Block expression

```
BLOCK_EXPR : { STMT* }
```

A block expression, or block, is a control flow expression. As a control flow expression, a block sequentially executes its component statements and then its optional tail expression.

The syntax for a block is {, then any number of statements, then an optional tail expression, and finally a }.

Statements are usually required to be followed by a semicolon, with one exception:

Expression statements usually require a following semicolon except if its outer expression is a flow control expression. Furthermore, extra semicolons between statements are allowed, but these semicolons do not affect semantics.

The type of a block is the type of the final operand if exists. Otherwise, if the last statement is a never type emitting statement (e.g. return, break) the block's type is the never type, otherwise it is the unit type.

```
// Evaluated to unit-type.
let _: () = {
    fn_call();
};

// Evaluated to i32 type.
let five: i32 = {
    fn_call();
    5
};

// Block evaluates to never-type, which is coerced to explicitly set i32.
let never: i32 = {
    return 6;
};
```