链滴

如何实现一个单例用 JavaScript

作者：Vanessa

原文链接：https://ld246.com/article/1595900739815

来源网站：链滴

2020-07-27

描述

单例是一种面向对象的软件设计模式，用来确保给定的类只能被实例化一次，这在很多不同的情况下非常有用，例如创建在应用程序之间共享的全局对象和组件。虽然 JavaScript 支持面向对象编程，但并没有提供许多简单的方式来实现这个模式。

说明

Proxy object 虽然某些方面比较超前，但他最容易扩展。Proxy 对象可用于定义所谓的 traps，这些法允许为某些操作，如属性查找、赋值等行为进行自定义。单例模式规定给定的类只能有一个实例，意味着最有用的 trap 是 handler.construct()，该 trap 为 new 操作。

由于 handler 本身为一个对象，因此我们可以使用他来存储我们所需类的唯一实例，如果他已实被例，同时还可以通过 handler.construct() 为 new 操作提供 trap。这样，我们就能为任何我们想要转为单例的类简单复用的创建一个对象，当然，我们还可以为其他任何我们可能需要的操作提供额外的定义 trap。

代码

将一个 class 转换为一个单例最基本的方法：

```
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;
    }
  });
}
```

这有一个简单的使用示例可以让你更好的理解他的原理：

```
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
```

总结

在上面的示例中，你可以看到 MySingletonClass 并没有被第二次实例化，这是因为实例已经存在，此返回实例本身来替代新对象的创建。以上代码虽然只是 singletonify 方法的最小实现，但他能很容的进行扩展：我们可以进一步修改其行为；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来更新他自己的 instance。

返回总目录

每天 30 秒系列之 JavaScript 代码