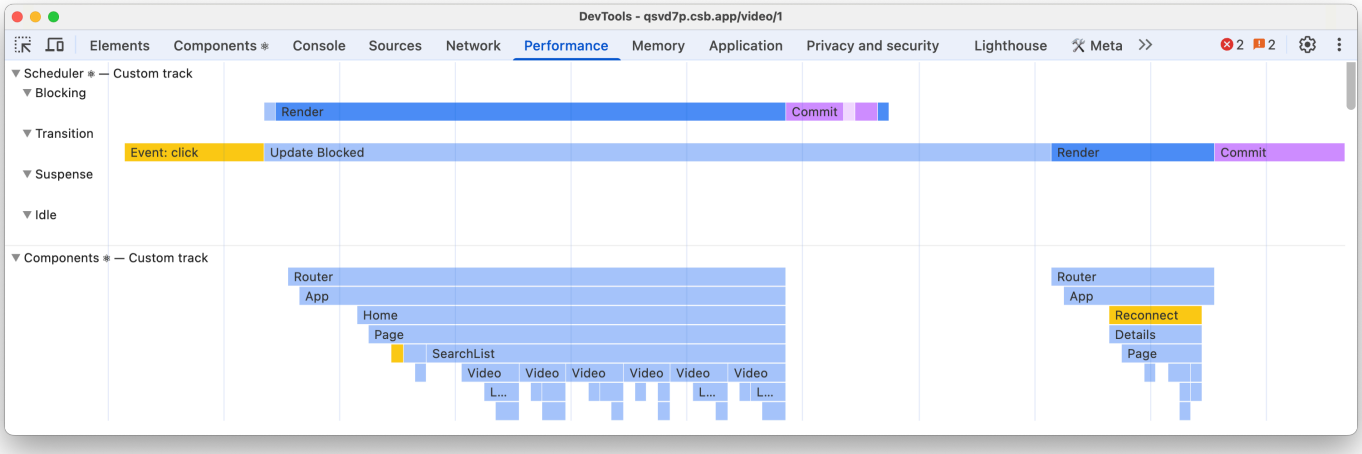
React Performance 트랙

React Performance(성능) 트랙은 브라우저 개발자 도구의 Performance 패널 타임라인에 표시되는 특수한 맞춤형 항목입니다.

이 트랙은 개발자에게 React 애플리케이션의 성능에 대한 포괄적인 인사이트를 제공하도록 설계되었습니다. 네트워크 요청, 자바스크립트 실행, 이벤트 루프 활동과 같은 주요 데이터 소스와 함께 React 전용 이벤트와 메트릭을 시각화하며, 애플리케이션 동작을 완전히 이해할 수 있도록 Performance(성능) 패널 내의 통합된 타임라인에서 모든 정보가 동기화되어 표시됩니다.



사용법

React Performance 트랙은 개발 및 프로파일링 빌드에서만 사용할 수 있습니다.

- **개발**: 기본적으로 활성화되어 있습니다.
- **프로파일링**: 기본적으로 Scheduler 트랙만 활성화되어 있습니다. 컴포넌트 트랙은 `<Profiler>` 로 감싼 서브트리 내의 컴포넌트만 나열합니다. React 개발자 도구 확장 프로그램을 활성화했다면, `<Profiler>` 로 감싸지 않았더라도 모든 컴포넌트가 컴포넌트 트랙에 포함됩니다. 서버 트랙은 프로파일링 빌드에서 사용할 수 없습니다.

활성화된 경우 확장성 API를 제공하는 브라우저의 Performance(성능) 패널로 기록한 트레이스에서 트랙이 자동으로 표시됩니다.

주의하세요!

React Performance 트랙을 구동하는 프로파일링 도구는 추가적인 오버헤드를 발생시키므로, 프로덕션 빌드에서는 기본적으로 비활성화되어 있습니다.
서버 컴포넌트 및 서버 요청(Server Requests) 트랙은 개발 빌드에서만 사용할 수 있습니다.

프로파일링 빌드 사용하기

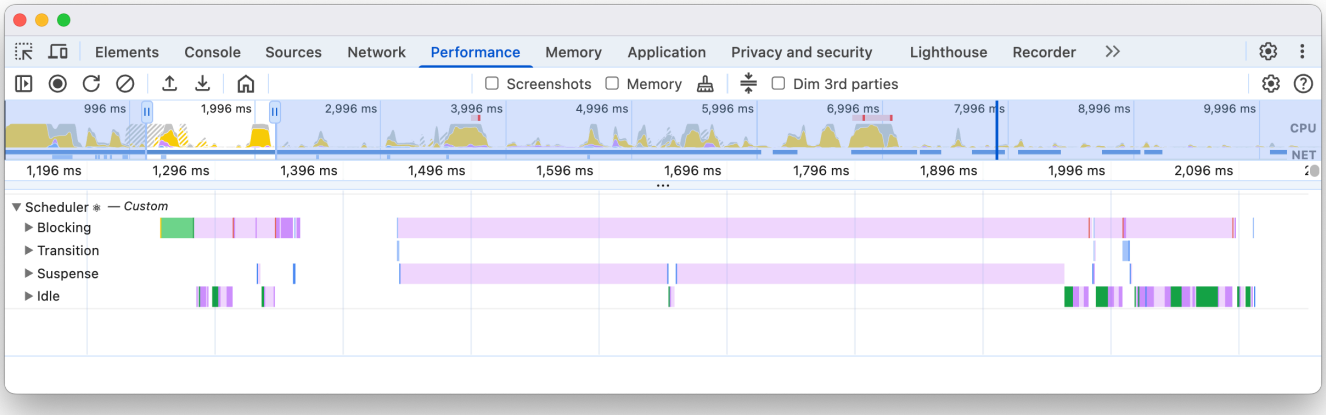
프로덕션 및 개발 빌드 외에도 React는 특수한 프로파일링 빌드를 제공합니다.
프로파일링 빌드를 사용하려면 `react-dom/client` 대신 `react-dom/profiling` 을 사용해야 합니다. 모든 `react-dom/client` 임포트(import) 구문을 수동으로 수정하는 대신 **번들러 별칭(alias)** 을 설정하여 빌드 시점에 `react-dom/client` 가 `react-dom/profiling` 을 가리키도록 하는 것을 권장합니다.
사용 중인 프레임워크에 React 프로파일링 빌드를 활성화하는 기능이 내장되어 있을 수 있습니다.

트랙

Scheduler

Scheduler는 우선순위가 다른 작업을 관리하는 데 사용되는 React의 내부 개념입니다. 이 트랙은 특정 우선순위의 작업을 나타내는 네 개의 서브트랙으로 구성됩니다.

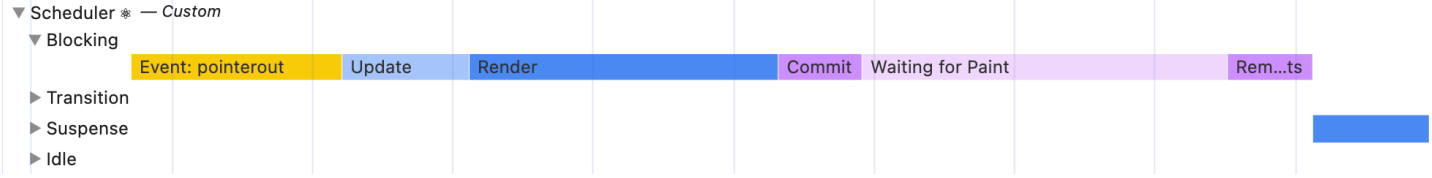
- **Blocking** - 사용자 상호작용에 의해 시작될 수 있는 동기 업데이트입니다.
- **Transition** - 백그라운드에서 발생하는 논블로킹 작업으로 주로 `startTransition`을 통해 시작됩니다.
- **Suspense** - 폴백(fallback) 표시나 콘텐츠 노출 등 Suspense 경계와 관련된 작업입니다.
- **Idle** - 우선순위가 더 높은 다른 작업이 없을 때 수행되는 가장 낮은 우선순위의 작업입니다.



렌더링

모든 렌더링 과정(render pass)은 타임라인에서 확인할 수 있는 여러 단계로 구성됩니다.

- **Update** - 새로운 렌더링 과정을 발생시킨 원인입니다.
- **Render** - React가 컴포넌트의 렌더링 함수를 호출하여 업데이트된 서브트리를 렌더링하는 단계입니다. 렌더링된 컴포넌트 서브트리는 동일한 색 구성표를 따르는 컴포넌트 트랙에서 확인할 수 있습니다.
- **Commit** - 컴포넌트 렌더링 후, React가 DOM에 변경 사항을 반영하고 `useLayoutEffect`와 같은 레이아웃 Effect를 실행하는 단계입니다.
- **Remaining Effects** - React가 렌더링된 서브트리의 패시브(passive) Effect를 실행하는 단계입니다. 이는 주로 브라우저가 화면을 그린(paint) 후에 발생하며, 이 시점에 `useEffect`와 같은 Hook이 실행됩니다. 클릭과 같은 사용자 상호작용이나 다른 불연속 이벤트(discrete events)는 예외적으로 페인트 이전에 실행될 수 있습니다.

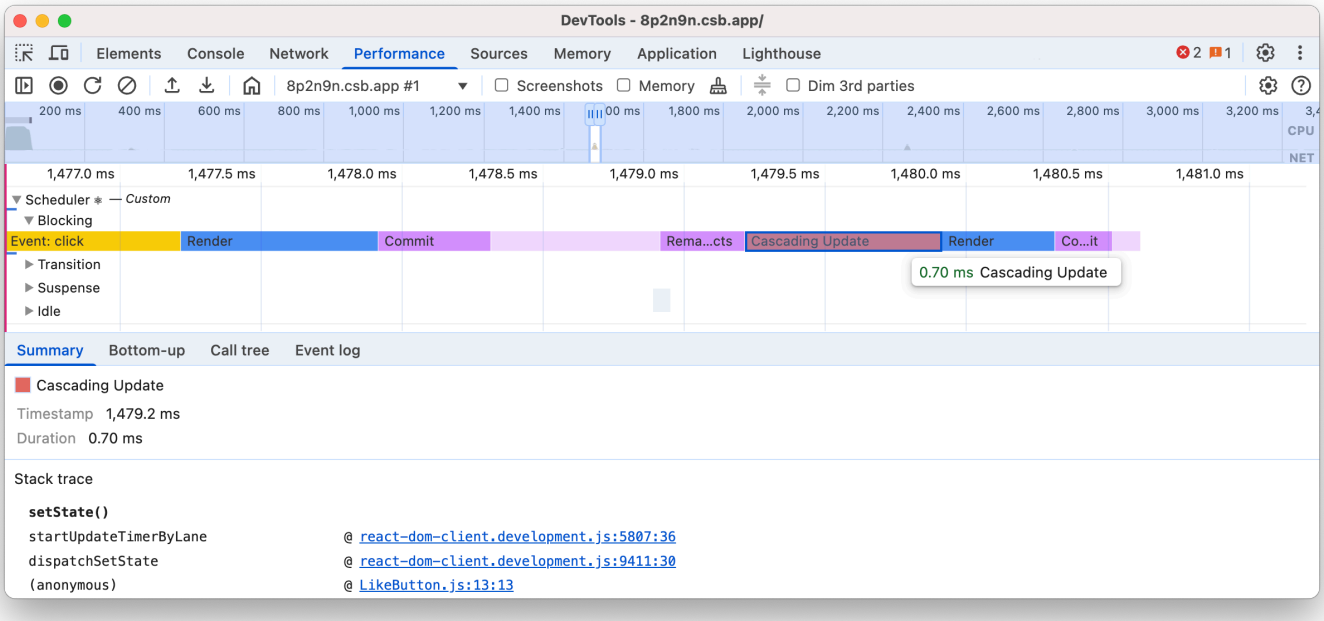


렌더링과 커밋에 대해 더 알아보기.

연쇄 업데이트

연쇄 업데이트(Cascading updates)는 성능 저하를 유발하는 대표적인 패턴 중 하나입니다. 렌더링 과정 중에 업데이트가 예약되면, React는 이미 완료된 작업을 폐기하고 새로운 과정을 다시 시작할 수 있습니다.

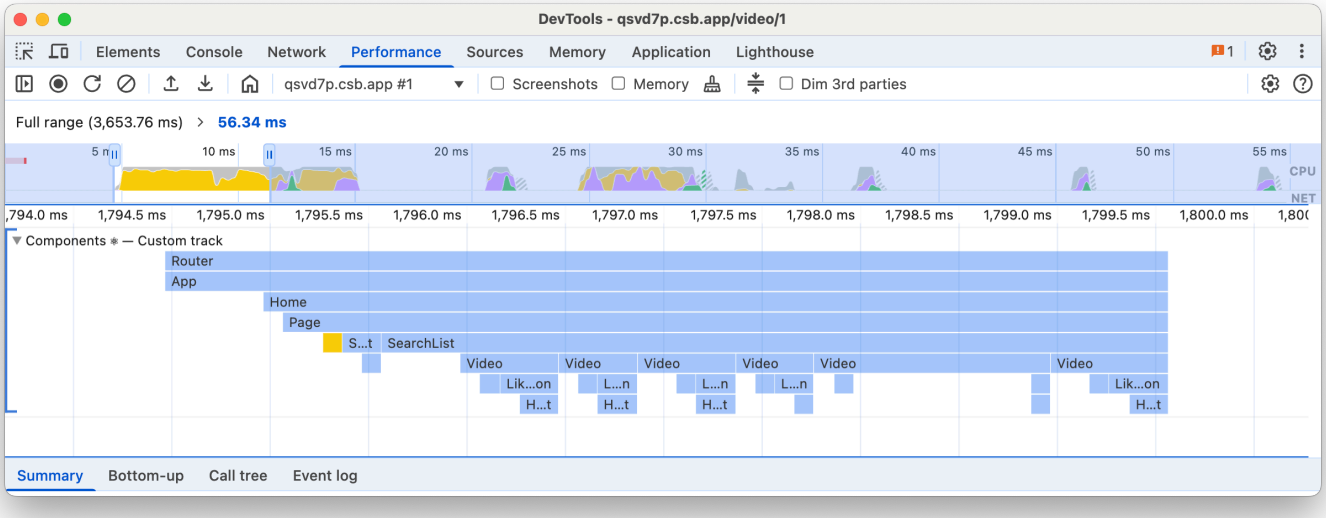
개발 빌드에서는 어떤 컴포넌트가 새로운 업데이트를 예약했는지 확인할 수 있습니다. 여기에는 일반적인 업데이트와 연쇄적으로 발생하는 업데이트가 모두 포함됩니다. “Cascading update” 항목을 클릭하면 업데이트를 예약한 메서드 이름이 포함된 상세한 스택 트레이스를 확인할 수 있습니다.



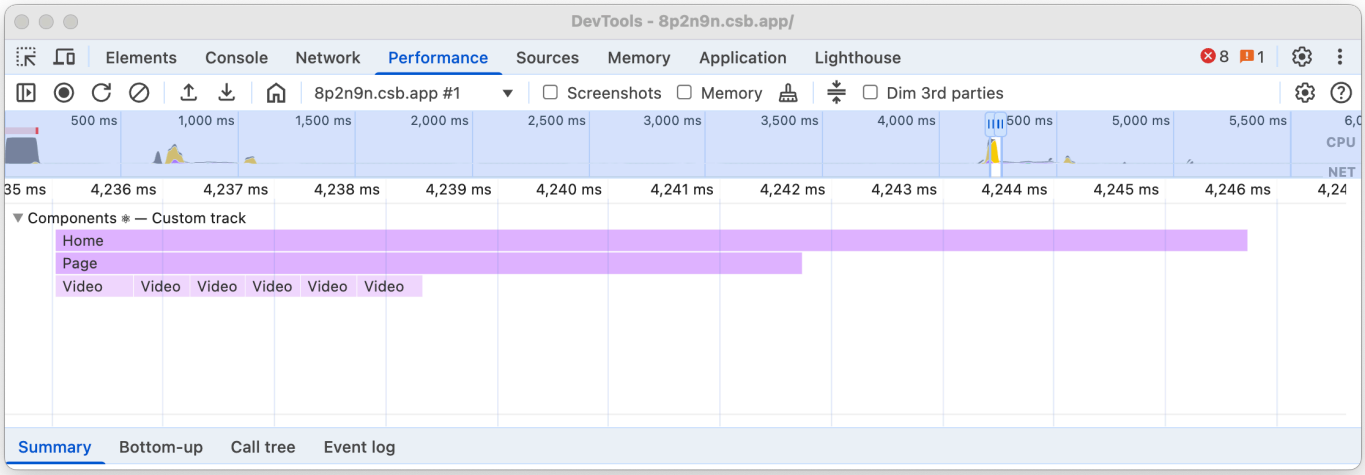
Effect에 대해 더 알아보기.

컴포넌트

컴포넌트 트랙은 React 컴포넌트의 지속 시간을 시각화합니다. 각 항목은 해당 컴포넌트와 그 **모든 하위 컴포넌트**의 렌더링 지속 시간을 나타내는 플레임그래프로 표시됩니다.



렌더링 지속 시간과 유사하게 Effect 지속 시간도 플레임그래프로 표현되지만 Scheduler 트랙의 해당 단계와 일치하는 다른 색 구성표를 사용합니다.



중요합니다!

렌더링과 달리 모든 Effect가 기본적으로 컴포넌트 트랙에 표시되는 것은 아닙니다.

성능을 유지하고 **UI가 복잡해지는 것을 방지하기 위해** React는 0.05ms 이상의 지속 시간을 가지거나 업데이트를 트리거한 Effect만 표시합니다.

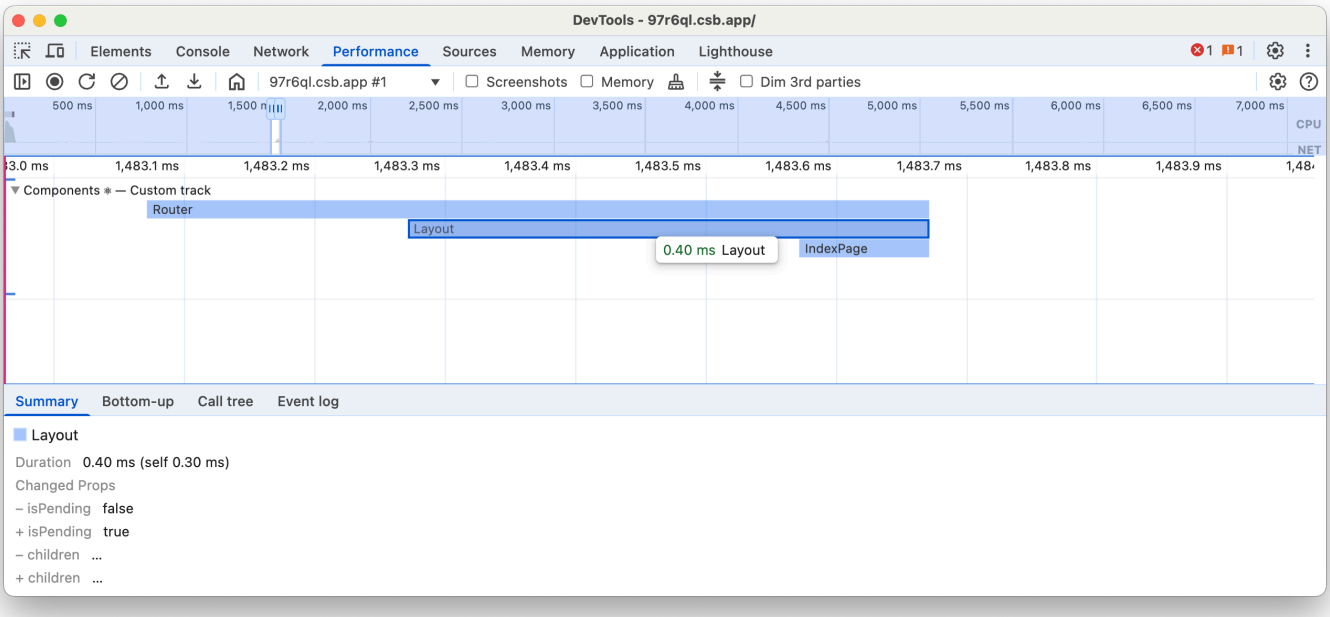
렌더링 및 Effect 단계 중에 추가 이벤트가 표시될 수 있습니다.

- 마운트 - 컴포넌트 렌더링 또는 Effect에 해당하는 서브트리가 마운트되었습니다.

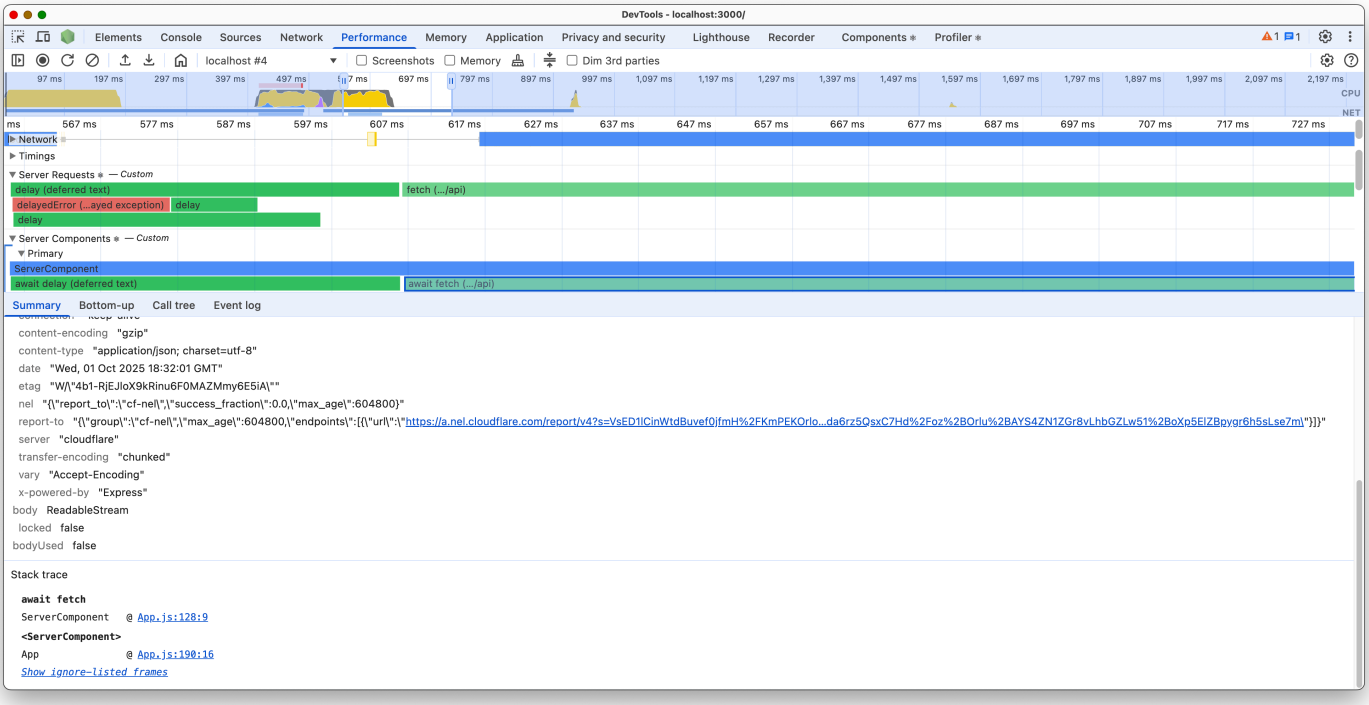
- 마운트 해제 - 컴포넌트 렌더링 또는 Effect에 해당하는 서브트리가 마운트 해제되었습니다.
- Reconnect - 마운트와 유사하지만 `<Activity>` 를 사용하는 경우에만 발생합니다.
- Disconnect - 마운트 해제와 유사하지만 `<Activity>` 를 사용하는 경우에만 발생합니다.

변경된 props

개발 빌드에서 컴포넌트 렌더링 항목을 클릭하면 props의 잠재적인 변경 사항을 확인할 수 있습니다. 이 정보를 사용하여 불필요한 렌더링을 식별할 수 있습니다.



서버



서버 요청

서버 요청 트랙은 최종적으로 React 서버 컴포넌트로 귀결되는 모든 Promise를 시각화합니다. 여기에는 `fetch` 호출이나 비동기 Node.js 파일 작업과 같은 모든 비동기(async) 작업이 포함됩니다.

React는 서드 파티 코드 내부에서 시작된 Promise를 **퍼스트 파티(1st party) 코드**를 차단하는 전체 작업의 지속 시간을 나타내는 단일 스팬으로 결합하려고 시도합니다.
예를 들어, 내부적으로 `fetch` 를 여러 번 호출하는 서드 파티 라이브러리 메서드 `getUser` 는 여러 `fetch` 스팬을 표시하는 대신 `getUser` 라는 단일 스팬으로 표현됩니다.

스팬을 클릭하면 Promise가 생성된 위치의 스택 트레이스와 함께, 가능한 경우 **Promise가 해결(resolve)된 값**의 뷰가 표시됩니다.

거부된(Rejected) Promise는 거부된 값과 함께 빨간색으로 표시됩니다.

서버 컴포넌트

서버 컴포넌트 트랙은 React 서버 컴포넌트의 렌더링 지속 시간과 해당 컴포넌트가 **대기한(awaited) Promise**를 시각화합니다. 시간 정보는 플레임그래프 형태로 표시되며, 각 항목은 해당 컴포넌트와 모든 하위 컴포넌트의 렌더링 지속 시간을 나타냅니다.

Promise를 await하면 React가 해당 Promise의 지속 시간을 **확인할 수 있습니다.** 모든 I/O 작업을 확인하려면 서버 요청 트랙을 사용하세요.

렌더링 지속 시간에 따라 다른 색상으로 표시됩니다. 색이 어두울수록 지속 시간이 더 길다는 것을 의미합니다.

서버 컴포넌트 트랙 그룹에는 항상 “Primary” 트랙이 포함됩니다. React가 서버 컴포넌트를 **동시(concurrently)에** 렌더링할 수 있는 경우 추가적인 “Parallel” 트랙이 표시됩니다.
8개 이상의 서버 컴포넌트가 동시에 렌더링되면 React는 트랙을 계속 추가하는 대신 마지막 “Parallel” 트랙에 연결하여 표시합니다.

커뮤니티

더 보기