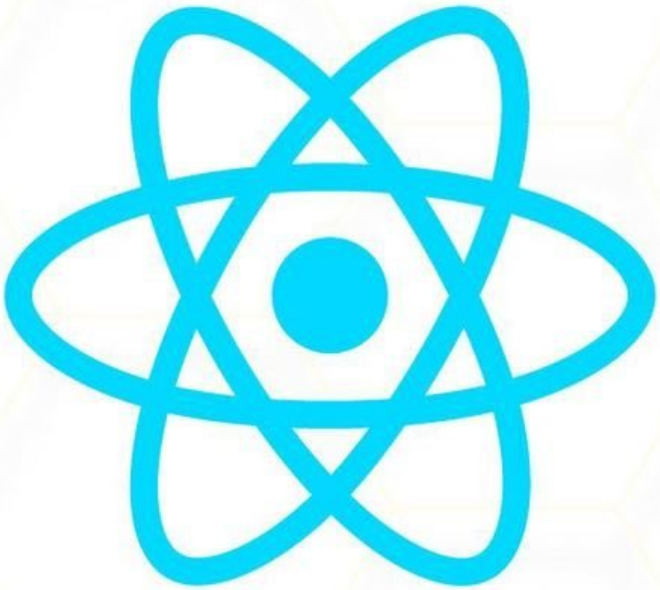
FOLLOW TO LEARN MORE

Profiler in React



SWIPE

@nishasingla05

@codewithHarshad

The Profiler API is the recommended way of measuring the rendering times of our components

The **Profiler** measures how often a React application renders and what the **"cost" of rendering** is.

Its purpose is to help identify parts of an application that are slow so that we can work on optimizing them

Let's understand it


@nishasingla05


@codewithHarshad

To use Profiler, Wrap a component tree in a **<Profiler>** to measure its rendering performance

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

Props

id: A string identifying the part of the UI you are measuring.

onRender: An onRender callback that React calls every time components within the profiled tree update. It receives information about what was rendered and how much time it took.

@nishasingla05

@codewithHarshad

onRender callback

React will call your onRender callback with information about what was rendered.

```
function onRender(id,
                  phase,
                  actualDuration,
                  baseDuration,
                  startTime,
                  commitTime
                 ) {
  // Aggregate or log render timings...
}
```

Profiler Component:

- If we want to have programmatic access to the performance measurements of a specific component, we can use the Profiler component.

- It wraps part or all of our app tree, and gives us metrics on how long it took for that tree to render.

```
import {Profiler} from 'react';
import User from './User';
function App() {


  const logTimes = () => {};

  return (
    <div className="App">
      <Profiler id="App" onRender={logTimes}>
        <User />
      </Profiler>

    </div>
  );
}

export default App;
```

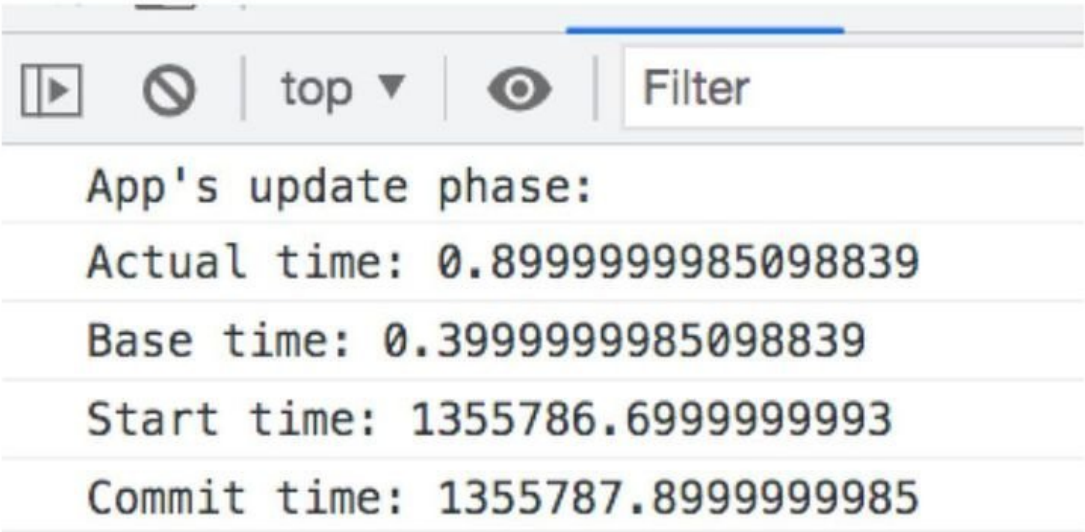
@nishasingla05

@codewithHarshad

onRender callback

```
const logTimes = (id, phase, actualTime, baseTime, startTime, commitTime) =>
  console.log(`${id}'s ${phase} phase:`);
  console.log(`Actual time: ${actualTime}`);
  console.log(`Base time: ${baseTime}`);
  console.log(`Start time: ${startTime}`);
  console.log(`Commit time: ${commitTime}`);
};
```

- When User renders, the Profiler's onRender callback will be invoked with a bunch of useful information.
-
- In our example, it'll print something like this to the console:



More about Profiler

- Profiling adds some additional overhead, so it is disabled in the production build.

- To opt into production profiling, React provides a special production build with profiling enabled.

check here

fb.me/react-profiling

- <Profiler> lets you gather measurements programmatically. If you're looking for an interactive profiler, try the Profiler tab in React Developer Tools. It exposes similar functionality as a browser extension

Want more?

Follow
&
Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like **Comment** **Repost**