Javascript数据结构

栈

Skipper

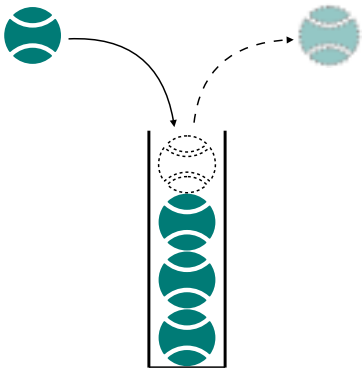
红点工场
redpoint.live

大纲

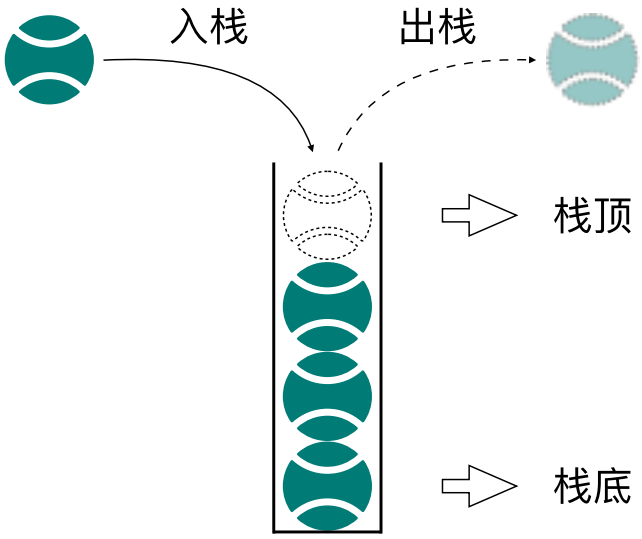
栈概念、结构、操作
栈操作实现
栈实战：十进制转2进制

什么是栈?

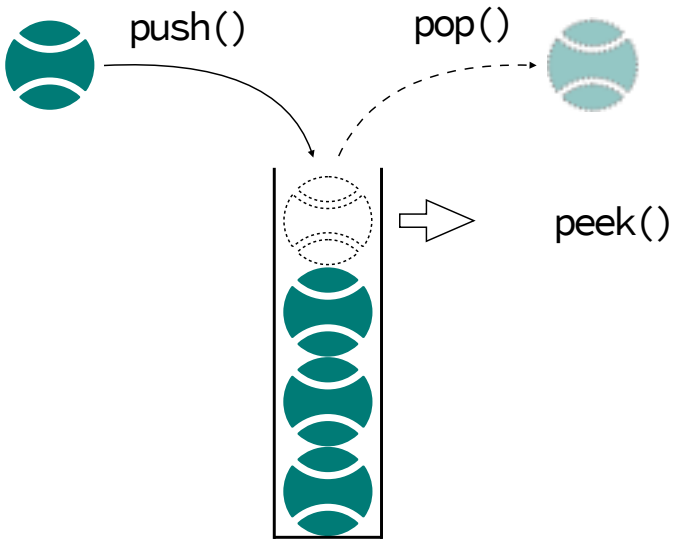
一种先进先出（LIFO）的数据结构



栈结构概念

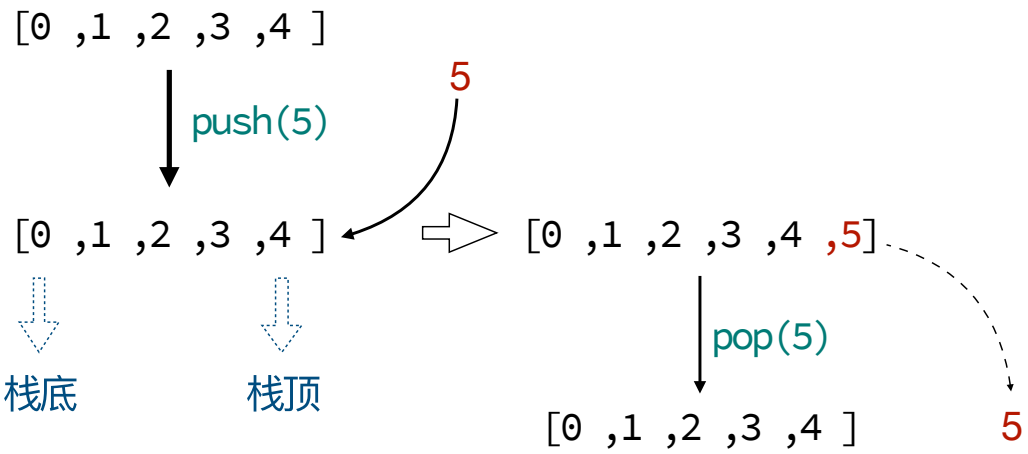


栈结构操作



isEmpty()	是否为空?
clear()	清空栈
size()	栈元素个数

js实现栈结构——数组



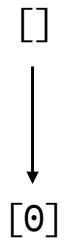
使用类封装栈操作

```
var Stack = function(){
    var items = [];
}
```

方法实现

```
this.push()
this.pop()
this.peek()
this.isEmpty()
this.clear()
this.size()
```

向栈顶添加元素：push




```
this.push = function(element){
    items.push(element);
}
```

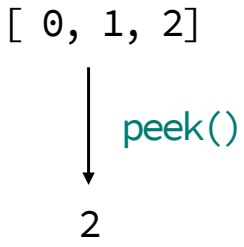
*使用数组的push方法添加元素到数组末端

向栈顶移除元素：pop


```
this.pop = function(){
    return items.pop();
}
```

*使用数组的pop方法：删除数组最后元素

查看栈顶元素：peek




```
this.peek = function(){
    return items[length - 1];
}
```

```
items.length = 3 ,
items.length - 1 = 2 ,
items[2] = 2
```

其他操作

检查栈是否为空：isEmpty()

```
this.isEmpty = function(){
    return items.length == 0;
}
```

*使用布尔型运算判断数组长度是否为0

清除栈：clear()

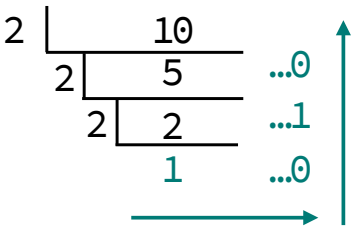
```
this.clear = function(){
    items = [];
}
```

获取栈大小：size()

```
this.size = function(){
    return items.length;
}
```

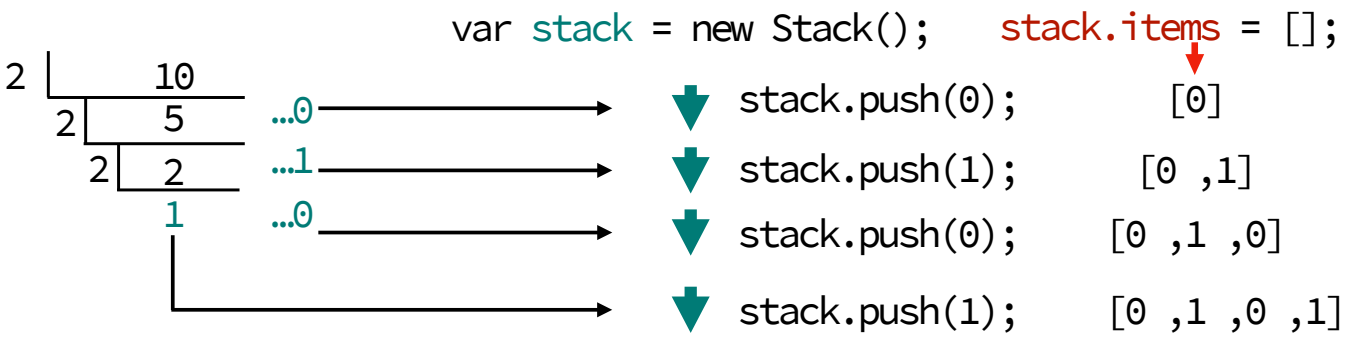
栈实例：十进制到二进制转换

进制转换算法：余数法

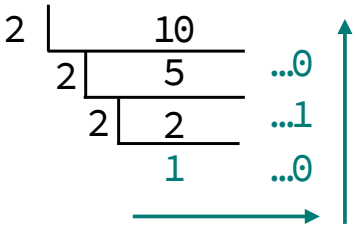


$(10)_{10} = (1010)_2$

使用栈实现余数法：入栈



出栈




```
stack.items = [0, 1, 0, 1];
```

循环执行 stack.pop()

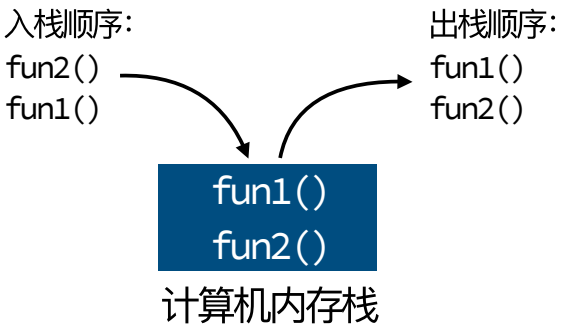
“1010”

栈的一些思考

栈作用：在编程语言的编译器和内存中保存变量、方法调用

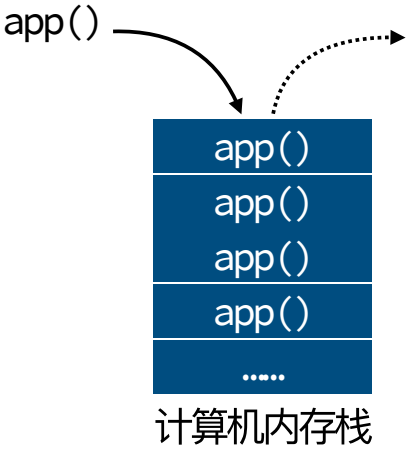
栈和函数

```
var fun1 = function(){
    return console.log('fun1 finish');
}
var fun2 = function(){
    fun1();
    return console.log('fun2 finish')
}
fun2();
```



递归

如果不停递归而不出栈，
会导致栈溢出



栈是一个基本的计算机数据结构
是高级编程语言的实现基础