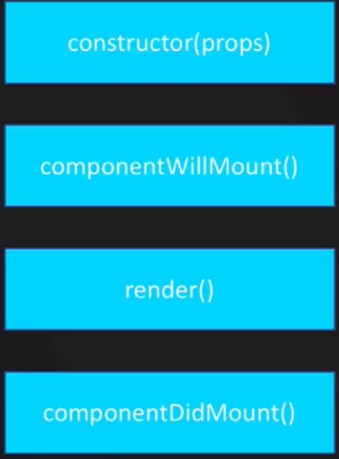
Життєвий цикл компонента (“lifecycle methods”)

<https://reactjs.org/docs/state-and-lifecycle.html>



componentWillMount (**deprecated**)

componentDidMount

componentWillUnmount

componentWillReceiveProps(**deprecated**)

shouldComponentUpdate

componentWillUpdate

componentDidUpdate

|  |  |
| --- | --- |
|  |  |
| class Клас extends React.Component {  constructor(props) {  super(props);  . . . . . . . .  }  //1.Викликається після того, як елемент було вставлено в DOM  componentDidMount() {  }  //2.Викликається перед тим, як елемент буде видалено з DOM  componentWillUnmount() {  }  render() {  return (  . . . . . . . .  );  }  } | class Clock extends React.Component {  constructor(props) {  super(props);  this.state = {  date: new Date()  };  }  **componentDidMount**() {  this.timerID = setInterval(  () => this.tick(),  1000  );  }  **componentWillUnmount**() {  clearInterval(this.timerID);  }  tick() {  this.setState({  date: new Date()  });  }  render() {  return (  <div>  <h1>Hello, world!</h1>  <h2>It is {this.state.date.toLocaleTimeString()}.</h2>  </div>  );  }  }  ReactDOM.render(  <Clock />,  document.getElementById('root')  ); |

Зауваження. Якщо метод render повертає null, то елемент виведено не буде, але метод componentDidMount буде викликано.

Модифікація попередньої задачі таймер змінюється кожні 5 секунд

import React, {Component} from 'react';

class Clock extends Component {

constructor(props) {

super(props);

this.state = {

date: new Date()

};

}

componentDidMount() {

this.timerID = setInterval(

() => this.tick(),

1000

);

}

componentWillUnmount() {

clearInterval(this.timerID);

}

tick() {

this.setState({

date: new Date()

});

console.log('tick')

}

shouldComponentUpdate(nextProps,nextState){

let ms=nextState.date.getSeconds()

if(ms%5===0)

return true

return false

}

render() {

return (

<div>

<h1>Hello, world!</h1>

<h2>It is {this.state.date.toLocaleTimeString()}.</h2>

</div>

);

}

}

export default Clock

==========================================================================================================================

Приклад. Зробити елемент «Передбачувач», який випадковим чином вибирає передбачення з наперед заданого масиву.



|  |  |
| --- | --- |
|  |  |
|  | *//-------------------------* **class** PredictionViewer **extends** Component{  constructor(props)  {  **super**(props);  **this**.onClick=**this**.onClick.bind(**this**);  **this**.**state**={  **prediction**:**this**.getPrediction()  }  }  //------ Випадковим чином вибираємо передбачення з масиву -----------  getPrediction(){  **let** randomIndex=***Math***.floor(***Math***.random()\***this**.**props**.**items**.**length**);  **return this**.**props**.**items**[ randomIndex ];  }  //--------- Викликається після монтування компонента у сторінку --------  componentDidMount() {  **this**.**timerID** = setInterval(  () => **this**.tick(),  1000  );  }  //---------- Компонент буде вилучено --------  componentWillUnmount() {  clearInterval(**this**.**timerID**);  }  //---------- Метод, який буде викликатися при спрацюванні таймера -------------  tick()  {  **this**.setState({  **prediction**:**this**.getPrediction()  })  }  //---------- Зупинення таймера -----------  onClick(){  clearInterval(**this**.**timerID**);  }  //---------------------------------  render(){  **return** (  <**div**>  <**label** >  Your prediction:  <**input type="text" value=**{**this**.**state**.**prediction**}/>  <**button onClick=**{**this**.onClick}> Stop </**button**>  </**label**>  </**div**>  )  } } |
|  | **class** App **extends** Component {  render() {  **return** (  <**div className="App"**>   <**PredictionViewer items=**{[**"Love"**,**"Great love"**, **"Money"**, **"Happy"**,**"Great Heppy"**]}/>  . . . . . .  } |