官方开发文档：

- 每个微信小程序都可以自己的本地缓存，即对本地缓存进行设置、获取和清理。同一个微信用户，同一个小程序 storage 上限为 10MB，单个 key 允许存储的最大数据长度为 1MB。

- 注意：如果用户储存空间不足，小程序会清空最近久未使用的小程序的本地缓存。我们不建议将关键信息全部缓存起来，以防储存空间不足或用户换设备的情况。

1.设置缓存

```
wx.setStorage(OBJECT)
wx.setStorageSync(KEY,DATA) //是setStorage的同步版本
```

```
wx.setStorage(Object object)
```

将数据存储在本地缓存中指定的 key 中。

会覆盖掉原来该 key 对应的内容。

除非用户主动删除或因存储空间原因被系统清理，否则数据都一直可用。

单个 key 允许存储的最大数据长度为 1MB，所有数据存储上限为 10MB

参数

属性	类型	默认值	必填	说明
key	string		是	本地缓存中指定的 key
data	any		是	需要存储的内容。只支持原生类型、Date、及能够通过 `JSON.stringify` 序列化的对象。

2.获取缓存数据

```
wx.getStorage(Object object)
wx.getStorageSync(string key) //wx.getStorage的同步版本
```

```
wx.getStorage(Object object)
```

- 从本地缓存中异步获取指定 key 的内容

属性	类型	默认值	必填	说明
key	string		是	本地缓存中指定的 key
success	function		否	接口调用成功的回调函数
fail	function		否	接口调用失败的回调函数
complete	function		否	接口调用结束的回调函数（调用成功、失败都会执行）

object.success 回调函数

参数 Object res

属性	类型	说明
data	any	key对应的内容

any 类型用于描述一个我们根本不知道类型的变量，或者说可以是任意类型的变量

3.移除缓存

```
wx.removeStorage(Object object)
wx.removeStorageSync(string key)
```

4.清除缓存

```
wx.clearStorage()
wx.clearStorageSync()
```

--- 提示：同步与异步区别

以Sync（同步，同时）结尾的都是都是同步缓存，二者的区别是，异步不会阻塞当前任务，同步缓存直到同步方法处理完才能继续往下执行。

通俗点说,异步就是不管保没保存成功,程序都会继续往下执行.同步是等保存成功了,才会执行下面的代码.

使用异步,性能会更好;而使用同步,数据会更安全

一般都使用同步，异步是为了用户体验的情况而选择，同步相对简单。

5.案例一：缓存电影数据

wx.js

```
onLoad: function (options) {
    // 从缓存中取数据
    var filmList=wx.getStorageSync('filmList')
    if (filmList) {
        // 从本地加载数据
        console.log("本地加载数据")
        this.setData({
            filmList
        })
    } else {
        // 设置URL地址
        const url = "https://api.vvhan.com/api/douban"
        //发起网络请求
        wx.request({
            url,
            method: 'GET',
            success: res => {
                console.log("访问网络")
                this.setData({
                    filmList:res.data.data
                })
                // 缓存数据
                wx.setStorageSync('filmList', res.data.data)
            }
        })
    }
},
```

6.案例2：缓存表单数据

1.创建storage页面

2.页面布局 storage.wxml

```
<!--pages/storage/storage.wxml-->
<view>
    <form bind:submit='login'>
        <view class="bot-input">
            <view class="bot">用户名:</view>
            <view class="input">
                <input type="text" name="username" placeholder="请输入用户名" value="{{account}}">
</input>
            </view>
        </view>

        <view class="bot-input">
            <view class="bot">密码:</view>
            <view class="input">
```

```
                <input type="password" name="password" placeholder="请输入密码" value="{{pwd}}">
</input>
            </view>
        </view>

        <view><button type="primary" form-type="submit">登录</button></view>
    </form>
</view>
```

3.样式设置storage.wxss

```
/* pages/storage/storage.wxss */
.bot-input{
    display: flex;
}

.input{
    border: 1px solid #ccc;
    margin: 10rpx;
}

.bot{
    width: 20%;
    text-align: right;
}
```

4.写入缓存与读取缓存数据逻辑 storage.js

```
// pages/storage/storage.js
Page({
    /**
    * 页面的初始数据
    */
    data: {
        account:"",
        pwd:""
    },

    /**
    * 生命周期函数--监听页面加载
    */
    onLoad: function (options) {
        // 从缓存加载数据
        const account= wx.getStorageSync('account')
        const pwd= wx.getStorageSync('pwd')
        this.setData({
            account,
            pwd
        })
    },

    login:function(e){
        console.log(e.detail.value)
        var info=e.detail.value
```

```
        // 把表单中输入的信息写入到缓存中
        wx.setStorage({key:"account",data:info.username})
        wx.setStorage({key:"pwd",data:info.password})
    }
})
```

5.效果