**1. JSON数据的格式要求:**

**字符串与属性名必须使用双引号**

**数据由逗号分隔**

**不能有注释**

**不能有多余的逗号**

**JSON 数据的书写格式是：名称/值对。**

**名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值："firstName" : "John"**

**值必须为以下类型：**

**Number   数字（整数或浮点数）**

**String     字符串（在双引号中）**

**Boolean   逻辑值（true 或 false）**

**Array      数组（在方括号中）**

**Object    对象（在花括号中）**

**null       null**

**常用方法：JSON.stringify()**

**JSON.parse()**

**2. 类数组与数组的区别:**

**类数组：**

**拥有length属性，其它属性（索引）为非负整数（对象中的索引会被当做字符串来处理）;**

**不具有数组所具有的方法；**

**类数组是一个普通对象，而真实的数组是Array类型。**

**常见的类数组有:**

**函数的参数arugments,**

**DOM对象列表(比如通过document.querySelectorAll 得到的列表),**

**jQuery 对象 (比如 $("div")).**

**类数组的原型对象与数组不同，所以不能使用数组的方法；**

**类数组如何使用数组的方法：**

**Array.prototype.map.call(jiaArray,function(item,idx,arr){})**

**3. Cookie/webStorage/sessiond的区别：**

**应用场景的区别**

**Cookie的应用场景**

**（1）判断用户是否登录过网站，以便下次登录时能够实现自动登录（或者记住密码）。**

**（2）保存上次登录的事件等信息。**

**（3）保存上次查看的页面**

**（4）浏览计数**

**Session的应用场景**

**（1）网上商城中的购物车**

**（2）保存用户登录信息**

**（3）将某些数据放入session中，供同一用户的不同页面使用**

**（4）防止用户非法登录**

**除此之外Session的安全性要高于Cookie ，Cookie只能保存字符串类型，Cookie的大小有限制，保存在客户端，session保存数据大小无限制，在服务器端；**

**HTML5的WebStorage提供了两种API：localStorage（本地存储）和sessionStorage（会话存储）**

**SessionStorange ：数据大小(2M) 有效期(Session)无法设置有效期 发送网络请求的时候不会提交到服务器端 H5新推出**

**LocalStorage ：数据大小(20M) 永久有效保存到电脑的硬盘中**

**附：Cookie的特点:**

**(1)使用cookie来进行数据存储的大小有限制，4K每个**

**(2)每个网页中存储cookie的个数（最多50）和每个网站中存储cookie的个数都有限制（200）**

**各个不同的浏览器对cookie的数量也不相同**

**(3) Cookie有效期(Session)**

**(4) Cookie数据不能跨浏览器访问**

**(5) 每次发送网络请求的时候，Cookie数据都会默认一起被提交到服务器端**

**cookie格式：name=value;expires=xxx;path=/html/img;domain=xxx**

**4. H5新特性：**

**（一） 语义标签**

**<footer></footer> 这种的一堆**

**（二）增强型表单**

**H5中的input type ：email/url/number/tel/search /range /color/month/week /date**

**H5中新增的表单元素：datalist /progress/meter/output**

**（三）视频和音频**

**视频播放：<video src=""><video>**

**音频播放：<audio src=""></audio>**

**（四）Canvas绘图**

**<canvas id="lala" width="400" height="300"></canvas>**

**（五）SVG绘图**

**（六）地理定位**

**（七）拖放API**

**（八） WebWorker**

**（九） WebStorage:**

**sessionStorage**

**localStorage**

**（十）WebSocket**

**（十一）History**

**（十二）FormData**

**、、、、、、、、、、、、、、、、、、、、、、、、、、、**

1. **未知宽高的图片居中**

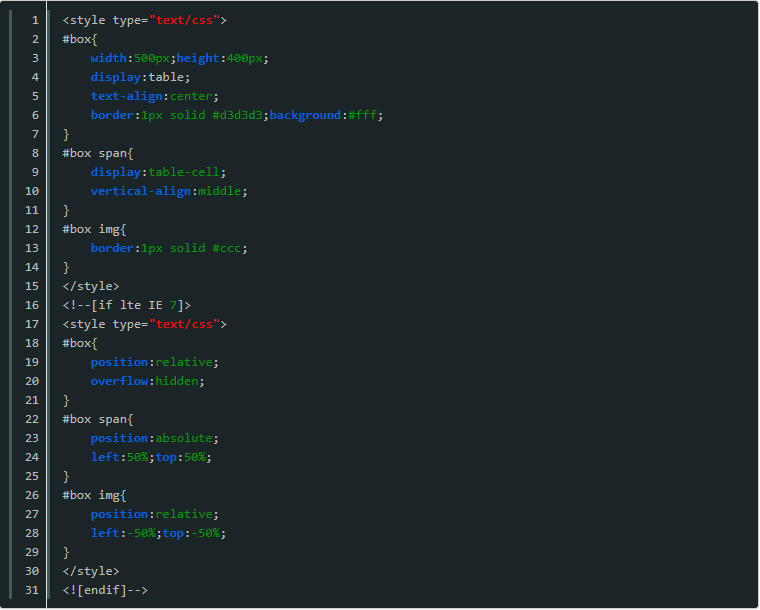
## **方法一：**

## 该方法是将外部容器的显示模式设置成display:table，img标签外部再嵌套一个span标签，并设置span的显示模式为display:table-cell，这样就可以很方便的使用vertical-align象表格元素那样对齐了。

## HTML代码



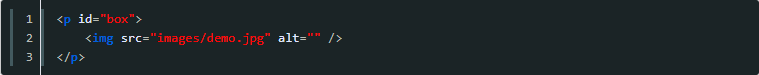
**CSS代码**

****

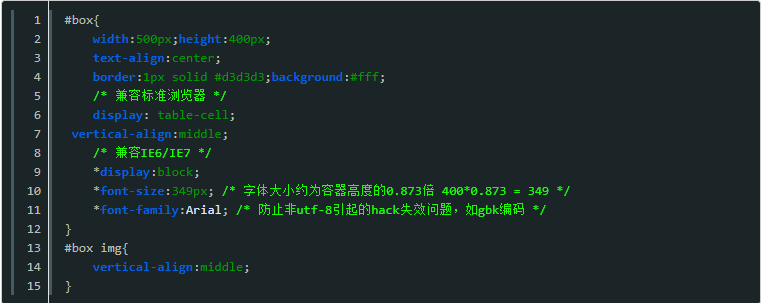
**兼容IE6/IE7**

## 该方法针对IE6/IE7，将图片外部容器的字体大小设置成高度的0.873倍就可以实现居中。

**HTML代码**



**CSS代码**

****

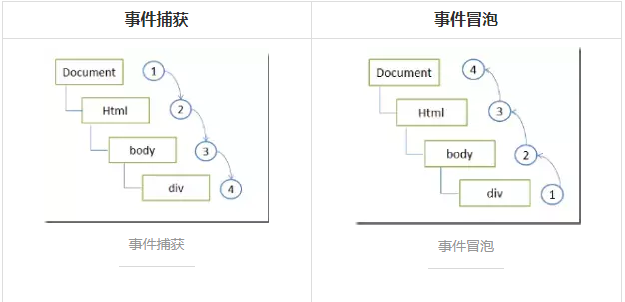
1. **事件传播的理解**

## 捕获模式(capturing)，捕获模式又称为“滴流模式”(trickling)，就是“从上向下”。

## 例如：（子元素有点击事件，当点击父元素时也可触发子元素的点击事件）。

## 冒泡模式(bubbling)，“从下向上”（默认的时间传播方式）。

## 例如：（当父元素有点击事件时，点击子元素可以触发父元素所绑定的事件）。



### 使用addEventListener(type, listener, useCapture)来注册事件的时候，第三个参数默认为false，代表事件捕获方式为冒泡。

### 使用.stopPropagation()可以阻断事件的传播，当为事件捕获时，阻断事件向下传播，当为事件冒泡时，阻断事件向上传播。

### .preventDefault()是用来阻止DOM元素默认的事件的，比如a标签的跳转动作，表单button的提交动作，与事件传播关系不大。

**应用场景：事件委托。**

1. **对象的复制**
2. **浅拷贝的方法（拷贝一层）**
3. ES6实现浅拷贝: Object.assign()
4. 2.扩展运算符（…） var b = { …a}
5. 利用for循环遍历
6. **深拷贝的方法（拷贝多层）**
7. **JSON方法实现**

JSON.stringify(); 将对象转换为json字符串形式。

JSON.parse(); 将转换而来的字符串转换为原生js对象。

注意点：当对象中有函数时，不能使用此方法。因为函数不能转换为字符串。

1. **利用递归方法**

**、、、、、、、、、、、、、、、、、、、、、、、、、、、、**

### 1.Get与Post的区别

**第一点：**

**get重点在从服务器上获取资源，post重点在向服务器发送数据；  
第二点：**  
**get请求参数放在URL中，post放在请求体中；  
第三点：**  
**Get传输的数据量小，因为受URL长度限制，但效率较高；  
Post可以传输大量数据，所以上传文件时只能用Post方式；  
第四点：**  
**get 方式只能支持ASCII字符，向服务器传的中文字符可能会乱码。  
post支持标准字符集，可以正确传递中文字符**

第五点：

**GET产生一个TCP数据包,POST产生两个TCP数据(firefox只发一个),**

**2 节点的操作**

**创建节点：**

document.createElement("标签名")

**删除节点：**

父节点.removeChild(子节点)

### 插入节点

**父节点.appendChild(新的子节点);**

**父节点.insertBefore(新的子节点,作为参考的子节点)**

**节点的获取**

**document.getElementById("box1")**

**document.getElementsByTagName("div1")**

**document.getElementsByClassName("hehe")**

### 复制节点（克隆节点）

**要复制的节点.cloneNode();**

### 获取节点的属性值

**节点.getAttribute("属性名称");**

**Git的常用命令**

创建git库

**git init  #在当前目录中生成一个.git 目录（含有.git目录的目录即是git仓库）**

2、注册git用户

**--->用于在团队合作开发中，表明代码作者。**

**git config --global user.name XXX  #用户名**

**git config --global user.email XXX   #用户邮箱**

**git config --list  #查看用户信息**

**注：加--global，全局设置。**

3、向git库添加修改

**git add [path］ #会把对应目录或文件，添加到stage状态  
　　git add .  #会把当前所有的untrack files和changed but not updated添加到stage状态**

**实际上是为修改内容添加index索引。**

4、向版本库提交修改

**git commit –m “XXXX”     #提交修改,添加注释**5、查看当前代码库的状态

**git status**

6、查看版本信息

　　--**->实际是查看修改提交信息**

**git log**

**git log --graph  #以图形化（节点）展示当前git库的提交信息。**

7、查看指定版本信息

**git show sdjf974654dd….  #(show后面为每次提交系统自动生成的一串哈希值)**

**git show sdji97 #一般只使用版本号的前几个字符即可**

8、撤销修改（又称版本回退）

**git reset  
（1）撤销整体修改  
git reset --hard  #回到原来编辑的地方,改动会丢失。（同样适用于团队对于其他人  
的修改恢复）**

9、向远端库推送修改（提交修改）

**git pull --rebase  #（将服务器项目与本地项目合并）  
　　git push    #（将本地项目上传至远端库）**

10、暂存修改

**git stash可以把当前的改动（stage和unstage，但不包括untrack的文件）暂存。**

**、、、、、、、、、、、、、、、、、、、、、、、、、、**

**面试题**

**1.同步与异步**

**同步是指在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务。**

**异步是指不进入主线程、而进入“任务队列”的任务，只有等主线程任务执行完毕，“任务队列”开始通知主线程，请求执行任务，该任务才会进入主线程执行。**

**异步的有setInterval、setTimeout定时器、Promise、Ajax请求等方法。**

**同步的有for等循环、if语句等。**

**2.jQuery链式调用的实现**

**链式调用的实现原理是jQuery在封装方法的时候最后都会return this，这里的this是调用这个方法的对象，把对象返回，所以可以在后面继续调用其他方法。**

**3.模块化开发**

**模块化是一种处理复杂系统分解为更好的可管理模块的方式。将一个复杂的项目功能分成多个模块，每个模块实现一个功能，效率高，利于维护，方便灵活，多人协作互不干扰。**

**模块化的规范有：**

**commonJS（nodeJS，同步）**

**AMD（requireJS，异步，预加载）**

**CMD（seaJS，异步，延迟加载，已经不维护了）**

**4.说说最近的项目**

**最近做的项目是no5时尚广场的仿写。这个项目是一个电商网站，类型不多，只卖护肤产品，页面简洁。**

**我负责的部分有登陆注册验证，主页面、列表页、详情页、购物车页面的数据渲染，以及一些点击跳转、删除、加入购物车等功能。**

**在前端，我使用到的技术用html、css、原生js、jQuery框架、模块化开发等。后端用到的有php，mysql等。**

**、、、、、、、、、、、、、、、、、、、、、**

**原型链的理解**

当前对象到Object.prototype的过程就是原型链。

**作用域**

作用域分为三种：

全局作用域（window.变量名），在函数外部进行定义的就是全局变量，全局变量可以在全局进行访问。

局部作用域（比如function里面），在函数里面进行定义的就是局部变量，局部变量只能在函数内被访问。

块级作用域（比如for）

变量查找规则：从当前作用域开始往前面查找直到全局作用域，找到就返回，找不到就返回not  defind

**ES6的新特性**

**解构：对赋值运算符的扩展，能够简化代码的书写；**

**箭头函数；**

**Class：在ES6才被引入，作为一个对象的模板；**

**let、const定义变量，let定义变量不会声明提前，而且只在当前代码块内有效；const用于声明一个常量，而且这个常量只可以读，不能改变；**

**promise**

**Var、let、const的异同**

**三种都是用于定义变量，但是var进行声明的变量会声明提前，let与const声明的变量不会声明提前；并且let所声明的变量只能在当前的代码块内使用；const用于声明一个常量，声明一个变量的时候就要立马进行赋值，否则将会报错。**

**、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、**

#8/19下午面试题#

1、web页面的重绘和重排

重绘是一个元素外观的改变（更改CSS样式）所触发的浏览器行为，影响比较小。例如改变visibility、outline、背景色等属性。浏览器会根据元素的新属性重新绘制，使元素呈现新的外观。

重排是改变布局（定位、标签结构改变），影响较大。

重绘不一定造成重排，但重排肯定造成重绘。

2、js中的继承

原型链继承

重点：让新实例的原型等于父类的实例。

特点：实例可继承的属性有：实例的构造函数的属性，父类构造函数属性，父类原型的属性。

　　缺点：1、新实例无法向父类构造函数传参。

　　　　　2、继承单一。

　 3、所有新实例都会共享父类实例的属性。

借用构造函数

call/apply

ES6的继承

extends

super

3、如何改变this指向

bind、call、apply

fn.bind(obj);// 把fn的this改成obj，返回一个新的函数

fn.call(obj);// 把fn的this改成obj，并执行fn，参数可以为多个

fn.apply(obj);//把fn的this改成obj，并执行fn，参数只能为1个数组

#8/20下午面试题#

1.如何阻止事件冒泡和默认事件？

答：防止冒泡和捕获：w3c的方法是e.stopPropagation()，IE则是使用e.cancelBubble = true；

取消默认事件：w3c的方法是 e.preventDefault()，IE则是使用e.returnValue = false;

2，对象深拷贝和浅拷贝定义

答：浅拷贝只是复制了对象的引用地址，两个对象指向同一个内存地址，会有共享问题，所以修改其中任意的值，另一个值都会随之变化， （例：assign()）；深拷贝是将对象及值复制过来，没有共享问题，两个对象修改其中任意的值另一个值不会改变（例：JSON.parse()和 JSON.stringify()，但是此方法无法复制函数类型）

3，二次封装的理解

答：我们在开发过程中二次封装的应用场景还是比较多，在某些封装好的工具上进行进一步的封装，能提高开发效率。

4，防抖和节流的理解和应用

答：在前端开发的过程中，我们经常会需要绑定一些持续触发的事件，如 resize、scroll、mousemove 等等，但有些时候我们并不希望在事 件持续触发的过程中那么频繁地去执行函数。为了优化性能，这里就可以用到防抖和节流。防抖（debounce：所谓防抖，就是指触发事件后 在 n 秒内函数只能执行一次，忽略后面的所有操作，如果在 n 秒内又触发了事件，则会重新计算函数执行时间。应用场景：滚动加载等等 。。。节流（throttle：就是指连续触发事件但是在 n 秒中只执行最后一次函数，忽略前面的操作，节流会稀释函数的执行频率。应用场景 ：如百度搜索等等。。。

#8/21下午面试题#

1、nodejs版本

nvm主要用来在不同的nodejs版本中切换，以便当node出新版本时，可以使用一些新的特性 。

nvm install stable 安装最新稳定版 node

nvm install <version> 安装指定版本，如：安装v4.4.0，nvm install v4.4.0

nvm uninstall <version> 删除已安装的指定版本，语法与install类似

nvm use <version> 切换使用指定的版本node

nvm ls 列出所有安装的版本

nvm alias <name> <version> 如： nvm alias default v11.1.0 给不同的版本号添加别名

nvm current 显示当前的版本

nvm unalias <name> 删除已定义的别名

2、乱码的根源

因为源编码与目标编码的不一致，造成乱码，后面统一规范用了uincode。但是缺点是字节的编码内存太大，最后用了utf-8 国际编码，基于unicode优化而来。

3、自适应布局问题

flex

（1）给父盒子设置display：flex；

flex-driection设置项目的排列方向，默认为row，设置为flex-driection: column

（2）给上下盒子设置高度

（3）设置中间盒子flex：1；

定位

父盒子设置相对定位position:relative; 上下盒子设置固定高度并绝对定位position:absolution，上面的盒子设置top：0；下面的盒子设置bottom：0；中间盒子设置为父盒子高度减去上下盒子的高度

4、 gulp的方法（常用）

gulp.task() 用来定义具体任务

gulp.src() 入口文件路径

gulp.dest() 用来写入文件

gulp.watch() 用来监视文件变化

gulp.run() 用来执行任务