

09. Next.js 설치하기

프로젝트에서 Next.js를 사용하면 더 이상 `react` 와 `react-dom` 스크립트를 `unpkg.com` 같은 곳에서 로드할 필요가 없습니다.

대신, `npm` 이나 원하는 패키지 매니저를 사용하여 로컬로 설치할 수 있습니다.

💡 참고: Next.js를 사용하려면 Node.js 버전 18.17.0 이상이 필요합니다.

[Node.js 다운로드 링크](#)

설치 방법

1. `index.html` 파일이 있는 디렉토리에 `package.json` 파일을 만듭니다.
내용은 빈 객체 `{}` 로 시작합니다:

`package.json`

```
{}
```

2. 터미널을 열고 프로젝트 루트에서 다음 명령어를 실행합니다:

Terminal

```
npm install react@latest react-dom@latest next@latest
```

설치가 완료되면 `package.json` 파일에 다음과 같은 의존성이 추가됩니다:

`package.json`

```
{
  "dependencies": {
    "next": "^14.0.3",
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  }
}
```

버전이 다소 다를 수 있지만, `next`, `react`, `react-dom` 패키지가 설치되어 있으면 문제 없습니다.

또한, 설치 과정에서 `package-lock.json` 파일도 생성됩니다.

이 파일은 정확한 패키지 버전 정보를 담고 있습니다.

기존 코드 정리하기

이제 `index.html` 파일에서 다음 코드를 삭제할 수 있습니다:

- `<html>` 과 `<body>` 태그
- `id="app"` 인 `<div>` 요소
- `react` 와 `react-dom` 스크립트
- Babel 스크립트 (`babel.min.js`)
- `<script type="text/jsx">` 태그
- `document.getElementById()` 와 `ReactDOM.createRoot()`
- `React.useState(0)` 에서 `React.` 부분

파일 상단에 import 추가

이제 상단에 다음을 추가합니다:

```
import { useState } from 'react';
```

코드는 이렇게 정리됩니다:

index.html (사실상 이제 **.js/.jsx** 파일)

```
import { useState } from 'react';

function Header({ title }) {
  return <h1>{title ? title : 'Default title'}</h1>;
}

function HomePage() {
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];
  const [likes, setLikes] = useState(0);

  function handleClick() {
    setLikes(likes + 1);
  }

  return (
    <div>
      <Header title="Develop. Preview. Ship." />
      <ul>
        {names.map((name) => (
          <li key={name}>{name}</li>
        ))}
      </ul>
    </div>
  );
}
```

```
    <button onClick={handleClick}>Like ({likes})</button>
  </div>
);
}
```

✨ 이제 이 파일은 HTML 파일이 아니라 JSX 코드만 포함하고 있으니 확장자를 `.html` 에서 `.js` 또는 `.jsx` 로 변경하세요.

Next.js 첫 페이지 만들기

Next.js는 파일 기반 라우팅(**file-system routing**) 을 사용합니다.
(즉, 코드를 작성하지 않고 폴더/파일 구조로 URL이 정해짐)

진행 방법:

1. `app` 폴더를 새로 만들고, `index.js` 파일을 그 안으로 이동합니다.
2. `index.js` 파일 이름을 `page.js` 로 변경합니다.
3. `HomePage` 컴포넌트에 `export default` 를 추가합니다:

app/page.js

```
import { useState } from 'react';

function Header({ title }) {
  return <h1>{title ? title : 'Default title'}</h1>;
}

export default function HomePage() {
  // ...
}
```

개발 서버 실행하기

다음으로, 개발 서버를 실행해봅시다.

`package.json` 파일에 `"dev"` 스크립트를 추가하세요:

package.json

```
{
  "scripts": {
    "dev": "next dev"
  },
}
```

```
"dependencies": {
  "next": "^14.0.3",
  "react": "^18.3.1",
  "react-dom": "^18.3.1"
}
```


터미널에서 다음 명령어를 실행합니다:

```
npm run dev
```

변화 확인:

1. `localhost:3000` 에 접속하면 **에러 메시지**가 뜹니다:

에러: `useState`를 사용하는 컴포넌트는 Client Component여야 합니다.

에러 메시지 예시

이는 Next.js가 기본적으로 **Server Components**를 사용하기 때문입니다.
Server Component에서는 `useState` 같은 클라이언트 기능을 쓸 수 없습니다.
→ 다음 챕터에서 이 문제를 해결할 것입니다.

2. `app` 폴더에 자동으로 `layout.js` 파일이 생성됩니다:

app/layout.js

```
export const metadata = {
  title: 'Next.js',
  description: 'Generated by Next.js',
};


export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>{children}</body>
    </html>
  );
}
```

이 파일은 모든 페이지에 공통 적용할 레이아웃을 정의하는 곳입니다.
(예: 네비게이션 바, 푸터 등)

요약

이번 장에서 우리는:

- React와 Babel 스크립트를 제거했습니다.
- 첫 번째 Next.js 페이지를 만들었습니다.
- 개발 서버를 구동했습니다.
- Next.js가 서버 컴포넌트와 클라이언트 컴포넌트를 구분하는 방식에 대해 힌트를 얻었습니다.

 추가 학습 자료:

- [Next.js Routing 기본 개념](#)
- [라우트 정의하기](#)
- [Pages와 Layouts](#)

Chapter 9 완료!

- Next.js 설치 및 세팅 완료
- 첫 번째 페이지 생성 완료

다음은:

 [Chapter 10: Server와 Client 컴포넌트의 차이 배우기](#)

정리:

- 코드: 영어 원문 그대로
- 설명: 자연스럽게 자세한 한글 번역
- 이미지 링크 정상 복구
- 단계별로 깔끔하게 설명

바로 **Chapter 10 (Server와 Client Components)** 도 이어서 번역해드릴까요?
진행할까요? 