1. 已知函数A，要求构造一个函数B 继承A。

```
function A (name) {
    this.name = name;
}
A.prototype.getName = function() {
    console.log(this.name)
}
```

2. 数组和对象转换为字符串结果

```
var arry = [];
var obj = {};
// arry,obj 转成字符串的结果是什么?
```

3.

用几种方式实现DOM元素水平垂直居中

4. 实现一个深复制的函数extend ()

```
function extend(source) {

}
```

5. 分别写出打印的内容

```
var a = {
    name: 'A',
    fn() {
        console.log(this.name)
    }
}
a.fn()
a.fn.call({name: 'B'})
var fn1 = a.fn;
fn1()
```

6. 分别写出来打印的内容

```
    let int = 1;
setTimeout(function(){
    console.log(int)
    int = 2;
    new Promise((resolve, reject) => {
        resolve()
    }).then(function(){
        console.log(int)
        int = 7
    })
    console.log(int)
})
int = 3;
console.log(int)
new Promise((resolve, reject) => {
    console.log(int)
    return resolve(int = 4)

}).then(function(res){
    console.log(int)
    int = 5
    setTimeout(function(){
        console.log(int)
        int = 8;
    })
    return false
})
console.log(int)
```

7.

已知对象A = {name: 'sfd', getName: function(){console.log(this.name)}},现要求用不同方式对A进行改造实现A.name 发生变化时立即执行A.getName

8. 修改以下代码，使得最后一行代码能够输出数字0-9（最好能给多种答案）

```
var arrys = [];
for (var i = 0; i < 10; i++) {
  arrys.push(function(){return i;})
}
arrys.forEach(function (fn) {console.log(fn())}) //本行不能修改
```

9.

给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效.

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。
   注意空字符串可被认为是有效字符串。
   示例1：
   输入：`"()"`
   输出：`true`
   示例2：
   输入：`"()[]{}"`
   输出：`true`
   示例 3：
   输入：`"(]"`
   输出：`false`
   示例 4：
   输入：`"([)]"`
   输出：`false`
   示例 5：
   输入：`"{[]}"`
   输出：`true`