链滴

记 react-native 离线缓存实现

作者：gmw-zjw

原文链接：https://ld246.com/article/1572608456027

来源网站：链滴

在rn中使用离缓存，首先我们使用AsyncStorege 来实现缓存，下面开始实现：

```
export default class DataStore {
    /// 保存数据
   saveData(url,data, callback) {
   if (!data || !url) return;
   AsyncStorage.setItem(url, JOSN.stringify(this._wrapData(data)), callback)
   }


// 获取本地数据
   fetchLocalData(url) {
       return new Promise((resolve, reject) => {
           AsyncStorage.getItem(url, (error, result) => {
               if (!error) {
                   try {
                       resolve(JSON.parse(result))
                   } catch (e) {
                       reject(e);
                       console.error(e);
                   }
               } else {
                   reject(error);
                   console.error(error);
               }
           })
       })
   }

}
```

对data进行处理

```
/// 处理data
   _wrapData(data) {
       return {
           data: data,
           timestamp: new Date().getTime()
       }
   }
```

获取网络数据

```
// 获取网络数据
   fetchDNetData(url) {
       return new Promise((resolve, reject) => {
           fetch(url)
               .then((response) => {
                   // 判断是否请求成功，返回正确的字段
                   if (response.ok) {
                       return response.json()
                   }
                   throw new Error(response.error())
```

原文链接：记 react-native 离线缓存实现

```
            })
            .then((responseData) => {
                /// 为了节流，将数据保存在本地，设置过期时间
                this.saveData(url, responseData);
                resolve(responseData);
            })
            .catch((e) => {
                /// 抛出拦截的错误信息
                reject(e);
                console.error(e);
            })
    })
}
```

离线缓存主要实现

```
/// 离线缓存实现
    /// 主要思想: 首先会在本地查找数据是否存在，如果存在那么使用本地，否则使用网络数据
    fetchData(url) {
        return new Promise((resolve, reject) => {
            /// 首先会在本地查找数据是否存在，如果存在那么使用本地，否则使用网络数据
            this.fetchDNetData(url).then((wrapData) => {
                /// 如果本地有数据，并且数据在有效时期内
                if (wrapData && DataStore.checkTimestampValid(wrapData)) {
                    resolve(wrapData);
                } else {
                    this.fetchDNetData(url)
                        .then(data => {
                            resolve(this._warperData(data))
                        })
                        .catch(e => {
                            reject(e);
                            console.error(e);
                        })
                }
            })
                .catch(e => {
                    /// 如果出错会直接再次获取数据
                    this.fetchDNetData(url)
                        .then(data => {
                            resolve(this._warperData(data));
                        })
                        .catch(e => {
                            reject(e);
                            console.error(e);
                        })
                })
        })
    }
```

检查是否过期

```
/// 检查保存数据时效性是否过期
    static checkTimestampValid(timestamp) {
```

```
    const currentDate = new Date();
    const targetDate = new Date();
    targetDate.setTime(timestamp);
    if (currentDate.getFullYear() !== targetDate.getFullYear()) return false;
    if (currentDate.getMonth() !== targetDate.getMonth()) return false;
    if (currentDate.getDate() !== targetDate.getDate()) return false;
    if (currentDate.getHours() !== targetDate.getHours() > 4) return false;
    return true;
}
```