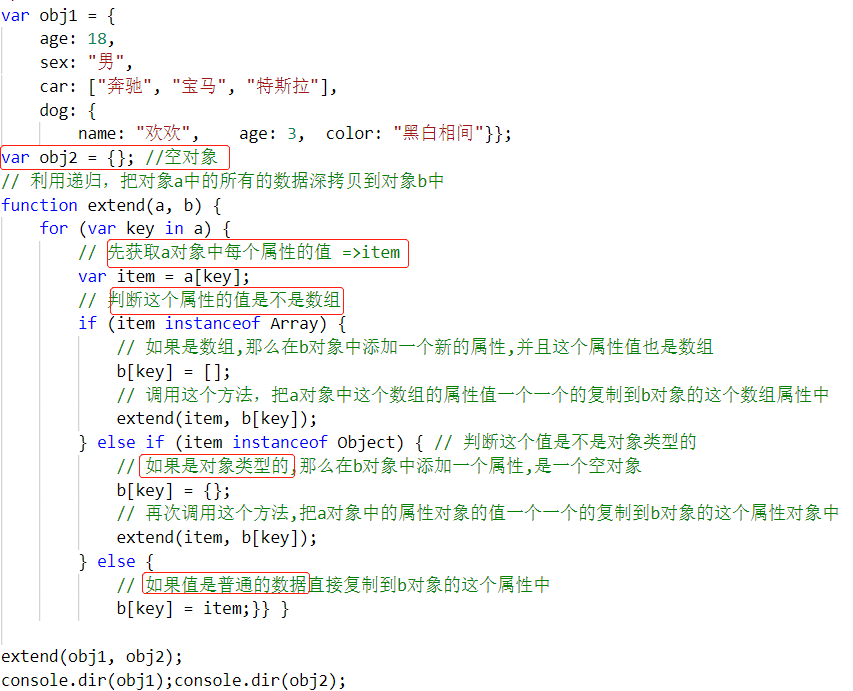
1.获取对象属性和原型属性

|  |  |  |
| --- | --- | --- |
|  | 返回该对象上 | 原型链 |
| **Object.keys** | **可枚举**的属性 | 不会获取原型链上的值 |
| **hasOwnPropery** |
| 返回一个布尔值，判断对象是否包含特定的**自身**（非继承/原型）属性 |
| **getOwnProperyNames** | **自身**的属性(可枚举和不可枚举的属性) |
| **for..in.../. []** | 所有**可**遍历**枚举**的属性,包括原型上 | |

### **for...in循环对象的所有枚举属性，然后再使用hasOwnProperty()方法来忽略继承属性**

 可枚举属性是指那些内部 “可枚举” 标志设置为 true 的属性。对于通过直接的赋值和属性初始化的属性，该标识值默认为即为 true。但是对于通过 [Object.defineProperty](https://links.jianshu.com/go?to=https://link.zhihu.com/?target=https%3A//developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty" \t "_blank) 等定义的属性，该标识值默认为 false.  
其中js中**基本包装类型**的**原型属性**是不可枚举的，如Object, Array, Number等。

可枚举的属性可以通过**[for...in](https://links.jianshu.com/go?to=https://link.zhihu.com/?target=https%3A//developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in" \t "https://www.jianshu.com/p/_blank)循环进行遍历**（除非该属性名是一个[Symbol](https://links.jianshu.com/go?to=https://link.zhihu.com/?target=https%3A//developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol" \t "https://www.jianshu.com/p/_blank)），或者通过**Object.keys()**方法返回一个可枚举属性的数组。

1. 深拷贝递归结束的条件

递归的应用通常是把一个大型的比较复杂的问题，通过层层转化为一个与原问题相似的小的问题来求解

1. [HttpRequest中常见的四种ContentType](https://www.cnblogs.com/xiaozong/p/5732332.html)

HTTP/1.1 协议规定的 HTTP 请求方法有 OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT 这几种。其中 POST 一般用来向服务端提交数据，本文主要讨论 POST 提交数据的几种方式。HTTP 协议是以 ASCII 码传输，建立在 TCP/IP 协议之上的应用层规范。规范把 HTTP 请求分为三个部分：状态行、请求头、消息主体。数据发送出去，还要服务端解析成功才有意义。一般服务端语言如 php、python 等，以及它们的 framework，都内置了自动解析常见数据格式的功能。服务端通常是根据请求头（headers）中的 Content-Type 字段来获知请求中的消息主体是用何种方式编码，再对主体进行解析。所以说到 POST 提交数据方案，包含了 Content-Type 和消息主体编码方式两部分。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 应用 | 数据被转换为 |  |
| application/x-www-form-url**encoded** | 最常见的 POST 提交数据的方式;  1)浏览器的**原生 form 表单，**不设置 **enctype** 属性，以 此 方式提交数据。  2)用 **Ajax 提交数据**时，很多时候用这种方式,Content-Type 默认值都是「application/x-www-form-urlencoded;charset=utf-8」。  提交的数据按照 key1=val1&key2=val2 的方式进行编码，**key 和 val 都进行了 URL 转码。** | | 都是浏览器原生支持的，而且现阶段原生 form 表单也只支持这两种方式。  大部分服务端语言对第一种方式有很好的支持 |
| **multipart/form-data** | 使用表单上传文件时  它会将表单的数据**处理为一条消息**，以标签为单元，用分隔符分开。**既可以上传键值对，也可以上传文件。**  当上传的字段是文件时，会有Content-Type来表名文件类型(image/jpg)；content-disposition，用来说明字段的一些信息；  由于有boundary隔离，所以multipart/form-data既可以上传文件，也可以上传键值对，它采用了键值对的方式，所以可以上传多个文件。 | |
| **application/json** | 用来告诉服务端**消息主体**是序列化后的 **JSON 字符串**。   AngularJS 中的 Aja  x 功能，默认就是提交 JSON 字符串。  由于 JSON 规范的流行，除了**低版本 IE** 之外的各大浏览器都原生支持 JSON.stringify，服务端语言也都有处理 JSON 的函数，使用 JSON 不会遇上什么麻烦。  JSON 格式支持比键值对复杂得多的结构化数据，这一点也很有用。 | | |
| **text/xml:**用于XML-RPC（XML Remote Procedure Call）,它是一种使用 HTTP 作为传输协议，XML 作为编码方式的远程调用规范。 | | | |

XML-RPC 协议简单、功能够用，各种语言的实现都有。它的使用也很广泛，如 WordPress 的 XML-RPC Api，搜索引擎的 ping 服务等等。JavaScript 中，也有现成的库支持以这种方式进行数据交互，能很好的支持已有的 XML-RPC 服务。不过，我个人觉得 XML 结构还是过于臃肿，一般场景用 JSON 会更灵活方便。

UTF-8是UTF-8编码是一种目前广泛应用于网页的编码，它其实是一种Unicode编码，即致力于把全球所有语言纳入一个统一的编码。

1. link与import差别

|  |  |  |  |
| --- | --- | --- | --- |
|  | link | | @import |
| 类型 | XHTML标签:link标签除了可以加载css外，还可以做很多其他的事情，比如定义RSS，定义rel连接属性等 | | css提供的一种方式,@import只能加载CSS。 |
| 加载顺序 | 当一个页面被加载的时候（就是被浏览者浏览的时候），link引用的CSS会同时被加载 | | @import引用的CSS会等到页面全部被下载完再加载。 |
| 兼容性 | @import是CSS2.1提出的所以老的浏览器不支持，@import只有在**IE5**以上的才能识别  而link标签无此问题，完全兼容。 | | |
| 使用dom控制样式时的差别 | | 当时用JavaScript控制dom去改变样式的时候，只能使用link标签，因为@import不是dom可以控制的（不支持）。 | |