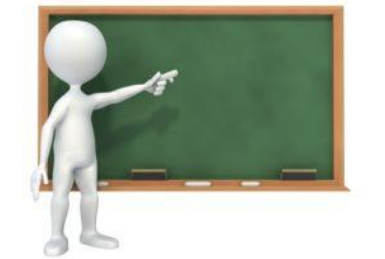
Centro Paula Souza

Fatec Jahu

IMPLEMENTAÇÃO DE ARRAYS, PILHAS E FILAS MANUALMENTE

ESTRUTURA DE DADOS

CST em Desenvolvimento de Software Multiplataforma



Prof. Me. Tiago A. Silva

@prof.tiagotas

PARA SOBREVIVER AO JAVASCRIPT

Non-zero value

0

null

undefined

ARRAY

Exemplo de Implementação em JavaScript, sem o uso de funções nativas

IMPLEMENTAÇÃO DE UM ARRAY

- Estrutura interna (**`items`**): Usamos um objeto simples para armazenar os itens do array, onde as chaves são os índices do array.
  - **`adicionar(elemento)`**: Adiciona um novo elemento ao final do array, usando o índice correspondente ao valor do tamanho atual (**`this.tamanho`**). Depois, incrementamos o tamanho.
  - **`remover()`**: Remove o último elemento do array, usando o índice que corresponde a tamanho - 1, e decrementa o tamanho.
  - **`obterElemento(indice)`**: Acessa e retorna o elemento no índice especificado, se ele for válido (entre 0 e tamanho - 1).
  - **`tamanhoArray()`**: Retorna o número de elementos no array.
  - **`limpar()`**: Remove todos os elementos do array, redefinindo o objeto **`items`** e o tamanho para zero.
- Essa implementação manual simula o comportamento de um array básico sem recorrer às funções nativas do JavaScript, como **`push()`**, **`pop()`**, ou **`length`**, e usa a lógica de manipulação de índices para armazenar e acessar os dados.

```
class MeuArray {

    constructor()
    {
        this.items = {};  // Usamos um objeto para armazenar os itens do array
        this.tamanho = 0;  // Mantemos o controle do tamanho do array
    }

    // Adiciona um elemento ao final do array
    adicionar(elemento)
    {
        this.items[this.tamanho] = elemento;  // Insere o elemento na posição atual do tamanho
        this.tamanho++;  // Incrementa o tamanho
    }

    // Remove o último elemento do array
    remover()
    {
        if (this.tamanho === 0) {
            return undefined;  // Se o array estiver vazio, não há o que remover
        }

        const ultimoItem = this.items[this.tamanho - 1];  // Armazena o último item
        delete this.items[this.tamanho - 1];  // Remove o último item do array
        this.tamanho--;  // Decrementa o tamanho

        return ultimoItem;  // Retorna o item removido
    }
```

```
29
30      // Acessa o elemento em um índice específico
31      obterElemento(indice)
32      {
33          if (indice < 0 || indice >= this.tamanho) {
34              return undefined;  // Se o índice estiver fora do alcance, retorna undefined
35          }
36          return this.items[indice];  // Retorna o item no índice solicitado
37      }
38
39      // Retorna o tamanho do array
40      tamanhoArray()
41      {
42          return this.tamanho;  // Retorna o valor do tamanho atual do array
43      }
44
45      // Remove todos os elementos do array
46      limpar()
47      {
48          this.items = {};  // Limpa o objeto
49          this.tamanho = 0;  // Reinicializa o tamanho
50      }
51  }
```

ARRAY – COMO USAR

```
53  // Exemplo de uso
54  let minha_variavel = new MeuArray();
55
56  minha_variavel.adicionar(10);
57  minha_variavel.adicionar(20);
58  minha_variavel.adicionar(30);
59
60  console.log(minha_variavel.obterElemento(1));  // Saída: 20
61  console.log(minha_variavel.tamanhoArray());  // Saída: 3
62
63  console.log(minha_variavel.remover());  // Saída: 30 (Remove o último elemento)
64  console.log(minha_variavel.tamanhoArray());  // Saída: 2
```

FILA

Exemplo de Implementação em JavaScript, sem o uso de funções nativas

IMPLEMENTAÇÃO DE UMA FILA

- Estrutura interna (**`items`**): Usamos um objeto para armazenar os elementos da fila. As chaves do objeto representam as posições dos elementos na fila.
  - **`enqueue(elemento)`**: Adiciona um novo elemento ao final da fila. O índice do fim da fila (**`this.fim`**) é incrementado cada vez que um novo elemento é inserido.
  - **`dequeue()`**: Remove o primeiro elemento da fila. O índice do início (**`this.inicio`**) é incrementado após a remoção de um elemento, e o item correspondente é excluído do objeto.
  - **`front()`**: Retorna o elemento na frente da fila sem removê-lo.
  - **`isEmpty()`**: Verifica se a fila está vazia, comparando os índices de início e fim.
  - **`size()`**: Calcula o número de elementos na fila subtraindo o índice do início do índice do fim.
  - **`clear()`**: Limpa completamente a fila, reinicializando os valores.
- Essa implementação manual simula o comportamento de uma fila usando apenas um objeto como armazenamento interno e gerenciando manualmente os índices para garantir o funcionamento correto da fila sem o uso de funções nativas como **`push()`** e **`shift()`**.

```
class Fila {
    constructor()
    {
        this.items = {};  // Usamos um objeto para armazenar os itens
        this.inicio = 0;  // Representa o índice do início da fila
        this.fim = 0;     // Representa o índice do fim da fila
    }

    // Adiciona um elemento ao final da fila (enqueue)
    enqueue(elemento)
    {
        this.items[this.fim] = elemento;  // Coloca o elemento no fim da fila
        this.fim++;  // Incrementa o índice do fim da fila
    }
```

```
15
16      // Remove e retorna o primeiro elemento da fila (dequeue)
17      dequeue()
18      {
19          if (this.isEmpty()) {
20              return undefined;  // Se a fila estiver vazia, retorna undefined
21          }
22
23          const item = this.items[this.inicio];  // Obtém o primeiro elemento
24          delete this.items[this.inicio];  // Remove o item do início da fila
25          this.inicio++;  // Move o índice do início para o próximo item
26
27          // Quando o início e o fim estiverem alinhados, redefine a fila
28          if (this.inicio === this.fim) {
29              this.inicio = 0;
30              this.fim = 0;
31          }
32
33          return item;  // Retorna o item removido
34      }
35
36      // Retorna o primeiro elemento da fila sem removê-lo (peek)
37      front()
38      {
39          if (this.isEmpty()) {
40              return undefined;  // Se a fila estiver vazia, retorna undefined
41          }
42          return this.items[this.inicio];  // Retorna o primeiro elemento
43      }
```

```
44
45      // Verifica se a fila está vazia
46      isEmpty()
47      {
48          return this.fim === this.inicio;  // Verifica se os índices estão iguais
49      }
50
51      // Retorna o tamanho da fila
52      size()
53      {
54          return this.fim - this.inicio;  // Calcula a diferença entre fim e início
55      }
56
57      // Limpa a fila
58      clear()
59      {
60          this.items = {};
61          this.inicio = 0;
62          this.fim = 0;
63      }
64  }
```

FILA – COMO USAR

```
65
66  // Exemplo de uso
67  let minha_variavel = new Fila();
68
69  minha_variavel.enqueue("Cliente 1");
70  minha_variavel.enqueue("Cliente 2");
71  minha_variavel.enqueue("Cliente 3");
72
73  console.log(minha_variavel.front());  // Saída: "Cliente 1"
74
75  console.log(minha_variavel.dequeue());  // Saída: "Cliente 1"
76  console.log(minha_variavel.dequeue());  // Saída: "Cliente 2"
77
78  minha_variavel.enqueue("Cliente 4");
79
80  console.log(minha_variavel.size());  // Saída: 2 (Cliente 3 e Cliente 4 ainda estão na fila)
81  console.log(minha_variavel.front());  // Saída: "Cliente 3"
```

PILHA

Exemplo de Implementação em JavaScript, sem o uso de funções nativas

IMPLEMENTAÇÃO DE UMA PILHA

- Estrutura interna (**`items`**): Usamos um objeto para armazenar os elementos da pilha, com chaves representando as posições.
  - **`adicionar(elemento)`**: Insere o elemento no topo da pilha, na posição correspondente ao tamanho atual. Em seguida, incrementa o tamanho da pilha.
  - **`remover()`**: Remove e retorna o elemento do topo da pilha, ajustando o tamanho após a remoção. Verifica se a pilha está vazia antes de tentar remover.
  - **`topo()`**: Retorna o elemento no topo da pilha sem removê-lo.
  - **`estaVazia()`**: Verifica se a pilha está vazia, retornando **`true`** ou **`false`**.
  - **`tamanhoPilha()`**: Retorna o número de elementos na pilha.
  - **`limpar()`**: Remove todos os elementos da pilha, redefinindo os valores de **`items`** e tamanho.
- Essa implementação simula o comportamento básico de uma pilha, sem recorrer a métodos nativos como **`push()`** e **`pop()`**, e realiza o gerenciamento de dados e índices de maneira manual.

```
class MinhaPilha {

    constructor() {
        this.items = {};  // Usamos um objeto para armazenar os elementos da pilha
        this.tamanho = 0;  // Mantemos o controle do tamanho da pilha
    }

    // Adiciona um elemento ao topo da pilha
    adicionar(elemento)
    {
        this.items[this.tamanho] = elemento;  // Insere o elemento na posição atual do tamanho
        this.tamanho++;  // Incrementa o tamanho
    }

    // Remove e retorna o elemento do topo da pilha
    remover()
    {
        if (this.tamanho === 0) {
            return undefined;  // Se a pilha estiver vazia, retorna undefined
        }

        const ultimoItem = this.items[this.tamanho - 1];  // Pega o item no topo da pilha
        delete this.items[this.tamanho - 1];  // Remove o item do topo
        this.tamanho--;  // Decrementa o tamanho

        return ultimoItem;  // Retorna o item removido
    }
```

```
28
29      // Retorna o elemento no topo da pilha sem removê-lo
30      topo()
31      {
32          if (this.tamanho === 0) {
33              return undefined;  // Se a pilha estiver vazia, retorna undefined
34          }
35          return this.items[this.tamanho - 1];  // Retorna o item no topo
36      }
37
38      // Verifica se a pilha está vazia
39      estaVazia()
40      {
41          return this.tamanho === 0;  // Verifica se o tamanho da pilha é zero
42      }
43
44      // Retorna o número de elementos na pilha
45      tamanhoPilha()
46      {
47          return this.tamanho;  // Retorna o tamanho da pilha
48      }
49
50      // Limpa a pilha
51      limpar()
52      {
53          this.items = {};  // Reseta os itens
54          this.tamanho = 0;  // Reinicializa o tamanho
55      }
56  }
```

PILHA – COMO USAR

```
57
58  // Exemplo de uso
59  let minha_variavel = new MinhaPilha();
60
61  minha_variavel.adicionar(10);
62  minha_variavel.adicionar(20);
63  minha_variavel.adicionar(30);
64
65  console.log(minha_variavel.topo());  // Saída: 30 (Elemento no topo)
66
67  console.log(minha_variavel.remover());  // Saída: 30 (Remove o elemento do topo)
68  console.log(minha_variavel.topo());  // Saída: 20 (Agora o topo é 20)
69
70  console.log(minha_variavel.tamanhoPilha());  // Saída: 2 (Dois elementos restantes)
```

OBRIGADO!

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