Expression → has two property → Type → can't be reference
↳ Value Category (VC)

int& r = x
↳ type is int
not int&

Lvalue → has memory location.
Variables don't have value category, expressions have (VC)

int* p = &x
↳ type is int*

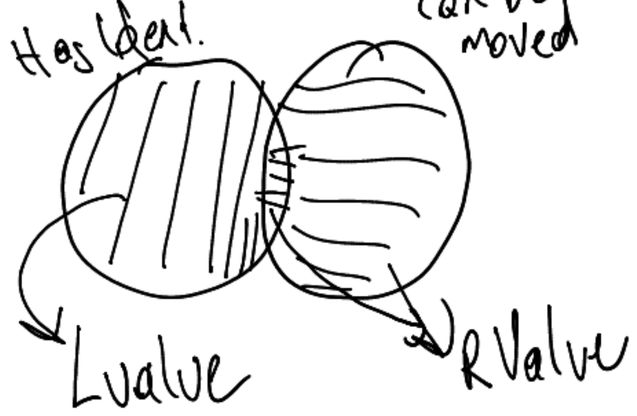
int&& r = x+y

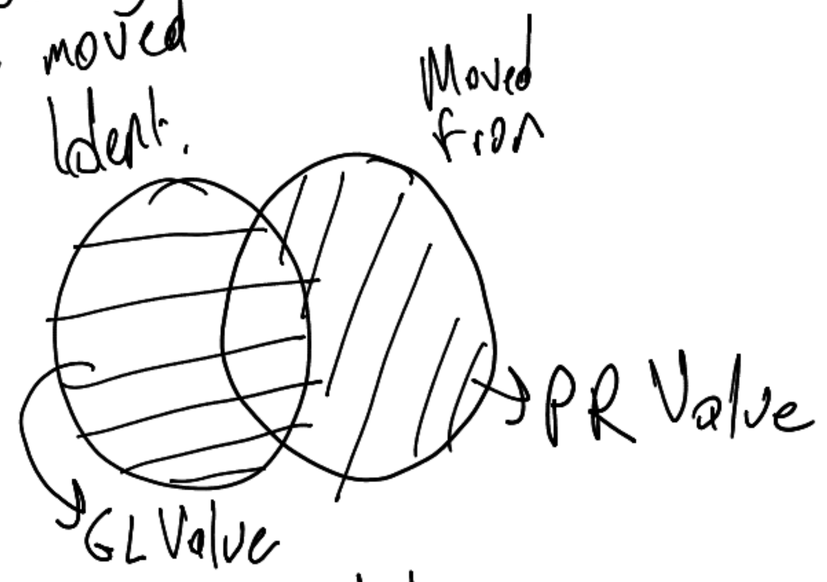
r → What is value cat? → LValue

Primary VC: PR Value (Pure RValue): No identity, but can be moved
LValue: Has identity, can't be moved from.
Expression can only be one of ✗ Value: Has identity, can be moved from.

Mixed (Combined) VC: GLValue: LValue or XValue → Has Identity
RValue: PRValue or XValue → Can be moved

1. Having identity
2. Can be stolen (Movability)



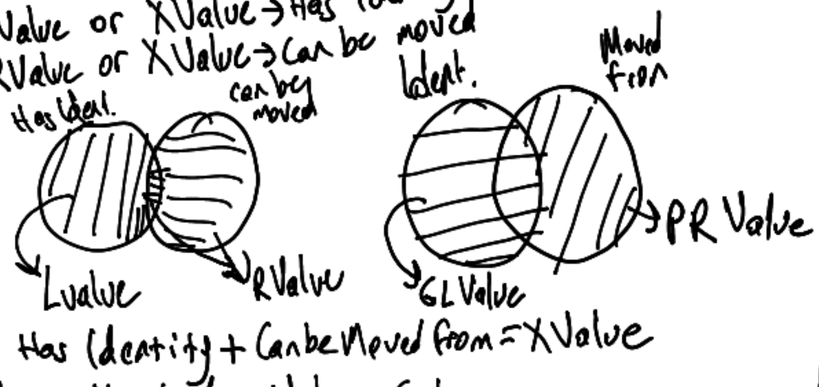


Has Identity + Can be Moved from = XValue

Primary VC: PRValue (Pure RValue): No identity, but can be moved
Expression can ONLY be one of: LValue: Has identity, can't be moved from.
XValue: Has identity, can be moved from.

Mixed (Combined) VC: GLValue: LValue or XValue → Has Identity
RValue: PRValue or XValue → can be moved

1. Having identity
2. Can be stolen (Movability)



Has Identity + Can be Moved from = XValue

if name is used
decltype(x) or expression → its diffrent
decltype(expression)
decltype((name)) → calls expression version

decltype(expression) version deducts
PRValue ==) T
LValue → T&
XValue → T&&

For examples check code Week1 - Value Category.

	T&	const T&	T&&	const T&&
LValue	1	2	X	X
const LValu	X	1	X	X
RValue	X	3	1	2
const RValue	X	2	X	1

1. foo(S&)
2. foo(const S&)
3. foo(S&&)
4. foo(const S&&)