<Profiler>

`<Profiler>`를 통해 React 트리의 렌더링 성능을 프로그래밍 방식으로 측정할 수 있습니다.

```
<Profiler id="App" onRender={onRender}>
  <App />
</Profiler>
```

레퍼런스

`<Profiler>`

렌더링 성능을 측정하기 위해서 컴포넌트 트리를 `<Profiler>`로 감싸줍니다.

```
<Profiler id="App" onRender={onRender}>
  <App />
</Profiler>
```

Props

- `id` : 성능을 측정하는 UI 컴포넌트를 식별하기 위한 문자열입니다.
- `onRender` : 프로파일링된 트리 내의 컴포넌트가 업데이트될 때마다 React가 호출하는 `onRender` 콜백입니다. 렌더링된 내용과 소요된 시간에 대한 정보를 받습니다.

Caveats

- Profiling adds some additional overhead, so **it is disabled in the production build by default.** To opt into production profiling, you need to enable a special production build with profiling enabled.
- 프로파일링은 추가적인 오버헤드를 더하기 때문에, **프로덕션 빌드에서는 기본적으로 비활성화 되어있습니다.** 프로덕션 프로파일링을 사용하려면, 프로파일링 기능이 활성화된 특수한 프로덕션 빌드를 사용해야 합니다.

`onRender` 콜백

React는 `onRender` 콜백을 렌더링된 내용과 같이 호출합니다.

```
function onRender(id, phase, actualDuration, baseDuration, startTime, commitTime) {
  // 렌더링 시간 집계 혹은 로그...
}
```

매개변수

- `id` : 커밋된 `<Profiler>` 트리의 문자열 `id` 프로퍼티입니다. 프로파일러를 다중으로 사용하고 있는 트리 내에서 어떤 부분이 커밋 되었는지 식별할 수 있도록 해줍니다.
- `phase` : `"mount"` , `"update"` 혹은 `"nested-update"` . 트리가 최초로 마운트되었는지 또는 Props, State, Hook의 변경으로 인해 리렌더링 되었는지 알 수 있습니다.
- `actualDuration` : 현재 업데이트에 대해 `<Profiler>` 와 자식들을 렌더링하는데 소요된 시간(밀리초)입니다. 이는 하위 트리가 메모이제이션(예: `memo` 와 `useMemo`)을 얼마나 잘 사용하는지를 나타냅니다. 많은 자식들이 특정 Props가 변경되는 경우에만 다시 렌더링되어야 하므로, 이상적으로는 이 값은 최초 마운트 이후에는 많이 감소해야 합니다.
- `baseDuration` : 최적화 없이 전체 `<Profiler>` 하위 트리에 대해 걸리는 시간을 추정하는 소요된 시간(밀리초)입니다. 트리에 있는 각 컴포넌트의 가장 최근 렌더링 시간을 합산하여 계산됩니다. 이 값은 최악의 렌더링 비용(예: 최초 마운트 또는 메모이제이션이 없는 트리)을 추정합니다. `actualDuration` 과 비교하여 메모이제이션이 작동하는지 확인합니다.

- `startTime` : React가 현재 업데이트 렌더링을 시작한 시점에 대한 숫자 타임스탬프입니다.
- `commitTime` : React가 현재 업데이트를 커밋한 시점에 대한 숫자 타임스탬프입니다. 이 값은 커밋된 모든 프로파일러 간에 공유되므로 원하는 경우 그룹화할 수 있습니다.

사용법

프로그래밍 방식으로 렌더링 성능 측정

React 트리를 `<Profiler>` 컴포넌트로 감싸서 렌더링 성능을 측정합니다.

```
<App>
  <Profiler id="Sidebar" onRender={onRender}>
    <Sidebar />
  </Profiler>
  <PageContent />
</App>
```

UI 컴포넌트를 식별하기 위한 `id` 문자열과 트리 내의 컴포넌트가 업데이트를 커밋할 때마다 React가 호출하는 `onRender` 콜백 함수 두 개의 Props가 요구됩니다.

주의하세요!

Profiling adds some additional overhead, so **it is disabled in the production build by default.** To opt into production profiling, you need to enable a special production build with profiling enabled.

중요합니다!

`<Profiler>` 는 프로그래밍 방식으로 측정값들을 모아줍니다. 상호작용할 수 있는 프로파일러를 찾고 있다면, React 개발자 도구의 프로파일러 탭을 사용해 보세요. 브라우저

확장 프로그램으로써 유사한 기능을 제공합니다.

Components wrapped in `<Profiler>` will also be marked in the Component tracks of React Performance tracks even in profiling builds.
In development builds, all components are marked in the Components track regardless of whether they're wrapped in `<Profiler>`.

애플리케이션의 부분별 측정

`<Profiler>` 컴포넌트를 여러개 사용하여 애플리케이션을 부분별로 측정할 수 있습니다.

```
<App>
  <Profiler id="Sidebar" onRender={onRender}>
    <Sidebar />
  </Profiler>
  <Profiler id="Content" onRender={onRender}>
    <Content />
  </Profiler>
</App>
```

`<Profiler>` 컴포넌트들을 중첩해서 사용할 수 있습니다.

```
<App>
  <Profiler id="Sidebar" onRender={onRender}>
    <Sidebar />
  </Profiler>
  <Profiler id="Content" onRender={onRender}>
    <Content>
      <Profiler id="Editor" onRender={onRender}>
        <Editor />
      </Profiler>
      <Preview />
    </Content>
  </Profiler>
```

`<Profiler>` 는 가벼운 컴포넌트이지만 사용할 때마다 애플리케이션에 약간의 CPU 및 메모리 오버헤드를 추가하기 때문에 필요할 때만 사용해야 합니다.

이전
<Fragment> (<>)

다음
<StrictMode>

uwu?