6기_이희제

JSX

JSX 란?

- 리액트에서 생김새를 정의할 때, 사용하는 문법
- React “엘리먼트(element)” 를 생성
- 자바 스크립트에서 쓰는 html 이라고 생각하면 된다.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;  // 중괄호를 통해서 js값을 넣어줄 수 있다.

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

JSX 속성 정의

```
const element = <div tabIndex="0"></div>;
```

```
const element = <img src={user.avatarUrl}></img>;
```

ReactDOM.render()

ReactDOM.render(element, container[, callback])

-> container DOM 에 element 를 렌더링한다. (container 가 target DOM)

브라우저에 있는 실제 DOM 내부에 리액트 컴포넌트를 렌더링하겠다는 것을 의미

Element Rendering

Element

- 리액트 앱에서 가장 작은 단위
- 화면에 표시할 내용을 나타냄.
- 일반 객체 (Plain object)

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

Components and Props

Components

함수형 =>

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

class =>

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Props

-어떠한 값을 컴포넌트에게 전달해줘야 할 때, props 를 사용
- 읽기 전용

```
function Welcome(props) {    // 함수 정의 컴포넌트.
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

State and Lifecycle

state

```
function Clock(props) {
  return (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}

function tick() {
  ReactDOM.render(
    <Clock date={new Date()} />,
    document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

state

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

왜 super(props) 호출? 🤔

1. Component 상속
2. 생성자를 작성하게되 기존 클래스 생성자 덮어씀
3. 리액트 컴포넌트 생성자 실행 후
4. state 설정.

LifeCycle Method

```
componentDidMount() {          // 생명주기 메서드
  this.timerID = setInterval(
    () => this.tick(),
    1000
  );
}

componentWillUnmount() {     // 생명주기 메서드
  clearInterval(this.timerID);
}
```

-컴포넌트 클래스에서 메서드를 선언하여 컴포넌트가 마운트되거나 언마운트 될 때 일부 코드를 작동할 수 있음.

- componentDidMount() 메서드는 컴포넌트 출력물이 DOM에 렌더링 된 후에 실행

-DOM으로부터 한 번이라도 삭제된 적이 있다면 React는 타이머를 멈추기 위해 componentWillUnmount() 호출

이벤트 처리

이벤트 처리

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

e.preventDefault() -> 이벤트의 기본 동작 방지.

이벤트 처리

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

e.preventDefault() -> 이벤트의 기본 동작 방지.

binding

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // 콜백에서 `this`가 작동하려면 아래와 같이 바인딩 해주어야 합니다.
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

ReactDOM.render(
  <Toggle />,
  document.getElementById('root')
);
```

감사합니다