Expression statement

```
EXPR_STMT : EXPR_WITHOUT_BLOCK ";"
          | EXPR_WITH_BLOCK ";"?
```

An *expression statement* evaluates an expression and ignores its result. Its purpose is to trigger side effects of expression evaluation only.

When an expression that ends with a block (i.e. '}') is used in a context where a statement is permitted, the trailing semicolon can be omitted without changing the semantic meaning. This is different than omitting the semicolon after a non-block expression. This can include if, match, for, etc. This can cause an ambiguity between it being parsed as a standalone statement and as a part of another expression; in this case, it is parsed as a statement.

```
v.pop();          // Ignore the element returned from pop
if v.is_empty() {
    v.push(5);
} else {
    v.remove(0);
}                 // Semicolon can be omitted.
[1];              // Separate expression statement, not an indexing expression
```

When the trailing semicolon is omitted, the result return type of the expression must be the unit type.

```
// bad: the block's type is i32, not ()
// Error: expected `()` because of default return type
// if true {
//   1
// }

// good: the block's type is i32
if true {
  1
} else {
  2
};
```