Performance no React

Diego Schell Fernandes - CTO @Rocketseat

O que veremos?

Imutabilidade para performance
Utilização do React DevToools
Aplicação dos hooks para performar componentes
Memorização de componentes com React.memo
Virtualização de grandes listas de dados
Code Splitting no React para Web
Espiando as novas APIs do React

Imutabilidade

```
const users = ['John', 'Robert'];

users.push('Ralph');

updateState(users);

// newUsers === users
```

vs.

```
const users = ['John', 'Robert'];

updateState([...users, 'Ralph']);

// newUsers !== users
```

Ferramentas de desenvolvedor

React DevTools: Identificar gargalos na renderização de componentes;

Let's use it!

useMemo

Memorizar um valor calculado dentro do componente.

Exemplo do useMemo

```jsx
function UserGroups({ user }) {
  return (
    <>
      <strong>Total de {users.groups.length} grupos.</strong>
      <p>{user.groups.map(group => group.name).join(', ')}</p>
    </>
  );
}
```

```jsx
function UserGroups({ user }) {
  const totalGroups = useMemo(() => users.groups.length, [users]);

  const groupNames = useMemo(() =>
    user.groups.map(group => group.name).join(', '),
    [users]
  );

  return (
    <>
      <strong>Total de {totalGroups} grupos.</strong>
      <p>{groupNames}</p>
    </>
  );
}
```

Dados externos sem useMemo

```
function AccessLog() {
  useEffect(...); // load logs

  return (
    <ul>
      {logs.map(log => (
        <li key={log.id}>
          {log.user.name} acessou em {format(parseISO(log.date), 'dd/MM/YYYY HH:mm')}
        </li>
      ))}
    </ul>
  );
}
```

Dados externos com useMemo

```
function AccessLog() {
  useEffect(...); // load logs

  const logList = useMemo(() => {
    return logs.map(log => ({
      id: log.id,
      user: log.user.name,
      formattedDate: format(parseISO(log.date), 'dd/MM/YYYY HH:mm'),
    }))
  }, [logs]);

  return (
    <ul>
      {logList.map(log => (
        <li key={log.id}>{log.user} acessou em {log.formattedDate}</li>
      ))}
    </ul>
  );
}
```

useCallback

Memorizar uma função criada dentro do componente.

Exemplo do useCallback

```
function UserList() {
  const [users, setUsers] = useState([]);

  function deleteUser(id) {
    const newUsers = users.filter(user => user.id !== id);

    setUsers(newUsers);
  }

  return (...);
}
```

```
function UserList() {
  const [users, setUsers] = useState([]);

  const deleteUser = useCallback((id) => {
    const newUsers = users.filter(user => user.id !== id);

    setUsers(newUsers);
  }, [users]);

  return (...);
}
```

Reconciliation

A cada atualização de propriedade e estado, o React reconstrói a árvore de um elemento e a compara com a anterior.

Caso tenha mudado, atualiza na DOM.

E os elementos filhos?

Todos elementos filhos também são renderizados novamente 😵

```jsx
function Header({ user }) {
  return (
    <User>
      <strong>{user.name}</strong>
      <Avatar src={user.thumbnail} />
    </User>
  );
}
```

Utilizando React.memo

```
// Avatar.js

function Avatar({ src }) {
  return (...)
}

export default React.memo(Avatar);
```

```
// Header.js

function Header({ user }) {
  return (
    <User>
      <strong>{user.name}</strong>
      <Avatar src={user.thumbnail} />
    </User>
  );
}
```

Quando usar?

Devemos usar o `React.memo` principalmente em situações que componentes sofrem múltiplos ciclos de renderização durante o tempo de vida da aplicação.

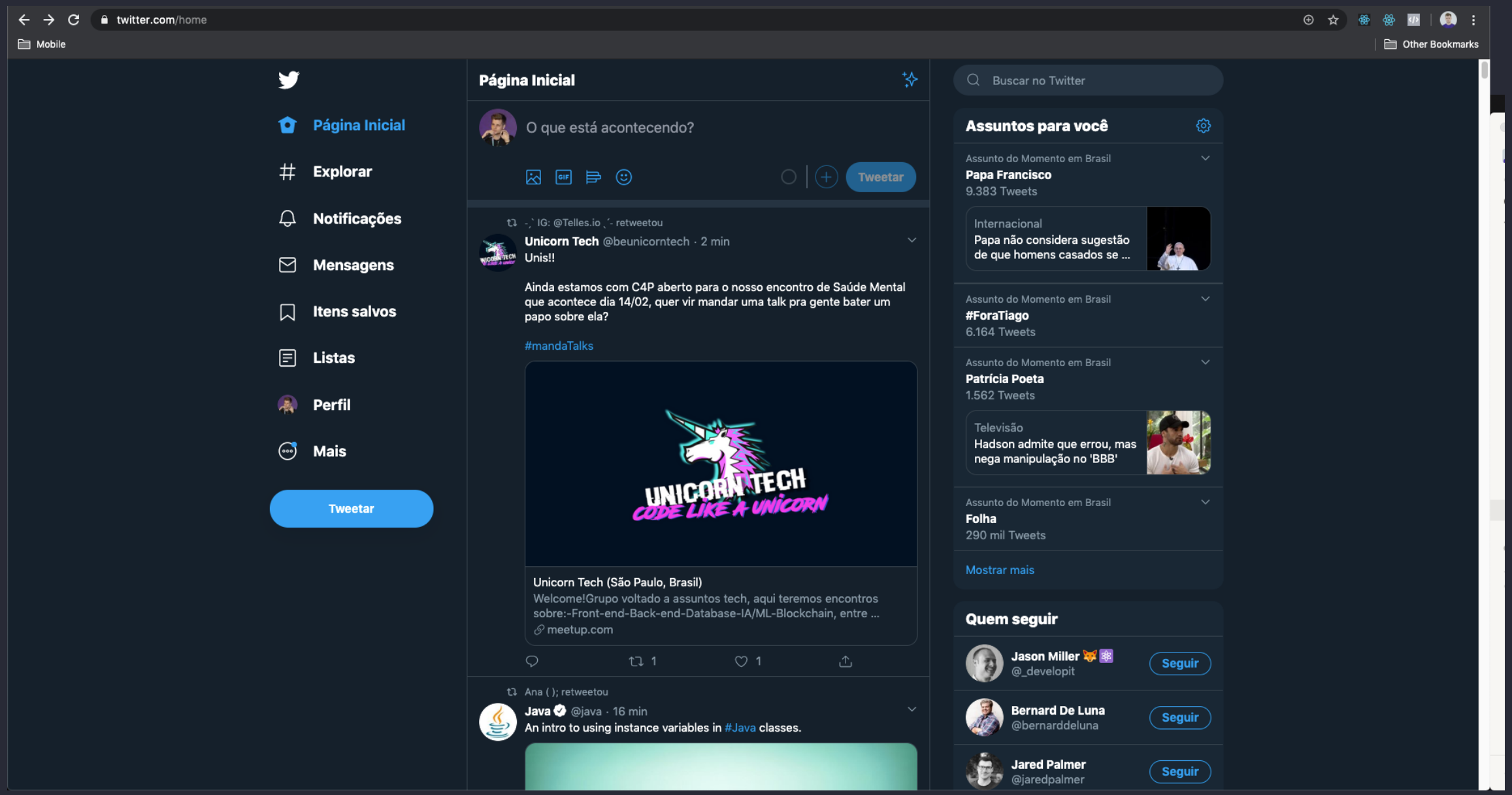
Se aplica muito bem para componentes totalmente visuais que retornam os mesmos dados baseado nas propriedades e estados que recebe.

Virtualização

Como renderizar grandes listas e tabelas de dados?

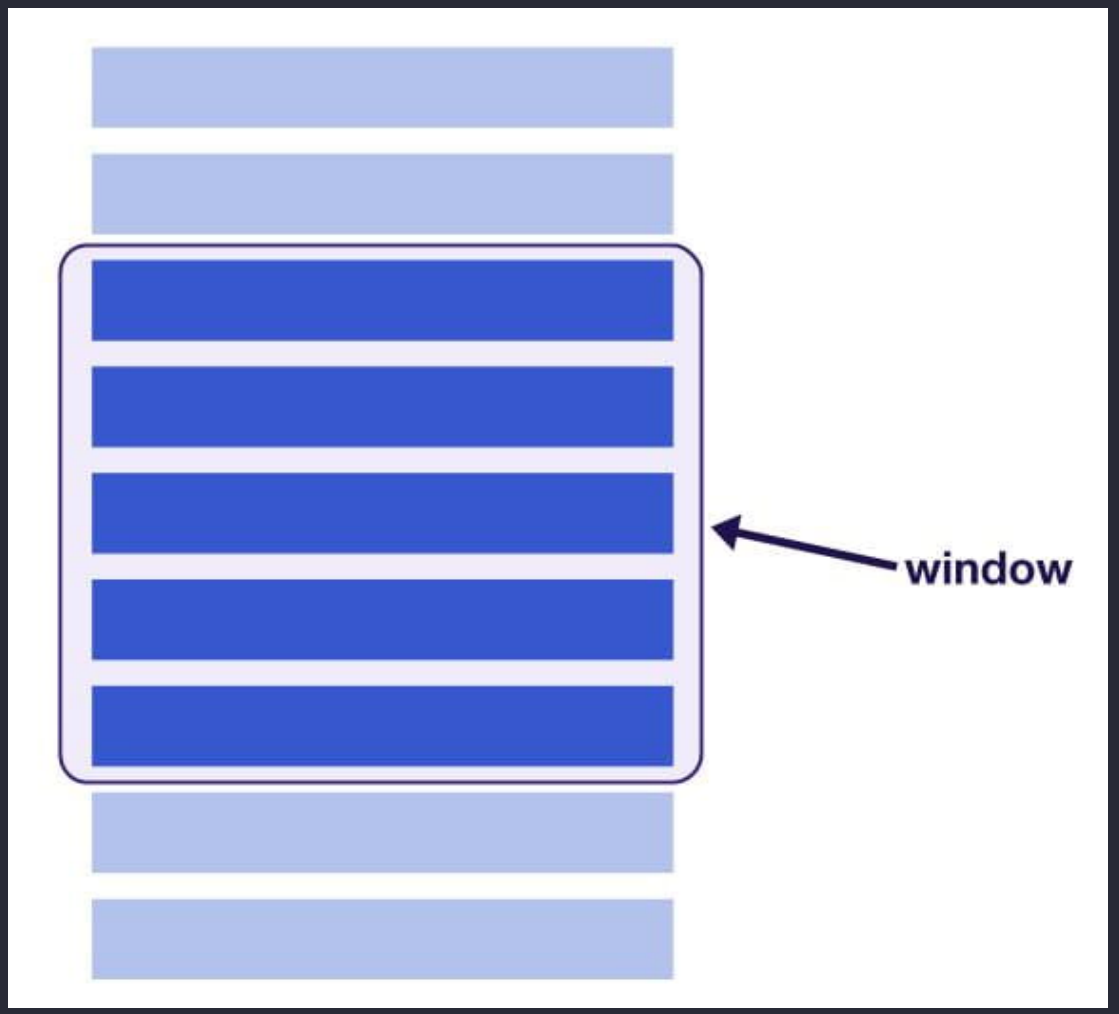
A virtualização permite renderizar apenas os componentes que aparecem em tela.

Exemplo


twitter.com/home
Mobile
Other Bookmarks
Página Inicial
Explorar
Notificações
Mensagens
Itens salvos
Listas
Perfil
Mais
Tweetar
Página Inicial
O que está acontecendo?
Tweetar
-,` IG: @Telles.io ¸´- retweetou
Unicorn Tech @beunicorntech · 2 min
Unis!!
Ainda estamos com C4P aberto para o nosso encontro de Saúde Mental que acontece dia 14/02, quer vir mandar uma talk pra gente bater um papo sobre ela?
#mandaTalks
UNICORN TECH
CODE LIKE A UNICORN
Unicorn Tech (São Paulo, Brasil)
Welcome!Grupo voltado a assuntos tech, aqui teremos encontros sobre:-Front-end-Back-end-Database-IA/ML-Blockchain, entre ...
meetup.com
1
1
Ana ( ); retweetou
Java @java · 16 min
An intro to using instance variables in #Java classes.
Buscar no Twitter
Assuntos para você
Assunto do Momento em Brasil
Papa Francisco
9.383 Tweets
Internacional
Papa não considera sugestão de que homens casados se ...
Assunto do Momento em Brasil
#ForaTiago
6.164 Tweets
Assunto do Momento em Brasil
Patrícia Poeta
1.562 Tweets
Televisão
Hadson admite que errou, mas nega manipulação no 'BBB'
Assunto do Momento em Brasil
Folha
290 mil Tweets
Mostrar mais
Quem seguir
Jason Miller
@_developit
Seguir
Bernard De Luna
@bernarddeluna
Seguir
Jared Palmer
@jaredpalmer
Seguir

Como implementar?

[React Window]



Code example

```
import { FixedSizeList as List } from 'react-window';

const Row = ({ index, style }) => (
  <div style={style}>Row {index}</div>
);

const Example = () => (
  <List
    height={150}
    itemCount={1000}
    itemSize={35}
    width={300}
  >
    {Row}
  </List>
);
```

Code splitting

Diminuir o bundle dividindo-o em pedaços menores;

Importar funções ou componentes somente se precisarmos;

Import dinâmico

```
export default function App() {
  function doSomeStuff(data) {
    import('my-super-heavy-function').then(superHeavy => {
      superHeavy(data);
    })
  }

  return (
    <button onClick={doSomeStuff}>Click here</button>
  );
}
```

E para componentes?

```jsx
import Comments from './Comments';

export default function Post({ post }) {
  return (
    <div>
      <strong>{post.title}</strong>
      <p>{post.description}</p>

      <Comments comments={post.comments} />
    </div>
  );
}
```

React.lazy

```
import { Suspense, lazy } from 'react';

const Comments = lazy(() => import('./Comments'));

export default function Post({ post }) {
  return (
    <div>
      <strong>{post.title}</strong>
      <p>{post.description}</p>

      <Suspense fallback={<p>Loading</p>}>
        { post.comments.length && <Comments comments={post.comments} /> }
      </Suspense>
    </div>
  );
}
```

Concurrent mode (Nova API)

⚒️ Totalmente experimental ⚒️

Utilizar o Suspense para aguardar qualquer tipo de código assíncrono finalizar.

Isso inclui chamadas à API ou efeitos colaterais de modo geral.

Isso nos permite utilizar um novo conceito chamado Render-as-You-Fetch.

Aguardar promises de componentes internos finalizarem antes de alterarmos algum estado.

Let's code