## 2-1变量类型和计算

* **变量大的两种类型：** 值类型、引用类型（包括null、存储方式与值类型不同，存储时是以指针的方式存储）
* **变量的详细类型有几个（使用typeof判断，只能区分值类型）：**

六个 undefined number string boolean object function

* **变量的计算，强制类型转换（发生在值类型中）**

四种情景：字符串拼接、==、if语句、逻辑运算

100==’100‘ //true 0==’‘ //true null==undefined //true

var a=100 a=true a=’’; if(a) 转换a变量为布尔值

&& || ！

* **关于何时使用==和===的问题**

if（obj.a==null）可以使用==，因为此时可以理解为obj.a===null||obj.a===undefined。

剩下所有的情况，**全部使用===，**可以避免不必要的问题。

* **页面渲染过程**

onload事件触发时所有页面上的东西已经全部加载完毕。

DOMContetntLoaded事件触发时仅仅是DOM加载完毕。

* **JS自带已经封装好的函数**

Array、Boolean、Number、String、Function、Date、RegExp、Error、Math（对象）、JSON（对象）

## 2-2原型和原型链

* **构造函数 显式原型 隐式原型 继承 原型链**

首先构造一个函数Fun()，函数名称开头要大写，写完后new这个构造函数并且赋给一个新的对象a。当new构造函数时会返回到构造函数里面传参并且执行里面的程序，此时里面的this是一个空对象，执行时为this的属性赋值，之后构造函数默认会返回this，返回this后就指向了新的对象a，所以此时新的对象a就有了新的属性和方法了。此时a不光有了新的属性和方法，并且也被默认的赋予了隐式原型\_\_proto\_\_，而构造函数被默认赋予了一个显示原型prototype。这些原型也是对象，也能被赋变量和方法，也有它们的隐式原型。当查找a的一个属性时，如果在它的属性里找不到，就会自动到\_\_proto\_\_中去查找，如果还是找不到，就会去\_\_proto\_\_的\_\_proto\_\_中再找，但是不会永远循环下去，当还是找不到时，第二级的\_\_proto\_\_就为null了，js设计者把这里掐住了，通过原型向上查找的过程就形成了原型链。js里有一个特性，新对象a的隐式原型===构造函数Fun()的显式原型prototype。instanceof就是利用新特性来判断一个函数是不是一个变量的构造函数。原型链继承的原理就是让子类型的原型指向父类型的实例。即”子类型.prototype = new 父类型()”。这样新对象的隐式原型里就有了父类型的属性和方法。