

*a melhor maneira de limpar seu **JSX** com **objetos literais***

```
export default function Home() {  
  return (  
    <main>  
      {data.map(item => (  
        <>  
          {item.status.type === StatusType.approved && (  
            <Status variant="success">approved</Status>  
          )}  
          {item.status.type === StatusType.failed && (  
            <Status variant="error">failed</Status>  
          )}  
          {item.status.type === StatusType.adjustment &&  
            <Status variant="warn">adjustment</Status>  
          }  
        </>  
      )})  
    </main>  
  );  
}
```



*usando o melhor do **Typescript***



Isaac Gomes

TS

*Não é incomum termos que mapear Componentes no **Front-end** devido alguma **variant***

```
export default function Home() {  
  return (  
    <main>  
      {data.map(item => (  
        <>  
          {item.status.type === StatusType.approved && (  
            <Status variant="success">approved</Status>  
          )}  
          {item.status.type === StatusType.failed && (  
            <Status variant="error">failed</Status>  
          )}  
          {item.status.type === StatusType.adjustment && (  
            <Status variant="warn">adjustment</Status>  
          )}  
        </>  
      )}}  
    </main>  
  );  
}
```

*com isso a escolha mais comum são condicionais dentro do **JSX***




Isaac Gomes

TS

*podemos substituir isso por um **objeto***

```
export default function Home() {  
  return (  
    <main>  
      {data.map(item => (  
        <>  
          {item.status.type === StatusType.approved && (  
            <Status variant="success">approved</Status>  
          )}  
          {item.status.type === StatusType.failed && (  
            <Status variant="error">failed</Status>  
          )}  
          {item.status.type === StatusType.adjustment && (  
            <Status variant="warn">adjustment</Status>  
          )}  
        </>  
      )})}  
    </main>  
  );  
}
```

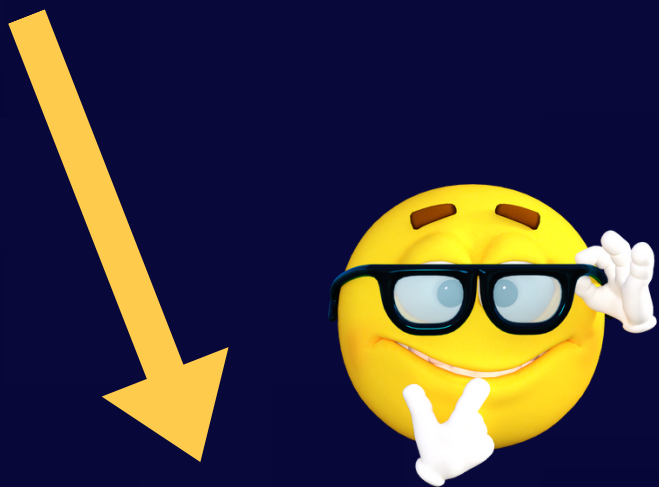

*mas, qual o melhor jeito de
definir o tipo do **objeto** ?????*



Isaac Gomes

TS

```
const statusElementsMap:  
  Record<StatusType, React.ReactElement> =  
  {}
```



*Podemos definir o tipo como um **Record** cujo a **key** vai ser um **enum** do tipo **StatusType** e o **value** vai ser do tipo **React.ReactElement***



Isaac Gomes

TS

```
const StatusElements:
  Record<StatusType, React.ReactElement> = {
  [StatusType.adjustment]: <Status variant="warn">adjustment</Status>,
  [StatusType.approved]: <Status variant="success">approved</Status>,
  [StatusType.failed]: <Status variant="error">failed</Status>,
}
```

*agora com um **objeto definido** tratando todos as opções do **enum***

```
const StatusElement = ({ status }: StatusElementParams):
  React.ReactElement => {
  return StatusElements[status] ?? <div />
}
```

*agora adicionamos uma **função** que retorna um **React.ReactElement** ela vai pegar o componente referente ao **enum**... em caso de erro retorna uma div*



Isaac Gomes

TS

Com isso Saímos disso →

```
export default function Home() {  
  return (  
    <main>  
      {data.map(item => (  
        <>  
          {item.status.type === StatusType.approved && (  
            <Status variant="success">approved</Status>  
          )}  
          {item.status.type === StatusType.failed && (  
            <Status variant="error">failed</Status>  
          )}  
          {item.status.type === StatusType.adjustment && (  
            <Status variant="warn">adjustment</Status>  
          )}  
        </>  
      )})}  
    </main>  
  );  
}
```



Isaac Gomes

TS

para isso

```
export default function Home() {  
  return (  
    <main>  
      {data.map(item =>  
        <StatusElement key={item.id} status={item.status.type} />  
      )}  
    </main>  
  );  
}
```



*perceba que nossa tratativa de
Variant fica no nosso **objeto literal**
limpando nosso **JSX***



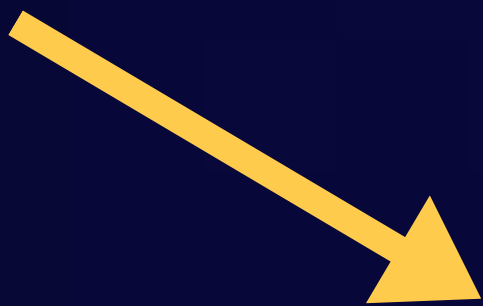
Isaac Gomes

TS

*qual vantagem de Usar dessa maneira ao inves de **IF**?*



```
enum StatusType {  
  approved = 'approved',  
  failed = 'failed',  
  adjustment = 'adjustment',  
  partial = 'partial'  
}
```



*vamos supor que um novo **StatusType** foi adicionado*



Isaac Gomes

TS

*Dessa maneira temos que **Mapear** todos os nosso casos*

```
const StatusElements: Record<StatusType, React.ReactElement> = {  
  [StatusType.adjustment]: <Status variant="warn">adjustment</Status>,  
  [StatusType.approved]: <Status variant="success">approved</Status>,  
  [StatusType.failed]: <Status variant="error">failed</Status>,  
}
```



```
/*
```

```
Property '[StatusType.partial]' is missing in type  
'{  
  adjustment: JSX.Element;  
  approved: JSX.Element;  
  failed: JSX.Element;  
'
```

```
*/
```



Isaac Gomes

TS

*podemos criar um type de strings
restrita para **substituir** o **enum***

```
type StatusType =  
  'approved' |  
  'failed' |  
  'adjustment' |  
  'partial'
```

*ambas são **validos** tudo
depende do seu **cenário***



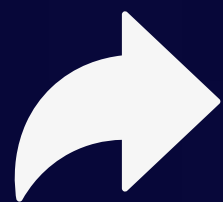
Isaac Gomes

TS

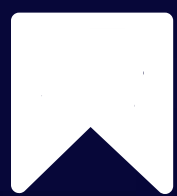
Gostou?



Curta



Compartilhe



Salve



Isaac Gomes

TS