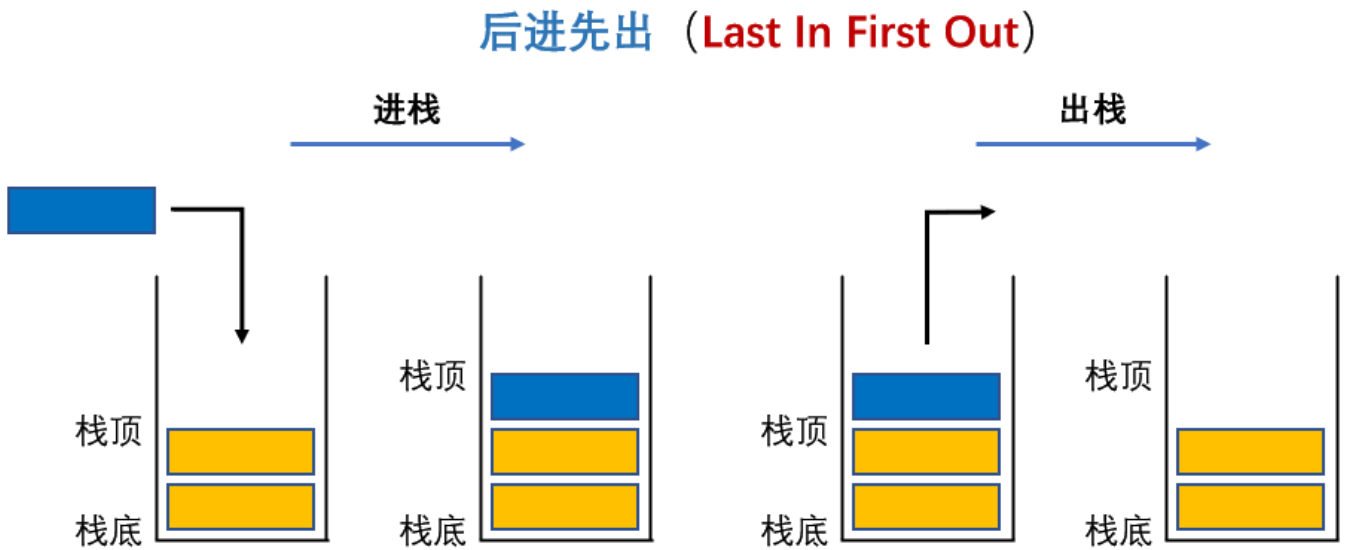
栈结构（Stack）

王红元 coderwhy

目录
content

认识栈结构

- 栈也是一种 非常常见 的数据结构， 并且在程序中的 应用非常广泛。
- 数组
  - 我们知道数组是一种线性结构， 并且可以在数组的 任意位置 插入和删除数据。
  - 但是有时候， 我们为了实现某些功能， 必须对这种任意性 加以 限制。
  - 而 栈和队列 就是比较常见的 受限的线性结构， 我们先来学习栈结构。
- 栈结构示意图



认识栈结构

- **栈（stack），它是一种受限的线性结构，后进先出(LIFO)**
  - 其限制是仅允许在 表的一端 进行插入和删除运算。这一端被称为栈顶，相对地，把另一端称为栈底。
  - LIFO(last in first out)表示就是后进入的元素， 第一个弹出栈空间。 类似于自动餐托盘， 最后放上的托盘， 往往先把拿出去使用。
  - 向一个栈插入新元素又称作进栈、入栈或压栈，它是把新元素放到栈顶元素的上面，使之成为新的栈顶元素；
  - 从一个栈删除元素又称作出栈或退栈，它是把栈顶元素删除掉，使其相邻的元素成为新的栈顶元素。
- **生活中类似于栈的**
  - 自助餐的托盘， 最新放上去的， 最先被客人拿走使用。
  - 收到很多的邮件(实体的)， 从上往下依次处理这些邮件。 (最新到的邮件， 最先处理)
  - 注意: 不允许改变邮件的次序， 比如从最小开始， 或者处于最紧急的邮件， 否则就不再是栈结构了。 而是队列或者优先级队列结构。

栈结构面试题

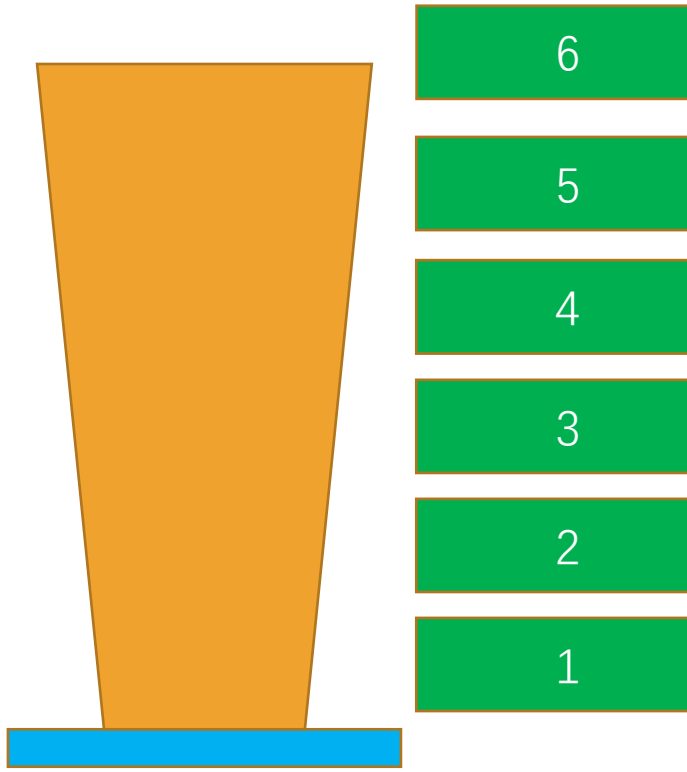
- **面试题目:**

题目

有六个元素6,5,4,3,2,1 的顺序进栈,问下列哪一个不是合法的出栈序列?( )

A. 5 4 3 6 1 2 B. 4 5 3 2 1 6 C. 3 4 6 5 2 1 D. 2 3 4 1 5 6

- **题目答案: C**
  - A答案: 65进栈， 5出栈， 4进栈出栈， 3进栈出栈， 6出栈， 21进栈，1出栈， 2出栈
  - B答案: 654进栈， 4出栈， 5出栈， 3进栈出栈， 2进栈出栈， 1进栈出栈， 6出栈
  - D答案: 65432进栈， 2出栈， 3出栈， 4出栈， 1进栈出栈， 5出栈， 6出栈



栈结构的实现

- **实现栈结构有两种比较常见的方式:**
  - 基于数组实现
  - 基于链表实现
- **什么是链表?**
  - 也是一种数据结构，目前我们还没有学习，并且JavaScript中并没有自带链表结构。
  - 后续，我们会自己来实现链表结构，并且对比数组和链表的区别。
- **因此，我们这里实现的栈结构基于数组。**

创建栈的类

- 我们先来创建一个栈的类，用于封装栈相关的操作

```
class Stack<T> {
  private data: T[] = [];
}
```

- 代码解析:
  - 我们创建了一个Stack，用户创建栈的类，可以定义一个泛型类。
  - 在构造函数中，定义了一个变量，这个变量可以用于保存当前栈对象中所有的元素。
  - 这个变量是一个数组类型。
  - 我们之后无论是压栈操作还是出栈操作，都是从数组中添加和删除元素。
  - 栈有一些相关的操作方法，通常无论是什么语言，操作都是比较类似的。

栈的操作

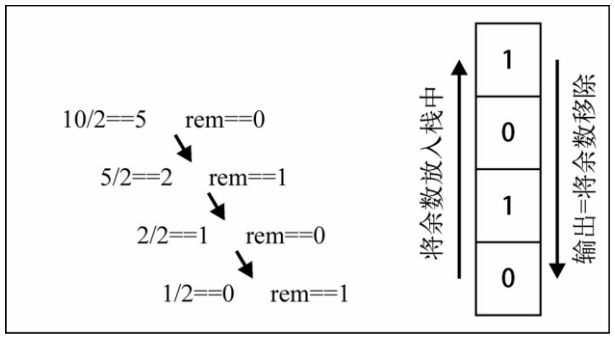
- **栈常见有哪些操作呢?**
  - push(element): 添加一个新元素到栈顶位置。
  - pop(): 移除栈顶的元素，同时返回被移除的元素。
  - peek(): 返回栈顶的元素，不对栈做任何修改（这个方法不会移除栈顶的元素，仅仅返回它）。
  - isEmpty(): 如果栈里没有任何元素就返回true，否则返回false。
  - size(): 返回栈里的元素个数。这个方法和数组的length属性很类似。

- **现在，我们可以在类中一一实现这些方法。**

```ts
class Stack<T> {
  private data: T[] = [];

  push(item: T): void {
    this.data.push(item);
  }

  pop(): T | undefined {
    return this.data.pop();
  }

  peek(): T | undefined {
    return this.data[this.data.length - 1];
  }

  isEmpty(): boolean {
    return this.data.length === 0;
  }

  clear(): void {
    this.data = []
  }
}
```

十进制转二进制（面试题）

- 我们已经学会了如何使用Stack类，现在就用它解决一些计算机科学中的问题。
- 为什么需要十进制转二进制?
  - 现实生活中，我们主要使用十进制。
  - 但在计算科学中，二进制非常重要，因为计算机里的所有内容都是用二进制数字表示的（0和1）。
  - 没有十进制和二进制相互转化的能力，与计算机交流就很困难。
  - 转换二进制是计算机科学和编程领域中经常使用的算法。
- 如何实现十进制转二进制?
  - 要把十进制转化成二进制，我们可以将该十进制数字和2整除（二进制是满二进一），直到结果是0为止。
  - 举个例子，把十进制的数字10转化成二进制的数字，过程大概是这样：




```
import Stack from "./Stack";

function decimalToBinary(decimal: number): string {
  const stack = new Stack();
  let remainder;
  let binary = "";

  while (decimal > 0) {
    remainder = decimal % 2;
    stack.push(remainder);
    decimal = Math.floor(decimal / 2);
  }
  while (!stack.isEmpty()) {
    binary += stack.pop()!.toString();
  }
  return binary;
}

console.log(decimalToBinary(100))

export {}
```

有效的括号 – 字节、华为等面试题

- 国内字节、华为、京东都考过的面试题。
- 给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串 s，判断字符串是否有效。
  - Leetcode 20：https://leetcode.cn/problems/valid-parentheses/description
- 有效字符串需满足：
  1. 左括号必须用相同类型的右括号闭合。
  2. 左括号必须以正确的顺序闭合。
  3. 每个右括号都有一个对应的相同类型的左括号。

示例 1：

```
输入：s = "()"
输出：true
```

示例 2：

```
输入：s = "()[]{}"
输出：true
```

示例 3：

```
输入：s = "(]"
输出：false
```


```typescript
function isValid(s: string): boolean {
  const stack = new Stack<string>()
  const length = s.length
  for (let i = 0; i < length; i++) {
    const x: string = s[i];
    switch (x) {
      case "(":
        stack.push(")");
        break;
      case "[":
        stack.push("]");
        break;
      case "{":
        stack.push("}");
        break;
      default:
        if (stack.pop() !== x) return false;
        break;
    }
  }
  return stack.isEmpty();
}
```