课堂案例

1. 函数和对象的关系

1.1 JS中的数据类型

- 原始类型（值类型）： Number、String、Boolean、Null、Undefined
- 对象类型（引用类型）：Object、Function、Array .....

1.2 关于对象

- 狭义上对象：Object对象
- 广义的对象：一切皆对象 包括 Function、Array、String、Boolean、Number、自定义的构造函数产生的对象

1.3 函数和构造函数

- 一个函数是普通函数还是构造函数，取决于怎么去用。调用就是普通函数，new 就是构造函数。
- 为了能够区分，一般把构造函数首字母大写。

1.4 对象和构造函数

- 对象是构造函数的实例，构造函数是对象的抽象（构造函数描述对象）。构造函数就像模型，对象是参照模型生成出来的产品。
- 对象由构造函数产生。
- 系统有很多定义好的构造函数：Object、Function、Array、String、Boolean、Number.....
- 构造函数决定了对象的数据类型。相同类型的对象构造函数是一样的。

2. 函数的返回值(return)问题

2.1 函数本身和函数调用

- 如果函数名后面没有()，是对函数的引用，函数不会调用。
- 函数名后面加（），函数会被调用，函数内的代码会执行。函数名加括号，组成了函数调用表达式，表达式的值就是函数的返回值。

> **函数名后面有括号是调用， 没有括号是引用这个函数**

2.2 return 语法规则

- 函数调用表达式的值 就是 return 的值（返回值）
- 如果函数中没有 return，默认返回的是 undefined
- 在函数内，return 后面的语句都不执行。return 把函数结束了。

> **什么时候需要写 return：**

> 如果函数的作用是为了计算某个结果，就可以把计算好的结果 return，这样的**函数**通常是有参数的。
>
> 如果函数的作用是为了进行某个操作，没有一个最终的运算结果，不需要 return，这样的函数**通常**没有参数。

3. this 指向总结

3.1 函数或方法中

> 全局函数本质上是 window 对象一个方法

在函数或方法中使用 this， this指向调用该函数或方法的对象。谁调用指向谁。

3.2 构造函数中

在构造函数中，this 指向构造函数创造的对象。

> 在 new 一个构造函数的时候，会把 this 指向所 new 出来的对象。

4. 原型总结

4.1 原型

- 每个对象都有一个原型，对象的原型仍然是一个对象。
- 当我们使用对象中某个属性的时候，如果对象不存在，会去对象的原型上找。（对象会继承原型的属性）

4.2 原型链

- 对象的原型仍然是一个对象，原型作为一个对象也有自己的原型，一直到最顶部一个没有原型的对象。组成了原型链。
- 当我们使用对象中某个属性的时候，如果对象不存在，会去对象的原型上找，如果还没有会继续在原型的原型找，一直找到原型链的结束。找不到才返回 undefined。

4.3 获取原型

- 隐式原型的获取方式：`对象.__proto__`
- 显示原型的获取方式：`构造函数.prototype`

> 同一个构造函数的不同实例，原型是一样的

> 所有数组的原型都是 Array.prototype