**说一下React?**

1.虚拟DOM：react也是一数据驱动的，每次数据变化React都会扫码整个虚拟DOM树，自动计算与上次虚拟DOM的差异变化，然后针对需要变化的部分实际的浏览器DOM更新。

2.组件化：react可以从功能角度和你选哪个划分，将UI分解成不同组件，各组件独立封装，整个UI是一个个小组件构成的一个大组件，每个组件只关系自身的逻辑，彼此独立。

3.单项数据流：react设计者认为数据双向绑定虽然便捷，但是复杂场景副作用也是很明显的，所以react更倾向于单项数据流，从父节点传递到子节点。

## React 的优点:

- 丰富的第三方资源,支持各种中大型项目的开发.

- 声明式渲染: {{msg}} 非声明式: document.getElementById.innerHTML = msg;

- 组件系统非常强大

- 平台化好,可以运行在各种终端,例如,PC, 手机, 原生APP, 大型终端等等

- 内部运行机制: 效率非常高.

- 虚拟DOM树, 当数据更新时, 触发差异化算法, 以最低的成本局部刷新页面.

- JSX --- 长得像HTML的JS

- 单向数据流: 数据是从顶层组件传递到子组件中

**2.说一下webpack？gulp**

webpack是现代的JavaScript应用模块打包器，它能把各种资源，如js。样式（less/sass）图片等作为模块来处理和使用。 大大提高了程序员的工作效率。

特点： 1打包工具，2.模块化识别3.编译模块代码方案用。

1.打包：可以把多个js文件打包成一个文件，减少服务器压力和下载宽带。

2.转换：把拓展语言转换成为普通的js，让浏览器顺利进行。

3.优化：具有优化和提升性能的作用。

gulp：是一个自动化构建工具。文件合并，文件压缩，愈发检查，监听文件变化。

gulp是工具链、构建工具，可以配合各种插件做js压缩，css压缩，less编译 替代手工实现自动化工作。

特点：

1.构建工具，2自动化，3.提高效率

Gulp 严格的插件指南确保插件如你期望的那样简洁高质得工作

3：SVN 和git使用方法，区别？

SVN ：集中式的代码管理工具。

缺点：1.回家后，不好工作。在公司才能访问公司的服务器。

2.上传文件的速度太慢。

git：分布式的代码管理工具。 长传和下载。

使用git得使用Linux命令：mac（终端）；

ls ：查看所有文件 -1显示一些详细信息。

mkdir 文件夹名称 ：新建文件夹；

touch 文件名称：创建文件。

rm 文件名：删除文件。clear：清空界面

步骤： 右击文件夹 --》git bash

1.git init：创建一个git仓库（在一个文件夹里创建一次即可）

2.git status：查看git仓库里面此时的文件件状态

3.git add 文件名：将文件夹里面的文件加入到缓冲区

4.git commit -m 提交的解释说明 提交到了git仓库里面

5.git checkout 文件名：从仓库里面下载某个文件

6.git log ：信息

git的三个区域：

1工作区：本地文件；2缓冲区：执行了add命令后的文件；3.仓库；

**4.http 和https？**

基本概念：

HTTP：是互联网上应用最为广泛的一种网络协议，是一个客户端和服务端请求和应答的标准（tcp），用于从www服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS ：使以安全为目标的http通道，简单讲使HTTP的安全版本，即HTTP下加入ssl层，https的安全基础是ssl，因此加密的详细内容就需要ssl。

https协议的主要作用可分两种：一个是建立一个信息安全通道，来保证数据传输安全；一个是确认网站的真实性。

区别：

1.https协议需要到ca申请证书，一般免费证书比较少，因此需要一定费用。

2.http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

3.http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443.

4.http的连接很简单，是无状态的；https协议是用ssl+http协议构建的了进行加密传输、身份认证的网络协议，比http协议安全。

**5.单线程和多线程?**

单线程：只有一个主线程的程序称作单线程程序。

主线程负责至此那个程序的所有代码（UI展现以及跟新，网络请求，本地储存等等）这些代码只能顺序执行，无法并发执行。

多线程：拥有多个线程的程序。

**解析数据时候使用:**

JSON.parse(): 将字符串 --> json对象

JSON.stringify(): 将对象 --> 字符串

**面向对象:**

原型链:

先从自己的原型找-->上一级找-->上一级-->Object-->null

obj.\_\_proto\_\_ --> obj.\_\_proto\_\_.\_\_proto\_\_-->obj.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_

设计模式: 二十三种设计模式 (单例,工厂模式,委托)

创建:

工厂模式创建对象:

function creatObj (name,age) {

var obj = new Object(); // 原料

// 加工

obj.name = name;

obj.age = age;

obj.eat = function () {}

// 出厂

return obj;

}

构造函数的方式创建对象: 方法始终重复创建

new Person()

1.自动帮你创建了一个obj

2.将obj的值赋给了this

3.return this

function Person (name,age) {

this.name = name;

this.age = age;

this.eat = function () {}

}

构造函数+原型的方式创建:

原型: 构造函数名.prototype

对象获取原型: obj.\_\_proto\_\_

function Person (name,age) {

this.name = name;

this.age = age;

}

Person.prototype.eat = function () {}

原型对象可以扩展系统的方法:

String.prototype.trim = String.prototype.trim || function () {

// this

return this.replace(/^\s+|\s+$/g,'');

}

继承:

改变this指向

call():会自动调用函数

apply():会自动调用函数

bind(): 不会调用,方法的返回值就是一个函数

function Person (name,age) {

this.name = name;

this.age = age;

}

Person.prototype.eat = function () {

}

function Worker (name,age) {

//Person.call(this,name,age);

Person.apply(this,arguments);// 继承属性

}

// 继承方法

Worker.prototype = new Person();

Worker.prototype.constructor = Worker;

**cookie session:**

cookie存储在客户端,session存储在服务器端;

cookie大小只有4kb,最多存储20个

cookie存在过期时间

不安全

必须在服务器环境下执行

**localStorage sessionStorage**

都是存储在客户端

localStorage5M,sessionStorage5M

localStorage永久存储,sessionStorage,会话关闭,就消失

不安全

不用在服务器环境下

**ajax: 原生ajax**

get和post:

1.get只能发送4kb的数据,post1个G

2.get参数是拼接url后面,post放到请求头里面

3.get的send方法不需要参数,post需要将参数放到send()里面

4.get不需要设置请求头,post需要设置请求头

**ajax兼容?**

//1.创建Ajax对象

if(window.XMLHttpRequest)

{

// 兼容所有浏览器

var oAjax=new XMLHttpRequest();

}

else

{

var oAjax=new ActiveXObject("Microsoft.XMLHTTP");// IE6

}

// 2.打开连接 是否异步

oAjax.open('GET',url,true)

//post

oAjax.open('POST',url,true);

oAjax.setRequestHeader("CONTENT-TYPE","application/x-www-form-urlencoded");

// 3.发送

oAjax.send();

// post

oAjax.send('a=10&b=20')

// 4.接受

oAjax.onreadystatechange = function () {

if (oAjax.readyState == 4) {

if (oAjax.status >= 200 && oAjax <= 299 || oAjax.status == 304) {

// 执行成功的回调函数

} else {

// 执行失败的回调函数

}

}

}

如何用ajax实现跨域请求数据?

不能实现跨域,使用jsonp实现跨域

在script中,写src属性,将接口放入src中,得自己手动设置回调函数

1.先拿接口往浏览器里面测试

2.手动创建script,赋值src

3.设置全局的函数

4.用完即扔(从head里面移除)

$.ajax({

type: 'get post',

url: 接口,

data: 参数,

success: fn,

error: fn,

cache: false,缓存

async: true 异步,

timeout: 超时

complete: fn

})

jsonp:

$.ajax({

dataType: 'jsonp',

url:,

data:,

success: function(str) {

var data = JSON.parse(str);

// data = {list: [{},{},{},{}]};

// 渲染页面

}

});

angular的四大特性:

1.双向数据绑定

2.依赖注入: 形参位置不固定,但是形参的名字是固定的 (java)

3.指令机制

4.MVC 或者是 MVVM

angular使用了哪些机制?

1.脏检查

2.指令机制

3.路由机制

4.消息机制 (父子控制器的通信) $emit $on

MVC和MVVM的区别就是：

1.MVC是单向的数据绑定，MVVM是双向数据绑定。

2.MVVM是MVC衍生而来，而MVC是从后台而来的。

3.MVVM的js文件是外部引入而来的，修改时也比较方便。提高了用户体验，增快网页加载速度。

**请写出几个常见的排序算法?**

冒泡: 从第一个数开始,比较后面的每一个数据,如果第一个数大于后面的数,则调换位置

选择排序: 从当前下标开始,找从该下标+1以后的数的最小值,然后调换位置

归并排序(正宗二分法): 采用二分法,左边一个排序好的数组,右边一个排序好的数组,每次比较左右第一个数,小的,放到一个新的数组

快速排序(二分法): 采用二分法,取出中间数数组,也有左右两边的数组,和中间数组值比较,小的放在左边,大的放到右边,最后连接左中右三个数组

**常见兼容性问题:**

* png24 位的图片在ie6浏览器上出现背景,解决方案是做成PNG8.
* 浏览器默认的margin和padding不同.解决方案是加一个全局的\*{margin:0;padding:0}来统一
* IE6 上边距bug:块属性标签float后,又又横行的margin情况下,在ie6显示margin比设置的大
* 浮动ie产生的双倍距离#box{float:left;width:10px:margin:0 0 0 100px} 这种情况下IE会产生20px的距离,解决方案是在float的标签样式控制中加入--\_display:inline;将其转化为行内属性(\_只在ie6会识别
* Ie,even对象有x,y属性,但是没有pageX,pageY属性;Firefox下,event对象有pageX,pageY,单没有x,y

解决方法(条件注释)缺点是在IE浏览器下可能会增加额外的HTTP请求数.

* 超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了解决防范是改变css属性的排列顺序:L-V-H-A:a:link{} a:visited{} a:hover{} a:active{}

**HTTP协议**

http协议是一个属于应用层的面相对象的协议,由于其简捷,快捷的方式,适用于分布式超媒体信息系统.

主要特点: 1.支持客户/服务器模式.

2.简单快速3.灵活4.无连接5.无状态协议

**缺点及解决方法**: 因为传统的HTTP明文传输模式，存在着大量的灰色中间环节，明文信息在中间代理服务器、路由器、WIFI热点、通信服务运营商等层层节点中转，每一层都有可能导致传输数据被窃取或被第三方篡改，HTTP协议已经被认为是非常不安全的传输协议。而HTTPS加密协议，在用户浏览器之间建立SSL加密通道，将数据加密后传输，大降低了被第三方窃取或篡改的风险。现在，绝大多数网民、搜索引擎企业和浏览器企业都已经纷提出弃用HTTP的计划，提倡全网HTTPS化

* **移动端性能优化**

1. 尽量使用css3动画，可以用transform: translateZ(0)来开启硬件加速。
2. 适当使用touch事件代替click事件。
3. 避免使用css3渐变阴影效果。
4. 不滥用Float。Float在渲染时计算量比较大，尽量减少使用
5. 不滥用Web字体。Web字体需要下载，解析，重绘当前页面，尽量减少使用。
6. 合理使用requestAnimationFrame动画代替setTimeout
7. CSS中的属性（CSS3 transitions、CSS3 3D transforms、Opacity、Canvas、WebGL、Video）会触发GPU渲染，请合理使用。过渡使用会引发手机耗电增加
8. PC端的在移动端同样适用

网页重构

概述：

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：

表格(table)布局改为DIV+CSS

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)

针对于SEO进行优化

深层次的网站重构应该考虑的方面：

减少代码间的耦合

1. 让代码保持弹性
2. 严格按规范编写代码
3. 设计可扩展的API
4. 代替旧有的框架、语言
5. 增强用户体验
6. 通常来说对于速度的优化也包含在重构中
7. 压缩JS、CSS、image等前端资源(通常是由服务器来解决)
8. 程序的性能优化(如数据读写)
9. 采用CDN来加速资源加载
10. 对于JS DOM的优化
11. HTTP服务器的文件缓存

* **浏览器内核与兼容性**

.内核类别

IE： trident内核

 Firefox： gecko内核

 Safari： webkit内核

 Opera： 以前是presto内核，Opera现已改用Google Chrome的Blink内核

 Chrome： Blink(基于webkit，Google与Opera Software共同开发

* **什么样的前端代码是好的?**

高复用 低耦合 好维护 好扩展

**面相对象:**

三大特性是：封装,继承,多态

所谓封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。封装是面向对象的特征之一，是对象和类概念的主要特性。 简单的说，一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体。在一个对象内部，某些代码或某些数据可以是私有的，不能被外界访问。通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分意外的改变或错误的使用了对象的私有部分。

所谓继承是指可以让某个类型的对象获得另一个类型的对象的属性的方法。它支持按级分类的概念。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。 通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程，就是从一般到特殊的过程。要实现继承，可以通过“继承”（Inheritance）和“组合”（Composition）来实现。继承概念的实现方式有二类：实现继承与接口继承。实现继承是指直接使用基类的属性和方法而无需额外编码的能力；接口继承是指仅使用属性和方法的名称、但是子类必须提供实现的能力；

所谓多态就是指一个类实例的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，它们（那些操作）可以通过相同的方式予以调用。

五大基本原则   
单一职责原则SRP(Single Responsibility Principle)  
是指一个类的功能要单一，不能包罗万象。如同一个人一样，分配的工作不能太多，否则一天到晚虽然忙忙碌碌的，但效率却高不起来。  
  
开放封闭原则OCP(Open－Close Principle)   
一个模块在扩展性方面应该是开放的而在更改性方面应该是封闭的。比如：一个网络模块，原来只服务端功能，而现在要加入客户端功能，  
那么应当在不用修改服务端功能代码的前提下，就能够增加客户端功能的实现代码，这要求在设计之初，就应当将服务端和客户端分开，公共部分抽象出来。  
  
替换原则(the Liskov Substitution Principle LSP)   
子类应当可以替换父类并出现在父类能够出现的任何地方。比如：公司搞年度晚会，所有员工可以参加抽奖，那么不管是老员工还是新员工，  
也不管是总部员工还是外派员工，都应当可以参加抽奖，否则这公司就不和谐了。  
  
依赖原则(the Dependency Inversion Principle DIP) 具体依赖抽象，上层依赖下层。假设B是较A低的模块，但B需要使用到A的功能，  
这个时候，B不应当直接使用A中的具体类： 而应当由B定义一抽象接口，并由A来实现这个抽象接口，B只使用这个抽象接口：这样就达到  
了依赖倒置的目的，B也解除了对A的依赖，反过来是A依赖于B定义的抽象接口。通过上层模块难以避免依赖下层模块，假如B也直接依赖A的实现，那么就可能造成循环依赖。一个常见的问题就是编译A模块时需要直接包含到B模块的cpp文件，而编译B时同样要直接包含到A的cpp文件。

接口分离原则(the Interface Segregation Principle ISP)   
模块间要通过抽象接口隔离开，而不是通过具体的类强耦合起来

浏览器兼容性问题：

1. 各浏览器在没有给定标签样式时的margin和padding的差异。

解决方法：使用通配符\*｛margin：0；padding：0｝

2.块级属性float之后又有横行的margin时，在IE6中margin值比较大。

常见症状是IE6中后面的一块被顶到下一行

解决方法：在float的标签样式控制中使用display：inline：将其改成行内标签

3.行内属性标签，设置display:block后采用float布局，又有横行的margin？IE6艰巨bug

解决方法：display：block后面加入display：inline；display：table。

4.设置较小高度标签（一般小于10px），在IE6，7 傲游中高度超出自己设置的高度

解决方法：给超出的高度标签设置overflow：hidden；或者设置行高line-hength系哦啊与设置的高度。

5.图片默认有间距：

解决方法：使用float属性为img布局

IE盒子模型 w3c标准盒子模型

盒子模型 box-sizing 的属性值： content-box border-box inherit

Ajax 封装：

If(window.XMLHttprequest){

Var oAjax =new XMLHttprequest()

}else{

var oAjax =new ActiveXObiect(“Microsoft.XMLHTTP”)

}

2.打开链接 是否异步

oAjax.open(“GET”,url,true)

3.发送

oAjax.send();

4.接受

oAjax.onreadystatechange =function (){成功的函数}else｛失败的函数｝

vue全家桶

vuex，vue.cli vue.axios vue.router es6 sass

模版字符串 ：``;

箭头函数 =》

结构赋值 ：【】 ｛｝；

1. 提高开发效率，减少体力耀东

使用剪头[函数](https://www.baidu.com/s?wd=%E5%87%BD%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHbzujDLn1IhPvn1uyuW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWcsPjcLnWm1P1msPjn1rjRz)不需要敲完整的 function 关键字， 同时如果只有行 return 语句的[函数](https://www.baidu.com/s?wd=%E5%87%BD%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHbzujDLn1IhPvn1uyuW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWcsPjcLnWm1P1msPjn1rjRz" \t "_blank)，还可以进一步简写：

例如 要定义一个 trim [函数](https://www.baidu.com/s?wd=%E5%87%BD%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHbzujDLn1IhPvn1uyuW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWcsPjcLnWm1P1msPjn1rjRz)，不使用箭头函数：

        const trim = function( str ) {

            return trim.replace( /^\s+|\s+$/g, '' );

        };

        使用箭头函数：

        const trim = str => trim.replace(  /^\s+|\s+$/g, '' );

2. 在函数内部不需要自己的 this 指针的时候，非常方便，因为箭头函数作用域内没有 this

    例如下面不使用箭头函数的代码， 要通过将 this 赋值给 me，调用 me 来调用 Obj：

    const Obj = {

        text : 'ABC',

        replace : function( arr ) {

            var me = this;

            arr.forEach( function( item ) {

                return me.text;

            } );

        }

    };

    使用箭头函数：

    const Obj = {

        text : 'ABC',

        replace : function( arr ) {

            arr.forEach( item => this.text );

        }

    };

还有一点是 箭头函数没有 arguments 变量，在某些时候也可以带来方便，和上面差不多。

**Git的使用步骤**：

**第一次建立的时候：**

1.git init 创建仓库

2.git status 查看仓库里面的一个文件状态

3.git add 文件名（一般是各.点，代表全部文件）添加文件到缓冲区

4.git commit –m “提交的解释说明” 提交到git仓库里面

5.git checkout 文件名：从仓库里面下载某个文件

6.git log ：信息

**步骤2：真正准备工作：**

1.本地生成一个密钥：ssh-keygen -t rsa –C ‘登录的邮箱地址’；直接回车

2.找密钥 c/Users/Administrator/.ssh

3.把密钥放在github进行绑定

4.配置：git config –global user.email “邮箱名”

5.配置：git config –global user.name “github的网名”

6.查看此时的配置文件：git config –l

**步骤3:创建连接：**

1. git init
2. git add .
3. git commit –m “first commit” (添加说明)
4. git remote add origin github里面的那个网址
5. git push –u origin master

**前后端分离的意思是**，前后端只通过 JSON 来交流，组件化、工程化不需要依赖后端去实现。前后端分离意味着，前后端之间使用 JSON 来交流，两个开发团队之间使用 API 作为契约进行交互。从此，后台选用的技术栈不影响前台。当后台开发人员选择 Java 的时候，我可以不用 JSP 来编写前端页面，继续使用我的 React 又或者 Angular。而我使用 React 时，也不影响后台使用某一个框架。