- In variable and member declarations with a type annotation and an initializer, the initializer expression is contextually typed by the type of the variable or property.
- In return statements, if the containing function includes a return type annotation, return expressions are contextually typed by that return type. Otherwise, if the containing function is contextually typed by a type *T*, return expressions are contextually typed by *T*'s return type.
- In typed function calls, argument expressions are contextually typed by their parameter types.
- In type assertions, the expression is contextually typed by the indicated type.
- In || operator expressions without a contextual type, the right hand expression is contextually typed by the type of the left hand expression.
- In assignment expressions, the right hand expression is contextually typed by the type of the left hand expression.
- In contextually typed object literals, property assignments are contextually typed by their property types.
- In contextually typed array literals, element expressions are contextually typed by the array element type.
- In contextually typed || operator expressions, the operands are contextually typed as well.
- In contextually typed conditional operator expressions, the operands are contextually typed as well.

Contextual typing of an expression *e* by a type *T* proceeds as follows:

- If *e* is an *ObjectLiteral* and *T* is an object type, *e* is processed with the contextual type *T*, as described in section 4.5.
- If *e* is an *ArrayLiteral* and *T* is an object type with a numeric index signature, *e* is processed with the contextual type *T*, as described in section 4.6.
- If *e* is a *FunctionExpression* or *ArrowFunctionExpression* with no type parameters and no parameter or return type annotations, *T* is a function type with exactly one call signature and *T*'s call signature is non-generic, then any inferences made for type parameters referenced by the parameters of *T*'s call signature are fixed (section 4.12.2) and *e* is processed with the contextual type *T*, as described in section 4.9.3.
- If *e* is a || operator expression and *T* is an object type, *e* is processed with the contextual type *T*, as described in section 4.15.7.
- If *e* is a conditional operator expression and *T* is an object type, *e* is processed with the contextual type *T*, as described in section 4.16.
- Otherwise, *e* is processed without a contextual type.

The rules above require expressions be of the exact syntactic forms specified in order to be processed as contextually typed constructs. For example, given the declaration of the variable 'f' above, the assignment

```
f = s => s.toLowerCase();
```

causes the function expression to be contextually typed, inferring the String primitive type for 's'. However, simply enclosing the construct in parentheses