# Day 9-1. Create_6_shouldComponentUpdate (Mar 4, 21)

push() 대신 concat을 사용하는게 유리한 이유)

TOC가 화면에 표시되기 위해서 필요한 data:

App컴포넌트의 contents가 지니고 있는 리스트

```
App.js

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'}
      ]
```

이 리스트가 바뀌면, TOC컴포넌트의 render가 호출되는 것을 통해서 TOC가 다시 그려지게 된다.

```
TOC.js

import React, { Component } from 'react';
class TOC extends Component{
    render(){
```

App컴포넌트의 contents가 바뀌지 않았다면, TOC 컴포넌트의 render는 호출되지 않는게 유리하다.

```
TOC.js

class TOC extends Component{
    shouldComponentUpdate(newProps, newState){

      return true;
    }
    render(){
```

render 전에 shouldComponentUpdate가 실행된다.

shouldComponentUpdate 결과값이 true면, render가 호출되고,

false면 render가 호출되지 않도록 한다.

shouldComponentUpdate 업데이트된 값과 업데이트 되기 전의 값에 접근할 수 있다.

이제부터 할 것)

TOC로 들어오는 data props의 값이 바뀔 때 render가 호출되고, 바뀌지 않았을 땐 render가 호출되지 않도록 한다.

```
TOC.js

class TOC extends Component{
    shouldComponentUpdate(newProps, newState){
      it(this.props.data === newProps.data){
        return false;
      }
      return true;
    }
    render(){
```

쓸데없는 redering을 막기위해 shouldComponentUpdate를 사용했고, 원본값과 변경값을 비교하여 변경값이 있을 때만 TOC가 render된다는 조건을 추가했다고 치자. 이 때, state.contents[]를 push로서 값을 추가했다면 TOC에서 this.props.data를 했을 때 원본 배열에 값을 추가하였기 때문에 shouldComponentUpdate함수 내에 if문으로 조건을 붙일 수 없다. 그러나 concat()을 사용한다면 원본값은 두고 그 원본값을 복제하여 변경값을 추가한다음 render()하기 때문에 원본값과 변경값을 비교할 수 있는 환경을 만들 수 있다