4.11 The new Operator

A new operation has one of the following forms:

```
new ConstructExpr

new ConstructExpr ( Args )
```

where *ConstructExpr* is an expression and *Args* is an argument list. The first form is equivalent to supplying an empty argument list. *ConstructExpr* must be of type Any or of an object type with one or more construct or call signatures. The operation is processed as follows at compile-time:

- If *ConstructExpr* is of type Any, *Args* can be any argument list and the result of the operation is of type Any.
- If *ConstructExpr*'s apparent type (section 3.8.1) is an object type with one or more construct signatures, the expression is processed in the same manner as a function call, but using the construct signatures as the initial set of candidate signatures for overload resolution. The result type of the function call becomes the result type of the operation.
- If *ConstructExpr*'s apparent type is an object type with no construct signatures but one or more call signatures, the expression is processed as a function call. A compile-time error occurs if the result of the function call is not Void. The type of the result of the operation is Any.

4.12 Function Calls

Function calls are extended from JavaScript to optionally include type arguments.

Arguments: (Modified)
TypeArguments$_{opt}$ (*ArgumentList*$_{opt}$)

A function call takes one of the forms

```
FuncExpr ( Args )

FuncExpr < TypeArgs > ( Args )
```

where *FuncExpr* is an expression of a function type or of type Any, *TypeArgs* is a type argument list (section 3.4.2), and *Args* is an argument list.

If *FuncExpr* is of type Any, or of an object type that has no call signatures but is a subtype of the Function interface, the call is an ***untyped function call***. In an untyped function call no *TypeArgs* are permitted, *Args* can be any argument list, no contextual types are provided for the argument expressions, and the result is always of type Any.

If *FuncExpr*'s apparent type (section 3.8.1) is a function type, the call is a ***typed function call***. TypeScript employs ***overload resolution*** in typed function calls in order to support functions with multiple call signatures. Furthermore, TypeScript may perform ***type argument inference*** to automatically determine type arguments in generic function calls.