컴포넌트 만들기

Life Cycle

render() 함수

constructor 메서드

getDerivedStateFromProps 메서드

componentDidMount 메서드

shouldComponentUpdate 메서드

getSnapshotBeforeUpdate 메서드

componentDidUpdate 메서드

componentWillUnmount 메서드

componentDidCatch 메서드

컴포넌트를 만드는 방법은 크게 두가지가 있습니다.

클래스형 컴포넌트

```
import React, { Component } from 'react';

class SampleClassComponent extends Component {
  constructor(props) {
    super(props);

    this.state = {
      value: 0,
    };
  }

  // constructor를 사용하지 않고, state를 constructor 밖으로 꺼낼 때.
  // state = {
  //  value : 0
  // };
```

```
  onClick = () => {
    this.setState({
      value: this.state.value + 1,
    });
  };

  render() {
    return (
      <div>
        <p>value : {this.state.value}</p>
        <button onClick={this.onClick}>버튼</button>
      </div>
    );
  }

  componentDidMount() {
    console.log('componentDidMount');
  }

  componentDidUpdate(prevProps, prevState, snapshot) {
    console.log('componentDidUpdate');
  }
}

export default SampleClassComponent;
```

함수형 컴포넌트

```
import React, { useState, useEffect } from 'react';

const SampleFuncComponent = () => {
  const [value, setValue] = useState(0);

  useEffect(() => {
    console.log('렌더링 완료');
  });

  // useEffect(() => {
  //  console.log('마운트 될때만 실행');
  // }, []);

  // useEffect(() => {
  //  console.log('특정 값이 업데이트될 때만 실행');
  // }, [value]);

  // useEffect(() => {
  //  console.log('렌더링 완료');
  //   () => {
  //    console.log('언마운트 되기 전이나 업데이트 되기 직전에 실행');
  //  }
  // });
```

```
  const onClick = () => {
    setValue(value + 1);
  };

  return (
    <div>
      <p>value : {value}</p>
      <button onClick={onClick}>버튼</button>
    </div>
  );
};

export default SampleFuncComponent;
```