高级Day 01---面向对象入门

一、面向对象的概念

1.1 什么是面向过程

就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

生活中的的例子举例。

1.2 什么是面向过程

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

1. 面向对象是一种思维方法
2. 面向对象是一种编程方法
3. 面向对象并不只针对某一种编程语言

1.3 面向对象和面向过程的区别和联系

1. 面向过程过程侧重整个问题的解决步骤，着眼局部或者具体
2. 面向对象侧重具体的功能，让某个对象具有这样的功能。更加侧重于整体。

各自的优缺点

面向过程的优点：
流程化使得编程任务明确，在开发之前基本考虑了实现方式和最终结果；
效率高,面向过程强调代码的胆小精悍，善于结合数据结构来开发高效率的程序。。
流程明确，具体步骤清楚，便于节点分析。

缺点是：需要深入的思考，耗费精力，代码重用性低，扩展能力差，维护起来难度比较高，
对复杂业务来说，面向对象的模块话难度较高，耦合度也比较高。

面向对象的优点：结构清晰，程序便于模块化，结构化，抽象化，更加符合人类的思维方式；
封装性，将事务高度抽象，从而便于流程中的行为分析，也便于操作和自省；
容易扩展，代码重用率高，可继承，可覆盖；
实现简单，可有效地减少程序的维护工作量，软件开发效率高。

缺点是：效率低，面向对象在面向过程的基础上高度抽象，从而和代码底层的直接交互非常少机会，
从而不适合底层开发和游戏甚至多媒体开发；
复杂性，对于事务开发而言，事务本身是面向过程的，过度的封装导致事务本身的复杂性提高。

二、编程语言中面向对象的实现方式

编程语言对对面向对象的实现主流的有两种方式：基于类的面向对象和基于原型的面向对象。

不管以什么方式实现，都具有面向对象的三大特征：

- 封装

  也就是把客观事物封装成抽象的类或具体的对象，并且类或对象可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

- 继承

  可以让某个类型的对象获得另一个类型的对象的属性的方

- 多态

  不同实例的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。

2.1基于类的面向对象

> 典型的语言：Java、C#

对象（object）依靠 类（class)来产生

2.2 基于原型的面向对象

> 典型的语言：JavaScript

对象（object）则是依靠 构造器（constructor）利用 原型（prototype）构造出来的

三、对JavaScript对象的进一步认识

ECMA-262 把对象定义为：“无序属性的集合，其属性可以包含基本值、对象或者函数。”严格来讲，这就相当于说对象是一组没有特定顺序的值。对象的每个属性或方法都有一个名字，而每个名字都映射到一个值。正因为这样（以及其他将要讨论的原因），我们可以把 ECMAScript 的对象想象成散列表：无非就是一组名值对，其中值可以是数据或函数。

```
<script type="text/javascript">
    //用大括号括起来的一系列的键值对，就构成了JavaScript对象。这种对象称之为字面量对象。
    var person = {
        name : "张三",     // 一个键值对
        age : 20,
        sex : "男",
        eat : function () { //属性的值是函数，这个时候我们更喜欢把这样的属性称之为方法。
            alert("吃东西");
        }
    }
</script>
```

> 说明：
>
> 1. name : "张三" 一个键值对表示JavaScript对象的一个属性。 name是属性名， "张三" 属性值。
> 2. 属性可以是任意类型的。可以包括我们以前学的简单数据类型，也可以是函数，也可以是其他的对象。
> 3. 当一个属性的值是函数的时候，我们更喜欢说这个属性为方法。(如果函数不和对象关联起来的时候，应该叫函数不应该叫方法。只是一种称呼，你完全可以不用理会)。 我们一般说person对象具有了一个方法eat. 将来访问eat的时候，也和调用一个函数一样一样的。

3.1 访问对象的属性

> 访问一个对象的属性，我们可以直接通过 对象.属性名 和 对象[属性名] 来访问。

```
alert(person.name);  // 访问person对象的 name属性值
person.age = 30;  //修改person对象的 age 属性
person.eat();  //既然是调用方法(函数) 则一定还要添加 ()来表示方法的调用
alert(person["name"]);  //
```

> 两种使用方式有一些不同的地方：
>
> 1. 对象.属性名的方式，只适合知道了属性的名字，可以直接写。比如： person.age 。如果属性名是个变量，则这种方法无效， 对象.变量名 会出现语法错误。
> 2. 对象[属性名]，这种方式使用无限制。如果是字符串常量，则应该用""引起来，如果是变量，可以直接使用。

```
person.age = 100; // ok
var n = "age";
person.a = 101; //  no ok 语法错误
person["age"] = 102; // ok
person[n] = 103;  //ok
```

3.2 给对象添加属性

> JavaScript是一种动态语言，可以在代码执行过程中，动态去添加和修改对象的属性。这是与其他面向对象语言一个很大的不同点。
>
> 备注：对那些基于类的语言，属性一旦在类中定义完成，对象是不能去动态添加和删除属性的。

```
//给person对象的属性 girlFriend 赋值。在赋值的过程中，首先会判断这个属性在JavaScript中是否存在，如果存在就对这个
//属性重写赋值。如果不存在，就给这个对象添加这个属性，并赋值。
person.girlFrient = "小丽";

//给对象添加方法
person.play = funcion(){
    alert("打击high起来");
}
```

3.3 删除对象属性

> 对JavaScript来说，我们不仅可以动态的添加属性，也可以动态的删除属性。
>
> 使用操作符：delete
>
> 注意：delete是个操作符，不是方法，所以后面没有必要添加括号啊

```
// 使用delete操作关键字，删除person对象的属性age
delete person.age;
alert(person.age);  //弹出undefined。表示这个属性没有定义
```

3.4 修改对象属性

```
// 把person对象的sex属性的值修改为 女
person.sex = "女";
person.eat = funcion(){
    alert("吃货");
}
person.eat();  //吃货
```

3.5 使用for...in遍历对象的属性

> for...in可以用来遍历对象的所有属性。

```
// 在用for...in遍历的时候， in前面的变量pn指的是属性的名称。
for (pn in person) {
    alert(pn + " " + person[pn]);
}
```

四、多种创建对象的方式

> 除了上面的使用对象直接量，JavaScript还支持多种方式创建对象

4.1 使用new Object()创建

```
<script type="text/javascript">
    //使用object创建一个对象     完全等同于 var person = {};
    var person = new Object();
    //给对象添加属性
    person.name = "李四";
    //给对象添加方法
    person.eat = function () {
        alert("好好吃")
    }
</script>
```

4.2 工厂模式创建

> 虽然 Object 构造函数或对象字面量都可以用来创建单个对象，但这些方式有个明显的缺点：使用同一个接口创建很多对象，会产生大量的重复代码。为解决这个问题，人们开始使用工厂模式的一种变体。

> 工厂模式是软件工程领域一种广为人知的设计模式，这种模式抽象了创建具体对象的过程，考虑到在 ECMAScript 中无法创建类，开发人员就发明了一种函数，用函数来封装以特定接口创建对象的细节。

```
<script type="text/javascript">
    function createPerson(name, age, job) {
        var o = new Object();
        o.name = name;
        o.age = age;
        o.job = job;
        o.sayName = function() {
            alert(this.name);
        };
        return o;
    }
    var person1 = createPerson("张三", 29, "js开发者");
    var person2 = createPerson("李四", 27, "java开发者");
</script>
```

> createPerson()函数可以多次调用，没调用一次这个函数就会返回一个对象，而且对象的类型仍然是Object类型的。虽然解决了多个相似对象的问题，但却没有解决对象类型识别的问题。

4.3 构造函数模式创建

> 为了解决对象类型识别问题，又提出了构造函数模式。这种模式，其实在我们创建一些原生对象的时候，比如Array、Object都是调用的他们的构造函数。
>
> 看下面的代码

```
<script type="text/javascript">
    function Person (name, age, sex) {
        this.name = name;
        this.age = age;
        this.sex = sex;
        this.eat = function () {
            alert(this.name + "在吃东西");
        }
    }
    var p1 = new Person("张三", 20, "男");
    p1.eat();   //张三在在吃东西
        var p1 = new Person("李四", 30, "男");
    p1.eat();   //李四在在吃东西
    alert(p1 instanceof Person);    //
</script>
```

> 说明：

1. 使用构造函数创建对象，必须使用关键字new ，后面跟着构造函数的名，根据需要传入相应的参数。
2. 其实使用new 构造函数()的方式创建对象，经历了下面几个步骤。
    - 创建出来一个新的对象
    - 讲构造函数的作用域赋给新对象。意味着这个时候 this就代表了这个新对象。
    - 执行构造函数中的代码。 在本例中就是给新对象添加属性，并给属性初始化值。
    - 构造函数执行完毕之后，默认返回新对象。 所以外面就可以拿到这个刚刚创建的新对象了。

五、构造函数与普通函数的关系

1. 他们都是函数。构造函数也是函数，也可以像普通的函数一样进行调用。 做普通函数调用的时候，因为没有创建新的对象，所以this其实指向了window对象。

```
function Person(){
    this.name = "张三";    // 把name属性添加到了window对象上面
    alert(this === window);  //如果不作为构造方法调用，则 是true
}
Person();  // 把构造函数当做普通方法调用。这个时候内部的this指向了weindow
alert(window.name);  //张三
function Human(){
    this.name = "王五";
    alert(this instanceof window);  // false
    alert(this instanceof Human);  //true
}
var h = new Human();  //当做构造函数来调用，创建一个对象
alert(h.name);
```

2. 构造函数和普通函数仅仅也仅仅是调用方式的不同。也就是说，随便一个函数你如果用new 的方式去使用，那么他就是一个构造函数。
3. 为了区别，如果一个函数想作为构造函数，作为国际惯例，最好把这个函数的首字母大写。