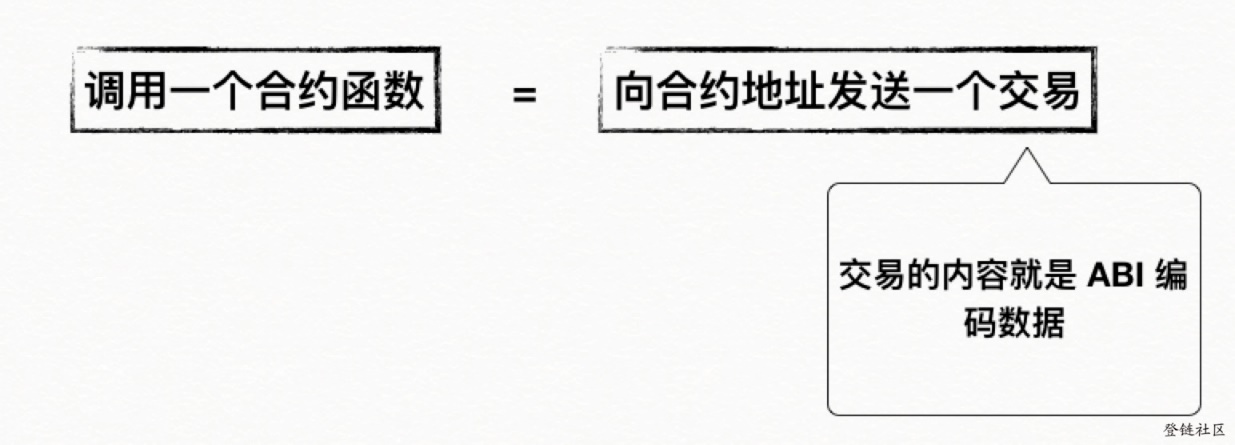
ABI 是什么

ABI 全称是 Application Binary Interface，翻译过来就是：应用程序二进制接口，简单来说就是 以太坊的调用合约时的接口说明。还不是很理解，没关系。

调用合约函数发生了什么

从外部施加给以太坊的行为都称之为向以太坊网络提交了一个交易， 调用合约函数其实是向合约地址（账户）提交了一个交易，这个交易有一个附加数据，这个附加的数据就是ABI的编码数据。  


目前，使用引用类型必须明确地提供存储该类型的数据位置，引用类型包括：

1、struct(结构体)

2、array(数组)：动态大小字节数组(bytes)、字符串(string)、动态数组(T[])

3、mapping(映射)，

**数据存储位置**

**1、Memory**

Memory：总是会重新分配内存，不会释放内存，当有重名未使用的变量时，不会利用前者内存空间，而会重新分配新的空间，memory标注的数据可读可修改。

memory(生存期存在于function(函数)内，超过作用域即失效)；

**2、calldata**

外部函数的参数必须使用calldata，但是也可用于其他变量，该位置避免了数据拷贝，并确保数据不被修改。函数也可以返回使用calldata声明的数组和结果，但是不可能赋值这些类型。

**Note1:** 尽量使用calldata作为数据位置，因为可以避免被复制，且不能修改数据。用calldata修饰array(数组)和struct(结构)也可以作为函数返回值，但无法给其分配空间。

**Note2:** 0.6.9版本之前的数据存储位置定义为：calldata用于external function(外部函数)，memory用于public function(公开函数)，memory和storage均可用于internal function(内部函数)和private function(私有函数)。当前0.8.0.版本允许memory和calldata用于所有函数类型。

**Note3:** 在版本0.5.0之前，数据位置可省略，并且根据变量的类型，函数类型等有默认数据位置，但是所有复杂类型现在必须提供明确的数据位置

**3、storage**

storage存储在EVM的storage区域，在交易执行之后，storage区域的数据会写回到合约账户的storage中

**4、stack**

**值类型的局部变量存储在栈上**

**合约中存储位置分类：**

**1、全局storage**：状态变量

**2、局部storage：**函数内声明的storage 变量，包括函数参数；

**3、memory:**  函数中的非storage、非calldata变量；

**4、calldata:**  这种数据位置是只读的。

**状态变量和局部变量：**

1，在合约中定义的变量为状态变量，在方法中定义的变量为局部变量；

memory和storage：

1，在合约中声明的变量默认是storage的；

2，函数参数中的变量默认是memory的；

函数中定义的**复杂类型**变量(如 string，array和struct）必须明确指定存储位置，memory或storage；

函数中定义的**值类型**变量默认是memory的 。值类型的局部变量（即既不是数组，也不是结构，也不是映射）存储在堆栈中