WEB前端开发面试题总结

***百度***

1. AJAX流程

step1. 创建XMLHttpRequest对象（异步调用对象）；

Step2. 创建http请求，并指定http请求的方法、url以及验证信息；

Step3.设置响应http状态变化的函数；

Step4. 发送http请求；

Step5. 获取返回数据；

Step6. 使用js和dom实现局部刷新；

1. promise简单说一下
2. ，Promise是ES6的新增的一个构造函数，用来执行一些异步操作。本身有all、reject、resolve等方法，原型上有then、catch等方法。 
3. 手写一个箭头函数

Eg1:

var fn = ()=> alert(‘箭头函数’)

Eg2:

(x,y)=>{

var z=x\*y;

Return z;

}

Eg3: 箭头函数中的this指向 定义它时,它所处的对象，而不是执行时的对象

普通函数中的this指向 执行它时，调用它的对象

* **var** obj={
* num:3,
* fn:**function**(){
* setTimeout(**function**(){

* console.log(**this**.num);
* });
* }
* }
* obj.fn();//undefined
* // 普通函数中的this出现在全局函数setTImeout()中的匿名函数里，并没有某个对象进行显示调用，所以this指向window对象
* //............................................................
* **var** obj1={
* num:4,
* fn:**function**(){
* setTimeout(() => {
* console.log(**this**.num);
* });
* }
* }
* obj1.fn();//4
* //箭头函数的this绑定看的是this所在的函数定义在哪个对象下，绑定到哪个对象则this就指向哪个对象

Eg4:

* **var** obj1={
* num:4,
* fn:**function**(){
* **var** f=**function**(){
* console.log(**this**); //window,因为函数f定义后并没有对象调用，this直接绑定到最外层的window对象
* setTimeout(() => {
* console.log(**this**);//window，外层this绑定到了window,内层也相当于定义在window层（全局环境）
* });
* }
* f();
* }
* }
* obj1.fn();

Eg5:

* **var** obj1={
* num:4,
* fn:**function**(){
* **var** f=() => {
* console.log(**this**); //object,f()定义在obj1对象中，this就指向obj1,这就是箭头函数this指向的关键
* setTimeout(**function**() {
* console.log(**this**);//window，非箭头函数的情况下还是要看宿主对象是谁，如果没有被对象调用，函数体中的this就绑定的window上
* });
* }
* f();
* }
* }
* obj1.fn();

1. 链式调用

案例1：

function show(str) {

console.log(str);

return show;

}

show(123)(456)(789);

1. .简单的观察者模式

观察者模式（Observer Pattern），也被称为“发布/订阅模型（publisher/subscriber model）”。在这种模式中，有两类对象，分别是“观察者-Observer”和“目标对象-Subject”。**当目标对象的状态发生改变的时候就主动向观察者发出通知（调用观察者提供的方法）**。

Eg1:

***Dom中的观察者模式***

例如常用的一些onclick、 attachEvent 、addEventListener 都是观察者模式的具体应用。

document.body.onclick = function(){

alert('我是一个观察者，你一点击，我就知道了');

}

body是发布者，即目标对象，当被点击的时候，向观察者反馈这一事件

***Js观察者模式（订阅/发布）(案例见百度云web前端---疑难问题--js观察者模式)***

1. let const var区别

Var 变量提升 重复定义相同的变量会覆盖 作用域是语句所在的函数内

Let 变量不会提升 重复定义相同的变量会报错 作用域是该语句所在的代码块内

Const 定义常量,一旦定义无法修改

Var 与let作用域详解案例：

Eg1:

**var** a = [];

**for** (**var** i = 0; i < 10; i++) {

a[i] = **function** () {

console.log(i);

};

}

a[6](); // 10

Eg2:

var a = [];

for (let i = 0; i < 10; i++) {

a[i] = function () {

console.log(i);

};

}

a[6](); // 6

Eg3

Var a = [];

for (var i=0;i<10;i++) {

a[i]=(function(i){

return function() {

console.log(i);

})(i);

}

a[6](); // 6

循环变量i传到一个立即执行函数里，变成了局部变量i（其实这个函数的参数可以任意取名，取成i为了加深你的理解），在这个局部变量的作用域内定义了一个函数引用了它，js就会把它作为函数的上下文保存起来，所以console.log得到的是0到9不同的值。

1. 数组去重

// 方法一 遍历删除相同元素

function norepeat(arr){

for(var i=0;i<arr.length;i++){

for(var j=i+1;j<arr.length;){

if(arr[i]==arr[j]){

arr.splice(j,1);

}

j++;

}

}

return arr;

}

var arr2=[11,6,5,4,8,6,1,15,4,6];

var arr1=norepeat(arr2);

console.log(arr1);

// 方法二 indexOf返回元素在数组中第一次出现的位置;

// push向数组添加新元素;

function norepeat1(arr){

var arr1=[];

for(var i=0;i<arr.length;i++){

if(arr1.indexOf(arr[i])==-1){

arr1.push(arr[i]);

}

}

return arr1;

}

var arr2=[11,6,5,4,8,6,1,15,4,6];

console.log(norepeat1(arr2));

方法三：

思路：先将原数组排序，在与相邻的进行比较，如果不同则存入新数组

function unique(arr){

　　var arr2 = arr.sort();

　　var res = [arr2[0]];

　　for(var i=1;i<arr2.length;i++){

　　　　if(arr2[i] !== res[res.length-1]){

　　　　　　res.push(arr2[i]);

　　　　}

　　}

　　return res;

}

1. 判断数据类型，null怎么判断

<!-- 方法一 原理：排除法（非真的四种情况：null undefined 0 ''）-->

<!-- typeof(exp)!="undefined" 排除 ''; -->

<!-- exp!=0 排除 0; -->

<!-- !exp 排除 真的; -->

var exp = null;

if (!exp && typeof(exp)!="undefined" && exp!=0)

{

alert("is null");

}

1. 目前主流的架构模式： .MVC、MVP、MVVM （Model View Controller）

**简介MVC：**

* 视图（View）：用户界面
* 控制器（Controller）：业务逻辑
* 模型（Model）：数据保存



互动过程：

* View 传送指令到 Controller
* Controller 完成业务逻辑后，要求 Model 改变状态
* Model 将新的数据发送到 View，用户得到反馈

注意： 所有过程都是单向的

**简介MVP （**Presenter**）**

MVP 模式将 Controller 改名为 Presenter，同时改变了通信方向。



1. 各部分之间的通信，都是双向的。

2. View 与 Model 不发生联系，都通过 Presenter 传递。

**简介MVVM**

MVVM 模式将 Presenter 改名为 ViewModel，基本上与 MVP 模式完全一致。



唯一的区别是，它采用双向绑定（data-binding）：View的变动，自动反映在 ViewModel，

1. TopK用的什么排序

问题描述：有 N (N>1000000)个数,求出其中的前K个最小的数（又被称作topK问题）。

**思路1：**最基本的思路，将N个数进行完全排序，从中选出排在前K的元素即为所求。

**思路2：**可以采用数据池的思想，选择其中前K个数作为数据池，

后面的N-K个数与这K个数进行比较，若小于其中的任何一个数，则进行替换。这种方法能够优化算法，提高效率。

1. 闭包

广义定义：有权访问另一个函数 作用域 内变量的函数都是闭包。

本质：闭包就是一个函数引用另外一个函数的变量，因为变量被引用着所以不会被回收，因此可以用来封装一个私有变量。

案例：

|  |  |
| --- | --- |
|  | 闭包案例：  function a(){  Var n = 0;  function inc(){  n++;  console.log(n)  };  return inc;  }  Var c = a();  C(); // 1  C(); // 2 |

Vue常见面试题

1. **active-class是哪个组件的属性？嵌套路由怎么定义？**  
   答：vue-router模块的router-link组件。
2. **怎么定义vue-router的动态路由？怎么获取传过来的动态参数？**

首先，在路径path里边所需要传递参数的地方，前面加上： 例如path属性上加上new/:id/:txt 在router-link里边写路径时，参数位置添加相应的参数值 ，例如： new/001/china

最后，在跳转后的组件里边使用$router.params 获取到传递的参数。例如：$router.params ------>{“id”:001,”txt”:”china”};

1. **vue-router有哪几种导航钩子？**

三种：

一种是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。

第二种：组件内的钩子；

第三种：单独路由独享组件

1. **scss是什么？安装使用的步骤是？有哪几大特性？**

答：css预处理器.

特性：1，可以用变量，例如（$变量名称=值）；

2、可以用混合器，例如（）

3、可以嵌套

**5、mint-ui是什么？怎么使用？说出至少三个组件使用方法？**

基于vue的移动端组件库

npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from ‘mint-ui’。

组件一：Toast(‘登录成功’)；组件二：mint-header；组件三：mint-swiper

**6、v-model是什么？怎么使用？ vue中标签怎么绑定事件？**

答：可以实现双向绑定，指令（v-class、v-for、v-if、v-show、v-on）。vue的model层的data属性。

1. **axios是什么？怎么使用？描述使用它实现登录功能的流程？**

**答**：请求后台资源的模块。

npm install axios -S装好，

在main.js中引入进来 import axios from 'axios'

将axious改写为Vue的原型属性，方便在其他组件中引用 Vue.prototype.$ajax = axios

Eg:

methods: {

submitForm () {

this.$ajax({

**method: 'post',**

**url: '/user',**

**data: {**

**name: 'wise',**

**info: 'wrong'**

**}**

}).then（function(res){

}).catch(function(err){

})

}

js中使用import进来，然后.get或.post。返回在.then函数中如果成功，失败则是在.catch函数中

**8、vuex是什么？怎么使用？哪种功能场景使用它？**

答：vue框架中状态管理。

在main.js引入store，注入。新建了一个目录store，….. export 。

场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

**9、mvvm框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？**

答：一个model+view+viewModel框架，数据模型model，viewModel连接两个

区别：vue数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

**10、vuex是什么？怎么使用？哪种功能场景使用它？**

答：vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，….. export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

**11、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？**

答：全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称另外一个是函数。组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding

Eg:

**Html:**

<div v-color='colorStatus'>一个普通的div元素</div>

**Js:**

Vue.directive('color',function(el,binding){

//指令名称注意不要加\*\*v-\*\*

console.log(el);

//当前绑定自定义指令的元素，可以用来直接操作DOM

console.log(binding);

//一些参数，常用的 => binding.value(指令的值)

el.style.backgroundColor = binding.value;

})

**12、说出至少4种vue当中的指令和它的用法？**

答：v-if：判断是否隐藏；

v-for：数据循环出来；

v-bind:class：绑定一个属性；

v-model：实现双向绑定

**13、vue-router是什么？它有哪些组件？**

答：vue用来写路由一个插件。router-link、router-view

**14、Vue的双向数据绑定原理是什么？**

vue数据双向绑定是通过数据劫持结合发布者-订阅者模式的方式来实现的.

vue是通过Object.defineProperty()来实现数据劫持的。

Eg:

var Book = {}

var name = '';

Object.defineProperty(Book, 'name', {

set: function (value) {

name = value;

console.log('你取了一个书名叫做' + value);

},

get: function () {

return '《' + name + '》'

}

})

Book.name = 'vue权威指南'; // 你取了一个书名叫做vue权威指南

console.log(Book.name); // 《vue权威指南》

详细解析：

我们通过Object.defineProperty( )设置了对象Book的name属性，对其get和set进行重写操作，顾名思义，get就是在读取name属性这个值触发的函数，set就是在设置name属性这个值触发的函数，所以当执行 Book.name = 'vue权威指南' 这个语句时，控制台会打印出 "你取了一个书名叫做vue权威指南"，紧接着，当读取这个属性时，就会输出 "《vue权威指南》"，因为我们在get函数里面对该值做了加工了。

**15.请详细说下你对vue生命周期的理解？**

答：总共分为8个阶段创建前/后，挂载前/后，更新前/后，销毁前/后。

**创建前/后：** 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。

在created阶段，vue实例的数据对象data有了，$el还没有。

**挂载前/后：**在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点。

在mounted阶段，vue实例挂载完成，data.message成功渲染。

**更新前/后：**当data变化时，会触发beforeUpdate和updated方法。

**销毁前/后：**在执行destroy方法后，对data的改变不会再触发周期函数，

说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

**18、请说下封装 vue 组件的过程？**

答：首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：**效率低**、**难维护**、**复用性**等问题。

首先，使用**Vue.extend方法创建一个组件**，

然后，使用**Vue.component方法注册组件**。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

1. **你是怎么认识vuex的？**

一种状态管理器，集中管理一些公用的数据，方法；

一般分为四个：state(保存数据)，mutations(保存方法的)，getter(对数据进行筛选加工的)，action(异步操作)

**20、vue-loader是什么？使用它的用途有哪些？**

**什么**：解析.vue格式文件的一个加载器，每个.vue文件都包含最基本的template, script, style三个语言块，vue-loader就是简析每个语言块的；

**用途**： 1, js可以使用es6语法；

2, style样式可以使用scss或者less

3, template中的html可以使用jade

(jade，编译html,比如：传统的HTML标签写尖括号很麻烦，Jade里可以省略尖括号，直接写标签名。标签间的嵌套关系用换行加空格来实现。紧接在标签名后加上.xx或#xx，就能给标签添加css类名和id。标签名后第一个空格后面的内容会被编译成标签内的文本内容)

例子：

<h1 id="titleID" class="titleClass">My First Jade Page</h1>

编译成

h1.titleClass#titleID My First Jade Page

扩展：

vue-loader通过指定语言块的lang属性支持css预编译、html编译模板等语言格式。

例如：

<style>

.classA{

color: “red”

}

</style>

使用less编译：

<style lang=”less”>

.classA

color: “red”

</style>

**21, 请说出vue.cli项目中src目录每个文件夹和文件的用法？**

**a**ssets：存放静态资源的；

Components: 放组件的；

Router: 定义路由配置的；

Main.js : 入口文件；

App.vue ： 一个最外层的组件；（主组件）

1. **vue.cli中怎样使用自定义的组件？有遇到过哪些问题吗？**

**第一步**，在components目录下新建一个vue文件，比如：homeButton.vue;

**第二步**，在需要的页面导入

import homeButton from‘./components/homeButton.vue’

**第三步**，注入到需要引用的页面的components里面

**components:{ homeButton }**

**第四步**，在template里使用；<home-button></home-button>

**问题**： 组件使用驼峰命名时（homeButton）,使用时要用还原，用-相连（home-button）

23, 为什么要选vue？与其它框架对比的优势和劣势？

Vue特征： 1，轻量级别的数据驱动框架；

2，双向数据绑定；

3，指令；

4，组件化；

Vue和angular比较：

同：都支持双向绑定、 指令 、 过滤器 、 不支持低端浏览器；

异：1，二者虽然都实现了双向绑定；但其原理不同；

**Angular**是利用数据脏检测，当model发生变化，会检测所有视图，查看是否绑定了相关数据，从而再更新视图；

**vue**则是采用了es5中的set和get方法，结合发布订阅模式来实现双向绑定；当视图发生变化时，由于采用的是发布订阅模式，故能够快速定位，从而更新视图。

2，想比较angular而言，vue更容易上手；

Vue和react比较：

同：二者都使用了 虚拟DOM，都具有可组合可复用的组件。

异：vue比较简单，容易学习。代码量相比react也会少。

1. 前端发展历史和框架

**前端发展历史背景**

上个世纪的1989年，欧洲核子研究中心的物理学家Tim Berners-Lee发明了超文本标记语言（HyperText Markup Language），简称HTML；最早的HTML页面是完全静态的网页。

浏览器请求某个URL时，Web服务器把对应的html文件扔给浏览器，就可以显示html文件的内容了。

但是如果要针对不同的用户显示不同的页面，显然不可能给成千上万的用户准备好成千上万的不同的html文件，这时候人们利用php,jsp，asp等创建动态的html;

但是，一旦浏览器显示了一个HTML页面，要更新页面内容，唯一的方法就是重新向服务器获取一份新的HTML内容。如果浏览器想要自己修改HTML页面的内容，怎么办？那就需要等到1995年年底，Java被引入到浏览器。

有了Java后，浏览器就可以运行Java，然后，对页面进行一些修改。Java还可以通过修改HTML的DOM结构和CSS来实现一些动画效果，而这些功能没法通过服务器完成，必须在浏览器实现。

**前端发展阶段**

**【第一阶段】 浏览器简析原生JS直接操作DOM ;**

**【第二阶段】 jQuery时代。对原生js进行封装，精简js命令。**

**【第三阶段】MVC模式，需要服务器端配合。（Java可以在前端修改服务器渲染后的数据）**

其最大特点就是所有通信都是单向的：提交一次反馈一次，比如提交表单：

填写内容 → 点击提交 →处理业务逻辑 →存入数据库 → 刷新页面→服务器取数据库数据→渲染到客户端页面→ 展示上一次你提交的内容



处理过程:

View 传送指令到 Controller

Controller 完成业务逻辑后，要求 Model 改变状态

Model 将新的数据发送到 View，用户得到反馈

Mvc模式的缺点：

缺点一：页面的更新修改必须等待服务器端的指示，也就是服务器端发送过来的数据。比如说html节点、数据、页面结构等。浏览器只需要直接简析就行。前端几乎没有什么工作。

缺点二：牵一发而动全身。数据、显示不分离！比如说：一个表单中，年龄这个字段我不需要了，我就得从新发送一次请求，然后后台把控制年龄字段的相应命令去掉，再发送给前端接受，前端接受后再渲染一遍；

**【第四阶段】 MVVM框架模式（数据驱动），数据和视图分离**

流行的mvvm框架：angular.js vue.js react

所谓的数据视图分离，数据驱动视图， 视图不影响数据，再也不用管繁琐的DOM结构操作了，世界顿时清净，完美！

MVVM最早由微软提出来，在前端页面中，把Model用纯Java对象表示，View负责显示，两者做到了最大限度的分离，也就是常说的前后端分离。



把Model和View关联起来的就是ViewModel。ViewModel负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model。

数据驱动模式的精髓在于【数据】和【视图】分离，所以我们首先并不关心DOM结构，而是关心数据的展现。

最简单的数据存储方式是什么呢？显然不是mysql、数据库而是使用Java对象。

视图的变更，不需要再操作DOM节点，而是只需要修改相应数据就行；

1. 混合app开发

定义：混合开发的App（Hybrid App）就是在一个App中内嵌一个轻量级的浏览器，一部分原生的功能改为Html 5来开发，这部分功能不仅能够在不升级App的情况下动态更新，而且可以在Android或iOS的App上同时运行，让用户的体验更好又可以节省开发的资源。

**一个Hybrid开发的App中必须要要有的功能就是Html 5页面和Native App怎么进行交互。**比如，我点了一个Html 5页面上的一个按钮或链接，我能不能够跳转到Native App的某个页面。又比如说我点了Html 5页面上的分享按钮，我能不能调用Native App的分享功能。

**Html5和Native App的交互技术**：WebView本来就支持js和Java相互调用，你只需要开启WebView的JavaScript脚本执行，然后通过代码mWebView.addJavascriptInterface(new JsBridge(), "bxbxbai");向Html 5页面时注入一个Java对象，然后就可以在Html 5页面中调用Native的功能了。

*参考链接：* [*http://blog.csdn.net/liuxu841911548/article/details/54906660*](http://blog.csdn.net/liuxu841911548/article/details/54906660)

1. 项目管理规范

代码管理：

《1》代码书写规范（注意空格），命名尽量要有意义；

《2》勤注释；

《3》使用频率比较高的代码要抽取和封装；

《4》视图层，逻辑层，数据尽量分开存放；

1. 浏览器的渲染过程

《1》首先获取html, 然后构建dom树；

《2》其次根据css构建render树，render树中不包含定位和几何信息；

《3》最后构建布局，即定位和几何信息；

1. 重构、回流

浏览器重构指的是 改变每个元素外观时所触发的浏览器行为；比如说：颜色，背景；（重构不会引发页面的重新布局）

回流指的是 为了重新渲染页面的需要， 进行的重新计算元素的大小和位置；

**JavaScript部分面试题**

**（1）JavaScript的数据类型**

基本数据类型（值类型）：Number，String，Boolean，Undefined，Null

复杂数据类型（引用类型）：Object，Array，Function，RegExp，Date，Error

全局数据类型：Math

注释：对引用类型的复制其实是引用复制，相当于复制着地址，对象并没有真正的复制。

**（2）JavaScript的闭包**

闭包简单的说就是一个函数能访问外部函数的变量，这就是闭包。



闭包的另一种作用是隔离作用域：



**闭包的缺点是： 因为内部闭包函数可以访问外部函数的变量，所以外部函数的变量不能被释放，如果闭包嵌套过多，会导致内存占用大，**

**（3）new 操作符到底做了什么**

首先，创建一个新的空对象；

其次，为这个空对象添加属性；（构造函数）

最后，将this指向这个对象；

**（4）改变函数内部this指针的指向函数**

假设要改变fn函数内部的this的指向，指向obj，那么可以fn.call(obj);或者fn.apply(obj);

两个的第一个参数都是一样的，表示调用该函数的对象；

apply的第二个参数是数组形式。eg:[arg1,arg2,arg3];

call的第二个参数是arg1,arg2,arg3这样的形式。

**（5）JavaScript的继承**

原型继承，

B.prototype = new A();

构造函数继承，

function B(age,name){

this.age=age;

this.name=name;

}

var foo=new B(18,"wmy");

foo.name; //wmy

foo.age; //18

原型+构造函数继承

**（6）JavaScript的变量提升**



**（7）JavaScript事件模型**

原始事件模型，捕获型事件模型，冒泡事件模型，

原始事件模型就是ele.onclick=function(){}这种类型的事件模型

冒泡事件模型是指事件从事件的发生地（目标元素），一直向上传递，直到document。

捕获型则恰好相反，事件是从document向下传递，直到事件的发生地（目标元素）

IE是只支持冒泡事件模型的。

**（8）内存泄漏**

内存泄漏指的是浏览器不能正常的回收内存的现象

**（9）同源策略**

所谓同源是指，域名，协议，端口相同。

**（10）跨域的几种方式**

jsonp（利用script标签的跨域能力）跨域、

websocket（html5的新特性，是一种新协议）跨域、

iframe跨域、

postMessage(包含iframe的页面向iframe传递消息)

**（11）异步和同步**

同步指下一个程序的执行需要等到上一个程序执行完毕，也就是得出结果后下一个才能执行，

异步指的是上一个程序指向后，下一个程序不用等到上一个程序出结果就能执行，等上一个出结果了调用回调函数处理结果就好。

**（12）优化下面代码**

这里的优化主要是对加号操作符的优化，因为加号在JavaScript中非常耗时和耗内存。



**（13）封装cookie的添加，删除，查询方法**

cookie是存储在浏览器端的，可以用于存储sessionID，也可以用于自动登陆，记住密码等，但是在浏览器端并没有官方的操作cookie的方法，下面我们来封装一下：

CookieUtil=｛

addCookie:function(key,value,options){

var str=key+"="+escape(value);

if(options.expires){

var curr=new Date(); //options.expires的单位是小时

curr.setTime(curr.getTime()+options.expires\*3600\* 1000);

options.expires=curr.toGMTString();

}

for(var k in options){ //有可能指定了cookie的path，cookie的domain

str+=";"+k+"="+options[k];

}

document.cookie=str;

},

queryCookie:function(key){

var cookies=document.cookie;

//获得浏览器端存储的cookie,格式是key=value;key=value;key=value

cookies+=";";

var start=cookies.indexOf(key);

if(start<=-1){ return null; } //说明不存在该cookie

var end=cookies.indexOf(";",start);

var value=cookies.slice(start+key.length+1,end);

return unescape(value);

},

deleteCookie:function(key){

var value=CookieUtil.queryCookie(key);

if(value===null){return false;}

CookieUtil.addCookie(key,value,{expires:0});//把过期时间设置为0，浏览器会马上自动帮我们删除cookie

}

｝

**（14）事件委托机制**

事件委托指的是，不再事件的发生地设立监听函数，而是在事件发生地的父元素或者祖先元素设置监听器函数，这样可以大大提高性能，因为可以减少绑定事件的元素，比如：

<ul>

<li></li>

<li></li>

<li></li>

</ul>

要给li元素绑定click事件，使用事件委托机制的话，就只需要给ul绑定click事件就行了，这样就不需要给每个li'绑定click事件，减小内存占用，提高效率

**其他部分**

**（1）http状态码**

http状态码是表示服务器对请求的响应状态，主要分为以下几个部分

1\*\*：这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束

2\*\*：表示请求成功，

3\*\*：表示重定向

4\*\*：表示客户端错误

5\*\*：表示服务器端错误

具体如下：

200（OK），表示请求成功；

202（Accepted），服务器已接受请求，但尚未处理。

204（No-Content），服务器成功处理了请求，但不需要返回任何实体内容

301（Moved-Permanently），永久性重定向

302（Moved-Temporarily），暂时性重定向

400（Bad-Reques），请求有误，当前请求无法被服务器理解。

401（Unauthorized），当前请求需要用户验证。

403（Forbidden），服务器已经理解请求，但是拒绝执行它。

404（Not-Found），请求的资源没有被找到

500（Interval Server Error），服务器内部错误

502（Bad GateWay），网关出错

503（Service Unavailable），由于临时的服务器维护或者过载，服务器当前无法处理请求。

**（2）xss，csrf的概念以及防范方法**

参考链接：

XSS攻击和防御： <http://blog.csdn.net/ghsau/article/details/17027893>

CSR攻击方式：

<http://www.cnblogs.com/hyddd/archive/2009/04/09/1432744.html>

**（3）CommonJs，AMD，CMD规范**

对于前端模块化来说，这三个规范是必须要了解的。

参考链接：<http://blog.csdn.net/vuturn/article/details/51970567>

**（4）谈谈对前端模块化的理解**

前端模块话就是把复杂的文件分成一个个独立的模块，比如js文件，分成独立的模块之后有利于代码的重用和维护，但是这样又会引来模块与模块之间的依赖问题，所以就有了CommonJS、AMD、CMD规范，最后出现了webpack，webpack就是前端模块话的一种解决方案，基本上大公司都会使用webpack。

**（5）优雅降级和渐进增强**

优雅降级指的是一开始就构建功能完好的网站，然后在慢慢兼容低版本的浏览器，使得各个浏览器之间的差异不要太大。

渐进增强是指在基本功能得到满足的情况下，对支持新特性的浏览器使用新特性，带给用户更换的体验。

优雅降级和渐进增强的出发点不同，前者是慢慢向下兼容，是向后看，后着是慢慢向上，增强功能，是向前看。

**（6）前端优化（提高网页的加载速度）**

1、使用css sprites，可以有效的减少http请求数

2、使用缓存

3、压缩js，css文件，减小文件体积

4、使用cdn，减小服务器负担

5、懒加载图片

6、预加载css，js文件

7、避免dom结构的深层次嵌套

8、给DOM元素添加样式时，把样式放到类中，直接给元素添加类，减少重构，回流