### **在solana Anchor中定义和触发事件**

在Anchor中定义事件很简单，你只需要在合约逻辑中定义一个Rust结构体，并使用**#[event]**宏来标注它。然后，在合约的某个函数中，通过**emit!**宏来触发这个事件。

use anchor\_lang::prelude::\*;

#[program]

mod my\_anchor\_program {

use super::\*;

pub fn emit\_my\_event(ctx: Context<EmitMyEvent>, data: String) -> ProgramResult {

emit!(MyEvent {

data: data,

authority: \*ctx.accounts.authority.key,

});

Ok(())

}

}

#[derive(Accounts)]

pub struct EmitMyEvent<'info> {

#[account(signer)]

pub authority: AccountInfo<'info>,

}

#[event]

pub struct MyEvent {

pub data: String,

pub authority: Pubkey,

}

在这个例子中，我们定义了一个**MyEvent**事件，它包含了一些数据和触发事件的账户的公钥。然后在**emit\_my\_event**函数中触发这个事件。

### **监听Anchor事件**

监听由Anchor合约触发的事件通常涉及到监听合约地址的交易，然后解析这些交易的日志。你可以使用Solana的Web3.js库来完成这项工作。

以下是使用JavaScript监听Anchor事件的示例：

const anchor = require('@project-serum/anchor');

const { Connection, PublicKey } = require('@solana/web3.js');

// 设置连接和程序ID

const connection = new Connection("https://api.devnet.solana.com", "confirmed");

const programId = new PublicKey("YourProgramIdHere");

// 监听程序ID下的所有交易

const subscriptionId = connection.onLogs(programId, (logs, context) => {

for (const log of logs.logs) {

if (log.startsWith("Program log:")) {

const eventData = log.slice("Program log: ".length);

try {

const event = JSON.parse(eventData);

if (event.name === "MyEvent") {

console.log("Event detected:", event);

}

} catch (e) {

// 不是有效的JSON，可能是其他类型的日志

}

}

}

}, "confirmed");

// 确保取消订阅以避免内存泄露

// connection.removeOnLogsListener(subscriptionId);

这个脚本连接到Solana网络，并监听与指定程序ID相关的所有交易的日志。每当检测到日志时，它会尝试解析日志内容为JSON，如果日志表示一个事件（在这个例子中是**MyEvent**），则打印出事件内容。