MessageChannel是什么，怎么使用？

ceido (/u/fcdf49ef65bb) ＋关注
2018.06.27 16:16* 字数 775 阅读 315 评论 0 喜欢 0
(/u/fcdf49ef65bb)

我们知道：在浏览器环境中，常见的 **macro task** 有 `setTimeout`、 `MessageChannel`、 `postMessage`、 `setImmediate`。而常见的 **micro task** 有 `MutationObsever` 和 `Promise.then`。

Vue中对于 macro task 的实现，优先检测是否支持原生 `setImmediate`，这是一个高版本 IE 和 Edge 才支持的特性，不支持的话再去检测是否支持原生的 `MessageChannel`，如果也不支持的话就会降级为 `setTimeout 0`。

现在我想了解一下这个 `MessageChannel`，查了一下还是看规范比较好。额，平时都没有去看过，现在的问题是：去哪里看规范？这个问题有点废，但我还是希望能搞清楚一点。

W3Schools 跟 W3C 组织没有关系 (https://www.v2ex.com/t/303172)

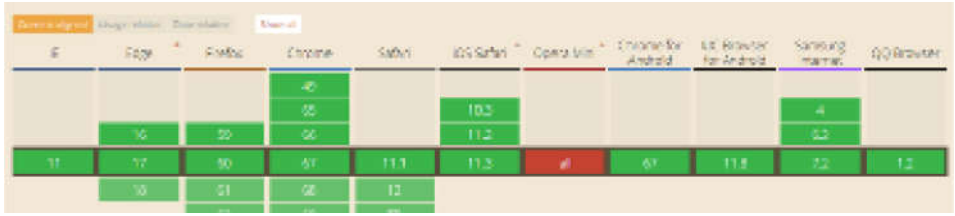
> W3Schools 的内容漏洞百出，而且没有给阅读者深入了解的参考，对所有内容浅尝辄止，不是一个教程该有的态度。

另外还有W3CSchool，W3CSchool 是 W3C 中国社区的成员，网上也讲了两者的区别。总之，感觉都不是特别靠谱。

所以还是看**MDN**吧：2017年10月18日，W3C宣布加入Mozilla开发者网络合作，与Mozilla、微软、谷歌、三星一起，共同支持MDN Web 文档（MDN Web Docs）(https://developer.mozilla.org/en-US/)项目。而且也有中文的呢，虽然只是一部分。

MessageChannel (https://developer.mozilla.org/en-US/docs/Web/API/MessageChannel)

MessageChannel API允许我们创建一个新的消息通道，并通过它的两个 `MessagePort` 属性发送数据。



目前的支持度

使用：

```
var channel = new MessageChannel();
```

这样就创建了一个管道。

实例属性：

```
channel.port1
channel.port2
```

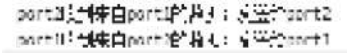
获取实例的两个端口，注意的是，两个端口都是只读的。

简单来说，MessageChannel创建了一个通信的管道，这个管道有两个口子，每个口子都可以通过postMessage发送数据，而一个口子只要绑定了onmessage回调方法，就可以接收从另一个口子传过来的数据。

一个简单的例子：

```
var channel = new MessageChannel();
var port1 = channel.port1;
var port2 = channel.port2;
port1.onmessage = function(event) {
    console.log("port1收到来自port2的数据：" + event.data);
}
port2.onmessage = function(event) {
    console.log("port2收到来自port1的数据：" + event.data);
}

port1.postMessage("发送给port2");
port2.postMessage("发送给port1");
```



运行结果截图

此特性在 Web Worker 中可以使用。

当我们使用多个web worker并想要在两个web worker之间实现通信的时候，MessageChannel就可以派上用场：


```
<script>
    var worker1 = new Worker("worker1.js");
    var worker2 = new Worker("worker2.js");
    var channel = new MessageChannel();
    worker1.postMessage({ port1: channel.port1 });
    worker2.postMessage({ port2: channel.port2 });
    worker2.onmessage = function(event) {
        console.log(event.data);
    }
</script>
```

```
  self.onmessage = function(event) {
    const port1 = event.data.port1;
    setTimeout(function() {
      port1.postMessage("this is from worker2")
    }, 2000)
  }
```

```
    self.onmessage = function(event) {
        const port2 = event.ports;
        port2.onmessage = function(event) {
            self.postMessage(event.data);
        }
   }
```

一开始我写出如上代码，结果报了这样的错误：

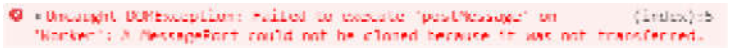


image.png

worker的数据传递是深复制的，这里报错说MessagePort不能复制。
于是我查了一下Worker.postMessage() (https://developer.mozilla.org/zh-CN/docs/Web/API/Worker/postMessage)。

发现这个方法有第二个可选的数组参数，可以将MessagePort传入，然后将控制权交给要发送到的worker。（这两句是我翻译的（如果还没有被大神改掉的话），翻译得不好别打我）

于是我把代码改为：

```
// index.html
<script>
    var w1 = new Worker("worker1.js");
    var w2 = new Worker("worker2.js");
    var ch = new MessageChannel();
    w1.postMessage("port1", [ch.port1]);
    w2.postMessage("port2", [ch.port2]);
    w2.onmessage = function(e) {
        console.log(e.data);
    }
</script>
```

```
// worker1.js
  onmessage = function(e) {
    const  port = e.ports[0];
    port.postMessage("this is from worker1")
  }
```

```
// worker2.js
    onmessage = function(e) {
        const port = e.ports[0];
        port.onmessage = function(e) {
            postMessage(e.data)
        }
    }
```

由于在worker中无法使用console.log，因此我们通过给w2绑定onmessage回调函数来验证传递是否成功。最终我们可以看到控制台中输出

> this is from worker1

传递的路径为：
w1=> ch1 => ch2 => w2

哈哈。

小礼物走一走，来简书关注我

赞赏支持