<script>

내장 브라우저 `<script>` 컴포넌트를 사용하면 문서에 스크립트를 추가할 수 있습니다.

```
<script> alert("hi!") </script>
```

레퍼런스

`<script>`

문서에 인라인 또는 외부 스크립트를 추가하려면 내장 브라우저 `<script>` 컴포넌트를 렌더링하세요. `<script>` 는 어떤 컴포넌트에서든 렌더링할 수 있으며, React는 특정 경우에 해당 DOM 요소를 문서의 `<head>` 에 배치하고 중복된 동일 스크립트를 제거합니다.

```
<script> alert("hi!") </script>
<script src="script.js" />
```

아래 예시를 참고하세요.

Props

`<script>` 는 모든 공통 엘리먼트 Props를 지원합니다.

`children` 또는 `src` 속성을 가져야 합니다.

- `children` : 문자열. 인라인 스크립트의 소스 코드.
- `src` : 문자열. 외부 스크립트의 URL.

지원하는 다른 속성들:

- `async` : 불리언 값. 브라우저가 문서의 남은 부분을 처리할 때까지 스크립트 실행을 연기할 수 있도록 합니다. 이는 성능을 위한 우선적인 동작 방식입니다.
- `crossOrigin` : 문자열. 사용할 CORS 정책입니다. 가능한 값은 `anonymous` 와 `use-credentials` 입니다.
- `fetchPriority` : 문자열. 여러 스크립트를 동시에 가져올 때 브라우저가 스크립트를 우선순위로 순위 지정할 수 있도록 합니다. `"high"` , `"low"` , 또는 `"auto"` (기본값)일 수 있습니다.
- `integrity` : 문자열. 스크립트의 암호화 해시로, 진위성을 검증합니다.
- `noModule` : 불리언 값. ES 모듈을 지원하는 브라우저에서 스크립트를 비활성화합니다. ES 모듈을 지원하지 않는 브라우저에 대한 대체 스크립트를 허용합니다.
- `nonce` : 문자열. 엄격한 콘텐츠 보안 정책을 사용할 때 리소스를 허용하기 위해 사용하는 암호화된 nonce입니다.
- `referrer` : 문자열. 스크립트를 가져오고 스크립트가 다시 가져온 리소스를 가져올 때 보낼 Referer 헤더를 지정합니다.
- `type` : 문자열. 스크립트가 전통적인 스크립트, ES 모듈 또는 import 맵인지를 나타냅니다.

React의 스크립트 특수 처리를 비활성화하는 속성들:

- `onError` : 함수. 스크립트의 로딩을 실패하였을 때 호출됩니다.
- `onLoad` : 함수. 스크립트의 로딩을 완료하였을 때 호출됩니다.

React에서 **권장하지 않는** 속성들:

- `blocking` : 문자열. `"render"` 로 설정하면 페이지가 스크립트시트를 로드할 때까지 브라우저에게 페이지를 렌더링하지 않도록 지시합니다. React는 Suspense를 사용하여 더 세밀한 제어를 제공합니다.
- `defer` : 문자열. 문서가 로딩될 때까지 브라우저가 스크립트를 실행하지 못하도록 합니다. 스트리밍 서버에 렌더링된 컴포넌트와 호환되지 않습니다. 대신 `async` 속성을 사용하세요.

특별한 렌더링 동작

React는 `<script>` 컴포넌트를 문서의 `<head>` 로 이동시키고, 중복된 동일 스크립트를 제거합니다.

이 동작을 사용하려면 `src` 와 `async={true}` 속성을 제공하세요. React는 `src` 가 동일한 경우에만 중복된 스크립트를 제거합니다. 스크립트를 안전하게 이동하려면 `async` 속성이 반드시 `true` 여야 합니다.

이 특별한 처리에는 두 가지 주의 사항이 있습니다.

- React는 스크립트를 렌더링한 후에 속성 변경을 무시합니다. (개발 환경에서는 이러한 경우에 경고가 발생합니다.)
- React는 컴포넌트를 마운트 해제한 후에도 DOM에 스크립트를 남길 수 있습니다. (스크립트는 DOM에 삽입될 때 한 번만 실행되므로 이것은 영향을 미치지 않습니다.)

사용법

외부 스크립트 렌더링

특정 스크립트에 의존하여 컴포넌트를 올바르게 표시해야 한다면, 컴포넌트 내에서 `<script>` 를 렌더링할 수 있습니다.
그러나 스크립트 로딩이 완료되기 전에 컴포넌트가 커밋될 수 있습니다.
`load` 이벤트가 발생하면 스크립트 내용에 따라 시작할 수 있습니다. 예를 들어 `onLoad` prop 을 이용할 수 있습니다.

React는 동일한 `src` 를 가진 중복된 스크립트를 제거하여 여러 컴포넌트를 렌더링하더라도 그 중 하나만 DOM에 삽입합니다.

App.js ShowRenderedHTML.js 새로고침 Clear 포크

```
import ShowRenderedHTML from './ShowRenderedHTML.js';

function Map({lat, long}) {
  return (
    <>
      <script async src="map-api.js" onLoad={() => console.log('script loaded')} />
      <div id="map" data-lat={lat} data-long={long} />
    </>
  );
```

```
}

export default function Page() {
```

자세히 보기

중요합니다!

스크립트를 사용하려는 경우, `preinit` 함수를 호출하는 것이 유리할 수 있습니다. 이 함수를 호출하면 `<script>` 컴포넌트를 그냥 렌더링하는 것보다 브라우저가 스크립트를 더 빨리 가져오도록 할 수 있습니다. 예를 들어 HTTP Early Hints 응답을 통해 스크립트를 더 빨리 가져올 수 있습니다.

인라인 스크립트 렌더링

인라인 스크립트를 포함하려면 `<script>` 컴포넌트를 자식으로 스크립트 소스 코드와 함께 렌더링하세요. 인라인 스크립트는 중복 제거되거나 문서의 `<head>` 로 이동하지 않습니다.

App.js ShowRenderedHTML.js 새로고침 Clear 포크

```
import ShowRenderedHTML from './ShowRenderedHTML.js';

function Tracking() {
  return (
    <script>
      ga('send', 'pageview');
    </script>
  );
}

export default function Page() {
  return (
```

자세히 보기

이전
<meta>

다음
<style>

Meta Open Source

uwu?

React 학습하기

- 빠르게 시작하기
- 설치하기
- UI 표현하기
- 상호작용성 더하기
- State 관리하기
- 탈출구

API 참고서

- React APIs
- React DOM APIs

커뮤니티

- 행동 강령
- 팀 소개
- 문서 기여자
- 감사의 말

더 보기

- 블로그
- React Native
- 개인 정보 보호
- 약관