What does a program or website consist of?

Information

What is information?

- ✓ Number
- ✓ Boolean (True/False)
- ✓ Text
- ✓ Picture
- ✓ Music / Sound
- ✓ Video / Camera
- ✓ Table
- ✓ Availability (e.g. internet connection, printer, scanner, webcam, sensor, etc.)

In short, everything in the computer, every program and every website consists of information.

What types are there?

Primitive Information:
This is the smallest piece of information there is: Number, Booleans, Text

Arrays:
An array can be empty, contain one or more pieces of information. It's a kind of list.

Objects:
These are complex structures that can contain various types of primitive information, arrays, and also objects. An object can also contain functions.

Instructions

What is an instruction?

An instruction is a command to the computer. through instructions

- ✓ Action
- ✓ Calculation
- ✓ Input (Keyboard, Mouse, Touch-Screen)
- ✓ And much more

Remember: Data alone is of no use.
Statements cannot be created without data.

Everything has an identifier (LABEL) !!!!!!!!!!!

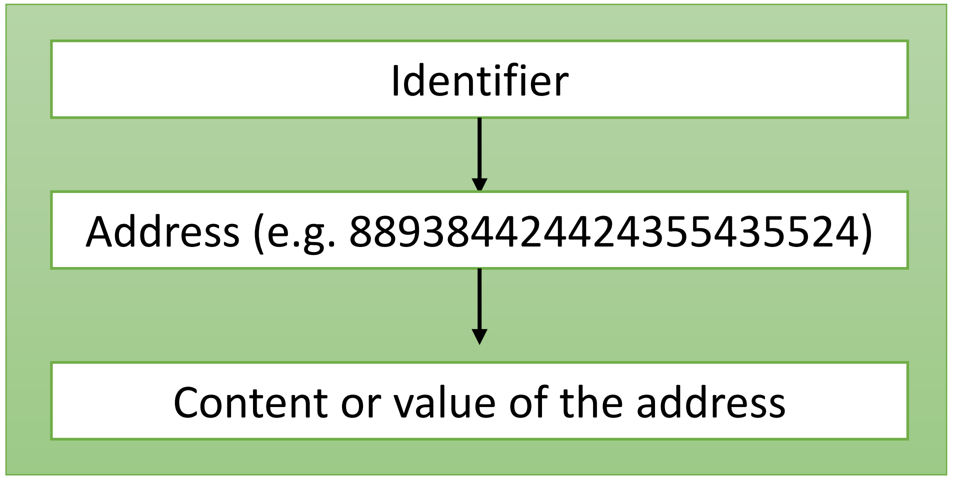
What are identifiers?

Each element that you define in the source code gets an identifier. All variables have an identifier:

```
let a = 6; // a is the identifier, 6 is the value to assign

const b = 7; // b is the identifier, 7 is the value of the assignment

let liste = [ 1, 2, 3 ]; // list is the identifier, [1,2,3] the value of the assignment

function doSomething ()  // doSomething is the identifier of the function
{
}

const vehicle = {};  // vehicle is the identifier of the object

class device {    // device is the identifier of the class
};
```

What are identifiers?

Since everything in the computer actually consists of addresses and these are only known when the program or website is loaded, the addresses are not used when programming, but placeholders for the addresses. Those are the identifiers. An identifier points to an address in memory that we base our value on.


Identifier
Address (e.g. 889384424424355435524)
Content or value of the address

What is a variable, array or object?

Variable

A variable is a placeholder for just a single value.

NULL or 0 is also information. Also NOTHING or EMPTY is information.

A variable can hold an integer (e.g. 5), a decimal number (e.g. 1.7), a boolean value (e.g. true / false) or a text (e.g. 'Hello World‘).

Array

An array is basically a list. It can

- ✓ To be empty
- ✓ Can contain at least one value
- ✓ Or several values

An array makes it easy to loop the information it contains

- ➢ to evaluate
- ➢ to manipulate

Object

An object is a concept as in real life. It contains properties and methods. An example would be the car. For example, a car has the following properties:

- ➢ Red color
- ➢ Four doors
- ➢ A diesel engine
- ➢ 160 hp

The methods of a car are for example:

- ➢ Drive
- ➢ change gear
- ➢ brakes

Objects can contain other objects, primitive variables, and arrays alike.

Types of variables, arrays and objects

Variable

- ✓ zero (NOTHING)
- ✓ undefined (no reference)
- ✓ Integers
- ✓ Float (Decimal Numbers)
- ✓ Boolean (true / false)
- ✓ string
- ✓ Anonymous Functions

Array

An array can collect any kind of information in a list.

Object

JavaScript offers a number of standard objects provided by the browser.
Examples:

- ➢ Event object (Event)
- ➢ DOM (Element Manipulation)
- ➢ document (document manipulation, DOM is part of it)
- ➢ window (manipulation of the browser tab or window)
- ➢ Date
- ➢ Math (mathematical means)

Commonalities of variables, arrays, objects and all data

Editability

All variables, arrays and objects can be defined in three ways:

var
The declared address is available either in the global scope or in a local scope. It can be read and written.

let
a declared address is always only available in the immediately defined scope. It can be read and written.

const
like let. The address is only readable, can only be written once.

Availability

Each piece of information is an address. Therefore, you can check its availability by typing:

```
If ( element )
    It is available or
    contains information
    not equal to 0.
else
    It is not available or
    is equal to 0.
```

We can also determine the type of an address:

```
console.log(typeof element);
```

When do you need a variable, when an array, when an object?

Variable

A variable is needed when only a single piece of information is required. Typical cases are:

- ✓ Simple calculations like c=a+b
- ✓ Displaying a single piece of information like console.log(x)
- ✓ Check for availability like if (x==y)
- ✓ To use little memory
- ✓ Examples:
- ✓ Login process
- ✓ contact form
- ✓ calculator

Array

- ✓ An array is usually needed when you need list logic.
- ✓ An array is also used when you need to quickly reach and manipulate or read each element in an array with less code. Key word: loop
- ✓ An array can hold data statically or dynamically. Static data is passed in the source code of the array. Dynamic data is passed into the array at runtime.

Object

- ✓ Objects are used on larger projects to divide the overall functionality of the program into meaningful chunks.
- ✓ This makes it easy later to create extensions, find and fix bugs, perform optimizations, and so on.
- ✓ Example scenario:
- ✓ A website could consist of the following objects:
- ✓ customer data
- ✓ product data
- ✓ Current offers
- ✓ order process

The variable, array, object, function, and its contents

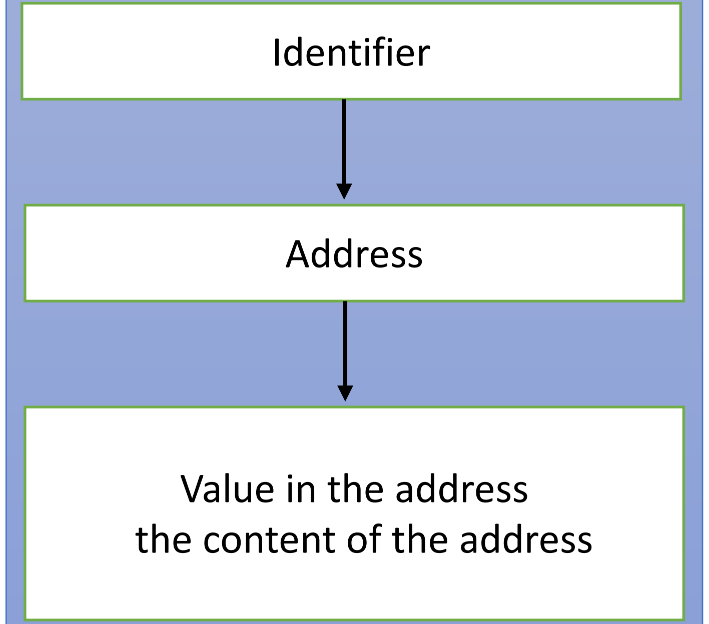
Identifier

The identifier is the name given to a variable, array, object, or function:

```
const x = 2;
let n = [];
const object = {

};

const doWhat = () =>
{

}

or

function doWhat ()
{

}
```

Content

The content or value of an identifier is a place in the main memory. The identifier doesn't really have the value, just the address that points to the value:



identify sources of error

- ✓ One must never equate the source with the destination
- ✓ Destination and source are always independent things

Examples:

```
const row1 = sheet.insertRow(1)
```

The two 1's have nothing to do with each other. "row1" is the variable. insertRow is a method of the sheet object and the 1 is a parameter of the method.

```
const name = ´This is his name´;
```

The two "name" expressions have nothing in common. "const name" creates a variable that has the identifier "name". The other "name" is just text in a string.

Working with a variable

Create

var x;
var x = wert;

let x;
let x = wert;

const x = Wert; -> must have a value!!!

Values:

x = **true** or x = **false** -> Boolean value

x = 5 or x = -5 -> IntegerInteger
x = 0.5 or x = -4.62 -> Floating point number

x = ‘Hello‘ -> Text

x = [] -> Empty Array

x = [52, 31, 60, -94] -> An array with 4 items

Write

As with Values Next Door, but it goes further:

x = a + b

x = (a * b) / c

x = “My Name is“ + name

x = ((a – 0.5) * b) + 4

x = product[0] + “ is sold“

x[5] = x[2] - (a * y[z])

Read

The basis of reading a variable is the same as writing it :

WriteInto = ReadFrom

Sample:

x = a + b

The values of a and b are read, added, and written to the x. Other reading methods:

- ✓ console.log(x)
- ✓ alert(x)
- ✓ Object.value = x
- ✓ Object.innerText = x
- ✓ Object.innerHTML = x

What is an instruction and how to create it and what do they do?

An instruction

A statement is a piece of code that does something. The instruction needs information to be able to be defined at all.

Simple examples of an instruction are:

- ✓ x = a + b
- ✓ functionalCall()
- ✓ z = functionalCall(x,y)
- ✓ if (…) …

Alone, in block

Instructions must always be included in the context. If the context only needs one statement, then you only specify the one statement. If the context requires multiple statements, then the statements are written in {} brackets. The context can be:

- ✓ If, else if, else
- ✓ for, while

Functions must have the {}.

Function, global scope

In JavaScript, the global level is already a kind of function. You can write instructions in it:

```
<script>
  let x = 5;
  let y = 3;
  let r = 0;
  r = x + 5;
</script>
```

But that's not a good solution. It is always best to put statements in a function:

```
function myFoo ()
{
 ….
}
```

Recognizing scope and avoiding mistakes

What is a scope?

The top-level scope is an empty JavaScript file or the immediate content of a script tag. This is the so-called global scope. Each block is in turn another scope. Blocks are always {} inside brackets.

```
Top-level scope
{                     sub-scope 1
  {                   sub-scope 2
    {                 sub-scope 3
      …
    }
  }
}
```

Validity of identifiers

Each identifier is always valid in the scope and its sub-scopes in which the identifier was defined:

```
let a = 5;    // Valid on all scopes
{
let b = 2;  // Valid here & in sub-scope
    {
          let c = 0;   // Valid only here
    }
}
```

Types of instructions

Target – Source

A target-source statement has two operands and one operator.

- ✓ An operand is either a destination or a source.
- ✓ The operator indicates the type of processing.

Sample: x = y
x and y are operands
= is the operator

The target receives the result from the source. The source can be a value or a complex operation:

```
x = 5              -> x is assigned the value 5
x = 8 – f          -> x gets the value from 8 – fx =
funktion() -> x receives the result of the function
x = (a + foo(b – 1, 3) ) * 0.5 -> x gets the result
```

Invocation

A statement can simply be calling a function when the function returns nothing:

```
// Without parameters
x()

// With one parameter
x(a)

// With multiple parameters
x(a, b, c )
```

Command statement

Special language rules like if, else if, for, switch, case, while are such command statements.
examples :

```
If ( x == 5 ) -> If x is equal to 5

for ( x of n ) -> Go through each element x of n

while ( r != 0 ) -> As long as r is not 0...

Switch ( x ) -> Consider the value of x

case 2:      -> If the value of x = 2...
```

Instructions : target and source

Target – Source - Relationship

The target of such a statement is always a variable, array or object. The goal is always just one element. The source can either be just a number, a boolean expression (true / false), a null, a string ("Hello"), another variable, array, object or a function. As long as the source is meant to be primitive, it can be a combination of primitive variables, numbers, booleans, and strings.

Primitive:

```
        x = a + b + 5 – 0.5 + ´ oh my God ☺´
        x = (a + b)
        x = ´Hello, my name ´ + ´ is Anton´
        x = 5
```

Array:

```
        x = []
        x = [ 2, 3, 4, ´fish´, ´pear´, true, false ]
```

Object:

```
        x = object
        x  = {
                 …
           }
```

Functions

Model 1:

```
x = function ()
{
   …
}
```

Model 2:

```
x = function (a, b, c)
{
   …
}
```

Model 3:

```
x = () =>
{
   …
}
```

Model 4:

```
x = (a, b, c) =>
{
   …
}
```

Operators:

=	Overwrites the variable x = 2 + 5 -> 7

+=	Added to the existing Value:
	x = 2
	x += 7 -> 9

-=	pulls from existing value off
	x = 6
	x -= 2 -> 4

*=	multiplied on it
/=	divided by the value

What do you need functions for?

Conzept

A function combines a whole block of instructions under a common identifier, the function name. It has the advantage that once created, it can be called up throughout the program.

Another advantage of a function is that you can pass parameters to it. Thanks to the parameters, the function always remains the same, but can produce different results.

Sample function

```
function calc ( z1, z2, weg)
{
    if ( weg == ´+´ )
      return z1 + z2;
    else if ( weg == ´-´ )
      return z1 - z2;
    else if ( weg == ´*´ )
      return z1 * z2;
   else if ( weg == ´/´ )
      return z1 / z2;
   else
      return 0;
}
```

Function test

```
// 5 + 3 = 8
console.log(    calc( 5, 3, ´+´ )    );

// -2 + 7 = 5
console.log(calc( -2, 7, ´+´ )    );

// 9 - 5 = 4
console.log(calc( 9, 5, ´-´ )    );

// 6 * 2 = 12
console.log(calc( 6, 2, ´*´ )    );

// 21 / 3 = 7
console.log(calc( 21, 3, ´/´ )    );

// 5 % 3 = 0, because % is not intended as a calculation method
console.log(    calc( 5, 3, ´%´ )    );
```

Controller - If

What is that?

A control structure allows the program flow to be controlled in a targeted manner. This means that you can use data to decide what to do next.

When the weather is sunny, I go out to play soccer. Otherwise I prefer to stay at home and watch TV. So the length of the weather is a factor that I use to make a decision about what I'm going to do.

Key elements

Logical Operators:

&&	and
||	or
!	The contrary

An instruction can come directly after the rule, if more than one instruction is required, they must be specified in {} brackets.

Structure

```
if ( expression )    if (expression && expression ||  expression )
{                    {
  …                    …
}                     }
```

If after an IF control structure, another "or if" should follow, or an "otherwise", that's how it works:

```
if (…)               if (…)       if (…)         if (…)
else if (…)          else         else if (…)    else if (…)
                                  else if (…)    else if (…)
                                  else if (…)    else if (…)
                                                 else
```

An expression consists of two sources and a comparison operator:

```
if( X == Y)   oder   if(x === y)    -> When x and y are equal
if( X != Y )                        -> When x and y are different
if( X > Y   )                       -> When x is greater than y
if( X >= Y )                        -> When x is greater than or equal to y
if( X < Y   )                       -> When x is less than y
if( X <= Y )                        -> When x is less than or equal to y
if(x)                               -> x is a valid expression
```

```
// if x
// equals 5
// not to be
if ( !x == 5)

// If the
// element
// not to be
// exist
if ( ! element )

// if a
// and b
// are not
// different
if ( ! a != b )
```

Controller – If - Samples

Login Process

```
if ( user === dbuser &&
      pass === dbpass )
{
    login = true;
    userid = dbuserid;
    username = dbuser;
    console.log(`Hallo ${username}`);
}
else
{
     console.log(`Benutzername oder
Passwort falsch`);
}
```

What if…

```
if ( money > 0 )
{
    console.log(`Du bist mit ${money}
im Plus! Herzlichen Glückwunsch.`);
}
else if ( money == 0 )
{
    console.log(`Kein Geld auf Konto!`);
}
else if ( money < 0 )
{
    console.log(`Du bist mit ${money}
im Minus! Verdiene Geld!!!`);
}
```

Check availability

```
If ( document.getElementById(`element`)  )
{
    console.log(`Das Objekt existiert`);
}
else
{
    console.log(`Das Objekt existiert NICHT!`);
}
```

Controller - switch

What is that?

Auch switch ist eine Kontrollstruktur. Sie prüft aber nur auf “ist gleich“ und „sonst“.

Wenn das WETTER…

- ✓ Sonnig ist, gehe ich Fußball spielen
- ✓ Bewölkt ist, gehe ich einkaufen
- ✓ Regnerisch ist, gehe ich zum Rathaus
- ✓ Sonst bleibe ich zuhause und schaue fern

Key elements

The switch statement always has a {} bracket. Below this are all cases and the otherwise case (default).

Aufbau

```
switch ( expression )
{
    case value1:
        instruction(en);
        break;
    case value2:
        instruction(en);
        break;
    default:
        instruction(en);
        break;
}
```

The expression of a "switch" is usually a single variable, but it can be anything:

```
switch( x )
switch( array.length )
switch( array[element] )
switch( foo() )
switch (foo(a,b,c) )
```