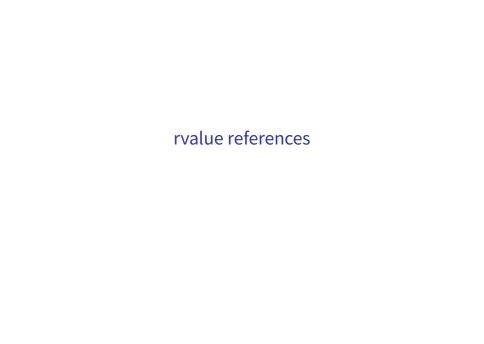
Modern C++

Rvalue references, move semantics & rule of five

Christian Paffhausen

July 11, 2019



What is a rvalue?

A lvalue is a value that could be on the left hand side of an assignment. Correspondingly, an rvalue is a value that could only be the right hand side.

```
#include <vector>
#include <string>
int main()
    std::vector<std::string> v;
    std::string s = "aaa";
    v.push back(s);  // s is a lvalue
    v.push_back("bbb"); // "bbb" is a rvalue
```

How to declarate a rvalue reference?

A lvalue reference, or simply reference, can be simply declarated by:

int &i;

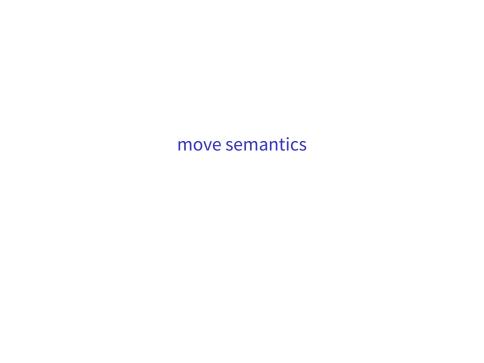
So a rvalue reference can be declarated by:

int &&i;

Pass by rvalue references

Pass by a std::string as rvalue reference to a function.

```
#include <vector>
#include <string>
// template<typename T>
// void vector::push_back(T &&value);
int main()
    std::vector<std::string> v;
    v.push_back("bbb");
```



how to move a object

std::move is used to indicate that an object s may be "moved from" from s to another object.

```
#include <vector>
#include <string>
#include <utility>
int main()
    std::vector<std::string> v;
    std::string s = "aaa";
    v.push_back(std::move(s)); // The rvalue
       reference push_back will be called
```

What happend with the moved object?

A move operation should move and leave its source in a valid state

```
#include <vector>
#include <string>
#include <utility>
#include <iostream>
int main()
    std::vector<std::string> v;
    std::string s = "aaa";
    v.push_back(std::move(s));
    std::cout << s << "\n";
```



Rule of five

If a class requires:

- a user-defined destructor
- a user-defined copy constructor
- a user-defined move constructor
- a user-defined copy assignment operator
- a user-defined move assignment operator

it almost certainly requires all three.

Signatures of the rule of five

The rule of five operations have the following declarations:

```
class Foo
 public:
    Foo(const Foo &);
    Foo(Foo &&);
    Foo & operator = (const Foo &);
    Foo &operator=(Foo &&);
    virtual ~Foo();
};
```

Rule of zero

- ▶ If you can avoid defining default operations, do.
- ► If you define or =delete any default operation, define or =delete them all.