The following code fragment captures a small subset of jQuery behavior, just enough to use jQuery in a simple way.

```
interface JQuery {
    text(content: string);
}

interface JQueryStatic {
    get(url: string, callback: (data: string) => any);
    (query: string): JQuery;
}

declare var $: JQueryStatic;

$.get("http://mysite.org/divContent",
      function (data: string) {
          $("div").text(data);
      }
);
```

The 'JQueryStatic' interface references another interface: 'JQuery'. This interface represents a collection of one or more DOM elements. The jQuery library can perform many operations on such a collection, but in this example the jQuery client only needs to know that it can set the text content of each jQuery element in a collection by passing a string to the 'text' method. The 'JQueryStatic' interface also contains a method, 'get', that performs an Ajax get operation on the provided URL and arranges to invoke the provided callback upon receipt of a response.

Finally, the 'JQueryStatic' interface contains a bare function signature

```
(query: string): JQuery;
```

The bare signature indicates that instances of the interface are callable. This example illustrates that TypeScript function types are just special cases of TypeScript object types. Specifically, function types are object types that contain only a call signature, but no properties. For this reason we can write any function type as an object type literal. The following example uses both forms to describe the same type.

```
var f: { (): string; };
var sameType: () => string = f;     // Ok
var nope: () => number = sameType;  // Error: type mismatch
```

We mentioned above that the '$' function behaves differently depending on the type of its parameter. So far, our jQuery typing only captures one of these behaviors: return an object of type 'JQuery' when passed a string. To specify multiple behaviors, TypeScript supports *overloading* of function signatures in object types. For example, we can add an additional call signature to the 'JQueryStatic' interface.

```
(ready: () => any): any;
```