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;
};
```