H5第1周第1天 讲议

一、对象原型

1.函数本身就是一个对象，可以称之为函数对象

```
function show() {
}
show.a = 100;
console.log(show.a);//100
```

2.请大家分清楚下面3种情况下的函数

(1)Function函数， 由js内置。即天生就存在的。

(2)Object函数，由Function函数创建

(3)自定义函数，由Function函数创建

可以说Object函数与自定函数一样，底层都是由Function new出来的。

```
function f1() {}
//上面的函数等同于：
var f1 = new Function();

function sum(a, b) {
    return a + b;
}
//上面的函数等同于：
var sum = new Function("a", "b", "return a + b");//前面的是函数形参名，最后一个参数是函数体
```

3. 变量引用对象,当一个对象没有变量引用时，就会被会GC

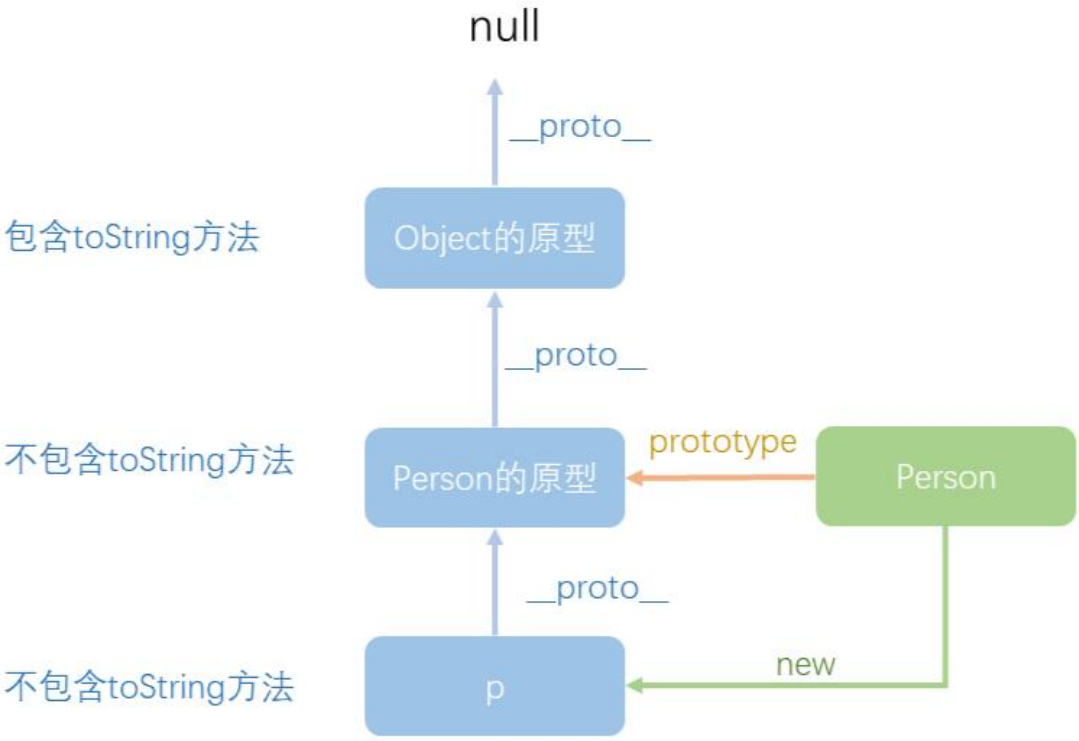
```
function Stu() {
    this.teacher = null;
    this.name = "小学生";
}
function Teacher() {
    this.major = "教学";
    this.stu = null;
}
let s = new Stu();
s.teacher = new Teacher();
s.teacher.stu = s;

let s1 = new Stu();
let s2 = s1;
```

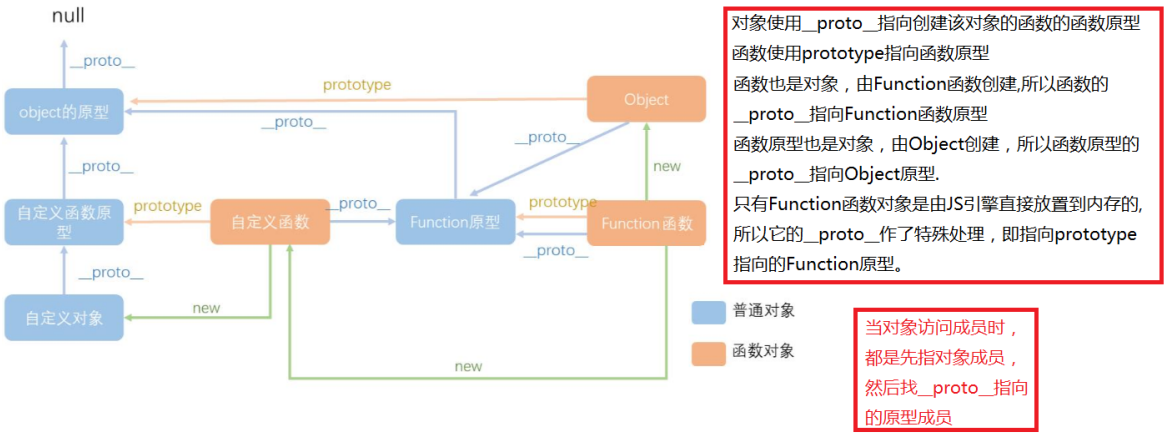
4.每人函数对象都唯一的一个函数原型对象与之对应。实计上所有的函数原型对象都底层里都是被Object new出来的。

5.函数对象都有一个prototype属性指向函数原型对象。函数原型对象都有一个construct引用函数对象。

6.对象都有一个 __proto__属性，引用创建该对象的函数对象的函数原型对象

```
//以下代码执行的结果是什么？为什么
function Person() {
}
const p = new Person();
p.toString();
```





课后作业：

```
//以下代码执行的结果是什么？为什么
function Person() {
    this.toString = function () {
        return "这是对象自身的toString";
    }
}
Person.prototype.toString = function () {
    return "这是对象隐式原型中的toString";
}
const p = new Person();
console.log(p.toString());
```

二、This

1基本使用

a. this出现在函数体中，执行时表示调用该函数的对象

b.练习题，下面的题的打印结果是多少

```
        function fn() {
           console.log(this.a);
        }
        var a = 2;
        fn();
```

```
        var a = 20;
        function fm() {
            console.log(this.a);
        }
        var obj = {
            a: 2,
            fn: fm
        };
        fm();
        obj.fn();
```

```
        function fn() {
            console.log(this.a);
        }
        var obj2 = {
            a: 42,
            fn: fn
        };
        var obj1 = {
            a: 2,
            obj2: obj2
        };
        obj1.obj2.fn();
```

```
function fn() {
    console.log(this.a);
}
var obj = {
    a: 2,
    fn: fn
};
var bar = obj.fn;
var a = "全局";
bar();
obj.fn();
```

2.绑定this

call方法：

语法：call([thisObj[,arg1[, arg2[, [,.argN]]]]])

定义：调用一个对象的一个方法，以另一个对象替换当前对象。

说明：call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

apply方法：

语法：apply([thisObj[,argArray]])

定义：应用某一对象的一个方法，用另一个对象替换当前对象。

说明：如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。

如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

bind方法：

语法：bind(thisArg[, arg1[, arg2[, ...]]])

定义：将接受多个参数的函数变换成接受一个单一参数。

说明：bind()方法所返回的函数的length（形参数量）等于原函数的形参数量减去传入bind()方法中的实参数量（第一个参数以后的所有参数），因为传入bind中的实参都会绑定到原函数的形参。

示例：

```
let teacher = {
    name:'秦可卿',
    show:function(age, sex) {
        console.log(this.name, age, sex);
    }
}
let stu = {
    name:'宝玉',
}
teacher.show(30,'女');
teacher.show.call(stu,20,'男');
teacher.show.apply(stu,[20,'男']);
teacher.show.bind(stu)(20,'男');
```

3 练习题

下面代码执行后打印的结果是什么？

例1：

```
var name = '小华';
let stu = {
    name: '小明',
    showName: function() {
        console.log(this.name);
    }
}
stu.showName();
stu.showName.bind(window)();
stu.showName.bind(stu)();
```

例2：

```
var name = '小华';
let stu = {
    name: '小明',
    showName: ()=> {
        console.log(this.name);
    }
}
stu.showName();
stu.showName.bind(window)();
```

三、函数

3.1.匿名函数

可以为变量赋值; 当作实参; 用于自调用

3.2.箭头函数

a.可以为变量赋值； 当作实参；用于自调用

b.箭头函数的形式

b1.去掉function和函数名，就一定是箭头函数了

b2.如果只有一个参数时，则包含参数的括号可省

b3.如果只有一个语句时，方法体的{}可省。

b4.如果只有一个语句，并且该语句是return 值。则 方法体的{}可省，语句中的return 必省

c.通过上面示例感受匿名函数和箭头函数的区别

3.3.自调用函数

函数都是可以自调用的

3.4.函数的回调

当有异步执行的时候，回调是必须的。

四、Let、var和Const

五、类

1.创建空类 2.类的构造函数 3.通过类创建对象 4.实例成员 5.原型成员 6.静态成员

六、继承

1.继承是什么 2.自动继承Object 3.使用extends继承父类 4.继承的意义 5.子类的构造函数中super 6.instanceof 7.方法重写(overwrite), 重载(overload) 8.super在方法中的使用