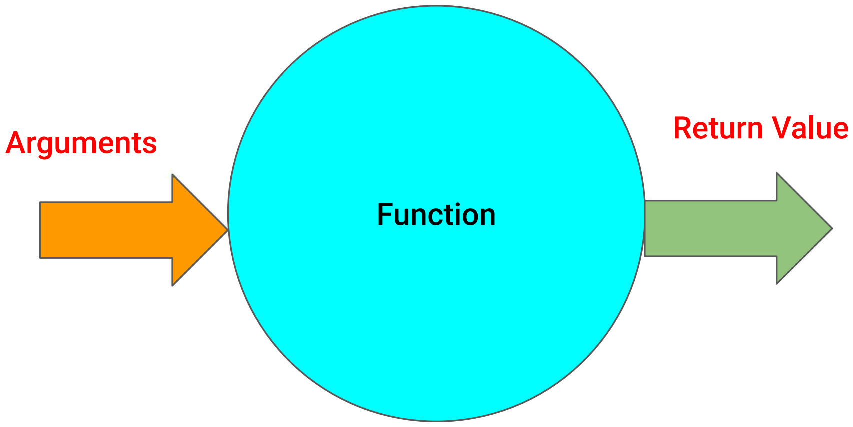
# UI Component and React

Dr Harshad Prajapati
30 Nov 2023

## Function: Let’s Understand What a Function Does



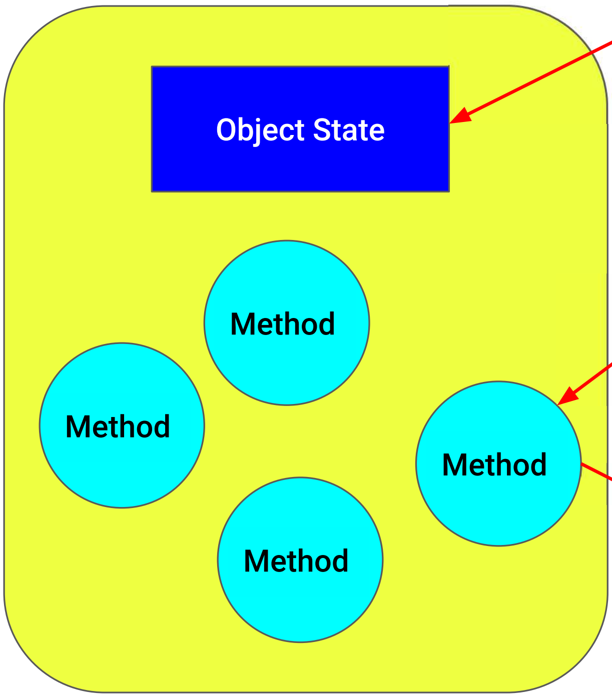
Output will change as input changes

Example:
sortedStudentList = sort(studentList)

# Object

Representation in Software
(Student object)

Real world object
(Student)



If the state of the object changes (for example, age and semester), we need to update its view to the outside world

Some method can provide view of the object to outside world.

For example toString():

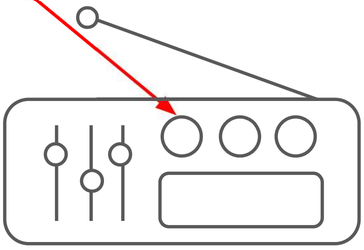
Information about Tom
Age: 20
Sem: 6

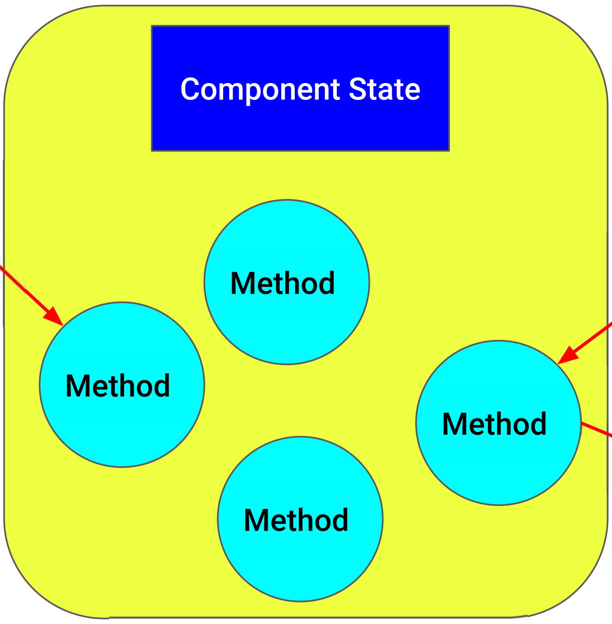
# Component in General

Some methods can provide changing component state.

For example: event handler.

Knob (event handler) for Radio



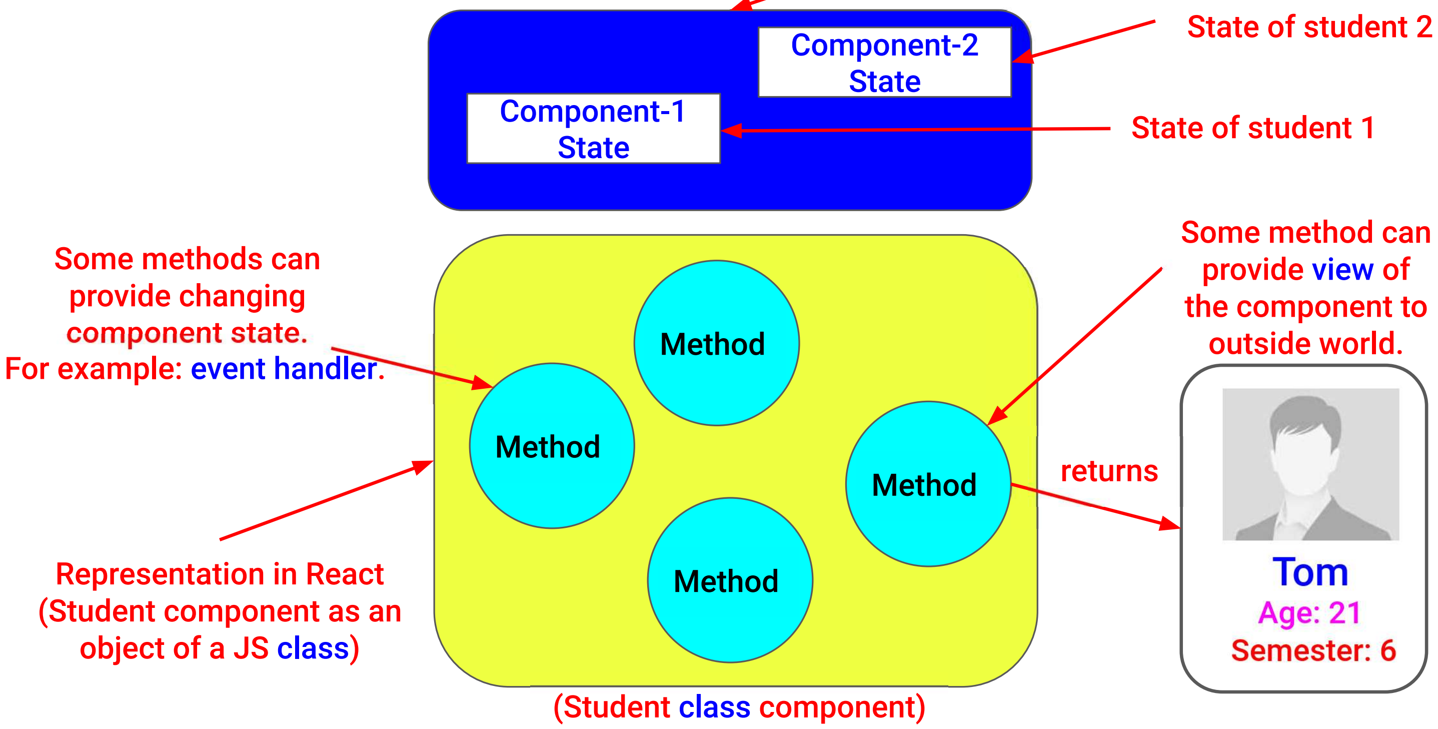


Representation in Software
(Student component)

Some method can provide view of the component to outside world.

Tom
Age: 21
Semester: 6

# What is a React Component



React Library

Component-2 State

Component-1 State

State of student 2

State of student 1

Some methods can provide changing component state. For example: event handler.

Method

Method

Method

Method

Some method can provide view of the component to outside world.

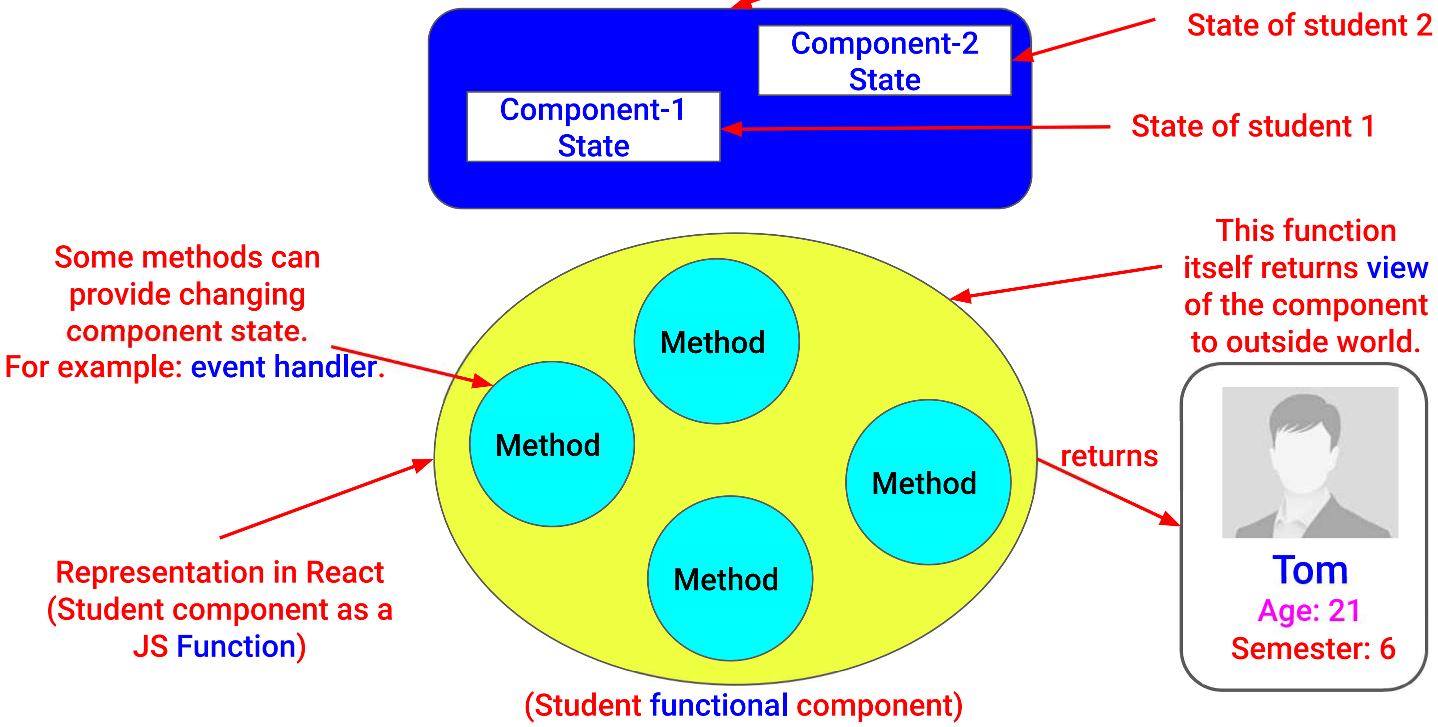
returns

Tom
Age: 21
Semester: 6

Representation in React (Student component as an object of a JS class)

(Student class component)

# What is a React Component



React Library

Component-2 State

Component-1 State

State of student 2

State of student 1

Some methods can provide changing component state. For example: event handler.

Method

Method

Method

Method

This function itself returns view of the component to outside world.

returns

Tom
Age: 21
Semester: 6

Representation in React (Student component as a JS Function)

(Student functional component)

# Instantiate Object versus Component

If there is a class named Student, we instantiate it using the following:

```
student = new Student(“Tom”, 21, 6);
```

When we want to display student object, we use its toString() method.

If there is a component named Student, we instantiate it using the following:

```
<Student name=“Tom” age=21 sem=6 />
```

When we want to display student component, we use render() method of class component or return value of functional component.

# View of Object versus UI Component for Outside World

Return value of method (toString()):

```
“Information about” + “Tom” + “\n” +
“\tAge:” + 20 + “\tSem:” + 6
```

This is what method returns

```
Information about Tom
    Age: 20
    Sem: 6
```

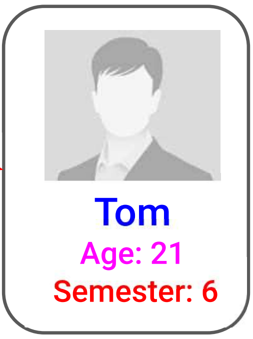
This is what is printed

This is what component returns

Return value of component:

```
<div ...>
        <img src=“...” width=“...” height=“...” .../>
        <div ...>Tom</div>
        <div ...>Age: 21</div>
        <div ...>Semester: 6</div>
</div
```

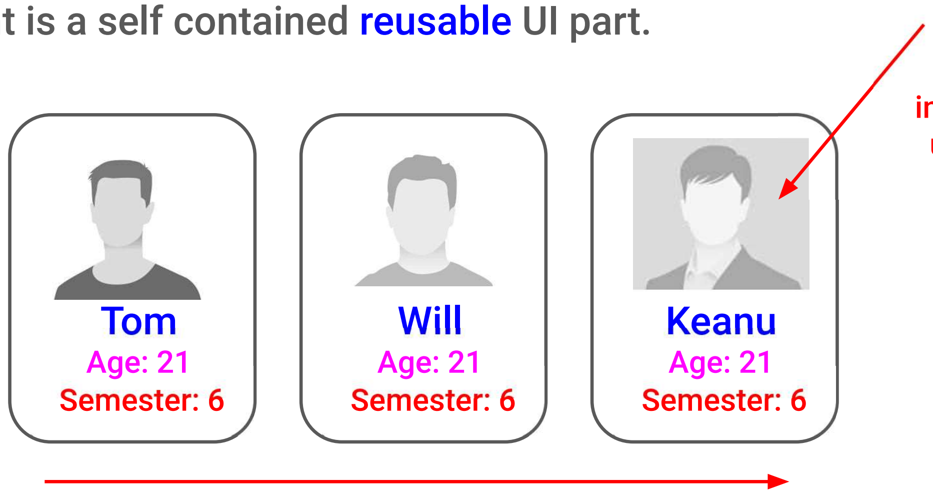
This is what component looks like



# What else a React Component can do?

- A component is a self contained **reusable** UI part.

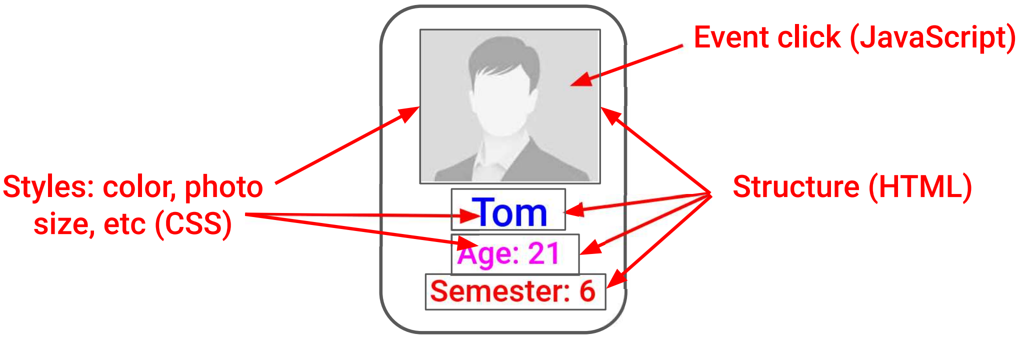
Component can allow a user to **interact** with itself. For example, change profile picture by interacting with it using keyboard /touch/mouse



We define a Student component once and then can create multiple instances of the component for different students.

# React Component

- A React component is composed of:
  - A template (HTML)
  - Interactivity (JavaScript)
  - Styles (CSS)



# Separation of Data and View

Keanu

Age: 21

Semester: 6

View

```
<div …>
        <img src={imageFile} width="…"
height="…" …/>
        <div …>{studentName}</div>
        <div …>Age: {studentAge}</div>
        <div …>Semester: {studentSem}</div>
</div
```

Data

```
imageFile= "…"
studentName=Tom
studentAge=21
studentSem=6
```

# Data Related to Component

- A React component can have **dynamic information** in **two forms**:
    - **props** (Properties): Initialization values.
    - **state**: State is within the component.
- Props:
    - **Props** are **arguments** passed into **React component**, similar to arguments passed to a function.
    - Props are **immutable**.
- State:
    - State is **mutable**.
    - The **state object** can contain **multiple properties**.

# Props (Initialization values)

- Information that is passed from one component to another is called as props in React.
- A component's props is an object and it holds initialization information about the component.
- These props are transferred to the component through the constructor of the class component and through function arguments in functional component.
- We can access props object inside a component using this.props.<propName> in class component or props.<propName> in a functional component.

# What is a React Component with Props

- We can pass data as props to the component.
- Suppose, the student component is defined (class or function) and is named as Student.
- And Student component is used in StudentList component.

The parent component (StudentList) can pass values to three props (name, age, and sem)

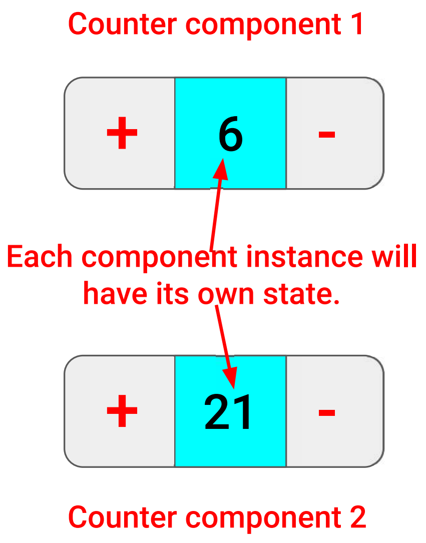
```
< div … >
      <Student name="Keanu" age=21 sem=6 />
      <Student name="Tom" age=21 sem=6 />
</div>
```

Each component instance will have its own props.

Student component

# What is a React Component with State

- The state is not passed from the outside. A component decides its own state.
- To create state, we need to assign an object (called state object) to this.state property in class component OR using hook (useState) in a functional component.
- For example,
    - When we click + or - button, the counter gets incremented or decremented.
    - This count value is the state of the component.



Counter component 1

Each component instance will have its own state.

Counter component 2

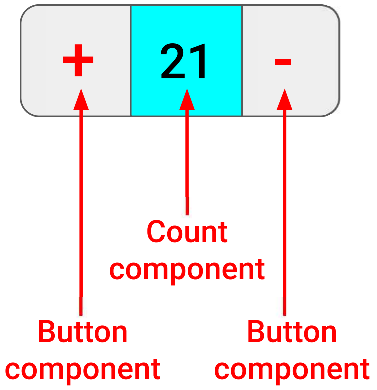
# React Component with Composition

- We can compose existing components to create a new component.
- For example, We can implement Counter using existing component Button and Count.

```
<div ...>
        <Button ... />
        <Count .../>
        <Button ... />
</div
```

Code fragment of Counter component

Counter component composed of three components.

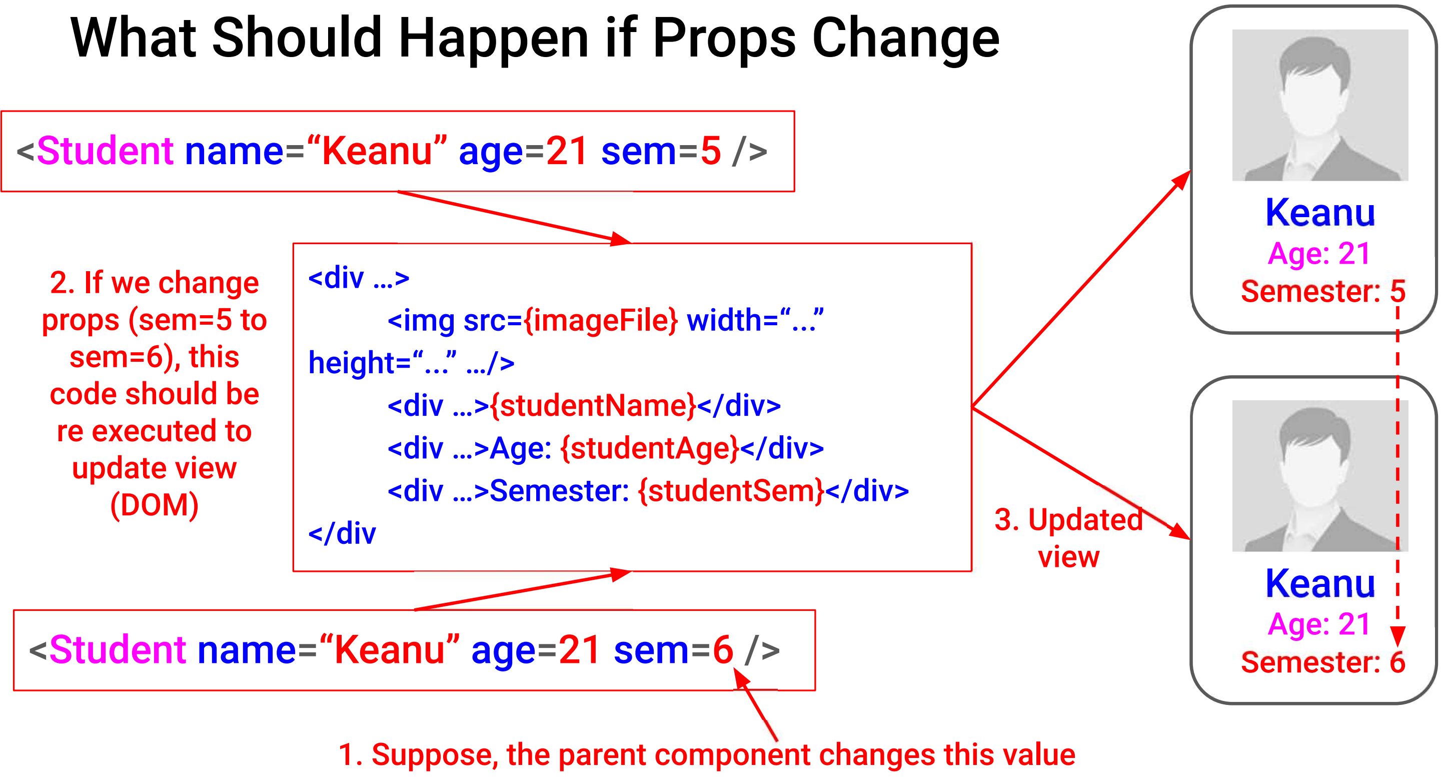


Count component

Button component

Button component

# What Should Happen if Props Change



<Student name="Keanu" age=21 sem=5 />

```
<div …>
        <img src={imageFile} width="…"
height="…" …/>
        <div …>{studentName}</div>
        <div …>Age: {studentAge}</div>
        <div …>Semester: {studentSem}</div>
</div
```

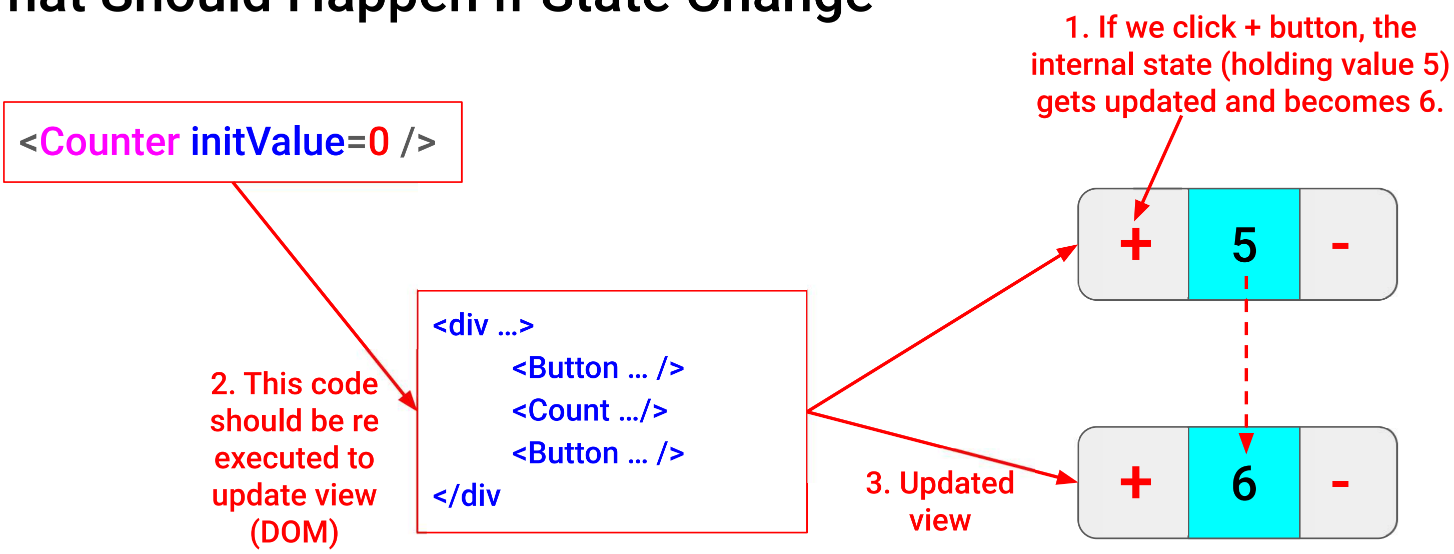
<Student name="Keanu" age=21 sem=6 />

2. If we change props (sem=5 to sem=6), this code should be re executed to update view (DOM)

3. Updated view

1. Suppose, the parent component changes this value

# What Should Happen if State Change



<Counter initValue=0 />

```
<div …>
        <Button … />
        <Count …/>
        <Button … />
</div
```

2. This code should be re executed to update view (DOM)

1. If we click + button, the internal state (holding value 5) gets updated and becomes 6.

3. Updated view

# What React is Responsible for

- React is responsible for:
  - Rendering and re-rendering component (creating virtual DOM).
  - Updating virtual DOM into actual browser DOM efficiently.