### js复习

js是运行在浏览器端的，是面向对象的脚本语言。

|  |  |
| --- | --- |
| **js是一门编程语言，编程的三要素** | |
| 数据 | 变量 → 数据类型 → 运算符 |
| 流程控制 | 流程分支if（单分支、双分支、多分支）/switch、循环for/while  continue跳出当前循环，break 结束整个循环 |
| 函数 | 声明、调用 |

面向对象：构造函数（类）、对象（实例的属性和方法）

面向对象的特性：抽象性（①只抽出需要的核心信息 / ②数学对象就是抽象出来的）、封装性、继承性、多态性。

多态：当子类继承父类后，每个子类可以单独修改继承下来的方法，然后实现各自不同的功能。

js没有多态，js是原型链继承，所有子类（子对象）继承的方法只有一份，保存在原型对象中，子对象无法对这些方法进行修改。

名词提炼法找对象

使用面向对象进行编程：使用现有对象，自定义对象

复合数据类型有：对象（正则表达式、Array Function Date Object）

var arr = [1,2,3];

var fn = function a (){};

var date = new Date();

var o = new Object();

console.log(typeof undefined); //undefined

console.log(typeof null); //object null是一个空指针对象

console.log(typeof arr); //object

console.log(typeof fn); //function

console.log(typeof date); //object

console.log(typeof o); //object

除了函数，其他的数据类型无法使用typeof获得数据类型，要知道对象的数据类型，用**O**bject.prototype.toString.apply(arr); 结果是[object Array]

console.log(Object.prototype.toString.apply(null)); 结果是[object Null]

js中对象的动态特性，即想要什么属性就可以提供什么属性，在js中，对象如果没有指定的属性，只要利用赋值就可以给对象提供该属性。

|  |  |
| --- | --- |
| o.name = 'jim'; // 点语法赋值 | console.log( o.name ); // 点语法取值 |
| o[ **'**name**'** ] = 'tom'; // 关联数组语法赋值 | console.log( o[ **'**name**'** ] ); // 关联数组语法取值 |

Q：o[ name ] = 'jack'; 对不对？

A：对，如果name 是一个变量, 里面存储的是字符串, 也是可以的。而window里正好有一个name属性，所以不会报错，换成别的就不行。相当于此时的键是个变量，你还可以给它赋值name=0，会打印出0：'jack'

Q：var num = “name”; [].push( { num : “Lily”} ); 会出现什么结果？

A：[{num : “Lily”}]，在对象中键名不能是变量，写什么就是什么。如果想把“name”放进去，可以用

var obj = {};

obj[num] = "Lily";

arr.push(obj);

### 构造函数（constructor）

1. 函数声明式：

function fn (a,b) {

代码段；

}

1. 函数表达式（匿名函数、字面量函数、lambda函数）：

var fn = function(a,b) {

代码段 }**;**

1. var fn = new Function();

函数就是韩国美容院，不管什么样的人进去，都按照定好的流程给改造一番，究竟具体的流程“鬼知道要经历什么。”

创建对象的方法一：

var obj = new Object(); //var obj = { }; 也可以

obj.name = Lily;

obj.act = function(){ content };

创建对象的方法二：

var laptop = {

color: “silver”,

price: 5000,

memory: 4G

}

批量创建对象：

1. 工厂函数：

function Student (name,age,score) { //声明一个函数，该函数的作用是创建一个对象

var stu = new Object(); 或var stu = { };

stu.name = name;

stu.age = age;

stu.score = score;  
 stu.sayHi = function () {  
 console.log("hello,I am "+ stu.name);  
 }; //创建一个对象，说明该对象的属性和方法

return stu; //返回这个对象

}  
 var stu1 = Student("Lily",15,400); //直接调用函数

stu1.sayHi();

工厂函数无法将对象和创建它的模板函数联系起来。

所以console.log(stu1 instanceof Student); 打印的结果是false

1. 构造函数模式：

function Student (name, age, score) {  
 this.name = name; // 不需要new Object;

this.age = age;

this.score = score;

this.sayHi = function () {  
 console.log("hello");  
 } // 不需要return;  
} //调用constructor时用new引导，相当于var stu = { };  
var stu1 = new Student ("Lily",16,100); //Student相当于类名

stu1.sayHi();

构造函数内部的this把构造函数创建出来的对象的真实类型指向了构造函数名称。new会改变构造函数中的this的指向，并让构造函数把它返回。

1. 原型模式：

function Student ( name, age, score) {  
 this.name = name;

this.age = age;

this.score = score;  
}  
Student.prototype.sayHi = function () {  
 console.log("hello");  
}  
var stu1 = new Student ("Lily",16,100);

stu1.sayHi();

### prototype是构造函数的原型属性，凡是绑定到构造函数原型属性上的属性或方法，每一个通过构造函数创建的实例对象都可以使用。

\_\_proto\_\_是实例对象的原型对象，实例对象的原型对象指向该对象的构造函数的原型属性。就像女人通过“丈夫”找到的人和孩子通过“爸爸”找到的人是同一个人一样。

### this

this关键字指向的是函数的调用者，即调用该方法的对象。

全局上下文中（任何函数体的外部），this指向全局对象window

函数上下文指调用该函数的对象，普通函数实际是被全局对象window调用了。

### 继承性

在js中，属性定义在实例对象中，方法定义在原型对象中，调用方法的时候，实例对象本身没有该成员，但是我们依然可以调用该方法，就像该实例对象本来就有这个方法一样，因此我们称该实例对象继承自原型对象。任何一个实例都继承自其原型对象，即**原型式继承。**这样可以提高复用性，共享性。

属性访问原则：

1. 对象在调用方法或属性的时候，首先在当前对象中查找，如果有就使用并停止查找；
2. 如果没有，就在其原型对象中查找，如果有即使用并停止查找；
3. 如果还没有找到，就到该对象的原型对象的原型对象中查找；
4. 最后一直找到Object.prototype上，如果还没有就返回null.

##### 原型式继承

写一个构造函数，如果想要其实例对象继承自某个特定的对象o，那么只需要设置该构造函数的prototype属性为o即可。

##### 混合式继承

混合就是将多个对象的多个功能混合到一起，加到构造函数的原型属性上，那么由该构造函数new出来的实例就继承自多个对象了。

typeof 用来判断基本数据类型；

instanceof用来判断构造函数的原型属性是否在对象的原型链上。

instanceof用来判断复杂数据类型（对象）的类型 console.log([ ] instanceof Array);

null instanceof Object false

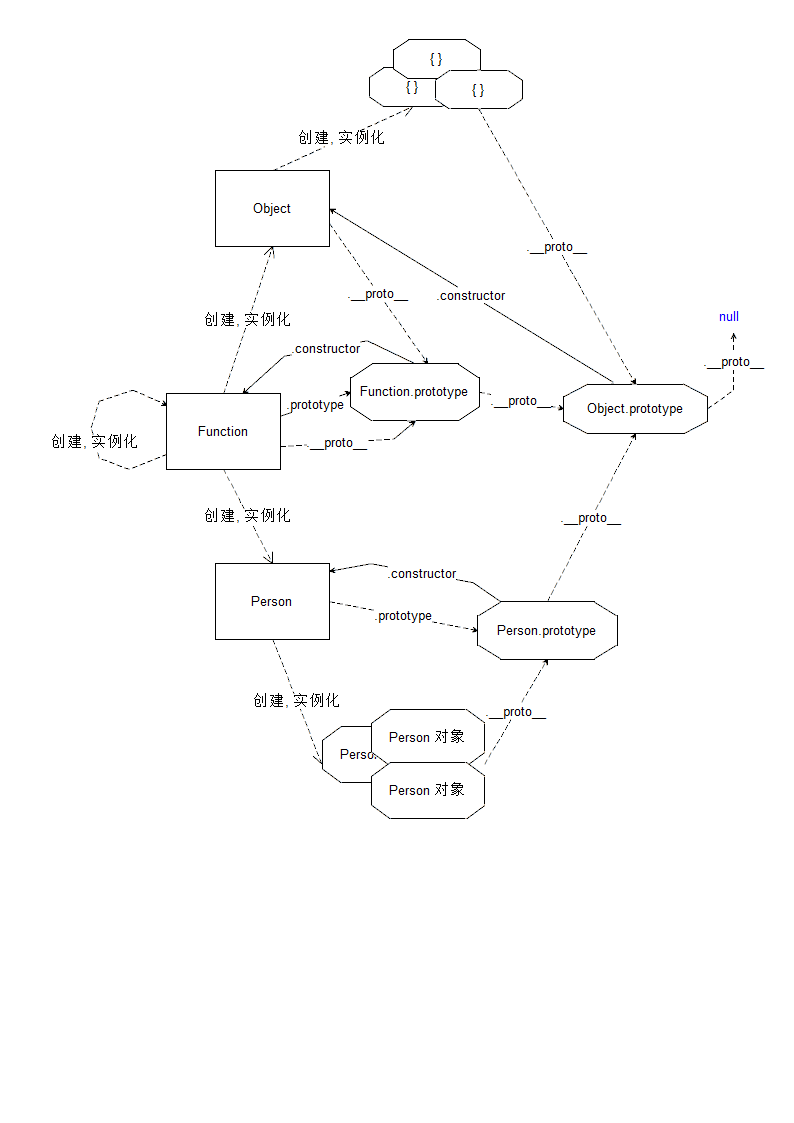
Function instanceof Object true

Array instanceof Object true

undefined instanceof Object false

Function instanceof Object true

Object.prototype instanceof Object false



万物皆对象（广义上的对象），但并不是每一个对象都是Object构造函数的实例，原型对象（属性）也是对象，**所有的.prototype都是Object的实例**，通过\_\_proto\_\_指向Object.prototype。

如果不是window而是别的对象想调用某方法/函数，就要先看看这个对象有没有这个方法/函数。

js在预解析的过程中完成了①声明部分的标记与②变量作用域的设定。声明又包括：变量的声明和函数的声明。

自调用函数是被window调用的，其中的this指的是window。

|  |  |  |
| --- | --- | --- |
|  | caller | callee |
| 语法 | 函数名.caller | arguments.callee |
| 例子 | function fn (){  console.log ( fn.caller); }  fn();  //此时caller是null，不是window | function fn (){  console.log(arguments.callee===fn);  } |
| 用途 | 表示调用当前函数的函数 | 在递归时表示函数的引用 |

json格式有两种结构：**{ },** **[ ]**，键名必须用双引号引起来。

Person.say = function(){

console.log("hello");

} //这种叫静态方法，只能由Person调用。

|  |  |  |
| --- | --- | --- |
| 静态成员 | 静态方法 | 直接绑定在（构造）函数上的方法 |
| 静态属性 | 直接绑定在（构造）函数上的属性 |

function Person (){

this.say = "hi";

} //这种构造函数内部定义的方法或是在Person.prototype上绑定的方法叫动态方法（或实例方法），因为new出来的每个实例对象都有该方法。