
BETA
p5*JS

PROGRAMMING WITH P5.JS

@FH-Potsdam

by Fabian Morón Zirfas

Winter 2015/2016

"Don’t Panic"

The Hitchhiker's Guide to the Galaxy

Mother Tongues

Tracing the roots of computer languages through the ages

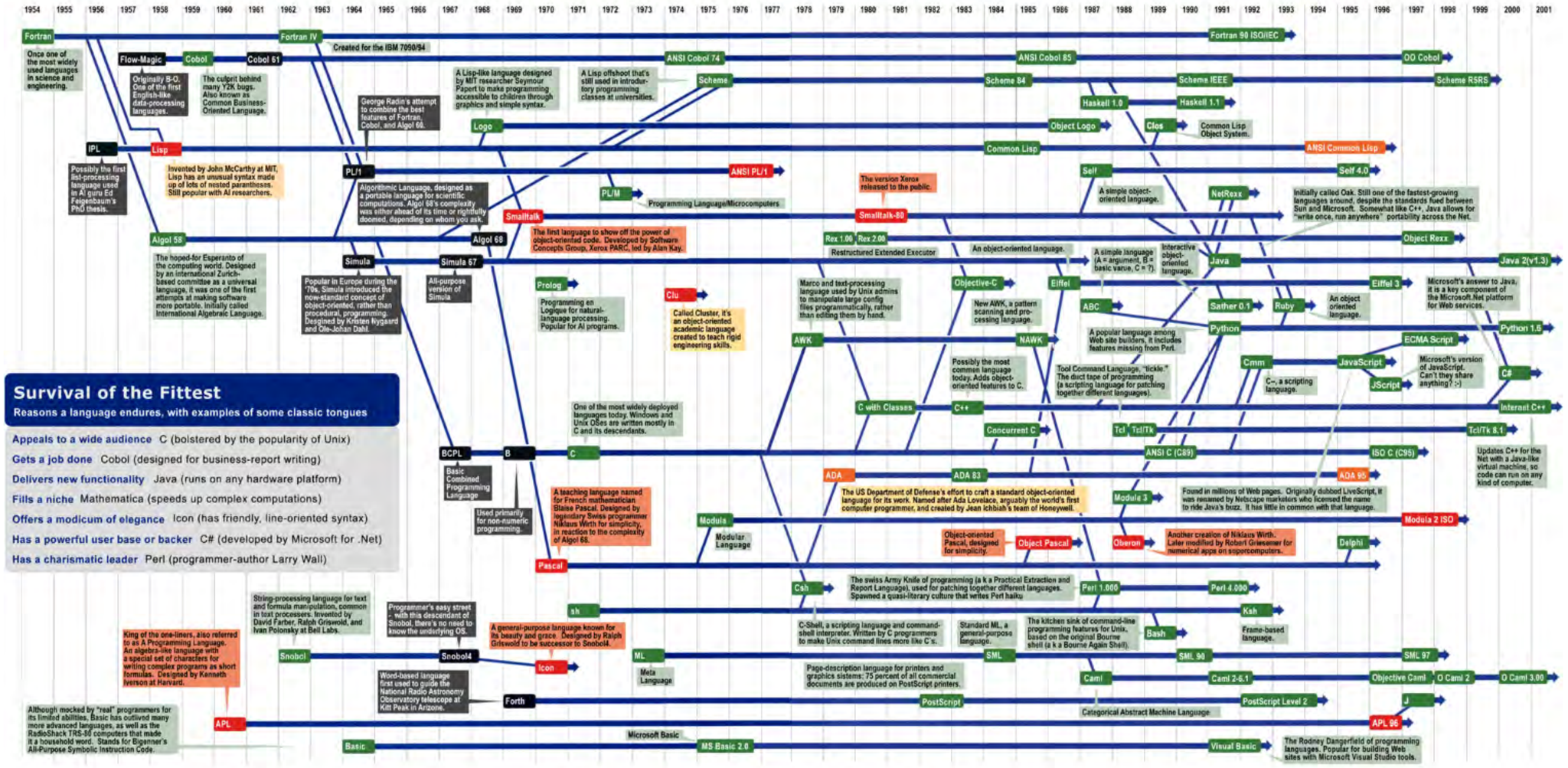
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**

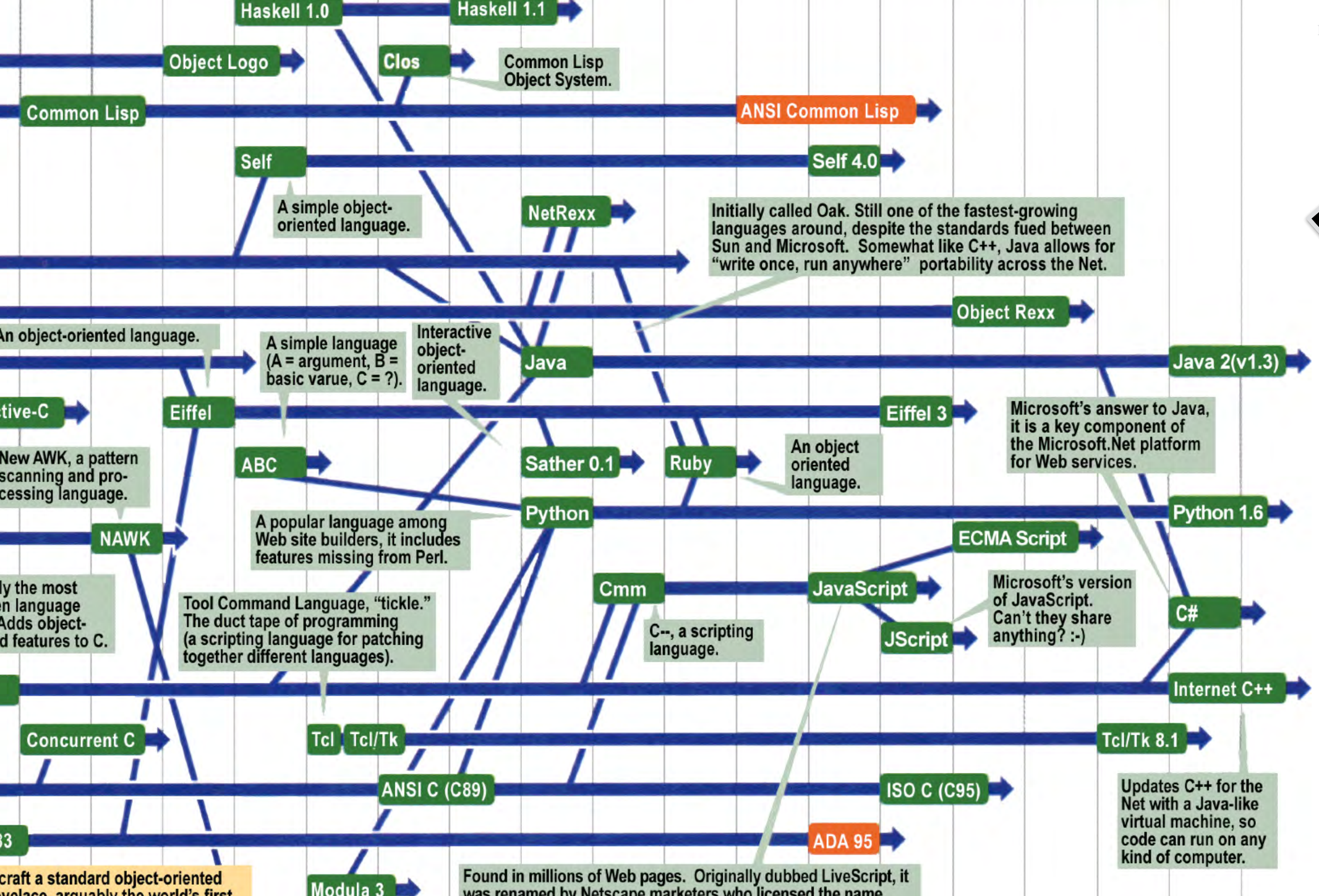


Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University


Haskell 1.0
Haskell 1.1
Object Logo
Clos
Common Lisp Object System.
Common Lisp
ANSI Common Lisp
Self
Self 4.0
A simple object-oriented language.
NetRexx
Initially called Oak. Still one of the fastest-growing languages around, despite the standards fued between Sun and Microsoft. Somewhat like C++, Java allows for "write once, run anywhere" portability across the Net.
Object Rexx
An object-oriented language.
A simple language (A = argument, B = basic varue, C = ?).
Interactive object-oriented language.
Java
Java 2(v1.3)
tive-C
Eiffel
Eiffel 3
Microsoft's answer to Java, it is a key component of the Microsoft.Net platform for Web services.
New AWK, a pattern scanning and pro-cessing language.
ABC
Sather 0.1
Ruby
An object oriented language.
Python
Python 1.6
NAWK
A popular language among Web site builders, it includes features missing from Perl.
ECMA Script
Cmm
JavaScript
Microsoft's version of JavaScript. Can't they share anything? :-)
y the most n language Adds object-d features to C.
Tool Command Language, "tickle." The duct tape of programming (a scripting language for patching together different languages).
C--, a scripting language.
JScript
C#
Internet C++
Concurrent C
Tcl
Tcl/Tk
Tcl/Tk 8.1
ANSI C (C89)
ISO C (C95)
Updates C++ for the Net with a Java-like virtual machine, so code can run on any kind of computer.
33
ADA 95
craft a standard object-oriented
Modula 3
Found in millions of Web pages. Originally dubbed LiveScript, it was renamed by Netscape marketers who licensed the name
We are here

"Processing is (…) built for (…) the purpose of teaching the fundamentals of computer programming in a visual context (…)"

Wikipedia


BETA
p5*JS

PREREQUISITES

- P5.js Editor (Beta)

oder

- Texteditor - Atom.io oder sublimetext.com/3
- p5.JS Libray - p5js.org/download/
- Browser Google Chrome
- NodeJs https://nodejs.org

USEFUL ATOM PACKAGES

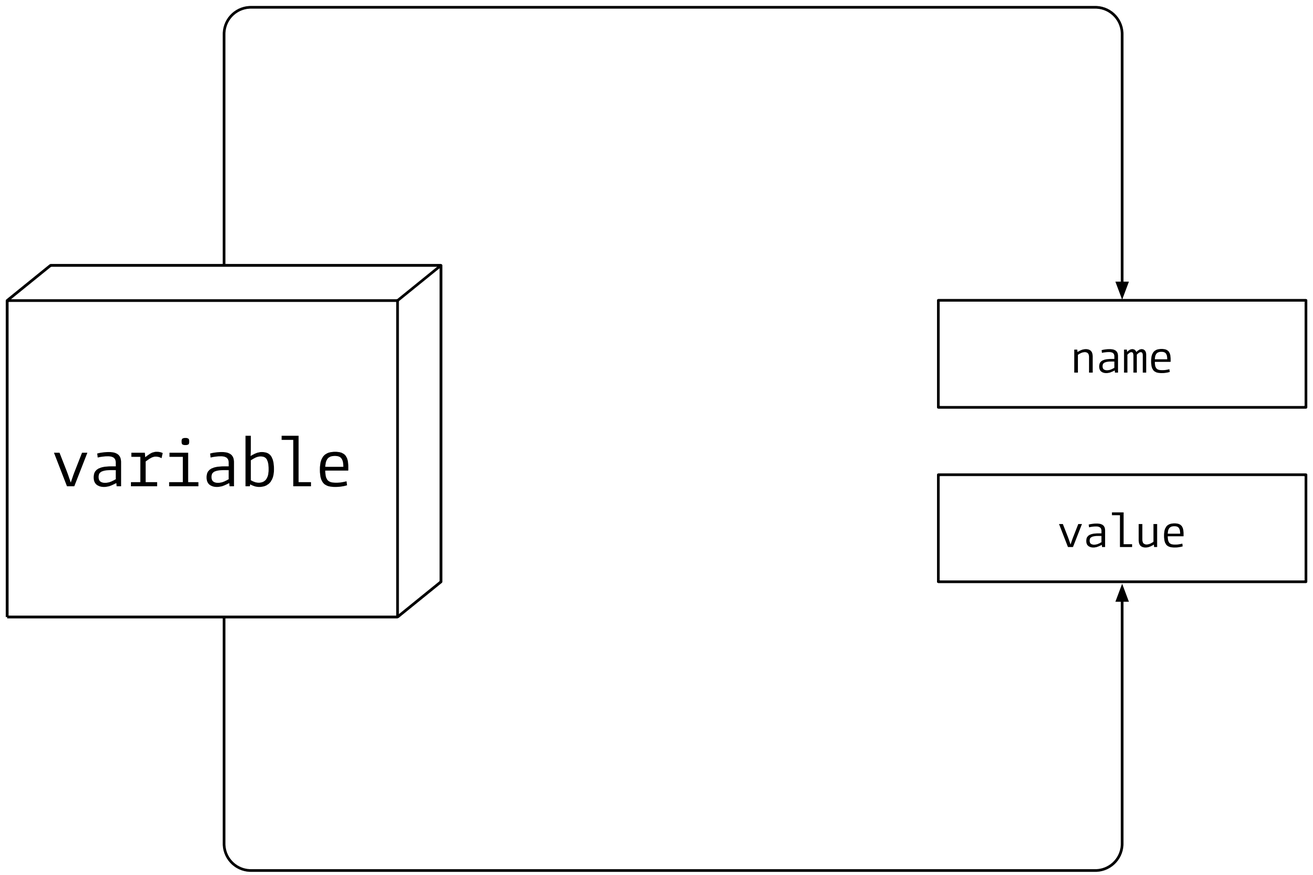
install from within the atom jshint, emmet,
formatter, linter, linter-htmlhint

USEFUL SUBLIME PACKAGES

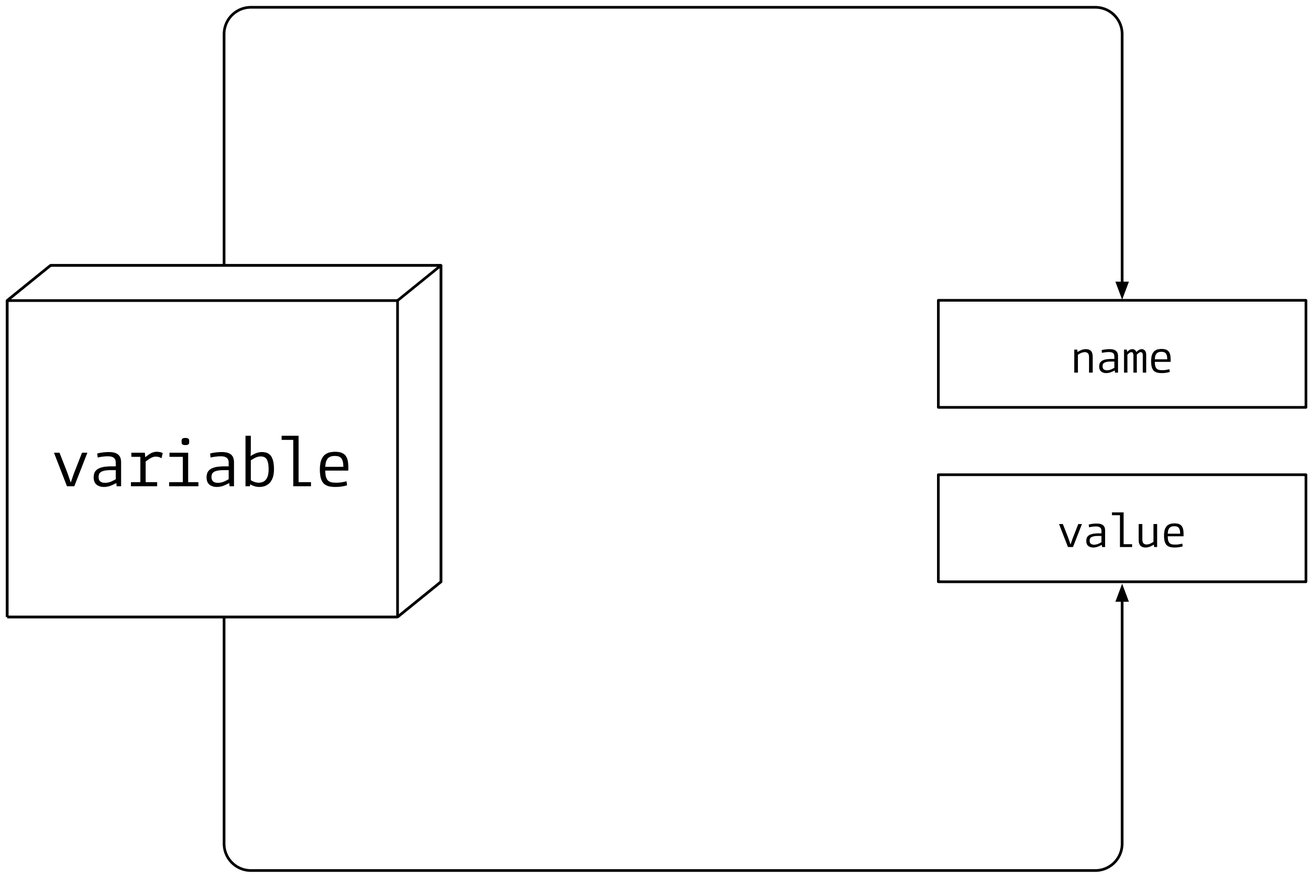
install via packagecontrol.io, Emmet, CodeFormatter, SublimeLinter, SublimeLinter-contrib-eslint, SublimeLinter-contrib-htmlhint

7 BASIC THINGS IN PROGRAMMING

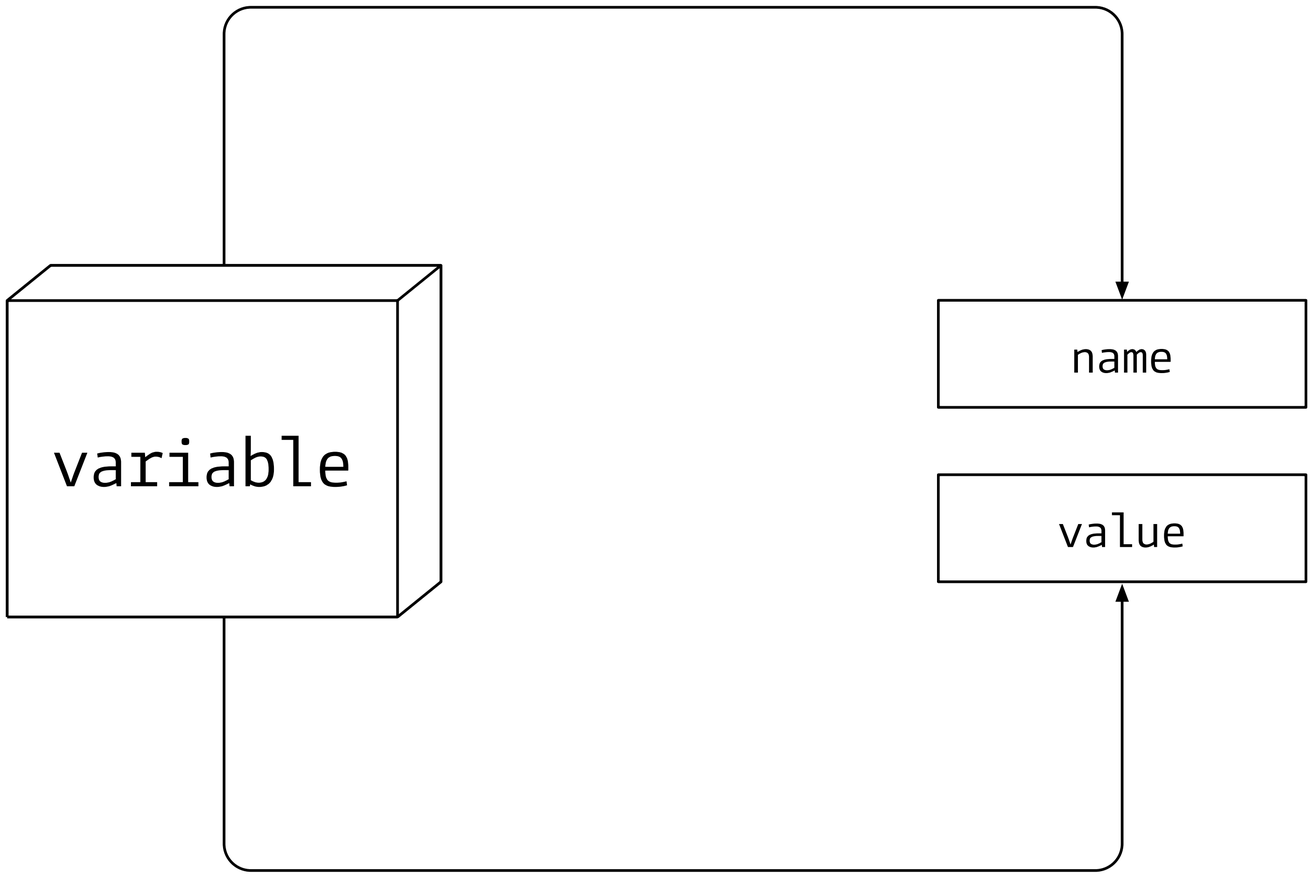
1. Variablen
2. Objekte
3. Arrays
4. Konditionen
5. Schleifen
6. Funktionen
7. Algorithmus

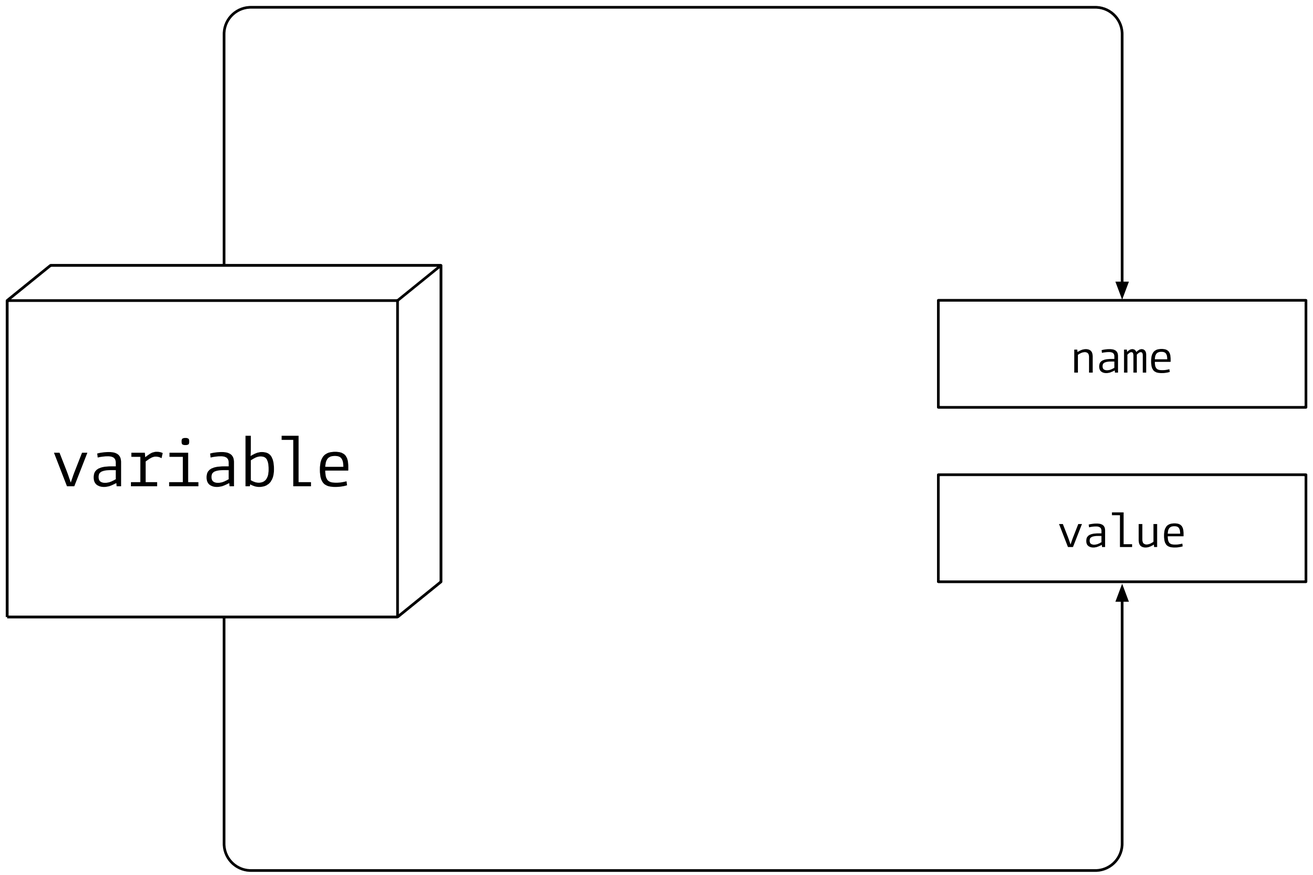

variable
name
value

var

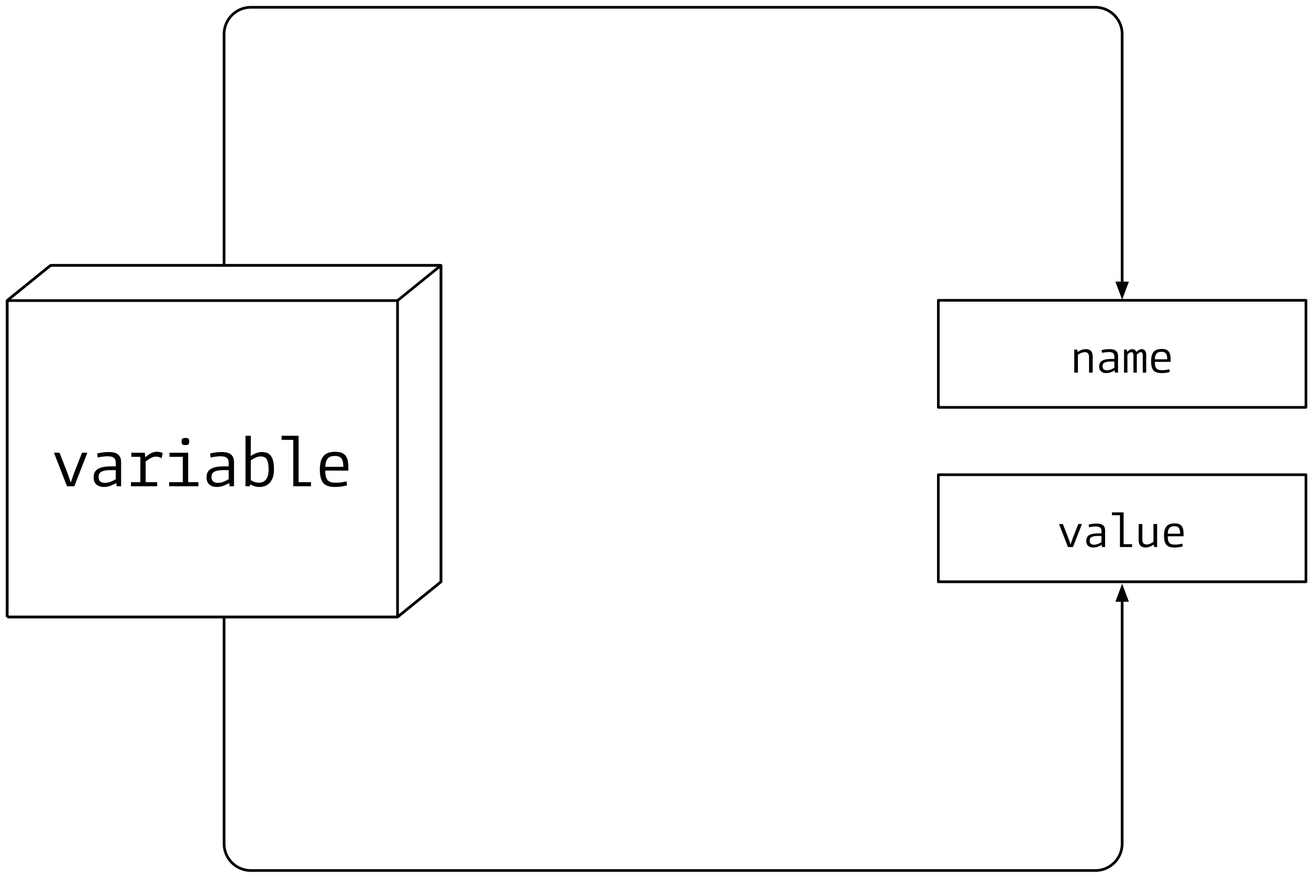


var

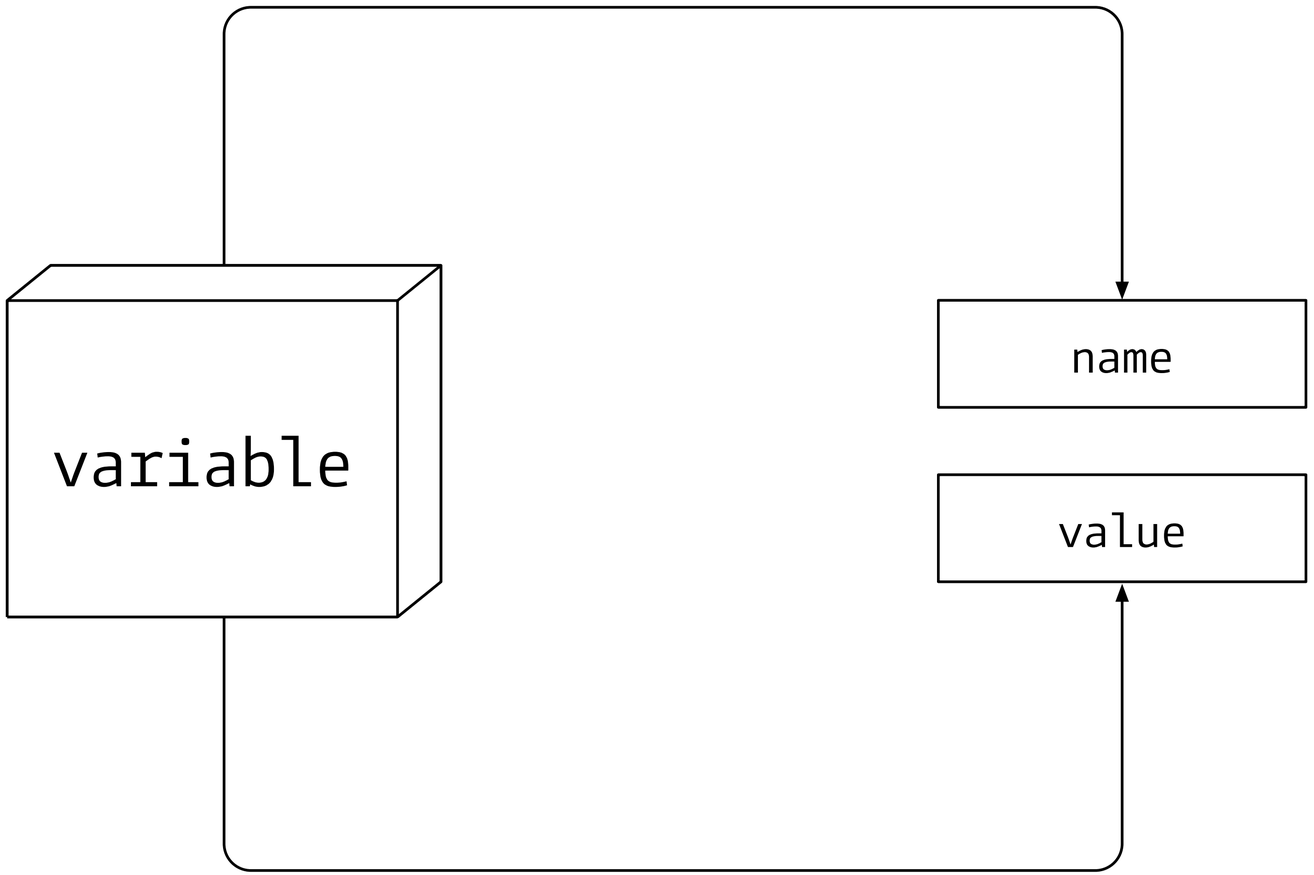




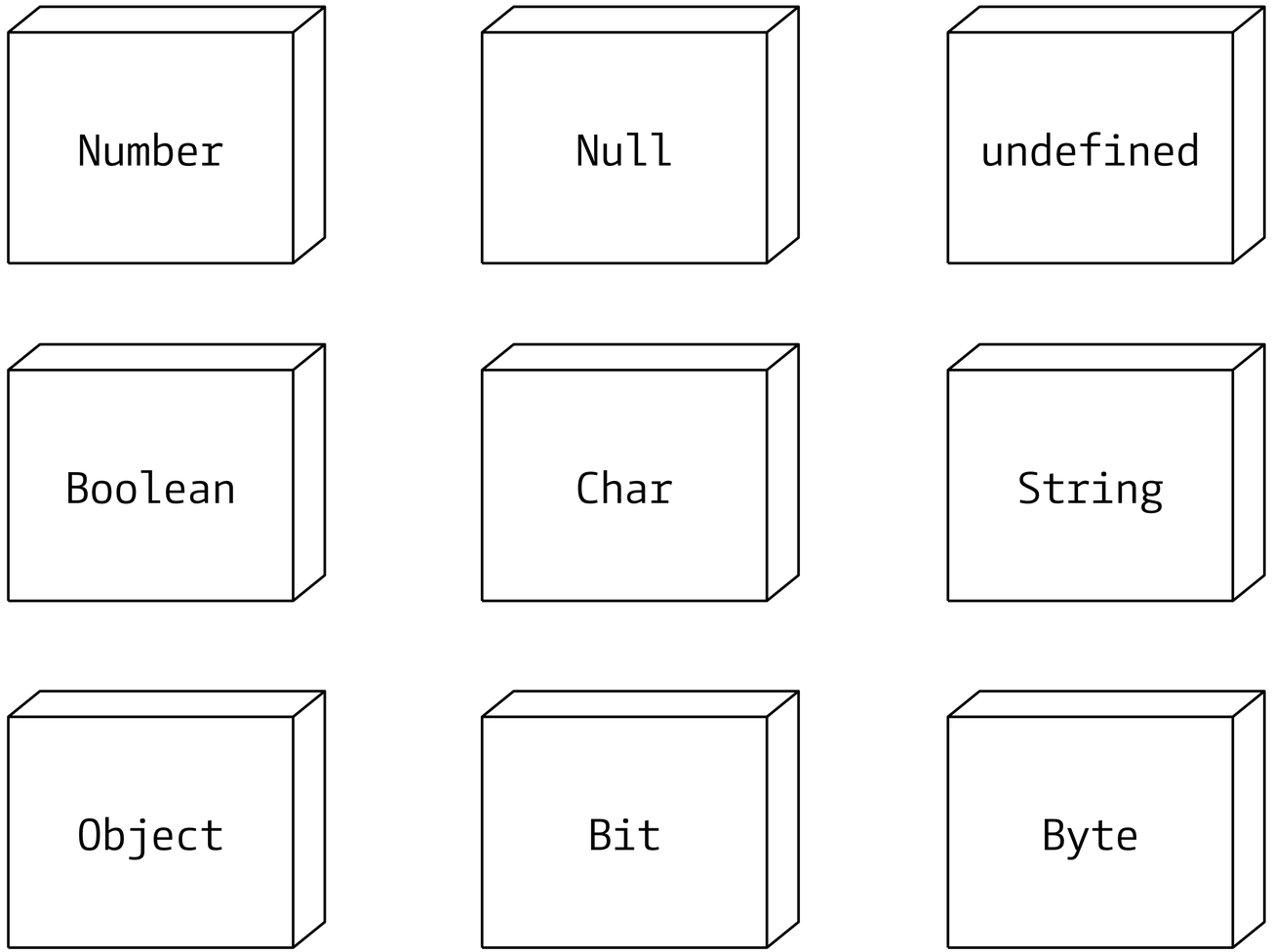

```
var jim
```

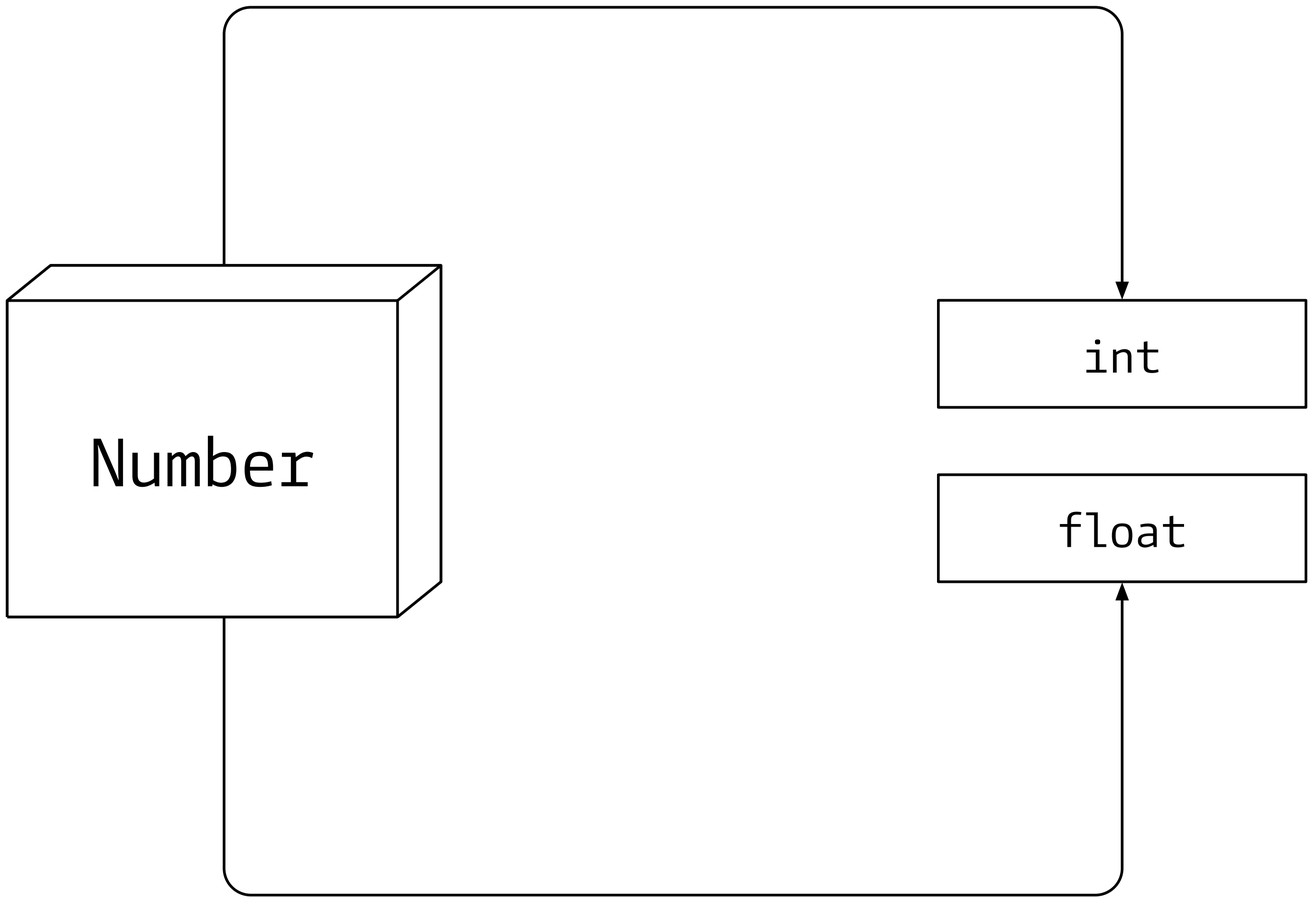


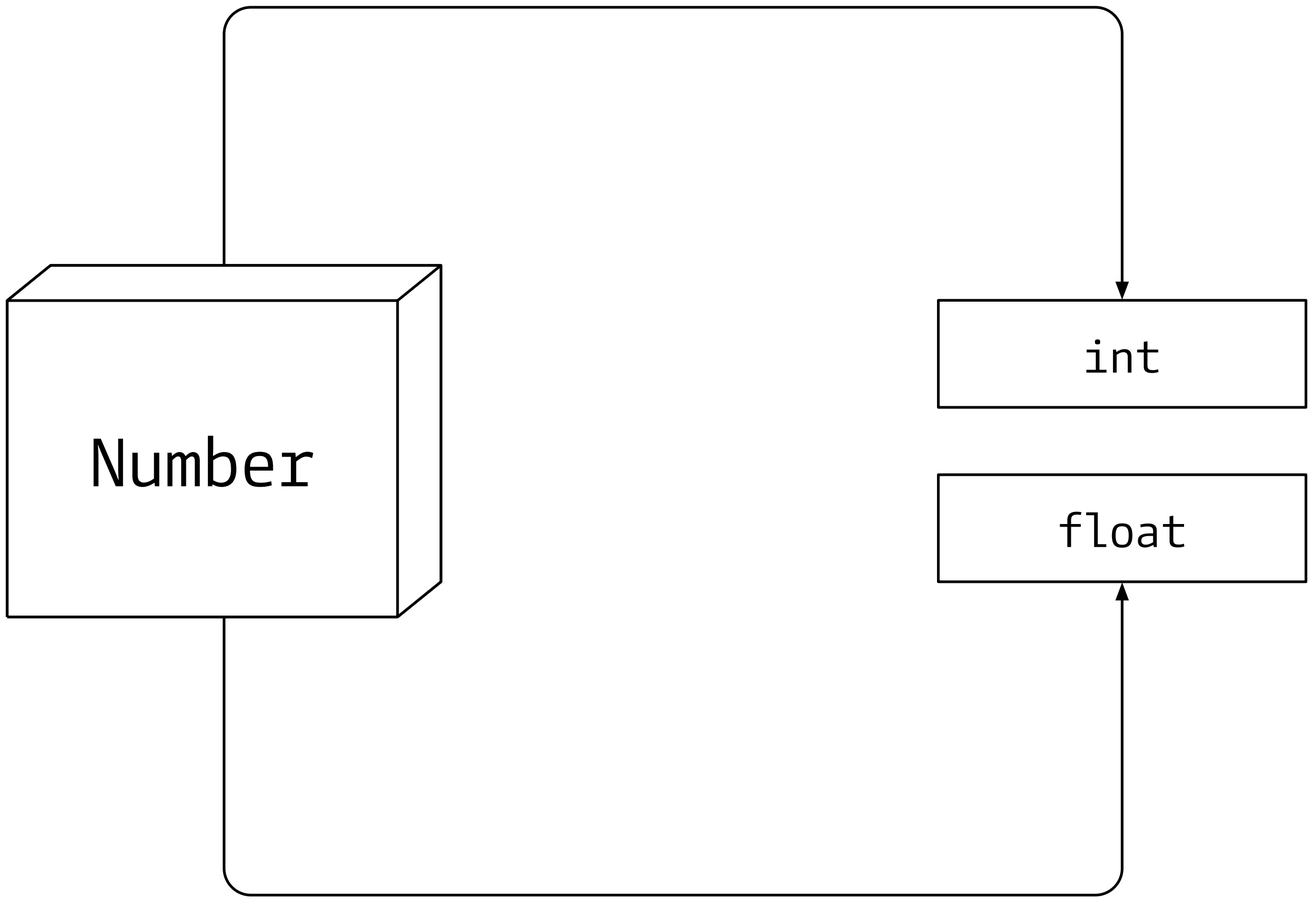

```
var jim = 100
```



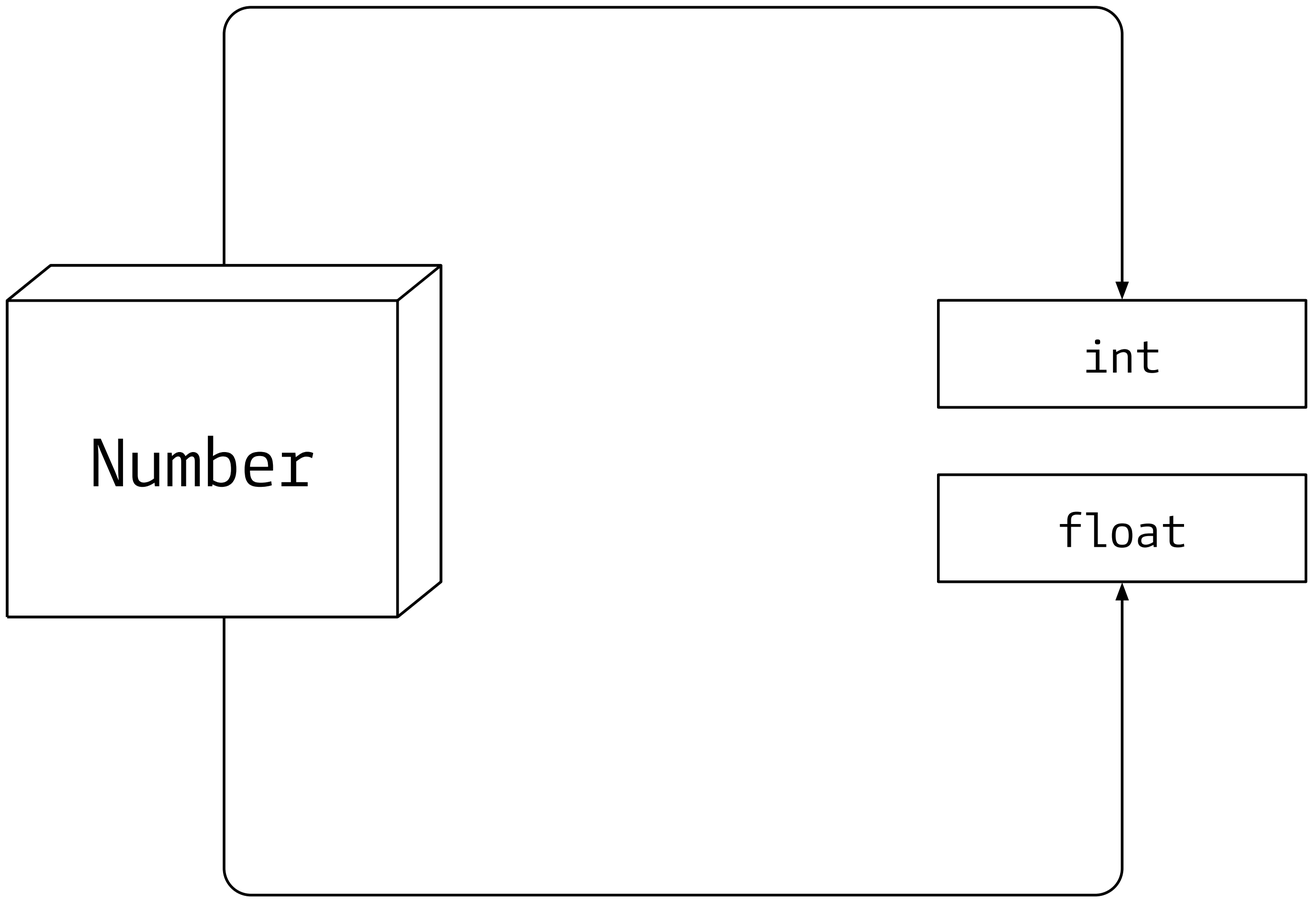

```
var jim = 100;
```


Number
Null
undefined
Boolean
Char
String
Object
Bit
Byte

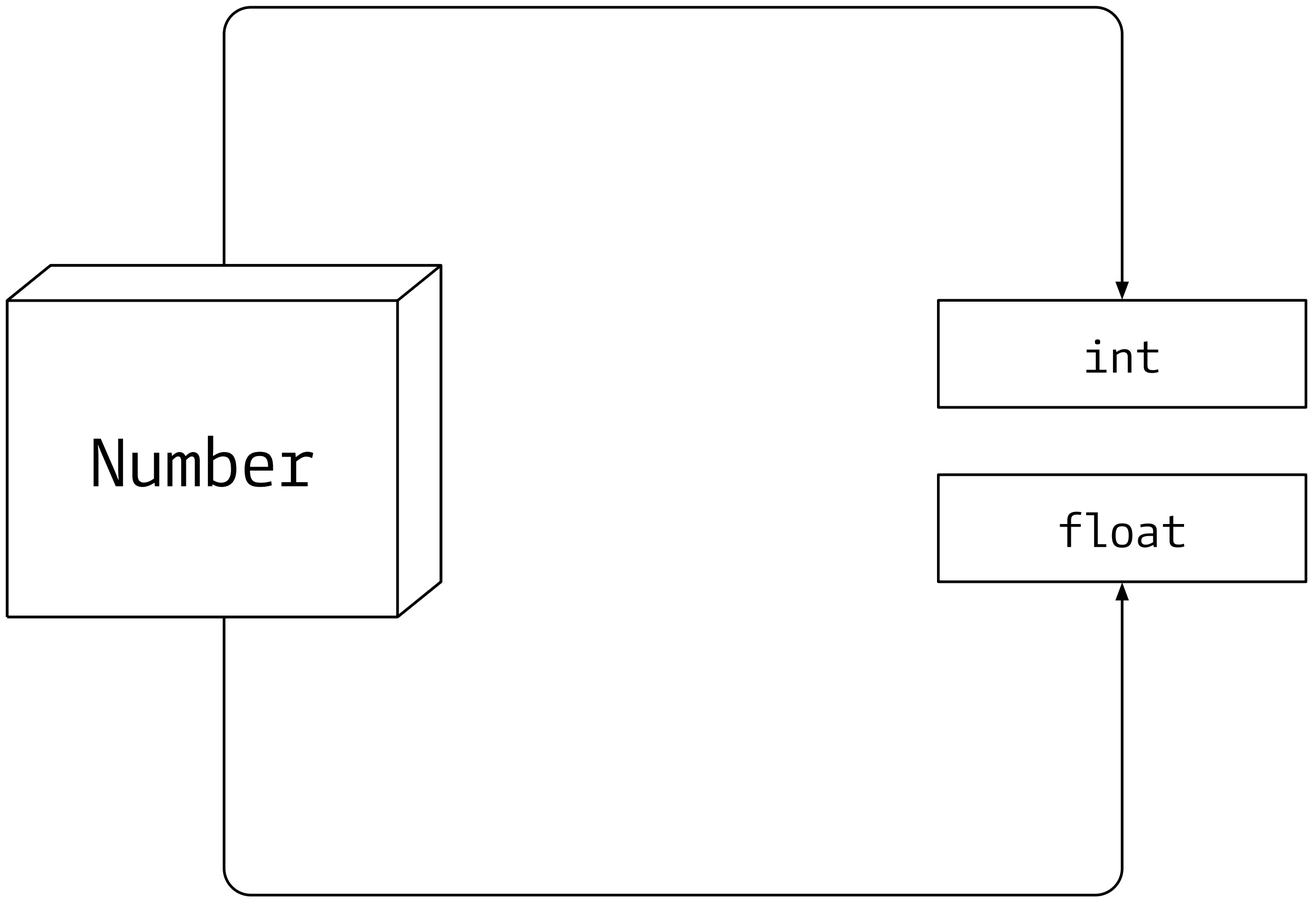

Number
int
float



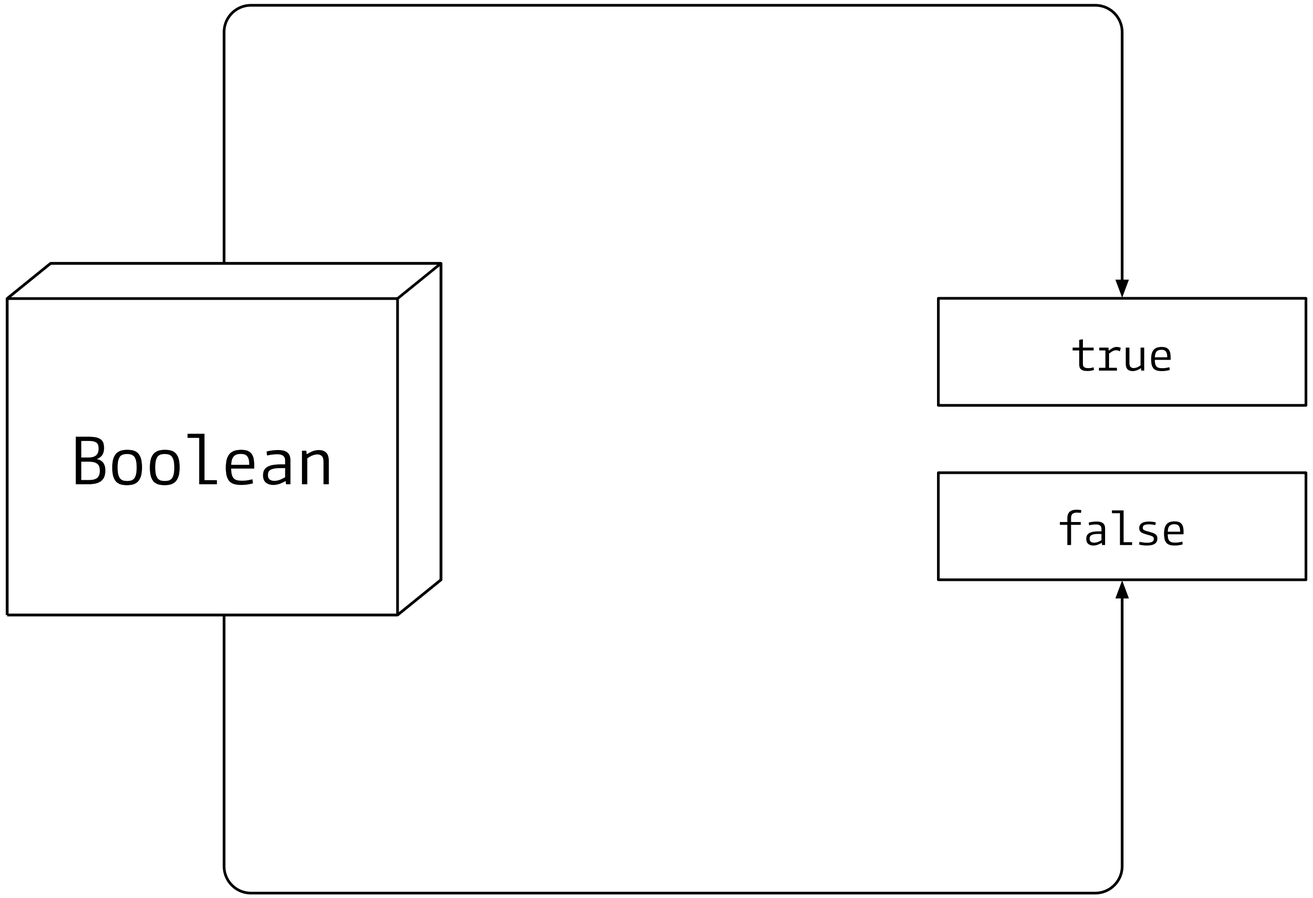

```
var x = 13;
```

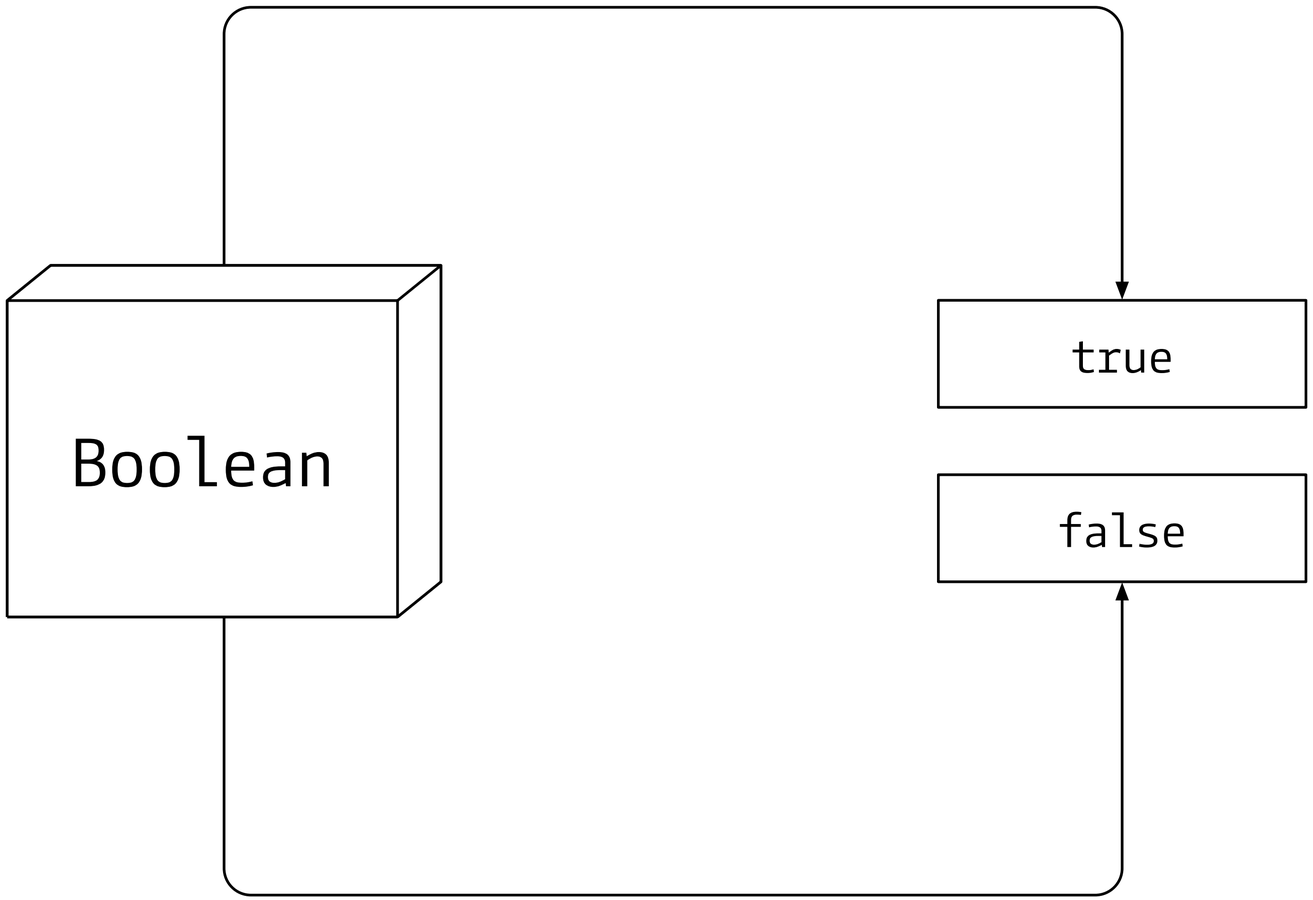



```
var x = 13;
var y = -42;
```

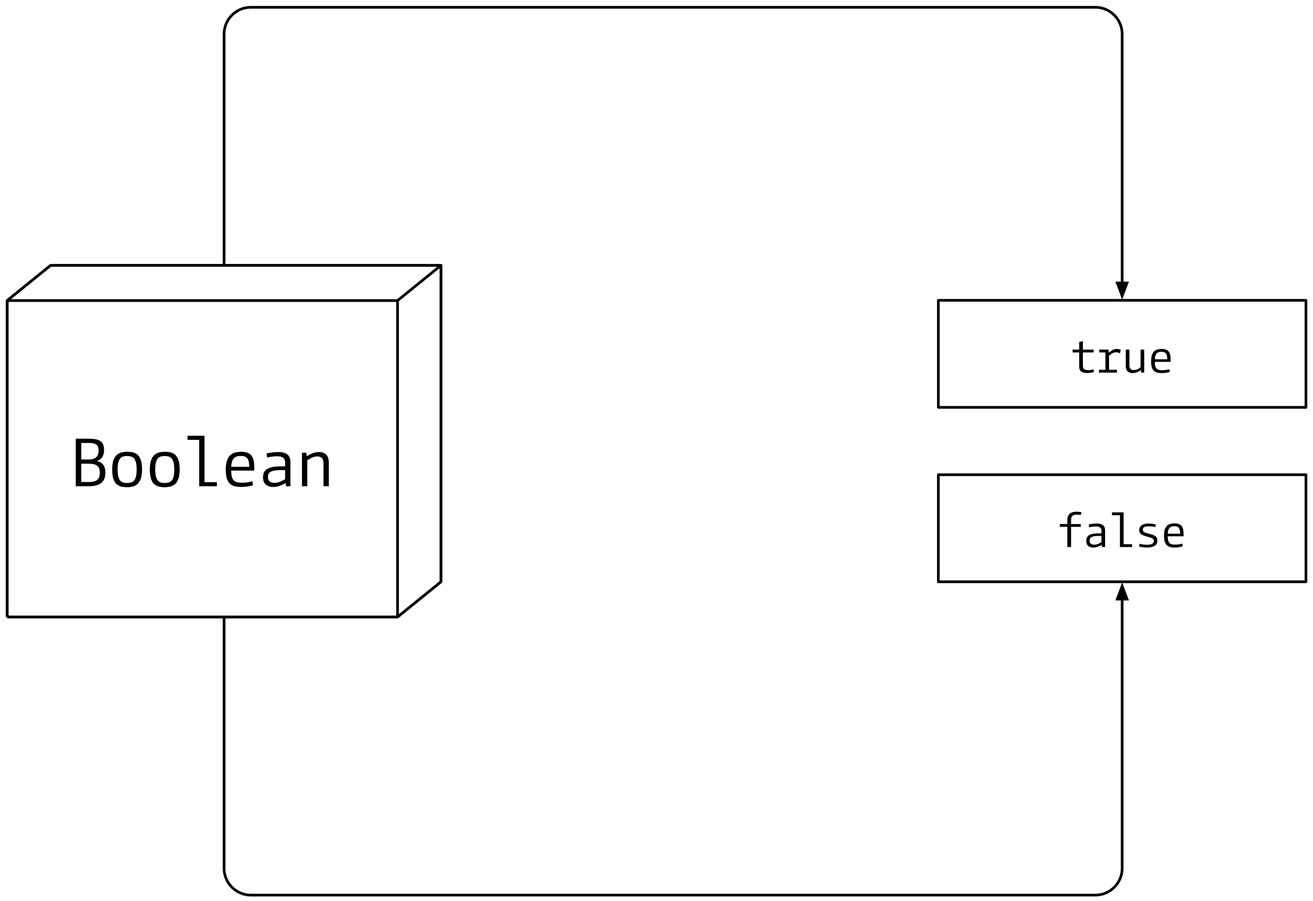



```
var x = 13;
var y = -42;
var z = -23.5;
```

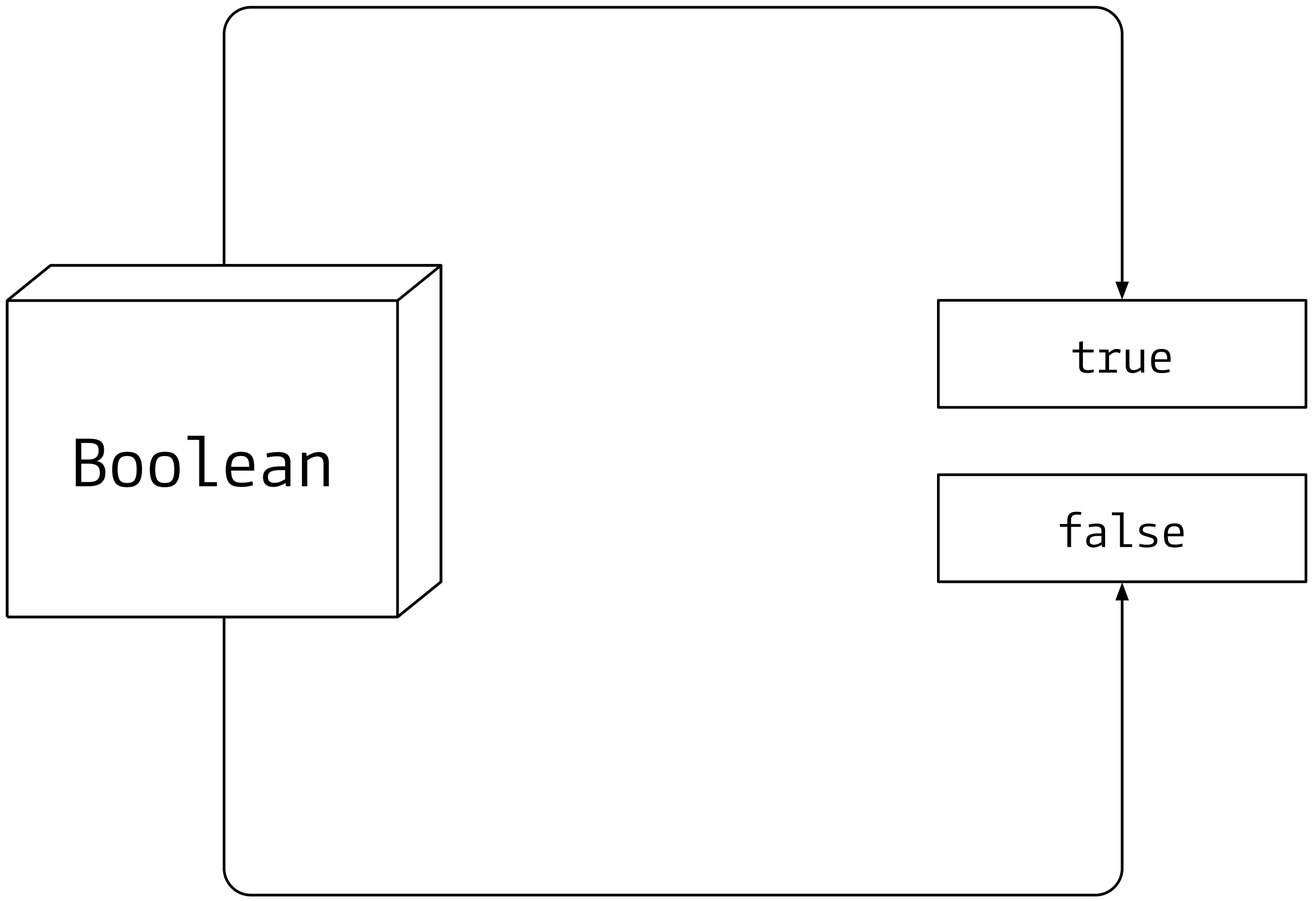

Boolean
true
false



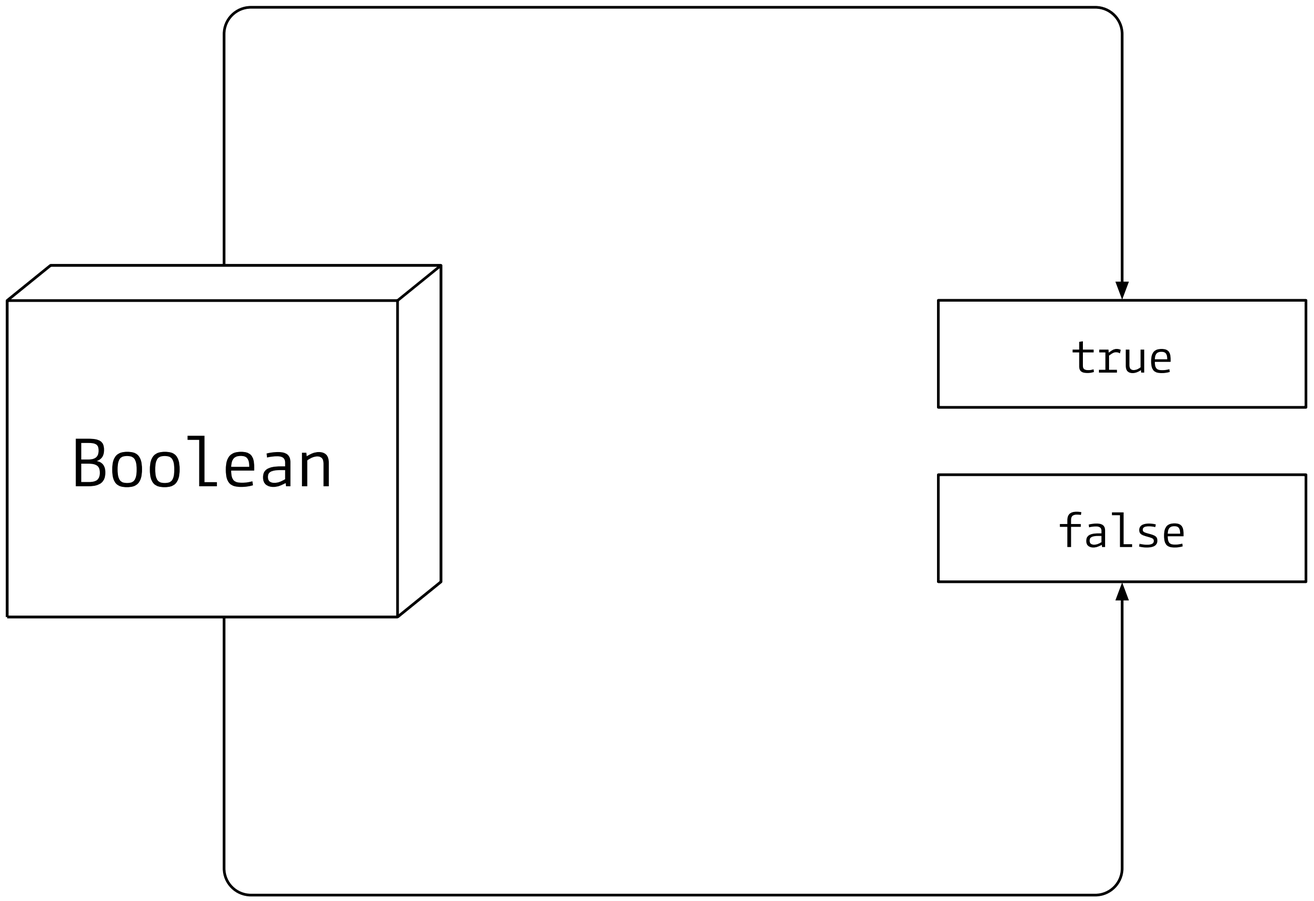

```
var status = true;
```



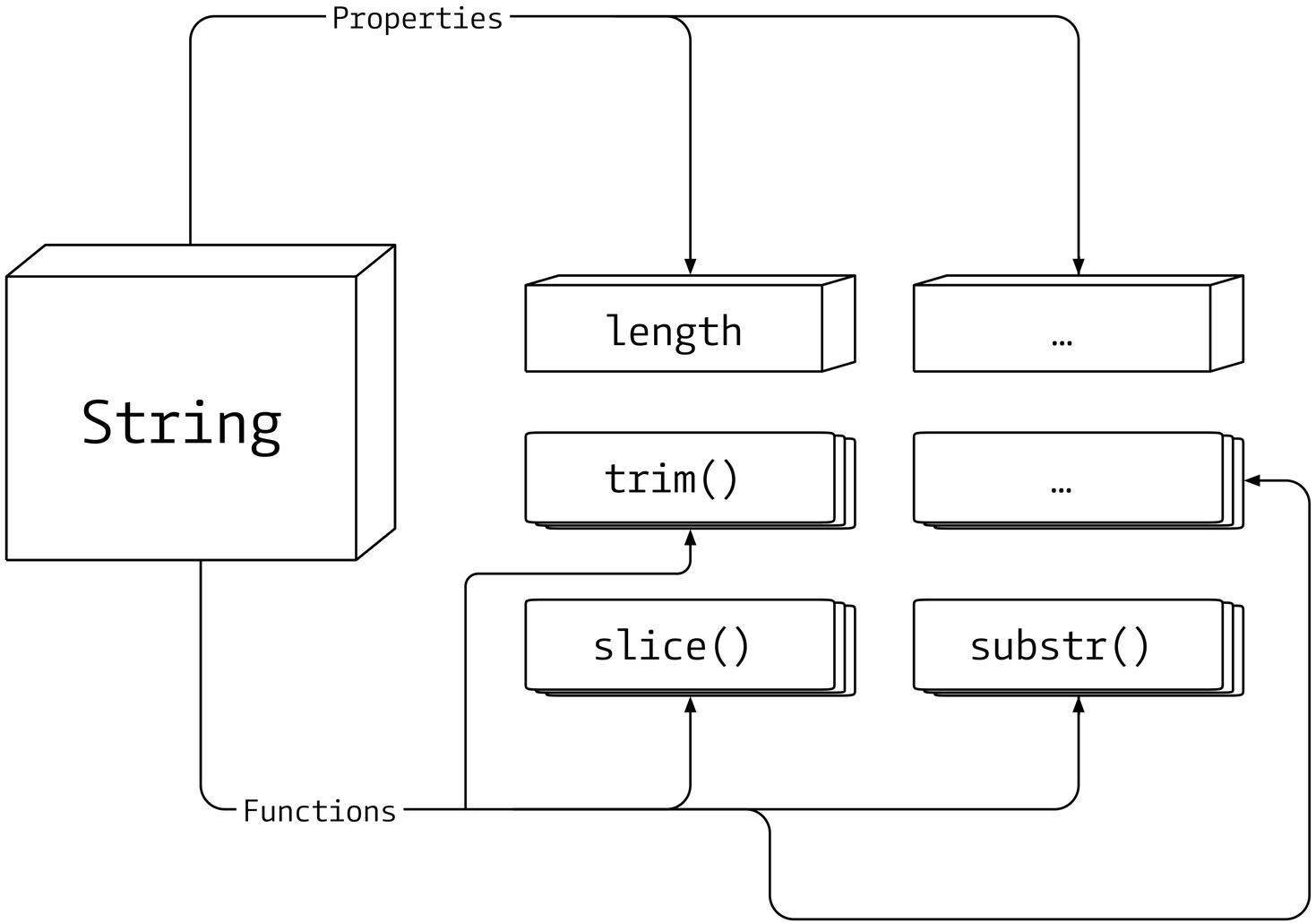

```
var status = false;
```



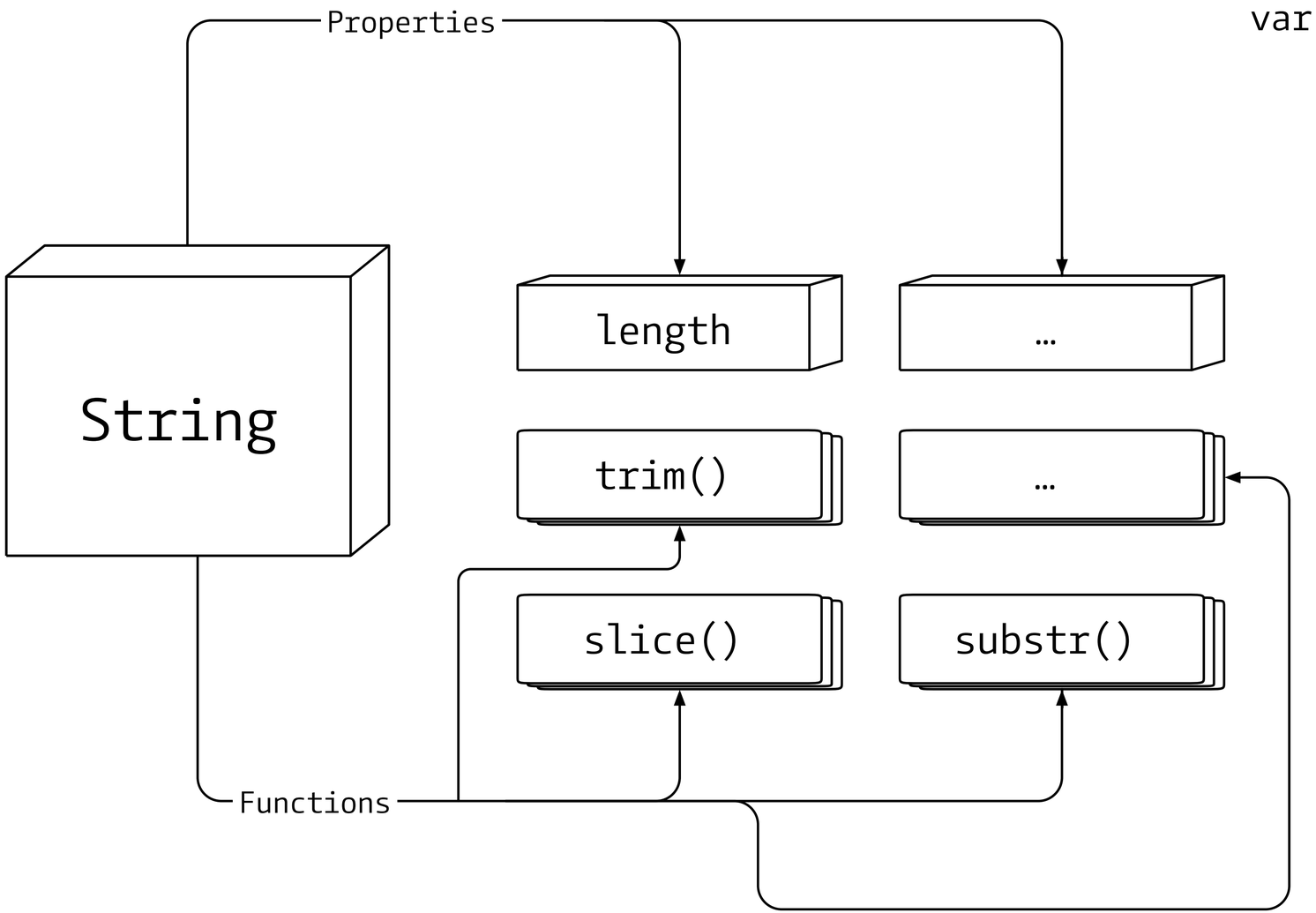

```
var status = false;

// something happens
// so we set status to true

status = true;
```

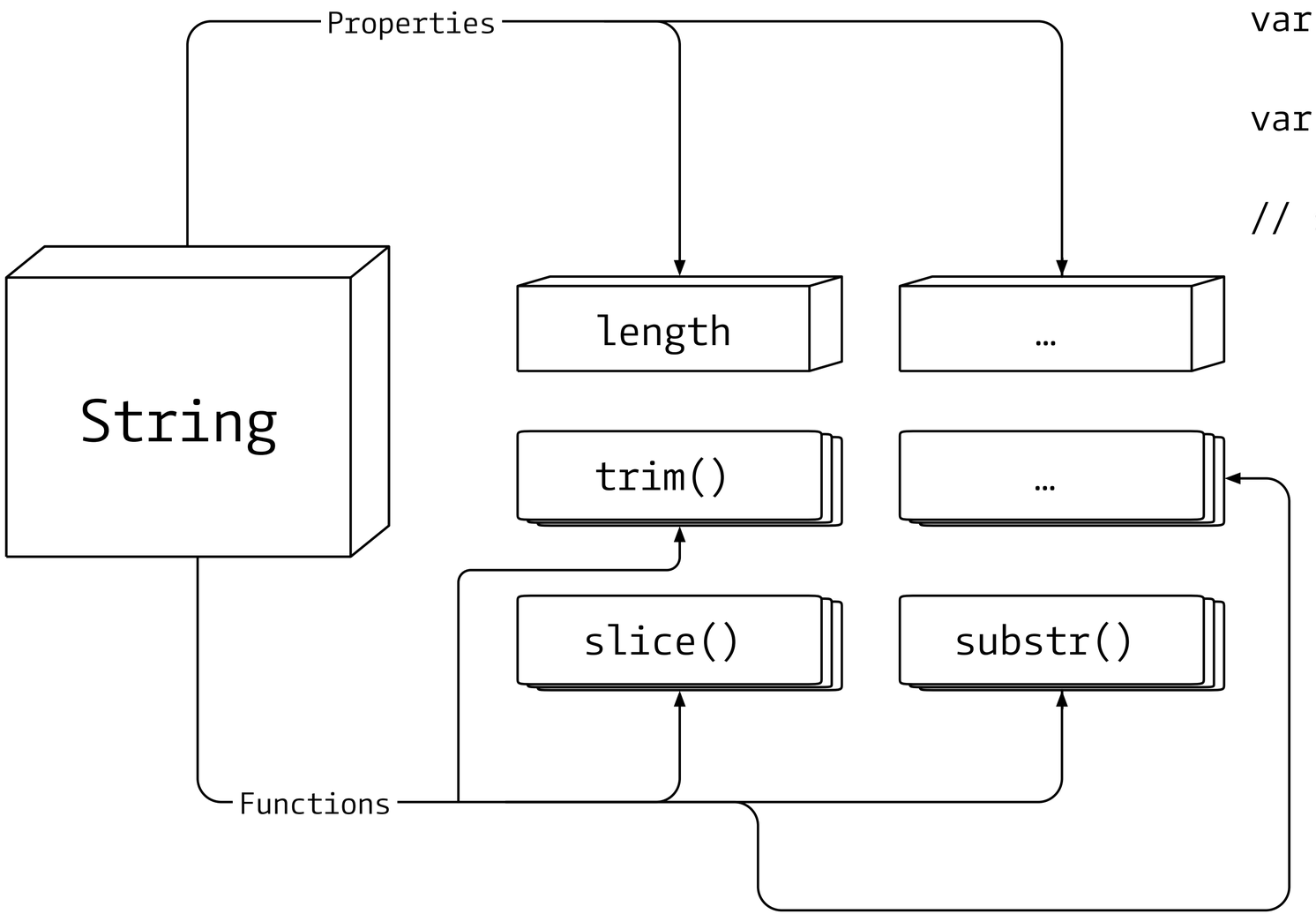



```
var status = false;

// something happens
// lets switch status

status = !status; // <— is true

// something else happens
// lets switch status again

status = !status; // <- its false
```

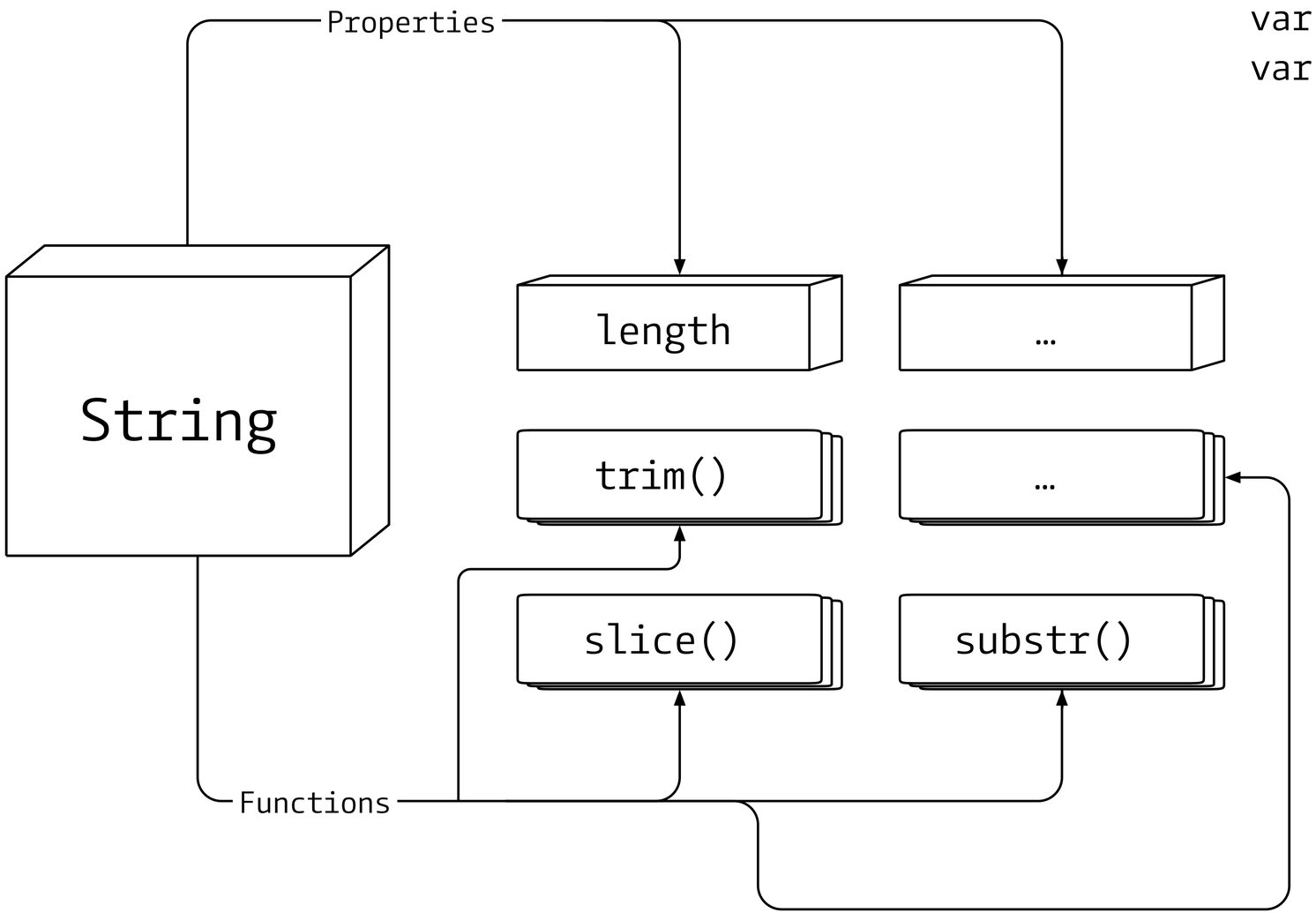

Properties
String
length
…
trim()
…
slice()
substr()
Functions

```
var str = "Hello World";
```

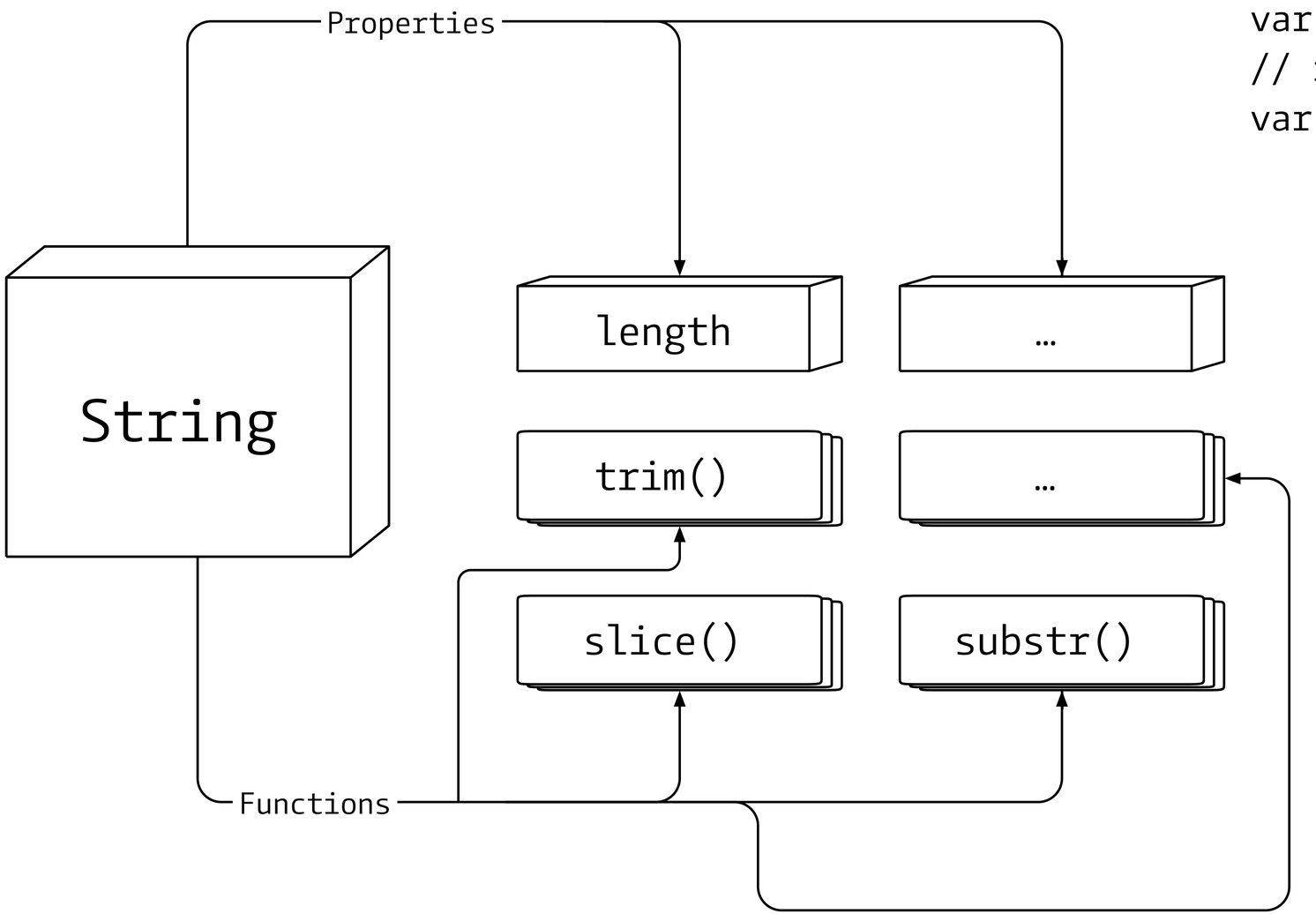




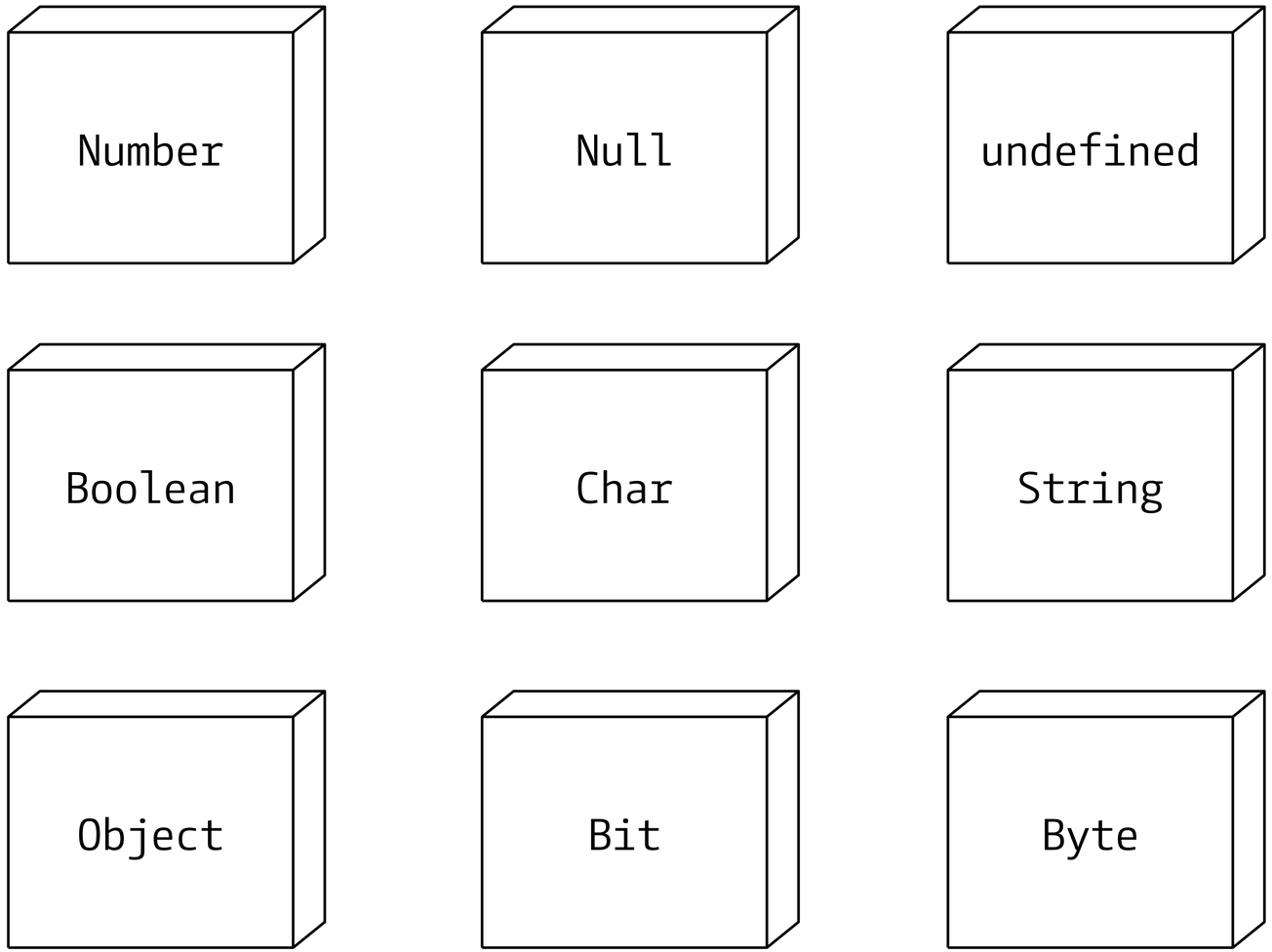

```
var str = "Hello World";

var sub = str.substr(6);

// sub = "Hello"
```




```
var str = "Hello World";
var len = str.length; // 11
```

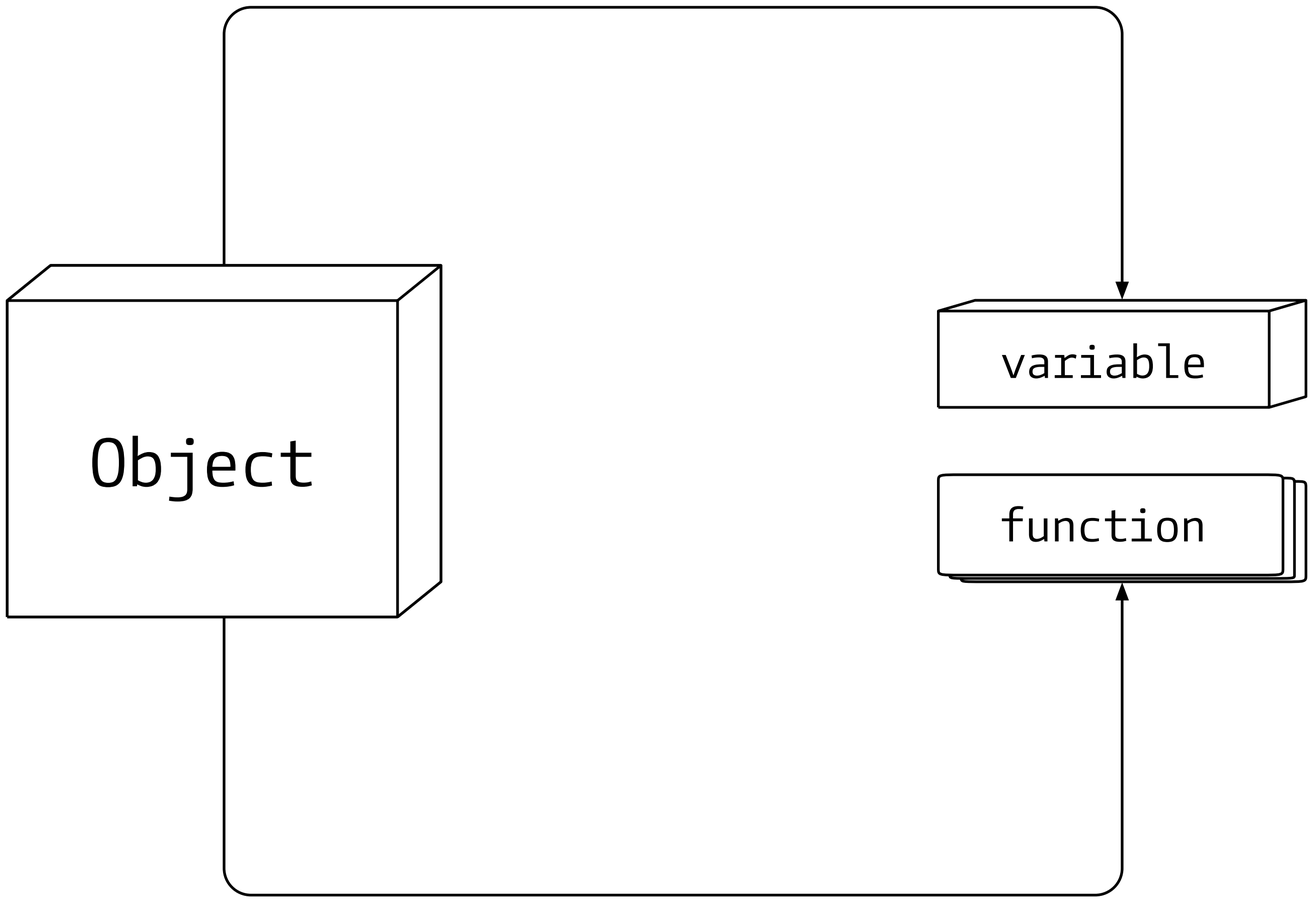


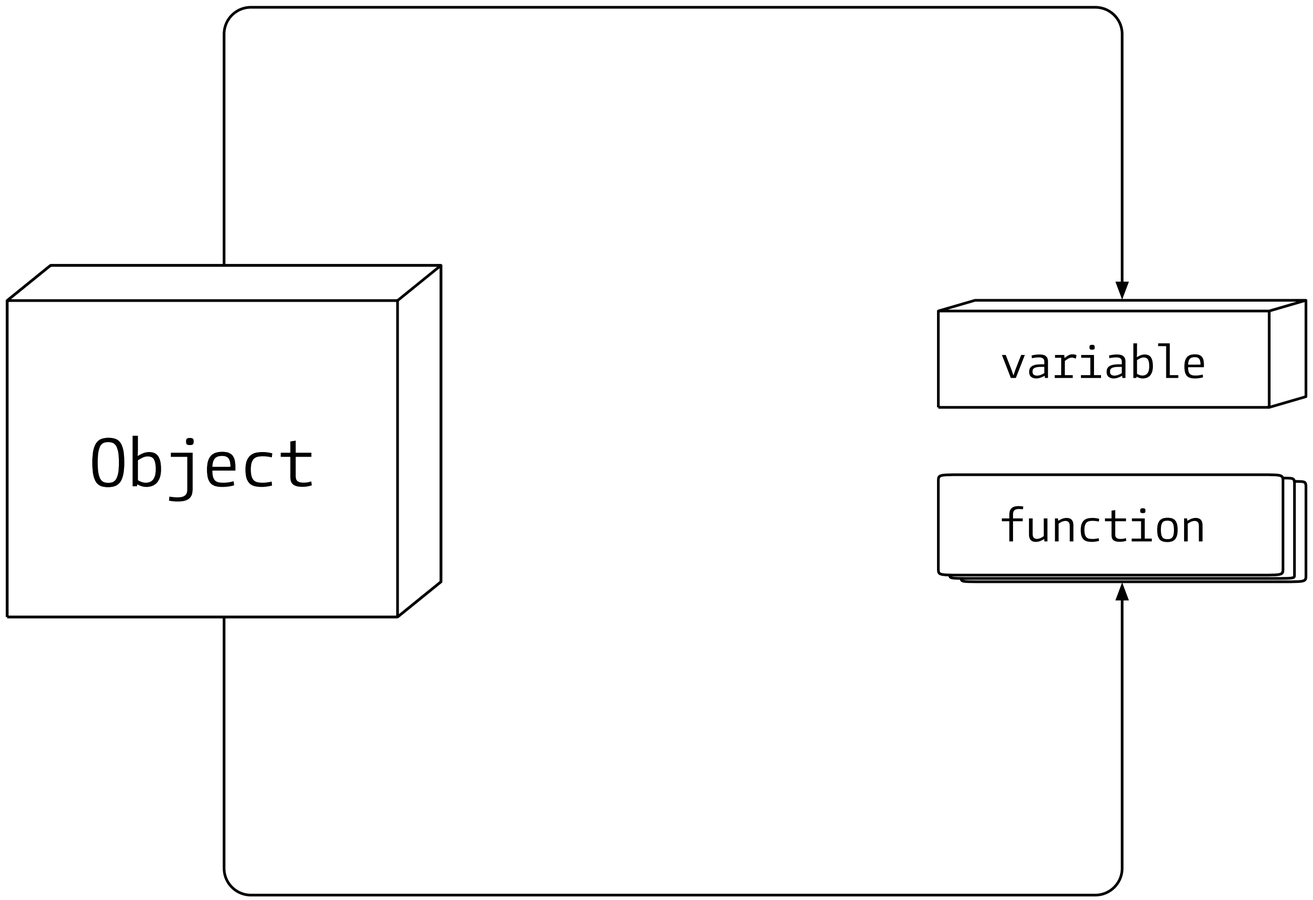

```
var str = "  Hello World  ";
// remove whitespace
var clean = str.trim();
```


Number
Null
undefined
Boolean
Char
String
Object
Bit
Byte

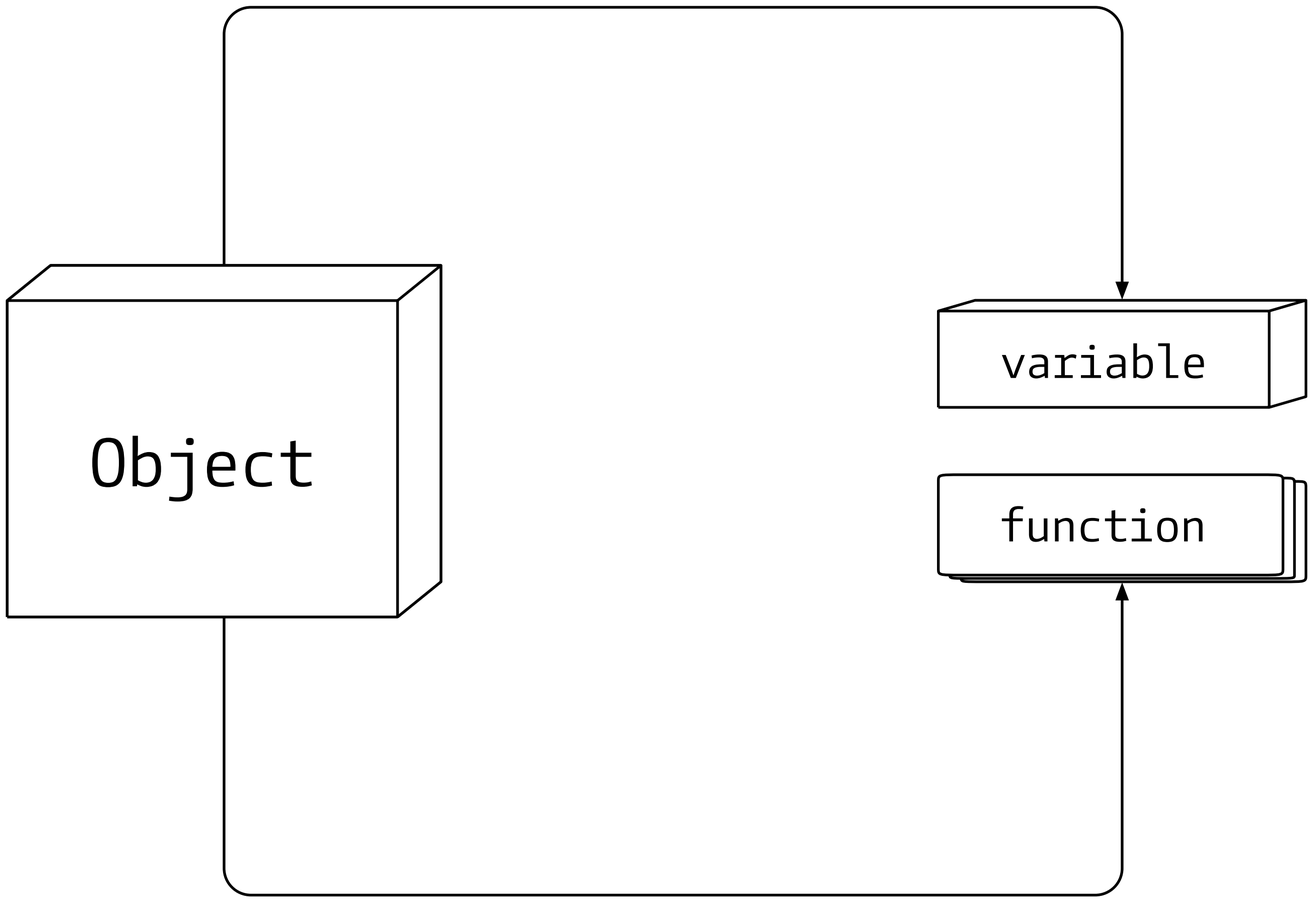
7 BASIC THINGS IN PROGRAMMING

1. Variablen ✓
2. Objekte
3. Arrays
4. Konditionen
5. Schleifen
6. Funktionen
7. Algorithmus

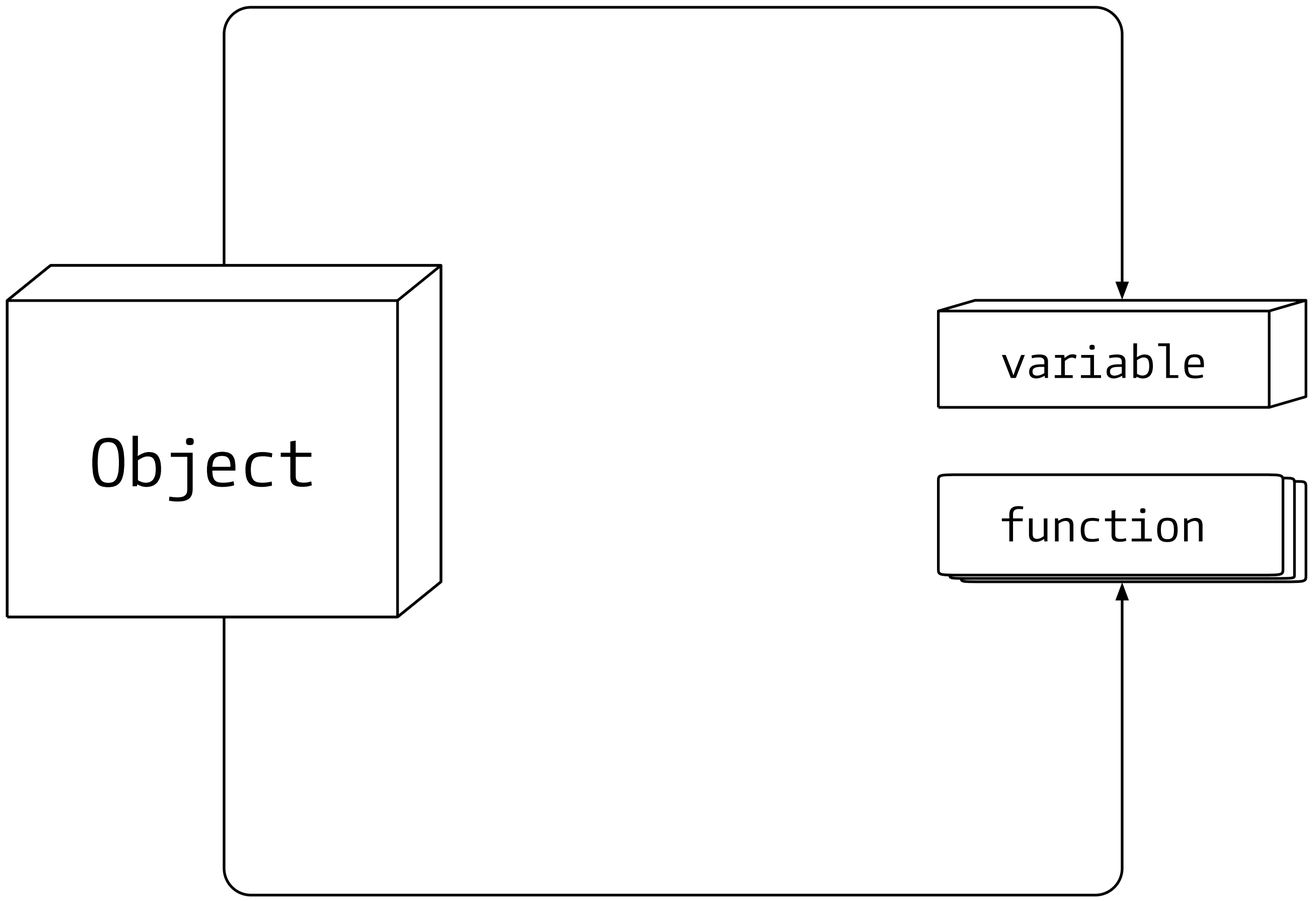

Object
variable
function



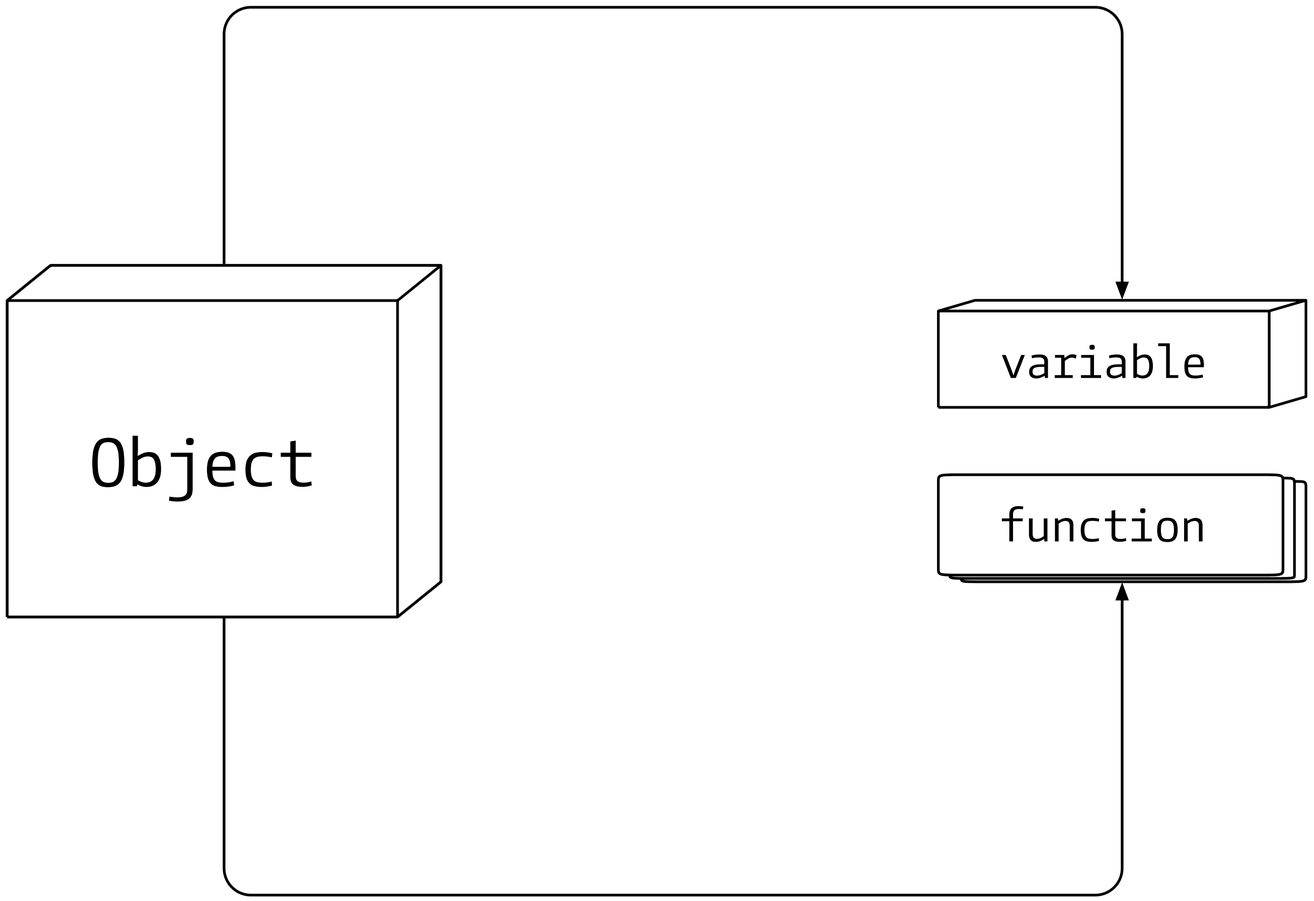

```
var obj = {};
```



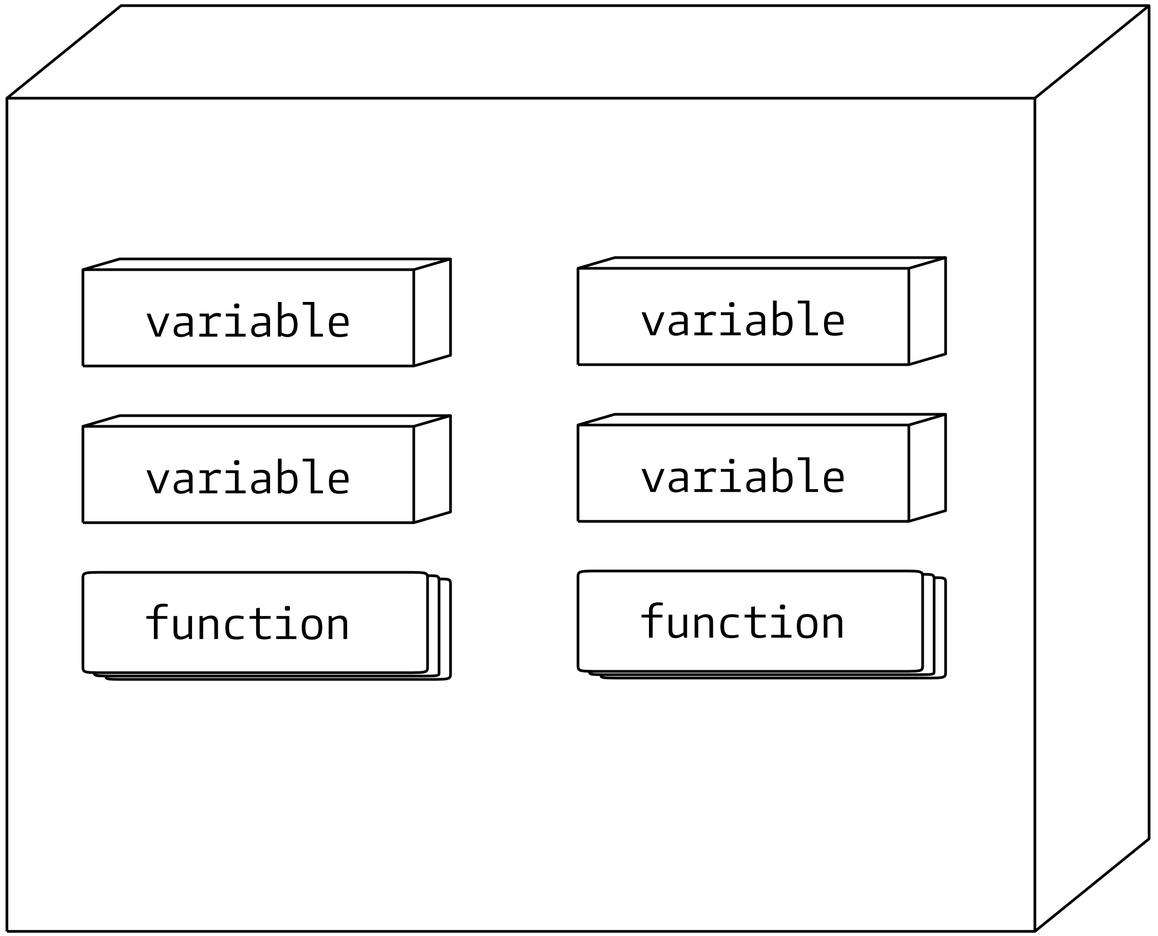

```
var obj = {"posx":5,"posy":20};
```



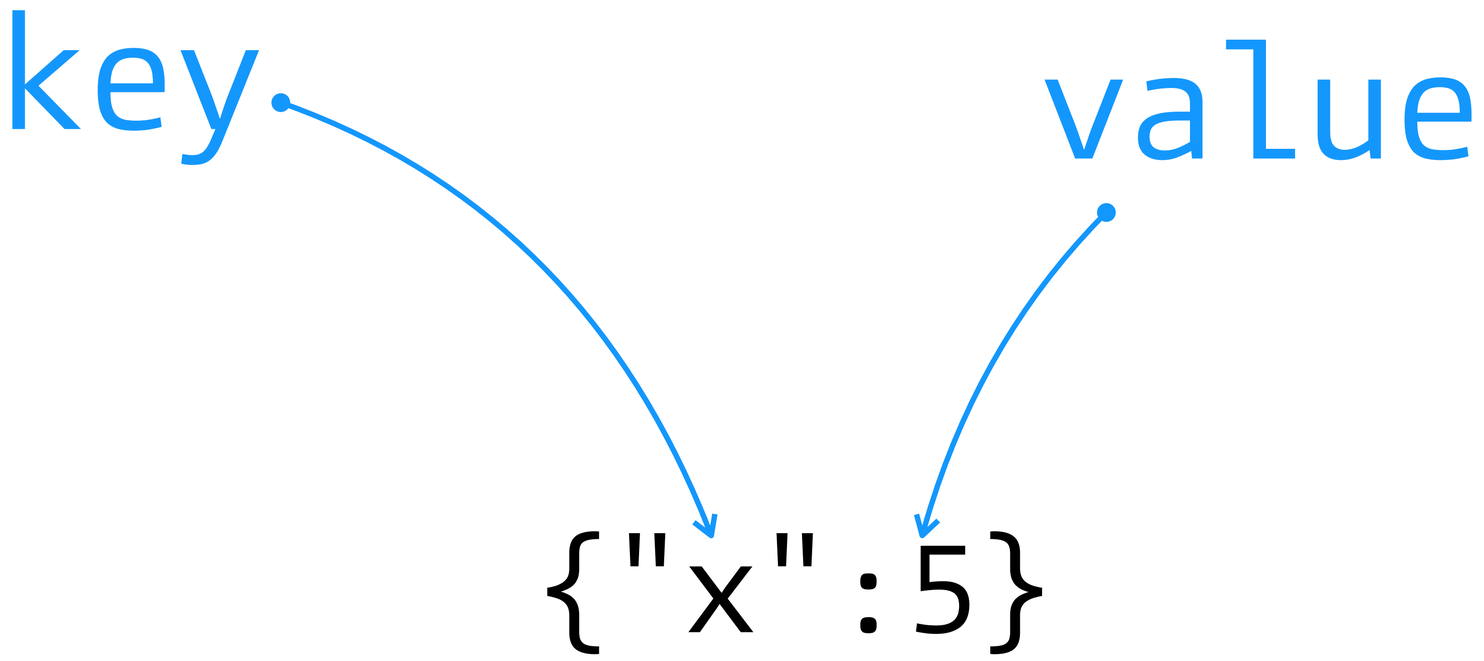

```
var obj = {"posx":5,"posy":20};

obj.something = true;

console.log(obj);
>{ posx: 5,
   posy: 20,
   something: true }
```




```
var obj = {
"add":function(a,b){
  return a+b;
  },
 "subtract":function(a,b){
    return a-b;
  }
};

console.log(obj.add(1,2));
> 3
console.log(subtract(5,5));
> 0
```

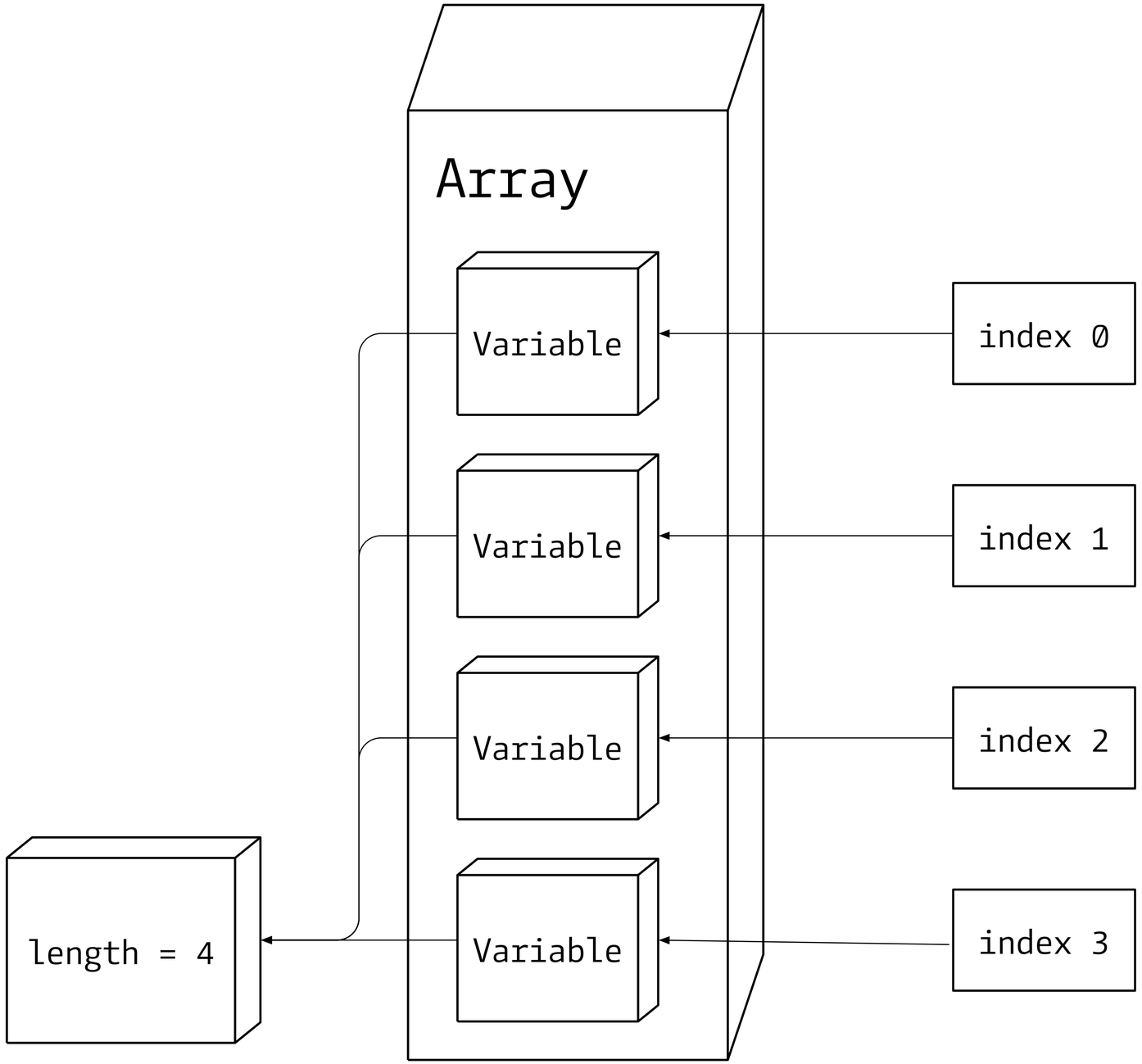

variable
variable
variable
variable
function
function

```
{"x":5}
```

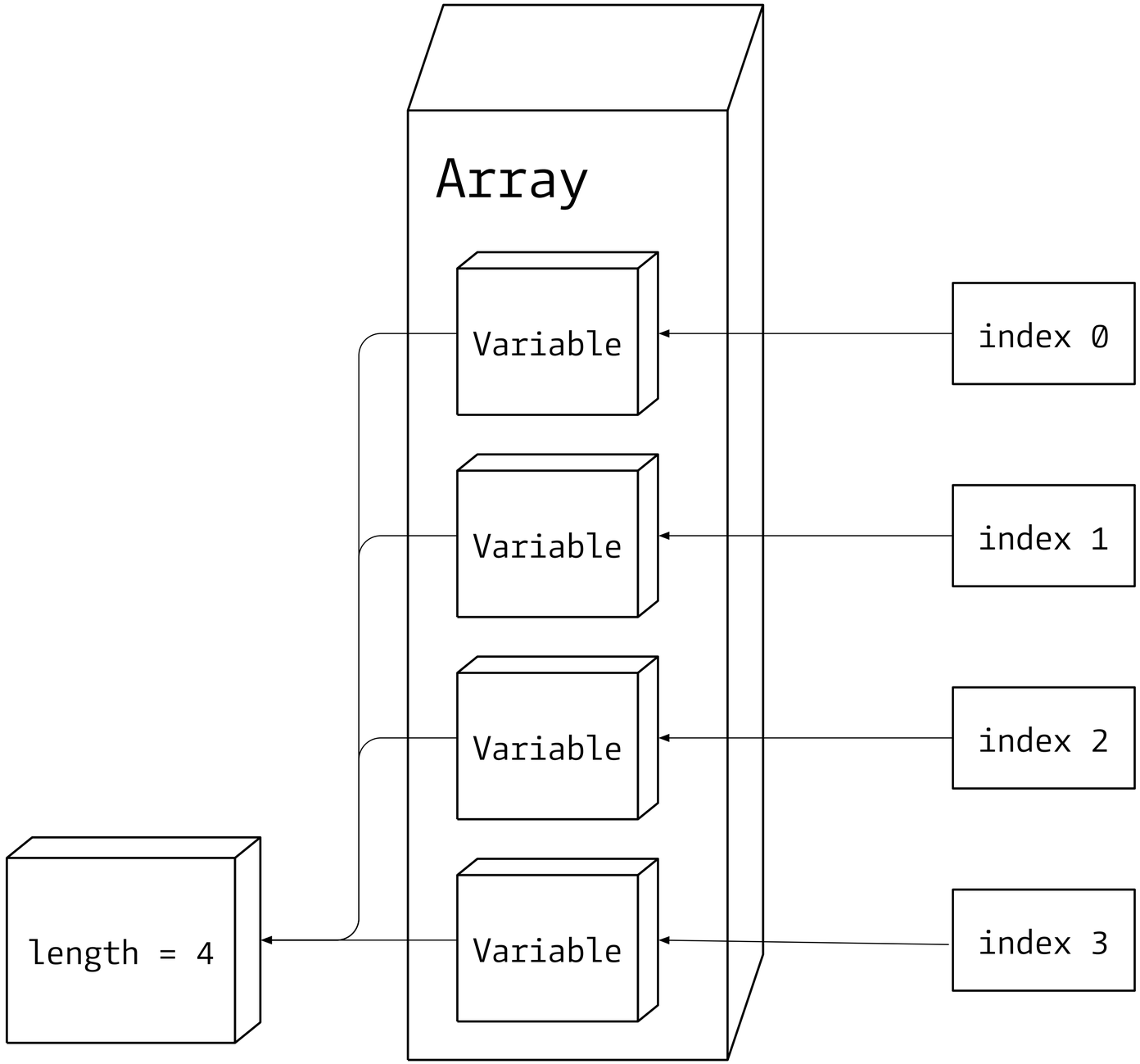

key
value
{"x":5}

7 BASIC THINGS IN PROGRAMMING

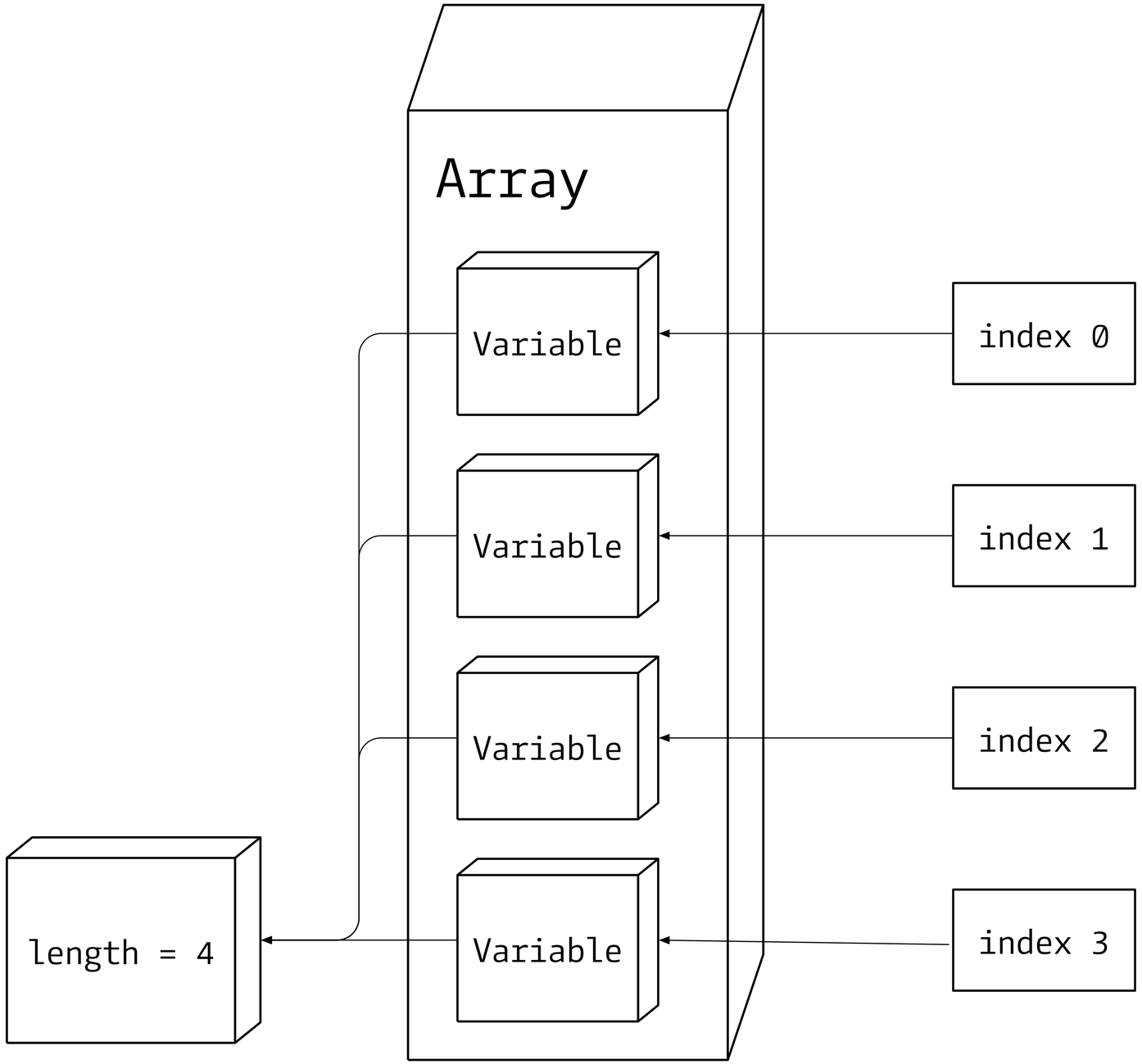
1. Variablen ✓
2. Objekte ✓
3. Arrays
4. Konditionen
5. Schleifen
6. Funktionen
7. Algorithmus


Array
Variable
Variable
Variable
Variable
index 0
index 1
index 2
index 3
length = 4

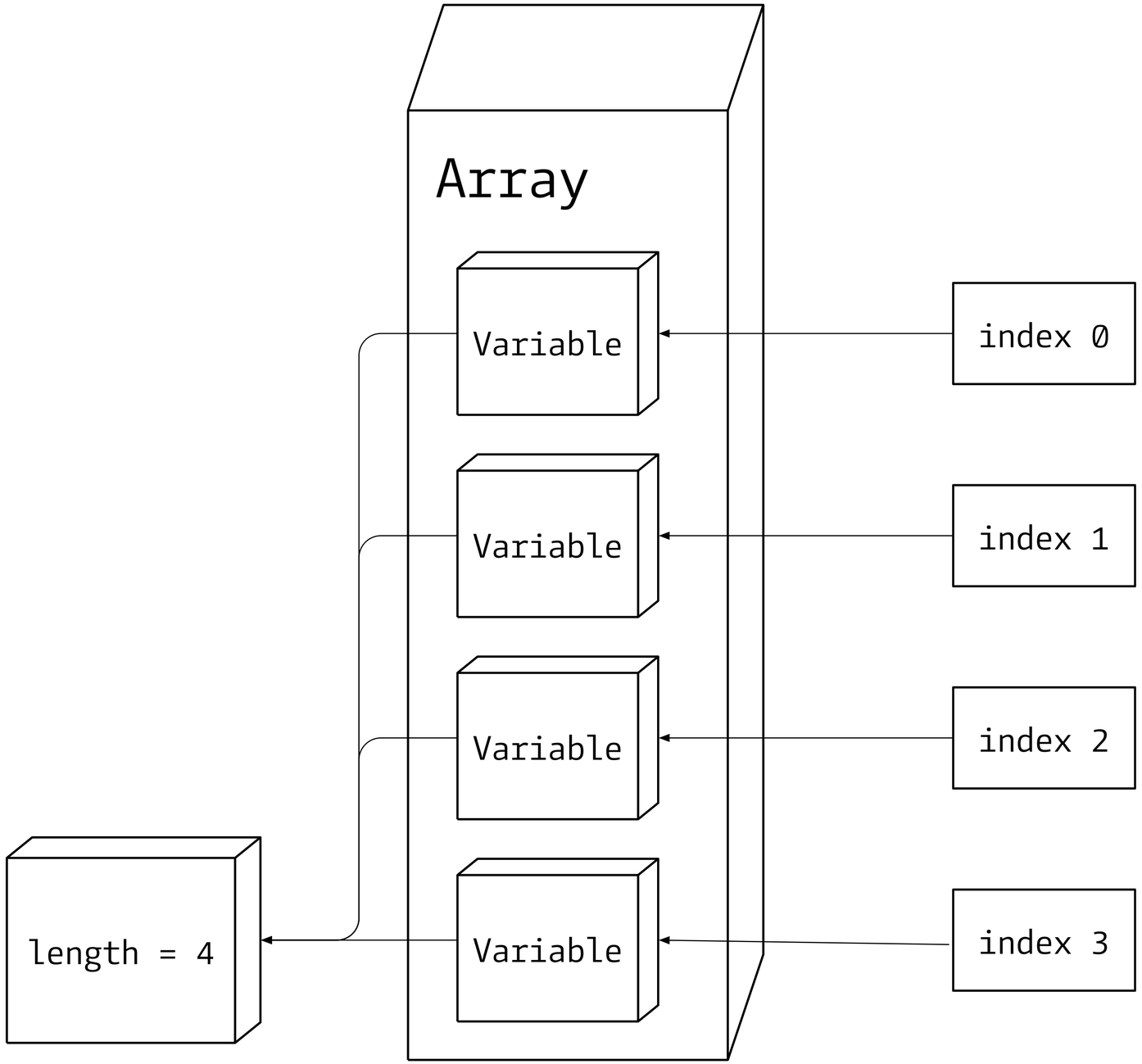
```
var arr = [];
```


Array
Variable
Variable
Variable
Variable
index 0
index 1
index 2
index 3
length = 4

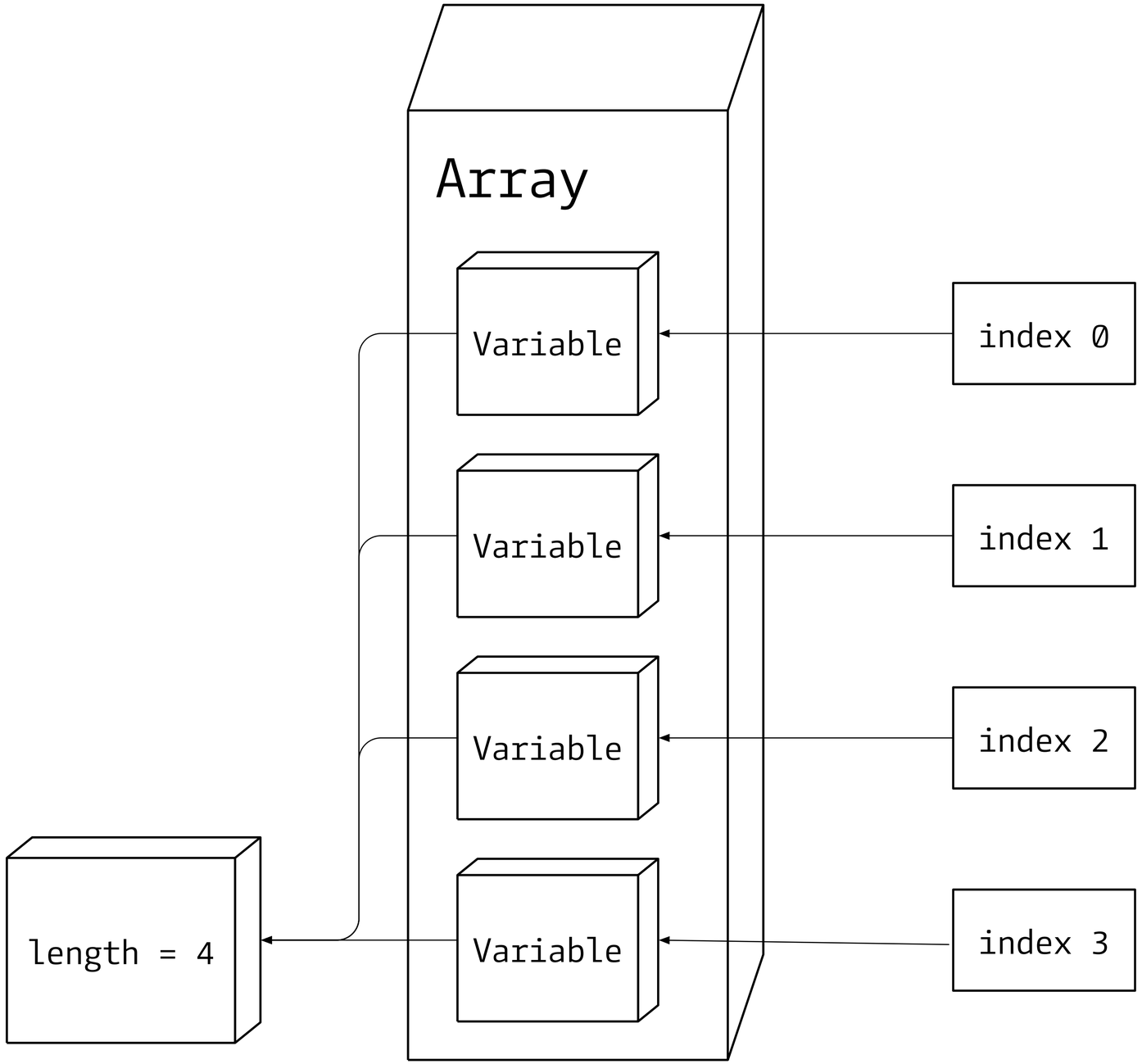
```
var arr = [1,7,3,5];
```



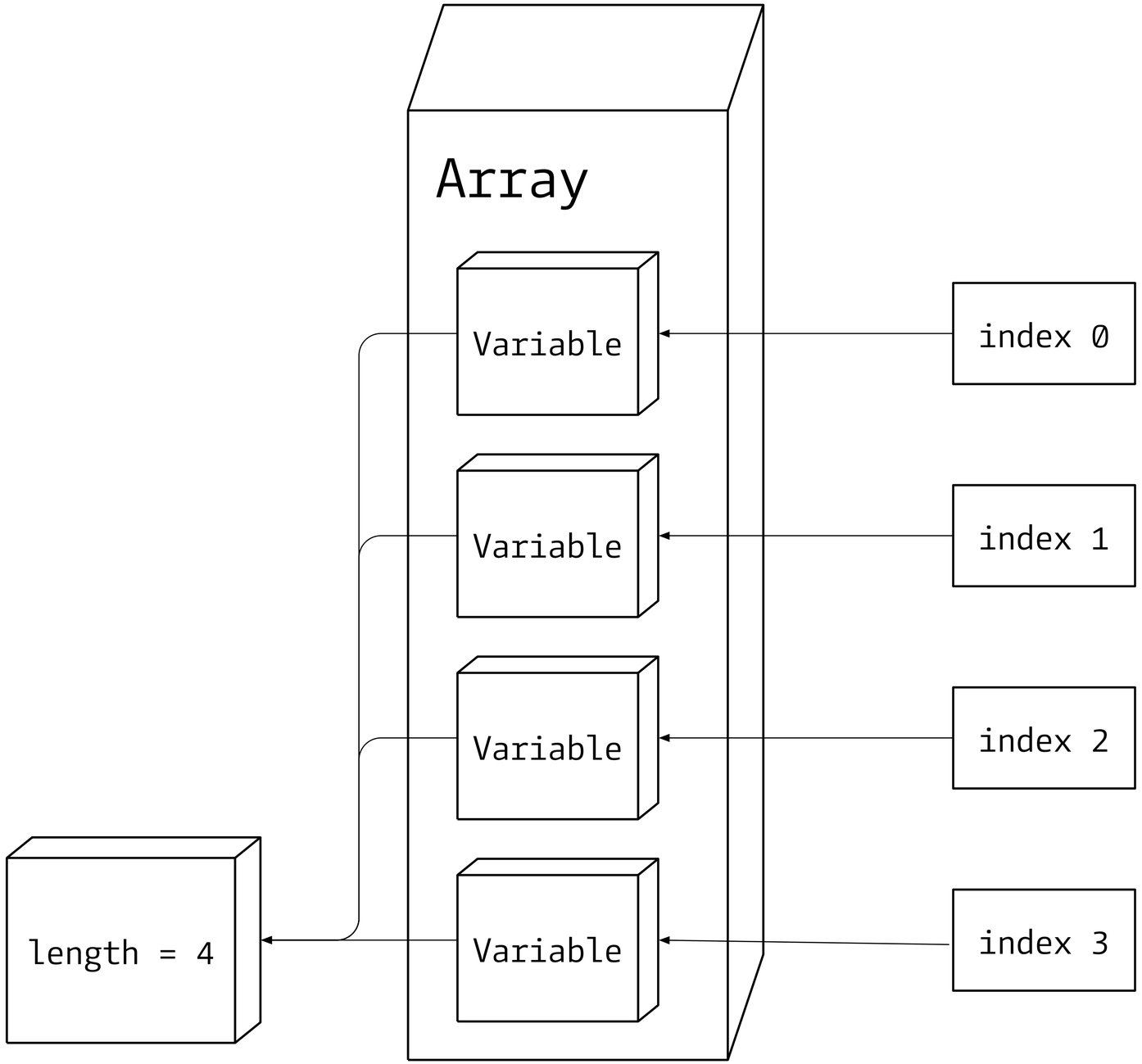

```
var arr = [1,7,3,5];
console.log(arr[2]);
> 3
```



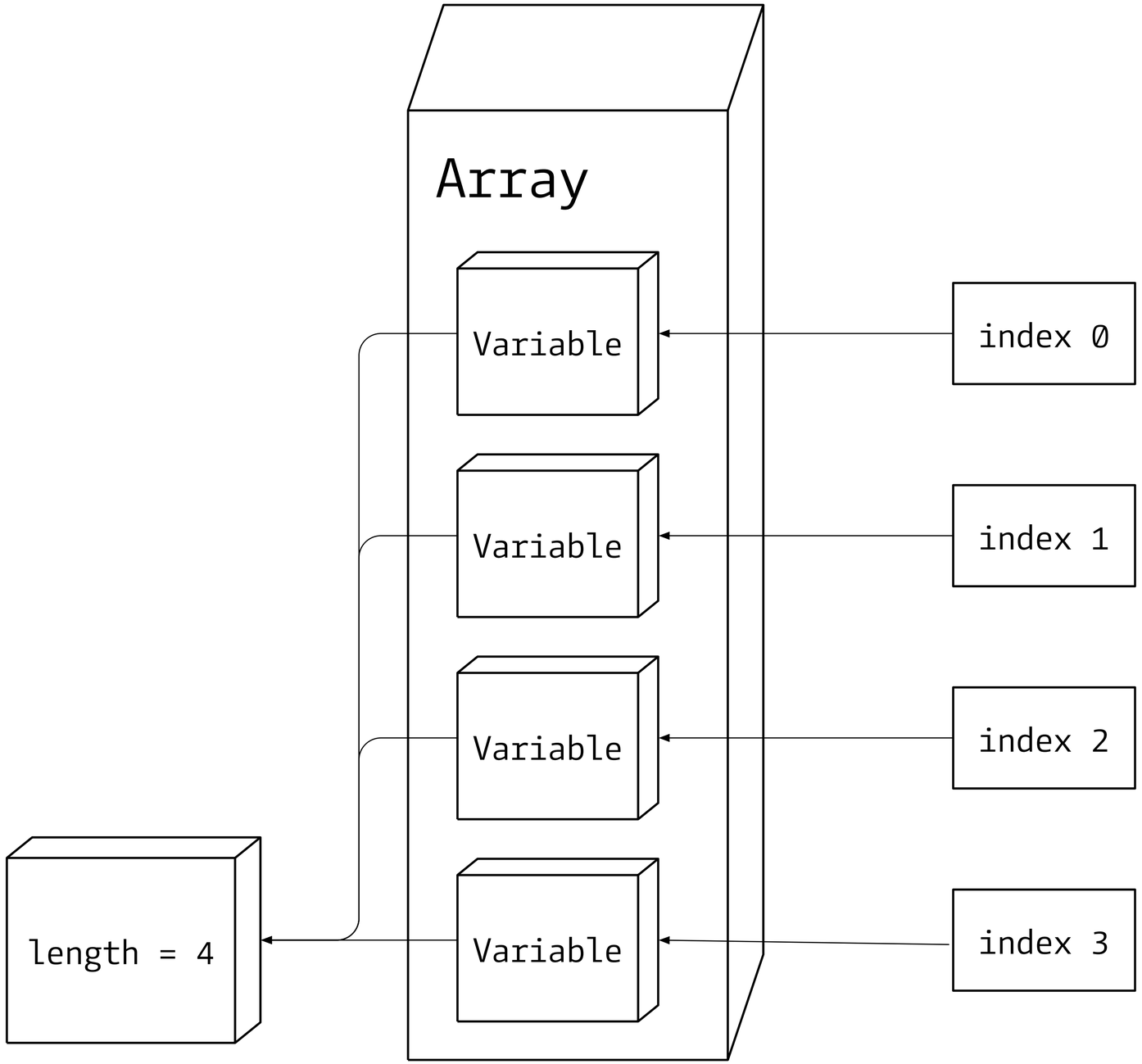

```
var arr = [];

arr[0] = 10;
arr[1] = 5;
arr[2] = 32.5;
arr[3] = -42;

console.log(arr[1]);
> 5
```



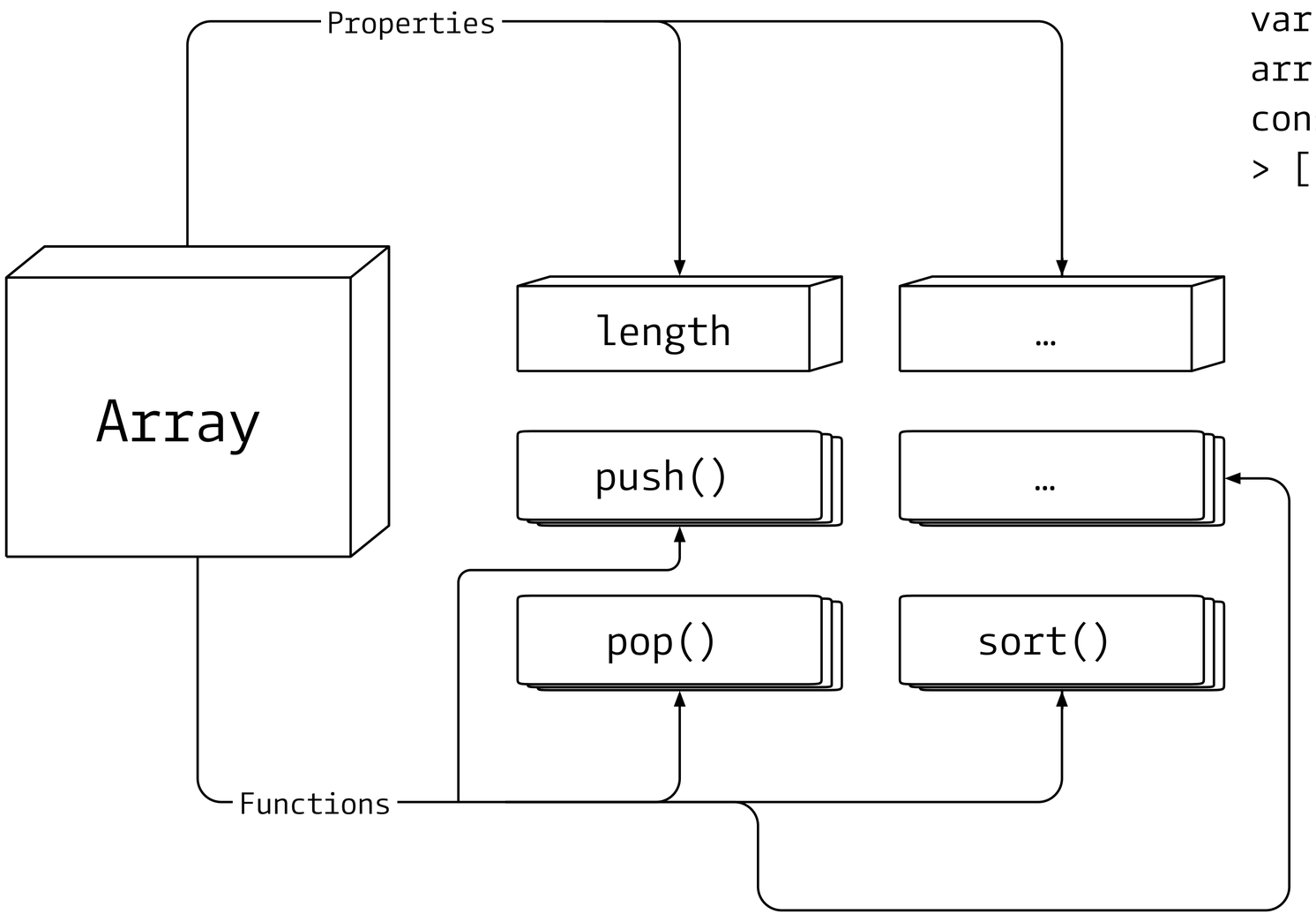

```
var arr = [1,7,3,5];
console.log(arr.length);
```




```
var arr = [1,7,3,5];
console.log(arr.length);
> 4
```



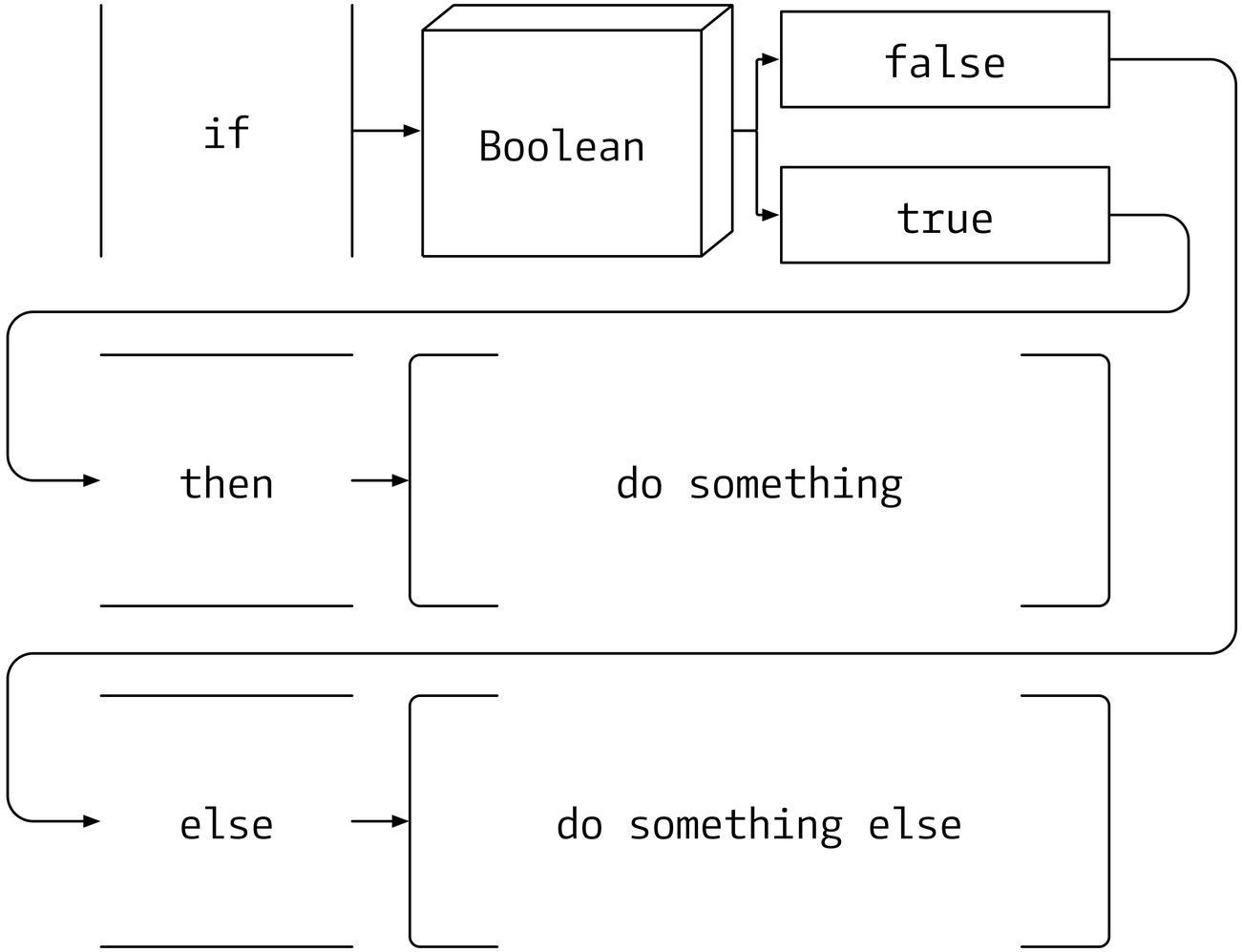

```
var arr = [1,7,3,5];
var last = arr[arr.length -1];
console.log(last);
> 5
```

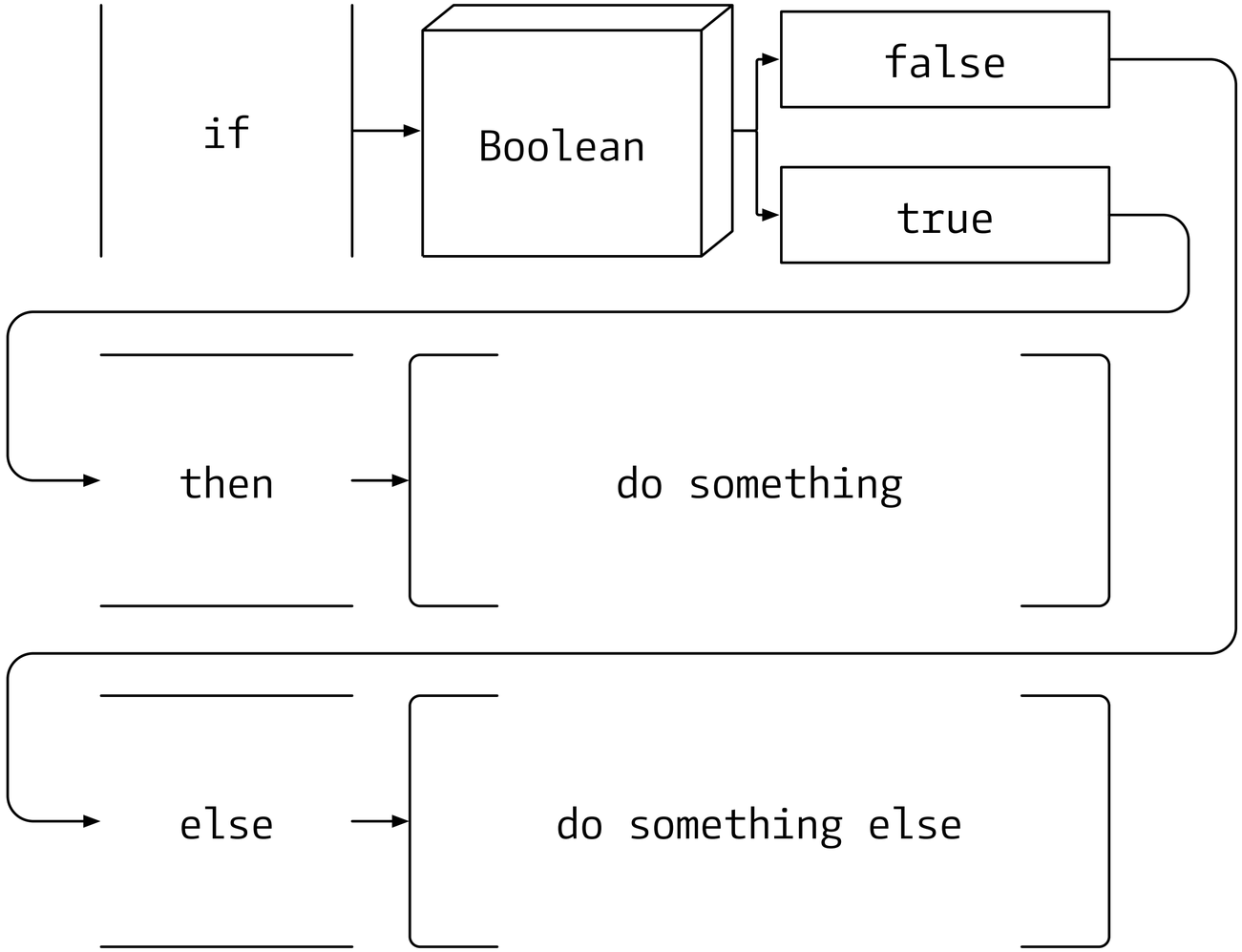



```
var arr = [1,7,3,5];
arr.push(23);
console.log(arr);
> [ 1, 7, 3, 5, 23 ]
```

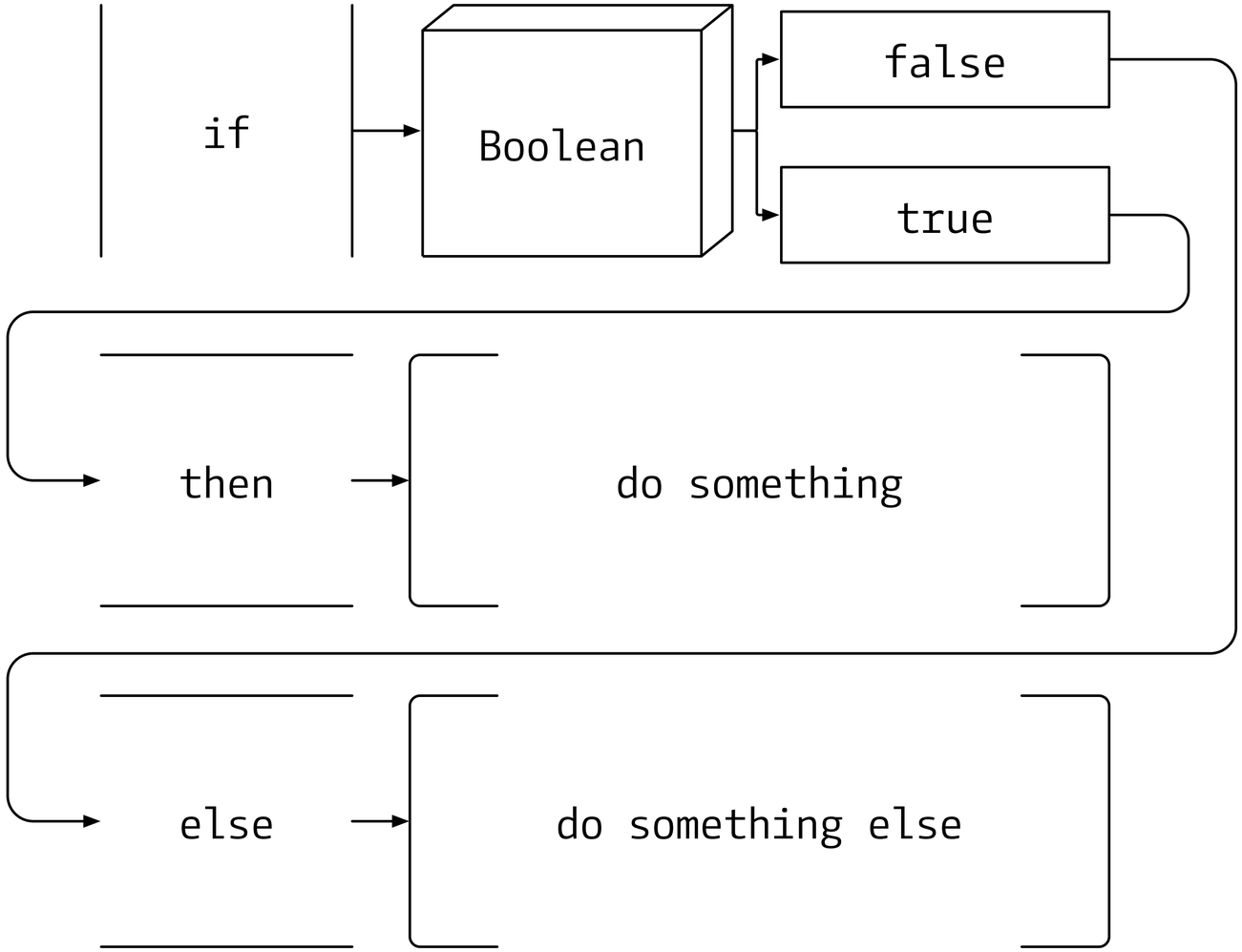
7 BASIC THINGS IN PROGRAMMING

1. Variablen ✓
2. Objekte ✓
3. Arrays ✓
4. Konditionen
5. Schleifen
6. Funktionen
7. Algorithmus

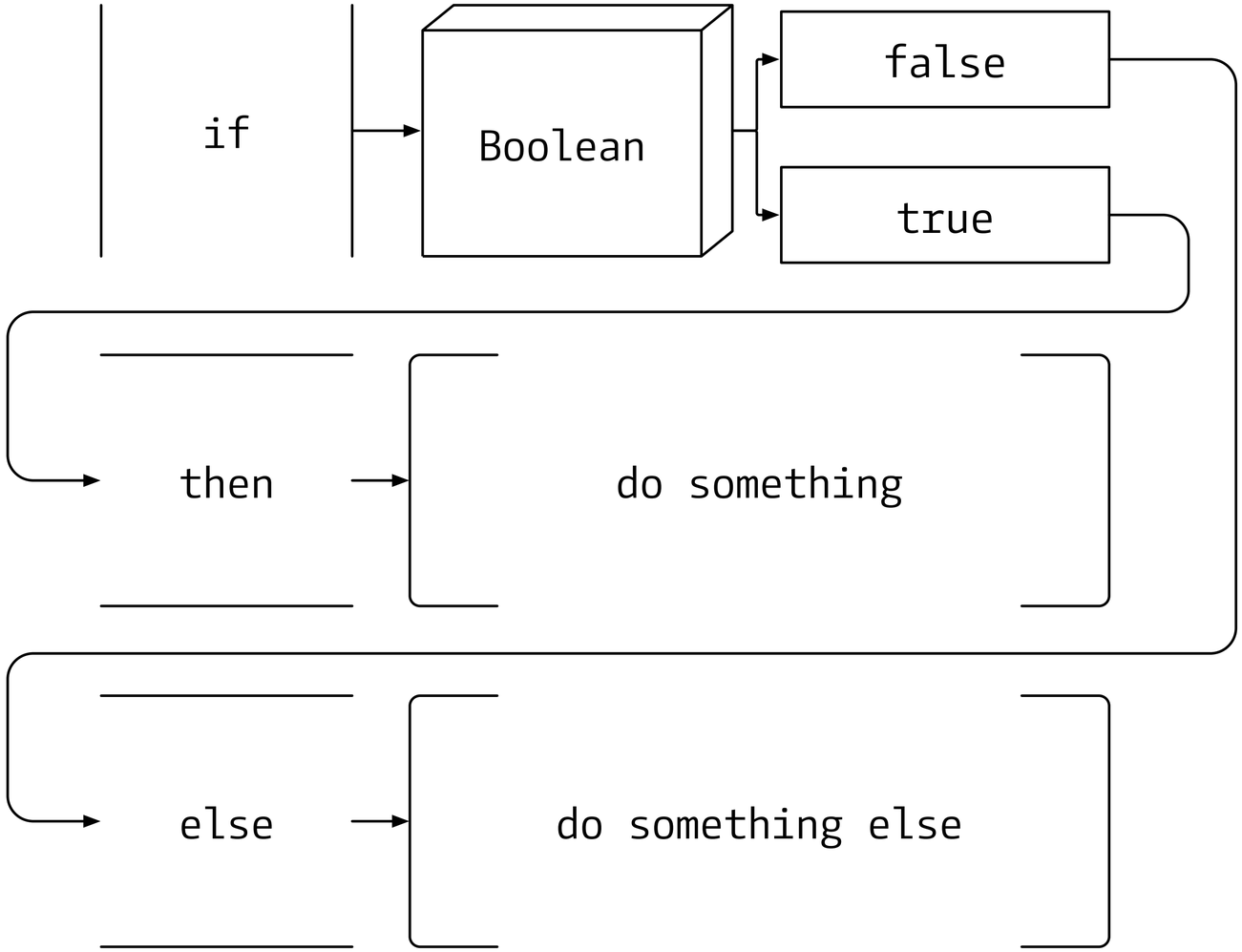

if
Boolean
false
true
then
do something
else
do something else



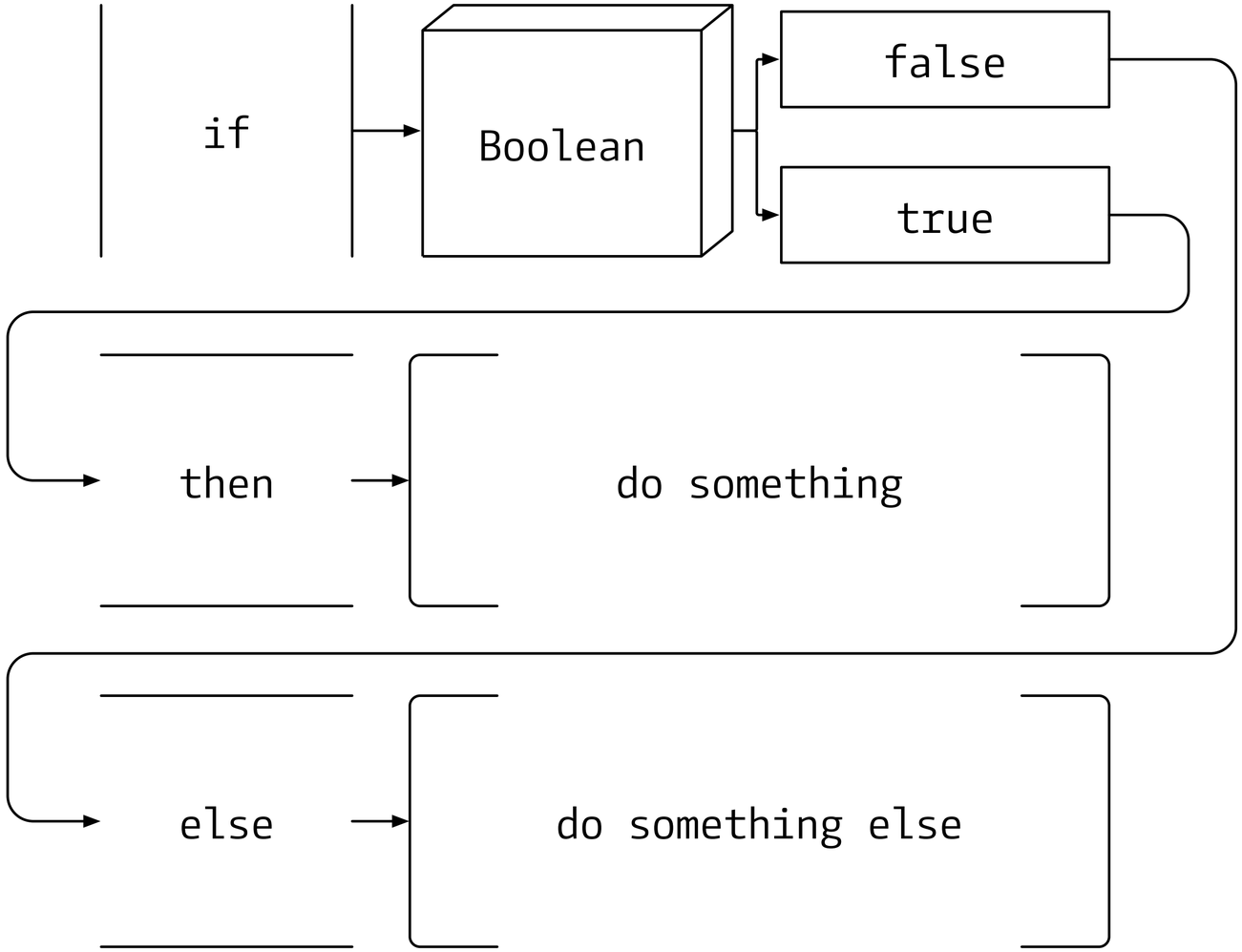

```
if
    dog chases cat
        open the garden door
```



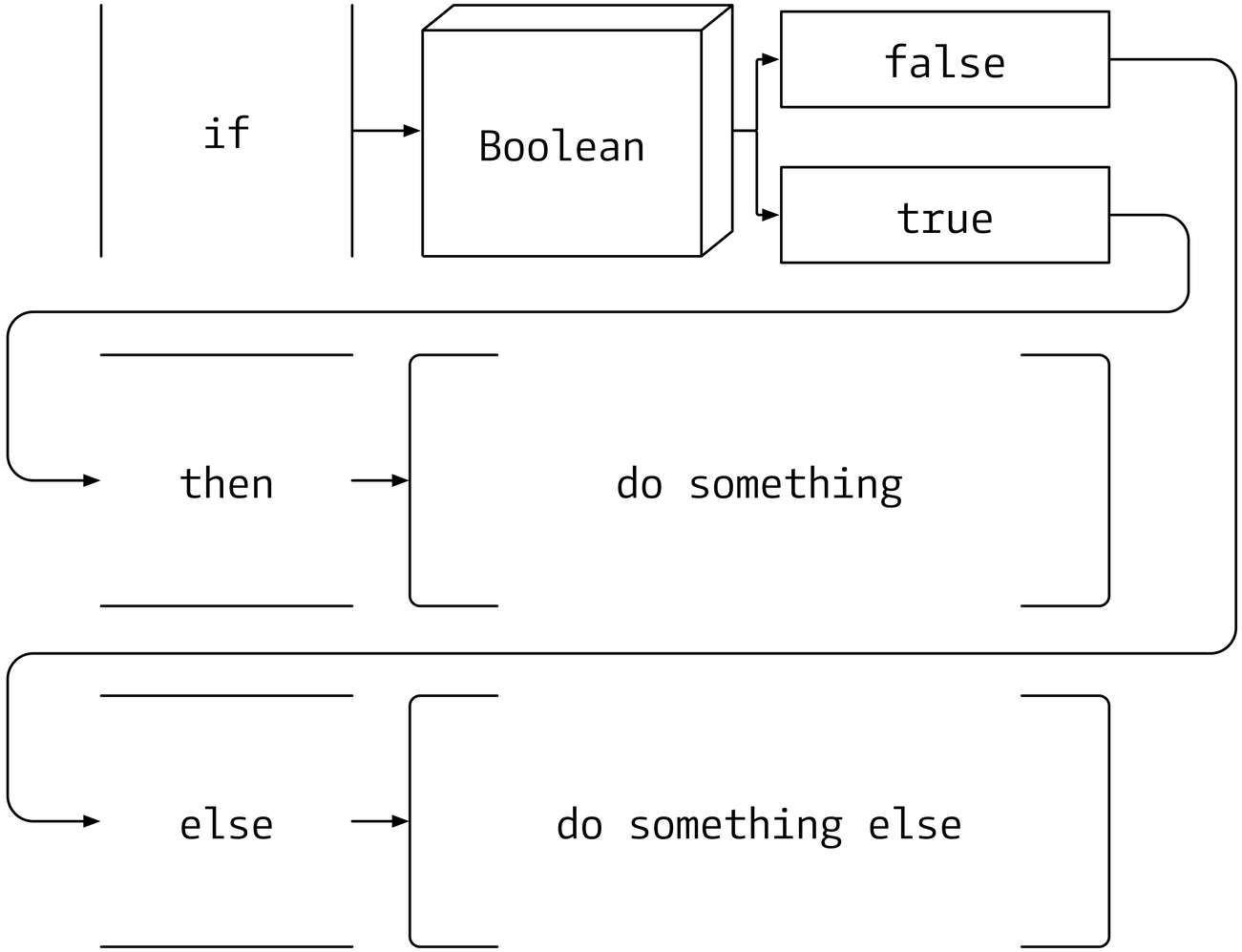

```
if
    dog chases cat
    and
    and they are inside
        open the garden door


// and, or, not
```



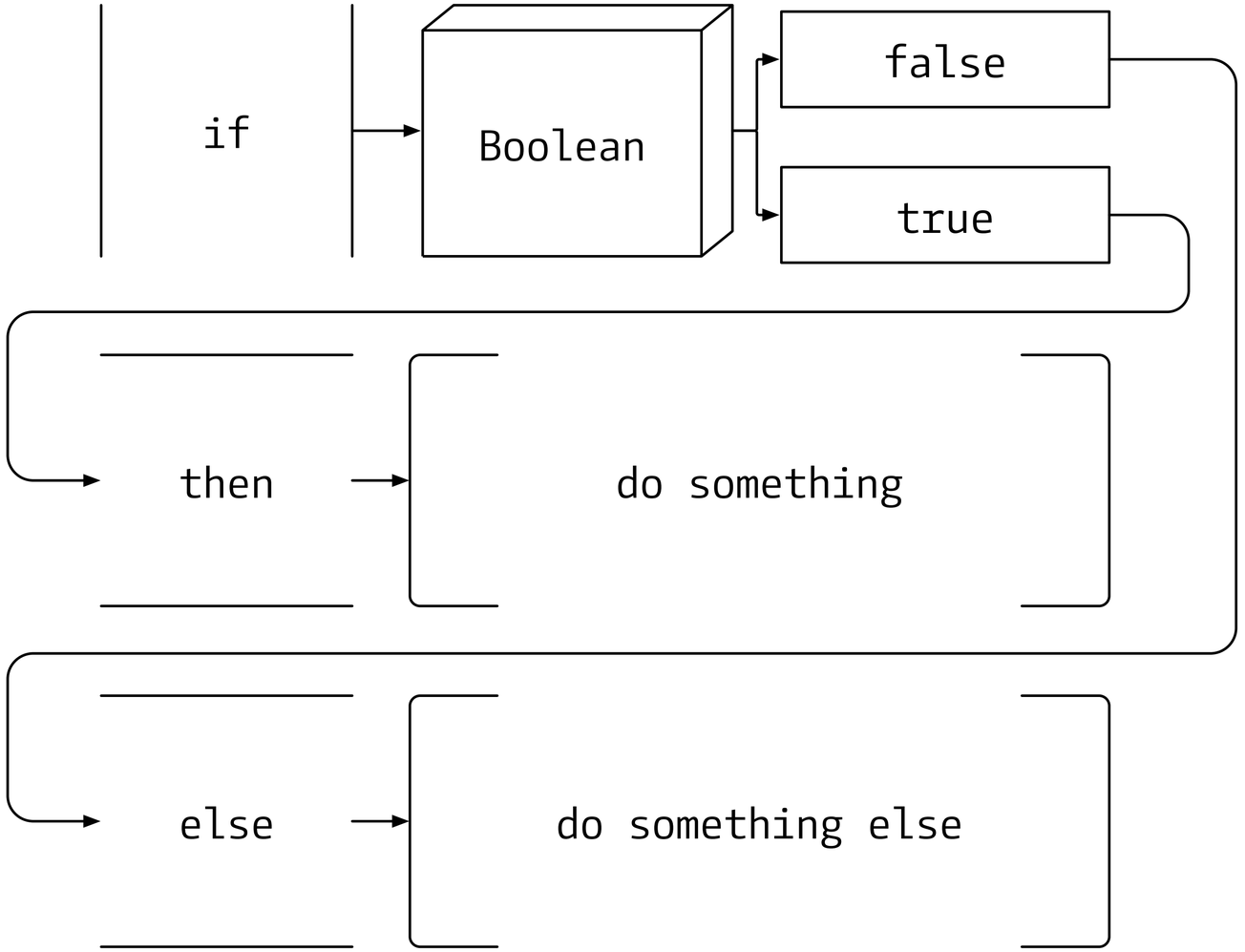

```
if
     x is smaller than y
          do this
```


if
Boolean
false
true
then
do something
else
do something else

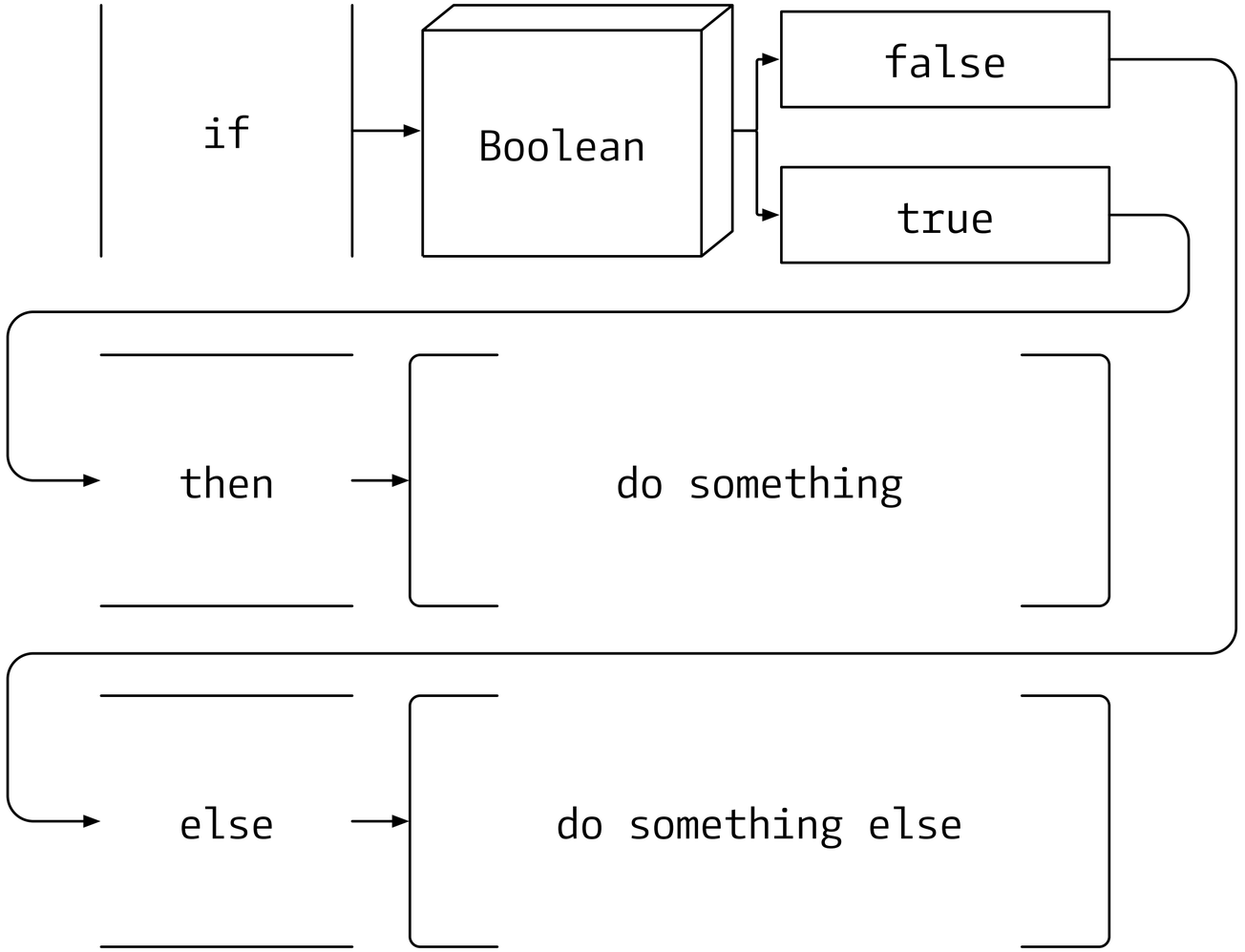
```
if
    x < y
        do this
```



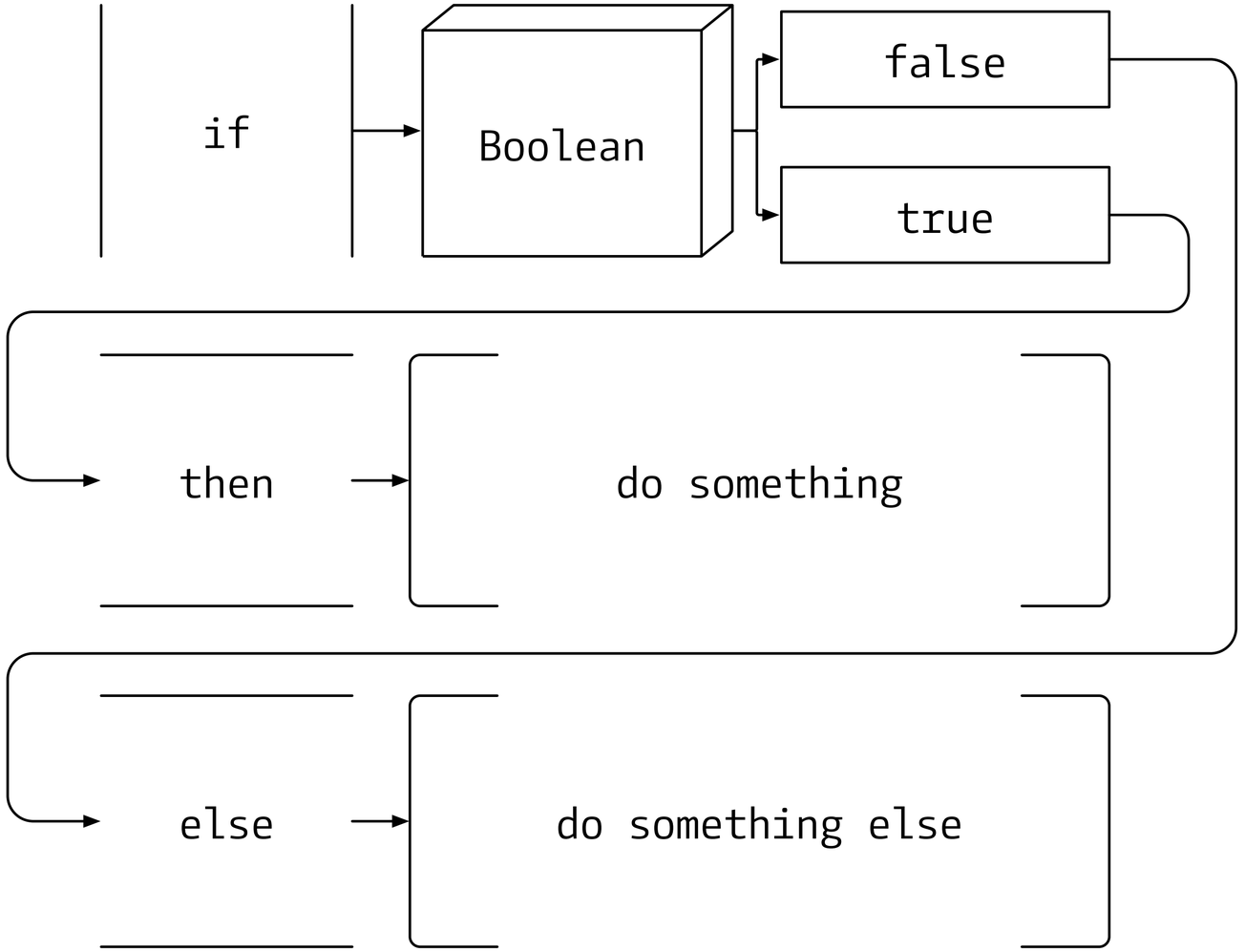

```
if
    x < y
        do this
else
        do that
```



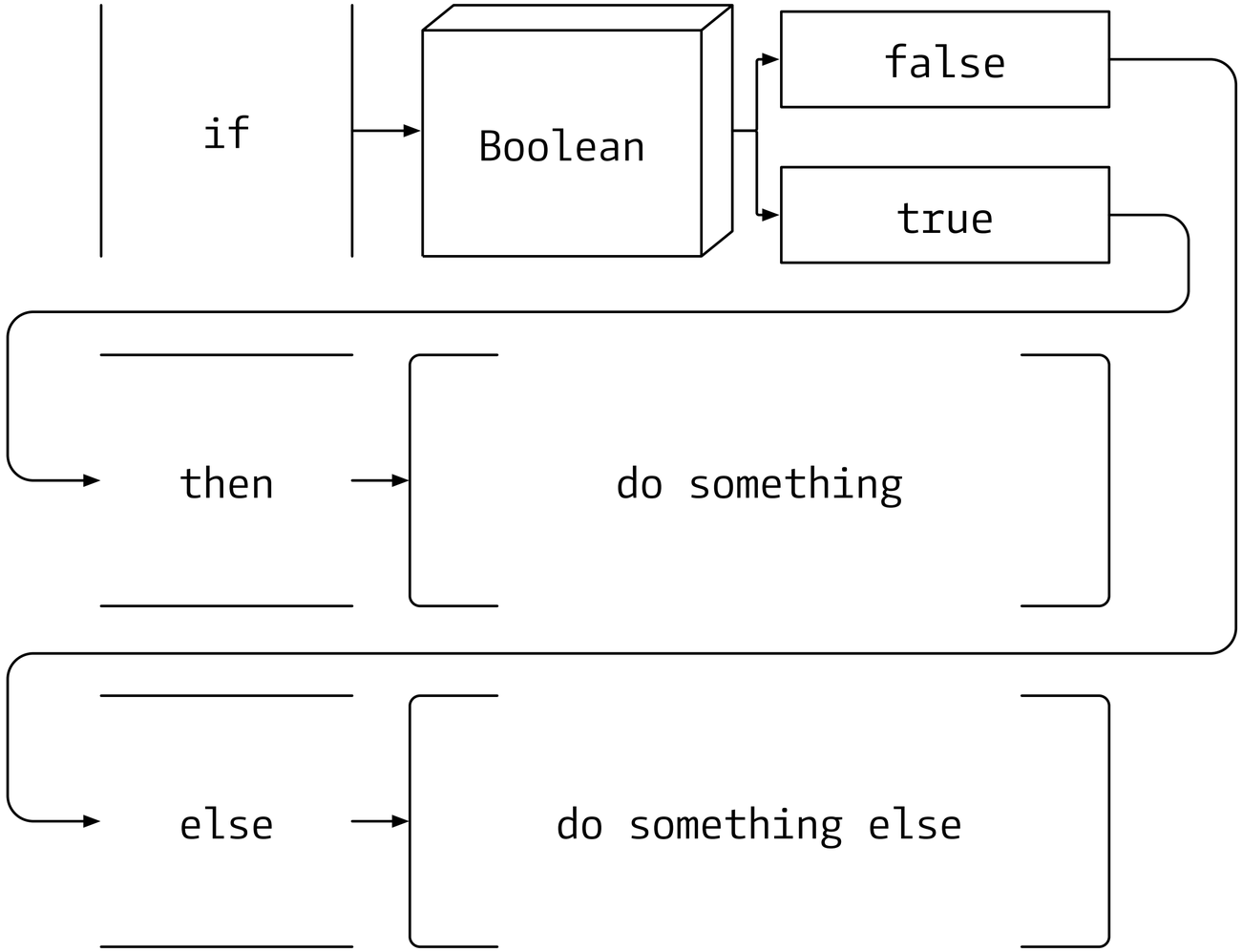

```
x = 10
y = 20
if
    x < y
        do this
else
        do that
```



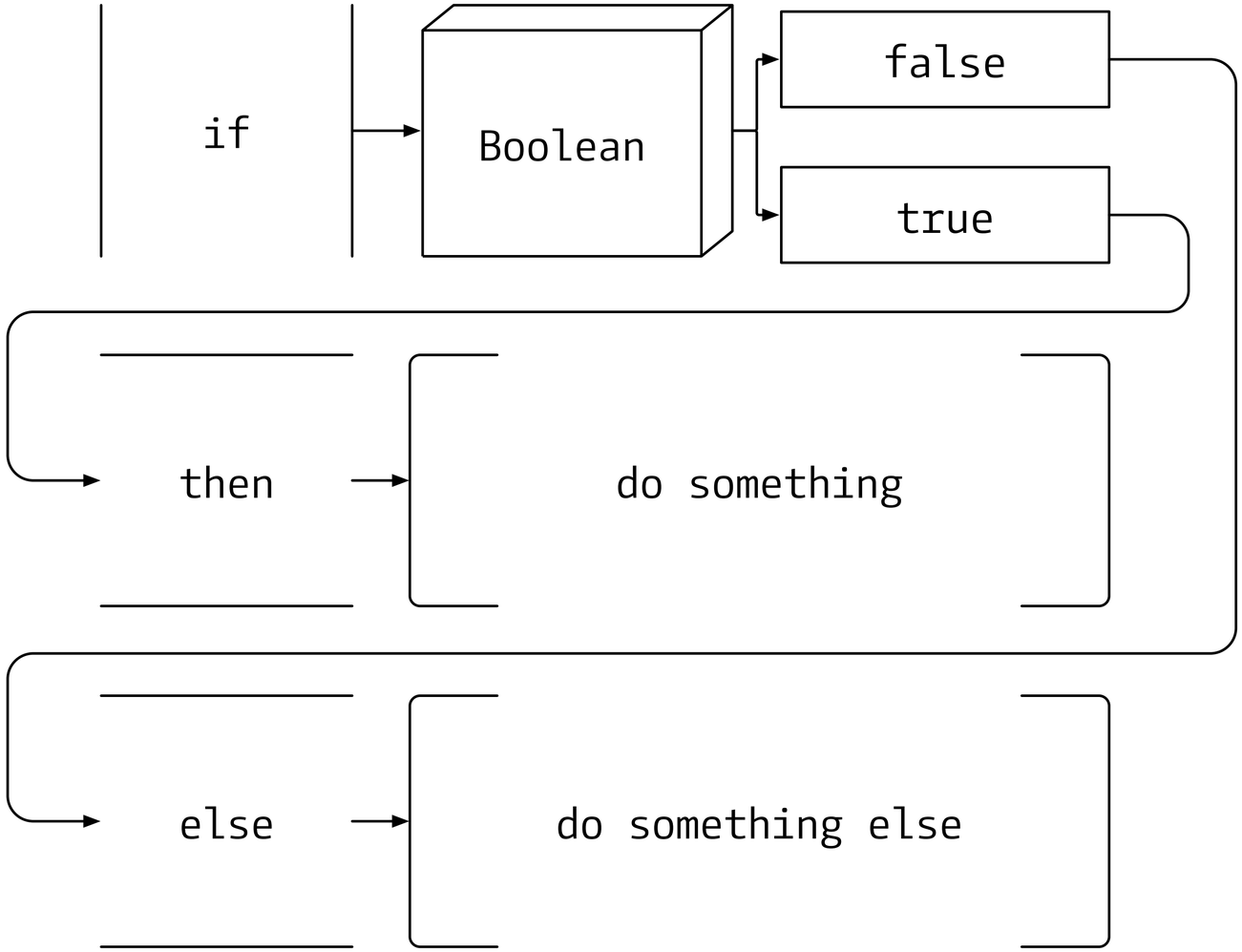

```
x = 42
y = 55
if
    x < y
        do this
else
        do that
```



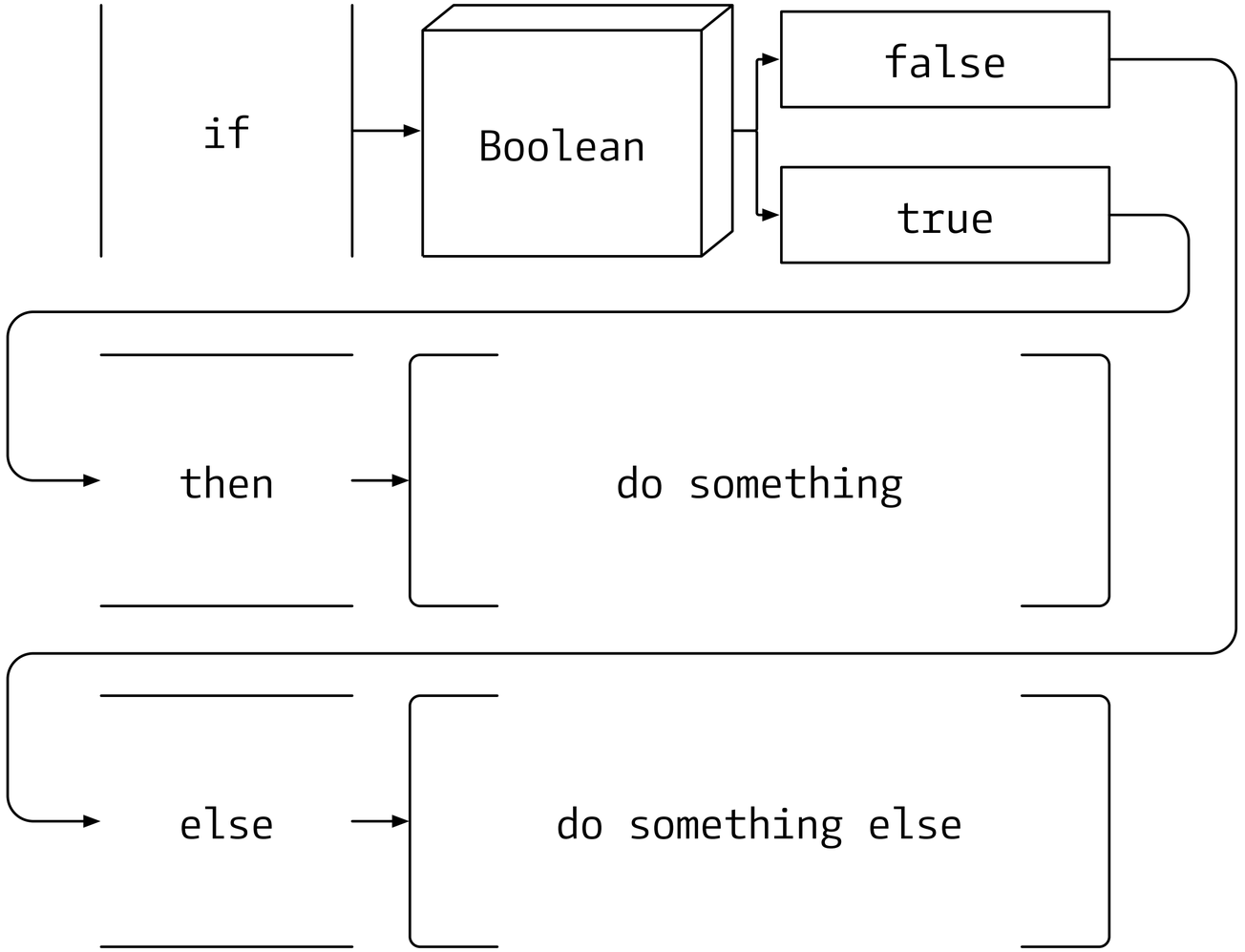

```
x = 42
y = 23
if
    x < y
        do this
else
        do that
```



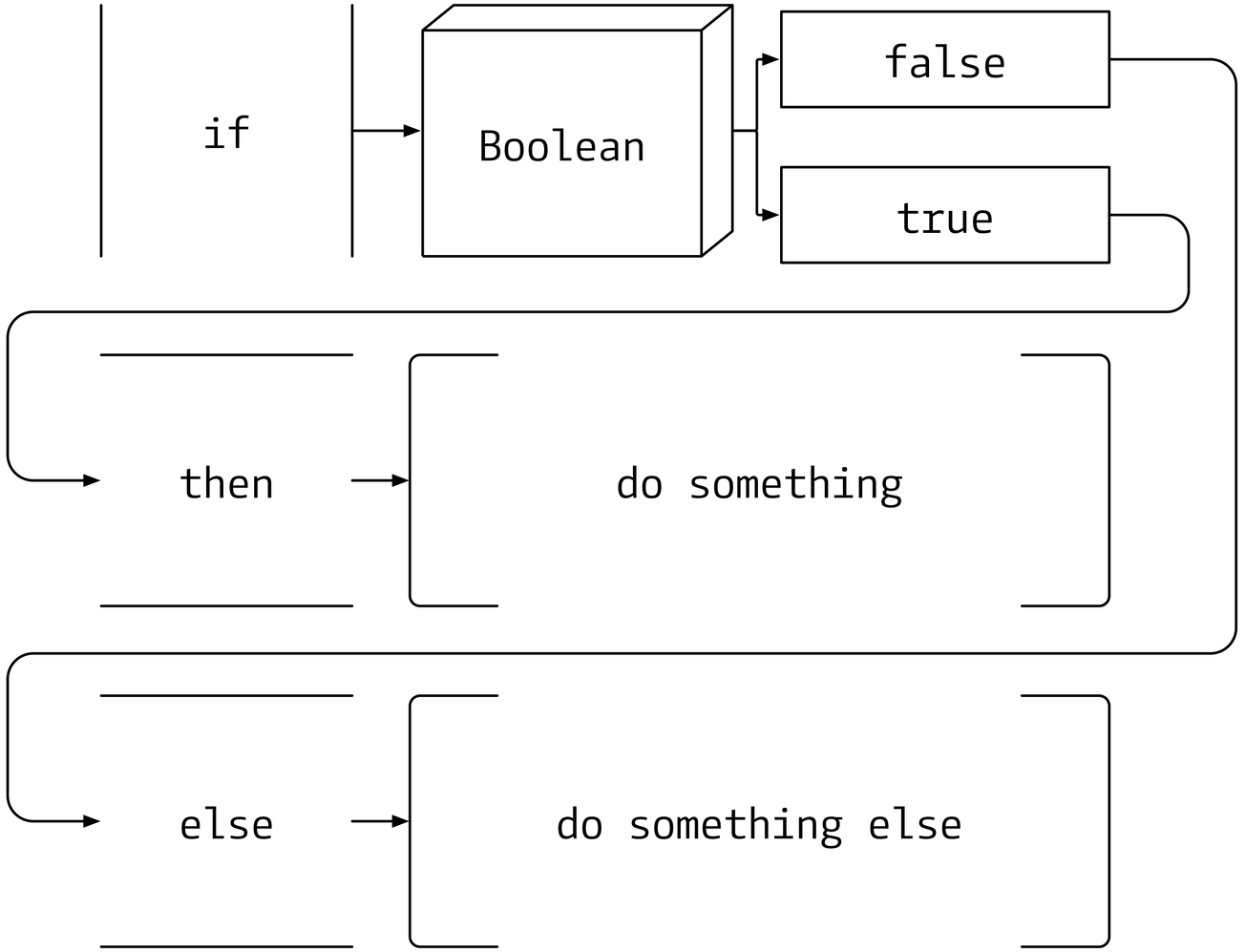

```
if
    x is smaller than y
    and
    x is smaller then 10
        do this
```



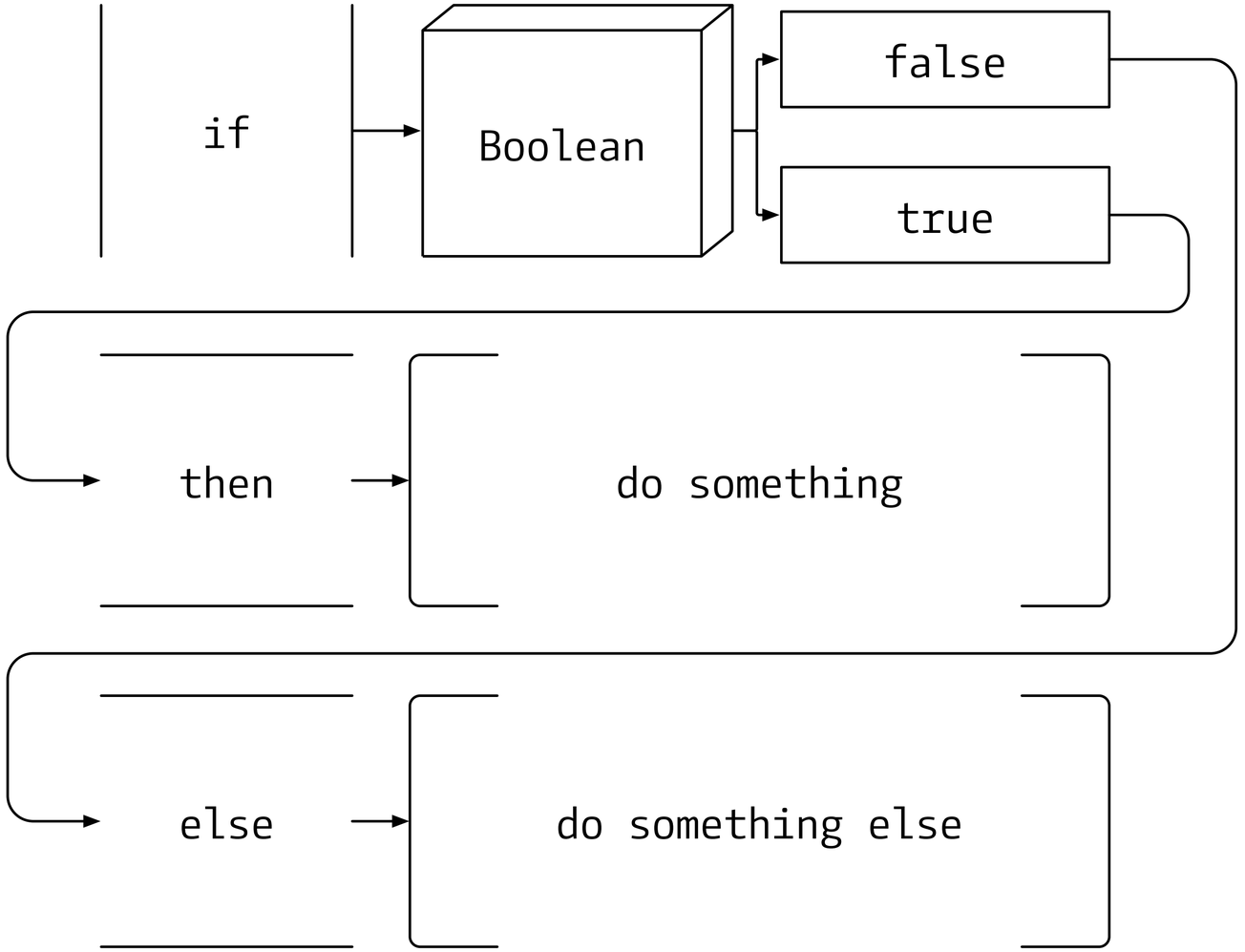

```
if
    x < y && x < 10
        do this
```



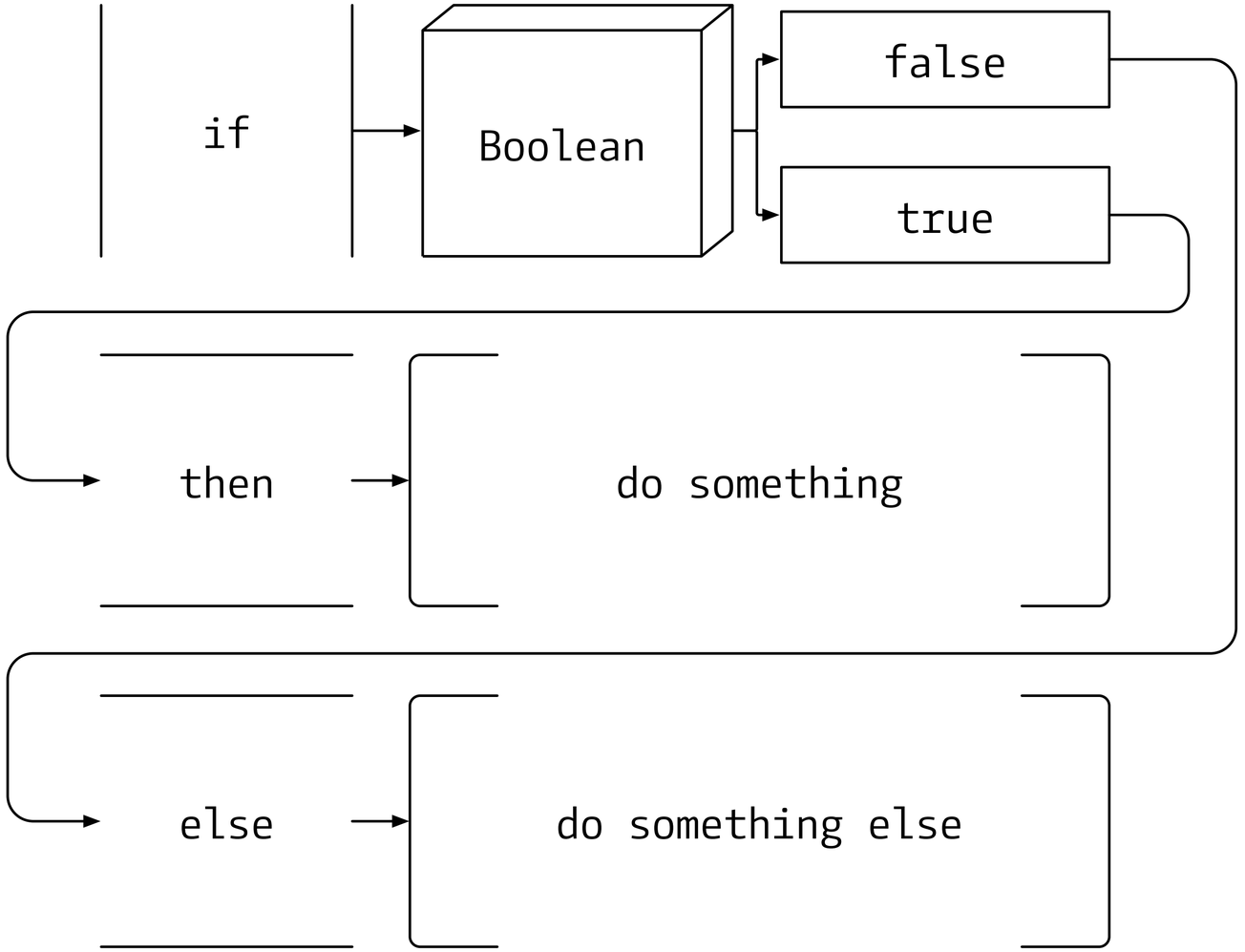

```
if
    x < y && x < 10
        do this
else
        do that
```



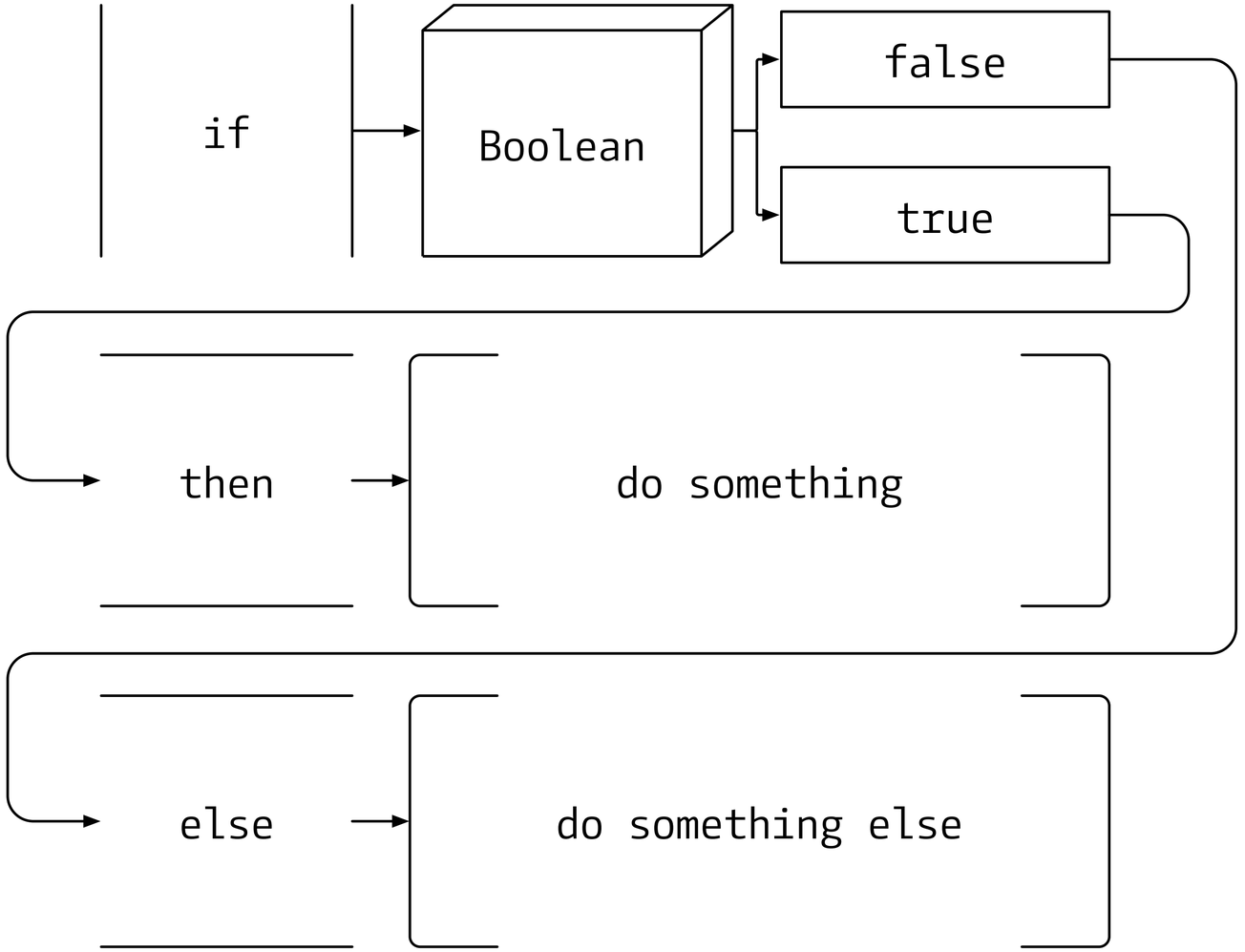

```
x = 5
y = 3
if
    x < y && x < 10
        do this
else
        do that
```



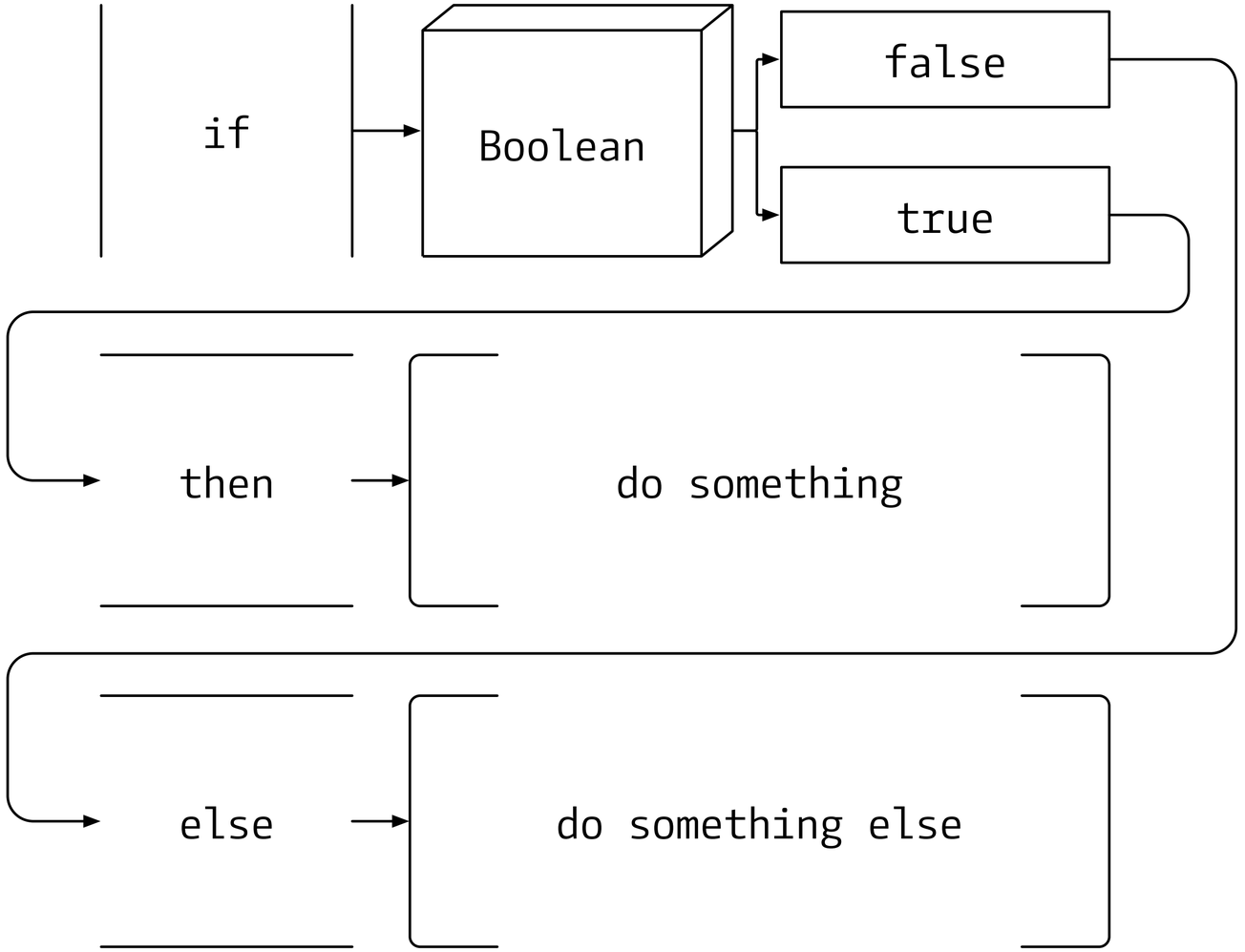

```
x = 11
y = 12
if
    x < y && x < 10
        do this
else
        do that
```



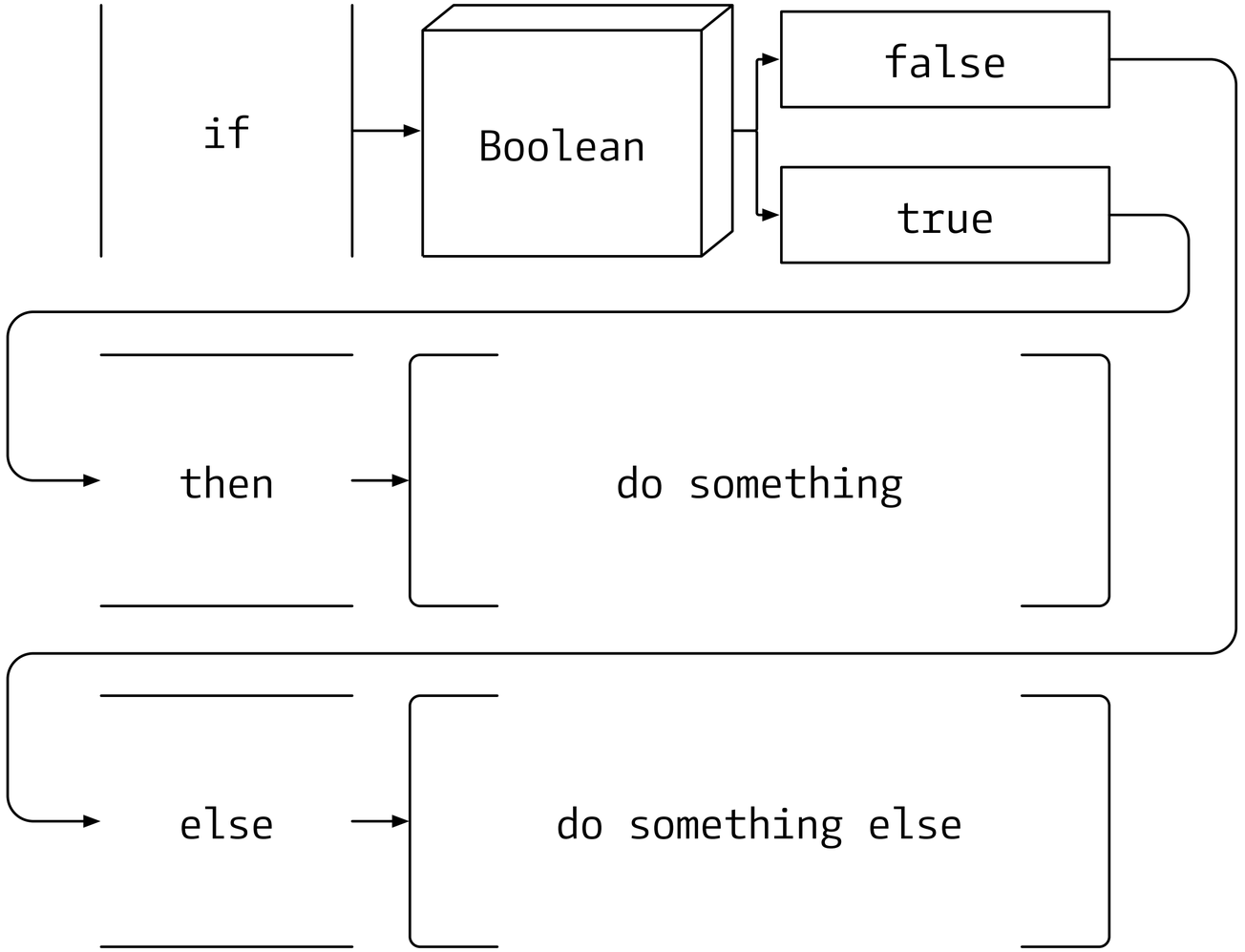

```
x = 2
y = 10
if
    x < y && x < 10
        do this
else
        do that
```



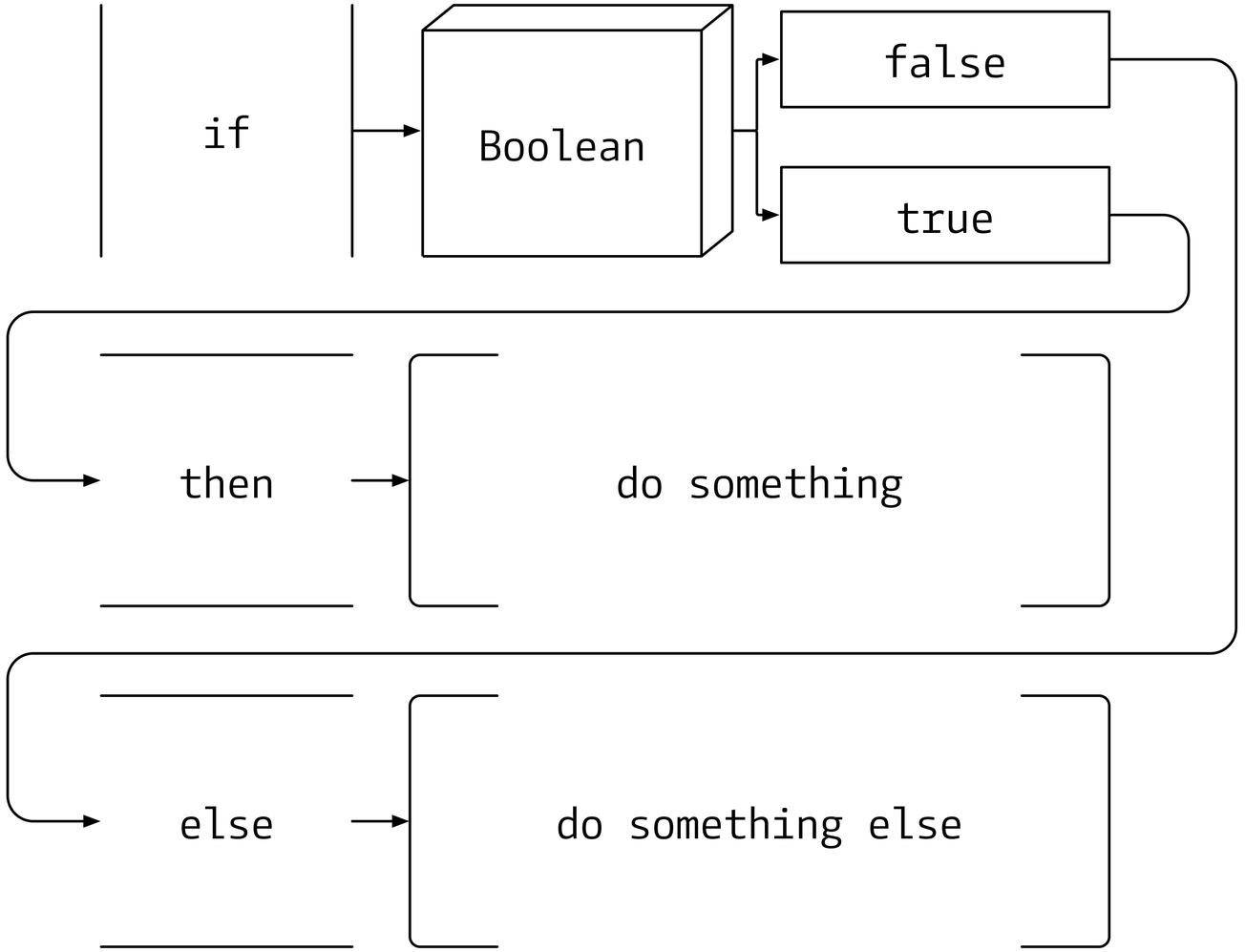

```
if
    x < y && x < 10
        do this
else if
    x < y && x > 10
        do that

else
        do something
```



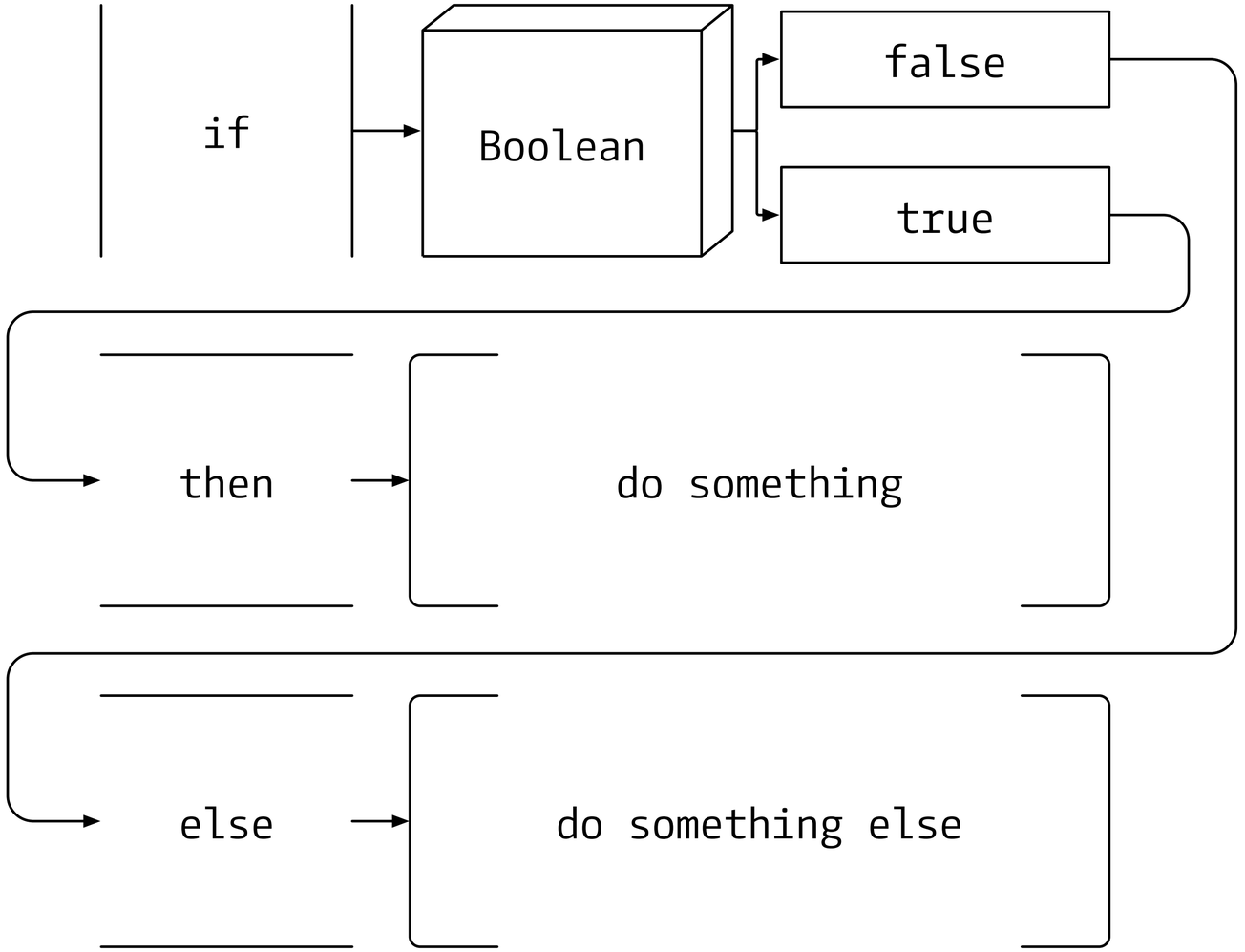

```
x = 23
y = 42
if
    x < y && x < 10
        do this
else if
    x < y && x > 10
        do that

else
        do something
```




```
x = 5
y = 42
if
    x < y && x < 10
        do this
else if
    x < y && x > 10
        do that

else
        do something
```

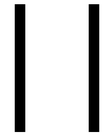



```
x = 42
y = 5
if
    x < y && x < 10
        do this
else if
    x < y && x > 10
        do that

else
        do something
```




```
if(x < y && x < 10){
        // do this

} else if (x < y && x > 10){
        //do that

}else{
        // do something

}
```

&&



```
x > 5 && y < 10
```

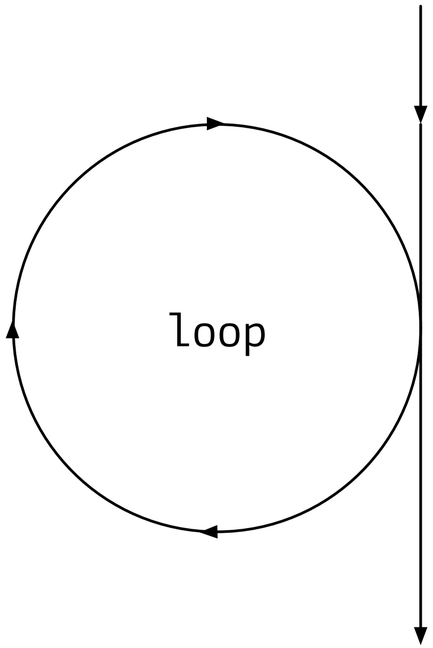
```
x > 5 || y < 10
```

```
x >= 5 || y =< 10
```

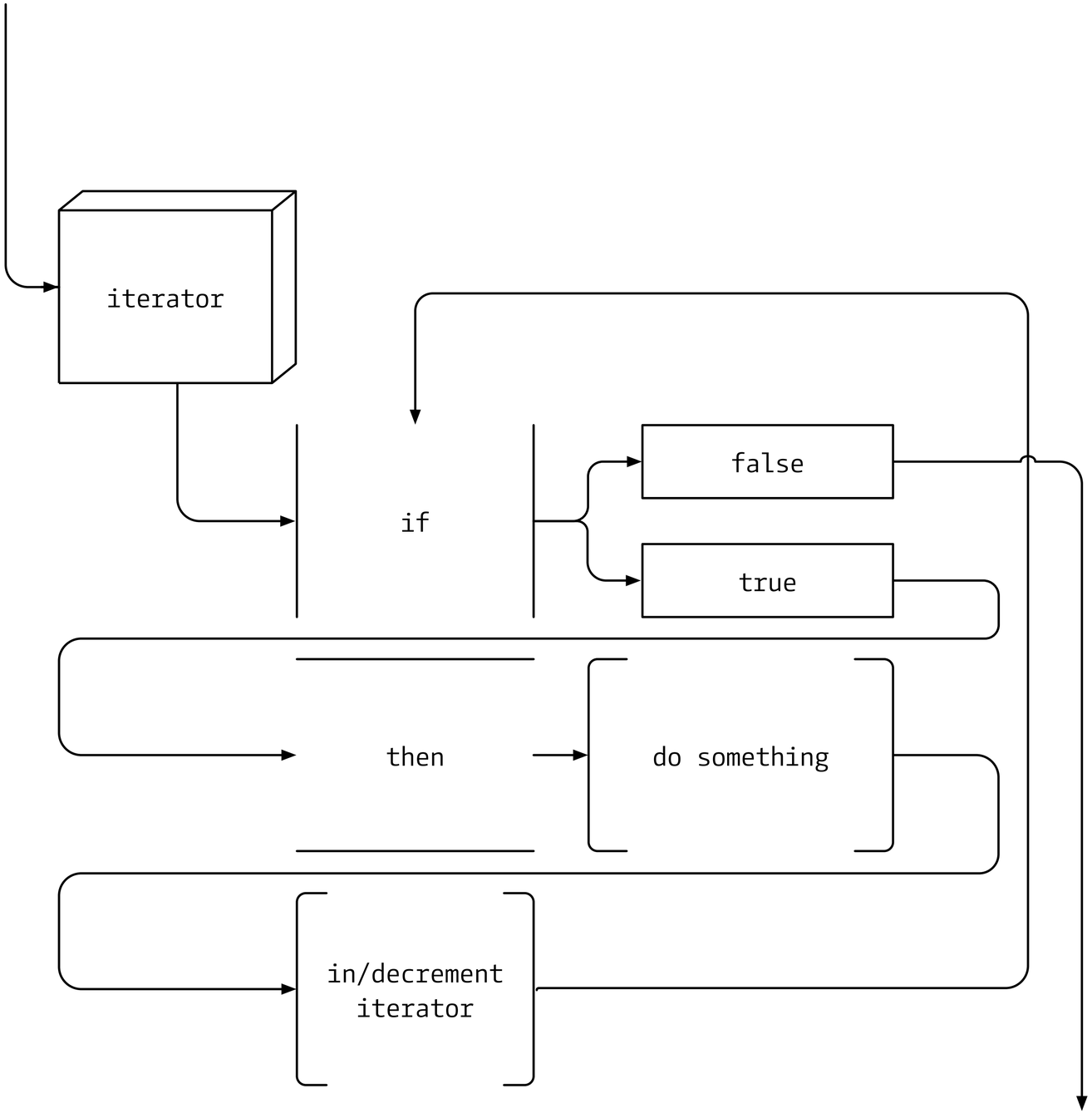
```
x !== 5 || y === 10
```

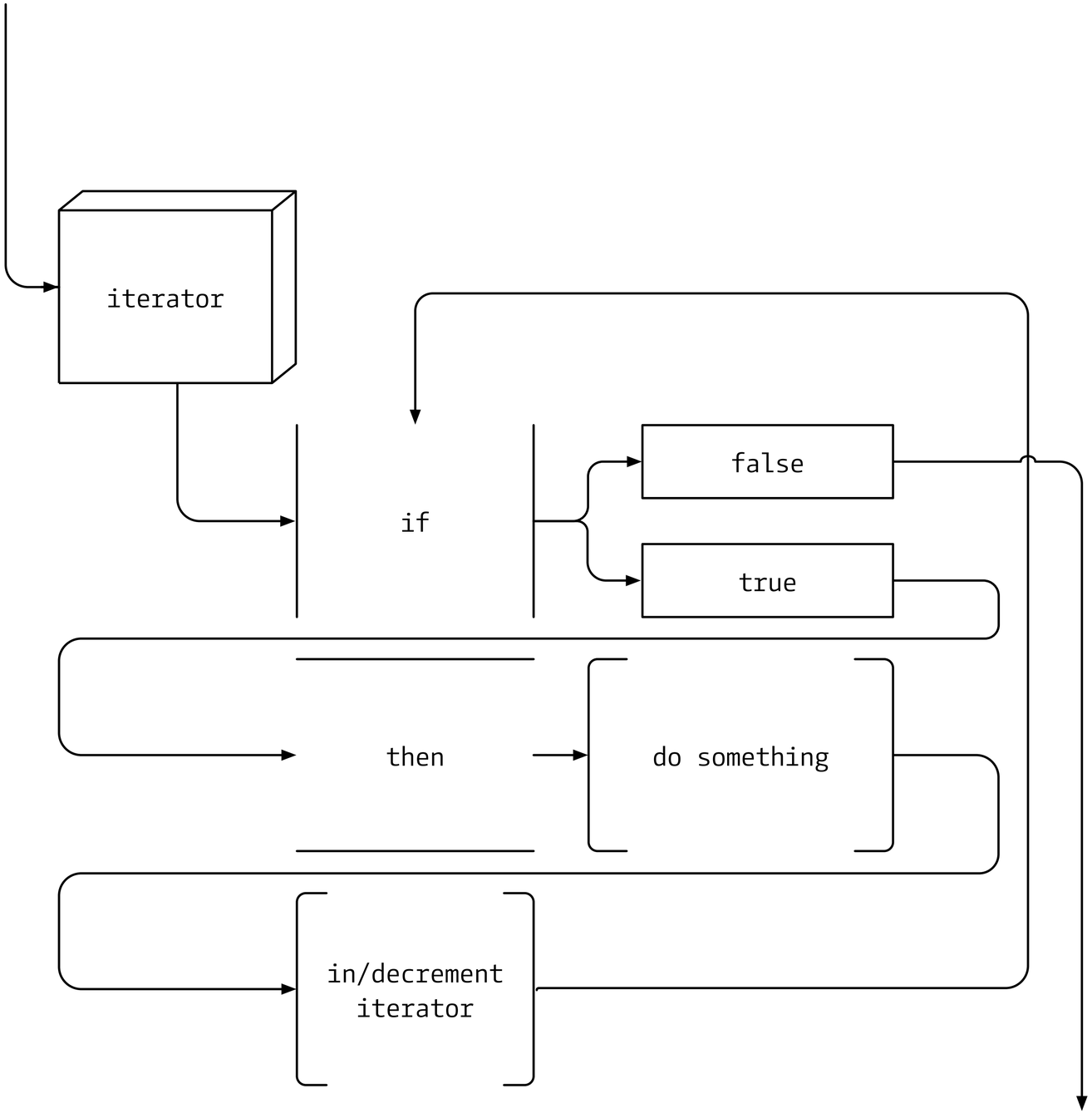
7 BASIC THINGS IN PROGRAMMING

1. Variablen ✓
2. Objekte ✓
3. Arrays ✓
4. Konditionen ✓
5. Schleifen
6. Funktionen
7. Algorithmus

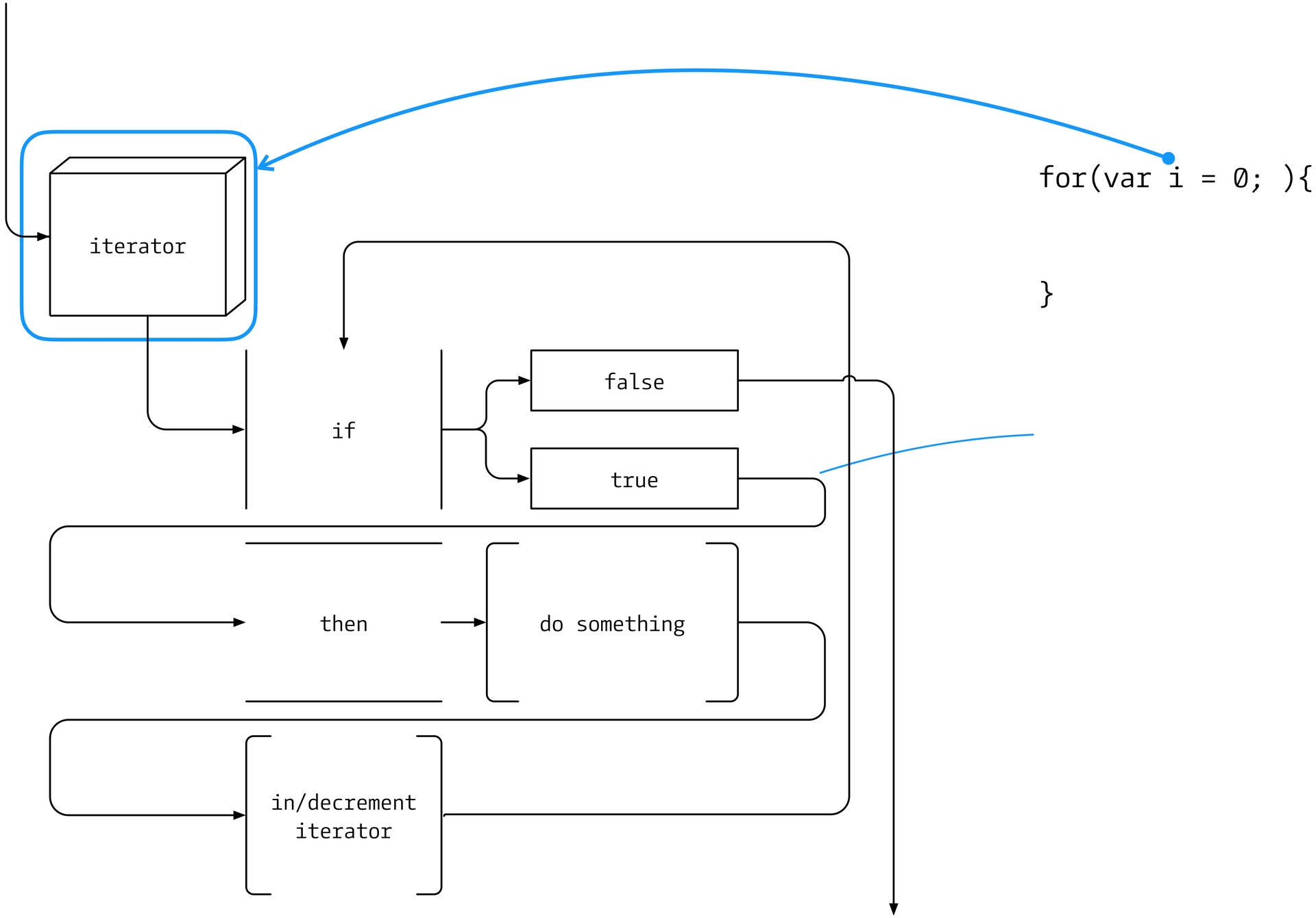

loop

Ranges

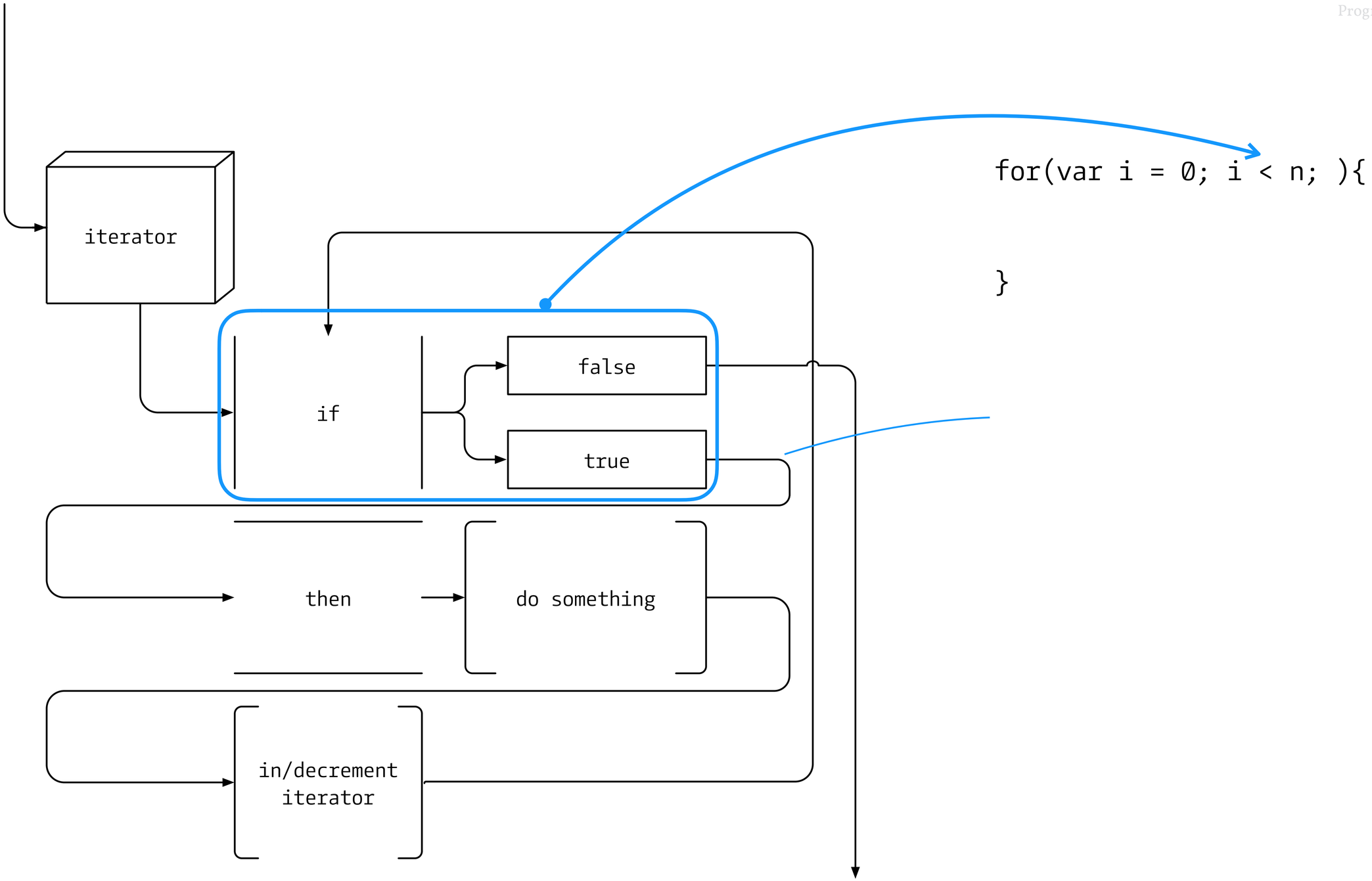

iterator
if
false
true
then
do something
in/decrement
iterator

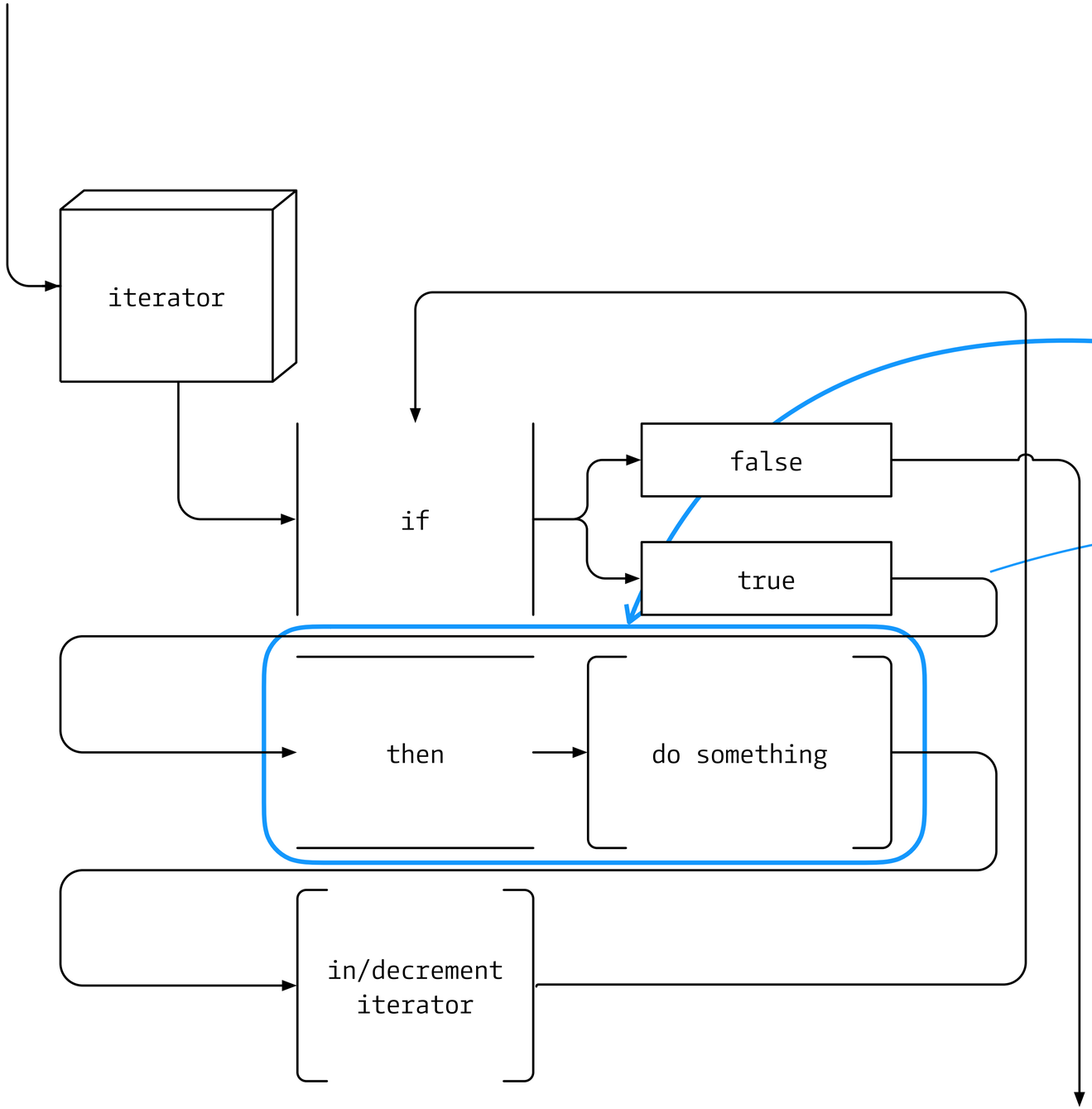

iterator
if
false
true
then
do something
in/decrement
iterator

```
for(){


}
```

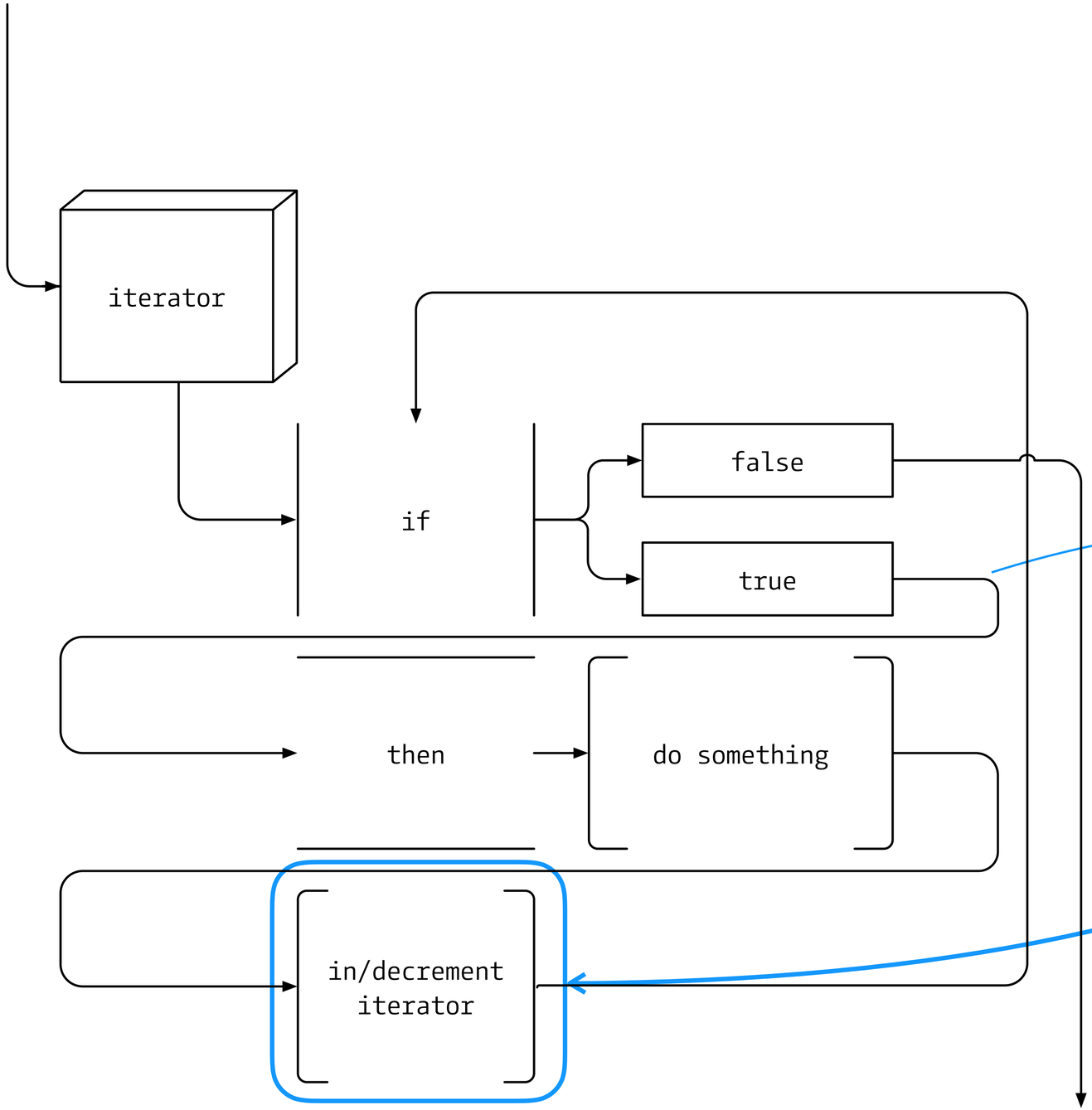

iterator
if
false
true
then
do something
in/decrement
iterator

```
for(var i = 0; ){


}
```


iterator
if
false
true
then
do something
in/decrement
iterator
for(var i = 0; i < n;){
}




```
for(var i = 0; i < n; ){

 // do something totally
 // fancy herewith the
 // curly brackets

}
```




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

 // do something totally
 // fancy herewith the
 // curly brackets

}
```

```
i++; // increase i by 1
i = i + 1; // means the same
i+=1; // also the same
```

```
i--; // decrease i by 1
i = i - 1; // means the same
i-=1; // also the same
```

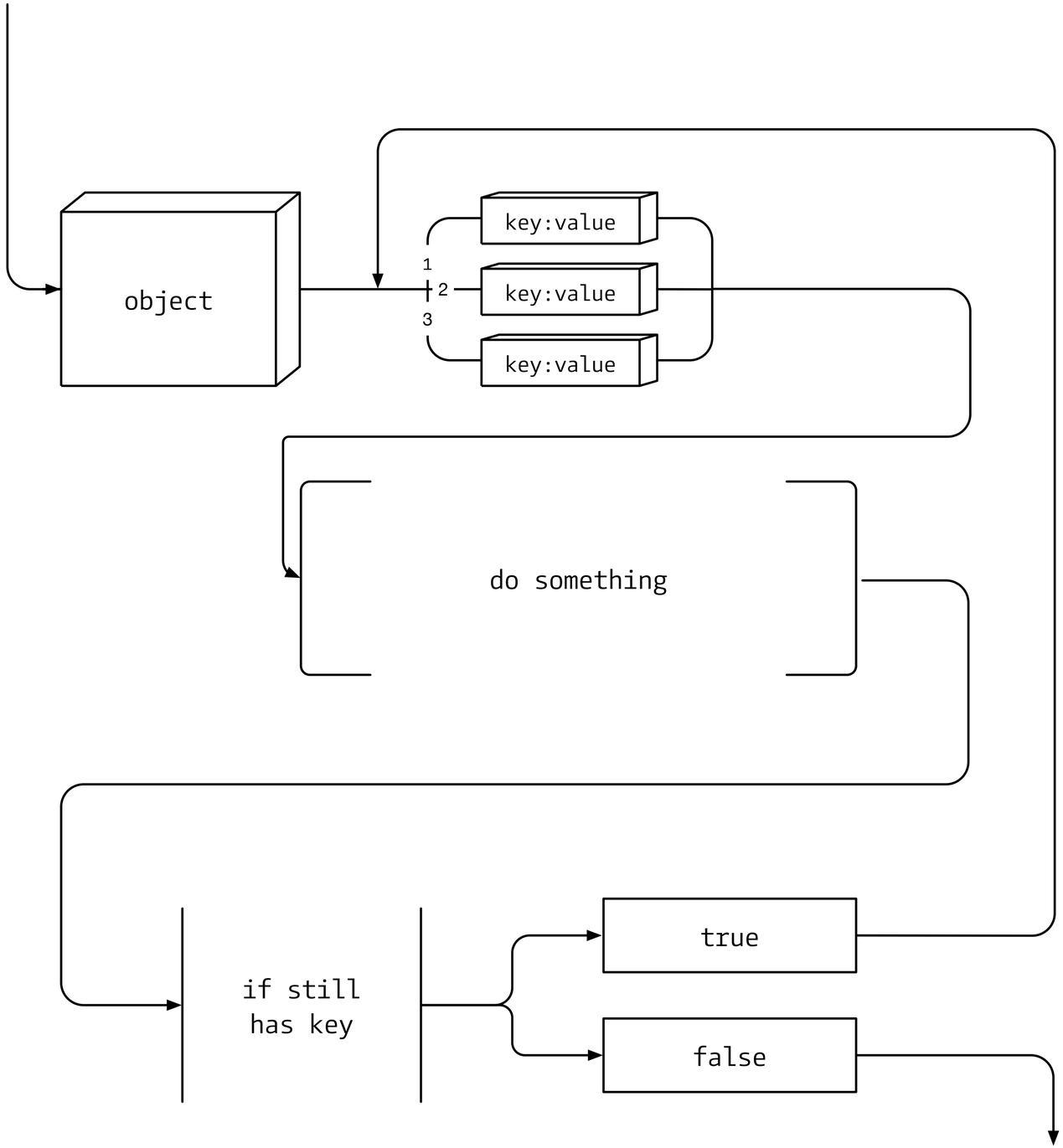
```
// does not need to be by 1
i+=5;
i=i-2;
```

```
var n = 10;
for(var i = 0; i < n; i++){
 console.log("%s × 5 = %s",i ,i * 5);
}
```

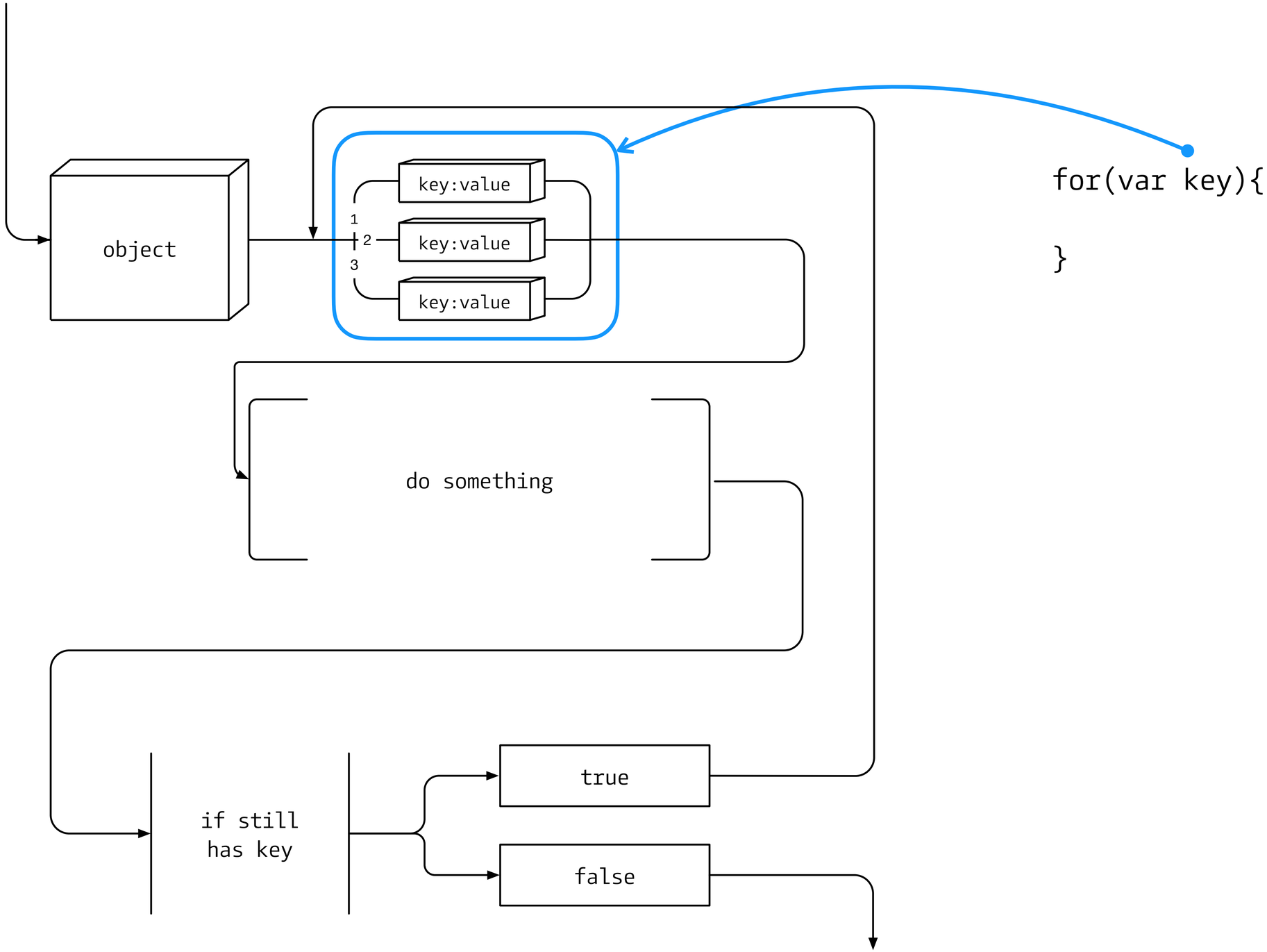
```
var n = 5;
for(var i = 100; i >= n; i-=5){
 console.log(i);
}
```

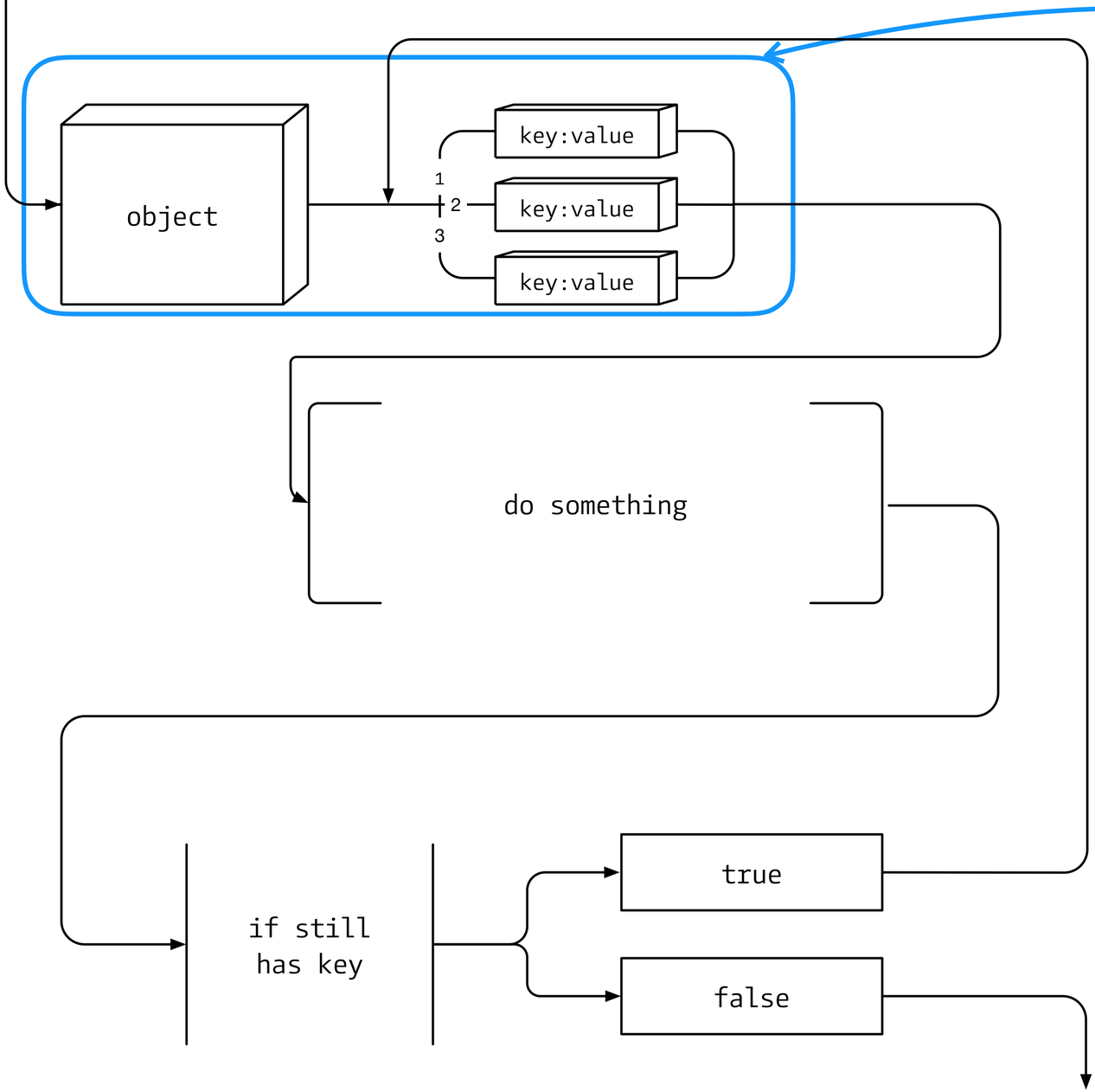
```
var arr = ["a","b","c","d","e","f"];
for(var i =0; i < arr.length ;i+=2){
 // log every second item
 console.log(arr[i]);
}
```

Object

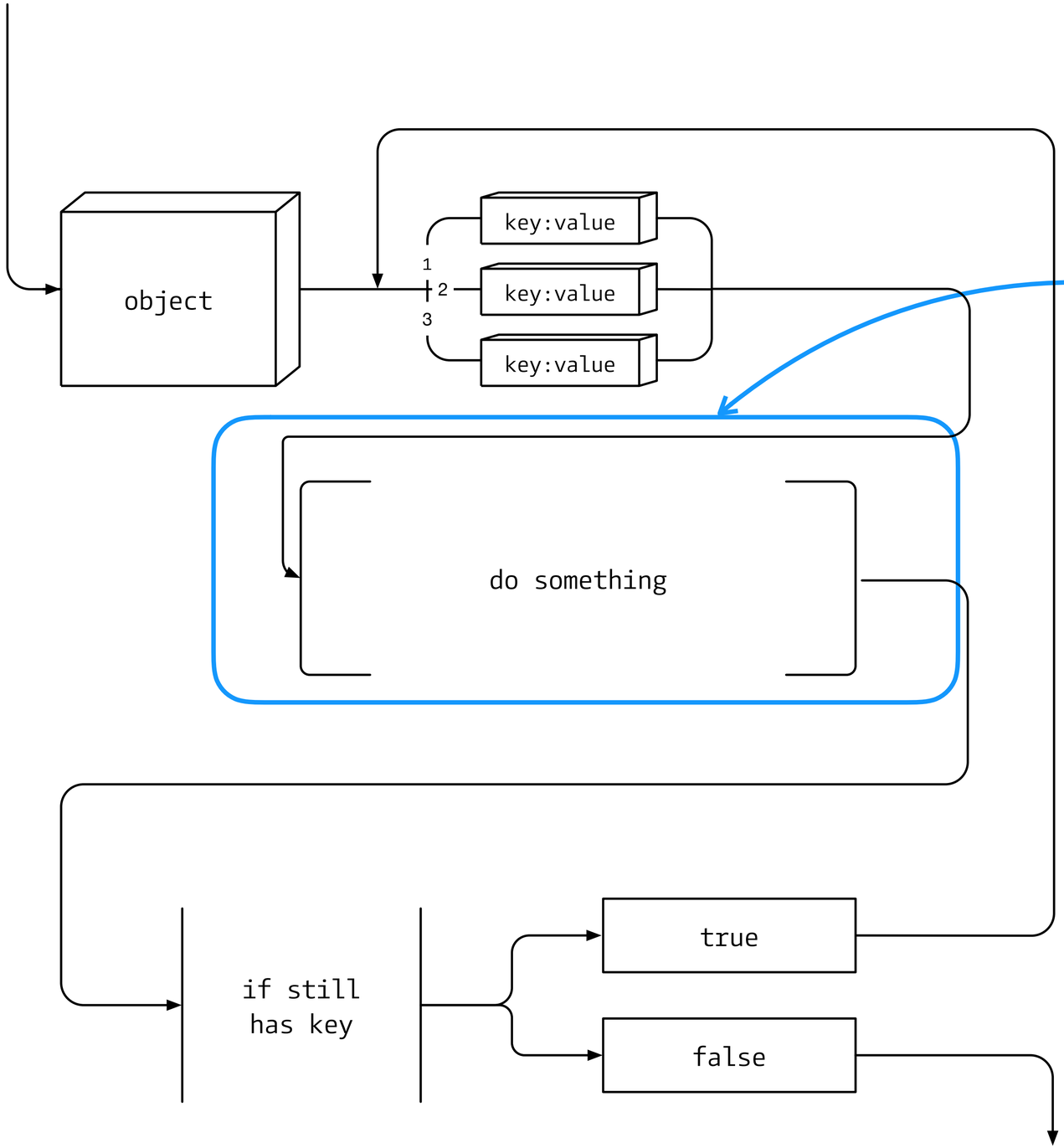

object
key:value
key:value
key:value
1
2
3
do something
if still
has key
true
false

```
for(){

}
```

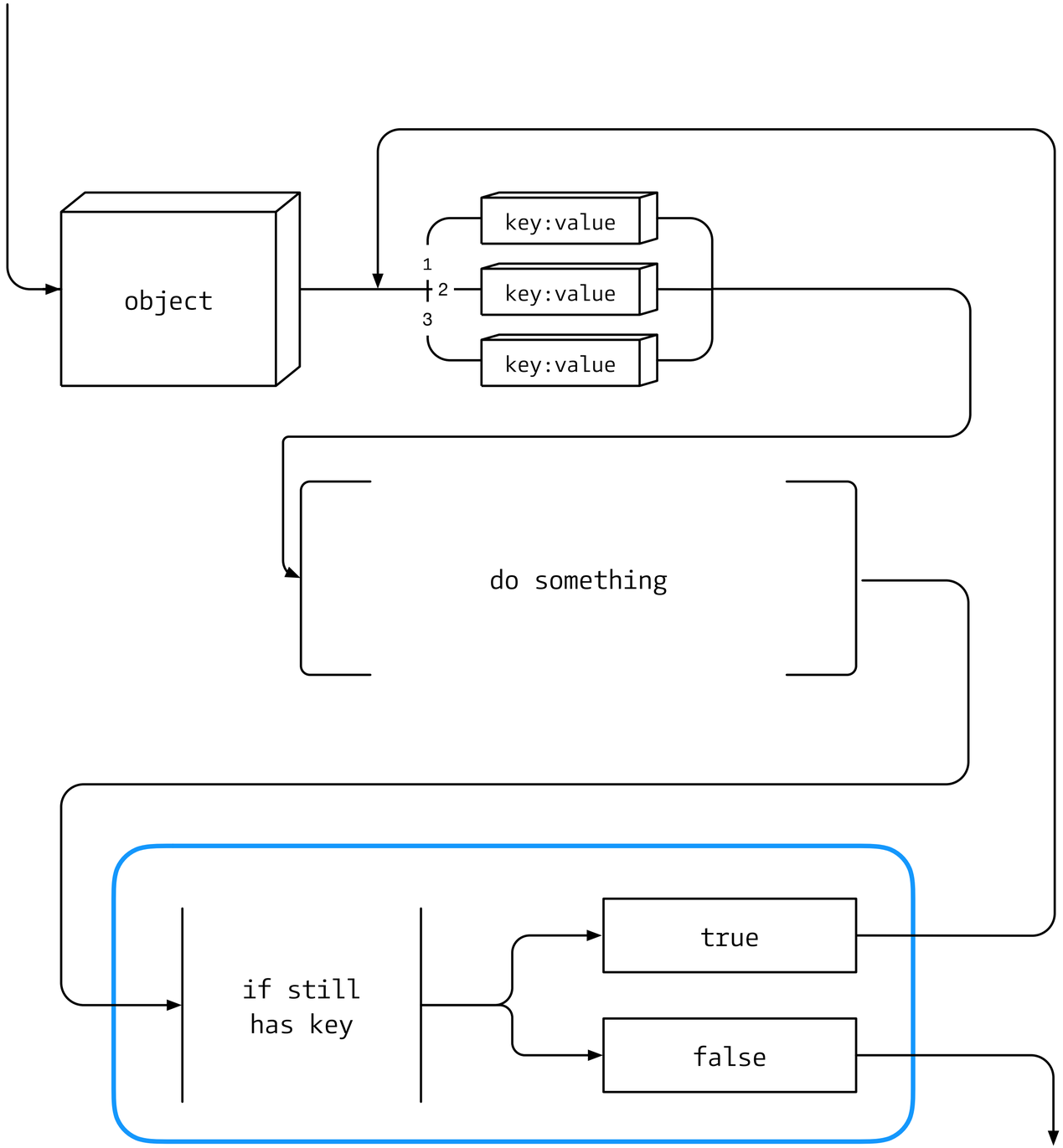

object
key:value
key:value
key:value
1
2
3
for(var key){
}
do something
if still
has key
true
false




```
for(var key in object){

}
```




```
for(var key in object){

 // do something fancy

}
```



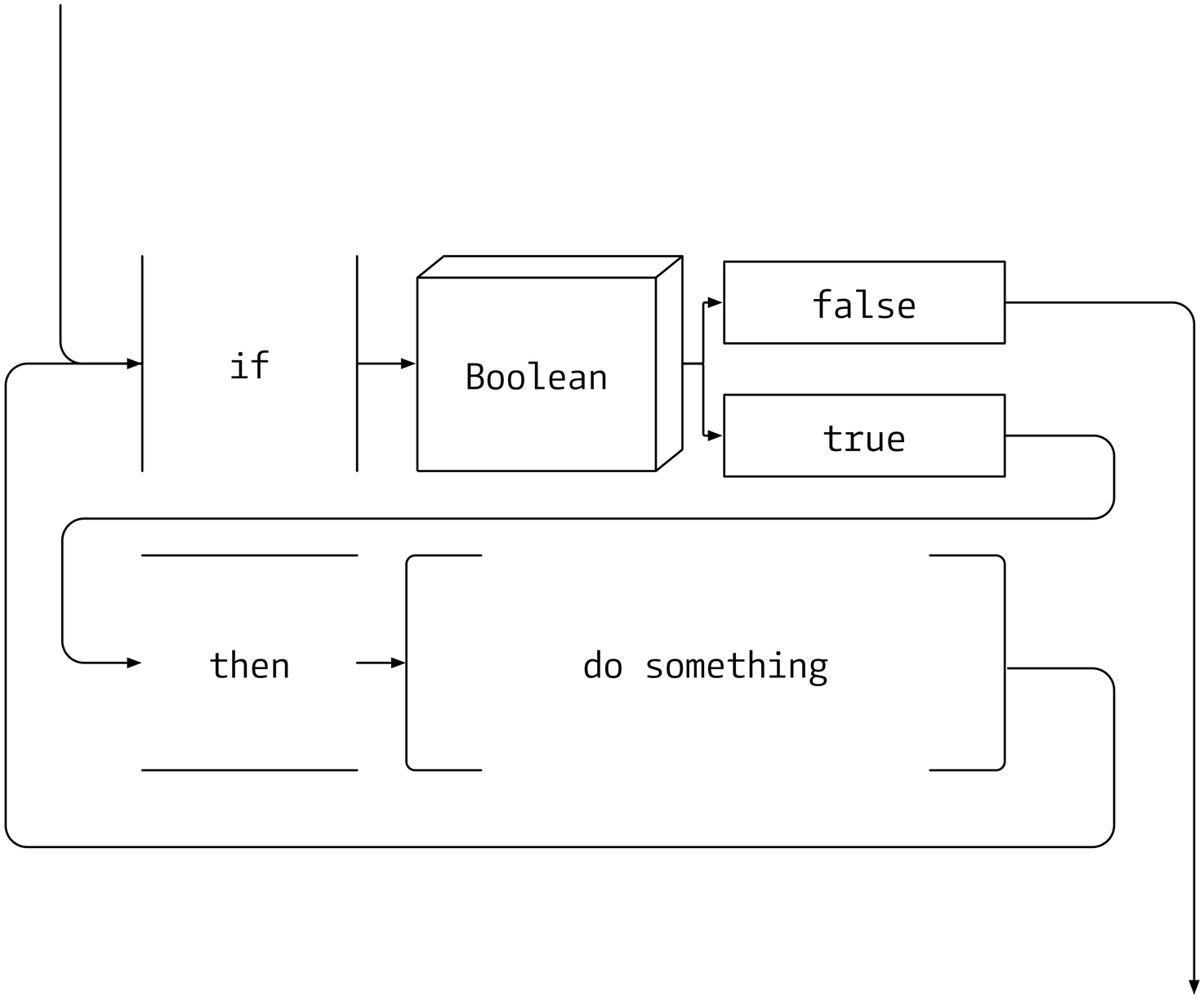

```
for(var key in object){

 // do something fancy

}
```

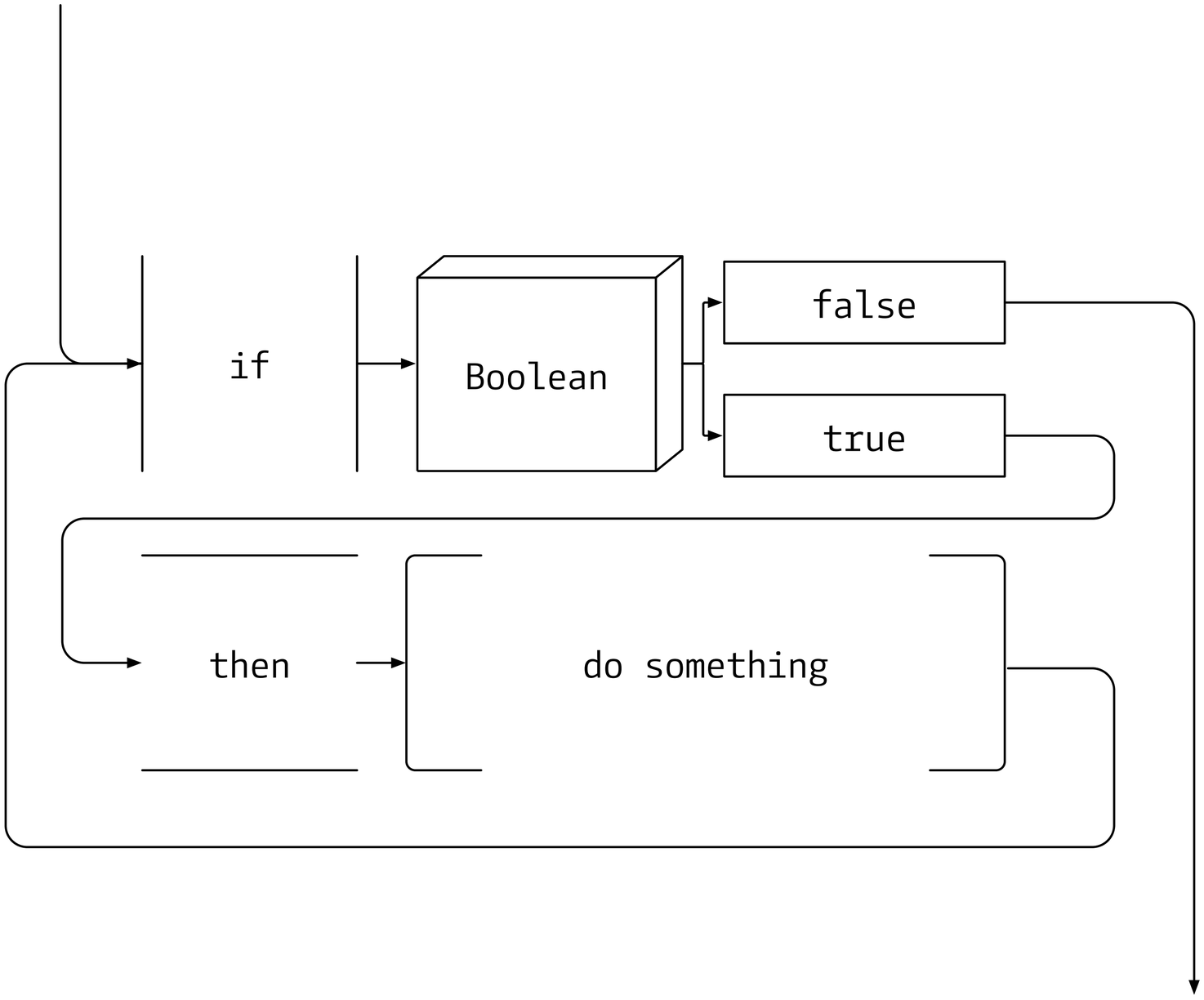
```
var obj = {"x":10,"y":20, "z":-20};
for(var key in obj){
 console.log("The key is %s",key);
 console.log("The value is %s", obj[key]);
}
```

```
var obj = {"a":"Hello","b":"World"};
for(var key in obj){
 console.log(key);
}
```

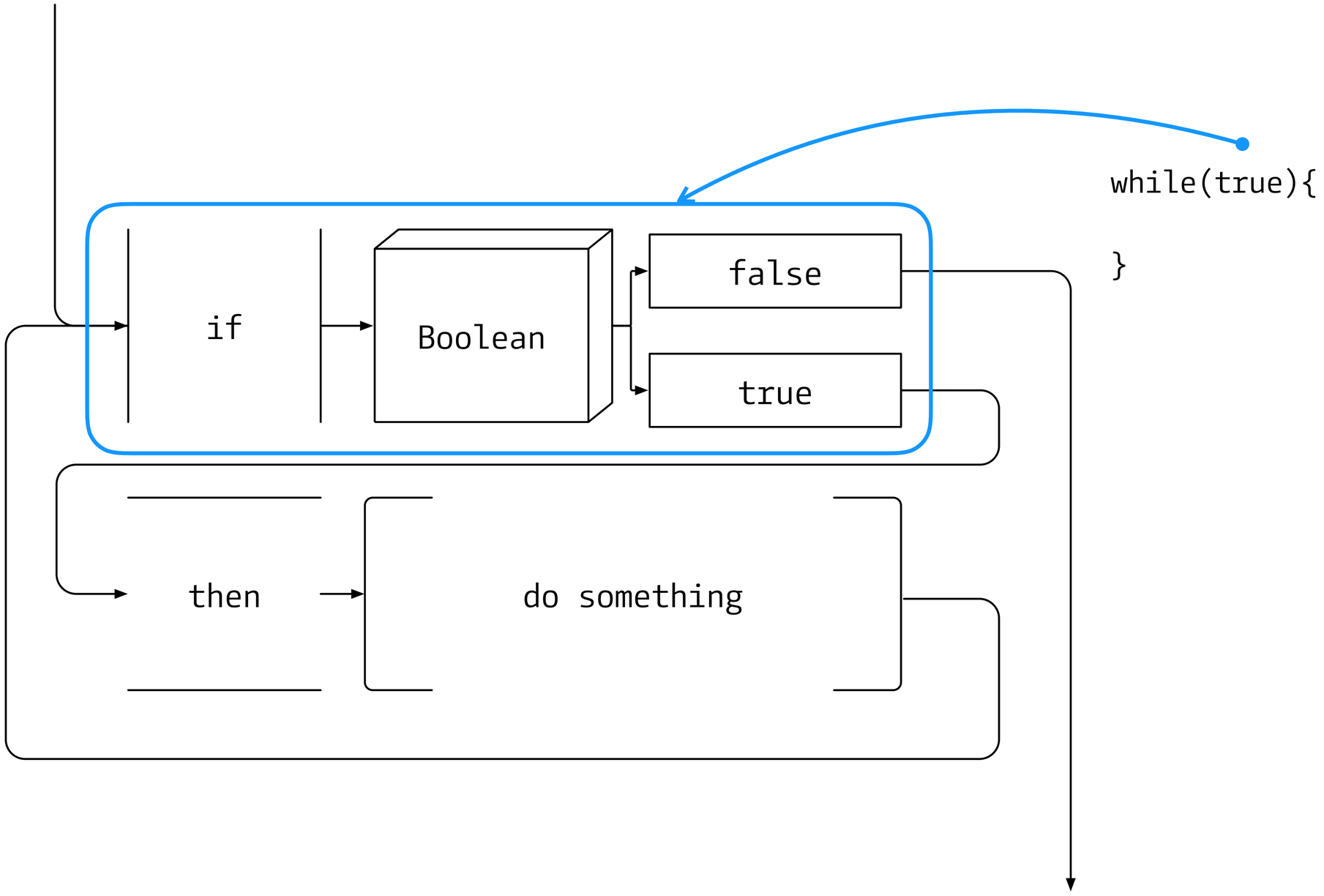
```
var obj = {"a":"Hello","b":"World"};
for(var key in obj){
 console.log(obj[key]);
}
```

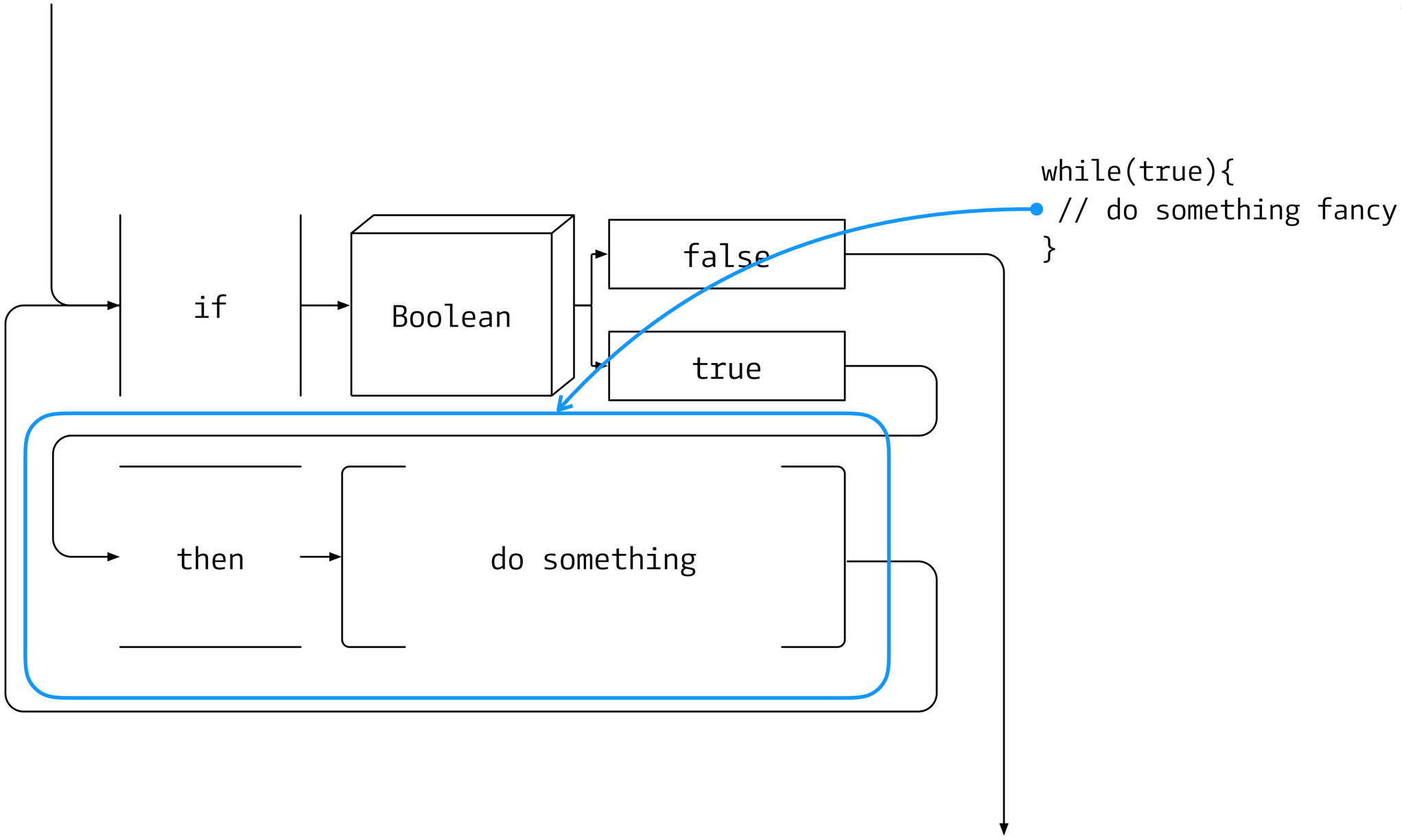
Condition


if
Boolean
false
true
then
do something




```
while(){

}
```


while(true){
}
if
Boolean
false
true
then
do something


if
Boolean
false
true
then
do something
while(true){
// do something fancy
}

```
var bool = true;
while(bool === true){
 var x = Math.random() * 5;
 console.log(x);
 if(x > 2.5){
  bool = false;
 }
}
```

```
var x = 0;
while(x < 5){
 console.log("x is %s", x);
 x++;
}
```

```
while(true){

}
```

break && continue

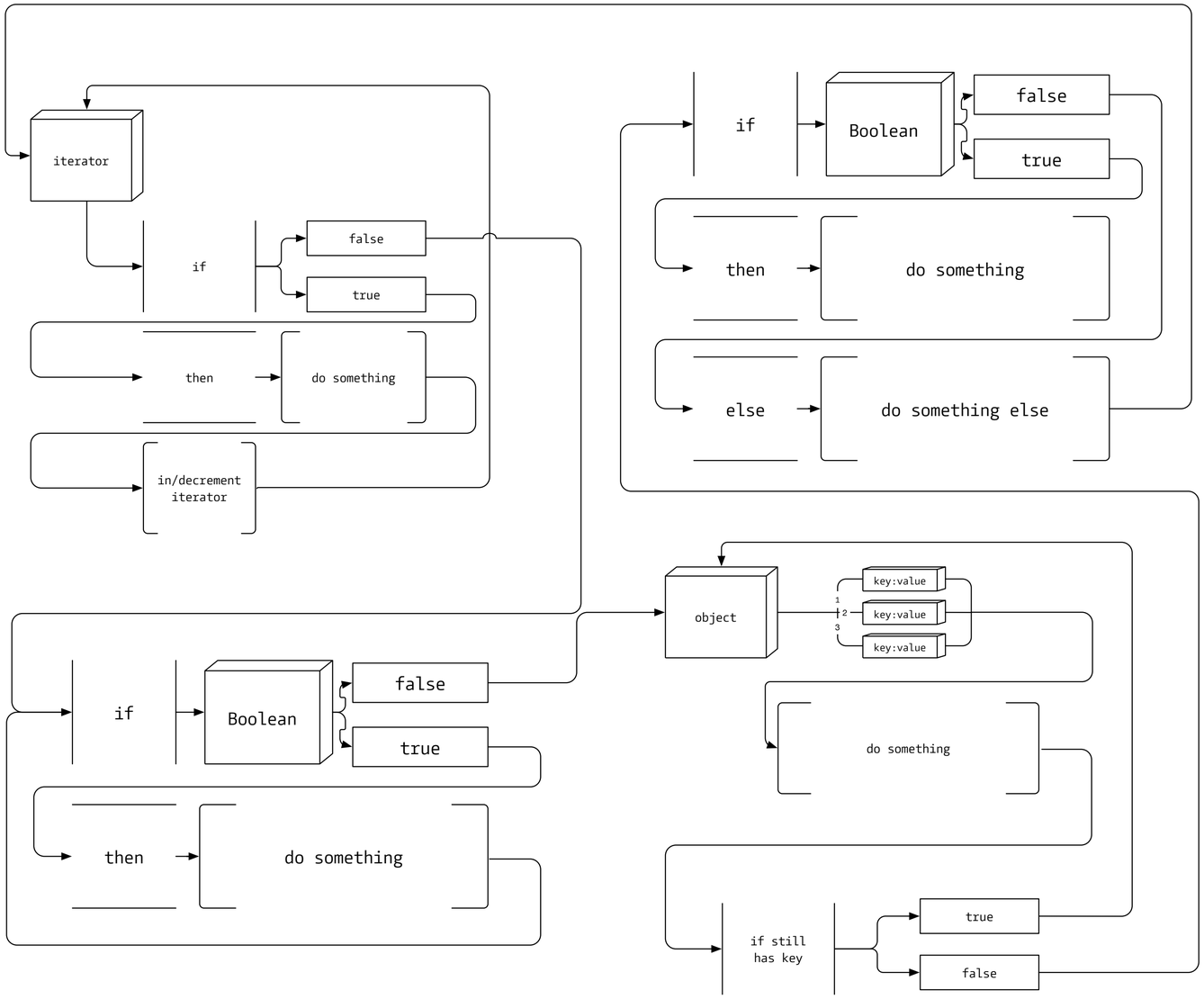
```
for(var i = 100; i >= n; i-=5){
 console.log(i);
  if(i < 30){
   break;
  }
}
```

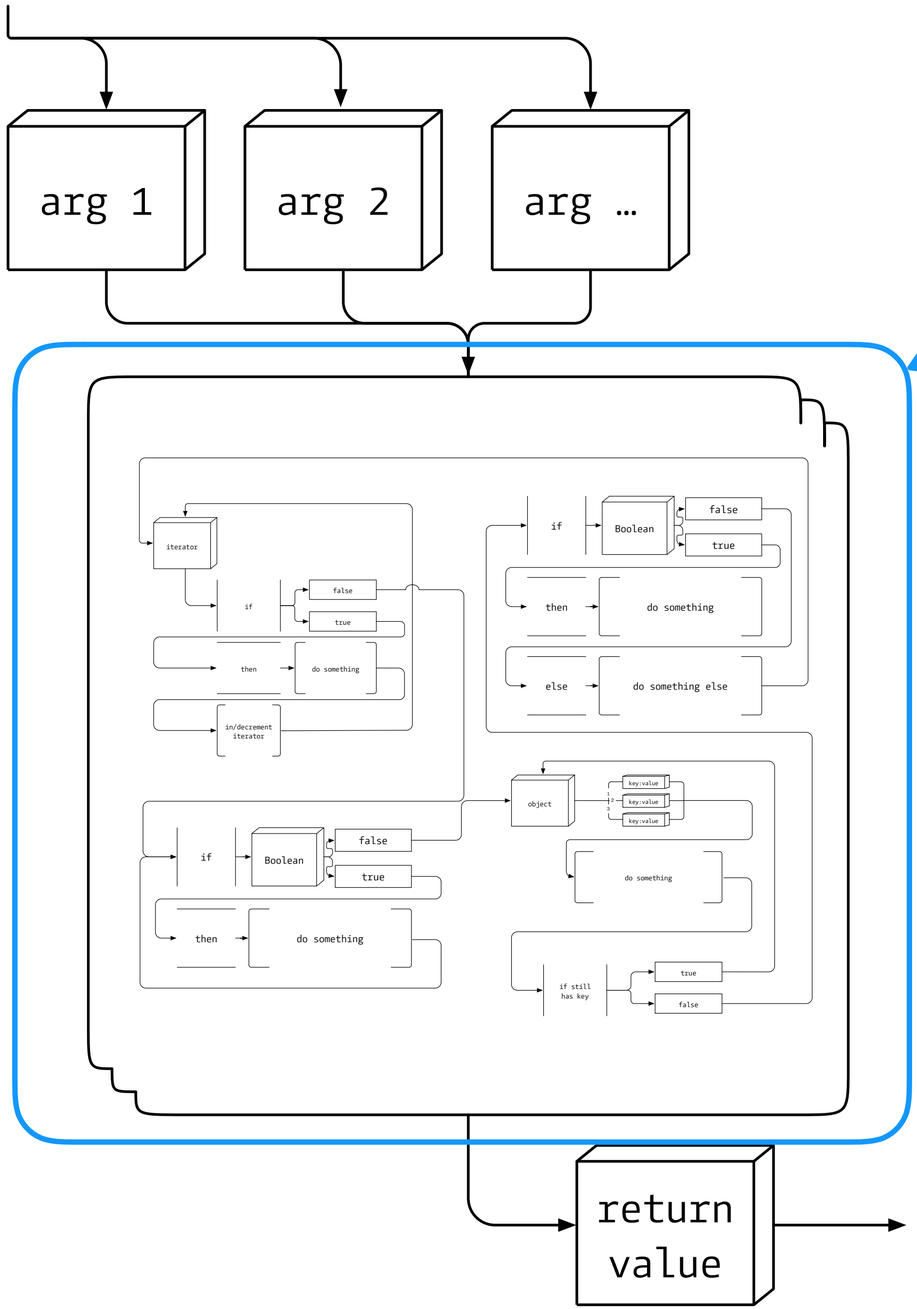
```
for(var i = 0; i < 6; i++){
  // the term below is module
  // it finds the even numbers
  if(i%2 == 0){
   // even
   continue;
  }
  console.log(i);
}
```

7 BASIC THINGS IN PROGRAMMING

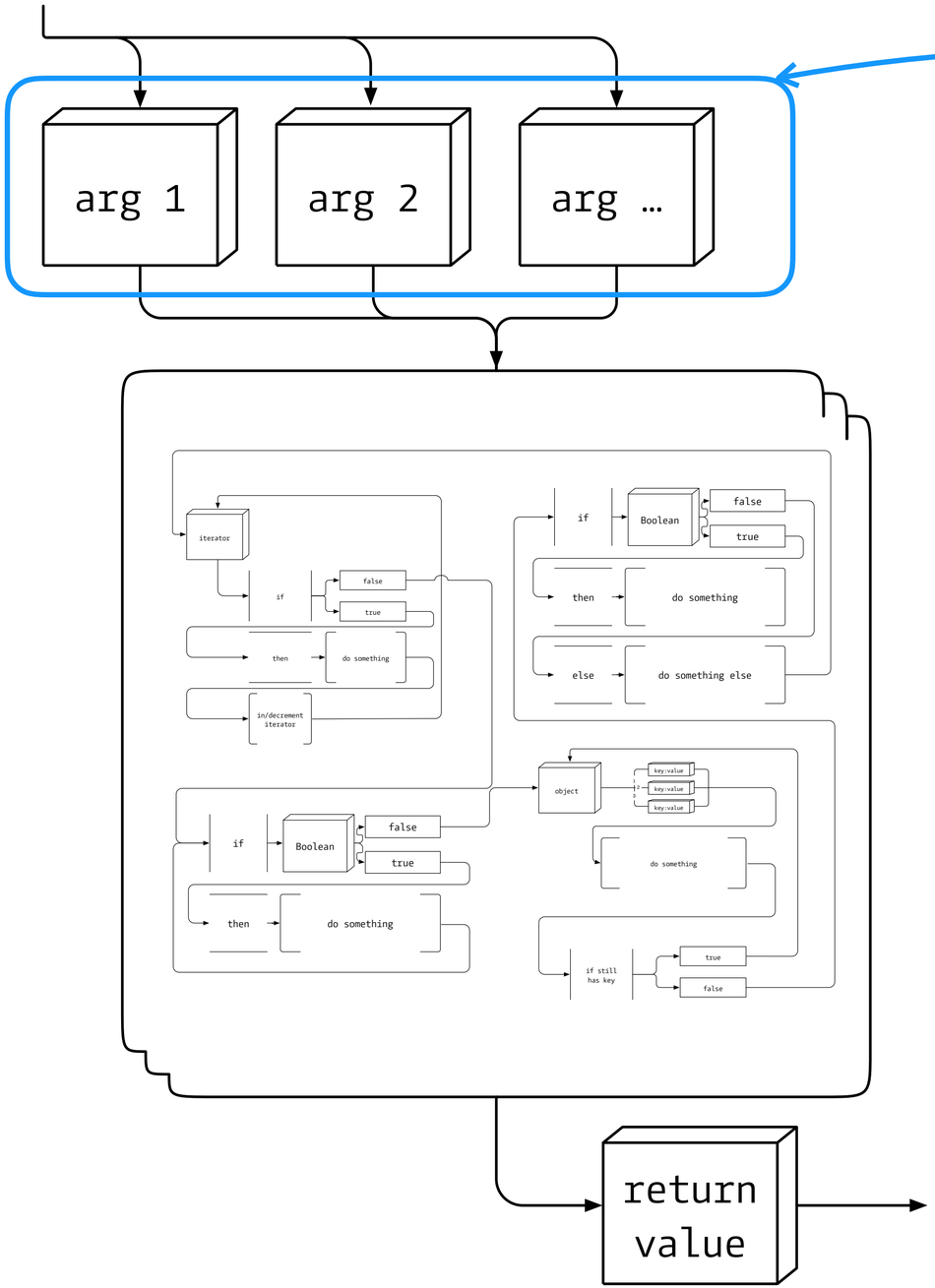
1. Variablen ✓
2. Objekte ✓
3. Arrays ✓
4. Konditionen ✓
5. Schleifen ✓
6. Funktionen
7. Algorithmus

function

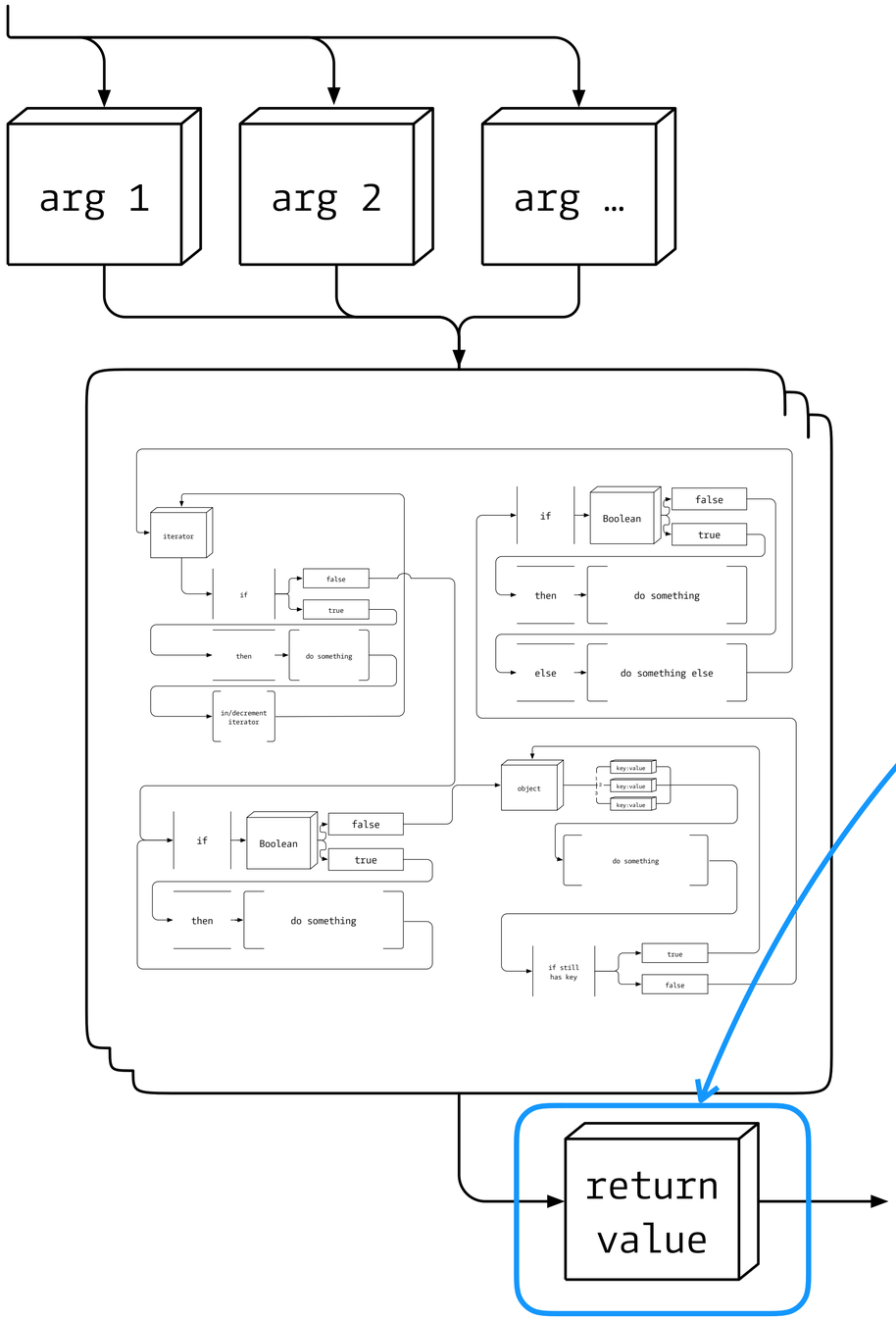

iterator
if
false
true
then
do something
in/decrement iterator
if
Boolean
false
true
then
do something
else
do something else
object
key:value
key:value
key:value
1
2
3
if
Boolean
false
true
then
do something
do something
if still has key
true
false




```
function name(){
 // lots of complex code
 var c = 5;
}

name();
```




```
function name(a, b){
  // lots of complex code
  var c = a * b;
 }

 name(10, 5);
 name(3, 5);
 name(100, 5);
 name(1000, 5);
```



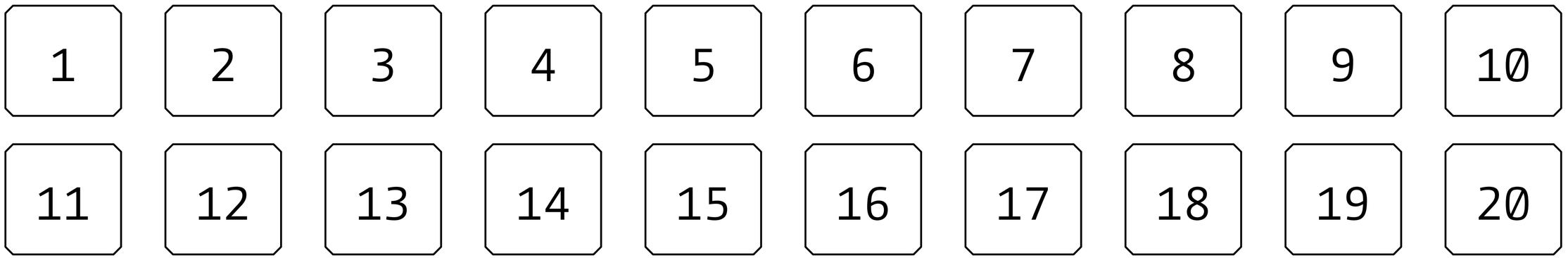

```
function name(a, b){
  // lots of complex code
  var c = a * b;
  return c;
 }
```

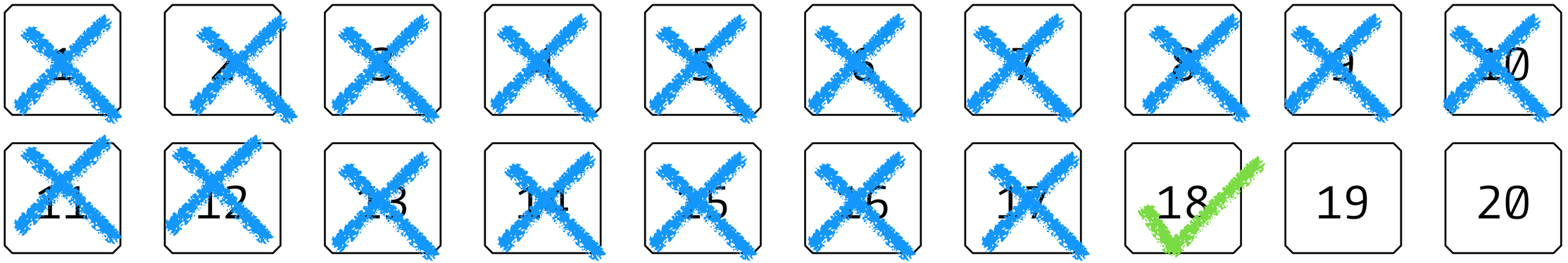
```
function calculator (a, b){
 return a * b;
}

console.log(calculator(10, 5));
console.log(calculator(3, 2));
console.log(calculator(123, 456));
```

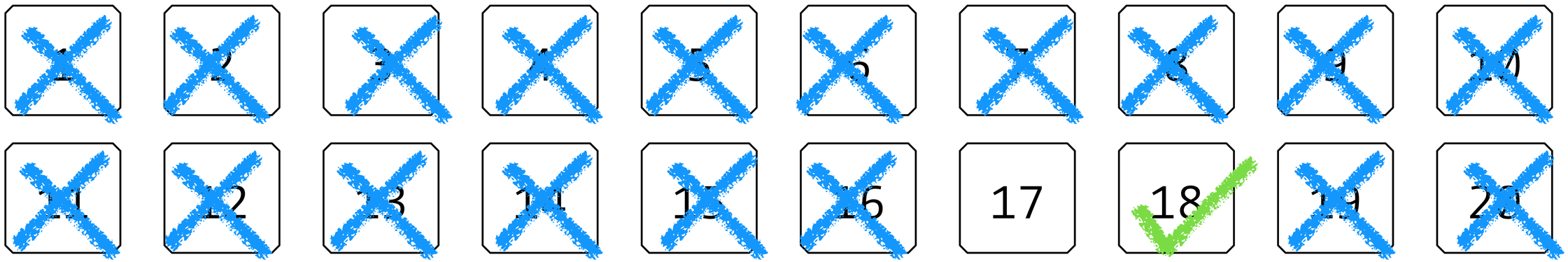
7 BASIC THINGS IN PROGRAMMING

1. Variablen ✓
2. Objekte ✓
3. Arrays ✓
4. Konditionen ✓
5. Schleifen ✓
6. Funktionen ✓
7. Algorithmus

BINÄRE SUCHE


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20


19
20


17
18

```
function binarySearch(values, target, start, end) {
  if (start > end) { return -1; } //does not exist

  var middle = Math.floor((start + end) / 2);
  var value = values[middle];

  if (value > target) { return binarySearch(values, target, start, middle-1); }
  if (value < target) { return binarySearch(values, target, middle+1, end); }
  return middle; //found!
}
var values = [1, 4, 6, 7, 12, 13, 15, 18, 19, 20, 22, 24];
var target = 12;

var result = binarySearch(values, target, 0, values.length - 1);
console.log('The target %d is at index %d' ,target, result);
```

LICHT ALGORITHMUS

```
Fabian!
  wenn das Licht an ist
    Gehe zum Schalter und schalte es aus
  wenn das Licht aus ist
    Gehe zum Schalter und schalte es an
```

```
if light.is_on
  fabian.goto(switch.location)
  fabian.set(switch, false)
else
  fabian.goto(switch.location)
  fabian.set(switch, true)
```

DRY-CODE

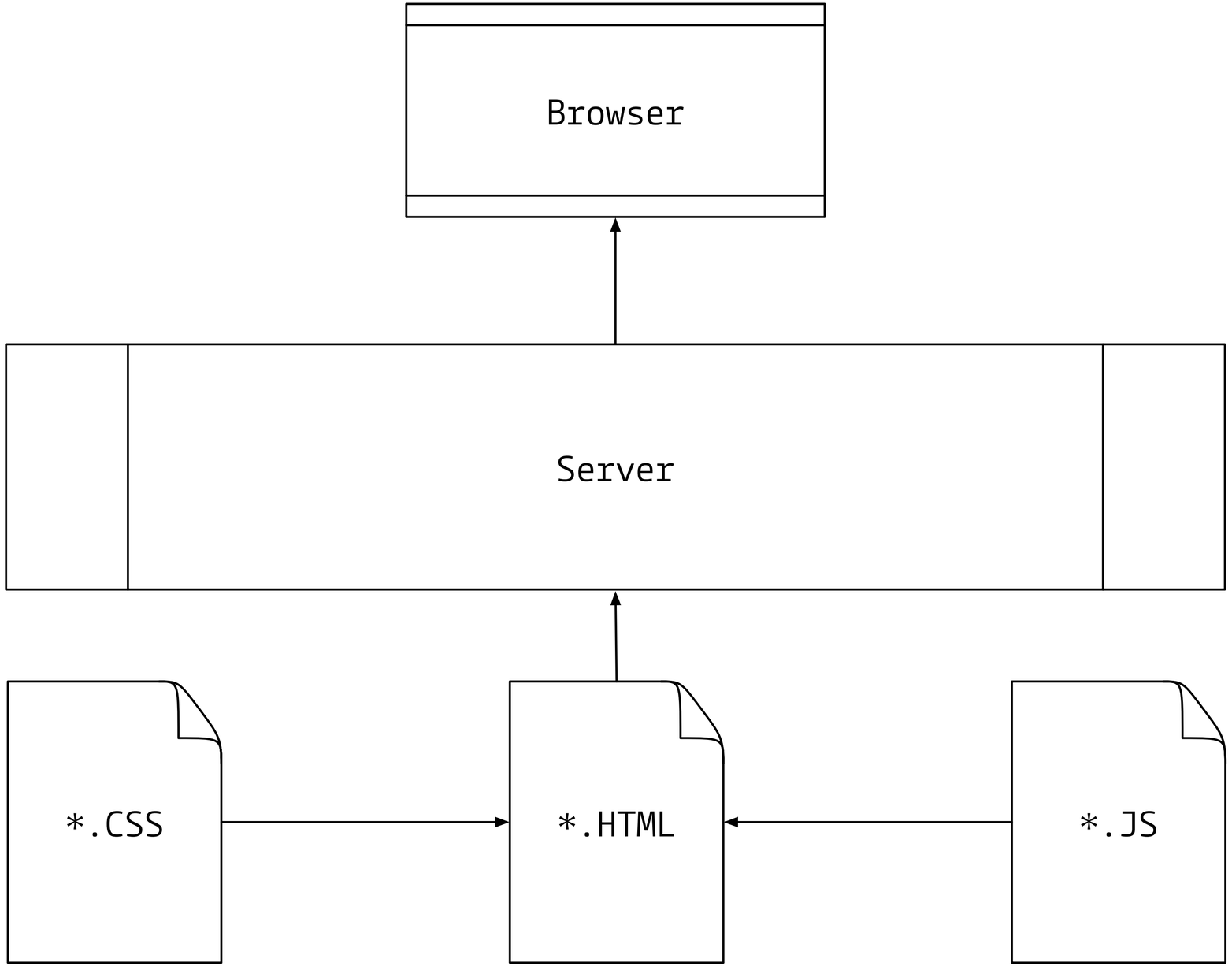
(Don’t Repeat Yourself)

```
fabian.goto(switch.location)
if light.is_on
  fabian.set(switch, false)
else
  fabian.set(switch, true)
```

```
fabian.goto(switch.location)
fabian.set(switch, !light.is_on)
```

0. SETUP

HTML + CSS + JS + Server


Browser
Server
*.CSS
*.HTML
*.JS

1. SHAPES

point, line, ellipse, rect, vertex

AUFGABE

Zeichne einen Menschen mit primitiven Formen!

10 MINUTEN

9 MINUTEN

8 MINUTEN

7 MINUTEN

6 MINUTEN

5 MINUTEN

4 MINUTEN

3 MINUTEN

2 MINUTEN

1 MINUTEN

PENCILS DOWN!

2. COLORS

stroke, fill, background

AUFGABE

Erzeuge eine spannende Farbkomposition/Farbreihe!

Hint: colorMode(HSB,360,100,100);

10 MINUTEN

9 MINUTEN

8 MINUTEN

7 MINUTEN

6 MINUTEN

5 MINUTEN

4 MINUTEN

3 MINUTEN

2 MINUTEN

1 MINUTEN

PENCILS DOWN!

3. INTERACTION

setup, draw, mouse

AUFGABE

Schreibe ein Programm das basierend auf der Position der Maus Parameter wie Form oder Farbe verändert!

10 MINUTEN

9 MINUTEN

8 MINUTEN

7 MINUTEN

6 MINUTEN

5 MINUTEN

4 MINUTEN

3 MINUTEN

2 MINUTEN

1 MINUTEN

PENCILS DOWN!

4. CONDITION

mousePressed, keyPressed

AUFGABE

Schreibe ein Programm das basierend auf einem Maus
oder Tastendruck Parameter wie Farbe oder Form ändert!

10 MINUTEN

9 MINUTEN

8 MINUTEN

7 MINUTEN

6 MINUTEN

5 MINUTEN

4 MINUTEN

3 MINUTEN

2 MINUTEN

1 MINUTEN

PENCILS DOWN!

5. LOOPS

for

AUFGABE

Fülle die Zeichenfläche mit primitiven Formen
indem du einen Loop verwendest!

10 MINUTEN

9 MINUTEN

8 MINUTEN

7 MINUTEN

6 MINUTEN

5 MINUTEN

4 MINUTEN

3 MINUTEN

2 MINUTEN

1 MINUTEN

PENCILS DOWN!

AMA

(Ask me anything)

VIELEN DANK

für eure Aufmerksamkeit.