Відслідковування стану (властивість **state**)

<https://reactjs.org/docs/react-component.html#setstate>

Кожен компонент може зберігати свій стан (значення його властивостей). Якщо при зміні таких значень необхідно змінити відображення компонента, то такі властивості можна зберігати у спеціальній властивості state. Властивості у state слід сприймати як приватні властивості. Розглянемо типову схему використання властивості state

|  |  |
| --- | --- |
| Загальна схема | Приклад |
| class Клас extends React.Component {  constructor(props) {  super(props);  . . . . . . . .  **//--- Задаємо поч.значення властивостей стану ---**  **this.state** = {  властивість1: значення1,  властивість2: значення2,  . . . . . . . .  };  }  //--- Змінюємо деякі (які потрібно) значення  //--- властивостей стану ---  метод () {  . . . . . .  **this.setState**({  властивість1: значення1,  властивість2: значення2,  . . . . . . . .  });  }  render() {  return (  . . . . . . . .  Звертання до властивостей з state  . . . . . . . .  );  }  } | class Clock extends React.Component {  constructor(props) {  super(props);  **this.state** = {  date: new Date() //🡨 Встановлюємо поч. значення date  };  }  componentDidMount() {  this.timerID = setInterval( () => this.tick(), 1000);  }  componentWillUnmount() {  clearInterval(this.timerID);  }  tick() {  this.setState({  date: new Date() //🡨 Змінюємо значення властивості date  });  }  render() {  return (  <div>  <h1>Hello, world!</h1>  <h2>It is {**this.state.date.**toLocaleTimeString()}.</h2>  </div>  );  }  } |

**Особливості використання властивості state**

* Властивості стану, що зберігаються у компоненті вважаються приватними (доступними тільки всередині об’єкта), хоча їх значення і можуть передаватися у дочірні компоненти;
* пряме звертання до властивості state з метою зміни значення полів допустиме тільки у конструкторі (для ініціалізації полів);
* зміна значень властивостей стану здійснюється з використанням метода setState;
* під час виклику setState автоматично викликається render для оновлення компонента;
* даний метод може використовувати у двох формах:

|  |  |
| --- | --- |
| this.setState({  властивість1: значення1,  властивість2: значення2,  . . . . . . . .  }); | this.setState({  counter: this.state.counter + this.props.increment,  }); |
| this.setState(function(previousState, props) {  return {  властивість1: значення1,  властивість2: значення2,  . . . . . . . .  });  такий виклик гарантує, що значення state, props під час роботи методу будуть актуальними | this.setState(function(state, props) {  return {  counter: state.counter + props.increment  };  }); |

* оновлення стану за допомогою setState є асинхронним;
* оновлення декількох полів можна розділяти у окремі виклики setState

|  |  |
| --- | --- |
|  |  |
| this.setState({  prop1: newValue1,  prop2: newalue2  }); | this.setState({  prop1: newValue1  });  this.setState({  prop2: newalue2  }); |

* setState та колбек

|  |  |
| --- | --- |
|  |  |
| this.setState({  prop1: newValue1,  prop2: newalue2  },  callback)  але рекомендоване замість цього колбека використовувати componentDidMount | this.setState({  prop1: newValue1  },  ()=>console.log(“updated”)); |

Приклад. Без використання ES6

|  |  |
| --- | --- |
|  |  |
|  |  |