# 6 Functions

TypeScript extends JavaScript functions to include type parameters, parameter and return type annotations, overloads, default parameter values, and rest parameters.

## 6.1 Function Declarations

Function declarations consist of an optional set of function overloads followed by an actual function implementation.

*FunctionDeclaration:* *( Modified )*
    *FunctionOverloads*$_{opt}$ *FunctionImplementation*

*FunctionOverloads:*
    *FunctionOverload*
    *FunctionOverloads* *FunctionOverload*

*FunctionOverload:*
    `function` *Identifier* *CallSignature* `;`

*FunctionImplementation:*
    `function` *Identifier* *CallSignature* `{` *FunctionBody* `}`

A function declaration introduces a function with the given name in the containing declaration space. Function overloads, if present, must specify the same name as the function implementation. If a function declaration includes overloads, the overloads determine the call signatures of the type given to the function object and the function implementation signature must be assignable to that type. Otherwise, the function implementation itself determines the call signature. Function overloads have no other effect on a function declaration.

## 6.2 Function Overloads

Function overloads allow a more accurate specification of the patterns of invocation supported by a function than is possible with a single signature. The compile-time processing of a call to an overloaded function chooses the best candidate overload for the particular arguments and the return type of that overload becomes the result type the function call expression. Thus, using overloads it is possible to statically describe the manner in which a function's return type varies based on its arguments. Overload resolution in function calls is described further in section 4.12.

Function overloads are purely a compile-time construct. They have no impact on the emitted JavaScript and thus no run-time cost.

The parameter list of a function overload cannot specify default values for parameters. In other words, an overload may use only the `?` form when specifying optional parameters.