

# Teste para Desenvolvedor Front-end

---

O teste é composto por duas partes: Exercícios e Projeto. A submissão do teste deve ser através do envio, por email, do repositório e da url do deploy do projeto. Pode ser utilizado o mesmo repositório para o projeto e os exercícios.

## Exercícios

### Exercício 1

Geralmente, quando você compra algo, é perguntado se o número do seu cartão de crédito, telefone ou resposta para sua pergunta secreta ainda está correto. No entanto, como alguém pode olhar por cima do seu ombro, você não quer que isso apareça em sua tela. Em vez disso, nós o mascaramos. Sua tarefa é escrever uma função `maskify`, que altera todos, exceto os últimos quatro caracteres, para "#".

Exemplos:

"4556364607935616" --> "#####5616"

"64607935616" --> "#####5616"

"1" --> "1"

"" --> ""

"Skippy" --> "##ippy"

"Nananananananananana Batman!" --> "#####man!"

Código:

```
function maskify(string) {  
  your code here...  
}  
  
module.exports = maskify;
```

### Exercício 2

Faça uma função que recebe um objeto como primeiro parâmetro e, como segundo parâmetro, um objeto com dados que vão atualizar o objeto do primeiro parâmetro.

Obs: se no objeto do segundo parâmetro tiver dados que o objeto do primeiro não tem, o valor não deve persistir no objeto de retorno da função. Somente são atualizados os dados que o objeto do primeiro parâmetro possuir.

Ex: `updateData({ name: "Marcos", country: "Brasil", age: 22 }, { country: "Japão", age: 33 })` --> saída: `{ name: 'Marcos', country: 'Japão', age: 33 }`

Ex: `updateData({ name: "Marcos", country: "Brasil", age: 22 }, { price: 89.9, amount: 15, description: "camiseta 100% algodão" })` --> saída: `{ name: "Marcos", country: "Brasil", age: 22 }`

Ex: `updateData({ name: "Rafael", country: "Chile", age: 42 }, { name: "Camiseta Polo", price: 59.9, amount: 30 })` --> saída: `{ name: "Rafael", country: "Chile", age: 42 }`

Código:

```
function updateData(currentObject, newDataObject) {  
  your code here...  
}  
  
module.exports = updateData;
```

### Exercício 3

Faça uma chamada para a api "rick and morty" e resgate informações dos seguintes personagens: Rick Sanchez, Morty Smith, Summer Smith, Beth Smith, Jerry Smith. Ajuste os dados para que fiquem igual à saída de exemplo abaixo.

Documentação

<https://rickandmortyapi.com/documentation/#rest>

Exemplo de Saida:

```
[  
  {  
    nome: 'Rick Sanchez',  
    genero: 'Homem',  
    avatar: 'https://rickandmortyapi.com/api/character/avatar/1.jpeg',  
    especie: 'Humano'  
  },  
  {  
    nome: 'Morty Smith',  
    genero: 'Homem',  
    avatar: 'https://rickandmortyapi.com/api/character/avatar/2.jpeg',  
    especie: 'Humano'  
  },  
  {  
    nome: 'Summer Smith',  
    genero: 'Mulher',  
    avatar: 'https://rickandmortyapi.com/api/character/avatar/3.jpeg',  
    especie: 'Humano'  
  },  
  {  
    nome: 'Beth Smith',  
    genero: 'Mulher',  
    avatar: 'https://rickandmortyapi.com/api/character/avatar/4.jpeg',  
    especie: 'Humano'  
  },  
  {  
    nome: 'Jerry Smith',  
    genero: 'Homem',  
    avatar: 'https://rickandmortyapi.com/api/character/avatar/5.jpeg',  
    especie: 'Humano'  
  }  
]
```

```
}  
]
```

Código:

```
async function getRickAndMortyCharacters() {  
  your code here...  
}  
  
module.exports = getRickAndMortyCharacters;
```

## Exercício 4

Faça uma função que verifica se a primeira letra de uma string é maiúscula, retornando true ou false.

Exemplos:

checkIfTheFirstLetterIsUppercase("Brasil") --> true

checkIfTheFirstLetterIsUppercase("mobiauto") --> false

checkIfTheFirstLetterIsUppercase("xXx xXx") --> false

checkIfTheFirstLetterIsUppercase("xDD") --> false

checkIfTheFirstLetterIsUppercase("Deu Certo!") --> true

Código:

```
function checkIfTheFirstLetterIsUppercase(word) {  
  your code here...  
}  
  
module.exports = checkIfTheFirstLetterIsUppercase;
```

## Projeto

O projeto consiste em construir duas páginas (Busca e Resultado), de acordo com as imagens no final desse documento, utilizando React.

Para obter os dados, consulte esta [API](#).

Para os componentes (Button e Select/Autocomplete), recomenda-se utilizar o framework Material-UI.

Para a estilização, utilizar alguma biblioteca CSS-in-JS, podendo ser o próprio Material-UI ou outra, como Styled-Components. A fonte deve ser a "Roboto".

Será avaliada a qualidade do código (semântica e legibilidade).

### Requisitos obrigatórios

- Next.js

- Redux ou Context API
- Deploy na Vercel

## Diferenciais

- Typescript
- Algum tipo de teste (unitário, integração ou e2e)

## Layout

### Busca sem modelo selecionado

### Tabela Fipe

Consulte o valor de um veículo de forma gratuita

Marca

Modelo

Consultar preço

### Busca com modelo selecionado

### Tabela Fipe

Consulte o valor de um veículo de forma gratuita

Marca  
Chevrolet

Modelo  
Cruze

Ano

Consultar preço

### Busca com ano selecionado

## Tabela Fipe

Consulte o valor de um veículo de forma gratuita

Marca  
Chevrolet

Modelo  
Cruze

Ano  
2019

Consultar preço

### Resultado

## Tabela Fipe: Preço Chevrolet Cruze 2019

**R\$ 91.618**

Este é o preço de compra do veículo