面向对象的几个概念：

* 一切事物皆对象
* 对象具有封装和继承特性
* 对象与对象之间使用消息通信，各自存在信息隐藏

根据以上三点，C++是半面向对象半面向过程语言（虽然实现了类的封装、继承和多态，但存在非对象性质的全局函数和变量），而像Java、C#是完全的面向对象的语言。

JavaScript语言是通过一种叫做原型（prototype）的方式来实现面向对象编程的。

基于**类的面向对象**和基于原**型的面向对象**两种方式差别：

在基于类的面向对象方式中，**对象（object）**依靠 **类（class）**来产生。而在基于原型的面向对象方式中，对象（object）则是依靠**构造器（constructor）**利用**原型（prototype）**构造出来的。例如：工厂造一辆车，一方面，工人必须参照一张工程图纸，设计规定这辆车应该如何制造。这里的工程图纸就好比是语言中的类(class)，而车就是按照这个类（class）制造出来的；另一方面，工人和机器(相当于constructor)利用各种零部件如发动机，轮胎，方向盘 (相当于prototype的各个属性)将汽车构造出来。

事实上关于这两种方式谁更为彻底地表达了面向对象的思想，目前尚有争论。但笔者更偏向于原型式面向对象更彻底表达了面向对象的思想！

首先，客观世界中的对象的产生都是其它实物对象构造的结果，而抽象的“图纸”是不能产生“汽车”的，也就是说，类是一个抽象概念而并非实体，而对象的产生是一个实体的产生；

其次，按照一切事物皆对象这个最基本的面向对象的法则来看，类 (class) 本身并不是一个对象，然而原型方式中的构造器 (constructor) 和原型 (prototype) 本身也是其他对象通过原型方式构造出来的对象。

再次，在类式面向对象语言中，对象的状态 (state) 由对象实例 (instance) 所持有，对象的行为方法 (method) 则由声明该对象的类所持有，并且只有对象的结构和方法能够被继承；而在原型式面向对象语言中，对象的行为、状态都属于对象本身，并且能够一起被继承，这也更贴近客观实际。

最后，类式面向对象语言比如 Java，为了弥补无法使用面向过程语言中全局函数和变量的不便，允许在类中声明静态 (static) 属性和静态方法。而实际上，客观世界不存在所谓静态概念，因为一切事物皆对象！而在原型式面向对象语言中，除内建对象 (build-in object) 外，不允许全局对象、方法或者属性的存在，也没有静态概念。所有语言元素 (primitive) 必须依赖对象存在。但由于函数式语言的特点，语言元素所依赖的对象是随着运行时 (runtime) 上下文 (context) 变化而变化的，具体体现在 this 指针的变化。正是这种特点更贴近 “万物皆有所属，宇宙乃万物生存之根本”的自然观点。

最基本的面向对象

ECMAScript是一门彻底的面向对象的编程语言（参考资源），JavaScript 是其中的一个变种 (variant)。它提供了 6 种基本数据类型，即 Boolean、Number、String、Null、Undefined、Object。为了实现面向对象，ECMAScript设计出了一种非常成功的数据结构 - JSON(JavaScript Object Notation), 这一经典结构已经可以脱离语言而成为一种广泛应用的数据交互格式。

具有基本数据类型和 JSON 构造语法的 ECMAScript 已经基本可以实现面向对象的编程了。开发者可以随意地用字面式声明（literal notation）方式来构造一个对象，并对其不存在的属性直接赋值，或者用**delete**将**属性删除** ( 注：JS 中的 delete 关键字用于删除对象属性，经常被误作为 **C++** 中的 delete，而后者是用于**释放**不再使用的**对象** )

使用函数构造器构造对象

除了字面式声明（literal notation）方式之外，ECMAScript允许通过**构造器**（constructor）创建对象。每个构造器实际上是一个函数（function）对象, 该函数对象含有一个“prototype”属性用于实现基于原型的继承（prototype-based inheritance）和共享属性（shared properties）。对象可以由“new 关键字 + 构造器调用”的方式来创建。

彻底理解原型链 (prototype chain)

在 ECMAScript 中，每个由构造器创建的对象拥有一个指向构造器 prototype 属性值的 隐式引用（implicit reference），这个引用称之为 原型（prototype）。进一步，每个原型可以拥有指向自己原型的 隐式引用（即该原型的原型），如此下去，这就是所谓的 原型链（prototype chain） （参考资源）。在具体的语言实现中，每个对象都有一个 \_\_proto\_\_ 属性来实现对原型的 隐式引用。

有了**原型链**，便可以定义一种所谓的**属性隐藏机制**，并通过这种机制实现继承。

JavaScript 私有成员实现

JavaScript 必须依赖闭包实现信息隐藏，是由其函数式语言特性所决定的。

