# 前端面试知识整理

## 前端知识

### 手写ajax，原生ajax

### 跨域

配置  前端 http header 设置Origin与服务端的源一致，如果服务端是\*的话无所谓

**jsonp怎么做的**

同源 <http://www.ruanyifeng.com/blog/2016/04/same-origin-policy.html>

协议，域名，端口都相同

浏览器允许通过设置document.domain共享 Cookie

同源政策规定，AJAX请求只能发给同源的网址，否则就报错。

1.后端服务器解决

2.JSONP 网页通过添加一个<script>元素，向服务器请求JSON数据，这种做法不受同源政策限制；服务器收到请求后，将数据放在一个指定名字的回调函数里传回来。只能发get请求

3.WebSocket

4.CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出[XMLHttpRequest](http://www.ruanyifeng.com/blog/2012/09/xmlhttprequest_level_2.html)请求，从而克服了AJAX只能[同源](http://www.ruanyifeng.com/blog/2016/04/same-origin-policy.html)使用的限制。

### 前端性能优化

<https://mp.weixin.qq.com/s/9xBlHltYImi_0liSZKQSNw>

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。

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。

资源的合并（减少http请求数量）资源的压缩（减少请求资源的大小）webpack

懒加载和预加载

重排&重绘

图标使用IconFont

### 事件冒泡机制

### 异步事件

### 异步编程

<https://mp.weixin.qq.com/s/Y21LMWcEatoBvtnqvQyxXg>

单线程，

### cookie，有哪些属性

计算机网络自顶向下方法p72

### cookie与localStorage、sessionStorage <https://www.cnblogs.com/zr123/p/8086525.html>

1.cookie在浏览器和服务器之间来回传递，sessionStorage和localStorage仅存在本地

2.cookie有路径的概念，可以限制cookie只属于某个路径下

3.cookie数据不能超过4K，同时因为每次http请求都会携带cookie、所以cookie只适合保存很小的数据；sessionStorage和localStorage虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大

4.数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭之前有效；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie：只在设置的cookie过期时间之前有效，即使窗口关闭或浏览器关闭

5.作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；cookie和localstorage在所有同源窗口中都是共享的

### 模块有哪几种标准

amd cmd commonjs es6

### 缓存方法有哪些

<https://www.cnblogs.com/chenqf/p/6386163.html>

<https://juejin.im/post/59c602276fb9a00a3d135f2e>

**强缓存和协商缓存**

好处：加快页面打开速度、降低服务器压力、减少网络损耗

### 加载一个页面的过程

**从地址栏输入地址到用户看到东西**

<https://blog.csdn.net/weixin_38150378/article/details/79408886>

1、浏览器（客户端）进行了地址解析。

2、将解析出的域名进行DNS解析。

3、通过IP寻址和arp，找到目标（服务器）地址。

4、进行TCP三次握手，建立tcp连接。

5、浏览器发送数据，等待服务器响应。

6、服务器处理请求，并对请求做出响应。

7、浏览器收到服务器响应，得到html代码。

8、渲染页面。

### Web性能优化有哪些

### web安全

<https://blog.csdn.net/c_kite/article/details/80045020>

### 纯函数是什么

### 事件代理

## JS基础

### This

改变this的指向有哪些方法

### prototype

### 原型链

### 判断数据类型的方法

### 闭包

function foo() {

  var a = 2;

  function bar() {

    Console.log(a);

  return bar;

}

// bar在定义自己的作用域以外执行

foo()();  // 2

### 闭包的缺陷

### 为什么js要设计成原型链模式

### 基本数据类型 null和undefined和NaN

### 引用类型

### 作用域

### 提升

### = = 和 = = =

### 异步加载方案

### js发送网络请求的方法

### call和apply和bind的区别

<https://www.jianshu.com/p/56a9c2d11adc>

相似：第一个参数用来改变函数的this对象的指向，后续参数传参

区别：

call和bind直接排列参数，apply用一个数组

call和apply都是对函数的直接调用，而bind方法返回的仍然是一个函数，调用要再加一个括号

### JS为什么设计成单线程

作为浏览器脚本语言，JavaScript的主要用途是与用户互动，以及操作DOM。这决定了它只能是单线程，否则会带来很复杂的同步问题。比如，假定JavaScript同时有两个线程，一个线程在某个DOM节点上添加内容，另一个线程删除了这个节点，这时浏览器应该以哪个线程为准

### js如何实现sleep

### 正则匹配的语法 正则表达式 RegExp

字面量方式： ／pattern／flags

flags：g：全局 i：忽略大小写 m：多行

pattern 模式

构造函数方式：var pattern1 = new RegExp(“pattern”, ”flag”)

### 字符串的模式匹配方法

<https://segmentfault.com/a/1190000006172772>

match( ) 参数：一个正则表达式或RegExp对象；返回一个数组。在字符串上调用这个方法本质上与调用RegExp的exec()方法相同。

search( ) 参数：一个正则表达式或RegExp对象；返回字符串中第一个匹配项的索引，如果没有找到，则返回-1

replace( ) 参数：一个RegExp对象或者一个字符串（这个字符串不会被转换成正则表达式），一个字符串或一个函数。利用replace()进行替换的时候，如果传入的是字符串，则只会替换第一个子字符串，要想替换所有的子字符串，则需要传入一个正则表达式，而且要指定全局（g）标志

### 浅拷贝和深拷贝

### js创建对象的方式

高级教程第六章

1.工厂模式：用函数封装，参数是接口字段，返回一个对象。

2.构造函数：将属性和方法赋给this对象 就可以new

3.class

### 继承

基于构造函数继承和基于原型继承

### js如何实现sleep方法

### var a=b=3；

(function(){

    var a = b = 3;  // b = 3;  var a = b;

})();

console.log(typeof a);  // "undefined"

console.log(b);  // 3 b被创建在全局作用域

### SetTimeout(function,time)

<https://www.jeffjade.com/2016/01/10/2016-01-10-javacript-setTimeout/>

1.如果被setTimeout推迟执行的回调函数是某个对象的方法，那么该方法中的this关键字将指向全局环境，而不是定义时所在的那个对象。

要等到当前脚本的同步任务和“任务队列”中已有的事件，全部处理完以后，才会执行setTimeout指定的任务。也就是说，setTimeout的真正作用是，在“任务队列”的现有事件的后面再添加一个事件，规定在指定时间执行某段代码。setTimeout添加的事件，会在下一次Event Loop执行。

## ES6

### 了解／用过的有哪些

let／const、箭头函数、解构、模块导入／导出、Promise、模版字符串、类

### Promise

Promise 是异步编程的一种解决方案。简单说就是一个容器，保存着一个异步操作的结果。本质上是一个对象，从它可以得知异步操作的信息。此外，Promise对象提供统一的接口，使得控制异步操作更加容易。

和回调函数的优势：将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调地狱；不需要每次检查错误参数，如果没有指定拒绝处理程序，错误不会反映到外部。

和事件模型的不同：事件模型必须保证事件发生之前已经给元素的事件属性赋值，promise变为完成状态后添加的处理程序也会执行。

还有一些回调函数和事件模型不能解决的情景，例如同时执行多个异步操作的情景，promise可以监控最先完成的，也可以监控到所有异步都完成的时间点。

缺点：无法取消Promise；当处于pending状态时，无法得知目前进展到哪一个阶段：状态不能以编程的形式检测。

### promise.then()和setTimeout(func, 0)哪个先执行

### var与let区别

### 箭头函数

## CSS

### Position

stick

### 浮动、清除浮动

### 垂直居中<https://www.cnblogs.com/zhouhuan/p/vertical_center.html>

1.**绝对定位+负边距** 必须提前知道被居中块级元素的尺寸

#box {

    position: relative;

}

#child {

    height: 100px;

    position: absolute;

    top: 50%;

    margin: -50px, 0, 0, 0;

}

2.**绝对定位+transform**

#box {

    width: 300px;

    height: 300px;

    background: #ddd;

    position: relative;

}

#child {

    background: #93BC49;

    position: absolute;

**top**: **50%**;

    transform: translate(0, -50%);

}

transform中translate偏移的百分比就是相对于元素自身的尺寸而。（x轴平移，y轴平移）

**3.绝对定位+margin：auto**

#box {

    position: relative;

}

#child {

    width: 200px;

    height: 100px;

    background: #A1CCFE;

    position: absolute;

**top**: **0**;

    bottom: 0;

    margin: auto;

    line-height: 100px;

}

top和bottom设为相等的值

**4.父元素padding**

#box {

    width: 300px;

    background: #ddd;

    padding: 100px 0;

}

#child {

    width: 200px;

    height: 100px;

    background: #F7A750;

    line-height: 50px;

}

父元素是不能设置高度的，要让它自动被填充起来

line-hight  适用于单行文字的垂直居中

vertical-align 只对 table-cell 以及 inline-element 起作用

CSS表格法

父元素 display: table;

子元素：display: table-cell;vertical-align: middle;

### 两栏布局

1.absolute+margin-left

2.float+margin-left： 父元素overflow:清除浮动

3.双float：父元素overflow: auto; 右栏width: calc(100% - 140px);

4.flex

### 三列自适应

### 自适应

### 响应式

### CSS3新特性

### 盒模型，box-sizing

### 动画，有哪些属性

### 选择器优先级

### 自适应搜索框，输入框部分宽度自适应，搜索按钮宽度固定

<https://my.oschina.net/zyxchuxin/blog/496821>

### flex布局

## HTML

### HTML5新特性 <https://www.cnblogs.com/vicky1018/p/7705223.html>

**新标签**

语义化标签使得页面的内容结构化，见名知义  <footer></footer>  <nav></nav>

HTML5 拥有多个新的表单 Input 输入类型  datetime  email

新增以下表单元素

视频和音频  播放音频文件的标准，即使用 <audio> 元素  通过 video 元素来包含视频的标准方法。

Canvas - 图形

Web Storage

WebSocket

拖放

Web Worker标准，允许JavaScript脚本创建多个线程，但是子线程完全受主线程控制，且不得操作DOM

对Html语义化的理解

简述flex布局，常用于什么

## 框架

React和Vue比较<https://www.jianshu.com/p/6e124ad23c68>

相似：virtual Dom

1.模版 vs. JSX    -> 入门难度

react的思路是all in js，通过js来生成html，所以设计了jsx，还有通过js来操作css，社区的styled-component、jss等

vue是把html，css，js组合到一起，用各自的处理方式，vue有单文件组件，可以把html、css、js写到一个文件中，html提供了模板引擎来处理。

2.双向绑定 vs. 单项数据流   ->vue适用小型应用

View -> Model

都是通过事件监听实现, v-model只是语法糖

Model -> View

vue的思想是响应式的，也就是基于是数据可变的，通过对每一个属性建立Watcher来监听，当属性变化的时候，响应式的更新对应的虚拟dom。当state特别多的时候，Watcher也会很多，会导致卡顿，所以大型应用（状态特别多的）一般用react，更加可控。

vue中实现数据绑定靠的是数据劫持（Object.defineProperty()）+发布-订阅模式。在 Vue 应用中，组件的依赖是在渲染过程中自动追踪的，所以系统能精确知晓哪个组件需要被重渲染

Vue提供了优化的重新渲染，其中系统在渲染过程中跟踪依赖关系并相应地工作。依赖追踪是通过Object.defineProperty 把 data 对象的属性全部转为 getter/setter 来实现的； v-model说到底不过是 （value 的单向绑定 + onChange 事件侦听）的一个语法糖.

react在state状态管理存储数据的，不能修改数据，修改数据用Setstate。在setState之后会重新走渲染的流程

react 是单向数据流 onChange/setState()模式。在 React 应用中，当某个组件的状态发生变化时，它会以该组件为根，重新渲染整个组件子树。这可以通过shouldComponentUpdate这个生命周期方法来进行控制purerender，但Vue将此视为默认的优化。

### React生命周期  <https://blog.csdn.net/qq_29311407/article/details/79861522>

在组件初始化阶段会执行

1. constructor

2. static getDerivedStateFromProps()

3. componentWillMount() / UNSAFE\_componentWillMount()

4. render()

5. componentDidMount()

props或state的改变可能会引起组件的更新，组件重新渲染的过程中会调用以下方法：

1. componentWillReceiveProps() / UNSAFE\_componentWillReceiveProps()

2. static getDerivedStateFromProps()

3. shouldComponentUpdate()

4. componentWillUpdate() / UNSAFE\_componentWillUpdate()

5. render()

6. getSnapshotBeforeUpdate()

7. componentDidUpdate()

### React为什么要使用setState进行更改数据

<https://www.jianshu.com/p/54a8a0688197>

只修改 this.state 并不能实现页面数据的更新。 React的数据更新，不仅仅需要修改当前状态数据值，还需要驱动UI的更新使组件重新渲染，这个过程就是 this.setState 过程

### 为什么我即使没在这个js中用到React也需要引入React库

<https://github.com/rdmclin2/blog/issues/1>

### 为什么React的组件名一定要大写

### React怎样定义一个组件

### 虚拟DOM

### redux的基本原理

### 原生和框架区别

### Vue双向绑定

指的是 vue 实例中的 data 与其渲染的 DOM 元素的内容保持一致，无论谁被改变，另一方会相应的更新为相同的数据。这是通过设置属性访问器实现的。

### NextTick

### 何为MVC、MVVM <http://www.ruanyifeng.com/blog/2015/02/mvcmvp_mvvm.html>

MVC（Model-View-Controller）model->view->controller->model  接收用户指令：view或controller

MVVM：Model<->ViewMode<->View 双向绑定

### vue是怎么实现mvvm的

<http://www.cnblogs.com/keepfool/p/5619070.html>

从View侧看，ViewModel中的DOM Listeners工具会帮我们监测页面上DOM元素的变化，如果有变化，则更改Model中的数据；

从Model侧看，当我们更新Model中的数据时，Data Bindings工具会帮我们更新页面中的DOM元素。

### Express 中间件 <https://expressjs.com/zh-cn/guide/using-middleware.html>

Express 是一个路由和中间件 Web 框架，其自身只具有最低程度的功能：Express 应用程序基本上是一系列中间件函数调用。

中间件函数能够访问[请求对象](https://expressjs.com/zh-cn/4x/api.html#req) (req)、[响应对象](https://expressjs.com/zh-cn/4x/api.html#res) (res) 以及应用程序的请求/响应循环中的下一个中间件函数。下一个中间件函数通常由名为 next 的变量来表示。

### Koa

## 项目

### 项目介绍

与抓拍机配套的一系列功能和业务，管理：如设备管理，展示：实时监控，任务管理，报警，抓拍列表，以图搜图等

我负责的部分介绍

我主要负责布控&建档任务管理、一人一档（档案列表、实时抓拍、档案详情三个页面）

建档任务：选ipc，选初始库，用于小区安保

布控任务：选ipc，选布控库，主要用与公安局等

用到：页面展示：列表、分页、筛选；输入：筛选、表单；

细节：边界值处理：无数据，loading，; 容错 ||[]:避免render函数中崩溃

优化：查看详情、新增、修改用一个modal传入不同的数据；

组件抽取：易于维护和复用，比如表单抽成一个，复杂的表单组件再抽；减小文件大小；nodes抽取不同模块复用。网络操作最好放在父组件

组件编写：适用于具体项目；复用，比如，rangedate。slider，libraries

简化：模块按需引入：例子：装饰器；get／post

节点渲染（递归） 不同抓拍机细微不同，写成模块按需引入

路由实现 新增一个页面的步骤

请求数据用的什么axios

 面包屑

echarts怎么用的

utils里有什么

规范编码包括：propTypes

代码优化：

### 遇到的问题

1.getDerivedStateFromProps(nextProps,prevState)并不是props改变时才被调用，只要父组件重新渲染，这些生命周期函数就会被调用。 解决：严格控制判断条件

2.导致state状态丢失或死循环 父组件更新→子组件动作→触发父组件更新 解决：console来确定组件周期函数的实际运行情况

3.跳转至人像检索 跨域问题

### 大量重复的工作怎么处理

举例

## 计网

### TCP、UDP区别

1.TCP面向连接

因为一个应用进程开始向另一个应用进程发送数据之前，这两个进程必须先相互“握手”，即相互发送某些预备报文段，以建立确保数据传输的参数。计算机网络自顶向下方法p155

TCP协议只在端系统中运行，中间的网络元素（路由器和链路层交换机）不会维持TCP连接状态，连接状态完全保留在两个端系统中。

2.可靠数据传输

TCP在IP不可靠的尽力而为服务之上创建了一种可靠数据传输服务，确保一个进程从其接收缓存中读出的数据流逝无损坏、无间隔、非冗余、按序的数据流。与连接的另一方端系统发出的字节流是完全相同。

3.首部开销

4.拥塞控制 UDP没有拥塞机制，因此网络出现拥堵不会使源主机的发送效率降低（有利于**实时**会议视频等）

5.TCP的连接只能是点到点的,UDP支持一对一，多对一，多对多的交互通信

### 七层模型

应用层 http ftp smtp dns

运输层 TCP UDP

网络层 IP路由选择

链路层

物理层

### DNS是什么

域名系统Domain Name System。进行主机名到ip地址转换的目录服务，将用户提供的主机名解析为IP地址

### HTTP1.1和HTTP1.0的区别，改进

<https://blog.csdn.net/elifefly/article/details/3964766>

<https://blog.csdn.net/ForgotAboutGirl/article/details/6936982>

HTTP 1.0规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。

HTTP 1.1支持持久连接，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟；HTTP 1.1还允许客户端不用等待上一次请求结果返回，就可以发出下一次请求

cache新特性

### HTTPS 协议

<https://mp.weixin.qq.com/s/7rn4ruT2KemKRJRUkCwyfA>

<https://juejin.im/post/5afe23756fb9a07ac90d7080>

Https是Http的安全版，Https的原理就是：在传输层和应用层之间加了一层SSL/TLS。

SSL/TLS是一种介于传输层和应用层的协议，它是Https的精髓所在。

### https的对称加密和非对称加密

<https://zhuanlan.zhihu.com/p/34732244>

对称加密算法：发送方和接收方需要持有同一把密钥，发送消息和接收消息均使用该密钥。

-相对于非对称加密，对称加密具有更高的加解密速度，但双方都需要事先知道密钥，密钥在传输过程中可能会被窃取，因此安全性没有非对称加密高。

非对称加密算法：接收方在发送消息前需要事先生成公钥和私钥，然后将公钥发送给发送方。发送方收到公钥后，将待发送数据用公钥加密，发送给接收方。接收到收到数据后，用私钥解密。

-在这个过程中，公钥负责加密，私钥负责解密，数据在传输过程中即使被截获，攻击者由于没有私钥，因此也无法破解。非对称加密算法的加解密速度低于对称加密算法，但是安全性更高。

非对称加密算法的性能是非常低

### get和post请求的区别

GET把参数包含在URL中，POST通过request body传递参数

### Http状态码

<http://www.runoob.com/http/http-status-codes.html>

|  |  |
| --- | --- |
| 1\*\* | 信息，服务器收到请求，需要请求者继续执行操作 |
| 2\*\* | 成功，操作被成功接收并处理 |
| 3\*\* | 重定向，需要进一步的操作以完成请求 |
| 4\*\* | 客户端错误，请求包含语法错误或无法完成请求 |
| 5\*\* | 服务器错误，服务器在处理请求的过程中发生了错误 |

### 三次握手四次挥手

<https://github.com/jawil/blog/issues/14>

保证双工

为了保证服务端能收接受到客户端的信息并能做出正确的应答而进行前两次(第一次和第二次)握手，为了保证客户端能够接收到服务端的信息并能做出正确的应答而进行后两次(第二次和第三次)握手。信道不可靠, 数据传输要可靠

### xss和csrf

## 其它

### 数组去重（set）

### 取url里面的参数，作为对象的键值对

### 替换成目标字符串

尽可能考虑多的异常情况

### Git怎么用的

### webpack，模块加载时是怎么处理的，写过loader吗

### gulp

### 设计模式

### websocket连接过程

### Https怎么实现的

### 哪些设计模式

### MongoDB与SQL

事务

### 前后端分离

### 如何学习前端

### 新知识如何学