Reading and writing to an address

If something is to be read and written from one address to another address, the procedure is always from right to left:

The target	The operator	The source
Is usually an identifier that is defined with LET, CONST, VAR: let a; const b:	The operator for an object attribute is ":" In other cases it is = += -= *= /=	null = Empty, Boolean (true,false), String (Text), Number, Array, Object

For an object, the target-source relationship is made with a colon operator:

```
const object = {
        Attribute : Value
}
```

In all other cases, the usual operators are used. Usually the = operator :

```
const content = document.getElementsByTagName(,body')
```

Additional operators :

A variable can be both target and source if ++ or - is appended after it:

X++ Increase the value in X by 1 > x = 2 > x = 3 > x = 4

X-- Decreases the value in X by 1 > x = 2 > x = 1 > x = 0

Creates a variable whose value can be overridden at any time

Let

const

Creates a variable that only gets a value once and can no longer be changed

The name of the variable. This is a placeholder and points to the address in memory that contains the value reachable through the address

LABEL

=

This is the left operand and thus the target in which something is written.

The assignment operator decides what kind of operation is done on destination and source.

= Overwrites the destination with the source.

More operators:
+= Adds the source to the target
-= Subtracts the source from the target
*= Multiplies the target by the source
/= Divide the target by the source

'Hello world'

false

true

5.4

5

[]

document.getElement...
document.querySelector...

[10, 20, 30, 40, 50]

```
{
   attribute_1 : Value,
   attribute_2 : Value
}
```

function () { }

() => { }

null

These are all sources and return a value. So they will only be read. The who of a source is written to the target. The source is the right operand.

The address points to a string (text)

The address points to the value 0/Falsch/False

The address points to the value 1/True

The address points to the Decimal number 5.4

The address points to the integer 5

The address points to an empty array

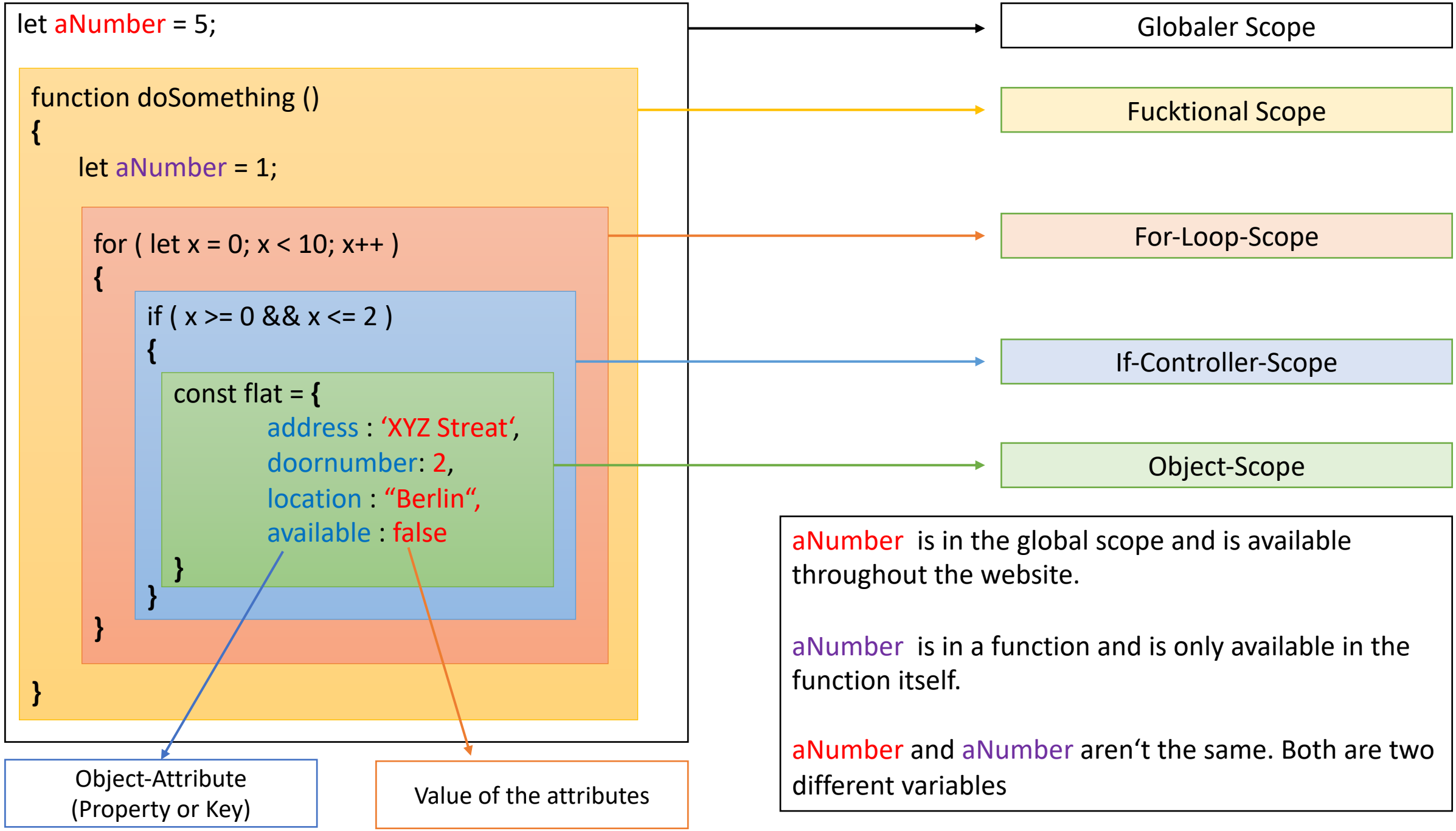
The address points to a DOM object

The address points to an array of values

The address points to an object

The address points to an anonymous function

The address is empty, it points to nothing!!!!


let aNumber = 5;
function doSomething ()
{
let aNumber = 1;
for (let x = 0; x < 10; x++)
{
if (x >= 0 && x <= 2)
{
const flat = {
address : 'XYZ Streat',
doornumber: 2,
location : "Berlin",
available : false
}
}
}
}
Globaler Scope
Fucktional Scope
For-Loop-Scope
If-Controller-Scope
Object-Scope
Object-Attribute
(Property or Key)
Value of the attributes
aNumber is in the global scope and is available throughout the website.
aNumber is in a function and is only available in the function itself.
aNumber and aNumber aren't the same. Both are two different variables

JavaScript Variable:

let LABEL **=** 12

let LABEL **=** 9.5

let LABEL **=** null

let LABEL **=** true

let LABEL **=** false

let LABEL **=** “Hello World“

let LABEL **=** []

let LABEL **=** [10, 20, 30, 40, 50]

let LABEL **=** document.getEleme..

let LABEL **=** document.querySel..

let LABEL LABEL **=** function() {}

let Bezeichner **=** () => {}

Target (The address of the variable):

let LABEL

Allocation :

=

Source (The value):

12

9.5

null

true

false

‘Hello World‘

[]

[10, 20, 30, 40, 50]

document.getElement…

document.querySelector…

function () {}

() => {}

The target becomes overwritten from the value of the source

JavaScript Object:

```
const LABEL = {

        Attribute: Value,

        Attribute: [ 10, 20, 30, 40, 50 ],

        Attribute: 'Hello World',

        Attribute: true,

        Attribute: null,

        Attribute : function () {
                // …
        },

        Attribute : () => {
                // …
        }

}
```

Target (Address of the varaible):

```
const LABEL
```

Allocation:

=

Source (The object):

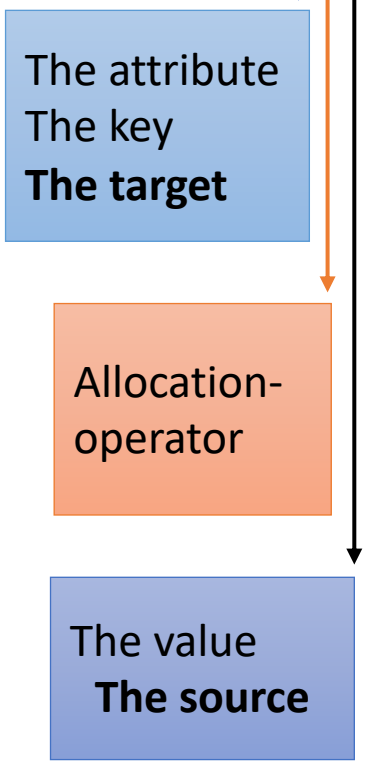
```
{

        Attribute: Value,

        Attribute: [ 10, 20, 30, 40, 50 ],

        Attribute: 'Hello World',

        Attribute: true,

        Attribute: null,

        Attribute : function () {
                // …
        },

        Attribute : () => {
                // …
        }

}
```

The target ← becomes overwritten ← from the value of the source

Pass the object as a parameter to a function

Structure of an object:

```
const object = {
        a : 2,
        b : 6
}
```



Sample 1:

```
function ( object )
{

    let result = object.a + object.b;

}
```

Sample 2:

```
function ( { a, b } )
{

    let result = a + b;

}
```

Sample 3:

```
const myFoo = (object ) =>
{

    let ergebnis = object.a + object.b;

}
```

Sample 4:

```
const myFoo =( { a, b } ) =>
{

    let result = a + b;

}
```