5. props 를 통해 컴포넌트에게 값 전달하기

이번에는 컴포넌트의 props 라는 개념에 대해서 알아보겠습니다. props 는 properties 의 줄임말입니다. 우리가 어떠한 값을 컴포넌트에게 전달해줘야 할 때, props 를 사용합니다.

props 의 기본 사용법

예를 들어서, App 컴포넌트에서 Hello 컴포넌트를 사용 할 때 `name` 이라는 값을 전달해주고 싶다고 가정해봅시다. 그러면, 이렇게 코드를 작성하면 됩니다.

App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <Hello name="react" />
  );
}

export default App;
```

이제, Hello 컴포넌트에서 name 값을 사용 하고 싶을 땐 어떻게 하면 되는지 알아볼까요?

Hello.js

```
import React from 'react';

function Hello(props) {
  return <div>안녕하세요 {props.name}</div>
}

export default Hello;
```

컴포넌트에게 전달되는 props 는 파라미터를 통하여 조회 할 수 있습니다. props 는 객체 형태로 전달되며, 만약 `name` 값을 조회하고 싶다면 `props.name` 을 조회하면 됩니다.

여러개의 props, 비구조화 할당

Hello 컴포넌트에 또 다른 props 를 전달해봅시다. `color` 라는 값을 설정해보세요.

App.js

```
import React from 'react';
import Hello from './Hello';
```

```
function App() {
  return (
    <Hello name="react" color="red"/>
  );
}

export default App;
```

그 다음에는, Hello 컴포넌트에서 `color` 값을 조회해서 폰트의 색상으로 설정을 해보겠습니다.

Hello.js

```
import React from 'react';

function Hello(props) {
  return <div style={{ color: props.color }}>안녕하세요 {props.name}</div>
}

export default Hello;
```

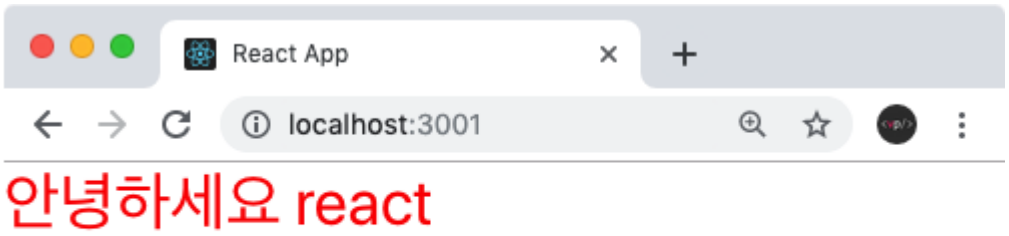
props 내부의 값을 조회 할 때마다 `props.` 를 입력하고 있는데요, 함수의 파라미터에서 비구조화 할당 (혹은 구조 분해라고도 불립니다) 문법을 사용하면 조금 더 코드를 간결하게 작성 할 수 있습니다.

Hello.js

```
import React from 'react';

function Hello({ color, name }) {
  return <div style={{ color }}>안녕하세요 {name}</div>
}

export default Hello;
```



defaultProps 로 기본값 설정

컴포넌트에 props 를 지정하지 않았을 때 기본적으로 사용 할 값을 설정하고 싶다면 컴포넌트에 `defaultProps` 라는 값을 설정하면 됩니다.

Hello.js

```
import React from 'react';

function Hello({ color, name }) {
  return <div style={{ color }}>안녕하세요 {name}</div>
}

Hello.defaultProps = {
  name: '이름없음'
}

export default Hello;
```

한번 App 에서 name 이 없는 Hello 컴포넌트를 렌더링해보세요.

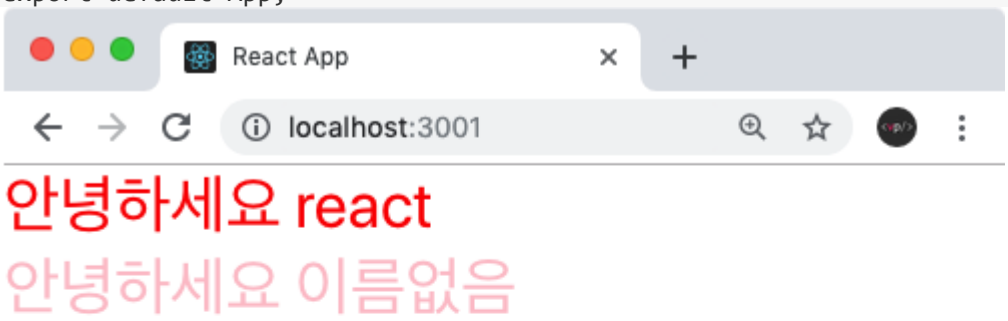
App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <>
      <Hello name="react" color="red"/>
```

```
      <Hello color="pink"/>
    </>
  );
}

export default App;
```



props.children

컴포넌트 태그 사이에 넣은 값을 조회하고 싶을 땐, `props.children` 을 조회하면 됩니다.
이번에, `props.children` 을 사용하는 새로운 컴포넌트를 만들어보겠습니다.
Wrapper.js 를 src 디렉터리에 만들어보세요.

Wrapper.js

```
import React from 'react';

function Wrapper() {
  const style = {
    border: '2px solid black',
    padding: '16px',
  };
  return (
    <div style={style}>

    </div>
  )
}

export default Wrapper;
```

이 컴포넌트를 App 에서 사용해봅시다!

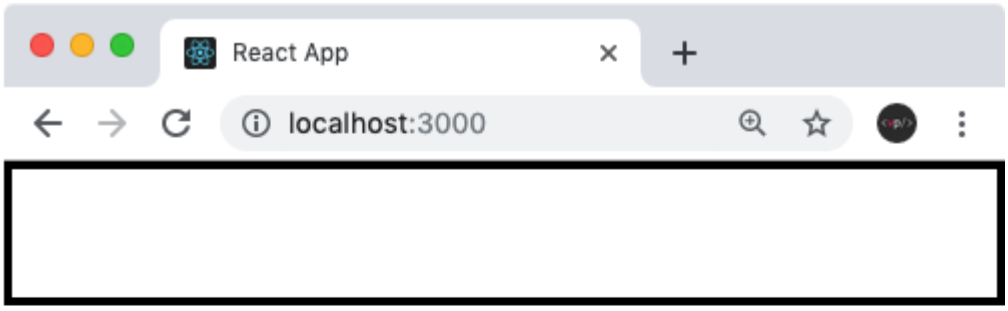
App.js

```
import React from 'react';
import Hello from './Hello';
import Wrapper from './Wrapper';

function App() {
  return (
    <Wrapper>
      <Hello name="react" color="red"/>
      <Hello color="pink"/>
    </Wrapper>
  );
}

export default App;
```

이렇게 Wrapper 태그 내부에 Hello 컴포넌트 두개를 넣었는데요, 브라우저를 확인하면 다음과 같이 Hello 컴포넌트들은 보여지지 않을 것입니다.



내부의 내용이 보여지게 하기 위해서는 Wrapper 에서 `props.children` 을 렌더링해주어야 합니다.

Wrapper.js

```
import React from 'react';

function Wrapper({ children }) {
  const style = {
```

```
    border: '2px solid black',
    padding: '16px',
  };
  return (
    <div style={style}>
      {children}
    </div>
  )
}

export default Wrapper;
```