Day 6-1. Component Event_2 (Mar 1st, 21) #어려움

Day 5-4. Component Event_1 (Feb 27, 21)에 이어서, 목록(HTML, CSS, JavaScript)을 클릭했을 때, App 컴포넌트의 state의 mode를 'read'로 바꾸고,

이제, 클릭한 목록의 content가 본문에 나오도록 하는 이벤트를 만들어 보자.

이전 시간에,

App.js

```
        <TOC
          onChangePage={function(){
            this.setState({mode='read'});

          }.bind(this)}
          data={this.state.contents}

        ></TOC>
```

TOC 태그에 onChangePage함수 세트 설치 & bind

TOC.js

```
class TOC extends Component{
    render(){
      console.log('TOC render');
      var lists = [];
      var data = this.props.data
      var i = 0;
      while(i < data.length){
        lists.push(
          <li key={data[i].id}>
            <a
              href={"/content/"+data[i].id}
              onClick={function(){
                e.preventDefault();
                this.props.onChangePage();
              }.bind(this)}
            >{data[i].title}</a>
          </li>);
        i = i + 1;
      }
```

```
      return (
        <nav>
          <ul>
            {lists}
          </ul>
        </nav>
      )
    }
  }
```

TOC 컴포넌트에 onClick 함수 설치 & bind

목록(HTML, CSS, JavaScript)을 클릭했을 때, App 컴포넌트의 state의 mode를 'read'로 바꾸는 것 까지 진행했다.

이제, 클릭한 목록의 content가 본문에 표시되도록하는 과정을 진행한다.

방법 : App의 state에 'selected content id'와 같은 개념을 이용해서, 현재 선택된 content를 본문에 표시하도록 한다. (contents라고 하는 객체에 있는 id값 중에, selected content id 값과 일치하는 것을 본문에 표시하도록 한다.)

App.js

```
class App extends Component{
  constructor(props){
    super(props);
    this.state={
      mode:'read',
      selected_content_id:2,
      subject:{title:'WEB', sub:'World Wide Web'},
      welcome:{title:'Welcome', desc:'Hello, React!'},
      contents:[
        {id:1, title:'HTML', desc:'HTML is for information'},
        {id:2, title:'CSS', desc:'CSS is for design'},
        {id:3, title:'JavaScript', desc:'JavaScript is for interactive'}
      ]
    }
  }
```

우선 2번 컨텐트를 선택하도록 초기화.

selected_content_id에 따라서

```
        {id:1, title:'HTML', desc:'HTML is for information'},
        {id:2, title:'CSS', desc:'CSS is for design'},
        {id:3, title:'JavaScript', desc:'JavaScript is for interactive'}
```

중에 어떤 것이 본문에 나올지를 지정하는 것은,

```
render() {
    console.log('App render');
    var _title, _desc = null;

    if(this.state.mode === 'welcome'){
      _title = this.state.welcome.title;
      _desc = this.state.welcome.desc;
    } else if(this.state.mode === 'read'){
      _title = this.state.contents[0].title;
      _desc = this.state.contents[0].desc;
    }
```

코드를

```
App.js

render() {
    console.log('App render');
    var _title, _desc = null;

    if(this.state.mode === 'welcome'){
      _title = this.state.welcome.title;
      _desc = this.state.welcome.desc;
    } else if(this.state.mode === 'read'){
      var i = 0;
      while(i < this.state.contents.length){
        var data = this.state.contents[i];
        if(data.id === this.state.selected_content_id){
          _title = data.title;
          _desc = data.desc;
        }
        i = i + 1;
      }
    }
```

로 바꿔준다.

App.js

```
<TOC
          onChangePage={function(){
            alert('hi');
            this.setState({mode='read'});

          }.bind(this)}
          data={this.state.contents}

        ></TOC>
```

TOC(리스트)에 OnChangePage 이벤트가 발생했을 때, thisSetState를 통해서 mode값과 함께 selected_content_id 값을 0으로 선언해준다.

App.js

```
        <TOC
          onChangePage={function(){
            alert('hi');
            this.setState({
              mode='read',
              selected_content_id:0
            });

          }.bind(this)}
          data={this.state.contents}

        ></TOC>
```

TOC.js

```
class TOC extends Component{
    render(){
      console.log('TOC render');
      var lists = [];
      var data = this.props.data
      var i = 0;
      while(i < data.length){
        lists.push(
          <li key={data[i].id}>
            <a
              href={"/content/"+data[i].id}
              onClick={function(){
                e.preventDefault();
                this.props.onChangePage();
```

```
              }.bind(this)}
            >{data[i].title}</a>
          </li>);
        i = i + 1;
      }
      return (
        <nav>
          <ul>
            {lists}
          </ul>
        </nav>
      )
    }
  }
```

onChangePage라고 하는 이벤트를 실행시키는 부분이

TOC.js의 onClick을 했을 때, this.props.onChangePage(); 의 props를 실행시키는 것을 통해서

```
TOC.js

              onClick={function(){
                e.preventDefault();
                this.props.onChangePage();
              }.bind(this)}
```

```
App.js

        <TOC
          onChangePage={function(){
            alert('hi');
            this.setState({
              mode='read',
              selected_content_id:0
            });

          }.bind(this)}
          data={this.state.contents}

        ></TOC>
```

App.js의 onChangePage 함수를 실행시켜주는 것이다.

(TOC.js가 App.js에 있는 함수를 실행시키는 것)

즉, 그것을 실행할 때, 인자로 우리가 클릭한 항목의 id값을 주도록 한다.

```
TOC.js

class TOC extends Component{
    render(){
      console.log('TOC render');
      var lists = [];
      var data = this.props.data
      var i = 0;
      while(i < data.length){
        lists.push(
          <li key={data[i].id}>
            <a
              href={"/content/"+data[i].id}
              data-id={data[i].id}
              onClick={function(){
                e.preventDefault();
                this.props.onChangePage();
              }.bind(this)}
            >{data[i].title}</a>
          </li>);
        i = i + 1;
      }
      return (
        <nav>
          <ul>
            {lists}
          </ul>
        </nav>
      )
    }
  }
```