The Implicit Parameter

Consider the implementation of the `hours()` member function of the `Time` class shown here:

```
int Time::hours() const
{
    return m_hours;
}

int main()
{
    Time t;                                 // a Time object
    cout << t.hours() << endl;              // value of t::m_hours
}
```

The `hours()` member function **does not** contain a local variable named `m_hours`. But, the function still compiles and runs correctly. Why?

In a **member function**, you may **directly access and manipulate** any or all of the class's **data members** by referring to them by name. You don't need to indicate that `m_hours` is a data member, nor do you specify **which** `Time` object it belongs to.

C++ assumes that all data members **are the data members of the receiver object**, and so the line `return m_hours` means *“return the value of the `m_hours` data member of the object on which this function was invoked.”* In such a case, **the receiver object is known as the implicit parameter**, passed to every member function.