欢迎继续我们的学习。

在上一课的内容中，我们已经完成了自己的第一个智能合约。

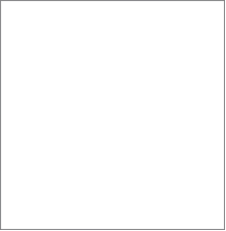
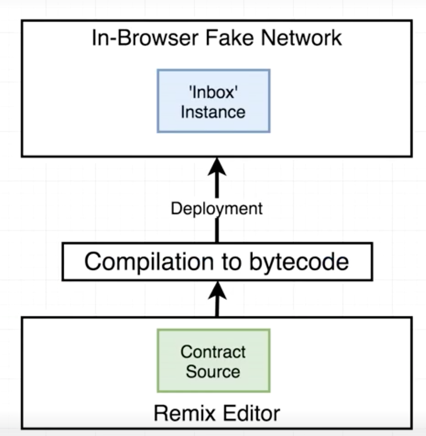
接下来要做的就是在remix中进行测试。

补充说明一下：

身在瓷器国的开发者，有时候会发现访问remix异常的缓慢，甚至是没办法访问。这个时候只有两种策略，一种是耐心等待，而如果想要比较顺畅的访问remix，有时候就得翻墙了。

关于翻墙，我想这是瓷器国开发者必备的技能了，这里不详细说明如何翻墙，无非就是VPN,SS这些工具。

好了，在进行测试之前，先让我们看一下remix测试流程背后的机制：



当前我们在最下面的一部分，也就是Remix编辑器中，里面有智能合约的源代码。

接下来remix会将智能合约的源代码编译成字节码。

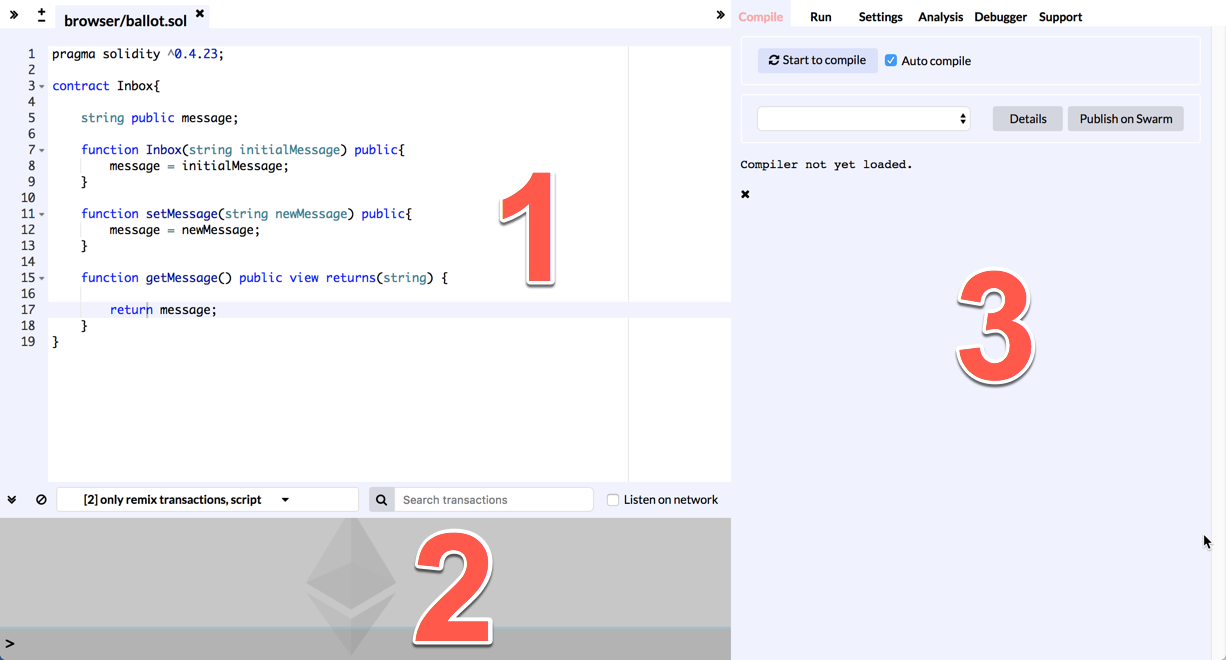
然后字节码会被部署到浏览器内的“伪”以太坊网络，并生成一个”Inbox”实例对象。

这最后一步听起来似乎有点恐怖，不过Don’t panic。

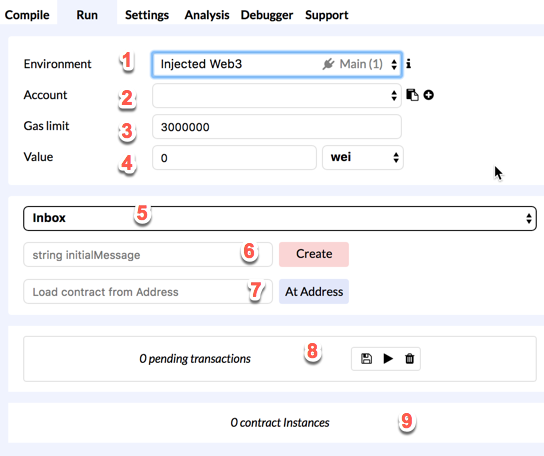
这里终于可以解释一下为什么我们会首选remix作为solidity的开发工具了。因为remix不仅仅是一个代码编辑器，还包含了一个迷你的Ethereum以太坊网络，所以它的作用不是我们熟悉的任何常规代码编辑器可以替代的~

好了，现在回到remix的页面，我们可以在页面上通过拖曳的方式调整布局，具体操作大家可以自行尝试。

调整到类似下图所示的方式：



在上图中，我们把remix编辑器的页面调整成了只显示3个区域，其中红色数字1代表的区域是代码编辑区，数字2代表的区域是输出用的console，数字3是一些重要的设置面板所在。

如果你仔细看上图中数字3的区域，发现Compiler not yet loaded，这就是没有翻墙导致加载缓慢的原因。因此，对于瓷器国的开发者，强烈建议翻墙。

在右侧的设置区切换到Run选项卡，会看到类似下面的信息：

这里简单说明一下其中一些重要选项的设置：

1.Environment

默认情况下，这里显示的是Injected Web3，需要先手动更改为下拉列表中的JavaScript VM



2.Account

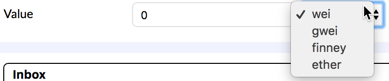
当我们选择JavaScript VM后，在Account的下拉列表中会自动列出几个带有100ETH的账户，不过可惜的是这些账户只存在于浏览器之内，不然按市价也是一笔不小的收入~



3.Gas limit

关于什么是gas,我们稍后会提到，这里先跳过。

4.Value

value指的是合约中的交易价值，在后面的单位部分有四种选择：

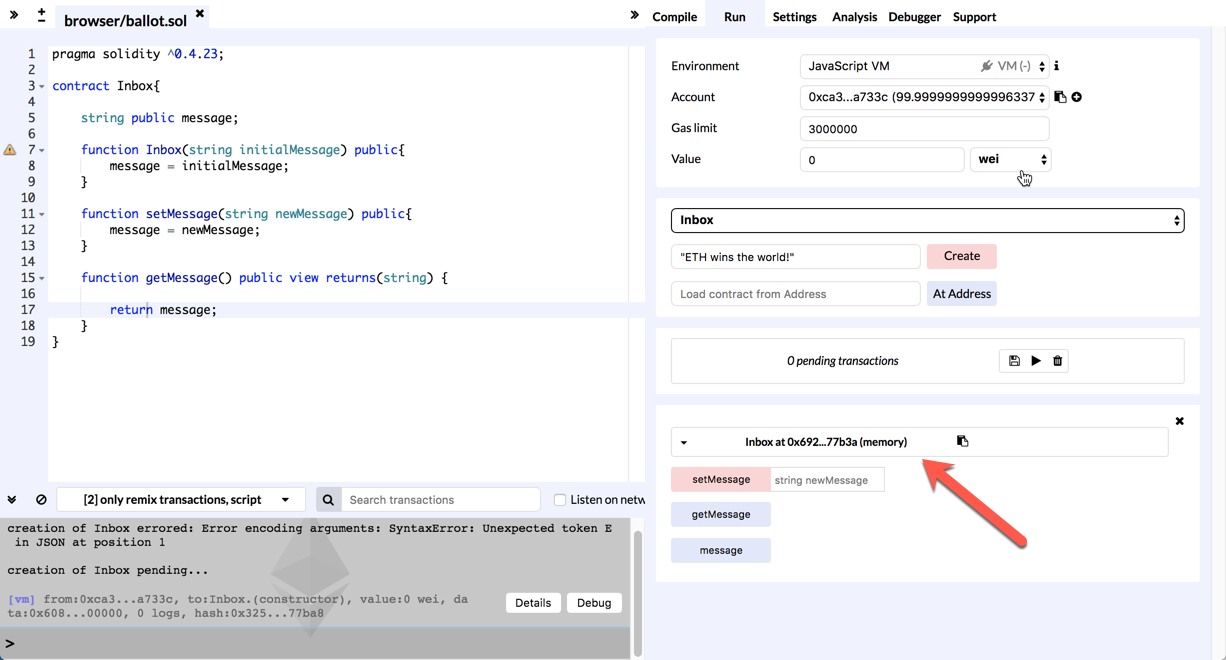
在稍后的内容中，我们会对此进行解释。

5.这里其实是我们定义的智能合约的名称。

6.左侧的空白区域用于输入构造函数的参数，点击Create就会生成一个实例对象

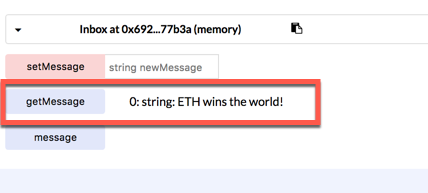
好了，在string initialMessage部分输入一段文本，比如”ETH wins the world!”需要特别注意的是，因为我们在构造函数Inbox中设置的参数是string类型，所以这里输入的信息必须加上半角的引号，否则会报错。

点击Create，会看到remix编辑器中的诸多区域都发生了变化：



在上图中，右下角用红色箭头标出的就是所生成的Inbox合约的实例对象。

可以看到，getMessage和message两个按钮是灰色的，我们可以点击getMessage按钮试试看：



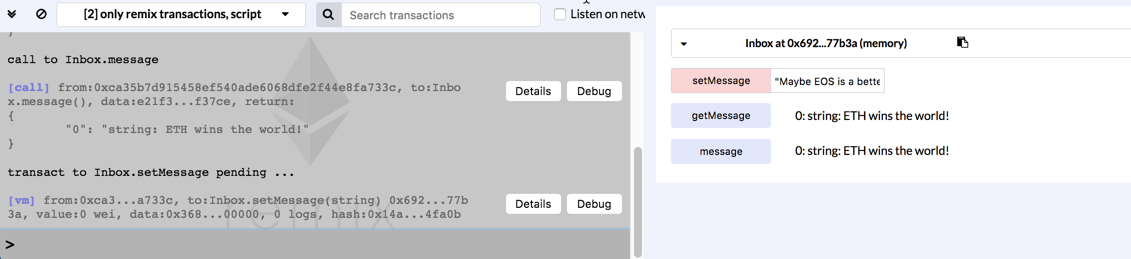
之所以在返回信息前会有一个数字编号0，是因为我们也可以设置返回多个信息，那么0就是返回的第1条信息，1就是第2条，以此类推。

此外，在0之后有一个string,说明了返回的信息类型。最后才是实际的返回信息。

点击message按钮，可以看到类似的返回信息：



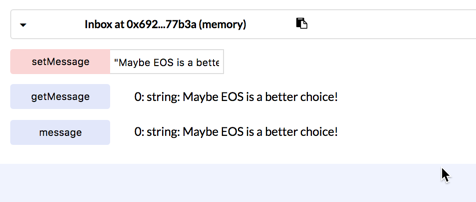
然后就是setMessage按钮，我们需要在后面的文本框里面输入一个字符串，比如”Maybe EOS is a better choice!”

然后点击setMessage按钮，看看会发生些什么：

你会看到console区中输出了新的信息，transact to Inbox.setMessage pending …

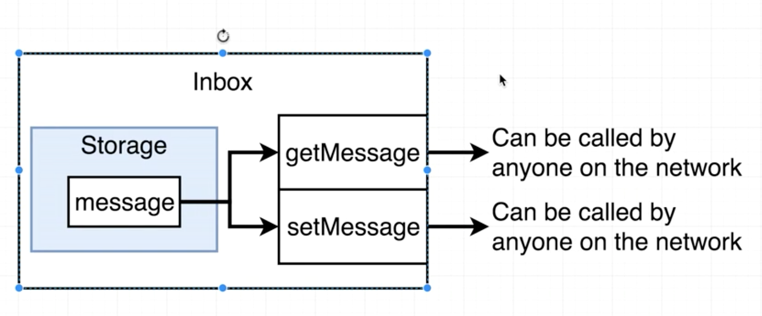
此时右下角中getMessage和message所输出的信息并不会自动更新。

不过，当我们手动点击这两个按钮后，会看到信息发生了变化：



虽然我们写的第一个智能合约炒鸡简单，但至少大家知道在remix中是怎么对合约进行测试的了。

这里让我们重温一下这个合约的机制:



在结束本课的内容之前，让我们来了解solidity语言的一个小技巧。

在Inbox的合约内容定义中，我们定义了一个变量，一个和合约名同名的构造函数，还有两个普通函数setMessage和getMessage。

当我们输入构造函数中的参数，点击Create创建了一个实例对象后，在右下角会看到除了getMessage和setMessage，还多了一个可以交互的message按钮。

而且它所输出的信息和getMessage相同。

那么，这里究竟想说明什么？

当我们在solidity的合约定义中创建了一个public类型的变量时，solidity会自动帮我们创建一个同名的函数，通过该函数可以获取该变量的值。

换句话说，getMessage函数根本就是多余的~

所以，我们完全可以从源代码中删除getMessage这个函数的相关代码。

不过如果你是个处女座，那么或许会觉得有个getMessage的名称会更好？

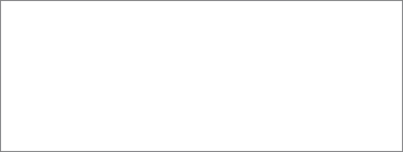
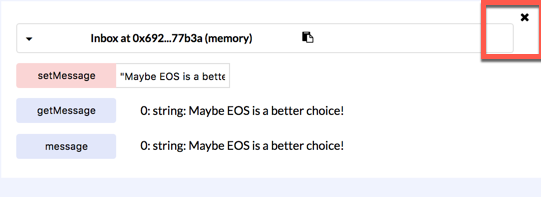
好了，现在让我们从代码编辑器中删除getMessage函数。

然后看看如何重新部署。

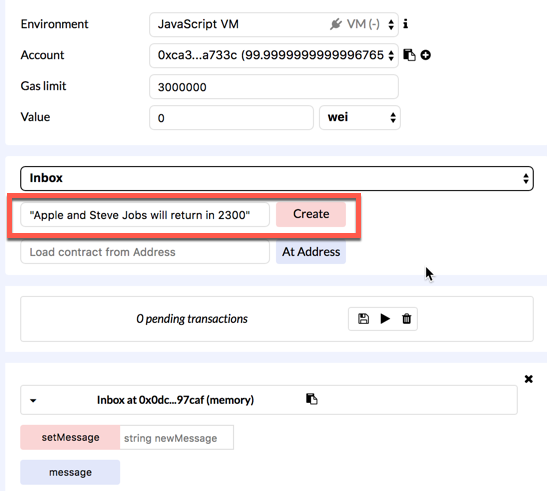
在删除代码之后，右下角的输出信息并没有发生任何变化。点击getMessage，还是可以得到所需要的信息。

这是因为仅仅更改代码区的代码，并没有生成新的实例变量。

在重新部署之前，首先删除之前的实例对象，只需要点击下图中的❌号就好：



现在string intialMessage处输入新的信息（注意要加上半角双引号），然后点击Create按钮，生成新的实例变量：



可以看到，此时已经没有getMessage按钮了。我们可以用之前的方式来测试这两个函数。

好了，到这里为止，我们已经学会了如何在remix中测试所开发的智能合约。

在下一课的内容中，我们将继续了解智能合约部署背后的机制。

我们下一课再见。

知乎专栏：

<https://zhuanlan.zhihu.com/kidscoding>

新浪博客:

<http://blog.sina.com.cn/eseedo>

Github:

<https://github.com/eseedo>

个人网站：

<http://icode.ai/>