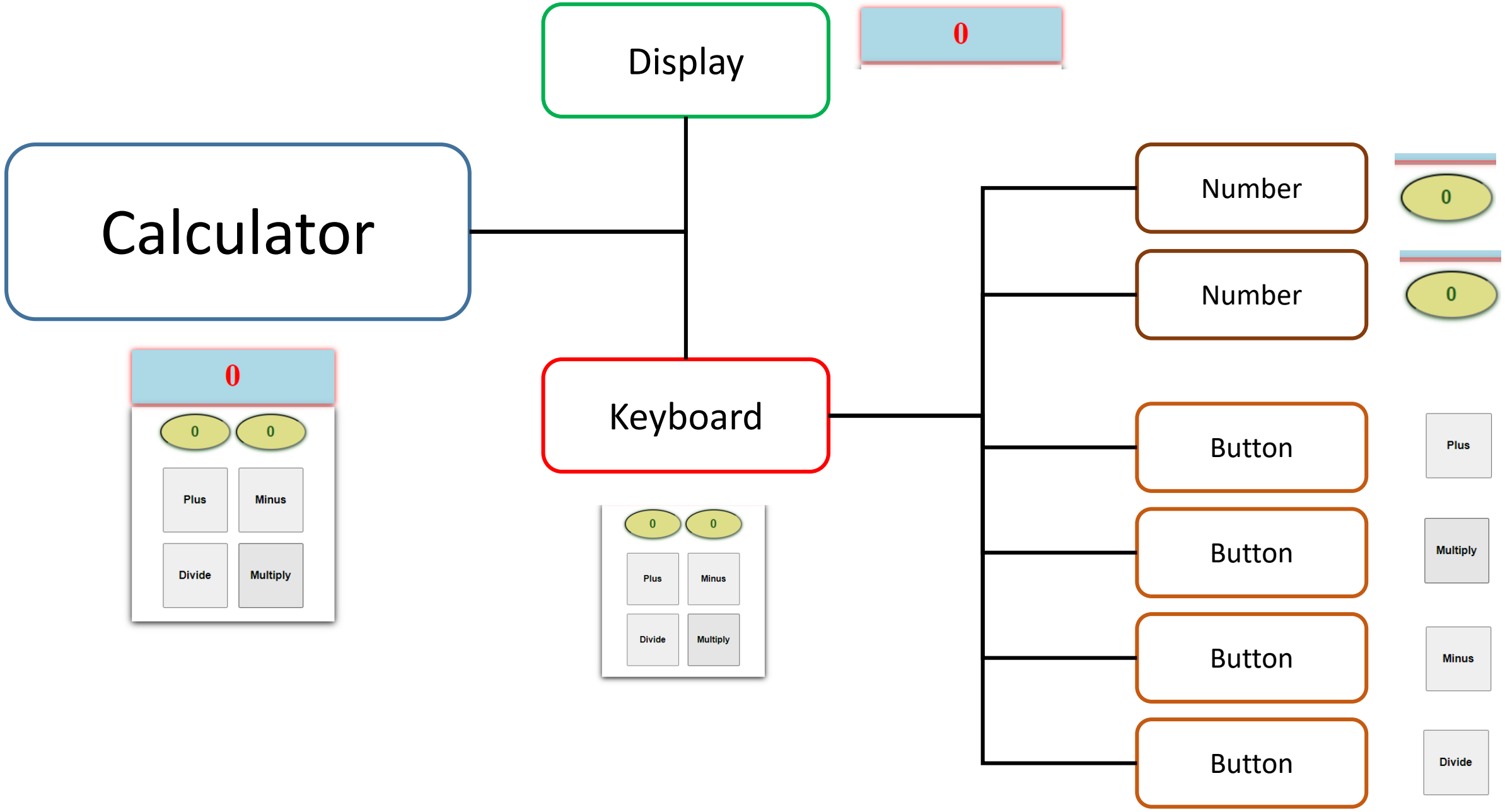
List of Components

- Calculator
  - Display
  - Keyboard
    - Button
    - Number

In react is good to have a division between components. This division is between „dumb“ and „smart“ components. What we mean by that is that is that one component will have all the logic (smart) and the other components will be only for display (dumb). In this exercise the Calculator component will be the „smart“ components and the others will be our „dumb“ components




Calculator
This is the main component. It contains the other important ones like Display and Keyboard. It contains all the logic for the calculations.
Display
This Component have the header which have the results of the calculations. It will receive the result from the Calculator component
Keyboard
This Component have the inputs for numbers and the buttons for the calculations. It will also pass the information to the input and button components from the Calculator component
Button
This Component is a general button which will receive event functions and information from the Keyboard component.
Number
This Component is a general input which will receive event functions and information from the Keyboard component.


Calculator
Display
0
Keyboard
0
0
Plus
Minus
Divide
Multiply
0
0
Plus
Minus
Divide
Multiply
Number
0
Number
0
Button
Plus
Button
Multiply
Button
Minus
Button
Divide

Calculator

```
class Calculator extends Component {

    state={
        num1:0,
        num2:0,
        result:0
    }

    onChangeFunc = (event) => {···
    }

    onClickFunc = (event) =>{···
    }
```

We need this state as we need to keep record of some variables in the app. The variables that will be changing are the inputs of the calculator and the result of the calculations. Then the state will have num 1 as the value of the first number/input and num2 for the second number/input. The result property will store the result of the calculations.

This onChangeFunc is a method from our class that will be used to read the values from the inputs. This method will be lately pass as a „prop“ to the dumb components. Also the value from the inputs received will be saved in the state.

This onClickFunc is a method to realize a certain operation according to which button was clicked. The method will be passed as a „prop“ to the Keyboard and then Button components. Also the method will perform the respective calculation and save the result in the state. S

NOTE: If you have questions about the code inside the methods please ask me directly in slack.

Calculator

Here we are sending the result property of the state to the Display component

```
render() {
    return(
        <React.Fragment>
            <Display result={this.state.result} />
            <KeyBoard onchangeFunc={this.onChangeFunc} num1={this.state.num1} num2={this.state.num2} onclickFunc={this.onClickFunc}/>
        </React.Fragment>
    )
}
```

Here we pass the onChangeFun method to the Keyboard component.

Here we pass the property num1 from the state to the Keyboard component

Here we pass the property num1 from the state to the Keyboard component

Here we pass the onClickFunc method to the Keyboard component.

Display

```
import React, { Component } from 'react'
import './Display.css'
export class Display extends Component {
    render() {
        return (
            <header>
                <h3 id="result">{this.props.result}</h3>
            </header>
        )
    }
}

export default Display
```

Here we are using the result that we obtain from the Calculator component

Keyboard

Here we pass as a „prop“ the respective name to the Number component.

Here we pass as a „prop“ the respective id to the Number component. This is needed for the css grid design

Here first we use „this.props“ to acces the props passed, specifically the num1 or num2 variables comming from the Calculator component state. Now we pass this also as a „prop“ to the Number component. Keep in mind that to each Number we pass a different number num1 or num2.

```
<Number name="firstNumber" id="firstNumber" number={this.props.num1} onchangeFunc={this.props.onchangeFunc}/>
<Number name="secondNumber" id="secondNumber" number={this.props.num2} onchangeFunc={this.props.onchangeFunc}/>

<Button classNprop="btn1" name="Plus" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn2" name="Minus" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn3" name="Divide" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn4" name="Multiply" onclickFunc={this.props.onclickFunc}/>
```

Here first we use „this.props“ to acces the props passed, specifically the onchangeFunc method obtained from the Calculator component.
Now we pass this also as a „prop“ to the Number component. As you may notice we pass the same prop to all Number components. That is ok because in the method we have a switch statement that help us differentiate between the inputs thanks to the name that we assign to each Number component.

Keyboard

```
<Number name="firstNumber" id="firstNumber" number={this.props.num1} onchangeFunc={this.props.onchangeFunc}/>
<Number name="secondNumber" id="secondNumber" number={this.props.num2} onchangeFunc={this.props.onchangeFunc}/>

<Button classNprop="btn1" name="Plus" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn2" name="Minus" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn3" name="Divide" onclickFunc={this.props.onclickFunc}/>
<Button classNprop="btn4" name="Multiply" onclickFunc={this.props.onclickFunc}/>
```

Here we assign the respective css class as a „prop“ to the Button component. As you see each button has a different css class as this is how it was designed so it have the correct position on the grid.

Here we pass as a „prop“ the respective name to the Button component depending on which operation we want.

Here first we use „this.props“ to acces the props passed, specifically the onclickFunc method obtained from the Calculator component.
Now we pass this also as a „prop“ to the Button component.
As you may notice we pass the same prop to all Button components. That is ok because in the method we have a switch statement that help us differentiate between the buttons thanks to the name that we assign to each Button.

Number

```
export class Number extends Component {

    render() {
        return (
            <input
                type="text"
                name={this.props.name}
                id={this.props.id}
                onChange={this.props.onchangeFunc}
                placeholder="Insert a Number"
                value={this.props.number}
            />
        )
    }
}
```

Here we are using the name that we obtain from the Keyboard component as a „prop“. Here we use it for the name of the input

Here we are using the id that we obtain from the Keyboard component as a „prop“. Here we use it for the id of the input.

Here we are using the onchangeFunc method that we obtain from the Keyboard component as a „prop“. And remember that this comes directly from the Calculator component. Here we assign this method to the „onChange“ event of the input.

Here we are using the number that we obtain from the Keyboard component as a „prop“. Remember that this comes directly from the state of the Calculator component. We assign it as the value for our input. More info on this later.

Button

```
export class Button extends Component {
    render() {
        return (
            <button
                className={this.props.classNprop}
                onClick={this.props.onclickFunc}
                name={this.props.name}
                >
                    {this.props.name}
            </button>
        )
    }
}
```

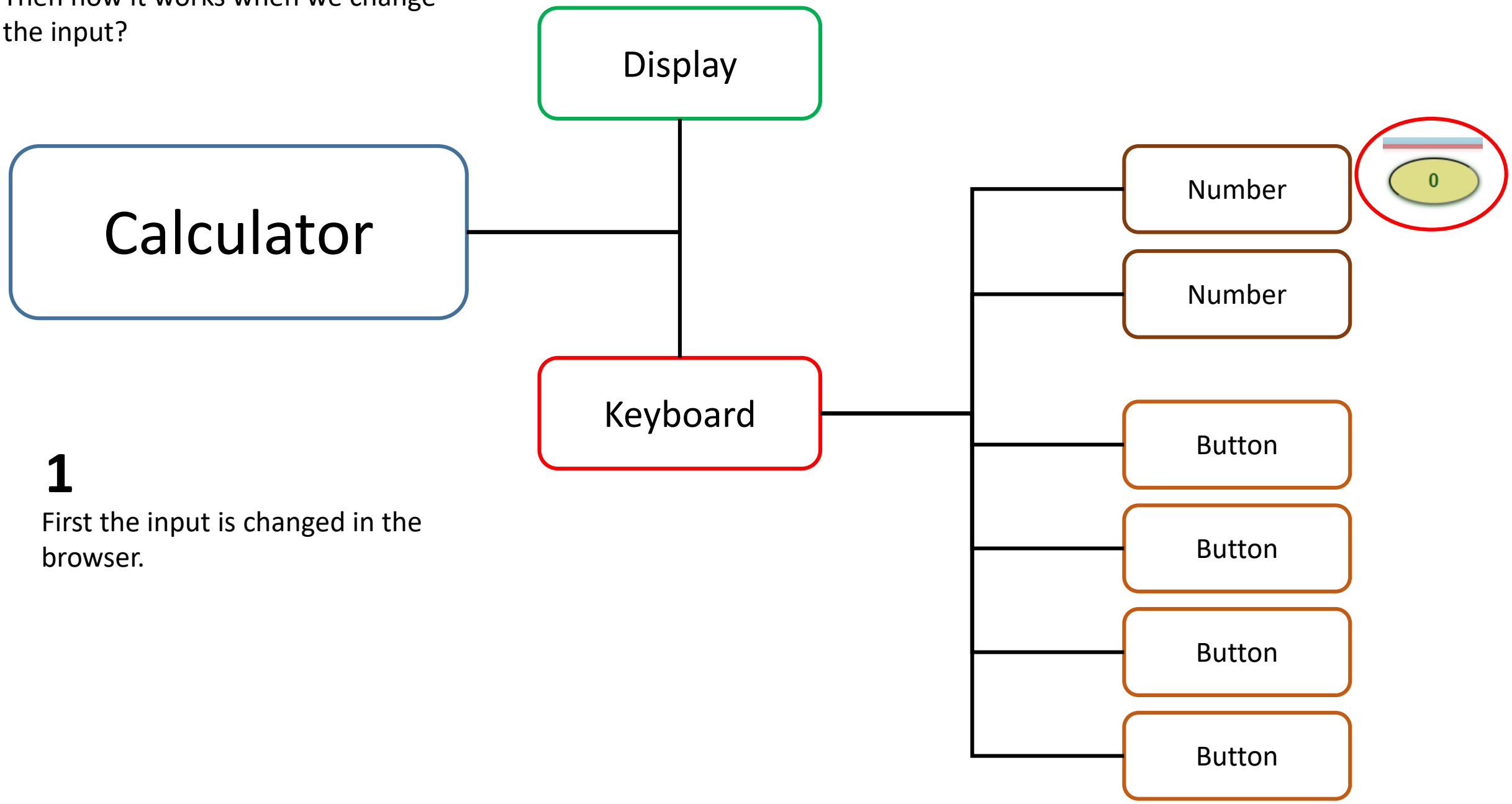
Here we are using the classNprop that we obtain from the Keyboard component as a „prop“. This property contains the css class for the button

Here we are using the onclickFunc method that we obtain from the Keyboard component as a „prop“. And remember that this comes directly from the Calculator component. This method is assigned to the „onClick“ event from the button

Here we are using the classNprop that we obtain from the Keyboard component as a „prop“. This property contains the name for the button

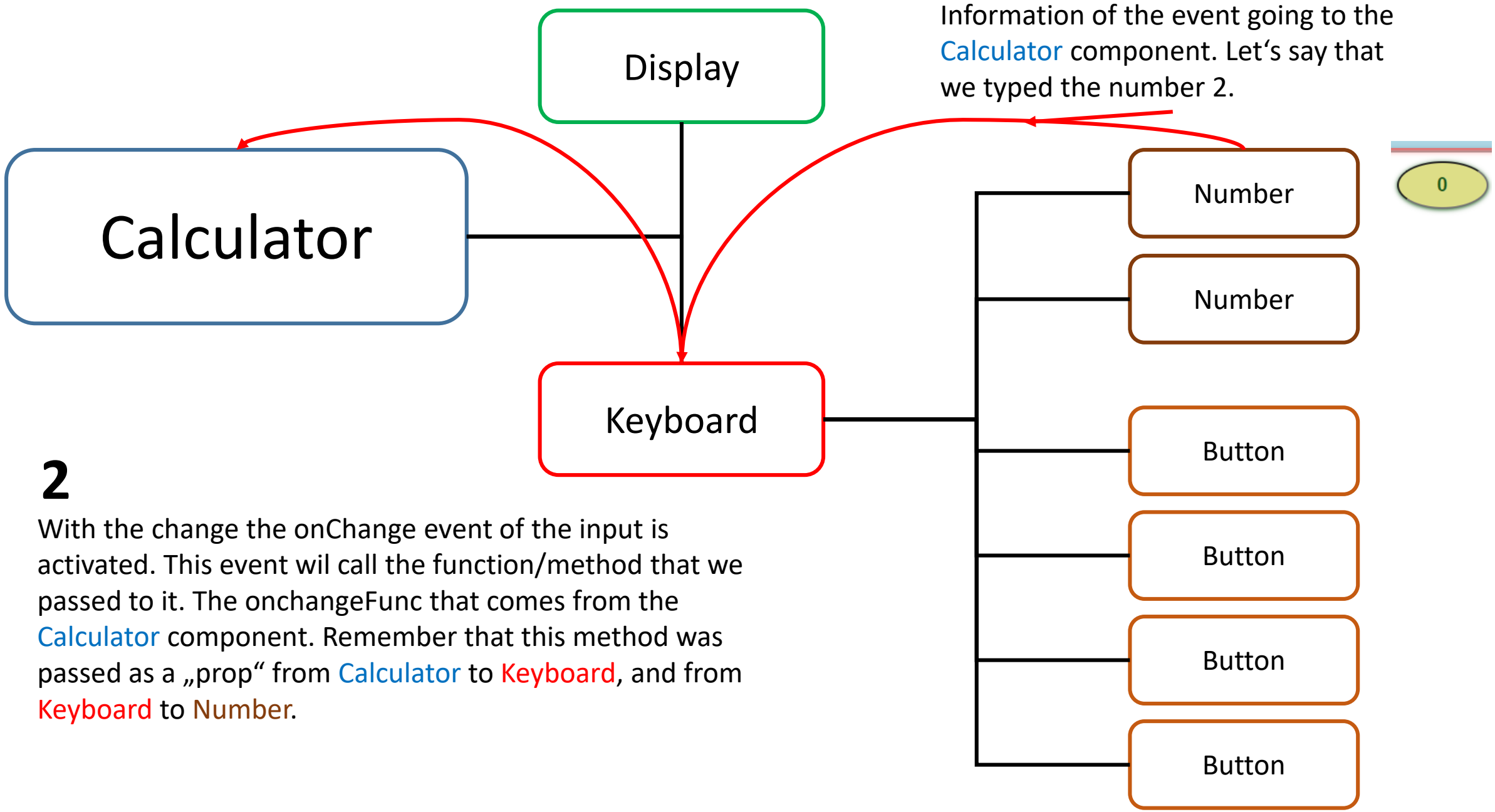
Here we are using the classNprop that we obtain from the Keyboard component as a „prop“. Here we are assigning it to the innerHTML of the button.

Then how it works when we change the input?



1

First the input is changed in the browser.



2

With the change the onChange event of the input is activated. This event wil call the function/method that we passed to it. The onchangeFunc that comes from the Calculator component. Remember that this method was passed as a „prop“ from Calculator to Keyboard, and from Keyboard to Number.

this method was passed as a „prop“

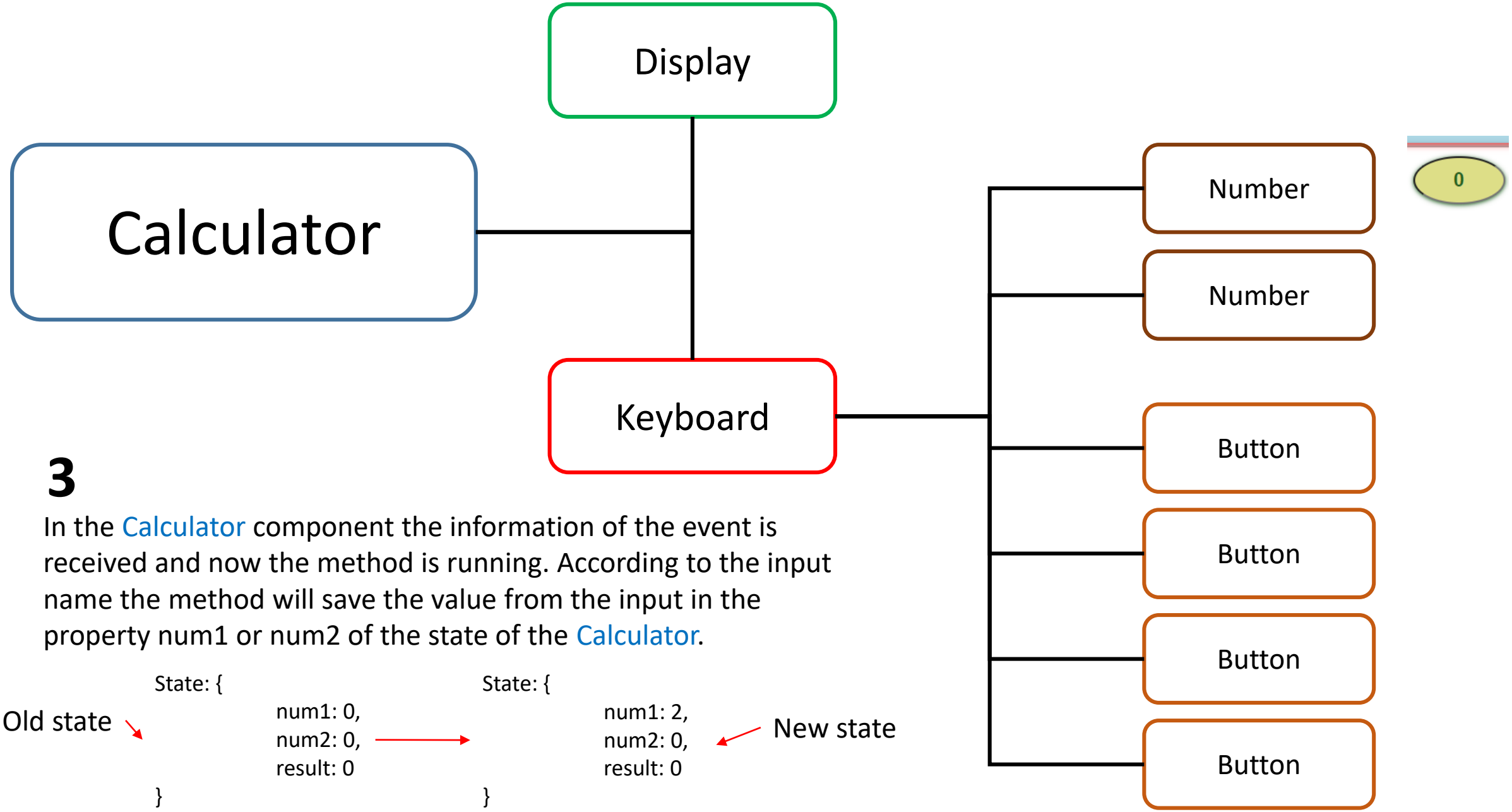
```
<React.Fragment>
    <Display result={this.state.result} />
    <KeyBoard onchangeFunc={this.onChangeFunc} num1={this
</React.Fragment>
```

from Calculator to Keyboard

```
<Number name="firstNumber" id="firstNumber" number={this.props.num1} onchangeFunc={this.props.onchangeFunc}/>
```

and from Keyboard to Number

Then the information of the event does the inverse trip, from Number to Keyboard and from Keyboard to Calculator.



3

In the Calculator component the information of the event is received and now the method is running. According to the input name the method will save the value from the input in the property num1 or num2 of the state of the Calculator.

Old state

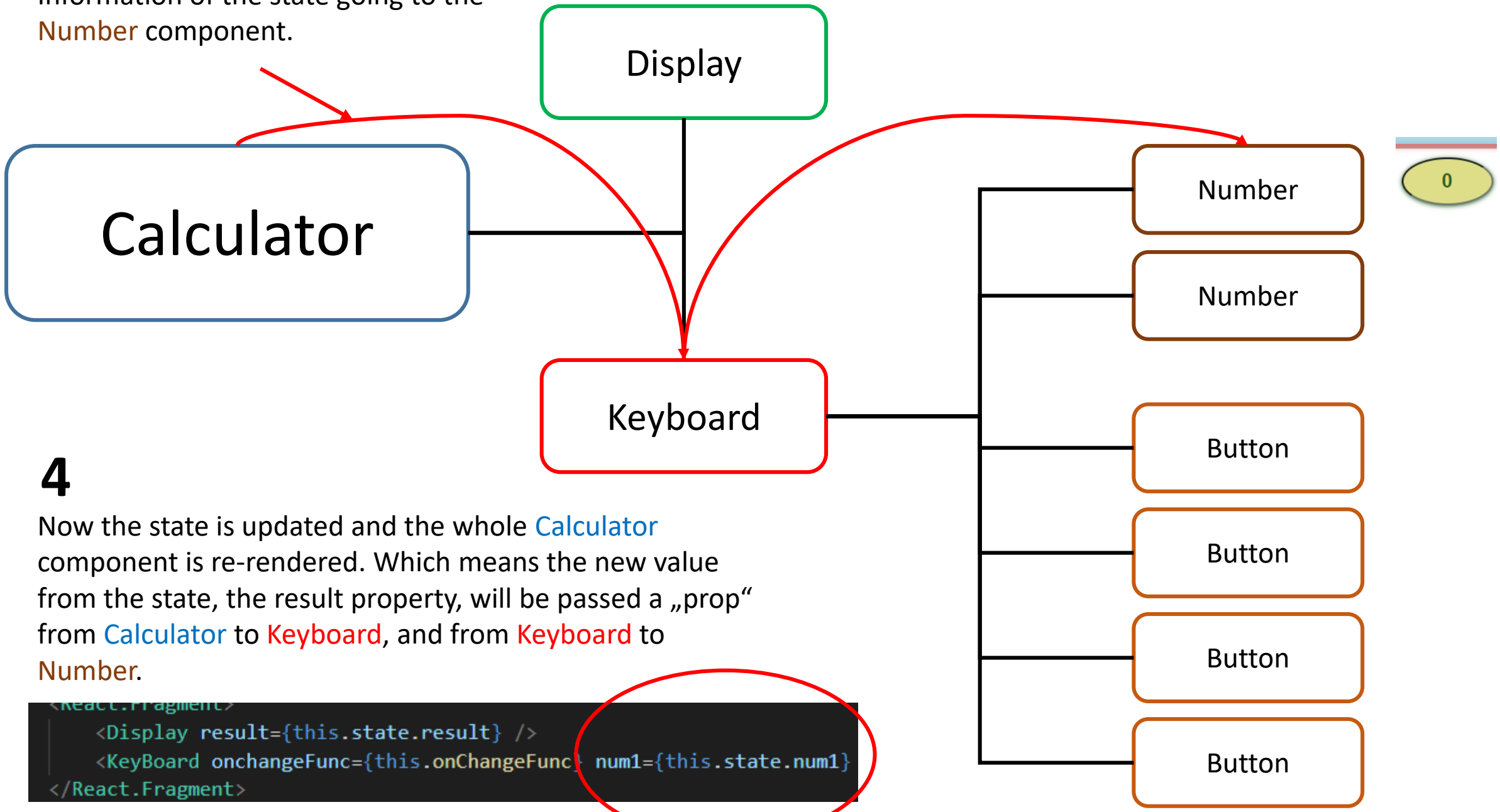
```
State: {
        num1: 0,
        num2: 0,
        result: 0
}
```

→

```
State: {
        num1: 2,
        num2: 0,
        result: 0
}
```

New state

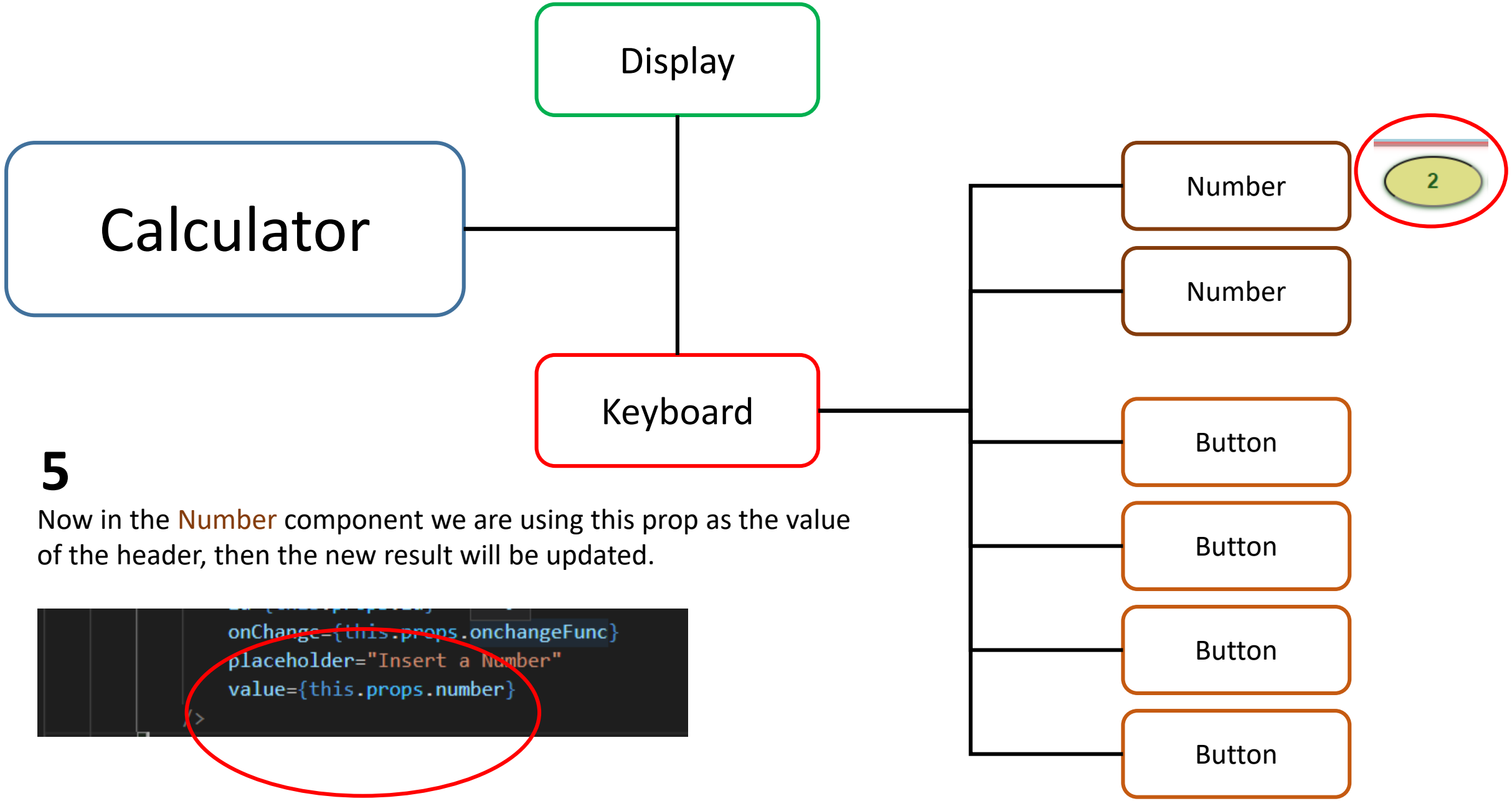
Information of the state going to the Number component.



4

Now the state is updated and the whole Calculator component is re-rendered. Which means the new value from the state, the result property, will be passed a „prop“ from Calculator to Keyboard, and from Keyboard to Number.

```
<React.Fragment>
    <Display result={this.state.result} />
    <KeyBoard onchangeFunc={this.onChangeFunc} num1={this.state.num1}
</React.Fragment>
```

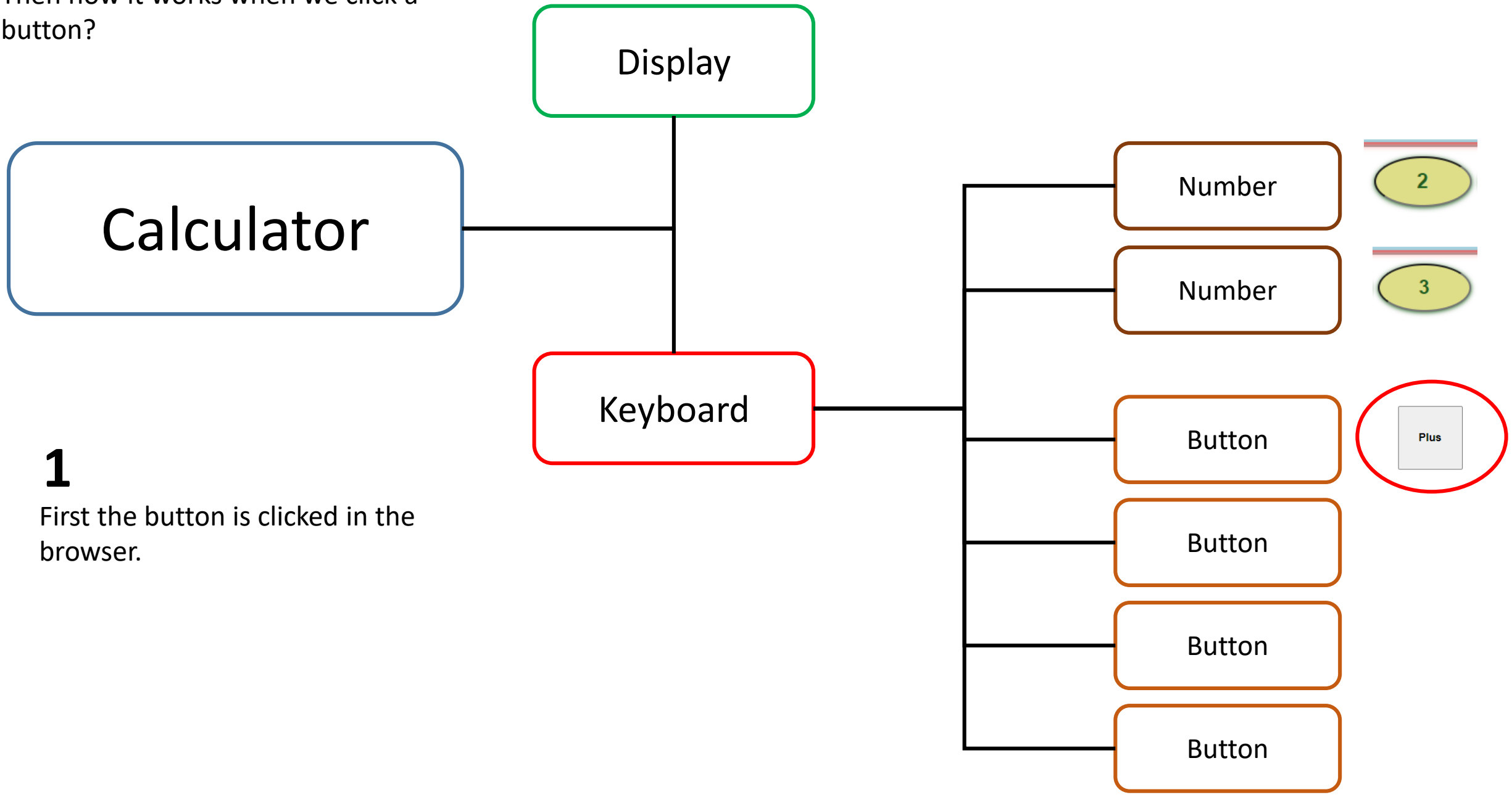


5

Now in the Number component we are using this prop as the value of the header, then the new result will be updated.

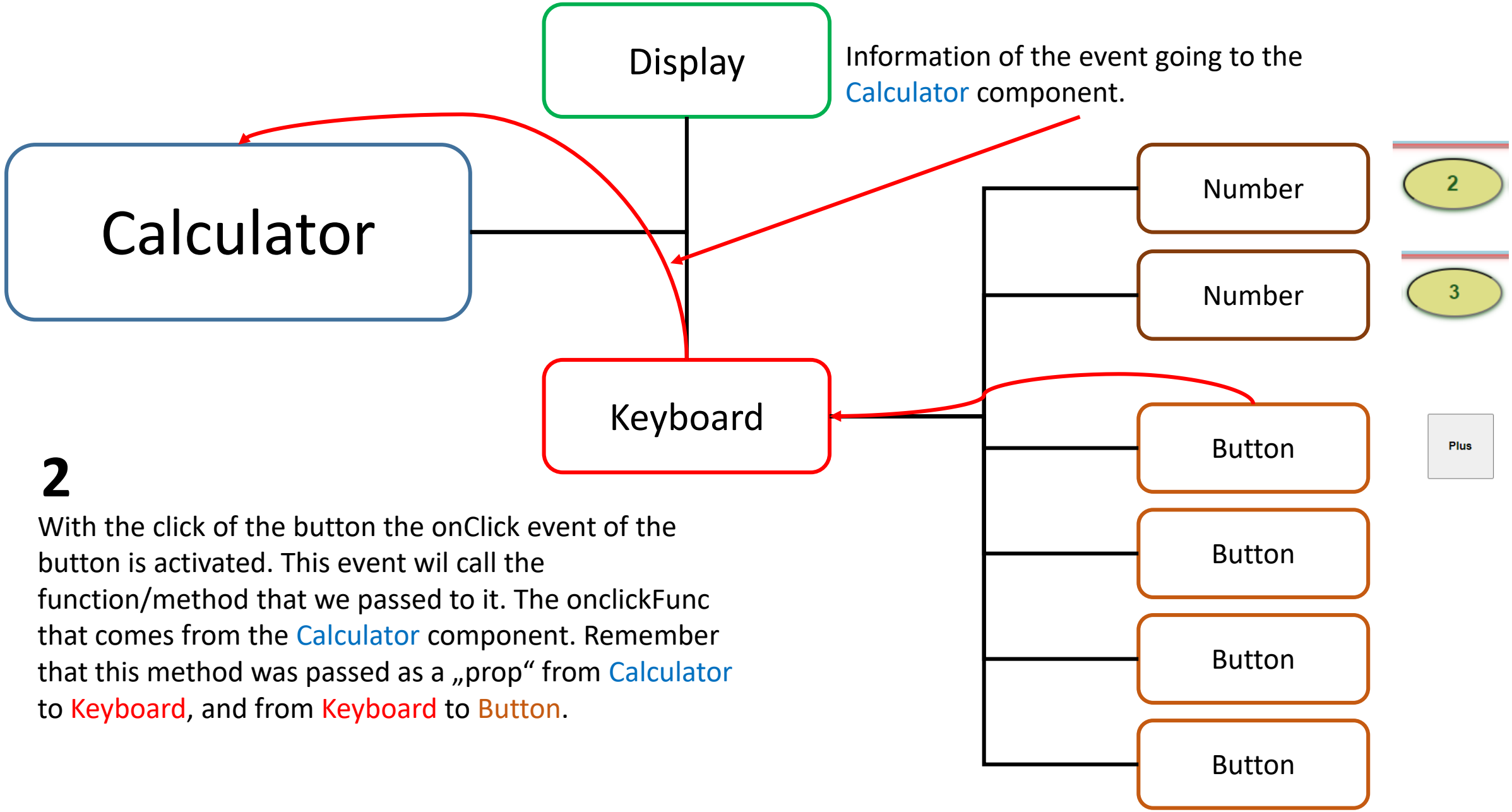
```
onChange={this.props.onchangeFunc}
placeholder="Insert a Number"
value={this.props.number}
/>
```

Then how it works when we click a button?



1

First the button is clicked in the browser.



2

With the click of the button the onClick event of the button is activated. This event wil call the function/method that we passed to it. The onclickFunc that comes from the Calculator component. Remember that this method was passed as a „prop“ from Calculator to Keyboard, and from Keyboard to Button.

this method was passed as a „prop“

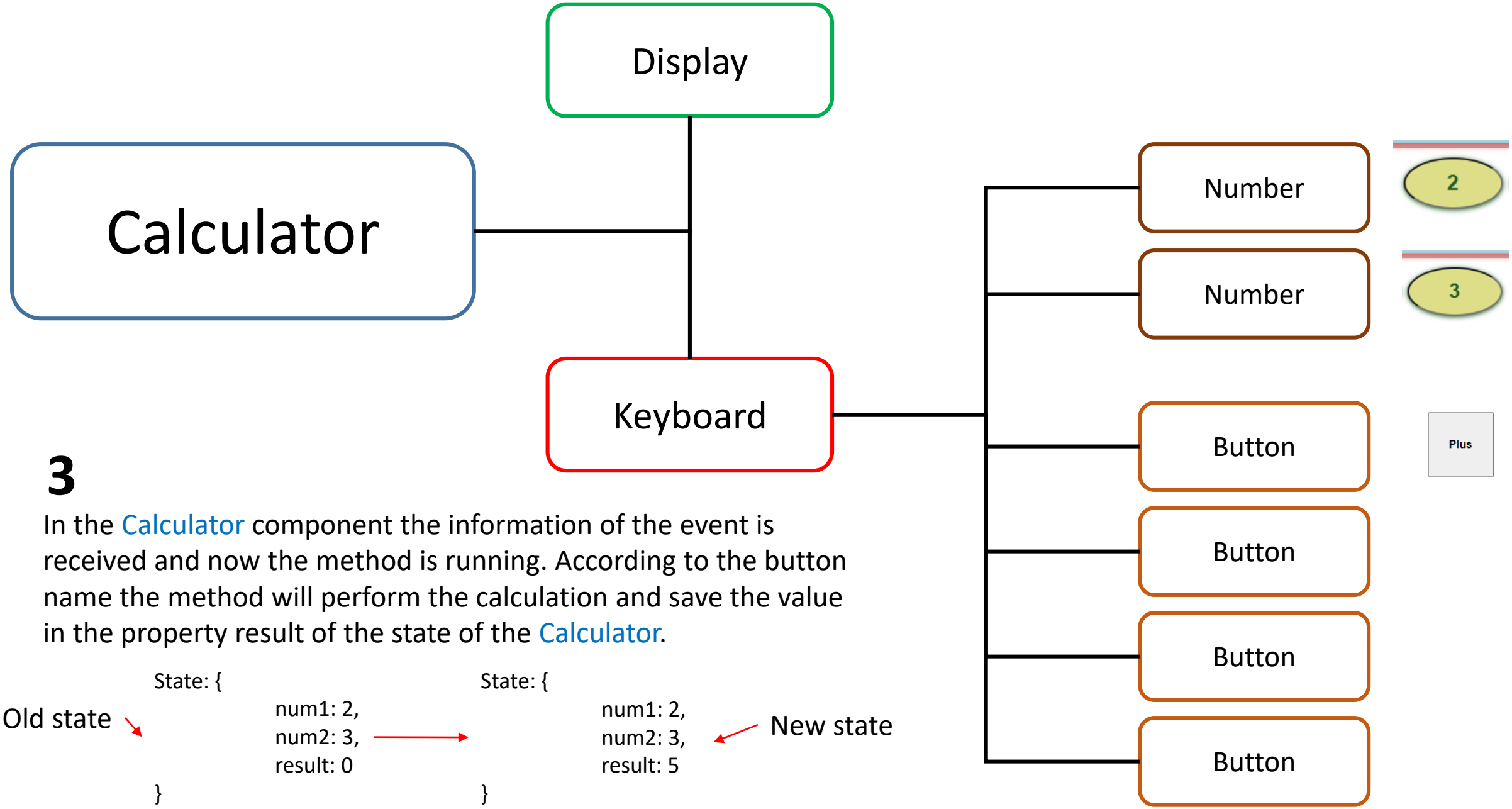
```
  <Display result={this.state.result} />
  <KeyBoard onchangeFunc={this.onChangeFunc} num1={this.state.num1} num2={this.state.num2} onclickFunc={this.onClickFunc}/>
React.Fragment>
```

from Calculator to Keyboard

```
<Button classNprop="btn1" name="Plus" onclickFunc={this.props.onclickFunc}/>
```

and from Keyboard to Button

Then the information of the event does the inverse trip, from Button to Keyboard and from Keyboard to Calculator.



3

In the Calculator component the information of the event is received and now the method is running. According to the button name the method will perform the calculation and save the value in the property result of the state of the Calculator.

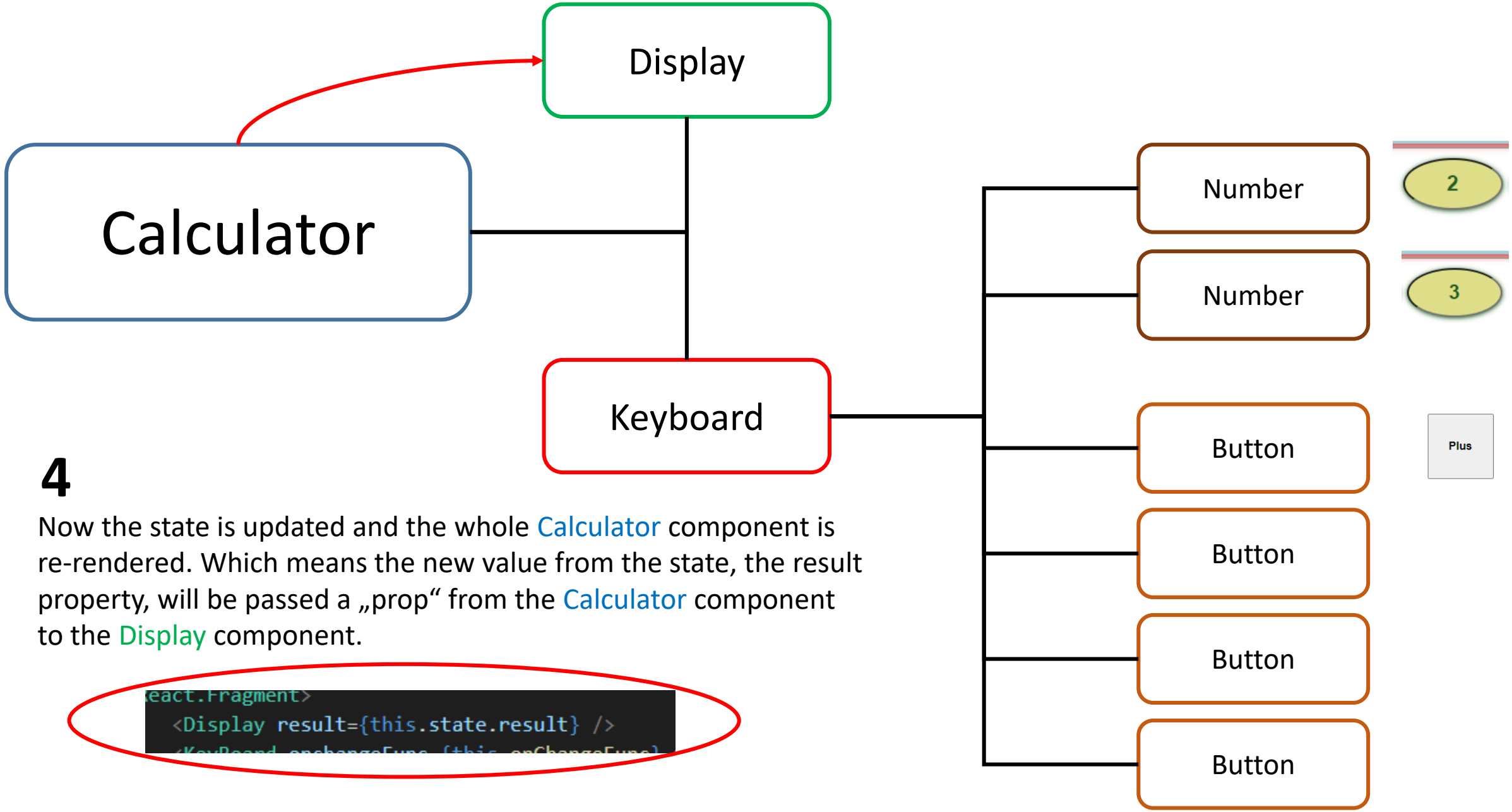
Old state

```
State: {
        num1: 2,
        num2: 3,
        result: 0
}
```

→

```
State: {
        num1: 2,
        num2: 3,
        result: 5
}
```

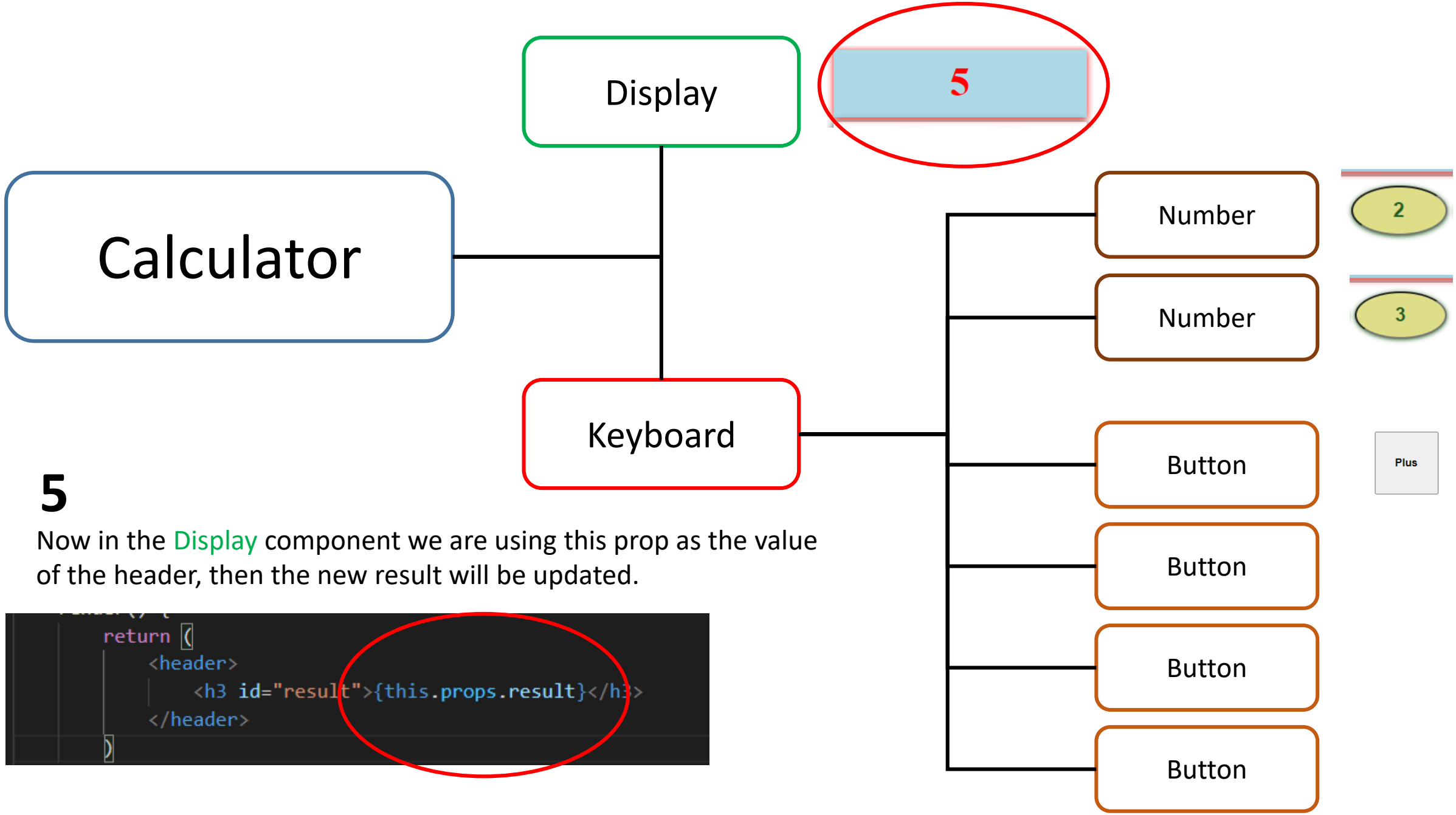
New state



4

Now the state is updated and the whole Calculator component is re-rendered. Which means the new value from the state, the result property, will be passed a „prop“ from the Calculator component to the Display component.

```
eact.Fragment>
  <Display result={this.state.result} />
```



5

Now in the Display component we are using this prop as the value of the header, then the new result will be updated.

```
return (
    <header>
        <h3 id="result">{this.props.result}</h3>
    </header>
)
```

Quick question... Why do we need to set the value in the input before? I mean this:

```
export class Number extends Component {

    render() {
        return (
            <input
                type="text"
                name={this.props.name}
                id={this.props.id}
                onChange={this.props.onchangeFunc}
                placeholder="Insert a Number"
                value={this.props.number}
            />
        )
    }
}
```

Well... It has it reasons.

First, the main idea of React is to control the view with the state, then if we want to follow this we need to save the value of the input in the state.
Also we would like to use this value of the input to do the calculations, then at the end of the day we need to save and what better place than the state. Then this explains why we want the onChange event... But what about the input? Well there are 3 cases:

- We put a value property with a static value like in the image. Well if we do this then we would not be able to change the value of the input. The state will change, but the value of the inpot wont.

```
    type="text"
    name={this.props.name}
    id={this.props.id}
    onChange={this.props.onchangeFunc}
    placeholder="Insert a Number"
    value= "0"
/>
```

- Then what if we dont use any value property? Like in this image. Well if we do this it will work... But it will be not the best practice and the input will be „uncontrolled“ like the input could have a value and the state another which could lead to problems in the web application. (NOTE: don`t confuse this with the „uncontrolled binding“ when you search on google, that is another topic)

```
<input
    type="text"
    name={this.props.name}
    id={this.props.id}
    onChange={this.props.onchangeFunc}
    placeholder="Insert a Number"
/>
```

Then the third case is as we have it now, with the value related to the state.

```
<input
    type="text"
    name={this.props.name}
    id={this.props.id}
    onChange={this.props.onchangeFunc}
    placeholder="Insert a Number"
    value={this.props.number}
/>
```

This is the best because then when the component re-renders we can sure that the value of the input and the state is the same. Now we have the control of the input and everything will be stable :) . You can see the image to the right to have a general idea of the flow of information (if it is still not clear with the previous diagram)