

# Performance no React



# Olá!

## Meu nome é John Victor

Pesquisador

Laboratório de Inovação Tecnológica em Saúde  
(LAIS/HUOL/UFRN)

Github: @JohnVict0r

Instagram: @Johnv\_alves



# Agenda

- ▶ Performance
- ▶ Reconciliation
- ▶ Utilizar Hooks para performance
- ▶ Memorizando componentes
- ▶ Virtualização de grandes listas de dados
- ▶ React Dev Tools na prática



1

# Performance



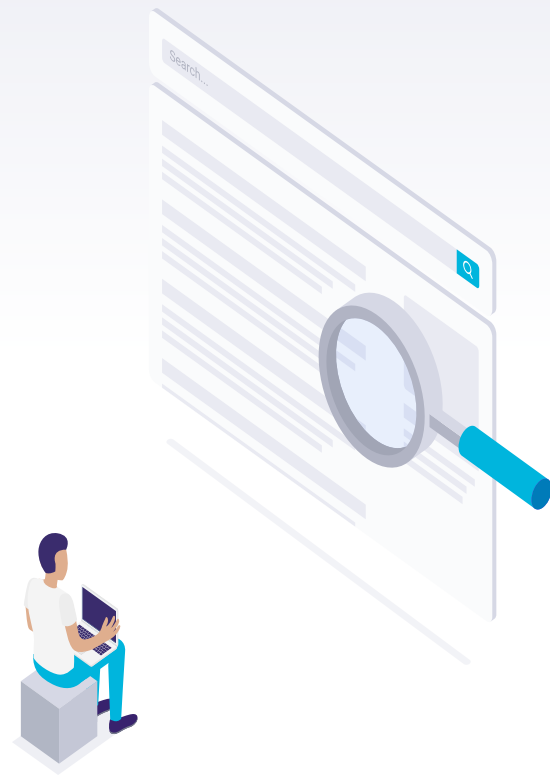


A **performace** da aplicação está na usabilidade e na experiência de usuário (UX) que a aplicação consegue entregar.

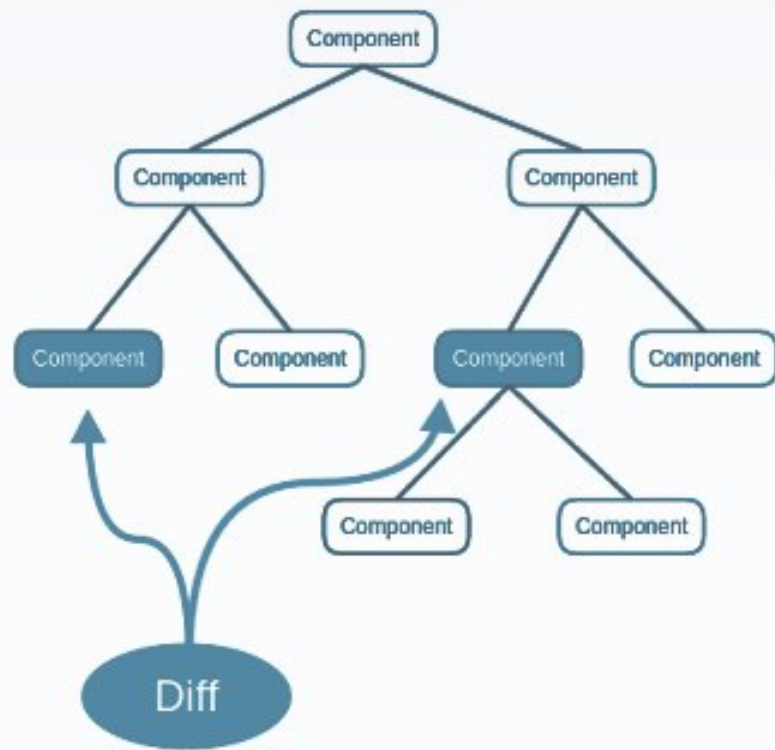


2

# Reconciliation



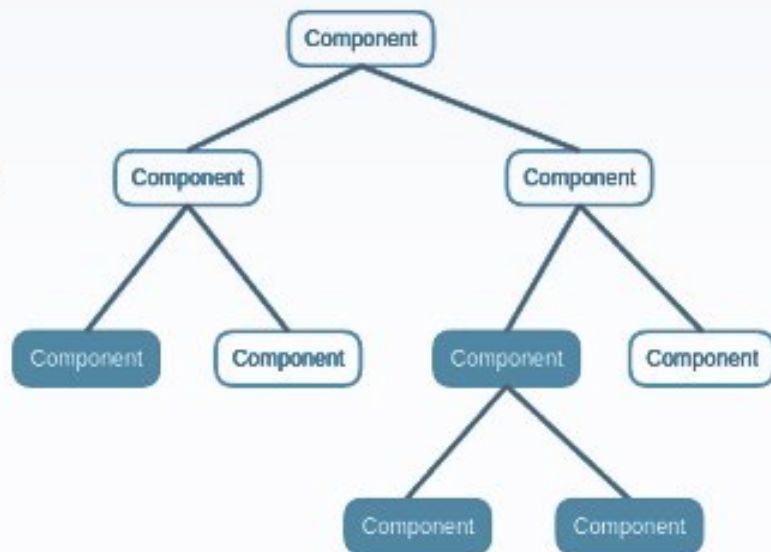
# Virtual DOM



Re-rendered



# DOM





Time que tá ganhando não  
se mexe...  
ou quase isso...

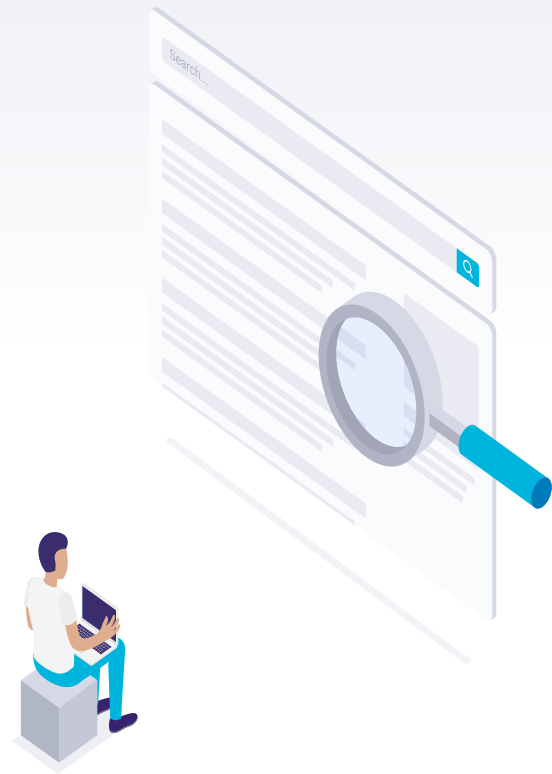




3

# Hooks para performance

useMemo e useCallback






# useMemo

Memorizar um valor calculado dentro do componente

# Exemplo



```
function DevInfo({ dev }){  
  return (  
    <div>  
      <h1>Nome: {dev.name}</h1>  
      <strong>Total de repositorios: {dev.repositories.length}</strong>  
      <p>Tecnologias: {dev.techs.map(tech => tech.name).join(', ')}</p>  
    </div>  
  )  
}
```



```
import { useMemo } from 'react'
```

```
function DevInfo({ dev }){
```

```
  const devName = useMemo(() => dev.name);
```

```
  const totalRepositories = useMemo(() => dev.repositories.length);
```

```
  const techNames = useMemo(() => dev.techs.map(tech => tech.name).join(', '));
```

```
  return (
```

```
    <div>
```

```
      <h1>Nome: {devName}</h1>
```

```
      <strong>Total de repositorios: {totalRepositories}</strong>
```

```
      <p>Tecnologias: {techNames}</p>
```

```
    </div>
```

```
  )
```

```
}
```



# useCallback

Memorizar uma função criada dentro do componente

# Exemplo

```
function TodoList({ dev }){  
  const [todos, setTodos] = useState([]);  
  
  function deleteTodo(id){  
    const newTodos = todos.filter(todo => todo.id !== id)  
  
    setTodos(newTodos)  
  };  
  
  return ( ... );  
}
```



```
function TodoList({ dev }){  
  const [todos, setTodos] = useState([]);  
  
  const deleteTodo = useCallback((id){  
    const newTodos = todos.filter(todo => todo.id !== id)  
  
    setTodos(newTodos)  
  }, [todos]);  
  
  return ( ... );  
}
```

# useCallback x useMemo

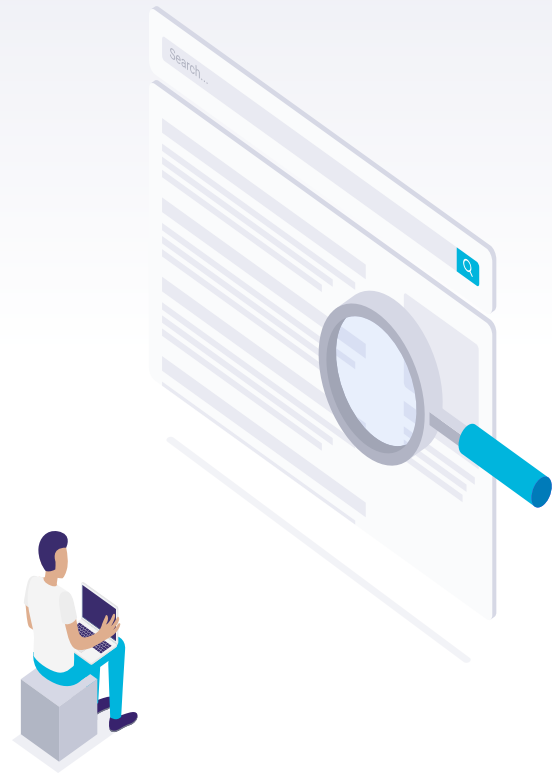
```
function myName() {  
  return 'John Victor';  
}  
  
const memorizedCallback = useCallback(myName, []);  
const memorizedResult = useMemo(myName, []);  
  
memorizedCallback;  
// f myName() {  
//   return 'John Victor';  
// }  
memorizedResult; // 'John Victor'  
memorizedCallback(); // 'John Victor'  
memorizedResult(); // ● TypeError
```



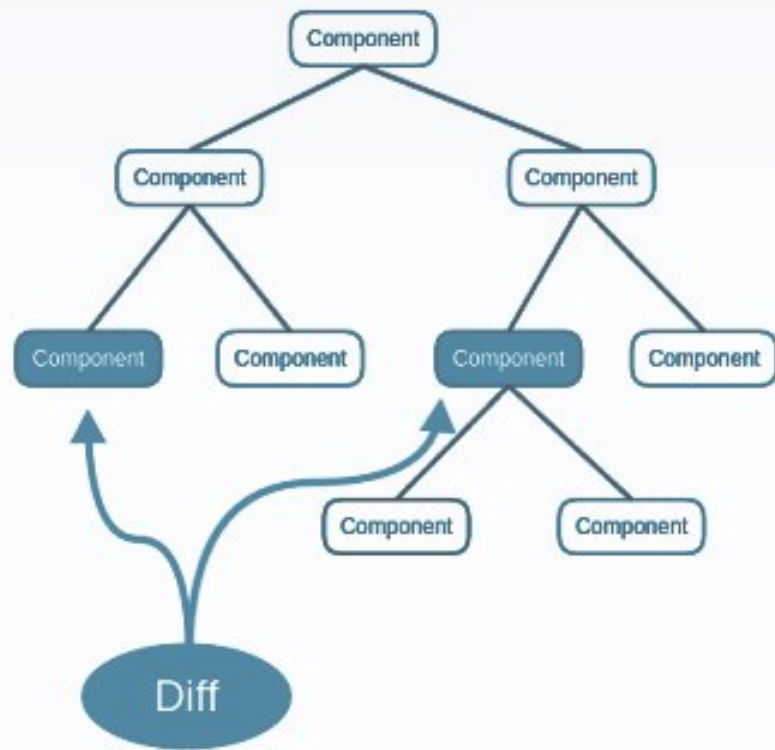
4

# Memorização

É possível memorizar componentes?



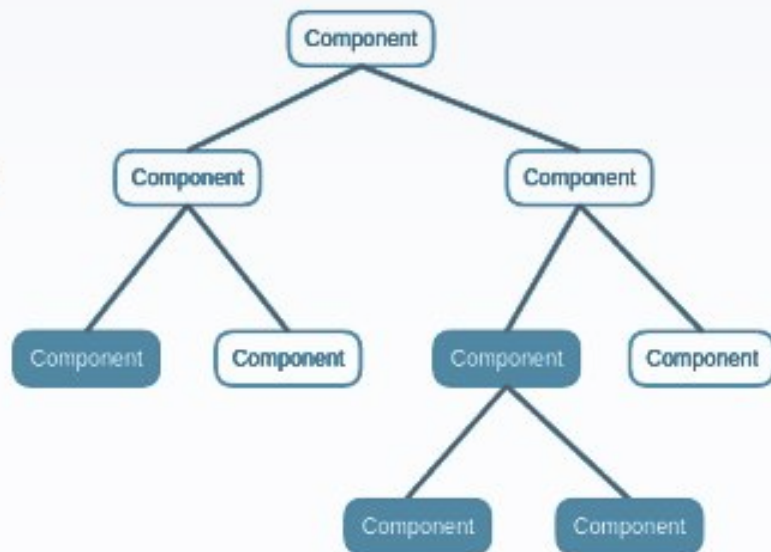
## Virtual DOM



Re-rendered



## DOM



# Exemplo

```
function Card({ name, description }) {  
  return (  
    <div>  
      <p>{name}</p>  
      <p>{description}</p>  
    </div>  
  )  
}
```

```
class Card extends React.PureComponent {  
  render() {  
    const { name, description } = this.props;  
    return (  
      <div>  
        <p>{name}</p>  
        <p>{description}</p>  
      </div>  
    )  
  }  
}
```

# React.memo

Resolvendo o problema de re-renderização, utilizando apenas uma função.

```
function Card({ name, description }) {  
  return (  
    <div>  
      <p>{name}</p>  
      <p>{description}</p>  
    </div>  
  )  
}  
  
export default React.memo(Card);
```

5

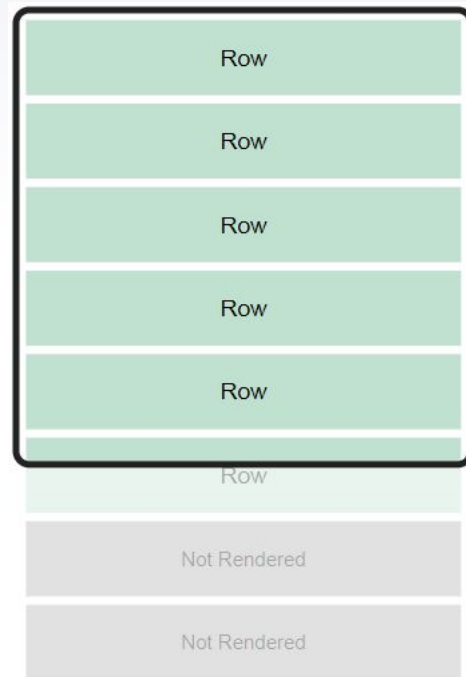
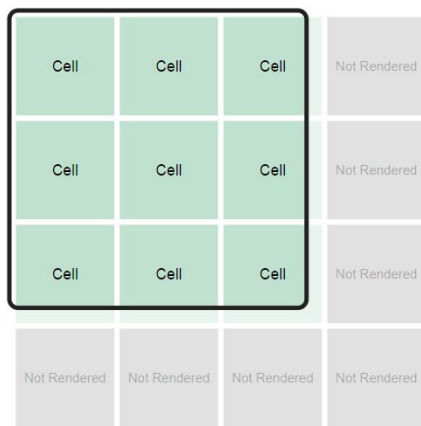
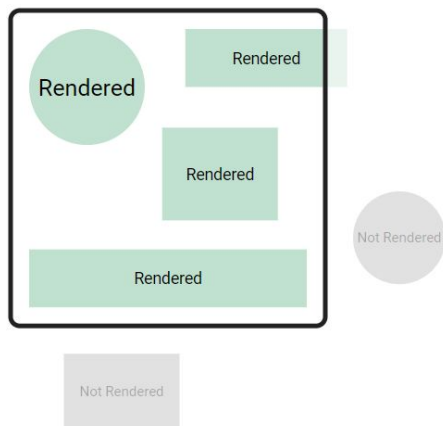
# Virtualização

Como renderizar grandes listas de dados?



# Qual estratégia utilizar?

- ▶ Tables, lists, spreadsheets (react-virtualized)
- ▶ Drop-down menus (react-virtualized-select)
- ▶ Calendar & date-pickers (react-infinite-calendar)
- ▶ Tree views (react-sortable-tree)
- ▶ Image carousels, news feeds, chat applications, etc



# Let's code

Hora de praticar...



# Desempenho

Hora da reflexão...





## Expectativa

- Código tem que ser o mais eficiente possível.

## Realidade

- Otimizações prematuras devem ser evitadas.
- Outros fatores são mais relevantes que o desempenho:
  - Simplicidade
  - Legibilidade
  - Manutenibilidade
- Soluções simples para problemas Complexos.



# Obrigado!

## Dúvidas?

Agradecimentos:

- ▶ React Natal
- ▶ Rocketseat

Contato: Johnvictorio3@gmail.com

