面向对象的理解

**目录：**

1. 抽象——流水线相同特征的提取

2. 封装——流水线作业的搭建

3. 继承——流水线的可扩展

4. 多态——流水线的多功能

5. 面向对象的3大特征

6. 类 和 对象

类：流水线

对象：流水线上面的产品

7. 创建面向对象的方法

es5——fn

es6 ——class

8. new的过程的理解

new fn 或 new fn() 或 new fn(参)

在函数内部参数数据arguments和新建一个对象，并且this指向这个对象

函数执行完返回这个新建的对象

注意：

函数本身的return

9. 原型和原型链的理解

构造函数.prototype=原型

new 构造函数=实例

实例.\_proto\_=构造函数.prototype

原型.contuctor=构造函数

10. 属性和方法

构造函数下的属性和方法是静态的

实例下的属性和方法是动态的

**1. 抽象——流水线相同特征的提取**

如果一个东西需要量产（复用），我们需要分析这个东西特点，根据这个特点来创建这个模具，后期我们会通过这个模具来批量的进行生产，或者根据要生产的内容创建流水线，需要对这些要批量创建的内容进行共同特征的提取，我们把这个提取事物共同特征的过程称为：抽象

抽象即抽取相同的部分。

特点：

相同的特征

不具体的

**2. 封装——流水线作业的搭建**

根据抽象进行流水线作业的搭建，我们把这个过程称为：封装

特点：对内进行封装，隐藏细节，对外提供接口

**3. 继承——流水线的可扩展**

一个流水线可以扩展出另外一个流水线，不用重新创建流水线。

比如：根据小车的生产线扩展出SUV的生产线，这个时候没有必要重头到尾的去创建一个新的生产线，只需要把原来的小车生产线修改一下就可以变成新的SUV的生成线了，这个过程我们称为：继承

**4. 多态——流水线的多功能**

一种类型的事物，如果拥有多个不同的子类型，那么针对这个类型来说的话，我们可以说同一个主类型具有不同的功能

比如：同样是车，车又有小车，还有卡车，小车可以装人，卡车可以装货，我们可以说车这中类型具有不同的状态，那么我们把这种现象称为：多态。

特点：多态的特性可以通过 继承 来实现

5. 面向对象的3大特征

封装

继承

多态

**6. 类 和 对象**

我们把抽象后的**生产线**称为：类，类型；

我们把通过生成线生成出来的**产品**称为：对象；

面向对象编程其实就是自定义数据类型；

类和对象的关系：

**类是对一类具有相同特征的事物的抽象化描述；**

**对象是类的实例化（具体化）。**

**7. 创建面向对象的方法**

方法一：

function Student(name){

this.name=name;

}

Student.prototype.study=function(){}

Student.prototype.a='aaa';

或：

Student.prototype={

constructor:Student,

study:function(){},

show:function(){}

}

var

方法二：

es6的创建方法，class类的方法

class Student(){

stactic a='aaa'

constructor(){

this.name=name;

}

study()

}

Student.prototype.a='aaa';

var s=new Student();

约定：大驼峰命名，首字母也要大写

本质：

用new运算符创建一个student这个类相关的对象，new后面是构造函数，是Student这个类下面的construct方法，construct是这个类的构造函数。当new之后，就会直接调用执行construct这个函数。

construct构造函数的作用：初始化对象。

new之后，创建空对象，系统默认将this指向这个创建的新对象，并且将this（产生的新对象）返回出去。

Student类这个下面有属性和方法。

**8. new的过程的理解**

运算符，new的后面只能跟函数；使用new也可以调用函数，不需要加括号调用，如果我们希望在调用函数的时候传入参数的时候，可以加括号；

当我们调用函数的时候，函数在执行的时候会创建一些的内置的数据，比如arguments；当我们使用new的方式来调用的时候，除了上面的一些对象以外，函数还会在内部创建一个空对象，同时还会把这个函数的this指向该对象；

在函数执行完成以后，函数还会把这个自动的创建的对象作为函数的返回值进行返回；

当我们通过new来调用一个函数的时候，其实就是创建对象；

**注意：**

a. 函数里面没有return，返回创建的对象；

当我们使用new的时候，js会保证这个函数的返回一定是一个对象，如果该函数有明确的return，那么会出现下面两种情况：

b. 如果return的是非对象的值，那么返回默认创建的对象；

c. 如果return的是一个对象，那么return的对象就是返回的对象；

**9. 原型和原型链的理解**

把对象私有的数据添加到对象自身上

为了能够让同一个类型的对象所拥有的公有值方便存储和查找，不影响其他的对象，所以js为我们提供了一个特殊位置来存放某个类所拥有的公有特性。因为函数也是一个对象，所以js给这个函数自动创建一个属性，prototype，原型，我们可以把与这个函数（构造函数）有关的公有数据都存放在函数的prototype下。

每一个函数会自动创建prototype，同时还会在prototype添加默认的属性constructor，值指向构造函数，这样的话，我们就可以通过对象来调用这个属性，获取该对象的构造函数了。

**特征：**

当一个函数被创建的时候，函数下会自动添加一些属性，其中有一个属性：prototype，他的值是一个对象；

当一个对象被创建的时候，对象也会有一些默认添加的属性，其中有一个属性：\_\_proto\_\_，他的值也是一个对象；

对象的\_\_proto\_\_其实就是创建该对象的构造函数下的prototype；

**原型链：**类似作用域链，对象属性或方法的查找或调用也有自己的规则，当我们去调用一个对象下的属性或方法的时候，首先会该对象自身上查找调用，如果没有，则会在该对象的\_\_proto\_\_下去查找调用，直到找到object为止，如果object还没有找到，就是null
**属性和方法:**

动态属性/动态方法：通过对象去调用的，又称为对象属性，对象方法

静态属性/静态方法：通过构造函数调用的，又称为类属性，类方法