链滴

如何在 JavaScript 中实现单例?

作者：Vanessa
原文链接：https://ld246.com/article/1583293909265
来源网站：链滴

2020-03-04

回答

较为高级灵活的方式是使用 Proxy，以下是将一个 class 转换为一个单例的最基本实现：

```
const singletonify = (className) => {
  return new Proxy(className.prototype.constructor, {
    instance: null,
    construct: (target, argumentsList) => {
      if (!this.instance)
        this.instance = new target(...argumentsList);
      return this.instance;
    }
  });
}

class MyClass {
  constructor(msg) {
    this.msg = msg;
  }

  printMsg() {
    console.log(this.msg);
  }
}

MySingletonClass = singletonify(MyClass);

const myObj = new MySingletonClass('first');
myObj.printMsg();          // 'first'
const myObj2 = new MySingletonClass('second');
myObj2.printMsg();          // 'first'
```

在上面代码中，我们可以看到 MySingletonClass 并没有被第二次实例化，这是由于当前的实例已经在，因此返回该实例来替代新对象的创建。

加分回答

- 单例模式是一种面向对象的软件设计模式，以确保给定的类只会进行一次实例化，这在很多不同的景下都非常有用。例如，在应用中创建一个全局共享的对象和组件
- 虽然 JavaScript 支持了面向对象的编程，但似乎没有提供很多简单的方式来实现此模式
- Proxy 对象可以用于定义所谓的捕获器，他的方法允许为某些操作进行自定义行为，如属性查找、值、枚举、函数调用等
- 单例模式规定给定的类只能实例化一次，这意味着最有用的捕获器是 handler.construct()，这是ew 操作的捕获获器
- 由于 Proxy 中的 handler 本身是一个对象，当类被实例化后，我们可以使用他存储我们想要的的唯一实例，同时还可以通过 handler.construct() 为 new 操作提供一个捕获器
- 如上，我们就可以创建一个对象，让任何类都可以简单的转换为单一模式。同时，他还允许我们为能需要到的其他自定义操作提供额外的捕获器

- 以上代码虽然是一个 singletonify 方法的最小实现，但他能很容易的进行扩展：我们可以进一步改其行为；我们甚至可以在后续的调用中使用一些传递给构造器的数据来更新他自己的 instance

返回总目录

每天 30 秒系列之前端面试

原文链接：如何在 JavaScript 中实现单例?