

# DESAFIO... LÓGICA AFIADA

21  
dias

by CódigoFonte



## RESOLUÇÕES



DIA 13



## DESAFIO 01

```
● ● ●

const tarefas = [] // Deque para armazenar tarefas

// Inserir uma tarefa no início (alta prioridade)
function inserirInicio(tarefa) {
    tarefas.unshift(tarefa);
    console.log(`Tarefa '${tarefa}' adicionada com alta
prioridade.`);
}

// Inserir uma tarefa no final (baixa prioridade)
function inserirFim(tarefa) {
    tarefas.push(tarefa);
    console.log(`Tarefa '${tarefa}' adicionada com baixa
prioridade.`);
}

// Remover a tarefa de alta prioridade
function removerInicio() {
    if (!estaVazio()) {
        console.log(`Tarefa '${tarefas.shift()}' com alta prioridade
foi removida.`);
    } else {
        console.log("Não há tarefas para remover.");
    }
}

// Remover a tarefa de baixa prioridade
function removerFim() {
    if (!estaVazio()) {
        console.log(`Tarefa '${tarefas.pop()}' com baixa prioridade
foi removida.`);
    } else {
        console.log("Não há tarefas para remover.");
    }
}

// Verificar se o deque está vazio
function estaVazio() {
    return tarefas.length === 0;
}

// Obter tarefas
function obterTarefas() {
    return tarefas.slice(); // Retorna uma cópia do array usando o
método slice
}
```



```
// Verificar se o deque está vazio
function estaVazio() {
    return tarefas.length === 0;
}

// Obter tarefas
function obterTarefas() {
    return tarefas.slice(); // Retorna uma cópia do array usando o
método slice
}

// Aumentar a prioridade de uma tarefa
function aumentarPrioridade(tarefa) {
    let index = tarefas.indexOf(tarefa);
    if (index > 0) {
        let temp = tarefas[index - 1];
        tarefas[index - 1] = tarefas[index];
        tarefas[index] = temp;
        console.log(`Prioridade da tarefa '${tarefa}' foi
aumentada.`);
    } else {
        console.log("A tarefa já está com a máxima prioridade ou não
existe.");
    }
}

// Diminuir a prioridade de uma tarefa
function diminuirPrioridade(tarefa) {
    let index = tarefas.indexOf(tarefa);
    if (index !== -1 && index < tarefas.length - 1) {
        let temp = tarefas[index + 1];
        tarefas[index + 1] = tarefas[index];
        tarefas[index] = temp;
        console.log(`Prioridade da tarefa '${tarefa}' foi
diminuída.`);
    } else {
        console.log("A tarefa já está com a mínima prioridade ou não
existe.");
    }
}

// Demonstração do Gerenciamento de Tarefas
inserirFim("Comprar café");
inserirInicio("Responder e-mails");
inserirFim("Agendar reunião");

console.log("Tarefas atuais:", obterTarefas());

aumentarPrioridade("Comprar café"); // Tentativa de aumentar a
prioridade
diminuirPrioridade("Responder e-mails"); // Diminuir a prioridade

console.log("Tarefas após ajustes de prioridade:",
obterTarefas());

removerInicio(); // Remover a tarefa de maior prioridade
console.log("Tarefas após remoção:", obterTarefas());
```



Vamos começar com um array vazio, que chamamos de tarefas. Assim como já fizemos antes, será a estrutura responsável por armazenar os dados do Deque.

Para adicionar tarefas ao Deque iremos criar 2 funções: **inserirInicio** e **inserirFim**. Na primeira nós adicionamos a tarefa no array utilizando um método interno do JavaScript chamada **unshift** e na segunda **push**, como já usamos anteriormente.

Na verdade, todas essas funções estão aí para facilitar nossas vidas, mas como já mostramos no início do DLA como implementar elas, o entendimento já está contigo. Para remover uma tarefa, também vamos precisar de 2 funções: **removerInicio** e **removerFim**. Dessa vez utilizamos o método **shift** e na outra o **pop**.

Só lembrando que esses métodos além de realizar a tarefa, ele também retorna o próprio elemento, nesse caso uma tarefa. Então quando removemos, mesmo que o elemento não esteja mais no array, nós podemos identificar quem ele é.

Agora chegamos nas funções mais complexas desse desafio: **aumentarPrioridade** e **diminuirPrioridade**. Essas funções simplesmente alteram a posição de uma tarefa dentro do deque para refletir mudanças de urgência. Por exemplo, se uma tarefa se torna mais crítica, ela é movida para uma posição mais próxima do início; se for menos crítica, será movida para o fim.

Vamos entender como isso foi feito, na primeira linha nós precisamos encontrar o lugar da tarefa dentro do

array, ou seja, qual é o índice (a posição) dessa tarefa. O método `indexOf` no JavaScript faz exatamente isso. Ela vai item a item verificando cada posição do array até encontrar (ou não) a tarefa.

Nesse caso se ela encontrar, então o valor será positivo, na verdade precisa ser maior que 1 pois se ela estiver na posição 0 ela já está como prioritária. Para trocar ela de posição é possível fazer isso rapidamente no JavaScript utilizando os colchetes para dizer quais posições iremos atribuir ao mesmo tempo.

Nesse caso usamos a variável **index** onde temos a posição da tarefa “-1”. Assim garantimos que ela será posicionada numa posição mais prioritária e ao mesmo tempo a tarefa que estava na posição atual será atribuída com a tarefa que estava com maior prioridade.

A mesma lógica foi feita na **diminuirPrioridade** porém utilizando o atributo **length**, pois a ideia é posicionar a tarefa com menos prioridade no fim do array e trocá-la com a que está na última posição.

Continuando, temos a função **obterTarefas** que retorna o deque completo, porém no JavaScript é preciso utilizar o método **slice** para fazer isso pois assim ele retornará uma cópia na memória e assim não corremos o risco de alterar o deque original.

Pra testar finalmente essa implementação, adicionamos algumas tarefas ao deque para começar o dia: Comprar Café, Responder E-mails, Agendar reunião e resolver um Bug. Já de cara, mostramos as tarefas utilizando o **obterTarefas**.

Durante o dia tivemos uma mudança de prioridade e Agendar reunião se tornou prioritário e Responder E-mails perdeu prioridade. Chamando as funções que criamos elas vão executar essa tarefa pra gente. Mais uma vez podemos visualizar as tarefas para ver agora como ficaram com a mudança de prioridade.

Por fim, depois de resolver a tarefa de maior prioridade, podemos remover ela e depois ver como ficou agora nossa lista de tarefas.

Eu sei que deu um certo trabalho fazer isso tudo, mas é assim que a gente vai construindo algo grande, aos poucos. Não esqueça de baixar o código para comparar com a sua versão.

# DESAFIO... LÓGICA AFIADA

**21**  
dias

by CódigoFonte

[desafiologicaafiada.com.br](http://desafiologicaafiada.com.br)