Métodos de String

- Strings no JS tem métodos específicos que facilitam o manuseio;
- Alguns métodos foram inseridos em novas atualizações do javascript, como os métodos padStart e padEnd.

Métodos para extrair partes de uma string

- slice(start, end)
  - [EXEMPLO]
  - `let str = "Apple, Banana, Kiwi";`
  - `let part = str.slice(7, 13);`
- substring(start, end)
  - `let str = "Apple, Banana, Kiwi";`
  - `let part = substring(7, 13);`
- substr(start, length)
  - `let str = "Apple, Banana, Kiwi";`
  - `let part = str.substr(7);`

Método para modificar o conteúdo de uma string

- string.replace()
  - `let text = "Please visit Microsoft!";`
  - `let newText = text.replace("Microsoft", "W3Schools");`
- expressões regulares
  - O parâmetro /i nos diz que o conteudo será trocado independentemente se a palavra for escrita maiscula ou minuscula
    - `let text = "Please visit Microsoft!";`
    - `let newText = text.replace(/MICROSOFT/i, "W3Schools");`
  - O parâmetro /g nos diz para trocar todas as aparições da palavra na string
    - `let text = "Please visit Microsoft and Microsoft!";`
    - `let newText = text.replace(/Microsoft/g, "W3Schools");`

Mais alguns métodos específicos de string

- toUpperCase(): utilizado para transformar o conteúdo da string para maiusculo
  - `let text1 = "Hello World!";`
  - `let text2 = text1.toUpperCase();`
- toLowerCase(): utilizado para transformar o conteúdo da string para minusculo

    - `let text1 = "Hello World!";`
    - `let text2 = text1.toLowerCase();`
- trim(): utilizado para remover espaços no início e final de uma string
    - `let text = "    Hello World!      ";`
    - `text.trim()`
- padStart(): complemento da string com o conteudo passado como parâmetro, mas no início
    - `let text = "5";`
    - `let padded = text.padStart(4,0);`
- padEnd(): complemento da string com o conteudo passado como parâmetro, mas ao final da string
    - `let text = "5";`
    - `let padded = text.padEnd(4,0);`
- split(): converte uma string em array através de um separado passado como parâmetro
    - `let text = "hello, world"`
    - `text.split(",")`

Referências

- https://www.w3schools.com/js/js_string_methods.asp

Exercícios

1. Através da string “Maria, Paulo, Moisés, Joel, Ana”, imprima todos os nomes um em cada linha no console
2. Troque todas as vírgulas na string abaixo por ponto final:
    a. “Olá, mundo, meu, nome, é, Juca”

Métodos de array

- concat: cria um novo array com os conteudos dos arrays
    - `const myGirls = ["Cecilie", "Lone"];`
    - `const myBoys = ["Emil", "Tobias", "Linus"];`
    - `const myChildren = myGirls.concat(myBoys);`
- join: converte um array para string onde é possível passar um separador para cada elemento
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.join(",");`
- push: insere um elemento ao final do array
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.push("Kiwi");`
- pop: remove o último elemento do array
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`

    - `fruits.pop();`
- shift: remove o primeiro elemento do array
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.shift();`
- unshift: adiciona um elemento no inicio do array
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.unshift("Lemon");`
- slice: pega um “pedaço” do array
    - `const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];`
    - `const citrus = fruits.slice(1);`

- splice: pode ser usado para inserir novos elementos no array. O primeiro argumento define a posição onde serão inseridos os elementos, o segundo define quantos elementos serão removidos.
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.splice(2, 0, "Lemon", "Kiwi");`
    - Também pode ser usado para remover elementos
        - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
        - `fruits.splice(0, 1);`

- reverse: inverte as posições do array
    - `const fruits = ["Banana", "Orange", "Apple", "Mango"];`
    - `fruits.reverse()`

Métodos de Array específicos que iteram elementos

- Além dos métodos que vimos anteriormente, no JS a classe de Array tem uma série de métodos que entraram na linguagem com a evolução do ES6+ que se propõe a percorrer o array e realizar uma ação específica.
- Vamos usar muito esses métodos para conseguir trabalhar com nossos arrays.

forEach

- forEach é o método mais genérico, ele não tem um propósito específico.
- O forEach percorre o array e para cada elemento executa o bloco que definimos.
- Na maior parte dos casos ele pode ser substituído por um método específico.
- Parâmetros:
    - função de callback que será executada para cada um dos elementos (obrigatório).
    - argumento this (opcional).
- Retorno:
    - undefined (sempre!)

- Função de callback:
    - Será executada para cada um dos elementos do array.
    - Parâmetros:
        - currentValue (obrigatório): elemento que está sendo iterado.
        - index (opcional): índice do elemento que está sendo iterado.
        - array (opcional): array original.
    - Retorno: não impacta em nada na execução.
- [EXEMPLO]

```
const users = ['João', 'Juca', 'Gabriel', 'Matheus']
users.forEach(user => {
  console.log(user)
})
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

map

- Método que percorre os elementos do array e cria um novo array modificando ou mapeando os elementos do array original à partir da função passada como argumento.
- Parâmetros:
    - função de callback que será executada para cada um dos elementos (obrigatório).
    - argumento this (opcional).
- Retorno:
    - Novo array com os elementos modificados.
- Função de callback:
    - Será executada para cada um dos elementos do array.
    - Parâmetros:
        - currentValue (obrigatório): elemento que está sendo iterado.
        - index (opcional): índice do elemento que está sendo iterado.
        - array (opcional): array original.
    - Retorno: Valor que será o novo elemento da posição do array que está sendo iterado.
- [EXEMPLO]

```
const array = [1, 3, 5, 8]
const mappedArray = array.map(elem => elem * 10)
console.log(array)
console.log(mappedArray)
```

- [EXEMPLO]

```
const users = ['João', 'Juca', 'Gabriel', 'Matheus']
const values = users.map(userName => {
```

```
    return {
      name: userName,
      role: 'Estudante',
    }
})
console.log(users)
console.log(values)
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

filter

- Método que percorre os elementos do array e cria um novo array contendo SOMENTE os elementos que satisfaçam uma condição.
- Parâmetros:
  - função de callback que será executada para cada um dos elementos (obrigatório).
  - argumento this (opcional).
- Retorno:
  - Novo array com os elementos que satisfizeram a condição.
- Função de callback:
  - Será executada para cada um dos elementos do array.
- Parâmetros:
  - currentValue (obrigatório): elemento que está sendo iterado.
  - index (opcional): índice do elemento que está sendo iterado.
  - array (opcional): array original.
- Retorno:
  - true: caso o elemento iterado passe pelo filtro.
  - false: caso contrário.
- [EXEMPLO]

```
const users = [
  { name: 'João', role: 'Professor' },
  { name: 'Juca', role: 'Estudante' },
  { name: 'Márcia', role: 'Estudante' },
  { name: 'Pedro', role: 'Estudante' },
  { name: 'Matheus', role: 'Professor' },
  { name: 'Júlia', role: 'Estudante' },
]
const students = users.filter((user) => {
  return user.role === 'Estudante'
```

```
})
console.log(users)
console.log(students)
```

- [EXEMPLO]

```
const numbers = [-8, 12, 76, 100, -230, -7, 120]
const negatives = numbers.filter((number) => {
  return number < 0
})
console.log(numbers)
console.log(negatives)
```

Referência

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

find

- Método para percorrer os elementos do array buscando o PRIMEIRO que satisfaça a condição informada.
- Parâmetros:
  - função de callback que será executada para cada um dos elementos (obrigatório).
  - argumento this (opcional).
- Retorno:
  - Elemento que satisfaz a condição, caso exista.
  - undefined caso contrário.
- Função de callback:
  - Será executada para cada um dos elementos do array.
  - Parâmetros:
    - currentValue (obrigatório): elemento que está sendo iterado.
    - index (opcional): índice do elemento que está sendo iterado.
    - array (opcional): array original.
  - Retorno:
    - true: caso o elemento seja o elemento buscado.
    - false: caso contrário.
- [EXEMPLO]

```
const numbers = [-8, 12, 76, 100, -230, -7, 120]
const firstPositive = numbers.find((number) => {
  return number > 0
})
console.log(numbers)
console.log(firstPositive)
```

- [EXEMPLO]

```
const users = [
  { name: 'João', role: 'Professor', id: 1 },
  { name: 'Juca', role: 'Estudante', id: 3 },
  { name: 'Márcia', role: 'Estudante', id: 4 },
  { name: 'Pedro', role: 'Estudante', id: 5 },
  { name: 'Matheus', role: 'Professor', id: 2 },
  { name: 'Júlia', role: 'Estudante', id: 6 },
]
const findUser = users.find((user) => {
  return user.id === 3
})
console.log(users)
console.log(findUser)
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find

every

- Método que percorre os elementos do array e informa se TODOS eles satisfazem uma condição.
- Parâmetros:
  - função de callback que será executada para cada um dos elementos (obrigatório).
  - argumento this (opcional).
- Retorno:
  - true: caso todos os elementos do array satisfizeram a condição.
  - false: caso contrário.
- Função de callback:
  - Será executada para cada um dos elementos do array.
- Parâmetros:
  - currentValue (obrigatório): elemento que está sendo iterado.
  - index (opcional): índice do elemento que está sendo iterado.
  - array (opcional): array original.
- Retorno:
  - true: caso o elemento iterado satisfaça a condição.
  - false: caso contrário.
- [EXEMPLO]

```
const users = [
  { name: 'João', role: 'Professor', id: 1 },
  { name: 'Juca', role: 'Estudante', id: 3 },
  { name: 'Márcia', role: 'Estudante', id: 4 },
```

```
    { name: 'Pedro', role: 'Estudante', id: 5 },
    { name: 'Matheus', role: 'Professor', id: 2 },
    { name: 'Júlia', role: 'Estudante', id: 6 },
  ]
  const allStudents = users.every((user) => {
    return user.role === 'Estudante'
  })
  console.log(users)
  console.log(allStudents)
```

- [EXEMPLO]

```
  const numbers = [-8, 12, 76, 100, -230, -7, 120]
  const allNumbers = numbers.every((number) => {
    return !Number.isNaN(number)
  })
  console.log(numbers)
  console.log(allNumbers)
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/every

some

- Método para percorrer os elementos para saber se existe algum que satisfaça a condição informada.
- Parâmetros:
  - função de callback que será executada para cada um dos elementos (obrigatório).
  - argumento this (opcional).
- Retorno:
  - true: Caso exista algum elemento que satisfez a condição.
  - false: Caso não exista algum elemento que satisfez a condição.
- Função de callback:
  - Será executada para cada um dos elementos do array.
- Parâmetros:
  - currentValue (obrigatório): elemento que está sendo iterado.
  - index (opcional): índice do elemento que está sendo iterado.
  - array (opcional): array original.
- Retorno:
  - true: caso o elemento se encaixe na condição buscada.
  - false: caso contrário.
- [EXEMPLO]

```
  const users = [
```

```
    { name: 'João', role: 'Professor', id: 1 },
    { name: 'Juca', role: 'Estudante', id: 3 },
    { name: 'Márcia', role: 'Estudante', id: 4 },
    { name: 'Pedro', role: 'Estudante', id: 5 },
    { name: 'Matheus', role: 'Professor', id: 2 },
    { name: 'Júlia', role: 'Estudante', id: 6 },
  ]
  const someStudent = users.some((user) => {
    return user.role === 'Estudante'
  })
  console.log(users)
  console.log(someStudent)
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/some

sort

- Método de ordenação do array.
- Parâmetro:
  - Função de callback que irá comparar dois elementos identificando a precedência entre eles.
- Retorno:
  - Array ordenado (mesmo assim altera o array original)
- Função de callback:
  - Utilizada para identificar qual a precedência entre dois elementos do array.
  - Parâmetros:
    - firstElem (obrigatório): elemento qualquer do array.
    - secondElem (obrigatório): elemento qualquer do array.
  - Retorno:
    - Número negativo (ex: -1): caso firstElem deva aparecer ANTES de secondElem no array ordenado.
    - Número positivo (ex: 1): caso firstElem deva aparecer DEPOIS de secondElem no array ordenado.
    - Zero (0): Caso não haja diferença de precedência entre firstElem e secondElem
- [EXEMPLO]

```
  const users = [
    { name: 'João', role: 'Professor', id: 1 },
    { name: 'Juca', role: 'Estudante', id: 3 },
    { name: 'Pedro', role: 'Estudante', id: 5 },
    { name: 'Matheus', role: 'Professor', id: 2 },
```

```
    { name: 'Júlia', role: 'Estudante', id: 6 },
    { name: 'Márcia', role: 'Estudante', id: 4 },
  ]
  const sortedArray = users.sort((a, b) => {
    if (a.id < b.id) return -1
    if (a.id > b.id) return 1
    return 0
  })
  console.log(users)
  console.log(sortedArray)
```

- [EXEMPLO]

```
  const sortedArray = users.sort((a, b) => {
    return a.id - b.id
  })
```

- EXEMPLO

```
  const sortedArray = [...users].sort((a, b) => {
    return a.id - b.id
  })
```

- EXEMPLO

```
  const sortedArray = users.slice().sort((a, b) => {
    return a.id - b.id
  })
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

reduce

- Método que percorre os elementos do array e realiza uma redução resultando em um único valor.
- Parâmetros:
  - callback: função de callback que será executada para cada um dos elementos (obrigatório).
  - valorInicial: valor que será usado como primeiro argumento na chamada da função de callback na primeira execução.
- Retorno:
  - Valor resultado da redução.
- Função de callback:
  - Parâmetros:

        - acc (obrigatório): valor inicial ou valor retornado pela execução da função de callback no elemento anterior.
        - currentValue (obrigatório): elemento que está sendo iterado.
        - index (opcional): índice do elemento que está sendo iterado.
        - array (opcional): array original.
    - Retorno:
    - Valor que será o novo acc para o próximo elemento iterado. Caso seja a última iteração, será o valor que o método reduce retornará.
- [EXEMPLO]

```
const numbers = [-8, 12, 76, 100, -230, -7, 120]
const sum = numbers.reduce((acc, number) => {
  return acc + number
})
console.log(numbers)
console.log(sum)
```

- [EXEMPLO]

```
const products = [
  { name: 'banana', category: 'fruta' },
  { name: 'alface', category: 'verdura' },
  { name: 'maçã', category: 'fruta' },
  { name: 'cenoura', category: 'legume' },
  { name: 'pêssego', category: 'fruta' },
  { name: 'couve', category: 'verdura' },
]
const categoryQtd = products.reduce(
  (acc, product) => {
    acc[product.category] = acc[product.category] + 1
    return acc
  },
  {
    fruta: 0,
    verdura: 0,
    legume: 0,
  }
)
console.log(products)
console.log(categoryQtd)
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce