Class Functions Declared with `static` Prefix

To **avoid the implicit argument**, **prefix** the function declaration **with `static`**.

Such functions are called **class functions**.

Their **function signatures**
- are **exactly the same as in C**
- (except that in **C**, the static qualifier restricts access to the file;
- class function access is handled differently in **C++**).

Use Class:: Prefix to Invoke a Class Function

How does one invoke a class function?

As with class (static) variables,
- **everything in a class**
- is **prefixed with the name of the class**.
- For example, `MyClass::`

These **prefixes reduce naming conflicts** and avoid the need for having programmers do it by hand (as with **C**).

Compiler Infers Which Class to Use for Member Functions

For member functions, a **compiler infers the class from the instance type**. For example,

```
MyClass m;

m.someFunc ();
```

The **compiler knows** that
- **`m` has type `MyClass`**, so
- it **looks first at `MyClass::someFunc`**.

Use Class:: Prefix to Invoke a Class Function

The **compiler does not guess** which class holds a class function. If `MyClass` contains …

```
static void doSomething (int32_t q);
```

use the namespace operator (::) to invoke the class function:

```
MyClass::doSomething (42);
```

C++ supports hierarchical namespaces with ::, but for now you need merely understand the use with classes.

