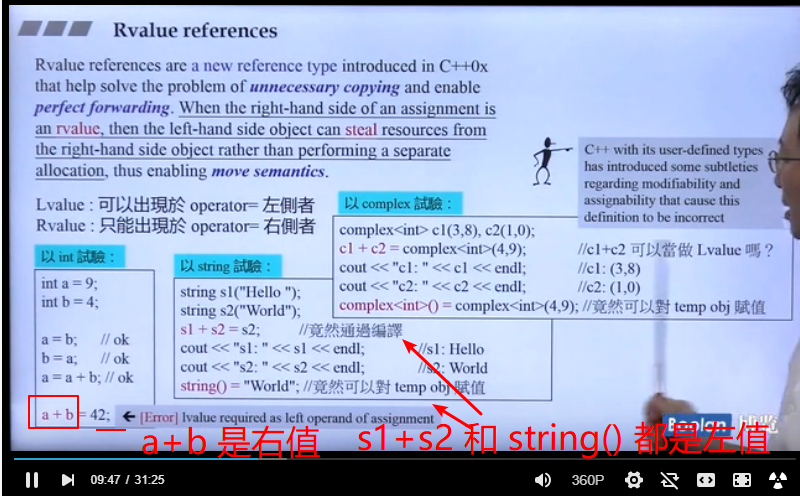
[Rvalue references and Move Semantics(移动语义)](https://www.bilibili.com/video/av24180269/?p=23)

1. 右值引用：为了解决不必要的copy，同时可以进行完美转发

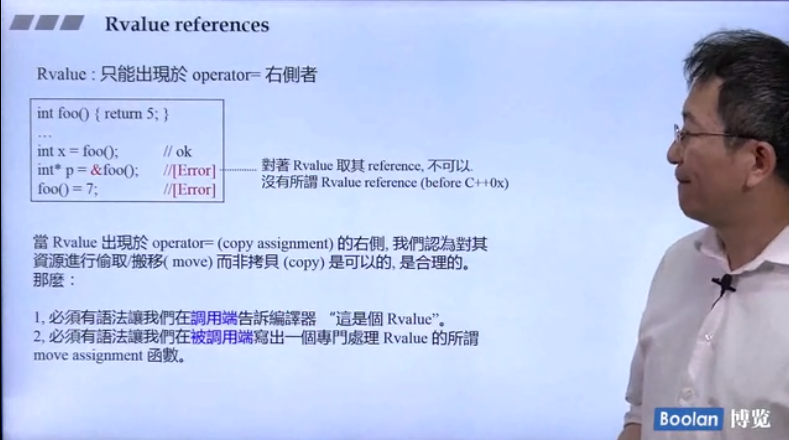
LValue：可以出现在operate= 左侧者

RValue: 只能出现在operate= 右侧者

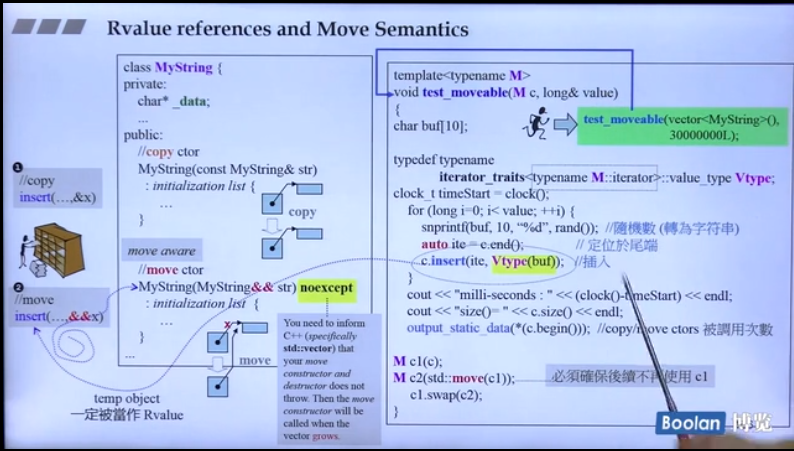


C++夹带着自己定义的类型，会引来一些对于modified 或者 赋值动作的一些让人意想不到的情况，从而造成一些定义的不正确性。所以说，虽然上图中临时对象string()/complex<int>可以在左边，但是我们还是会将临时对象设定为右值。综上所述：只能出现在operate= 右侧的是右值，临时对象是右值。

C++2.0前面，我们是无法对右值进行取地址，但是C++2.0就给了我们一个语法，用来取右值的地址，于是就有了一个右值引用：RValue References。



1. 右值引用和移动语义



MyString(const MyString& str) -> 指针的深拷贝

MyString(const MyString&& str) -> 右值引用 ，其实就是指针的浅拷贝，拷贝以后原来的指针就没用了。从中我们看出，被move以后，原来的东西就不能再用了。我们看右边的Vtype(buf)，因为Vtype(buf)是临时对象，我们接下来起来是不用再用了，所以这里用的很合理。

右边我们可以看到std::move，其实是当我明明是一个左值，但是我想当作右值用，我们就用std::move。