对象探析

> ! 本文简单介绍JavaScript语言中的Object对象类型，包括但不限于对象的结构、核心概念、类型检查、创建对象的方式以及对象的常用操作等内容。

1.1 对象简单介绍

javaScript是一门基于 弱类型、支持面向对象编程且基于原型继承的脚本语言 。

对象说明

在JavaScript语言中，对象是可变的键值对集合（或者称为属性的容器），通过对象我们可以方便的管理一组变量和函数。

用通俗的话来说，对象其实就是一堆变量和函数的集合。只是“定义”在对象中的变量，我们称为属性，“定义”在对象中的函数，我们称之为方法。

数据类型和对象

JavaScript中的数据类型可以分为 基本数据类型和复杂数据类型 。
其中基本数据类型有：数字(number)、字符串(string)、布尔值(boolean)、undefined值和null。

复杂数据类型可以简单理解为对象类型。在JavaScript中，数组是对象、函数是对象、正则表达式是对象，对象自然也是对象，它们在使用typeof 运算符时得到的结果为object，即对象类型。

> ! typeof对函数运算的结果为function，对null运算的结果为object，但这被认为是一个错误。

JavaScript语言中的对象是无类型的，对象可以拥有属性和方法，属性和方法都是以key-value的方式存储的。我们可以笼统的把方法和属性统一归类为属性（因为方法名其实和属性名没有任何本质的区别，属性和方法的分类方式只是对它们存储的内容进行人为区分），所以 对象其实就是键值对的集合 。

对象通过引用来传递，它们永远不会被复制。

对象的检查

① 可以使用typeof来对对象类型进行简单的检查，但需要注意排除null的情况。

② 在开发中，经常会用到hasOwnProperty方法来过滤对象原型成员。

1.2 javaScript中对象的获取方式

JavaScript中对象的创建有多种方式，根据特定的应用场景，我们可以选择不同的更合适的方式来创建对象，简单可以归纳为以下情况：

> ① 字面量的方式创建对象
> ② 内置构造函数创建对象
> ③ 封装工厂函数创建对象
> ④ 定义构造函数创建对象
> ⑤ 调用系统方法创建对象

下面分别对上面列出的这些方式进行逐一介绍。

字面量的方式创建对象

基本写法

```
var  bookObj = {
       name:"声名狼藉者的生活",
       price:42.00,
       author:"福柯",
       press:"北京大学出版社",
       read:function () {
           console.log("我的书名为:声名狼藉者的的生活,作者为福柯....");
       }
   };
```

上面的代码中通过字面量的方式创建了bookObj对象，该对象拥有name、price、author和press属性，还拥有read方法。

对象字面量提供了一种非常方便的创建新对象的表示方法，一个对象就是包围在{}中的N(N>=0)个键值对。对象字面量可以出现在任何允许表达式出现的地方。

内置构造函数创建对象

JavaScript中的内置构造函数主要有：

```
String
Number
Boolean
Date
Array
Function
Object
RegExp
```

> ! 注意：(String Number Boolean 是区别于string number boolean的基本包装类型)

基本写法

```
var book = new Object();
   book.name = "声名狼藉者的生活";
   book.price = 42.00;
   book.author = "福柯";
   book.press = "北京大学出版社";
   book.read = function () {
       console.log("我的书名为:声名狼藉者的的生活,作者为福柯....");
   };
```

这种写法相对字面量创建方式而言不够简洁和直观，而且本身的代码复用性不好，不推荐。

工厂函数创建对象

工厂函数方式创建对象其本质是对内置构造函数创建对象的过程进行了封装，适用于大规模“批量生产”同类型的对象。
基本写法

```
//提供工厂函数
function createBookNew (name,price,author,press) {

    var book = new Object();
    book.name = name;
    book.price = price;
    book.author = author;
    book.press = press;
    book.read = function () {
        console.log("我的书名为:"+book.name+",作者为"+book.author+"....");
    };

    return book;
}

//使用工厂函数来创建对象
var book1 = createBookNew("声名狼藉者的的生活","42.00","福柯","北京大学出版社");
var book2 = createBookNew("人性的枷锁","49.00","毛姆","华东师范大学出版社");
var book3 = createBookNew("悟空传","28.00","今何在","湖南文艺出版社");

//打印对象的属性,调用对象的方法
console.log(book1.name);
console.log(book2.name);
console.log(book3.name);

book1.read();
book2.read();
book3.read();
```

总结工厂函数创建对象的实现过程

> ① 提供一个创建对象的函数（参数）
> ② 在函数内使用new 关键字和构造器创建对象

③ 设置对象的属性和方法
④ 返回加工过的对象

自定义构造函数创建对象

基本写法

```
 //提供构造函数
function CreateBook (name,price,author,press) {

   //使用new调用该构造函数时,默认在内部会先创建Object类型的实例对象
   //并把该对象关联到当前构造函数的原型对象上，并把函数内的this绑定到该对象
   //通过this来给实例对象设置属性和方法
    this.name = name;
    this.price = price;
    this.author = author;
    this.press = press;
    this.read = function () {
        console.log("我的书名为:"+this.name+",作者为"+this.author+"....");
    };

  //默认总是把新创建的实例对象返回
}

//使用new + 函数名() 的方式来调用
var bookObj = new CreateBook("声名狼藉者的的生活","42.00","福柯","北京大学出版社");
```

! 构造函数和普通函数没有本质区别，约定使用new调用的构造函数的首字母应该大写。构造函数的作用在于完成对象的初始化，对象的创建等工作由new关键字完成，组合使用。

工厂函数和构造函数创建对象过程简单对象

① 函数的首字母大写(用于区别构造函数和普通函数)
② 创建对象的过程是由new关键字实现
③ 在构造函数内部会自动的创建新对象,并赋值给this指针
④ 自动返回创建出来的对象

构造函数调用方式的返回值

① 如果在构造函数中没有显示的return，则默认返回的是新创建出来的对象
② 如果在构造函数中显示的return，则依照具体的情况处理
❒ return 的是对象类型数据，则直接返回该对象
❒ return 的是null或其他基本数据类型数据，则返回新创建的对象（即this）

提示 在开发中我们通过把自定义构造函数和原型对象结合在一起使用，这样可以充分的利用JavaScript原型链继承的特性并解决方法的共享问题。下面给出基本的代码示例：

```
// (1) 提供Person构造函数
function Person(name) {}
// (2) 替换Person默认的原型对象
Person.prototype ={
    //修正构造器属性 Object --> Person
    constructor:Person,
    //提供实例对象的初始化方法
    init:function(name,age){
        this.name = name || "默认的姓名";
        this.age = age || 18;
    },
    //所有实例对象都需要访问的原型方法
    showName:function () {
        console.log(this.name);
    }
};
// (3) 使用new来调用构造函数以创建实例对
var p = new Person();
// (4) 调用init方法对实例对象进行初始化操作
p.init("文顶顶",20);
```

使用Object.create方法创建对象

ES5提供了Object.create法来创建一个新对象，该方法在使用的时候会把传入的指定对象连接为新对象的原型对象。

语法

```
Object.create(proto, [propertiesObject])
```

参数说明
第一个参数proto：新创建对象的原型对象。
第二个参数propertiesObject：可选的参数，如果没有指定为 undefined，则表示要添加到新创建对象的可枚举属性信息，存放对象的属性描述符以及相应的属性名称。

> 如果传入的参数为null，则创建出来的空对象不会继承Object原型成员，没有基础方法。
> 如果传入的参数为Object.prototype，那么创建出来的对象等同于{}空对象。

代码示例

```
//01 字面量方式创建对象obj
var obj = {name:"wendingding",age:18};

//02 使用Object.create方法来创建新对象
var o = Object.create(obj);
//o是一个空对象，o.__proto__指向obj对象
console.log(o);
//wendingding 访问原型对象obj上面的name属性
console.log(o.name);

//03 测试传入null的情况
var o1 = Object.create(null);
//打印结果为空对象，No properties 该对象身上没有任何成员
```

```
console.log(o1);

//04 测试传入Object.prototype的情况
var o2 = Object.create(Object.prototype);
//o2 是空对象，等价于{}
console.log(o2);
```

1.3 javaScript中对象的操作

我们知道对象可以简单理解为键值对的集合，通过前面的阅读我们已经了解到如何创建对象，接下来我们接着探讨对象内部键值对的相关操作。

对象属性和方法的访问方式：点语法或者是[]语法。
对象属性和方法的操作方式：对象的操作方式和数据的操作方式保持一致，可以简单总结为 增删改查和遍历操作。

代码示例

```
 //00 提供obj对象
//通过字面量方式创建obj对象，该对象现在拥有name属性和showName方法
//因使用字面量方式创建，所有obj的原型对象(__proto__)指向object.prototype
var obj = {
    name:"wendingding",
    showName:function () {
        console.log(this.name);
    }
};

//01 添加属性或方法
//a 使用点语法来为obj对象添加age属性和showAge方法
obj.age = 18;
obj.showAge = function (){
    console.log(this.age);
};

//b 使用中括号语法来为obj对象添加age属性和showAge方法
obj["class-name"] = 41;
obj["showClassName"] = function () {
    console.log(this["class-name"]);
};

//02 修改属性或方法
//如果对象的属性已经存在，那么设置该属性的时候表示修改
obj.age = 20;
obj.showAge = function (){
    console.log("年龄" + this.age);
};

//03 查询属性或者调用方法
console.log(obj.name);      //wendingding
console.log(obj["age"]);    //20
obj.showName();             //wendingding
obj["showName"]();          //注意，不推荐使用这种方法
```

```
//04 删除对象中的属性或方法
//语法形式：delete 对象.属性 | delete 对象[属性]
delete obj.name;
delete obj["showName"];
console.log(obj);

//05 对象的遍历
for (key in obj)
{
     console.log(key, obj["key"]);
}
```

delete关键字说明

```
01 具体使用：
        (1) 可以用来删除对象中指定的属性
        (2) 用来删除没有使用var关键字声明的变量  delete 变量名|window.变量
02 使用注意点
        (1) 关键字在使用的时候有返回值,true|false 删除成功返回true
        (2) 删除对象中不存在的属性,返回true
        (3) 使用var声明的变量不能被直接删除
        (4) 不能删除使用var声明的全局变量但是可以删除直接添加在window上面的属性
        (5) 如果删除数组中的元素（数组也是对象）则数组中对应的元素内容被替换为undefined，索引保留。
```

对象属性说明

> ❒ 对象的属性名可以是任意字符串，包括空字符。
> ❒ 对象的属性名如果是合法的JavaScript标识符，则不必强制要求使用双引号括住属性名。
> ❒ 属性值可以是任何值（包括undefined）。
> ❒ 如果对象的属性名并非合法标识符，则建议使用[]语法来访问对象。
> ❒ 标识符规范：字母开头，后面跟1个或多个下划线、数字或字母。