

JavaScript 习题及面试题

1.

[单选题]

有以下 ES6 代码

```
function * gen() {  
    yield 1;  
    yield 2;  
    yield 3;  
}
```

下面选项描述正确的是哪个？

- A.gen()执行后返回 2
- B.gen()执行后返回 undefined
- C.gen()执行后返回一个 Generator 对象
- D.gen()执行后返回 1

来自：百度 2016 研发工程师笔试真题（三）

答案：C

提示：

这是 ES6 的新 feature， function 后面带 * 的叫做 generator function。函数运行时，返回一个迭代器。

2.

[不定项选择题]

语句 var arr=[a,b,c,d];执行后，数组 arr 中每项都是一个整数，下面得到其中最大整数语句正确的是哪几项？

- A.Math.max(arr)
- B.Math.max(arr[0], arr[1], arr[2], arr[3])
- C.Math.max.call(Math, arr[0], arr[1], arr[2], arr[3])
- D.Math.max.apply(Math,arr)

来自：百度 2016 研发工程师笔试真题（三）

答案：B C D

提示：

A 选项错误

因为函数 Math.max(x);的参数是 Number 类型，可以使小数，整数，正数，负数或者是 0.如果不是上面所述类型就会返回 NaN.

3.

[问答题]

写一个 `traverse` 函数，输出所有页面宽度和高度大于 50 像素的节点。

来自：阿里巴巴 2016 前端开发工程师笔试(二)

参考：

```
<script language="javascript">
    function traverse() {
        var arr = [];
        var elements = [];
        if (document.all) {
            elements = document.all;
        } else {
            elements = document.getElementsByTagName("*");
        }
        //console.log(elements.length);
        for (var i = 0; i < elements.length; i++) {
            var ele = elements[i];
            //console.log(ele.tagName);

            //width 返回的是字符串    offsetWidth 返回的是带边框的 Number 型的数字
            var width = parseFloat(ele.style.width) || ele.offsetWidth;
            //console.log(width);
            var height = parseFloat(ele.style.height) || ele.offsetHeight;
            //console.log(height);
            if (width > 50 && height > 50) {
                arr.push(elements[i].tagName);
            }
        }
        return arr;
    }
    window.onload=function() //注意 onload 的使用方式
    {
        console.log(traverse());
        console.log("a");
        console.log('a');
    }
</script>
```

4.

[问答题]

请写一个表格以及对应的 CSS，使表格奇数行为白色背景，偶数行为灰色背景，鼠标移上去时为黄色背景。

来自：阿里巴巴 2016 前端开发工程师笔试(二)

参考：

```
<table class="table">
    <tr><td>第一行</td></tr>
    <tr><td>第二行</td></tr>
    <tr><td>第三行</td></tr>
    <tr><td>第四行</td></tr>
</table>
<style>
    .table tr:nth-child(odd){
        background-color:white;
    }
    .table tr:nth-child(even){
        background-color:gray;
    }
    .table tr:hover{
        background-color:yellow;
    }
</style>
```

5.

[问答题]

写一个求和的函数 sum，达到下面的效果

```
// Should equal 15
sum(1, 2, 3, 4, 5);
// Should equal 0
sum(5, null, -5);
// Should equal 10
sum('1.0', false, 1, true, 1, 'A', 1, 'B', 1, 'C', 1, 'D', 1, 'E', 1, 'F', 1, 'G', 1);
// Should equal 0.3, not 0.30000000000000004
sum(0.1, 0.2);
```

来自：阿里巴巴 2016 前端开发工程师笔试(二)

参考：

```
function sum() {  
    var nResult = 0;  
    for (var i = 0, l = arguments.length; i < l; i++) {  
        nResult += window.parseFloat(arguments[i]) || 0;  
    }  
    return nResult.toFixed(3) * 1000 / 1000;  
}
```

6.

[填空题]

删除给定数组中的第二项和第三项，并且在得到的新的数组中第二项后面添加一个新的值：

```
var arr1 = ['a','b','c','d','e'];  
var arr2 = arr1.1( 2,3,'newvalue')
```

来自：阿里巴巴 2016 前端开发工程师笔试(二)

答案： splice 1 2

7.

[填空题]

在空白处填入适当的代码使输出结果成立：

```
function showMoney( ){  
    1  
};  
var personA = new Object;  
var personB = new Object;  
personA.money= "100";  
personB.money= "150";  
personA.showMoney= showMoney;  
personB.showMoney= showMoney;
```

输出结果：

```
personA.showMoney( ); //"100"  
personB.showMoney( ); //"150"
```

来自：阿里巴巴 2016 前端开发工程师笔试(二)

答案： return this.money;

8.

[填空题]

使用 `for in` 循环数组中的元素会枚举原型链上的所有属性，过滤这些属性的方式是使用 1 函数

来自：阿里巴巴 2016 前端开发工程师笔试(二)

答案：`hasOwnProperty`

9.

[问答题]

请实现一个 `fibonacci` 函数，要求其参数和返回值如下所示：

```
/**  
 * @desc: fibonacci  
 * @param: count {Number}  
 * @return: result {Number} 第 count 个 fibonacci 值，计数从 0 开始  
 *          fibonacci 数列为：[1, 1, 2, 3, 5, 8, 13, 21, 34 ...]  
 *          则 getNthFibonacci(0) 返回值为 1  
 *          则 getNthFibonacci(4) 返回值为 5  
 */  
function getNthFibonacci(count) {  
}
```

来自：阿里巴巴 2016 前端开发工程师笔试(一)

参考：

```
function getNthFibonacci(count) {  
    if (count <= 1) {  
        return 1;  
    }  
    return getNthFibonacci(count - 1) + getNthFibonacci(count - 2);  
}
```

10.

[填空题]

输出对象中值大于 2 的 key 的数组

```
var data = {a: 1, b: 2, c: 3, d: 4};  
Object.keys(data).filter(function(x) { return 1; })
```

期待输出：[“c”, “d”]

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：`data[x] > 2`

11.

[填空题]

填写内容让下面代码支持 `a.name = "name1"; b.name = "name2";`

```
function obj(name){
```

```
    1
```

```
}
```

```
obj.2 = "name2";
```

```
var a = obj("name1");
```

```
var b = new obj;
```

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：`if (name) { this.name = name; } return this; prototype.name`

12.

[填空题]

javascript 语言特性中，有很多方面和我们接触的其他编程语言不太一样，比如说，
javascript 语言实现继承机制的核心就是 1，而不是 Java 语言那样的类式继承。Javascript 解析引擎在读取一个 Object 的属性的值时，会沿着 2 向上寻找，如果最终没有找到，则该属性值为 3； 如果最终找到该属性的值，则返回结果。与这个过程不同的是，当 javascript 解析引擎执行“给一个 Object 的某个属性赋值”的时候，如果当前 Object 存在该属性，则改写该属性的值，如果当前的 Object 本身并不存在该属性，则赋值该属性的值。

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：`prototype` 原型链 `undefined`

13.

[单选题]

下面有关 html 的描述，不推荐的是？

- A. 在页面顶部添加 `doctype` 声明；
- B. 在 `</head> ... <body>` 中间插入 HTML 代码；
- C. 避免使用 `` 标签；
- D. 使用 `<table>` 元素展现学生成绩表等数据。

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：B

14.

[单选题]

下面关于 CSS 布局的描述，不正确的是？

- A.块级元素实际占用的宽度与它的 width 属性有关；
- B.块级元素实际占用的宽度与它的 border 属性有关；
- C.块级元素实际占用的宽度与它的 padding 属性有关；
- D.块级元素实际占用的宽度与它的 background 属性有关。

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：D

15.

[单选题]

下列事件哪个不是由鼠标触发的事件（）

- A.click
- B.contextmenu
- C.mouseout
- D.keydown

来自：阿里巴巴 2016 前端开发工程师笔试(一)

答案：D

16.

[问答题]

js 如何获取和设置 cookie？

来自：前端工程师进阶检测

参考：

```
// 创建 cookie
function setCookie(name, value, expires, path, domain, secure) {
    var cookieText = encodeURIComponent(name) + '=' + encodeURIComponent(value);
```

```
if (expires instanceof Date) {
    cookieText += '; expires=' + expires;
}
if (path) {
    cookieText += '; path=' + path;
}
if (domain) {
    cookieText += '; domain=' + domain;
}
if (secure) {
    cookieText += '; secure';
}
document.cookie = cookieText;
}

// 获取 cookie
function getCookie(name) {
    var cookieName = encodeURIComponent(name) + '=';
    var cookieStart = document.cookie.indexOf(cookieName);
    var cookieValue = null;
    if (cookieStart > -1) {
        var cookieEnd = document.cookie.indexOf(';', cookieStart);
        if (cookieEnd == -1) {
            cookieEnd = document.cookie.length;
        }
        cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));
    }
    return cookieValue;
}

// 删除 cookie
function unsetCookie(name) {
    document.cookie = name + "=; expires=" + new Date(0);
}
```

17.

[问答题]

请说说 cache-control 是怎么回事？

来自：前端工程师进阶检测

参考：

网页的缓存是由 HTTP 消息头中的“Cache-control”来控制的，常见的取值有 private、no-cache、max-age、must-revalidate 等，默认为 private。

Expires 头部字段提供一个日期和时间，响应在该日期和时间后被认为失效。允许客户端在这个时间之前不去检查（发请求），等同 **max-age** 的效果。但是如果同时存在，则被 **Cache-Control** 的 **max-age** 覆盖。

Expires = "Expires" ":" HTTP-date

例如：

Expires: Thu, 01 Dec 1994 16:00:00 GMT （必须是 **GMT** 格式）

如果把它设置为-1，则表示立即过期

Expires 和 **max-age** 都可以用来指定文档的过期时间，但是二者有一些细微差别

1. **Expires** 在 **HTTP/1.0** 中已经定义，**Cache-Control:max-age** 在 **HTTP/1.1** 中才有定义，为了向下兼容，仅使用 **max-age** 不够。

2. **Expires** 指定一个绝对的过期时间(**GMT** 格式),这么做会导致至少 2 个问题：

2.1 客户端和服务器时间不同步导致 **Expires** 的配置出现问题。

2.2 很容易在配置后忘记具体的过期时间，导致过期来临出现浪涌现象

3. **max-age** 指定的是从文档被访问后的存活时间，这个时间是个相对值(比如:3600s)，相对的是文档第一次被请求时服务器记录的 **Request_time**(请求时间)

4. **Expires** 指定的时间可以是相对文件的最后访问时间(**Atime**)或者修改时间(**MTime**)，而 **max-age** 相对对的是文档的请求时间(**Atime**)

5. 如果值为 **no-cache**,那么每次都会访问服务器。如果值为 **max-age**，则在过期之前不会重复访问服务器。

18.

[问答题]

你了解 **HTTP** 状态码吗，请随便介绍一下。

来自：前端工程师进阶检测

参考：

100 Continue 继续，一般在发送 **post** 请求时，已发送了 **http header** 之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置

302 Found 临时性重定向

303 See Other 临时性重定向，且总是使用 **GET** 请求新的 **URI**

304 Not Modified 自从上次请求后，请求的网页未修改过

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求

401 Unauthorized 请求未授权

403 Forbidden 禁止访问

404 Not Found 找不到如何与 **URI** 相匹配的资源

500 Internal Server Error 最常见的服务器端错误

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）

19.

[问答题]

js 数组去重。

来自：前端工程师进阶检测

参考：

有两个地方需要注意：1、返回值是否是当前引用，2、“重复”的判断条件。

```
Array.prototype.uniq = function () {
    // 长度只有 1，直接返回当前的拷贝
    if (this.length <= 1) {
        return this.slice(0);
    }
    var aResult = [];
    for (var i = 0, l = this.length; i < l; i++) {
        if (!_fExist(aResult, this[i])) {
            aResult.push(this[i]);
        }
    }
    return aResult;
    // 判断是否重复
    function _fExist(aTmp, o) {
        if (aTmp.length === 0) {
            return false;
        }
        var tmp;
        for (var i = 0, l = aTmp.length; i < l; i++) {
            tmp = aTmp[i];
            if (tmp === o) {
                return true;
            }
        }
        // NaN 需要特殊处理
        if (!o && !tmp && tmp !== undefined && o !== undefined && isNaN(tmp) &&
isNaN(o)) {
            return true;
        }
    }
    return false;
}
```

20.

[问答题]

如何获取 UA?

来自：前端工程师进阶检测

参考：

```
function whatBrowser() {  
    document.Browser.Name.value=navigator.appName;  
    document.Browser.Version.value=navigator.appVersion;  
    document.Browser.Code.value=navigator.appCodeName;  
    document.Browser.Agent.value=navigator.userAgent;  
}
```

21.

[问答题]

说说对网站重构的理解。

来自：前端工程师进阶检测

参考：

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变 UI 的情况下，对网站进行优化，在扩展的同时保持一致的 UI。

对于传统的网站来说重构通常是：

1. 表格(table)布局改为 DIV + CSS
2. 使网站前端兼容于现代浏览器(针对于不合规范的 CSS、如对 IE6 有效的)
3. 对于移动平台的优化
4. 针对于 SEO 进行优化
5. 深层次的网站重构应该考虑的方面
6. 减少代码间的耦合
7. 让代码保持弹性
8. 严格按照规范编写代码
9. 设计可扩展的 API
10. 代替旧有的框架、语言(如 VB)
11. 增强用户体验
12. 通常来说对于速度的优化也包含在重构中
13. 压缩 JS、CSS、image 等前端资源(通常是由服务器来解决)
14. 程序的性能优化(如数据读写)
15. 采用 CDN 来加速资源加载
16. 对于 JS DOM 的优化

17. HTTP 服务器的文件缓存

22.

[问答题]

js 对象的深度克隆代码实现。

来自：前端工程师进阶检测

参考：

```
function clone(Obj) {  
    var buf;  
    if (Obj instanceof Array) {  
        buf = []; // 创建一个空的数组  
        var i = Obj.length;  
        while (i--) {  
            buf[i] = clone(Obj[i]);  
        }  
        return buf;  
    } else if (Obj instanceof Object){  
        buf = {}; // 创建一个空对象  
        for (var k in Obj) { // 为这个对象添加新的属性  
            buf[k] = clone(Obj[k]);  
        }  
        return buf;  
    }else{  
        return Obj;  
    }  
}
```

23.

[问答题]

Ajax 是什么？Ajax 的交互模型？同步和异步的区别？如何解决跨域问题？

来自：前端工程师进阶检测

参考：

- AJAX 的全称是异步的 Javascript 和 XML ，是一种创建快速动态的技术，通过在后台与服务器进行少量数据交互，实现网页的异步更新，在不重新加载整个界面的情况下，做到网页的部分刷新；
- AJAX 的交互模型（AJAX 的过程）
 - 用户发出异步请求；

- 创建 XMLHttpRequest 对象;
 - 告诉 XMLHttpRequest 对象哪个函数会处理 XMLHttpRequest 对象状态的改变，为此要把对象的 onreadystatechange 属性设置为响应该事件的 JavaScript 函数的引用
 - 创建请求，用 open 方法指定是 get 还是 post，是否异步， url 地址；
 - 发送请求， send 方法
 - 接收结果并分析
 - 实现刷新
- 同步：脚本会停留并等待服务器发送回复然后再继续
 异步：脚本允许页面继续其进程并处理可能的回复
 - 跨域问题的解决
 - 使用 document.domain+iframe 解决跨子域问题
 - 使用 window.name
 - 使用 flash
 - 使用 iframe+location.hash
 - 使用 html5 的 postMessage；
 - 使用 jsonp（创建动态 script）

24.

[问答题]

事件、IE 与火狐的事件机制有什么区别？如何阻止冒泡？

来自：前端工程师进阶检测

参考：

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 捕获到的行为
2. 事件处理机制：IE 是事件冒泡、firefox 同时支持两种事件模型，也就是：捕获型事件和冒泡型事件
3. ev.stopPropagation();
注意旧 ie 的方法：ev.cancelBubble = true;

25.

[问答题]

WEB 应用从服务器主动推送 Data 到客户端有那些方式？

来自：前端工程师进阶检测

参考：

1. html5 websocket
2. WebSocket 通过 Flash

3. XHR 长时间连接
4. XHR Multipart Streaming
5. 不可见的 Iframe
6. <script>标签的长时间连接(可跨域)

26.

[问答题]

怎么重构页面？

来自：前端工程师进阶检测

参考：

1. 编写 CSS
2. 让页面结构更合理化，提升用户体验
3. 实现良好的页面效果和提升性能

27.

[问答题]

JavaScript 原型，原型链？有什么特点？

来自：前端工程师进阶检测

参考：

1. 原型对象也是普通的对象，是对象一个自带隐式的 `__proto__` 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 `null` 的话，我们就称之为原型链
2. 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

28.

[问答题]

Node.js 的适用场景

来自：前端工程师进阶检测

参考：

1. 高并发
2. 聊天
3. 实时消息推送

29.

[问答题]

写一个通用的事件侦听器函数

来自：前端工程师进阶检测

参考：

```
// event(事件)工具集，来源：github.com/markyun
markyun.Event = {
    // 页面加载完成后
    readyEvent : function(fn) {
        if (fn==null) {
            fn=document;
        }
        var oldonload = window.onload;
        if (typeof window.onload != 'function') {
            window.onload = fn;
        } else {
            window.onload = function() {
                oldonload();
                fn();
            };
        }
    },
    // 视能力分别使用 dom0||dom2||IE 方式 来绑定事件
    // 参数： 操作的元素,事件名称 ,事件处理程序
    addEvent : function(element, type, handler) {
        if (element.addEventListener) {
            //事件类型、需要执行的函数、是否捕捉
            element.addEventListener(type, handler, false);
        } else if (element.attachEvent) {
            element.attachEvent('on' + type, function() {
                handler.call(element);
            });
        } else {
            element['on' + type] = handler;
        }
    },
    // 移除事件
    removeEvent : function(element, type, handler) {
        if (element.removeEventListener) {
            element.removeEventListener(type, handler, false);
        }
    }
};
```

```
        } else if (element.datachEvent) {
            element.detachEvent('on' + type, handler);
        } else {
            element['on' + type] = null;
        }
    },
// 阻止事件 (主要是事件冒泡, 因为 IE 不支持事件捕获)
stopPropagation : function(ev) {
    if (ev.stopPropagation) {
        ev.stopPropagation();
    } else {
        ev.cancelBubble = true;
    }
},
// 取消事件的默认行为
preventDefault : function(event) {
    if (event.preventDefault) {
        event.preventDefault();
    } else {
        event.returnValue = false;
    }
},
// 获取事件目标
getTarget : function(event) {
    return event.target || event.srcElement;
},
// 获取 event 对象的引用, 取到事件的所有信息, 确保随时能使用 event;
getEvent : function(e) {
    var ev = e || window.event;
    if (!ev) {
        var c = this.getEvent.caller;
        while (c) {
            ev = c.arguments[0];
            if (ev && Event == ev.constructor) {
                break;
            }
            c = c.caller;
        }
    }
    return ev;
}
};
```

30.

[问答题]

`eval` 是做什么的，有什么建议？

来自：前端工程师进阶检测

参考：

1. 它的功能是把对应的字符串解析成 JS 代码并运行
2. 应该避免使用 `eval`，不安全，非常耗性能（2 次，一次解析成 js 语句，一次执行）

31.

[问答题]

哪些地方会出现 css 阻塞，哪些地方会出现 js 阻塞？

来自：前端工程师进阶检测

参考：

js 的阻塞特性：所有浏览器在下载 JS 的时候，会阻止一切其他活动，比如其他资源的下载，内容的呈现等等。直到 JS 下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。为了提高用户体验，新一代浏览器都支持并行下载 JS，但是 JS 下载仍然会阻塞其它资源的下载（例如图片，css 文件等）。

由于浏览器为了防止出现 JS 修改 DOM 树，需要重新构建 DOM 树的情况，所以就会阻塞其他的下载和呈现。

嵌入 JS 会阻塞所有内容的呈现，而外部 JS 只会阻塞其后内容的显示，2 种方式都会阻塞其后资源的下载。也就是说外部样式不会阻塞外部脚本的加载，但会阻塞外部脚本的执行。

CSS 怎么会阻塞加载了？CSS 本来是可以并行下载的，在什么情况下会出现阻塞加载了（在测试观察中，IE6 下 CSS 都是阻塞加载）

当 CSS 后面跟着嵌入的 JS 的时候，该 CSS 就会出现阻塞后面资源下载的情况。而当把嵌入 JS 放到 CSS 前面，就不会出现阻塞的情况了。

根本原因：因为浏览器会维持 html 中 css 和 js 的顺序，样式表必须在嵌入的 JS 执行前先加载、解析完。而嵌入的 JS 会阻塞后面的资源加载，所以就会出现上面 CSS 阻塞下载的情况。

嵌入 JS 应该放在什么位置？

1. 放在底部，虽然放在底部照样会阻塞所有呈现，但不会阻塞资源下载。
2. 如果嵌入 JS 放在 head 中，请把嵌入 JS 放在 CSS 头部。
3. 使用 `defer`（只支持 IE）。
4. 不要在嵌入的 JS 中调用运行时间较长的函数，如果一定要用，可以用 `setTimeout` 来调用。

Javascript 无阻塞加载具体方式：

1. 将脚本放在底部。`<link>`还是放在 head 中，用以保证在 js 加载前，能加载出正常显

示的页面。<script>标签放在</body>前。

2. 阻塞脚本：由于每个<script>标签下载时阻塞页面解析过程，所以限制页面的<script>总数也可以改善性能。适用于内联脚本和外部脚本。

3. 非阻塞脚本：等页面完成加载后，再加载 js 代码。也就是，在 window.onload 事件发出后开始下载代码。

4. defer 属性：支持 IE4 和 fierfox3.5 更高版本浏览器

5. 动态脚本元素：文档对象模型（DOM）允许你使用 js 动态创建 HTML 的几乎全部文档内容。代码如下：

```
<script>
    var script=document.createElement("script");
    script.type="text/javascript";
    script.src="file.js";
    document.getElementsByTagName("head")[0].appendChild(script);
</script>
```

此技术的重点在于：无论在何处启动下载，文件额下载和运行都不会阻塞其他页面处理过程，即使在 head 里（除了用于下载文件的 http 链接）。

32.

[问答题]

GET 和 POST 的区别，何时使用 POST？

来自：前端工程师进阶检测

参考：

GET：一般用于信息获取，使用 URL 传递参数，对所发送信息的数量也有限制，一般在 2000 个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制

GET 方式需要使用 Request.QueryString 来取得变量的值

POST 方式通过 Request.Form 来获取变量的值

也就是说 Get 是通过地址栏来传值，而 Post 是通过提交表单来传值。

在以下情况中，请使用 POST 请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）

2. 向服务器发送大量数据（POST 没有数据量限制）

3. 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

33.

[问答题]

什么是 "use strict";？使用它的好处和坏处分别是什么？

来自：前端工程师进阶检测

参考：

ECMAScript 5 添加了第二种运行模式：“严格模式”（strict mode）。顾名思义，这种模式使得 Javascript 在更严格的条件下运行。

设立“严格模式”的目的，主要有以下几个：

1. 消除 Javascript 语法的一些不合理、不严谨之处，减少一些怪异行为；
2. 消除代码运行的一些不安全之处，保证代码运行的安全；
3. 提高编译器效率，增加运行速度；
4. 为未来新版本的 Javascript 做好铺垫。

注：经过测试 IE6,7,8,9 均不支持严格模式。

缺点：

现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

34.

[问答题]

请解释一下 JavaScript 的同源策略。

来自：前端工程师进阶检测

参考：

概念：

同源策略是客户端脚本（尤其是 Javascript）的重要安全度量标准。它最早出自 Netscape Navigator 2.0，其目的是防止某个文档或脚本从多个不同源装载。

这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议，指一段脚本只能读取来自同一来源的窗口和文档的属性。

为什么要有同源限制：

我们举例说明：比如一个黑客程序，他利用 Iframe 把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过 Javascript 读取到你的表单中 input 中的内容，这样用户名，密码就轻松到手了。

35.

[问答题]

Flash、Ajax 各自的优缺点，在使用中如何取舍？

来自：前端工程师进阶检测

参考：

Flash：

1. Flash 适合处理多媒体、矢量图形、访问机器
2. 对 CSS、处理文本上不足，不容易被搜索

Ajax:

1. Ajax 对 CSS、文本支持很好，支持搜索
 2. 多媒体、矢量图形、机器访问不足
- 共同点：

1. 与服务器的无刷新传递消息
2. 可以检测用户离线和在线状态
3. 操作 DOM

36.

[问答题]

什么叫优雅降级和渐进增强？

来自：前端工程师进阶检测

参考：

1. 优雅降级：Web 站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于 IE 独特的盒模型布局问题，针对不同版本的 IE 的 hack 实践过优雅降级了，为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效。
2. 渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能，向页面增加无害于基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

37.

[问答题]

哪些操作会造成内存泄漏？

来自：前端工程师进阶检测

参考：

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的唯一引用是循环的，那么该对象的内存即可回收。

1. `setTimeout` 的第一个参数使用字符串而非函数的话，会引发内存泄漏。
2. 闭包
3. 控制台日志
4. 循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

38.

[问答题]

.call() 和 .apply() 的作用？

来自：前端工程师进阶检测

参考：

动态改变某个类的某个方法的运行环境

39.

[单选题]

现有如下 html 结构

```
<ul>
    <li>click me</li>
    <li>click me</li>
    <li>click me</li>
    <li>click me</li>
</ul>
```

运行如下代码：

```
var elements=document.getElementsByTagName('li');
var length=elements.length;
for(var i=0;i<length;i++){
    elements[i].onclick=function(){
        alert(i);
    }
}
```

依次点击 4 个 li 标签，哪一个选项是正确的运行结果（）？

- A. 依次弹出 1, 2, 3, 4
- B. 依次弹出 0, 1, 2, 3
- C. 依次弹出 3, 3, 3, 3
- D. 依次弹出 4, 4, 4, 4

来自：搜狗 2015 前端工程师笔试题

答案：D

40.

[单选题]

```
function Foo(){
```

```
var i=0;
return function(){
    document.write(i++);
}
}

var f1=Foo(),
f2=Foo();
f1();
f1();
f2();

请问以上程序的输出是（）
A.010
B.012
C.000
D.011
```

来自：搜狗 2015 前端工程师笔试题

答案：A

这是一个闭包，闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

这里的局部变量 i，对 f1()来说是全局变量，对 f2()来说也是全局变量，但是 f1()的 i 跟 f2()的 i 又是相互独立相互不可见的，f1()每执行一次，f1()的 i 就加 1，f2 () 每次执行一次，f2()的 i 就加油，但是相互之间不影响，因此结果是 010。

41.

[单选题]

以下 Js 程序的输出是什么（）

```
<SCRIPT LANGUAGE=""JavaScript>
    var a="undefined";
    var b="false";
    var c="";
    function assert(aVar){
        if(aVar)
            alert(true);
        else
            alert(false);
    }
    assert(a);
    assert(b);
    assert(c);
</SCRIPT>
```

- A.true, true, true
 - B.true, true, false
 - C.false, false, true
 - D.false, false, false
-

来自：搜狗 2015 前端工程师笔试题

答案：B

undefined 和 false 都是 Javascript 的数据类型，但是用双引号引起了就不是本身类型了，这是字符串，空串相当于 false，否则是 true

42.

[单选题]

JavaScript 定义 var a="40",var b=7,则执行 a%b 会得到()。

- A.5
 - B."5"
 - C.undefined
 - D.null
-

来自：搜狗 2015 前端工程师笔试题

答案：A

Javascript 是弱类型语言，但是明显字符串“40”不能用于 % 运算符，所以会根据后面的类型进行转化，最后结果是 5。

43.

[问答题]

请实现 javascript 中的 indexOf 功能，判断一个字符串 a 中是否包含另一个字符串 b。

- a) 如果包含，需要返回匹配字符串 b 的位置。
- b) 如果不包含，需要返回-1。

如：indexOf("hello","el") return 1。

来自：去哪儿网前端工程师笔试题

参考：

```
function indexOf(str,subStr){  
    var result =str.match(subStr);  
    return result ? result.index : -1;  
}  
或  
function indexOf(a, b){
```

```
        return a.search(b);
    }
```

44.

[单选题]

```
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();
程序的输出是什么？

- A.bar bar bar bar
- B.bar bar bar undefined
- C.bar bar undefined bar
- D.undefined bar undefined bar

来自：百度前端工程师笔试题

答案：C

理解关键：方法/函数是由谁(对象) 调用 的，方法/函数内部的 this 就指向谁(该对象)；

注意：被谁调用，不是处于谁的作用域，即使在作用域

- 1、func 是由 myObject 调用的，this 指向 myObject。
- 2、self 指向 myObject，相当于 myObject 的 this 的副本。
- 3、这个立即执行匿名函数表达式（IIFE）是由 window 调用的，this 指向 window。
- 4、IIFE 的作用域处于 myObject.func 的作用域中，本作用域找不到 self 变量，沿着作用域链向上查找 self 变量，找到了指向 myObject 对象的 self。

45.

[不定项选择题]

下面关于 IE、FF 下面脚本的区别描述错误的是？

- A.innerText IE 支持，FIREFOX 不支持
- B.documentElement.createElement FIREFOX 支持，IE 不支持
- C.setAttribute('class', 'styleClass') FIREFOX 支持，IE 不支持

D.用 `setAttribute` 设置事件 FIREFOX 不支持, IE 支持

来自：百度前端工程师笔试题

参考：

A 描述正确, 请参考这里 <http://w3help.org/zh-cn/causes/SD9017>

B 描述错误, 请参考这里 <http://w3help.org/zh-cn/causes/SD9010>

C 描述正确, 请参考这里 <http://w3help.org/zh-cn/causes/SD9006>

D 选项描述不清楚, 其实都可以的, 只是形式不同, 具体参考这里 <http://w3help.org/zh-cn/causes/SD9006>

可以看到, 在 IE8(S) Firefox Chrome Safari Opera 中, 结果符合规范。而在 IE6 IE7 IE8(Q) 中, 无法通过 `setAttribute` 方法传入一段代码字符串设置一个元素的内联事件, 而必须传入一个 `function` 类型的对象; 获取一个已有的内联事件的属性值也是 `function` 类型, 而不是规范中的字符串类型。

46.

[单选题]

下面有关 JavaScript 中 `call` 和 `apply` 的描述, 错误的是?

A.`call` 与 `apply` 都属于 `Function.prototype` 的一个方法, 所以每个 `function` 实例都有 `call`、`apply` 属性

B.两者传递的参数不同, `call` 函数第一个参数都是要传入给当前对象的对象, `apply` 不是

C.`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

D.`call` 传入的则是直接的参数列表。`call` 方法可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。

来自：前端工程师能力评估

答案: B

`call ()` 方法和 `apply ()` 方法的作用相同, 他们的区别在于接收参数的方式不同。对于 `call ()`, 第一个参数是 `this` 值没有变化, 变化的是其余参数都直接传递给函数。(在使用 `call ()` 方法时, 传递给函数的参数必须逐个列举出来。使用 `apply ()` 时, 传递给函数的是参数数组) 如下代码做出解释:

```
function add(c, d){  
    return this.a + this.b + c + d;  
}  
var o = {a:1, b:3};  
add.call(o, 5, 7); // 1 + 3 + 5 + 7 = 16  
add.apply(o, [10, 20]); // 1 + 3 + 10 + 20 = 34
```

47.

[不定项选择题]

下面哪些方法可以用作 javascript 异步模式的编程？

- A. 回调函数
 - B. 事件监听
 - C. 发布/订阅
 - D. Promises 对象
-

来自：前端工程师能力评估

答案：abcd

1. 回调函数

```
f1();
f2();
function f1(callback){
    setTimeout(function () {
        // f1 的任务代码
        callback();
    }, 1000);
}
f1(f2);
```

2. 事件监听

```
f1.on('done', f2);
function f1(){
    setTimeout(function () {
        // f1 的任务代码
        f1.trigger('done');
    }, 1000);
}
```

3. 发布/订阅

```
jQuery.subscribe("done", f2);
function f1(){
    setTimeout(function () {
        // f1 的任务代码
    }, 1000);
}
jQuery.unsubscribe("done", f2);
```

4. Promises 对象

```
f1().then(f2);
function f1(){
    var dfd = $.Deferred();
    setTimeout(function () {
        // f1 的任务代码
        dfd.resolve();
    }, 500);
```

```
    return dfd.promise;
}

指定多个回调函数:
f1().then(f2).then(f3);
指定发生错误时的回调函数:
f1().then(f2).fail(f3);
```

48.

[单选题]

下面有关 javascript 常见事件的触发情况，描述错误的是？

- A.onmousedown: 某个鼠标按键被按下
- B.onkeypress: 某个键盘的键被按下或按住
- C.onblur: 元素获得焦点
- D.onchange: 用户改变域的内容

来自：前端工程师能力评估

答案：c

- onBlur:当失去输入焦点后产生该事件
- onFocus:当输入获得焦点后，产生该文件
- onchange:当文字值改变时，产生该事件
- onselect:当文字加亮后，产生该事件
- onClick: 当组件被点击时产生的事件

49.

[不定项选择题]

下面有关 javascript 内部对象的描述，正确的有？

- A.History 对象包含用户（在浏览器窗口中）访问过的 URL
- B.Location 对象包含有关当前 URL 的信息
- C.Window 对象表示浏览器中打开的窗口
- D.Navigator 对象包含有关浏览器的信息

来自：腾讯 2015 春招 web 前端开发练习卷

答案：abcd

Navigator: 提供有关浏览器的信息。

Window: Window 对象处于对象层次的最顶层，它提供了处理 Navigator 窗口的方法和属性。

Location: 提供了与当前打开的 URL 一起工作的方法和属性，是一个静态的对象。

History: 提供了与历史清单有关的信息。

Document: 包含与文档元素一起工作的对象，它将这些元素封装起来供编程人员使用。

50.

[单选题]

下面有关 javascript 系统方法的描述，错误的是？

- A.parseFloat 方法：该方法将一个字符串转换成对应的小数
 - B.isNaN 方法：该方法用于检测参数是否为数值型，如果是，返回 true，否则，返回 false。
 - C.escape 方法：该方法返回对一个字符串编码后的结果字符串
 - D.eval 方法：该方法将某个参数字符串作为一个 JavaScript 执行
-

来自：前端工程师能力评估

答案：B

NaN,即非数值（Not a Number）是一个特殊的数值，这个数值用来表示一个本来要返回数值的操作未返回数值的情况（这样就不会抛出错误了）。

针对 NaN 的特点，ECMAScript 定义了 isNaN() 函数。这个函数接受一个参数，该参数可以是任何类型，而函数会帮我们确定这个参数是否“不是数值”。isNaN() 在接受一个值后，会尝试将这个值转换为数值。某些不是数值的值会直接转换为数值，例如字符串“10”或 Boolean 值。而任何不能被转换为数值的值都会导致这个函数返回 true。

51.

[单选题]

下面符合一个有效的 javascript 变量定义规则的是？

- A._2
 - B.with
 - C.a bc
 - D.2a
-

来自：前端工程师能力评估

答案：A

52.

[不定项选择题]

下面属于 javascript 基本数据类型的有？

- A.字符串
- B.数字
- C.null
- D.undefined

来自：腾讯 2015 春招 web 前端开发练习卷

答案：abcd

ECMAScript 中有 5 个简单数据类型(也成为基本数据类型): Undefined、Null、Boolean、Number 和 String。还有一种复杂数据类型-Object，Object 本质上是由一种无序的名值对组成的。

53.

[单选题]

js 中字符串连接用那个比较高效？

- A.a+=b
- B.a = a+b
- C.Array.join()
- D.Array.push()

来自：前端工程师能力评估

答案：C

javascript 中字符串连接时用 Array.join() 替换 string += "xx"，换来几十倍的速度提升。

54.

[单选题]

请给出这段代码的运行结果（）

```
<SCRIPT LANGUAGE="JavaScript">
var bb = 1;
function aa(bb) {
    bb = 2;
    alert(bb);
};
aa(bb);
alert(bb);
</SCRIPT>
```

- A.1 1
- B.1 2
- C.2 1
- D.2 2

来自：前端工程师能力评估

答案: C

ECMAScript 中所有函数的参数都是按值传递的。也就是说，把函数外部的值复制给函数内部的参数，就和把一个值从一个变量复制到另外一个变量一样。基本类型的传递如同基本类型变量的复制一样，而引用类型值的传递，则如同引用类型变量的复制一样。

在向参数传递基本类型的值时，被传递的值会被复制给一个局部变量（即命名参数，或者用 ECMAScript 的概念来说，就是 `arguments` 对象中的一个元素）。在向参数传递引用类型的值时，会把这个值在内存中的地址复制给一个局部变量，因此这个局部变量的变化会反映在函数的外部。

55.

[单选题]

下面这个 JS 程序的输出是什么：

```
function Foo() {  
    var i = 0;  
    return function() {  
        console.log(i++);  
    }  
}  
var f1 = Foo(),  
    f2 = Foo();  
f1();  
f1();  
f2();
```

- A.0 1 0
- B.0 1 2
- C.0 0 0
- D.0 0 2

来自：搜狐

答案：A

56.

[单选题]

在文件 `/home/somebody/workspace/somemodule.js` 中第一行引用了一个模块：

`require('othermodule'),` 请问 `required` 的查找模块的顺序

- A. `/home/somebody/workspace/module_modules/othermodule/index.js`
- B. `/home/somebody/workspace/module_modules/othermodule.js`
- C. CORE MODULES named othermodule
- D. `/home/somebody/module_modules/othermodule/index.js`

- A.C D A B
 - B.C B D A
 - C.C B A D
 - D.C D B A
-

来自：阿里巴巴

答案：C

(1):首先，Node 在当前目录下查找 package.json(CommonJS 包规范定义的包描述文件)，通过 JSON.parse()解析出包描述对象，从中取出 main 属性指定的文件名进行定位。如果文件缺少扩展名，将会进入扩展名分析的步骤。

(2):而如果 main 属性制定的文件名错误，或者压根没有 package.json 文件，Node 会将 index 当做默认文件名，然后依次查找 index.js、index.node、index.json.

(3):如果在目录分析的过程中没有定位成功任何文件，则自定义模块进入下一个模块路径进行查找。如果模块路径数组都被遍历完毕，依然没有查找到目标文件，则会抛出查找失败异常。

按照上面的思路，首先应该查找 package.json 文件，看看里面有没有核心模块，应该是 C 最先，othermodule 不是核心模块，那么接着应该进入扩展名分析的步骤，就应该是查找 othermodule.js，对应 B，紧接着就是以 index 为默认文件名，也就是 A，再接下来就是上一个文件目录 D 了，

57.

[不定项选择题]

下面哪些方式可以用于 javascript 延迟加载？

- A.通过 ajax 下载 js 脚本，动态添加 script 节点
 - B.通过监听 onload 事件，动态添加 script 节点
 - C.直接将 script 节点放置在之前，这样 js 脚本就会在页面显示出来之后再加载
 - D.使用 script 标签的 defer 和 async 属性
-

来自：百度

答案：ABD

58.

[单选题]

如下代码输出的结果是什么：

```
console.log(1+ "2"+"2");
console.log(1+ +"2"+ "2");
console.log("A"- "B"+ "2");
console.log("A"- "B"+2);
```

- A.122 122 NaN NaN
B.122 32 NaN NaN2
C.122 32 NaN2 NaN
D.122 32 NaN2 NaN2
-

来自：百度

答案：C

1.

```
console.log(1+ "2"+"2");
```

做加法时要注意双引号，当使用双引号时，JavaScript 认为是字符串，字符串相加等于字符串合并。

因此，这里相当于字符串的合并，即为

2.

```
console.log(1+ +"2"+"2");
```

第一个+"2"中的加号是一元加操作符，+"2"会变成数值 2，因此 1+ +"2"相当于 1+2=3.

然后和后面的字符串“2”相合并，变成了字符串"32".

3.

```
console.log("A"- "B"+"2");
```

"A"- "B"的运算中，需要先把"A"和"B"用 Number 函数转换为数值，其结果为 NaN，在剪发操作中，如果有一个是 NaN，则结果是 NaN，因此"A"- "B"结果为 NaN。

然后和"2"进行字符串合并，变成了 NaN2.

4.

```
console.log("A"- "B"+2);
```

根据上题所述，"A"- "B"结果为 NaN，然后和数值 2 进行加法操作，在加法操作中，如果有一个操作数是 NaN，则结果为 NaN。

59.

[单选题]

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
-

来自：阿里巴巴

答案：A

只要协议、域名、端口有任何一个不同，都被当作是不同的域。

60.

[问答题]

阅读下列程序，写出 x, y, z 最后的值。

```
var x = 1, y = z = 0;
function add(n)
{
    return n = n + 1;
}
y = add(x);
function add(n)
{
    return n = n + 3;
}
z = add(x);
```

来自：百度

参考：

x=1

y=4

z=4

js 的预编译会将

```
function add(n)
{
    return n = n + 3;
}
```

去覆盖前面的 add 方法。

```
var y = z = 0
```

这里面 z 变量会变为 window.z=0; 这里需要注意的

61.

[不定项选择题]

如何规避 javascript 多人开发函数重名问题。

- A. 根据不同的开发人员实现的功能，在函数名加前缀
 - B. 每个开发人员都把自己的函数封装到类中，然后调用的时候即使函数名相同，但是因为是要类.函数名来调用，所以也减少了重复的可能性
 - C. 以上都不正确
-

来自：腾讯 2015 春招

答案：AB

- A, 函数名之前加上开发人员特有的前缀，可以有效避免重名问题
- B, 类的封装是面向对象程序设计语言规避重名问题的有效途径

62.

[单选题]

下列 js 可以让一个 input 的背景颜色变成红色的是？

- A.inputElement.style.backgroundColor = 'red';
- B.inputElement.backgroundColor = 'red';
- C.inputElement.style.backgroundColor = '#0000';
- D.inputElement.backgroundColor = '#0000';

来自：百度

答案：A

B 没有 style

C 和 D 是因为红色#0000 应该为#FF0000

63.

[不定项选择题]

下列关于比较 Ajax 与 Flash 的优缺点，相关描述正确的是？

- A.Ajax 的优势在意在于可搜索性，开放性，易用性及易于开发
- B.Flash 的优势在于多媒体处理，可以更容易的调用浏览器以外的外部资源
- C.Ajax 最主要的批评就是它可能破坏浏览器的后退功能
- D.flash 文件经常会很大，用户第一次使用的时候需要忍耐较长的等待时间

来自：百度

答案：A B C D

1.Ajax 的优势： 1.可搜索性 2.开放性 3.费用 4.易用性 5.易于开发。

2.Flash 的优势： 1.多媒体处理 2.兼容性 3.矢量图形 4.客户端资源调度

3.Ajax 的劣势： 1.它可能破坏浏览器的后退功能 2.使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中，不过这些都有相关方法解决。

4.Flash 的劣势： 1.二进制格式 2.格式私有 3.flash 文件经常会很大，用户第一次使用的时候需要忍耐较长的等待时间 4.性能问题

64.

[不定项选择题]

下面哪些语句可以在 JS 里判断一个对象 oStringObject 是否为 String。

- A.oStringObject instanceof String

- B.typeof oStringObject == 'string'
 - C.oStringObject is String
 - D.以上答案都不正确
-

来自：百度

答案：A

通常来说判断一个对象的类型使用 `typeof`,但是在 `new String` 的情况下的结果会是 `object` 此时需要通过 `instanceof` 来判断

65.

[单选题]

flash 和 js 通过什么类如何交互？

- A.ExtensionContex
 - B.ExternalInterface
 - C.IInterpolator
 - D.FlexContentHolder
-

来自：网易

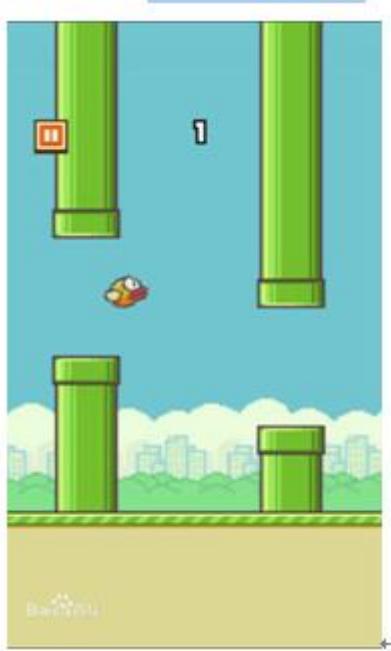
答案：B

Flash 提供了 `ExternalInterface` 接口与 JavaScript 通信，`ExternalInterface` 有两个方法，`call` 和 `addCallback`，`call` 的作用是让 Flash 调用 js 里的方法，`addCallback` 是用来注册 flash 函数让 js 调用。

66.

[问答题]

`Flappy Bird` 是风靡一时的手机游戏，玩家要操作一只小鸟穿过无穷无尽的由钢管组成的障碍。如果要在 HTML 前端开发这个游戏，为了保证游戏的流畅运行，并长时间运行也不会崩溃，请列举开发要注意的性能问题和解决的方法。



来自：百度 2015 前端研发笔试卷

参考：

1.长时间运行会崩溃的原因就是‘内存泄漏’。我们在日常的 JS 程序中并不太在意内存泄漏问题，因为 JS 解释器会垃圾回收机制，大部分无效内存会被回收，另一方面 JS 运行在客户端，即使出现内存泄漏也不是太大的问题，简单的刷新页面即可。但是，如果出现要预防内存泄漏的场景还是要注意一些问题。

2.针对这个场景来说，即时长期运行出现内存泄漏可能还是很低。第一方面，数据量很少，水管维护一个数组即可，然后每隔一段时间更新数组，来达到水管长度不同的效果。小鸟只要维护一个对象即可。通过移动水管检查碰撞来就可以实现游戏逻辑。因为在浏览器端，JS 程序和页面 UI 渲染共用一条线程，如果计算时间过长会使渲染阻塞，在 HTML5 中利用 webworker 已经可以开辟一个新线程专门负责计算解决这个问题了。

3.

背景的卷轴效果优化。背景不能是无限长的图片拼接，必须有回收已移出的场景的方法。

将复杂运算从主 UI 线程中解耦。比如场景中小鸟的运动轨迹、碰撞算法等，需要在空闲时间片运算，不能和 UI 动画同时进行。

将比较大的运算分解成不同的时间片，防止阻塞主 UI 线程。最好使用 webworker。

注意内存泄漏和回收。使用对象池管理内存，提高内存检测和垃圾回收。

进行预处理。将一些常用的过程进行预处理，

控制好帧率。将 1 秒分解成多个时间片，在固定间隔时间片进行 UI 动画，其他时间片用在后台运算。

通过 GPU 加速和 CSS transition 将小鸟飞行动画和背景动画分离

.....

67.

[问答题]

用 javascript 实现用户登录验证的代码。

来自：腾讯 2015 春招 web 前端开发练习卷

参考：

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title></title>
    <meta name="viewport" content="width=device-width,minimum-scale=1.0,maximum-scale=1.0" />
<body>
<form id="form1" action="" method="get">
    <input name="text1" type="text" value="" placeholder="name">
    <input type="password" name="pwd" placeholder="密码">
    <input name="text3" value="submit" type="submit" onclick="subimtonclick()">
</form>
<script language="JavaScript">
    function subimtonclick(){
        var form1=document.getElementById('form1');
        if(form1.text1.value==""){
            alert("用户名不能为空");
            form1.text1.focus();
            return;
        }
        if(form1.text1.value.length<6 || form1.text1.value.length>10){
            alert("用户名不能少于六个字符，不能超过十个字符");
            form1.text1.focus();
            return;
        }

        if(form1.pwd.value==""){
            alert("密码不能为空");
            form1.pwd.focus();
            return;
        }
        if(form1.pwd.value.length<6 || form1.pwd.value.length>10){
            alert("密码不能少于六个字符，不能超过十个字符");
            form1.pwd.focus();
            return;
        }

        var str="用户名：" +form1.text1.value+"<br>" +
               "密码：" +form1.pwd.value+"<br>";
    }
</script>
```

```
        document.writeln(str);
    }
</script>
</body>
</html>
```

68.

[问答题]

请说明下面各种情况的执行结果，并注明产生对应结果的理由。

```
function doSomething() {
    alert(this);
}
```

- ① element.onclick = doSomething，点击 element 元素后。
- ② element.onclick = function() {doSomething()}, 点击 element 元素后。
- ③ 直接执行 doSomething()。

来自：阿里巴巴 2011 前端工程师面试题

参考：

- ① 弹出 element object，通过函数赋值方式，this 直接指向 element 对象
- ② 弹出 window object，this 是写在 doSomething 这个函数里面的，而这种方式的事件绑定写法并没有将 element 对象传递给 this，而在默认情况下 this 指向 window
- ③ 弹出 window object，没有绑定对象的情况下 this 默认指向 window

69.

[问答题]

请用 JavaScript 语言实现 sort 排序函数，要求：sort([5, 100, 6, 3, -12]) // 返回 [-12, 3, 5, 6, 100]。

如果你有多种解法，请阐述各种解法的思路及优缺点。（仅需用代码实现一种解法，其它解法用文字阐述思路即可）

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
/*快速排序实现方法*/
function quickSort(arr){
    //长度少于 1 直接返回
    if(arr.length <= 1){
        return arr;
    }
    var less  = new Array();
```

```
var greater = new Array();
//选择一个值做为基准
var pivotIndex = Math.floor(arr.length / 2);
var pivot = arr.splice(pivotIndex, 1) [0];
//遍历数组
for(var i = 0; i<arr.length; i++ ){
  if( arr[i] < pivot ){
    less.push(arr[i]);
  }else{
    greater.push(arr[i]);
  }
}
//递归调用返回排序好的数组
return quickSort(greater).concat([pivot], quickSort(less));
}
```

70.

[问答题]

请根据下面的描述，用 JSON 语法编写一个对象：“小明今年 22 岁，来自杭州。兴趣是看电影和旅游。他有两个姐姐，一个叫小芬，今年 25 岁，职业是护士。还有一个叫小芳，今年 23 岁，是一名小学老师。”

```
var person = ?
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
var person = {
  name:"小明",
  from:"杭州",
  interest:["电影","旅游"],
  sisters:[
    { name:"小芬", age:25, occupation:"护士"},
    { name:"小芳", age:23, occupation:"小学老师"}
  ]
}
```

71.

[问答题]

请给 `Array` 本地对象增加一个原型方法，它用于删除数组条目中重复的条目(可能有多个)，返回值是一个包含被删除的重复条目的新数组。

来自：腾讯 2015 春招 web 前端开发练习卷

参考：

```
Array.prototype.distinct = function() {
    var ret = [];
    for (var i = 0; i < this.length; i++) {
        {
            for (var j = i+1; j < this.length;) {
                if (this[i] === this[j]) {
                    ret.push(this.splice(j, 1)[0]);
                } else {
                    j++;
                }
            }
        }
        return ret;
    }
    //for test
    alert(['a','b','c','d','b','a','e'].distinct());
}
```

72.

[问答题]

请把以下用于连接字符串的 JavaScript 代码修改为更有效率的方式

```
var htmlString =
    "<div class='container'>" + "<ul id='news-list'>";
for (var i = 0; i < NEWS.length; i++) {
    htmlString += "<li><a href='"
        + NEWS[i].LINK + ">"
        + NEWS[i].TITLE + "</a></li>";
}
htmlString += "</ul></div>";
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

JavaScript 赋值效率问题

```
var htmlString = "<div class='container'>" + "<ul id='news-list'>";
for (var i = 0; i < NEWS.length; i++)
{
    var newsItem = NEWS[i];
    htmlString += "<li><a href='"
        + newsItem.LINK + ">"
        + newsItem.TITLE + "</a></li>";
}
```

```
htmlString += "</ul></div>";
```

73.

[问答题]

请编写一个 JavaScript 函数 `parseQueryString`, 它的用途是把 URL 参数解析为一个对象, 如:

```
var url = "http://www.taobao.com/index.php?key0=0&key1=1&key2=2....."
var obj = parseQueryString(url);
alert(obj.key0) // 输出 0
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
var url = "http://www.taobao.com/index.php?key0=0&key1=1&key2=2.....";
var obj = parseQueryString(url);
alert(obj.key2);
function parseQueryString(argu){
    var str = argu.split('?')[1];
    var result = {};
    var temp = str.split('&');
    for(var i=0; i<temp.length; i++)
    {
        var temp2 = temp[i].split('=');
        result[temp2[0]] = temp2[1];
    }
    return result;
}
```

74.

[问答题]

请给 JavaScript 的 String 原生对象添加一个名为 `trim` 的原型方法, 用于截取空白字符。

要求

```
alert(" taobao ".trim());      // 输出 "taobao"
alert(" taobao ".trim());      // 输出 "taobao"
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
//版本一、速度最快
String.prototype.trim=function(){
    var str=this;
```

```

var
whitespace='\n\r\t\f\x0b\xA0\u2001\u2002\u2003\n\u2004\u2005\u2006\u2007\u2008\u
2009\u200a\u200b\u2028\u2029\u3000';
for(var i=0;i<str.length;i++){
    if(whitespace.indexOf(str.charAt(i))==-1){
        str=str.substring(i);
        break;
    }
}
for(i=str.length-1;i>=0;i--){
    if(whitespace.indexOf(str.charAt(i))==-1){
        str=str.substring(0,i+1);
        break;
    }
}
return whitespace.indexOf(str.charAt(0)) === -1 ? str : '';
}

//版本二、速度也很快
String.prototype.trim=function(){
var str=this;
str.replace(/^\s+/,"");
for(var i=str.length-1;i>=0;i++){
if(/\S/.test(str.charAt(i))){
str=str.substring(0,i+1);
break;
}
}
return str;
}

//版本三 很通用
String.prototype.trim=function(){
var str=this;
return str.replace(/^\s\s*/,"").replace(/\s\s*\$/,"");
}

```

75.

[问答题]

下面是个输入框

当没有获取焦点时，显示灰色的提示信息：

当用户输入时，隐藏提示文字，且恢复为默认色：

请输入内容

当输入框失去焦点，如果输入为空，需还原提示信息：

要求： a) 写出 HTML 和 CSS 代码 b) 用 JavaScript 实现功能

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Document</title>
</head>
<style>
#content{
color:grey;
}
</style>
<body>
<input type="text" id="content" value="请输入内容">
</body>
<script>
var input = document.getElementById("content");
input.onfocus = function(){
if(this.value == "请输入内容"){
this.value = "";
this.style.color = "black";
}
}
input.onblur = function(){
if(this.value == ""){
this.style.color = "grey";
this.value = "请输入内容";
}
}
</script>
</html>
```

76.

[问答题]

请编写一个通用的事件注册函数（请看下面的代码）。

```
function addEvent(element, type, handler)
{
```

```
// 在此输入你的代码，实现预定功能  
}
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
function addEvent (ele,type,handler) {  
    if ("addEventListener" in window) {  
        ele.addEventListener(type,handler,false);  
    } else if("attachEvent" in window){  
        ele.attachEvent('on' + type,function(){  
            handler.call(ele);  
        });  
    } else{  
        ele["on" + type] = handler;  
    }  
}
```

77.

[问答题]

请编写一段 JavaScript 脚本生成下面这段 DOM 结构。要求：使用标准的 DOM 方法或属性。

```
<div id="example">  
    <p class="slogan">淘！你喜欢</p>  
</div>
```

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
window.onload = function() {  
    var div = document.createElement('div');  
    div.id = "example";  
    var p = document.createElement('p');  
    p.className = "slogan";  
    p.innerHTML = '淘！你喜欢';  
    div.appendChild(p);  
    document.body.appendChild(div);  
}
```

78.

[问答题]

请分别列出 HTML、JavaScript、CSS、Java、php、python 的注释代码形式。

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
<!-- HTML 注释 -->
// JavaScript 注释
/*
 *   JavaScript 多行注释
 */
/* CSS 注释 */
// Java 注释
/**
 * Java 多行注释
 */
// php 单行注释
# php 单行注释
/**
 * php 多行注释
 */
# Python 单行注释
...
Python 多行注释
'''
```

79.

[问答题]

实现一个所见即所得编辑器。需提供以下功能： 1. 字体加粗； 2. 文本左对齐、右对齐、居中 3. 设置字体； 4. 设置字号； 5. 设置字体颜色； 6. 插入超链接； 7. 插入图片。

来自： 人人网 2011 前端工程师面试题

参考：

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>富文本编辑器</title>
<script src="script/dialog.js"></script>
<script>
window.onload = function(){
    var editor = document.getElementById("editor").children[0];
```

```
function setBtn(id, styleName){
    var elm = document.getElementById(id);
    elm.onclick = function(){
        editor.style[styleName] = setting[id] = !setting[id] ? id : "";
    }
}

//setBtn("bold", "fontWeight");
setBtn("left", "textAlign");
setBtn("center", "textAlign");
setBtn("right", "textAlign");

function setMenu(id, styleName){
    var elm = document.getElementById(id);
    function change(){
        editor.style[id] = elm.value || elm.children[elm.selectedIndex].innerHTML;
    }
    change();
    elm.onchange = change;
}

setMenu("color");
setMenu("fontSize");
setMenu("fontFamily");

function setChecked(id, styleName){
    var elm = document.getElementById(id);
    function change(){
        editor.style[elm.name] = elm.checked ? this.value : "";
    }
    change();
    elm.onchange = elm.onpropertychange = change;
}

setChecked("bold");
setChecked("left");
setChecked("center");
setChecked("right");
var link = document.getElementById("link"),
    img = document.getElementById("img");

document.getElementById("img").onclick = function(){
    var url = prompt("请输入图片 url", "http://"),
        img;
```

```
        if(url){
            img = new Image();
            img.src = url;
            editor.appendChild(img)
        }
    }

document.getElementById("lnk").onclick = function(){
    var url = prompt("请输入链接 url", "http://"),
        lnk;
    if(url){
        lnk = document.createElement("a");
        lnk.href = url;
        lnk.innerHTML = prompt("请输入链接文字或者点击取消", "") || url;
        editor.appendChild(lnk)
    }
}

};

</script>
<link href="style/base.css" rel="stylesheet" type="text/css">
<style type="text/css">
#wrap {
    padding: 100px;
}
#wrap p {
    padding-bottom: 20px;
    line-height: 30px;
}
#wrap p * {
    vertical-align: middle;
}
button {
    padding: 0 10px;
}
#editor {
    border: 2px inset #ccc;
    font-family: SimSun;
    overflow: hidden;
    cursor: text;
    margin: auto;
}
#editor div{
    height: 500px;
```

```
outline: none;
padding: 10px;
}
</style>
</head>

<body>
<div id="wrap">
<p>
    对齐方式:
    <input type="radio" id="left" name="textAlign" value="left" checked>
    <label for="left">左对齐</label>
    <input type="radio" id="center" name="textAlign" value="center">
    <label for="center">居中对齐</label>
    <input type="radio" id="right" name="textAlign" value="right">
    <label for="right">右对齐</label>
    &nbsp;
    &nbsp;
    &nbsp;
    <input type="checkbox" id="bold" name="fontWeight" value="bold">
    <label for="bold"><b>字体加粗</b></label>
    &nbsp;
    &nbsp;
    &nbsp;
    <button id="lnk">插入超链接</button>
    <button id="img">插入图片</button>
    <br>
    设置字体
    <select id="fontFamily">
        <option value="SimSun" selected>宋体</option>
        <option value="NSimSun" selected>新宋体</option>
        <option value="SimHei" selected>黑体</option>
        <option value="FangSong">仿宋</option>
        <option value="FangSong_GB2312">仿宋_GB2312</option>
        <option value="KaiTi">楷体</option>
        <option value="KaiTi_GB2312">楷体_GB2312</option>
    </select>
    &nbsp; 设置字号
    <select id="fontSize">
        <option selected>12px</option>
        <option>14px</option>
        <option>16px</option>
        <option>18px</option>
        <option>20px</option>
```

```
</select>
    &nbsp; 设置字体颜色:
<select id="color">
    <option selected>black
    <option>silver</option>
    <option>gray</option>
    <option>white</option>
    <option>maroon</option>
    <option>red</option>
    <option>purple</option>
    <option>fuchsia</option>
    <option>green</option>
    <option>lime</option>
    <option>olive</option>
    <option>yellow</option>
    <option>navy</option>
    <option>blue</option>
    <option>teal</option>
    <option>aqua</option>
</select>
</p>
<div id="editor">
    <div contentEditable></div>
</div>
</div>
</body>
</html>
(预览效果: http://gucong3000.github.io/renren/editor.html)
```

80.

[问答题]

请编写一个 JavaScript 函数，它的作用是校验输入的字符串是否是一个有效的电子邮件地址。要求： a) 使用正则表达式。 b) 如果有效返回 true ，反之为 false。

来自：阿里巴巴 2011 前端工程师面试题

参考：

```
function isEmail(emailstr){
    var reg=/[a-zA-Z]+([._\\-]*[a-zA-Z0-9])*@[([a-zA-Z]+[-a-zA-Z]*[a-zA-Z0-9]+.){1,63}[a-zA-Z0-9]+\$/i;
    return reg.test(emailstr);
}
```

81.

[问答题]

尝试实现注释部分的 Javascript 代码，可在其他任何地方添加更多代码（如不能实现，说明一下不能实现的原因）：

```
var Obj = function(msg){  
    this.msg = msg;  
    this.shout = function(){  
        alert(this.msg);  
    }  
    this.waitAndShout = function(){  
        //隔五秒钟后执行上面的 shout 方法  
    }  
}
```

来自：阿里巴巴 2011 前端工程师面试题

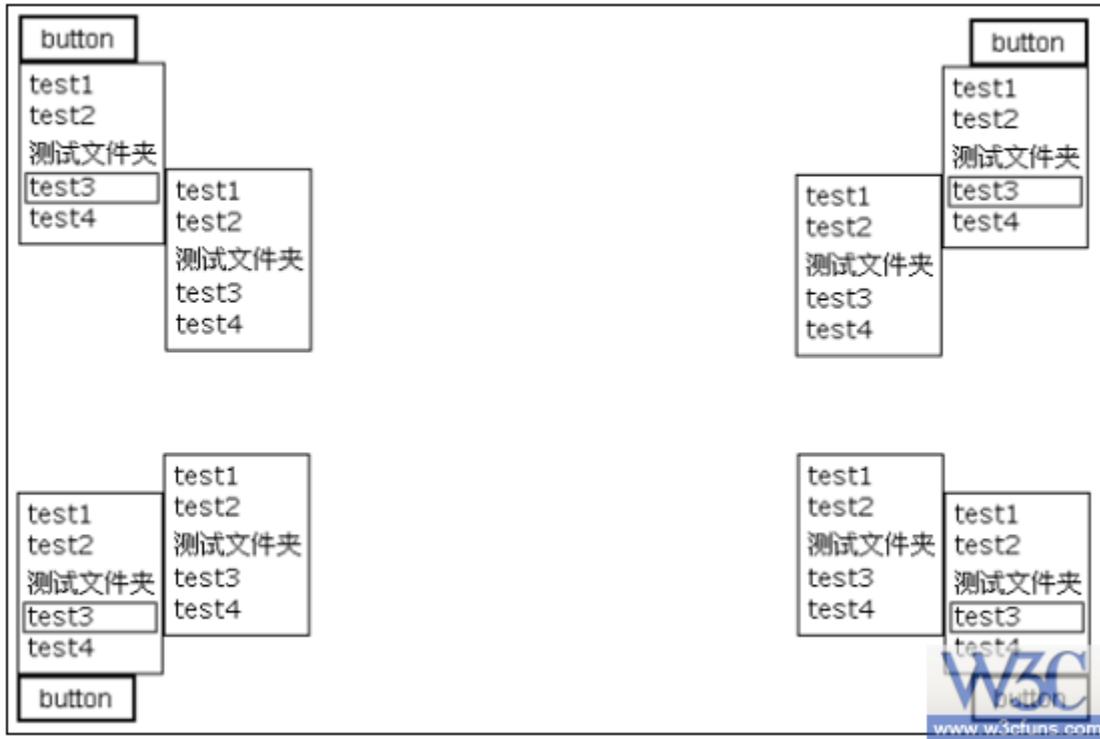
参考：

```
var Obj = function(msg){  
    this.msg = msg;  
    this.shout = function(){  
        alert(this.msg);  
    }  
    this.waitAndShout = function(){  
        var that = this;  
        setTimeout(that.shout, 5000);  
        //隔五秒钟后执行上面的 shout 方法  
    }  
    return this;  
}  
Obj("shouting").waitAndShout();
```

82.

[问答题]

在页面上实现一个二级菜单控件



1. 这个控件可以绑定到页面上的任意一个元素，当点击页面元素出现菜单；
2. 菜单出现的方向根据所在页面的位置自动进行调整，例如：
3. 一级菜单中的元素，鼠标划过后，将会在相应的位置出现二级菜单，二级菜单中的元素点击将会有事件响应。

来自：人人网 2011 前端工程师面试题

参考：无

83.

[问答题]

请编写一个 JavaScript 函数 toRGB，它的作用是转换 CSS 中常用的颜色编码。要求：

```
alert(toRGB("#0000FF"));           // 输出 rgb(0, 0, 255)
alert(toRGB("invalid"));           // 输出 invalid
alert(toRGB("#G00"));              // 输出 #G00
```

来自：阿里巴巴 2011 前端工程师面试题

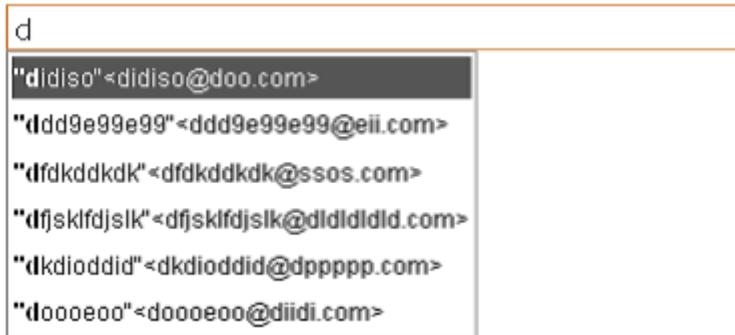
参考：

```
function toRGB(color) {
    var regex = /^#[0-9a-fA-F]{2})([0-9a-fA-F]{2})([0-9a-fA-F]{2})$/;
    match = color.match(regex);
    return      match      ?      'rgb('+parseInt(match[1],      16)+','+parseInt(match[2],
16)+','+parseInt(match[3], 16)+')' : color
}
```

84.

[问答题]

实现 input 输入框的自动匹配



1. 对 input 框中输入的字符进行匹配，将匹配到的内容以菜单的形式展现在 input 框的下方；
2. 只针对英文字母进行匹配，并且匹配到的内容在菜单中加粗；
3. 通过键盘上的上下箭头可以对菜单进行选择，按下回车后将选中的内容写入到 input 框中；

来自：人人网 2011 前端工程师面试题

参考：

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>文本框自动补全</title>
<script src="../script/base.js"></script>
<script>
document.createElement("datalist");

function autocomplete(textBox){
    var $ = window.gucong;
    textBox = document.querySelector(textBox);
    var dataList = document.getElementById(textBox.getAttribute("list"));
    optList = dataList.children;
    dataList = document.createElement("ul");

    $.each(optList, function(){
        var li = document.createElement("li");
        li.setAttribute("label" ,this.getAttribute("label"));
        li.setAttribute("value" ,this.getAttribute("value"));
        dataList.appendChild(li);
    });
}
```

```

textbox.setAttribute("list", "");
$.addClass(datalist, "datalist");
document.documentElement.appendChild(datalist);

textbox.autocomplete="off";
textbox.addEventListener("keydown", function(e){
    switch(e.keyCode) {
        case 38: // up
            e.preventDefault();
            moveSelect(-1);
            break;
        case 40: // down
            e.preventDefault();
            moveSelect(1);
            break;
        case 9: // tab
        case 13: // return
            var opt = datalist.querySelector(".highlight");
            if(opt){
                selectOpt(opt)
                e.preventDefault();
                return false;
            }
            break;
        default:
            window.setTimeout(onChange,0);
    }
}, true);

textbox.addEventListener("input", onChange, false);
textbox.addEventListener("blur", function(){
    datalist.style.display = "none";
}, false);

function onChange() {
    if(textbox.value){
        optList = [];
        var match = new RegExp( textbox.value , "ig"),
        pos = $.findPos(textbox),
        count = 0;
        $.each(datalist.children, function(){
            var str = this.getAttribute("label");
            if(match.test(str)){
                this.innerHTML = str.replace(match, function(str){

```

```

        return "<b>" + str + "</b>"
    });
    this.style.display = "block";
    optList[count] = this;
    count++;
} else{
    this.style.display = "none";
}
});
datalist.style.display = count ? "block" : "none";
datalist.style.left = pos.x + "px";
datalist.style.top = pos.y + textbox.offsetHeight + "px";
} else {
    datalist.style.display = "none";
}
}

function filter(elm){
    return /li/i.test(elm.tagName) ? elm : null;
}

function moveSelect(setp){
    var sel = datalist.querySelector(".highlight"),
    index;
    if(sel){
        $.each(optList,function(i){
            if(sel == this){
                index = i;
                return false;
            }
        });
    } else {
        index = optList.length * -setp;
    }
    index += setp
    if(index < 0){
        index = 0;
    } else if(index >= optList.length){
        index = optList.length - 1;
    }
    hoverOpt(optList[index]);
}

function selectOpt(elm){

```

```

elm = filter(elm);
if(elm){
    textbox.value = elm.getAttribute("value");
}
datalist.style.display = "none";
}

function hoverOpt(elm){
    elm = filter(elm);
    $.each(datalist.children, function(){
        if(this == elm){
            $.addClass(this, "highlight");
        }else{
            $.removeClass(this, "highlight");
        }
    });
}

datalist.addEventListener("mouseover", function(e){
    hoverOpt(e.target);
});

datalist.addEventListener("click", function(e){
    selectOpt(e.target);
});
}

document.realy(function($){
    autocomplete("input");
});
</script>
<link href="style/base.css" rel="stylesheet" type="text/css">
<style type="text/css">
#wrap {
    padding: 100px;
}
.datalist {
    -webkit-user-select: none;
    -moz-user-select: none;
    user-select: none;
    box-shadow: 2px 2px 2px #777;
    border: 1px solid black;
    background: window;
    position: absolute;
}

```

```
        overflow: hidden;
        cursor: default;
        color: menutext;
        z-index: 99999;
        display: none;
        font: menu;
        padding: 0;
        margin: 0;
        behavior: url("PIE.htc");
    }
:root .datalist {
    behavior: none;
    -webkit-box-shadow: 2px 2px 2px rgba(0,0,0,.5);
    -moz-box-shadow: 2px 2px 2px rgba(0,0,0,.5);
    box-shadow: 2px 2px 2px rgba(0,0,0,.5);
}
.datalist li {
    line-height: 20px;
    list-style: none;
    font-size: 14px;
    padding: 0 10px;
    display: block;
}
.datalist li.highlight {
    background: highlight;
    color: highlighttext;
}
</style>
</head>

<body>
<div id="wrap">
    <input type="email" list="url_list" autofocus>
    <datalist id="url_list">
        <option label="gucong@gmail.com" value="mailto:gucong@gmail.com" />
        <option label="gucongbbs@163.com" value="mailto:gucongbbs@163.com" />
        <option value="mailto:gucong.student@sina.com" />
            <option label="gucong@student@sina.com" />
        <option value="mailto:gucong520@hotmail.com" />
            <option label="gucong520@hotmail.com" />
        <option label="gucong@foxmail.com" value="mailto:gucong@foxmail.com" />
        <option label="35803719@qq.com" value="mailto:35803719@qq.com" />
        <option label="64272001@qq.com" value="mailto:64272001@qq.com" />
    </datalist>
```

```
<br>
请输入 gucong 来作为测试数据
</div>
</body>
</html>
(预览效果: http://gucong3000.github.io/renren/autocomplete.html)
```

85.

[问答题]

阅读以下 JavaScript 代码:

```
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);
        });
    };
}
```

请阐述 a) 代码的功能; b) 代码的优点和缺点; c) `listener.apply(el)` 在此处的作用; d) 如果有可改进之处, 请给出改进后的代码, 并说明理由。

来自: 阿里巴巴 2011 前端工程师面试题

参考:

- 1、事件绑定的一个简单函数封装;
- 2、优点: 函数封装可随时引用, 没有解决兼容性的问题; 缺点: 代码冗余
- 3、作用是改变 `this` 指向的问题, 不用则指向 `window` 而不是调动它的事件对象

```
function bind(obj, evname, fn){
    if(obj.addEventListener){
        obj.addEventListener(evname, fn, false);
    } else {
        obj.attachEvent('on'+evname, function(){
            fn.apply(obj);
        });
    }
}
```

86.

[问答题]

用 js、html、css 实现一个弹出提示控件

1. 分别实现类似于系统的 alert、confirm、prompt 对话框；
 2. 对话框大小根据提示内容进行自适应（有一个最小宽高），默认出现在页面的水平垂直居中的位置；
 3. 对话框可拖动；
 4. 对话框中的事件模拟系统对话框的事件（例如：alert 对话框，点击确定按钮，对话框消失）；
 5. 解决 IE6 被 select 控件遮挡的问题。
-

来自：人人网 2011 前端工程师面试题

参考：

```
<!DOCTYPE HTML>
<html class="as">
<head>
<meta charset="utf-8">
<title>对话框</title>
<!--[if lt IE 9]>
<script src="../script/base.js"></script>
<![endif]-->
<link href="style/base.css" rel="stylesheet" type="text/css">
<style type="text/css">
#wrap {
    padding: 100px;
}
#dialogbg {
    background: rgba(0,0,0,.2);
    overflow: hidden;
    position: fixed;
}
#dialogbg,
#dialogbg .bgcolor {
    height: 100%;
    width: 100%;
    margin: 0;
    left: 0;
    top: 0;
}
#dialogbg .bgcolor {
    filter: alpha(opacity=20);
```

```

background: #000;
position: absolute;
}
:root #dialogbg .bgcolor {
    display: none;
}
* html #dialogbg {
    _position: absolute;
    _top: expression(offsetParent.scrollTop);
}
#dialogbg .dialog {
    -webkit-border-radius: 5px 5px 0 0;
    -moz-border-radius: 5px 5px 0 0;
    border-radius: 5px 5px 0 0;
border: 1px solid #666;
display: inline-block;
position: absolute;
background: white;
text-align: left;
overflow: hidden;
margin: auto;
left: 50%;
top: 50%;
behavior: url("PIE.htc");
*behavior: none;
}
.dialog .title {
    -webkit-user-select: none;
    -moz-user-select: none;
    user-select: none;
    filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr=#00aaee,endColorstr=#0066cc);
background: #0095cd;
background: -webkit-gradient(linear, left top, left bottom, from(#00aaee), to(#0066cc));
background: -webkit-linear-gradient(top, #00aaee, #0066cc);
background:      -moz-linear-gradient(top, #00aaee, #0066cc);
background:      -ms-linear-gradient(top, #00aaee, #0066cc);
background:      -o-linear-gradient(top, #00aaee, #0066cc);
background:          linear-gradient(top, #00aaee, #0066cc);
text-shadow: 0 0 1px black;
font-weight: bold;
line-height: 28px;
overflow: hidden;
cursor: move;

```

```
color: white;
*width: expression(this.parentNode.clientWidth);
}

.dialog .title div {
    padding: 0 8px;
}

.dialog .text,
.dialog .content {
    margin: 20px;
}

.dialog .text{
    text-align: center;
}

.dialog .content {
    word-break: keep-all;
    white-space: nowrap;
}

.dialog input,
.dialog .content {
    min-width: 200px;
    _width: 200px;
}

.dialog input {
    -webkit-box-sizing:border-box;
    -moz-box-sizing:border-box;
    -ms-box-sizing:border-box;
    box-sizing:border-box;
    display: block;
    width: 100%;
    *width: auto;
}

.dialog .btns {
    text-align: center;
    margin: 10px 5px;
}

.dialog .btns * {
    border: 1px solid #CCC;
    text-decoration: none;
    background: none;
    padding: 5px 20px;
    margin: 0 5px;
    color: #000;
}

#dialogbg, .dialog,.dialog .winbtns .size,.dialog input {
```

```
/*display: none ;*/
}

#dialogbg, .dialog {
    display: block;
}

</style>
</head>

<body>
<div id="wrap">
    <select class="select">
        <option>看看有没有遮住对话框</option>
        <option>看看有没有遮住对话框</option>
        <option>看看有没有遮住对话框</option>
    </select>
    <br>
    请点击下面的链接弹出对话框
    <br>
    <a href="javascript:msgbox('你好， 欢迎使用 Windows 8 旗舰版')">alert</a>
    <br>
    <a href="javascript:msgbox(' 您 确 定 要 关 闭 计 算 机
吗?',function(t){alert(t)})">confirm</a>
    <br>
    <a href="javascript:msgbox(' 请 输入 您 的 姓 名 ',function(t){alert(t)},' 匿 名
')">prompt</a>
</div>
<script>
function msgbox(msg, fun, text){
    var dialogbg = document.querySelector("#dialogbg");
    if(!dialogbg){
        dialogbg = document.createElement("div");
        dialogbg.id = "dialogbg";
        document.body.appendChild(dialogbg);
    }
    dialogbg.innerHTML = '<div class="bgcolor"></div><div class="dialog"><div class="title"
unselectable="on"><div>消息</div></div><div class="content"></div><div class="text"><input
type="text"></div><div class="btns"><button class="ok">确定</button><button class="cancel">
取消</button></div></div>';
    dialogbg.querySelector(".content").innerHTML = msg || " ";
    function hide( sel ){
        dialogbg.querySelector(sel).style.display = "none";
    }
    var btnOk = dialogbg.querySelector(".ok"),

```

```

btnCancel = dialogbg.querySelector(".cancel");
if(fun){
    if(text == undefined){
        hide(".text");
        btnOk.onclick = function(){
            fun(true);
        }
        btnCancel.onclick = function(){
            fun(false);
        }
    }else{
        var textbox = dialogbg.querySelector("input");
        textbox.value = text;
        btnOk.onclick = function(){
            fun.call(textbox, textbox.value);
        }
    }
}else{
    hide(".cancel");
    hide(".text");
}

function btnClose(){
    dialogbg.style.display = "none";
}

btnOk.addEventListener("click", btnClose, true);
btnCancel.addEventListener("click", btnClose, true);

dialogbg.style.display = "block";

var dialog = dialogbg.querySelector(".dialog");
dialog.style.left = (dialogbg.offsetWidth - dialog.offsetWidth) / 2 + "px";
dialog.style.top = (dialogbg.offsetHeight - dialog.offsetHeight) / 2 + "px";
drag(dialog, dialog.querySelector(".title"));
}

function drag(dialog, titleBar){
    var onDrag,
        pos = {},
        dialogbg = document.querySelector("html");

    titleBar.addEventListener("mousedown", function(e){
        pos.left = dialog.offsetLeft;
        pos.top = dialog.offsetTop;
    })
}

```

```

        pos.x = e.pageX;
        pos.y = e.pageY;
        onDrag = true;
    }, false);

dialogbg.addEventListener("mousemove", function(e){
    if(onDrag){
        dialog.style.left = (pos.left + e.pageX - pos.x) + "px";
        dialog.style.top = (pos.top + e.pageY - pos.y) + "px";
    }
}, false);

titleBar.onselectstart = function(){
    return false;
}

dialogbg.onmouseup = function(){
    onDrag = false;
}
}

</script>
</body>
</html>

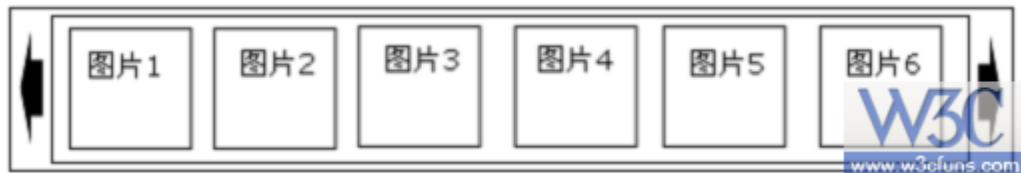
```

(预览效果: <http://gucong3000.github.io/renren/dialog.html>)

87.

[问答题]

在页面的固定区域内实现图片的展示(使用原生代码实现, 不可使用任何框架)。



1. 每点击一次右箭头, 图片区域向左滚动出一张图片, 反之相同
2. 当发现图片滚动到末尾时, 响应的箭头变成不可点击状态
3. 鼠标在图片区域内滑动滚轮, 图片会随着鼠标滚轮的方向进行响应的滚动

来自：人人网 2011 前端工程师面试题

参考：

```

<!DOCTYPE HTML>
<html>
<head>
```

```

<meta charset="utf-8">
<title>缩略图滚动显示</title>
<script src="../script/base.js"></script>
<script>
document.realy(function($){
    var thumbs = document.querySelector(".thumbs"),
        left = thumbs.querySelector(".scroll_left"),
        right = thumbs.querySelector(".scroll_right"),
        cont = thumbs.querySelector("ul"),
        items = cont.querySelectorAll("li");
    //console.log(scrollStep);

    function scrollFun(val){
        val = cont.scrollLeft += (items[1].offsetLeft - items[0].offsetLeft) * val;
        if(val<=0){
            $.addClass(left, "disabled");
        }else if (val >= (cont.scrollWidth - cont.clientWidth)){
            $.addClass(right, "disabled");
        }
    }

    function sLeft(){
        scrollFun(-1);
        $.delClass(right, "disabled");
        return false;
    }

    function sRight(){
        scrollFun(1);
        $.delClass(left, "disabled");
        return false;
    }

    left.onclick = sLeft;
    right.onclick = sRight;

    cont.addEventListener("onmousewheel" in cont ? "mousewheel" : "DOMMouseScroll",
function(e){
    if(( e.detail || -e.wheelDelta) > 0){
        sRight();
    }else{
        sLeft();
    }
    e.preventDefault();
}

```

```
        return false;
    },true);
});
</script>
<link href="style/base.css" rel="stylesheet" type="text/css">
<style type="text/css">
#wrap {
    padding-top: 100px;
}
.thumbs {
    overflow: hidden;
    margin: 0 100px;
}
.thumbs ul {
    word-break: keep-all;
    white-space: nowrap;
    overflow: hidden;
    margin: auto;
    font-size: 1px;
    _width: expression(document.documentElement.clientWidth-280);
}
.thumbs li {
    list-style: none;
    display: inline;
}
.thumbs a {
    border: 2px solid #B7B7B7;
    text-decoration: none;
    display: inline-block;
    outline: none;
    margin: 0 9px;
}
.thumbs img {
    vertical-align: top;
    width: 115px;
    height: 85px;
}
.thumbs a:hover {
    border-color: #65B6DD;
}
.thumbs a:focus,
.thumbs a:active {
    border-color: #0084B4;
}
```

```
.thumbs .scroll_left,  
.thumbs .scroll_right {  
    -webkit-user-select: none;  
    -moz-user-select: none;  
    user-select: none;  
    font-family: SimHei;  
    background: #65B6DD;  
    text-align: center;  
    line-height: 85px;  
    overflow: hidden;  
    font-size: 20px;  
    cursor: pointer;  
    height: 85px;  
    width: 20px;  
    color: #fff;  
    behavior: url("PIE.htc");  
}  
.thumbs .scroll_left {  
    -webkit-border-radius: 10px 3px 3px 10px;  
    -moz-border-radius: 10px 3px 3px 10px;  
    border-radius: 10px 3px 3px 10px;  
    margin-right: 15px;  
    float: left;  
}  
.thumbs .scroll_right {  
    -webkit-border-radius: 3px 10px 10px 3px;  
    -moz-border-radius: 3px 10px 10px 3px;  
    border-radius: 3px 10px 10px 3px;  
    margin-left: 15px;  
    float: right;  
}  
* html .thumbs .scroll_left,  
* html .thumbs .scroll_right {  
    _margin: 0;  
}  
.thumbs .disabled {  
    cursor: not-allowed;  
    background: #CCC;  
    color: #999;  
}  
</style>  
</head>  
  
<body>
```

```

<div id="wrap">
    <div class="thumbs" unselectable="on">
        <a class="scroll_left disabled"><--</a>
        <a class="scroll_right">--></a>
        <ul>
            <li><a href="#"></a></li>
            <li><a href="#"></a></li>
        </ul>
    </div>
</div>
</body>
</html>
(效果预览: http://gucong3000.github.io/renren/thumbs.html)

```

88.

[单选题]

JavaScript 中 document.getElementById 的返回值的类型为?

- A.Array
- B.Object
- C.String
- D.Function

来自：微博

答案：B

89.

[问答题]

在网页里显示一个 `div` 浮层，位于网页正中，该浮层内的文本显示用户电脑当前时间，格式 `YYYY-MM-DD hh:mm:ss`，如 `2013-08-16 10:22:05`。浮层居中可以使用 JavaScript 或者 CSS 实现。

来自：腾讯

参考：

```
<!DOCTYPE html>
<html>
<head lang="ch">
    <meta charset="UTF-8">
    <title></title>
    <style>
        html,body{width: 100%; height: 100%; margin: 0;}
        .container {display:table; width: 100%; height: 100%; text-align:center;}
        .time{vertical-align:middle; display:table-cell;}
    </style>
</head>
<body>
    <div class="container">
        <span id="time" class="time"></span>
    </div>
</body>
<script>
    function TwoDigit(args){
        return args < 10 ? "0" + args : args;
    };
    var String
    var dom = document.getElementById("time")
    with( new Date() ) {
        String = getFullYear() + "年" + TwoDigit(getMonth() + 1) + "月" +
        TwoDigit(getDate()) + "日" + TwoDigit(getHours()) + "时" + TwoDigit(getMinutes()) + "分"
        "+TwoDigit.getSeconds()) + "秒";
    };
    dom.innerHTML = String;
</script>
</html>
```

90.

[问答题]

请用 JavaScript 实现，控制一个文本框只能输入正整数，如输入不符合条件则文本框全部字体标红。要求写出完整的文本框 HTML 代码和 JavaScript 逻辑代码。

来自：腾讯

参考：

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>无标题文档</title>

        <script type="text/javascript">
        <!--
            function isNum() {
                var txtField = document.getElementById('txt');
                var txt = txtField.value;
                var regexp = /^\d*$/;
                if (regexp.test(txt)) {
                    txtField.style.color = "#000";
                } else {
                    txtField.style.color = "#f00";
                }
            }
        //-->
        </script>
    </head>

    <body>
        <input id="txt" type="text" onKeyUp="isNum();"/>
    </body>
</html>
```

91.

[问答题]

以下代码输出的值为？（ ）

```
function f1() {
    var n = 100;
    nAdd = function() {
```

```
n += 1
}
function f2() {
    alert(n);
}
return f2;
}
var result = f1();
result();
nAdd();
result();
```

来自：搜狐

参考：

100

undefined

101

92.

[问答题]

如何判断浏览器是 IE 还是火狐，用 ajax 实现。

来自：阿里巴巴

参考：

用 Request Headers 里的 User-Agent 判断

93.

[问答题]

请使用原生 js 实现一个 div 可拖拽，需要考虑浏览器兼容性。

来自：阿里巴巴

参考：

```
var drag1=document.getElementById('drag1');
drag1.onmousedown=function(ev){
    var x=ev.clientX-drag1.offsetLeft;
    var y=ev.clientY-drag1.offsetTop;
    drag1.onmousemove=function(ev){
        drag1.style.left=ev.clientX-x+'px';
```

```
drag1.style.top=ev.clientY-y+'px';
}
drag1.onmouseup=function(ev){
drag1.onmousemove=drag1.onmouseup=null;
}
}
```

94.

[问答题]

请填充代码，使 mySort()能使传入的参数按照从小到大的顺序显示出来。

```
function mySort() {
    var tags = new Array();//使用数组作为参数存储容器
    请补充你的代码
    return tags;//返回已经排序的数组
}
var result = mySort(50,11,16,32,24,99,57,100);//传入参数个数不确定
console.info(result);//显示结果
```

来自：腾讯 2015 春招 web 前端开发练习卷

参考：

```
function mySort() {
    var tags = new Array();//使用数组作为参数存储容器
    tags = Array.prototype.slice.call(arguments)
    tags.sort(function(pre,next){
        return pre - next;
    })
    return tags;//返回已经排序的数组
}
```

95.

[单选题]

下面哪些是 NodeJS 官方模块？

- A.Querystring
- B.Request
- C.Async
- D.Dns

来自：阿里巴巴

答案：D

96.

[问答题]

用 javascript 实现控制一个文本框的输入字数限制，超出字数限制文本框飘红显示。

来自：百度

参考：

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>textarea字数限制</title>
6 <script>
7     window.onload=function(){
8         document.getElementById("tBox").onkeydown=function(){
9             if(this.value.length>20){
10                 var str="字数超过限制了！";
11                 document.getElementById("maxLength").innerHTML=str;
12                 document.getElementById("maxLength").style.color="#ff0000";
13             }
14         }
15     }
16 </script>
17 </head>
18 <body>
19     <textarea id="tBox">ddd</textarea>
20     <div id="maxLength" style="font-size:12px;">最多输入20个字！</div>
21 </body>
22 </html>
23
24
```

97.

[问答题]

使用 Javascript 打印出 1-10000 之间的所有对称数（例如 121 1331 等）。

来自：百度

参考：

```
function findSymmetryNum(s,o){
    var arr=[];
    for(var i=s;i<=o;i++){
        var str=+i,s1=str.length,middle=0,flag=true;
        if(s1%2==0){
            middle=s1/2;
        }else{
            middle = (s1-1)/2;
        }
        for(var m=0;m<middle;m++){
            if(str.substr(0+m,1)!=str.substr(-1-m,1)){
                flag = false;
            }
        }
        if(flag&&arr.push(i));
    }
}
```

```
    }
    console.log(arr);
    return arr;
}
findSymmetryNum(1,1001);
```

98.

[问答题]

使用 javascript 实现，将文档中 className 有 “test” 的 li 标签背景色设为黄色。

来自：百度

参考：

```
<script type="text/javascript">
window.onload=function(){
    var list = document.getElementsByTagName("li");
    console.log(list.length);
    for(var i=0;i<list.length;i++){
        if(list[i].className=='test'){
            list[i].style.backgroundColor="gold";
        }
    }
}
</script>
```

99.

[问答题]

用 js 实现如下功能，将给定的数字转化成千分位的格式，如把“10000”转化成“10,000”，并考虑到性能方面的因素

来自：百度

参考：

```
<!DOCTYPE>
<html>
<head>
<script>
var a="150355660";
var re=(?=(!\b)(\d{3})+\$)/g
alert(a.replace(re','));
</script>
```

```
</head>
<body>
</body>
</html>
```

100.

[问答题]

请通过 js 来实现一个函数 once，这个函数在整个应用运行的时候只被访问一次。如果再次访问就会访问上次的执行结果。

来自：百度

参考：

```
var singleton = (function() {
    //缓存实例
    var instance;
    var randomNum = Math.random();
    //单例初始化代码
    function init() {
        return randomNum;
    }
    //如果没有初始化，则初始化，否则返回已经执行的结果。
    if (!instance) {
        instance = init();
    }
    return instance;
})()
console.log(singleton);
console.log(singleton);
```

101.

[问答题]

如果要把在手机百度的搜索结果做成一个可以无限向上滑动的卡片列表(图就不展示了，就是一行是一块内容这种形式)，请你设计这个可以无限滑动的卡片列表方案，注意不能造成浏览器奔溃。

来自：百度

参考：

可以用瀑布流布局来实现

```
var lock = false,
    maxId = 0;

function addMore() {
    if (!lock) {
        lock = true;
        $.ajax({
            url: 'www.xxx.com/test.php?maxId=' + maxId,
            type: 'GET',
            dataType: 'json',
            success: function(data) {
                var buffer = [],
                    m = 0;
                buffer[m++] = data.something1;
                buffer[m++] = data.something2;
                //otherthings....
                var temp = buffer.join("");
                $(temp).appendTo('div');
                maxId++;
            }
        })
        lock = false;
    }
}
$(window).scroll(function() {
    if (($(window).height() + $(window).scrollTop()) >= $('body').height()) {
        addMore();
    }
});
```

102.

[问答题]

在用户填写表单内容时，为了避免误操作，有时会在用户“试图关闭”网页时进行警告，让用户确认数据未提交会丢弃。请尽可能列举，你所知道的，用户导致这种影响的“误操作行为”有哪些，并适当归类。

来自：百度

参考：无

103.

[问答题]

如何解决跨域通信的问题，简述有哪些方法？

来自：百度

参考：

- (1)、document.domain+iframe 的设置
- (2)、动态创建 script，要就是 jsonp
- (3)、利用 iframe 和 location.hash
- (4)、window.name 实现的跨域数据传输
- (5)、使用 HTML5 postMessage
- (6)、利用 flash

104.

[问答题]

实现点击表头排序功能，点一次降序，再点一次升序。

来自：百度

参考：无

105.

[问答题]

实现一个监听 load 事件的接口 window.load(callback):多次调用时保证执行顺序，先绑定的回调先执行：如果 load 事件已触发，调用时会直接执行该回调。

来自：美团

参考：

```
window.load = (function(){  
    var loaded = false,  
        func = new Array();  
  
    window.onload = function(){  
        loaded = true;  
        for(var i = 0;i<func.length;i++){  
            func[i]();  
        }  
    }  
})()
```

```
        }
        return function(callback){
            if(typeof callback!="function") return;
            if(!loaded) {
                callback();
            }else{
                func.push(callback)
            }

        }
    }()
}
```

106.

[问答题]

画图描述 CSS 盒模型，用 JS 实现获取元素宽和位置，注意兼容性。

来自：去哪儿

参考：

```
var element = document.getElementById("div0");
function getWeizhi (element) {
    var top = document.documentElement.clientTop;
    var left = document.documentElement.clientLeft;
    var ele = element.getBoundingClientRect();
    return{
        top:ele.top - top,
        right:ele.right - left,
        bottom:ele.bottom - top,
        left:ele.left - left,
        width:ele.right - ele.left,
        height:ele.bottom - ele.top
    }
}
```

107.

[问答题]

用 Javascript 实现乱序函数 randomSort(array)函数，输出排序后的函数。如[1,2,3,4,5]，输出[3,2,4,5,1]。要求 N 次以内不重复。

来自：去哪儿

参考：

```
function randomSort(array){  
    var x=array.sort(function(a,b){  
        return Math.random()>0.5?1:-1;  
    });  
    return x;  
}  
  
或  
  
function randomSort(array) {  
    var n = array.length, t, i;  
    while (n) {  
        i = Math.random() * n-- | 0;  
        t = array[n];  
        array[n] = array[i];  
        array[i] = t;  
    }  
    return array;  
}
```

108.

[问答题]

某饭店要开发一个排队软件。非 VIP 用户需要排队，先到先得。VIP 用户可以插队，但是 VIP 用户之间需要按到达时间先后排队。

要求实现：①addCustomer(String phoneNumber)函数②有空位时获取排到的用户③用户进店排队时的函数 current()

来自：去哪儿

参考：使用 Python

class queue:

```
    def __init__(self,vip,nor):  
        if(type(vip)==list and type(nor)==list):  
            self.vip = vip  
            self.nor = nor  
        else :print 'vip and nor must be list'  
    def addcustom(self,name,phonenumber,ifvip):  
        if (ifvip):  
            self.vip.append([name,phonenumber])  
        else :  
            self.nor.append([name,phonenumber])  
    def next(self):  
        if (self.vip):  
            n = self.vip[0][0]
```

```

        del(self.vip[0])
        print 'next is ' + n
    else :
        n = self.nor[0][0]
        del(self.nor[0])
        print 'next is ' + n
def current(self):
    str1 = '现在队列为：'
    for i in self.vip:
        str1 += i[0]
        str1 += ','
    for i in self.nor:
        str1 += i[0]
        str1 += ','
    print str1

```

109.

[问答题]

写一个命令行字符的解析函数。

例： -name lily -age 25 -school "chengdu university" 返回的是[-name lily,-age 25, -school "chengdu university"]

来自：去哪儿

参考：

```

<script>
function getdata(str)
{
    var json={};
    var gets =str.split('\"') [0];
    gets =gets.split(' ');
    for(var i=0;i<gets.length-1;i++)
    {
        if(i%2==0)
        {
            json[gets[i]]=gets[i+1];
        }
    }
    json[gets[gets.length-1]]=str.split('\"') [1];
    return json;
}
</script>

```

110.

[问答题]

写一个函数 padstare(string str1,min_length,string str2)。

例： padstare ('5', 3, '0') 返回的是 '005'; padstare ('798', 5, '0') 返回的是 '798'；

来自：去哪儿

参考：

```
<script>
function padstare(str,min,str_char)
{
    var ret=[];
    if(str.length>=min)
    {
        return str;
    }
    else
    {
        for(var i=0;i<min-str.length;i++)
        {
            ret.push(str_char);
        }
        return ret.join("")+str;
    }
}
</script>
```

111.

[问答题]

请实现 javascript 中的 indexOf 功能，判断一个字符串 a 中是否包含另一个字符串 b。

- a) 如果包含，需要返回匹配字符串 b 的位置
- b) 如果不包含，需要返回-1

例如 indexOf("hello","el") returns 1;

来自：去哪儿

参考：

```
function indexOf (a,b){
    var result = a.match(b);
    return result? result.index:-1;
```

```
}

console.log(indexOf("hello", "el"));//1

或

function indexOf(strA, strB) {
    var lenA = strA.length,
        lenB = strB.length;
    if (lenA < lenB) {
        return -1;
    } else if (lenA == lenB) {
        return 0;
    } else {
        for (var j = 0; j < lenA; j++) {
            if (strA.charAt(j) == strB[0] && strA.substr(j, lenB) == strB) {
                return j;
            }
        }
        return -1;
    }
}
console.log(indexOf("hello", "el")); //1
```

112.

[问答题]

angularjs 里面的双向数据绑定和依赖注入的原理?

来自：阿里巴巴

参考：

双向数组绑定主要是通过 angular 的 digest 流程，依赖注入主要是 function.toString 分析

参见：<https://github.com/xufeil/blog/issues/10>

113.

[问答题]

javascript 里面的继承怎么实现，如何避免原型链上面的对象共享。

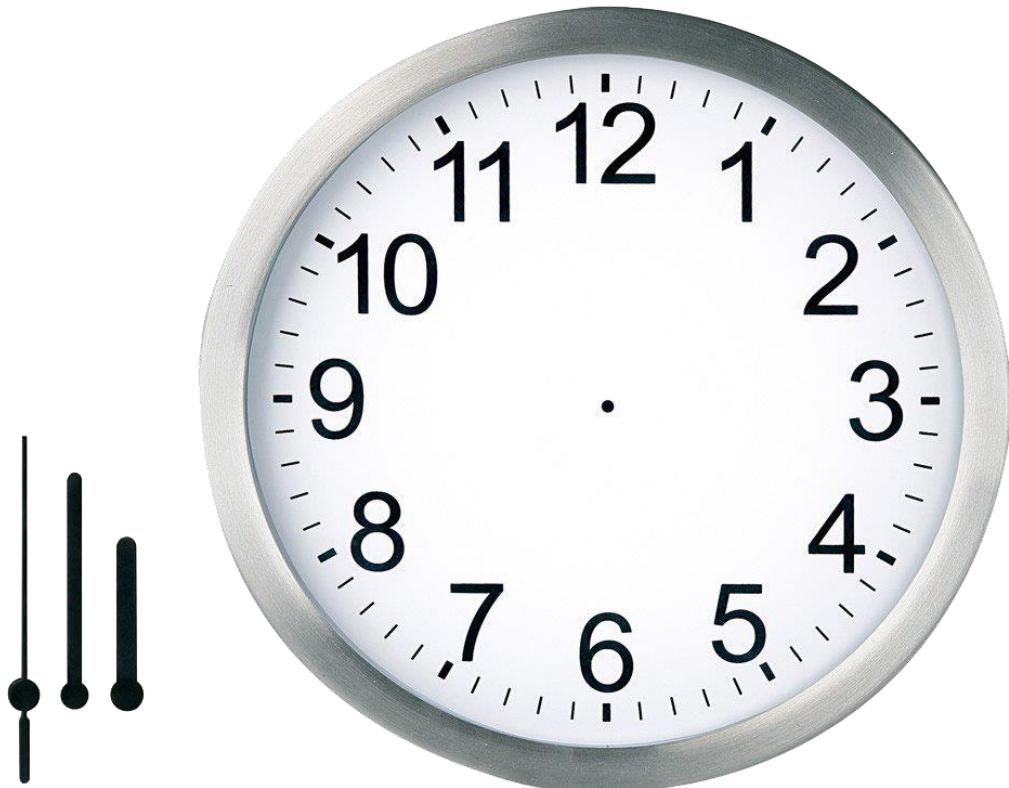
来自：阿里巴巴

参考：

114.

[问答题]

网页时钟 第一问：使用 HTML, CSS 绘制一个时钟，效果图和素材如下图（注意指针可以旋转到任意位置）



第二问：用 Javascript 编写一个 Clock 类，实现如下接口：

1. 构造函数 Clock(HTMLDivElement dom)：在参数 dom 中生成上述时钟
2. setTime(hour,minute,second)：设置时钟时间，指针指定到正确位置
3. run()时钟开始工作 可以使用 jQuery 等任何前端框架

来自：大众点评

参考：

第一问：

html：

```
<div class="clock">
  <div class="clock-panel"></div>
  <div class="clock-hour"></div>
  <div class="clock-mini"></div>
  <div class="clock-second"></div>

</div>
```

CSS:

```
.clock {  
    position: relative;  
    margin: 50px auto;  
    width: 800px;  
  
}  
.clock .clock-panel {  
    position: absolute;  
    left:0;  
    right:0;  
    background: url("clock.png");  
    background-size: 1024px;  
    background-position: -188px -50px;  
    background-repeat: no-repeat;  
    z-index: 1;  
    height: 800px;  
    width: 800px;  
}  
.clock .clock-hour {  
    /* display:none; */  
    background: url("clock.png");  
    background-size: 1024px;  
    position: absolute;  
    transform-origin: 15px 145px;  
    z-index: 2;  
    height: 161px;  
    width: 31px;  
    left: 375px;  
    top: 250px;  
    background-position: -120px -558px;  
}  
.clock .clock-mini {  
    /* display:none; */  
    background: url("clock.png");  
    background-size: 1024px;  
    position: absolute;  
    z-index: 3;  
    height: 220px;  
    width: 28px;  
    background-position: -73px -498px;  
    transform-origin: 14px 205px;  
    left: 376px;  
    top: 185px;
```

```
}

.clock .clock-second {
    /* display:none; */
    background: url("clock.png");
    background-size: 1024px;
    position: absolute;
    z-index: 4;
    height: 321px;
    width: 31px;
    background-position: -26px -466px;
    transform-origin: 15px 240px;
    left: 375px;
    top: 153px;
}
```

第二问：

```
function Clock(dom){
    this.domClock = document.createElement('div');
    this.domPanl = document.createElement('div');
    this.domHour = document.createElement('div');
    this.domMini = document.createElement('div');
    this.domSecond = document.createElement('div');
    this.domClock.className = "clock";
    this.domPanl.className = "clock-panel";
    this.domHour.className = "clock-hour";
    this.domMini.className = "clock-mini";
    this.domSecond.className = "clock-second";
    this.domClock.appendChild(this.domPanl);
    this.domClock.appendChild(this.domHour);
    this.domClock.appendChild(this.domMini);
    this.domClock.appendChild(this.domSecond);
    dom.appendChild(this.domClock);
    return this;
}

Clock.prototype.setTime = function(hour, minute, second){
    this.hour = hour;
    this.minute = minute;
    this.second = second;
    this.domHour.style.transform = "rotate(" + (hour * 30 + 0.5 * minute) + "deg)";
    this.domMini.style.transform = "rotate(" + minute * 6 + "deg)";
    this.domSecond.style.transform = "rotate(" + second * 6 + "deg)";
}

Clock.prototype.run = function(){
    var self = this;
```

```
var clockId = setInterval(function(){
    var now = new Date(0, 0, 0, self.hour, self.minute, self.second + 1);
    self.second = self.second + 1;
    self.domHour.style.transform = "rotate(" + (now.getHours() * 30 + 0.5 * now.getMinutes()) + "deg)";
    self.domMini.style.transform = "rotate(" + now.getMinutes() * 6 + "deg)";
    self.domSecond.style.transform = "rotate(" + now.getSeconds() * 6 + "deg)";
}, 1000);
this.domHour.style.display = "block";
this.domMini.style.display = "block";
this.domSecond.style.display = "block";
})
```

或

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>时钟</title>
    <style type="text/css">
        html,body,div{
            margin: 0;
            padding: 0;
        }
        .clock-wrap{
            width: 300px;
            height: 300px;
            margin:20px auto;
            background: url("clock1.png") -75px -20px no-repeat;
            position: relative;
        }
        .clock-wrap>div{
            position: absolute;
            width: 12px;
            height: 100px;
            background-color: #ccc;
            top:80px;
            left: 50%;
            margin-left: -1px;
            /* -webkit-transition: all 0.5s ease-in-out;
            -moz-transition: all 0.5s ease-in-out;
            -ms-transition: all 0.5s ease-in-out;
            -o-transition: all 0.5s ease-in-out;
            transition: all 0.5s ease-in-out; */
        }
    </style>
</head>
<body>
</body>
```

```
#sec{
    top:60px;
    height: 128px;
    background: url('clock1.png') -10px -182px no-repeat;
    -webkit-transform-origin: center 94px;
    -moz-transform-origin: center 94px;
    -ms-transform-origin: center 94px;
    -o-transform-origin: center 94px;
    transform-origin: center 94px;
    -webkit-transform: rotateZ(30deg);
    -moz-transform: rotateZ(30deg);
    -ms-transform: rotateZ(30deg);
    -o-transform: rotateZ(30deg);
    transform: rotateZ(30deg);
}
#min{
    height: 88px;
    background: url('clock1.png') -28px -202px no-repeat;
    -webkit-transform-origin: center 75px;
    -moz-transform-origin: center 75px;
    -ms-transform-origin: center 75px;
    -o-transform-origin: center 75px;
    transform-origin: center 75px;
    -webkit-transform: rotateZ(145deg);
    -moz-transform: rotateZ(145deg);
    -ms-transform: rotateZ(145deg);
    -o-transform: rotateZ(145deg);
    transform: rotateZ(145deg);
}
#hour{
    height: 84px;
    background: url('clock1.png') -47px -202px no-repeat;
    -webkit-transform-origin: center 75px;
    -moz-transform-origin: center 75px;
    -ms-transform-origin: center 75px;
    -o-transform-origin: center 75px;
    transform-origin: center 75px;
    -webkit-transform: rotateZ(60deg);
    -moz-transform: rotateZ(60deg);
    -ms-transform: rotateZ(60deg);
    -o-transform: rotateZ(60deg);
    transform: rotateZ(60deg);
}
</style>
```

```

</head>
<script type="text/javascript">
    var Clock=function(json){
        this.hour=json.hour;
        this.min=json.min;
        this.sec=json.sec;
    }
    Clock.prototype.run=function(){
        var clock=this;
        var date=new Date();
        var hh=date.getHours();
        var mm=date.getMinutes();
        var ss=date.getSeconds();
        clock.setTime(hh,mm,ss);
        setInterval(function(){
            date=new Date();
            hh=date.getHours();
            mm=date.getMinutes();
            ss=date.getSeconds();
            clock.setTime(hh,mm,ss);
        },1e3);
    }
    Clock.prototype.setTime=function(h,m,s){
        var
trans=['webkitTransform','msTransform','mozTransform','oTransform','transform'];
        for(var i=0;i<trans.length;i++){
            this.hour.style[trans[i]]='rotateZ('+((h/12)*360)+'deg)';
            this.min.style[trans[i]]='rotateZ('+((m/60)*360)+'deg)';
            this.sec.style[trans[i]]='rotateZ('+((s/60)*360)+'deg)';
        }
    }
    window.onload=function(){
        var oH=document.getElementById('hour');
        var oM=document.getElementById('min');
        var oS=document.getElementById('sec');
        var c1=new Clock({
            hour:oH,
            min:oM,
            sec:oS
        });
        c1.run();
    }
</script>
<body>

```

```
<div class="clock-wrap">
    <div id="hour"></div>
    <div id="min"></div>
    <div id="sec"></div>
</div>
</body>
</html>
```

115.

[问答题]

编写一个 JavaScript 函数，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。
/* @param selector {String} 传入的 CSS 选择器。 * @return {Array} */ var query = function(selector){ //返回查找到的节点数组 return [];}

来自：阿里巴巴

参考：无

116.

[问答题]

如何配置让 nginx 对 js、html、css 文件进行 gzip 压缩输出？

来自：阿里巴巴

参考：

```
gzip on;
gzip_min_length 1k;
gzip_buffers     4 16k;
gzip_types       text/plain application/x-javascript text/css application/xml;
```

117.

[问答题]

有这样一个 URL: <http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e>，请写一段 JS 程序提取 URL 中的各个 GET 参数(参数名和参数个数不确定)，将其按 key-value 形式返回到一个 json 结构中，如{a:'1', b:'2', c:"", d:'xxx', e:undefined}。

来自：阿里巴巴

参考：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>

<script>
var s="http://item.taobao.com/item.htm?a=1&b=2&c=3&d=4"
var re=/.+?/g ;
var re2=/=/g;
var re3=/&/g;
var s1,s2,s3,json;
s1=s.replace(re,"");
s2=s1.replace(re2,:").replace(re3,'');
s3=s2.replace(new RegExp(s2,'g'),function($0,$1){json=eval('({'+$0+'})');});
console.log(json);
</script>
</head>
<body>

</body>
</html>
```

或

```
var get_param=function(url){
    var arr=url.split(/[\?]=\&]/g);
    var result={};
    var len=arr.length%2==0?arr.length+1:arr.length;
    for(var i=1;i<len-1;i=i+2){
        result[arr[i]]=arr[i+1];
    }
    return JSON.stringify(result);
};
```

118.

[问答题]

```
<div class='mod-spm' data-spmid='123'>
    <div class='child_a'></div>
    <div class='child_b'></div>
    <div class='child_c'></div>
    <div class='child_d'></div>
</div>
<div class='mod-spm' data-spmid='456'>
    <div class='child_a'></div>
```

```
<div class='child_b'></div>
<div class='child_c'></div>
<div class='child_d'></div>
</div>
<div class='mod-spm' data-spmid='789'>
    <div class='child_a'></div>
    <div class='child_b'></div>
    <div class='child_c'></div>
    <div class='child_d'></div>
</div>
```

有 dom 结构如上,请用原生代码 (禁用 jQuery 作答) 实现以下功能: (a)计算鼠标在 mod-spm 区域内的停留时长, data-spm 不同视为不同区域 (b)尽量减少性能损耗 (c)重复进入计时累加

来自: 阿里巴巴

参考: 无

119.

[不定项选择题]

按照 CommonJS 规范, 在任何模块代码的作用域下内置了以下哪些变量?

- A.module
- B.context
- C.require
- D.exports

来自: 阿里巴巴

答案: ACD

120.

[问答题]

写出至少 5 个前端优化的方法, 并写明理由。

来自: 百度

参考:

1. 将 CSS 放再顶部 --- 能加快页面内容显示,并且能避免页面产生白屏。
2. 将 JS 放在底部 --- ①JS会阻塞对其后面内容的呈现;②JS会阻塞对其后面内容的下载。
3. 将 JS,CSS 放在外部文件中 (代码和样式的分离) --- 便于优化和管理。
4. 重置 CSS 文件 --- 清除 HTML 标签默认的属性, 让页面按编写者的意愿变化。

5.HTML 尽量使用标准规范的写法 --- 提高渲染引擎的执行效率。

6.对 JS 和 CSS 进行压缩，去重，合并等处理 --- ①减小了文件的体积； ②减小了网络传输量和带宽占用； ③减小了服务器的处理的压力； ④提高了页面的渲染显示的速度。

7.减少页面的图片数目 --- 浏览器拉取页面图片的开销是比较大的，而实际上，我们的页面为了提升用户体验使用了大量图片，这里我们常采用 cdn 存放，图片合并(几个图片合成一个，然后使用 css 进行截取片断显示)，永久 cache (存在图片变更的维护成本,工具的建设等)，甚至有些效果是可以用 css 来实现的代替图片。

121.

[问答题]

我们在进行组件开发的时候，经常会需要用到大量颜色。有两种方法。方法 1：预先定义好大量的颜色；方法 2：自定义函数，采用随机生成颜色的方式。请采用方法 2 实现随机颜色汲取。

来自：百度

参考：

```
function renderColour() {  
    return "#" + (~~(Math.random()*(1<<24))).toString(16);  
}
```

122.

[问答题]

请简要叙述：ActionScript 与 JavaScript 如何进行交互？请用简要的代码说明。

来自：百度

参考：无

123.

[问答题]

请问在 JavaScript 中如何调用以下几个 CSS 属性： font-size， border-top-width, -moz-user-select?

来自：百度

参考：

考察 CSS 在 JS 中的驼峰写法

fontSize

borderTopWidth
mozUserSelect

124.

[问答题]

用 JavaScript 脚本为 Array 对象添加一个去除重复项的方法。

来自：百度

参考：

```
Array.prototype.uniq = function () {
    // 长度只有 1，直接返回当前的拷贝
    if (this.length <= 1) {
        return this.slice(0);
    }
    var aResult = [];
    for (var i = 0, l = this.length; i < l; i++) {
        if (!_fExist(aResult, this[i])) {
            aResult.push(this[i]);
        }
    }
    return aResult;
    // 判断是否重复
    function _fExist(aTmp, o) {
        if (aTmp.length === 0) {
            return false;
        }
        var tmp;
        for (var i = 0, l = aTmp.length; i < l; i++) {
            tmp = aTmp[i];
            if (tmp === o) {
                return true;
            }
        }
        // NaN 需要特殊处理
        if (!o && !tmp && tmp !== undefined && o !== undefined && isNaN(tmp) &&
isNaN(o)) {
            return true;
        }
    }
    return false;
}
或
```

```
Array.prototype.rmRepeat = function(){
    var res = [],hash={};
    var len=this.length;
    for (var i=0;i<len ;i++) {
        if( !hash.hasOwnProperty('_'+this[i]) )
        {
            hash['_'+this[i]] = 1;
            res.push(this[i]);
        }
    }
    return res;
}
```

125.

[问答题]

flash 中的事件处理分哪几个过程？ Event 对象的 target 和 currentTarget 有什么区别？

来自：网易

参考：无

126.

[单选题]

以下哪一条 Javascript 语句会产生运行错误？

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

来自：网易

答案：A

- A 是语法错误
- B 是创建一个数组对象
- C 是创建一个对象
- D 是一个创建正则对象；若为 var obj=/ /; 即赋值被注释掉，及运行被结束；

127.

[单选题]

从四个选项选出不同的一个。

- A.JQuery
 - B.Node.js
 - C.Prototype
 - D.CommonJS
-

来自：搜狐研发工程师模拟笔试题

答案：d

Jquery 是继 prototype 之后又一个优秀的 JavaScript 框架。它是轻量级的 js 库(压缩后只有 21k)，它兼容 CSS3，还兼容各种浏览器。

Node.js 是一套用来编写高性能网络服务器的 JavaScript 工具包。

在 JavaScript 中，prototype 对象是实现面向对象的一个重要机制。每个函数就是一个对象（Function），函数对象都有一个子对象 prototype 对象，类是以函数的形式来定义的。

prototype 表示该函数的原型，也表示一个类的成员的集合。prototype.js 是由 Sam Stephenson 写的一个 javascript 类库。

CommonJS API 定义很多普通应用程序（主要指非浏览器的应用）使用的 API，从而填补了这个空白。

128.

[问答题]

extjs 里对一个支持事件监听的控件，取出监听器的方法有哪些？

来自：阿里巴巴

参考：无

129.

[问答题]

当你打开浏览器在地址栏中输入 “<http://www.baidu.com/>” 后在百度的搜索框中输入 “HTML5”，然后点击百度一下按钮，在所有的信息一一被列举出来的过程中，计算机和网络都发生了什么变化？你有什么建议？

来自：百度

参考：

输入框挂载 onchange 事件，获取客户端输入，ajax 传回后台，查数据库，查出若干匹配项，装进 list，返回给 jsp 输出，ajax 回调方法接收 jsp 的输出，将输出转为一段段字符串，装进输入框下方的 table 或 div 里。

130.

[问答题]

我们把一个数字倒着读和原数字相同的数字称之为对称数，(例如 1,121,88,8998)，不考虑性能，请找出 1—10000 之间的对称数，要求用 javaScript 实现；

来自：百度

参考：

```
function symmetry(num) {  
    var arr = [];  
    while(--num > 10) {  
        var reverseNum = num.toString().split("").reverse().join("");  
        (reverseNum == num) && (arr.push(num));  
    }  
    return arr;  
}  
var result = symmetry(10000);  
console.log(result);  
  
或  
function findSymmetryNum(s,o){  
    var arr=[];  
    for(var i=s;i<=o;i++){  
        var str=""+i,sl=str.length,middle=0,flag=true;  
        if(sl%2==0){  
            middle=sl/2;  
        }else{  
            middle = (sl-1)/2;  
        }  
  
        for(var m=0;m<middle;m++){  
            if(str.substr(0+m,1)!==str.substr(-1-m,1)){  
                flag = false;  
            }  
        }  
        flag&&arr.push(i);  
    }  
  
    console.log(arr);  
    return arr;  
}  
findSymmetryNum(1,10000);
```

131.

[问答题]

浏览器的缓存和本地存储相关内容有哪些？这些在什么环境下都各自能起到什么作用？

来自：百度

参考：

html 页面、图片等。

在联网时已访问的内容可以直接从缓存调出来，提高访问速度离线时，可以本地存储做离线访问（google gears）

以后我们每次访问网站时，IE 会首先搜索这个目录，如果其中已经有访问过的内容，那 IE 就不必从网上下载，而直接从缓存中调出来，从而提高了访问网站的速度。

132.

[问答题]

怎样优化网页性能

来自：百度

参考：

1.尽量减少 HTTP 请求次数

2. 减少 DNS 查找次数

3.资源合并与压缩

4.CSS Sprites

5.Inline Images

6.将外部脚本置底

7.缓存

133.

[问答题]

如何解决跨域问题，这些解决方法应用于什么场景，有什么特点。

来自：百度

参考： 无

134.

[问答题]

ajax 原理、如何实现刷新数据及优点？

来自：迅雷

参考：

Ajax 的工作原理相当于在用户和服务器之间加了一个中间层,使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像一些数据验证和数据处理等都交给 Ajax 引擎自己来做,只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求.

优点：1.减轻服务器负担 2.无刷新更新页面 3 更好的用户体验

135.

[问答题]

用 js 脚本写去除字符串的前后空格。

来自：百度

参考：

```
String.prototype.trim = function(mode)
{//前后去空格
    if (mode=='left') {
        return ((this.charAt(0) == " " && this.length > 0) ? this.slice(1).trim('left') :
this);
    } else
        if (mode == 'right') {
            return ((this.charAt(this.length - 1) == " " && this.length > 0) ?
this.slice(0, this.length - 1).trim('right') : this);
        } else {
            return this.trim('left').trim('right');
        }
};
```

136.

[问答题]

说出 3 条以上 ff 和 ie 的脚本兼容问题。

来自：百度

参考：

IE 有 children, FF 没有；IE 有 parentElement, FF 没有；IE 有 innerText, outerText, outerHTML, FF 没有；FF 有 HTMLElement, HTMLDivElement, XMLDocument, DocumentFragment, Node, Event, Element 等等，IE 没有；IE 有数据岛，FF 没有；IE 跟 FF 创建 XMLHttpRequest 实例的方法不一样。

137.

[问答题]

prototype.js 实现的基本原理。

来自：百度

参考：

将功能封装

比如 Ajax.Request, 还是有判断浏览器的代码; Position 这样的实现页面元素位置的计算

138.

[问答题]

使用 JavaScript 深度克隆一个对象？

来自：百度

参考：

```
function Object.prototype.cloneObj()
{
    function NEWOBJECT(){}
    NEWOBJECT.prototype = this;
    var anObj = new NEWOBJECT();
    for ( var ele in anObj )
    {
        if ( typeof anObj[ele] == "object" ) return anObj[ele].cloneObj();
    }
    return anObj;
}
```

139.

[问答题]

编写一个 JavaScript 函数，实时显示当前时间，格式“年-月-日 时:分:秒”。

来自：百度
参考：

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>time</title>

</head>
<body>
    <div id="time">time</div>
    <script type="text/javascript">
        function startTime(){
            var today=new Date();
            var year=today.getFullYear();
            var month=today.getMonth()+1;
            var date=today.getDate();
            var hour=today.getHours();
            var minute=today.getMinutes();
            var second=today.getSeconds();

            month=checkTime(month);
            date=checkTime(date);
            hour=checkTime(hour);
            minute=checkTime(minute);
            second=checkTime(second);
            var currentTime="";
            currentTime = year+"-"+month+"-"+date+" "+hour+":"+minute+":"+second;
            document.getElementById("time").innerHTML=currentTime;
            setInterval('startTime()',1000);
        }
        startTime();
        function checkTime(t){
            return t<10?"0"+t:t;
        }

    </script>
</body>
</html>
```

140.

[单选题]

蔺相如，司马相如；魏无忌，长孙无忌。下列哪一组对应关系与此类似()

- A.PHP, Python
- B.JSP, servlet
- C.Java, Javascript
- D.C, C++

来自：程序员文化水平闯关挑战卷

答案：C

蔺相如 和 司马相如，都有 相如
魏无忌 和 长孙无忌 都有 无忌
名字里面由重复地方，但没有什么关系
但是 D 选项 C 和 C plus plus ，后者是对前者的进阶版，不是很贴切

141.

[问答题]

根据下图，编写 HTML 结构。要求：符合 xHTML 1.0 规范。

国家	网站名	URL	Alexa 排名
中国	淘宝网	www.taobao.com	38
美国	Ebay	www.ebay.com	22
	Amazon	www.amazon.com	27
Alexa.com 提供数据			

来自：阿里巴巴

参考：

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>阿里巴巴面试表格</title>
</head>
<body>
    <table align = "center" border="1" cellspacing = "0" cellpadding = "0" width = "50%">
        <tr>
            <th>国家</th>
            <th>网站名</th>
            <th>URL</th>
            <th>Alexa 排名</th>
        </tr>
        <tr align = "center">
```

```
<td>中国</td>
<td>淘宝网</td>
<td>www.taobao.com</td>
<td>38</td>
</tr>
<tr align = "center">
<td rowspan = "2">美国</td>
<td>Ebay</td>
<td>www.ebay.com</td>
<td>22</td>
</tr>
<tr align = "center">
<td>Amazon</td>
<td>www.amazon.com</td>
<td>27</td>
</tr>
<tr align = "right">
<td colspan = "4">Alexa.com 提供</td>
</tr>
</table>
</body>
</html>
```

142.

[问答题]

判断字符串是否是这样组成的，第一个必须是字母，后面可以是字母、数字、下划线，总长度为 5-20。

来自：前端工程师练习卷

参考：

```
var reg = /^[a-zA-Z][a-zA-Z_0-9]{4,19}$/;
reg.test("a1a__a1a__a1a__a1a__");
```

143.

[问答题]

截取字符串 abcdefg 的 efg。

来自：前端工程师练习卷

参考：

```
var str = "abcdefg";
if (/efg/.test(str)) {
    var efg = str.substr(str.indexOf("efg"), 3);
    alert(efg);
}
```

144.

[问答题]

判断一个字符串中出现次数最多的字符，统计这个次数。

来自：前端工程师练习卷

参考：

```
//将字符串的字符保存在一个 hash table 中，key 是字符，value 是这个字符出现的次数
var str = "abcdefgaddda";
var obj = {};
for (var i = 0, l = str.length; i < l; i++) {
    var key = str[i];
    if (!obj[key]) {
        obj[key] = 1;
    } else {
        obj[key]++;
    }
}

/*遍历这个 hash table，获取 value 最大的 key 和 value*/
var max = -1;
var max_key = "";
var key;
for (key in obj) {
    if (max < obj[key]) {
        max = obj[key];
        max_key = key;
    }
}
alert("max:"+max+" max_key:"+max_key);
```

145.

[问答题]

IE 与 FF 脚本兼容性问题。

来自：前端工程师练习卷

参考：

(1) window.event:

表示当前的事件对象，IE 有这个对象，FF 没有，FF 通过给事件处理函数传递事件对象

(2) 获取事件源

IE 用 srcElement 获取事件源，而 FF 用 target 获取事件源

(3) 添加，去除事件

IE: element.attachEvent(“onclick”, function) element.detachEvent(“onclick”, function)

FF: element.addEventListener(“click”, function, true) element.removeEventListener(“click”, function, true)

(4) 获取标签的自定义属性

IE: div1.value 或 div1[“value”]

FF: 可用 div1.getAttribute(“value”)

(5) document.getElementByName() 和 document.all[name]

IE: document.getElementByName() 和 document.all[name] 均不能获取 div 元素

FF: 可以

(6) input.type 的属性

IE: input.type 只读

FF: input.type 可读写

(7) innerText textContent outerHTML

IE: 支持 innerText, outerHTML

FF: 支持 textContent

(8) 是否可用 id 代替 HTML 元素

IE: 可以用 id 来代替 HTML 元素

FF: 不可以

146.

[问答题]

规避 javascript 多人开发函数重名问题。

来自：前端工程师练习卷

参考：

(1) 可以开发前规定命名规范，根据不同开发人员开发的功能在函数前加前缀

(2) 将每个开发人员的函数封装到类中，调用的时候就调用类的函数，即使函数重名只要类名不重复就 ok

147.

[问答题]

javascript 面向对象中继承实现。

来自：前端工程师练习卷

参考：

javascript 面向对象中的继承实现一般都使用到了构造函数和 Prototype 原型链，简单的代码如下：

```
function Animal(name) {  
    this.name = name;  
}  
  
Animal.prototype.getName = function() {alert(this.name)}  
function Dog() {};  
Dog.prototype = new Animal("Buddy");  
Dog.prototype.constructor = Dog;  
var dog = new Dog();
```

148.

[问答题]

FF 下面实现 outerHTML

来自：前端工程师练习卷

参考：

FF 不支持 outerHTML，要实现 outerHTML 还需要特殊处理。

思路如下：

在页面中添加一个新的元素 A，克隆一份需要获取 outerHTML 的元素，将这个元素 append 到新的 A 中，然后获取 A 的 innerHTML 就可以了。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />  
<title>获取 outerHMTL</title>  
<style>  
div{ background:#0000FF;width:100px;height:100px;}  
span{ background:#00FF00;width:100px;height:100px;}  
p{ background:#FF0000;width:100px;height:100px;}  
</style>  
</head>  
<body>  
<div id="a"><span>SPAN</span>DIV</div>  
<span>SPAN</span>  
<p>P</p>
```

```
<script type="text/javascript">
function getOuterHTML(id){
var el = document.getElementById(id);
var newNode = document.createElement("div");
document.appendChild(newNode);
var clone = el.cloneNode(true);
newNode.appendChild(clone);
alert(newNode.innerHTML);
document.removeChild(newNode);
}
getOuterHTML("a");
</script>
</body>
</html>
```

149.

[问答题]

编写一个方法 求一个字符串的字节长度。

来自：前端工程师练习卷

参考：

假设：一个英文字符占用一个字节，一个中文字符占用两个字节。

```
function GetBytes(str){
    var len = str.length;
    var bytes = len;
    for(var i=0; i<len; i++){
        if (str.charCodeAt(i) > 255) bytes++;
    }
    return bytes;
}
alert(GetBytes("你好,as"));
```

150.

[问答题]

编写一个方法 去掉一个数组的重复元素

来自：前端工程师练习卷

参考：

```
var arr = [1,1,2,3,3,2,1];
```

```

Array.prototype.unique = function(){
    var ret = [];
    var o = {};
    var len = this.length;
    for (var i=0; i<len; i++){
        var v = this[i];
        if (!o[v]){
            o[v] = 1;
            ret.push(v);
        }
    }
    return ret;
};
alert(arr.unique());

```

151.

[问答题]

写出 3 个使用 this 的典型应用。

来自：前端工程师练习卷

参考：

1) 在 html 元素事件属性中使用，如：

```
<input type="button" onclick="showInfo(this); value="点击一下" />
```

2) 构造函数

```
function Animal(name, color) {
    this.name = name;
    this.color = color;
}
```

3)

```
<input type="button" id="text" value="点击一下" />
```

```
<script type="text/><a href="http://www.bairuiw.com/tag/javascript" class="st_tag internal_tag" rel="tag" title="Posts tagged with Javascript">javascript</a>>
```

```
var btn = document.getElementById("text");
```

```
btn.onclick = function() {
```

```
    alert(this.value); //此处的 this 是按钮元素
```

```
}
```

```
</script>
```

4) CSS expression 表达式中使用 this 关键字

```
<table width="100px" height="100px">
```

```
<tr>
```

```
<td>
```

```
<div style="width:expression(this.parentNode.width);>div element</div>
```

```
</td>
</tr>
</table>
```

152.

[问答题]

如何显示/隐藏一个 DOM 元素？

来自：前端工程师练习卷

参考：

```
el.style.display = "";
el.style.display = "none";
el 是要操作的 DOM 元素。
```

153.

[问答题]

JavaScript 中如何检测一个变量是一个 String 类型？请写出函数实现。

来自：前端工程师练习卷

参考：

String 类型有两种生成方式：

```
(1)Var str = "hello world";
(2)Var str2 = new String("hello world");
function IsString(str){
    return (typeof str == "string" || str.constructor == String);
}
var str = "";
alert(IsString(1));
alert(IsString(str));
alert(IsString(new String(str)));
```

154.

[问答题]

网页中实现一个计算当年还剩多少时间的倒数计时程序，要求网页上实时动态显示“××年还剩××天××时××分××秒”。

来自：前端工程师练习卷

参考：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>倒计时</title>
</head>
<body>
<input type="text" value="" id="input" size="1000"/>
<script type="text/javascript">
function counter() {
    var date = new Date();
    var year = date.getFullYear();
    var date2 = new Date(year, 12, 31, 23, 59, 59);
    var time = (date2 - date)/1000;
    var day =Math.floor(time/(24*60*60))
    var hour = Math.floor(time%(24*60*60)/(60*60))
    var minute = Math.floor(time%(24*60*60)%(60*60)/60);
    var second = Math.floor(time%(24*60*60)%(60*60)%60);
    var str = year + "年还剩"+day+"天"+hour+"时"+minute+"分"+second+"秒";
    document.getElementById("input").value = str;
}
window.setInterval("counter()", 1000);
</script>
</body>
</html>
```

155.

[问答题]

补充代码，鼠标单击 Button1 后将 Button1 移动到 Button2 的后面。

```
<div><input type="button" id="button1" value="1" onclick="?">
<input type="button" id="button2" value="2" /></div>
```

来自：前端工程师练习卷

参考：

```
<div>
<input type="button" id="button1" value="1" onclick="moveBtn(this);">
<input type="button" id="button2" value="2" />
</div>
<script type="text/javascript">
```

```
function moveBtn(obj) {  
    var clone = obj.cloneNode(true);  
    var parent = obj.parentNode;  
    parent.appendChild(clone);  
    parent.removeChild(obj);  
}  
</script>
```

156.

[问答题]

JavaScript 有哪几种数据类型。

来自：前端工程师练习卷

参考：

简单：Number, Boolean, String, Null, Undefined

复合：Object, Array, Function

157.

[问答题]

下面 css 标签在 JavaScript 中调用应如何拼写，border-left-color, -moz-viewport。

来自：前端工程师练习卷

参考：

borderLeftColor

mozViewport

158.

[问答题]

JavaScript 中如何对一个对象进行深度 clone。

来自：前端工程师练习卷

参考：

```
function cloneObject(o) {  
    if(!o || 'object' !== typeof o) {  
        return o;  
    }
```

```
var c = 'function' === typeof o.pop ? [] : {};
var p, v;
for(p in o) {
    if(o.hasOwnProperty(p)) {
        v = o[p];
        if(v && 'object' === typeof v) {
            c[p] = Ext.ux.clone(v);
        }
        else {
            c[p] = v;
        }
    }
}
return c;
};
```

159.

[问答题]

如何控制 alert 中的换行。

来自：前端工程师练习卷

参考：

\n alert("p\np");

160.

[问答题]

请实现，鼠标点击页面中的任意标签，alert 该标签的名称. (注意兼容性)。

来自：前端工程师练习卷

参考：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>鼠标点击页面中的任意标签，alert 该标签的名称</title>
<style>
div{ background:#0000FF;width:100px;height:100px;}
span{ background:#00FF00;width:100px;height:100px;}
```

```
p{ background:#FF0000; width:100px; height:100px; }
</style>
<script type="text/javascript">
document.onclick = function(evt){
var e = window.event || evt;
var tag = e["target"] || e["srcElement"];
alert(tag.tagName);
};
</script>
</head>
<body>
<div id="div"><span>SPAN</span>DIV</div>
<span>SPAN</span>
<p>P</p>
</body>
</html>
```

161.

[问答题]

请编写一个 JavaScript 函数 `parseQueryString`, 它的用途是把 URL 参数解析为一个对象, 如: `var url = "http://witmax.cn/index.php?key0=0&key1=1&key2=2";`

来自: 前端工程师练习卷

参考:

```
function parseQueryString(url){
    var params = {};
    var arr = url.split("?");
    if (arr.length <= 1)
        return params;
    arr = arr[1].split("&");
    for(var i=0, l=arr.length; i<l; i++){
        var a = arr[i].split("=");
        params[a[0]] = a[1];
    }
    return params;
}
var url = "http://witmax.cn/index.php?key0=0&key1=1&key2=2";
var ps = parseQueryString(url);
alert(ps["key1"]);
```

162.

[问答题]

ajax 是什么? ajax 的交互模型? 同步和异步的区别? 如何解决跨域问题?

来自: 前端工程师练习卷

参考:

Ajax 是多种技术组合起来的一种浏览器和服务器交互技术, 基本思想是允许一个互联网浏览器向一个远程页面/服务做异步的 http 调用, 并且用收到的数据来更新一个当前 web 页面而不必刷新整个页面。该技术能够改进客户端的体验。包含的技术:

XHTML: 对应 W3C 的 XHTML 规范, 目前是 XHTML1.0。

CSS: 对应 W3C 的 CSS 规范, 目前是 CSS2.0

DOM: 这里的 DOM 主要是指 HTML DOM, XML DOM 包括在下面的 XML 中

JavaScript: 对应于 ECMA 的 ECMAScript 规范

XML: 对应 W3C 的 XML DOM、XSLT、XPath 等等规范

XMLHttpRequest : 对应 WhatWG 的 Web Applications1.0 规范
(<http://whatwg.org/specs/web-apps/current-work/>)

同步: 脚本会停留并等待服务器发送回复然后再继续

异步: 脚本允许页面继续其进程并处理可能的回复

跨域问题简单的理解就是因为 JS 同源策略的限制, a.com 域名下的 JS 无法操作 b.com 或 c.a.com 下的对象, 具体场景如下:

PS:

(1)如果是端口或者协议造成的跨域问题前端是无能为力的

(2) 在跨域问题上, 域仅仅通过 URL 的头部来识别而不会尝试判断相同的 IP 地址对应的域或者两个域是否对应一个 IP

前端对于跨域的解决办法:

(1) document.domain+iframe

(2) 动态创建 script 标签

163.

[问答题]

什么是闭包? 下面这个 ul, 如何点击每一列的时候 alert 其 index?

来自: 前端工程师练习卷

参考:

```
<ul id="test">
  <li>这是第一条</li>
  <li>这是第二条</li>
  <li>这是第三条</li>
</ul>
```

内部函数被定义它的函数的外部区域调用的时候就产生了闭包。

```
(function A() {
    var index = 0;
    var ul = document.getElementById("test");
    var obj = {};
    for (var i = 0, l = ul.childNodes.length; i < l; i++) {
        if (ul.childNodes[i].nodeName.toLowerCase() == "li") {
            var li = ul.childNodes[i];
            li.onclick = function() {
                index++;
                alert(index);
            }
        }
    }
})();
```

164.

[问答题]

请给出异步加载 js 方案，不少于两种。

来自：前端工程师练习卷

参考：

默认情况 javascript 是同步加载的，也就是 javascript 的加载时阻塞的，后面的元素要等待 javascript 加载完毕后才能进行再加载，对于一些意义不是很大的 javascript，如果放在页头会导致加载很慢的话，是会严重影响用户体验的。

异步加载方式：

(1) defer，只支持 IE

(2) async:

(3) 创建 script，插入到 DOM 中，加载完毕后 callBack，见代码：

```
function loadScript(url, callback){
    var script = document.createElement("script")
    script.type = "text/javascript";
    if (script.readyState){ //IE
        script.onreadystatechange = function(){
            if (script.readyState == "loaded" || 
                script.readyState == "complete"){
                script.onreadystatechange = null;
                callback();
            }
        };
    } else { //Others: Firefox, Safari, Chrome, and Opera
    }
}
```

```
        script.onload = function(){
            callback();
        };
    }
    script.src = url;
    document.body.appendChild(script);
}
```

165.

[问答题]

请设计一套方案，用于确保页面中 JS 加载完全。

来自：前端工程师练习卷

参考：

```
var n = document.createElement("script");
n.type = "text/javascript";
//以上省略部分代码
//ie 支持 script 的 readystatechange 属性
if(ua.ie){
    n.onreadystatechange = function(){
        var rs = this.readyState;
        if('loaded' === rs || 'complete'===rs){
            n.onreadystatechange = null;
            f(id,url); //回调函数
        }
    };
    //省略部分代码
    //safari 3.x supports the load event for script nodes(DOM2)
    n.addEventListener('load',function(){
        f(id,url);
    });
    //firefox and opera support onload(but not dom2 in ff) handlers for
    //script nodes. opera, but no ff, support the onload event for link
    //nodes.
}else{
    n.onload = function(){
        f(id,url);
    };
}
```

166.

[问答题]

js 中如何定义 class, 如何扩展 prototype?

来自：前端工程师练习卷

参考：

Ele.className = “***”; //***在 css 中定义，形式如下：.*** {…}

A.prototype.B = C;

A 是某个构造函数的名字

B 是这个构造函数的属性

C 是想要定义的属性的值

167.

[问答题]

如何添加 html 元素的事件, 有几种方法.

来自：前端工程师练习卷

参考：

- (1) 为 HTML 元素的事件属性赋值
- (2) 在 JS 中使用 ele.on*** = function() {…}
- (3) 使用 DOM2 的添加事件的方法 addEventListener 或 attachEvent

168.

[问答题]

document.write 和 innerHTML 的区别。

来自：前端工程师练习卷

参考：

document.write 只能重绘整个页面。

innerHTML 可以重绘页面的一部分。

169.

[问答题]

多浏览器检测通过什么？

来自：前端工程师练习卷

参考：

- (1) navigator.userAgent
- (2) 不同浏览器的特性，如 addEventListener

170.

[问答题]

js 的基础对象有那些, window 和 document 的常用的方法和属性列出来。

来自：前端工程师练习卷

参考：

String,Number,Boolean

Window:

方法： setInterval,setTimeout,clearInterval,clearTimeout,alert,confirm,open

属性： name,parent,screenLeft,screenTop,self,top,status

Document

方 法 :

createElement,execCommand,getElementById,getElementsByTagName,getElementByTagName,write,writeln

属性： cookie,doctype,documentElement,readyState,URL,

171.

[问答题]

前端开发的优化问题。

来自：前端工程师练习卷

参考：

- (1) 减少 http 请求次数：css spirit,data uri
- (2) JS, CSS 源码压缩
- (3) 前端模板 JS+数据，减少由于 HTML 标签导致的带宽浪费，前端用变量保存 AJAX 请求结果，每次操作本地变量，不用请求，减少请求次数
- (4) 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能
- (5) 用 setTimeout 来避免页面失去响应
- (6) 用 hash-table 来优化查找
- (7) 当需要设置的样式很多时设置 className 而不是直接操作 style
- (8) 少用全局变量
- (9) 缓存 DOM 节点查找的结果

- (10) 避免使用 CSS Expression
- (11) 图片预载
- (12) 避免在页面的主体布局中使用 table, table 要等其中的内容完全下载之后才会显示出来, 显示比 div+css 布局慢

172.

[问答题]

如何控制网页在网络传输过程中的数据量。

来自: 前端工程师练习卷

提示:

启用 GZIP 压缩

保持良好的编程习惯, 避免重复的 CSS, JavaScript 代码, 多余的 HTML 标签和属性

173.

[问答题]

Flash、Ajax 各自的优缺点, 在使用中如何取舍?

来自: 前端工程师练习卷

提示:

Ajax 的优势

- (1) 可搜索性
- (2) 开放性
- (3) 费用
- (4) 易用性
- (5) 易于开发

Flash 的优势

- (1) 多媒体处理
- (2) 兼容性
- (3) 矢量图形 比 SVG, Canvas 优势大很多
- (4) 客户端资源调度, 比如麦克风, 摄像头

174.

[问答题]

写出 Math Array String 的方法。

来自：腾讯最新前端面试题

参考：无

175.

[问答题]

http 状态码都有哪些。

来自：腾讯最新前端面试题

提示：

404: 页面不存在

301: 永久转移

.....

越多越好，可以反映你基础

176.

[问答题]

反转字符串。

来自：腾讯最新前端面试题

提示：

`str.split("").reverse().join("")`

177.

[问答题]

把字符串的单词放到数组中。

来自：腾讯最新前端面试题

提示：

`arr = str.split(/\s+/);`

怎么把标点符号也去掉

`str.replace(/[^w]/,' ').split(/\s+/);`

字符串包括数字，不要把数字也去掉

178.

[问答题]

表格的的语义化标签。

来自：腾讯最新前端面试题

提示：

<caption>

....

179.

[问答题]

遍历表格，把 td 内容连接。

来自：腾讯最新前端面试题

提示：

getElementsByName 两个 for 循环。

或

使用 innerHTML

180.

[问答题]

对二维数组的某个进行排序。

来自：腾讯最新前端面试题

提示：

arr.sort(a,b){ return a[0] > b[0] }

181.

[问答题]

设计一套方案判断 css 加载完成。

来自：腾讯最新前端面试题

参考：无

182.

[问答题]

ie 标准浏览器事件对象。

来自：腾讯最新前端面试题

提示：

ev || window.event
srcElement target 捕获
ie 只支持冒泡

183.

[问答题]

ajax 的底层，原生怎么搞。

来自：腾讯最新前端面试题

提示：

readystate : 4 , 0 1 2 3 4 这几种状态,
status : 200 判断正确返回数据
response 有几种属性
json 类型字符串返回后怎么转换成对象?
json.parse
ie 怎么处理
ie6,7 引入 json.js

184.

[问答题]

数组去重，原型扩展，两种方法

来自：腾讯最新前端面试题

提示：

```
Array.prototype.unique = function(){
    var arr = this, len=this.length, obj={}, newArr=[];
    while(--len>=0){
        if(obj[ arr[len] ] !== arr[len]){
            obj[arr[len]] = arr[len];    newArr.push( arr[len] );
        }
    }
    return newArr.reverse();
}
```

185.

[问答题]

图片，内容延时加载。

来自：腾讯最新前端面试题

提示：

用 scrolltop 判断有没有到第二屏，然后用 ajax 去取数据

.....

186.

[问答题]

如何在 ie 下模拟 DOMContentLoaded 事件。

来自：腾讯最新前端面试题

提示：

创建一个指向空的 `src=http://void(0); defer`, ie 支持这个用 `defer`, 浏览器再 DOM 加载完才触发, 所以在 `script` 的 `readstate == "complete"` 的时候说明 DOM 已经加载完成了

另一种方法看了一下是用定时器, 当 `try{ (document.documentElement || document.body).doscroll("left") }` 也说明加载完了

187.

[问答题]

请看以下代码, 按照下面的要求回答

```
if(window.addEventListener){  
    var fn = function (type,fn,useCapture){  
        el.addEventListener(type,fn,useCapture);  
    }  
}  
else if(window.attachEvent){  
    fn = function (type,fn){  
        el.attachEvent('on'+type,);  
    }  
}
```

a)以上代码的作用。

b)以上代码的优点。

c)以上代码中的问题, 如果你有更好的, 请把它编写出来。

来自：淘宝 UED Web 前端开发面试题

参考：

考察对事件绑定的深入理解。

a)以上代码主要是为 HTML 元素绑定一个事件，并且兼容 IE 和 DOM 标准下的浏览器。

b)以上代码的优点是做到了事件绑定的兼容性。

c)以上代码中 fn 这个变量是在 DOM 标准下的浏览器中才会声明，在 IE 下它将是一个全局变量。

简单的编写：

```
function bind(el, type, fn, useCapture){  
    if (window.addEventListener) {  
        el.addEventListener(type, function(){  
            fn.apply(el, arguments); //始终将 this 指向 DOM  
        }, useCapture);  
    }  
    else if (window.attachEvent) {  
        el.attachEvent('on' + type, function(){  
            fn.apply(el, arguments); //始终将 this 指向 DOM  
        });  
    }  
}  
  
var el = document.getElementById('demo');  
var test = function(){  
    alert(this.nodeName);  
}  
bind(el, 'click', test);
```

注意代码中的注释部分。因为在符合 DOM 标准的浏览器中，addEventListener 方法将把 this 指针指向绑定的函数，而 IE 中 attachEvent 方法将始终指向 window 对象，为了将 this 指针始终指向当前绑定事件的 DOM，我们必须要使用 apply 或者 call 方法来改变函数的作用域。

188.

[问答题]

请计算下面变量的值：

```
var a= (Math.PI++);  
var b = (Math.PI++);  
alert(a);  
alert(b);
```

来自：淘宝 UED Web 前端开发面试题

提示：

考察对 javascript 中 Math 对象的深入理解。

189.

[问答题]

注释的代码是否可以实现？如不能实现，请修改

```
function test(){
    this.name = 'taobao';
    this.waitMes = function (){
        //隔 5 秒钟执行 this.name
    }
}
```

来自：淘宝 UED Web 前端开发面试题

参考：

考察 javascript 闭包。

```
function test(){
    this.name = 'taobao';
    var waitMes = function (){
        //每隔 5 秒钟执行 this.name
        setTimeout(function (){alert(self.name)},5000);
    }
    return waitMes;
}
var _test = test();
_test();
```

190.

[问答题]

要求：

- 1、只能在指定的位置填写自己的代码，本文件里的其他代码不能修改
- 2、所有题目都不允许添加全局变量名
- 3、本文件应该能在 firebug 的 console 里正常执行，并输出结果
- 4、代码最优化，效率最高
- 5、代码注释明确

实现一个遍历数组或对象里所有成员的迭代器。

```
var each = function(obj, fn){
    //++++++++++答题主区域+++++
    //++++++++++答题主区域+++++
};

try{
    var data1 = [4,5,6,7,8,9,10,11,12];
```

```

var data2 = {
    "a": 4,
    "b": 5,
    "c": 6
};

console.group(data1);
each(data1, function(o){
    if( 6 == this )
        return true;
    else if( 8 == this )
        return false;
    console.log(o + ": \"" + this + "\"");
});

console.groupEnd();
/*-----[执行结果]-----
1: "4"
2: "5"
4: "7"
-----*/
console.group(data2);
each(data2, function(v, n){
    if( 5 == this )
        return true;
    console.log(n + ": \"" + v + "\"");
});

console.groupEnd();
/*-----[执行结果]-----
a: "4"
c: "6"
-----*/
}catch(e){
    console.error("执行出错， 错误信息: " + e);
}

```

来自：搜狐 JavaScript 面试题

参考：无

191.

[问答题]

要求：

- 1、只能在指定的位置填写自己的代码，本文件里的其他代码不能修改
- 2、所有题目都不允许添加全局变量名

3、本文件应该能在 firebug 的 console 里正常执行，并输出结果

4、代码最优化，效率最高

5、代码注释明确

实现一个叫 Man 的类，包含 attr, words, say 三个方法。

```
var Man;  
//++++++答题区域++++++  
  
//++++++答题结束++++++  
try{  
    var me = Man({ fullname: "小红" });  
    var she = new Man({ fullname: "小红" });  
    console.group();  
    console.info("我的名字是: " + me.attr("fullname") + "\n 我的性别是: " +  
me.attr("gender"));  
    console.groupEnd();  
    /*----[执行结果]----  
    我的名字是: 小红  
    我的性别是: <用户未输入>  
    -----*/  
    me.attr("fullname", "小明");  
    me.attr("gender", "男");  
    me.fullname = "废柴";  
    me.gender = "人妖";  
    she.attr("gender", "女");  
    console.group();  
    console.info("我的名字是: " + me.attr("fullname") + "\n 我的性别是: " +  
me.attr("gender"));  
    console.groupEnd();  
    /*----[执行结果]----  
    我的名字是: 小明  
    我的性别是: 男  
    -----*/  
    console.group();  
    console.info("我的名字是: " + she.attr("fullname") + "\n 我的性别是: " +  
she.attr("gender"));  
    console.groupEnd();  
    /*----[执行结果]----  
    我的名字是: 小红  
    我的性别是: 女  
    -----*/  
    me.attr({  
        "words-limit": 3,  
        "words-emote": "微笑"  
    });
```

```

me.words("我喜欢看视频。");
me.words("我们的办公室太漂亮了。");
me.words("视频里美女真多！");
me.words("我平时都看优酷！");
console.group();
console.log(me.say());
/*----[执行结果]-----
小明微笑： "我喜欢看视频。我们的办公室太漂亮了。视频里美女真多！"
-----*/
me.attr({
    "words-limit": 2,
    "words-emote": "喊"
});
console.log(me.say());
console.groupEnd();
/*----[执行结果]-----
小明喊： "我喜欢看视频。我们的办公室太漂亮了。"
-----*/
}catch(e){
    console.error("执行出错， 错误信息: " + e);
}

```

来自： 搜狐 JavaScript 面试题
参考： 无

192.

[问答题]

要求：

- 1、只能在指定的位置填写自己的代码，本文件里的其他代码不能修改
- 2、所有题目都不允许添加全局变量名
- 3、本文件应该能在 firebug 的 console 里正常执行，并输出结果
- 4、代码最优化，效率最高
- 5、代码注释明确

实现一个 URI 解析方法，把 url 里#之后的参数解析成指定的数据结构。

```

function urlParser(s){
    //++++++++++答题区域+++++
    //++++++++++答题结束+++++
}

try{
    var url1 = "http://www.abc.com/m/s/#page/2/?type=latest_videos&page_size=20";
}

```

```
var url2 = "http://www.abc.com/m/s/#type=latest_videos&page_size=20";
var url3 = "http://www.abc.com/m/s/#page?type=latest_videos&page_size=20";
console.group();
console.info( urlParser(url1) );
console.info( urlParser(url2) );
console.info( urlParser(url3) );
console.groupEnd();
/*----[执行结果]----*/
[{"page": "2", "type": "latest_videos", "page_size": 20 }]
[{"type": "latest_videos", "page_size": 20 }]
[{"page": "2", "type": "latest_videos", "page_size": 20 }]
-----*/
}catch(e){
    console.error("执行出错， 错误信息: " + e);
}
```

来自： 搜狐 JavaScript 面试题
参考： 无

193.

[问答题]
javascript 的 typeof 返回哪些数据类型。

来自： 前端工程师练习题
参考：
Object number function boolean undefined

194.

[问答题]
例举 3 种强制类型转换和 2 种隐式类型转换？

来自： 前端工程师练习题
参考：
强制（parseInt,parseFloat,number）
隐式（== - ===）

195.

[问答题]

split() join()的区别

来自：前端工程师练习题

参考：

前者是切割成数组的形式，后者是将数组转换成字符串。

196.

[问答题]

数组方法 pop() push() unshift() shift()。

来自：前端工程师练习题

参考：

Push()尾部添加 pop()尾部删除。

Unshift()头部添加 shift()头部删除。

197.

[问答题]

事件绑定和普通事件有什么区别，IE 和 DOM 事件流的区别。

来自：前端工程师练习题

参考：

1.执行顺序不一样、

2.参数不一样

3.事件加不加 on

4.this 指向问题

198.

[问答题]

IE 和标准下有哪些兼容性的写法。

来自：前端工程师练习题

参考：

```
Var ev = ev || window.event  
document.documentElement.clientWidth || document.body.clientWidth  
Var target = ev.srcElement || ev.target
```

199.

[问答题]

ajax 请求的时候 get 和 post 方式的区别。

来自：前端工程师练习题

参考：

一个在 url 后面 一个放在虚拟载体里面。

有大小限制。

安全问题。

应用不同：一个是论坛等只需要请求的，一个是类似修改密码的。

200.

[问答题]

call 和 apply 的区别。

来自：前端工程师练习题

参考：

Object.call(this,obj1,obj2,obj3)

Object.apply(this,arguments)

201.

[问答题]

ajax 请求时，如何解释 json 数据。

来自：前端工程师练习题

参考：

使用 eval、parse，鉴于安全性考虑 使用 parse 更靠谱。

202.

[问答题]

写一个获取非行间样式的函数。

来自：前端工程师练习题

参考：

```
function getStyle(obj,attr,value)
{
if(!value)
{
if(obj.currentStyle)
{
return obj.currentStyle(attr)
}
else
{
obj.getComputedStyle(attr,false)
}
}
else
{
obj.style[attr]=value
}
}
```

203.

[问答题]

事件委托是什么。

来自：前端工程师练习题

参考：

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行。

204.

[问答题]

闭包是什么，有什么特性，对页面有什么影响。

来自：前端工程师练习题

参考：

闭包就是能够读取其他函数内部变量的函数。

205.

[问答题]

如何阻止事件冒泡和默认事件。

来自：前端工程师练习题

提示：

cancelBubble return false

206.

[问答题]

添加 删除 替换 插入到某个接点的方法。

来自：前端工程师练习题

提示：

obj.appendChild()

obj.innerHTML

obj.replaceChild

obj.removeChild

207.

[问答题]

解释 jsonp 的原理，以及为什么不是真正的 ajax。

来自：前端工程师练习题

提示：

动态创建 script 标签，回调函数

Ajax 是页面无刷新请求数据操作

208.

[问答题]

javascript 的本地对象，内置对象和宿主对象。

来自：前端工程师练习题

提示：

本地对象为 array obj regexp 等可以 new 实例化。

内置对象为 global Math 等不可以实例化的。

宿主为浏览器自带的 document,window 等。

209.

[问答题]

document load 和 document ready 的区别。

来自：前端工程师练习题

提示：

Document.onload 是在结构和样式加载完才执行 js

Document.ready 原生没有这个方法， jquery 中有 \$().ready(function)

210.

[问答题]

==和 ===” 的不同。

来自：前端工程师练习题

提示：

前者会自动转换类型。

后者不会。

211.

[问答题]

javascript 的同源策略。

来自：前端工程师练习题

提示：

一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

212.

[问答题]

编写一个数组去重的方法。

来自：前端工程师练习题

提示：

```
function oSort(arr)
{
    var result = {};
    var newArr=[];
    for(var i=0;i<arr.length;i++)
    {
        if(!result[arr])
        {
            newArr.push(arr)
            result[arr]=1
        }
    }
    return newArr
}
```

213.

[问答题]

下面代码的输出值是？

```
alert(1&&2);
```

来自：奇虎 360Web 前端开发工程师面试题一面

参考：无

214.

[问答题]

正则表达式匹配，开头为 11N, 12N 或 1NNN，后面是-7-8 个数字的电话号码。

来自：奇虎 360Web 前端开发工程师面试题一面

参考：无

215.

[问答题]

写出下面代码的输出值：

```
var obj = { a: 1, b: function () {console.log(this.a)} }; var a = 2; var objb = obj.b; obj.b(); objb(); obj.b.call(window);
```

来自：奇虎 360Web 前端开发工程师面试题一面
参考：无

216.

[问答题]

写出下列代码的输出值：

```
function A() {} function B(a) { this.a = a; } function C(a) { if (a) { thia.a = a; } } A.prototype.a = 1; B.prototype.a = 1; C.prototype.a = 1; console.log(new A()); console.log(new B()); console.log(new C(2));
```

来自：奇虎 360Web 前端开发工程师面试题一面
参考：无

217.

[问答题]

写出下列代码的输出值：

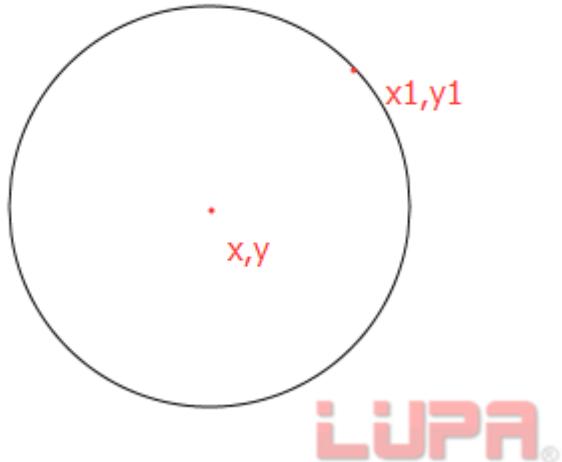
```
var a = 1; function b() { var a = 2; function c() { console.log(a); } return c; } b()();
```

来自：奇虎 360Web 前端开发工程师面试题一面
参考：无

218.

[问答题]

已知圆心(x,y)，求圆上任一点(x1,y1)的坐标。



LUPA®

来自：奇虎 360Web 前端开发工程师面试题二面
参考：无

219.

[问答题]

随机抛五枚硬币，求三枚及以上朝上的概率。



来自：奇虎 360Web 前端开发工程师面试题二面
参考：无

220.

[问答题]

JS 主要数据类型？

来自：百度校园招聘 web 前端开发面试题

参考：

主要的类型有 number、string、object 以及 Boolean 类型,其他两种类型为 null 和 undefined。

221.

[问答题]

CSS 的 JS 调用？如 font-family, -moz-border-radius 。

来自：百度校园招聘 web 前端开发面试题

参考：

fontFamily、MozBorderRadius

222.

[问答题]

js 对象的深度克隆？

来自：百度校园招聘 web 前端开发面试题

参考：

```
Object.prototype.deepClone=function(){
    function cloneObj(){}
    cloneObj.prototype=this;
    var obj=new cloneObj();
    for(var o in obj){
        if(typeof(obj[o])=="object")
            obj[o]=obj[o].deepClone();
    }
    return obj;
}
```

223.

[问答题]

动态打印时间，格式为 yyyy-MM-dd hh:mm:ss?

来自：百度校园招聘 web 前端开发面试题

参考：

```
function printTime(){
    var timer1=new Date();
    var timer=timer1.toLocaleString();
    timer=timer.replace(/\[年月\]/g,"-");
    timer=timer.replace(/\日/,"\"");
    time.innerHTML=timer;
}
setInterval("printTime()",1000);
```

224.

[问答题]

如何提高网页运行性能？

来自：百度校园招聘 web 前端开发面试题

参考：无

225.

[问答题]

linux 下删除当前目录下扩展名为 c 的文件（如 a.c, b.c）。

来自：百度校园招聘 web 前端开发面试题

参考：

```
rm -r *.c
find . -name “*.doc” -type f -exec cp {} /tmp/doc \; 找到当前目录(.)下扩展名为(doc)
的文件并拷贝到指定目录【注意-type f 指普通文件, -exec ls-l{}列出文件, 最后加上\】
cp [options] source dest 复制
```

226.

[问答题]

flash as2.0 和 flash as3.0 在面向对象方面的异同？

来自：百度校园招聘 web 前端开发面试题

参考：

面向对象方面，2.0 像 javascript，3.0 像 java。

到了 AS 2.0，面向对象被引入了，但它实质上是动态脚本语言，虽然已经有了类的概念和 class 关键字，但对象支持还是基于类似 JavaScript 的 prototype 机制——动态继承。

3.0 同时支持静态类型，即基于类的继承方式；以及动态类型，即基于 prototype 的继承方式。推荐用静态类型。

227.

[问答题]

Flash、Ajax 各自的优缺点，在使用中如何取舍？

来自：百度校园招聘 web 前端开发面试题

参考：

Flash 适合处理多媒体、矢量图形、访问机器；对 CSS、处理文本上不足，不容易被搜索。

Ajax 对 CSS、文本支持很好，支持搜索；多媒体、矢量图形、机器访问不足。

共同点：与服务器的无刷新传递消息、用户离线和在线状态、操作 DOM

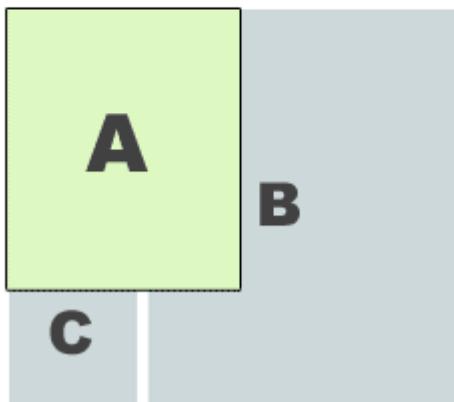
228.

[问答题]

用 javascript 优化布局。由于我们的用户群喜欢放大看页面，于是我们给页面布局做一次优化。当鼠标略过某个区块的时候，该区块会放大 25%，并且其他的区块仍然固定不动。



当鼠标略过A区，A区被放大，
BC区块在原来位置不变



提示：

也许，我们其他的布局也会用到这个放大的效果哦。可以使用任何开源代码，包括曾经你自己写的。

关键字：

javascript、封装、复用

来自：阿里巴巴 Web 前端开发面试题

参考：无

229.

[不定项选择题]

声明一个对象，给它加上 name 属性和 show 方法显示其 name 值，以下代码中正确的是（）

- A. var obj = [name:"zhangsan",show:function(){alert(name)}];
- B. var obj = {name:"zhangsan",show:"alert(this.name)"}
- C. var obj = {name:"zhangsan",show:function(){alert(name)}};
- D. var obj = {name:"zhangsan",show:function(){alert(this.name)}};

来自：前端开发工程师练习题

答案：D

230. 230

[不定项选择题]

以下关于 `Array` 数组对象的说法不正确的是（ ）

- A. 对数组里数据的排序可以用 `sort` 函数，如果排序效果非预期，可以给 `sort` 函数加一个排序函数的参数
 - B. `reverse` 用于对数组数据的倒序排列
 - C. 向数组的最后位置加一个新元素，可以用 `pop` 方法
 - D. `unshift` 方法用于向数组删除第一个元素
-

来自：前端开发工程师练习题

答案：CD

231.

[不定项选择题]

要将页面的状态栏中显示“已经选中该文本框”，下列 JavaScript 语句正确的是（ ）

- A. `window.status=“已经选中该文本框”`
 - B. `document.status=“已经选中该文本框”`
 - C. `window.screen=“已经选中该文本框”`
 - D. `document.screen=“已经选中该文本框”`
-

来自：前端开发工程师练习题

答案：A

232.

[不定项选择题]

点击页面的按钮，使之打开一个新窗口，加载一个网页，以下 JavaScript 代码中可行的是（ ）

- A. `<input type="button" value="new" onclick="open('new.html', '_blank')"/>`
 - B. `<input type="button" value="new" onclick="window.location='new.html';"/>`
 - C. `<input type="button" value="new" onclick="location.assign('new.html');"/>`
 - D.
-

来自：前端开发工程师练习题

答案: AD

233.

[不定项选择题]

使用 JavaScript 向网页中输出 hello，以下代码中可行的是（）

- A. document.write(hello);
 - B. document.write("hello");
 - C. hello
 - D. document.write("hello");
-

来自：前端开发工程师练习题

答案：BD

234.

[不定项选择题]

分析下面的代码：

```
function writelt (value) { document.myfm.first_text.value=value;}
```

以下说法中正确的是（）

- A. 在页面的第二个文本框中输入内容后，当鼠标离开第二个文本框时，第一个文本框的内容不变
 - B. 在页面的第一个文本框中输入内容后，当鼠标离开第一个文本框时，将在第二个文本框中复制第一个文本框的内容
 - C. 在页面的第二个文本框中输入内容后，当鼠标离开第二个文本框时，将在第一个文本框中复制第二个文本框的内容
 - D. 在页面的第一个文本框中输入内容后，当鼠标离开第一个文本框时，第二个文本框的内容不变
-

来自：前端开发工程师练习题

答案：CD

235.

[不定项选择题]

下面的 JavaScript 语句中，（）实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

```
A. for(var i=0;i< form1.elements.length;i++) {  
if(form1.elements.type=="text")  
form1.elements.value="";}
```

```
B. for(var i=0;i<document.forms.length;i++) {  
    if(forms[0].elements.type=="text")  
        forms[0].elements.value="";  
    }  
C. if(document.form.elements.type=="text")  
    form.elements.value="";  
D. for(var i=0;i<document.forms.length; i++){  
    for(var j=0;j<document.forms.elements.length; j++){  
        if(document.forms.elements[j].type=="text")  
            document.forms.elements[j].value="";  
    }  
}
```

来自：前端开发工程师练习题

答案： D

236.

[不定项选择题]

在表单(form1)中有一个文本框元素(fname)，用于输入电话号码，格式如：010-82668155，要求前 3 位是 010，紧接一个“-”，后面是 8 位数字。要求在提交表单时，根据上述条件验证该文本框中输入内容的有效性，下列语句中，() 能正确实现以上功能。

```
A. var str= form1.fname.value;  
if(str.substr(0,4)!="010-" || str.substr(4).length!=8 ||  
isNaN(parseFloat(str.substr(4))))  
alert( “无效的电话号码！” );  
B. var str= form1.fname.value;  
if(str.substr(0,4)!="010-" && str.substr(4).length!=8 &&  
isNaN(parseFloat(str.substr(4))))  
alert( “无效的电话号码！” );  
C. var str= form1.fname.value;  
if(str.substr(0,3)!="010-" || str.substr(3).length!=8 ||  
isNaN(parseFloat(str.substr(3))))  
alert( “无效的电话号码！” );  
D. var str= form1.fname.value;  
if(str.substr(0,4)!="010-"&& str.substr(4).length!=8 &&  
!isNaN(parseFloat(str.substr(4))))  
alert( “无效的电话号码！” );
```

来自：前端开发工程师练习题

答案： A

237.

[不定项选择题]

关于正则表达式声明 6 位数字的邮编，以下代码正确的是（ ）

- A. var reg = /\d6/;
- B. var reg = \d{6}\;
- C. var reg = /\d{6}/;
- D. var reg = new RegExp("\d{6}");

来自：前端开发工程师练习题

答案：C

238.

[不定项选择题]

关于 JavaScript 里的 xml 处理，以下说明正确的是（ ）

- 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

来自：前端开发工程师练习题

答案：BCD

239.

[问答题]

列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个。

来自：前端开发工程师练习题

参考：

对象：Window document location screen history navigator

方法：Alert() confirm() prompt() open() close()

240.

[问答题]

简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法并做简单说明。

来自：前端开发工程师练习题

参考：

Document.getElementById 根据元素 id 查找元素

Document.getElementsByName 根据元素 name 查找元素

Document.getElementsByTagName 根据指定的元素名查找元素

241.

[问答题]

JavaScript 原型，原型链？有什么特点？

来自：前端开发工程师练习题

参考：无

242.

[问答题]

eval 是做什么的？

来自：前端开发工程师练习题

参考：

它的功能是把对应的字符串解析成 JS 代码并运行； 应该避免使用 eval，不安全，非常耗性能（2 次，一次解析成 js 语句，一次执行）。

243.

[问答题]

null, undefined 的区别？

来自：前端开发工程师练习题

参考：无

244.

[问答题]

Node.js 的适用场景？

来自：前端开发工程师练习题

提示：

高并发、聊天、实时消息推送

245.

[问答题]

介绍 js 的基本数据类型

来自：前端开发工程师练习题

提示：

number,string,boolean,object,undefined

246.

[问答题]

Javascript 如何实现继承？

来自：前端开发工程师练习题

提示：

通过原型和构造器

247.

[问答题]

[“1” , “2” , “3”].map(parseInt) 答案是多少？

来自：前端开发工程师练习题

参考：

[1, NaN, NaN]

因为 parseInt 需要两个参数 (val, radix)，其中 radix 表示解析时用的基数。map 传了 3 个(element, index, array)，对应的 radix 不合法导致解析失败。

248.

[问答题]

如何创建一个对象？（画出此对象的内存图）

来自：前端开发工程师练习题

参考：

```
function Person(name, age) {  
    this.name = name;  
    this.age = age;  
    this.sing = function() { alert(this.name) }  
}
```

249.

[问答题]

谈谈 `this` 对象的理解。

来自：前端开发工程师练习题

参考：

`this` 是 js 的一个关键字，随着函数使用场合不同，`this` 的值会发生变化。

但是有一个总原则，那就是 `this` 指的是调用函数的那个对象。

`this` 一般情况下：是全局对象 Global。作为方法调用，那么 `this` 就是指这个对象

250.

[问答题]

什么是闭包（closure），为什么要用它？

来自：前端开发工程师练习题

参考：

执行 `say667()` 后，`say667()` 闭包内部变量会存在，而闭包内部函数的内部变量不会存在。使得 Javascript 的垃圾回收机制 GC 不会收回 `say667()` 所占用的资源，因为 `say667()` 的内部函数的执行需要依赖 `say667()` 中的变量。这是对闭包作用的非常直白的描述。

```
function say667() {  
    // Local variable that ends up within closure  
    var num = 666;  
    var sayAlert = function() { alert(num); }  
    num++;  
    return sayAlert;  
}  
var sayAlert = say667();  
sayAlert() // 执行结果应该弹出的 667
```

251.

[问答题]

“use strict”;是什么意思？使用它的好处和坏处分别是什么？

来自：前端开发工程师练习题

参考：无

252.

[问答题]

如何判断一个对象是否属于某个类？

来自：前端开发工程师练习题

参考：

使用 instanceof (待完善)

```
if(a instanceof Person){  
    alert('yes');  
}
```

253.

[问答题]

new 操作符具体干了什么呢？

来自：前端开发工程师练习题

参考：

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this。

```
var obj = {};  
obj.__proto__ = Base.prototype;  
Base.call(obj);
```

254.

[问答题]

Javascript 中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

来自：前端开发工程师练习题

参考：

hasOwnProperty

255.

[问答题]

JSON 的了解？

来自：前端开发工程师练习题

参考：

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

它是基于 JavaScript 的一个子集。数据格式简单，易于读写，占用带宽小

{'age':'12', 'name':'back'}

256.

[问答题]

js 延迟加载的方式有哪些？

来自：前端开发工程师练习题

参考：

defer 和 async、动态创建 DOM 方式（用得最多）、按需异步载入 js

257.

[问答题]

模块化怎么做？立即执行函数,不暴露私有成员

来自：前端开发工程师练习题

参考：

```
var module1 = (function(){
    var _count = 0;
    var m1 = function(){
        //...
    };
    var m2 = function(){
```

```
//...
};

return {
    m1 : m1,
    m2 : m2
};

})();
```

258.

[问答题]

AMD (Modules/Asynchronous-Definition)、CMD (Common Module Definition) 规范区别？

来自：前端开发工程师练习题

参考：无

259.

[问答题]

异步加载的方式有哪些？

来自：前端开发工程师练习题

参考：

(1) defer，只支持 IE

(2) async:

(3) 创建 script，插入到 DOM 中，加载完毕后 callBack

260.

[问答题]

document.write 和 innerHTML 的区别。

来自：前端开发工程师练习题

参考：

document.write 只能重绘整个页面

innerHTML 可以重绘页面的一部分

261.

[问答题]

.call() 和 .apply() 的区别?

来自：前端开发工程师练习题

参考：

例子中用 add 来替换 sub，`add.call(sub,3,1) == add(3,1)`，所以运行结果为：`alert(4);`

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

```
function add(a,b)
{
    alert(a+b);
}
function sub(a,b)
{
    alert(a-b);
}
add.call(sub,3,1);
```

262.

[问答题]

Jquery 与 jQuery UI 有啥区别？

来自：前端开发工程师练习题

参考：

*jQuery 是一个 js 库，主要提供的功能是选择器，属性修改和事件绑定等等。

*jQuery UI 则是在 jQuery 的基础上，利用 jQuery 的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

263.

[问答题]

JQuery 的源码看过吗？能不能简单说一下它的实现原理？

来自：前端开发工程师练习题

参考：

jquery 中如何将数组转化为 json 字符串，然后再转化回来？

JQuery 中没有提供这个功能，所以你需要先编写两个 jquery 的扩展：

```
$.fn.stringifyArray = function(array) {  
    return JSON.stringify(array)  
}  
$.fn.parseArray = function(array) {  
    return JSON.parse(array)  
}  
然后调用：  
$("") stringifyArray(array)
```

264.

[问答题]

针对 jQuery 的优化方法？

来自：前端开发工程师练习题

参考：

- * 基于 Class 的选择性的性能相对于 Id 选择器开销很大，因为需遍历所有 DOM 元素。
- * 频繁操作的 DOM，先缓存起来再操作。用 Jquery 的链式调用更好。

如： var str=\$("a").attr("href"); *for (var i = size; i < arr.length; i++) {}
for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：

```
for (var i = size, length = arr.length; i < length; i++) {}
```

265.

[问答题]

JavaScript 中的作用域与变量声明提升？

来自：前端开发工程师练习题

参考：无

266.

[问答题]

如何编写高性能的 Javascript？

来自：前端开发工程师练习题

参考：无

267.

[问答题]

那些操作会造成内存泄漏？

来自：前端开发工程师练习题

参考：

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的唯一引用是循环的，那么该对象的内存即可回收。

`setTimeout` 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

268.

[问答题]

JQuery 一个对象可以同时绑定多个事件，这是如何实现的？

来自：前端开发工程师练习题

参考：无

269.

[问答题]

如何判断当前脚本运行在浏览器还是 node 环境中？

来自：阿里前端开发工程师面试题

参考：

通过判断 Global 对象是否为 `window`，如果不为 `window`，当前脚本没有运行在浏览器中

270.

[问答题]

对 Node 的优点和缺点提出了自己的看法？

来自：前端开发工程师练习题

参考：

* (优点) 因为 Node 是基于事件驱动和无阻塞的，所以非常适合处理并发请求，因此构建在 Node 上的代理服务器相比其他技术实现(如 Ruby)的服务器表现要好得多。此外，与 Node 代理服务器交互的客户端代码是由 javascript 语言编写的，因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

* (缺点) Node 是一个相对新的开源项目，所以不太稳定，它总是一直在变，而且缺少足够多的第三方库支持。看起来，就像是 Ruby/Rails 当年的样子。

271.

[问答题]

你有哪些性能优化的方法？（看雅虎 14 条性能优化原则）

来自：前端开发工程师练习题

参考：

(1) 减少 http 请求次数：CSS Sprites, JS, CSS 源码压缩、图片大小控制合适；网页 Gzip, CDN 托管, data 缓存，图片服务器。

(2) 前端模板 JS+数据，减少由于 HTML 标签导致的带宽浪费，前端用变量保存 AJAX 请求结果，每次操作本地变量，不用请求，减少请求次数

(3) 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能。

(4) 当需要设置的样式很多时设置 className 而不是直接操作 style。

(5) 少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作。

(6) 避免使用 CSS Expression (css 表达式)又称 Dynamic properties(动态属性)。

(7) 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

(8) 避免在页面的主体布局中使用 table，table 要等其中的内容完全下载之后才会显示出来，显示比 div+css 布局慢。

272.

[问答题]

http 状态码有那些？分别代表是什么意思？

来自：前端开发工程师练习题

参考：

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。

400 1、语义有误，当前请求无法被服务器理解。

401 当前请求需要用户验证

403 服务器已经理解请求，但是拒绝执行它。
500-599 用于支持服务器错误。 503 - 服务不可用

273.

[问答题]

一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）

来自：前端开发工程师练习题

参考：

查找浏览器缓存

DNS 解析、查找该域名对应的 IP 地址、重定向（301）、发出第二个 GET 请求

进行 HTTP 协议会话

客户端发送报头(请求报头)

服务器回馈报头(响应报头)

html 文档开始下载

文档树建立，根据标记请求所需指定 MIME 类型的文件

文件显示

{浏览器这边做的工作大致分为以下几步：

加载：根据请求的 URL 进行域名解析，向服务器发起请求，接收文件（HTML、JS、CSS、图象等）。

解析：对加载到的资源（HTML、JS、CSS 等）进行语法解析，建议相应的内部数据结构（比如 HTML 的 DOM 树，JS 的（对象）属性表，CSS 的样式规则等等）}

274.

[问答题]

JavaScript 是一门什么样的语言，它有哪些特点？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考： 没有标准答案。

275.

[问答题]

JavaScript 的数据类型都有什么？如何判断某变量是否为数组数据类型？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

基本数据类型：String,boolean,Number,Undefined, Null

引用数据类型：Object(Array,Date,RegExp,Function)

判断某变量是否为数组数据类型：

- 方法一.判断其是否具有“数组性质”，如 slice()方法。可自己给该变量定义 slice 方法，故有时会失效

•方法二.`obj instanceof Array` 在某些 IE 版本中不正确

- 方法三.方法一二皆有漏洞，在 ECMA Script5 中定义了新方法 `Array.isArray()`，保证其兼容性，最好的方法如下：

```
if(typeof Array.isArray==="undefined")  
{  
    Array.isArray = function(arg){  
        return Object.prototype.toString.call(arg)==="[object Array]"  
    };  
}
```

276.

[问答题]

已知 ID 的 Input 输入框，希望获取这个输入框的输入值，怎么做？(不使用第三方框架)

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
document.getElementById("ID").value
```

277.

[问答题]

希望获取到页面中所有的 checkbox 怎么做？(不使用第三方框架)

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
var domList = document.getElementsByTagName('input')  
var checkBoxList = [];  
var len = domList.length;      //缓存到局部变量  
while (len--) {      //使用 while 的效率会比 for 循环更高  
    if (domList[len].type ==  'checkbox' ){  
        checkBoxList.push(domList[len]);  
    }  
}
```

278.

[问答题]

设置一个已知 ID 的 DIV 的 html 内容为 xxxx，字体颜色设置为黑色(不使用第三方框架)

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
var dom = document.getElementById("ID");
dom.innerHTML = "xxxx"
dom.style.color = "#000"
```

279.

[问答题]

当一个 DOM 节点被点击时候，我们希望能够执行一个函数，应该怎么做？Javascript 的事件流模型都有什么？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

- 直接在 DOM 里绑定事件：`<div onclick=" test()"></div>`
- 在 JS 里通过 `onclick` 绑定：`xxx.onclick = test`
- 通过事件添加进行绑定：`addEventListener(xxx, 'click', test)`

Javascript 的事件流模型：

- “事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播
- “事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的
- “DOM 事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

280.

[问答题]

什么是 Ajax 和 JSON，它们的优缺点。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

Ajax 是异步 JavaScript 和 XML，用于在 Web 页面中实现异步数据交互。

优点：

- 可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量
- 避免用户不断刷新或者跳转页面，提高用户体验

缺点：

- 对搜索引擎不友好（
- 要实现 ajax 下的前后退功能成本较大
- 可能造成请求数的增加
- 跨域问题限制

JSON 是一种轻量级的数据交换格式，ECMA 的一个子集。

优点：轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）

281.

[问答题]

看下列代码输出为何？解释原因。

```
var a;  
alert(typeof a); // undefined  
alert(b); // 报错
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

`Undefined` 是一个只有一个值的数据类型，这个值就是“`undefined`”，在使用 `var` 声明变量但并未对其赋值进行初始化时，这个变量的值就是 `undefined`。而 `b` 由于未声明将报错。注意未申明的变量和声明了未赋值的是不一样的。

282.

[问答题]

看下列代码，输出什么？解释原因。

```
var a = null;  
alert(typeof a); // object
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

`null` 是一个只有一个值的数据类型，这个值就是 `null`。表示一个空指针对象，所以用 `typeof` 检测会返回“`object`”。

283.

[问答题]

看下列代码，输出什么？解释原因。

```
var undefined;  
undefined == null; // true
```

```
1 == true;    // true
2 == true;    // false
0 == false;   // true
0 == "";      // true
NaN == NaN;   // false
[] == false;  // true
[] == ![];    // true
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

- undefined 与 null 相等，但不恒等 (==)
- 一个是 number 一个是 string 时，会尝试将 string 转换为 number
- 尝试将 boolean 转换为 number, 0 或 1
- 尝试将 Object 转换成 number 或 string，取决于另外一个对比量的类型
- 所以，对于 0、空字符串的判断，建议使用 “==” 。 “==” 会先判断两边的值类型，类型不匹配时为 false。

考官会继续追问，看下面的代码，输出什么， foo 的值为什么？

```
var foo = "11"+2-"1";
console.log(foo);
console.log(typeof foo);
```

执行完后 foo 的值为 111， foo 的类型为 String。

284.

[问答题]

看代码给答案。

```
var a = new Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

答案：2（考察引用数据类型细节）

285.

[问答题]

已知数组 var stringArray = [“This”， “is”， “Baidu”， “Campus”]， Alert 出”This is Baidu Campus”。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

答案：alert(stringArray.join(" "))

286.

[问答题]

已知有字符串 `foo=“get-element-by-id”`,写一个 `function` 将其转化成驼峰表示法“`getElementById`”。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
function combo(msg){  
    var arr=msg.split("-");  
    for(var i=1;i<arr.length;i++){  
        arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);  
    }  
    msg=arr.join("");  
    return msg;  
}  
(考察基础 API)
```

287.

[问答题]

`var numberArray = [3,6,2,4,1,5];` （考察基础 API）

- 1) 实现对该数组的倒排，输出`[5,1,4,2,6,3]`
 - 2) 实现对该数组的降序排列，输出`[6,5,4,3,2,1]`
-

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
function combo(msg){  
    var arr=msg.split(" ");  
    for(var i=1;i<arr.length;i++){  
        arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);  
    }  
    msg=arr.join("");  
    return msg;  
}
```

288.

[问答题]

输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2014 年 9 月 26 日，则输出
2014-09-26

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
var d = new Date();
// 获取年, getFullYear()返回 4 位的数字
var year = d.getFullYear();
// 获取月, 月份比较特殊, 0 是 1 月, 11 是 12 月
var month = d.getMonth() + 1;
// 变成两位
month = month < 10 ? '0' + month : month;
// 获取日
var day = d.getDate();
day = day < 10 ? '0' + day : day;
alert(year + '-' + month + '-' + day);
```

289.

[问答题]

将字符串”<tr><td>{\$id}</td><td>{\$name}</td></tr>”中的{\$id}替换成 10, {\$name}替换成 Tony（使用正则表达式）

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

答 案：“<tr><td>{\$id}</td><td>{\$id}_{\$name}</td></tr>”.replace(/\{\$id\}/g, '10').replace(/\{\$name\}/g, 'Tony');

290.

[问答题]

为了保证页面输出安全，我们经常需要对一些特殊的字符进行转义，请写一个函数 escapeHtml，将<, >, &, “进行转义

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
function escapeHtml(str) {
```

```
return str.replace(/[<>"&]/g, function(match) {
    switch (match) {
        case "<":
            return "&lt;";
        case ">":
            return "&gt;";
        case "&":
            return "&amp;";
        case "\'":
            return "&quot;";
    }
});
```

291.

[问答题]

`foo = foo || bar`，这行代码是什么意思？为什么要这样写？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

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。

292.

[问答题]

看下列代码，将会输出什么?(变量声明提升)

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

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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。

293.

[问答题]

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。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
var iArray = [];
function getRandom(istart, iend){
    var iChoice = istart - iend +1;
    return Math.floor(Math.random() * iChoice + istart);
}
for(var i=0; i<10; i++){
    iArray.push(getRandom(10,100));
}
iArray.sort();
```

294.

[问答题]

把两个数组合并，并删除第二个元素。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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

295.

[问答题]

怎样添加、移除、移动、复制、创建和查找节点（原生 JS）

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

1) 创建新节点

```
createDocumentFragment() //创建一个 DOM 片段
```

```
createElement() //创建一个具体的元素
```

```
createTextNode() //创建一个文本节点
```

2) 添加、移除、替换、插入

```
appendChild() //添加
```

```
removeChild() //移除
```

```
replaceChild() //替换
```

```
insertBefore() //插入
```

3) 查找

```
getElementsByName() //通过标签名称
```

```
getElementsByName() //通过元素的 Name 属性的值
```

```
getElementById() //通过元素 Id, 唯一性
```

296.

[问答题]

有这样一个 URL: <http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e>, 请写一段 JS 程序提取 URL 中的各个 GET 参数(参数名和参数个数不确定), 将其按 key-value 形式返回到一个 json 结构中, 如{a: '1' , b: '2' , c: "" , d: 'xxx' , e:undefined}。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
function serilizeUrl(url) {
    var result = {};
    url = url.split("?")[1];
    var map = url.split("&");
    for(var i = 0, len = map.length; i < len; i++) {
        result<script>jQuery(function($)
            ${"#google-maps-1").gMap({controls:
false,scrollwheel: false,markers: [{address: "",icon: {image:
"http://blog.jobbole.com/wp-content/themes/jobboleblogv3/_assets/img/_colors/red/pin.png",i
consize: [32, 32],iconanchor: [16,27],infowindowanchor: [16, 27]}]},address: "",zoom: 15,icon:
{image:
"http://blog.jobbole.com/wp-content/themes/jobboleblogv3/_assets/img/_colors/red/pin.png",i
consize: [32, 32],iconanchor: [16,27],infowindowanchor: [16, 27]}});});</script><div
class="google-maps" id="google-maps-1" style="width: 100%; height:
200px;"></div>.split("=")[0]] = map[i].split("=")[1];
    }
    return result;
}
```

```
}
```

297.

[问答题]

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？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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\}$/;
```

298.

[问答题]

看下面代码，给出输出结果。

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

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

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
```

299.

[问答题]

写一个 function，清除字符串前后的空格。（兼容所有浏览器）

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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+$/, "");  
    }  
}  
  
// test the function  
var str = "\t\n test string ".trim();  
alert(str == "test string"); // alerts "true"
```

300.

[问答题]

avascript 中 callee 和 caller 的作用？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

caller 是返回一个对函数的引用，该函数调用了当前函数；

callee 是返回正在被执行的 function 函数，也就是所指定的 function 对象的正文。

那么问题来了？如果一对兔子每月生一对兔子；一对新生兔，从第二个月起就开始生兔子；假定每对兔子都是一雌一雄，试问一对兔子，第 n 个月能繁殖成多少对兔子？（使用 callee 完成）

```
var result=[];  
function fn(n){ //典型的斐波那契数列  
    if(n==1){  
        return 1;  
    }else if(n==2){  
        return 1;  
    }else{  
        if(result[n]){  
            return result[n];  
        }else{  
            //argument.callee()表示 fn()  
            result[n]=arguments.callee(n-1)+arguments.callee(n-2);  
        }  
    }  
}
```

```

        return result[n];
    }
}
}

```

301.

[问答题]

实现一个函数 clone，可以对 JavaScript 中的 5 种主要的数据类型（包括 Number、String、Object、Array、Boolean）进行值复制。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

- 考察点 1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚

- 考察点 2：是否知道如何判断一个变量是什么类型的

- 考察点 3：递归算法的设计

// 方法一：

```

Object.prototype.clone = function(){
    var o = this.constructor === Array ? [] : {};
    for(var e in this){
        o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];
    }
    return o;
}

```

//方法二：

```

/**
 * 克隆一个对象
 * @param Obj
 * @returns
 */
function clone(Obj) {
    var buf;
    if (Obj instanceof Array) {
        buf = [];
                    //创建一个空的数组
        var i = Obj.length;
        while (i--) {
            buf[i] = clone(Obj[i]);
        }
        return buf;
    }else if (Obj instanceof Object){
        buf = {};
                    //创建一个空对象
        for (var k in Obj) {
                    //为这个对象添加新的属性
            buf[k] = clone(Obj[k]);
        }
        return buf;
    }
}

```

```
        buf[k] = clone(Obj[k]);
    }
    return buf;
}else{ //普通变量直接赋值
    return Obj;
}
}
```

302.

[问答题]

如何消除一个数组里面重复的元素？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];
function deRepeat(){
    var newArr=[];
    var obj={};
    var index=0;
    var l=arr.length;
    for(var i=0;i<l;i++){
        if(obj[arr[i]]==undefined)
        {
            obj[arr[i]]=1;
            newArr[index++]=arr[i];
        }
        else if(obj[arr[i]]==1)
            continue;
    }
    return newArr;
}
var newArr2=deRepeat(arr);
alert(newArr2); //输出 1,2,3,4,5,6,9,25
```

303.

[问答题]

小贤是一条可爱的小狗(Dog)，它的叫声很好听(wow)，每次看到主人的时候就会乖乖叫一声(yelp)。从这段描述可以得到以下对象：

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
function Dog() {
    this.wow = function() {
        alert('Wow');
    }
    this.yelp = function() {
        this.wow();
    }
}
```

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了(MadDog)，一看到人就会每隔半秒叫一声(wow)地不停叫唤(yelp)。请根据描述，按示例的形式用代码来实现。(继承，原型，setInterval)

答案：

```
function MadDog() {
    this.yelp = function() {
        var self = this;
        setInterval(function() {
            self.wow();
        }, 500);
    }
}
MadDog.prototype = new Dog();
//for test
var dog = new Dog();
dog.yelp();
var madDog = new MadDog();
madDog.yelp();
```

304.

[问答题]

下面这个 ul，如何点击每一列的时候 alert 其 index? (闭包)

```
<ul id="test">
<li>这是第一条</li>
<li>这是第二条</li>
<li>这是第三条</li>
</ul>
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

// 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);
}

```

305.

[问答题]

编写一个 JavaScript 函数，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。

```
/** @param selector {String} 传入的 CSS 选择器。 * @return {Array}*/
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```

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;
    }
}

```

```
}
```

306.

[问答题]

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);
        });
    }
}
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

- 不应该在 if 和 else 语句中声明 addListener 函数，应该先声明；
- 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；

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;
    }
}
```

307.

[问答题]

给 String 对象添加一个方法，传入一个 string 类型的参数，然后将 string 的每个字符间价格空格返回，例如：

```
addSpace("hello world") // -> 'h e l l o   w o r l d'
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

```
String.prototype.spacify = function(){
    return this.split('').join(' ');
};
```

接着上述问题答案提问，1) 直接在对象的原型上添加方法是否安全？尤其是在 Object 对象上。(这个我没能答出？希望知道的说一下。) 2) 函数声明与函数表达式的区别？

答案：

在 js 中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

308.

[问答题]

定义一个 log 方法，让它可以代理 console.log 的方法。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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。

答案：

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])`。

309.

[问答题]

在 Javascript 中什么是伪数组？如何将伪数组转化为标准数组？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

伪数组（类数组）：无法直接调用数组方法或期望 `length` 属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的 `argument` 参数，还有像调用 `getElementsByTagName`,`document.childNodes` 之类的，它们都返回 `NodeList` 对象都属于伪数组。可以使用 `Array.prototype.slice.call(fakeArray)` 将数组转化为真正的 `Array` 对象。

假设接第八题题干，我们要给每个 `log` 方法添加一个”(app)”前缀，比如’hello world!’->’(app)hello world!’。方法如下：

```
function log(){
    var args = Array.prototype.slice.call(arguments); //为了使用 unshift 数组方法，将
    argument 转化为真正的数组
    args.unshift('(app)');
    console.log.apply(console, args);
}
```

310.

[问答题]

对作用域上下文和 `this` 的理解，看下列代码：

```
var User = {
    count: 1,
    getCount: function() {
        return this.count;
    }
};
console.log(User.getCount()); // what?
var func = User.getCount;
console.log(func()); // what?
问两处 console 输出什么？为什么？
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

1 和 undefined。

func 是在 window 的上下文中被执行的，所以会访问不到 count 属性。

继续追问，那么如何确保 User 总是能访问到 func 的上下文，即正确返回 1。正确的方法是使用 Function.prototype.bind。兼容各个浏览器完整代码如下：

```
Function.prototype.bind = Function.prototype.bind || function(context){  
    var self = this;  
    return function(){  
        return self.apply(context, arguments);  
    };  
}  
  
var func = User.getCount.bind(User);  
console.log(func());
```

311.

[问答题]

原生 JS 的 window.onload 与 Jquery 的 \$(document).ready(function(){})) 有什么不同？如何用原生 JS 实现 Jq 的 ready 方法？

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

window.onload()方法是必须等到页面内包括图片的所有元素加载完毕后才能执行。

\$(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。

```
/*  
 * 传递函数给 whenReady()  
 * 当文档解析完毕且为操作准备就绪时，函数作为 document 的方法调用  
 */  
  
var whenReady = (function() { //这个函数返回 whenReady() 函数  
    var funcs = []; //当获得事件时，要运行的函数  
    var ready = false; //当触发事件处理程序时，切换为 true  
    //当文档就绪时，调用事件处理程序  
    function handler(e) {  
        if(ready) return; //确保事件处理程序只完整运行一次  
        //如果发生 onreadystatechange 事件，但其状态不是 complete 的话，那么文档  
        尚未准备好  
        if(e.type === 'onreadystatechange' && document.readyState !== 'complete') {  
            return;  
        }  
        //运行所有注册函数  
        //注意每次都要计算 funcs.length  
        //以防这些函数的调用可能会导致注册更多的函数  
        for(var i=0; i<funcs.length; i++) {  
            funcs[i].call(document);  
        }  
    }  
    return handler;  
});
```

```

        }
        //事件处理函数完整执行,切换 ready 状态, 并移除所有函数
        ready = true;
        funcs = null;
    }
    //为接收到的任何事件注册处理程序
    if(document.addEventListener) {
        document.addEventListener('DOMContentLoaded', handler, false);
        document.addEventListener('readystatechange', handler, false);
    } //IE9+
    window.addEventListener('load', handler, false);
} else if(document.attachEvent) {
    document.attachEvent('onreadystatechange', handler);
    window.attachEvent('onload', handler);
}
//返回 whenReady()函数
return function whenReady(fn) {
    if(ready) { fn.call(document); }
    else { funcs.push(fn); }
}
})();
如果上述代码十分难懂, 下面这个简化版:
function ready(fn){
    if(document.addEventListener) { //标准浏览器
        document.addEventListener('DOMContentLoaded', function() {
            //注销事件, 避免反复触发
            document.removeEventListener('DOMContentLoaded', arguments.callee,
false);
            fn(); //执行函数
        }, false);
    } else if(document.attachEvent) { //IE
        document.attachEvent('onreadystatechange', function() {
            if(document.readyState == 'complete') {
                document.detachEvent('onreadystatechange', arguments.callee);
                fn(); //函数执行
            }
        });
    }
};

```

312.

[问答题]

(设计题) 想实现一个对页面某个节点的拖曳? 如何做? (使用原生 JS)

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

回答出概念即可，下面是几个要点

- 1.给需要拖拽的节点绑定 mousedown, mousemove, mouseup 事件
- 2.mousedown 事件触发后，开始拖拽
- 3.mousemove 时，需要通过 event.clientX 和 clientY 获取拖拽位置，并实时更新位置
- 4.mouseup 时，拖拽结束
- 5.需要注意浏览器边界的情况

313.

[问答题]

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('<p><a href="{0}">{1}</a><span>{2}</span></p>');
    console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));
})();
```

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：

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

314.

[问答题]

用面向对象的 Javascript 来介绍一下自己。

来自：BAT 及各大互联网公司 2014 前端笔试面试题集

参考：无

315.

[问答题]

实现输出 document 对象中所有成员的名称和类型。

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

316.

[问答题]

如何获得一个 DOM 元素的绝对位置？（获得元素位置，不依赖框架）

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

317.

[问答题]

如何利用 JS 生成一个 table？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

318.

[问答题]

实现预加载一张图片，加载完成后显示在网页中并设定其高度为 50px，宽度为 50px；

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

319.

[问答题]

假设有一个 4 行 td 的 table，将 table 里面 td 顺序颠倒。

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

320.

[问答题]

模拟一个 HashTable 类，包含有 add、remove、contains、length 方法。

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

321.

[问答题]

Ajax 读取一个 XML 文档并进行解析的实例。

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

322.

[问答题]

JS 如何实现面向对象和继承机制？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

323.

[问答题]

JS 模块的封装方法，比如怎样实现私有变量，不能直接赋值，只能通过公有方法。

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

324.

[问答题]

对闭包的理解，闭包的好处和坏处？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

325.

[问答题]

对 this 指针的理解，可以列举几种使用情况？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

326.

[问答题]

对 JS 中函数绑定的理解？函数绑定可以使用哪两个函数？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题

参考：无

327.

[问答题]

jQuery 的特点？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

328.

[问答题]

简述 Ajax 的异步机制。Ajax 有哪些好处和弊端？

来自：百度流程信息管理部 Web 前端实习研发工程师笔试题
参考：无

329.

[问答题]

阅读下面代码，判断输出结果？

考察知识点： this。

```
var length = 10;
function fn() {
    console.log(this.length);
}

var obj = {
    length: 5,
    method: function(fn) {
        fn();
        arguments[0]();
    }
};

obj.method(fn, 1);
```

来自：2015 年十大热点 Javascript 笔试题

输出：10 2

第一次输出 10 应该没有问题。我们知道取对象属于除了点操作符还可以用中括号，所以第二次执行时相当于 arguments 调用方法，this 指向 arguments，而这里传了两个参数，故输出 arguments 长度为 2。

330.

[问答题]

阅读下面代码，判断输出结果？

考察知识点：var 和函数的提前声明

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

来自：2015 年十大热点 Javascript 笔试题

输出：function a() {} 2

我们知道 var 和 function 是会提前声明的，而且 function 是优先于 var 声明的(如果同时存在的话)，所以提前声明后输出的 a 是个 function，然后代码往下执行 a 进行重新赋值了，故第二次输出是 2。

331.

[问答题]

阅读下面代码，判断输出结果？

考察知识点：局部变量和全局变量

```
var f = true;  
if (f === true) {  
    var a = 10;  
}  
  
function fn() {  
    var b = 20;  
    c = 30;  
}  
  
fn();  
console.log(a);  
console.log(b);  
console.log(c);
```

来自：2015 年十大热点 Javascript 笔试题

输出：10 报错 30

这是个我犯了很久的错误，很长一段时间我都以为{}内的新声明的变量是局部变量，后来我才发现 function 内的新声明的变量才是局部变量，而没有用 var 声明的变量在哪里都是全局变量。再次提醒切记只有 function(){}内新声明的才能是局部变量，while{}、if{}、for(..) 之内的都是全局变量(除非本身包含在 function 内)。

332.

[问答题]

阅读下面代码，判断输出结果？

考察知识点：变量隐式声明

```
var f = true;
if (f === true) {
    var a = 10;
}
```

```
function fn() {
    var b = 20;
    c = 30;
}
```

```
fn();
console.log(a);
console.log(b);
console.log(c);
```

来自：2015 年十大热点 Javascript 笔试题

答案：10

前面说过 function 和 var 会提前声明，而其实{}内的变量也会提前声明。于是代码还没执行前，a 变量已经被声明，于是 'a' in window 返回 true，a 被赋值。

333.

[问答题]

阅读下面代码，判断输出结果？

考察知识点：给基本类型数据添加属性，不报错，但取值时是 undefined

```
var a = 10;
a.pro = 10;
console.log(a.pro + a);
```

```
var s = 'hello';
s.pro = 'world';
```

```
console.log(s.pro + s);
```

来自：2015年十大热点 Javascript 笔试题

答案：NaN undefinedhello

给基本类型数据加属性不报错，但是引用的话返回 undefined，10+undefined 返回 NaN，而 undefined 和 string 相加时转变成了字符串。

334.

[问答题]

阅读下面代码，判断输出结果？

考察知识点：函数声明优于变量声明

```
console.log(typeof fn);
function fn() {};
var fn;
```

来自：2015年十大热点 Javascript 笔试题

答案：function

因为函数声明优于变量声明。我们知道在代码逐行执行前，函数声明和变量声明会提前进行，而函数声明又会优于变量声明，这里的优于可以理解为晚于变量声明后，如果函数名和变量名相同，函数声明就能覆盖变量声明。所以上述代码将函数声明和变量声明调换顺序还是一样结果。

335.

[问答题]

判断一个字符串中出现次数最多的字符，并统计次数。

来自：2015年十大热点 Javascript 笔试题

参考：

hash table 方式：

```
var s = 'aaabbcccaaabbbaaa';
var obj = {};
var maxn = -1;
var letter;
for(var i = 0; i < s.length; i++) {
    if(obj[s[i]]) {
        obj[s[i]]++;
        if(obj[s[i]] > maxn) {
            maxn = obj[s[i]];
```

```

        letter = s[i];
    }
} else {
    obj[s[i]] = 1;
    if(obj[s[i]] > maxn) {
        maxn = obj[s[i]];
        letter = s[i];
    }
}
}

alert(letter + ':' + maxn);

```

正则方式：

```

var s = 'aaabbcccaaabbbaaabbbbbbbb';
var a = s.split('');
a.sort();
s = a.join('');
var pattern = /(\w)\1*/g;
var ans = s.match(pattern);
ans.sort(function(a, b) {
    return a.length < b.length;
});
console.log(ans[0][0] + ':' + ans[0].length);

```

336.

[问答题]

设计经典闭包。

来自：2015 年十大热点 Javascript 笔试题

参考：

```

<!-- 实现一段脚本，使得点击对应链接 alert 出相应的编号 -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<body>
    <a href='#'> 第一个链接 </a> <br>
    <a href='#'> 第二个链接 </a> <br>
    <a href='#'> 第三个链接 </a> <br>
    <a href='#'> 第四个链接 </a> <br>
</body>

```

dom 污染法：

```

<!-- 实现一段脚本，使得点击对应链接 alert 出相应的编号 -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<body>

```

```

<a href="#"> 第一个链接 </a><br>
<a href="#"> 第二个链接 </a><br>
<a href="#"> 第三个链接 </a><br>
<a href="#"> 第四个链接 </a><br>
<script type="text/javascript">
    var lis = document.links;
    for(var i = 0, length = lis.length; i < length; i++) {
        lis[i].index = i;
        lis[i].onclick = function() {
            alert(this.index);
        };
    }
</script>
</body>
闭包：
<!-- 实现一段脚本，使得点击对应链接 alert 出相应的编号 -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<body>
    <a href="#"> 第一个链接 </a><br>
    <a href="#"> 第二个链接 </a><br>
    <a href="#"> 第三个链接 </a><br>
    <a href="#"> 第四个链接 </a><br>
    <script type="text/javascript">
        var lis = document.links;
        for(var i = 0, length = lis.length; i < length; i++) {
            (function(i) {
                lis[i].onclick = function() {
                    alert(i + 1);
                };
            })(i);
        }
    </script>
</body>

```

337.

[问答题]

阅读下面代码，判断输出结果？

考察知识点： this

```

function JSClass() {
    this.m_Text = 'division element';
    this.m_Element = document.createElement('div');
    this.m_Element.innerHTML = this.m_Text;
    this.m_Element.addEventListener('click', this.func);
}

```

```
// this.m_Element.onclick = this.func;
}

JSClass.prototype.Render = function() {
    document.body.appendChild(this.m_Element);
}

JSClass.prototype.func = function() {
    alert(this.m_Text);
};

var jc = new JSClass();
jc.Render(); // add div
jc.func(); // 输出 division element
//click 添加的 div 元素 division element 会输出 undefined，为什么？
```

来自：2015 年十大热点 Javascript 笔试题

答案：division element undefined

第一次输出很好理解，第二次的话仔细看，this 其实已经指向了 this.m_Element，因为是 this.m_Element 调用的 addEventListener 函数，所以内部的 this 全指向它了。可以试着加上一行代码 this.m_Element.m_Text = 'hello world'，就会 alert 出 hello world 了。

338.

[问答题]

请编写一个 JavaScript 函数 parseQueryString，它的用途是把 URL 参数解析为一个对象，如： var url = “http://witmax.cn/index.php?key0=0&key1=1&key2=2”

来自：2015 年十大热点 Javascript 笔试题

参考：

```
function parseQueryString(url) {
    var obj = {};
    var a = url.split('?');
    if(a === 1) return obj;
    var b = a[1].split('&');
    for(var i = 0, length = b.length; i < length; i++) {
        var c = b[i].split('=');
        obj[c[0]] = c[1];
    }
    return obj;
}
var url = 'http://witmax.cn/index.php?key0=0&key1=1&key2=2';
```

```
var obj = parseQueryString(url);
console.log(obj.key0, obj.key1, obj.key2); // 0 1 2
```

339.

[问答题]

找出数字数组中最大的元素(使用 Math.max 函数)。

来自：外国 Javascript 编程题精选

参考：

```
var a = [1, 2, 3, 6, 5, 4];
var ans = Math.max.apply(null, a);
console.log(ans); // 6
```

这题很巧妙地用了 apply，如果不是数组，是很多数字求最大值，我们知道可以这样：

```
var ans = Math.max(1, 2, 3, 4, 5, 6);
console.log(ans); // 6
```

而 apply 的第二个参数正是一个数组，都不用进行转换了。

```
var a = [1, 2, 3, 6, 5, 4];
var ans = eval('Math.max(' + a.toString() + ')');
console.log(ans); // 6
```

还有一种用 eval+toString 的实现：

340.

[问答题]

转化一个数字数组为 function 数组(每个 function 都弹出相应的数字)。

来自：外国 Javascript 编程题精选

参考：

```
var a = [1, 2, 3, 4, 5, 6];
var len = a.length;
for(var i = 0; i < len; i++) {
    var num = a[i];
    (function(num) {
        var f = function() {
            console.log(num);
        };
        a[i] = f;
    })(num);
}
```

```
for(var i = 0; i < len; i++)  
    a[i]();  
// 1  
// 2  
// 3  
// 4  
// 5  
// 6
```

这跟给 n 个 a 标签，弹出相应标签对应的编号是一个类型的题，用闭包保存变量到内存即可。

341.

[问答题]

给 object 数组进行排序(排序条件是每个元素对象的属性个数)。

来自：外国 Javascript 编程题精选

参考：

```
var a = {  
    name: 'hanzichi',  
    age: 10,  
    location: 'china'  
};
```

```
var b = {  
    name: 'curry'  
};
```

```
var c = {  
    name: 'kobe',  
    sex: 'male'  
};
```

```
Object.prototype.getLength = function() {  
    var num = 0;  
    for(var key in this) {  
        if(this.hasOwnProperty(key))  
            num++;  
    }  
    return num;  
};
```

```
var arr = [a, b, c];
```

```
arr.sort(function(a, b) {
    return a.getLength() > b.getLength();
});
console.log(arr);
```

这题不难，数组排序，当然是 `sort`，排序条件是对象的属性个数，可以写个函数计算，注意可能要用 `hasOwnProperty` 判断下。

342.

[问答题]

利用 JavaScript 打印出 Fibonacci 数(不使用全局变量)。

来自：外国 Javascript 编程题精选

参考：

```
(function(a, b) {
    var c = a + b;
    console.log(c);
    if(c > 100) return;
    arguments.callee(b, c);
})(-1, 1);
```

这题没看明白，是打出斐波那契数列的前 n 项么？还是第 n 项...

```
function fn(n) {
    var a = [];
    a[0] = 0, a[1] = 1;
    for(var i = 2; i < n; i++)
        a[i] = a[i - 1] + a[i - 2];
    for(var i = 0; i < n; i++)
        console.log(a[i]);
}
```

`fn(5); // 10 表示需要的斐波那契数列个数`

`// 0
// 1
// 1
// 2
// 3`

不使用全局变量，把它们写在函数里了应该算是局部变量，难道这样就好了？

343.

[问答题]

实现如下语法的功能：`var a = (5).plus(3).minus(6); //2`

来自：外国 Javascript 编程题精选

参考：

```
Number.prototype.plus = function(a) {  
    return this + a;  
};
```

```
Number.prototype.minus = function(a) {  
    return this - a;  
};
```

```
var a = (5).plus(3).minus(6);  
console.log(a); // 2
```

直接在 Number 对象上加扩展方法即可，传说中这样很不好，but 我也想不到更好的办法了...

344.

[问答题]

实现如下语法的功能： var a = add(2)(3)(4); //9

来自：外国 Javascript 编程题精选

参考：

```
function add(a) {  
    var temp = function(b) {  
        return add(a + b);  
    }  
    temp.valueOf = temp.toString = function() {  
        return a;  
    };  
    return temp;  
}  
var ans = add(2)(3)(4);  
console.log(ans); // 9
```

对 valueOf 和 toString 的考察，具体可以参考《valueOf 和 toString》

另看到一种很飘逸的写法(来自 Gaubee):

```
function add(num){  
    num += ~~add;  
    add.num = num;  
    return add;  
}
```

```
add.valueOf = add.toString = function(){return add.num};  
var ans = add(3)(4)(5)(6); // 18  
alert(ans);
```

345.

[问答题]

原生 JavaScript 实现字符串长度截取

来自：原生 Javascript 编程练习题

参考：

```
function cutstr(str, len) {  
    var temp;  
    var icount = 0;  
    var patrn = /^[^\x00-\xff]/;  
    var strre = "";  
    for (var i = 0; i < str.length; i++) {  
        if (icount < len - 1) {  
            temp = str.substr(i, 1);  
            if (patrn.exec(temp) == null) {  
                icount = icount + 1  
            } else {  
                icount = icount + 2  
            }  
            strre += temp  
        } else {  
            break  
        }  
    }  
    return strre + "..."  
}
```

346.

[问答题]

原生 JavaScript 获取域名主机

来自：原生 Javascript 编程练习题

参考：

```
function getHost(url) {  
    var host = "null";
```

```
if(typeof url == "undefined" || null == url) {  
    url = window.location.href;  
}  
var regex = /^\\w+\\:\\/\\/([\\^\\/]*)\\.*/;  
var match = url.match(regex);  
if(typeof match != "undefined" && null != match) {  
    host = match[1];  
}  
return host;  
}
```

347.

[问答题]

原生 JavaScript 清除空格

来自：原生 Javascript 编程练习题

参考：

```
String.prototype.trim = function() {  
    var reExtraSpace = /^\\s*(.*?)\\s+$/;  
    return this.replace(reExtraSpace, "$1")  
}
```

348.

[问答题]

原生 JavaScript 替换全部

来自：原生 Javascript 编程练习题

参考：

```
String.prototype.replaceAll = function(s1, s2) {  
    return this.replace(new RegExp(s1, "gm"), s2)  
}
```

349.

[问答题]

原生 JavaScript 转义 html 标签

来自：原生 Javascript 编程练习题

参考：

```
function HtmlEncode(text) {  
    return text.replace(/&/g, '&').replace(/\"/g, '\"').replace(/</g, '<').replace(/>/g, '>')  
}
```

350.

[问答题]

原生 JavaScript 时间日期格式转换

来自：原生 Javascript 编程练习题

参考：

```
Date.prototype.Format = function(formatStr) {  
    var str = formatStr;  
    var Week = ['日', '一', '二', '三', '四', '五', '六'];  
    str = str.replace(/\yyyy|YYYY/, this.getFullYear());  
    str = str.replace(/yy|YY/, (this.getYear() % 100) > 9 ? (this.getYear() % 100).toString() : '0' +  
    (this.getYear() % 100));  
    str = str.replace(/MM/, (this.getMonth() + 1) > 9 ? (this.getMonth() + 1).toString() : '0' + (th  
    is.getMonth() + 1));  
    str = str.replace(/M/g, (this.getMonth() + 1));  
    str = str.replace(/w|W/g, Week[this.getDay()]);  
    str = str.replace(/dd|DD/, this.getDate() > 9 ? this.getDate().toString() : '0' + this.getDate()  
    ;  
    str = str.replace(/d|D/g, this.getDate());  
    str = str.replace(/hh|HH/, this.getHours() > 9 ? this.getHours().toString() : '0' + this.getHou  
    rs());  
    str = str.replace(/h|H/g, this.getHours());  
    str = str.replace(/mm/, this.getMinutes() > 9 ? this.getMinutes().toString() : '0' + this.getMi  
    nutes());  
    str = str.replace(/m/g, this.getMinutes());  
    str = str.replace(/ss|SS/, this.getSeconds() > 9 ? this.getSeconds().toString() : '0' + this.getSe  
    conds());  
    str = str.replace(/s|S/g, this.getSeconds());  
    return str  
}
```

351.

[问答题]

原生 JavaScript 判断是否为数字类型

来自：原生 Javascript 编程练习题

参考：

```
function isDigit(value) {  
    var patrn = /^[0-9]*$/;  
    if (patrn.exec(value) == null || value == "") {  
        return false  
    } else {  
        return true  
    }  
}
```

352.

[问答题]

原生 JavaScript 设置 cookie 值

来自：原生 Javascript 编程练习题

参考：

```
function setCookie(name, value, Hours) {  
    var d = new Date();  
    var offset = 8;  
    var utc = d.getTime() + (d.getTimezoneOffset() * 60000);  
    var nd = utc + (3600000 * offset);  
    var exp = new Date(nd);  
    exp.setTime(exp.getTime() + Hours * 60 * 60 * 1000);  
    document.cookie = name + "=" + escape(value) + ";path=/;expires=" + exp.toGMTString() +  
    ";domain=360doc.com;"  
}
```

353.

[问答题]

原生 JavaScript 获取 cookie 值

来自：原生 Javascript 编程练习题

参考：

```
function getCookie(name) {  
    var arr = document.cookie.match(new RegExp("(^| )" + name + "=([^\;]*)(;\$)"));  
    if (arr != null) return unescape(arr[2]);
```

```
    return null  
}
```

354.

[问答题]

原生 JavaScript 加入收藏夹

来自：原生 Javascript 编程练习题

参考：

```
function AddFavorite(sURL, sTitle) {  
    try {  
        window.external.addFavorite(sURL, sTitle)  
    } catch(e) {  
        try {  
            window.sidebar.addPanel(sTitle, sURL, "")  
        } catch(e) {  
            alert("加入收藏失败，请使用 Ctrl+D 进行添加")  
        }  
    }  
}
```

355.

[问答题]

原生 JavaScript 设为首页

来自：原生 Javascript 编程练习题

参考：

```
function setHomepage() {  
    if (document.all) {  
        document.body.style.behavior = 'url(#default#homepage)';  
        document.body.setHomePage('http://www.jq-school.com')  
    } else if (window.sidebar) {  
        if (window.netscape) {  
            try {  
                netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect")  
            } catch(e) {  
                alert("该操作被浏览器拒绝，如果想启用该功能，请在地址栏内输入 about:config,然后将项 signed.applets.codebase_principal_support 值该为 true")  
            }  
        }  
    }  
}
```

```
        }
        var prefs = Components.classes['@mozilla.org/preferences-service;1'].getService(Components.interfaces.nsIPrefBranch);
        prefs.setCharPref('browser.startup.homepage', 'http://www.jq-school.com')
    }
}
```

356.

[问答题]

原生 JavaScript 判断 IE6

来自：原生 Javascript 编程练习题

参考：

```
var ua = navigator.userAgent.toLowerCase();
var isIE6 = ua.indexOf("msie 6") > -1;
if (isIE6) {
    try {
        document.execCommand("BackgroundImageCache", false, true)
    } catch(e) {}
}
```

357.

[问答题]

原生 JavaScript 加载样式文件

来自：原生 Javascript 编程练习题

参考：

```
function LoadStyle(url) {
    try {
        document.createStyleSheet(url)
    } catch(e) {
        var cssLink = document.createElement('link');
        cssLink.rel = 'stylesheet';
        cssLink.type = 'text/css';
        cssLink.href = url;
        var head = document.getElementsByTagName('head')[0];
        head.appendChild(cssLink)
    }
}
```

358.

[问答题]

原生 JavaScript 返回脚本内容

来自：原生 Javascript 编程练习题

参考：

```
function evalscript(s) {
    if(s.indexOf('<script') == -1) return s;
    var p = /<script[^>]*?(>([^\x00]*?)</script>/ig;
    var arr = [];
    while(arr = p.exec(s)) {
        var p1 = /<script[^>]*?src=\"([^\>]*?)\"[^>]*?(reload=\\"1\\")?(:charset=\"([\w\-\+?)\\\")?></script>/i;
        var arr1 = [];
        arr1 = p1.exec(arr[0]);
        if(arr1) {
            appendscript(arr1[1], ", arr1[2], arr1[3]);
        } else {
            p1 = /<script(.*)?(>([^\x00]+?)</script>/i;
            arr1 = p1.exec(arr[0]);
            appendscript('', arr1[2], arr1[1].indexOf('reload=') != -1);
        }
    }
    return s;
}
```

359.

[问答题]

原生 JavaScript 清除脚本内容

来自：原生 Javascript 编程练习题

参考：

```
function stripscript(s) {
    return s.replace(/<script.*?>.*?</script>/ig, '');
}
```

360.

[问答题]

原生 JavaScript 动态加载脚本文件

来自：原生 Javascript 编程练习题

参考：

```
function appendScript(src, text, reload, charset) {
    var id = hash(src + text);
    if(!reload && in_array(id, evalscripts)) return;
    if(reload && $(id)) {
        $(id).parentNode.removeChild($(id));
    }

    evalscripts.push(id);
    var scriptNode = document.createElement("script");
    scriptNode.type = "text/javascript";
    scriptNode.id = id;
    scriptNode.charset = charset ? charset : (BROWSER.firefox ? document.characterSet : document.charset);
    try {
        if(src) {
            scriptNode.src = src;
            scriptNode.onloadDone = false;
            scriptNode.onload = function () {
                scriptNode.onloadDone = true;
                JSLOADED[src] = 1;
            };
            scriptNode.onreadystatechange = function () {
                if((scriptNode.readyState == 'loaded' || scriptNode.readyState == 'complete') && !scriptNode.onloadDone) {
                    scriptNode.onloadDone = true;
                    JSLOADED[src] = 1;
                }
            };
        } else if(text){
            scriptNode.text = text;
        }
        document.getElementsByTagName('head')[0].appendChild(scriptNode);
    } catch(e) {}
}
```

361. 361

[问答题]

原生 JavaScript 返回按 ID 检索的元素对象

来自：原生 Javascript 编程练习题

参考：

```
function $(id) {
    return !id ? null : document.getElementById(id);
}
```

362.

[问答题]

原生 JavaScript 返回浏览器版本内容

来自：原生 Javascript 编程练习题

参考：

```
function browserVersion(types) {
    var other = 1;
    for(i in types) {
        var v = types<i> ? types<i> : i;
        if(USERAGENT.indexOf(v) != -1) {
            var re = new RegExp(v + '(\V|\s)([\d\.\.]+)', 'ig');
            var matches = re.exec(USERAGENT);
            var ver = matches != null ? matches[2] : 0;
            other = ver !== 0 && v != 'mozilla' ? 0 : other;
        }else {
            var ver = 0;
        }
        eval('BROWSER.' + i + '= ver');
    }
    BROWSER.other = other;
}
```

363.

[问答题]

原生 JavaScript 元素显示的通用方法

来自：原生 Javascript 编程练习题

参考：

```
function $(id) {
    return !id ? null : document.getElementById(id);
}

function display(id) {
    var obj = $(id);
    if(obj.style.visibility) {
        obj.style.visibility = obj.style.visibility == 'visible' ? 'hidden' : 'visible';
    } else {
        obj.style.display = obj.style.display == "" ? 'none' : "";
    }
}
```

364.

[问答题]

原生 JavaScript 中有 insertBefore 方法,可惜却没有 insertAfter 方法?用如下函数实现

来自：原生 Javascript 编程练习题

参考：

```
function insertAfter(newChild,refChild){
    var parElem=refChild.parentNode;
    if(parElem.lastChild==refChild){
        refChild.appendChild(newChild);
    }else{
        parElem.insertBefore(newChild,refChild.nextSibling);
    }
}
```

365.

[问答题]

原生 JavaScript 中兼容浏览器绑定元素事件

来自：原生 Javascript 编程练习题

参考：

```
function addEventSamp(obj,evt,fn){
    if (obj.addEventListener) {
        obj.addEventListener(evt, fn, false);
    }
```

```
 }else if(obj.attachEvent){  
    obj.attachEvent('on'+evt,fn);  
}  
}  
}
```

366.

[问答题]

原生 JavaScript 光标停在文字的后面，文本框获得焦点时调用

来自：原生 Javascript 编程练习题

参考：

```
function focusLast(){  
    var e = event.srcElement;  
    var r = e.createTextRange();  
    r.moveStart('character',e.value.length);  
    r.collapse(true);  
    r.select();  
}
```

367.

[问答题]

原生 JavaScript 检验 URL 链接是否有效

来自：原生 Javascript 编程练习题

参考：

```
function getUrlState(URL){  
    var xmlhttp = new ActiveXObject("microsoft.xmlhttp");  
    xmlhttp.Open("GET",URL, false);  
    try{  
        xmlhttp.Send();  
    }catch(e){  
    }finally{  
        var result = xmlhttp.responseText;  
        if(result){  
            if(xmlhttp.Status==200){  
                return(true);  
            }else{  
                return(false);  
            }  
        }  
    }  
}
```

```
        }else{
            return(false);
        }
    }
}
```

368.

[问答题]

原生 JavaScript 格式化 CSS 样式代码

来自：原生 Javascript 编程练习题

参考：

```
function formatCss(s){//格式化代码
    s = s.replace(/\s*(\{\}\:\;\])\s*/g, "$1");
    s = s.replace(/;\s*/g, ";"); //清除连续分号
    s = s.replace(/\,\[\s\.\#\d]*{/g, "{");
    s = s.replace(/(^s)]\{([^\s])/g, "$1 {\n\t$2");
    s = s.replace(/(^s)]\{([^\n]*)/g, "$1\n\n$2");
    s = s.replace(/(^s);([^\s])/g, "$1;\n\t$2");
    return s;
}
```

369.

[问答题]

原生 JavaScript 压缩 CSS 样式代码

来自：原生 Javascript 编程练习题

参考：

```
function yasuoCss (s) {//压缩代码
    s = s.replace(/\/*(.|\n)*?\*\//g, ""); //删除注释
    s = s.replace(/\s*(\{\}\:\;\])\s*/g, "$1");
    s = s.replace(/\,\[\s\.\#\d]*\{/g, "{"); //容错处理
    s = s.replace(/;\s*/g, ";"); //清除连续分号
    s = s.match(/^\s*(\S+(\s+\S+)*\s*)\s*$/); //去掉首尾空白
    return (s == null) ? "" : s[1];
}
```

370.

[问答题]

原生 JavaScript 获取当前路径

来自：原生 Javascript 编程练习题

参考：

```
var currentPageUrl = "";
if (typeof this.href === "undefined") {
    currentPageUrl = document.location.toString().toLowerCase();
}
else {
    currentPageUrl = this.href.toString().toLowerCase();
}
```

371.

[问答题]

原生 JavaScript IP 转成整型

来自：原生 Javascript 编程练习题

参考：

```
function _ip2int(ip){
    var num = 0;
    ip = ip.split(".");
    num = Number(ip[0]) * 256 * 256 * 256 + Number(ip[1]) * 256 * 256 + Number(ip[2]) * 25
6 + Number(ip[3]);
    num = num >>> 0;
    return num;
}
```

372.

[问答题]

原生 JavaScript 整型解析为 IP 地址

来自：原生 Javascript 编程练习题

参考：

```
function _int2ip(num){
```

```
var str;
var tt = new Array();
tt[0] = (num >>> 24) >>> 0;
tt[1] = ((num << 8) >>> 24) >>> 0;
tt[2] = (num << 16) >>> 24;
tt[3] = (num << 24) >>> 24;
str = String(tt[0]) + "." + String(tt[1]) + "." + String(tt[2]) + "." + String(tt[3]);
return str;
}
```

373.

[问答题]

原生 JavaScript 实现 checkbox 全选与全不选

来自：原生 Javascript 编程练习题

参考：

```
function checkAll() {
    var selectall = document.getElementById("selectall");
    var allbox = document.getElementsByName("allbox");
    if (selectall.checked) {
        for (var i = 0; i < allbox.length; i++) {
            allbox[i].checked = true;
        }
    } else {
        for (var i = 0; i < allbox.length; i++) {
            allbox[i].checked = false;
        }
    }
}
```

374.

[问答题]

原生 JavaScript 判断是否移动设备

来自：原生 Javascript 编程练习题

参考：

```
function isMobile(){
    if (typeof this._isMobile === 'boolean'){
        return this._isMobile;
    }
}
```

```
        }
        var screenWidth = this.getScreenWidth();
        var fixViewPortsExperiment = rendererModel.runningExperiments.FixViewport || rendererModel.runningExperiments.fixviewport;
        var fixViewPortsExperimentRunning = fixViewPortsExperiment && (fixViewPortsExperiment.toLowerCase() === "new");
        if(!fixViewPortsExperiment){
            if(!this.isAppleMobileDevice()){
                screenWidth = screenWidth/window.devicePixelRatio;
            }
        }
        var isMobileScreenSize = screenWidth < 600;
        var isMobileUserAgent = false;
        this._isMobile = isMobileScreenSize && this.isTouchScreen();
        return this._isMobile;
    }
}
```

375.

[问答题]

原生 JavaScript 判断是否移动设备访问

来自：原生 Javascript 编程练习题

参考：

```
function isAppleMobileDevice(){
    return (/iphone|ipod|ipad|Macintosh/i.test(navigator.userAgent.toLowerCase()));
}
```

376.

[问答题]

原生 JavaScript 判断是否苹果移动设备访问

来自：原生 Javascript 编程练习题

参考：

```
function isAppleMobileDevice(){
    return (/iphone|ipod|ipad|Macintosh/i.test(navigator.userAgent.toLowerCase()));
}
```

377.

[问答题]

原生 JavaScript 判断是否安卓移动设备访问

来自：原生 Javascript 编程练习题

参考：

```
function isAndroidMobileDevice(){
    return (/android/i.test(navigator.userAgent.toLowerCase()));
}
```

378.

[问答题]

原生 JavaScript 判断是否 Touch 屏幕

来自：原生 Javascript 编程练习题

参考：

```
function isTouchScreen(){
    return ('ontouchstart' in window) || window.DocumentTouch && document instanceof
DocumentTouch;
}
```

379.

[问答题]

原生 JavaScript 判断是否在安卓上的谷歌浏览器

来自：原生 Javascript 编程练习题

参考：

```
function isNewChromeOnAndroid(){
    if(this.isAndroidMobileDevice()){
        var userAgent = navigator.userAgent.toLowerCase();
        if(/chrome/i.test(userAgent)){
            var parts = userAgent.split('chrome/');
            var fullVersionString = parts[1].split(" ")[0];
            var versionString = fullVersionString.split('.')[0];
            var version = parseInt(versionString);
            if(version >= 27){

```

```
        return true;
    }
}
return false;
}
```

380.

[问答题]

原生 JavaScript 判断是否打开视窗

来自：原生 Javascript 编程练习题

参考：

```
function isViewportOpen() {
    return !!document.getElementById('wixMobileViewport');
}
```

381.

[问答题]

原生 JavaScript 获取移动设备初始化大小

来自：原生 Javascript 编程练习题

参考：

```
function getInitZoom(){
    if(!this._initZoom){
        var screenWidth = Math.min(screen.height, screen.width);
        if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){
            screenWidth = screenWidth/window.devicePixelRatio;
        }
        this._initZoom = screenWidth /document.body.offsetWidth;
    }
    return this._initZoom;
}
```

382.

[问答题]

原生 JavaScript 获取移动设备最大化大小

来自：原生 Javascript 编程练习题

参考：

```
function getZoom(){  
    var screenWidth = (Math.abs(window.orientation) === 90) ? Math.max(screen.height, screen.width) : Math.min(screen.height, screen.width);  
    if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){  
        screenWidth = screenWidth/window.devicePixelRatio;  
    }  
    var FixViewPortsExperiment = rendererModel.runningExperiments.FixViewport || rendererModel.runningExperiments.fixviewport;  
    var FixViewPortsExperimentRunning = FixViewPortsExperiment && (FixViewPortsExperiment === "New" || FixViewPortsExperiment === "new");  
    if(FixViewPortsExperimentRunning){  
        return screenWidth / window.innerWidth;  
    }else{  
        return screenWidth / document.body.offsetWidth;  
    }  
}
```

383.

[问答题]

原生 JavaScript 获取移动设备屏幕宽度

来自：原生 Javascript 编程练习题

参考：

```
function getScreenWidth(){  
    var smallerSide = Math.min(screen.width, screen.height);  
    var fixViewPortsExperiment = rendererModel.runningExperiments.FixViewport || rendererModel.runningExperiments.fixviewport;  
    var fixViewPortsExperimentRunning = fixViewPortsExperiment && (fixViewPortsExperiment.toLowerCase() === "new");  
    if(fixViewPortsExperiment){  
        if(this.isAndroidMobileDevice() && !this.isNewChromeOnAndroid()){  
            smallerSide = smallerSide/window.devicePixelRatio;  
        }  
    }  
    return smallerSide;  
}
```

384.

[问答题]

原生 JavaScript 完美判断是否为网址

来自：原生 Javascript 编程练习题

参考：

```
function IsURL(strUrl) {  
    var regular = /^\b(((https?|ftp):\//)?[-a-z0-9]+(\.[-a-z0-9]+)*\.(?:com|edu|gov|int|mil|net|org|biz|info|name|museum|asia|coop|aero|[a-z][a-z]|((25[0-5])|(2[0-4]\d)|(1\d\d)|([1-9]\d)|\d))\b(\/[-a-z0-9_:@&?=+,.!/~%\$\"]*)?)$/i  
    if (regular.test(strUrl)) {  
        return true;  
    }  
    else {  
        return false;  
    }  
}
```

385.

[问答题]

原生 JavaScript 根据样式名称检索元素对象

来自：原生 Javascript 编程练习题

参考：

```
function getElementsByClassName(name) {  
    var tags = document.getElementsByTagName('*') || document.all;  
    var els = [];  
    for (var i = 0; i < tags.length; i++) {  
        if (tags.className) {  
            var cs = tags.className.split(' ');\n            for (var j = 0; j < cs.length; j++) {  
                if (name == cs[j]) {  
                    els.push(tags);  
                    break  
                }  
            }  
        }  
    }  
    return els
```

```
}
```

386.

[问答题]

原生 JavaScript 判断是否以某个字符串开头

来自：原生 Javascript 编程练习题

参考：

```
String.prototype.startsWith = function (s) {  
    return this.indexOf(s) == 0  
}
```

387.

[问答题]

原生 JavaScript 判断是否以某个字符串结束

来自：原生 Javascript 编程练习题

参考：

```
String.prototype.endsWith = function (s) {  
    var d = this.length - s.length;  
    return (d >= 0 && this.lastIndexOf(s) == d)  
}
```

388.

[问答题]

原生 JavaScript 返回 IE 浏览器的版本号

来自：原生 Javascript 编程练习题

参考：

```
function getIE(){  
    if (window.ActiveXObject){  
        var v = navigator.userAgent.match(/MSIE ([^;]+)[/][1];  
        return parseFloat(v.substring(0, v.indexOf(".")))  
    }  
    return false  
}
```

389.

[问答题]

原生 JavaScript 获取页面高度

来自：原生 Javascript 编程练习题

参考：

```
function getPageHeight(){  
    var g = document, a = g.body, f = g.documentElement, d = g.compatMode == "BackCompat"  
    ? a  
    : g.documentElement;  
    return Math.max(f.scrollHeight, a.scrollHeight, d.clientHeight);  
}
```

390.

[问答题]

原生 JavaScript 获取页面 scrollLeft

来自：原生 Javascript 编程练习题

参考：

```
function getPageScrollLeft(){  
    var a = document;  
    return a.documentElement.scrollLeft || a.body.scrollLeft;  
}
```

391.

[问答题]

原生 JavaScript 获取页面可视宽度

来自：原生 Javascript 编程练习题

参考：

```
function getPageViewWidth(){  
    var d = document, a = d.compatMode == "BackCompat"  
    ? d.body  
    : d.documentElement;  
    return a.clientWidth;
```

```
}
```

392.

[问答题]

原生 JavaScript 获取页面宽度

来自：原生 Javascript 编程练习题

参考：

```
function getPageWidth(){
    var g = document, a = g.body, f = g.documentElement, d = g.compatMode == "BackCompat"
    ? a
    : g.documentElement;
    return Math.max(f.scrollWidth, a.scrollWidth, d.clientWidth);
}
```

393.

[问答题]

原生 JavaScript 获取页面 scrollTop

来自：原生 Javascript 编程练习题

参考：

```
function getPageScrollTop(){
    var a = document;
    return a.documentElement.scrollTop || a.body.scrollTop;
}
```

394.

[问答题]

考察知识点：作用域

考虑如下代码：

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

请问控制台上会输出什么？

来自：Javascript 开发者面试典型题集

输出：5

这一题的陷阱是，在函数表达式中有两个赋值，但 `a` 是用关键字 `var` 来声明的，这意味着 `a` 是局部变量，而 `b` 则被赋予为全局变量。

另一个陷阱是，它并没有使用严格模式（`use strict`）。在函数里面，如果启用了严格模式，代码就会报错：“`Uncaught ReferenceError: b is not defined`”。请记住，严格模式需要你显式地引用全局作用域，代码应该写成：

```
(function() {  
  'use strict';  
  var a = window.b = 5;  
}());  
console.log(b);
```

395.

[问答题]

考察知识点：创建“内置”方法

给 `String` 对象定义一个 `repeatify` 方法。该方法接收一个整数参数，作为字符串重复的次数，最后返回重复指定次数的字符串。例如：

```
console.log('hello'.repeatify(3));
```

输出应该是

`hellohellohello.`

来自：Javascript 开发者面试典型题集

参考：

一个可行的做法如下：

```
String.prototype.repeatify = String.prototype.repeatify || function(times) {  
  var str = "";  
  for (var i = 0; i < times; i++) {  
    str += this;  
  }  
  return str;  
};
```

这题测试开发者对 Javascript 的继承及原型属性的知识，它同时也检验了开发者是否能扩展内置数据类型的方法。

这里的另一个关键点是，看你怎样避免重写可能已经定义了的方法。这可以通过在定义自己的方法之前，检测方法是否已经存在。

```
String.prototype.repeatify = String.prototype.repeatify || function(times) {/* code here */};
```

当被问起去扩展一个 Javascript 方法时，这个技术非常有用。

396.

[问答题]

考察知识点：声明提前

下面这段代码的结果是什么？为什么？

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

来自：Javascript 开发者面试典型题集

参考：

代码的运行结果：`undefined` 和 `2`

理由是，变量和函数的声明都被提前至函数体的顶部，而同时变量并没有被赋值。因此，当打印变量 `a` 时，它虽存在于函数体（因为 `a` 已经被声明），但仍然是 `undefined`。换句话说，上面的代码等同于下面的代码：

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

397.

[问答题]

考察知识点：JavaScript 中的 `this`

下面代码的运行结果是什么并做解释。

```
var fullname = 'John Doe';  
var obj = {  
    fullname: 'Colin Ihrig',  
    prop: {  
        fullname: 'Aurelio De Rosa',  
        getfullname: function() {
```

```
return this.fullname;  
});  
console.log(obj.prop.getfullname());  
var test = obj.prop.getfullname;  
console.log(test());
```

来自： Javascript 开发者面试典型题集

参考：

代码输出： Aurelio De Rosa 和 John Doe

理由是， Javascript 中关键字 this 所指代的函数上下文，取决于函数是怎样被调用的，而不是怎样被定义的。

在第一个 console.log()，getfullname()被作为 obj.prop 对象被调用。因此，当前的上下文指代后者，函数返回这个对象的 fullname 属性。相反，当 getfullname()被赋予 test 变量，当前的上下文指代全局对象 window，这是因为 test 被隐式地作为全局对象的属性。基于这一点，函数返回 window 的 fullname，在本例中即为代码的第一行。

398.

[问答题]

考察知识点： call()和 apply()

修复前一个问题，让最后一个 console.log() 打印输出 Aurelio De Rosa.

来自： Javascript 开发者面试典型题集

参考：

这个问题可以通过运用 call () 或者 apply()方法强制转换上下文环境。下面的代码中用了 call(),但 apply () 也能产生同样的结果：

```
console.log(test.call(obj.prop));
```

399.

[问答题]

真假判断

```
var aLinks=document.getElementsByTagName('a');  
for(i=0;i<aLinks.length;i++)  
{  
  
}
```

来自： 百度 2011 年 6 月前端开发面试题

参考：

复制代码修改后：

```
var aLinks=document.getElementsByTagName('a');
for(var i=0,l= aLinks.length;i<l;i++)
{
}

}
```

400.

[问答题]

参照上题代码，给 a 添加事件，要求点击弹出提示相应的 index 值。

来自：百度 2011 年 6 月前端开发面试题

参考：

(1)、第一种方法（加索引）

```
var aLinks=document.getElementsByTagName('a');
for(var i=0,l= aLinks.length;i<l;i++)
{
    aLinks[i].Index=i;
    aLinks[i].onclick=function(){alert(this.Index)};
}
```

(2)、第二种方法（闭包）

```
var aLinks=document.getElementsByTagName('a');
for(var i=0,l= aLinks.length;i<l;i++)
{
    aLinks[i].onclick=function(a){
        return function(){alert(a);}
    }(i);
}
```

401.

[问答题]

用户上传图片，没有刷新过程显示图片的功能【ajax】

来自：百度 2011 年 6 月前端开发面试题附加题

参考：无

402.

[问答题]

写出下面 JS 的运行结果:

```
var a = 10,b = 20,c = 10;  
alert(a=b);alert(a==b);alert(a==c);
```

来自：搜道网前端开发面试题

参考：

20 true false

403.

[问答题]

a: javascript 如何深度克隆一个对象？

b: javascript 如何消除一个数组里面重复的元素？

来自：搜道网前端开发面试题

参考：

a:

```
function Object.prototype.cloneObj()  
{  
    function NEWOBJECT(){};  
    NEWOBJECT.prototype = this;  
    var anObj = new NEWOBJECT();  
    for ( var ele in anObj )  
    {  
        if ( typeof anObj[ele] == "object" ) return anObj[ele].cloneObj();  
    }  
    return anObj;  
}
```

b:

```
function getNewArr(oldArr){  
    if(typeof oldArr != "object") return oldArr;  
    var newArr = [];  
    var oldArrLen = oldArr.length-1, newArrLen = -1, flag = false;  
    for(var i=oldArrLen; i>=0; i--){  
        flag = false;  
        for(var j=newArrLen; j>=0; j--){  
            if(oldArr === newArr[j]){  
                flag = true;  
            }  
        }  
        if(flag) continue;  
        newArr[newArrLen] = oldArr[i];  
        newArrLen--;  
    }  
    return newArr;  
}
```

```
        break;
    }
}
if(!flag) newArrLen = newArr.push(oldArr)-1;
}
return newArr;
}
```

404.

[问答题]

你玩微博吗？现有这样的字符串：@小甜甜 (xiaotiantian) 或者 @小王的后宫 (hougong123) ..这样的字符串如何在计算字数的时候，省略括号里面的内容？比如 @小甜甜 (xiaotiantian) 他的长度应该是省略 (xiaotiantian) 这个字符串后的长度，就是 3. 你有多少种方法来实现，请写下代码。

来自：搜道网前端开发面试题

参考：

a: 使用正则替换掉 (XXX)，然后使用 string.length 返回长度，代码：

```
function getStrLen(str){return (str.replace(/[\(\w\)]/g, "")).length;}
```

b: 使用 indexOf 查找"("与")"，记录相关位置，然后使用 str.substring()函数截取，得到新串，再.length 得到长度，代码：

```
function getStrLen(str){
    return (str.substring(0, str.indexOf("(")+str.substring(str.indexOf(")")+1)).length;
}
```

405.

[问答题]

用 Html5+js 重现"新浪微博手机体验版"首页。

来自：UC(优视)前端面试题

参考：无

406.

[问答题]

设计一个鼠标移入移出代理组件，在 UL 或 OL 上绑定事件，使得鼠标经过里面的 li 时可触发移入移出函数。

来自：乐视前端面试题
参考：无

407.

[问答题]

有多个模块会用到同一个接口的数据（需要异步请求，并且只能请求一次），但各模块在什么时候执行不确定，请设计一个组件解决这个问题。

来自：乐视前端面试题
参考：无

408.

[问答题]

请把以下用于连接字符串的 JavaScript 代码修改为更高效的方式

```
var htmlString = '<div class="container">' +  
    '<ul id="news-list">;  
for (var i = 0; i < NEWS.length; i++) {  
    htmlString += '<li><a href="" +  
        NEWS[i].LINK + "> +  
        NEWS[i].TITLE + '</a></li>;  
}  
htmlString += '</ul></div>;
```

来自：前端开发面试题
参考：无

409.

[问答题]

尝试实现注释部分的 Javascript 代码，可在其他任何地方添加更多代码（如不能实现，说明一下不能实现的原因）：

```
var Obj = function(msg){  
    this.msg = msg;  
    this.shout = function(){  
        alert(this.msg);  
    }  
    this.waitAndShout = function(){
```

```
//隔五秒钟后执行上面的 shout 方法  
}  
}
```

来自：前端开发面试题

参考：无

410.

[问答题]

请编写一个 JavaScript 函数 toRGB，它的作用是转换 CSS 中常用的颜色编码。要求：

```
alert(toRGB("#0000FF"));           // 输出 rgb(0, 0, 255)  
alert(toRGB("invalid"));           // 输出 invalid  
alert(toRGB("#G00"));             // 输出 #G00
```

来自：前端开发面试题

参考：无

411.

[问答题]

javascript 有哪几种方法定义函数

来自：前端开发面试题

参考：

1. [函 数 声 明 表 达 式](<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function>)
2. [function 操 作 符](<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function>)
3. [Function 构 造 函 数](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function)
4. [ES6:arrow function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/arrow_functions)

412.

[问答题]

应用程序存储和离线 web 应用

来自：前端开发面试题

参考：

HTML5 新增应用程序缓存，允许 web 应用将应用程序自身保存到用户浏览器中，用户离线状态也能访问。

1.为 html 元素设置 manifest 属性:<html manifest="myapp.appcache">，其中后缀名只是一个约定，真正识别方式是通过 text/cache-manifest 作为 MIME 类型。所以需要配置服务器保证设置正确

2.manifest 文件首行为 CACHE MANIFEST，其余就是要缓存的 URL 列表，每个一行，相对路径都相对于 manifest 文件的 url。注释以#开头

3.url 分为三种类型：CACHE:为默认类型。NETWORK: 表示资源从不缓存。FALLBACK: 每行包含两个 url，第二个 URL 是指需要加载和存储在缓存中的资源，第一个 URL 是一个前缀。任何匹配该前缀的 URL 都不会缓存，如果从网络中载入这样的 URL 失败的话，就会用第二个 URL 指定的缓存资源来替代。以下是一个文件例子：

```
CACHE MANIFEST
CACHE:
myapp.html
myapp.css
myapp.js
FALLBACK:
videos/ offline_help.html
NETWORK:
cgi/
```

客户端存储 localStorage 和 sessionStorage

localStorage 有效期为永久，sessionStorage 有效期为顶层窗口关闭前

同源文档可以读取并修改 localStorage 数据，sessionStorage 只允许同一个窗口下的文档访问，如通过 iframe 引入的同源文档。

Storage 对象通常被当做普通 javascript 对象使用：通过设置属性来存取字符串值，也可以通过 setItem(key, value) 设置， getItem(key) 读取，removeItem(key) 删除，clear() 删除所有数据，length 表示已存储的数据项数目，key(index) 返回对应索引的 key

```
localStorage.setItem('x', 1); // storage x->1
localStorage.getItem('x'); // return value of x
// 枚举所有存储的键值对
for (var i = 0, len = localStorage.length; i < len; ++i ) {
    var name = localStorage.key(i);
    var value = localStorage.getItem(name);
}
localStorage.removeItem('x'); // remove x
localStorage.clear(); // remove all data
```

413.

[问答题]

cookie 及其操作

来自：前端开发面试题

参考：

cookie 是 web 浏览器存储的少量数据，最早设计为服务器端使用，作为 HTTP 协议的扩展实现。cookie 数据会自动在浏览器和服务器之间传输。

通过读写 cookie 检测是否支持

cookie 属性有名，值，有效期，作用域，secure；

cookie 默认有效期为浏览器会话，一旦用户关闭浏览器，数据就丢失，通过设置 max-age=seconds 属性告诉浏览器 cookie 有效期

cookie 作用域通过文档源和文档路径来确定，通过 path 和 domain 进行配置，web 页面同目录或子目录文档都可访问

通过 cookie 保存数据的方法为：为 document.cookie 设置一个符合目标的字符串如下

读取 document.cookie 获得';'分隔的字符串，key=value,解析得到结果

```
document.cookie = 'name=qiu; max-age=9999; path=/; domain=domain; secure';
```

```
document.cookie = 'name=aaa; path=/; domain=domain; secure';
```

// 要改变 cookie 的值，需要使用相同的名字、路径和域，新的值

// 来设置 cookie，同样的方法可以用来改变有效期

// 设置 max-age 为 0 可以删除指定 cookie

// 读取 cookie，访问 document.cookie 返回键值对组成的字符串，

// 不同键值对之间用';'分隔。通过解析获得需要的值

cookieUtil.js：自己写的 cookie 操作工具

414.

[问答题]

javascript 有哪些方法定义对象

来自：前端开发面试题

参考：

对象字面量： var obj = {};

构造函数： var obj = new Object();

Object.create(): var obj = Object.create(Object.prototype);

415.

[问答题]

==运算符判断相等的流程是怎样的

来自：前端开发面试题

参考：

如果两个值不是相同类型，它们不相等

如果两个值都是 `null` 或者都是 `undefined`，它们相等

如果两个值都是布尔类型 `true` 或者都是 `false`，它们相等

如果其中有一个是 `Nan`，它们不相等

如果都是数值型并且数值相等，他们相等，`-0` 等于 `0`

如果他们都是字符串并且在相同位置包含相同的 16 位值，他它们相等；如果在长度或者内容上不等，它们不相等；两个字符串显示结果相同但是编码不同`==`和`===`都认为他们不相等

如果他们指向相同对象、数组、函数，它们相等；如果指向不同对象，他们不相等

416.

[问答题]

`==`运算符判断相等的流程是怎样的

来自：前端开发面试题

参考：

如果两个值类型相同，按照`==`比较方法进行比较

如果类型不同，使用如下规则进行比较

如果其中一个值是 `null`，另一个是 `undefined`，它们相等

如果一个值是数字另一个是字符串，将字符串转换为数字进行比较

如果有布尔类型，将 `true` 转换为 `1`, `false` 转换为 `0`，然后用`==`规则继续比较

如果一个值是对象，另一个是数字或字符串，将对象转换为原始值然后用`==`规则继续比较

其他所有情况都认为不相等

417.

[问答题]

对象到字符串的转换步骤

来自：前端开发面试题

参考：

如果对象有 `toString()`方法，`javascript` 调用它。如果返回一个原始值 (primitive value 如：

`string number boolean`) ,将这个值转换为字符串作为结果

如果对象没有 `toString()`方法或者返回值不是原始值，`javascript` 寻找对象的 `valueOf()`方法，如果存在就调用它，返回结果是原始值则转为字符串作为结果

否则，`javascript` 不能从 `toString()`或者 `valueOf()`获得一个原始值，此时 `throws a TypeError`
对象到数字的转换步骤

1. 如果对象有 `valueOf()`方法并且返回元素值，`javascript` 将返回值转换为数字作为结果

2. 否则，如果对象有 `toString()` 并且返回原始值，javascript 将返回结果转换为数字作为结果

3. 否则，`throws a TypeError`

`<,>,<=,>=` 的比较规则

所有比较运算符都支持任意类型，但是比较只支持数字和字符串，所以需要执行必要的转换然后进行比较，转换规则如下：1. 如果操作数是对象，转换为原始值：如果 `valueOf` 方法返回原始值，则使用这个值，否则使用 `toString` 方法的结果，如果转换失败则报错 2. 经过必要的对象到原始值的转换后，如果两个操作数都是字符串，按照字母顺序进行比较（他们的 16 位 `unicode` 值的大小）3. 否则，如果有一个操作数不是字符串，将两个操作数转换为数字进行比较

418.

[问答题]

+运算符工作流程

来自：前端开发面试题

参考：

1. 如果有操作数是对象，转换为原始值
2. 此时如果有**一个操作数是字符串**，其他的操作数都转换为字符串并执行连接
3. 否则：**所有操作数都转换为数字并执行加法**

419.

[问答题]

函数内部 `arguments` 变量有哪些特性，有哪些属性，如何将它转换为数组

来自：前端开发面试题

参考：

`arguments` 所有函数中都包含的一个局部变量，是一个类数组对象，对应函数调用时的实参。如果函数定义同名参数会在调用时覆盖默认对象

`arguments[index]` 分别对应函数调用时的实参，并且通过 `arguments` 修改实参时会同时修改实参

`arguments.length` 为实参的个数（`Function.length` 表示形参长度）

`arguments.callee` 为当前正在执行的函数本身，使用这个属性进行递归调用时需注意 `this` 的变化

`arguments.caller` 为调用当前函数的函数（已被遗弃）

转换为数组：`var args = Array.prototype.slice.call(arguments, 0);`

420.

[问答题]

DOM 事件模型是如何的，编写一个 EventUtil 工具类实现事件管理兼容

来自：前端开发面试题

参考：

DOM 事件包含捕获（capture）和冒泡（bubble）两个阶段：捕获阶段事件从 window 开始触发事件然后通过祖先节点一次传递到触发事件的 DOM 元素上；冒泡阶段事件从初始元素依次向祖先节点传递直到 window

标准事件监听 elem.addEventListener(type, handler, capture)/elem.removeEventListener(type, handler, capture): handler 接收保存事件信息的 event 对象作为参数，event.target 为触发事件的对象，handler 调用上下文 this 为绑定监听器的对象，event.preventDefault() 取消事件默认行为，event.stopPropagation()/event.stopImmediatePropagation() 取消事件传递

老版本 IE 事件监听 elem.attachEvent('on'+type, handler)/elem.detachEvent('on'+type, handler): handler 不接收 event 作为参数，事件信息保存在 window.event 中，触发事件的对象为 event.srcElement，handler 执行上下文 this 为 window 使用闭包中调用 handler.call(elem, event) 可模仿标准模型，然后返回闭包，保证了监听器的移除。event.returnValue 为 false 时取消事件默认行为，event.cancelBubble 为 true 时取消时间传播

通常利用事件冒泡机制托管事件处理程序提高程序性能。

```
/**  
 * 跨浏览器事件处理工具。只支持冒泡。不支持捕获  
 * @author (qiu_deqing@126.com)  
 */
```

```
var EventUtil = {  
    getEvent: function (event) {  
        return event || window.event;  
    },  
    getTarget: function (event) {  
        return event.target || event.srcElement;  
    },  
    // 返回注册成功的监听器，IE 中需要使用返回值来移除监听器  
    on: function (elem, type, handler) {  
        if (elem.addEventListener) {  
            elem.addEventListener(type, handler, false);  
            return handler;  
        } else if (elem.attachEvent) {  
            var wrapper = function () {  
                var event = window.event;  
                event.target = event.srcElement;
```

```

        handler.call(elem, event);
    };
    elem.attachEvent('on' + type, wrapper);
    return wrapper;
}
},
off: function (elem, type, handler) {
    if (elem.removeEventListener) {
        elem.removeEventListener(type, handler, false);
    } else if (elem.detachEvent) {
        elem.detachEvent('on' + type, handler);
    }
},
preventDefault: function (event) {
    if (event.preventDefault) {
        event.preventDefault();
    } else if ('returnValue' in event) {
        event.returnValue = false;
    }
},
stopPropagation: function (event) {
    if (event.stopPropagation) {
        event.stopPropagation();
    } else if ('cancelBubble' in event) {
        event.cancelBubble = true;
    }
}
};

```

421.

[问答题]

评价一下三种方法实现继承的优缺点，并改进

```

function Shape() {}
function Rect() {}
// 方法 1
Rect.prototype = new Shape();
// 方法 2
Rect.prototype = Shape.prototype;
// 方法 3
Rect.prototype = Object.create(Shape.prototype);
Rect.prototype.area = function () {
    // do something
};

```

来自：前端开发面试题

参考：

方法 1：

优点：正确设置原型链实现继承

优点：父类实例属性得到继承，原型链查找效率提高，也能为一些属性提供合理的默认值

缺点：父类实例属性为引用类型时，不恰当地修改会导致所有子类被修改

缺点：创建父类实例作为子类原型时，可能无法确定构造函数需要的合理参数，这样提供的参数继承给子类没有实际意义，当子类需要这些参数时应该在构造函数中进行初始化和设置

总结：继承应该是继承方法而不是属性，为子类设置父类实例属性应该是通过在子类构造函数中调用父类构造函数进行初始化

方法 2：

优点：正确设置原型链实现继承

缺点：父类构造函数原型与子类相同。修改子类原型添加方法会修改父类

方法 3：

优点：正确设置原型链且避免方法 1.2 中的缺点

缺点：ES5 方法需要注意兼容性

改进：

所有三种方法应该在子类构造函数中调用父类构造函数实现实例属性初始化

```
function Rect() {
    Shape.call(this);
}

用新创建的对象替代子类默认原型，设置 Rect.prototype.constructor = Rect;保证一致性
第三种方法的 polyfill:
function create(obj) {
    if (Object.create) {
        return Object.create(obj);
    }
    function f() {};
    f.prototype = obj;
    return new f();
}
```

422.

[问答题]

完成一个函数，接受数组作为参数，数组元素为整数或者数组，数组元素包含整数或数组，函数返回扁平化后的数组

如： [1, [2, [[3, 4], 5], 6]] => [1, 2, 3, 4, 5, 6]

来自：前端开发面试题

参考：

```
var data = [1, [2, [ [3, 4], 5], 6]];
function flat(data, result) {
    var i, d, len;
    for (i = 0, len = data.length; i < len; ++i) {
        d = data[i];
        if (typeof d === 'number') {
            result.push(d);
        } else {
            flat(d, result);
        }
    }
}
var result = [];
flat(data, result);
console.log(result);
```

423.

[问答题]

如何判断一个对象是否为数组

来自：前端开发面试题

参考：

如果浏览器支持 `Array.isArray()` 可以直接判断否则需进行必要判断

```
/**
 * 判断一个对象是否是数组，参数不是对象或者不是数组，返回 false
 *
 * @param {Object} arg 需要测试是否为数组的对象
 * @return {Boolean} 传入参数是数组返回 true，否则返回 false
 */
function isArray(arg) {
    if (typeof arg === 'object') {
        return Object.prototype.toString.call(arg) === '[object Array]';
    }
    return false;
}
```

424.

[问答题]

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);
        });
    };
}
```

来自：前端开发面试题

参考：

作用：浏览器功能检测实现跨浏览器 DOM 事件绑定

优点：

测试代码只运行一次，根据浏览器确定绑定方法

通过 `listener.apply(el)` 解决 IE 下监听器 `this` 与标准不一致的地方

在浏览器不支持的情况下提供简单的功能，在标准浏览器中提供捕获功能

缺点：

`document.all` 作为 IE 检测不可靠，应该使用 `if(el.attachEvent)`

`addListener` 在不同浏览器下 API 不一样

`listener.apply` 使 `this` 与标准一致但监听器无法移除

未解决 IE 下 `listener` 参数 `event`。`target` 问题

改进：

```
var addListener;
if (window.addEventListener) {
    addListener = function (el, type, listener, useCapture) {
        el.addEventListener(type, listener, useCapture);
        return listener;
    };
}
else if (window.attachEvent) {
    addListener = function (el, type, listener) {
        // 标准化 this, event, target
        var wrapper = function () {
            var event = window.event;
            event.target = event.srcElement;
            listener.call(el, event);
        };
    };
}
```

```
        el.attachEvent('on' + type, wrapper);
        return wrapper;
        // 返回 wrapper。调用者可以保存，以后 remove
    };
}
```

425.

[问答题]

如何判断一个对象是否为函数

来自：前端开发面试题

参考：

```
/**
 * 判断对象是否为函数，如果当前运行环境对可调用对象（如正则表达式）
 * 的 typeof 返回'function'，采用通用方法，否则采用优化方法
 *
 * @param {Any} arg 需要检测是否为函数的对象
 * @return {boolean} 如果参数是函数，返回 true，否则 false
 */
function isFunction(arg) {
    if (arg) {
        if (typeof (./) !== 'function') {
            return typeof arg === 'function';
        } else {
            return Object.prototype.toString.call(arg) === '[object Function]';
        }
    } // end if
    return false;
}
```

426.

[问答题]

编写一个函数接受 url 中 query string 为参数，返回解析后的 Object，query string 使用 application/x-www-form-urlencoded 编码

来自：前端开发面试题

参考：

```
/**
 * 解析 query string 转换为对象，一个 key 有多个值时生成数组
```

```

*
* @param {String} query 需要解析的 query 字符串, 开头可以是?,
* 按照 application/x-www-form-urlencoded 编码
* @return {Object} 参数解析后的对象
*/
function parseQuery(query) {
    var result = {};
    // 如果不是字符串返回空对象
    if (typeof query !== 'string') {
        return result;
    }
    // 去掉字符串开头可能带的?
    if (query.charAt(0) === '?') {
        query = query.substring(1);
    }
    var pairs = query.split('&');
    var pair;
    var key, value;
    var i, len;
    for (i = 0, len = pairs.length; i < len; ++i) {
        pair = pairs[i].split('=');
        // application/x-www-form-urlencoded 编码会将' '转换为+
        key = decodeURIComponent(pair[0]).replace(/\+/g, ' ');
        value = decodeURIComponent(pair[1]).replace(/\+/g, ' ');
        // 如果是新 key, 直接添加
        if (!(key in result)) {
            result[key] = value;
        }
        // 如果 key 已经出现一次以上, 直接向数组添加 value
        else if (isArray(result[key])) {
            result[key].push(value);
        }
        // key 第二次出现, 将结果改为数组
        else {
            var arr = [result[key]];
            arr.push(value);
            result[key] = arr;
        } // end if-else
    } // end for
    return result;
}
function isArray(arg) {
    if (arg && typeof arg === 'object') {
        return Object.prototype.toString.call(arg) === '[object Array]';
    }
}

```

```

    }
    return false;
}
/**/
console.log(parseQuery('sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8'));
*/

```

427.

[问答题]

解析一个完整的 url，返回 Object 包含域与 window.location 相同

来自：前端开发面试题

参考：

```

/**
 * 解析一个 url 并生成 window.location 对象中包含的域
 * location:
 * {
 *   href: '包含完整的 url',
 *   origin: '包含协议到 pathname 之前的内容',
 *   protocol: 'url 使用的协议，包含末尾的:',
 *   username: '用户名', // 暂时不支持
 *   password: '密码', // 暂时不支持
 *   host: '完整主机名，包含:和端口',
 *   hostname: '主机名，不包含端口'
 *   port: '端口号',
 *   pathname: '服务器上访问资源的路径/开头',
 *   search: 'query string, ?开头',
 *   hash: '#开头的 fragment identifier'
 * }
 *
 * @param {string} url 需要解析的 url
 * @return {Object} 包含 url 信息的对象
 */
function parseUrl(url) {
  var result = {};
  var keys = ['href', 'origin', 'protocol', 'host',
             'hostname', 'port', 'pathname', 'search', 'hash'];
  var i, len;
  var regexp = /(([^\:]+\:)\:\/\/(([^\:\?\#]+)(\:(\d+)?))(\?[^?\#]*)(\?[^\#]*)(#[\.\#]*?)/;
  var match = regexp.exec(url);
  if (match) {
    for (i = keys.length - 1; i >= 0; --i) {
      result[keys[i]] = match[i];
    }
  }
  return result;
}

```

```

        result[keys[i]] = match[i] ? match[i] : "";
    }
}
return result;
}

```

428.

[问答题]

完成函数 getScrollOffset 返回窗口滚动条偏移量

来自：前端开发面试题

参考：

```

/**
 * 获取指定 window 中滚动条的偏移量，如未指定则获取当前 window
 * 滚动条偏移量
 *
 * @param {window} w 需要获取滚动条偏移量的窗口
 * @return {Object} obj.x 为水平滚动条偏移量,obj.y 为竖直滚动条偏移量
 */
function getScrollOffset(w) {
    w = w || window;
    // 如果是标准浏览器
    if (w.pageXOffset != null) {
        return {
            x: w.pageXOffset,
            y: w.pageYOffset
        };
    }
    // 老版本 IE，根据兼容性不同访问不同元素
    var d = w.document;
    if (d.compatMode === 'CSS1Compat') {
        return {
            x: d.documentElement.scrollLeft,
            y: d.documentElement.scrollTop
        }
    }
    return {
        x: d.body.scrollLeft,
        y: d.body.scrollTop
    };
}

```

429.

[问答题]

现有一个字符串 `richText`, 是一段富文本, 需要显示在页面上。有个要求, 需要给其中只包含一个 `img` 元素的 `p` 标签增加一个叫 `pic` 的 `class`。请编写代码实现。可以使用 `jQuery` 或 `KISSY`。

来自：前端开发面试题

参考：

```
function richText(text) {  
    var div = document.createElement('div');  
    div.innerHTML = text;  
    var p = div.getElementsByTagName('p');  
    var i, len;  
    for (i = 0, len = p.length; i < len; ++i) {  
        if (p[i].getElementsByTagName('img').length === 1) {  
            p[i].classList.add('pic');  
        }  
    }  
    return div.innerHTML;  
}
```

430.

[问答题]

请实现一个 `Event` 类, 继承自此类的对象都会拥有两个方法 `on` 和 `trigger`

来自：前端开发面试题

参考：无

431.

[问答题]

编写一个函数将列表子元素顺序反转

来自：前端开发面试题

参考：无

```
<ul id="target">  
    <li>1</li>
```

```

<li>2</li>
<li>3</li>
<li>4</li>
</ul>

<script>
    var target = document.getElementById('target');
    var i;
    var frag = document.createDocumentFragment();

    for (i = target.children.length - 1; i >= 0; --i) {
        frag.appendChild(target.children[i]);
    }
    target.appendChild(frag);
</script>

```

432.

[问答题]

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 = __1__;
        return this.str.replace(__2__, function (a, b) {
            return arg[b] || "";
        });
    };
    window.fn = fn;
})(window);
// use
(function () {
    var t = new fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');
    console.log(t.format('http://www.alibaba.com', 'Alibaba', 'Welcome'));
})();

```

来自：前端开发面试题

参考：无

`define` 部分定义一个简单的模板类，使用`{}`作为转义标记，中间的数字表示替换目标，`format` 实参用来替换模板内标记

横线处填：

```
Array.prototype.slice.call(arguments, 0)
/\{\s*(\d+)\s*\}/g
```

433.

[问答题]

编写一个函数实现 form 的序列化（即将一个表单中的键值序列化为可提交的字符串）

来自：前端开发面试题

参考：无

```
/*
 * 将一个表单元素序列化为可提交的字符串
 *
 * @param {FormElement} form 需要序列化的表单元素
 * @return {string} 表单序列化后的字符串
 */
function serializeForm(form) {
    if (!form || form.nodeName.toUpperCase() !== 'FORM') {
        return;
    }
    var result = [];
    var i, len;
    var field, fieldName, fieldType;

    for (i = 0, len = form.length; i < len; ++i) {
        field = form.elements[i];
        fieldName = field.name;
        fieldType = field.type;

        if (field.disabled || !fieldName) {
            continue;
        } // enf if

        switch (fieldType) {
            case 'text':
            case 'password':
            case 'hidden':
            case 'textarea':
                result.push(encodeURIComponent(fieldName) + '=' +
                           encodeURIComponent(field.value));
                break;
        }
    }
}
```

```

        case 'radio':
        case 'checkbox':
            if (field.checked) {
                result.push(encodeURIComponent(fieldName) + '=' +
                    encodeURIComponent(field.value));
            }
            break;

        case 'select-one':
        case 'select-multiple':
            for (var j = 0, jLen = field.options.length; j < jLen; ++j) {
                if (field.options[j].selected) {
                    result.push(encodeURIComponent(fieldName) + '=' +
                        encodeURIComponent(field.options[j].value || field.options[j].text));
                }
            } // end for
            break;

        case 'file':
        case 'submit':
            break; // 是否处理?

        default:
            break;
        } // end switch
    } // end for

    return result.join('&');
}

```

434.

[问答题]

使用原生 javascript 给下面列表中的 li 节点绑定点击事件，点击时创建一个 Object 对象，兼容 IE 和标准浏览器

来自：前端开发面试题

参考：无

```

<ul id="nav">
    <li><a href="http://11111">111</a></li>
    <li><a href="http://22222">222</a></li>
    <li><a href="http://33333">333</a></li>
    <li><a href="http://44444">444</a></li>

```

```
</ul>

Object:
{
    "index": 1,
    "name": "111",
    "link": "http://1111"
}

script:

var EventUtil = {
    getEvent: function (event) {
        return event || window.event;
    },
    getTarget: function (event) {
        return event.target || event.srcElement;
    },
    // 返回注册成功的监听器，IE 中需要使用返回值来移除监听器
    on: function (elem, type, handler) {
        if (elem.addEventListener) {
            elem.addEventListener(type, handler, false);
            return handler;
        } else if (elem.attachEvent) {
            function wrapper(event) {
                return handler.call(elem, event);
            };
            elem.attachEvent('on' + type, wrapper);
            return wrapper;
        }
    },
    off: function (elem, type, handler) {
        if (elem.removeEventListener) {
            elem.removeEventListener(type, handler, false);
        } else if (elem.detachEvent) {
            elem.detachEvent('on' + type, handler);
        }
    },
    preventDefault: function (event) {
        if (event.preventDefault) {
            event.preventDefault();
        } else if ('returnValue' in event) {
            event.returnValue = false;
        }
    },
}
```

```

stopPropagation: function (event) {
    if (event.stopPropagation) {
        event.stopPropagation();
    } else if ('cancelBubble' in event) {
        event.cancelBubble = true;
    }
}
};

var DOMUtil = {
    text: function (elem) {
        if ('textContent' in elem) {
            return elem.textContent;
        } else if ('innerText' in elem) {
            return elem.innerText;
        }
    },
    prop: function (elem, propName) {
        return elem.getAttribute(propName);
    }
};

var nav = document.getElementById('nav');

EventUtil.on(nav, 'click', function (event) {
    var event = EventUtil.getEvent(event);
    var target = EventUtil.getTarget(event);

    var children = this.children;
    var i, len;
    var anchor;
    var obj = {};

    for (i = 0, len = children.length; i < len; ++i) {
        if (children[i] === target) {
            obj.index = i + 1;
            anchor = target.getElementsByTagName('a')[0];
            obj.name = DOMUtil.text(anchor);
            obj.link = DOMUtil.prop(anchor, 'href');
        }
    }

    alert('index: ' + obj.index + ' name: ' + obj.name +
        ' link: ' + obj.link);
});

```

435.

[问答题]

有一个大数组，`var a = ['1', '2', '3', ...];` a 的长度是 100，内容填充随机整数的字符串。请先构造此数组 a，然后设计一个算法将其内容去重

来自：前端开发面试题

参考：无

```
/***
 * 数组去重
 */
function normalize(arr) {
    if (arr && Array.isArray(arr)) {
        var i, len, map = {};
        for (i = arr.length; i >= 0; --i) {
            if (arr[i] in map) {
                arr.splice(i, 1);
            } else {
                map[arr[i]] = true;
            }
        }
        return arr;
    }

    /**
     * 用 100 个随机整数对应的字符串填充数组。
     */
    function fillArray(arr, start, end) {
        start = start == undefined ? 1 : start;
        end = end == undefined ? 100 : end;

        if (end <= start) {
            end = start + 100;
        }

        var width = end - start;
        var i;
        for (i = 100; i >= 1; --i) {
            arr.push(" " + (Math.floor(Math.random() * width) + start));
        }
        return arr;
    }
}
```

```
}

var input = [];
fillArray(input, 1, 100);
input.sort(function (a, b) {
    return a - b;
});
console.log(input);

normalize(input);
console.log(input);
```

436.

[谈话题]

前端是庞大的，包括 HTML、CSS、Javascript、Image、Flash 等等各种各样的资源。前端优化是复杂的，针对方方面面的资源都有不同的方式。随便聊聊有关优化方面的技术话题。

来自：前端开发面试题

参考：

前端优化的目的是什么？

1. 从用户角度而言，优化能够让页面加载得更快、对用户的操作响应得更及时，能够给用户提供更为友好的体验。

2. 从服务商角度而言，优化能够减少页面请求数、或者减小请求所占带宽，能够节省可观的资源。

总之，恰当的优化不仅能够改善站点的用户体验并且能够节省相当的资源利用。

前端优化的途径有很多，按粒度大致可以分为两类，第一类是页面级别的优化，例如 HTTP 请求数、脚本的无阻塞加载、内联脚本的位置优化等；第二类则是代码级别的优化，例如 Javascript 中的 DOM 操作优化、CSS 选择符优化、图片优化以及 HTML 结构优化等等。另外，本着提高投入产出比的目的，后文提到的各种优化策略大致按照投入产出比从大到小的顺序排列。

一、页面级优化

1. 减少 HTTP 请求数

这条策略基本上所有前端人都知道，而且也是最重要最有效的。都说要减少 HTTP 请求，那请求多了到底会怎么样呢？首先，每个请求都是有成本的，既包含时间成本也包含资源成本。一个完整的请求都需要经过 DNS 寻址、与服务器建立连接、发送数据、等待服务器响应、接收数据这样一个“漫长”而复杂的过程。时间成本就是用户需要看到或者“感受”到这个资源是必须要等待这个过程结束的，资源上由于每个请求都需要携带数据，因此每个请求都需要占用带宽。另外，由于浏览器进行并发请求的请求数是有上限的（具体参见此处），因此请求数多了以后，浏览器需要分批进行请求，因此会增加用户的等待时间，会给用户造成站点速度慢这样一个印象，即使可能用户能看到的第一屏的资源都已经请求完了，但是浏览器的进度条会一直存在。

减少 HTTP 请求数的主要途径包括：

(1). 从设计实现层面简化页面

如果你的页面像百度首页一样简单，那么接下来的规则基本上用不着了。保持页面简洁、减少资源的使用时最直接的。如果不是这样，你的页面需要华丽的皮肤，则继续阅读下面的内容。

(2). 合理设置 HTTP 缓存

缓存的力量是强大的，恰当的缓存设置可以大大的减少 HTTP 请求。以有啊首页为例，当浏览器没有缓存的时候访问一共会发出 78 个请求，共 600 多 K 数据(如图 1.1)，而当第二次访问即浏览器已缓存之后访问则仅有 10 个请求，共 20 多 K 数据(如图 1.2)。(这里需要说明的是，如果直接 F5 刷新页面的话效果是不一样的，这种情况下请求数还是一样，不过被缓存资源的请求服务器是 304 响应，只有 Header 没有 Body，可以节省带宽)

怎样才算合理设置?原则很简单，能缓存越多越好，能缓存越久越好。例如，很少变化的图片资源可以直接通过 HTTP Header 中的 `Expires` 设置一个很长的过期头;变化不频繁而又可能会变的资源可以使用 `Last-Modified` 来做请求验证。尽可能的让资源能够在缓存中待得更久。关于 HTTP 缓存的具体设置和原理此处就不再详述了，有兴趣的可以参考下列文章：

[HTTP1.1 协议中关于缓存策略的描述](#)

[Fiddler HTTP Performance 中关于缓存的介绍](#)

(3). 资源合并与压缩

如果可以的话，尽可能的将外部的脚本、样式进行合并，多个合为一个。另外，CSS、Javascript、Image 都可以用相应的工具进行压缩，压缩后往往能省下不少空间。

(4). CSS Sprites

合并 CSS 图片，减少请求数的又一个好办法。

(5). Inline Images

使用 `data: URL scheme` 的方式将图片嵌入到页面或 CSS 中，如果不考虑资源管理上的问题的话，不失为一个好办法。如果是嵌入页面的话换来的是增大了页面的体积，而且无法利用浏览器缓存。使用在 CSS 中的图片则更为理想一些。

(6). Lazy Load Images

这条策略实际上并不一定能减少 HTTP 请求数，但是却能在某些条件下或者页面刚加载时减少 HTTP 请求数。对于图片而言，在页面刚加载的时候可以只加载第一屏，当用户继续往后滚屏的时候才加载后续的图片。这样一来，假如用户只对第一屏的内容感兴趣时，那剩余的图片请求就都节省了。有啊首页曾经的做法是在加载的时候把第一屏之后的图片地址缓存在 `Textarea` 标签中，待用户往下滚屏的时候才“惰性”加载。

2. 将外部脚本置底

前文有谈到，浏览器是可以并发请求的，这一特点使得其能够更快的加载资源，然而外链脚本在加载时却会阻塞其他资源，例如在脚本加载完成之前，它后面的图片、样式以及其他脚本都处于阻塞状态，直到脚本加载完成后才会开始加载。如果将脚本放在比较靠前的位置，则会影响整个页面的加载速度从而影响用户体验。解决这一问题的方法有很多，在这里有比较详细的介绍(这里是译文和更详细的例子)，而最简单可依赖的方法就是将脚本尽可能的往后挪，减少对并发下载的影响。

3. 异步执行 inline 脚本

`inline` 脚本对性能的影响与外部脚本相比，是有过之而无不及。首页，与外部脚本一样，`inline` 脚本在执行的时候一样会阻塞并发请求，除此之外，由于浏览器在页面处理方面是单线程的，当 `inline` 脚本在页面渲染之前执行时，页面的渲染工作则会被推迟。简而言之，`inline` 脚本在执行的时候，页面处于空白状态。鉴于以上两点原因，建议将执行时间较长的 `inline` 脚本异步执行，异步的方式有很多种，例如使用 `script` 元素的 `defer` 属性(存在兼

容性问题和其他一些问题，例如不能使用 `document.write()`、使用 `setTimeout`，此外，在 HTML5 中引入了 Web Workers 的机制，恰恰可以解决此类问题。

4. Lazy Load Javascript

随着 Javascript 框架的流行，越来越多的站点也使用起了框架。不过，一个框架往往包括了很多的功能实现，这些功能并不是每一个页面都需要的，如果下载了不需要的脚本则算得上是一种资源浪费-既浪费了带宽又浪费了执行花费的时间。目前的做法大概有两种，一种是为那些流量特别大的页面专门定制一个专用的 mini 版框架，另一种则是 Lazy Load。YUI 则使用了第二种方式，在 YUI 的实现中，最初只加载核心模块，其他模块可以等到需要使用的时候才加载。

5. 将 CSS 放在 HEAD 中

如果将 CSS 放在其他地方比如 BODY 中，则浏览器有可能还未下载和解析到 CSS 就已经开始渲染页面了，这就导致页面由无 CSS 状态跳转到 CSS 状态，用户体验比较糟糕。除此之外，有些浏览器会在 CSS 下载完成后才开始渲染页面，如果 CSS 放在靠下的位置则会导致浏览器将渲染时间推迟。

6. 异步请求 Callback

在某些页面中可能存在这样一种需求，需要使用 `script` 标签来异步的请求数据。类似：

Javascript:

```
/*Callback 函数*/
function myCallback(info){
//do something here
}
```

HTML:

```
<script type="text/javascript" src="http://abc.com/cb"></script>
cb 返回的内容:
myCallback('Hello world!');
```

像以上这种方式直接在页面上写`<script>`对页面的性能也是有影响的，即增加了页面首次加载的负担，推迟了 `DOMContentLoaded` 和 `window.onload` 事件的触发时机。如果时效性允许的话，可以考虑在 `DOMContentLoaded` 事件触发的时候加载，或者使用 `setTimeout` 方式来灵活的控制加载的时机。

7. 减少不必要的 HTTP 跳转

对于以目录形式访问的 HTTP 链接，很多人都会忽略链接最后是否带' /'，假如你的服务器对此是区别对待的话，那么你也需要注意，这其中很可能隐藏了 301 跳转，增加了多余请求。具体参见下图，其中第一个链接是以无' /'结尾的方式访问的，于是服务器有一次跳转。

8. 避免重复的资源请求

这种情况主要是由于疏忽或页面由多个模块拼接而成，然后每个模块中请求了同样的资源时，会导致资源的重复请求

二、代码级优化

1. Javascript

(1). DOM

DOM 操作应该是脚本中最耗性能的一类操作，例如增加、修改、删除 DOM 元素或者对 DOM 集合进行操作。如果脚本中包含了大量的 DOM 操作则需要注意以下几点：

a. HTML Collection

在脚本中 `document.images`、`document.forms`、`getElementsByName()` 返回的都是 `HTMLCollection` 类型的集合，在平时使用的时候大多将它作为数组来使用，因为它有 `length` 属性，也可以使用索引访问每一个元素。不过在访问性能上则比数组要差很多，原因是这个集合并不是一个静态的结果，它表示的仅仅是一个特定的查询，每次访问该集合时都会重新执行这个查询从而更新查询结果。所谓的“访问集合”包括读取集合的 `length` 属性、访问集合中的元素。

因此，当你需要遍历 `HTML Collection` 的时候，尽量将它转为数组后再访问，以提高性能。即使不转换为数组，也请尽可能少的访问它，例如在遍历的时候可以将 `length` 属性、成员保存到局部变量后再使用局部变量。

b. Reflow & Repaint

除了上面一点之外，DOM 操作还需要考虑浏览器的 `Reflow` 和 `Repaint`，因为这些都是需要消耗资源的，具体的可以参加以下文章：

如何减少浏览器的 `repaint` 和 `reflow`?

[Understanding Internet Explorer Rendering Behaviour](#)

[Notes on HTML Reflow](#)

(2). 慎用 `with`

`with(obj){ p = 1;}` 代码块的行为实际上是修改了代码块中的执行环境，将 `obj` 放在了其作用域链的最前端，在 `with` 代码块中访问非局部变量是先从 `obj` 上开始查找，如果没有再依次按作用域链向上查找，因此使用 `with` 相当于增加了作用域链长度。而每次查找作用域链都是要消耗时间的，过长的作用域链会导致查找性能下降。

因此，除非你能肯定在 `with` 代码中只访问 `obj` 中的属性，否则慎用 `with`，替代的可以使用局部变量缓存需要访问的属性。

(3). 避免使用 `eval` 和 `Function`

每次 `eval` 或 `Function` 构造函数作用于字符串表示的源代码时，脚本引擎都需要将源代码转换成可执行代码。这是很消耗资源的操作——通常比简单的函数调用慢 100 倍以上。

`eval` 函数效率特别低，由于事先无法知晓传给 `eval` 的字符串中的内容，`eval` 在其上下文中解释要处理的代码，也就是说编译器无法优化上下文，因此只能有浏览器在运行时解释代码。这对性能影响很大。

`Function` 构造函数比 `eval` 略好，因为使用此代码不会影响周围代码；但其速度仍很慢。

此外，使用 `eval` 和 `Function` 也不利于 `Javascript` 压缩工具执行压缩。

(4). 减少作用域链查找

前文谈到了作用域链查找问题，这一点在循环中是尤其需要注意的问题。如果在循环中需要访问非本作用域下的变量时请在遍历之前用局部变量缓存该变量，并在遍历结束后再重写那个变量，这一点对全局变量尤其重要，因为全局变量处于作用域链的最顶端，访问时的查找次数是最多的。

低效率的写法：

```
//全局变量
var globalVar = 1;
function myCallback(info){
    for( var i = 100000; i--;){
        //每次访问 globalVar 都需要查找到作用域链最顶端，本例中需要访问 100000 次
        globalVar += i;
```

```
    }
}

更高效的写法:

//全局变量
var globalVar = 1;
function myCallback(info){
    //局部变量缓存全局变量
    var localVar = globalVar;
    for( var i = 100000; i--;){
        //访问局部变量是最快的
        localVar += i;
    }
    //本例中只需要访问 2 次全局变量
    globalVar = localVar;
}
```

此外，要减少作用域链查找还应该减少闭包的使用。

(5). 数据访问

Javascript 中的数据访问包括直接量(字符串、正则表达式)、变量、对象属性以及数组，其中对直接量和局部变量的访问是最快的，对对象属性以及数组的访问需要更大的开销。当出现以下情况时，建议将数据放入局部变量：

- a. 对任何对象属性的访问超过 1 次
- b. 对任何数组成员的访问次数超过 1 次

另外，还应当尽可能的减少对对象以及数组深度查找。

(6). 字符串拼接

在 Javascript 中使用 "+" 号来拼接字符串效率是比较低的，因为每次运行都会开辟新的内存并生成新的字符串变量，然后将拼接结果赋值给新变量。与之相比更为高效的做法是使用数组的 `join` 方法，即将需要拼接的字符串放在数组中最后调用其 `join` 方法得到结果。不过由于使用数组也有一定的开销，因此当需要拼接的字符串较多的时候可以考虑用此方法。

关于 Javascript 优化的更详细介绍请参考：

[Write Efficient Javascript\(PPT\)](#)

[Efficient JavaScript](#)

2. CSS 选择符

在大多数人的观念中，都觉得浏览器对 CSS 选择符的解析式从左往右进行的，例如

```
#toc A { color: #444; }
```

这样一个选择符，如果是从右往左解析则效率会很高，因为第一个 ID 选择基本上就把查找的范围限定了，但实际上浏览器对选择符的解析是从右往左进行的。如上面的选择符，浏览器必须遍历查找每一个 A 标签的祖先节点，效率并不像之前想象的那样高。根据浏览器的这一行为特点，在写选择符的时候需要注意很多事项，有人已经一一列举了，详情参考此处。

3. HTML

对 HTML 本身的优化现如今也越来越多的受人关注了，详情可以参见这篇总结性文章。

4. Image 压缩

图片压缩是个技术活，不过现如今这方面的工具也非常多，压缩之后往往能带来不

错的效果，具体的压缩原理以及方法在《Even Faster Web Sites》第 10 章有很详细的介绍，有兴趣的可以去看看。

总结

上面从页面级以及代码级两个粒度对前端优化的各种方式做了一个总结，这些方法基本上都是前端开发人员在开发的过程中可以借鉴和实践的，除此之外，完整的前端优化还应该包括很多其他的途径，例如 CDN、Gzip、多域名、无 Cookie 服务器等等，由于对于开发人员的可操作性并不强大，在此也就不多叙述了，详细的可以参考 Yahoo 和 Google 的网站优化。

437.

[谈话题]

简单说说 Javascript 的历史。

来自：前端开发面试易考知识点

参考：

HTML 是纯静态的的页面，而 Javascript 让页面有了动态的效果，比如;OA 中模块的拖拉所有的浏览器都会内置 Javascript 的解释器

1992 年 Nombas 公司开发出 C 减减的嵌入式脚本语言。这是最好的 HTML 页面的脚本语言。

Netscape 为了扩展其浏览器的功能，开发了一套 LiveScript，并与 1995 年与 SUN 公司联合把其改名为 javascript,它的主要目的是处理一些输入的有效性验证，而之前这个工作是留给 perl 之类的服务器端语言完成，在以前使用电话线调制解调器(28.8kb/s)的时代，如此慢的与服务器交互，这绝对是一件很痛苦的事情。Javascript 的出现，解决了这个问题，因为它的验证是基于客户端的。

微软公司早期版本的浏览器仅支持自己的 vbscript，但后来不得不加入 javascript

IE3 中搭载 Javascript 的克隆版本，命名为 jscript

在一次技术会议中，sun, microsoft, netscape 公司联合制定了 ECMA-Script 标准

在 2005 前，网页上提示框，广告越来越多，把 javascript 滥用，使 javascript 背上了大量的罪名。

2005 年，google 公司的网上产品（google 地图，gmail，google 搜索建议）等使得 ajax 兴起，而 javascript 便是 ajax 最重要的元素之一

Javascript 有三个部分组成

ECMAScript DOM BOM

WEB 标准

网页主要有三部分组成

（结构 HTML,XHTML，表现 CSS，行为 DOM，ECMA）

438.

[思考题]

Javascript 的基本语法

来自：前端开发面试易考知识点

重点提示：

1. 区分大小写
2. 弱类型变量 var age=10 var name=" dd"
3. 每行结尾的分号可有可无，但建议还是加上
4. 注释与 java 相同

变量

变量是通过 var 关键字来声明的。（Variable）

变量的命名规则与 java 一致

注释有三种：// /**/ <!-- --> 这个只能注释单行

439.

[比较题]

slice()、substring()、substr

来自：前端开发面试易考知识点

重点提示：

Slice 和 substring (2,5) 指的是从第 3 为开始，取 (5-2) =3 个数，其中 slice 的参数可以为负

Substr (2,5) 指的是从第 3 为开始，取 5 个数。但 ECMAScript 没有对该方法进行标准化，因此尽量少使用该方法

440.

[比较题]

indexOf()和 lastIndexOf(), isNaN, typeOf

来自：前端开发面试易考知识点

重点提示：

indexOf (" i") //从前往后,i 在第几位

indexOf (" i",3) 可选参数，从第几个字符开始往后找

lastIndexOf (" i") //从后往前,i 在第几位

lastIndexOf (" i",3) //从后往前,i 在第几位

如果没找到，则返回-1

String 类型的变量，在 Java 中，用“”符号表示字符串，用' ' 表示单个字符。而在 javascript 中这两种都可以

Nan (not a number)

Alert(nan ==nan)返回 false,因此不推荐使用 nan 本身，推荐函数 isNaN

```
Alert(isNaN( "ab" ));//返回 false  
typeof 运算符  
var stmp = "test";  
alert(typeof stmp); //输出 string  
alert(type of 1);//输出 number  
此外：还有 boolean,undefined,object(如果是引用类型或者 null 值,null 值返回 object，这  
其实是 ECMAScript 的一个错误)
```

当声明的变量未初始化的时候，它的值就是 undefined. 当没有这个变量的时候，typeof 变

返回的值也是 undefined。但是没声明的变量是不能参与计算的。

当函数无明确返回值时，返回的也是 undefined

```
Function a(){  
}  
Alert(a() == undefined) //返回 true  
Alert(null == undefined)//返回 true
```

441.

[技巧题]

数值计算

来自：前端开发面试易考知识点

重点提示：

```
var mynum1 = 23.345;  
var mynum2 = 45;  
var mynum3 = -34;  
var mynum4 = 9e5;      //科学计数法 为 9*10 五次方  
var fNumber = 123.456;  
alert(fNumber.toExponential(1));//保留的小数点数 1.2e+2  
alert(fNumber.toExponential(2));//1.23e+2
```

442.

[技巧题]

布尔值

来自：前端开发面试易考知识点

重点提示：

```
var married = true;  
alert("1." + typeof(married));//Boolean  
married = "true";
```

```
alert("2." + typeof(married));//String
```

443.

[技巧题]

类型转换

来自：前端开发面试易考知识点

重点提示：

转换成 string 类型有三种方式

```
var a = 3;  
var b = a + "";  
var c = a.toString();  
var d = "student" + a;
```

toString()

```
var a=11;  
document.write(a.toString(2) + "<br>");//转成 2 进制  
document.write(a.toString(3) + "<br>");//转成 3 进制  
如果不是数值，则转换报错
```

parseInt()

```
document.write(parseInt("1red6") + "<br>");//返回 1，后面非数值的将全部忽略  
document.write(parseInt("53.5") + "<br>");//返回 53  
document.write(parseInt("0xC") + "<br>"); //直接十进制转换 12  
document.write(parseInt("isaacshun@gmail.com") + "<br>");//NAN  
document.write(parseInt("011",8) + "<br>"); 返回 9  
document.write(parseInt("011",10) + "<br>"); //指定为十进制 返回 11  
parseFloat()与 parseInt()类似
```

444.

[思考题]

数组

来自：前端开发面试易考知识点

重点提示：

```
var aMap = new Array("China","USA","Britain");  
aMap[20] = "Korea";  
alert(aMap.length + " " + aMap[10] + " " + aMap[20]);//aMap[10]返回 undefined  
document.write(aMap.join("[") + "<br>"); //用 “][” 来连接  
var sFruit = "apple,pear,peach,orange";
```

```
var aFruit = sFruit.split(",");
var aFruit = ["apple","pear","peach","orange"];
alert(aFruit.reverse().toString());

var aFruit = ["pear","apple","peach","orange"];
aFruit.sort();

var stack = new Array();
stack.push("red");
stack.push("green");
stack.push("blue");
document.write(stack.toString() + "<br>");
var vItem = stack.pop(); // blue
document.write(vItem + "<br>");
document.write(stack.toString()); // red green
```

445.

[思考题]

if 语句

来自：前端开发面试易考知识点

用法提示：

```
//首先获取用户的一个输入，并用 Number()强制转换为数字
var iNumber = Number(prompt("输入一个 5 到 100 之间的数字", ""));//第二个参数,用于显示输入框的默认值
if(isNaN(iNumber))      //判断输入的是否是数字 NaN “Not a Number”
    document.write("请确认你的输入正确");
else if(iNumber > 100 || iNumber < 5)      //判断输入的数字范围
    document.write("你输入的数字范围不在 5 和 100 之间");
else
    document.write("你输入的数字是：" + iNumber);
```

446.

[思考题]

switch

来自：前端开发面试易考知识点

用法提示：

```
iWeek = parseInt(prompt("输入 1 到 7 之间的整数",""));
switch(iWeek){
    case 1:
        document.write("Monday");
        break;
    case 2:
        document.write("Tuesday");
        break;
    case 3:
        document.write("Wednesday");
        break;
    case 4:
        document.write("Thursday");
        break;
    case 5:
        document.write("Friday");
        break;
    case 6:
        document.write("Saturday");
        break;
    case 7:
        document.write("Sunday");
        break;
    default:
        document.write("Error");
}
```

447.

[思考题]
while 语句

来自：前端开发面试易考知识点

用法提示：

```
var i=iSum=0;
while(i<=100){
    iSum += i;
    i++;
}
alert(iSum);
do{while for break continue (与 JAVA 语法一致)}
```

448.

[思考题]

函数

来自：前端开发面试易考知识点

用法提示：

```
function ArgsNum(){  
    return arguments.length;  
}  
  
document.write(ArgsNum("isaac",25) + "<br>");  
document.write(ArgsNum() + "<br>");  
document.write(ArgsNum(3) + "<br>");
```

从这个例子可以看出，方法可以没有参数，也可以没有返回值，但是照样可以传入参数和返回值。

449.

[思考题]

Date 对象

来自：前端开发面试易考知识点

重点提示：

```
var myDate1 = new Date(); //运行代码前的时间  
for(var i=0;i<3000000;i++);  
  
var myDate2 = new Date(); //运行代码后的时间  
document.write(myDate2);  
  
var oMyDate = new Date();  
var iYear = oMyDate.getFullYear();  
var iMonth = oMyDate.getMonth() + 1; //月份是从 0 开始的  
var iDate = oMyDate.getDate();  
var iDay = oMyDate.getDay(); //0 为 星期日 1 为 星期一  
function disDate(oDate, iDate){  
    var ms = oDate.getTime(); //换成毫秒数  
    ms -= iDate*24*60*60*1000; //计算相差的毫秒数  
    return new Date(ms); //返回新的时间对象  
}  
  
var oBeijing = new Date(2008,7,8);  
var iNum = 100; //前 100 天  
var oMyDate = disDate(oBeijing, iNum);
```

450.

[思考题]

检测浏览器和操作系统

来自：前端开发面试易考知识点

提示：

```
var sUserAgent = navigator.userAgent;
//检测 Opera、KHTML
var isOpera = sUserAgent.indexOf("Opera") > -1;
var isKHTML = sUserAgent.indexOf("KHTML") > -1 || sUserAgent.indexOf("Konqueror") > -1
|| sUserAgent.indexOf("AppleWebKit") > -1;
//检测 IE、Mozilla
var isIE = sUserAgent.indexOf("compatible") > -1 && sUserAgent.indexOf("MSIE") > -1
&& !isOpera;
var isMoz = sUserAgent.indexOf("Gecko") > -1 && !isKHTML;
//检测操作系统
var isWin = (navigator.platform == "Win32") || (navigator.platform == "Windows");
var isMac = (navigator.platform == "Mac68K") || (navigator.platform == "MacPPC") ||
(navigator.platform == "Macintosh");
var isUnix = (navigator.platform == "X11") && !isWin && !isMac;
if(isOpera) document.write("Opera ");
if(isKHTML) document.write("KHTML ");
if(isIE) document.write("IE ");
if(isMoz) document.write("Mozilla ");
if(isWin) document.write("Windows");
if(isMac) document.write("Mac");
if(isUnix) document.write("Unix");
```

451.

[思考题]

Global 对象

来自：前端开发面试易考知识点

重点提示：

其实 isNaN, parseInt 等都是 Global 对象的方法

EncodeURI.因为有效的 URI 不能包含某些字符，如空格。这个方法就是用于将这个字符转换成 UTF-8 编码，使浏览器可以接受他们。

```
Var suil = "www.oseschool.com/pro file/a.html";
Alert(encodeURI(suil));//www.oseschool.com/pro%20file/a.html
```

即将空格编码成%20

Eval 方法

```
Eval("alert('hi')");
```

当解析程序发现 eval() 时，它将把参数解析成真正的 ECMA-script 语句，然后插入该语句所在位置。

Global 除了有内置方法外，还有很多内置的属性：如：undefined, nan, Array, String, Number, Date, RegExp 等

452.

[思考题]

Math 对象

来自：前端开发面试易考知识点

提示：

Max 方法， min 方法,ceil,floor,round,sqrt,random

```
Max(1,2,3);min(1.2,3.4);
```

想取到 1~10 的数据

```
Math.floor(Math.random()*10+1)
```

2~9 的数据

```
Math.floor(Math.random()*9+2);
```

453.

[思考题]

getElementsByTagName

来自：前端开发面试易考知识点

提示：

```
function searchDOM(){
```

//放在函数内，页面加载完成后才用<body>的 onload 加载，这时如果把 alert 这句改成用 document.write 则会把原内容覆盖掉，因为是后面才执行的

```
var oLi = document.getElementsByTagName("li");
```

//输出长度、标签名称以及某项的文本节点值

```
alert(oLi.length + " " + oLi[0].tagName + " " + oLi[3].childNodes[0].nodeValue);
```

```
var oUl = document.getElementsByTagName("ul");
```

```
var oLi2 = oUl[1].getElementsByTagName("li");
```

```
alert(oLi2.length + " " + oLi2[0].tagName + " " + oLi2[1].childNodes[0].nodeValue);
```

```
}
```

```
<body onload="searchDOM()">
```

```
<ul>客户端语言
    <li>HTML</li>
    <li>JavaScript</li>
    <li id="cssLi">CSS</li>
</ul>
<ul>服务器端语言
    <li>ASP.NET</li>
    <li>JSP</li>
    <li>PHP</li>
</ul>
</body>
```

454.

[思考题]

getElementById

来自：前端开发面试易考知识点

提示：

```
var oLi = document.getElementById("cssLi");
oLi.style.backgroundColor="yellow"
//输出标签名称以及文本节点值
alert(oLi.tagName + " " + oLi.childNodes[0].nodeValue);
```

455.

[思考题]

getElementsByName

来自：前端开发面试易考知识点

提示：

```
alert(document.getElementsByName("a").length);
```

456.

[技巧题]

访问子节点

来自：前端开发面试易考知识点

参考：

```
function myDOMInspector(){  
    var oUl = document.getElementById("myList"); //获取<ul>标记  
    var DOMString = "";  
    if(oUl.hasChildNodes()){ //判断是否有子节点  
        var oCh = oUl.childNodes;  
        for(var i=0;i<oCh.length;i++) //逐一查找  
            DOMString += oCh[i].nodeName + "\n";  
    }  
    alert(DOMString);  
}
```

457.

[技巧题]

访问父节点

来自：前端开发面试易考知识点

参考：

```
nodeName 如果为文本节点，则返回#text  
tagName 如果为文本节点，则返回 undefined  
function myDOMInspector(){  
    var myItem = document.getElementById("myDearFood");  
    alert(myItem.parentNode.tagName);  
}  
  
function myDOMInspector(){  
    var myItem = document.getElementById("myDearFood");  
    var parentElm = myItem.parentNode;  
    while(parentElm.className != "colorful" && parentElm != document.body)  
        parentElm = parentElm.parentNode; //一路往上找  
    alert(parentElm.tagName);  
}  
<body onload="myDOMInspector()">  
<div class="colorful">  
    <ul>  
        <li>糖醋排骨</li>  
        <li>圆笼粉蒸肉</li>  
        <li>泡菜鱼</li>  
        <li id="myDearFood">板栗烧鸡</li>  
        <li>麻婆豆腐</li>  
    </ul>  
</div>
```

```
</body>
```

458.

[技巧题]

访问兄弟节点

来自：前端开发面试易考知识点

参考：

```
function myDOMInspector(){
    var myItem = document.getElementById("myDearFood");
    //访问兄弟节点
    var nextListItem = myItem.nextSibling;
    var preListItem = myItem.previousSibling;
    alert(nextListItem.tagName + " " + preListItem.tagName);
}
```

在 Firefox 中不支持，但是 IE 中却是支持的。

459.

[技巧题]

第一个 最后一个 子节点

来自：前端开发面试易考知识点

参考：

```
function nextSib(node){
    var tempLast = node.parentNode.lastChild;
    //判断是否是最后一个节点，如果是则返回 null
    if(node == tempLast)
        return null;
    var tempObj = node.nextSibling;
    //逐一搜索后面的兄弟节点，直到发现元素节点为止
    while(tempObj.nodeType!=1 && tempObj.nextSibling!=null)
        tempObj = tempObj.nextSibling;
    //三目运算符，如果是元素节点则返回节点本身，否则返回 null
    return (tempObj.nodeType==1)?tempObj:null;
}
function prevSib(node){
    var tempFirst = node.parentNode.firstChild;
    //判断是否是第一个节点，如果是则返回 null
    if(node == tempFirst)
```

```

        return null;
    var tempObj = node.previousSibling;
    //逐一搜索前面的兄弟节点，直到发现元素节点为止
    while(tempObj.nodeType!=1 && tempObj.previousSibling!=null)
        tempObj = tempObj.previousSibling;
    return (tempObj.nodeType==1)?tempObj:null;
}

function myDOMInspector(){
    var myItem = document.getElementById("myDearFood");
    //获取后一个元素兄弟节点
    var nextListItem = nextSib(myItem);
    //获取前一个元素兄弟节点
    var preListItem = prevSib(myItem);
    alert("后一项:" + ((nextListItem!=null)?nextListItem.firstChild.nodeValue:null) + " 前一项:" + ((preListItem!=null)?preListItem.firstChild.nodeValue:null) );
}

```

460.

[思考题]

nodeType

来自：前端开发面试易考知识点

参考：

元素 element 1

属性 attr 2

文本 text 3

注释 comments 8

文档 document 9

function showAttr(){

var btnShowAttr=document.getElementById("btnShowAttr"); //演示按钮，有很多属性

var attrs=btnShowAttr.attributes;

for(var i=0;i<attrs.length ;i++){

 var attr=attrs[i];

 alert('nodeType:' + attr.nodeType); //attribute 的 nodeType=2

 alert('attr:' + attr);

 alert('attr.name:' + attr.name + '=' + attr.value);

}

function showDocument(){

 alert('nodeType:' + document.nodeType); //9

 alert('nodeName:' + document.nodeName);

```
    alert(document);
}
```

461.

[技巧题]
getAttribute setAttribute

来自：前端开发面试易考知识点

参考：

```
function myDOMInspector(){
    //获取图片
    var myImg = document.getElementsByTagName("img")[0];
    //获取图片 title 属性
    alert(myImg.getAttribute("title"));

}

function changePic(){
    //获取图片
    var myImg = document.getElementsByTagName("img")[0];
    //设置图片 src 和 title 属性
    myImg.setAttribute("src","02.jpg");
    myImg.setAttribute("title","紫荆公寓");
}
```

462.

[技巧题]
创建新节点

来自：前端开发面试易考知识点

参考：

```
function createP(){
    var op = document.createElement("p");
    var otext = document.createTextNode("HHHHH");
    op.appendChild(otext);
    op.setAttribute("style","text-align:center");
    document.body.appendChild(op);

    //创建完节点，就马上会影响到下面的操作，比如 P 的数量就会多 1 个
}
```

463.

[技巧题]

删除节点

来自：前端开发面试易考知识点

参考：

需要注意的是标签之间的父子关系!!!

```
function deleteP(){
    var oP = document.getElementsByTagName("p")[0];
    oP.parentNode.removeChild(oP);      //删除节点
}
```

464.

[技巧题]

替换节点

来自：前端开发面试易考知识点

参考：

```
function replaceP(){
    var oOldP = document.getElementsByTagName("p")[0];
    var oNewP = document.createElement("p");      //新建节点
    var oText = document.createTextNode("这是一个感人肺腑的故事");
    oNewP.appendChild(oText);
    oOldP.parentNode.replaceChild(oNewP,oOldP);      //替换节点
}
```

465.

[技巧题]

插入节点

来自：前端开发面试易考知识点

参考：

```
function insertP(){
    var oOldP = document.getElementsByTagName("p")[0];
    var oNewP = document.createElement("p");      //新建节点
    var oText = document.createTextNode("这是一个感人肺腑的故事");
```

```

oNewP.appendChild(oText);
oOldP.parentNode.insertBefore(oNewP,oOldP);      //插入节点
}

没有 insertAfter，但是可以自己写一个
function insertAfter(newElement, targetElement){
    var oParent = targetElement.parentNode; //首先找到目标元素的父元素
    if(oParent.lastChild == targetElement) //如果目标元素已经是最后一个子元素了
        oParent.appendChild(newElement); //则直接用 appendChild()加到子元素列表的最后
    else                                //否则用 insertBefore()插入到目标元素的下一个兄弟元素之前
        oParent.insertBefore(newElement,targetElement.nextSibling);
}

function insertP(){
    var oOldP = document.getElementById("myTarget");
    var oNewP = document.createElement("p"); //新建节点
    var oText = document.createTextNode("这是一个感人肺腑的故事");
    oNewP.appendChild(oText);
    insertAfter(oNewP,oOldP);      //插入节点
}

```

其实这个也是通过 insertBefore 原理来实现的

466.

[技巧题]
cloneNode(deepBoolean)

来自：前端开发面试易考知识点

参考：

复制并返回当前节点的复制节点，这个复制得到的节点是一个孤立的节点，不在 document 树中。复制原来节点的属性值，包括 ID 属性，所以在把这个新节点加到 document 之前，一定要修改 ID 属性，以便使它保持唯一。当然如果 ID 的唯一性不重要可以不做处理。

这个方法支持一个布尔参数，当 deepBoolean 设置 true 时，复制 当前节点的所有子节点，包括该节点内的文本。

```

<p id="mypara">11111</p>
p=document.getElementById("mypara")
pclone = p.cloneNode(true);
p.parentNode.appendChild(pclone)

```

467.

[技巧题]

文档碎片

来自：前端开发面试易考知识点

参考：

```
function insertPs(){
    var aColors = [
        "red", "green", "blue", "magenta", "yellow", "chocolate", "black", "aquamarine", "lime", "fuchsia", "br
        "ass", "azure", "brown", "bronze", "deeppink", "aliceblue", "gray", "copper", "coral", "feldspar", "orange
        ", "orchid", "pink", "plum", "quartz", "purple"];
    var oFragment = document.createDocumentFragment(); //创建文档碎片
    for(var i=0;i<aColors.length;i++){
        var oP = document.createElement("p");
        var oText = document.createTextNode(aColors[i]);
        oP.appendChild(oText);
        oFragment.appendChild(oP); //将节点先添加到碎片中
    }
    document.body.appendChild(oFragment); //最后一次性添加到页面
}
```

468.

[技巧题]

innerHTML

来自：前端开发面试易考知识点

参考：

```
function myDOMInnerHTML(){
    var myDiv = document.getElementById("myTest");
    alert(myDiv.innerHTML); //直接显示 innerHTML 的内容
    //修改 innerHTML，可直接添加代码
    myDiv.innerHTML = "<img src='01.jpg' title='情人坡'>";
}
innerHTML 可同时显示没有的代码
```

469.

[问答题]

换皮肤

来自：前端开发面试易考知识点

参考：

```

<style type="text/css">
.myUL1{
    color:#0000FF;
    font-family:Arial;
    font-weight:bold;
}
.myUL2{
    color:#FF0000;
    font-family:Georgia, "Times New Roman", Times, serif;
}
</style>
<script language="javascript">
function check(){
    var oMy = document.getElementsByTagName("ul")[0];
    oMy.className =(oMy.className=="myUL1"? "myUL2":"myUL1"); //修改 CSS 类
}
</script>
</head>

<body>
<ul onclick="check()" class="myUL1">
    <li>HTML</li>
    <li>JavaScript</li>
    <li>CSS</li>
</ul>
</body>

```

470.

[技巧题]
动态添加行

来自：前端开发面试易考知识点

参考：

```

<script language="javascript">
window.onload=function(){
    var oTr = document.getElementById("member").insertRow(2); //插入一行
    var aText = new Array();
    aText[0] = document.createTextNode("fresheggs");
    aText[1] = document.createTextNode("W610");
    aText[2] = document.createTextNode("Nov 5th");
    aText[3] = document.createTextNode("Scorpio");
}

```

```
aText[4] = document.createTextNode("1038818");
for(var i=0;i<aText.length;i++){
    var oTd = oTr.insertCell(i);
    oTd.appendChild(aText[i]);
}
</script>
```

471.

[技巧题]
修改单元格内容

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
window.onload=function(){
    var oTable = document.getElementById("member");
    oTable.rows[3].cells[4].innerHTML = "lost";      //修改单元格内容
}
</script>
```

472.

[技巧题]
动态删除

来自：前端开发面试易考知识点

参考：

parentElement 是 IE dom,
parentNode 是标准 DOM

```
<script language="javascript">
window.onload=function(){
    var oTable = document.getElementById("member");
    oTable.deleteRow(2);      //删除一行，后面的行号自动补齐//指从 table 中的第 2
行开始进行删除
    oTable.rows[2].deleteCell(1); //删除一个单元格，后面的也自动补齐
}
</script>
```

```
<script language="javascript">
function myDelete(){
    var oTable = document.getElementById("member");
    //删除该行
    this.parentNode.parentNode.parentNode.removeChild(this.parentNode.parentNode);
}
window.onload=function(){
    var oTable = document.getElementById("member");
    var oTd;
    //动态添加 delete 链接
    for(var i=1;i<oTable.rows.length;i++){
        oTd = oTable.rows[i].insertCell(5);
        oTd.innerHTML = "<a href='#>delete</a>";
        oTd.firstChild.onclick = myDelete; //添加删除事件
    }
}
</script>
```

473.

[技巧题]

动态删除列

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
function deleteColumn(oTable,iNum){
    //自定义删除列函数，即每行删除相应单元格
    for(var i=0;i<oTable.rows.length;i++)
        oTable.rows[i].deleteCell(iNum);
}
window.onload=function(){
    var oTable = document.getElementById("member");
    deleteColumn(oTable,2);
}
</script>
```

474.

[技巧题]

控制 textarea 的字符个数

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
function LessThan(oTextArea){
    //返回文本框字符个数是否符号要求的 boolean 值
    return oTextArea.value.length < oTextArea.getAttribute("maxlength");
}
</script>
```

475.

[思考题]

BOM 模型

来自：前端开发面试易考知识点

参考：

浏览器对象模型，可以对浏览器窗口进行访问和操作，使用 BOM，开发者可以移动窗口，改变状态栏中的文本等与页面内容不相关的操作

Window 对象

这里可以用 `window.frames[0]` 或者用 `windows.frames[“topFrame”]` 来引用框架，也可以用 `top` 来代替 `window` 属性。`Top.frames[0]`。`window` 对象可以忽略

提供的方法有 `moveto,moveBy,resizeTo,resizeBy` 等方法。但尽量避免使用它们，因为会对用户浏览产生影响

Open 方法

除了 `open` 方法，还有 `alert,comfirm,prompt` 方法

状态栏

`Settimeout` 与 `setInterval`

`Settimeout`

下面的代码都是在 1 秒钟后显示一条警告

```
Settimeout("alert('aa'),1000");
```

```
Settimeout(function(){alert('aa')},1000);
```

如果要还未执行的暂停，可调用 `clearTimeOut()` 方法

```
Var si = Settimeout(function(){alert('aa')},1000);
```

```
clearTimeout(si);
```

`setInterval`

`History`

向后一页 `window.history.go(-1)` 等于 `history.back()`；

向前一页 `window.history.go(1)` 等于 `history.forward()`；

`Document`

`LastModified,title,URL` 属性都是比较常用

`Location` 对象

`Navigator` 对象

Screen 对象

476.

[思考题]

冒泡型事件

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
function add(sText){
    var oDiv = document.getElementById("display");
    oDiv.innerHTML += sText;    //输出点击顺序
}
</script>
</head>
<body onclick="add('body<br>');">
    <div onclick="add('div<br>');">
        <p onclick="add('p<br>');">Click Me</p>
    </div>
    <div id="display"></div>
</body>
```

先执行最里面的 p，再往外执行

477.

[思考题]

监听函数

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
window.onload = function(){
    var oP = document.getElementById("myP"); //找到对象
    oP.onclick = function(){                //设置事件监听函数
        alert('我被点击了');
    }
}
</script>
</head>
```

```
<body>
  <div>
    <p id="myP">Click Me</p>
  </div>
</body>
```

Function a(){}

oP.onclick = a

这样会先把 a 函数加载到缓存，不是最佳方案

Var A = Function(){}
oP.onclick = a

这样只有在 onclick 事件发生的时候，加载该函数

若以上的监听函数，在 onclick 的时候，需要执行多个函数，那就只能用以下的方法：

IE 标准：

```
<script language="javascript">
function fnClick(){
  alert("我被点击了");
  oP.detachEvent("onclick",fnClick);      //点击了一次后删除监听函数
}
var oP;//声明在函数外，这样就可以两个函数一起使用
window.onload = function(){
  oP = document.getElementById("myP");    //找到对象
  oP.attachEvent("onclick",fnClick);      //添加监听函数
}
</script>
</head>
```

```
<body>
  <div>
    <p id="myP">Click Me</p>
  </div>
</body>
```

也可以添加多个监听器

oP.attachEvent("onclick",fnClick1); //添加监听函数 1

oP.attachEvent("onclick",fnClick2); //添加监听函数 2

执行顺序为 fnClick2-> fnClick1

但是以上的监听器均为 IE 中的标准，而符合标准 DOM (firefox) 的监听器如下

oP.addEventListener("click",fnClick1,false); //添加监听函数 1

oP.addEventListener("click",fnClick2,false); //添加监听函数 2

因此这种方式在 Firefox 中支持，而在 IE 中不支持

为了兼容性，可这样写

```
if (el.addEventListener){
  el.addEventListener('click', KindDisableMenu, false);
```

```
    } else if (el.attachEvent){
        el.attachEvent('onclick', KindDisableMenu);
    }
```

第三个参数为 useCapture

而 useCapture 这个参数就是在控制这时候两个 click 事件的先后顺序。如果是 false，那就会使用 bubbling，他是从内而外的流程，所以会先执行蓝色元素的 click 事件再执行红色元素的 click 事件，如果是 true，那就是 capture，和 bubbling 相反是由外而内

478.

[思考题]

事件的类型 event.type

来自：前端开发面试易考知识点

参考：

IE 浏览器中事件对象是 window 对象的一个属性 event

```
Op.onclick=function(){ var o = window.event}
```

而标准 DOM 中规定 event 对象必须作为唯一的参数传给事件处理函数

```
Op.onclick=function(oevent){
```

```
}
```

因此为了兼容两种浏览器，通常采用下面的方法

```
Op.onclick=function(o){
```

```
If(window.event)//假如不等于空，则为 IE 浏览器
```

```
    O = window.event;
```

```
}
```

```
<script language="javascript">
function handle(oEvent){
    var oDiv = document.getElementById("display");
    if(window.event) oEvent = window.event;      //处理兼容性，获得事件对象
    if(oEvent.type == "click")                  //检测事件名称
        oDiv.innerHTML += "你点击了我 & ";
    else if( oEvent.type == "mouseover")
        oDiv.innerHTML += "你移动到我上方了 & ";
}
window.onload = function(){
    var oImg = document.getElementsByTagName("img")[0];
    oImg.onclick = handle;
    oImg.onmouseover = handle;
}
</script>
还有很多鼠标事件
window.onload = function(){
```

```
var oImg = document.getElementsByTagName("img")[0];
oImg.onmousedown = handle; //将鼠标事件除了 mousemove 外都监听
oImg.onmouseup = handle;
oImg.onmouseover = handle;
oImg.onmouseout = handle;
oImg.onclick = handle;
oImg.ondblclick = handle;
}
```

479.

[思考题]

事件的激活元素 event.srcElement 或者 target

来自：前端开发面试易考知识点

参考：

```
<script language="javascript">
function handle(oEvent){
    if(window.event) oEvent = window.event; //处理兼容性，获得事件对象
    var oTarget;
    if(oEvent.srcElement) //处理兼容性，获取事件目标
        oTarget = oEvent.srcElement; //IE 支持的写法
    else
        oTarget = oEvent.target; //Firefox 支持的写法
    alert(oTarget.tagName); //弹出目标的标记名称
}
window.onload = function(){
    var oImg = document.getElementsByTagName("img")[0];
    oImg.onclick = handle;
}
</script>
event.button
<script language="javascript">
function TestClick(oEvent){
    var oDiv = document.getElementById("display");
    if(window.event)
        oEvent = window.event;
    oDiv.innerHTML += oEvent.button; //输出 button 的值
}
document.onmousedown = TestClick;
window.onload = TestClick; //测试未按下任何键
</script>
</head>
```

```
<body>
<div id="display"></div>
</body>
```

在 IE/Opera 中，是 window.event，而在 Firefox 中，是 event
而事件的对象，在 IE 中是 window.event.srcElement，在 Firefox 中是 event.target，而在 Opera 中则两者都支持。

在 IE 里面

左键是 window.event.button = 1
右键是 window.event.button = 2
中键是 window.event.button = 4
没有按键动作的时候 window.event.button = 0

在 Firefox 里面

左键是 event.button = 0
右键是 event.button = 2
中键是 event.button = 1
没有按键动作的时候 event.button = 0

在 Opera 7.23/7.54 里面

鼠标左键是 window.event.button = 1
没有按键动作的时候 window.event.button = 1
右键和中键无法获取
键盘事件

```
window.onload = function(){
    var oTextArea = document.getElementsByTagName("textarea")[0];
    oTextArea.onkeydown = handle; //监听所有键盘事件
    oTextArea.onkeyup = handle;
    oTextArea.onkeypress = handle;
}
```

e.keyCode;

onkeypress 是在用户按下并放开任何字母数字键时发生。系统按钮（例如，箭头键和功能键, Shift、Ctrl、Alt、F1、F2）无法得到识别。

onkeyup 是在用户放开任何先前按下的键盘键时发生。

onkeydown 是在用户按下任何键盘键（包括系统按钮，如箭头键和功能键）时发生
屏蔽鼠标右键

第一种方式：

```
<script language="javascript">
function block(oEvent){
    if(window.event)
        oEvent = window.event;
    if(oEvent.button == 2)
        alert("鼠标右键不可用");
}
document.onmousedown = block;
```

```
</script>
第二种方式:
<script language="javascript">
function block(oEvent){
    if(window.event){
        oEvent = window.event;
        oEvent.returnValue = false; //取消默认事件 支持 IE
    }else
        oEvent.preventDefault(); //取消默认事件 支持 Firefox
}
document.oncontextmenu = block;
</script>
```

480.

[技巧题]

伸缩的菜单

来自：前端开发面试易考题

参考：

```
<script language="javascript">
function change(){
    //通过父元素 li，找到兄弟元素 ul
    var oSecondDiv = this.parentNode.getElementsByTagName("ul")[0];//这里的 this 就是
下面的 OA
    //CSS 交替更换来实现显、隐
    if(oSecondDiv.className == "myHide")
        oSecondDiv.className = "myShow";
    else
        oSecondDiv.className = "myHide";
}
window.onload = function(){
    var oUl = document.getElementById("listUL");
    var aLi = oUl.childNodes; //子元素
    var oA;
    for(var i=0;i<aLi.length;i++){
        //如果子元素为 li，且这个 li 有子菜单 ul
        if(aLi[i].tagName == "LI" && aLi[i].getElementsByName("ul").length){
            oA = aLi[i].firstChild; //找到超链接
            oA.onclick = change; //动态添加点击函数
        }
    }
}
```

```
</script>
```

481.

[思考题]

Event DOM 常用属性

来自：前端开发面试易考知识点

提示：

tagName
nodeValue
nodeName
nodeType
parentNode
childNodes
firstChild
lastChild
nextSibling (IE)
previousSibling (IE)
attributes
innerHTML
style
className

方法

getElementById
getElementsByName
getElementsByTagName
hasChildNodes()
getAttribute
setAttribute
createElement
createTextNode
appendChild
removeChild
replaceChild
insertBefore
cloneNode
createDocumentFragment
detachEvent
attachEvent (IE) addEventListener(DOM)

event 属性

```
type  
keyCode  
srcElement(IE) target(DOM)  
button
```

鼠标事件

```
onclick  
onmouseover  
onmousedown  
onmouseup  
onmouseout  
ondblclick
```

键盘事件

```
onkeydown  
onkeyup  
onkeypress
```

window 事件

```
onload
```

document 事件

```
oncontextmenu  
write  
onmousedown
```

482.

[思考题]

错误和异常

来自：前端开发面试易考知识点

参考：

拼写错误、访问不存在的变量，括号不匹配，等号与赋值

声明变量时，要记住局部变量和全局变量的区别

```
Function square(num){  
    Total = num*num;  
    Return total;  
}  
Var total = 50;
```

```
Var number = Square(20);
Alert(total);
这些代码将不可避免地导致全局变量 total 的值发生变化。
Function square(num){
    Var Total = num*num;
    Return total;
}
```

483.

[思考题]

错误处理

来自：前端开发面试易考知识点

参考：

```
onerror
<head>
<title>onerror</title>
<script language="javascript">
window.onerror = function(){
    alert("出错啦！");
    return true; //屏蔽系统事件
}
</script>
</head>
<body onload="nonExistent()">
</body>
```

```
Try..catch
<script language="javascript">
try{
    alert("this is an example");
    alert(fresheggs);
} catch(exception){
    var sError = "";
    for(var i in exception)
        sError += i + ":" + exception[i] + "\n";
    alert(sError);
}
</script>
```

484.

[思考题]

调试器

来自：前端开发面试易考知识点

参考：

IE-工具-Internet 选项-高级->禁用调试，显示脚本错误

Firefox 错误控制台

Microsoft script debugger

Venkman firefox 的插件

485.

[思考题]

JavaScript 优化

来自：前端开发面试易考知识点

参考：

1. 提高 JavaScript 下载时间。将 JavaScript 写到同一行

2. 尽量使用内置函数（因为内置函数是通过 C 语言编译到浏览器中的）

. 实例

1 图片查看器

```
<html>
<head>
    <title></title>
    <script>
```

```
        function showPic(obj){
            var h = obj.getAttribute("href");
            document.getElementById("image").setAttribute("src",h);
        }
    </script>
</head>
<body>
<h1>Snaaphots</h1>
<ul>
    <li><a href="photo/01.jpg" title="a" onclick="showPic(this);return false;">01</a></li>
    <li><a href="photo/02.jpg" title="b" onclick="showPic(this);return false;">02</a></li>
    <li><a href="photo/03.jpg" title="c" onclick="showPic(this);return false;">03</a></li>
    <li><a href="photo/04.jpg" title="d" onclick="showPic(this);return false;">04</a></li>
```

```
<li><a href="photo/05.jpg" title="e" onclick="showPic(this);return false;">05</a></li>
</ul>
</img>
</body>
</html>

Return false 指的是把默认的 noclick 事件取消
给其加上 css
<style>
    body{
        color:#333;
        background:#ccc;
        margin:1em 10%;
    }
    a{
        text-decoration:none;
        padding:10px;
        color:#c60;
    }
    a:link, a:visited{
        color: #A62020; background-color: #ecd8db; text-decoration: none; padding:4px
        10px 4px 10px;
        border-top:1px solid #EEE;
        border-left:1px solid #EEE;
        border-bottom:1px solid #717171;
        border-right:1px solid #717171;
    }

    a:hover{
        color:#821818; background-color:#e2c4c9; padding:5px 8px 3px 12px;
        border-top:1px solid #717171;
        border-left:1px solid #717171;
        border-bottom:1px solid #EEE;
        border-right:1px solid #EEE;}
    ul{
        margin:0px;
        padding:0px;
    }
    li{
        list-style-type:none;
        display:inline;
    }
    img{
        margin:10px 0px;
    }
```

```
</style>
```

```
当时现在有个缺陷，就是 onclick 的事件直接写在了 HTML 上，分离  
先给 ul 加上个属性 id<ul id=" img_ul " >  
window.onload = prepareGalley;  
function prepareGalley(){  
    var img_ul = document.getElementById("img_ul");  
    var links = img_ul.getElementsByTagName("a");  
  
    for(var i=0;i<links.length;i++){  
        links[i].onclick = function(){  
            showPic(this);  
            return false;  
        }  
    }  
}
```

有一个问题，如果 onload 的函数有多个怎么办？

```
window.onload = prepareGalley1;  
window.onload = prepareGalley2;
```

显然，这样第一个函数就会被第二个函数覆盖。

可以这样写

```
Window.onload = function(){  
    prepareGalley1();  
    prepareGalley2();  
}
```

还有一个比这个更 NB 的写法，由 Simon Willison 写的

```
function addLoadEvent(func){  
    var oldonload = window.onload;  
    if(typeof window.onload !='function'){  
        window.onload = func;  
    }else{  
        window.onload = function(){  
            oldonload();  
            func();  
        }  
    }  
}
```

```
addLoadEvent(prepareGalley1);  
addLoadEvent(prepareGalley2);
```

486.

[问答题]

编写一个方法 求一个字符串的字节长度

来自：前端开发面试易考题

参考：

```
new function(s){  
    if(!arguments.length||!s) return null;  
    if("")==s) return 0;  
    var l=0;  
    for(var i=0;i<s.length;i++){  
        if(s.charCodeAt(i)>255) l+=2; else l++;  
    }  
    alert(l);  
}("hello world!");
```

487.

[问答题]

如何控制 alert 中的换行

来自：前端开发面试易考题

参考：

```
alert("hello\nworld");
```

488.

[问答题]

解释 document.getElementById("ElementID").style.fontSize="1.5em"

来自：前端开发面试易考题

参考：

设置 id 为 ElementID 的元素的字体大小为 1.5 个相对单位

Em 为相对长度单位。相对于当前对象内文本的字体尺寸。

如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。

1em=16px

489.

[问答题]

按照格式 xxxx 年 xx 月 xx 日 xx 时 xx 分 xx 秒动态显示时间 要求不足 10 的补 0

来自：前端开发面试易考题

参考：

```
<script type="text/javascript" language="javascript">
function tick(){
    var d=new Date();
    var t=function(a){return a<10?"0"+a:a;}
    Clock.innerHTML=d.getFullYear()+" 年 "+t(d.getMonth()+1)+" 月 "+t(d.getDate())+" 日 "+t(d.getHours())+" 时 "
        +t(d.getMinutes())+" 分 "+t(d.getSeconds())+" 秒";
    window.setTimeout("tick()",1000);
}
window.onload=tick;
</script>

<body>
<div id="Clock"></div>
</body>
```

490.

[问答题]

编写一个方法 去掉一个数组的重复元素

来自：前端开发面试易考题

参考：

```
Array.prototype.strip=function(){
    if(this.length<2) return [this[0]]||[];
    var arr=[];
    for(var i=0;i<this.length;i++){
        arr.push(this.splice(i--,1));//将本数组中第一个元素取出放入到数组 arr 中
        for(var j=0;j<this.length;j++){
            if(this[j]==arr[arr.length-1]){
                this.splice(j--,1); //删除本数组中与数组 arr 中最后一个元素相同的元素
            }
        }
    }
    return arr;
}

var arr=["abc",85,"abc",85,8,8,1,2,5,4,7,8];
```

```
alert(arr.strip());
```

491.

[问答题]

说出 3 条以上 ff 和 ie 的脚本兼容问题

来自：前端开发面试易考题

参考：

IE 有 children, FF 没有；

IE 有 parentElement, FF 没有；

IE 有 innerText,outerText,outerHTML, FF 没有；

IE 有数据岛，FF 没有；

FF 有

HTMLElement,HTMLDivElement,XMLDocument,DocumentFragment,Node,Event,Element

等等，IE 没有；

IE 跟 FF 创建 HttpRequest 实例的方法不一样

492.

[问答题]

DIV 中 border、margin 和 padding 的区别和用法

来自：前端开发面试易考题

参考：

边框属性(border)用来设定一个元素的边线

外边距属性(margin)是用来设置一个元素所占空间的边缘到相邻元素之间的距离

内边距属性(padding)是用来设置元素内容到元素边界的距离

493.

[问答题]

为 Array 写一个 indexOf 方法

来自：前端开发面试易考题

参考：

```
Array.prototype.indexOf = function(e){  
    for(var i=0,j; j=this[i]; i++){  
        if(j==e){return i;}  
    }  
}
```

```

        }
        return -1;
    }

Array.prototype.lastIndexOf = function(e){
    for(var i=this.length-1,j; j=this[i]; i--){
        if(j==e){return i;}
    }
    return -1;
}

var arr=[1,2,3,4,5];
alert(arr.indexOf(5));

```

494.

[问答题]

说说元素克隆

来自：前端开发面试易考题

参考：

浅复制(影子克隆):只复制对象的基本类型,对象类型,仍属于原来的引用

深复制(深度克隆):不紧复制对象的基本类,同时也复制原对象中的对象.就是说完全是新对象产生的

```

Object.prototype.Clone = function(){
    var objClone;
    if( this.constructor == Object )
        objClone = new this.constructor();
    else
        objClone = new this.constructor(this.valueOf());
    for ( var key in this ){
        if ( objClone[key] != this[key] ){
            if ( typeof(this[key]) == 'object' ){
                objClone[key] = this[key].Clone();
            }else{
                objClone[key] = this[key];
            }
        }
    }
    objClone.toString = this.toString;
    objClone.valueOf = this.valueOf;
    return objClone;
}

```

495.

[问答题]

兼容 IE 和 FF 的换行 CSS 推荐样式

来自：前端开发面试易考题

参考：

`word-wrap:break-word; overflow:hidden;`

`word-wrap` 是控制换行的。使用 `break-word` 时，是将强制换行。中文没有任何问题，英文语句也没问题。但是对于长串的英文，就不起作用。

`word-break` 是控制是否断词的。

`normal` 是默认情况，英文单词不被拆开。

`break-all`，是断开单词。在单词到边界时，下个字母自动到下一行。主要解决了长串英文的问题。

`keep-all`，是指 Chinese, Japanese, and Korean 不断词。即只用此时，不用 `word-wrap`，中文就不会换行了。（英文语句正常。）

496.

[问答题]

手型 Cursor 的兼容 IE 和 FF 写法

来自：前端开发面试易考题

参考：

`cursor:pointer`

497.

[问答题]

元素的 alt 和 title 有什么异同？

来自：前端开发面试易考题

参考：

`alt` 作为图片的替换文字出现，`title` 是图片的解释文字

图片存在

只有 `alt` 图片的解释文字

只有 `title` 图片的解释文字

两者都有 图片的解释文字

两者都没有 图片既没有替换文字，也没有解释文字

图片不存在
只有 alt 图片既有替换文字，又有解释文字
只有 title 图片没有替换文字，只有解释文字
两者都有 图片既有替换文字，又有解释文字
两者都没有 图片既没有替换文字，也没有解释文字
当然不同的浏览器处理方式也会不一样
border-color-left、marin-left、-moz-viewport 改写成 JavaScript 格式
border-color-left:borderLeftColor
marin-left:marinLeft
-moz-viewport:MozViewport

498.

[问答题]

用 css、html 编写一个两列布局的网页，要求右侧宽度为 200px，左侧自动扩展。

来自：前端开发面试易考题
参考：
CSS:

```
#right{
    float:right;
    width:200px;
}
#left{
    margin-right:200px;
}
```

HTML:

```
<body>
    <div id="right">...</div>
    <div id="left">...</div>
</body>
```

499.

[问答题]

如何提高网页的运行速度

来自：前端开发面试易考知识点
参考：
内容与形式分离，模块化开发，优化 CSS
减少页面文档大小

尽量减少图片的使用或注意图片的大小，优化图片：格式、质量、图片长宽标志
减少响应的次数，用 Ajax
网址后面加一个 “/”

500.

[问答题]

按要求写一个简单的 ajax 示例

来自：前端开发面试易考题

参考：

```
<body>
    <div id="load">数据正在加载.....</div>
    <script type="text/javascript">
        var Browser={/**Browser 对象用于检测浏览器，其中用到了 IE 的条件编译*/
            isFF:window.navigator.appName.toUpperCase().indexOf("NETSCAPE")!=-1?true:false,
            isOpera:window.navigator.appName.toUpperCase().indexOf("OPERA")!=-1?true:false
        };

        Function.prototype.bind=function(object){
            var _this=this;
            return function(){
                _this.apply(object,arguments);
            }
        }

        function HttpRequest(){
            this.async=true;
            this.cache=false;
            this.xmlhttp=function(){
                if(Browser.isFF&&window.XMLHttpRequest){
                    try{
                        return new XMLHttpRequest();
                    }catch(e){}
                }else if(Browser.isIE&&window.ActiveXObject){
                    var Version = =
                    ["Msxml2.XMLHTTP.6.0","Msxml2.XMLHTTP.5.0","Msxml2.XMLHTTP.4.0","Msxml2.XMLHTTP.3.0"
                    ,+
                    "Msxml2.XMLHTTP.2.6","Msxml2.XMLHTTP","Microsoft.XMLHTTP.1.0","Microsoft.XMLHTTP.1",
                    "Microsoft.XMLHTTP"];
                }
            }
        }
    </script>

```

```

        for(var i=0;i<Version.length;i++){
            try{
                return new ActiveXObject(Version[i]);
            }catch(e){}
        }
    }()
    | | false;
}HttpRequest.prototype={
    send:function(object,url,callback){
        if(!this.xmlhttp) return;
        this.xmlhttp.open(object?"post":"get",url,!this.async);
        if(object)
            this.xmlhttp.setRequestHeader("content-type","application/x-www-form-urlencoded");

        if(!this.cache){
            this.xmlhttp.setRequestHeader("No-Cache","1");
            this.xmlhttp.setRequestHeader("Pragma","no-cache");

            this.xmlhttp.setRequestHeader("Cache-Control","no-cache");
            this.xmlhttp.setRequestHeader("Expire","0");
            this.xmlhttp.setRequestHeader("Last-Modified","Wed, 1 Jan 1997
00:00:00 GMT");
            this.xmlhttp.setRequestHeader("If-Modified-Since","-1");
        }
        if(!this.callback) this.callback=callback;
        if(!this.async){
            if(typeof(this.callback)=="string"){
                eval(this.callback);
            }else if(typeof(this.callback)=="function"){
                this.callback(this.xmlhttp);
            }
        }else{
            this.xmlhttp.onreadystatechange=function(){
                if(this.xmlhttp.readyState==4){
                    if(this.xmlhttp.status==0| |this.xmlhttp.status==200){

                        if(typeof(this.callback)=="string"){

                            eval(this.callback);
                        }else if(typeof(this.callback)=="function"){

                            this.callback(this.xmlhttp);
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}.bind(this);
}
this.xmlhttp.send(object);
},abort:function(){
    if(this.xmlhttp&&this.xmlhttp.abort) this.xmlhttp.abort();
}
}; //ajax 类定义结束
new HttpRequest().send(null,"http://bbs.51js.com/index.php",function(r){

document.getElementById("load").innerHTML=r.responseText.match(/<img.*?(?:\/>)/img).join("");
}); });
</script>
</body>

```

501.

[思考题]

IE6、IE7、IE8、Firefox 兼容性 CSS HACK

来自：前端开发面试易考知识点

参考：

整理关于 IE6、IE7、IE8、Firefox 兼容性 CSS HACK 问题

1. 区别 IE 和非 IE 浏览器 CSS HACK 代码

```
#divcss5{
background:blue; /*非 IE 背景蓝色*/
background:red \9; /*IE6、IE7、IE8 背景红色*/
}
```

2. 区别 IE6,IE7,IE8,FF CSS HACK

【区别符号】：「\9」、「*」、「_」

【示例】：

```
#divcss5{
background:blue; /*Firefox 背景变蓝色*/
background:red \9; /*IE8 背景变红色*/
*background:black; /*IE7 背景变黑色*/
_background:orange; /*IE6 背景变橘色*/
}
```

【说明】：因为 IE 系列浏览器可读「\9」，而 IE6 和 IE7 可读「*」(米字号)，另外 IE6 可辨识「_」(底线)，因此可以依照顺序写下来，就会让浏览器正确的读取到自己看得懂得 CSS 语法，所以就可以有效区分 IE 各版本和非 IE 浏览器(像是 Firefox、Opera、Google Chrome、Safari 等)。

3. 区别 IE6、IE7、Firefox (EXP 1)

【区别符号】: 「*」、「_」

【示例】:

```
#divcss5{  
background:blue; /*Firefox 背景变蓝色*/  
*background:black; /*IE7 背景变黑色*/  
_background:orange; /*IE6 背景变橘色*/  
}
```

【说明】: IE7 和 IE6 可读「*」(米字号), IE6 又可以读「_」(底线), 但是 IE7 却无法读取「_」, 至于 Firefox(非 IE 浏览器)则完全无法辨识「*」和「_」, 因此就可以透过这样的差异性来区分 IE6、IE7、Firefox

4. 区别 IE6、IE7、Firefox (EXP 2)

【区别符号】: 「*」、「!important」

【示例】:

```
#divcss5{  
background:blue; /*Firefox 背景变蓝色*/  
*background:green !important; /*IE7 背景变绿色*/  
*background:orange; /*IE6 背景变橘色*/  
}
```

【说明】: IE7 可以辨识「*」和「!important」, 但是 IE6 只可以辨识「*」, 却无法辨识「!important」, 至于 Firefox 可以读取「!important」但不能辨识「*」因此可以透过这样的差异来有效区隔 IE6、IE7、Firefox。

5. 区别 IE7、Firefox

【区别符号】: 「*」、「!important」

【示例】:

```
#divcss5{  
background:blue; /*Firefox 背景变蓝色*/  
*background:green !important; /*IE7 背景变绿色*/  
}
```

【说明】: 因为 Firefox 可以辨识「!important」但却无法辨识「*」, 而 IE7 则可以同时看懂「*」、「!important」, 因此可以两个辨识符号来区隔 IE7 和 Firefox。

6. 区别 IE6、IE7 (EXP 1)

【区别符号】: 「*」、「_」

【示例】:

```
#tip {  
*background:black; /*IE7 背景变黑色*/  
_background:orange; /*IE6 背景变橘色*/  
}
```

【说明】: IE7 和 IE6 都可以辨识「*」(米字号), 但 IE6 可以辨识「_」(底线), IE7 却无法辨识, 透过 IE7 无法读取「_」的特性就能轻松区隔 IE6 和 IE7 之间的差异。

7. 区别 IE6、IE7 (EXP 2)

【区别符号】: 「!important」

【示例】:

```
#divcss5{  
background:black !important; /*IE7 背景变黑色*/  
background:orange; /*IE6 背景变橘色*/  
}
```

【说明】: 因为 IE7 可读取「!important;」但 IE6 却不行，而 CSS 的读取步骤是从上到下，因此 IE6 读取时因无法辨识「!important」而直接跳到下一行读取 CSS，所以背景色会呈现橘色。

8. 区别 IE6、Firefox

【区别符号】: 「_」

【示例】:

```
#divcss5{  
background:black; /*Firefox 背景变黑色*/  
  
_background:orange; /*IE6 背景变橘色*/  
}
```

【说明】: 因为 IE6 可以辨识「_」(底线)，但是 Firefox 却不行，因此可以透过这样的差异来区隔 Firefox 和 IE6，有效达成 CSS hack。

以上包括了 IE6\IE8\IE7\火狐浏览器兼容问题及解决方法。

502.

[单选题]

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 = / /;

来自：JavaScript 基本功检测

参考：A

503.

[单选题]

以下哪个单词不属于 javascript 保留字: ()

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

来自：JavaScript 基本功检测
参考：B

504.

[单选题]

请选择结果为真的表达式：()

- A. null instanceof Object
- B. null === undefined
- C. null == undefined
- D. NaN == NaN

来自：JavaScript 基本功检测
参考：C

505.

[不定项选择题]

请选择对 javascript 理解有误的：()

- A. JScript 是 javascript 的简称
- B. javascript 是网景公司开发的一种 Java 脚本语言，其目的是为了简化 Java 的开发难度
- C. FireFox 和 IE 存在大量兼容性问题的主要原因在于他们对 javascript 的支持不同上
- D. AJAX 技术一定要使用 javascript 技术

来自：JavaScript 基本功检测
参考：ABCD

506.

[不定项选择题]

foo 对象有 att 属性，那么获取 att 属性的值，以下哪些做法是可以的：()

- A. foo.att
- B. foo("att")
- C. foo["att"]
- D. foo{"att"}
- E. foo["a"+ "t" + "t"]

来自：JavaScript 基本功检测
参考：ACE

507.

[判断题]
continue 语句只能用在循环语句和 switch 语句中。

来自：JavaScript 基本功检测
参考：无

508.

[判断题]
函数是一个独立的程序模块，当页面被浏览时会自动执行其中包含的程序语句。

来自：JavaScript 基本功检测
参考：无

509.

[判断题]
表单元素的 name 属性用来标识表单元素的类型。

来自：JavaScript 基本功检测
参考：无

510.

[填空题]
JavaScript 单行注释用_____表示，多行注释用_____表示。

来自：JavaScript 基本功检测
参考：
// /* */

511.

[填空题]

JavaScript 中的对象由_____和_____两个基本元素构成。

来自：JavaScript 基本功检测

参考：

Object Function

512.

[填空题]

JavaScript 中布尔常量只有两种状态：_____或者_____。

来自：JavaScript 基本功检测

参考：

True False

513.

[问答题]

什么数据类型仅有一个值？什么数据类型只有两个值？

来自：JavaScript 基本功检测

参考：无

514.

[问答题]

```
var a = 1.5,b;  
b=parseInt(a);
```

b 的值为？

来自：JavaScript 基本功检测

参考：

1.

515.

[问答题]

```
var a="10",d=2;  
document.write(a>2);  
document.write(a+2);  
输出结果?
```

来自：JavaScript 基本功检测

参考： 无

516.

[问答题]

求 y 和 z 的值是多少？

```
var x = 1;  
var y = 0;  
var z = 0;  
function add(n){n=n+1;}  
y = add(x);  
function add(n){n=n+3;}  
z = add(x);
```

来自：JavaScript 基本功检测

参考：

都为 undefined，因为没有返回值。

517.

[问答题]

求 y 和 z 的值是多少？

```
var x = 1;  
var y = 0;  
var z = 0;  
function add(n){return n+1;}  
y = add(x);  
function add(n){ return n+3;}  
z = add(x);
```

来自：JavaScript 基本功检测

参考：

都为 45。

518.

[问答题]

javascript 是面向对象的，怎么体现 javascript 的继承关系？

来自：JavaScript 基本功检测

参考：

使用 `prototype` 来实现。

519.

[问答题]

javascript 怎样选中一个 checkbox，怎样设置它无效？

来自：JavaScript 基本功检测

参考：

`document.all.cb1[0].disabled = true;`

520.

[问答题]

form 中的 input 可以设置为 `readonly` 和 `disabled`，请问 2 者有什么区别？

来自：JavaScript 基本功检测

参考：

`readonly` 不可编辑，但可以选择和复制；值可以传递到后台

`disabled` 不能编辑，不能复制，不能选择；值不可以传递到后台

521.

[问答题]

js 中的 3 种弹出式消息提醒（警告窗口，确认窗口，信息输入窗口）的命令式什么？

来自：JavaScript 基本功检测

参考：

alert

confirm

prompt

522.

[问答题]

form 中的 input 有哪些类型？

来自：JavaScript 基本功检测

参考：无

523.

[问答题]

javaScript 的 2 种变量范围有什么不同？

来自：JavaScript 基本功检测

参考：

全局变量：当前页面内有效

局部变量：方法内有效

524.

[问答题]

列举 javaScript 的 3 种主要数据类型，2 种复合数据类型和 2 种特殊数据类型。

来自：JavaScript 基本功检测

参考：

主要数据类型：string, boolean, number

复合数据类型：function, object

525.

[问答题]

程序中捕获异常的方法？

来自：JavaScript 基本功检测

参考：

window.error

try{}catch(){}}finally{}

526.

[问答题]

写出函数 DateDemo 的返回结果，系统时间假定为今天

```
function DateDemo(){
    var d, s="今天日期是: ";
    d = new Date();
    s += d.getMonth() + "/";
    s += d.getDate() + "/";
    s += d.getFullYear();
    return s;
}
```

来自：JavaScript 基本功检测

参考：

结果：今天日期是：8/08/2008

527.

[问答题]

写出程序运行的结果？

```
for(i=0, j=0; i<10, j<6; i++, j++){
    k = i + j;
}
alert(k)
```

来自：JavaScript 基本功检测

参考：

10（小心陷阱）

528.

[问答题]

运行的结果？

```
function hi() {
```

```
    var a;  
    alert(a);  
}  
hi()
```

来自：JavaScript 基本功检测

参考：

undefined

529.

[问答题]

运行的结果？

```
function hi() {  
    var a = null;  
    alert(a);  
}  
hi()
```

来自：JavaScript 基本功检测

参考：

null

530.

[问答题]

浏览器的对象模型？

来自：JavaScript 基本功检测

参考：

window

顶级对象

```
window.alert(msg)  
window.prompt()  
window.confirm()  
if(window.confirm()){  
...  
}  
window.open()  
window.close()
```

`document`
`document.write()`

`history`

当用户浏览网页时，浏览器保存了一个最近所访问网页的 `url` 列表。这个列表就是用 `history` 对象表示。

`history.back():`后退

`history.forward():`前进

`history.go(n):`正数表示向前，负数表示向后

`location`

表示当前打开的窗口或框架的 URL 信息。

`location.href:` 重定向

等价于 `location.assign(url)`

`location.host:` 类似 `www.163.com:80`

`navigator`

表示浏览器的信息及 js 运行的环境

`navigator.cookieEnabled:` 该属性表示是否启用 cookie

`screen`

用于显示网页的显示器的大小和颜色

`screen.width/screen.height:` 表示显示器的分辨率（总的宽度，高度）

531.

[问答题]

`XMLHttpRequest` 对象是什么？

来自：JavaScript 基本功检测

参考：

Ajax 原理

532.

[问答题]

超链接的属性 `target` 的可选值：`_blank`, `_parent`, `_self`, `_top` 和框架名称有什么区别？

来自：JavaScript 基本功检测

参考：无

533.

[问答题]

javascript 的常用对象有哪些？

来自：JavaScript 基本功检测

参考：

String, Math, Date 和 Array 对象

534.

[问答题]

innerHTML, innerText, outerHTML, innerText 的区别？

来自：JavaScript 基本功检测

参考：无

535.

[问答题]

```
var i,m,n;  
i=18;  
m=++i;  
document.write(m+n+i);
```

程序的运行结果为_____

来自：JavaScript 基本功检测

参考：

NaN (n 没定义)

536.

[问答题]

```
switch(n){  
    case "a": document.write("85-100");break;  
    case "b": document.write("70-84");break;  
    case "c": document.write("60-69");break;  
    case "d": document.write("<60");break;  
    default: document.write("error");  
}
```

程序运行，当 n 为 d 时，输出结果是_____

来自：JavaScript 基本功检测
参考：无

537.

[问答题]

```
var x,y;
y=1;
x=y++;
if(x>=0)
    if(x>0)
        y=1;
    else
        y=-1;
else
    y=1;
document.write(y);
```

运行结果为_____

来自：JavaScript 基本功检测
参考：
1

538.

[问答题]

```
var reg1=/fa*/;
var str1="";
var str2="";
var res1=reg1.exec(str1);
var res2=reg1.exec(str2);
document.write("str1 中的匹配字符串为: "+res1+"<br>");
document.write("str2 中的匹配字符串为: "+res2+"<br>");
```

运行结果为_____

来自：JavaScript 基本功检测
参考：
null

539.

[问答题]

如果 sum 最终的值为 39， 请完善以下程序

```
var x=      ;
var y=15;
function inc(n){
    var x=n+1;
    var y=++x;
    return(y);
}
var sum=inc(x)+y;
alert(sum);
```

来自： JavaScript 基本功检测

参考：

22

540.

[问答题]

请为以下语句加上注释

```
function calar(){
    return this.width*this.height;
}
function rec(w,h){
    this.width=w;
    this.height=h;
    this.ar=calar;
}
var recA=new rec(10,20);
```

来自： JavaScript 基本功检测

参考： 无

541.

[问答题]

分别用 while 语句和 for 语句编写求 $1+2+\dots+100$ 的程序？

来自：JavaScript 基本功检测

参考：

```
//while
var sum = 0, i = 1;
while(i <= 100){
    sum += i++;
}
document.write('1+2+3+...+100=' + sum); //5050
var sum = 0, i = 1;
while(i <= 100){
    sum = sum + i;
    i++;
}
//for
var sum = 0;
for(i = 1; i <= 100; i++){
    sum += i;
}
document.write('1+2+3+...+100=' + sum); //5050
var sum = 0;
for(i=1;i<=100;i++)
sum = sum + i;
```

542.

[问答题]

将计算阶乘的程序写成一个函数，当用户在表单 formjc 的文本域 txtshr 中输入某个整数并单击“计算阶乘”超链接，JavaScript 主程序就会调用该函数求出该整数的阶乘值。请编写函数及主程序语句？

来自：JavaScript 基本功检测

参考：

```
var txtshr ={value:"3"};
function func(){
    var end = 1 *txtshr.value;
    var result = 1;
    for(i=1;i<=end;i++)
        result *= i;
    return result;
}
alert(func());
```

543.

[问答题]

写出通过动态网页访问 Web 数据库的运行过程。

来自：JavaScript 基本功检测

参考：

用户 web 浏览器请求动态网页

web 服务器查找该网页并将其传送给应用程序服务器

应用程序服务器查找该网页中的指令

应用程序服务器将查询指令发送到数据库驱动程序

数据库驱动程序对数据库执行查询

记录集被返回给数据库驱动程序

数据库驱动程序将记录集传递给应用程序服务器

应用程序服务器将数据插入网页中，然后将该网页传递给 web 服务器

web 服务器将完成的网页发送到用户 web 浏览器

544.

[单选题]

以下哪条语句会产生运行错误：()

A.var obj = ()//语法错误

B.var obj = [];//创建数组

C.var obj = {};//创建对象

D.var obj = //;

来自：JavaScript 基本功检测

参考：a

原因：var obj = new Array ();是对的；JavaScript 中大括号表示创建对象。var obj = { id:1, name:"jacky" };alert(obj.name);上例表示创建一个具有属性 id（值为 1）、属性 name（值为 jacky）的对象。属性名称可以用引号引起来成 "id"、"name"，也可以不引。

当然除了属性，也可以创建方法。

试验代码

```
/* window.onload=function()
{
// var obj = ();
var obj1 = [];//object
var obj2 = {};//object
var obj3 = //;//undefined
alert(typeof(obj1));
alert(typeof(obj2));
```

```
    alert(typeof(obj3));
  }*/
  function showName()
  {
    alert(this.name);
  }
  var obj = { id:1, name:"jacky", showName:showName };
  obj.showName();
```

545.

[单选题]

以下哪个单词不属于 javascript 保留字: ()

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

来自：JavaScript 基本功检测

参考： b

以下的保留字不可以用作变量,函数名,对象名等,其中有的保留字是为以后 JAVASCRIPT 扩展用的.

546.

[单选题]

请选择结果为真的表达式: ()

- A.null instanceof Object (if(!(null instanceof Object)))
- B.null === undefined
- C.null == undefined
- D.NaN == NaN

来自：JavaScript 基本功检测

参考： c

(1) null 确实可以理解为原始类型, 不能当 Object 理解!

null,int,float.....等这些用关键字表示的类型,都不属于 Object.

至于可以把 null 作为参数,只是特殊规定而已.

可以这么理解:

对象的引用代表的是一个内存的值,null 是一个空引用,可以理解为内存的值为 0;按这个意思对代码

(2) function f1(){

```

}

1. alert(f1 instanceof Function);//true
2. alert(f1 instanceof Object);//true
3. alert(Function instanceof Object);//true
4. alert(Object instanceof Function);//true

Function 是 Object 的实例, Object 又是 Function 的实例
Function 是函数的构造函数, 而 Object 也是函数, Function 自身也是函数
Object.prototype 是一切原型链的顶点, instanceof 会查找整个原型链
alert(Function);
alert(Function.prototype);
alert(Function.__proto__);
alert(Object);
alert(Object.prototype);
alert(Object.__proto__);
alert((function(){}).prototype);
alert((function(){}).__proto__);
alert((function(){}).__proto__.prototype);
alert((function(){}).prototype.__proto__);
alert(Array.__proto__);
alert((123).__proto__);
alert((Number).__proto__);
alert(("test").__proto__);
alert((String).__proto__);
alert((true).__proto__);
alert((Boolean).__proto__);
/* window.onload=function()
{
if(NaN == NaN)
{
alert("ddd");
}
}
*/

```

547.

[不定项选择题]

请选择对 javascript 理解有误的: ()

- A.JScript 是 javascript 的简称
 - B.javascript 是网景公司开发的一种 Java 脚本语言, 其目的是为了简化 Java 的开发难度
 - C.FireFox 和 IE 存在大量兼容性问题的主要原因在于他们对 javascript 的支持不同上
 - D.AJAX 技术一定要使用 javascript 技术
-

来自：JavaScript 基本功检测

参考：abcd

548.

[不定项选择题]

foo 对象有 att 属性，那么获取 att 属性的值，以下哪些做法是可以的：()

- A.foo.att
- B.foo("att")
- C.foo["att"]
- D.foo{"att"}
- E.foo["a"+ "t" + "t"]

来自：JavaScript 基本功检测

参考：无

549.

[不定项选择题]

以下哪些是 javascript 的全局函数：()

- A.escape
- B.parseFloat
- C.eval
- D.setTimeout
- E.alert

来自：JavaScript 基本功检测

参考：abc

550.

[不定项选择题]

关于 IFrame 表述正确的有：()

- A.通过 IFrame，网页可以嵌入其他网页内容，并可以动态更改
- B.在相同域名下，内嵌的 IFrame 可以获取外层网页的对象
- C.在相同域名下，外层网页脚本可以获取 IFrame 网页内的对象
- D.可以通过脚本调整 IFrame 的大小

来自：JavaScript 基本功检测

参考: abcd

551.

[不定项选择题]

关于表格表述正确的有: ()

- A. 表格中可以包含 TBODY 元素
- B. 表格中可以包含 CAPTION 元素
- C. 表格中可以包含多个 TBODY 元素
- D. 表格中可以包含 COLGROUP 元素
- E. 表格中可以包含 COL 元素

来自: JavaScript 基本功检测

参考: abcde

552.

[不定项选择题]

关于 IE 的 window 对象表述正确的有: ()

- A. window.opener 属性本身就是指向 window 对象
- B. window.reload()方法可以用来刷新当前页面
- C. window.location=" a.html" 和 window.location.href=" a.html" 的作用都是把当前页面替换成 a.html 页面
- D. 定义了全局变量 g; 可以用 window.g 的方式来存取该变量

来自: JavaScript 基本功检测

参考: acd

553.

[问答题]

谈谈 javascript 数组排序方法 sort()的使用，重点介绍 sort()参数的使用及其内部机制

来自: JavaScript 基本功检测

参考:

sort 的实现的功能类似 JAVA 的比较器，数据排序从多维数组的第一维开始排序
可以自己定义排序方法，很不多的函数

554.

[问答题]

简述 DIV 元素和 SPAN 元素的区别。

来自：JavaScript 基本功检测

参考：

DIV 有回车，SPAN 没有

555.

[问答题]

结合 text 这段结构，谈谈 innerHTML outerHTML innerText 之间的区别。

来自：JavaScript 基本功检测

参考：

这个问题只要写一下看的很清楚

innerHTML 对象里面的 HTML,outerHTML 包括对象和里面的

innerText 对象里面的文本

556.

[问答题]

说几条 XHTML 规范的内容（至少 3 条）

来自：JavaScript 基本功检测

参考：

属性加引号，不能有不匹配的标签，加定义

557.

[问答题]

对 Web 标准化（或网站重构）知道哪些相关的知识，简述几条你知道的 Web 标准？

来自：JavaScript 基本功检测

参考：

网页主要由三部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。

对应的网站标准也分三方面：结构化标准语言，主要包括 XHTML 和 XML；表现标准语言主要包括 CSS；行为标准主要包括对象模型（如 W3C DOM）、ECMAScript 等。

558.

[问答题]

使用过类库吗？最喜欢哪个？为什么？自己有写过类库吗？比如 DOM 的扩展。有使用过服务端 JavaScript 框架吗？ECMAScript 和 JavaScript 的区别是什么？有用过 JavaScript 代码校验工具吗？有读过或推荐的 JavaScript 书籍吗？会为你的 JavaScript 代码写单元测试吗？

来自：JavaScript 通用面试题型

参考：无

559.

[问答题]

为什么基本上所有对象都有 `toString` 方法？知道 Mozilla Firefox 用的是哪个解析器吗？其他浏览器呢？JavaScript 支持 `lambda` 函数吗？你用过或写过的最有用的 JavaScript 函数是什么？JavaScript 有块级作用域吗？能解释下 `Ajax/XMLHttpRequest` 是如何工作的吗？JavaScript 支持类继承吗？能写一个用了 `with` 表达式的代码片段吗？知道什么是 Greasemonkey 吗？有用过吗？你认为 `innerHTML` 是魔鬼吗？什么是 JSON？

来自：JavaScript 通用面试题型

参考：无

560.

[问答题]

Can you give me an example of a generator？JSONP 是如何工作的？请举个单例模式的例子。未定义和未声明之间有什么区别？有用 `Raphael` 或 `Canvas` 元素做过动画吗？熟悉 `Web Worker` 吗？做过 `profiling` 吗？都有用过哪些工具？有读过新的 `ECMAScript` 规范吗？都有哪些新特性？

来自：JavaScript 通用面试题型

参考：无

561.

[问答题]

谁最初写了 ECMAScript? 知道他在哪工作, 以及他的 title 是什么吗? 写 jQuery 的那男孩叫什么? 谁写了 JSLint?

来自: JavaScript 通用面试题型

参考: 无

562.

[问答题]

哪些浏览器支持标准的 addEventListener ? 哪些浏览器对于 getElementById 的实现有问题? 比如它会返回 name 属性一致的元素。

来自: JavaScript 通用面试题型

参考: 无

563.

[问答题]

如何在没有定义 toString 方法的对象上调用 toString() ? 在调用函数时使用 new 会发生什么? 什么是作用域链? 如何在函数里创建静态变量? 如果给你一个类名的字符串, 你如何实例化他? 什么是 currying? 如何在 JavaScript 里用他? 什么是匿名函数? 什么是 lambda 函数? 什么是 ‘live’ 容器? (应该是指 getElementsByTagName 等方法返回的元素) var 为什么重要? 如何调试 JavaScript?

来自: JavaScript 通用面试题型

参考: 无

564.

[单选题]

在 IE 中要想获得当前窗口的位置可以使用 window 对象的()方法

- A. windowX
 - B. screenX
 - C. screenLeft
 - D. windowLeft
-

来自: JavaScript 基础练习题

答案: C

565.

[单选题]

分析下面的 JavaScript 代码段

```
a=new Array(2,3,4,5,6);
sum=0;
输出结果是().(选择一项)
for(i=1;i<a.length;i++ )
    sum +=a[i];
document.write(sum);
```

- A. 20
- B. 18
- C. 14
- D. 12

来自：JavaScript 基础练习题

答案：B

566.

[不定项选择题]

下面对于 JavaScript 中的单选按钮（Radio）的说法正确的是（）。(选择两项)

- A. 单选按钮可以通过单击“选中”和“未选中”选项来进行切换
- B. 单选按钮没有 checked 属性
- C. 单选按钮支持 onClick 事件
- D. 单选按钮的 Length 属性返回一个选项组中单选项的个数

来自：JavaScript 基础练习题

答案：AC

567.

[单选题]

下面哪个选项中的对象与浏览列表有关()

- A. location,history
- B. window,location
- C. navigator,window
- D. historylist,location

来自：JavaScript 基础练习题

答案：A

568.

[单选题]

下列（）标记符属性为布尔属性（即只需要指定属性的存在，而不用指定其值的标记符属性）。（选择一项）

- A. noshade
 - B. width
 - C. bold
 - D. size
-

来自：JavaScript 基础练习题

答案：A

569.

[单选题]

在某一页面下载时，要自动显示出另一页面，可通过在<body>中使用下边的哪一事件来完成（）。（选择一项）

- A. onload
 - B. onunload
 - C. onclick
 - D. onchange
-

来自：JavaScript 基础练习题

答案：A

570.

[单选题]

在 HTML 中，Location 对象的()属性用于设置或检索 URL 的端口号。（选择一项）

- A. hostname
 - B. host
 - C. pathname
 - D. href
-

来自：JavaScript 基础练习题

答案: B

571.

[单选题]

下面哪个选项中的对象与浏览列表有关()

- A. location,history
- B. window,location
- C. navigator,window
- D. historylist,location

来自: JavaScript 基础练习题

答案: A

572.

[单选题]

下列 JavaScript 语句中, () 能实现单击一个按钮时弹出一个消息框。(选择一项)

- A. <BUTTON VALUE ="鼠标响应" onClick=alert("确定")></BUTTON>
- B. <INPUT TYPE="BUTTON" VALUE ="鼠标响应" onClick=alert("确定")>
- C. <INPUT TYPE="BUTTON" VALUE ="鼠标响应" onChange=alert("确定")>
- D. <BUTTON VALUE ="鼠标响应" onChange=alert("确定")></BUTTON>

来自: JavaScript 基础练习题

答案: B

573.

[单选题]

在 HTML 页面中, 下面关于 Window 对象的说法不正确的是 ()。(选择一项)

- A. Window 对象表示浏览器的窗口, 可用于检索有关窗口状态的信息
- B. Window 对象是浏览器所有内容的主容器
- C. 浏览器打开 HTML 文档时, 通常会创建一个 Window 对象
- D. 如果文档定义了多个框架, 浏览器只为原始文档创建一个 Window 对象, 无须为每个框架创建 Window 对象

来自: JavaScript 基础练习题

答案: D

574.

[单选题]

在 JavaScript 中,表单文本框(Text)不支持的事件包括()。(选择一项)

- A. onBlur
 - B. onLostFocused
 - C. onFocus
 - D. onChange
-

来自：JavaScript 基础练习题

答案：B

575.

[单选题]

分析下面的 javascript 代码:

```
x=11;  
y="number";  
m= x+y ;  
m 的值为 ( )。(选择一项)  
A. 11number  
B. number  
C. 11  
D. 程序报错
```

来自：JavaScript 基础练习题

答案：A

576.

[单选题]

在 HTML 页面中使用外部 javaScript 文件的正确语法是 ()。(选择一项)

- A. <language="JavaScript"src="scriptfile.js">
 - B. <script language="JavaScript"src="scriptfile.js"></script>
 - C. <script language="JavaScript" =scriptfile.js></script>
 - D. <language src=" scriptfile.js">
-

来自：JavaScript 基础练习题

答案：B

577.

[单选题]

分析如下的 JavaScript 代码段，则运行后在页面上输出() (选择一项)

```
var c="10",d=10;  
document.write(c+d)
```

- A. 10
- B. 20
- C. 1010
- D. 页面报错

来自：JavaScript 基础练习题

答案：C

578.

[单选题]

网页编程中，运行下面的 javascript 代码：

```
<script language="javascript">  
x=3;  
y=2;  
z=(x+2)/y;  
alert(z);  
</script>
```

则提示框中显示()。 (选择一项)

- A. 2
- B. 2.5
- C. 32/2
- D. 16

来自：JavaScript 基础练习题

答案：B

579.

[单选题]

在 JAVASCRIPT 中,命令按钮(Button)支持的事件包括() (选择一项)

- A. onClick
- B. onChange
- C. onSelect

D. onSubmit

来自：JavaScript 基础练习题

答案：A

580.

[单选题]

在当前页面的同一目录下有一名 show.js 的文件，下列()代码可以正确访问该件。(选择一项)

- A. <script language= "show.js"></script>
- B. <script type="show.js"></script>
- C. <script src="show.js"></script>
- D. <script runat="show.js"></script>

来自：JavaScript 基础练习题

答案：C

581.

[单选题]

在 javaScript 中，可以使用 Date 对象的 () 方法返回该对象的日期。(选择一项)

- A. getDate
- B. getYear
- C. getMonth
- D. gerTime

来自：JavaScript 基础练习题

答案：A

582.

[单选题]

那一个对象可以获得屏幕的大小()

- A. window
- B. screen
- C. navigator
- D. screenX

来自：JavaScript 基础练习题

答案：B

583.

[单选题]

分析下面的 JavaScript 语句：

Str = "This apple costs "+5.05;

执行后 str 的结果是（）。(选择一项)

- A. This apple costs 50.5
- B. This apple costs 5.5
- C. "This apple costs" 50.5
- D. "This apple costs" 5.5

来自：JavaScript 基础练习题

答案：A

584.

[单选题]

setInterval("alert('welcome')",1000);

这段代码的意思是()

- 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. 语句报错,语法有问题
- D. 每隔一秒钟弹出一个对话框

来自：JavaScript 基础练习题

答案：D

585.

[单选题]

要求用 JavaScript 实现下面的功能：在一个文本框中内容发生改变后，单击页面的其他部分将弹出一个消息框显示文本框中的内容，下面语句正确的是（）(选择一项)

- A. <input type="text" onChange="alert(this.value)">
- B. <input type="text" onClick="alert(this.value)">
- C. <input type="text" onChange="alert(text.value)">
- D. <input type="text" onClick="alert(value)">

来自：JavaScript 基础练习题

答案：A

586.

[单选题]

window 对象的 open 方法返回的是()

- A. 没有返回值
- B. boolean 类型，表示当前窗口是否打开成功
- C. 返回打开新窗口的对象
- D. 返回 int 类型的值，开启窗口的个数

来自：JavaScript 基础练习题

答案：C

587.

[单选题]

分析下面的 JavaScript 代码段：

```
function employee(name,code)
{
    this.name="wangli";
    this.code="A001";
}
newemp=new employee("zhangming",'A002');
document.write("雇员姓名:"+ newemp.name+ "<br>");
document.write("雇员代号:"+ newemp.code +"<br>");
```

输出的结果是().(选择一项)

- A. 雇员姓名:wangli 雇员代码:A001
- B. 雇员姓名： zhangming 雇员代码： A002
- C. 雇员姓名： null, 雇员代码： null
- D. 代码有错误， 无输出结果

来自：JavaScript 基础练习题

答案：A

588.

[单选题]

在 HTML 页面中，下面有关的 Document 对象的描述错误的是（）。(选择一项)

- A. Document 对象用于检查和修改 HTML 元素和文档中的文本
 - B. Document 对象用于检索浏览器窗口中的 HTML 文档的信息
 - C. Document 对象提供客户最近访问的 URL 的列表
 - D. Document 对象的 location 属性包含有关当前 URL 的信息
-

来自：JavaScript 基础练习题

答案：C

589.

[单选题]

分析下面的 JavaScript 代码段：

```
a = new Array("100","2111","41111");
for(var i = 0;i < a.length;i  ){
document.write(a[i] "");
}
```

输出结果是（）。(选择一项)

- A. 100 2111 41111
 - B. 1 2 3
 - C. 0 1 2
 - D. 1 2 4
-

来自：JavaScript 基础练习题

答案：A

590.

[单选题]

分析下面的 JavaScript 代码段：

```
var a=15.49;
document.write(Math.round(a));
```

输出的结果是（）。(选择一项)

- A. 15
- B. 16
- C. 15.5

D. 15.4

来自：JavaScript 基础练习题

答案：A

591.

[单选题]

以下（）为 JavaScript 声明变量的语句。（选择一项）

- A. dim x;
- B. int x;
- C. var x;
- D. x;

来自：JavaScript 基础练习题

答案：C

592.

[单选题]

分析如下的 JavaScript 代码片段, b 的值为（）(选择一项)

```
Var a = 1.5,b;  
b=parseInt(a);
```

- A. 2
- B. 0.5
- C. 1
- D. 1.5

来自：JavaScript 基础练习题

答案：C

593.

[不定项选择题]

声明一个对象，给它加上 name 属性和 show 方法显示其 name 值，以下代码中正确的是（ ）

- A. var obj = [name:"zhangsan",show:function(){alert(name)}];
- B. var obj = {name:"zhangsan",show:"alert(this.name)"}

-
- C. var obj = {name:"zhangsan",show:function(){alert(name);}};
 - D. var obj = {name:"zhangsan",show:function(){alert(this.name);}};

来自：Javascript 面试笔试题

答案：D

594.

[不定项选择题]

以下关于 `Array` 数组对象的说法不正确的是（ ）

- A. 对数组里数据的排序可以用 `sort` 函数，如果排序效果非预期，可以给 `sort` 函数加一个排序函数的参数
 - B. `reverse` 用于对数组数据的倒序排列
 - C. 向数组的最后位置加一个新元素，可以用 `pop` 方法
 - D. `unshift` 方法用于向数组删除第一个元素
-

来自：Javascript 面试笔试题

答案：CD

595.

[不定项选择题]

要将页面的状态栏中显示“已经选中该文本框”，下列 JavaScript 语句正确的是（ ）

- A. `window.status="已经选中该文本框"`
 - B. `document.status="已经选中该文本框"`
 - C. `window.screen="已经选中该文本框"`
 - D. `document.screen="已经选中该文本框"`
-

来自：Javascript 面试笔试题

答案：A

596.

[不定项选择题]

点击页面的按钮，使之打开一个新窗口，加载一个网页，以下 JavaScript 代码中可行的是（ ）

- A. `<input type="button" value="new" onclick="open('new.html', '_blank') "/>`
- B. `<input type="button" value="new"`

```
onclick="window.location='new.html';"/>
C. <input type="button" value="new"
onclick=" location.assign('new.html');"/>
D. <form target="_blank" action="new.html">
<input type="submit" value="new"/>
</form>
```

来自：Javascript 面试笔试题

答案：AD

597.

[不定项选择题]

使用 JavaScript 向网页中输出

hello

，以下代码中可行的是（ ）

- A. <script type="text/javascript">
document.write(<h1>hello</h1>);
</script>
 - B. <script type="text/javascript">
document.write("<h1>hello</h1>");
</script>
 - C. <script type="text/javascript">
<h1>hello</h1>
</script>
 - D. <h1>
<script type="text/javascript">
 document.write("hello");
</script>
</h1>
-

来自：Javascript 面试笔试题

答案：BD

598.

[不定项选择题]

分析下面的代码：

```
<html>
<head>
<script type="text/javascript">
    function writelt (value) { document.myfm.first_text.value=value;}
</script>
```

```
</head>
<body bgcolor="#ffffff">
    <form name="myfm">
        <input type="text" name="first_text">
        <input type="text" name="second_text" onchange="writelt(value)">
    </form>
</body>
</html>
```

以下说法中正确的是（ ）

- A. 在页面的第二个文本框中输入内容后，当鼠标离开第二个文本框时，第一个文本框的内容不变
- B. 在页面的第一个文本框中输入内容后，当鼠标离开第一个文本框时，将在第二个文本框中复制第一个文本框的内容
- C. 在页面的第二个文本框中输入内容后，当鼠标离开第二个文本框时，将在第一个文本框中复制第二个文本框的内容
- D. 在页面的第一个文本框中输入内容后，当鼠标离开第一个文本框时，第二个文本框的内容不变

来自：Javascript 面试笔试题

答案：CD

599.

[不定项选择题]

下面的 JavaScript 语句中，（ ）实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

- A.

```
for(var i=0;i< form1.elements.length;i++) {
if(form1.elements[i].type=="text")
form1.elements[i].value="";
}
```
- B.

```
for(var i=0;i<document.forms.length;i++) {
if(forms[0].elements[i].type=="text")
forms[0].elements[i].value="";
}
```
- C.

```
if(document.form.elements.type=="text")
form.elements[i].value="";
};
```
- D.

```
for(var i=0;i<document.forms.length; i++){
for(var j=0;j<document.forms[i].elements.length; j++){
if(document.forms[i].elements[j].type=="text")
document.forms[i].elements[j].value="";
}
};
```

来自：Javascript 面试笔试题

答案：D

600.

[不定项选择题]

在表单(form1)中有一个文本框元素(fname)，用于输入电话号码，格式如：010-82668155，要求前 3 位是 010，紧接一个“-”，后面是 8 位数字。要求在提交表单时，根据上述条件验证该文本框中输入内容的有效性，下列语句中，（ ）能正确实现以上功能

- A. var str= form1.fname.value;
 if(str.substr(0,4)!="010-" || str.substr(4).length!=8 ||
 isNaN(parseFloat(str.substr(4))))
 alert("无效的电话号码！");
- B. var str= form1.fname.value;
 if(str.substr(0,4)!="010-" && str.substr(4).length!=8 &&
 isNaN(parseFloat(str.substr(4))))
 alert("无效的电话号码！");
- C. var str= form1.fname.value;
 if(str.substr(0,3)!="010-" || str.substr(3).length!=8 ||
 isNaN(parseFloat(str.substr(3))))
 alert("无效的电话号码！");
- D. var str= form1.fname.value;
 if(str.substr(0,4)!="010-" && str.substr(4).length!=8 &&
 !isNaN(parseFloat(str.substr(4))))
 alert("无效的电话号码！");

来自：Javascript 面试笔试题

答案：A

601.

[不定项选择题]

关于正则表达式声明 6 位数字的邮编，以下代码正确的是（ ）

- A. var reg = /\d6/;
- B. var reg = \d{6};
- C. var reg = /\d{6}/;
- D. var reg = new RegExp("\d{6}");

来自：Javascript 面试笔试题

答案：C

602.

[不定项选择题]

关于 JavaScript 里的 xml 处理，以下说明正确的是（ ）

- 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
-

来自：Javascript 面试笔试题

答案：BCD

603.

[问答题]

列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个。

来自：Javascript 面试笔试题

参考：

对象：Window document location screen history navigator

方法：Alert() confirm() prompt() open() close()

604.

[问答题]

简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法并做简单说明

来自：Javascript 面试笔试题

参考：

Document.getElementById 根据元素 id 查找元素

Document.getElementsByName 根据元素 name 查找元素

Document.getElementsByTagName 根据指定的元素名查找元素

605.

[程序题]

补充按钮事件的函数，确认用户是否退出当前页面，确认之后关闭窗口；

来自：Javascript 面试笔试题

参考：

```
<html>
<head>
<script type="text/javascript" >
function closeWin(){
//在此处添加代码
if(confirm("确定要退出吗？")){
    window.close();
}
</script>
</head>
<body>
<input type="button" value="关闭窗口" onclick="closeWin()"/>
</body>
</html>
```

606.

[程序题]

写出简单描述 html 标签（不带属性的开始标签和结束标签）的正则表达式，并将以下字符串中的 html 标签去除掉

```
var str = "<div>这里是 div<p>里面的段落</p></div>";
```

来自：Javascript 面试笔试题

参考：

```
<script type="text/javascript">
var reg = /<\?\\w+\\?>/gi;
var str = "<div>这里是 div<p>里面的段落</p></div>";
alert(str.replace(reg,""));
</script>
```

607.

[程序题]

完成 foo()函数的内容，要求能够弹出对话框提示当前选中的是第几个单选框。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```

</head>
<body>
<script type="text/javascript" >
function foo() {
//在此处添加代码
    var rdo = document.form1.radioGroup;
    for(var i =0 ;i<rdo.length;i++){
        if(rdo[i].checked){
            alert("您选择的是第"+(i+1)+"个单选框");
        }
    }
}

</script>
<body>
<form name="form1" onsubmit="return foo();">
<input type="radio" name="radioGroup"/>
<input type="radio" name="radioGroup"/>
<input type="radio" name="radioGroup"/>
<input type="radio" name="radioGroup"/>
<input type="submit"/>
</form>
</body>
</html>

```

来自：Javascript 面试笔试题

参考：参考上面代码

608.

[程序题]

完成函数 showImg(), 要求能够动态根据下拉列表的选项变化，更新图片的显示（15分）

```

<body>
<script type="text/javascript" >
function showImg (oSel) {
//在此处添加代码
var str = oSel.value;
    document.getElementById("pic").src = str+".jpg";
}
</script>

<br />

```

```
<select id="sel" onchange="showImg(this)">
  <option value="img1">城市生活</option>
  <option value="img2">都市早报</option>
  <option value="img3">青山绿水</option>
</select></body>
```

来自： Javascript 面试笔试题

参考： 参考上面代码

609.

[单选题]

写 “Hello World” 的正确 javascript 语法是？

()

- A. document.write("Hello World")
 - B. "Hello World"
 - C. response.write("Hello World")
 - D. ("Hello World")
-

来自： Javascript 进阶练习题

参考： A

610.

[单选题]

JS 特性不包括()

- A.解释性
 - B.用于客户端
 - C.基于对象
 - D.面向对象
-

来自： Javascript 进阶练习题

参考： D

611.

[单选题]

下列 JS 的判断语句中()是正确的 ()

- A.if(i==0)
 - B.if(i=0)
 - C.if i==0 then
 - D.if i=0 then
-

来自： Javascript 进阶练习题

参考： A

612.

[单选题]

下列 JavaScript 的循环语句中()是正确的 ()

- A.if(i<10;i++)
 - B.for(i=0;i<10)
 - C.for i=1 to 10
 - D.for(i=0;i<=10;i++)
-

来自：Javascript 进阶练习题

参考：D

613.

[单选题]

下列的哪一个表达式将返回假 ()

- A.! (3<=1)
 - B.(4>=4)&&(5<=2)
 - C.("a"=="a")&&("c"!="d")
 - D.(2<3) || (3<2)
-

来自：Javascript 进阶练习题

参考：B

614.

[单选题]

下 列 选 项 中 , () 不 是 网 页 中 的 事 件
()

- A.onclick
 - B.onmouseover
 - C.onsubmit
 - D.onpressbutton
-

来自：Javascript 进阶练习题

参考：D

615.

[单选题]

有语句 “var x=0;while(____) x+=2;”，要使 while 循环体执行 10 次，空白处的循环判定式应写为： ()

- A. x<10
 - B. x<=10
 - C.x<20
 - D.x<=20
-

来自： Javascript 进阶练习题

参考： C

616.

[单选题]

JS 语句

()

`var a1=10;`

`var a2=20;`

`alert("a1+a2="+a1+a2)`

将显示()结果

- A.a1+a2=30 B.a1+a2=1020 C.a1+a2=a1+a2

来自： Javascript 进阶练习题

参考： B

617.

[单选题]

将字串 s 中的所有字母变为小写字母的方法是

()

A.s.toSmallCase() B.s.toLowerCase()

C.s.toUpperCase() D.s.toUpperChars()

来自： Javascript 进阶练习题

参考： B

618.

[单选题]

以下()表达式产生一个 0~7 之间(含 0,7)的随机整数.

()

A.Math.floor(Math.random()*6)

B.Math.floor(Math.random()*7)

C.Math.floor(Math.random()*8)

D.Math.ceil(Math.random()*8)

来自： Javascript 进阶练习题

参考： C

619.

[单选题]

产生当前日期的方法是 ()

- A.Now(); B.Date() C.new Date() D.new Now()
-

来自：Javascript 进阶练习题

参考：C

620.

[单选题]

如果想在网页显示后,动态地改变网页的标题 ()

- A.是不可能的 B.通过 `document.write("新的标题内容")`
C. 通过 `document.title="新的标题内容"`
D. 通过 `document.changeTitle("新的标题内容")`
-

来自：Javascript 进阶练习题

参考：C

621.

[单选题]

某网页中有一个窗体对象,其名称是 `mainForm`,该窗体对象的第一个元素是按钮,其名称是 `myButton`,表述该按钮对象的方法是 ()

- A.`document.forms.myButton` B.`document.mainForm.myButton`
C.`document.forms[0].element[0]` D.以上都可以
-

来自：Javascript 进阶练习题

参考：B

622.

[单选题]

HTML 文档的树状结构中, () 标签为文档的根节点, 位于结构中的最顶层。 ()

- A.<HTML>B.<HEAD>C.<BODY>D.<TITLE>
-

来自：Javascript 进阶练习题

参考：A

623.

[单选题]

- 在 HTML 页面中，CSS 样式的属性名为 background-image 对应的 style 对象的属性名是（ ）
A.background B.backgroungImage C.image D.background
-

来自：Javascript 进阶练习题

参考：B

624.

[单选题]

在使用 Javascript 实现省市级联菜单功能时，在添加城市列表前清空原来的下拉选项的代码是（ ）

- A.document.myform.selCity.options.clear()
B. document.myform.selCity.options.deleteAll()
C. document.myform.selCity.options.length=0
D. document.myform.selCity.options.size=0
-

来自：Javascript 进阶练习题

参考：C

625.

[单选题]

HMTL 表单的首要标记是<form>,</form>标记的参数 method 表示表单发送的方法，可能为 get 或 post，下列关于 get 和 post 的描述正确的是（ ）

- A.post 方法传递的数据对客户端是不可见的
B.get 请求信息以查询字符串的形式发送，查询字符串长度没有大小限制
C.post 方法对发送数据的数量限制在 255 个字符之内
D.get 方法传递的数据对客户端是不可见的
-

来自：Javascript 进阶练习题

参考：D

626.

[不定项选择题]

在 DOM 对象模型中，下列选项中的（）对象位于 DOM 对象模型的第二层。（选择二项）

()

- A. history
- B. document
- C. button
- D. text

来自：Javascript 进阶练习题

参考：AB

627.

[单选题]

在 HTML 文档对象模型中，`history` 对象的（）用于加载历史列表中的下一个 URL 页面。

()

- A. next()
- B. back()
- C. forward()
- D. go(-1)

来自：Javascript 进阶练习题

参考：C

628.

[单选题]

在 Javascript 中要改变页面文档的背景色，需要修改 `document` 对象的（）属性。 ()

- A. BackColor
- B. BackgroundColor
- C. BgColor
- D. Background

来自：Javascript 进阶练习题

参考：C

629.

[单选题]

在 HTML 页面中，不能与 `onChange` 事件处理程序相关联的表单元素有（ ）

- A. 文本框
- B. 复选框
- C. 列表框
- D. 按钮

来自：Javascript 进阶练习题

参考：D

630.

[单选题]

- 在 HTML 页面上编写 Javascript 代码时，应编写在（）标签中间。 ()
- A.<javascript>和</javascript> B.<script>和</script> C. <head>和</head> D. <body>和</body>
-

来自：Javascript 进阶练习题

参考：B

631.

[单选题]

- 在 Javascript 浏览器对象模型中，window 对象的（）属性用来指定浏览器状态栏中显示的 临 时 消 息 。
- ()
- A. status B.screen C.history D.document
-

来自：Javascript 进阶练习题

参考：A

632.

[单选题]

编写 Javascript 函数实现网页背景色选择器，下列选项中正确的是（）

- A.function change(color){
 window.bgColor=color;
}
B. function change(color){
 document.bgColor=color;
}
C. function change(color){
 body.bgColor=color;
}
D. function change(color){
 form.bgColor=color;
}
-

来自：Javascript 进阶练习题

参考: B

633.

[单选题]

在 Javascript 中，可以使用 Date 对象的()方法返回一个月中的每一天。 ()

- A. getDate
- B.getYear
- C.getMonth
- D.getTime

来自: Javascript 进阶练习题

参考: A

634.

[单选题]

在 Javascript 中，对于浏览器对象的层次关系理解正确的是 () (选择二项) ()

- A.window 对象是所有页面内容的根对象
- B.document 对象包含 location 对象和 history 对象
- C.location 对象包含 history
- D.document 对象包含 form 对象

来自: Javascript 进阶练习题

参考: AD

635.

[单选题]

下列选项中关于浏览器对象的说法错误的是 ()

- A.history 对象记录了用户在一个浏览器中已经访问过的 URLs
- B.location 对象相当于 IE 浏览器中的地址栏，包含关于当前 URL 地址的信息
- C.location 对象是 history 对象的父对象
- D.location 对象是 window 对象的子对象

来自: Javascript 进阶练习题

参考: C

636.

[单选题]

在 HTML 页面中包含一个按钮控件 mybutton，如果要实现点击该按钮时调用已定义的

Javascript 函数 compute，要编写的 HTML 代码是（ ）

- A.<input name="mybutton" type="button" onBlur="compute()" value="计算" >
 - B.<input name="mybutton" type="button" onFocus="compute()" value="计算" >
 - C.<input name="mybutton" type="button" onClick="function compute()" value="计算" >
 - D.<input name="mybutton" type="button" onClick="compute()" value="计算" >
-

来自：Javascript 进阶练习题

参考：D

637.

[单选题]

分析下面的 Javascript 代码段，输出结果是（ ）

```
var mystring="I am a student";
var a=mystring.substring(9,13);
document.write(a);
```

- A. stud
 - B.tuden
 - C.uden
 - D.udent
-

来自：Javascript 进阶练习题

参考：C

638.

[单选题]

Javascript 中制作图片代替按钮的提交效果需要手动提交方法 submit()，以下调用正确的是（ ）

- A.submit();
 - B.myform.submit()
 - C.document.myform.submit()
 - D.window.myform.submit();
-

来自：Javascript 进阶练习题

参考：C

639.

[单选题]

在 HTML 页面中包含如下所示代码，则编写 Javascript 函数判断是否按下键盘上的回车键正确的编码是（ ）

```
<input name="password" type="text" onkeydown="myKeyDown()">
```

- A. function myKeyDown(){

```
if (window.keyCode==13)
    alert(“你按下了回车键” );
B. function myKeyDown(){
if (document.keyCode==13)
    alert(“你按下了回车键” );
C. function myKeyDown(){
if (event.keyCode==13)
    alert(“你按下了回车键” );
D. function myKeyDown(){
if (keyCode==13)
    alert(“你按下了回车键” );
```

来自：Javascript 进阶练习题

参考：C

640.

[单选题]

如果在 HTML 页面中包含如下图片标签，则选项中的（）语句能够实现隐藏该图片的功能。

()

- ```

A.document.getElementById("pic").style.display="visible";
B.document.getElementById("pic").style.display="disvisible";
C.document.getElementById("pic").style.display="block";
D.document.getElementById("pic").style.display="none";
```
- 

来自：Javascript 进阶练习题

参考：D

## 641.

[单选题]

如果在 HTML 页面中包含如下图片标签，则在下划线处添加（）代码能够实现隐藏该图片的功能。

( )

- ```

A. style="display:visible";
B. style="display:disvisible";
C. style="display:block";
D. style="display:none";
```
-

来自：Javascript 进阶练习题

参考：D

642.

[不定项选择题]

下列选项中，()段 HTML 代码所表示的“返回”链接能够正确实现 IE 工具栏中“后退”按钮的功能。(选择二项) ()

- A. 返回
- B. 返回
- C. 返回
- D. 返回

来自：Javascript 进阶练习题

参考：AD

643.

[单选题]

在 HTML 文档中包含如下超链接，要实现当鼠标移入该链接时，超链接文本大小变为 30px，选项中的编码正确的是 ()

- A.注册
- B.注册
- C.注册
- D.注册

来自：Javascript 进阶练习题

参考：C

644.

[单选题]

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

来自：Javascript 进阶练习题

参考：D

645.

[不定项选择题]

在 HTML 页面中，定义了如下所示的 Javascript 函数，则正确调用该函数的 HTML 代码是（选择二项） ()

```
function compute(op){  
    alert(op);  
}  
A.<input name="a" type="button" onclick="compute(this.value)" value="+">  
B.<input name="b" type="button" onclick="compute('')" value="-">  
C.<input name="c" type="button" onclick="compute('*')" value="*>  
D.<input name="d" type="button" onclick="compute(/ )" value="/">
```

来自：Javascript 进阶练习题

参考：AB

646.

[单选题]

在 HTML 页面上包含如下创建层的语句，那么编写 Javascript 语句实现显示该层的语句错误的是 ()

```
<html>  
<body>  
<div id="imageLayer" style="display:none;">  
      
</body>  
<html>
```

- A. document.getElementByTagName("div")[0].style.display="block"
 - B. document.getElementById("imageLayer").style.display="block";
 - C. document.getElementByName("imageLayer")[0].style.display="block";
 - D. document.getElementByName("imageLayer").get(0).style.display="block";
-

来自：Javascript 进阶练习题

参考：D

647.

[单选题]

分析下面的 Javascript 代码段，输出结果是 ()

```
var s1=parseInt( "101 中学" );
```

document.write(s1);
A. NaN B.101 中学 C.101 D.出现脚本错误

来自：Javascript 进阶练习题
参考：C

648.

[单选题]

在 HTML 中，点击图片” previous.gif ” 上的超级链接后页面将加载历史列表中的上一个 URL 页面。代码如下所示，应在下划线处填入（）

A.”javascript:history.go(-1)”; B. “history.go(1)” C. “history.go(-1)” D. “javascript:history.go(1)”

来自：Javascript 进阶练习题
参考：A

649.

[单选题]

在 HTML 页面上包含如下所示的层对象，则 javascript 语句 document.getElementById(“info”).innerHTML 的值是（）

<div id=” info ” style=” display:block ” ><p>请填写</p></div>
A.请填写 B.<p>请填写</p> C.id=” info ” style=” display:block ”
D.<div id=” info ” style=” display:block ” ><p>请填写</p>

来自：Javascript 进阶练习题
参考：A

650.

[单选题]

以下哪条语句会产生运行错误（A）

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

来自：Javascript 进阶练习题
参考：

651.

[问答题]

阅读程序写结果

```
function replaceStr(inStr, oldStr, newStr){  
    var rep = inStr;  
    while (rep.indexOf(oldStr) > -1) {  
        rep = rep.replace(oldStr, newStr);  
    }  
    return rep;  
}  
alert(replaceStr("how do you do","do","are"));
```

来自： Javascript 进阶练习题

参考：

弹出警示对话框，显示 how are you are。

652.

[问答题]

阅读程序写结果

```
<script>  
var x,y=null;  
alert(x);  
alert(y);  
alert(x=y);  
alert(x==y);  
</script>
```

来自： Javascript 进阶练习题

参考：

依次弹出四个警示对话框，分别显示 undefined、null、null、true。

653.

[问答题]

阅读程序写结果

```
</html>  
x="a";  
y="b";
```

```
z=false;
function testOne(){
var x="c";
var y="d";
z=true;
    alert(x);
    alert(y);
    alert(z);
}
function testTwo(){
    alert(x);
    alert(y);
    alert(z);
}
testOne();
testTwo();
```

来自： Javascript 进阶练习题

参考：

依次弹出六个警示对话框，分别显示 c、d、true、a、b、true。

654.

[问答题]

阅读程序写结果

当单击 button 按钮时，出现什么结果。

```
<html>
    <head>
        <title>Untitled Document</title>
        <script language=JavaScript>
            function add(){
                var first=document.myForm.first.value ;
                var second=parseInt(document.myForm.second.value);
                var third= parseInt(document.myForm.third.value);
                alert(first+second+third);
            }
        </script>
    </head>
    <body>
        <form name="myForm">
            <input type=text name="first" value="40">
            <input type=text name="second" value="30">
            <input type=text name="third" value="70">
```

```
<input type=button value="add" onclick=add()>
</form>
</body>
```

来自： Javascript 进阶练习题

参考：

弹出警示对话框，显示 403070。

655.

[程序题]

实现在标题栏和状态栏上动态显示当前时间的效果。

来自： Javascript 进阶练习题

参考：

```
<html>
<head>
<title>新建网页 1</title>
</head>
<body onload="showTime()">
<script>
function showTime(){
now=new Date();
display=now.toLocaleString();
document.title=display;
status=display;
setTimeout("showTime()",1000)
}
</script>
</body>
</html>
```

656.

[程序题]

交换图像。

来自： Javascript 进阶练习题

参考：

```
<a href="#" onmouseover="document.p1.src='images/IMG02.JPG'">
```

```
onmouseout="document.p1.src='images/IMG01.JPG'">

</a>
```

657.

[程序题]

改变下拉列表框的选项时能显示当前选项的文本和值。

来自： Javascript 进阶练习题

参考：

```
<form name="a">
<select name="a" size="1" onchange="_sel(this)">
<option value="a">1</option>
<option value="b">2</option>
<option value="c">3</option>
</select>
</form>
<script>
function _sel(obj){
alert("显示文本：" + obj.options[obj.selectedIndex].text);
alert("值：" + obj.options[obj.selectedIndex].value);
}
</script>
```

658.

[程序题]

要求能够弹出对话框提示当前选中的是第几个单选框。

来自： Javascript 进阶练习题

参考：

```
<html>
<body>
<script>
Function foo(){
    var rg = document.getElementsByName("radioGroup");
    for(var i= 0; i<rg.length; i++)
    {
        if(rg[i].checked)
        {
```

```

        alert("你选择了第" + (i+1) + "个单选框");
    }
}
return false;
}

```

</script>

<body>

<form name="form1" onsubmit="return foo();">

<input type="radio" name="radioGroup"/>

<input type="submit"/>

</form>

</body>

</html>

659.

[程序题]

改变下拉列表框显示图片，并显示在文本框中。

来自：Javascript 进阶练习题

参考：

<html>

<head>

<title>图像切换</title>

<script>

```

function LoadImg(f){
    document.img1.src=document.form1.D1.options[document.form1.D1.selectedIndex].value;
    document.form1.T1.value=document.form1.D1.options[document.form1.D1.selectedIndex].
    value;
}

```

</script>

</head>

<body>

<form name="form1" >

<p><input type="text" name="T1" size="20">

<select size="1" name="D1" onchange="LoadImg(this.form)">

<option selected value="images\img01.jpg">图片一</option>

<option value="images\img02.jpg">图片二</option>

```
<option value="images\img03.jpg">图片三</option>
</select></p>

</form>
</body>
</html>
```

660.

[程序题]

在下面的 HTML 文档中，编写函数 test()，实现如下功能：

- (1) 当多行文本框中的字符数超过 20 个，截取至 20 个
- (2) 在 id 为 number 的 td 中显示文本框的字符个数

来自：Javascript 进阶练习题

参考：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>留言</title>
<script>
    function test(){
        var content = document.getElementById("feedBack").value;
        if(content.length>20){
            content = content.substr(0,20);
        }
        document.getElementById("feedBack").value=content;
        document.getElementById("number").innerHTML=content.length;
    }
</script>
</head>
<body>
<table>
<tr>
<td>
    留言
</td>
<td id="number">
    0
</td>
</tr>
<tr>
<td colspan=2>
```

```
<textarea id="feedBack" onkeyup="test()" rows=6></textarea>
</td>
</tr>
</table>
</body>
</html>
```

661.

[运算题]

1 && 3

来自：Javascript 表达式运算练习题

答案：3

布尔值在"&&"运算时候，如果左为 true 时，总是返回右边，反之则直接返回左边

662.

[运算题]

1 && "foo" || 0

来自：Javascript 表达式运算练习题

答案："foo"

布尔值在"||"运算时候，如果左为 false 时，总是返回右边，反之则直接返回左边

663.

[运算题]

1 || "foo" && 0

来自：Javascript 表达式运算练习题

答案：1

664.

[运算题]

(1,2,3)

来自：Javascript 表达式运算练习题

答案：3

","运算，直接输出最后一个

665.

[运算题]

```
x = {shift: [].shift};  
x.shift();  
x.length;
```

来自：Javascript 表达式运算练习题

答案：0

这个我也没看懂...望同仁指点...

666.

[运算题]

```
{foo:1}[0]
```

来自：Javascript 表达式运算练习题

答案：undefined

{foo:1}中无 key 为“0”的属性

667.

[运算题]

```
[true, false][+true, +false]
```

来自：Javascript 表达式运算练习题

答案：true

[+true, +false]为[1,0]，整体[true, false][1,0]中[1,0]可看做","运算，所以整体简化为[true, false][0]

668.

[运算题]

```
++'52'.split('')[0]
```

来自：Javascript 表达式运算练习题

答案：6

'52'.split("")为["5", "2"], ++["5", "2"][0]为 6

669.

[运算题]

a: b: c: d: e: f: g: 1, 2, 3, 4, 5;

来自：Javascript 表达式运算练习题

答案：语法错误

控制台内返回 5 是因为 eval(a: b: c: d: e: f: g: 1, 2, 3, 4, 5) 原因

670.

[运算题]

{a: 1, b: 2}["b"]

来自：Javascript 表达式运算练习题

答案：2

["b"] 转化为对象中 key= “b” 而输出

671.

[运算题]

"b" + 45

来自：Javascript 表达式运算练习题

答案： "b45"

隐式转换

672.

[运算题]

{a:{b:2}}

来自：Javascript 表达式运算练习题

答案：对象
返回含有属性 a，且 a 为一个属性 b 为 2 的对象

673.

[运算题]
(function(){}())

来自：Javascript 表达式运算练习题
答案：undefined
执行空函数返回一个未定义的值

674.

[运算题]
[1,2,3,4,5][0..toString.length]

来自：Javascript 表达式运算练习题
答案：2
[0..toString.length]相当于(0).toString.length

675.

[运算题]
({} + 'b' > {} + 'a')

来自：Javascript 表达式运算练习题
答案：true
相当于比较字符串"[object Object]b"和"[object Object]a"

676.

[运算题]
Number.prototype.x = function(){ return this === 123; };
(123).x();

来自：Javascript 表达式运算练习题
答案：false

严格测试 this 为对象，123 为 number

677.

[运算题]

Array(2).join()

来自：Javascript 表达式运算练习题

答案：","

两位的空数组

678.

[运算题]

vars: var vars = vars;

来自：Javascript 表达式运算练习题

答案：undefined

679.

[运算题]

{ foo = 123 }

来自：Javascript 表达式运算练习题

答案：123

返回块中计算的结果

680.

[运算题]

x = 1; (function(){return x; var x = 2;}())

来自：Javascript 表达式运算练习题

答案：undefined

没有变量接收自执行函数返回值

681.

[运算题]

```
delete [].length;
```

来自：Javascript 表达式运算练习题

答案：false

delete 只有在删除对象属性时候会返回 true

682.

[运算题]

```
RegExp.prototype.toString = function() {return this.source};/3/-/2/;
```

来自：Javascript 表达式运算练习题

答案：1

扩展方法相当于返回/../中的内容

683.

[运算题]

```
{break;4;}
```

来自：Javascript 表达式运算练习题

答案：语法错误

break 只能存在于循环中

684.

[运算题]

```
'foo' == new function(){ return String('foo'); };
```

来自：Javascript 表达式运算练习题

答案：false

"foo"为字符串， new function 为对象

685.

[运算题]

'foo'.split("") + []

来自：Javascript 表达式运算练习题

答案： "f,o,o"

自己试下 [1, 2] + [3, 4]