5. ref

- ref는 DOM을 직접 조작할 때 사용한다.
- 함수 컴퍼넌트에서 ref를 사용하려면 hooks를 사용해야 한다. 클래스형 컴퍼넌트에서 ref를 사용 실습을 진행해 보자

5.1 예제 컴퍼넌트

```
src/App.js
import "./App.css";
import React05Ref01 from "./mysrc/React05Ref01";

function App() {

  return (
    <>
      <React05Ref01 />
    </>
  );

}
export default App;

src/mysrc/React05Ref01.css
.success {
  background-color: mediumseagreen;
}

.failure {
  background-color: lightcoral;
}

src/mysrc/React05Ref01.js
import React, { Component } from 'react';
import './React05Ref01.css';

class React05Ref01 extends Component {

  state = {
    password: '',
    clicked: false,
    validated: false
  }

  handleChange = (e) => {
    this.setState({
      password: e.target.value
    })
  }

  handleButtonClick = () => {
    this.setState({
```

```
      clicked: true,
      validated: this.state.password === '0000'
    })
  }

  render() {
    return (
      <div>
        <input type="password" value={this.state.password}
          onChange={this.handleChange}
          className={this.state.clicked ? (this.state.validated ?
'success' : 'failure') : ''}
        />
        <button onClick={this.handleButtonClick}>검증하기</button>
      </div>

    );
  }
}

export default React05Ref01;
```

5.2 ref사용

- 참고 :https://velog.io/@hoon_dev/React-Ref-이용하기feat-createRef
- ref를 사용하는 방법은 2가지
  1. callback함수를 통한 ref 설정
  2. `createRef`를 통한 설정

5.2.1 콜백함수를 통한 ref설정

- ref를 만드는 가장 기본적인 방법은 callback함수를 상해서 ref를 달고자하는 요소에 `ref라는 코백함수를 props로 전달`
- 이 콜백함수는 ref값을 파라미커로 전달 받아서 `함수 내부에서 파라미터로 전달 받은 ref를 컴퍼넌트의 변수로 설정` 한다.

  ```
  <input ref={(ref) => {this.input=ref}} >
  ```

- 이렇게 하면 앞으로 `this.input은 input요소의 DOM`을 가리킨다.
- ref의 이름은 자유롭게 지정할 수 있다. DOM타입과 관계없이 this.superman = ref 처럼 마음대로 지정한다.

5.2.2 `createRef` 통한 ref설정

- ref를 만드는 또다른 방법은 `리액트에 내장되어 있는 createRef 함수를 사용`하는 것
- createRef를 이용하면 `더 적은 코드로 쉽게 사용`할 수 있다.
- 이 함수는 React V16.3부터 도입
- DOM에 접근하려면 `this.input_ref.current를 조회`하면 된다.
- 콜백함수를 사용할 때와 `다른 점은 뒤 부분에 .current를 넣어 주어야 한다.`
- button을 클릭했을 때 foucus를 input박스로 자동으로 넘어가도록 작성하는 방법

```
src/App.js
import "./App.css";
import React05Ref02 from "./mysrc/React05Ref02";
import React05Ref01 from './mysrc/React05Ref01';

function App() {

  return (
    <>
      <React05Ref01 />
      <hr/>
      <React05Ref02 />
    </>
  );

}
export default App;

src/mysrc/React05Ref02.js
import React, { Component } from 'react';
import './React05Ref01.css';

class React05Ref02 extends Component {

  // 1. createRef()를 통한 ref설정
  // input_ref = React.createRef();

  // handleFocus = () => {
  //   this.input.current.focus();
  // }

  state = {
    password: '',
    clicked: false,
    validated: false
  }

  handleChange = (e) => {
    this.setState({
      password: e.target.value
    })
  }

  handleButtonClick = () => {
    this.setState({
      clicked: true,
      validated: this.state.password === '0000'
    })

  // 2. 콜백함수를 이용한 ref설정
  // 1) button onClik이벤트에 focus()
    this.input_ref.focus();
  }

  render() {
    return (
      <div>
```

```
        <input type="password" value={this.state.password}
          // 3. input에 ref달기
          ref={(ref) => {this.input_ref=ref}}
          onChange={this.handleChange}
          className={this.state.clicked ? (this.state.validated ?
'success' : 'failure') : ''}
        />
        <button onClick={this.handleButtonClick}>검증하기</button>
      </div>

    );
  }
}

export default React05Ref02;
```

5.3 컴퍼넌트에 ref 달기

- 리액트에서는 컴퍼넌트에도 ref를 달 수가 있다.
- 이 방법은 주로 `컴퍼넌트 내부에 있는 DOM을 컴퍼넌트 외부에서 사용할 때 사용`
- 컴퍼넌트에 ref를 다는 방법은 `DOM에 ref를 다는 방법과 동일`

5.3.1 사용법

```
jsx
<MyComponent ref={(ref) => {this.myComponent=ref}}  />
```

- 이렇게 하면 MyComponent 내부의 메서드 및 멤버변수에도 접근할 수 있다.
- 즉, 내부의 ref에도 접근할 수 있다. (예: myComponent.handleClici, myComponent.input등...)

부모 컴퍼넌트에서 스크롤바 내리기

- 부모컴퍼넌트에서 스크롤바를 조정하는 작업

```
src/App.js
import "./App.css";
import { Component } from 'react';
import React05Ref01 from './mysrc/React05Ref01';
import React05Ref02 from "./mysrc/React05Ref02";
import React05Ref03 from "./mysrc/React05Ref03";

// App를 function이 아닌 class로 작성
class App extends Component {

  render() {
    return (
      <>
        <React05Ref01 />
        <hr/>
        <React05Ref02 />
        <hr/>
        <React05Ref03 ref={ref => (this.xxx = ref)} />
```

```
        {/* 문법상으로 onClick={this.xxx.scrollToBottom}으로 작성해
도 틀린 것은 아니지만
            처음 랜더링 될 때 this.xxx.scrollToBottom의 값은
undefined이기 때문에 오류가
            발생한다. 화살표 문법을 사용해서 새로운 함수를 만들고 그 내
부에서 this.scrollBox
            .scrollToBottom()메서드를 실행하면 버튼을 클릭(이미 랜더링
해서 this.scrollBox를
            설정한 시점)할 때 scrollToBottom값을 읽어와서 실행하므로
에러는 발생하지 않는다.
        */}
        <button onClick={() => this.xxx.scrollToBottom()}>
          맨 밑으로...
        </button>
      </>
    );
  }
}
export default App;

src/mysrc/React05Ref03.js
import { Component } from 'react';

class React05Ref03 extends Component {

  scrollToBottom = () => {
    const { scrollHeight, clientHeight } = this.box;
    /* 앞 코드에는 비구조화 할당 문법을 사용. 다음 코드와 같은 의미.
    const scrollHeight = this.box.scrollHeight;
    const clientHeight = this.box.cliengHeight;
    */
    this.box.scrollTop = scrollHeight - clientHeight;
  };

  render() {
    const style = {
      border: '1px solid black',
      height: '300px',
      width: '300px',
      overflow: 'auto',
      position: 'relative'
    };
    const innerStyle = {
      width: '100%',
      height: '650px',
      background: 'linear-gradient(white, black)'
    };
    return (
      <div style={style} ref={ref => {this.box = ref}}>
        <div style={innerStyle} />
      </div>
    );
  }
}
export default React05Ref03;
```

5.4 정리

- 컴퍼넌트 내부에서 DOM에 직접 접근해야 할 때는 ref를 사용한다.
- 서로 다른 컴퍼넌트끼리 데이터를 교류할 수 있지만 이는 잘못 사용하는 것이다. 이 방법은 react상상에 어긋난 설계이다.\
- 앱 규모가 커지젼 구조가 스파게티처럼 꼬여서 유지보수가 불가능하게 된다.
- 컴퍼넌트끼리 데이터를 교환할 때는 `Redux or Context API를 사용`하여 효율적으로 교환할 수 있다.
- `함수 컴퍼넌트에서는 useRef Hook함수를 사용`한다.