

5个经典的前端面试问题

杭州 - 网易 - @郑海波



波老师，这周三晚上有前端微专业的直播哦，《5个经典的前端面试问题》不要忘记准备呀~

课程链接：<http://study.163.com/course/introduction/1003674021.htm#/courseDetail>

15:58:40



好，我看下

2月23日 11:08:02

20:29:31

我发现有点被坑了啊。。。。直播内容也太广了。。

20:29:22

直播内容包括：

- (1) 了解目前前端招聘行情及趋势
- (2) 了解前端面试官最喜欢考察的5个经典问题
- (3) 掌握技术面试技巧
- (4) 了解前端学习过程中需要重点掌握的方向
- (5) 了解企业对前端开发工程师的要求及工作流程等

20:30:56

自我介绍

- 网易5年工作经验，4年面试经验
 - 多年友情或被迫承接各产品线的面试
- 光学工程专业本硕，研二自学前端
 - 所以别咨询我xx转前端，大几转前端这类问题了...
- 工作经历即单调又丰富
 - 前端技术部(公共支撑和框架开发)
 - 数据科学中心(2B产品线前端架构)
 - 网易云音乐(2C产品线前端架构)

2016前端行情持续高走

- 前端有大量被验证过的横向发展领域
 - Nodejs、可视化、游戏、Nativejs、跨平台富应用 ...
- 学习门槛相对较低，搭技术岗位快车的首选岗位
- 仍极度缺乏资深的前端人才

我希望看到怎样的简历

- 在精不再多，（ 99.9%一页足矣 ）
 - 量大反而暴露缺陷
- 亮点 (经历、技能)且有事实举例
 - 无法和面试官谈上个把小时的点都不需写上去
- 符合部门的岗位需求

• 特别在集中招聘时，90%的简历，我不会浏览超过2分钟

• 简历在面试中其实充当的是『地图』的角色，它可以引导面试以正确姿势对你进行提问

应聘岗位

Web 前端开发

郑海波 | 男

硕士 | 英语 四级

千万别忘记写联系方式

手机

邮箱

博客

地址

武汉光电国家实验室 F20

颜好请PO照，否则……

实习经历

淘宝：数据共享平台 | 2011.7 --- 2011.9

参与新产品淘宝指数内测版 (Index 1.0 alpha) 开发，独立完成其中部分数据可视化的开箱即用模块的开发

作了两次面向部门的前端分享，并整理内容上传到内部 Wiki

项目经历

重度 javascript 的一个单页面数据产品 (公测版本被作了较大修改)，基本所有应用逻辑都集中在页面前端，在前端构建过程中，对于前端 UI 与数据模型的分离以及模块间的解耦有了一定的认识。是个人参与的第一个团队项目。服务器端的前台使用的是 nodejs，使用到的库有 Mootools, Raphael, Mustache

推广页面 (未大规模公测)

2) Javascript 3D engine | 2011.4 --- 2011.5 | *独立完成*

小巧、独立、OOP，基于原生 JS 写的一个简单 3D 引擎 (gz 7K) 它并不依赖于固定的 API (虽然 demo 基于 canvas 2D context)，支持 WebGL 的 Render 引擎对其他场景的支持：我将其应用到了硕士阶段课题 (将生长出来的 3D 模型可视化)

示例：<http://blog.leeluolee.name/gz/archives/77/>

3) 数据可视化模块 html5 实现 | 2011.7 --- 2011.9 | *独立进行*

基于普通 Dom 或者 Raphael (svg&vml) 开发的跨浏览器的数据可视化模块，目标是可配置可扩展，目前只实现了基于普通 Canvas 的 TreeMap 以及相关的数据结构。

示例：[http://leeluolee.name/demo/beach](#) ; [http://leeluolee.name/demo/incise/](#)

4) 四个博客模板 | 2010.12 --- 2011.3 | *负责前端*

基于 typecho 博客程序的 php 模板，奠定了基本的 CSS+HTML+JS 构建经验

技能掌握

手写 HTML&CSS，理解标签的语义化，会简单的切图，处理一些常见的浏览器兼容问题。

掌握并重度痴迷 javascript，框架依赖性低，对自己常用的库都通过源码并有详细注释，在原生 JS 或一定框架的基础上，对一些常用库，精通 Mootools，熟悉 Raphael，了解 YUI, JQuery 等热门库

熟悉 Ajax，并理解构建工具上的 long polling, http stream 等 comet 技术原理。熟悉 JSON, JSONP。对前端 MVC 有一定理解。

服务器端：PHP 熟悉，NodeJS AVA 入门。

熟悉 javascript TDD，使用 Qunit 进行单元测试案例的书写

教育经历

2009/9--2012/3 华中科技大学 微电子科学与工程 硕士

2005/9--2009/6 华中科技大学 光子科学与技术 学士

一类地段

二类地段

三类地段

五环

我希望看到怎样的面试者

- 基础扎实

- 原型、作用域、this...

- 超越标准的解决问题能力

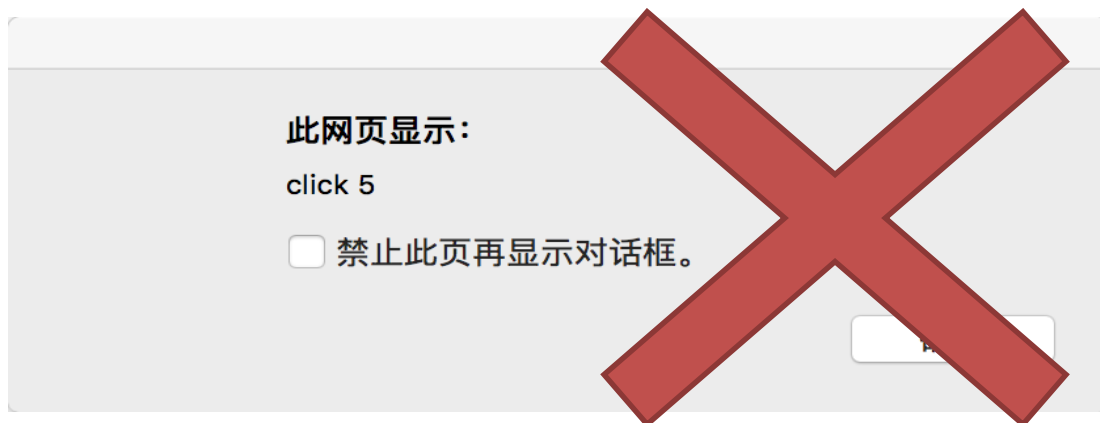
- 明白自己的优势，并对面试官做出引导

- 坦诚，适度谦虚，合理骄傲


套路1: 循环绑定事件

```
var nodes = document.querySelectorAll('ul li');  
  
for( var i = 0, len = nodes.length; i < len; i++){  
    nodes[i].addEventListener('click', function(){  
        alert('click ' + i )  
    })  
}
```

```
<ul>  
  <li>0</li>  
  <li>1</li>  
  <li>2</li>  
  <li>3</li>  
  <li>4</li>  
</ul>
```




```
for( var i = 0, len = nodes.length; i < len; i++){  
    !function(node, index){  
        node.addEventListener('click', function(){  
            alert('click' + index )  
        })  
    }(nodes[i], i);  
}
```



//注: *NodeList.prototype.forEach* 并非所有浏览器支持

```
[].slice.call(nodes).forEach(function(node, index){  
    node.addEventListener('click', function(){  
        alert('click ' + index )  
    })  
})
```

- 块级作用域变量声明let

```
for( let i = 0, len = nodes.length; i < len; i++){  
  nodes[i].addEventListener('click', function(){  
    alert('click ' + i )  
  })  
}
```

当列表项达到100或更多呢？

- 事件代理进行优化

套路2: 事件代理

- 事件代理解决什么问题
- 事件代理的基本原理（事件模型）
- 事件代理与普通事件绑定的优劣
 - 应用在频繁触发频繁触发事件时,如mouseover
 - 当列表频繁创建删除时...

套路3: this对象

1. 考察形参实参的理解
2. 考察调用方式对this指向的影响
3. 考察call和bind的理解

```
// 全局
var a = {
  a: 'haha',
  getA: function(){
    console.log(this.a)
  }
}

var b = {
  a: 'hello'
}

var getA = a.getA;
var getA2 = getA.bind(a);

function run(fn){
  fn()
}

// 分别输出?
a.getA();
getA();
run(a.getA);
getA2.call(b);
```

套路4: 原型

```
function clone( obj ){  
    // 实现  
    return Object.create(obj)  
}
```

```
function clone( obj ){  
    var ret = {};  
    // △警告: 并不是规范, 避免使用  
    ret.__proto__ = obj;  
    return ret;  
}
```

```
function clone( obj ){  
  
    function Noop(){};  
    Noop.prototype = obj;  
    return new Noop  
}
```

```
function clone( obj ){  
    // 实现  
}
```

```
var a = {name: 'a'};  
var b = clone(b);
```

```
console.log(b.name) // 'a'
```

```
a.name = 'a1'  
console.log(b.name) // 'a1'
```

```
b.name = 'b';  
console.log(a.name) // 'a1'
```

```
a.name = 'a2';  
  
console.log(b.name) // 'b'
```

套路5:

异步

```
// 现在有三个「异步接口」，分别是
// 1. getTodayUser: 获取当天用户id, callback返回userId
// 2. getTodayMovie: 获取当天的电影id, callback返回
// 3. bookMovieFor 为用户预定电影, 参数是userId和 movieId
//
// 现在要实现, 一个接口bookTodayMovieForTodayUser, 为当天用户预定当天影片

getTodayUser( function(userId){
    //获得当天的预定客户
})

getTodayMovie( function(movieId){
    //获得当天的电影
} )

// 为用户预定影片
bookMovieForUser(userId, movieId, function( isDone ){

})

// 根据以上接口封装这个函数
bookTodayMovieForTodayUser( function(isDone){
    // 为今天预定客户预定当天影片
    alert('预定成功')
})
```

标准解法

```
function bookTodayMovieForTodayUser( callback ){
    var params = []; //标记完成的请求

    function checkAndBook(){
        var uid = params[0];
        var mid = params[1];
        if(typeof uid !== 'undefined' && typeof mid !== 'undefined'){

            bookMovieForUser(uid, mid, callback)
        }
    }

    getTodayUser(function(userId){
        params[0] = userId
        checkAndBook(); // 检查
    })
    getTodayMovie(function(movieId){
        params[1] = movieId
        checkAndBook(); // 检查
    })
}
```


更通用的做法

```
function convertToPromise(fn){  
  return function(){  
    // slice arguments 不是高效做法，这里只为了简化起见  
    var args = [].slice.call(arguments);  
    var self = this;  
  
    return new Promise(function(resolve){  
      function callback(ret){  
        resolve(ret)  
      }  
      args.push( callback )  
      fn.apply(self, args)  
    })  
  }  
}
```

```
// 将所有请求转化为Promise的模式，利用Promise本身的特性来解决异步逻辑问题  
var getTodayUser2 = convertToPromise(getTodayUser);  
var getTodayMovie2 = convertToPromise(getTodayMovie);  
var bookMovieForUser2 = convertToPromise(bookMovieForUser);
```

更通用的做法

```
Promise.all([getTodayUser2(), getTodayMovie2()])  
  .then(function( param ){  
    return bookMovieForUser2(param[0], param[1])  
  }).then(function(){  
    alert('done')  
  })
```

事实上...

- 网易的校招面试是比较注重基础的
- 微专业的授课也是如此
 - 所以暂时你在前端微专业里是学不到『高大上』的React等热门框架的...

欢迎加入我们

了解前端开发微专业
跟网易资深工程师一起学前端



加入前端开发交流群
与更多小伙伴一起交流答疑



网易春季实习生推荐码
8D3K151