1、怎么理解JS面向对象

JS 基于类 和 基于原型 的面向对象；基于类的对象依靠类来产生，基于原型的，对象则是依靠构造器（constructor）利用原型（prototype）构造出来的。

ES中定义：无序属性的集合，其属性可以包含基本值、对象或者函数

2、Javascript如何实现继承？分解

function Parent (name) {//父类

this.name = 'Animal';

this.sleep = function(){

console.log(this.name + '正在睡觉！');

}

}

1、构造继承：复制父类的实例属性给子类

function Child (name){

Parent.call(this);// call方法或者是apply方法调用函数的构造函数

this.name = 'Tom';

}

Var child = new Child(‘tom’);// Tom

2、原型继承：将父类的实例作为子类的原型

function Child(){}   
Child.prototype = new Parent();

3、实例继承: 为父类实例添加新特性，作为子类实例返回

function Child (name){

var instance = new Parent ();

instance.name ='Tom';

return instance;

}

4、拷贝继承: 可多继承

function Child (name){

var Par = new Parent ();

for(var p in Par){

Child.prototype[p] = Par[p];

}

Child.prototype.name = 'Tom';

}

5、组合继承: 通过调用父类构造，继承父类属性，然后将父类实例作为子类原型，复用

function Child (name){

Parent.call(this);

this.name = 'Tom';

}

Child.prototype = new Parent();

3、JS创建对象几种方式：

1、对象字面量的；

、对象字面量ES6：可以在对象字面量里面定义原型；定义方法可以不用function关键字；直接调用父类方法、 使用内置对象：var str = new String("实例初始化String");

2、构造函数；

function Student(name,sex,grade){

 this.name = name;

 this.sayName = function(){}

}

3、工厂模式；

function createStudent(name,sex,grade){

 var o = new Object();

 o.name = name;

 o.sayName = function(){}

 return o;

}

4、原型模式；

function Student(){}

Student.prototype.name = 'Claiyre';

Student.prototype.sayName = function (){}

5、混合方式

4、CSS优先级：

!important >  id > class > tag

5、iframe缺点：

Iframe会阻塞主页面Onload事件；搜索引擎无法解读这种页面不利于SEO；

Iframe和主页面共享连接池，浏览器会相同域的连接有限制，会影响页面加载；

使用iframe最好通过js动态给src添加，绕开弊端；

6、HTML5的离线缓存怎么用，原理：

断网时，可以正常访问站点或应用，网络连接时，跟新用户机器上的缓存文件；

原理：HTML5的离线缓存基于一个新建的.appcache文件的缓存机制（不是存储技术），通过这个文件上的解析清单离线存储资源，就会想cookie一样被存储下来，离线时，浏览器会通过离线时存储的数据显示；

使用：

页面头部加入mainifest属性<html lant=’en’ mainifest=’..appcache’>

CACHE MAINIFEST

/aa.js //文件

NETWORK:

\*

Login.js //不被缓存

FALLBACK:

/AAA/ /404.html //失去连接404.html替换AAA文件夹内所有文件

HTML5存储类型：

Localstorage：k-v,永久，5M；

Sessionstorage：关闭浏览器就清空；

Applicationcache：离线缓存：mainfest文件；

Websql：关系数据

HTML5新增标签：

<article>定义外部内容；<section>文档中节；<time>时间日期；

<audio>声音；<video>视频；<canvas>; <dialog>会话；<footer>页脚；

<header>;<mark>;<nav>定义导航链接；<output>;<source>

7、同步和异步：

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,j进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

11、同源政策：协议、域名、端口

12、JSONP原理：

Ajax请求同源策略影响，不允许进行跨域请求，而script标签src属性中的链接可以访问跨域js脚本，利用这个特性，服务端不再返回json格式的数据，而是返回一段调用某个函数的js代码，在src中调用，实现跨域。

客户端传递一个callback参数给服务端，返回数据时将callback参数作为函数名包裹json数据，客户端就可以随意定制自己的函数来自动处理返回数据；

jsonp的核心则是动态添加<script>标签来调用服务器提供的js脚本

优缺点：

跨域，兼容性好，callback回传结果；

只支持get不支持post；只支持http请求；

$(document).ready(function(){

$.ajax({

type:”get”,

async:false,

[url:”http://www.test.com/test.js?code=1](url:)”,

dataType:’jsonp’,

jsonp:’callback’,//请求参数名

jsonpCallback:’testFunction’,//要执行的回调函数

success:function(data){

alert(‘数据:’+data.msg);

}

});

});

Window.name –只是字符串;

window.postMessage()—H5特性；

document.domain – iframe 中两个页面都设置相同主域

动态创建script标签；

、、、、、

跨域资源共享（CORS）：自定义HTTP头部让浏览器与服务器沟通，决定成功与否；



13、HTTP相关：

HTTP：一个基于请求与响应模式、无连接(每次连接只处理一个请求)、无状态(事务处理没有记忆能力)、应用层的协议，常基于TCP的连接；

HTTP1.1：持续连接机制，轮询(长轮询：客户端向服务端发送ajax，直到服务器返回数据后才关闭，像网页版聊天；短轮询：定时持续发送)setInterval；

头：Connection:keep-alive :实质TCP连接；

优点：减少浪费、节约时间，缺点：连接过多会增加服务器压力

Http 与TCP、IP 关系：

Http是应用层，TCP是传输层，IP是网络层；

IP是解决网络路由、寻址；TCP是IP层上传递数据包，可靠；

HTTP请求头：

请求行：1.0：get、post、head； 1.1：新增put、option、delete、connect





HTTP响应头： 状态行、响应头、空行、响应正文

常用方法：get、post、put、head、delete、options

Get、post区别：

1、Get从服务器获取数据，post向服务器传送数据；

2、get数据量小2k，post数据量大；

3、get较post安全性低；

4、get数据放在URL后、传递可能会乱码，post放在消息头head中；

建议：数据查询事可以get，做增删改时post；

CP传输的三次握手四次挥手策略：

用TCP协议把数据包发送出去后，然后会向对方确认是否送达，

即：1发送一个带SYN标志的数据包给服务器，接收后，2回传一个带有SYN、ACK标志的数据包以传达确认信息，3发送端再回传一个带ACK标志的数据包，表示握手结束；若中断，TCP会再次发送；

断开：

1、客户端发送FIN，客户端进入FIN\_WAIT\_1状态，

2、服务端收到FIN，发送ACK到客户端确认序号为收到序号+1，进入Wait

3、服务端发送FIN，进入LAST\_ACK状态；

4、客户端收到FIN，进入TIME\_WAIT状态，发送ACK给服务端，进入关闭状态；

TCP、UDP：

TCP，传输控制协议，基于连接，面向连接、传输可靠、大量数据、速度慢；

UDP，用户数据协议，非连接，不与对方建立连接直接发送数据包过去，适用于一次传送少量数据、对可可靠性要求不好的环境

HTTP、HTTPS：

通信明文不加密，内容会被抓包；

没有验证，会有伪请求；

HTTPS加密ssl认证；

HTTP优化：负载均衡、加速http应用、HTTP Cache优化；

WebSocket：基于ICP；允许服务端像客户端首次发送请求；

send();onmessage();close();

14、

Js原型、原型链：

Js中有两种类型值，原始值、对象值，每个对象都有一个内部属性prototype(原型)，原型值可以是对象、null，原型为对象，对象一定有自己的原型，形成一条线性链，原型链

原型对象：所有对象实例共享原型包含的属性和方法；

function Person(name, age, job) {

this.name = name;

this.age = age;

this.job = job;

this.lessons = ['Math', 'Physics'];

}

Person.prototype = {

constructor: Person,//原型字面量方式会将对象的constructor变为Object，此外强制指回Person

getName: function () {

return this.name;

}

}

var person1 = new Person('Jack', 19, 'SoftWare Engneer');

person1.lessons.push('Biology');

var person2 = new Person('Lily', 39, 'Mechanical Engneer');

alert(person1.lessons);//Math,Physics,Biology

alert(person2.lessons);//Math,Physics

alert(person1.getName === person2.getName);//true,//共享原型中定义方法

js闭包：

当一个函数返回它内部定义的一个函数时，就产生了一个闭包，包就是能够读取其他函数内部变量的函数；

或：**当函数a的内部函数b被函数a外的一个变量引用的时候，就创建了一个闭包**

用处：可以读取函数内部的变量，让这些变量的值始终保持在内存中

Js声明提前：

Js的函数作用域：在函数内声明的所有变量在函数体内始终可见。

在变量声明之前就可以使用了，此情况叫声明以前。

作用域链：

由于js的变量都是对象的属性，对象又可能是其他对象属性，所有对象是window的属性，这些对象的关系可以看做一条链；

保证执行环境里有权访问的变量和函数是有序的，作用域链的变量只能向上访问，变量访问到window对象即被终止，作用域链不允许向下访问变量；

局部变量优先级高于全局变量；

任何地方都能访问到的对象拥有全局作用域；

所有末定义直接赋值的变量自动声明为拥有全局作用域；

所有window对象的属性拥有全局作用域

15、BOM对象：

Window：js中顶级对象；navigator：客户端浏览器信息；

Screen:显示屏；history：浏览过的URL；

Location：当前URL；document：整个html文档

DOM对象：

Document：整个html文档；node：文档树节点；

Nodelist：多个node列表，数组；element：xml标签；

Attr：某标签中属性；

16、HTML 、CSS ：

对W3C理解：

标签闭合、标签小写、不乱嵌套、使用外链css、js、文件下载速度、页面流畅度、组件化、易维护性，易用性

行内元素，块级元素，css盒模型：

块：div p h1-h4 form ul tr

行：a b br span input select

Css盒：content border margin padding

清除浮动：

1. 父子类添加float；
2. 给父级加高度；

clear:both; overflow:auto; zoom:1；//兼容；

:after:伪类 :after{content:""; display:block;clear:both;} zoom:1

17、事件绑定和普通事件有什么区别：

普通事件onclick只支持单个事件，其他相同会被覆盖；时间绑定中addEventListener可添加多个不覆盖IE8；

18、冒泡原理，让自己的触发事件，让他的父元素执行；

事件委托：是通过事件冒泡的原理,利用父级去触发子级的事件

事件流：

1. 事件捕获：元素触发事件，顶层发出事件流，函数不会触发；
2. 事件目标：当达到目标元素后，执行目标元素函数；
3. 事件冒泡：从目标元素起，往顶层传递，途中有节点事件会被触发

19、如何阻止事件冒泡和默认事件：

function stopBubble(e){// 阻止事件冒泡

  if(e&&e.stopPropagation){//非IE

   e.stopPropagation();

  }

  else{//IE

   window.event.cancelBubble=true;

  }

 }

function stopDefault( e ) {

  if ( e && e.preventDefault ){ //阻止默认浏览器动作(W3C)

    e.preventDefault();

}else{//IE中阻止函数器默认动作的方式

  window.event.returnValue = false;

    return false;

 }

}

20、数组去重：

var arr = [12,23,23,43,12,55,66,77,55];

var brr = [];

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

if(brr.indexOf(arr[i]) < 0){

brr.push(arr[i]);

}

}

21、编写一个js函数，该函数有一个n（数字类型），其返回值是一个数组，

该数组内是n个随机且不重复的整数，且整数取值范围是[2,32] ：

var jsFun = function(n,min,max){

let reValue = [];

let isNum = !isNaN(Number(n));//判断n是不是一个数字

let isRandOk = (n >= min && n <= max && n <= (max - min)) ? true : false;//判断n的取值是否符合

if(isNum && isRandOk){

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

let rand = Math.round(Math.random()\*(max - min) + min);

if(reValue.indexOf(rand) > 0){

i--;

}else{

reValue.push(rand);

}

}

}

return reValue;

}

22、JS冒泡排序：

var arr = [5,4,3,2,11];var brr = 0;

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

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

if(arr[j] > arr[j+1]){

brr = arr[j+1];

arr[j+1] = arr[j];

arr[j] = brr;

}

}

}

23、快速排序：

function qSort(arr){

if(arr.length<=1){return arr; }

var pIndex = Math.floor(arr.length/2);

var piv = arr.splice(pIndex,1)[0];

var left = [];

var right = [];

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

if(arr[i] < piv){

left.push(arr[i]);

}else{

right.push(arr[i]);

}

}

return qSort(left).concat([piv],qSort(right));

}

1、前端页面有哪三层构成，分别是什么?作用是什么?

结构层 Html 表示层 CSS 行为层 js

2、解释css sprites，如何使用

把一堆小的图片整合到一张大的图片上，减轻服务器对图片的请求数量

3、javascript的typeof返回哪些数据类型

　　Object String number function boolean underfind

4、例举3种强制类型转换和2种隐式类型转换?

　　强制(parseInt,parseFloat,number)

　　隐式(== – ===)

5、call和apply的区别

　　Object.call(this,obj1,obj2,obj3)

　　Object.apply(this,arguments)

6、ajax请求时，如何解释json数据

　　使用eval parse 鉴于安全性考虑 使用parse更靠谱

7、传递到jQuery方法的四种不同值

1、$(document).ready(function()

{

$("#summary").bind("click", {foo:'abc'} ,GetCode);

});

2、$("#summary").bind("click", function()

{

GetCode("abc")

});

3、闭包：

|  |  |
| --- | --- |
|  | $("#summary").bind("click", GetCode("abc")); |

### 4、什么是CSS Hack?

针对不同浏览器的CSS

// 条件Hack

<!—[if IE]> <style> .test{color:red;} </style> <![endif]>

// 属性Hack

.text{

color:#090\9;/\*for IE8+\*/

\*coloe:#f00;/\*for IE7\*/

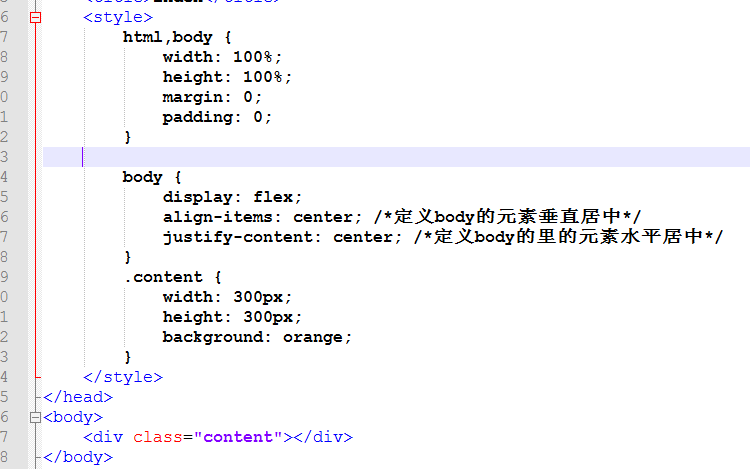
}

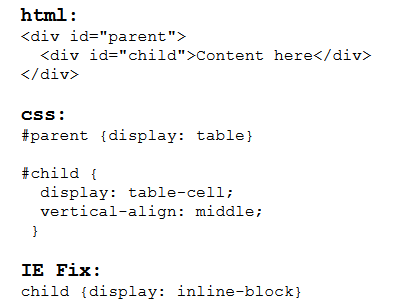
// 选择符Hack

\*html .test{}

5、px和em区别：em不是固定值，继承父级元素 1em=16px

6、垂直居中：display:flex;





或者：position:relative; position:absolute；

或者：margin:auto

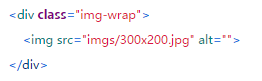
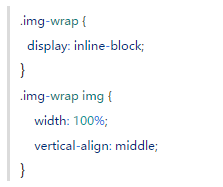
或者：绝对定位和transform

或者：vertical-align:middle;line-height:2px;//缺点：内容不能换行

兼容性：



7、图片自适应：

9、jQuery 扩展方法：

jQuery.extend(Obj);//jQ本身的扩展

jQuery.fn.extend(Obj);//jQ所选对象的扩展

jQuery.extend({

Mes:function(mess){},

MesTwo:function(two){}

})

//页面调用：jQuery.Mes(“黑”)

jQuery.fn.extend({

Show:function(show){

jQuery(this).html(show);

}

})

//页面调用 ：jQuery(“#div”).Show(“nihao”);

或：(function(obj){

Obj.fn.Show=function(show){

jQuery(this).html(show);

}

})

10、CSS动画：

@keyframe 规则或 animation 属性

11、java是块级作用域，js是函数作用域，没有块级作用域

12、JS算法：

1.冒泡排序：比较相邻的元素，if(a>b)，交换位置，重复

Function b(arr){

Var len = arr.length;

For(var i=0;i<len;i++){

For(varj=0;j<len-1-i;j++){

If(arr[j]>arr[j+1]){ //相邻元素比较

Var temp = arr[j+1]; //元素交换

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

Return arr;

}

改进版：设置一标志性变量pos,用于记录每趟排序中最后一次进行交换的位置

2.选择排序：在未排序的序列中找到最值元素，存放到起始位置；

Function a(){

Var len=arr.length;

Var minIndex,temp;

For(var i=0;i<len-1;i++){

minIndex = i;

for(var j=i+1;j<len;j++){

if(arr[j]<arr[minIndex]){//招最小的

minIndex = j;

}

}

Temp = arr[i];

Arr[i] = arr[minIndex];

Arr[minIndex] = temp;

}

}

3．插入排序：(扑克牌按小到大插入排列)通过构建有序排列，对未排序数据，在已排序中向后扫面，找到相应位置插入

Function(array){

If(Object.prototype.toString().call(array).slice(8,-1)==’array’){

For(var i=1;i<array.length;i++){

Var key = array[i];

Var j = i-1;

While(j>=0 && array[j]>key){

Array[j+1] = array[j];

j--;

}

Array[j+1] = key;

}

Return array;

}

Return ‘array is not an Array’;

}

4、快速排序：

通过一趟排序将待排序记录分隔成独立两部分，其中一部分记录的关键字比另一部分小，则可分别对这两部分记序排序，已达到整个序列有序

Function quickSort(arr){

If(arr.length<=1) return arr;

Var pivotIndex = Math.floor(arr.length/2);

Var pivot = arr.splice(pivotIndex,1)[0];

Var left = [];

Var right= [];

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

If(arr[i]<pivot){

Left.push(arr[i]);

}else{

Right,push(arr[i]);

}

}

Return quickSort(left).concat([pivot],quicksort(right));

}



1. 判断一个单词是否是回文：str.split(‘’).resverse().join(‘’);
2. 数据去重：indexOf()、 或者 通过key

Let unique = function(arr){

Let judge = {};

Let newArr= [];

Arr.forEach(item=>{

If(!judge[item]){

judge[item] = item;

newArr.push(item);

}

});

Return newArr;

}

1. 计算数组中最大的差值：

Math.max.apply(this,arr) - Math.min.apply(this,arr);

Function getMaxPro(arr){

let min = arr[0];

let maxPro = 0;

For(let i=0;i<arr.length;i++){

let current = arr[i];

let min = Math.min(current,min);

let pro = current – min;

let maxPro = Math.max(maxPro,pro);

}

Return maxPro;

}

8、统计字符串中出现最多的字母：

Function findMoustChar(str){

Let charObj = {};

For(let i=0;i<str.length;i++){

If(!charObj[str.charAt(i)]){

charObj[str.charAt(i)] = 1;

}else{

charObj[str.charAt(i)] +=1;

}

}

Let maxChar = “”;

Let maxValue= 1;

For(let k in charObj){

If(charObj[k]>maxValue){

maxValue = charObj[k];

maxChar = k;

}

}

Return maxChar；

}

1. 九九乘法表：

Var i,j;

For(i=1;i<10;i++){

For(j=1;j<10;j++){

Document.write(“”+i+”\*”+j+”=”+i\*j+” “);

If(i=j){document.write(“<br>”);break;}

}

}

1. 数组倒叙：arr = [];arr.reverse();
2. 数组降序：arr.sort(function(a,b){return b-a;});
3. YYYY-MM-DD:

Var d = new Date();

Var y = d.getFullYear(); var m = d.getMonth()+1; m=m<10?’0’+m:m;

Var d = d.getDate(); d = d<10?’0’+d:d;

1. Ul li 输出点击每列的index：

Var list = document.getElementById().getElementsByTagName(‘li’);

For(var i=0;i<3;i++){

List[i].index=i;

List[i].onclick=function(){alert(this.index)};

}

13、this在js中的指向；

1.全局：指向全局对象；

2.函数调用：指向全局对象；foo();

3．方法调用：指向test对象：test.foo();

4.调用构造函数：this指向新创建对象：new foo();

5.显示的设置this：使用call、apply ，this指向第一个参数

14、H5和CSS3 新特性：

1.localStorage、sessionStorage

2.canvas、video、audio

3.语义化内容元素：article，section、footer、header、nav

4.表单控件：calendar、date、time、url

5.离线缓存

CSS3：

新增选择器：nth-child()…

盒模型：

文字特效：背景填充、@Font-face …

动画：animation

16、css 三角形：把 上、左、右 三条边隐藏 颜色transparent

Demo{

width:0;height:0;border:20px solid;

border-color:transparent transparent red transparent;

}

17、HTML语义化：

让页面结构化，清晰、易读；利于SEO；利于网页分块便于维护理解；

18、label标签来定义表单控制间的关系，当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。

19、如何实现浏览器内多个标签页之间的通信？

调用 localStorage、cookies 等本地存储方式

20、JS基本数据类型：

Undefined、null、String、Boolean、Number、Object

JS内置对象：

数据封装类：Object、Array、Boolean、Number、String

其他：Function、Argument、Math、Date、Error

21、call、apply、bind作用、区别：

功能：实现继承，转换对象指针；

.call(对象，参数1，参数2，…)//参数罗列，对象的参数；

.apply(对象，参数组)

.bind();//返回一个函数，不会被执行

1. function Animal（name,weight）{

this.name = name;

this.weight=weight;

}

Function Cat(){

Animal.call(this,’cat’,’30’);

// Animal.apply(this,[‘cat’,’30’]);

This.say = function(){

Console.log(this.name + ‘,’+this.weight);

}

}

Var cat = new Cat();

Cat.say();// cat,30

1. var aaa = {

name:’小王’,

gender:’男’,

say : function(school,age){

console.log(this.name+’,’+this.gender+’,’+school+’,’+age);

}

}

Var bbb = {

Name:’小红’,

Gender:’女’,

Age:18

}

aaa.say.call(bbb,’小学’,10);

aaa.say.apply(bbb,’小学’,12);

aaa.say.bind(bbb)();//返回一个函数，不执行

aaa.say.bing(bbb)(‘小学’，20);// 传参调用

22、Web网站常见攻击

1.跨站脚本攻击(XSS): 在网页上发布包含攻击性代码的数据

特定的脚本会以浏览用户的身份和权限来执行，

解决：1.将重要的cookies标记为HTTP only；2.限制输入规则；

3.对数据进行编码处理；3过滤js事件的标签

出错的页面漏洞：页面找不到，会直接把URL显示；

2.跨站请求伪造攻击(CSRF): 伪造请求，模仿用户提交表单，修改用户数据

解决：1.post请求；

2.对请求进行认证，token

3.Hash验证；加验证码

3.Http Heads攻击：请求头执行js脚本

4．Cookie攻击、重定向 ：加token

23、<!DOCTYPE html>：告知浏览器使用哪种文档规范

怪异模式：浏览器使用自己的解析方式渲染界面；

标准模式：浏览器采用W3C标准解析渲染界面

另外：严格模式：根据规范渲染； 混杂模式：向后兼容IE5以下

触发：浏览器根据doctype是否存在和使用哪种DTD来决定

25、盒子模型：由 content、border、padding、margin

两种：IE盒子、W3C标准盒子

区别：width=border+padding+content；W3C，width=content

解决：doctype 加声明，使用W3C标准

26、AMD：异步加载，浏览器不会失去响应

CMD：模块定义和模块加载时机不同；

AMD依赖前置，在定义模块时就要声明依赖；

CMD就近依赖，只有用到哪个模块才去require

27、@media screen

Css:

@media screen and (min-width:450px) and (max-width:900px){}

Link 引入不同css：

<link rel=’stylesheet’href=’style.css’media=’screen and (min-width:450px)>

28、原生JS实现ajax：

Var xmlHttpRequest;

Function creteXmlHttpRequest(){

If(window.XMLHttpRequest){//非IE

xmlHttpRequest = new XMLHttpRequest();

}else if(window.ActiveObject){//IE6+

xmlHttpRequest = new ActiveObject(“Msxml2.XMLHTTP”);

}else{

xmlHttpRequest = new ActiveObject(“Microsoft.XMLHTTP”);

}

}

open(method,url);

send(content);

onreadystatechange(){

if(xmlHttpRequest.readyState = 4){

if(xmlHttpRequest.state == 200){

alert(“成功”);

}

}else{

Alert(“请求失败”);

}

}

29、DOM操作：

创建:createElement()、cretentTextNode();

appendChild()、removeChild()、replaceChild()、insertChild()

getElementsByTagName()、getElementsByName()、getElementsById)

30、window.setTimeout(‘function’,time);//只执行一次，没有生命周期

Window.setInterval(‘function’,time);//有生命周期，每一个time执行一次

Window.clearTimeout(对象);

Window.clearInterval(对象);

31、JS 大小写：toUpperCase(); toLowerCase();

32、eval : 把字符串解析成js，并运行返回执行结果；

把json字符串转化为json对象

Var json = “{name:’aa’，age:12}”;

Var jsonObj = eval(“(”+json+”)”);

两次执行，耗能，不安全 parse 来替代

33、JS事件处理机制：捕获-处理-冒泡

捕获：外层到内层，首相注册click，document、body、父节点、按钮；

处理：按钮本身的click；

冒泡：从按钮开始，内层到外层，依次调用父节点的冒泡监听直到window

IE8以下不支持捕获；通常处理：通过全局的event来访问，$().addEventListener(‘click,event,false);

Vue 部分：：：：：：：：：：：：：：

1、Vue 中MVVM：model，view，viewModel，Vue是以数据位驱动的，Vue本身将dom和数据进行绑定，一旦绑定，dom和数据将保持同步，数据发生变化时，dom发生变化。

viewModel是Vue的核心，是Vue的实例domListeners 监听所有view层变化，变化时Model层变化，Databingding监听model层数据，数据发生变化view的dom元素变化；

2、v-show，v-if：条件渲染无论v-show值是否是true，元素都在HTML中，v-if只要true就会存在；

3、Vue组件要先注册才能使用；

4、路由嵌套：路由嵌套会将其他组件渲染到该组件中，不是整个页面跳转，router-view本身就是将组件渲染到该位置，要跳转就要将页面渲染到跟组件，

在起始位置配置：var app = Vue.extend({root});router.start(app,'#app');

4.1、resolve =>require(URL,resolve)//懒加载；

5、绑定事件：DOM元素绑定，组件绑定；元素绑定：v-on:click='event'//1.0 小写

6、Vue是什么：构建用户界面的渐进式框架。自底向上，增量开发。核心库只关注视图层，API实现响应的数据绑定和组合的视图组件。

7、Vue特性：

简洁：页面由HTML模板+Json+Vue实例；

组件化：可复用、解耦组件；

轻量：不依赖其他库；

快速：精确批量dom,虚拟dom

模板友好：npm、bower多方式安装；

8、Vue、angular区别：

Vue是一个灵活的解决方案，它允许你希望的方式组织你的程序，而不是必须循序angular制定的规则。

Vue仅仅是一个视图层，可以嵌入一个现有的页面不会特别庞大；

Vue内部设计简单；

Vue性能好，没有太多的检查，当watcher越来越多时，angular会变得慢，因为作用域内的每一次数据变更，

所有的watcher都要被重新求值；

Vue指令和组件分的清晰。指令只负责封装dom操作，组件代表一个独立的部分；

9、Vue、react 区别：

相似：虚拟DOM快速渲染，轻量级，响应式组件，

易于继承路由打包状态管理。

区别：vue模板和渲染函数，简单语法项目构建，渲染速度体积小；

React适用大型应用可测性高，适用于webAPP，依赖虚拟dom；

但是，内部实现不同。react基于一种内存中描述的dom树状的结构，react中数据被看做是不可变的，而dom操作是

通过虚拟dom的 diff 来计算的。相比Vue，中数据默认是可变的，数据变化会直接对dom更新，Vue会使用实际的dom作为模板

并且保持对真实节点的引用来实现数据绑定。

react的虚拟dom，每次更新会重新渲染整个应用。react的dom更新完全交给虚拟dom管理，当你想要自己控制dom，违背了react的设计原理；

11、less和sass：

预处理语言，有变量、继承、运算、函数，弥补css单一古板；less是客户端基于js运行，sass是基于ruby，服务器端，sass功能大于less；sass有Compass库

12、JS 兼容性问题：

IE8 是 ES3，IE9+ ES5，js兼容问题是兼容ES规范的使用；

1. 浏览器宽高：

Width=document.body.clientWidth||document.documentElement.clientWidth;

scrollWidth 、scrollTop

1. event事件：

document.onclick=function(e){

var e||window.event;//事件对象// e.target||e.srcElement;//对象目标

var mousex = e.clientX;//var key = e.keyCode;//键盘

}

1. DOM节点：

Function (obj){

//obj.previousElementSibling//obj.fristElementChild//lastElementChild

If(obj.nextElementSibling){

Retrun obj.nextElementSibling;//下一个兄弟节点

}else{

//obj.previousSibling//obj.fristChild

Return obj.nextSibling;

}

1. document.getElementsByClassName();//ie8- 不支持；可用jQ；

或者：document.formName.elements[‘A’];

1. 添加监听事件：

Function (o){

If(o.addEventListener){//o.removeEventListener

o.addEventListener();

}else{

o.attachEvent();//o.detachEvent();

}

}

1. 阻止事件：

Document.onclick=function(e){

Var e=e||window.event;

If(e.stopPropagation){//e.preventDefault//阻止默认事件

e.stopPropagation();//W3C

}else{//e.returnValue=false;

e.cancelBubble = true;//ie

}

}

1. 兼容不同浏览器：

自定义属性：getAttribute();

尽量用document.getElementsById();

IE下不支持input.type 修改，可以隐藏做append添加新的；

尽量使用document.location();//href()FF低版本不适用；

setAttribute()//使用obj.ClassName=’类名称’;

设置事件都用 . 形式

8.获取父元：FF：obj.parentNode;IE:obj.parentElement\obj.parentNode

13、CSS兼容性问题：

1.样式初始化;

2.css Hack：<!--[if IE8]><html class=”ie8”><![endif]-->

css兼容性写法：Hack把浏览器分为3类：IE6、IE7和遨游、

IE8及Chrome，FF等；

IE6:下划线 \_和\*; IE7: \*;

: height:100px;\*height:100px;\_height:100px;

3.meta头添加content；

4、IE8问题：input设置padding输入内容过多会失效，解决：外加div；

IE11,div不添加高度，会比其他浏览器偏大；

margin加倍float下在IE中会加倍(设置display:inline)

5、FireFox 下文本无法撑开容器高度：

清除浮动: .clear{clear:both;height:0px;overflow:hidden;}

6、选择器优先级：子选择器比父级选择器优先级高；

7.设置较小高度标签(<10px),IE6、7中高度超出自己设置高度：

overflow:hidden;或者line-height小于设置高度；

8.行内标签，display:block后采用float会有IE6 margin问题，

display:inline;display:table;

1. 几个img一起，浏览器会有默认间距，添加block；或者负间距；
2. Float闭合：给父类加float、或者父类加overflow:auto; 或者子类末尾加div{clear:both;margin:0px;padding:0px;height:0px;font-size:0px;}
3. CSS透明度：

{filter:alpha(opacity=70);//IE

-moz-opacity:0.7;//FF

Opacity:0.7;//chrome、Opera等

}

各浏览器兼容性问题：

1. 圆角问题：IE，behavior: url(ie-css3.htc);图片做背景；
2. 垂直居中：字体line-height
3. input元素在设置了高时，在IE7、IE8、IE9下会出现文本和文本输入框不能对齐的现象: vertical-align:middle;或line-height:100px
4. 不同浏览器下宽度不同，比如说设置width：200px，在iE7、IE8、IE9下显示的是200px，在FF、Chrome、Opera中显示的是220px  
   width:200px; \*width:220px
5. IE7、IE8、IE9对min-height不识别；直接height双重定义
6. IE7下如果为li指定高度可能会出现排版错位：设置line-height

14、前端性能优化：

1、减少http请求：图片地图(小图放到一张图)，CSS Sprites，内联图片(base64压缩:减少http不影响加载速度；不会缓存图，兼容差>=IE8,小图)，合并脚本JS、CSS

2、CDN：内容发布网络；部署在不同地区的服务器来提高客户的下载速度；在用户和服务器之间增加Cache层，将请求引导到Cache上获取服务器的数据；

3、添加Expires头(服务端设置，限制：服务与客户端时间一致)

4、压缩组件：Gzip压缩

5、延时加载；样式在顶部，脚本在底部，减少CSS表达式，减少JS操作节点，(append)使用外部JS，CSS(可以被缓存);精简CSS

6、减少DOM元素数量

7、减少iframe数量：阻止页面加载

8、避免图空src；减少cookie；

9、可以index.css?201708;加缓存时间

15、前端性能优化JS部分：

1.避免重复JS、避免网页加JS；

2.尽量减少DOM访问；

3.代码压缩，Gzip压缩；

4.定义局部变量，JS查顺序：this开始局部变量，函数参数，本地变量，最后全局变量；

5．避免使用with(),会创建自己的作用域；

6.小心使用闭包，局部变量，内存；

7.避免使用数组，或缓存长度：JS对数据访问速度：数值、变量、对象属性、数组；var len = arr.length;减少查询；

8.for循环避免使用for(in);他会查询Hash键，增加额外变量；

9.避免使用+=拼字符串，用数据push、arr.join(“”);

10.转化字符串：性能：**("" +A) > String() > .toString() > new String()**

11.各类型转换：

Var str = “3.14159”;

Str = “”+ str;// 转化字符窜

Int = ~ ~str; // 转化integer// 或Math.floor();Math.round()

Float = 1 \* str; //to float

Boolean = !!str; // to Boolean

Array = [str]; // to array

12.直接量，避免new：var arr = [];arr.color = “red”;

13.使用innerHtml替代appendChild();最好一次赋值：$().innerHTML = “”;

14.删除DOM节点：一定要删除节点上的事件，innerHTML = “”;

15.写到局部变量，避免重复访问；

For(var i=0,var len = arr.length;i<len;i++){}

三目运算替代if条件；

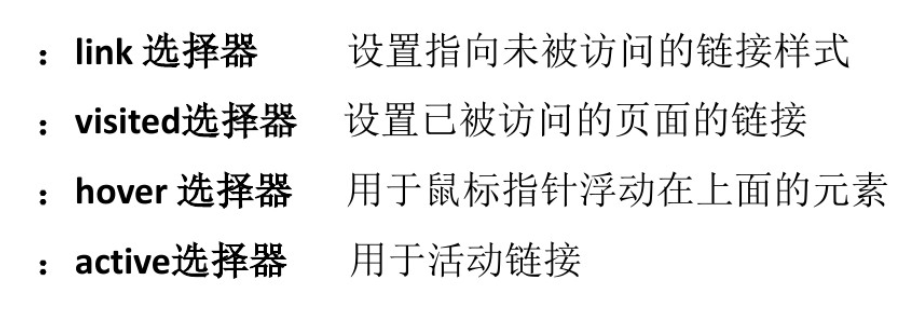
16.释放JS对象：object = null；…

17.使用位运算符会快一些、使用=== \ !==

18.避免使用eval；使用CDN；减少对象查找 arr.a.b.d.c; 使用setInterval替换setTimeout

怎么实现0.5px：transform:scale(0.5,0.5);

-webkit-transform:scale(0.5,0.5);



Vue是以数据为驱动，Vue自身将DOM和数据进行绑定，绑定，DOM和数据将进行同步，数据变化，DOM变化；



实现原理：把一个普通对象传给 Vue 实例作为它的 data 选项，Vue.js 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。这是 ES5 特性，不能打补丁实现，这便是为什么 Vue.js 不支持 IE8 及更低版本。



Vue 常见属性：

Computed：get set；

Methods：供组件；

Watch: 监听数据改变；

Data： vue暴露实例属性方法，前缀$;

$el:当前元素；#data:挂载数据;$option：自定义属性；

vm.$emit(event,[]);触发当前实例上的事件

vm.$on(event,callback);监听当前实例上自定义事件

Vue 修饰符：

v-on:click.stop=’方法’// 阻止事件冒泡；

v-on:submit.prevent=’’ //提交事件不在重载页面

v-on:click.stop.prevent=’’//修饰符串联

v-on:submit.prevent // 只有修饰符

v-on:click.self=’’// 只有事件在本身(非子元素)触发时回调；

按键修饰符：

v-on:keyup.13=’click’//vm.click()

对JSON的相关方法：

1. 字符串、对象：

$.parseJSON(str);//jQuery.parseJSON(str);

JSON.parse(str);JSON.stringify(str);//兼容IE8以上，引入官方json.js,支持IE67，不推荐eval会执行里面的表达式

2.添加删除：push、 delete array[2]

3.排序：

一维数据：

function sortN(a,b){return a-b;};[“2”,”1”].sort(sortN);

JS 浅复制，深复制：

1. 浅复制：同一个对象引用；
2. Js 的 slice\concat：var arr = array.slice(0);
3. 便利对象属性赋值：b.name = a.name;b.age=a.age;

数组方法：

1. pop：删除数组中最后一个元素并返回该元素，原数组被删除；
2. push：concat：合并；
3. unshift：插入数据开始位置；
4. splice：添加或删除str.splice(位置,删除量|0为不删除,要添加的新)
5. shift：删除第一个位置，并返回该元素，数组变化

JQuery 的事件绑定：

1. bind():$().bind(“click”,function{});//为每个匹配元素绑定事件处理函数，后来添加的页面元素不会生效；
2. live()[1.9被取消]\on()[1.7后版本]\delegate()[效率好]:动态绑定

$().on(‘click’,’li’,function(){});// 绑定具体元素

1. 对应解除事件：unbind\die\undelegate\off;

jQuery 常见的方法：

attr/ .html()/ .val()/ append()/ appendTo()/ prepend()/ after()/ before()/ insert()/ remover()/ empty()/ clone()/ warp()/ addClass()/

toggleClass()/ hasClass()/ text()/ next()/ sublings()/ prev()/ data()

JQuery的ajax写法：$.ajax({}), $.post(), $.get(), $.getJSON();

浏览器内核JS引擎：

浏览器：shell(壳子:菜单工具栏..) + 内核

内核：渲染引擎 + JS引擎

渲染引擎：对网页语法解释渲染页面

主流内核： Trident(IE)、Gecko(FF)、Presto(Chrome)、Webkit(Chrome)

Express:

基于node的web框架，快速搭建网站，有入口、路由概念；

基于node内置的http模块上，http服务器实例，req、res值

SEO：

Meta 标签title关键字；路径绝对引用；dom元素小写闭合；jscs合并；

JS面向对象：一种编程范式；面向对象设计、编程；封装继承多态；

JS数据类型：基本(string,null,number,boolean,undefined),对象(Object,Array,Fuction)

ES6新语法：

1. let const：块级作用域
2. 多行字符串：反引号；
3. 模板表达式：反引号中 ${NAME} 获取值；
4. 箭头函数：

不用写function和return：

(function(obj){return obj+1;}); (obj=> obj+1;)

//多行语句需要用{}括起来，单行表达式不需要{}，并且会作为函数返回值

没有自己的this，继承外围的this;

**constructor方法**，也**没有prototype**，所以**不支持new操作**

1. **promise:解决同步异步问题，多个回调嵌套问题：all、resolve、reject**

**链式多then（），catch()返回错误；**

1. **新增方法：**

**Object.is(a,b) 可以识别自身; 和 === ， +0===-0 //false**

**Object.assign(target,source);//复制对象**

**For(let s of str); includes();**

**数组：arr.find(()=>{});for(let s of arr.keys()){};//arr.values()**

**Array.of();//返回数组，嵌套数组[[]]**

Bootstrap相关：

1. 是快速开发web应用程序和网站框架，基于HTMLcssJavaScript；
2. 移动优先，浏览器支持，上手快，响应式设计；
3. 栅格布局：通过容器大小平分12分，调整内外边距，结合媒体查询实现网格布局；
4. 自定义5列：栅格布局：@grid-columns:5;或自定义CSS；

CSS3 新特性：

1. CSS3实现圆角（border-radius），阴影（box-shadow），

2. 对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

3.transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜

4. 增加了更多的CSS选择器  多背景 rgba

5. 在CSS3中唯一引入的伪元素是 ::selection.

6. 媒体查询，多栏布局

HTML5 新特性：

1. 拖拽释放(Drag and drop) API

2. 语义化更好的内容标签（header,nav,footer,aside,article,section）

3. 音频、视频API(audio,video)

4. 画布(Canvas) API

5. 地理(Geolocation) API

6. 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

7. sessionStorage 的数据在浏览器关闭后自动删除

8. 表单控件，calendar、date、time、email、url、search

9. 新的技术webworker, websocket, Geolocation

Gulp 和 webpack 区别：

Gulp：件压缩合并、雪碧图、版本控制，定义执行顺序，侧重于构建项目开发流程；

Webpack：强调前端模块化，侧重模块化打包，图片jscss 通过loader和插件进行资源管理，打包成符合生产部署的前端资源；

Cookie 和 session ：

1. cookie存在浏览器，session存放在服务器上；
2. coolie不安全，session；登录信息一般保留session；
3. session会一定时间内存在服务器，增加服务器性能压力；
4. 单个cookie<4k ， 浏览器一个站点可以有20个cookie；

堆栈区别：

1. 栈：编译器自动分配释放，存函数参数、局部变量等；
2. 堆：自定义分配释放；

<div class="father">

<div class="son"></div>

</div>

垂直居中：

固定宽度：

1、.father{} .son{positon:absolute;left:50%;top:50%;width:100px;

height:100px;margin-left:-50px;margin-top:-50;}

不固定宽高：

1、.father:{position:fixed;width:100%;height:100%;top:0;left:0}IE8+

.son{position:absolute;top:0;bottom:0;left:0;right:0;margin:auto;

width:50%;height:50%;}

2、.father{display:table;width:100%;} //高度不受控制

.son{display:table-cell;vertical-align:middle;text-align:center;}

3、.father{} .son{position:fixed;height:50%;width:50%;top:50%；

left:50%;-webkit-transform:translateX(-50%) translateY(-50%)}

4、.father{justify-content:center;align-items:center;display:flex;}



display和position ：

：none、block、inline、inline-block、flex、inherit

1.inline、block：行内元素、块级元素，block独占一行，垂直布局、inline水平方式布局，垂直方向margin,padding无效,无法宽高；

2.inline-block：两者之间；

3.flex:子元素的float、clear、vertical-align失效；IE9+

：static、relative、absolute、fixed、inherit

1. static:快速取消定位：top,left等失效；
2. absolute不会撑开父级，相对于第一个父类；
3. fixed：IE7-不支持；定位于窗口
4. inherit:继承父类，IE不支持；

9、RequireJS 和 Sea.js异同:

同：都是模块加载器，主导模块化开发；

异：1、定位差异，RequireJS想成为浏览器的模块加载器，同时也想Node等环境的模块加载器；Sea.js专注Web浏览器，通过node扩展方式在node环境中运行；

2、遵循规范不同，RequireJS为AMD，Sea.js为CMD；

3、RequireJs:防止js加载阻塞渲染；简化引用写法；

requireJS:

1.<script src=require.js data-main=main.js defer async=true>;

IE 支持defer

2.require([moudleA,moudleB],function(A,B){//其他模块加载完后执行});

3.require.config({

paths:{

baseUrl:”js”,

“jquery”:”jquery.min.js”}

});

1. define(function(){

var add=function(a,b){return a+b;};

return{add:add}

});

require([‘math’,function(math){alert(math.add(1,1))}]);