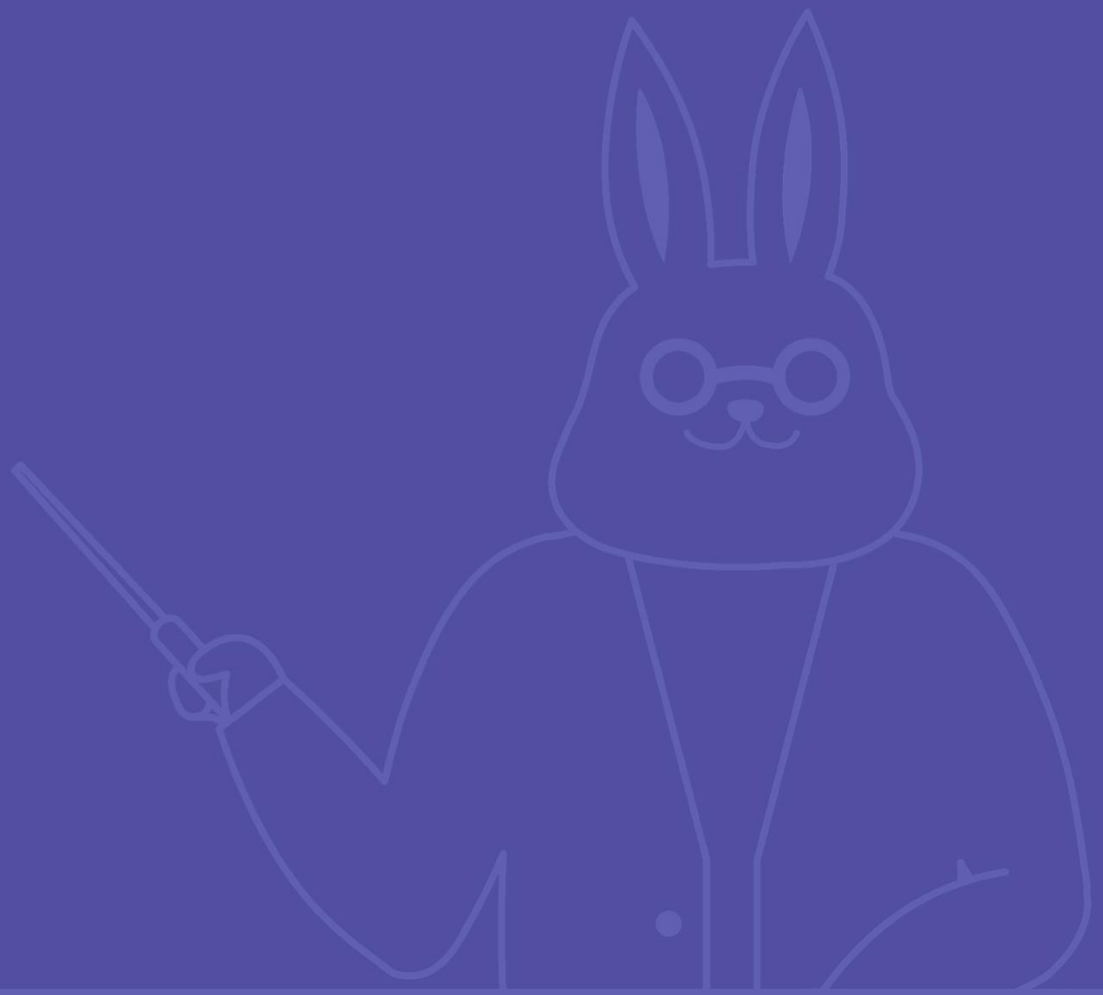
React 기초 I

04 JSX와 컴포넌트



목차

01

JSX

JSX란

코드

```
const App = () => {
  return (
    <div>
      <p>안녕</p>
      <MyComponent>반가워</MyComponent>
      <div>바이바이</div>
    </div>
  );
}
```

- JSX는 함수 호출과 객체 생성을 위한 문법적 편의를 제공하는 JavaScript의 확장
- HTML과 비슷하게 생겼으나 JavaScript이고 HTML과 다른 부분이 있음

JSX는 Babel에 의해서 Transcompile 됩니다.

JSX

```
(
  <div className="App">
    <header className="App-header">
      <img src={logo} className="App-logo" alt="logo" />
      <h1 className="App-title">Welcome to React</h1>
    </header>
    <p className="App-intro">
      Hello world
    </p>
  </div>
)
```

JS

```
React.createElement("div", {
  className: "App"
}, React.createElement("header", {
  className: "App-header"
}, React.createElement("img", {
  src: logo,
  className: "App-logo",
  alt: "logo"
}), React.createElement("h1", {
  className: "App-title"
}, "Welcome to React")), React.createElement("p", {
  className: "App-intro"
}, "Hello world"));
```

JSX의 장점

1. 개발자 편의성 향상
2. 협업에 용이 / 생산성 향상
3. 문법 오류와 코드량 감소

JSX 특징 / HTML과 차이점

1. HTML 태그 내에 JavaScript 연산
2. class → className
3. 스타일은 object로
4. 닫는 태그 필수
5. 최상단 element는 반드시 하나

HTML 태그 내에 JavaScript 연산

HTML + JS

```
<div>
  <span id="a"></span> +
  <span id="b"></span> =
  <span id="sum"></span>
</div>

<script>
  const a = 3;
  const b = 6;
  document.getElementById("a").innerText = a;
  document.getElementById("b").innerText = b;
  document.getElementById("sum").innerText = a + b;
</script>
```

JSX

```
const App = () => {
  const a = 3;
  const b = 6;
  return <div>{a} + {b} = {a+b}</div>
}
```

class → className

코드

```
(
  <div className="greeting" style={{ padding: 10, color: 'red' }}>
    {name}님 안녕하세요. <br />
    반갑습니다.
  </div>
)
```

스타일은 object로

코드

```
(
  <div className="greeting" style={{ padding: 10, color: 'red' }}>
    {name}님 안녕하세요. <br />
    반갑습니다.
  </div>
)
```

주의사항 위와 같은 Inline style의 Property name은 camelCase로 적습니다.

예시: font-size → fontSize, padding-left: paddingLeft

참고: https://www.w3schools.com/react/react_css.asp

닫는 태그 필수

HTML

```
<div>
  <input type="text">
  <br>
</div>
```

JSX

```
<div>
  <input type="text" />
  <br />
</div>
```

기존 HTML에서는 닫는 태그를 작성하지 않아도 에러가 발생하지 않으며 <input>,
같은 일부 태그의 경우 아예 닫는 태그를 생략하여 코드를 작성해도 되었으나 JSX에서는 닫는 태그를 필수로 작성하여야 합니다.

최상단 element는 반드시 하나

HTML

```
const App = () => {
  return (
    <div>Hello</div> // 에러 발생!
    <div>World</div>
  )
}
```

JSX

```
const App = () => {
  return (
    <> {/* React.Fragment */}
      <div>Hello</div>
      <div>World</div>
    </>
  )
}
```

JSX의 원칙상 최상단 Element는 한 개만 작성이 가능하기 때문에 이를 또는 <React.Fragment> 를 이용해 감쌉니다. 실제 렌더링 시에는 Fragment 안에 있는 내용만 출력됩니다.
<React.Fragment>는 간단히 <> 로 표기가 가능합니다.

02

컴포넌트

Component란

1. React에서 페이지를 구성하는 최소단위
2. Component의 이름은 대문자로 시작
3. Class Component / Function Component 로 나뉨
4. Controlled Component / Uncontrolled Component

Component란

코드

```
const MyComponent = ({ children }) => {
  return <div style={{
      padding: 20,
      color: "blue"
    }}>
      {children}
    </div>;
}
```

```
const App = () => {
  return (
    <div>
      <p>안녕</p>
      <MyComponent>반가워</MyComponent>
      <div>바이바이</div>
    </div>
  );
}
```

Component를 만들고(왼쪽) 다른 Component에서 자유롭게 활용(오른쪽)할 수 있습니다.

Component의 이름은 항상 대문자로 시작합니다.

Class Component와 Function Component

Class

```
class Hello extends Component {
  render() {
    const { name } = this.props
    return <div>{name}님 안녕하세요.</div>
  }
}
```

Function

```
const Hello = (props) => {
  const { name } = props
  return <div>{name}님 안녕하세요.</div>
}
```

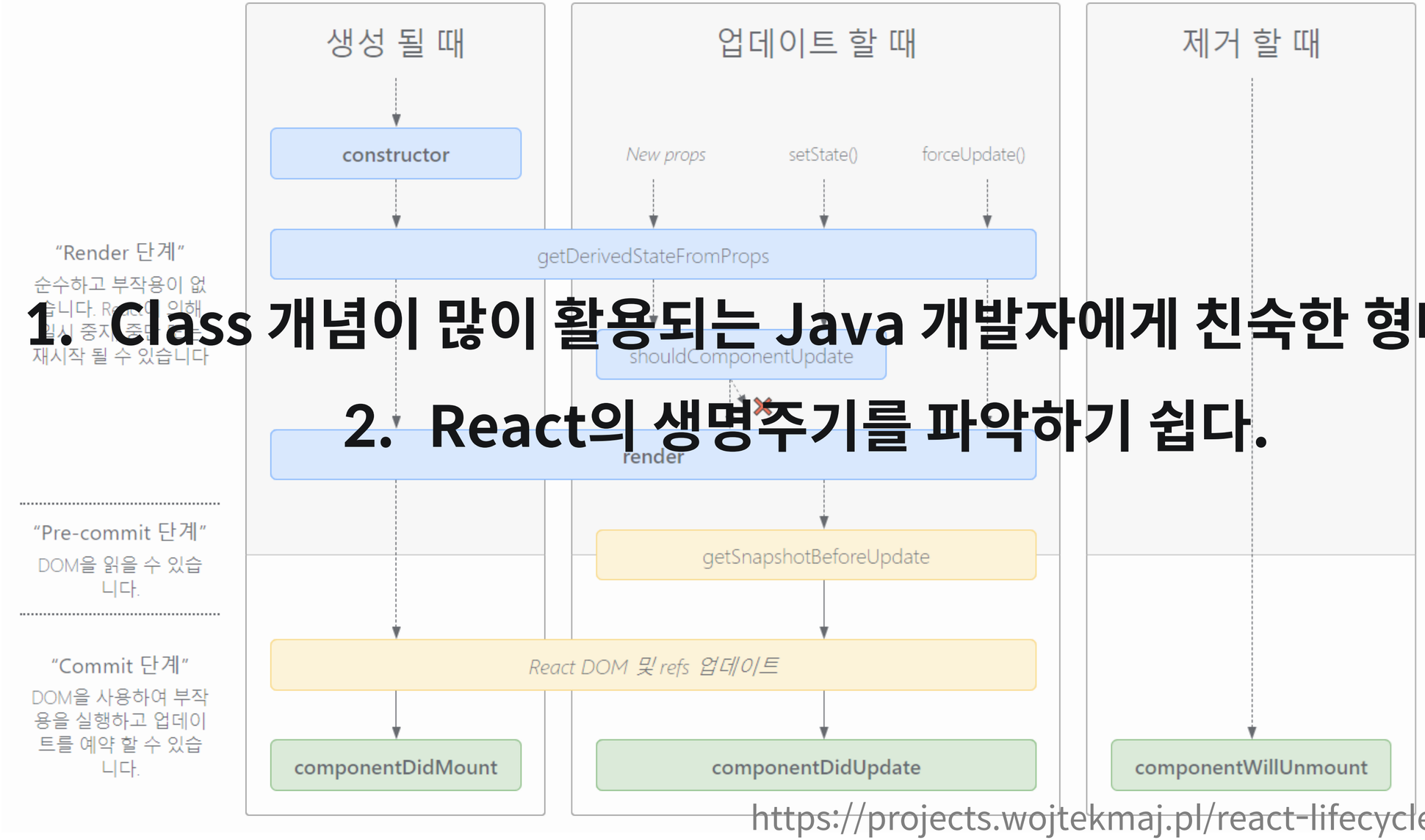
초기 React의 Component는 모두 Class Component 였습니다.

이후 v16 부터 새로운 Function Component와 Hooks 개념이 발표되었으며

현재는 Function Component가 주로 사용되고 있습니다.

React 기초 I 강의 역시 Function Component 위주로 수업이 진행됩니다.

Class Component 특징



1. Class 개념이 많이 활용되는 Java 개발자에게 친숙한 형태이다.
2. React의 생명주기를 파악하기 쉽다.

https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

너무 깊게 들어가진 않을게요…

Component의 특징

```
<MyComponent user={{name: "민수"}} color="blue">
  <div>안녕하세요.</div>
</MyComponent>
```

컴포넌트에 Attribute에 해당하는 부분을 Props(Properties)라고 합니다.

컴포넌트 안에 작성된 하위 Element를 children이라고 합니다.
그리고 children도 결국엔 props 중 하나입니다.

Component의 특징

코드

```
const MyComponent = (props) => {
  const { user, color, children } = props

  return (
    <div style={{ color }}>
      <p>{user.name}님의 하위 element는!</p>
      {children}
    </div>
  )
}
```

상위 Element로부터 전달받은 props를 활용하는 코드입니다.

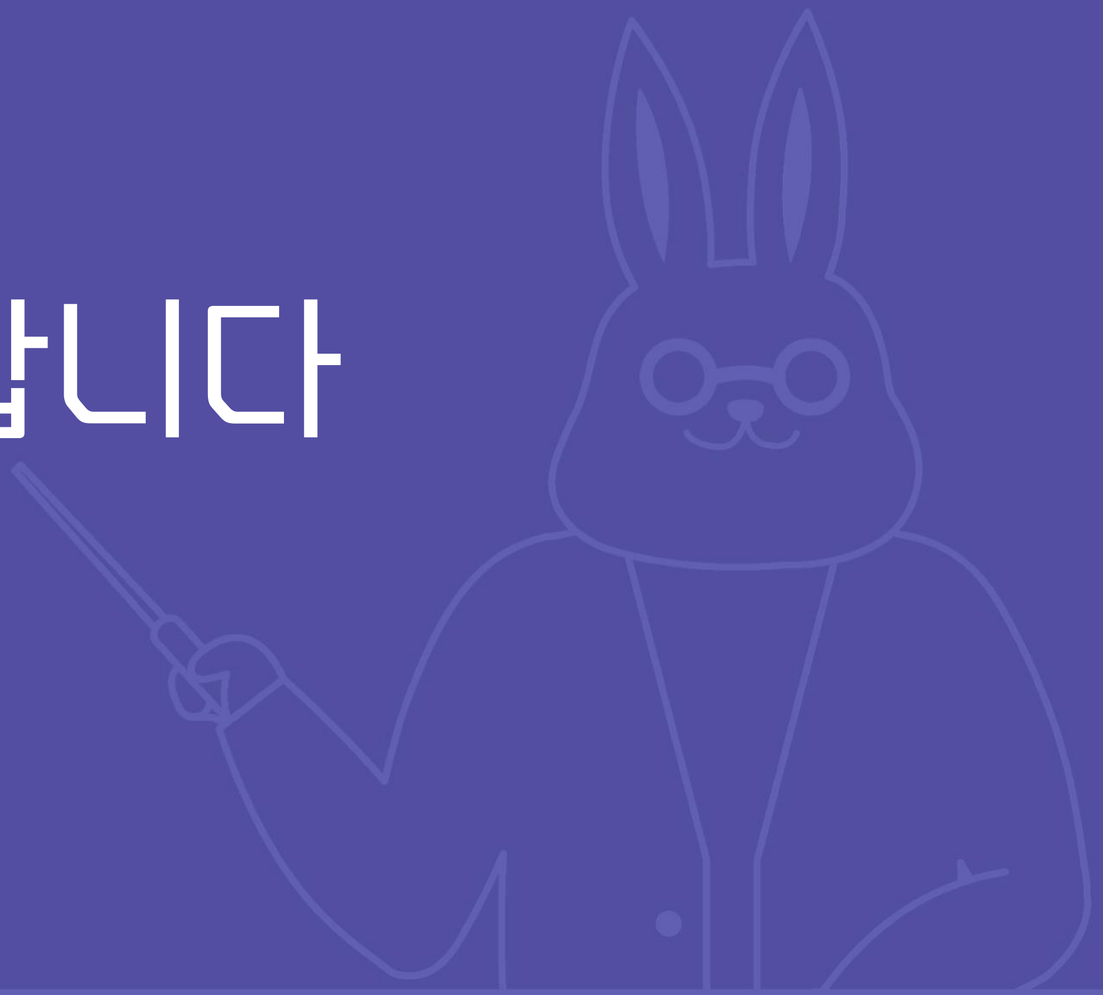
이 컴포넌트의 자식(children) 요소 역시 props 로부터 값을 받아오는 것을 볼 수 있습니다.

Component의 특징

1. 컴포넌트끼리 데이터를 주고받을 땐 Props
2. 컴포넌트 내에서 데이터를 관리할 땐 State
3. 데이터는 부모 → 자식으로만 전달

다음 시간부터 자세히 배웁니다.

감사합니다



크레딧

/* elice */

코스 매니저

콘텐츠 제작자
마로

강사
마로

감수자