EXAMPLE 6-14. Chaining method calls client-side

```
$fields = array('uid', 'name', 'books', 'pic', 'current_location');
$friend_uids = $facebook->api_client->friends_get();
$user_infos = users_getInfo($friend_uids, $fields);
```

This results in more calls to the data server (here, two calls), greater latency, and more points of possible failure. Instead, for viewing user number 8055 (yours truly), we render this in FQL syntax as the single call in Example 6-15.

EXAMPLE 6-15. Chaining method calls server-side with FQL

```
$fql = "SELECT uid, name, books, pic, current_location FROM profile
        WHERE uid IN (SELECT uid2 from friends where uid1 = 8055)";
$user_infos = $facebook->api_client->fql_query($fql);
```

We conceptually treat the data referred to by `users_getInfo` as a *table* with a number of selectable *fields*, based on an index (`uid`). If augmented appropriately, this new grammar enables a number of new data access capabilities:

- Range queries (for example, event times)
- Nested queries (`SELECT fields_1 FROM table WHERE field IN (SELECT fields_2 FROM .... )`)
- Result limits and ordering

Architecture of FQL

Developers invoke FQL through the API call `fql_query`. The crux of the problem involves unifying the named "objects" and "properties" of the external API with named "tables" and "fields" in FQL. We still inherit the flow of the standard API: fetching the data through our internal methods, applying the rules normally associated with API calls on this method, and transforming the output according to the Thrift system from the earlier section "Data: Creating an XML Web Service." For every data-reading API method there exists a corresponding "table" in FQL that abstracts the data behind that query. For instance, the API method `users_getInfo`, which makes the `name`, `pic`, `books`, and `current_location` fields available for a given user ID is represented in FQL as the `user` table with those corresponding fields. The external output of `fql_query` actually conforms to the output of the standard API as well (if the XSD is modified to allow for omitted fields in an object), so a call to `fql_query` on the `user` table returns output identical to an appropriate call to `users_getInfo`. In fact, often calls such as `user_getInfo` are implemented at Facebook on the server side as FQL calls!

> **NOTE**
>
> **At the time of this writing, FQL supports only `SELECT` rather than `INSERT`, `UPDATE`, `REPLACE`, `DELETE`, or others, so only *read* methods can be implemented using FQL. Most Facebook Platform API methods operating on this type of data are read-only at this point anyway.**