Improve UI Experience with useDeferredValue and useTransition

Peter Kellner

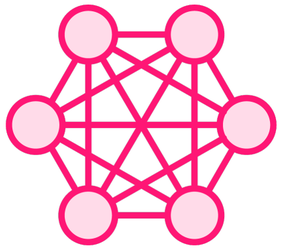
Developer, Consultant and Author

ReactAtScale.com @pkellner linkedin.com/in/peterkellner99

useDeferredValue and useTransition Goals

Designed to make React apps more responsive

Change priority of state updates



Allow developers more control

useDeferredValue and useTransition Differences

useDeferredValue and useTransition Differences

useDeferredValue

For use when your state is managed outside of your apps direct control

useDeferredValue and useTransition Differences

useDeferredValue

For use when your state is managed outside of your apps direct control

useTransition

For use when you have direct access to React component state

Improve Performance

useDeferredValue and useTransition gives our browser users a better experience

The Basics of useDeferredValue

/pages/demoUseDeferredValue.js

```
export default function App() {

  const [search, setSearch] = useState("");

  return (
    <>
      <input value={search} onChange={ e => setSearch(e.target.value) } />
      <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseDeferredValue.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const deferredSearch = useDeferredValue(search);

  return (
    <>
      <input value={search} onChange={ e => setSearch(e.target.value) } />
      <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseDeferredValue.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const deferredSearch = useDeferredValue(search);

  return (
    <>
      <input value={search} onChange={ e => setSearch(e.target.value) } />
      <SlowResults query={deferredSearch} />
    </>
  );
}
```

The Basics of useTransition

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");

  return (
    <>
      <input value={search}
             onChange={e => {
               setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();

  return (
    <>
      <input value={search}
             onChange={e => {
               setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();

  return (
    <>
      <input value={search}
             onChange={e => {
               startTransition(() => setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();

  return (
    <>
      <input value={search}
             onChange={e => {
               setSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();
  const [currentSearch, setCurrentSearch] = useState("");

  return (
    <>
      <input value={search}
             onChange={e => {
               setSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();
  const [currentSearch, setCurrentSearch] = useState("");

  return (
    <>
      <input value={currentSearch}
             onChange={e => {
               setSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();
  const [currentSearch, setCurrentSearch] = useState("");

  return (
    <>
      <input value={currentSearch}
             onChange={e => {
               setCurrentSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();
  const [currentSearch, setCurrentSearch] = useState("");

  return (
    <>
      <input value={currentSearch}
             onChange={e => {
               setCurrentSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        {isPending ? "refreshing..." : null}
        <SlowResults query={search} />
    </>
  );
}
```

/pages/demoUseTransition.js

```
export default function App() {

  const [search, setSearch] = useState("");
  const [isPending, startTransition] = useTransition();
  const [currentSearch, setCurrentSearch] = useState("");

  return (
    <>
      <input value={currentSearch}
             onChange={e => {
               setCurrentSearch(e.target.value);
               startTransition(() => setSearch(e.target.value));
             }}
      />
        {isPending ? "refreshing..." : null}
        <SlowResults query={search} />
    </>
  );
}
```

Takeaways

The difference between “nice to have” and “have to have”

useDeferredValue is easy

useTransition if you need more control