Why use static_cast<int>(x) instead of (int)x?

The main reason is that classic C casts make no distinction between what we call **static_cast<>()**, **reinterpret_cast<>()**, **const_cast<>()**, and **dynamic_cast<>()**. These four things are **completely different**.

A static_cast<>() is usually safe. There is a valid conversion in the language, or an appropriate constructor that makes it possible. **The only time it's a bit risky is when you cast down to an inherited class; you must make sure that the object is a actually the descendant that you claim it is**, by means external to the language (like a flag in the object). A dynamic_cast<>() is safe as long as the result is checked (pointer) or a possible exception is taken into account (reference).

A reinterpret_cast<>() (or a const_cast<>()) on the other hand is always dangerous. You tell the compiler: "trust me: I know this doesn't look like a foo (this looks as if it isn't mutable), but it is".

The first problem is that it's almost impossible to tell which one will occur in a C-style cast without looking at large and disperse pieces of code and knowing all the rules.

Let's assume these:

```
class CMyClass : public CMyBase {...};
class CMyOtherStuff {...} ;

CMyBase  *pSomething; // filled somewhere
```

Now, these two are compiled the same way:

```
CMyClass *pMyObject;
pMyObject = static_cast<CMyClass*>(pSomething); // Safe; as long as we checked

pMyObject = (CMyClass*)(pSomething); // Same as static_cast<>
                                     // Safe; as long as we checked
                                     // but harder to read
```

However, let's see this almost identical code:

```
CMyOtherStuff *pOther;
pOther = static_cast<CMyOtherStuff*>(pSomething); // Compiler error: Can't convert

pOther = (CMyOtherStuff*)(pSomething);            // No compiler error.
                                                  // Same as reinterpret_cast<>
                                                  // and it's wrong!!!
```

As you can see, there is no easy way to distinguish between the two situations without knowing a lot about all the classes involved.

The second problem is that the C-style casts are too hard to locate. In complex expressions it can be very hard to see C-style casts. It is virtually impossible to write an automated tool that needs to locate C-style casts (for example a search tool) without a full blown C++ compiler front-end. On the other hand, it's easy to search for "static_cast<" or "reinterpret_cast<".

```
pOther = reinterpret_cast<CMyOtherStuff*>(pSomething);
      // No compiler error.
      // but the presence of a reinterpret_cast<> is
      // like a Siren with Red Flashing Lights in your code.
      // The mere typing of it should cause you to feel VERY uncomfortable.
```

That means that, not only are C-style casts more dangerous, but it's a lot harder to find them all to make sure that they are correct.