JS编程实战99道面试题（https://github.com/minsion/）

面试题 1. 简述请写出如下代码的打印结果？

```
function Foo() {
Foo.a = function() {
console.log(1);
};
this.a = function() {
console.log(2);
};
}
Foo.prototype.a = function() {
console.log(3);
};
Foo.a = function() {
console.log(4);
};
Foo.a();
let obj = new Foo();
obj.a();
Foo.a();
```

推荐指数：★★★ 试题难度：中级 试题类型：编程题

试题回答参考思路：

4 2 1

面试题 2. 简述下面这个 ul，如何点击每一列的时候 alert 其 index？（闭包？

- 这是第一条
- 这是第二条
- 这是第三条

推荐指数：★★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

```
// 方法一：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++)
{
lis[i].index=i;
lis[i].onclick=function(){
alert(this.index);
};
```

```
}
//方法二：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++){
lis[i].index=i;
lis[i].onclick=(function(a){
return function() {
alert(a);
}
})(i);
}
```

面试题 3. 编写一个 JavaScript 函数实现以下逻辑？

输入指定类型的选择器(仅需支持 id，class，tagName 三种简单 CSS 选择器，无需兼容组合选择器)可以返回匹配的 DOM 节点，需考虑浏览器兼容性和性能

推荐指数：★★★★ 试题难度： 中级 试题类型： 编程题

试题回答参考思路：

```
/*** @param selector {String} 传入的 CSS 选择器。* @return {Array}*/
var query = function(selector) {
var reg = /^(#)?(\.)?(\w+)$/img;
var regResult = reg.exec(selector);
var result = [];
//如果是 id 选择器
if(regResult[1]) {
if(regResult[3]) {
if(typeof document.querySelector === "function") {
result.push(document.querySelector(regResult[3]));
}else {
result.push(document.getElementById(regResult[3]));
}
}
}
//如果是 class 选择器
else if(regResult[2]) {
if(regResult[3]) {
if(typeof document.getElementsByClassName === 'function') {
var doms = document.getElementsByClassName(regResult[3]);
if(doms) {
result = converToArray(doms);
}
}
//如果不支持 getElementsByClassName 函数
else {
var allDoms = document.getElementsByTagName("*") ;
```

```
for(var i = 0, len = allDoms.length; i < len; i++) {
if(allDoms[i].className.search(new RegExp(regResult[2])) > -1) {
result.push(allDoms[i]);
}
}
}
}
}
//如果是标签选择器
else if(regResult[3]) {
var doms = document.getElementsByTagName(regResult[3].toLowerCase());
if(doms) {
result = converToArray(doms);
}
}
return result;
}
function converToArray(nodes){
var array = null;
try{
array = Array.prototype.slice.call(nodes,0);//针对非 IE 浏览器
}catch(ex){
array = new Array();
for( var i = 0 ,len = nodes.length; i < len ; i++ ) {
array.push(nodes[i])
}
}
return array;
}
```

面试题 4. 简述请评价以下代码并给出改进意见？

```
if(window.addEventListener){
var addListener = function(el,type,listener,useCapture){
el.addEventListener(type,listener,useCapture);
};
}
else if(document.all){
addListener = function(el,type,listener){
el.attachEvent("on"+type,function(){
listener.apply(el);
});
}
}
```

推荐指数：★★★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

1不应该在 if 和 else 语句中声明 addListener 函数，应该先声明；
2不需要使用 window.addEventListener 或 document.all 来进行检测浏览器，应该使用能力检测；
3由于 attachEvent 在 IE 中有 this 指向问题，所以调用它时需要处理一下
改进如下：

```
function addEvent(elem, type, handler){
if(elem.addEventListener){
elem.addEventListener(type, handler, false);
}else if(elem.attachEvent){
elem['temp' + type + handler] = handler;
elem[type + handler] = function(){
elem['temp' + type + handler].apply(elem);
};
elem.attachEvent('on' + type, elem[type + handler]);
}else{
elem['on' + type] = handler;
}
}
```

面试题 5. 简述定义一个 log 方法，让它可以代理 console.log 的方法？

推荐指数：★★★★★ 试题难度：中级 试题类型：编程题

试题回答参考思路：

可行的方法一：

```
function log(msg) {
console.log(msg);
}
log("hello world!") // hello world!
```

如果要传入多个参数呢？显然上面的方法不能满足要求，所以更好的方法是：

```
function log(){
console.log.apply(console, arguments);
};
```

到此，追问 apply 和 call 方法的异同。
对于 apply 和 call 两者在作用上是相同的，即是调用一个对象的一个方法，以另一个对象替换当前对象。将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。但两者在参数上有区别的。对于第一个参数意义都一样，但对第二个参数： apply 传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而 call 则作为 call 的参数传入（从第二个参数开始）。 如 func.call(func1,var1,var2,var3)对应的 apply 写法为：
func.apply(func1,[var1,var2,var3])

面试题 6. 简述说出以下函数的作用是？空白区域应该填写什么？

```
//define
(function(window){
function fn(str){
this.str=str;
```

```
}
fn.prototype.format = function(){
var arg = ______;
return this.str.replace(_____,function(a,b){
return arg[b]||"";
});
}
window.fn = fn;
})(window);
//use
(function(){
var t = new fn('

{1}{2}

');
console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));
})();
```

推荐指数： ★★ 试题难度： 初级 试题类型： 编程题

试题回答参考思路：

答案：访函数的作用是使用 format 函数将函数的参数替换掉{0}这样的内容，返回一个格式
化后的结果：
第一个空是：arguments
第二个空是：/\{(\d+)\}/ig

面试题 7. 简述 foo = foo||bar ，这行代码是什么意思？为什么要这样写 ？

推荐指数： ★★★★★ 试题难度： 初级 试题类型： 编程题

试题回答参考思路：

这种写法称之为短路表达式
答案：if(!foo) foo = bar; //如果 foo 存在，值不变，否则把 bar 的值赋给 foo。
短路表达式：作为”&&”和”||”操作符的操作数表达式，这些表达式在进行求值时，只要最终的结果已经可以确定是真或假，求值过程便告终止，这称之为短路求值。
注意 if 条件的真假判定，记住以下是 false 的情况：
空字符串、false、undefined、null、0

面试题 8. 简述看下列代码，将会输出什么 ？

```
var foo = 1;
function(){
console.log(foo);
var foo = 2;
console.log(foo);
}
```

推荐指数： ★★ 试题难度： 初级 试题类型： 编程题

试题回答参考思路：

输出 undefined 和 2。上面代码相当于：

```
var foo = 1;
function(){
var foo;
console.log(foo); //undefined
foo = 2;
console.log(foo); // 2;
}
```

函数声明与变量声明会被 JavaScript 引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分

面试题 9. 简述用 js 实现随机选取 10—100 之间的 10 个数字，存入一个数组，并排序？

推荐指数：★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

```
var iArray = [];
funtion getRandom(istart, iend){
var iChoice = istart – iend +1;
return Math.floor(Math.random() * iChoice + istart;
}
Math.random()就是获取 0–1 之间的随机数（永远获取不到 1）
for(var i=0; i<10; i++){
var result= getRandom(10,100);
iArray.push(result);
}
iArray.sort();
```

面试题 10. 简述把两个数组合并，并删除第二个元素？

推荐指数：★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

```
var array1 = ['a','b','c'];
var bArray = ['d','e','f'];
var cArray = array1.concat(bArray);
cArray.splice(1,1);
```

面试题 11. 正则表达式构造函数 var reg=new RegExp("xxx")与正则表达字面量 var reg=//有什么不同？匹配邮箱的正则表达式？

推荐指数：★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

当使用 RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。
邮箱的正则匹配：

```
var regMail = /^([a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+((.[a-zA-Z0-9_-]{2,3}){1,2})$/;
```

24.看下面代码，给出输出结果。

```
for(var i=1;i<=3;i++){
setTimeout(function(){
console.log(i);
},0);
};
```

答案：4 4 4。
原因：Javascript 事件处理器在线程空闲之前不会运行。追问，如何让上述代码输出 1 2 3?

```
for(var i=1;i<=3;i++){
setTimeout((function(a){ //改成立即执行函数
console.log(a);
})(i),0);
};
1 //输出
2
3
```

面试题 12. 简述写一个 function，清除字符串前后的空格。（兼容所有浏览器）？

推荐指数：★★★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

使用自带接口 trim()，考虑兼容性：

```
if (!String.prototype.trim) {
String.prototype.trim = function() {
return this.replace(/^\s+/, "").replace(/\s+$/,"");
//\s 匹配空白字符：回车、换行、制表符 tab 空格
}
}
// test the function
var str = " \t\n test string ".trim();
alert(str == "test string"); // alerts "true"
```

面试题 13. 简述以下两个变量 a 和 b，a+b 的哪个结果是 NaN？

A、var a=undefind; b=NaN //拼写
B、var a=‘123’; b=NaN//字符串

C、var a =undefined , b =NaN
D、var a=NaN , b='undefined'//”Nan”

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

C

//var a=10; b=20; c=4; ++b+c+a++
//21+4+10=35

面试题 14. 简述var a=10; b=20; c=4; ++b+c+a++ 以下哪个结果是正确的？

A、 34 B、35 C、36 D、37

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

B

面试题 15. 简述以下哪条语句会产生运行错误： （ ） ？

A.var obj = ();
B.var obj = [];
C.var obj = {};
D.var obj = //;

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

A

面试题 16. 简述以下哪个单词不属于 javascript 保留字（） ？

A.with
B.parent
C.class
D.void

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

B

面试题 17. 简述JS实现一个 call 函数？

推荐指数：★★★★ 试题难度： 初级 试题类型： 编程题

试题回答参考思路：

```
Function.prototype.mycall = function (context) {
if (typeof this !== 'function') {
throw new TypeError('not funciton')
}
context = context || window
context.fn = this
let arg = [...arguments].slice(1)
let result = context.fn(...arg)
delete context.fn
return result
}
```

面试题 18. 简述JS实现一个 apply 函数？

推荐指数：★★★★ 试题难度： 初级 试题类型： 编程题

试题回答参考思路：

```
Function.prototype.myapply = function (context) {
if (typeof this !== 'function') {
throw new TypeError('not funciton')
}
context = context || window
context.fn = this
let result
if (arguments[1]) {
result = context.fn(...arguments[1])
} else {
result = context.fn()
}
delete context.fn
return result
}
```

面试题 19. 简述 JS实现一个 bind 函数?

推荐指数：★★★ 试题难度： 中级 试题类型： 编程题

试题回答参考思路：

```
Function.prototype.mybind = function (context) {
if (typeof this !== 'function') {
throw new TypeError('Error')
```

```
}
let _this = this
let arg = [...arguments].slice(1)
return function F() {
// 处理函数使用new的情况
if (this instanceof F) {
return new _this(...arg, ...arguments)
} else {
return _this.apply(context, arg.concat(...arguments))
}
}
}
```

面试题 20. 简述JS实现一个节流函数？

推荐指数：★★★★★ 试题难度：中级 试题类型：编程题

试题回答参考思路：

```
function throttle (fn, delay) {
// 利用闭包保存时间
let prev = Date.now()
return function () {
let context = this
let arg = arguments
let now = Date.now()
if (now - prev >= delay) {
fn.apply(context, arg)
prev = Date.now()
}
}
}
function fn () {
console.log('节流')
}
addEventListener('scroll', throttle(fn, 1000))
```

面试题 21. 简述JS实现一个防抖函数？

推荐指数：★★★★ 试题难度：初级 试题类型：编程题

试题回答参考思路：

```
function debounce (fn, delay) {
// 利用闭包保存定时器
let timer = null
return function () {
```

```
let context = this
let arg = arguments
// 在规定时间内再次触发会先清除定时器后再重设定时器
clearTimeout(timer)
timer = setTimeout(function () {
fn.apply(context, arg)
}, delay)
}
}
function fn () {
console.log('防抖')
}
addEventListener('scroll', debounce(fn, 1000))
```

面试题 22. 简述JS柯里化函数的实现？

推荐指数：★★★★★ 试题难度：高难 试题类型：编程题

试题回答参考思路：

柯里化函数的定义：将多参数的函数转换成单参数的形式。
柯里化函数实现的原理：利用闭包原理在执行可以形成一个不销毁的作用域，然后把需要预先处理的内
容都储存在这个不销毁的作用域中，并且返回一个最少参数函数。
第一种：固定传入参数，参数够了才执行
复制/**
* 实现要点：柯里化函数接收到足够参数后，就会执行原函数，那么我们如何去确定何时达到足够的参数
呢?
* 柯里化函数需要记住你已经给过他的参数，如果没给的话，则默认为一个空数组。
* 接下来每次调用的时候，需要检查参数是否给够，如果够了，则执行fn，没有的话则返回一个新的 curry
函数，将现有的参数塞给他。
*
*/

```
// 待柯里化处理的函数
let sum = (a, b, c, d) => {
return a + b + c + d
}
// 柯里化函数，返回一个被处理过的函数
let curry = (fn, ...arr) => { // arr 记录已有参数
return (...args) => { // args 接收新参数
if (fn.length <= (...arr,...args)) { // 参数够时，触发执行
return fn(...arr, ...args)
} else { // 继续添加参数
return curry(fn, [...arr, ...args])
}
```

```
}
}
var sumPlus = curry(sum)
sumPlus(1)(2)(3)(4)
sumPlus(1, 2)(3)(4)
sumPlus(1, 2, 3)(4)
第二种：不固定传入参数，随时执行
复制/**
* 实现要点：柯里化函数接收到足够参数后，就会执行原函数，那么我们如何去确定何时达到足够的参数
呢?
* 柯里化函数需要记住你已经给过他的参数，如果没给的话，则默认为一个空数组。
* 接下来每次调用的时候，需要检查参数是否给够，如果够了，则执行fn，没有的话则返回一个新的 curry
函数，将现有的参数塞给他。
*
*/
// 待柯里化处理的函数
let sum = (a, b, c, d) => {
return a + b + c + d
}
// 柯里化函数，返回一个被处理过的函数
let curry = (fn, ...arr) => { // arr 记录已有参数
return (...args) => { // args 接收新参数
if (fn.length <= (...arr,...args)) { // 参数够时，触发执行
return fn(...arr, ...args)
} else { // 继续添加参数
return curry(fn, [...arr, ...args])
}
}
}
var sumPlus = curry(sum)
sumPlus(1)(2)(3)(4)
sumPlus(1, 2)(3)(4)
sumPlus(1, 2, 3)(4)
复制/**
* 当然了，柯里化函数的主要作用还是延迟执行，执行的触发条件不一定是参数个数相等，也可以是其他的
条件。
* 例如参数个为0的情况，那么我们需要对上面curry函数稍微做修改
*/
// 待柯里化处理的函数
let sum = arr => {
return arr.reduce((a, b) => {
return a + b
})
}
```

```
let curry = (fn, ...arr) => { // arr 记录已有参数
return (...args) => { // args 接收新参数
if (args.length === 0) { // 参数为空时，触发执行
return fn(...arr, ...args)
} else { // 继续添加参数
return curry(fn, ...arr, ...args)
}
}
}
var sumPlus = curry(sum)
sumPlus(1)(2)(3)(4)()
sumPlus(1, 2)(3)(4)()
sumPlus(1, 2, 3)(4)()
```

面试题 23. 简述JS实现一个基本的 Event Bus ？

推荐指数： ★★★ 试题难度： 中级 试题类型： 编程题

试题回答参考思路：

```
class EventEmitter {
constructor () {
// 存储事件
this.events = this.events || new Map()
}
// 监听事件
addListener (type, fn) {
if (!this.events.get(type)) {
this.events.set(type, fn)
}
}
// 触发事件
emit (type) {
let handle = this.events.get(type)
handle.apply(this, [...arguments].slice(1))
}
}
// 测试
let emitter = new EventEmitter()
// 监听事件
emitter.addListener('ages', age => {
console.log(age)
})
// 触发事件
emitter.emit('ages', 18) // 18
```

面试题 24. 简述JS实现一个双向数据绑定 ？

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 编程题

试题回答参考思路：

制let obj = {}
let input = document.getElementById('input')
let span = document.getElementById('span')
// 数据劫持
Object.defineProperty(obj, 'text', {
configurable: true,
enumerable: true,
get() {
console.log('获取数据了')
},
set(newVal) {
console.log('数据更新了')
input.value = newVal
span.innerHTML = newVal
}
})
// 输入监听
input.addEventListener('keyup', function(e) {
obj.text = e.target.value
})

面试题 25. 简述JS实现一个简单路由？

推荐指数： ★★★ 试题难度： 中级 试题类型： 编程题

试题回答参考思路：

class Route{
constructor(){
// 路由存储对象
this.routes = {}
// 当前hash
this.currentHash = ''
// 绑定this，避免监听时this指向改变
this.freshRoute = this.freshRoute.bind(this)
// 监听
window.addEventListener('load', this.freshRoute, false)
window.addEventListener('hashchange', this.freshRoute, false)
}
// 存储
storeRoute (path, cb) {
this.routes[path] = cb || function () {}
}

```
// 更新
freshRoute () {
this.currentHash = location.hash.slice(1) || '/'
this.routes[this.currentHash]()
}
}
```

面试题 26. 简述 JS rem 基本设置?

推荐指数: ★★ 试题难度: 中级 试题类型: 编程题

试题回答参考思路:

```
setRem()
// 原始配置
function setRem () {
let doc = document.documentElement
let width = doc.getBoundingClientRect().width
let rem = width / 75
doc.style.fontSize = rem + 'px'
}
// 监听窗口变化
addEventListener("resize", setRem)
```

面试题 27. 简述JS手写实现 AJAX ?

推荐指数: ★★★★ 试题难度: 中级 试题类型: 编程题

试题回答参考思路:

```
// 实例化
let xhr = new XMLHttpRequest()
// 初始化
xhr.open(method, url, async)
// 发送请求
xhr.send(data)
// 设置状态变化回调处理请求结果
xhr.onreadystatechange = () => {
if (xhr.readyStatus === 4 && xhr.status === 200) {
console.log(xhr.responseText)
}
}
// 2. 基于promise实现
function ajax (options) {
// 请求地址
const url = options.url
// 请求方法
```

```
const method = options.method.toLocaleLowerCase() || 'get'
// 默认为异步true
const async = options.async
// 请求参数
const data = options.data
// 实例化
const xhr = new XMLHttpRequest()
// 请求超时
if (options.timeout && options.timeout > 0) {
xhr.timeout = options.timeout
}
// 返回一个Promise实例
return new Promise ((resolve, reject) => {
xhr.ontimeout = () => reject && reject('请求超时')
// 监听状态变化回调
xhr.onreadystatechange = () => {
if (xhr.readyState == 4) {
// 200-300 之间表示请求成功，304资源未变，取缓存
if (xhr.status >= 200 && xhr.status < 300 || xhr.status == 304) {
resolve && resolve(xhr.responseText)
} else {
reject && reject()
}
}
}
// 错误回调
xhr.onerror = err => reject && reject(err)
let paramArr = []
let encodeData
// 处理请求参数
if (data instanceof Object) {
for (let key in data) {
// 参数拼接需要通过 encodeURIComponent 进行编码
paramArr.push(encodeURIComponent(key) + '=' +
encodeURIComponent(data[key]))
}
encodeData = paramArr.join('&')
}
// get请求拼接参数
if (method === 'get') {
// 检测url中是否已存在 ？及其位置
const index = url.indexOf('?')
if (index === -1) url += '?'
else if (index !== url.length -1) url += '&'
// 拼接url
url += encodeData
}
```

```
// 初始化
xhr.open(method, url, async)
// 发送请求
if (method === 'get') xhr.send(null)
else {
// post 方式需要设置请求头
xhr.setRequestHeader('Content-Type','application/x-www-
form urlencoded;charset=UTF-8')
xhr.send(encodeData)
}
})
}
```

面试题 28. 实现JS区间反转？

反转从位置 m 到 n 的链表。请使用一趟扫描完成反转。
说明: 1 ≤ m ≤ n ≤ 链表长度。
示例:
输入: 1->2->3->4->5->NULL, m = 2, n = 4
输出: 1->4->3->2->5->NULL

推荐指数: ★★★★★ 试题难度: 高难 试题类型: 编程题

试题回答参考思路:

循环解法和递
归解法。
需要注意的问题就是 前后节点 的保存(或者记录),

关于前节点和后节点的定义，大家在图上应该能看的比较清楚了，后面会经常用到。
反转操作上一题已经拆解过，这里不再赘述。值得注意的是反转后的工作，那么对于整个区间反转后的
工作，其实就是一个 移花接木 的过程，首先将前节点的 next 指向区间终点，然后将区间起点的 next 指
向后节点。因此这一题中有四个需要重视的节点: 前节点 、 后节点 、 区间起点 和 区间终点 。接下来我们
开始实际的编码操作。
循环解法
输入: 1->2->3->4->5->NULL, m = 2, n = 4
输出: 1->4->3->2->5->NULL

```
/**
* @param {ListNode} head
* @param {number} m
* @param {number} n
* @return {ListNode}
*/
var reverseBetween = function(head, m, n) {
let count = n - m;
```

```
let p = dummyHead = new ListNode();
let pre, cur, start, tail;
p.next = head;
for(let i = 0; i < m - 1; i ++) {
p = p.next;
}
// 保存前节点
front = p;
// 同时保存区间首节点
pre = tail = p.next;
cur = pre.next;
// 区间反转
for(let i = 0; i < count; i++) {
let next = cur.next;
cur.next = pre;
pre = cur;
```

递归解法

对于递归解法，唯一的不同就在于对于区间的处理，采用递归程序进行处理，大家也可以趁着复习一下

递归反转的实现。

3.两个一组翻转链表

给定一个链表，两两交换其中相邻的节点，并返回交换后的链表。

你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。

示例:

```
cur = next;
}
// 前节点的 next 指向区间末尾
front.next = pre;
// 区间首节点的 next 指向后节点(循环完后的cur就是区间后面第一个节点，即后节点)
tail.next = cur;
return dummyHead.next;
};
```

递归解法

对于递归解法，唯一的不同就在于对于区间的处理，采用递归程序进行处理，大家也可以趁着复习一下

递归反转的实现

```
var reverseBetween = function(head, m, n) {
// 递归反转函数
let reverse = (pre, cur) => {
if(!cur) return pre;
// 保存 next 节点
let next = cur.next;
cur.next = pre;
return reverse(cur, next);
}
```

```
let p = dummyHead = new ListNode();
dummyHead.next = head;
let start, end; //区间首尾节点
let front, tail; //前节点和后节点
for(let i = 0; i < m - 1; i++) {
p = p.next;
}
front = p; //保存前节点
start = front.next;
for(let i = m - 1; i < n; i++) {
p = p.next;
}
end = p;
tail = end.next; //保存后节点
end.next = null;
// 开始穿针引线啦，前节点指向区间首，区间首指向后节点
front.next = reverse(null, start);
start.next = tail;
return dummyHead.next;
}
```

面试题 29. 简述如何实现JS两个一组翻转链表？

给定一个链表，两两交换其中相邻的节点，并返回交换后的链表。
你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。
示例:给定 1->2->3->4, 你应该返回 2->1->4->3.

推荐指数：★★★★★ 试题难度：高难 试题类型：编程题

试题回答参考思路：

循环解决
思路清楚了，其实实现还是比较容易的，代码如下：

```
var swapPairs = function(head) {
if(head == null || head.next == null)
return head;
let dummyHead = p = new ListNode();
let node1, node2;
dummyHead.next = head;
while((node1 = p.next) && (node2 = p.next.next)) {
node1.next = node2.next;
node2.next = node1;
p.next = node2;
p = node1;
}
return dummyHead.next;
};
```

递归方式

```
var swapPairs = function(head) {
if(head == null || head.next == null)
return head;
let node1 = head, node2 = head.next;
node1.next = swapPairs(node2.next);
node2.next = node1;
return node2;
};
```

面试题 30. 简述如何实现JS一组翻转链表 ？

给你一个链表，每 k 个节点一组进行翻转，请你返回翻转后的链表。
k 是一个正整数，它的值小于或等于链表的长度。
如果节点总数不是 k 的整数倍，那么请将最后剩余的节点保持原有顺序。
示例：
给定这个链表：1->2->3->4->5
当 k = 2 时，应当返回: 2->1->4->3->5
当 k = 3 时，应当返回: 3->2->1->4->5
说明：
你的算法只能使用常数的额外空间。
你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。

推荐指数：★★★★★ 试题难度：高难 试题类型：编程题

试题回答参考思路：

思路类似No.3中的两个一组翻转。唯一的不同在于两个一组的情况下每一组只需要反转两个节点，而在
K 个一组的情况下对应的操作是将 K 个元素 的链表进行反转

递归解法
这一题我觉得递归的解法更容易理解，因此，先贴上递归方法的代码。

```
/**
* @param {ListNode} head
* @param {number} k
* @return {ListNode}
*/
var reverseKGroup = function(head, k) {
let pre = null, cur = head;
let p = head;
// 下面的循环用来检查后面的元素是否能组成一组
for(let i = 0; i < k; i++) {
if(p == null) return head;
p = p.next;
}
for(let i = 0; i < k; i++){
let next = cur.next;
cur.next = pre;
```

```
pre = cur;
cur = next;
}
// pre为本组最后一个节点，cur为下一组的起点
head.next = reverseKGroup(cur, k);
return pre;
};

循环解法
重点都放在注释里面了

var reverseKGroup = function(head, k) {
let count = 0;
// 看是否能构成一组，同时统计链表元素个数
for(let p = head; p != null; p = p.next) {
if(p == null && i < k) return head;
count++;
}
let loopCount = Math.floor(count / k);
let p = dummyHead = new ListNode();
dummyHead.next = head;
// 分成了 loopCount 组，对每一个组进行反转
for(let i = 0; i < loopCount; i++) {
let pre = null, cur = p.next;
for(let j = 0; j < k; j++) {
let next = cur.next;
cur.next = pre;
pre = cur;
cur = next;
}
// 当前 pre 为该组的尾结点，cur 为下一组首节点
let start = p.next;// start 是该组首节点
// 开始穿针引线！思路和2个一组的情况一模一样
p.next = pre;
start.next = cur;
p = start;
}
return dummyHead.next;
};
```

面试题 31. 简述JS如何检测链表形成环？

给定一个链表，判断链表中是否形成环。

推荐指数：★★★★★ 试题难度：高难 试题类型：编程题

试题回答参考思路：

思路一: 循环一遍，用 Set 数据结构保存节点，利用节点的内存地址来进行判重，如果同样的节点走过两
次，则表明已经形成了环。
思路二: 利用快慢指针，快指针一次走两步，慢指针一次走一步，如果 两者相遇 ，则表明已经形成了环。
可能你会纳闷，为什么思路二用两个指针在环中一定会相遇呢?
其实很简单，如果有环，两者一定同时走到环中，那么在环中，选慢指针为参考系，快指针每次 相对参
考系 向前走一步，终究会绕回原点，也就是回到慢指针的位置，从而让两者相遇。如果没有环，则两者
的相对距离越来越远，永远不会相遇。
接下来我们来编程实现

方法一: Set 判重

```
/**
* @param {ListNode} head
* @return {boolean}
*/
var hasCycle = (head) => {
let set = new Set();
let p = head;
while(p) {
// 同一个节点再次碰到，表示有环
if(set.has(p)) return true;
set.add(p);
p = p.next;
}
return false;
}
```

方法二: 快慢指针

```
var hasCycle = function(head) {
let dummyHead = new ListNode();
dummyHead.next = head;
let fast = slow = dummyHead;
// 零个结点或者一个结点，肯定无环
if(fast.next == null || fast.next.next == null)
return false;
while(fast && fast.next) {
fast = fast.next.next;
slow = slow.next;
// 两者相遇了
if(fast == slow) {
return true;
}
}
```

```
return false;
};
```

面试题 32. 简述JS如何找到环的起点？

给定一个链表，返回链表开始入环的第一个节点。 如果链表无环，则返回 null。
说明：不允许修改给定的链表

推荐指数：★★★★★ 试题难度：高难 试题类型：编程题

试题回答参考思路：

刚刚已经判断了如何判断出现环，那如何找到环的节点呢？我们来分析一波。
看上去比较繁琐，我们把它做进一步的抽象:
设快慢指针走了 x 秒，慢指针一秒走一次。
对快指针，有: 2x - L = m * S + Y -----①
对慢指针，有: x - L = n * S + Y -----②
其中，m、n 均为自然数。
① - ② * 2 得:
L = (m - n) * S - Y-----③
好，这是一个非常重要的等式。我们现在假设有一个新的指针在 L 段的最左端，慢指针现在还在相遇
处。
让 新指针 和 慢指针 都每次走一步，那么，当 新指针 走了 L 步之后到达环起点，而与此同时，我们看看
慢指针情况如何 。
由③式，慢指针走了 (m - n) * S - Y 个单位，以环起点为参照物，相遇时的位置为 Y，而现在由 Y +
(m - n) * S - Y 即 (m - n) * S ，得知慢指针实际上参照环起点，走了整整(m - n)圈。也就是说，慢
指针此时也到达了环起点。 :::tip 结论 现在的解法就很清晰了，当快慢指针相遇之后，让新指针从头出
发，和慢指针同时前进，且每次前进一步，两者相遇的地方，就是环起点。 :::
编程实现
懂得原理之后，实现起来就容易很多了。

```
/**
* @param {ListNode} head
* @return {ListNode}
*/
var detectCycle = function(head) {
let dummyHead = new ListNode();
dummyHead.next = head;
let fast = slow = dummyHead;
// 零个结点或者一个结点，肯定无环
if(fast.next == null || fast.next.next == null)
return null;
while(fast && fast.next) {
fast = fast.next.next;
```

```
slow = slow.next;
// 两者相遇了
if(fast == slow) {
let p = dummyHead;
while(p != slow) {
p = p.next;
slow = slow.next;
}
return p;
}
}
return null;
};
```

面试题 33. 简述JS合并两个有序链表？

将两个有序链表合并为一个新的有序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成
的。
示例:
输入：1->2->4, 1->3->4
输出：1->1->2->3->4->4

推荐指数： ★★★★★ 试题难度： 高难 试题类型： 编程题

试题回答参考思路：

递归解法
递归解法更容易理解，我们先用递归来做一下:
循环解法

```
/**
* @param {ListNode} l1
* @param {ListNode} l2
* @return {ListNode}
*/
var mergeTwoLists = function(l1, l2) {
const merge = (l1, l2) => {
if(l1 == null) return l2;
if(l2 == null) return l1;
if(l1.val > l2.val) {
l2.next = merge(l1, l2.next);
return l2;
}else {
l1.next = merge(l1.next, l2);
return l1;
}
}
return merge(l1, l2);
```

```
};

循环解法

var mergeTwoLists = function(l1, l2) {
if(l1 == null) return l2;
if(l2 == null) return l1;
let p = dummyHead = new ListNode();
let p1 = l1, p2 = l2;
while(p1 && p2) {
if(p1.val > p2.val) {
p.next = p2;
p = p.next;
p2 = p2.next;
}else {
p.next = p1;
p = p.next;
p1 = p1.next;
}
}
// 循环完成后务必检查剩下的部分
if(p1) p.next = p1;
else p.next = p2;
return dummyHead.next;
};
```

面试题 34. 简述JS实现合并 K 个有序链表 ？

合并 k 个排序链表，返回合并后的排序链表。请分析和描述算法的复杂度。
示例:
自上而下(递归)实现
自下而上实现
在这里需要提醒大家的是，在自下而上的实现方式中，我为每一个链表绑定了一个虚拟头指针
(dummyHead)，为什么这么做?
这是为了方便链表的合并，比如 l1 和 l2 合并之后，合并后链表的头指针就直接是 l1 的 dummyHead.next 值，等于说两个链表都合并到了 l1 当中，方便了后续的合并操作。
输入:
[
1->4->5,
1->3->4,
2->6
]
输出: 1->1->2->3->4->4->5->6

推荐指数： ★★★★★ 试题难度： 高难 试题类型： 编程题

试题回答参考思路：

自上而下(递归)实现

```
/**
* @param {ListNode[]} lists
* @return {ListNode}
*/
var mergeKLists = function(lists) {
// 上面已经实现
var mergeTwoLists = function(l1, l2) {/*上面已经实现*/};
const _mergeLists = (lists, start, end) => {
if(end - start < 0) return null;
if(end - start == 0)return lists[end];
let mid = Math.floor(start + (end - start) / 2);
return mergeTwoList(_mergeLists(lists, start, mid), _mergeLists(lists,
mid + 1, end));
}
return _mergeLists(lists, 0, lists.length - 1);
};
```

自下而上实现

在这里需要提醒大家的是，在自下而上的实现方式中，我为每一个链表绑定了一个虚拟头指针
(dummyHead)，为什么这么做?
这是为了方便链表的合并，比如 l1 和 l2 合并之后，合并后链表的头指针就直接是 l1 的 dummyHead.next 值，等于说两个链表都合并到了 l1 当中，方便了后续的合并操作。

```
var mergeKLists = function(lists) {
var mergeTwoLists = function(l1, l2) {/*上面已经实现*/};
// 边界情况
if(!lists || !lists.length) return null;
// 虚拟头指针集合
let dummyHeads = [];
// 初始化虚拟头指针
for(let i = 0; i < lists.length; i++) {
let node = new ListNode();
node.next = lists[i];
dummyHeads[i] = node;
}
```

多个链表的合并到这里就实现完成了，在这里顺便告诉你这种归并的方式同时也是对链表进行归并排序
的核心代码。希望你能好好体会自上而下和自下而上两种不同的实现细节，相信对你的编程内功是一个
不错的提升

面试题 35. 计算以下Javascript代码计算结果（1） ?

```
["1", "2", "3"].map(parseInt)
A：["1", "2", "3"]
B：[1, 2, 3]
```

C：[0, 1, 2]
D：other

推荐指数：★★★★★ 试题难度： 高难 试题类型： 选择题

试题回答参考思路：

解释：该题目的答案为：[1, NaN, NaN]；即选择 D。该题用到了 map 与 parseInt；
parseInt()
函数的语法是 parseInt(string, radix)；
string 必需。要被解析的字符串。
radix 可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。
如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以“0x”或“0X”开头，将以 16 为基数。
如果该参数小于 2 或者大于 36，则 parseInt()将返回 NaN。
实际上 map 里面的 callback 函数接受的是三个参数 分别为元素 下标和数组（虽然很多情况只使用第一个参数）
回调函数的语法如下所示：
function callbackfn(value, index, array1)
可使用最多三个参数来声明回调函数。
例：
var a=["1", "2", "3", "4","5",6,7,8,9,10,11,12,13,14,15];
a.map(parseInt);
返回结果为：[1, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 9, 11, 13, 15, 17, 19]

面试题 36. 计算以下Javascript代码计算结果（2）？

[typeof null, null instanceof Object]
A: ["object", false]
B: [null, false]
C: ["object", true]
D: other

推荐指数：★★★★★ 试题难度： 高难 试题类型： 选择题

试题回答参考思路：

考察 typeof 运算符和 instanceof 运算符，上 MDN 上看一下 typeof 运算符，一些基础类型的结果为：
Undefined "undefined"
Null "object"Boolean "boolean"Number "number"String "string"
Any other object "object"Array "object"
null instanceof 任何类型 都是 false，所以选 A。

面试题 37. 计算以下Javascript代码计算结果（3）？

[[3,2,1].reduce(Math.pow), [].reduce(Math.pow)]
A: an error
B: [9, 0]

C: [9, NaN]
D: [9, undefined]

推荐指数：★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

解答：这题考的 Math.pow 和 Array.prototype.reduce
Math.pow(base, exponent)接受两个参数：基数、需要计算的次方
reduce 传递给其作为参数的函数几个值：
• previousValue：上一次计算的结果
• currentValue：当前元素的值
• index： 当前元素在数组中的位置
• array：整个数组
reduce 本身接受两个参数，callback 和 initialValue，分别是 reduce 的回调函数和计算初始值--也就是第一次 reduce 的 callback 被调用时的 previousValue 的值，默认为 0
reduce 在数组为空且没有定义 initialValue 时，会抛出错误，在火狐下报错为：TypeError:
reduce of empty array with no initial value
所以选 A

面试题 38. 计算以下Javascript代码计算结果（4）？

```
var val = 'smtg';
console.log('Value is ' + (val === 'smtg') ? 'Something' : 'Nothing');
A: Value is Something
B: Value is Nothing
C: NaN
D: other
```

推荐指数：★★★★★ 试题难度： 高难 试题类型： 选择题

试题回答参考思路：

解答：这题考的 javascript 中的运算符优先级，这里'+'运算符的优先级要高于'?'所以运算符，实际上是 'Value is true'?'Something' : 'Nothing'，当字符串不为空时，转换为 bool
为 true，所以结果为'Something'，选 D

面试题 39. 计算以下Javascript代码计算结果（5）？

```
var name = 'World!';
(function () {
if (typeof name === 'undefined') {
var name = 'Jack';
console.log('Goodbye ' + name);
} else {
console.log('Hello ' + name);
}
```

```
})();
A: Goodbye Jack
B: Hello Jack
C: Hello undefined
D: Hello World
```

推荐指数：★★★★★ 试题难度： 高难 试题类型： 选择题

试题回答参考思路：

这题考的是 javascript 作用域中的变量提升，javascript 的作用于中使用 var 定义的变量都
会被提升到所有代码的最前面，于是乎这段代码就成了：

```
var name = 'World!';
(function () {
var name;//现在还是 undefined
if (typeof name === 'undefined') {
name = 'Jack';
console.log('Goodbye ' + name);
} else {
console.log('Hello ' + name);
}
})();
```

这样就很好理解了，typeof name === 'undefined'的结果为 true，所以最后会输出 'Goodbye Jack'，选 A

面试题 40. 计算以下Javascript代码计算结果（6）？

```
var END = Math.pow(2, 53);var START = END - 100;var count = 0;for (var
i = START; i <= END; i++) {
count++;
}
console.log(count);
A: 0
B: 100
C: 101
D: other
```

推荐指数：★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

解答：这题考查 javascript 中的数字的概念：首先明确一点，javascript 和其他语言不同，仅有一种数字，IEEE 754 标准的 64 位浮点数，能够表示的整数范围是-2^53~2^53（包含边界值），所以 Math.pow(2, 53)即为 javascript 中所能表示的最大整数，在最大整数在继续增大就会出现精度丢失的情况，END + 1 的值其实是等于 END 的，这也就造成了死循环。

面试题 41. 计算以下Javascript代码计算结果（7）？

```
var ary = [0,1,2];
ary[10] = 10;
ary.filter(function(x) { return x === undefined;});
A: [undefined × 7]
B: [0, 1, 2, 10]
C: []
D: [undefined]
```

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

选择 C Array.prototype.filter is not invoked for the missing elements（缺少的元素，不会调用过滤器）。

面试题 42. 计算以下Javascript代码计算结果（8）？

```
var two = 0.2var one = 0.1var eight = 0.8var six = 0.6
[two - one == one, eight - six == two]
A: [true, true]
B: [false, false]
C: [true, false]
D: other
```

推荐指数：★★★★★ 试题难度： 高难 试题类型： 选择题

试题回答参考思路：

浮点数计算时的精度丢失问题。.chrome 中计算出来的结果：[0.1, 0.20000000000000007]，也就是[true, false]，最终答案选 C。

面试题 43. 计算以下Javascript代码计算结果（9）？

```
function showCase(value) {
switch(value) {
case 'A':
console.log('Case A');
break;
case 'B':
console.log('Case B');
break;
case undefined:
console.log('undefined');
break;
default:
console.log('Do not know!');
}
}
showCase(new String('A'));
A: Case A
B: Case B
C: Do not know!
D: undefined
```

推荐指数：★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

此题考察的是关于 new string（）；其实就是 new 一个实例对象。要匹配的也是object；所以答案是 Do not know!，选择 C。

面试题 44. 计算以下Javascript代码计算结果（10）？

```
function showCase2(value) {
switch(value) {
case 'A':
console.log('Case A');
break;
case 'B':
console.log('Case B');
break;
case undefined:
console.log('undefined');
break;
default:
console.log('Do not know!');
}
}
showCase(String('A'));
```

A: Case A
B: Case B
C: Do not know!
D: undefined

推荐指数：★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

和上题原理一样，不过这里没有使用 new 来生成字符串，所以生成的结果就是原始字符串，
相当于 showCase('A')，所以结果就是 A

面试题 45. 计算以下Javascript代码计算结果（11）？

```
function isOdd(num) {
return num % 2 == 1;
}function isEven(num) {
return num % 2 == 0;
}function isSane(num) {
return isEven(num) || isOdd(num);
}var values = [7, 4, '13', -9, Infinity];
values.map(isSane);
```

A: [true, true, true, true, true]
B: [true, true, true, true, false]
C: [true, true, true, false, false]
D: [true, true, false, false, false]

推荐指数：★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

'13'在进行计算前则会进行隐式类型转换（JS 最恶心的部分之一），详细参见$雨$的文章《Javascript 类型转换的规则》，这里的规则就是将字符串通过 Number()方法转换为数字，
所以结果为 13 % 2 ，也就是 true
而 JS 中负数取模的结果是负数，这里-9%2 的结果实际上是-1，所以为 false
而 Infinity 对任意数取模都是 NaN，所以是 false
综上，结果为[true, true, true, false, false]，也就是 C

面试题 46. 计算以下Javascript代码计算结果（12） ？

parseInt(3, 8)
parseInt(3, 2)
parseInt(3, 0)
A: 3, 3, 3
B: 3, 3, NaN
C: 3, NaN, NaN
D: other

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

还是 parseInt 的题，考的和第一题类似，第一个值为 3 没什么好说的。如果出现的数字不符合后面输入的进制，则为 NaN，所以第二个值为 NaN。而 radix 为 0 时的情况第一题下面有介绍，这里也是一样为默认 10，所以结果为 3，所以答案为 3, NaN, 3，选 D

面试题 47. 计算以下Javascript代码计算结果（13） ？

Array.isArray(Array.prototype)
A: true
B: false
C: error
D: other

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

D

面试题 48. 计算以下Javascript代码计算结果（14） ？

```
var a = [0];if ([0]) {
console.log(a == true);
} else {
console.log("wut");
```

```
}
A: true
B: false
C: "wut"
D: other
```

推荐指数：★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

同样是一道隐式类型转换的题，不过这次考虑的是'=='运算符，a 本身是一个长度为 1 的数组，而当数组不为空时，其转换成 bool 值为 true。
而==左右的转换，会使用如果一个操作值为布尔值，则在比较之前先将其转换为数值的规则来转换，Number([0])，也就是 0，于是变成了 0 == true，结果自然是 false，所以最终结果为 B

面试题 49. 计算以下Javascript代码计算结果（15）？

```
[] == []
A: true
B: false
C: error
D: other
```

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

这题考的是数组字面量创建数组的原理和==运算符，首先 JS 中数组的真实类型是 Object 这点很明显 typeof []的值为"object"，而==运算符当左右都是对象时，则会比较其是否指向同一个对象。而每次调用字面量创建，都会创造新的对象，也就是会开辟新的内存区域。所以指针的值自然不一样，结果为 false，选 B

面试题 50. 计算以下Javascript代码计算结果（16）？

```
'5' + 3 '5' - 3
A: 53, 2
B: 8, 2
C: error
D: other
```

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

又是一道隐式类型转换的题
加法： 加法运算中，如果有一个操作值为字符串类型，则将另一个操作值转换为字符串，最后连接起来

减法： 如果操作值之一不是数值，则被隐式调用 Number()函数进行转换
所以第一行结果为字符串运算，为'53'。第二行结果为 2，选 A

面试题 51. 计算以下Javascript代码计算结果（17） ？

```
1 + - + + + - + 1
A: 2
B: 1
C: error
D: other
```

推荐指数： ★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

A

面试题 52. 计算以下Javascript代码计算结果（18） ？

```
var ary = Array(3);
ary[0]=2
ary.map(function(elem) { return '1'; });
A: [2, 1, 1]
B: ["1", "1", "1"]
C: [2, "1", "1"]
D: other
```

推荐指数： ★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

又是考的 Array.prototype.map 的用法，map 在使用的时候，只有数组中被初始化过元素才会被触发，其他都是 undefined，所以结果为["1", undefined × 2]，选 D

面试题 53. 计算以下Javascript代码计算结果（19） ？

```
function sidEffecting(ary) {
ary[0] = ary[2];
}function bar(a,b,c) {
c = 10
sidEffecting(arguments);
return a + b + c;
}
bar(1,1,1)
A: 3
B: 12
C: error
D: other
```

推荐指数： ★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

这题考的是 JS 的函数 arguments 的概念：
在调用函数时，函数内部的 arguments 维护着传递到这个函数的参数列表。它看起来是一个数组，但实际上它只是一个有 length 属性的 Object，不从 Array.prototype 继承。所以无法使用一些 Array.prototype 的方法。
arguments 对象其内部属性以及函数形参创建 getter 和 setter 方法，因此改变形参的值会影响到 arguments 对象的值，反过来也是一样
具体例子可以参见 Javascript 秘密花园#arguments所以，这里所有的更改都将生效，a 和 c 的值都为 10，a+b+c 的值将为 21，选 D

面试题 54. 计算以下Javascript代码计算结果（20）？

```
var a = 11111111111111110000,
b = 1111;a + b;
A: 111111111111111111111
B: 11111111111111110000
C: NaN
D: Infinity
```

推荐指数：★★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

又是一道考查 JavaScript 数字的题，与第七题考察点相似。由于 JavaScript 实际上只有一
种数字形式 IEEE 754 标准的 64 位双精度浮点数，其所能表示的整数范围为-2^53~2^53(包
括边界值)。这里的 11111111111111110000 已经超过了 2^53 次方，所以会发生精度丢失的情况。综上选 B

面试题 55. 计算以下Javascript代码计算结果（21）？

```
var x = [].reverse;
x();
A: []
B: undefined
C: error
D: window
```

推荐指数：★★★★ 试题难度：高难 试题类型：选择题

试题回答参考思路：

这题考查的是函数调用时的 this 和 Array.prototype.reverse 方法。
首先看 Array.prototype.reverse 方法，首先举几个栗子：
console.log(Array.prototype.reverse.call("skyinlayer"));//skyinlayercon
sole.log(Array.prototype.reverse.call({}));//Object {}console.log(Array.
prototype.reverse.call(123));//123

这几个栗子可以得出一个结论，Array.prototype.reverse 方法的返回值，就是 this
Javascript 中 this 有如下几种情况：
全局下 this，指向 window 对象
console.log(this);//输出结果：//Window {top: Window, window: Window, loca
tion: Location, external: Object, chrome: Object...}
函数调用，this 指向全局 window 对象：
function somefun(){
console.log(this);
}
somefun();//输出结果：//Window {top: Window, window: Window, location: Lo
cation, external: Object, chrome: Object...}
方法调用，this 指向拥有该方法的对象：
var someobj = {};
someobj.fun = function(){
console.log(this);
};console.log(someobj.fun());//输出结果：//Object {fun: function}
调用构造函数，构造函数内部的 this 指向新创建对象：
function Con() {
console.log(this);
}
Con.prototype.somefun = function(){};console.log(new Con());//输出结果：/
/Con {somefun: function}
显示确定 this：
function somefun(){
console.log(this);
};
somefun.apply("skyinlayer");
somefun.call("skyinlayer");
//输出结果：
//String {0: "s", 1: "k", 2: "y", 3: "i", 4: "n", 5: "l", 6: "a", 7: "y
", 8: "e", 9: "r", length: 10}
//String {0: "s", 1: "k", 2: "y", 3: "i", 4: "n", 5: "l", 6: "a", 7: "y
", 8: "e", 9: "r", length: 10}
这里报错：Uncaught TypeError: Array.prototype.reverse called on null or
undefined，选 C。

面试题 56. 计算以下Javascript代码计算结果（22） ？

Number.MIN_VALUE > 0
A: false
B: true
C: error
D: other

推荐指数： ★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

考查的 Number.MIN_VALUE 的概念，MDN 传送门，关键的几句话
• The Number.MIN_VALUE property represents the smallest positive numeric value representable in JavaScript.
翻译：Number.MIN_VALUE 表示的是 JavaScript 中最小的正数
• The MIN_VALUE property is the number closest to 0, not the most negative number, that JavaScript can represent.
翻译：MIN_VALUE 是接近 0 的数，而不是最小的数
• MIN_VALUE has a value of approximately 5e-324. Values smaller than MIN_VALUE ("underflow values") are converted to 0.
翻译：MIN_VALUE 值约等于 5e-324，比起更小的值（大于 0），将被转换为 0
所以，这里是 true，选 B
顺带把 Number 的几个常量拉出来：
•
o Number.MAX_VALUE：最大的正数
•
o Number.MIN_VALUE：最小的正数
•
o Number.NaN：特殊值，用来表示这不是一个数
•
o Number.NEGATIVE_INFINITY：负无穷大
•
o Number.POSITIVE_INFINITY：正无穷大
如果要表示最小的负数和最大的负数，可以使用-Number.MAX_VALUE 和 -Number.MIN_VALUE

面试题 57. 计算以下Javascript代码计算结果（23）？

[1 < 2 < 3, 3 < 2 < 1]
A: [true, true]
B: [true, false]
C: error
D: other

推荐指数：★★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

运算符的运算顺序和隐式类型转换的题，从 MDN 上运算符优先级，'<'运算符顺序是从左到右，所以变成了[true < 3, false < 1]
接着进行隐式类型转换，'<'操作符的转换规则（来自$雨$的文章《Javascript 类型转换的规则》）：
• 如果两个操作值都是数值，则进行数值比较
• 如果两个操作值都是字符串，则比较字符串对应的字符编码值
• 如果只有一个操作值是数值，则将另一个操作值转换为数值，进行数值比较
• 如果一个操作数是对象，则调用 valueOf()方法（如果对象没有 valueOf()方法则调用 toString()方法），得到的结果按照前面的规则执行比较
• 如果一个操作值是布尔值，则将其转换为数值，再进行比较
所以，这里首先通过 Number()转换为数字然后进行比较，true 会转换成 1，而 false转换

成 0，就变成了[1 < 3, 0 < 1]
所以结果为 A

面试题 58. 计算以下Javascript代码计算结果（24） ？

```
// the most classic wtf2 == [[[2]]]
A: true
B: false
C: undefined
D: other
```

推荐指数： ★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

又是隐式类型转换的题（汗）
题目作者的解释是：
both objects get converted to strings and in both cases the resulting string is "2"
也就是说左右两边都被转换成了字符串，而字符串都是"2"
这里首先需要对==右边的数组进行类型转换，根据以下规则（来自 justjavac 的文章《「译」
JavaScript 的怪癖 1：隐式类型转换》）：
1. 调用 valueOf()。如果结果是原始值（不是一个对象），则将其转换为一个数字。
2. 否则，调用 toString() 方法。如果结果是原始值，则将其转换为一个数字。
3. 否则，抛出一个类型错误。
所以右侧被使用 toString()方法转换为"2"，然后又通过 Number("2")转换为数字 2 进行比
较，结果就是 true 了，选 A

面试题 59. 计算以下Javascript代码计算结果（25） ？

```
3.toString()3..toString()3...toString()
A: "3", error, error
B: "3", "3.0", error
C: error, "3", error
D: other
```

推荐指数： ★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

说实话这题有点常见了，很多人都踩过 3.toString()的坑（包括我）...虽然 JavaScript 会在调用方法时对原始值进行包装，但是这个点是小数点呢、还是方法调用的点呢，于是乎第一个就是 error 了，因为 JavaScript 解释器会将其认为是小数点。
而第二个则很好说通了，第一个点解释为小数点，变成了(3.0).toString()，结果就是"3"了
第三个也是，第一个点为小数点，第二个是方法调用的点，但是后面接的不是一个合法的方

法名，于是乎就 error 了
综上，选 C

面试题 60. 计算以下Javascript代码计算结果（26）？

```
(function(){
var x = y = 1;
})();console.log(y);console.log(x);
A: 1, 1
B: error, error
C: 1, error
D: other
```

推荐指数：★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

变量提升和隐式定义全局变量的题，也是一个 JavaScript 经典的坑...
还是那句话，在作用域内，变量定义和函数定义会先行提升，所以里面就变成了：

```
(function(){
var x;
y = 1;
x = 1;
})();
```

这点会问了，为什么不是 var x, y;，这就是坑的地方...这里只会定义第一个变量 x，而 y 则会通过不使用 var 的方式直接使用，于是乎就隐式定义了一个全局变量 y
所以，y 是全局作用域下，而 x 则是在函数内部，结果就为 1, error，选 C

面试题 61. 计算以下Javascript代码计算结果（27）？

```
var a = /123/,
b = /123/;
a == b
a === b
A: true, true
B: true, false
C: false, false
D: other
```

推荐指数：★★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

首先需要明确 JavaScript 的正则表达式是什么。JavaScript 中的正则表达式依旧是对象，
使用 typeof 运算符就能得出结果：

```
console.log(typeof /123/);//输出结果：//"object"
```

==运算符左右两边都是对象时，会比较他们是否指向同一个对象，可以理解为 C语言中两个指针的值是否一样（指向同一片内存），所以两个结果自然都是 false

面试题 62. 计算以下Javascript代码计算结果（28）？

var a = [1, 2, 3],
b = [1, 2, 3],
c = [1, 2, 4]a == ba === ba > ca < c
A: false, false, false, true
B: false, false, false, false
C: true, true, false, true
D: other

推荐指数：★★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

和上题类似，JavaScript 中 Array 的本质也是对象，所以前两个的结果都是 false，
而 JavaScript 中 Array 的'>'运算符和'<'运算符的比较方式类似于字符串比较字典序，会从第一个元素开始进行比较，如果一样比较第二个，还一样就比较第三个，如此类推，所以第三个结果为 false，第四个为 true。
综上所述，结果为 false, false, false, true，选 A

面试题 63. 计算以下Javascript代码计算结果（29）？

var a = {}, b = Object.prototype;
[a.prototype === b, Object.getPrototypeOf(a) === b]
A: [false, true]
B: [true, true]
C: [false, false]
D: other

推荐指数：★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

原型链的题（总会有的），考查的__proto__和 prototype 的区别。首先要明确对象和构造函数的关系，对象在创建的时候，其__proto__会指向其构造函数的 prototype 属性 Object 实际上是一个构造函数（typeof Object 的结果为"function"）,使用字面量创建对象和 new Object 创建对象是一样的，所以 a.__proto__也就是 Object.prototype，而 Object.getPrototypeOf(a)与 a.__proto__是一样的，所以第二个结果为 true而实例对象是没有prototype属性的，只有函数才有，所以 a.prototype其实是 undefined，第一个结果为 false
综上，选 A

面试题 64. 计算以下Javascript代码计算结果（30）？

function f() {}var a = f.prototype, b = Object.getPrototypeOf(f);
a === b
A: true
B: false
C: null
D: other

推荐指数： ★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

还是__proto__和 prototype 的区别，两者不是一个东西，所以选 B

面试题 65. 计算以下Javascript代码计算结果（31）？

```
function foo() { }var oldName = foo.name;
foo.name = "bar";
[oldName, foo.name]
A: error
B: ["", ""]
C: ["foo", "foo"]
D: ["foo", "bar"]
```

推荐指数： ★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

考察了函数的 name 属性，使用函数定义方式时，会给 function 对象本身添加一个 name 属性，保存了函数的名称，很好理解 oldName 为"foo"。name 属性时只读的，不允许修改，所以 foo.name = "bar";之后，foo.name 还是"foo"，所以结果为["foo", "foo"]，选C

PS：name 属性不是一个标准属性，不要去使用，除非你想要坑别人

面试题 66. 计算以下Javascript代码计算结果（32）？

```
"1 2 3".replace(/\d/g, parseInt)
A: "1 2 3"
B: "0 1 2"
C: "NaN 2 3"
D: "1 NaN 3"
```

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

String.prototype.replace、正则表达式的全局匹配和 parseInt（又是 parseInt...），可以

根据题意看出来题目上漏了一个''

首先需要确定 replace 会传给 parseInt 哪些参数。举个栗子：

```
"1 2 3".replace(/\d/g, function(){
console.log(arguments);
});
//输出结果：
//["1", 0, "1 2 3"]
//["2", 2, "1 2 3"]
//["3", 4, "1 2 3"]
```

一共三个：
1. match：正则表达式被匹配到的子字符串
2. offset：被匹配到的子字符串在原字符串中的位置
3. string：原字符串
这样就很好理解了，又回到之前 parseInt 的问题了，结果就是 parseInt("1", 10), parseInt("2", 2), parseInt("3", 4)所以结果为"1, NaN, 3"，选 D

面试题 67. 计算以下Javascript代码计算结果（33）？

```
function f() {}var parent = Object.getPrototypeOf(f);
f.name // ?
parent.name // ?typeof eval(f.name) // ?typeof eval(parent.name) // ?
A: "f", "Empty", "function", "function"
B: "f", undefined, "function", error
C: "f", "Empty", "function", error
D: other
```

推荐指数：★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

又是 Function.name 属性的题，和三十二题一样样，f.name 值为"f"，而 eval("f")则会输出 f 函数，所以结果为"function"
接着看 parent，parent 实际上就是 f.__proto__，需要明确的是 JavaScript 中的函数也是对象，其也有自己的构造函数 Function，所以 f.__proto__ === Function.prototype 结果是 true，而 Function.prototype 就是一个名为 Empty 的 function

```
console.log(Function.prototype);
console.log(Function.prototype.name);
//输出结果：
//function Empty() {}
//Empty
```

所以 parent.name 的值为 Empty
如果想直接在全局作用域下调用 Empty，显示未定义...因为 Empty 并不在全局作用域下
综上所述，结果为 C

面试题 68. 计算以下Javascript代码计算结果（34）？

```
var lowerCaseOnly = /^[a-z]+$/;
[lowerCaseOnly.test(null), lowerCaseOnly.test()]
A: [true, false]
B: error
C: [true, true]
D: [false, true]
```

推荐指数：★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

正则表达式的 test 方法会自动将参数转换为字符串，原式就变成了[lowerCaseOnly.test("null"), lowerCaseOnly.test("undefined")]，结果都是真，

所以选 C

面试题 69. 计算以下Javascript代码计算结果（35） ？

```
[,,,].join(", ")

A: ", , , "
B: "undefined, undefined, undefined, undefined"
C: ", , "
D: ""
```

推荐指数： ★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

JavaScript 中使用字面量创建数组时，如果最末尾有一个逗号','，会背省略，所以实际上这
个数组只有三个元素（都是 undefined）：
console.log([,,,].length);//输出结果：//3
而三个元素，使用 join 方法，只需要添加两次，所以结果为", , "，选 C

面试题 70. 计算以下Javascript代码计算结果（36） ？

```
var a = {class: "Animal", name: 'Fido'};
a.class
A: "Animal"
B: Object
C: an error
D: other
```

推荐指数： ★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

经典坑中的一个，class 是关键字。根据浏览器的不同，结果不同：
chrome 的结果： "Animal"
Firefox 的结果："Animal"
Opera 的结果："Animal"
IE 8 以上也是： "Animal"
IE 8 及以下： 报错

面试题 71. 简述在 HTML 页面上，当按下键盘上的任意一个键时都会触发 javascript 的（）事件?

```
A：onFocus
B：onBlur
C：onSubmit
D：onKeyDown
```

推荐指数： ★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

D

面试题 72. 简述下面哪个字符串定义语句不正确？

A：var mytext=”here is some text!”
B：var mytext=’here is some text!”
C：var mytext=’here is some text!’
D：var mytext=”here is \nsome text!”

推荐指数：★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

B

面试题 73. 简述下列选项中，能获取到input 节点的一项是？

```
<br />
<form class="file" name="upload"><br />
    <input id="file" name="file"/><br />
</form><br />
```

A：document.querySelectorAll('file')[0]
B：document.getElementById('file')[0]
C：document.getElementByTagName('file')[0]
D：document.getElementById('file')

推荐指数：★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

D

面试题 74. 简述执行以下程序，输出结果为？

4.
执行以下程序，输出结果为（）

```
function Person(age){

this.age = age;

}

Person.prototype = {

constructor:Person,

getAge:function(){
```

```
console.log(this.age);

},

}

var ldh = new Person(24);

Person.prototype.age = 18;

Object.prototype.age = 20;

ldh.getAge();
```

A：24
B：18
C：20
D：undefined

推荐指数： ★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

A

面试题 75. 问：控制台打印的结果是？？

```
for(let i=0;i<2;i++){
setTimeout(function(){
console.log(i)
},100);
}
for(var i=0;i<2;i++){
setTimeout(function(){
console.log(i)
},100);
}
```

问：控制台打印的结果是?

A：0 1 2 2
B：0 1 0 1
C：0 1 1 1
D：1 1 0 0

推荐指数： ★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

A

面试题 76. 简述在 JavaScript 中，用于阻止默认事件的默认操作的方法是？

A：stopDeafault()
B：stopPropagation()
C：preventPropagation()
D：preventDefault()

推荐指数：★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

D

面试题 77. 简述执行以下代码，输出结果为（） ？

```
function test(a){
a=a+10;
}
var a=10;
test(a);
console.log(a);
```

A：10
B：20
C：抛出异常
D：undefined

推荐指数：★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

A

面试题 78. 简述下面有关浏览器中使用js跨域获取数据的描述，说法错误的是 ？

A：域名、端口相同，协议不同，属于相同的域
B：js可以使用jsonp进行跨域
C：通过修改document.domain来跨子域
D：使用window.name来进行跨域

推荐指数：★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

A

面试题 79. 下面这段JavaScript代码的的输出是什么 ？

```
var myObject = {
foo: "bar",
func: function() {
var self = this;
console.log(this.foo);
console.log(self.foo);
```

```
(function() {
console.log(this.foo);
console.log(self.foo);
}());
}
};
myObject.func();
```

A：bar bar bar bar
B：bar bar bar undefined
C：bar bar undefined bar
D：undefined bar undefined bar

推荐指数：★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

C

面试题 80. 在很多时候，我们需要给网页中的一些元素不停的切换样式，那么要怎样实现给某个元素删除一个样式的同时，添加另外一个样式？

```
A：
$('#ele').removeClass('className');
$('#ele').addClass('ClassName');
B：
$('us').removeClass('className');
$('us').addClass('ClassName');
C：
$('.us').removeClass('className');
$('.us').addClass('ClassName');
D：
$('us').remove('className');
$('us').add('ClassName');
```

推荐指数：★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

A

面试题 81. 简述 如果修改obj里面的name属性时会发生什么？

```
let obj = {name: '222'};
let _name = obj.name;
Object.defineProperty(obj, 'name', {
get() {
return _name;
},
set(newVal) {
console.log(newVal, _name);
_name = newVal;
}
```

```
})
obj.name = '11';
```

A：打印11
B：打印222
C：打印11和222
D：什么都不打印

推荐指数：★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

C

面试题 82. 简述要求匹配以下16进制颜色值，正则表达式可以为： #ffbbad #Fc01DF #FFF #ffE ？

A：/#([0-9a-f]{6}|[0-9a-fA-F]{3})/g
B：/#([0-9a-fA-F]{6}|[0-9a-fA-F]{3})/g
C：/#([0-9a-fA-F]{3}|[0-9a-f]{6})/g
D：/#([0-9A-F]{3}|[0-9a-fA-F]{6})/g

推荐指数：★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

B

面试题 83. 简述在javascript中，( )变量在函数外声明，并可从脚本的任意位置访问？

A：局部
B：全局
C：typeOf
D：New

推荐指数：★★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

B

面试题 84. 简述执行以下程序后，x的值为（）？

```
var x=0;
switch(++x) {
case 0: ++x;
case 1: ++x;
case 2: ++x;
}
```

A：1
B：2

C：3
D：4

推荐指数：★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

C

面试题 85. 简述以下哪一项不属于浏览器Response Headers字段： ？

A：Referer
B：Connection
C：Content-Type
D：Server

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

A

面试题 86. 简述上面这段代码运行后的输出是: ？

```
(function() {
var a = b = 5;
})();
console.log(b);
console.log(a);
```

上面这段代码运行后的输出是:

A：5，5
B：undefined，undefined
C：5，undefined
D：5，Uncaught ReferenceError: a is not defined

推荐指数：★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

D

面试题 87. 简述setInterval(“alert(welcome)”,1000); ？

A：等待1000秒后，再弹出一个对话框
B：等待1秒钟后弹出一个对话框
C：每隔一秒钟弹出一个对话框
D：语句报错，语法有问题

推荐指数：★★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

D

面试题 88. 简述下列window方法中，可以显示对话框的一项是（） ？

A：confirm()
B：alert()
C：prompt()
D：open()

推荐指数：★★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

C

面试题 89. 简述上面这段代码运行后的输出是 ？

(function() {
var x=foo();
var foo=function foo() {
return "foobar"
};
return x;
})();
上面这段代码运行后的输出是

A：foo()
B：类型错误
C：undefined
D：foobar

推荐指数：★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

B

面试题 90. 简述关于ES6的使用以下描述错误的是 ？

A
const a = 1;
const b = 2;
const map = {a, b};
B
enum TYPE {
OK,
YES
}
C
class A {

```
constructor (a) {
this.a = a;
}
}
class AA extends A {
constructor (a, b) {
super(a);
this.b = b;
}
toString () {
return this.a + '' + this.b;
}
}
D
function* greet(){
yield "How";
yield "are";
yield "you";
}
var greeter = greet();
console.log(greeter.next().value);
console.log(greeter.next().value);
console.log(greeter.next().value);
```

推荐指数：★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

B

面试题 91. 简述请问以下JS代码输出结果是什么？ ？

```
console.log(typeof ''.prototype);
console.log(typeof ''.__proto__);
console.log(typeof ''.__proto__ === typeof ''.prototype);
```

A：object、object、true
B：undefined、undefined、true
C：undefined、object、false
D：object、undefined、false

推荐指数：★★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

C

面试题 92. 简述JavaScript中函数的调用方式有哪些？

A：直接调用
B：作为对象方法调用

C：作为构造函数调用
D：通过call和apply方法调用

推荐指数： ★★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

ABCD

面试题 93. 简述下面属于javascript基本数据类型的有？？

A：字符串
B：数字
C：null
D：undefined

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

ABCD

面试题 94. 简述下列哪些会返回false？？

A：null
B：undefined
C：0
D：'0'

推荐指数： ★★ 试题难度： 初级 试题类型： 选择题

试题回答参考思路：

ABC

面试题 95. 简述关于JavaScript里的xml处理，以下说法正确的是 ()？

A：Xml是种可扩展标记语言，格式更规范，是作为未来html的替代
B：Xml一般用于传输和存储数据，是对html的补充，两者的目的不同
C：在JavaScript里解析和处理xml数据时，因为浏览器的不同，其做法也不同
D：在IE浏览器里处理xml，首先需要创建ActiveXObject对象

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 选择题

试题回答参考思路：

BCD

面试题 96. 简述以下对闭包（closure）理解正确的有？

A：闭包是指有权访问另一个函数作用域中变量的函数；
B：函数内再嵌套函数，返回到外部形成闭包；
C：内部函数可以引用外层的参数和变量
D：参数和变量不会被垃圾回收机制回收

推荐指数：★★★★ 试题难度：高难 试题类型：选择题

试题回答参考思路：

ABCD

面试题 97. 在一个块元素中，存在了很多的行元素，现在要求，将这些行元素中，只要有显示“叮咚”的行元素全部删除，怎么来做（ ）？

A：$("span").detach("叮咚");
B：$(p").detach("叮咚");
C：$("div").detach("叮咚");
D：$("em").detach("叮咚");

推荐指数：★★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

AD

面试题 98. 简述给网页添加JavaScript的方式有？

A：使用script标签，将javascript代码写到之间
B：添加外部javascript文件
C：使用行内javascript
D：使用@import引入javascript文件

推荐指数：★★★ 试题难度：中级 试题类型：选择题

试题回答参考思路：

ABC

面试题 99. 简述以下哪些事件支持冒泡？？

A:mouseenter
B:scroll
C:focus
D:keypress

推荐指数：★★★★ 试题难度：初级 试题类型：选择题

试题回答参考思路：

BD