The *PropertyName* of a property signature must be unique within its containing type. If the property name is followed by a question mark, the property is optional. Otherwise, the property is required.

If a property signature omits a *TypeAnnotation*, the Any type is assumed.

3.7.2 Call Signatures

A call signature defines the type parameters, parameter list, and return type associated with applying a call operation (section 4.12) to an instance of the containing type. A type may ***overload*** call operations by defining multiple different call signatures.

CallSignature:
TypeParameters$_{opt}$ (*ParameterList*$_{opt}$) *TypeAnnotation*$_{opt}$

A call signature that includes *TypeParameters* (section 3.4.1) is called a ***generic call signature***. Conversely, a call signature with no *TypeParameters* is called a non-generic call signature.

As well as being members of object type literals, call signatures occur in method signatures (section 3.7.5), function expressions (section 4.9), and function declarations (section 6.1).

An object type containing call signatures is said to be a ***function type***.

It is an error for a type to declare multiple call signatures that are considered identical (section 3.8.2) or differ only in their return types.

3.7.2.1 Type Parameters

Type parameters (section 3.4.1) in call signatures provide a mechanism for expressing the relationships of parameter and return types in call operations. For example, a signature might introduce a type parameter and use it as both a parameter type and a return type, in effect describing a function that returns a value of the same type as its argument.

The scope of a type parameter extends over the entire call signature in which the type parameter is introduced. Thus, type parameters may be referenced in type parameter constraints, parameter types, and return type annotations in their associated call signature.

Type arguments (section 3.4.2) for call signature type parameters may be explicitly specified in a call operation or may, when possible, be inferred (section 4.12.2) from the types of the regular arguments in the call. An ***instantiation*** of a generic call signature for a particular set of type arguments is the call signature formed by replacing each type parameter with its corresponding type argument.

Some examples of call signatures with type parameters:

`<T>(x: T): T`	A function taking an argument of any type, returning a value of that same type.
`<T>(x: T, y: T): T[]`	A function taking two values of the same type, returning an array of that type.