JavaScript原型

函数的prototype

- 每个函数都有prototype属性，它指向一个空对象
- prototype属性有一个consturctor 对象 它指向函数本身
  - 比如：Date.prototype.consturctot===Date
- 给原型对象添加属性 就相当于所有实例对象都拥有这个属性或方法

显式原型和隐式原型

- 每个函数function都有一个prototype，即显式原型属性
- 每一个实例对象都有一个__proto__，叫隐式原型
- 对象的隐式原型的值为其对应构造函数的显示原型的值
  - var fn = new Fn()
  - fn.__proto__==Fn.prototype
- 函数的prototype属性：定义函数的时候自动添加的，默认是一个空对象
- 对象的__proto__属性，创建对象时自动添加，默认值为构造函数prototype属性值
- ES6之前程序员可以操纵显示原型，但是不可以操作隐式原型

原型链

就是说 到原型链最顶端 的是 Object 实例对象 而不是Object对象 ， 所以顶端的是 Object.__proto__

栈：fn ox123

fn.__proto__

Fn.prototype

栈：Fn 0x123

Fn.prototype

1.__proto__：不管是函数还是实例对象都有__proto__这个属性， 所有函数的__proto__都指向一个地方：Function的原型 *f* () { 【native code】 }
2.constructor：a. 除了Function的所有函数的constructor值和__proto一样都指向：Function的原型
b.Function函数的constructor指向的是它自己本身，
c.Function的原型对象的constructor指向Function函数
3.prototype：只有函数才有的属性，而实例对象没有（没有function的），原型对象都没有这个属性
4.Object原型对象（Object）：既不是Object也不是Function，负责存储特定的方法
5.Function原型对象（*f* () { 【native code】 }）：负责函数的创建，
所以所有的函数的__proto__r都指向这里，
除了Function函数的所有函数的constructo都指向这里，
因为Function可以自己创建自己：new Function()
6.Function：最特殊的是，__proto__和prototype相等，都指向Function原型对象

Function = new Function（）

Function这个函数比较特殊 proto 和prototype相同 都是Function的原型对象

constructor 只有 原型对象有 指向的是原来的函数

Object的原型对象 不属于Object 也 不属于 Function