



# DICIONÁRIOS

## ESTRUTURA DE DADOS

### CST em Desenvolvimento de Software Multiplataforma



**PROF. Me. TIAGO A. SILVA**



# LIVRO DE REFERÊNCIA DA DISCIPLINA

- **BIBLIOGRAFIA BÁSICA:**

- GRONER, Loiane. **Estrutura de dados e algoritmos com JavaScript**: escreva um código JavaScript complexo e eficaz usando a mais recente ECMAScript. **São Paulo: Novatec Editora, 2019.**

- **NESTA AULA:**

- **Capítulo 8**



# PARA SOBREVIVER AO JAVASCRIPT

Non-zero value



null



0



undefined



# DICIONÁRIOS EM JAVASCRIPT

- Os dicionários são estruturas de dados que armazenam **pares chave-valor**.
- Em JavaScript, são geralmente implementados usando objetos ou a classe Map, que foi introduzida no ECMAScript 6.
- Um dicionário permite **armazenar dados de forma associativa**, onde uma chave é usada para acessar um valor correspondente.

# USO DE DICIONÁRIOS EM JAVASCRIPT

- Dicionários em JavaScript (implementados com objetos ou a classe **Map**) e em estruturas de dados têm uma ampla variedade de aplicações devido à sua capacidade de armazenar e acessar dados rapidamente por meio de chaves.
- Dicionários são usados para indexar dados, facilitando pesquisas rápidas por chaves, útil em cenários como bancos de dados em memória.
- Dicionários são amplamente **usados para armazenar configurações, opções e parâmetros de aplicativos ou funções.**

# USO DE DICIONÁRIOS EM JAVASCRIPT

- Dicionários são usados para representar **respostas de APIs** e armazenar dados recebidos do servidor.

```
1  const apiResponse = {  
2      status: "success",  
3      data: {  
4          id: 1,  
5          nome: "Produto A",  
6          preco: 100  
7      }  
8  };  
9  console.log(apiResponse.data.nome); // Saída: Produto A
```

# POR QUE USAR DICIONÁRIOS?

- **Eficiência**
  - Acesso rápido aos dados por meio de chaves.
- **Versatilidade**
  - Suportam vários casos de uso em sistemas pequenos ou complexos.
- **Simplicidade**
  - São fáceis de implementar e ler no código.
- **Escalabilidade**
  - Funcionam bem em aplicações de pequeno e grande porte.

# DICIONÁRIOS EM JAVASCRIPT

```
1  // Criando um dicionário com um objeto
2  const dicionario = {
3      nome: "João",
4      idade: 25,
5      profissao: "Engenheiro"
6  };
7
8  console.log(dicionario.nome); // Saída: João
9  console.log(dicionario["idade"]); // Saída: 25
```



# DICIONÁRIOS COM OBJETOS

- Em JavaScript, os objetos são a forma mais comum de implementar dicionários. **Chaves podem ser adicionadas dinamicamente ou removidas.**

```
12  const dicionario = {};  
13  
14  // Adicionando chaves e valores  
15  dicionario["cor"] = "azul";  
16  dicionario.tamanho = "grande";  
17  
18  console.log(dicionario); // Saída: { cor: 'azul', tamanho: 'grande' }  
19  
20  // Removendo uma chave  
21  delete dicionario["cor"];  
22  console.log(dicionario); // Saída: { tamanho: 'grande' }
```

# ITERANDO SOBRE UM DICIONÁRIO

```
25  const dicionario = {
26    fruta: "maçã",
27    cor: "vermelho",
28    preco: 5
29  };
30
31  // Usando `for...in`
32  for (const chave in dicionario) {
33    console.log(`${chave}: ${dicionario[chave]}`);
34  }
```

# MÉTODOS PARA TRABALHAR COM DICIONÁRIOS

```
37  const dicionario = {
38      nome: "João",
39      idade: 25,
40      cidade: "Rio de Janeiro"
41  };
42
43  // Obter todas as chaves
44  console.log(Object.keys(dicionario)); // Saída: ["nome", "idade", "cidade"]
45
46  // Obter todos os valores
47  console.log(Object.values(dicionario)); // Saída: ["João", 25, "Rio de Janeiro"]
48
49  // Obter pares chave-valor
50  Object.entries(dicionario).forEach(([chave, valor]) => {
51      console.log(`${chave}: ${valor}`);
52  });
```

# VALIDANDO E TRABALHANDO COM CHAVES

```
56  const dicionario = { fruta: "maçã" };
57
58  if ("fruta" in dicionario) {
59      console.log("Chave encontrada!");
60  } else {
61      console.log("Chave não encontrada!");
62  }
63
64  const dados = { nome: "Carlos" };
65
66  if (!("idade" in dados)) {
67      dados["idade"] = 28;
68  }
69
70  console.log(dados); // Saída: { nome: "Carlos", idade: 28 }
```

# DICIONÁRIOS COM A CLASSE MAP

- Embora objetos sejam amplamente usados, a classe **Map** oferece **vantagens, como suporte a qualquer tipo de chave.**

```
37  const mapa = new Map();
38
39  // Adicionando pares chave-valor
40  mapa.set("nome", "Ana");
41  mapa.set("idade", 30);
42
43  // Obtendo valores
44  console.log(mapa.get("nome")); // Saída: Ana
45
46  // Verificando se uma chave existe
47  console.log(mapa.has("idade")); // Saída: true
48
49  // Removendo uma chave
50  mapa.delete("idade");
51  console.log(mapa.has("idade")); // Saída: false
```

## DICIONÁRIOS COM A CLASSE MAP - ITERANDO

```
53  const mapa = new Map([
54    ["chave1", "valor1"],
55    ["chave2", "valor2"]
56  ]);
57
58  // Iterando com for...of
59  for (const [chave, valor] of mapa) {
60    console.log(`${chave}: ${valor}`);
61  }
```

# DIFERENÇAS ENTRE OBJECT E MAP

Aspecto	Object	Map
Tipos de Chave	Apenas strings e símbolos	Qualquer tipo, incluindo objetos, funções, números, etc
Ordem das Chaves	Não garantida (pode variar)	Preserva a ordem de inserção
Iteração	Necessita <b>for...in</b> ou <b>Object.keys</b> .	Iteração direta com <b>for...of</b>
Tamanho	Precisa calcular manualmente	Acesso rápido com <b>map.size</b>
Métodos Utilitários	Limitados	Métodos como <b>set</b> , <b>get</b> , <b>delete</b> , <b>clear</b>
Desempenho em Grandes Conjuntos	Menor eficiência em certos casos	Melhor desempenho para adição e remoção frequente.

## MÉTODOS DA CLASSE MAP

- **Map** fornece métodos convenientes para operações comuns, como:
  - **set** Adicionar ou atualizar uma chave.
  - **get** Obter o valor associado a uma chave.
  - **delete** Remover uma chave.
  - **clear** Remover todos os itens.



# QUANDO USAR OBJECT OU MAP

- **Use Object**

- Quando as chaves são strings ou símbolos.
- Para estruturas de dados simples ou com pouca manipulação de chaves.
- Quando não for necessária a ordem de inserção das chaves.

- **Use Map**

- Quando as chaves podem ser de qualquer tipo.
- Para garantir a ordem de inserção.
- Quando precisa de operações frequentes de inserção e remoção em grandes conjuntos de dados.
- Para evitar colisões de nomes com as propriedades herdadas de objetos (**toString**, **hasOwnProperty**).

# EXEMPLO COMPARATIVO: USANDO OBJECT

```
74  const dicionario = {};  
75  dicionario["nome"] = "Carlos";  
76  dicionario[42] = "Número";  
77  
78  // Acesso  
79  console.log(dicionario["nome"]); // Saída: Carlos  
80  console.log(dicionario["42"]); // Saída: Número (chaves numéricas convertidas para string)  
81  
82  // Iteração  
83  for (const chave in dicionario) {  
84  |   console.log(`${chave}: ${dicionario[chave]}`);  
85  }
```

## EXEMPLO COMPARATIVO: USANDO MAP

```
88  const mapa = new Map();
89  mapa.set("nome", "Carlos");
90  mapa.set(42, "Número");
91
92  // Acesso
93  console.log(mapa.get("nome")); // Saída: Carlos
94  console.log(mapa.get(42));    // Saída: Número
95
96  // Iteração
97  for (const [chave, valor] of mapa) {
98  |   console.log(`${chave}: ${valor}`);
99  }
```

## EXERCÍCIOS

- 1) Crie um objeto para armazenar informações de alunos, onde a chave é o número de matrícula e o valor é o nome do aluno. Adicione pelo menos 3 alunos ao dicionário e exiba os nomes iterando sobre as chaves.
- 2) Implemente uma função que receba uma string e retorne um objeto onde as chaves são os caracteres e os valores, o número de vezes que aparecem na string. Ignore os espaços.
- 3) Crie um objeto para armazenar informações de produtos de uma loja. As chaves devem ser os códigos dos produtos, e os valores, objetos contendo nome e preço. Itere sobre o dicionário e exiba os produtos com preços acima de R\$ 50.

## EXERCÍCIOS

- 4) Crie uma função que receba um dicionário e remova todas as chaves cujo valor seja **null** ou **undefined**.
- 5) Crie uma função que receba uma frase e use um **Map** para contar quantas vezes cada palavra aparece.
- 6) Use um **Map** para criar um contador de frequência de letras em uma string, diferenciando letras maiúsculas e minúsculas.
- 7) Implemente uma função que converta um **object** para um **Map** e outra que faça o caminho inverso, convertendo um **Map** para um **object**.

# OBRIGADO!

- Encontre este **material on-line** em:
  - [www.tiago.blog.br](http://www.tiago.blog.br)
  - Plataforma Teams
- Em caso de **dúvidas**, entre em contato:
  - **Prof. Tiago:** [tiago.silva238@fatec.sp.gov.br](mailto:tiago.silva238@fatec.sp.gov.br)

