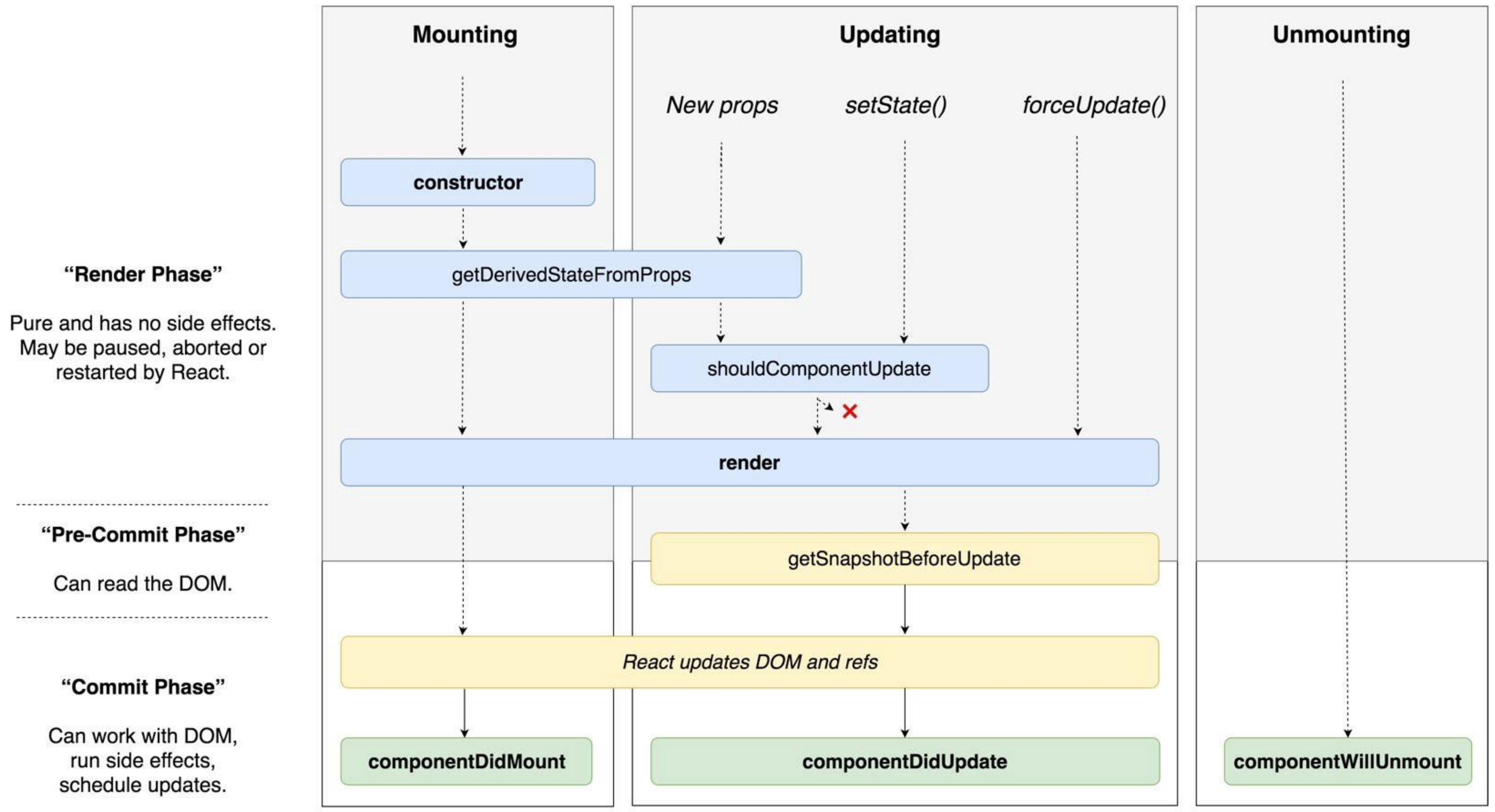
Components & Props



"Render Phase"

Pure and has no side effects. May be paused, aborted or restarted by React.

"Pre-Commit Phase"

Can read the DOM.

"Commit Phase"

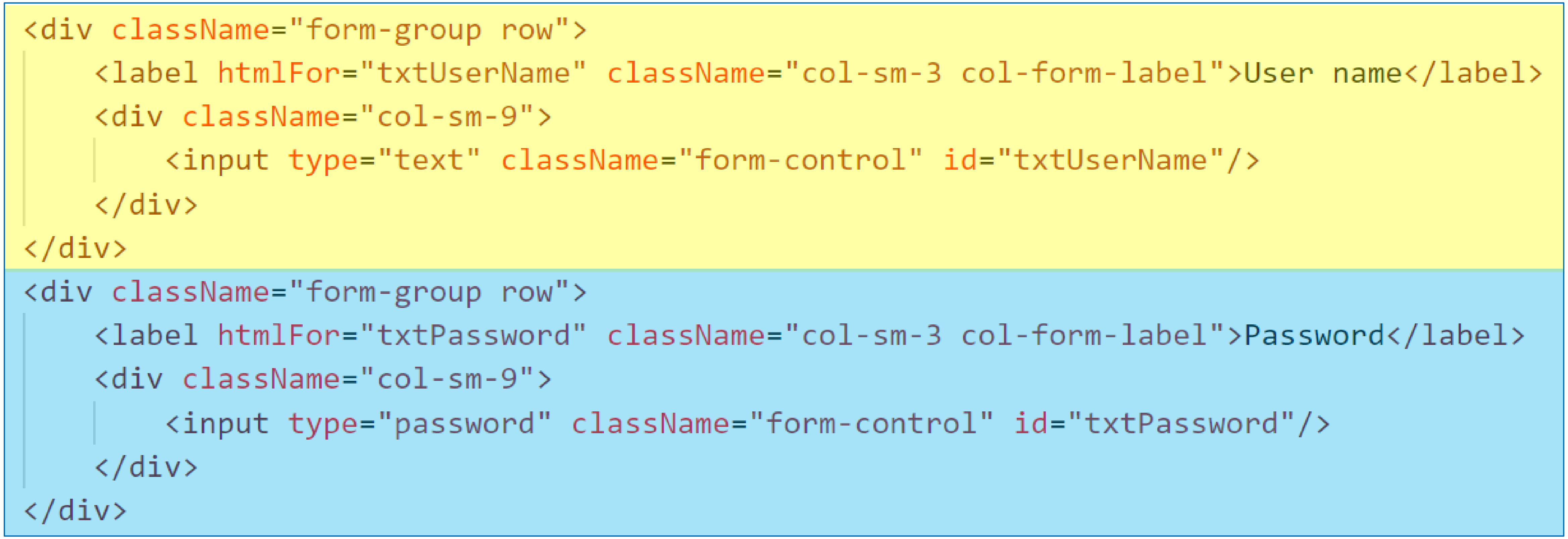
Can work with DOM, run side effects, schedule updates.

What **to do/not to do** with **lifecycle**

- **constructor**
  - Do
    - set initial state
    - if not using class properties syntax — prepare all class fields and bind functions that will be passed as callbacks
  - Don’t
    - cause any side effects (AJAX calls etc.)
- **componentDidMount**
  - Do
    - cause side effects (AJAX calls etc.)
  - Don’t
    - call **this.setState** as it will result in a re-render

- **componentDidUpdate(prevProps, prevState, snapshot)**
  - Do
    - cause side effects (AJAX calls etc.)
  - Don’t
    - call **this.setState** as it will result in a re-render
- **componentWillUnmount**
  - Do
    - remove any timers or listeners created in lifespan of the component
  - Don’t
    - call **this.setState**, start new listeners or timers

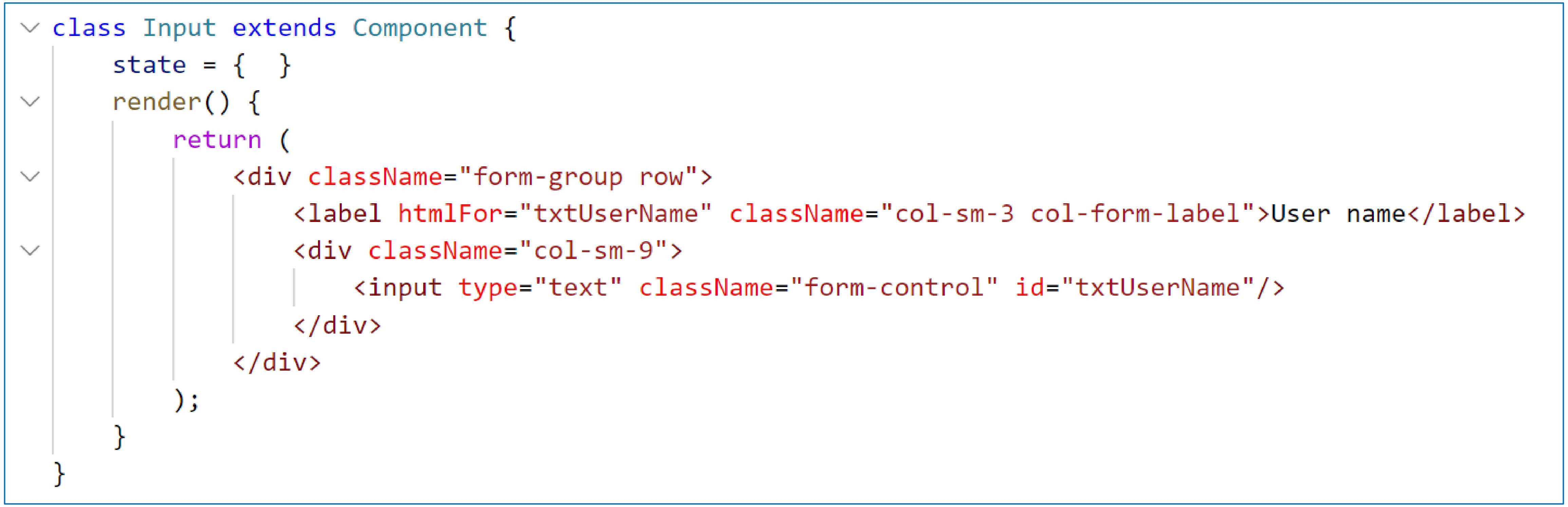
Why we need to create custom control?



```
<div className="form-group row">
    <label htmlFor="txtUserName" className="col-sm-3 col-form-label">User name</label>
    <div className="col-sm-9">
        <input type="text" className="form-control" id="txtUserName"/>
    </div>
</div>
<div className="form-group row">
    <label htmlFor="txtPassword" className="col-sm-3 col-form-label">Password</label>
    <div className="col-sm-9">
        <input type="password" className="form-control" id="txtPassword"/>
    </div>
</div>
```

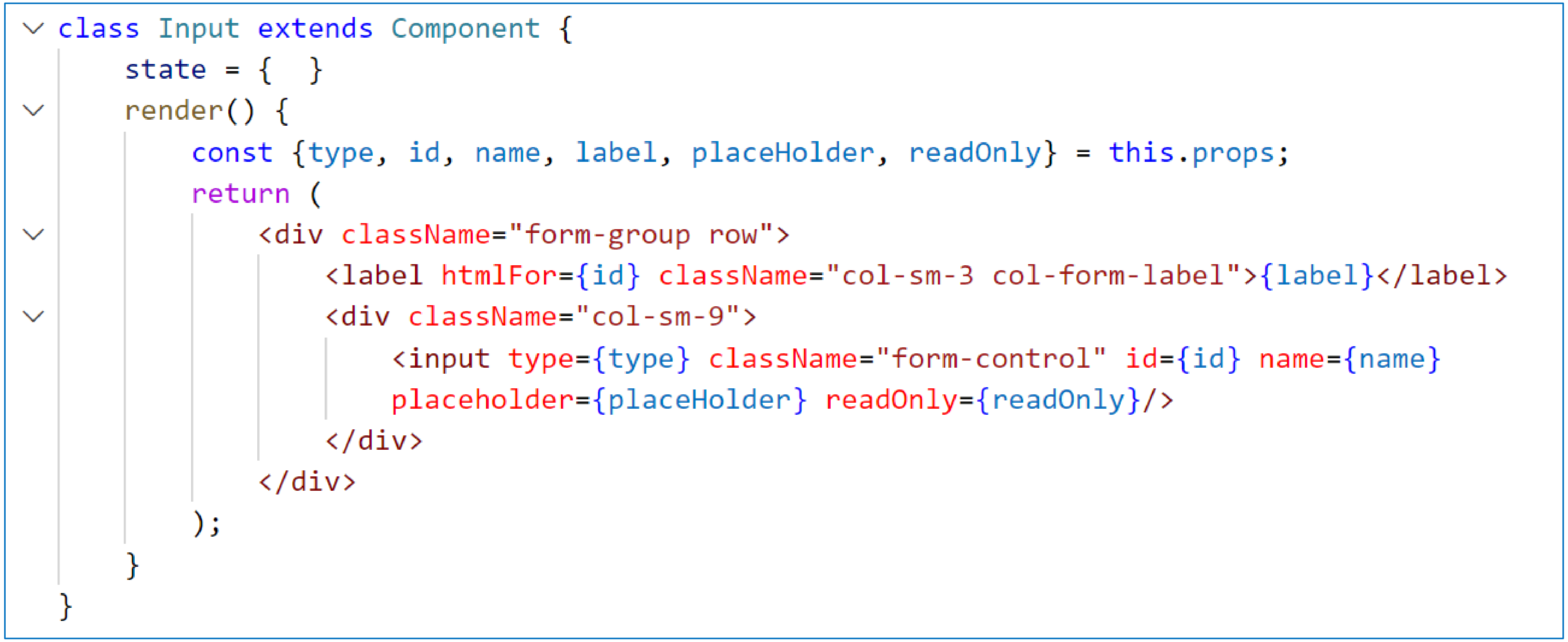
Create a new custom control

- Create a new folder named “controls”
- In “controls” folder, create a new file named “input.jsx”
- Create a new class component named “Input” as below



```
class Input extends Component {
    state = {  }
    render() {
        return (
            <div className="form-group row">
                <label htmlFor="txtUserName" className="col-sm-3 col-form-label">User name</label>
                <div className="col-sm-9">
                    <input type="text" className="form-control" id="txtUserName"/>
                </div>
            </div>
        );
    }
}
```

Working with props

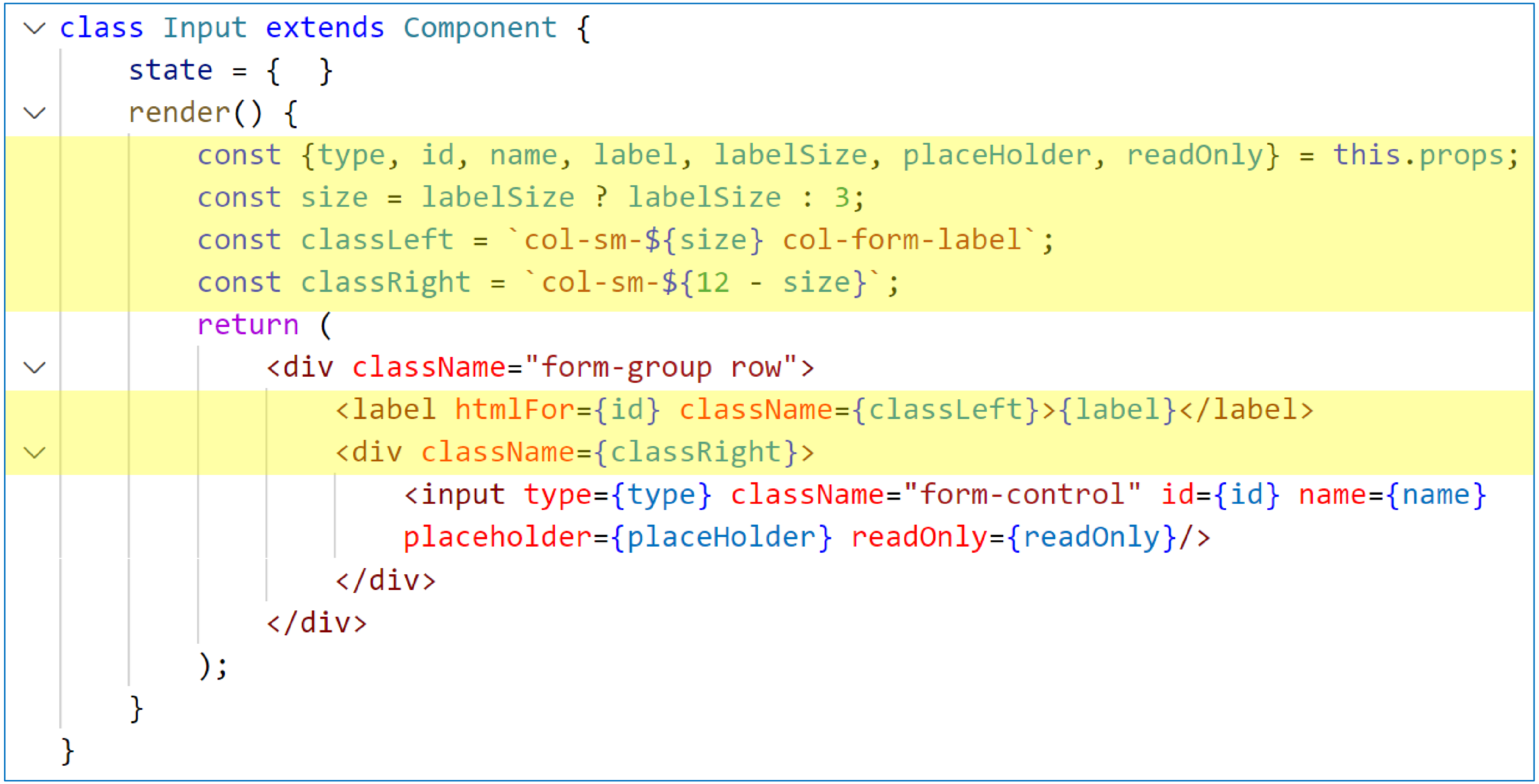


```
class Input extends Component {
    state = {  }
    render() {
        const {type, id, name, label, placeHolder, readOnly} = this.props;
        return (
            <div className="form-group row">
                <label htmlFor={id} className="col-sm-3 col-form-label">{label}</label>
                <div className="col-sm-9">
                    <input type={type} className="form-control" id={id} name={name}
                    placeholder={placeHolder} readOnly={readOnly}/>
                </div>
            </div>
        );
    }
}
```

Applying custom control in login.jsx

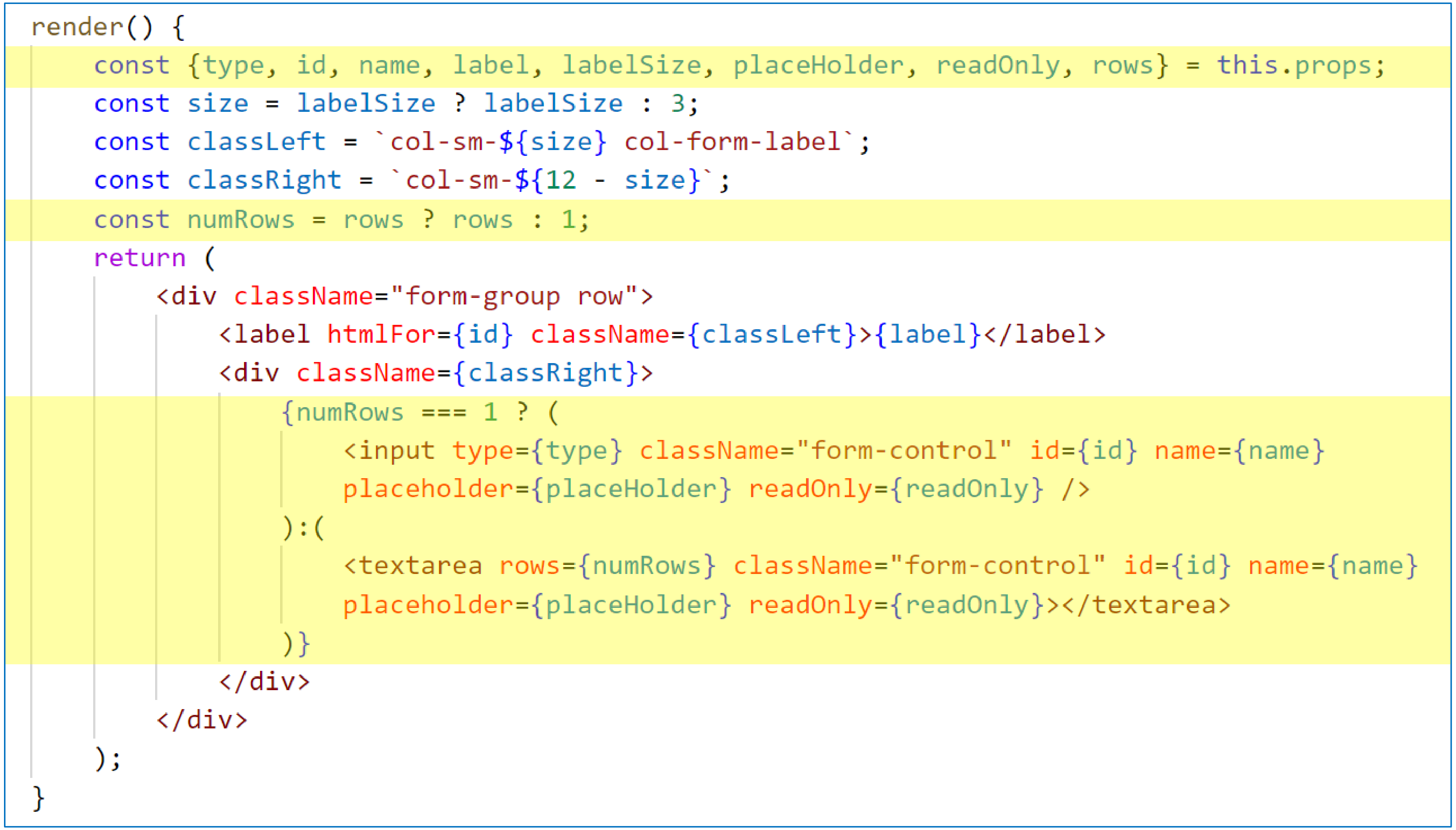
```
<form>
    <Input id="txtUserName" name="userName" label="User name" />
    <Input id="txtPassword" name="password" label="Password" />
    <div className="row">
        <div className="offset-sm-3 col-auto">
            <button type="button" className="btn btn-primary">Sign in</button>
        </div>
    </div>
</form>
```

Controlling label size of custom control



```
class Input extends Component {
    state = {  }
    render() {
        const {type, id, name, label, labelSize, placeHolder, readOnly} = this.props;
        const size = labelSize ? labelSize : 3;
        const classLeft = `col-sm-${size} col-form-label`;
        const classRight = `col-sm-${12 - size}`;
        return (
            <div className="form-group row">
                <label htmlFor={id} className={classLeft}>{label}</label>
                <div className={classRight}>
                    <input type={type} className="form-control" id={id} name={name}
                    placeholder={placeHolder} readOnly={readOnly}/>
                </div>
            </div>
        );
    }
}
```

Supporting textarea in custom control



```
render() {
    const {type, id, name, label, labelSize, placeHolder, readOnly, rows} = this.props;
    const size = labelSize ? labelSize : 3;
    const classLeft = `col-sm-${size} col-form-label`;
    const classRight = `col-sm-${12 - size}`;
    const numRows = rows ? rows : 1;
    return (
        <div className="form-group row">
            <label htmlFor={id} className={classLeft}>{label}</label>
            <div className={classRight}>
                {numRows === 1 ? (
                    <input type={type} className="form-control" id={id} name={name}
                    placeholder={placeHolder} readOnly={readOnly} />
                ):(
                    <textarea rows={numRows} className="form-control" id={id} name={name}
                    placeholder={placeHolder} readOnly={readOnly}></textarea>
                )}
            </div>
        </div>
    );
}
```