实现一个 Storage

描述

> 实现Storage，使得该对象为单例，基于 localStorage 进行封装。实现方法 setItem(key,value) 和 getItem(key)。

思路

拿到单例模式相关的面试题，大家首先要做的是回忆我们上个小节的“基本思路”部分——至少要记起来 `getInstance` 方法和 `instance` 这个变量是干啥的。

具体实现上，把判断逻辑写入静态方法或者构造函数里都没关系，最好能把闭包的版本也写出来，多多益善。

总之有了上节的基础，这个题简直是默写！

> 实现：静态方法版

```
// 定义Storage
class Storage {
    static getInstance() {
        // 判断是否已经new过1个实例
        if (!Storage.instance) {
            // 若这个唯一的实例不存在，那么先创建它
            Storage.instance = new Storage()
        }
        // 如果这个唯一的实例已经存在，则直接返回
        return Storage.instance
    }
    getItem (key) {
        return localStorage.getItem(key)
    }
    setItem (key, value) {
        return localStorage.setItem(key, value)
    }
}

const storage1 = Storage.getInstance()
const storage2 = Storage.getInstance()

storage1.setItem('name', '李雷')
// 李雷
storage1.getItem('name')
// 也是李雷
storage2.getItem('name')

// 返回true
storage1 === storage2
```

> 实现： 闭包版

```
// 先实现一个基础的StorageBase类，把getItem和setItem方法放在它的原型链上
function StorageBase () {}
StorageBase.prototype.getItem = function (key){
    return localStorage.getItem(key)
}
StorageBase.prototype.setItem = function (key, value) {
    return localStorage.setItem(key, value)
}

// 以闭包的形式创建一个引用自由变量的构造函数
const Storage = (function(){
    let instance = null
    return function(){
        // 判断自由变量是否为null
        if(!instance) {
            // 如果为null则new出唯一实例
            instance = new StorageBase()
        }
        return instance
    }
})()

// 这里其实不用 new Storage 的形式调用，直接 Storage() 也会有一样的效果
const storage1 = new Storage()
const storage2 = new Storage()

storage1.setItem('name', '李雷')
// 李雷
storage1.getItem('name')
// 也是李雷
storage2.getItem('name')

// 返回true
storage1 === storage2
```

实现一个全局的模态框

描述

> 实现一个全局唯一的Modal弹框

思路

这道题比较经典，基本上所有讲单例模式的文章都会以此为例，同时它也是早期单例模式在前端领域的最集中体现。

万变不离其踪，记住 `getInstance` 方法、记住 `instance` 变量、记住闭包和静态方法，这个题除了要多写点 HTML 和 CSS 之外，对大家来说完全不成问题。

实现

完整代码如下：

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>单例模式弹框</title>
</head>
<style>
    #modal {
        height: 200px;
        width: 200px;
        line-height: 200px;
        position: fixed;
        left: 50%;
        top: 50%;
        transform: translate(-50%, -50%);
        border: 1px solid black;
        text-align: center;
    }
</style>
<body>
    <button id='open'>打开弹框</button>
    <button id='close'>关闭弹框</button>
</body>
<script>
    // 核心逻辑，这里采用了闭包思路来实现单例模式
    const Modal = (function() {
        let modal = null
        return function() {
            if(!modal) {
                modal = document.createElement('div')
                modal.innerHTML = '我是一个全局唯一的Modal'
                modal.id = 'modal'
                modal.style.display = 'none'
                document.body.appendChild(modal)
            }
            return modal
        }
    })()

    // 点击打开按钮展示模态框
    document.getElementById('open').addEventListener('click', function() {
        // 未点击则不创建modal实例，避免不必要的内存占用;此处不用 new Modal 的形式调用也可以，和
Storage 同理
        const modal = new Modal()
        modal.style.display = 'block'
    })

    // 点击关闭按钮隐藏模态框
    document.getElementById('close').addEventListener('click', function() {
        const modal = new Modal()
        if(modal) {
            modal.style.display = 'none'
        }
    })
</script>
</html>
```

是不是发现又是熟悉的套路？又可以默写了？（ES6 版本的实现大家自己尝试默写一下，相信对现在的你来说已经非常简单了）。

这就是单例模式面试题的特点，准确地说，是所有设计模式相关面试题的特点——牢记核心思路，就能举一反三。所以说设计模式的学习是典型的一分耕耘一分收获，性价比极高。

（阅读过程中有任何想法或疑问，或者单纯希望和笔者交个朋友啥的，欢迎大家添加我的微信xyalinode与我交流哈~）