Centro Paula Souza

Fatec Jahu

IMPLEMENTAÇÃO DE FILAS EM JAVASCRIPT

ESTRUTURA DE DADOS

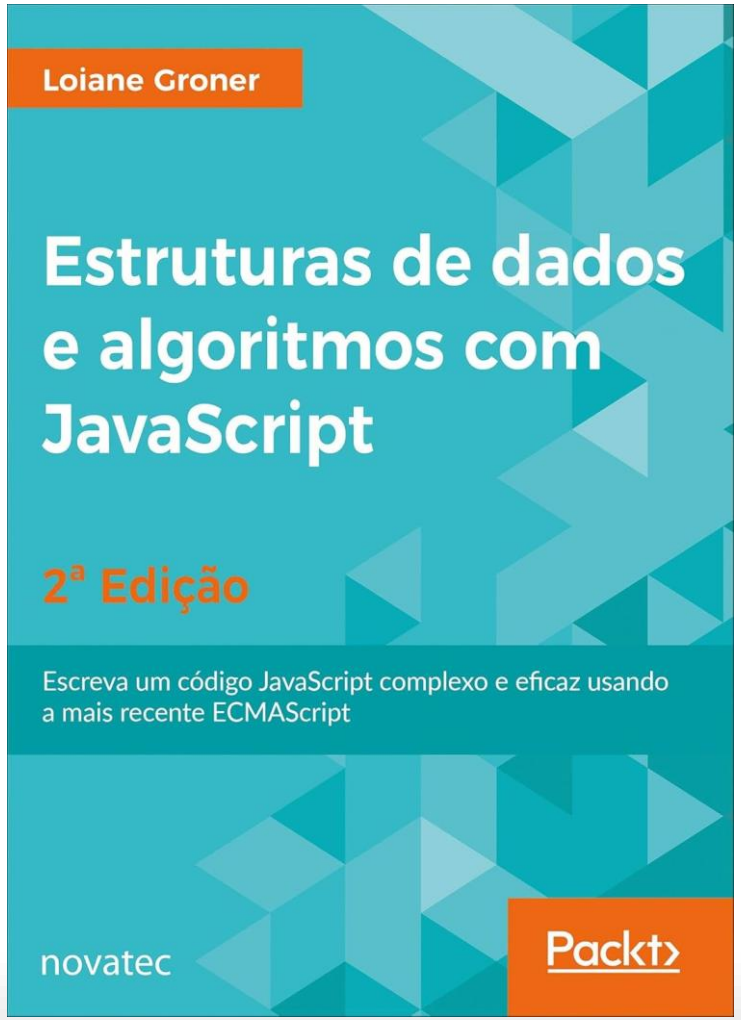
CST em Desenvolvimento de Software Multiplataforma

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 ECMASript. **São Paulo: Novatec Editora**, 2019.

- **NESTA AULA:**
  - **Capítulo 4**



PARA SOBREVIVER AO JAVASCRIPT

Non-zero value

0

null

undefined

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.

IMPLEMENTAÇÃO DE UMA FILA

- 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()`**.

REGRAS DA FILA

FIFO
First in – First out

Primeiro que entra

Primeiro que sai

LILO
Last in – Last out

Último que entra

Último que saí

ADICIONANDO ELEMENTOS À FILA (ENQUEUE)

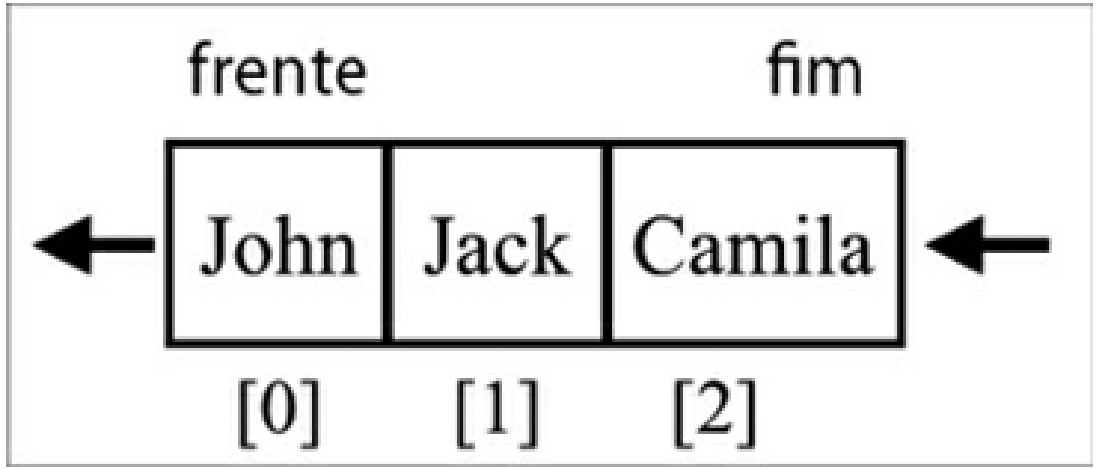


Figura 5.2

ADICIONANDO ELEMENTOS DA FILA (DEQUEUE)

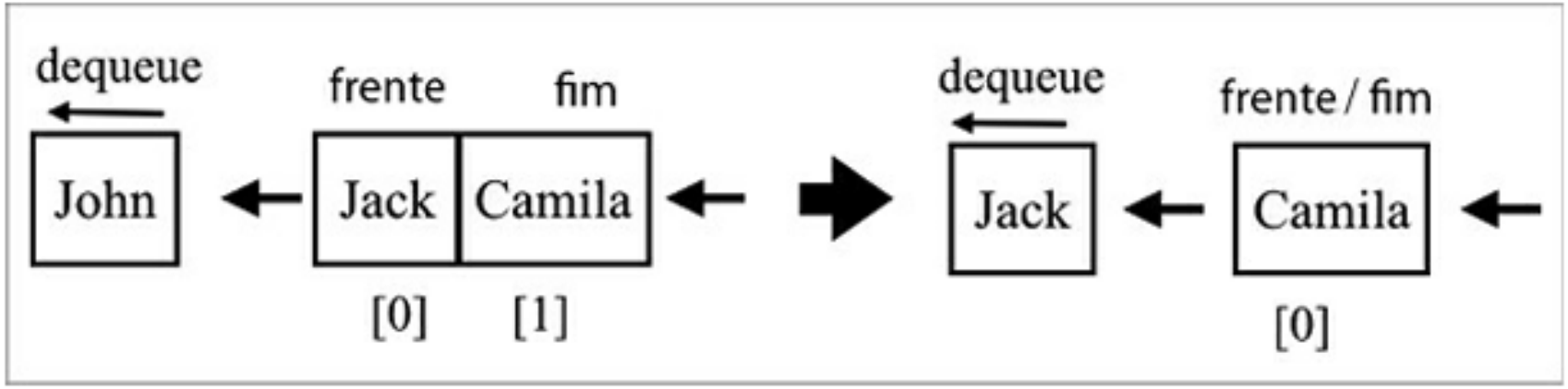


Figura 5.3

IMPLEMENTAÇÃO DO PROTÓTIPO DA CLASSE

Aula 05 - Filas - Fila_prototipo.js

```
1  class Fila
2  {
3      #items = [];
4      #inicio = 0;
5      #fim = 0;
6
7      enqueue(elemento) { }
8      dequeue() { }
9      front() { }
10     estaVazia() { }
11     tamanho() { }
12     limpar() { }
13 }
14
15 module.exports = Fila;
```

IMPLEMENTAÇÃO DOS ATRIBUTOS DA CLASSE

Aula 05 - Filas - Fila.js

```
1  class Fila
2  {
3      // Usamos uma lista para armazenar os itens
4      #items = [];
5
6      // Representa o índice do início da fila
7      #inicio = 0;
8
9      // Representa o índice do fim da fila
10     #fim = 0;
```

IMPLEMENTAÇÃO MÉTODO ENQUEUE

Aula 05 - Filas - Fila.js

```
12  // Adiciona um elemento ao final da fila (enqueue)
13  enqueue(elemento)
14  {
15      // Coloca o elemento no fim da fila
16      this.items[this.#fim] = elemento;
17
18      // Incrementa o índice do fim da fila
19      this.#fim++;
20  }
```

IMPLEMENTAÇÃO MÉTODO DEQUEUE

Aula 05 - Filas - Fila.js

```
22 // Remove e retorna o primeiro elemento da fila (dequeue)
23 dequeue()
24 {
25     // Se a fila estiver vazia, retorna undefined
26     if (this.estaVazia()) {
27         return undefined;
28     }
29
30     // Obtém o primeiro elemento
31     const item = this.#items[this.#inicio];
32
33     // Remove o item do início da fila
34     delete this.#items[this.#inicio];
35
36      // Move o índice do início para o próximo item
37     this.#inicio++;
38
39     // Quando o início e o fim estiverem alinhados, redefine a fila
40     if (this.#inicio === this.#fim) {
41         this.#inicio = 0;
42         this.#fim = 0;
43     }
44
45     return item;  // Retorna o item removido
46 }
```

IMPLEMENTAÇÃO MÉTODO FRONT

```
52  // Retorna o primeiro elemento da fila sem removê-lo (peek)
53  front()
54  {
55      // Se a fila estiver vazia, retorna undefined
56      if (this.estaVazia()) {
57          return undefined;
58      }
59      // Retorna o primeiro elemento
60      return this.#items[this.#inicio];
61  }
```

IMPLEMENTAÇÃO MÉTODO LIMPAR

Aula 05 - Filas - Fila.js

```
68  // Limpa a fila
69  limpar()
70  {
71      this.#items = {};
72      this.#inicio = 0;
73      this.#fim = 0;
74  }
```

IMPLEMENTAÇÃO DOS MÉTODOS ESTAVAZIA E TAMANHO

Aula 05 - Filas - Fila.js

```
60  // Verifica se a fila está vazia
61  // Verifica se os índices estão iguais
62  estaVazia = () => this.#fim === this.#inicio;
63
64  // Retorna o tamanho da fila
65  // Calcula a diferença entre fim e início
66  tamanho = () => this.#fim - this.#inicio;
```

COMO USAR A CLASSE?

Aula 05 - Filas - app.js

```
1  const Fila = require('./Fila.js');
2
3  const minha_variavel = new Fila();
4
5  minha_variavel.enqueue("Cliente 1");
6  minha_variavel.enqueue("Cliente 2");
7  minha_variavel.enqueue("Cliente 3");
8
9  console.log(minha_variavel.front());  // Saída: "Cliente 1"
10 console.log(minha_variavel.dequeue());  // Saída: "Cliente 1"
11 console.log(minha_variavel.dequeue());  // Saída: "Cliente 2"
12
13 minha_variavel.enqueue("Cliente 4");
14
15 // Saída: 2 (Cliente 3 e Cliente 4 ainda estão na fila)
16 console.log(minha_variavel.tamanho());
17 console.log(minha_variavel.front());  // Saída: "Cliente 3"
```

EXERCÍCIOS

Use a classe implementada acima para resolver os exercícios

EXERCÍCIO 1

- Crie um sistema de atendimento onde clientes entram na fila e são atendidos na ordem de chegada. A cada iteração, um cliente deve ser removido da fila e exibido no console. Novos clientes podem ser adicionados aleatoriamente.

- **Desafio Extra**: Simule diferentes tempos de atendimento para cada cliente, usando **`setTimeout()`**.

EXERCÍCIO 2

- Simule uma fila de impressão onde diferentes documentos são adicionados à fila. Cada documento deve ter um nome e um tamanho em páginas. Ao processar a fila, exiba no console qual documento está sendo impresso e remova-o da fila após a "impressão".

- **Desafio Extra**: Limite a capacidade da fila (ex: 5 documentos) e exiba uma mensagem quando a fila estiver cheia.

EXERCÍCIO 3

- Imagine um brinquedo que comporta apenas um passageiro por vez. Os visitantes entram na fila e são chamados um por um para embarcar. Crie um sistema que adicione visitantes à fila e simule o embarque e desembarque.

- **Desafio Extra**: Dê um tempo fixo para cada embarque usando **setTimeout()** e adicione um sistema de prioridade para visitantes VIP.

EXERCÍCIO 4

- Simule um restaurante onde os pedidos são adicionados a uma fila e preparados na ordem correta. Cada pedido deve ter um número e um nome (ex: "Pedido 1: Hambúrguer"). À medida que são preparados, os pedidos devem ser removidos da fila.

- **Desafio Extra**: Implemente um sistema onde os pedidos demoram tempos diferentes para serem preparados.

EXERCÍCIO 5

- Simule um caixa de supermercado onde os clientes entram na fila e passam pelo caixa um por um. Cada cliente pode ter um número aleatório de produtos (entre 1 e 10). O tempo de atendimento deve ser proporcional ao número de produtos.

- **Desafio Extra**: Crie várias filas e distribua os clientes entre elas para otimizar o tempo de atendimento.

OBRIGADO!

- Encontre este **material on-line** em:
  - Slides: Plataforma Microsoft Teams

- Em caso de **dúvidas**, entre em contato:
  - **Prof. Tiago**: tiago.silva238@fatec.sp.gov.br