1: 闭包

> 定义在函数内部的函数 ,内部的函数可以访问外部函数的作用域,参数和变量不会被垃圾回收机制回收

优点:

```
1.保护函数内的变量安全,加强了封装性
2.防止变量污染
```

缺点:

```
1.内存泄漏
2.空间浪费
3.消耗性能
```

2:arguments是什么？arguments是数组还是对象？如果是对象请问怎么将arguments转换数组?

> JavaScript 中每个函数内都能访问一个特别变量 arguments。这个变量维护着所有传递到这个函数中的参数列表。

arguments 变量不是一个数组（Array）。 尽管在语法上它有数组相关的属性 length，但它不从 Array.prototype 继承，实际上它是一个对象（Object）。

```
Array.prototype.slice.call(arguments);
```

3:什么是面向对象?

1. **“面向对象”是专指在程序设计中采用封装、继承、多态等设计方法**

2. **面向对象也是一种对现实世界理解和抽象的方法**

> （背会第一条和第二条任意一条的即可）

4:什么是原型？什么是原型链?

> **原型也是一个对象，通过原型可以实现对象的属性继承，JavaScript 的构造函数中都包含了一个 prototype 内部属性，这个属性所对应的就是该对象的原型。**

> **原型链的基本思想是利用原型让一个引用类型继承另一个引用类型的属性和方法。**

5: call 和 apply 还有bind

共同点：

- apply 、 call 、bind 三者都是用来改变函数的this对象的指向的
- apply 、 call 、bind 三者第一个参数都是this要指向的对象，也就是想指定的上下文
- apply 、 call 、bind 三者都可以利用后续参数传参；

不同点:

- apply接受的是数组参数
- call接受的是连续参数。
- bind 是返回对应函数，便于稍后调用；apply 、call 则是立即调用

```
var obj = {
    x: 81,
};

var foo = {
    getX: function() {
        return this.x;
```

```
    }
}

console.log(foo.getX.bind(obj)());  //81
console.log(foo.getX.call(obj));    //81
console.log(foo.getX.apply(obj));   //81
```

三个输出的都是81，但是注意看使用 bind() 方法的，他后面多了对括号。
也就是说，区别是，当你希望改变上下文环境之后并非立即执行，而是回调执行的时候，使用 bind() 方法。而 apply/call 则会立即执行函数。

6：js中数组和对象浅拷贝和深拷贝?

1:浅拷贝

> 浅拷贝只是单纯的将引用地址赋值给这个元素 2个引用类型关联性一致

```
//数组的浅拷贝
var arr = ["One","Two","Three"];
//通过直接引用类型直接赋值 他们2个指向的都是同一个地址
var arrToo = arr;



//对象的浅拷贝
var obj = { class:'95班',personNumber:'46人'};
//通过直接引用类型直接赋值 他们2个指向的都是同一个地址
var objToo = obj;
```

2:深拷贝

> 深拷贝是开辟一个新的引用空间地址。深拷贝出来的引用类型和原来没有关联

```
//数组的深拷贝--方法1
var arr = ["One","Two","Three"];
//数组中 slice方法会根据传递进去的参数返回一个新的数组  这个新数组和原来数组的引用地址不同
var arrtoo = arr.slice(0);

//数组的深拷贝--方法2 原理类似方法1
```

```
var arr = ["One","Two","Three"];
var arrtooo = arr.concat();

//简单的对象深拷贝
var obj = { class:'95班',personNumber:'46人'};
var objToo = {};
objToo.class = obj.class;
objToo.personNumber = obj.class;

/*
*通过JSON方法深拷贝
* 存在问题?
*1:无法复制函数
*2:原型链没了，对象就是object，所属的类没了。
*/

var obj = { class:'95班',personNumber:'46人'};
var objToo = JSON.parse(JSON.stringify(obj));
```

7:什么是构造函数？与普通函数有什么区别?

> 构造函数：是一种特殊的方法、主要用来创建对象时初始化对象，总与new运算符一起使用，创建对象的语句中构造函数的函数名必须与类名完全相同。

区别

- 构造函数首字母必须大写,普通函数则不需要
- 构造函数使用需要配合new操作符,普通函数则不需要
- 所谓“构造函数”，就是专门用来生成“对象”的函数。它提供模板，作为对象的基本结构。一个构造函数，可以生成多个对象，这些对象都有相同的结构。