# 代码面试题

## 1. bilibili 面试

不考虑兼容性且不能更改dom结构，需求如下：

1.完成经典的上 header ，下 footer ，左边是侧边栏，右边是内容。

2.去掉列表前面的 · ，列表项水平排列，注意里面的br标签需要换行，同时每两个li后有一条竖线。

3.点击列表项不跳转，弹出href内的内容

来源 题目 答案

## 2. 用setTimeout实现setInterval

```
function mySetInterval(fn, millisec,count){
  function interval(){
    if(typeof count===‘undefined’||count-->0){
      setTimeout(interval, millisec);
      try{
        fn()
      }catch(e){
        t = 0;
        throw e.toString();
      }
    }
  }
  setTimeout(interval, millisec)
}
```

参考：

用setTimeout实现setInterval

## 3. call模拟实现

```
Function.prototype.call2 = function (context) {
    var context = context || window;
    context.fn = this;
    var args = [];
    for(var i = 1, len = arguments.length; i < len; i++) {
        args.push('arguments[' + i + ']');
    }
    var result = eval('context.fn(' + args +')');
    delete context.fn
    return result;
}
```

## 4. apply模拟实现

```
Function.prototype.apply = function (context, arr) {
    var context = Object(context) || window;
    context.fn = this;
    var result;
    if (!arr) {
        result = context.fn();
    }
    else {
        var args = [];
        for (var i = 0, len = arr.length; i < len; i++) {
            args.push('arr[' + i + ']');
        }
        result = eval('context.fn(' + args + ')')
    }
    delete context.fn
    return result;
}
```

## 5. bind模拟实现

```
Function.prototype.bind2 = function (context) {
```

```
    if (typeof this !== "function") {
      throw new Error("Function.prototype.bind - what is trying to be bound is not callable");
    }
    var self = this;
    var args = Array.prototype.slice.call(arguments, 1);
    var fNOP = function () {};
    var fBound = function () {
        var bindArgs = Array.prototype.slice.call(arguments);
        return self.apply(this instanceof fNOP ? this : context, args.concat(bindArgs));
    }
    fNOP.prototype = this.prototype;
    fBound.prototype = new fNOP();
    return fBound;
}
```

## 6. new模拟实现

```
function create() {
    // 创建一个空的对象
    var obj = new Object(),
    // 获得构造函数，arguments中去除第一个参数
    Con = [].shift.call(arguments);
    // 链接到原型，obj 可以访问到构造函数原型中的属性
    obj.__proto__ = Con.prototype;
    // 绑定 this 实现继承，obj 可以访问到构造函数中的属性
    var ret = Con.apply(obj, arguments);
    // 优先返回构造函数返回的对象
    return ret instanceof Object ? ret : obj;
};
```

## 7. 两个升序数组合并成一个升序数组

```
function merge(left, right){
    let result  = [],
        il      = 0,
        ir      = 0;
    while (il < left.length && ir < right.length) {
        result.push(left[il] < right[ir] ? left[il++] : right[ir++]);
    }
    return result.concat(left.slice(il)).concat(right.slice(ir));
}
```

## 8. 数组去重

```
//reduce方法
const distinct = arr => arr.sort().reduce( (init, current) => {
    if (init.length === 0 || init[init.length - 1] !== current) {
        init.push( current );
    }
    return init;
}, []);
//过滤方法
const distinct = arr => arr.filter( (element, index, self) => {
    return self.indexOf( element ) === index;
});
```

## 9. 多重数组降维

```
const flattenDeep = arr => Array.isArray(arr)
  ? arr.reduce( (a, b) => [...a, ...flattenDeep(b)] , [])
  : [arr]
```

## 10. 给定两个数组，写一个方法来计算它们的交集。

```
var intersect = function(nums1, nums2) {
    var map1 = new Map();
    var number = [];
    for(var i = 0; i < nums1.length; i++) {
        var map1Value = map1.get(nums1[i]);
        map1.set( nums1[i], ( map1Value ? map1Value : 0 ) + 1 );
    }
    for(var i = 0; i < nums2.length; i++) {
        if( map1.has(nums2[i]) && map1.get(nums2[i]) != 0 ) {
            number.push(nums2[i]);
```

```
            map1.set( nums2[i], map1.get(nums2[i]) - 1 );
        }
    }
    return number;
};
```

## 11.promise实现ajax

```
//promise 实现ajax
function ajax(method, url, data) {
    var request = new XMLHttpRequest();
    return new Promise(function (resolve, reject) {
        request.onreadystatechange = function () {
            if (request.readyState === 4) {
                if (request.status === 200) {
                    resolve(request.responseText);
                } else {
                    reject(request.status);
                }
            }
        };
        request.open(method, url);
        request.send(data);
    });
}
var p = ajax('GET', '/api/categories');
p.then(function (text) {   // 如果AJAX成功，获得响应内容
    log.innerText = text;
}).catch(function (status) { // 如果AJAX失败，获得响应代码
    log.innerText = 'ERROR: ' + status;
});
```

12. 以下代码，使用三种方法，实现依次输出0-9

```
//使用立即执行函数（）（）
for (var i = 0; i < 10; i++) {
  funcs.push((function (value) {
    return function () {
      console.log(value)
    }
  }(i)))
}
//使用闭包
function show(i) {
  return function () {
    console.log(i)
  }
}
for (var i = 0; i < 10; i++) {
  funcs.push(show(i))
}
//使用let
for (let i = 0; i < 10; i++) {
  funcs.push(function () {
    console.log(i)
  })
}
```