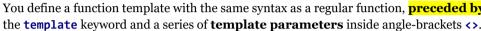
Function Templates

I hope the code on the preceding page bothers you as much as it bothers me. It doesn't take much to notice that the body of each function is identical. Why can't we define one version of the function that takes any kind of argument?

Surprise! We can!

C++ functions with **generic types** are called **function templates**. (In Java these are called generic functions, but **template** is used more often in C++).



```
template <typename T>
std::string to_string(const T& n)

{
    std::ostringstream out;
    out << n;
    return out.str();
}</pre>
```

The template parameters are separated by commas, and use **generic template type**names: names preceded by either the class or typename keyword followed by an identifier. Both keywords are synonyms in template declarations

When using separate compilation:

- Function templates ar **placed inside the header file**, unlike normal functions which are placed inside the implementation file.
- You must **fully qualify all library types**, such as **string** and **ostringstream** in the example shown here, since you are not allowed to add **using namespace std;** to a header file.



This course content is offered under a <u>CC Attribution Non-Commercial</u> license. Content in this course can be considered under this license unless otherwise noted.

d by			
s ⟨> .			
v <mark>e</mark>			
ns,			
om in d; to			