

5月3日

搞懂 Solidity 事件 Event - 如何在 DApp 中使用

很多同学对 [Solidity](#) 中的 Event 有疑问，这篇文章就来详细的看看 Solidity 中 Event 到底有什么用？

很多同学对 [Solidity](#) 中的 Event 有疑问，这篇文章就来详细的看看 Solidity 中 Event 到底有什么用？

写在前面

Solidity 是以太坊智能合约编程语言，阅读本文前，你应该对以太坊、智能合约有所了解，如果你还不了解，建议你先看[以太坊是什么](#)，另外本文在监听合约事件是对上一篇[Web3 与智能合约交互实战](#)进行补充，如果阅读了上一篇可以更好的理解本文。

什么是事件 Event

事件是[以太坊虚拟机\(EVM\)](#)日志基础设施提供的一个便利接口。当被发送事件（调用）时，会触发参数存储到交易的日志中（一种区块链上的特殊数据结构）。这些日志与合约的地址关联，并记录到区块链中。
来捋这个关系：区块链是打包一系列交易的区块组成的链条，每一个交易“收据”会包含 0 到多个日志记录，日志代表着[智能合约](#)所触发的事件。

关于 EVM 上如何处理事件的，可以参考：[理解以太坊上的事件日志](#)

在 DAPP 的应用中，如果监听了某事件，当事件发生时，会进行回调。
不过要注意：日志和事件在合约内是无法被访问的，即使是创建日志的合约。

在 Solidity 代码中，使用 `event` 关键字来定义一个事件，如：

```
event EventName(address bidder, uint amount);
```

这个用法和定义函数式一样的，[并且事件在合约中同样可以被继承](#)。触发一个事件使用 `emit`(说明，之前的版本里并不需要使用 `emit`)，如：

```
emit EventName(msg.sender, msg.value);
```

[触发事件可以在任何函数中调用](#)，如：

```
function testEvent() public {
```

```
    // 触发一个事件
```

```
emit eventName(msg.sender, msg.value);  
}
```

监听事件

通过上面的介绍，可能大家还是不清楚事件有什么作用，如果你跟过 [Web3 与智能合约交互实战](#) 这篇文章，你会发现点击"Update Info"按钮之后，虽然调用智能合约成功，但是当前的界面并没有得到更新。
使用 **事件监听**，就可以很好的解决这个问题，让看看如何实现。

修改合约，定义事件及触发事件

先回顾一下合约代码：

```
pragma solidity ^0.4.21;  
  
contract InfoContract {  
  
    string fName;  
    uint age;  
  
    function setInfo(string _fName, uint _age) public {  
        fName = _fName;  
        age = _age;  
    }  
  
    function getInfo() public constant returns (string, uint) {  
        return (fName, age);  
    }  
}
```

首先，需要定义一个事件：

```
event Instructor(  
    string name,  
    uint age  
);
```

这个事件中，会接受两个参数：name 和 age，**也就是需要跟踪的两个信息。**

然后，需要在 setInfo 函数中，触发 Instructor 事件，如：

```
function setInfo(string _fName, uint _age) public {
    fName = _fName;
    age = _age;
    emit Instructor(_fName, _age);
}
```

在 [Web3 与智能合约交互实战](#)，点击"Updata Info"按钮之后，会调用 setInfo 函数，函数时触发 Instructor 事件。

使用 Web3 监听事件，刷新 UI

现在需要使用 Web3 监听事件，刷新 UI。
先回顾下之前的使用 Web3 和智能合约交互的代码：

```
<script>
    if (typeof web3 !== 'undefined') {
        web3 = new Web3(web3.currentProvider);
    } else {
        // set the provider you want from Web3.providers
        web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
    }

    web3.eth.defaultAccount = web3.eth.accounts[0];

    var infoContract = web3.eth.contract(ABI INFO);

    var info = infoContract.at('CONTRACT ADDRESS');

    info.getInfo(function(error, result){
        if(!error)
        {
            $("#info").html(result[0]+' ('+result[1]+' years old)');
            console.log(result);
        }
        else
            console.error(error);
    });

    $("#button").click(function() {
        info.setInfo($("#name").val(), $("#age").val());
    });

</script>
```

现在可以不需要 `info.getInfo()` 来获取信息，而改用监听事件获取信息，先定义一个变量引用事件：

```
var instructorEvent = info.Instructor();
```


然后使用 `**watch()` 方法来添加一个回调函数：

```
instructorEvent.watch(function(error, result) {  
    if (!error)  
    {  
        $("#info").html(result.args.name + ' (' + result.args.age + ' years old)');  
    } else {  
        console.log(error);  
    }  
});
```

代码更新之后，可以在浏览器查看效果，这是点击"Update Info"按钮之后，会及时更新界面，如图：

Info Contract

Tiny 熊 (28 years old)



Name

Age

登录社区

事件高级用法-过滤器

有时我们会有这样的需求：获取当前所有姓名及年龄记录，或者是，要过滤出年龄 28 岁的记录，应该如何做呢？

以及另外一个常见的场景：想要获取到代币合约中所有的转账记录，也同样需要使用事件过滤器功能，这部分内容请大家订阅小专栏[区块链技术](#)阅读。

另外强烈安利两门视频课程给大家：

- [深入详解以太坊智能合约语言 Solidity](#) - Solidity 语言面面俱到
- [以太坊 DAPP 开发实战](#) - 轻轻松松学会 DAPP 开发

参考文章

<https://coursetro.com/posts/code/100/Solidity-Events-Tutorial---Using-Web3.js-to-Listen-for-Smart-Contract-Events>

<https://github.com/ethereum/wiki/wiki/JavaScript-API#contract-events>

[深入浅出区块链](#) - 打造高质量区块链技术博客，学区块链都来这里，关注[知乎](#)、[微博](#) 掌握区块链技术动态。

如果你想和我有密切的联系，欢迎加入知识星球[深入浅出区块链](#)，我会在星球为大家解答技术问题，作为星友福利，星友可加入我创建的区块链技术群，群内已经聚集了 300 多位区块链技术牛人和爱好者。

有时我们会有这样的需求：获取当前所有姓名及年龄记录，应该如何做呢？实际上事件支持过滤器，可以从所有的区块中过滤出符合要求的事件，如：

```
var instructorEvent = info.Instructor({}, {fromBlock: 0, toBlock: 'latest'});
```

或者是，要过滤出年龄 28 岁的记录，可以这样：

```
var instructorEvent = info.Instructor({ 'age': 28});
```

比如，我们要获取到代币合约中，所有的转账记录， 就可以使用：

```
var transferEvent = token.Transfer({}, {fromBlock: 0, toBlock: 'latest'})
var transferEvent.watch(function(error, result){
    // handle result.args.from result.args.to
});
```