Statements

Method Summary

void	break() Break statement exits a loop.
void	const() Constant declaration.
void	continue() Continue statement, jumps to next iteration of the loop.
void	do while() do while loop
void	for() for loop
void	for each in() foreach loop
void	if() If statement
void	if else() If/Else statement.
void	label() Provides a statement with an identifier that you can refer to using a break or continue statement.
void	switch() Switch statement.
void	try catch() try/catch statement
void	try catch finally() try/catch/finally statement
void	var() Variable declaration
void	while() while loop

Method Details

break

void **break** ()

Break statement exits a loop.

Returns

void

Sample

```
break
```

const

void **const** ()

Constant declaration.

Returns

void

Sample

```
const #;
```

continue

void **continue** ()

Continue statement, jumps to next iteration of the loop.

Returns

void
Sample

```
continue
```

do while

void **do while** ()
do while loop
Returns

void
Sample

```
do
{
}
while ( # )
```

for

void **for** ()
for loop
Returns

void
Sample

```
for ( var i = 0 ; i < # ; i++ )
{
}
```

for each in

void **for each in** ()
foreach loop
Returns

void
Sample

```
for ( var item in obj )
{
}
```

if

void **if** ()
If statement
Returns

void
Sample

```
if ( # )
{
}
```

if else

void **if else** ()

If/Else statement.

Returns

void

Sample

```
if ( # )
{
}
else
{
}
```

label

void **label** ()

Provides a statement with an identifier that you can refer to using a break or continue statement.

For example, you can use a label to identify a loop, and then use the break or continue statements to indicate whether a program should interrupt the loop or continue its execution.

Returns

void

Sample

```
var i = 0, j;
outer_loop: while (i < 10) {
        i++;
        j = 0;
        while (j < 10) {
                j++;
                if (j > i) continue outer_loop;
                application.output("i=" + i + ", j=" + j);
        }
}
```

switch

void **switch** ()

Switch statement.

Returns

void

Sample

```
switch( # )
{
case:
default:
}
```

try catch

void **try catch** ()

try/catch statement

Returns

void

Sample

```
try
{
        #
}
 catch(#)
{
        #
}
```

try catch finally

void **try catch finally** ()
try/catch/finally statement
Returns

void
Sample

```
try
{
        #
}
 catch(#)
{
        #
} finally
{
        #
}
```

var

void **var** ()
Variable declaration
Returns

void
Sample

```
var #;
```

while

void **while** ()
while loop
Returns

void
Sample

```
while ( # )
{
        #
}
```