1. cookie和localstorge，sessionstorge在原理上有什么不同？

共同点：都是保存在浏览器端，且同源的。

区别：

1. cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递；cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。存储大小限制也不同，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，如会话标识。

2. 而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。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。

3. 数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；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过期时间之前一直有效，即使窗口或浏览器关闭。

4. 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；localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 api 接口使用更方便。

1. css3动画中的阴影，位移，旋转？

box-shadow translate rotate

1. 如何用表单存储数据？

???

1. 如何得知页面布局是弹性盒布局？

1：看父盒子是否有属性display:flex;

2：看代码宽度是否是百分比

3：看页面效果如缩小页面是否影响

1. Css定义的权重？

行内样式>ID选择器>类，属性选择器和伪类选择器>元素和伪元素

1. dom和bom有什么区别？

DOM是文档对象模型用来用来获取设置文档中标签的属性。操作对象是文档。

Bom是浏览器对象模型，获取设置浏览器属性，行为。操作对象是浏览器。

1. 原生js中数组去重？

//数组去重

var arr=[1,2,4,1,2,3]

var arr1=[]

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

if(arr1.indexOf(arr[i])<0){

arr1.push(arr[i])

}

}

1. DOM节点增删的方法？

创建新节点

1. createDocumentFragment() 创建一个DOM片段

2. createElement\_x() 创建一个具体的元素

3. createTextNode() 创建一个文本节点

添加、移除、替换、插入

1．appendChild() 添加节点

2．removeChild() 移除节点

3．replaceChild() 替换节点

4．insertBefore() 插入节点

1. 数组入队出队，入栈出栈的方法？

push() pop() shift() unshift() splice()

1. 什么是Ajax？原生js中ajax的四步?jquery中ajax的请求方法？

复习大全37,38 JQ中请求Ajax.$.ajax({type:””,url:””,async:””,data:””,success:””,error:””})

1. for循环如何优化？

把var i = 0;以及获取数组长度放在for循环外边

1. 原生js中如何在不定义第三个变量的情况下，交换前两个变量的值？

var a = 6;

var b = 2;

b = b-a;

a = b+a;

b = a-b

console.log(a,b)

1. 排序算法？

**冒泡法**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var arr = [3,6,1,2,5];  var temp;  for(var i= 0;i<arr.length;i++){  for(var j=i+1;j<arr.length;j++){  if(arr[i] > arr[j]){  temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  console.log(arr); |

**快速排序法**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function quicksort (arr){   if(arr.length<=1){   return arr;   }   var left = [];   var right = [];   var middle = arr[0];   for(var i=1;i<arr.length;i++){   if(arr[i]<middle){   left.push(arr[i]);   }else{   right.push(arr[i]);   }   }   return quicksort(left).concat([middle],quicksort(right));  } |

1. 定义一个局部变量，想在全局中使用它，通过什么方法？

声明提升

function内的新声明的变量才是局部变量，而没有用var声明的变量在哪里都是全局变量。再次提醒切记只有function(){}内新声明的才能是局部变量，while{…}、if{…}、for(..) 之内的都是全局变量（除非本身包含在function内）

1. 描述一下事件流？

Javascript的事件流模型都有什么？

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

1. 面向对象的继承方式，写具体代码？

（1）构造继承

（2）原型继承

（3）实例继承

（4）拷贝继承

原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。

Javascript:

function Parent(){

this.name = 'wang';

}

function Child(){

this.age = 28;

}

Child.prototype = new Parent();//继承了Parent，通过原型

var demo = new Child();

alert(demo.age);

alert(demo.name);//得到被继承的属性

1. js中如何实现跨域，实现跨域的原理是什么？

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

1. 谈谈css预处理？

less/sass/compass 是一种动态样式语言. 将CSS赋予了动态语言的特性，如变量，继承，运算， 函数. LESS 既可以在客户端上运行 (支持IE 6+, Webkit, Firefox)，也可一在服务端运行 (借助 Node.js)。

为什么要使用它们？

* 结构清晰，便于扩展。
* 可以方便地屏蔽浏览器私有语法差异。这个不用多说，封装对浏览器语法差异的重复处理，减少无意义的机械劳动。
* 可以轻松实现多重继承。
* 完全兼容 CSS 代码，可以方便地应用到老项目中。 只是在 CSS 语法上做了扩展，所以老的 CSS 代码也可以一同编译。

1. js运算符的优先级？

文档里的大表格

1. 写出一些常见的状态码，并说明其意思？

http状态码：

100-199 用于指定客户端相应胡某些动作

200-299 用于表示请求成功

400-499 用于指出客户端的错误

400 语义有误，当前请求无法被服务器理解

401 当前请求需要用户验证

403 服务器已理解，但是拒绝执行它

500-599 用于指出服务器的错误

503 服务器不可用

1. angular的特点和使用方式

mvvm 模块化 自动化双向数据绑定 语义化标签 依赖注入

使用： 引入angular.js文件，下载依赖模块

1. jquery的ready事件，如果用原生js怎么写？

自行百度

1. 是否了解重构？

重构是通过调整程序代码改善软件质量，性能，使程序的设计模式和架构更合理，提高软件的扩展性和维护性

1. JS设计模式了解哪些？

单体模式 工厂模式 单例模式 观察者模式 等23种..

1. 为什么要用自执行函数？

自执行函数就是函数的封闭空间 （function（）{}）（）；隔离作用域，解决命名冲突问题

1. 什么是回调函数？回调函数是为了解决什么问题？

函数A作为参数(函数引用)传递到另一个函数B中，并且这个函数B执行函数A。我们就说函数A叫做回调函数。如果没有名称(函数表达式)，就叫做匿名回调函数。

1. 了解几种的html框架？js框架？它们的优势是什么？

Html: bootstrap

Js: angular 模板功能丰富 自带丰富的指令 依赖注入 数据双向绑定 路由 服务 过滤器 功能强大

vue 轻量级的框架 插件化 数据双向绑定 指令

react 代码模块化重用更容易 虚拟DOM 操作单向数据流

jquery 轻量级 丰富的DOM 操作 链式表达 可扩展 ajax操作 代码可读性增强（动画 样式 事件）

1. javascript的typeof返回哪些数据类型？

String number boolean object array date undefined null function

1. ajax请求的时候get 和post方式的区别？

Get 只能发送4KB的数据，post可以发送1GB 以上

Get的参数是拼接 URL后面，而Post不是，post需要设置请求头

Get 里面的send不需要设置内容，而post需要在send 里面放参数

Get没有post安全

1. ajax的增删改查功能分别怎么写？

自行百度

1. 拿到多条json数据以后怎么做数据的循环？如何通过循环来渲染dom页面？

把json数据转换成Object ,利用Object 的遍历方法,如keys, values, entries 等,再通过for of循环, .将需要的数据渲染出来, 整合到相应的DOM元素, 并进行页面的插入节点操作.

1. call和apply的区别？

作用：

call和apply都可以让B对象调用A对象的方法或属性，并且修改了this的当前作用对象。

区别：

1. call，apply都属于Function.prototype的一个方法，它是JavaScript引擎内在实现的，因为属于Function.prototype，所以每个Function对象实例(就是每个方法)都有call，apply属性。既然作为方法的属性，那它们的使用就当然是针对方法的了，这两个方法是容易混淆的，因为它们的作用一样，只是使用方式不同。

2. 语法：foo.call(this, arg1,arg2,arg3) == foo.apply(this, arguments) == this.foo(arg1, arg2, arg3);

3. 相同点：两个方法产生的作用是完全一样的。

4. 不同点：方法传递的参数不同。call的第二个及其以后的参数都是数组里面的元素，apply的第二个参数是一个参数数组

5. bind不会立即调用，call和apply是立即调用

1. 事件委托是什么？

使用事件委托技术能让你避免对特定的每个节点添加事件监听器；相反，事件监听器是被添加到它们的父元素上。事件监听器会分析从子元素冒泡上来的事件，找到是哪个子元素的事件。

事件委托就是利用事件冒泡，只指定一个事件处理程序，就可以管理某一类型的所有事件。

事件委托就是利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

事件冒泡，就是从最深的节点开始，逐渐向上传播事件。

1. 闭包是什么，有什么特性，对页面有什么影响？

闭包是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。

相关连接：http://www.jb51.net/article/24101.htm

1. 如何阻止事件冒泡和默认事件？

在元素中return false

1. 添加 删除 替换 插入到某个接点的方法?

添加、移除、替换、插入

1．appendChild() //添加节点

2．removeChild() //移除节点

3．replaceChild() //替换节点

4．insertBefore() //插入节点

1. javascript的本地对象，内置对象和宿主对象?

本地对象:

一、Array,Boolean,Date,Function,Global,Math,Number,Object,RegExp,String...

二、错误类对象 Error,RangeError,TypeError,EvalError,ReferenceError...

三、内置对象 Global和Math

内置对象: Global和Math

宿: 一、执行JS脚本环境提供的对象.对于嵌入网站中的JS来说,其宿主对象就是浏览器提供的对象,又称浏览器对象.

二、不同浏览器提供的宿主对象可能不同

三、即使宿主对象相同,其实现方式也可能不同

自定义对象: 开发者自己定义的对象,使应用及功能得到拓展.

1. 什么是强类型语言和弱类型语言?

强: 强类型定义语言,素有变量都必须先定义,一旦对其指定了数据类型,不经过强制转换,就永远是这个数据类型;

弱: 忽略数据类型,一个变量可以赋予不同的数据类型.

1. 脚本语言和编译语言的区别是什么？JS属于哪种语言类型？

脚本语言&编译语言

脚: 执行简单任务的解释性语言,一般没有 类 或者 此类 高级功能;

编: 在编译前将脚本语言转换为可以执行代码的语言.

1. 在JS中有哪些数据类型？

基本数据类型：number string null undefined boolean

引用类型： object

1. 基本类型和引用类型有什么区别？它们的存储原理是什么？

基本数据类型： 创建时，在栈上给其划分一块内存，将数值直接存储在栈上；

引用数据类型： 创建时，先在栈上给其引用分配一块内存，而对象的具体信息都存储在堆里，然后由栈上面的引用指向堆中的对象的地址。

1. 什么是局部变量和全局变量？在JS中最大的全局变量是谁？

1.作用域不同：作用域为整个程序，而局部变量的作用域为当前函数或循环等；

2.内存存储方式不同：全局变量存储在全局数据区中，局部变量存储在栈区；

3.生命期不同：全局变量的生命期和主程序一样，随程序的销毁而销毁，局部变量在函数内部或循环内部，随函数的退出或循环退出就不存在；

4. 使用方式不同：全局变量在声明后程序的各个部分都可以用到，但是局部变量只能在局部使用。函数内部会优先使用局部变量再使用全局变量；

js中最大的全局变量是 ： window

1. 作用域链?

在JavaScript中，函数也是对象，实际上，JavaScript里一切都是对象。函数对象和其它对象一样，拥有可以通过代码访问的属性和一系列仅供JavaScript引擎访问的内部属性。其中一个内部属性是[[Scope]]，由ECMA-262标准第三版定义，该内部属性包含了函数被创建的作用域中对象的集合，这个集合被称为函数的作用域链，它决定了哪些数据能被函数访问。

1. JS中内置了哪些常用的对象？

1. Array

2. String

3. Date

4. Math

5. Boolean

6. Number

7. Function

8. Global

9. Error

10.RegExp

11.Object

1. javascript继承的6种方法以及优点和缺点?

（一）简单原型链

具体实现

function Super(){

this.val = 1;

this.arr = [1];

}

function Sub(){

// ...

}

Sub.prototype = new Super(); // 核心

var sub1 = new Sub();

var sub2 = new Sub();

sub1.val = 2;

sub1.arr.push(2);

alert(sub1.val); // 2

alert(sub2.val); // 1

alert(sub1.arr); // 1, 2

alert(sub2.arr); // 1, 2

核心

拿父类实例来充当子类原型对象

优缺点

优点：

1．简单，易于实现

缺点：

1. 修改sub1.arr后sub2.arr也变了，因为来自原型对象的引用属性是所有实例共享的。

2. 可以这样理解：执行sub1.arr.push(2);先对sub1进行属性查找，找遍了实例属性（在本例中没有实例属性），没找到，就开始顺着原型链向上找，拿到了sub1的原型对象，一搜身，发现有arr属性。于是给arr末尾插入了2，所以sub2.arr也变了

3. 创建子类实例时，无法向父类构造函数传参

（二）借用构造函数

function Super(val){

this.val = val;

this.arr = [1];

this.fun = function(){

// ...

}

}

function Sub(val){

Super.call(this, val); // 核心

// ...

}

var sub1 = new Sub(1);

var sub2 = new Sub(2);

sub1.arr.push(2);

alert(sub1.val); // 1

alert(sub2.val); // 2

alert(sub1.arr); // 1, 2

alert(sub2.arr); // 1

alert(sub1.fun === sub2.fun); // false

核心

借父类的构造函数来增强子类实例，等于是把父类的实例属性复制了一份给子类实例装上了（完全没有用到原型）

优缺点

优点：

1. 解决了子类实例共享父类引用属性的问题

2. 创建子类实例时，可以向父类构造函数传参

缺点：

1．无法实现函数复用，每个子类实例都持有一个新的fun函数，太多了就会影响性能，内存爆炸。

（三）组合继承（最常用）

function Super(){

// 只在此处声明基本属性和引用属性

this.val = 1;

this.arr = [1];

}

// 在此处声明函数

Super.prototype.fun1 = function(){};

Super.prototype.fun2 = function(){};

//Super.prototype.fun3...

function Sub(){

Super.call(this); // 核心

// ...

}

Sub.prototype = new Super(); // 核心

var sub1 = new Sub(1);

var sub2 = new Sub(2);

alert(sub1.fun === sub2.fun); // true

核心

把实例函数都放在原型对象上，以实现函数复用。同时还要保留借用构造函数方式的优点，通过Super.call(this);继承父类的基本属性和引用属性并保留能传参的优点；通过Sub.prototype = new Super();继承父类函数，实现函数复用

优缺点

优点：

1. 不存在引用属性共享问题

2. 可传参

3. 函数可复用

缺点:

（一点小瑕疵）子类原型上有一份多余的父类实例属性，因为父类构造函数被调用了两次，生成了两份，而子类实例上的那一份屏蔽了子类原型上的。。。又是内存浪费，比刚才情况好点，不过确实是瑕疵

（四）寄生组合继承（最佳方式）

function beget(obj){ // 生孩子函数 beget：龙beget龙，凤beget凤。

var F = function(){};

F.prototype = obj;

return new F();

}

function Super(){

// 只在此处声明基本属性和引用属性

this.val = 1;

this.arr = [1];

}

// 在此处声明函数

Super.prototype.fun1 = function(){};

Super.prototype.fun2 = function(){};

//Super.prototype.fun3...

function Sub(){

Super.call(this); // 核心

// ...

}

var proto = beget(Super.prototype); // 核心

proto.constructor = Sub; // 核心

Sub.prototype = proto; // 核心

var sub = new Sub();

alert(sub.val);

alert(sub.arr);

核心

用beget(Super.prototype);切掉了原型对象上多余的那份父类实例属性

优缺点

优点：完美了

（五）原型式

function beget(obj){ // 生孩子函数 beget：龙beget龙，凤beget凤。

var F = function(){};

F.prototype = obj;

return new F();

}

function Super(){

this.val = 1;

this.arr = [1];

}

// 拿到父类对象

var sup = new Super();

// 生孩子

var sub = beget(sup); // 核心

// 增强

sub.attr1 = 1;

sub.attr2 = 2;

//sub.attr3...

alert(sub.val); // 1

alert(sub.arr); // 1

alert(sub.attr1); // 1

核心

用生孩子函数得到得到一个“纯洁”的新对象（“纯洁”是因为没有实例属性），再逐步增强之（填充实例属性）

优缺点

优点：

从已有对象衍生新对象，不需要创建自定义类型（更像是对象复制，而不是继承。。）

缺点：

1. 原型引用属性会被所有实例共享，因为是用整个父类对象来充当了子类原型对象，所以这个缺陷无可避

2. 无法实现代码复用（新对象是现取的，属性是现添的，都没用函数封装，怎么复用）

（六）寄生式

function beget(obj){ // 生孩子函数 beget：龙beget龙，凤beget凤。

var F = function(){};

F.prototype = obj;

return new F();

}

function Super(){

this.val = 1;

this.arr = [1];

}

function getSubObject(obj){

// 创建新对象

var clone = beget(obj); // 核心

// 增强

clone.attr1 = 1;

clone.attr2 = 2;

//clone.attr3...

return clone;

}

var sub = getSubObject(new Super());

alert(sub.val); // 1

alert(sub.arr); // 1

alert(sub.attr1); // 1

核心

给原型式继承穿了个马甲而已，看起来更像继承了（上面介绍的原型式继承更像是对象复制）

注意：beget函数并不是必须的，换言之，创建新对象 -> 增强 -> 返回该对象，这样的过程叫寄生式继承，新对象是如何创建的并不重要（用beget生的，new出来的，字面量现做的。。都可以）

优缺点

优点：

还是不需要创建自定义类型

缺点：

无法实现函数复用

1. 匿名函数?

函数字面量：首先声明一个函数对象，然后执行它。

(function () {

alert(1);

})();

优先表达式：由于Javascript执行表达式是从圆括号里面到外面，所以用圆括号强制执行声明的函数。

(function () {

alert(2);

}());

Void操作符：用void操作符去执行一个没有用圆括号包围的一个单独操作数。

void function () {

alert(3);

}()

匿名函数的应用

《Javascript的一种模块模式》中的第一句话就是“全局变量是魔鬼”。配合var关键字，匿名函数可以有效的保证在页面上写入Javascript，而不会造成全局变量的污染。这在给一个不是很熟悉的页面增加Javascript时非常有效，也很优美。实际上，YUI以及其相应的范例中大量使用匿名函数，其他的Javascript库中也不乏大量使用。

1. 模块化的概念?

模块化概念是将系统分离成独立功能部分的方法。

严格定义模块接口模块间具有透明性。

可维护性，方便单个模块功能的调试，多人协作互补干扰，可分单元测试

1. webpack的配置方式?

课堂笔记

1. pc端开发流程？

了解需求（后台接口，功能点，开发时间） --> 明确技术类型（页面布局）用什么工具 --> 分配任务 --> 后续问题(修复bug) --> 总结

1. 移动端开发流程？

产品经理提出需求（原型） --> 讨论需求 --> 通过需求 --> 设计 ui交互 --> 交付工程师开发 --> 修改细节和bug --> 上线

1. 代码优化做过哪些？

所谓的优化就是程序代码进行等价的变化（再不改变程序的运行结果的情况下。生成更加高效的目标代码）

途径： 1)改进算法

2)在源程序上进行等价变换

3）充分利用系统的程序库

4）编译时的优化

1. 做过哪些兼容性问题？

1.关于获取行外样式currentStyle和getComputedStyle 出现的兼容问题

IE: currentStyle

Chrome,ff : getComputedStyle(div,false)

function getStyle(obj，name){

if(obj.currentStyle){

return obj.currentStyle[name];

}else{

return getComputedStyle(obj,false)[name];

}

};

2.事件对象

document.onclick = function(evt){

evt = evt || window.event;

console.log(evt)

}

3.获取滚动条距离

IE,Chrome : document.body.scrollTop

FF : document.documentElement.scrollTop

var scrollTop = document.documentElement.scrollTop || document.body.scrollTop