第五课 函数与对象

学习目录

- instanceof
- 函数与对象

一．instanceof

在检测数据类型的时候，我们可以使用 typeof 操作符，凡是如果检测的数据是引用类型，这个操作符的用处不大。因为大多数情况下，我们并不是想知道某个值是不是对象，更多的时候是想知道它是什么类型的对象。因此我们可以使用 instanceof 操作符。

```
var obj = {};
console.log(obj instanceof Object);
```

数组可以看成是有序数据的集合仓库，而对象则可以看成无序数据的集合仓库，因此，某种意义上说，数组可以看成是对象的一种子集，但是函数跟对象之间并不是那么简单。

二．函数与对象

首先从之前的检测可以看出，函数是一种对象，不过函数跟数组还不太一样，函数与对象之间更像是一种相互依存的关系。

1.函数是一种对象

```
var fn = function(){};
console.log(fn instanceof Object);
console.log(fn instanceof Function);
```

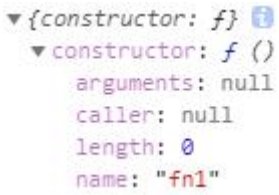
```
console.log(Function instanceof Object);
```

2.对象都是通过函数实例化来创建的

```
var obj1 = {};

var obj2 = new Object();
```

函数与对象的这种关系其实是通过函数的一个属性来作为关系纽带的，我们可以看到每个函数都有一个属性叫做 prototype，这个 prototype 的属性值本身也是一个包含各种属性的对象。默认情况下他下面有一个属性 constructor，这个属性就代表了当前这个函数。



我们也可以手动添加属性和方法到函数的原型上，这样就可以丰富封装我们的原型，因此可以实例出各种实例对象，他们就能拥有原型上的属性和方法，比如数组，通过实例出来的数组，可以使用 Array.prototype 提供的各种属性和方法，每个实例之间也不会有过多相互干扰，因此这也是面向对象封装的基础根据。

```
var Fn1 = function(){};

Fn1.prototype.abc = 123;

Fn1.prototype.cba = function(){console.log('cba');};

var fn1 = new Fn1();

console.log(Fn1.prototype);
```

Fn1 是函数，fn1 是从 Fn 函数 new 出来的实例对象，fn1 就可以调用 Fn.prototype 中的属性和方法。

谢谢观看！

我是星星课堂老师：周小周