The descriptions of function declarations provided in section 6.1 apply to function expressions as well, except that function expressions do not support overloading.

4.9.1 Standard Function Expressions

Standard function expressions are function expressions written with the `function` keyword. The type of `this` in a standard function expression is the Any type.

Standard function expressions are transformed to JavaScript in the same manner as function declarations (see section 6.5).

4.9.2 Arrow Function Expressions

TypeScript supports ***arrow function expressions***, a new feature planned for ECMAScript 6. Arrow function expressions are a compact form of function expressions that omit the `function` keyword and have lexical scoping of `this`.

An arrow function expression of the form

ArrowFormalParameters `=>` *AssignmentExpression*

is exactly equivalent to

ArrowFormalParameters `=> { return` *AssignmentExpression* `; }`

Furthermore, arrow function expressions of the forms

Identifier `=>` *Block*
Identifier `=>` *AssignmentExpression*

are exactly equivalent to

`(` *Identifier* `) =>` *Block*
`(` *Identifier* `) =>` *AssignmentExpression*

Thus, the following examples are all equivalent:

```
(x) => { return Math.sin(x); }
(x) => Math.sin(x)
x => { return Math.sin(x); }
x => Math.sin(x)
```

A function expression using the `function` keyword introduces a new dynamically bound `this`, whereas an arrow function expression preserves the `this` of its enclosing context. Arrow function expressions are particularly useful for writing callbacks, which otherwise often have an undefined or unexpected `this`.

In the example