



# Sciter Script

**home** <https://sciter.com>  
**discussion** <https://sciter.com/forums/>  
**github** <https://github.com/c-smile/sciter-sdk>  
**wrappers** <https://github.com/sciter-sdk>

## Control

```
if (...)
if (...) ... else ...

for[:label]( ... ;... ; ... ) ...
for[:label](... in ...) ...

while[:label](...) ...

do[:label] ... while(...);

break [label];
continue [label];

switch(...) {
  case const : ... break;
  like template : ... break;
  instanceof class : ... break;
  in collection : ... break;
  default: ... break;
}

with(object) {...}

try {...} catch(e) {...}
try {...} catch(e) {...} finally {...}
try {...} finally {...}

await promise-expr;

yield expr;

return;
return expr;
return (expr1,expr2,...);
```

## Declarations

```
var name [= expr];
const name = expr;

var (nm1,nm2,...) = multi-ret-expr;
const (nm1,nm2,...) = multi-ret-expr;
```

## Types and literals

literal	type
42, 'A' <i>//char code</i>	Integer
42.0	Float
"string"	String
#symbol	Symbol
/ab+c/	RegExp
[0,1,2]	Array
{a:0, b:1, c:2}	Object
[name: 0,1,2]	Tuple
12.5in	Length
45deg	Angle
200ms	Duration
color(0xFF,0,0)	Color
new Date(2009,7,15)	Date
Bytes.fromBase64("...")	Bytes

## Modules and classes

```
include "rel-path";
include string-expr;

namespace name [:parentns] {
  var name;
  const name;
  function name(...) {...}
  event name ... { ... }
  namespace subns {...}
  class subcls {...}
}

class name [:parentcls] {
  var name[= expr];
  this var name [= expr];
  const name = expr;
  function name(...) {...}
  property name(v){get{...} set{...}}
  event name ... { ... }
  namespace subns { ... }
  class subcls {...}
}
```

## Expressions

```
expr + expr
expr - expr
expr * expr
expr / expr
int-expr % int-expr
expr += expr
expr -= expr
expr *= expr
expr /= expr
expr %= expr
++expr expr++
--expr expr--
expr == expr
expr === expr
expr != expr
expr !== expr
expr < expr expr <= expr
expr > expr expr >= expr
expr && expr
expr || expr
!expr
int-expr & int-expr
int-expr &= int-expr
int-expr | int-expr
int-expr |= int-expr
int-expr ^ int-expr
expr << expr
expr <= expr
expr <<< expr
expr >> expr
expr >= expr
expr >>> expr
```

## String Expressions

```
"a,b,c,d" ~/ " " // "a"
"a,b,c,d" ~% " " // "b,c,d"
"a,b,c,d" /~ " " // "a,b,c"
"a,b,c,d" %~ " " // "d"
```

## Auxiliary Expressions

```
typeof expr
new classname
expr[expr]
expr like expr
expr !like expr
expr instanceof class
expr !instanceof class
expr in collection
expr !in collection
```

## Function calls

```
func()
func(p1,p2,...)

Stringizier calls:
$func(...text...)
$func(...{expr}...{expr})

Object passing calls:
func{ a:1, b:2, ... }

Method calls:
expr.name(p1,p2,...)
expr.$name(...text...)
expr.name{ a:1, b:2, ... }
```

## Function declarations

```
function name() {...}
function name(p1,p2,...) {...}

Addon methods
function class.name() {...}
function obj.name(p1,p2,...) {...}

Optional parameters
function name(p1=1,p2=2) {...}
function name(p1,rest...) {...}
```

## Lambda functions

```
"This" lambdas
: p1,p2 : statement
: p1,p2 { statements... }

"That" lambdas
| p1,p2 | statement
| p1,p2 { statements... }
```

## Async functions

```
async function (p1,p2) {
  statements...
  var r = await promise;
  statements...
}
```

## Generator functions

```
function* (p1,p2) {
  statements...
  yield expr;
  statements...
}
```

## Event handlers

```
event name { ... }
event name $(selector) { ... }
event name $(selector) (evt) { ... }
event name $(selector) (evt,that) { ... }
```

## Macro constants

```
__FILE__ // full path of the file
__FOLDER__ // path of the file's folder
__LINE__ // current line number
__TRACE__ // stack trace
```

## Debug

```
assert expr;
assert expr : expr1 expr2 ...;

debug [channel] : expr1 expr2 ...;
debug [channel] ( ... {expr1} ... );

debug break;
debug namespace;
debug stacktrace;
```