蚂蚁金服数据体验技术 Lv5

2017年09月15日 阅读 5145

关注

4

跨页面通信的各种姿势

作者简介：nekron 蚂蚁金服·数据体验技术团队

将跨页面通讯类比计算机进程间的通讯，其实方法无外乎那么几种，而web领域可以实现的技术方案主要是类似于以下两种原理：

- 获取句柄，定向通讯
- 共享内存，结合轮询或者事件通知来完成业务逻辑

由于第二种原理更利于解耦业务逻辑，具体的实现方案比较多样。以下是具体的实现方案，简单介绍下，权当科普：

一、获取句柄

具体方案

父页面通过 `window.open(url, name)` 方式打开的子页面可以获取句柄，然后通过postMessage完成通讯需求。

```js
// parent.html
const childPage = window.open('child.html', 'child')

childPage.onload = () => {
        childPage.postMessage('hello', location.origin)
}

// child.html
window.onmessage = evt => {
        // evt.data
}
```

tips

1. 当指定 `window.open` 的第二个name参数时，再次调用 `window.open('****', 'child')` 会使之前已经打开的同name子页面刷新
2. 由于安全策略，异步请求之后再调用 `window.open` 会被浏览器阻止，不过可以通过句柄设置子页面的url即可实现类似效果

复制代码

```
// 首先先开一个空白页
const tab = window.open('about:blank')

// 请求完成之后设置空白页的url
fetch(/* ajax */).then(() => {
        tab.location.href = '****'
})
```

优劣

缺点是只能与自己打开的页面完成通讯，应用面相对较窄；但优点是在跨域场景中依然可以使用该方案。

二、localStorage

具体方案

设置共享区域的storage，storage会触发storage事件

js 复制代码

```
// A.html
localStorage.setItem('message', 'hello')

// B.html
window.onstorage = evt => {
```

```
  // evt.key, evt.oldValue, evt.newValue
}
```

tips

1. 触发写入操作的页面下的**storage listener**不会被触发
2. storage事件只有在发生改变的时候才会触发，即重复设置相同值不会触发listener
3. safari隐身模式下无法设置localStorage值

优劣

API简单直观，兼容性好，除了跨域场景下需要配合其他方案，无其他缺点

三、BroadcastChannel

具体方案

和 `localStorage` 方案基本一致，额外需要初始化

js 复制代码

```
// A.html
const channel = new BroadcastChannel('tabs')
channel.onmessage = evt => {
        // evt.data
}

// B.html
const channel = new BroadcastChannel('tabs')
channel.postMessage('hello')
```

优劣

和 `localStorage` 方案没特别区别，都是同域、API简单， `BroadcastChannel` 方案兼容性差些（chrome > 58），但比 `localStorage` 方案生命周期短（不会持久化），相对干净些。

4

四、SharedWorker

具体方案

`SharedWorker` 本身并不是为了解决通讯需求的，它的设计初衷应该是类似总控，将一些通用逻辑放在SharedWorker中处理。不过因为也能实现通讯，所以一并写下：

js 复制代码

```
// A.html
var sharedworker = new SharedWorker('worker.js')
sharedworker.port.start()
sharedworker.port.onmessage = evt => {
        // evt.data
}

// B.html
var sharedworker = new SharedWorker('worker.js')
sharedworker.port.start()
sharedworker.port.postMessage('hello')

// worker.js
const ports = []
onconnect = e => {
        const port = e.ports[0]
```

```
        ports.push(port)
        port.onmessage = evt => {
                ports.filter(v => v!== port) // 此处为了贴近其他方案的实现，剔除自己
                .forEach(p => p.postMessage(evt.data))
        }
}
```

4

优劣

相较于其他方案没有优势，此外，API复杂而且调试不方便。

五、Cookie

具体方案

一个古老的方案，有点 `localStorage` 的降级兼容版，我也是整理本文的时候才发现的，思路就是往 `document.cookie` 写入值，由于cookie的改变没有事件通知，所以只能采取轮询脏检查来实现业务逻辑。

方案比较丑陋，势必被淘汰的方案，贴一下原版思路地址，我就不写demo了。

communication between browser windows (and tabs too) using cookies

优劣

相较于其他方案没有存在优势的地方，只能同域使用，而且污染cookie以后还额外增加AJAX的请求头内容。

六、Server

之前的方案都是前端自行实现，势必受到浏览器限制，比如无法做到跨浏览器的消息通讯，比如大部分方案都无法实现跨域通讯（需要增加额外的postMessage逻辑才能实现）。通过借助服务端，还有很多增强方案，也一并说下。

乞丐版

4

后端无开发量，前端定期保存，在tab被激活时重新获取保存的数据，可以通过校验hash之类的标记位来提升检查性能。

```js 复制代码
window.onvisibilitychange = () => {
        if (document.visibilityState === 'visible') {
                // AJAX
        }
}
```

Server-sent Events / Websocket

项目规模小型的时候可以采取这类方案，后端自行维护连接，以及后续的推送行为。

SSE

```js 复制代码
// 前端
const es = new EventSource('/notification')

es.onmessage = evt => {
        // evt.data
```

```
}
es.addEventListener('close', () => {
        es.close()
}, false)


// 后端，express为例
const clients = []

app.get('/notification', (req, res) => {
        res.setHeader('Content-Type', 'text/event-stream')
        clients.push(res)
        req.on('aborted', () => {
                // 清理clients
        })
})
app.get('/update', (req, res) => {
        // 广播客户端新的数据
        clients.forEach(client => {
                client.write('data:hello\n\n')
                setTimeout(() => {
                        client.write('event:close\ndata:close\n\n')
                }, 500)
        })
        res.status(200).end()
})
```

Websocket

`socket.io` 、 `sockjs` 例子比较多，略

消息队列

项目规模大型时，需要消息队列集群长时间维护长链接，在需要的时候进行广播。

提供该类服务的云服务商很多，或者寻找一些开源方案自建。

例如MQTT协议方案（阿里云就有提供），web客户端本质上也是websocket，需要集群同时支持ws和mqtt协议，示例如下：

4

js 复制代码

```
// 前端
// 客户端使用开源的Paho
// port会和mqtt协议通道不同
const client = new Paho.MQTT.Client(host, port, 'clientId')

client.onMessageArrived = message => {
        // message. payloadString
}
client.connect({
        onSuccess: () => {
                client.subscribe('notification')
        }
})
// 抑或，借助flash（虽然快要被淘汰了）进行mqtt协议连接并订阅相应的频道，flash再通过回调抛出消息

// 后端
// 根据服务商提供的Api接口调用频道广播接口
```

原文地址: github.com/ProtoTeam/b...

蚂蚁金服数据体验技术 Lv5 数据体验技术团队（ProtoTeam） @ 蚂蚁金服（杭州）网络技术有...
发布了 38 篇专栏 · 获得点赞 14,711 · 获得阅读 324,399

关注

4

评论

输入评论...

xhhcocos2d

storage事件 ios webview中不支持，visibilityState ios多个webview切换不改变状态，感觉借助服务端是个相对安全的方案，当然也有可能是我司webview的问题，纯属交流，SharedWorker和BroadcastChannel这两个没试过，我要涨下姿势，实验一把，谢谢

5天前 回复

了了君丶

postMessage感觉很好

8月前 回复

内小子 前端 @ 伟大的娱乐公司

localstorage如何做到不在子页面监听直接在父页面往子页面写数据

9月前 回复

使用中的小黄瓜 前端 @ 利马

公司最近在实现一个支持多核的mqtt broker，感觉通信这一块node还是差点

1年前 回复

Roger1489408382442

友情提示：postMessage跨窗口消息请在IE11下测试兼容性。
最佳解决方案还是opener

1年前 1 回复

蚂蚁金服数据体验技术 Lv5 (作者) 数据体验技术团队（ProtoTeam...

回复 Roger1489408382442: 赞，好思路，多谢您的分享~

1年前 回复

4

Nekron Lv2 一条咸鱼 @ 蚂蚁金服

回复 Roger1489408382442: = =! 是我孤陋寡闻了哈~我去了解一下[害羞]

1年前 回复

hustcc 前端攻城军 @ alipay

回复 Roger1489408382442: opener 是针对 window.open，无法针对 iframe 吧?

1年前 回复

曹磊 前端开发工程师 @ 蚂蚁金服

跨起来

1年前 回复

查看更多 >