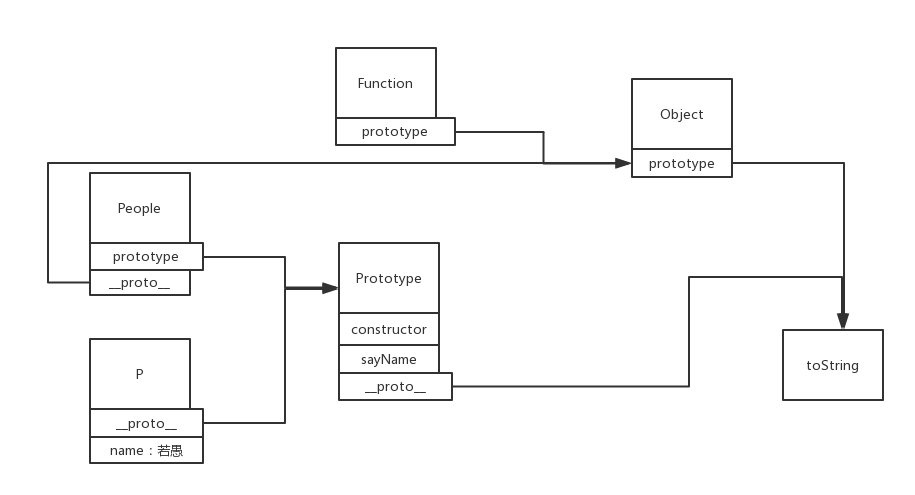
### ****有如下代码，解释****Person****、****prototype****、****\_\_proto\_\_****、****p****、****constructor****之间的关联。****

1. function Person(name){
2. this.name = name;
3. }
4. Person.prototype.sayName = function(){
5. console.log('My name is :' + this.name);
6. }
7. var p = new Person("若愚")
8. p.sayName();

Person 是一个构造函数，通过 new 生成实例 p。每个函数都有 prototype 属性，生成的实例有 \_\_proto\_\_属性，他们都指向 Person函数的原型对象 prototype ，原型对象都带有constructor 属性，它指向 Person 函数

### ****上例中，对对象 p可以这样调用****p.toString()****。****toString****是哪里来的? 画出原型图?并解释什么是原型链。****



    图上所示，p 自身并没有 toString() 这个方法，p 是构造函数 Person的实例，其 \_\_proto\_\_ 指向原型对象 Person 的Prototype，原型对象的 \_\_proto\_\_ 指向 Object 函数的的原型对象 toString，所以构成的链路就是原型链。

### ****对****String****做扩展，实现如下方式获取字符串中频率最高的字符****

1. *//todo....*
3. var str = 'ahbbccdeddddfg';
4. var ch = str.getMostOften();
5. console.log(ch); *//d , 因为d 出现了5次*
6. String.prototype.getMostOften = function(n){
7. var str = this,
8. obj = {},
9. key,
10. max = 0;
11. for(var i= 0;i<str.length;i++){
12. if(str[i] in obj){
13. obj[str[i]] +=1;
14. }else{
15. obj[str[i]] = 1;
16. }
17. }
18. for(n in obj){
19. if(obj[n]>max){
20. max = obj[n];
21. key = n;
22. console.log(key,max);
23. }
24. }
25. return (max,key)
26. }
27. var str = 'ahbbccdeddddfg';
28. var ch = str.getMostOften();
29. console.log(ch)

### instanceOf****有什么作用？内部逻辑是如何实现的？****

     判断一个变量是否某个构造函数创造的实例

1. var a = [1,2,3]
2. a instanceOf Array   *// true*
3. a instanceOf Object  *// true*

    实质是构造函数的原型对象(Function.prototype)是否在左边对象的原型链(\_\_proto\_\_)上

1. function isObjInstanceOffunc(obj,func) {
2. var \_\_proto\_\_=obj.\_\_proto\_\_;
3. do{
4. if(\_\_proto\_\_===func.prototype) return true;
5. }while(\_\_proto\_\_=\_\_proto\_\_.proto\_\_)
6. return false;
7. }