But consider that having such an overload set is typically contraindicated. Without the rvalue reference overload, invoking `h` on a *temporary* would simply fail to compile, thereby avoiding a runtime defect. With the rvalue-reference overload present, the code will in fact compile, but now any output written to that *temporary* will silently disappear along with that temporary.

Although seldom needed, Table 2[9] provides the relative priority in which all four potential pass-by-reference members of an overload set would be selected.

Table 2: Overload resolution priorities

Value Category		`g(C&)`	`g(const C&)`	`g(C&&)`	`g(const C&&)`
non**`const`** *lvalue*	`c`	1	2	N/A	N/A
`const` *lvalue*	`cc`	N/A	1	N/A	N/A
non**`const`** *rvalue*	`C()`	N/A	3	1	2
`const` *rvalue*	`fc()`	N/A	2	N/A	1

Note that the equivalent function to `fc` for a built-in type, **`const int`** `fi()`, would return a non**`const`** *rvalue*, as fundamental types returned by **`const`** value are treated as if they had been returned by non**`const`** value. The only way to have a function that returns a **`const`** *rvalue* of primitive type `T` is to have it return **`const`** `T&&`, such as **`const int`**`&& fi2()`.

***Rvalue* references in expressions** Recall from *Lvalues in C++11/14* on page 717 that any named variable, including an rvalue reference, is an *lvalue*:

```
#include <utility>  // std::move

struct S { /*...*/ };

void test1()
{
    S   s1;                  // local variable of type S
    S&& s2 = s1;             // Error, s1 is an lvalue.
    S&& s3 = std::move(s1);  // OK, std::move(s1) is an xvalue.
    S&& s4 = s3;             // Error, s3 is an lvalue.
    S&& s5 = std::move(s3);  // OK, std::move(s3) is an xvalue.
}
```

That a named rvalue reference itself was deliberately categorized as an *lvalue* helps to ensure that such a reference is not accidentally consumed as an *xvalue* until its current value is no longer needed elsewhere:

[9] Though Table 2 was derived and verified independently, a strikingly similar table can be found in **josuttis20b**, section 8.3.1, "Overload Resolution with Rvalue References," pp. 133–134, Table 8.1, p. 134.