서버 함수

React 서버 컴포넌트

서버 함수는 React 서버 컴포넌트에서 사용합니다.

참고: 2024년 9월까지, 우리는 모든 서버 함수를 "서버 액션"으로 불렀습니다. 만약 서버 함수를 action prop으로 전달하거나 action 내부에서 호출된다면 이는 서버 액션이지만, 모든 서버 함수가 서버 액션은 아닙니다. 이 문서의 명명 규칙은 서버 함수가 여러 용도로 사용될 수 있다는 점을 반영하여 업데이트했습니다.

서버 함수Server Functions를 사용하면 클라이언트 컴포넌트가 서버에서 실행되는 비동기 함수를 호출할 수 있습니다.

중요합니다!

서버 함수를 지원하려면 어떻게 해야 하나요?

React 19의 서버 함수는 안정적이며 마이너Minor 버전 간에는 변경되지 않습니다. 그러나 React 서버 컴포넌트 번들러나 프레임워크에서 서버 함수를 구현하는 데 사용되는 기본 API는 유의적 버전SemVer을 따르지 않으며 React 19.x의 마이너Minor 버전 간에 변경될 수 있습니다.

서버 함수를 번들러나 프레임워크로 지원하려면, 특정 React 버전에 고정하거나 Canary 릴리즈를 사용하는 것을 권장합니다. 향후 서버 함수를 구현하는 데 사용되는 API를 안정화하기 위해 번들러 및 프레임워크와 계속 협력할 것입니다.

서버 함수가 `"use server"` 지시어로 정의되면, 프레임워크는 자동으로 서버 함수에 대한 참조를 생성하고 해당 참조를 클라이언트 컴포넌트에 전달합니다. 클라이언트에서 해당 함수를 호출하면, React는 서버에 함수를 실행하라는 요청Request을 보내고 결과를 반환합니다.

서버 함수는 서버 컴포넌트에서 생성하여 클라이언트 컴포넌트에 Props로 전달할 수 있으며, 클라이언트 컴포넌트에서 가져와서 사용할 수도 있습니다.

사용법

서버 컴포넌트에서 서버 함수 만들기

서버 컴포넌트는 `"use server"` 지시어로 서버 함수를 정의할 수 있습니다.

```
// 서버 컴포넌트
import Button from './Button';

function EmptyNote () {
  async function createNote() {
    // 서버 함수
    'use server';

    await db.notes.create();
  }

  return <Button onClick={createNote}/>;
}
```

React가 `EmptyNote` 서버 컴포넌트를 렌더링할 때, `createNoteAction` 함수에 대한 참조를 생성하고, 그 참조를 `Button` 클라이언트 컴포넌트에 전달합니다. 버튼을 클릭하면, React는 제공된 참조를 통해 `createNoteAction` 함수를 실행하도록 서버에 요청Request을 보냅니다.

```
"use client";

export default function Button({onClick}) {
  console.log(onClick);
  // {$$typeof: Symbol.for("react.server.reference"), $$id: 'createNoteAction'}
  return <button onClick={() => onClick()}>Create Empty Note</button>
```

```
}
```

자세한 내용은 `"use server"` 문서를 참조하세요.

클라이언트 컴포넌트에서 서버 함수 가져오기

클라이언트 컴포넌트는 `"use server"` 지시어를 사용하는 파일에서 서버 함수를 가져올 수 있습니다.

```
"use server";

export async function createNote() {
  await db.notes.create();
}
```

번들러가 `EmptyNote` 클라이언트 컴포넌트를 빌드할 때, 번들에서 `createNote` 함수에 대한 참조를 생성합니다. 버튼을 클릭하면, React는 제공된 참조를 통해 `createNote` 함수를 실행하도록 서버에 요청Request을 보냅니다.

```
"use client";

import {createNote} from './actions';

function EmptyNote() {
  console.log(createNote);
  // {$$typeof: Symbol.for("react.server.reference"), $$id: 'createNote'}
  <button onClick={() => createNote()} />
}
```

자세한 내용은 `"use server"` 문서를 참조하세요.

액션으로 서버 함수 구성하기

서버 함수는 클라이언트의 액션Action과 함께 구성할 수 있습니다.

```
"use server";

export async function updateName(name) {
  if (!name) {
    return {error: 'Name is required'};
  }
  await db.users.updateName(name);
}
```

```
"use client";

import {updateName} from './actions';

function UpdateName() {
  const [name, setName] = useState('');
  const [error, setError] = useState(null);

  const [isPending, startTransition] = useTransition();

  const submitAction = async () => {
    startTransition(async () => {
      const {error} = await updateName(name);
      if (error) {
        setError(error);
      } else {
        setName('');
      }
    })
  }

  return (
    <form action={submitAction}>
      <input type="text" name="name" disabled={isPending}/>
      {error && <span>Failed: {error}</span>}
    </form>
  )
}
```

이렇게 하면 클라이언트의 액션으로 래핑하여 서버 함수의 `isPending` 상태에 접근할 수 있습니다.

자세한 내용은 `<form>` 외부에서 서버 함수 호출하기 문서를 참조하세요.

서버 함수를 사용한 폼 액션

서버 함수는 React 19의 새로운 폼Form 기능과 함께 동작합니다.

서버 함수를 폼에 전달하여 폼을 서버에 자동으로 제출할 수 있습니다.

```
"use client";

import {updateName} from './actions';

function UpdateName() {
  return (
    <form action={updateName}>
      <input type="text" name="name" />
    </form>
  )
}
```

폼 제출이 성공하면, React는 자동으로 폼을 재설정합니다. `useActionState` 를 추가하여 대기Pending 상태 혹은 마지막 응답Response에 접근하거나, 점진적 향상을 지원할 수 있습니다.

자세한 내용은 폼Form에서의 서버 함수 문서를 참조하세요.

`useActionState` 를 사용한 서버 함수

액션 대기Pending 상태와 마지막으로 반환된 응답Response에 접근하는 일반적인 경우에는 `useActionState` 를 사용하여 서버 함수를 호출할 수 있습니다.

```
"use client";

import {updateName} from './actions';

function UpdateName() {
```

```
  const [state, submitAction, isPending] = useActionState(updateName, {error: null})

  return (
    <form action={submitAction}>
      <input type="text" name="name" disabled={isPending}/>
      {state.error && <span>Failed: {state.error}</span>}
    </form>
  );
}
```

서버 함수 함께 `useActionState` 를 사용하는 경우, React는 Hydration이 완료되기 전에 입력된 폼 제출을 자동으로 다시 실행합니다. 즉, 사용자는 앱이 Hydration 되기 전에도 앱과 상호작용을 할 수 있습니다.

자세한 내용은 `useActionState` 문서를 참조하세요.

`useActionState` 를 통한 점진적 향상

서버 함수는 `useActionState` 의 세 번째 인수를 통한 점진적 향상도 지원합니다.

```
"use client";

import {updateName} from './actions';

function UpdateName() {
  const [, submitAction] = useActionState(updateName, null, `/name/update`);

  return (
    <form action={submitAction}>
      ...
    </form>
  );
}
```

permalink 가 `useActionState` 에 제공될 때, 자바스크립트 번들이 로드되기 전에 폼이 제출되면 React는 제공된 URL로 리디렉션합니다.

자세한 내용은 `useActionState` 문서를 참조하세요.

이전
서버 컴포넌트

다음
지시어

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