'use server'

React 서버 컴포넌트

`'use server'`는 React 서버 컴포넌트와 함께 사용합니다.

`'use server'`를 사용하여 클라이언트 측 코드에서 호출할 수 있는 서버 측 함수를 표시합니다.

레퍼런스

`'use server'`

함수를 클라이언트에서 호출할 수 있음을 표시하기 위해, 비동기 함수의 최상단에 `'use server';`를 추가하세요. 이를 *서버 함수*라고 부릅니다.

```
async function addToCart(data) {
  'use server';
```

```
  // ...
}
```

클라이언트에서 서버 함수를 호출하면, 전달된 모든 인수의 직렬화된 사본을 포함한 네트워크 요청을 서버로 전송합니다. 서버 함수가 값을 반환하면, 그 값을 직렬화하여 클라이언트로 반환합니다.

함수 각각에 `'use server'` 를 표기하는 대신, 파일의 최상단에 지시어를 추가하여 파일의 모든 내보내기Export를 클라이언트를 포함한 모든 곳에서 사용할 수 있는 서버 함수로 표기할 수 있습니다.

주의 사항

- `'use server'` 는 함수 또는 모듈의 최상단에 있어야 합니다. `import` 를 포함한 다른 코드보다 위에 있어야 합니다. (지시어 위의 주석은 괜찮습니다.) 백틱이 아닌 단일 또는 이중 따옴표로 작성해야 합니다.
- `'use server'` 는 서버 측 파일에서만 사용할 수 있습니다. 결과적으로 생성된 서버 함수는 Props를 통해 클라이언트 컴포넌트로 전달할 수 있습니다. 지원되는 직렬화 타입을 참고하세요.
- 서버 함수를 클라이언트 코드에서 가져오기Import 위해, 지시어를 모듈 수준에서 사용해야 합니다.
- 기본 네트워크 호출이 항상 비동기적이므로, `'use server'` 는 비동기 함수에서만 사용할 수 있습니다.
- 항상 서버 함수의 인수를 신뢰할 수 없는 입력으로 취급하고 모든 변경을 검토하세요. 보안 고려사항을 확인하세요.
- 서버 함수는 Transition 안에서 호출되어야 합니다. `<form action>` 또는 `formAction` 으로 전달된 서버 함수는 자동으로 Transition 내에서 호출됩니다.
- 서버 함수는 서버 측 상태를 업데이트하는 Mutation을 위해 설계되었으며, 데이터 가져오기Fetching에는 권장하지 않습니다. 따라서, 서버 함수를 구현하는 프레임워크는 일반적으로 한 번에 하나의 작업만 처리하며, 반환 값을 캐시하는 방법을 제공하지 않습니다.

보안 고려사항

서버 함수에 대한 인수는 클라이언트에서 완전히 제어됩니다. 보안을 위해 항상 신뢰할 수 없는 입력으로 취급하여, 인수를 적절하게 검증하고 이스케이프 하는지 확인하세요.

모든 서버 함수에서 로그인한 사용자가 해당 작업을 수행할 수 있는지 확인하세요.

개발중이에요!

서버 함수에서 중요한 데이터를 전송하지 않기 위해, 고유한 값과 객체를 클라이언트 코드로 전달하는 것을 방지하기 위한 실험적인 Taint API가 있습니다.

experimental_taintUniqueValue와 experimental_taintObjectReference를 참고하세요.

직렬화 가능 인수와 반환값

클라이언트 코드가 네트워크를 통해 서버 함수를 호출하므로, 전달하는 모든 인수는 직렬화 가능해야 합니다.

다음은 서버 함수의 인수로 지원되는 타입입니다.

- 원시 자료형
  - string
  - number
  - bigint
  - boolean
  - undefined
  - null
  - symbol (`Symbol.for` 를 통해 전역 심볼 레지스트리에 등록된 심볼만 해당)
- 직렬화할 수 있는 값을 포함하는 이터러블
  - String
  - Array
  - Map
  - Set
  - TypedArray와 ArrayBuffer
- Date
- FormData 인스턴스
- 일반 객체 (직렬화할 수 있는 프로퍼티를 사용하여 객체 이니셜라이저로 생성된 객체)

- 서버 함수로서의 함수
- Promise

단, 다음은 지원되지 않습니다.

- React 엘리먼트 또는 JSX
- 컴포넌트 함수 또는 서버 함수가 아닌 다른 함수를 포함하는 함수
- 클래스
- 클래스의 인스턴스인 객체(언급된 내장 객체 제외)또는 null 프로토타입이 있는 객체
- 전역에 등록되지 않은 Symbol (예: `Symbol('my new symbol')`)
- Events from event handlers

지원되는 직렬화 가능한 반환 값은 경계 클라이언트 컴포넌트의 직렬화 가능한 Props와 동일합니다.

사용법

Server Functions in forms

서버 함수의 가장 일반적인 사용 사례는, 데이터를 변경하는 서버 함수를 호출하는 것입니다. 브라우저의 HTML 폼 엘리먼트는 사용자가 Mutation을 제출하는 전통적인 접근 방식입니다. React 서버 컴포넌트를 통해, React는 폼Form에서 액션으로 사용되는 서버 함수에 대한 최상의 지원을 제공합니다.

여기, 사용자가 사용자 이름을 요청할 수 있는 폼Form이 있습니다.

```
// App.js

async function requestUsername( formData ) {
  'use server';
  const username = formData.get('username');
  // ...
}

export default function App() {
  return (
    <form action={requestUsername}>
      <input type="text" name="username" />
```

```
      <button type="submit">Request</button>
    </form>
  );
}
```

예시에서 `requestUsername` 은 `<form>` 을 통한 서버 함수입니다. 사용자가 이 폼Form을 제출하면 서버 함수인 `requestUsername` 에 네트워크 요청을 보냅니다. 폼Form에서 서버 함수를 호출할 때, React는 폼Form의 `formData` 를 서버 함수의 첫 번째 인자로 제공합니다.

서버 함수를 폼 `action` 에 전달하여, React는 폼을 점진적 향상할 수 있습니다. 이것은 자바스크립트 번들을 로드하기 전에 양식을 제출할 수 있다는 것을 의미합니다.

폼에서 반환 값 처리

In the username request form, there might be the chance that a username is not available. `requestUsername` should tell us if it fails or not.

점진적 향상을 지원하며 서버 함수의 결과를 기반으로 UI를 업데이트하려면, `useActionState` 를 사용하세요.

```
// requestUsername.js
'use server';

export default async function requestUsername(formData) {
  const username = formData.get('username');
  if (canRequest(username)) {
    // ...
    return 'successful';
  }
  return 'failed';
}
```

```
// UsernameForm.js
'use client';

import { useActionState } from 'react';
import requestUsername from './requestUsername';

function UsernameForm() {
```

```
  const [state, action] = useActionState(requestUsername, null, 'n/a');

  return (
    <>
      <form action={action}>
        <input type="text" name="username" />
        <button type="submit">Request</button>
      </form>
      <p>Last submission request returned: {state}</p>
    </>
  );
}
```

대부분의 Hook과 마찬가지로 `useActionState`는 클라이언트 코드에서만 호출할 수 있습니다.

`<form>` 외부에서 서버 함수 호출하기

서버 함수는 노출된 서버 엔드포인트이며 클라이언트 코드 어디에서나 호출할 수 있습니다.

폼Form 외부에서 서버 함수를 사용할 때, Transition에서 서버 함수를 호출하면 로딩 인디케이터Loading Indicator를 표시하고, 낙관적 상태 업데이트를 표시하며 예기치 않은 오류를 처리할 수 있습니다. 폼은 Transition의 서버 함수를 자동으로 래핑합니다.

```
import incrementLike from './actions';
import { useState, useTransition } from 'react';

function LikeButton() {
  const [isPending, startTransition] = useTransition();
  const [likeCount, setLikeCount] = useState(0);

  const onClick = () => {
    startTransition(async () => {
      const currentCount = await incrementLike();
      setLikeCount(currentCount);
    });
  };

  return (
    <>
```

```
      <p>Total Likes: {likeCount}</p>
      <button onClick={onClick} disabled={isPending}>Like</button>;
    </>
  );
}
```

```
// actions.js
'use server';

let likeCount = 0;
export default async function incrementLike() {
  likeCount++;
  return likeCount;
}
```

서버 함수의 반환 값을 읽으려면 반환된 Promise를 `await` 해야합니다.

이전

'use client'

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관