# C++11 braced initialization made the impossible possible (and how to fix it so it stays impossible)

devblogs.microsoft.com/oldnewthing/20210719-00

July 19, 2021

Raymond Chen

Suppose you have a private nested type. You might use this because you need your constructor to be public in order to work with some framework,[1] but you don't want people to do their own `make_unique`; you want them to go through your factory.

```
class Package
{
  struct private_constructor { };

public:
  // Do not call constructor directly. Use CreatePackage instead.
  Package(int id, private_constructor);

  static Package CreatePackage(int id, int flavor)
  {
    Package package(id, private_constructor());
    ... do other stuff that gets the package ready ...
    return package;
  }
};

void bad_boy()
{
  // This doesn't work. Wrong number of parameters.
  Package package(3);

  // This doesn't work. private_constructor is a private type.
  Package package(3, Package::private_constructor());
}
```

But C++11 introduced braced initialization, and the bad boy can use that to construct the type without naming it.

```
void bad_boy_got_through()
{
  // Bad boy uses empty braces to sneak past the gate!
  Package package(3, {});
}
```

To prevent this, you need to give your private type an explicit constructor so it cannot be used implicitly.

```
class Package
{
  struct private_constructor
    { explicit private_constructor() = default; };

public:
  // Do not call constructor directly. Use CreatePackage instead.
  Package(int id, private_constructor);

  ...
};
```

With this change, the bad boy has been foiled.

```
void bad_boy_foiled()
{
  // Can't sneak in with empty braces.
  Package package(3, {});
}
```

From Visual C++:

```
error C2664: 'Package::Package(Package &&)': cannot convert argument 2 from
'initializer list' to 'Package::private_constructor'
```

From clang:

```
error: converting to 'Package::private_constructor' from initializer list would use
explicit constructor 'constexpr Package::private_constructor::private_constructor()'
```

And the explicit constructor is inaccessible.

```
void bad_boy_foiled()
{
  // Can't use explicit constructor.
  Package package(3, Package::private_constructor{});
}

// error: cannot access private struct
```

[1] For example, `std::make_unique` requires that the object have a public constructor.

Raymond Chen

## Follow