Stack Data Structure - Detailed Guide

This document explains the Stack data structure in detail.

Stack is a linear data structure that follows the Last In First Out (LIFO) principle.

We will cover Stack concepts, JavaScript examples, and practical use cases.

Prepared for Tanveer.

1. What is Stack?

A Stack is a linear data structure that follows the LIFO (Last In First Out) principle.

The last element inserted in the stack is the first one to be removed.

Common examples include a stack of plates or browser history.

2. Basic Operations of Stack

Main operations in a stack are:

- push(element): Adds an element to the top of the stack.

- pop(): Removes the element from the top of the stack.

- peek(): Returns the top element without removing it.

- isEmpty(): Checks whether the stack is empty.

3. Using JavaScript Array as Stack

In JavaScript, arrays can be used to implement stacks using push() and pop() methods.

Example:

```
const stack = [];
stack.push(10);
stack.push(20);
console.log(stack.pop());  // 20
console.log(stack[stack.length - 1]);  // 10 (peek)
```

4. Example Walkthrough

Let's add multiple elements and remove some to see how the stack changes.

Example:

```
const stack = [];
stack.push(10);
stack.push(20);
stack.push(30);
stack.pop(); // removes 30
stack.push(40);
console.log(stack); // [10, 20, 40]
```

5. Custom Stack Class in JavaScript

We can also create a Stack class with its own methods for better abstraction.

Example:

```
class Stack {
  constructor() {
    this.items = [];
  }
  push(element) {
    this.items.push(element);
  }
  pop() {
    if(this.isEmpty()) return 'Stack is empty';
    return this.items.pop();
  }
  peek() {
    return this.items[this.items.length - 1];
  }
  isEmpty() {
    return this.items.length === 0;
  }
}
```

6. Practical Problems Using Stack

Stack Data Structure - Detailed Guide

Some common problems where stacks are useful:

- Reverse a string

- Check for balanced parentheses in expressions

- Implement undo functionality

Example: Reverse a string

Push all characters, then pop them to get reversed string.

```
function reverseString(str) {
  const stack = [];
  for(let char of str) {
    stack.push(char);
  }
  let reversed = '';
  while(stack.length > 0) {
    reversed += stack.pop();
  }
  return reversed;
}
console.log(reverseString('Tanveer')); // reevnaT
```