make the inner names available with this using-directive:

using namespace elements::fire;

Also, you can use using-directives and using-declarations inside namespaces:

namespace myth
{
    using Jill::fetch;
    using namespace elements;
    using std::cout;
    using std::cin;
}

Suppose you want to access Jill::fetch. Because Jill::fetch is now part of the myth namespace, where it can be called fetch, you can access it this way:

std::cin >> myth::fetch;

Of course, because it also is part of the Jill namespace, you still can call it Jill::fetch:

std::cout << Jill::fetch;  // display value read into myth::fetch

Or you can do this, providing no local variables conflict:

using namespace myth;
cin >> fetch;  // really std::cin and Jill::fetch

Now consider applying a using-directive to the myth namespace. The using-directive is transitive. We say an operation $op$ is transitive if $A \ op \ B$ and $B \ op \ C$ implies $A \ op \ C$. For example, the $>$ operator is transitive. (That is, $A$ bigger than $B$ and $B$ bigger than $C$ implies $A$ bigger than $C$.) In this context, the upshot is that the statement

using namespace myth;

results in the elements namespace being added via a using-directive also, so it's the same as the following: