# javascript保留字：

就是不能用来当变量,函数名,对象名等用的特殊字 符abstract boolean byte char class const debugger double enum export extends fimal float goto implements import int interface long mative package private protected public short static super synchronized throws transient volatile

# javascript关键字：

这些关键字可用于表示控制语句的开始或结束，或者用于执行特定操作等

break case catch continue default delete do else finally for function if in instanceof new return switch this throw try typeof var void while with

# 以下代码会输出什么？

var User={

count: 1,

getCount:function(){

return this.count;

}

}

console.log(User.getCount());

var func=User.getCount;

console.log(func());

//1和undefined

# alert(Null==undefined) true

# js事件流模型：

冒泡型（dubbed bubbling ）事件：指事件按照从最精确的对象到最不精确的对象的顺序逐一触发。

捕获型（event capturing ）事件：它与冒泡型事件相反，指事件按照从最不精确的对象到最精确的对象的顺序逐一触发。

# 如何获取页面中所有的checkbox?

var aInput=document.getElementsByTagName(‘Input’)

var len=aInput.length;

var arr=[];

while(len--){

if(aInput[len].type==’checkbox’){

arr.push(aInput[len]);

}

}

# 闭包：

----------函数嵌套函数, 形成一个独特的作用域, 利用这个作用域来做实际的开发

------------可以将全局变量转成局部变量(不影响变量的功能, 防止变量名之间的冲突)

-------------可以利用闭包存储变量的状态，外层函数执行结束不会影响到内层函数

请定义这样一个函数

function repeat (func, times, wait) {

}

这个函数能返回一个新函数，比如这样用

var repeatedFun = repeat(alert,10, 5000)

调用这个 repeatedFun ("hellworld")

会alert十次 helloworld, 每次间隔5秒

------ function repeat (func, times, wait) {

//不用匿名函数是为了方便调试

function repeatImpl(){

var handle=null,

\_arguments = arguments,

i = 0;

handle = setInterval(function(){

i = i + 1;

//到达指定次数取消定时器

if(i === times){

clearInterval(handle);

}

func.apply(null, \_arguments);

},wait);

}

return repeatImpl;

}

//测试用例

var repeatFun = repeat(alert, 3, 1000);

repeatFun("hellworld");

## 例子一

//闭包外部调用

function aaa(){

var a=1;

function bbb(){

a++;

alert(a);

};

//bbb();//2

//bbb();//3

return bbb;

};

var fn=aaa();

fn();//2

fn();//3

## 例子二

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

{

show(i);

}

function show(i){//闭包

//0

//1

//2

//3

aLi[i].onclick = function(){

alert(i);

};

};

# 移动端触摸事件

ontouchstart、ontouchmove、ontouchend、ontouchcancel

# 移动设备如何禁止用户缩放页面

<meta name="viewport" content="device-width,user-scalable=no, initial-scale=1 ">

# Session storage和local storage的区别

* + - sessionStorage
      * session临时回话，从页面打开到页面关闭的时间段
      * 窗口的临时存储，页面关闭，本地存储消失
    - localStorage

永久存储（可以手动删除数据）

* + Storage的特点
    - 存储量限制 ( 5M )
    - 客户端完成，不会请求服务器处理
    - sessionStorage数据是不共享、 localStorage共享
  + Cookie的特点
    - 存储限制
* 域名100个cookie,每组值大小4KB
  + - 客户端、服务器端，都会请求服务器（头信息）
    - 页面间的cookie是共享

# JS中的作用域与作用域链

* 作用域：js中的函数作用域是指在函数内声明的所有变量在函数体内始终是可见的。并且，变量在声明之前就可以使用了，这种情况就叫做声明提前（hoisting)
* 作用域链：由于js的变量都是对象的属性，而该对象可能又是其它对象的属性，而所有的对象都是window对象的属性，所以这些对象的关系可以看作是一条链   
  链头就是变量所处的对象，链尾就是window对象

# JS组成

* JavaScript是一种描述语言，基于对象和时间驱动。javascript应用程序都要下载到浏览器的客户端执行，从而减轻了服务器端的负担
* JavaScript由三部分组成：

a、核心与法(ECMAScript)：是 JavaScript的标准规范

b、浏览器对象模型(BOM)：与浏览器窗口进行交互

c、文档对象模型(DOM)：可以控制页面中的每一个元素，实现页面中的各种动态效果

* JS的执行过程：

a.浏览器接受用户的请求

b.向服务器端请求某个包含JS脚本的页面(浏览器把请求信息发送到应用服务器，等待响应)

c.应用服务器端向浏览器发送相应消息

# JS数据类型

基本数据类型：字符串string; 数字 number; 布尔boolean

复合数据类型：数组 array; 对象 object

特殊数据类型：空对象 null; 未定义 undefined

# get和post的区别

get 是从服务器上获取数据，post 是向服务器传送数据

1. [GET](http://www.nowamagic.net/librarys/veda/tag/GET)使用URL或Cookie传参。而POST将数据放在BODY中。
2. GET的URL会有长度上的限制，则POST的数据则可以非常大。
3. POST比GET安全，因为数据在地址栏上不可见。

1. get 是从服务器上获取数据，post 是向服务器传送数据。

get 请求返回 request - URI 所指出的任意信息。

Post 请求用来发送电子邮件、新闻或发送能由交互用户填写的表格。这是唯一需要在请求中发送body的请求。使用Post请求时需要在报文首部 Content - Length 字段中指出body的长度。

2. get 是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址，用户看不到这个过程。

3. 对于 get 方式，服务器端用Request.QueryString获取变量的值，对于 post 方式，服务器端用Request.Form获取提交的数据。

4. get 传送的数据量较小，不能大于2KB。post 传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。 用IIS过滤器的只接受get参数，所以一般大型搜索引擎都是用get方式。

5. get 安全性非常低，post 安全性相对较高。如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post 为好。

# 绑定事件与普通事件的区别

* 普通事件: 同一个元素，加相同的事件, 调用不同的函数，后面的会覆盖前面
* 绑定事件不会有这种问题

//绑定兼容

function bindEvent(obj,events, fn){

if (obj.addEventListener) {

obj.addEventListener(events, fn, false);

}

else

{

obj.attachEvent('on'+events, fn);

obj[fn]=function()

{fn.call(this);}

}

};

# addEventListener和attachEvent区别（IE和DOM事件流的区别）:

1. 参数是不一样的

2. 事件加不加on

3. 执行顺序不一样(addEventListener顺序执行

attachEvent逆序执行 )

4. this指向的问题（this -> div）:IE6~IE10都存在一个问题this -> window(设计上的失误)

# 取消绑定事件和普通事件

* 普通事件：oDiv.onclick=null;

绑定事件：//取消事件兼容

function delEvent(obj, events, fn){

if(removeEventListener){

obj.removeEventListener(events,

fn, false);

}

else

{

obj.detachEvent('on'+events,obj[fn]);

}

};

# call和apply的区别

* call 方法:调用一个对象的一个方法，以另一个对象替换当前对象

obj.call(thisobj,param1,param2)

* apply方法：应用某一对象的一个方法，用另一个对象替换当前对象

obj.apple(thisobj,[param1,param2])

# 事件委托

* 原理：利用冒泡原理实现
* 好处：提高性能（不需要循环操作）；后续添加内容直接拥有事件行为

# 事件源

* target

oUl.onclick = function(ev){

//this.style.background = 'red';

var ev = ev||window.event;

var target = ev.target || ev.srcElement;

//target.style.background = 'red';

if (target.nodeName.toLowerCase() == 'li')

{

target.style.background = 'red';

}

};

# JS鼠标坐标

ev.clientX ev.clientY—相对于可视区

ev.pageX ev.pageY-相对于整个屏幕--鼠标坐标相对于整个页面， IE6~8不支持该方法

# onmousedown onmouseup onclick 顺序触发

# 阻止冒泡事件

function stopBubble(e) {

//如果提供了事件对象，则这是一个非IE浏览器

if(e&& e.stopPropagation) {

　　//因此它支持W3C的stopPropagation()方法

　　e.stopPropagation();

} else {

　　//否则，我们需要使用IE的方式来取消事件冒泡

　　window.event.cancelBubble = true;

}

return false;

}

# 阻止默认事件

* function stopDefault(e) {

//如果提供了事件对象，则这是一个非IE浏览器

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;

}

* if (ev.preventDefault)

{

ev.preventDefault();

}

else

{

return false;

}

# 到整个页面的距离

function postLeft(obj){

var result=0;

while(obj)

{

result += obj.offsetLeft; //无定位: obj = obj.offsetParent; //有定位: }

return result;

};

# oDiv的宽度

console.log(oDiv.offsetWidth); //width + padding + border

console.log(oDiv.clientWidth); //width + padding

# 滚动条的高度

//获取

function getScrollY(){

return window.pageYOffset ||

document.documentElement.scrollTop;

};

//设置

function setScrollY(num){

document.documentElement.scrollTop(chrome不支持) = document.body.scrollTop(仅chrome支持)= num;

};

# 可视区的高度

function viewWidth(){

return document.documentElement.clientWidth;

};

# 文档的高度

document.body.offsetHeight//可能大于可视区， 可能小于可视区

# JQ可视区的宽度、页面的宽度、滚动距离

//$(window).width(); //可视区宽

//$(window).height(); //可视区高

//$(document).width(); //整个页面的宽

//$(document).height(); //整个页面的高

//alert($(document).scrollTop());

xhtml和html有什么区别  
HTML是一种基本的WEB网页设计语言，XHTML是一个基于XML的置标语言

最主要的不同：

XHTML 元素必须被正确地嵌套。

XHTML 元素必须被关闭。

标签名必须用小写字母。

XHTML 文档必须拥有根元素。

# CSS引入的方式有哪些? link和@import的区别是?

内嵌、外链、导入；

* link引用CSS时，在页面载入时同时加载；@import需要页面网页完全载入以后加载。
* link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。
* link支持使用Javascript控制DOM去改变样式；而@import不支持。
* link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。

# CSS选择符有哪些?哪些属性可以继承?优先级算法如何计算?内联和important哪个优先级高?

id选择符（#）

群组选择符（，）

class选择符（.）

类型选择符(div……)

包含选择符（div p）

通配符（\*）

* 不可继承的：display、margin、border、padding、background、height、min-height、max-height、width、min-width、max-width、overflow、position、left、right、top、bottom、z-index、float、clear、table-layout、vertical-align、page-break-after、page-bread-before和unicode-bidi。
* 所有元素可继承：visibility和cursor。
* 内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。
* 终端块状元素可继承：text-indent和text-align。
* 列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。
* 表格元素可继承：border-collapse。
* 优先级问题：!important>（JS修改-各浏览器不一样）>行间>id>class>类型

# Html中 input加上 readonly 和 disable的区别

* Readonly和Disabled它们都能够做到使用户不能够更改表单域中的内容。但是它们之间有着微小的差别，总结如下：
* Readonly只针对input(text / password)和textarea有效，而disabled对于所有的表单元素都有效，但是表单元素在使用了disabled后，当我们将表单以POST或GET的方式提交的话，这个元素的值不会被传递出去，而readonly会将该值传递出去（readonly接受值更改可以回传，disable接受改但不回传数据）。

# 响应式布局原理

## **第一步：Meta标签**

大多数移动浏览器将HTML页面放大为宽的视图(viewport)以符合屏幕分辨率。你可以使用视图的meta标签来进行重置。下面的视图标签告诉浏览器，使用设备的宽度作为视图宽度并禁止初始的缩放。在标签里加入这个meta标签。

<meta name="viewport" content="width=device-width, initial-scale=1

user-scalable=no">

IE8或者更早的浏览器并不支持Media Query。你可以使用media-queries.js或者respond.js来为IE添加Media Query支持。

<!--[if lt IE 9]>

<script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>

<![endif]-->

## 第二步：HTML结构

由于网页会根据屏幕宽度调整布局，所以不能使用绝对宽度的布局，也不能使用具有绝对宽度的元素。这一条非常重要。

字体也不能使用绝对大小(px)，而只能使用相对大小(em)。

## 第三步：媒介查询 -Media Queries

CSS3 Media Query-媒介查询是响应式设计的核心。它根据条件告诉浏览器如何为指定视图宽度渲染页面。

媒介查询的目的在于为指定的视图宽度指定不同的CSS规则，来实现不同的布局。媒介查询可以写在同一个或者单独的样式表中。

# Angula.js

AngularJS 诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：MVVM、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。

# Ajax相关

Asynchronous JavaScript and XML（异步JavaScript和XML)

* 节省用户操作，时间，提高用户体验，减少数据请求
* 传输获取数据，无需刷新页面，响应用户请求

## 步骤：

var xhr = null;

/\*

if(window.XMLHttpRequest)

{

xhr = new XMLHttpRequest();

}else{

xhr = ActiveXObject('Microsoft.XMLHTTP');

}

\*/

//异常捕获处理

try{

xhr = new XMLHttpRequest();

}catch(e){

xhr = ActiveXObject('Microsoft.XMLHTTP');

}

//2.在地址栏输入地址

/\*

open方法:

参数:

1. 打开方式: 地址栏-get方式 表单-post方式

2. 地址

3. 是否异步

异步: 非阻塞 前面的代码不会影响后面代码的执行

同步: 阻塞 前面的代码会影响后面代码的执行

\* \*/

xhr.open('get', '1.txt', true);

//true-异步 false-同步

//3. 提交

//alert(1);

xhr.send();

//alert(xhr.responseText);

//4. 等待服务返回内容

xhr.onreadystatechange = function(){

if(xhr.readyState == 4){

alert(xhr.responseText); //返回内容

};

};

## readyState属性：请求状态

* + 0 （未初始化）还没有调用open()方法
  + 1 （载入）已调用send()方法，正在发送请求
  + 2 （载入完成）send()方法完成，已收到全部响应内容
  + 3 （解析）正在解析响应内容
  + 4 （完成）响应内容解析完成，可以在客户端调用了

# Ajax封装

function ajax(method,url,data,success){

//1.打开浏览器

var xhr = null;

//异常捕获处理

try{

xhr = new XMLHttpRequest();

}catch(e){

xhr = ActiveXObject('Microsoft.XMLHTTP');

}

if (method=='get'&& data) {

url += '?'+data;

}

//2.在地址栏输入地址

xhr.open(method, url, true);

//true-异步 false-同步

//3. 提交

if (method=='get') {

xhr.send();

}else{

xhr.setRequestHeader('content-type',

'application/x-www-form-urlencoded');

xhr.send(data);

}

//4. 等待服务返回内容

xhr.onreadystatechange = function(){

if(xhr.readyState == 4){

if(xhr.status == 200)

{

success && success(xhr.responseText);

//返回内容(字符串)

}

else{

alert('出错了,err:'+xhr.status);

}

};

};

};

# 中文编码：encodeURI()

# 跨域请求

跨域请求（访问）：一个域名下的文件请求另外一个域名下的资源，就产生了跨域

## 跨域的解决

1. Jsonp : json with padding
2. Local proxy(本地代理)
3. Flash(服务器：xml/file)

## Jsonp:

* 用script标签的src加载需要的资源
* src的作用 ： 加载（包含指定的外部文件）
* 可以跨域包含
* 被包含的资源可以是任何类型的文件(可以是txt,php等)
* 他只关注被包含的文件的内容是否是合法的JS

# 面向对象

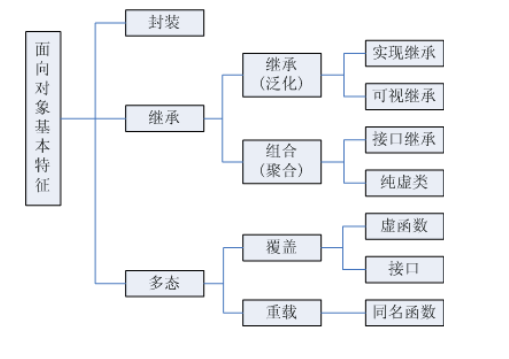
## 概念

使用对象时，只关注对象提供的功能，不关注其内部细节

比如: JQuery

## 特点

1. 继承； 从已有对象上，继承出新的对象
2. 封装；不考虑内部实现，只考虑功能使用
3. 多态；允许将子类类型的指针赋值给父类类型的指针



## 对象的组成

方法 -- 函数： 过程、动态的

属性 -- 变量 : 状态、静态的

var obj = new Object();

//属性

obj.aaa = 12;

//方法

obj.show=function(){

alert(this.aaa);

};

obj.show();

## 工厂方式：构造函数（--JS高级-面向对象）

问题：1. 没有new

2. 每个对象都有自己的函数(资源)

## Prototype(原型)

var arr1 = new Array(1,2,3,4,6,7,8,9,10);

var arr2 = new Array(5,6,7,8);

Array.prototype.sum=function(){

var result=0;

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

result += this[i];

}

return result;

};

//alert(arr1.sum());

//alert(arr2.sum());

alert(arr1.sum == arr2.sum)// true;

## 构造函数+原型

function CreatePerson(name, sex) //构造函数

{

//属性: 每个对象都各不相同

this.name = name;

this.sex = sex;

};

//方法：所有对象都一样

CreatePerson.prototype.showName=function(){

alert('我的名字:'+this.name);

};

CreatePerson.prototype.showSex=function(){

alert('我的性别:'+this.sex);

};

var p1=new CreatePerson('如花', '男');

var p2=new CreatePerson('花花', '女');

alert(p1.showName == p2.showName);//true

p1.showName();

p1.showSex();

p2.showName();

p2.showSex();

# 数组去重

## 方法一

var arr1=[2,3,3,4,32,4,32,43,32,2,11,43]

var arr2=[3,3,4,5,5,3,2343,32432,35,2,35]

Array.prototype.unique4 = function()

{

this.sort();

var ac=[this[0]];

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

{

if (this[i]!=ac[ac.length-1])

{

ac.push(this[i]);

}

}

return ac;

}

console.log(arr1.unique4());

console.log(arr2.unique4());

## 方法二

var arr=[2,3,3,4,32,4,32,43,32,2,11,43]

var arr2=[3,3,4,5,5,3,2343,32432,35,2,35]

Array.prototype.unique1 = function()

{

var n = []; //一个新的临时数组

for(var i = 0; i < this.length; i++) //遍历当前数组

{

//如果当前数组的第i已经保存进了临时数组，那么跳过，

//否则把当前项push到临时数组里面

if (n.indexOf(this[i]) == -1) n.push(this[i]);

}

return n;

}

console.log(arr1.unique1());

console.log(arr2.unique1());

# 兼容性问题：

## 1.在IE6下，内容会撑开设置好的宽高

计算一定要精确 不要让内容的宽高超出我们设置的宽高:

## 2. 在IE6下元素浮动，如果宽度需要由内容撑开，就给里边的元素添加浮动

## 3. IE6最小高度问题

在IE6下元素的高度小于19px，会被当做19px来进行处理

解决： overflow:hidden;

## 4.border: 1px(2px) dotted black在IE6不支持，会显示dashed效果；