decltype

Michael Park



What is it?

• A language feature in C++ to retrieve the type of an expression

Examples

```
int a = 42;  // decltype(a) == int

const int& b = a; // decltype(b) == const int&

int f();  // decltype(f()) == int

int x[10]; // decltype(x[0]) == int&
```

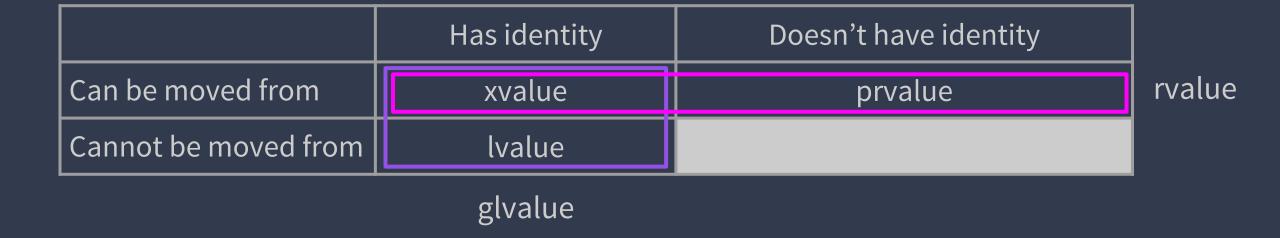
Few More Examples

The Rules

If the type of expr is T, the result of decltype(expr) is:

- T if the value category of expr is a prvalue
- T& if the value category of expr is an Ivalue
- T&& if the value category of expr is an xvalue

Value Categories



© 2015 Mesosphere, Inc. All Rights Reserved.

Example

```
int x = 42;
```

What does decltype(x) yield with these rules?

- The type of x is int.
- The value category of x is an Ivalue (has an identity, cannot be moved from)

```
→ decltype(x) == int&
```

But the result of decltype(x) is actually int...

An Overriding Rule

If expr is an **unparenthesized** id-expr or **unparenthesized** class member access, Then decltype yields the **declared type** of expr.

 \rightarrow decltype(x) == **declared** type of x == int

Parenthesis

If expr is an **unparenthesized** id-expr or **unparenthesized** class member access, Then decltype yields the **declared type** of expr.

 \rightarrow decltype((x)) == int&

Conflated Features

- Inspects
 - The declared type of an entity, or
 - The type and value category of an expression.
- Potentially could have been better off with:
 - decltype
 - o exprtype

#define exprtype(expr) decltype((expr))

decltype in function return types

```
template <typename T, typename U>
auto add(T t, U u) -> decltype(t + u) { return t + u; }
```

Duplicate expressions in decltype and return

decltype(auto) -- C++14

- auto is typically a type placeholder.
- The auto in decltype(auto) is a expression placeholder.

```
template <typename T, typename U>
decltype(auto) add(T t, U u) { return t + u; }
```

No more duplicate expressions!

Expression SFINAE

Expression SFINAE

```
template <typename T, typename U>
decltype(auto) add(T t, U u) { return t + u; }

struct S {} s;
add(s + s); // error: invalid operands to binary expression
```

Conclusion

- Understand the fundamental ambiguous question of decltype.
 - declared type vs. the type and value category of the expression.
- Expression SFINAE is new in C++11.
- decltype(auto) is new in C++14.
- Naively transforming a trailing return type with decltype in it with decltype(auto) will not always work.
- Consider using thinking in terms of decltype and exprtype.