captureOwnerStack

`captureOwnerStack` 는 개발 환경에서 현재 Owner Stack을 읽고, 사용할 수 있다면 문자열 반환합니다.

```
const stack = captureOwnerStack();
```

레퍼런스

captureOwnerStack()

`captureOwnerStack` 을 호출하여 현재 Owner Stack을 가져옵니다.

```
import * as React from 'react';

function Component() {
  if (process.env.NODE_ENV !== 'production') {
    const ownerStack = React.captureOwnerStack();
    console.log(ownerStack);
```

```
    }
}
```

매개변수

`captureOwnerStack` 는 매개변수를 받지 않습니다.

반환값

`captureOwnerStack` 은 `string` 이나 `null` 을 반환합니다.

Owner Stacks은 다음 경우에 사용할 수 있습니다.

- 컴포넌트 렌더링 시
- Effect (예: `useEffect`)
- React 이벤트 핸들러 (예: `<button onClick={...} />`)
- React 오류 핸들러 (React 루트 옵션 `onCaughtError` , `onRecoverableError` , `onUncaughtError`)

Owner Stack을 사용할 수 없는 경우, `null` 을 반환합니다. (문제해결: Owner Stack이 `null` 인 경우)

주의 사항

- Owner Stack은 개발 환경에서만 사용할 수 있습니다. `captureOwnerStack` 은 개발 환경 밖에서는 항상 `null` 을 반환합니다.

자세히 살펴보기

Owner Stack vs Component Stack

자세히 보기

사용법

커스텀 오류 오버레이 개선하기

```
import { captureOwnerStack } from "react";
import { instrumentedConsoleError } from "./errorOverlay";

const originalConsoleError = console.error;
console.error = function patchedConsoleError(...args) {
  originalConsoleError.apply(console, args);
  const ownerStack = captureOwnerStack();
  onConsoleError({
    // Keep in mind that in a real application, console.error can be
    // called with multiple arguments which you should account for.
    consoleMessage: args[0],
    ownerStack,
  });
};
```

`console.error` 호출을 가로채서 오류 오버레이에 표시하고 싶다면, `captureOwnerStack` 을 호출하여 `OwnerStack` 을 포함할 수 있습니다.

index.js errorOverlay.js App.js 새로고침 Clear 포크

```
import { captureOwnerStack } from "react";
import { createRoot } from "react-dom/client";
import App from './App';
import { onConsoleError } from "./errorOverlay";
import './styles.css';

const originalConsoleError = console.error;
console.error = function patchedConsoleError(...args) {
  originalConsoleError.apply(console, args);
  const ownerStack = captureOwnerStack();
  onConsoleError({
    // Keep in mind that in a real application, console.error can be
```

자세히 보기

문제 해결

Owner Stack이 `null` 인 경우

`captureOwnerStack` 이 `setTimeout` 콜백과 같이 React가 제어하지 않는 함수 바깥에서 호출됐을 경우, `fetch` 호출 후, 커스텀 DOM 이벤트 핸들러 등에서는 Owner Stack이 `null` 이 됩니다. 렌더링 중이나 Effect, React 이벤트 핸들러, React 오류 핸들러 (예: `hydrateRoot#options.onCaughtError`) 내에서만 생성됩니다.

아래 예시에서, 버튼을 클릭하면 빈 Owner Stack이 로그로 출력됩니다. 그 이유는 `captureOwnerStack` 이 커스텀 이벤트 핸들러 내에서 호출되었기 때문입니다. Owner Stack은 더 이른 시점, 예를 들어 이펙트 내부에서 `captureOwnerStack` 를 호출하도록 이동시켜야 올바르게 캡처할 수 있습니다.

App.js 다운로드 새로고침 Clear 포크

```
import {captureOwnerStack, useEffect} from 'react';

export default function App() {
  useEffect(() => {
    // Should call `captureOwnerStack` here.
    function handleEvent() {
      // Calling it in a custom DOM event handler is too late.
      // The Owner Stack will be `null` at this point.
      console.log('Owner Stack: ', captureOwnerStack());
```

```
    }
```

자세히 보기

`captureOwnerStack` 을 사용할 수 없는 경우

`captureOwnerStack` 은 개발 환경 빌드에서만 Export됩니다. 프로덕션 환경 빌드에서는 `undefined` 입니다. `captureOwnerStack` 이 개발과 프로덕션이 모두 번들링되는 파일에서 사용될 때는 네임스페이스 `import` 를 사용하고 조건부로 접근해야 합니다.

```
// Don't use named imports of `captureOwnerStack` in files that are bundled for develo
import {captureOwnerStack} from 'react';
// Use a namespace import instead and access `captureOwnerStack` conditionally.
import * as React from 'react';

if (process.env.NODE_ENV !== 'production') {
  const ownerStack = React.captureOwnerStack();
  console.log('Owner Stack', ownerStack);
}
```

이전
cacheSignal

다음
createContext

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