```
// Type definitions for [~THE LIBRARY NAME~] [~OPTIONAL VERSION NUMBER~]
// Project: [~THE PROJECT NAME~]
// Definitions by: [~YOUR NAME~] <[~A URL FOR YOU~]>

/*~ This is the module template file for function modules.
 *~ You should rename it to index.d.ts and place it in a folder with the same n
 *~ For example, if you were writing a file for "super-greeter", this
 *~ file should be 'super-greeter/index.d.ts'
 */

/*~ Note that ES6 modules cannot directly export callable functions.
 *~ This file should be imported using the CommonJS-style:
 *~   import x = require('someLibrary');
 *~
 *~ Refer to the documentation to understand common
 *~ workarounds for this limitation of ES6 modules.
 */

/*~ If this module is a UMD module that exposes a global variable 'myFuncLib' w
 *~ loaded outside a module loader environment, declare that global here.
 *~ Otherwise, delete this declaration.
 */
export as namespace myFuncLib;

/*~ This declaration specifies that the function
 *~ is the exported object from the file
 */
export = MyFunction;

/*~ This example shows how to have multiple overloads for your function */
declare function MyFunction(name: string): MyFunction.NamedReturnType;
declare function MyFunction(length: number): MyFunction.LengthReturnType;

/*~ If you want to expose types from your module as well, you can
 *~ place them in this block. Often you will want to describe the
 *~ shape of the return type of the function; that type should
 *~ be declared in here, as this example shows.
 */
declare namespace MyFunction {
    export interface LengthReturnType {
        width: number;
        height: number;
    }
    export interface NamedReturnType {
        firstName: string;
        lastName: string;
    }

    /*~ If the module also has properties, declare them here. For example,
     *~ this declaration says that this code is legal:
     *~   import f = require('myFuncLibrary');
     *~   console.log(f.defaultName);
     */
    export const defaultName: string;
    export let defaultLength: number;
}
```