ESTRUTURA DE DADOS

O que é estrutura de dados?

Uma maneira de organizar e ordenar informações como textos, números, booleanos, etc e registrá-los na memória do computador.

- organizar dados (informações) *textos, números, booleanos...
- como estão registrados na memória

array

[1, 2, 3] elementos 1,2,3

object

{name: 'fulano', idade: 20} elementos name: 'fulano', idade: 20

Gerenciando dados

Estrutura de dados tem a ver com a gestão das informações da aplicação.

Para esse gerenciamento, podemos dividir em 3 etapas:

- modelar a estrutura
- dar vida a estrutura(instanciar essa estrutura)
- criar as funcionalidades dessa estrutura. Ex: inserir, excluir, buscar, exibir, contar...

Arrays

Array, vetor ou arranjo, é uma estrutura amplamente utilizada e implementada em quase todas as linguagens de programação

Uma das estruturas mais básicas, simples de criar e utilizar.

['a', 10, 'd ', true] total de 4 elementos

Características

- acesso pelo index
- respeita a ordem de inserção dos elementos
- aceita valores duplicados
- dependendo do tamanho do arrays, para encontrar e/ou deletar um elemento, será necessário um uso maior de performance

Arrays no javascript

- são dinâmicos
- você poderá ter dados de diferentes tipos misturados dentro de um array.Strings, numbers, booleans, objetos, funções e até outros arrays
- Existem muitos métodos já implementados

push( ), pop( ), find( ), filter( ) entre outros

Array no código

Exemplo:

```
const pilotos = ['senna', 'prost', 'schumacher','hamilton']
a indexação (index) começa pelo 0
console.log(pilotos[0])
console.log(pilotos[3])
console.log(pilotos[1])
 acessar o tamanho do array
console.log(pilotos.length)
iterável
for (let piloto of pilotos){
    console.log(piloto)
}
```

adicionar elementos
pilotos.push('alonso')
console.log(pilotos)

encontrar um elemento
const senna = pilotos.find(piloto => piloto === 'senna')
console.log(senna)

deletar um elemento
pilotos.splice(1,1)
console.log(pilotos)

Matrix
Matriz ou vetor multidimensional
Significa que é um array, dentro de outro array.Poderemos ter muitos níveis.
Exemplo:
const students = [['leia', 35, 'SP'], ['clara', 25, 'MG'], ['bruna', 20, 'BA']]
console.log(students)
console.log(students[0])

Stack (tradução de stack é pilha)
Como uma pilha de livros
- linear, um após o outro
- o último a entrar na pilha é o primeiro a sair

Conceitos
- LIFO: Last In First Out
- O último elemento a entrar na pilha, aquele elemento do topo da pilha, é o primeiro a sair.

Stack no código
Métodos fundamentais
- push ( ): adicionar um elemento a pilha
- pop( ): remover o elemento do topo da pilha
- peek( ): obter o elemento do topo da pilha

Outros métodos poderão ser implementados como size( ) para mostrar o tamanho da pilha.
Exemplo:
passo 1 modelando

```
class Stack{
   constructor(){
            this.data =[]
            this.top = -1
   }

   push(value){
            this.top++
            this.data[this.top] = value
            return this.data

   }

   pop(){
            if(this.top < 0) return undefined
```

```
            const poppedTop = this.data[this.top]
        delete this.data[this.top]
        this.top--
        return poppedTop
        }

    peek(){
        return this.top >= 0 ? this.data[this.top]:
        undefined

    }
}
 passo 2 utilizando
const stack = new Stack()
adicionar dados
stack.push('learning')
stack.push('data')
console.log(stack.push('structures'))

console.log(stack.peek())
remover
stack.pop()
console.log(stack.pop())
console.log(stack.peek())
```

Queue (tradução "fila")
Como uma fila em uma loja ou restaurante

- linear
- o primeiro a entrar na fila é o primeiro a sair
- FIFO: first in first out

o primeiro elemento a entrar na fila, é o primeiro a sair dela

- FRONT (frente) é a referência do primeiro elemento a entrar na fila
- BACK (fundo) é a referência do último elemento a entrar na fila

Queue no código
Métodos fundamentais

- enqueue( ): adicionar um elemento ao final da fila
- dequeue( ): remover o primeiro elemento a entrar na fila

Outros métodos poderão ser implementados como size( ) para mostrar o tamanho da fila ou front( ) para pegar o primeiro elemento da fila, dentre tantos outros.
Exemplo:

```
 passo 1 modelando
class Queue{
    constructor(){
        this.data =[]
    }

    enqueue(item){
        this.data.push(item)
```

```
        console.log(`${item} chegou na fila!`)
    }

    dequeue(){
        const item = this.data.shift()
        console.log(`${item} saiu da fila!`)
    }
}
passo 2 utilizando
const fila = new Queue()
fila.enqueue('Mariana')
fila.enqueue('Joao')
fila.enqueue('Ariel')
fila.dequeue()
fila.dequeue()
fila.dequeue()
```