Reference-Returning Functions, Reading

You learned about "void" and "value-returning" functions previous to the course. The operator square bracket setter that we just learned about in the previous reading is neither of these. It's yet another type of C++ function. Now we have these:

- void functions, like `void fun( );`
- value-returning functions, like `int fun( );`
- reference-returning functions, like `int& fun( );`

Value-returning functions return exactly what their name suggests -- a *value*. You might have a statement in the function like `return result;`, where `result` is a variable, but before the return happens, the value stored in the variable is extracted and that's what's returned. So you can do `cout << fun( )`, but you could never do `fun( ) = something;` because you cannot put a value on the *left side* of an assignment statement -- no `100 = something;` allowed!

Reference-returning functions cannot return local variables. They cannot return values, like `return 0;`. They can only return global variables, such as *class data members*. Calls to reference-returning functions are *aliases* for the variable they return. So if `dummy` is a data member, and the function `fun` returns it, writing `a.fun( ) = something;` in the main program is like saying `a.dummy = something;`. The object `a` is used in that example because we're talking about returning data members here, and there needs to be an object to put this discussion in the proper context.