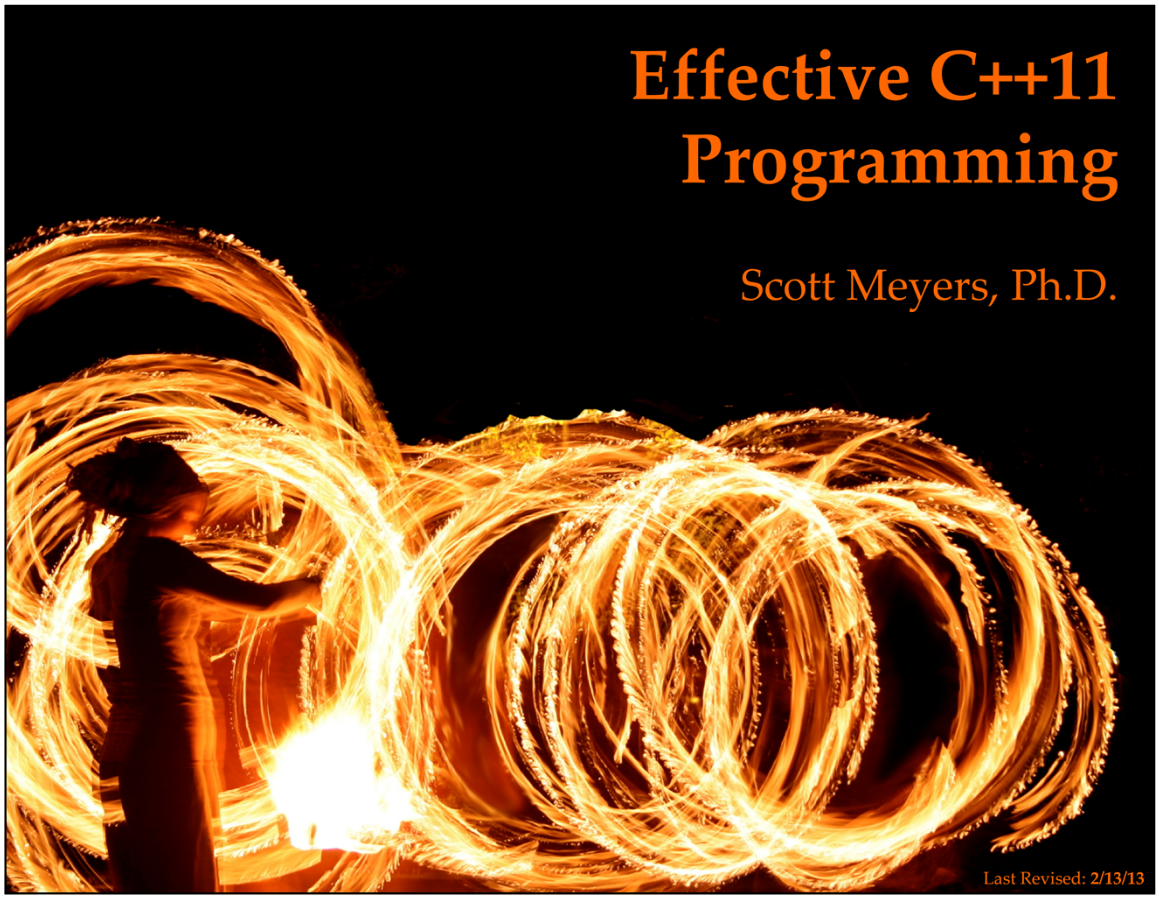


Image: iStockPhoto #6648802 .

Declare Overriding Functions `override`

A derived member function *overrides* a base class version if:

- Base version is `virtual`.
- Derived version matches base in:
  - Name
  - Parameter types
  - "Adornments," i.e., `const`, `volatile`, `&`, and `&&`

```
class Base {
public:
  virtual
    void f() const;
};
```

```
class Derived: public Base {
public:
  virtual              // overriding
    void f() const;    // function
};
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 2

Per 10.3/2, what I call "adornments" are officially called *cv-qualifications* and *ref-qualifiers*.

Overriding Functions

Overrides may omit `virtual`:

```
class Base {
public:
  virtual void f() const;
};

class Derived1: public Base {
public:
  virtual void f() const;       // overrides Base::f
};

class Derived2: public Base {
public:
  void f() const;               // also overrides Base::f
};
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Overriding Errors

Override-related errors easy to make:

```
class Base {
public:
  virtual void f() const;
};

class Derived: public Base {
public:
  virtual void f();            // missing const;
};                             // doesn't override;
                               // declares new virtual function
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 4

None of VC10, VC11, and gcc 4.7 issue a warning for this code.

Overriding Errors

```
class Base {
public:
  virtual void f(long) const;
};
class Derived: public Base {
public:
  virtual void f(int) const;  // wrong param type;
};                            // doesn't override;
                              // declares new virtual function
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 5

None of VC10, VC11, and gcc 4.7 issue a warning for this code.

Overriding Errors

```
class Base {
public:
  void f() const;             // missing virtual
};
class Derived: public Base {
public:
  void f() const;             // doesn't override;
};                            // declares new nonvirtual
                              // function
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 6

None of VC10, VC11, and gcc 4.7 issue a warning for this code.

override

`override`-declared functions *must* override inherited versions:

```
class Base {
public:
  virtual void f() const;
};

class Derived1: public Base {
public:
  void f() const override;       // fine
};

class Derived2: public Base {
public:
  virtual void f() override;     // error! not an override
};                               // (missing const)
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 7

`override` goes after cv- and ref qualifications.

Declaring an `override` function `virtual` is redundant. `override` functions are always `virtual`.

override

Prevents all earlier errors from compiling:

- Attempt to override non-`virtual` function:

```
class Base {
public:
  void f() const;
};

class Derived: public Base {
public:
  void f() const override;      // error!
};
```

override

- Mismatched parameter types in base and derived classes:

```
class Base {
public:
  virtual void f(long) const;
};

class Derived: public Base {
public:
  virtual void f(int) const override;     // error!
};
```

override

- Mismatched `const`/`volatile` declarations in base and derived classes:

```
class Base {
public:
  virtual void f(int) const;
};

class Derived: public Base {
public:
  virtual void f(int) override;      // error!
};
```

override

Other benefits:

- Helps identify virtuals in derived classes.

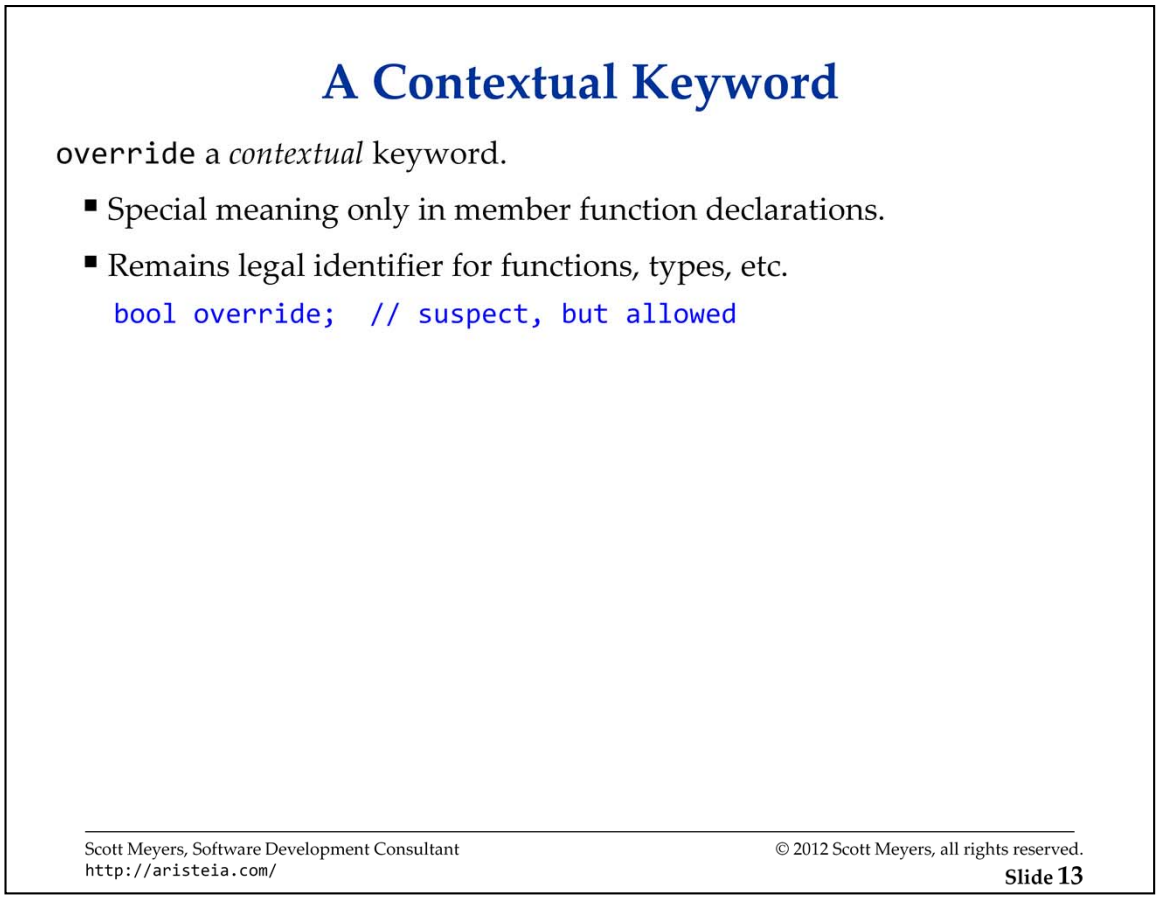
```
class Derived: public Base {
public:
  void f1();              // virtual?
  void f2() override;     // virtual!
  ...
};
```

- Identifies affected derived functions if base signature changes.
  - ➡ Reduces maintenance-induced errors.

override

Covariant return types remain legal:

```
class Base {
public:
  virtual Base& me();
};
class Derived: public Base {
public:
  virtual Derived& me() override;       // fine
};
```



The only reason I can think of for having code like this in your system is that it's legacy, and you don't want to change it.

Fun with "override"

Legal code you don't want to write:

```
class override {};

class Base {
public:
  virtual ::override override(::override);   // override takes
                                             // and returns an
};                                           // ::override

class Derived: public Base {
public:
  ::override override(::override) override;  // an override
};                                           // of above :-)
```

Scott Meyers, Software Development Consultant
http://aristeia.com/

Slide 14

This code compiles cleanly with both gcc 4.7 and VC11.

Guideline

Declare overriding functions `override`.