欢迎继续我们的学习。

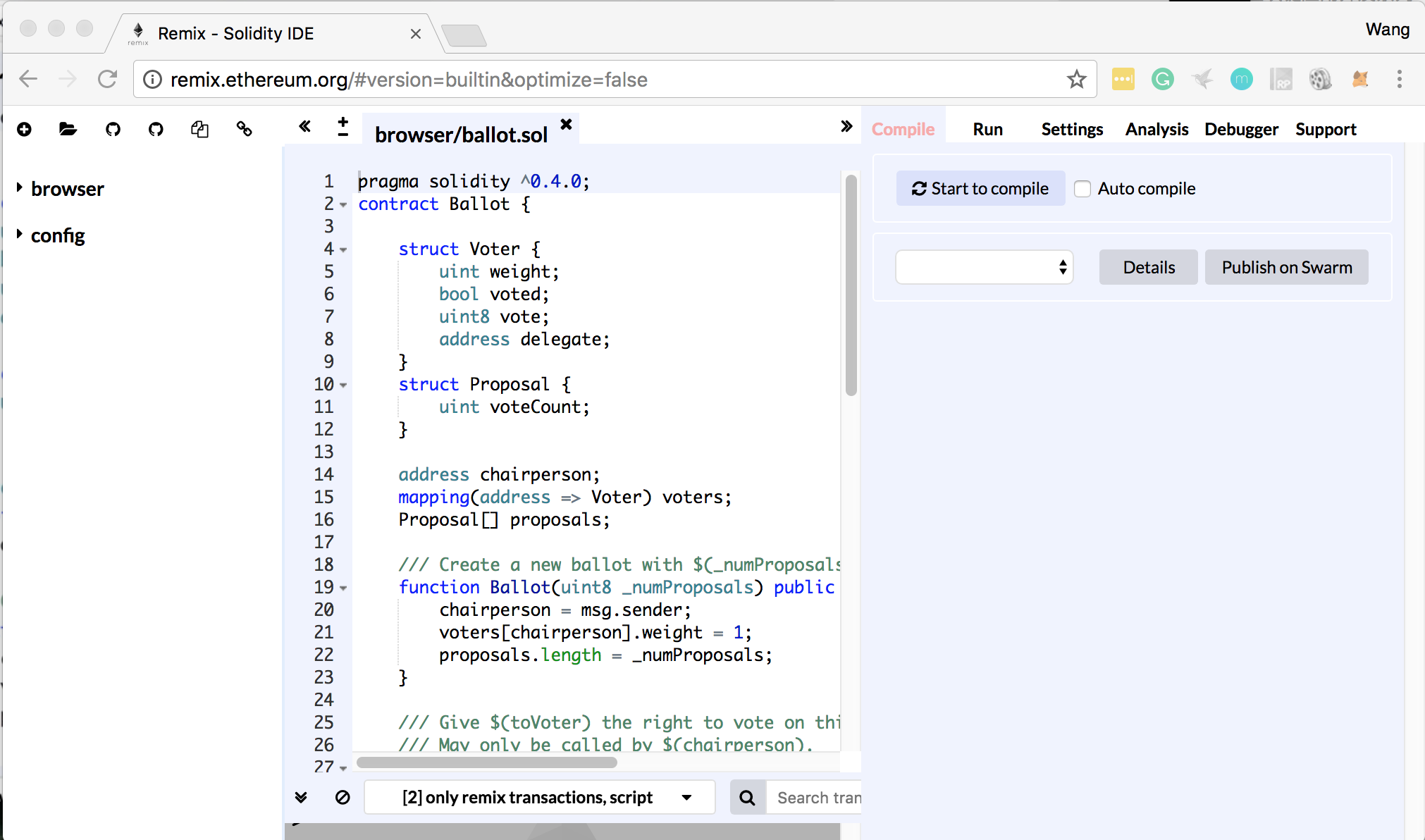
在这一课的内容中，我们将学习开发第一个真正意义上的智能合约。

为此，我们将使用一个名为Remix的在线代码编辑器。使用remix编辑器，可以轻松创建和测试基于solidity的智能合约。

当然理论上你也可以使用任何一个其它的代码编辑器。

因为remix是个在线代码编辑器，所以只需要在Chrome浏览器中输入以下网址就好：

<http://remix.ethereum.org/>

此时你会看到类似下面的界面：

可以看到，默认情况下代码编辑器中已经有一大堆代码了。

不过为了让大家对智能合约的开发有更清晰的认识，这里还是从零开始比较好。

选中已有的所有代码，然后删除。

好了，现在让我们从零开始写一个智能合约。

首先让我们写第一行代码：

pragma solidity ^0.4.17;

在这一行代码的内容中，我们指定了solidity的版本号。

接下来我们要创建自己的第一个智能合约：

contract Inbox{

}

通过这种方式，我们就创建了第一个contract。接着在里面填充具体的内容：

contract Inbox{

string public message;

function Inbox(string initialMessage) public{

message = initialMessage;

}

function setMessage(string newMessage) public{

message = newMessage;

}

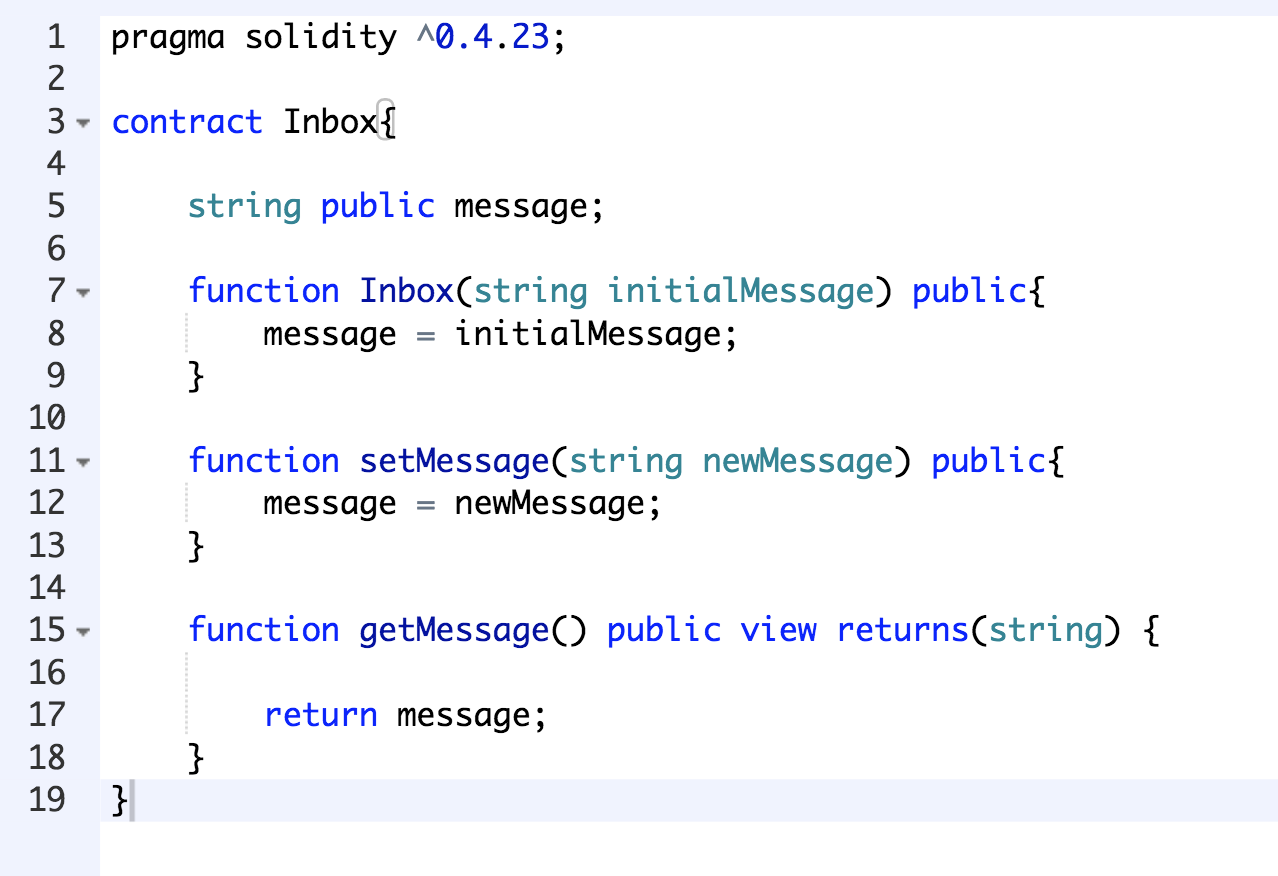
function getMessage() public view returns(string) {

return message;

}

}

接下来让我们按照代码行编号来仔细解释一下以上代码的具体作用：



1.在第一行代码中，我们指定了所使用的solidity语言的版本。

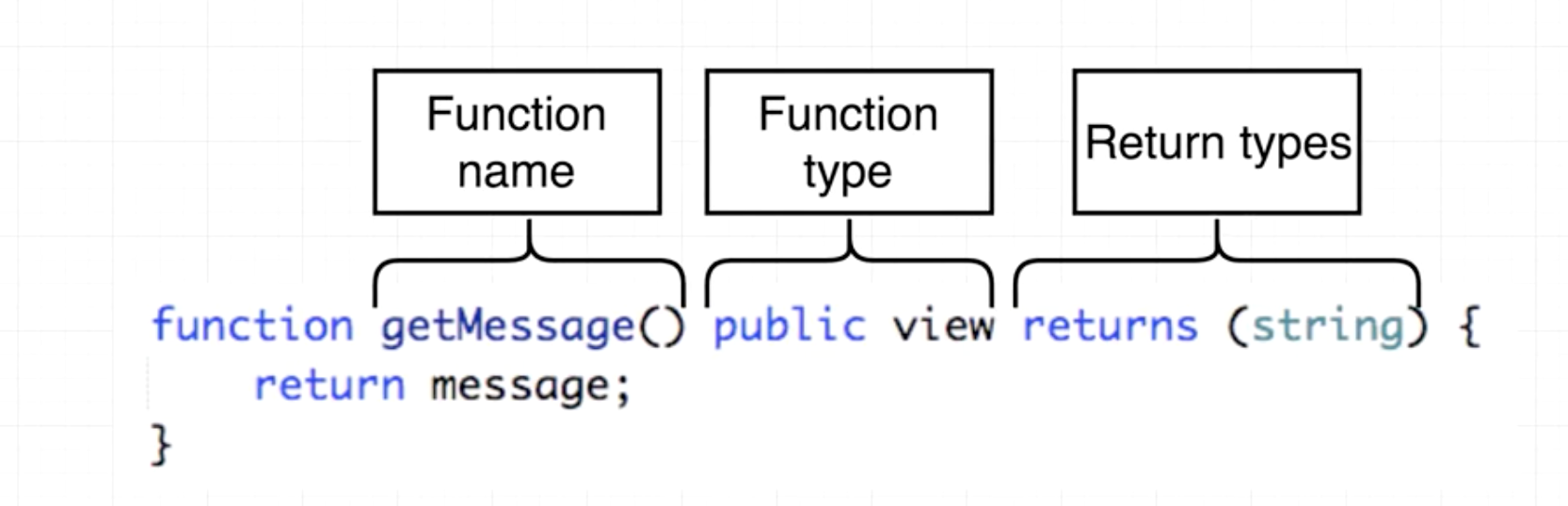
也就是说接下来的所有代码都会采用该版本的编译器(compiler)。

2.在第3行代码中，我们定义了一个新的合约（跟类的定义有点类似），其中包含了一些方法和变量。

3.在第5行代码中，我们定义了一个实例变量，其形式是：

dataType（数据类型） dataScope（范围） variableName（变量名）;

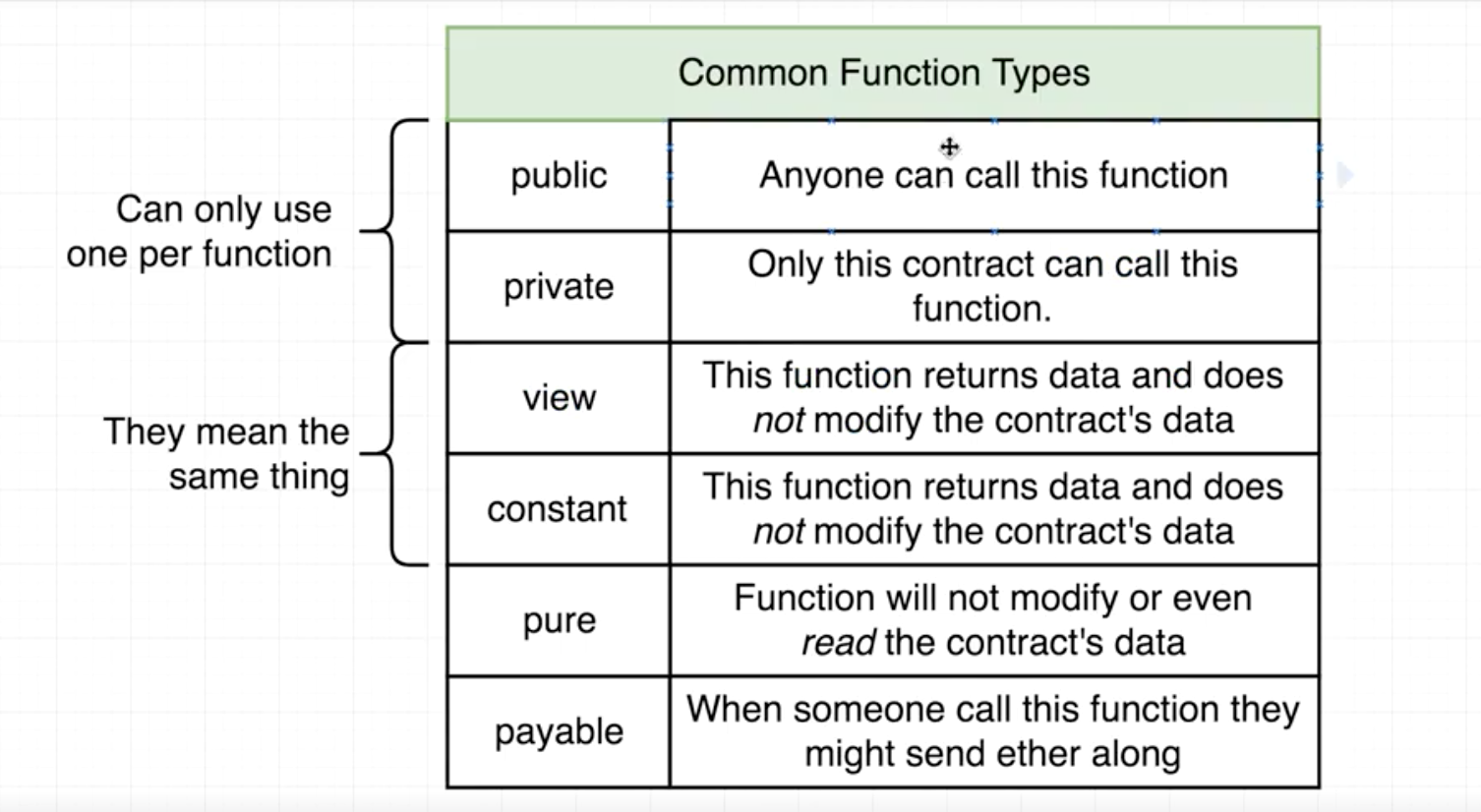
4.从第7行代码开始，我们定义了一个函数。这里的Inbox函数因为跟类的名字相同，所以属于构造函数。

5.对于getMessage函数，这里详细解释一下：

在solidity中，函数的定义方式是：

function 函数名（参数列表）+函数类型 +返回值类型（参数）

下表中列出了一些常用的函数类型：



需要特别注意的是，getMessage函数中用到了returns，而returns只能用于类型是view或者constant的函数。

因此，如果函数需要修改智能合约中的数据，则不能使用returns。

好了，到这里为止，我们应该对自己所写的智能合约有了一个基本的认识。

在下一课的内容中，我们将介绍如何使用remix来测试智能合约，以及更多的一些使用技巧。

我们下一课再见。