04. React 시작하기

이 챕터에서는 이전에 작성한 JavaScript 코드를 React로 전환하는 방법을 배웁니다.

React 스크립트 추가하기

먼저, 프로젝트에 React를 사용하려면 외부 CDN에서 두 개의 React 스크립트를 로드해야 합니다:

- `react` : React의 핵심 라이브러리
- `react-dom` : React를 DOM과 연동하기 위한 라이브러리

```
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@18/umd/react.development.js">
</script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
    <script type="text/javascript">
      const app = document.getElementById('app');
      const header = document.createElement('h1');
      const text = 'Develop. Preview. Ship.';
      const headerContent = document.createTextNode(text);
      header.appendChild(headerContent);
      app.appendChild(header);
    </script>
  </body>
</html>
```

위 코드는 이전 챕터에서 작성한 순수 JavaScript 코드입니다. 이제 이를 React 방식으로 변경해보겠습니다.

React로 DOM 업데이트하기

React를 사용하면 DOM을 직접 조작하는 대신, `ReactDOM.createRoot()` 를 사용하여 특정 DOM 요소에 React 컴포넌트를 렌더링할 수 있습니다.

```
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@18/umd/react.development.js">
</script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
    <script>
```

```
      const app = document.getElementById('app');
      const root = ReactDOM.createRoot(app);
      root.render(<h1>Develop. Preview. Ship.</h1>);
    </script>
  </body>
</html>
```

하지만 위 코드를 브라우저에서 실행하면 다음과 같은 문법 오류가 발생합니다:

```
Uncaught SyntaxError: expected expression, got '<'
```

이는 `<h1>...</h1>`이 JavaScript 문법이 아니기 때문입니다. 이 부분은 **JSX**로 작성된 코드입니다.

JSX란?

JSX는 JavaScript의 문법 확장으로, HTML과 유사한 문법을 사용하여 UI를 정의할 수 있게 해줍니다. JSX를 사용하면 더 직관적으로 UI를 구성할 수 있습니다.

하지만 브라우저는 JSX를 직접 이해하지 못하므로, 이를 일반 JavaScript로 변환해주는 **컴파일러**가 필요합니다. 대표적인 컴파일러로는 **Babel**이 있습니다.

Babel 추가하기

프로젝트에 Babel을 추가하려면, `index.html` 파일에 다음 스크립트를 추가하세요:

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

또한, JSX 코드를 포함하는 `<script>` 태그의 `type`을 `"text/jsx"`로 지정해야 Babel이 해당 코드를 변환할 수 있습니다.

```
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@18/umd/react.development.js">
</script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
    <!-- Babel 스크립트 추가 -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js">
</script>
    <script type="text/jsx">
      const domNode = document.getElementById('app');
      const root = ReactDOM.createRoot(domNode);
      root.render(<h1>Develop. Preview. Ship.</h1>);
    </script>
```

```
  </body>
</html>
```

이제 브라우저에서 HTML 파일을 열면, "Develop. Preview. Ship."이라는 텍스트가 `h1` 태그로 표시되는 것을 확인할 수 있습니다.

선언형 vs 명령형 코드 비교

React를 사용한 선언형 코드는 다음과 같습니다:

```
<script type="text/jsx">
  const domNode = document.getElementById("app");
  const root = ReactDOM.createRoot(domNode);
  root.render(<h1>Develop. Preview. Ship.</h1>);
</script>
```

반면, 이전에 작성한 명령형 JavaScript 코드는 다음과 같습니다:

```
<script type="text/javascript">
  const app = document.getElementById('app');
  const header = document.createElement('h1');
  const text = 'Develop. Preview. Ship.';
  const headerContent = document.createTextNode(text);
  header.appendChild(headerContent);
  app.appendChild(header);
</script>
```

이 비교를 통해 React를 사용하면 반복적인 코드를 줄이고, 더 간결하게 UI를 정의할 수 있음을 알 수 있습니다.

> 💡 추가 자료:
>
> React가 UI를 어떻게 업데이트하는지에 대한 자세한 내용을 알고 싶다면, 다음 리소스를 참고하세요:
>
> - React 공식 문서: ReactDOM.createRoot()
> - React 공식 문서: root.render()

이상으로 "Getting Started with React" 챕터의 내용을 마칩니다. 다음 챕터에서는 React 컴포넌트에 대해 더 자세히 알아보게 됩니다.