Jvav 24 Release

Released July 2024

Jvav 24 is the first major Jvav Release, supports a few syntaxes and features. The versions of Jvav are named by year, e.g. a Jvav version released in 2024 should be called Jvav 24.

This is an early release of Jvav, many if the details are have yet to be finalized, so this is just an introduction to the syntax

1. Compilation unit

In general, a Jvav source file corresponds to a *compilation unit*, and a *compilation unit* contains several members.

Members are:

- Function declaration *(since Jvav 24)*
- Global statement *(since Jvav 24)*

Global statement are any valid statement *(since Jvav 24)*

2. Scope

2.1. General

In a Jvav source file, the outermost scope are called *global scope*. When we declare a function, the scope within the function is called *local scope*.

A *local scope* can be created by surrounding statements with curly bracket syntax.

[Example

```
Complation unit #1:

...

// global scope
{ // local scope
  ...
}
```

– *end example*]

2.2. Block scope

3. Expressions

3.1. Preamble

3.2. Primary expressions

3.3. Compound expressions

4. Statements

Statements are executed in sequence except where noted elsewhere.

Statements are starts with a keyword, otherwise it is a expression statement. *(since Jvav 24)*

4.1. Block statement

A *block statement* (also known as a compound) groups a sequence of statements into a single statement.

compound-statement:

{ statement-sequence$_{\text{optional}}$ *}*

statement-sequence:

statement

statement-sequence statement

A block statement defines a block scope (Section 2.2).

4.2. Selection statements

4.2.1. If statement

If statement executes statements conditionally, if the condition yields *true* the first sub-statement is executed. If the *else* part is present and the condition yields *false*, the second sub-statement is executed.

if (*condition*)$_{\text{optional}}$ { *statement-true* } (1)

if (*condition*)$_{\text{optional}}$ { *statement-true* } *else* { *statement-false* } (2)

1. *If* statement without an *else* branch
2. *If* statement with an *else* branch

condition	-	a *expression* which will yield a value of type *bool*
statement-true	-	the *statement* to be executed if the *condition* yields *true*
statement-false	-	the *statement* to be executed if the *condition* yields *false*

The parentheses of the wrapping condition can be empty, e.g.

if condition {

statement-true

}

4.3. Iteration statements

4.3.1. General

4.3.2. While statement

4.3.3. Do-while statement

4.3.4. For statement

4.4. Jump statements

4.4.1. General

4.4.2. Break statement

4.4.3. Continue statement

4.4.4. Return statement

5. Declarations

5.1. Preamble

A declaration is a statement (Section 4)

5.2. Variable declaration

A variable declaration is a statement that introduces and optionally initialize one identifiers.

let variable-name: $type_{optional}$ = *initializer* (1)

var variable-name: $type_{optional}$ = *initializer* (1)

variable-name	-	the name of the variable, any valid identifier
type	-	possibly empty, the type of the variable
initializer	-	the initial value of the variable, any valid expression

1. Declare a variable with the type and the initializer, the value of the variable is mutable.
2. Declare a constant with the type and the initializer, which the value cannot be changed after declaration.

If the *type clause* is empty, then the type of the variable is deduced from the initializer.