Canal 14 (Frequência 75.4)

WORKSHOP 14

Como manter a sua app React acima dos 60fps

Rafael Lages e Henrique Elias


hotmart

React Performance

Henrique Elias

Sênior Front-end

Rafael Lages

Sênior Front-end

ReactJS

Concurrent Mode

Concurrent Mode

O que é?

- Conjunto de novas funcionalidades que impacta na renderização dos componentes
- Estágio experimental - React Conf 2019
- Foco na experiência do usuário

Instalação

- npm install react@experimental react-dom@experimental

- Concurrent Mode

```
ReactDOM
    .createRoot(document.getElementById('root'))
    .render(<App/>);
```

- Blocking Mode

```
ReactDOM
    .createBlockingRoot(document.getElementById('root'))
    .render(<App/>);
```

- React.StrictMode

Modos

	Legacy Mode	Blocking Mode	Concurrent Mode
String Refs	✅	🚫**	🚫**
Legacy Context	✅	🚫**	🚫**
findDOMNode	✅	🚫**	🚫**
Suspense	✅	✅	✅
SuspenseList	🚫	✅	✅
Suspense SSR + Hydration	🚫	✅	✅
Progressive Hydration	🚫	✅	✅
Selective Hydration	🚫	🚫	✅
Cooperative Multitasking	🚫	🚫	✅
Automatic batching of multiple setStates	🚫*	✅	✅
Priority-based Rendering	🚫	🚫	✅
Interruptible Prerendering	🚫	🚫	✅
useTransition	🚫	🚫	✅
useDeferredValue	🚫	🚫	✅
Suspense Reveal "Train"	🚫	🚫	✅

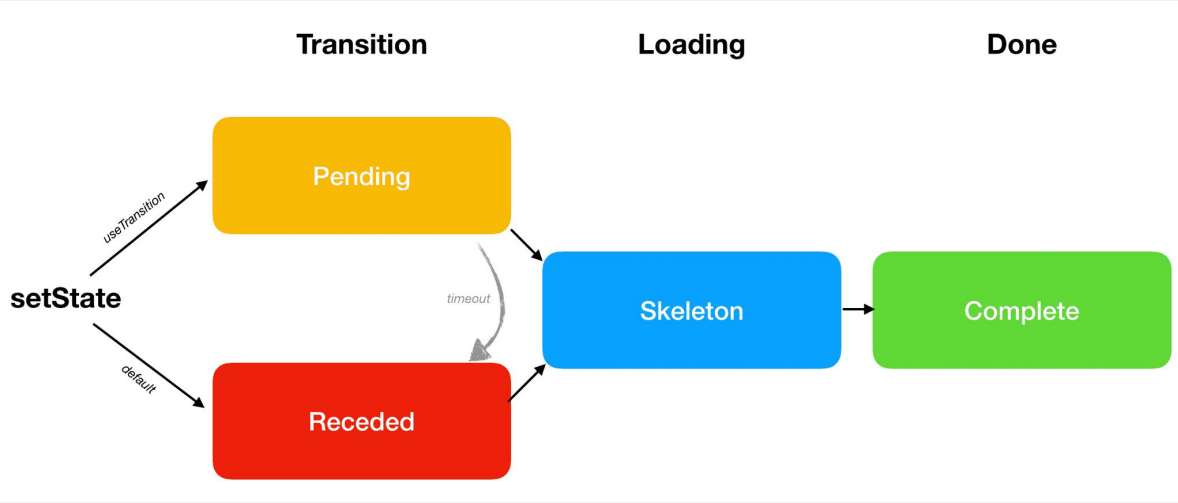
Concurrent UI Patterns

- Transitions - useTransition Hook
- Deferring a value - useDeferredValue Hook
- State isolation
- SuspenseList

Concurrent Mode

Transitions - useTransition Hook

- Gerencia a renderização de um componente a partir de um intervalo definido
- Normalmente, utilizado para renderizar ou não componentes de loading
- O parâmetro passado define o tempo máximo de espera para a transição finalizar.




```
function App() {
  const [resource, setResource] = useState(initialResource);
  const [startTransition, isPending] = useTransition({
    timeoutMs: 3000
  });
  return (
    <>
      <button
        disabled={isPending}
        onClick={() => {
          startTransition(() => {
            const nextUserId = getNextId(resource.userId);
            setResource(fetchProfileData(nextUserId));
          });
        }}
      >
        Next
      </button>
      {isPending ? " Loading..." : null}
      <ProfilePage resource={resource} />
    </>
  );
}
```

Concurrent Mode

Deferring a Value - useDeferredValue Hook

- Retorna o valor com um “atraso”
- Manter a responsividade da tela quando há componentes que precisam ser atualizados imediatamente e outros não
- Exemplo: Input text + componente pesado
- Configuração define o tempo de “atraso”

```
function App() {
  const [text, setText] = useState("hello");
  const deferredText = useDeferredValue(text, { timeoutMs: 2000 });

  return (
    <div className="App">
      {/* Keep passing the current text to the input */}
      <input value={text} onChange={handleChange} />
      ...
      {/* But the list is allowed to "lag behind" when necessary */}
      <MySlowList text={deferredText} />
    </div>
  );
}
```

Concurrent Mode

State Isolation

- Há casos onde a performance pode ser impactada ao renderizar o state totalmente
- Separar atualizações do state que são prioritárias
- Exemplo: 2 requests definindo o mesmo state - tempo de carregamento é o tempo do request mais longo

```
function handleChange(e) {
  const value = e.target.value;

  // Outside the transition (urgent)
  setQuery(value);

  startTransition(() => {
    // Inside the transition (may be delayed)
    setResource(fetchTranslation(value));
  });
}
```

Concurrent Mode

SuspenseList

- Suspense(React 16.6) permite esperar um evento acontecer para renderizar o componente, enquanto exibe outro componente
- É comum utilizar para renderizar o loading enquanto acontece o fetch dos dados
- SuspenseList resolve casos onde há múltiplos Suspenses, principalmente ao realizar fetch de requests em paralelo
- revealOrder (forwards, backwards, together) - Define a ordem de exibição dos fallbacks
- tail (collapsed, hidden) - Define comoos items não carregados são exibidos(experimental)

```
function ProfilePage({ resource }) {
  return (
    <SuspenseList revealOrder="forwards">
      <ProfileDetails resource={resource} />
      <Suspense fallback={<h2>Loading posts...</h2>}>
        <ProfileTimeline resource={resource} />
      </Suspense>
      <Suspense fallback={<h2>Loading fun facts...</h2>}>
        <ProfileTrivia resource={resource} />
      </Suspense>
    </SuspenseList>
  );
}
```

Workshop

Concurrent Mode

Workshop

- O objetivo deste workshop é melhorar a performance de uma aplicação React utilizando as features do Concurrent Mode
- A aplicação exibe uma lista de produtos e permite fazer busca por nome do produto
- Há uma prop ‘slow’ que se ativada irá colocar um delay na renderização dos componentes - simulação de componentes pesados

Concurrent Mode

Workshop

https://github.com/Hotmart-Org/workshop-react-performance

- Input List
  - Utilizar early fetch - “*fetchAllProductsWithSuspense*”
  - Utilizar deferred value para passar o texto de busca como props para a lista
  - Utilizar Suspense em torno do componente “*List*”

- List
  - Utilizar “memo”
  - Utilizar método read da interface provida pelo “*fetchAllProductsWithSuspense*”

- ListItem
  - Remover último parâmetro “true” da função “*callPromiseWithTimeout*” - irá transformar uma promise na interface que utilizamos no Suspense
  - Adicionar Suspense em torno do componente “*CommentsLength*”

- CommentsLength
  - Utilizar método read da interface provida pelo “*fetchAllProductsWithSuspense*”

Concurrent Mode

Workshop

- Iremos realizar um sorteio de dois kits Hotmart
- Realizar cadastro em: https://tinyurl.com/react-perf-hotmart
- Iremos utilizar o Google para sortear um número aleatoriamente

Troopers Onboarding

That's all folks!