埃森哲面试：

1. JS数据类型有哪些？

基本数据类型：Undefined,Null,Number,String,Boolean，复杂数据类型：Object

1. typeof返回值有哪些？数字1和z字符串1用==比较返回什么？

Undefined,number,string,boolean,object

1. JS的==和===的区别？

== 如果类型不同，先尝试转换类型，再做值的比较

=== 只有在相同类型下，才会比较值

1=='1'返回true

1. JS如何判断一个对象是否是数组？

方法一：使用instanceof来判断

方法二：

Function isArray(obj){

Return Object.prototype.toString.call(obj) === ‘[object Array]’

}

1. JS中改变this指向的方法有哪些？他们有什么区别？

call,apply,bind

call以参数的形式传递，apply是以数组的形式传参

call()

语法：函数.call(this, arg1,arg2,arg3)

第一个参数用来指定函数内部this的指向，后面参数是函数执行时所需的实参。

apply()

语法：函数.apply(this,[])

bind()

相同点：

都可以改变函数运行时的this

不同点：

call和apply是一次性改变this，bind是永久性改变this

1. 什么是原型？\_\_proto和prototype关系？原型链是什么？

原型1：

Javascript中，原型也是一个对象，原型可以实现对象的属性继承，javascript的对象中都包含了一个[[prototype]]内部属性，这个属性就是该对象的原型。

[[prototype]]作为对象的内部属性是不能直接访问的，通过\_\_proto\_\_来访问。

所有实例的原型引用的是函数的prototype

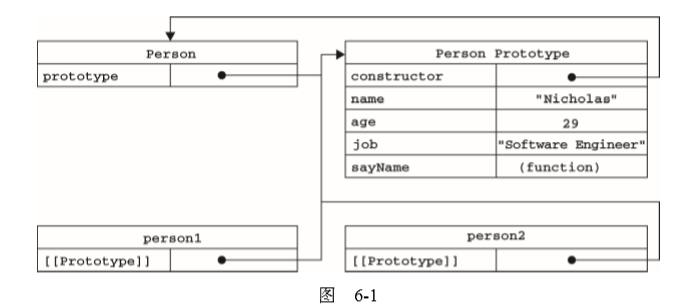
对于所有的对象，都有\_\_proto\_\_属性，这个属性对应该对象原型

对于函数对象，除了\_\_proto\_\_属性之外，还有prototype属性，当一个函数被用作构造函数来创建实例时，该函数的prototype属性值将被作为原型赋值给所有的对象实例（也就是设置实例的\_\_proto\_\_属性）

<https://blog.csdn.net/wxw_317/article/details/49617767>

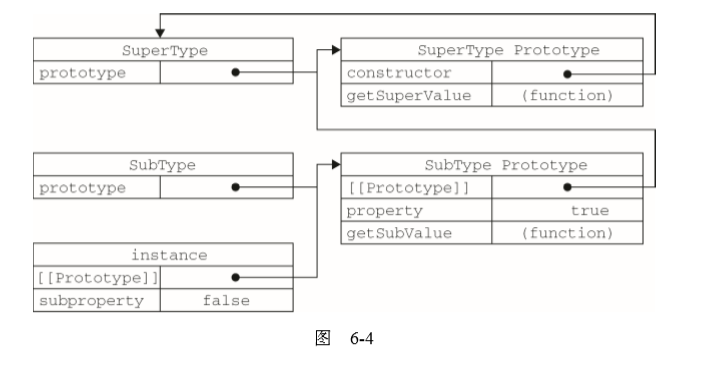
原型2：

我们创建的每一个函数都有一个prototype的属性，这个属性是一个指针指向一个对象，而这个对象的用途是包含可以由特定类型的所有实例共享的属性和方法。如果按照字面意思来理解，那么prototype就是通过调用构造函数而创建的那个对象实例的原型对象。



原型链：

JS中每个构造函数都有一个原型对象，原型对象又有一个指向构造函数的指针，而实例都包含一个直线原型的内部指针。那么假如我们让原型对象等于另一个类型的实例，结果会怎样呢？显然，此时的原型对象将包含一个指向另一个原型的指针，相应的，另一个原型中也包含一个指向另一个构造函数的指针，相应的，另一个原型中也包含一个指向另一个函数的构造函数的指针。假如另一个原型又是另一个类型的实例，那么上述关系依然成立，如此层层递进，就构成了原型链。这就是原型链的概念。



1. postion定位有哪些值？分别代表什么？

static：默认定位，元素出现在正常文档流中，忽略bottom,left,top,right等声明

relative：生成相对定位，相对于正常位置（文档流）进行定位，left:20px会向元素left添加20个像素

absolute：生成绝对定位，相对于static以外的第一个父级元素定位。元素可以通过bottom,left,top,right属性规定。

fixed：生成绝对定位，相对于浏览器窗口进行定位。元素可以通过bottom,left,top,right属性规定。

absolute和fixed会脱离文档流

1. display:none和visibility:hidden的区别？

display:none：将元素隐藏，在网页中不占任何位置

visibility:hidden：将元素隐藏，但是在网页中还占着位置

1. 设置display:none和opacity:0的div里面包裹img图片会加载吗？

都会加载

1. css如果垂直居中一个div?（不固定元素宽高）

方法一：

.center{

display: flex;

justify-content:center;

align-item:center;

}

1. 前端跨域解决方案

https://segmentfault.com/a/1190000011145364