p5.js im Browser http://p5js.org/

PROGRAMMING WITH P5.JS

@FH-Potsdam

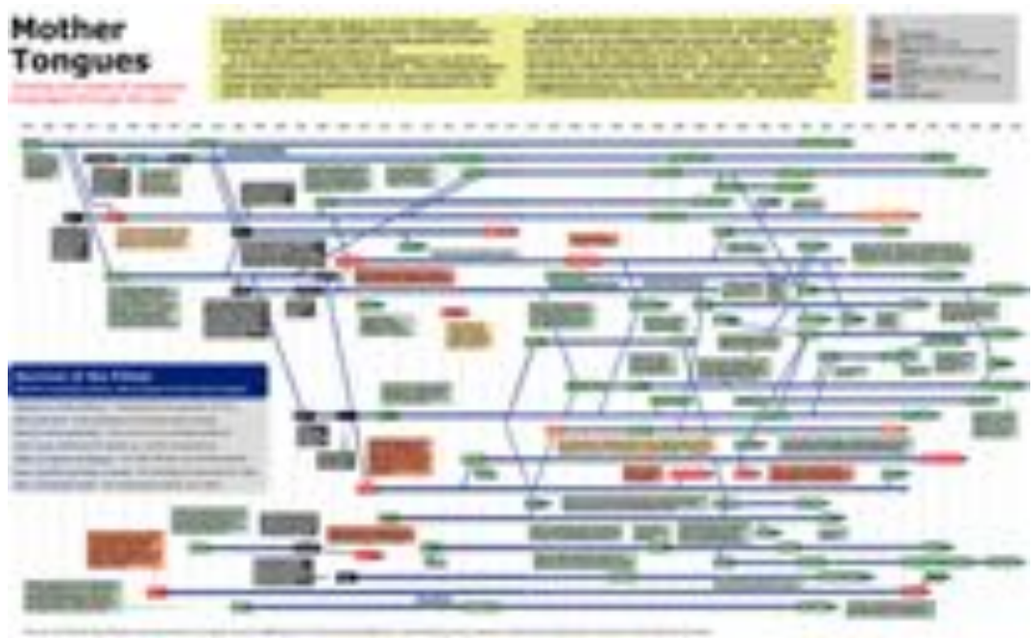
by Fabian Morón Zirfas

Winter 2015/2016

"Don’t Panic"

The Hitchhiker's Guide to the Galaxy

Keine Panik. Zu Beginn ist programmieren sehr komplex und abschreckend. Es wird nicht sofort einfach zugänglich sein. Wenn ihr jedoch Interesse habt, bleibt dran. Mit etwas Zeit kann es für euch ein Werkzeug werden wie Photoshop, ein Pinsel oder Ton.



Es gibt viele Sprachen.

We are here.

Processing

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

Wikipedia

https://en.wikipedia.org/wiki/Processing_(programming_language)

p5js.org

p5js enter the stage. Crowd goes wild. 2013

Hello! p5.js is a JavaScript library that starts with the original goal of Processing, to make coding accessible for artists, designers, educators, and beginners, and reinterprets this for today's web.

Using the original metaphor of a software sketchbook, p5.js has a full set of drawing functionality. However, you're not limited to your drawing canvas, you can think of your whole browser page as your sketch! For this, p5.js has addon libraries that make it easy to interact with other HTML5 objects, including text, input, video, webcam, and sound.

p5.js is a new interpretation, not an emulation or port, and it is in active development. An official editing environment is coming soon, as well as many more features!

Erster Commit auf Github ist im Februar 2013. Master Arbeit von Lauren McCarthy. Idee von Processing aber als Native JS Bib

PREREQUISITES

- Texteditor - Atom.io oder sublimetext.com/3
- p5.JS Libray - p5js.org/download/
- Browser Google Chrome
- NodeJs https://nodejs.org (v4.2.2)

TEST

TEST

Nodejs

1. open Terminal.app or CMD
2. type: `node -v`
3. hit: ↩
4. type: `cd`
5. add a whitespace behind the cd
6. drag + drop a desired folder behind it
7. hit: ↩
8. create a file (in that folder) called index.js with the content:
   `console.log("Hello Nodejs")`
9. type: `node index.js`
10. hit: ↩

CONGRATS

You did he first step in becoming a
super duper pro hacker

USEFUL ATOM PACKAGES

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

Okay. Wenn ihr den Atom editor verwendet solltet ihr euch später dies zustatzmodule installieren. Jetzt Butter bei die Fische.

USEFUL SUBLIME PACKAGES

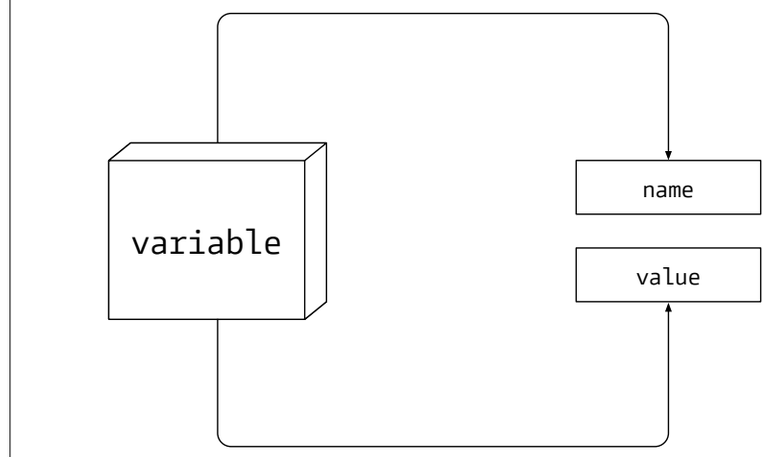
install via packagecontrol.io, Emmet, CodeFormatter, SublimeLinter, SublimeLinter-contrib-eslint, SublimeLinter-contrib-htmlhint

Okay. Wenn ihr den Sublime editor verwendet solltet ihr euch später dies Zustatzmodule installieren. https://packagecontrol.io/

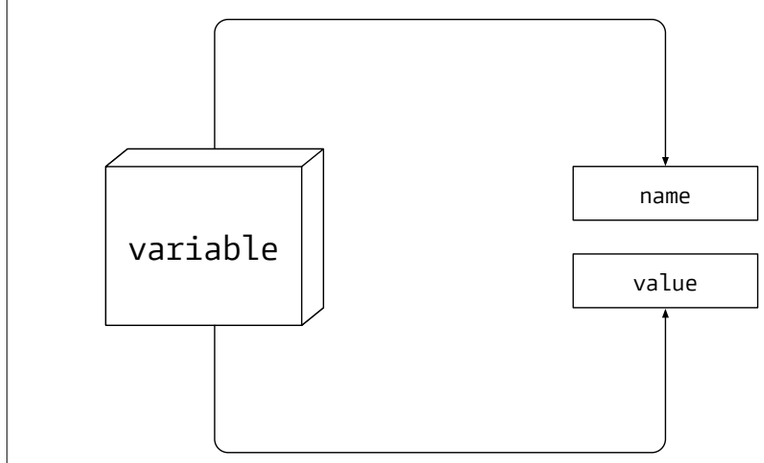
7 BASIC THINGS IN PROGRAMMING

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

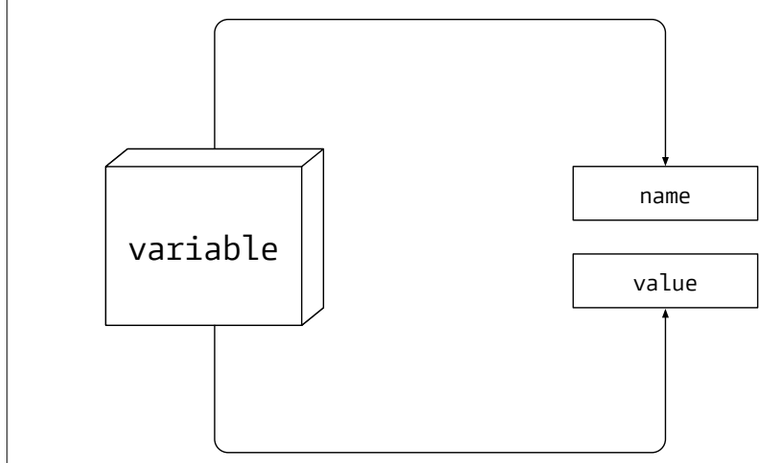
Bevor wir uns die Hände schmutzig machen möchte ich noch 6 Dinge mit euch besprechen.



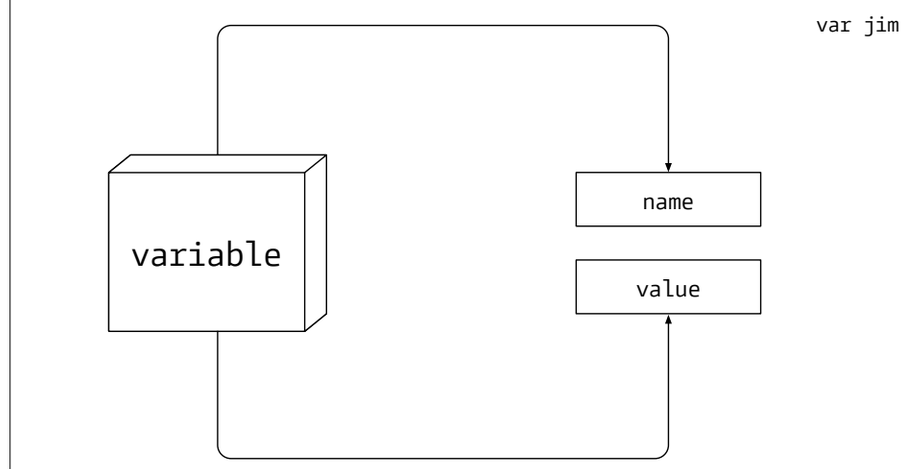
Eine variable ist wie ein Kiste



damit mein Programm weiss das es eine neuen Variable hat muss ic sie deklarieren. In JS mit var

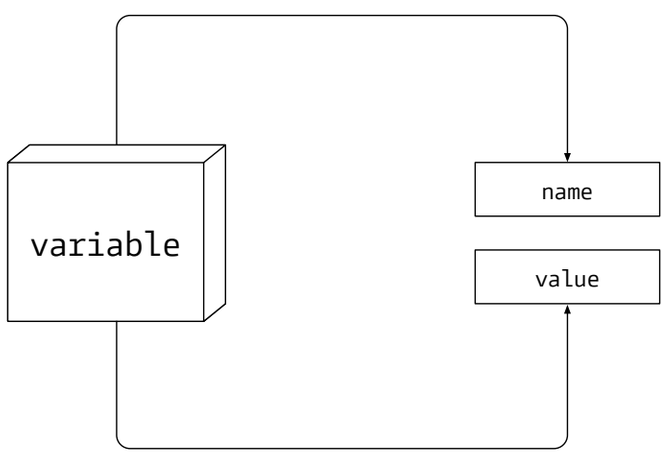


damit mein Programm weiss das es eine neuen Variable hat muss ic sie deklarieren. In JS mit var



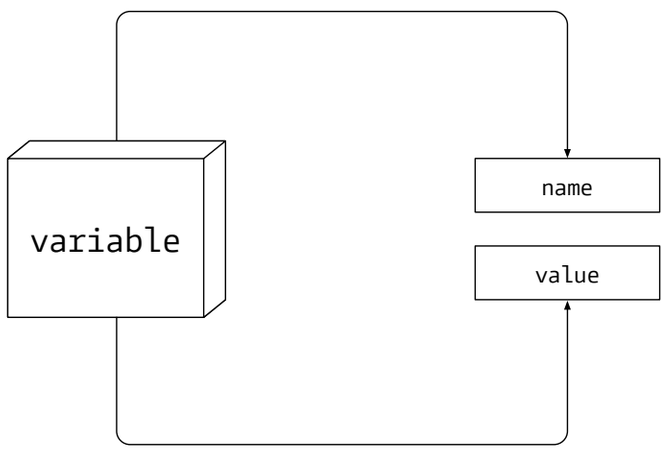

```
var jim
```

eine Variable hat einen Namen



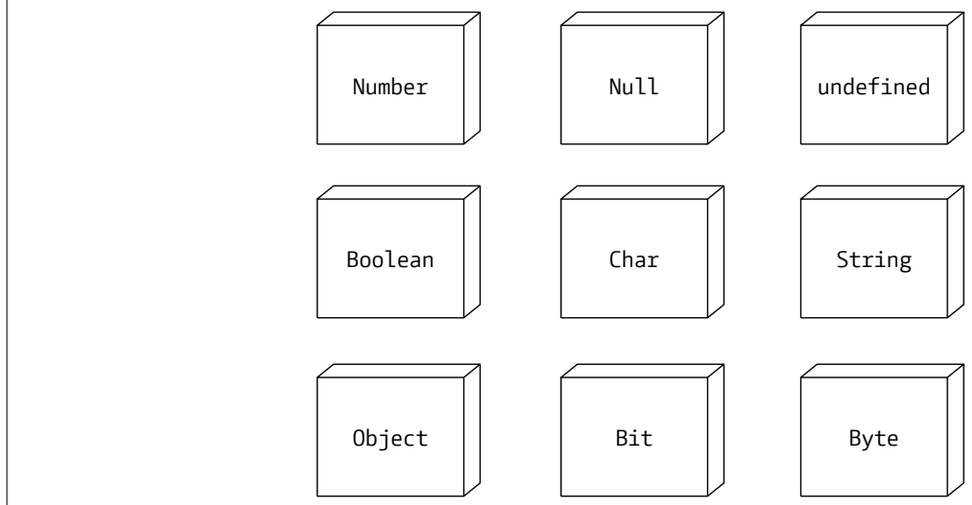

```
var jim = 100
```

eine Variable hat auch einen Wert der über = zugewiesen wird

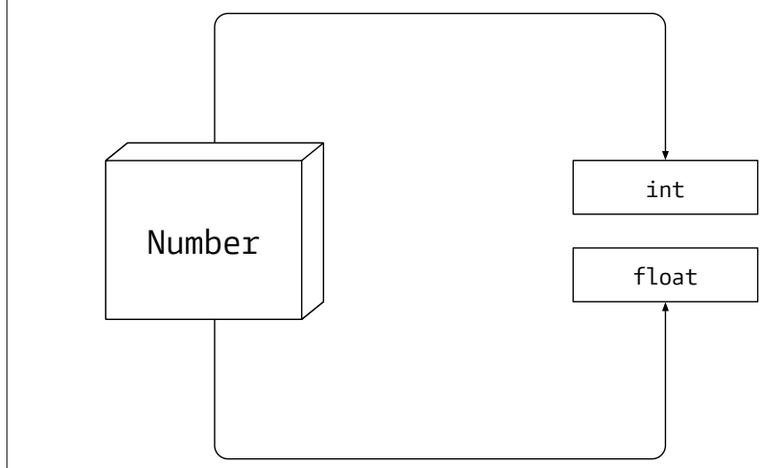



```
var jim = 100;
```

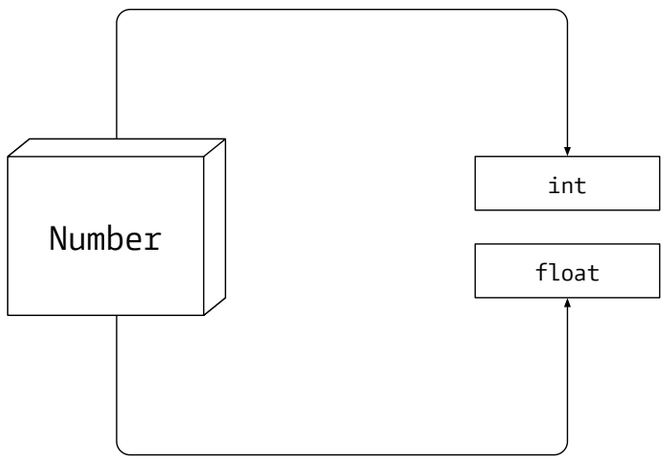
!ACHTUNG Jede Zeile endet mit einem ";"



Es gibt unterschiedliche Arten von Variablen

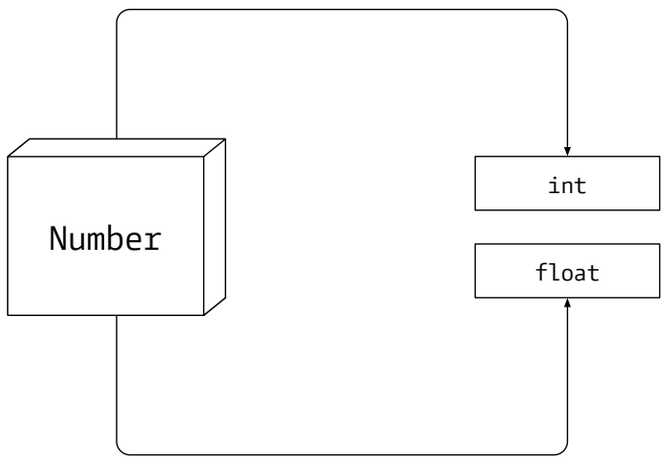


zB Zahlen, Ganze Zahlen oder integer



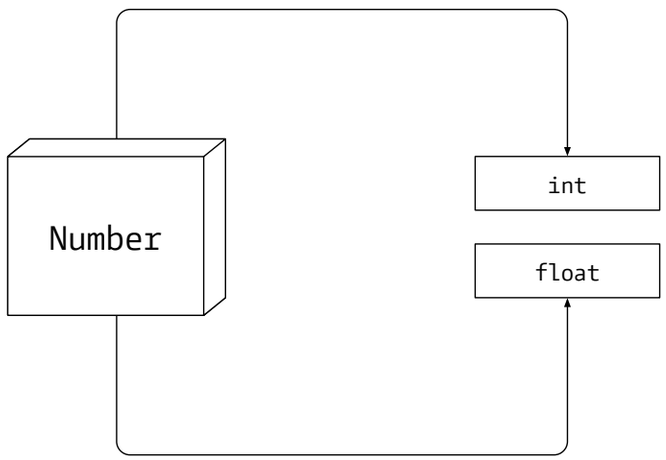

```
var x = 13;
```

zB Zahlen, Ganze Zahlen oder integer x = 13



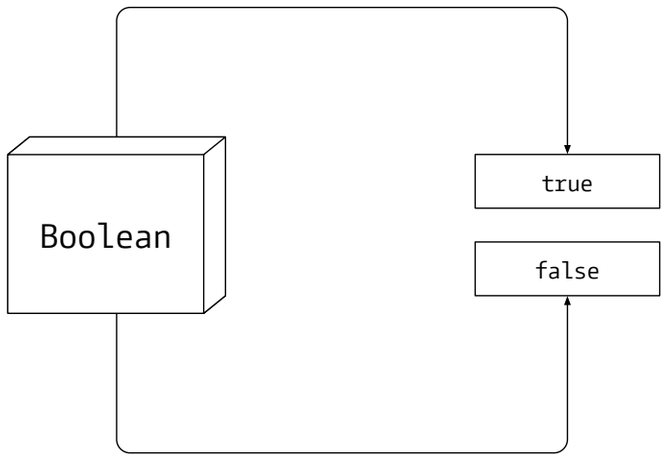

```
var x = 13;
var y = -42;
```

zB Zahlen, Ganze Zahlen oder integer y = -42;

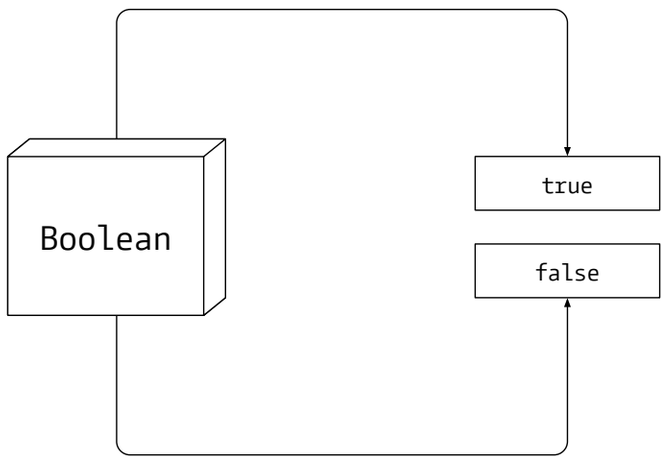



```
var x = 13;
var y = -42;
var z = -23.5;
```

JS unterscheidet grundsätzlich nicht zwischen Fließkomma Zahlen und integer

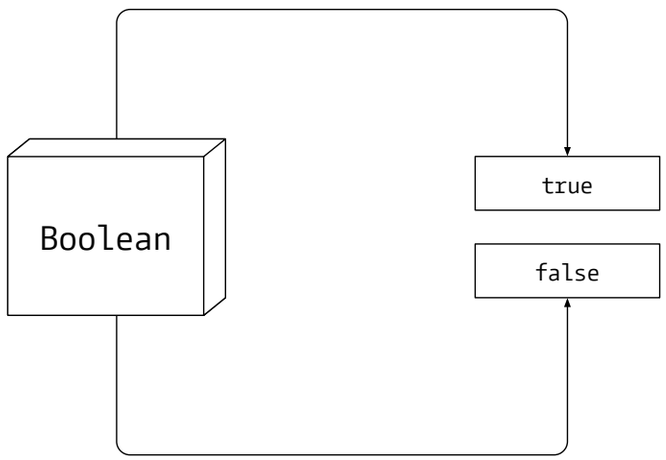


Ein anderer Typ an Variablen ist dr Boolean



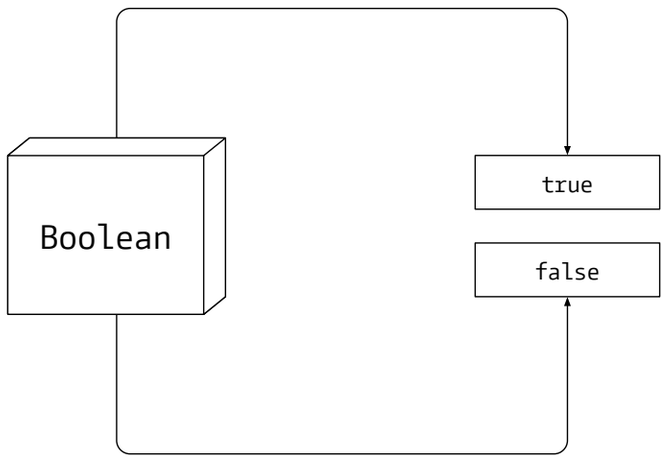

```
var status = true;
```

der kann entweder wahr oder falsch sein. Also true

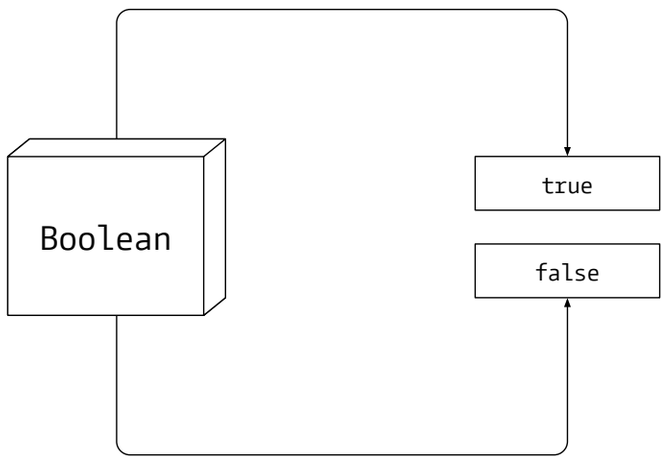



```
var status = false;
```

oder false

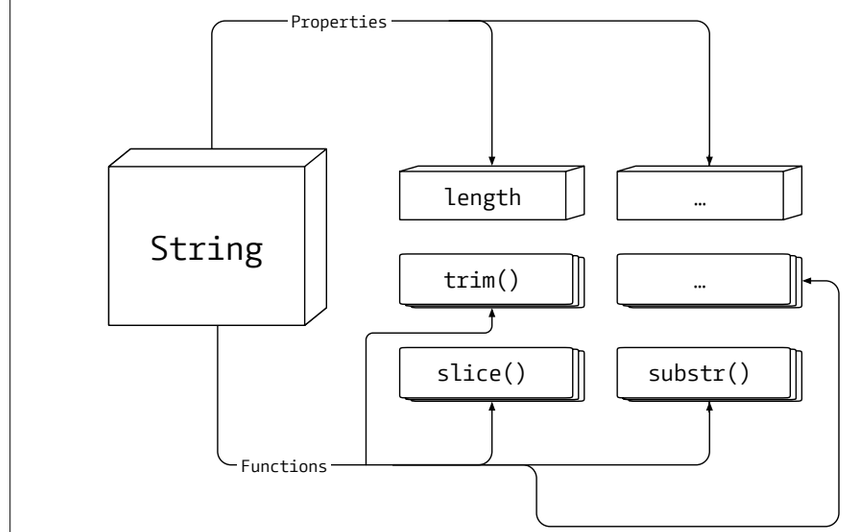



```
var status = false;

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

status = true;
```

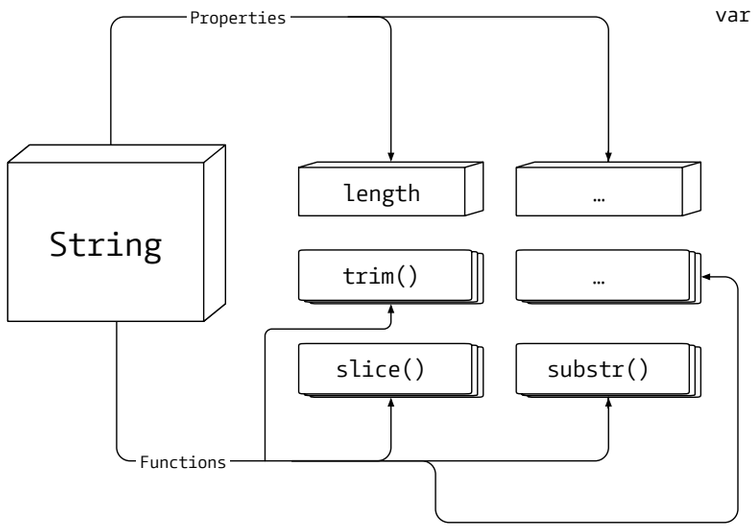



```
var status = false;

// something happens
// lets switch status

status = !status; // <— is true

// something else happens
// lets switch status again

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

Wir können bei boolean ebenfalls einfach sagen, dass wir gerne das Gegenteil hätten

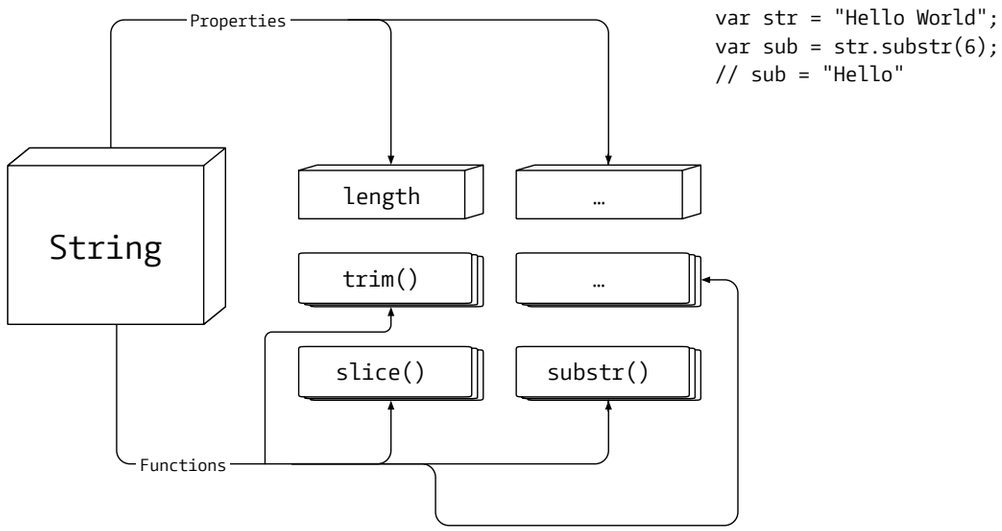


Dann gibts es auch noch Strings



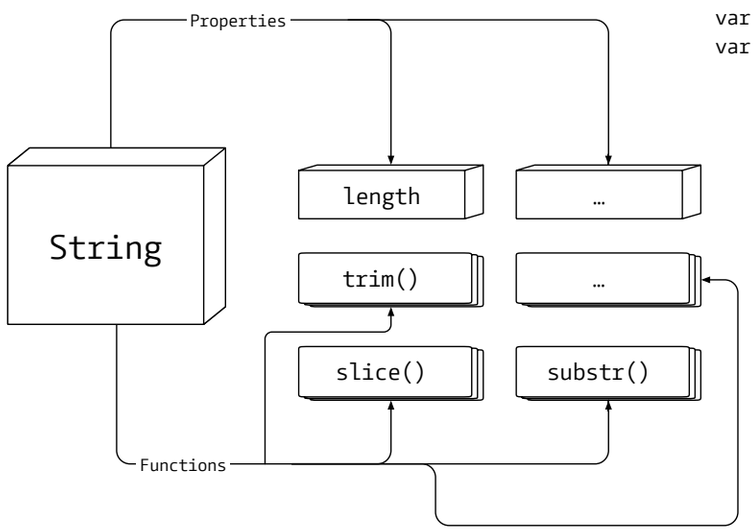

```
var str = "Hello World";
```

Strings sind Zeichenketten. zum Beispiel "Hello World" in Anführungszeichen geschrieben



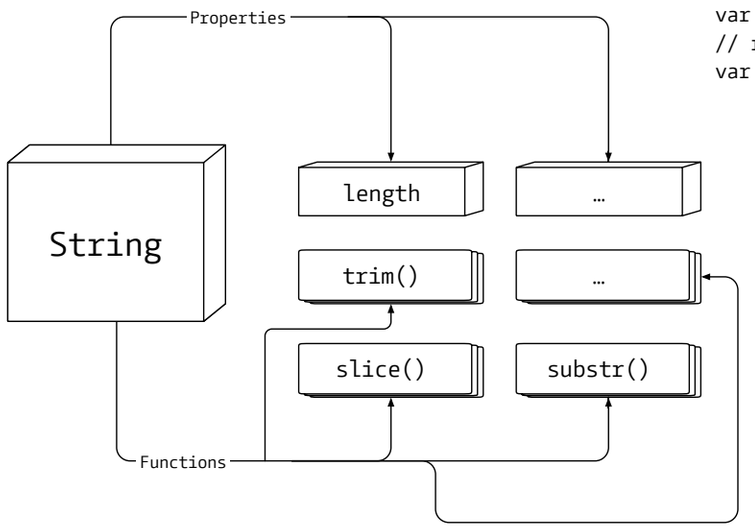

```
var str = "Hello World";
var sub = str.substr(6);
// sub = "Hello"
```

Strings. Haben auch noch zusätzliche Funktionen um mit ihnen besser arbeiten zu können. zB trim und substr() Sie haben auch Eigenschaften zB length



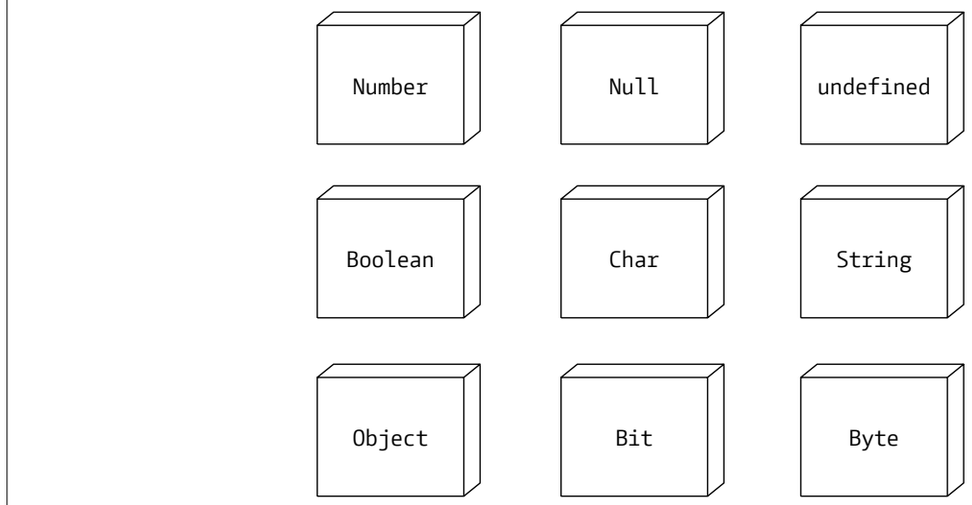

```
var str = "Hello World";
var len = str.length; // 11
```

Strings. Haben auch noch zusätzliche Funktionen um mit ihnen besser arbeiten zu können. zB trim und substr() Sie haben auch Eigenschaften zB length




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

oder die Funktion trim() nimmt lehrreichen von einem string weg.

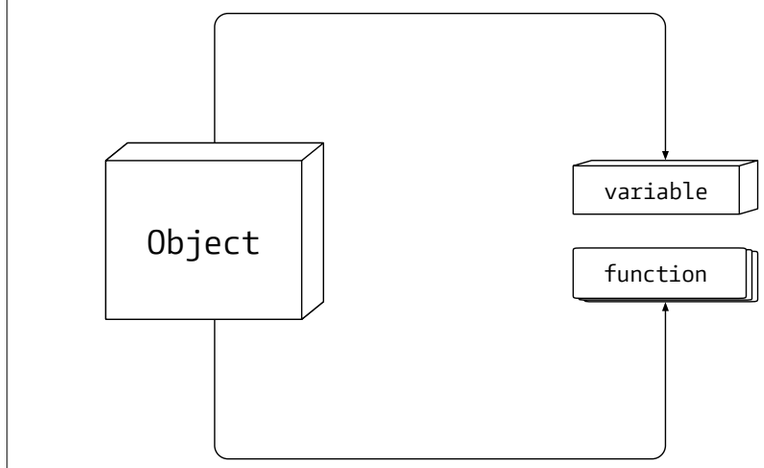


Es gibt noch andere Datentypen aber das sind erstmal genug.

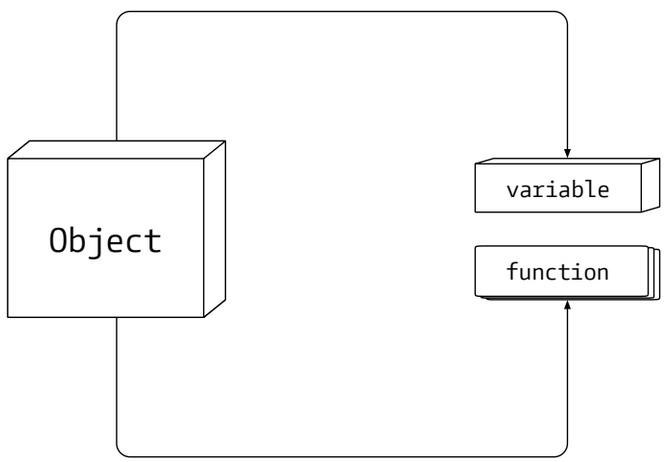
7 BASIC THINGS IN PROGRAMMING

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

Fragen? Objekte Hands on!

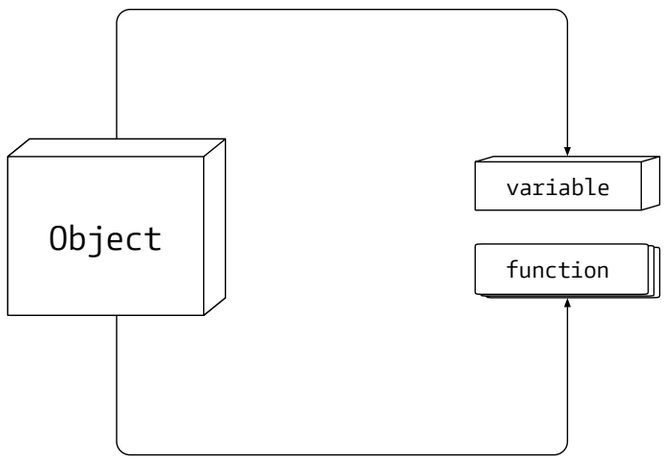


Objekte sind eigentlich auch nur Variablen. Der grosse unterschied ist, dass sie mehrere Variablen in sich haben können.



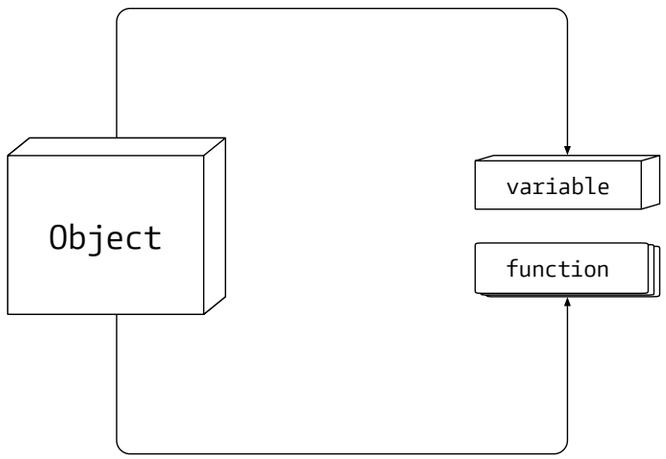

```
var obj = {};
```

Objekte sind eigentlich auch nur Variablen. Der grosse unterschied ist, dass sie mehrere Variablen in sich haben können.



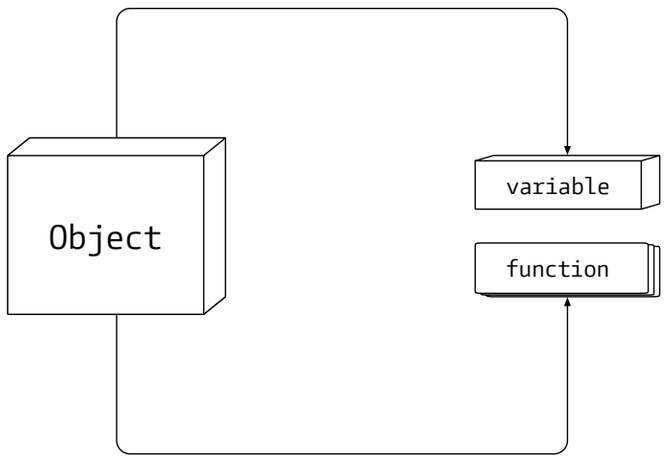

```
var obj = {"posx":5,"posy":20};
```

Objekte sind eigentlich auch nur Variablen. Der grosse unterschied ist, dass sie mehrere Variablen in sich haben können.



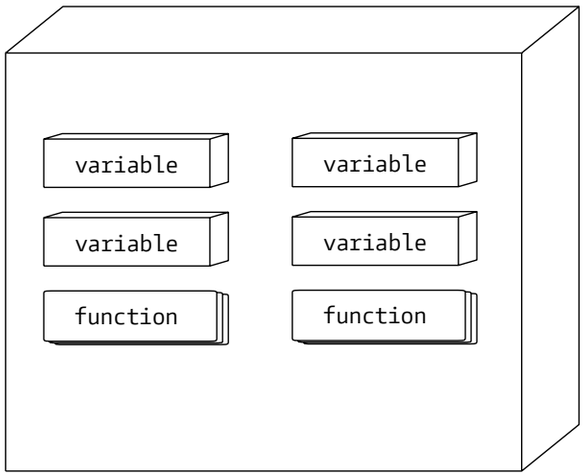

```
var obj = {"posx":5,"posy":20};

obj.something = true;

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

Objekte sind eigentlich auch nur Variablen. Der grosse unterschied ist, dass sie mehrere Variablen in sich haben können.

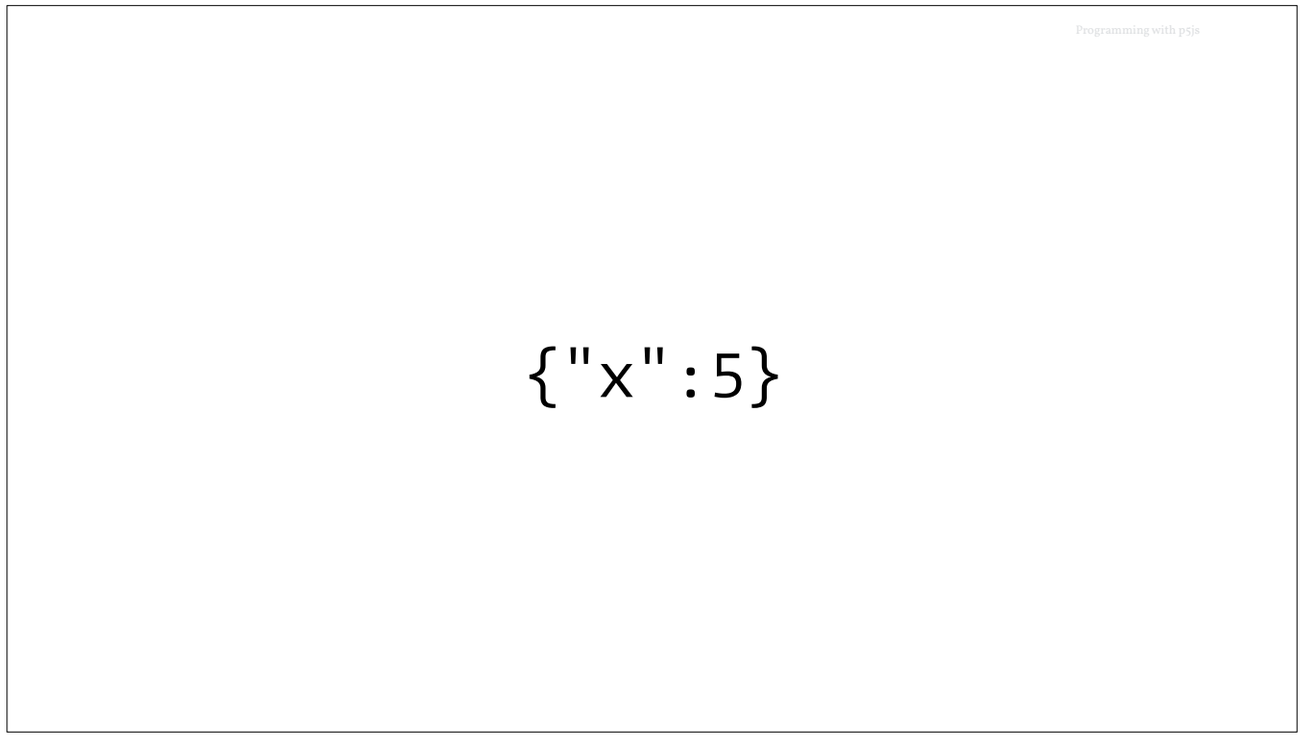



```
var obj = {
"add":function(a,b){
  return a+b;
  },
 "subtract":function(a,b){
    return a-b;
  }
};

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

Objekte sind eigentlich auch nur Variablen. Der grosse unterschied ist, dass sie mehrere Variablen in sich haben können.

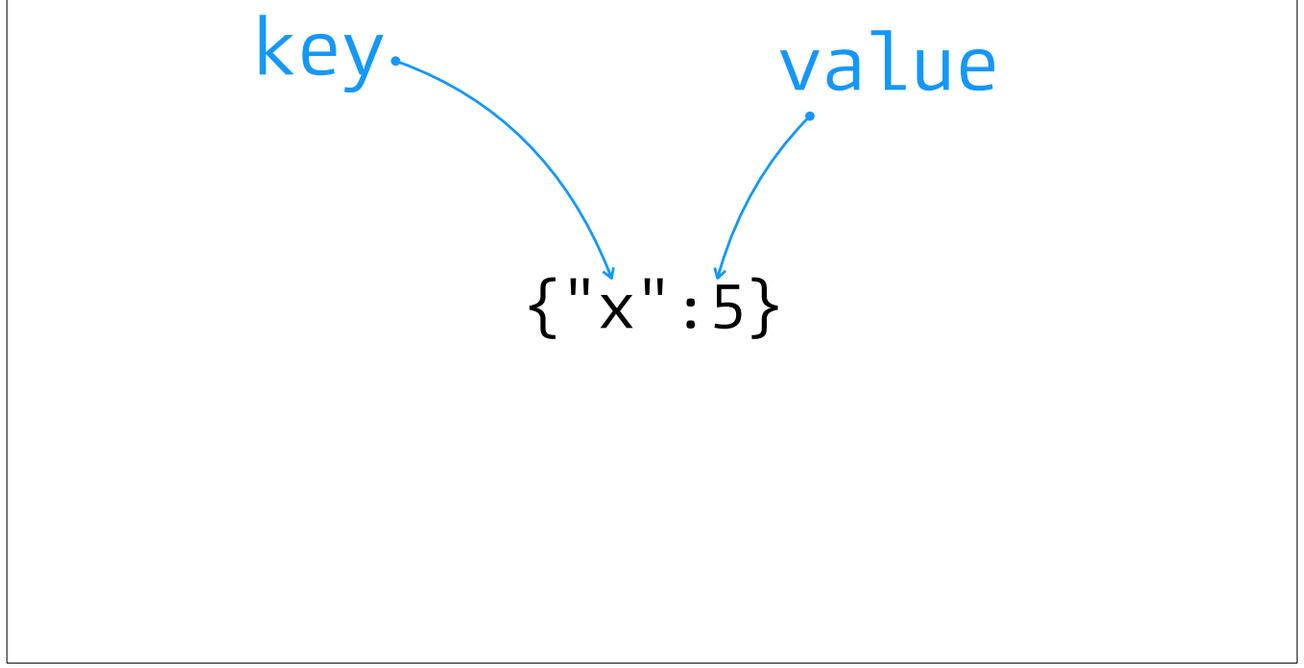


Ihr könnt euch das wie eine Kiste für andere Kisten vorstellen.



In einem Object haben wir immer einen "key" mit einem Wert.

Über den Key können wir einen Wert abrufen.



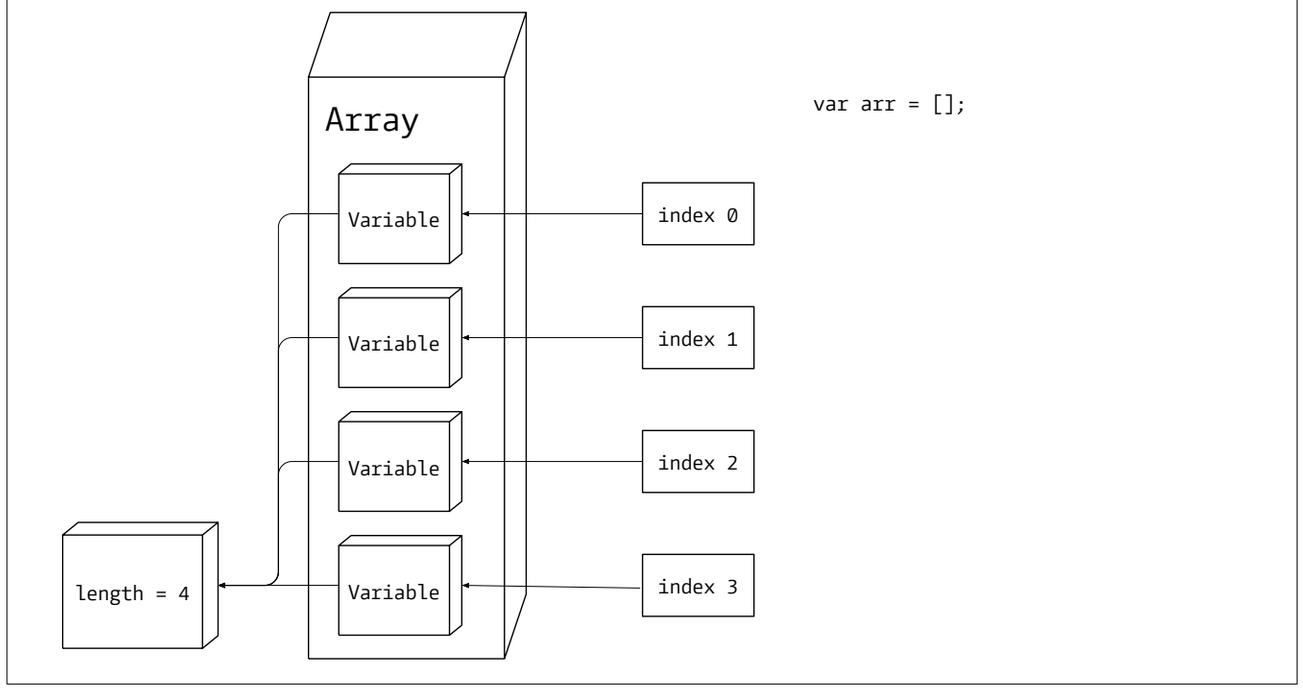
In einem Object haben wir immer einen "key" mit einem Wert.

Über den Key können wir einen Wert abrufen.

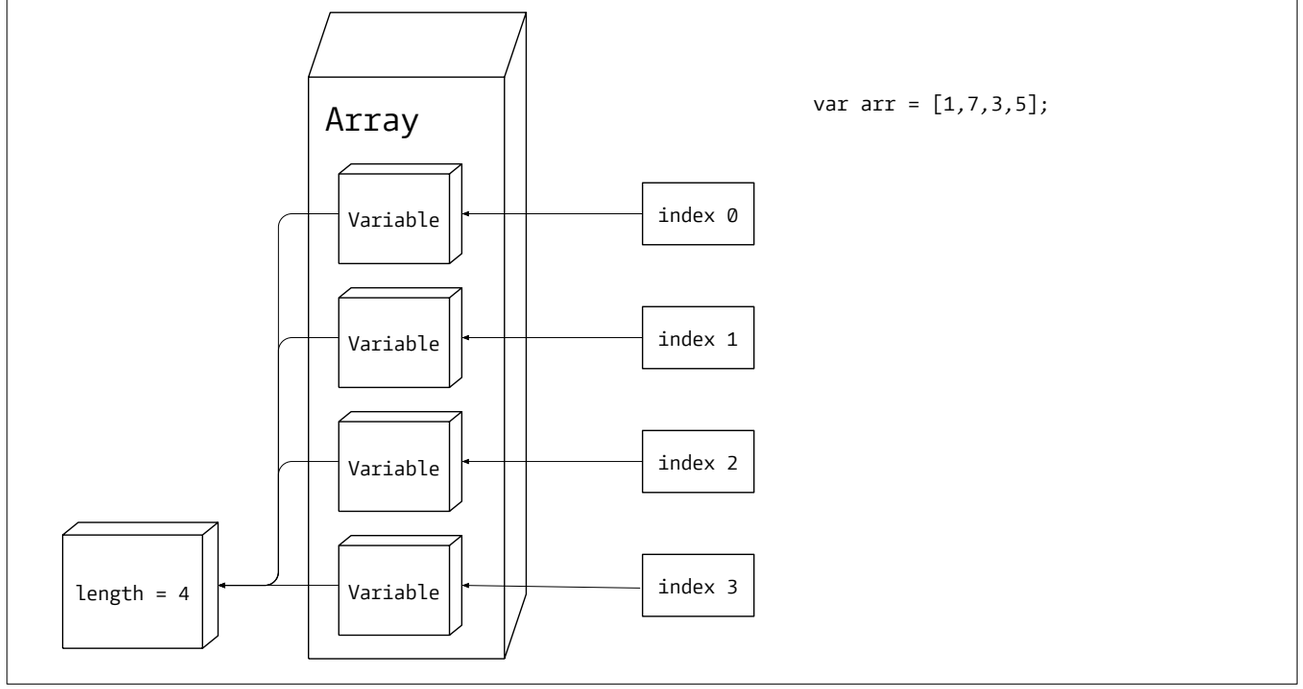
7 BASIC THINGS IN PROGRAMMING

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

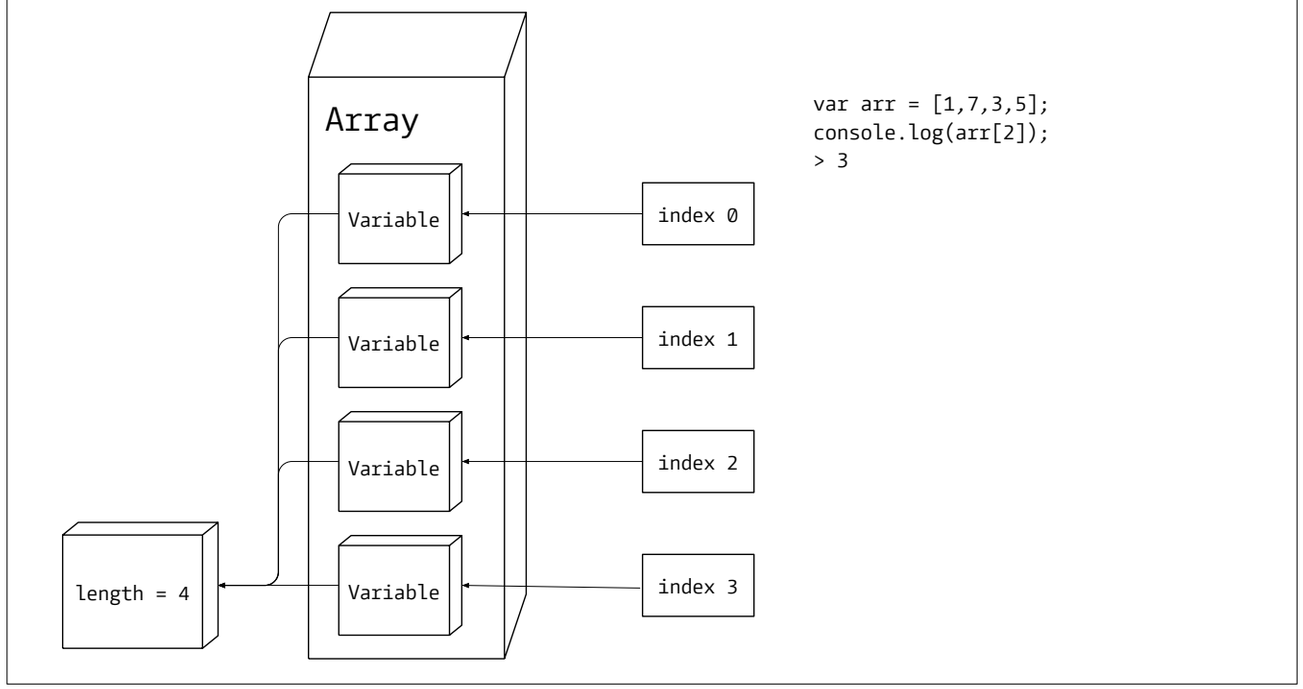
Fragen? Hands on



Ihr könnt euch das auch wie eine Kiste vorstellen in der andere Kisten sind oder eine Art Liste. Aus dieser grossen Kiste könnt ihr den Inhalt über einen Index ansprechen.

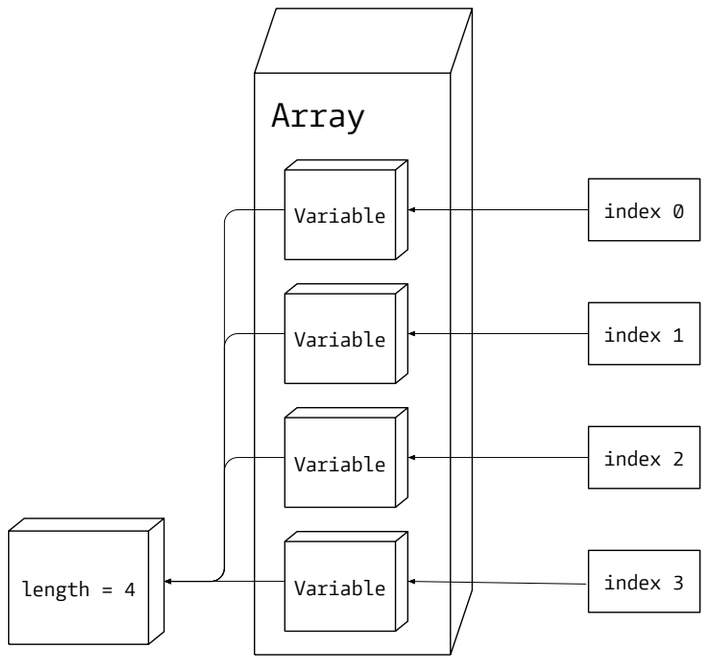


Wir können ein array gleich befüllen wenn wir es erschaffen



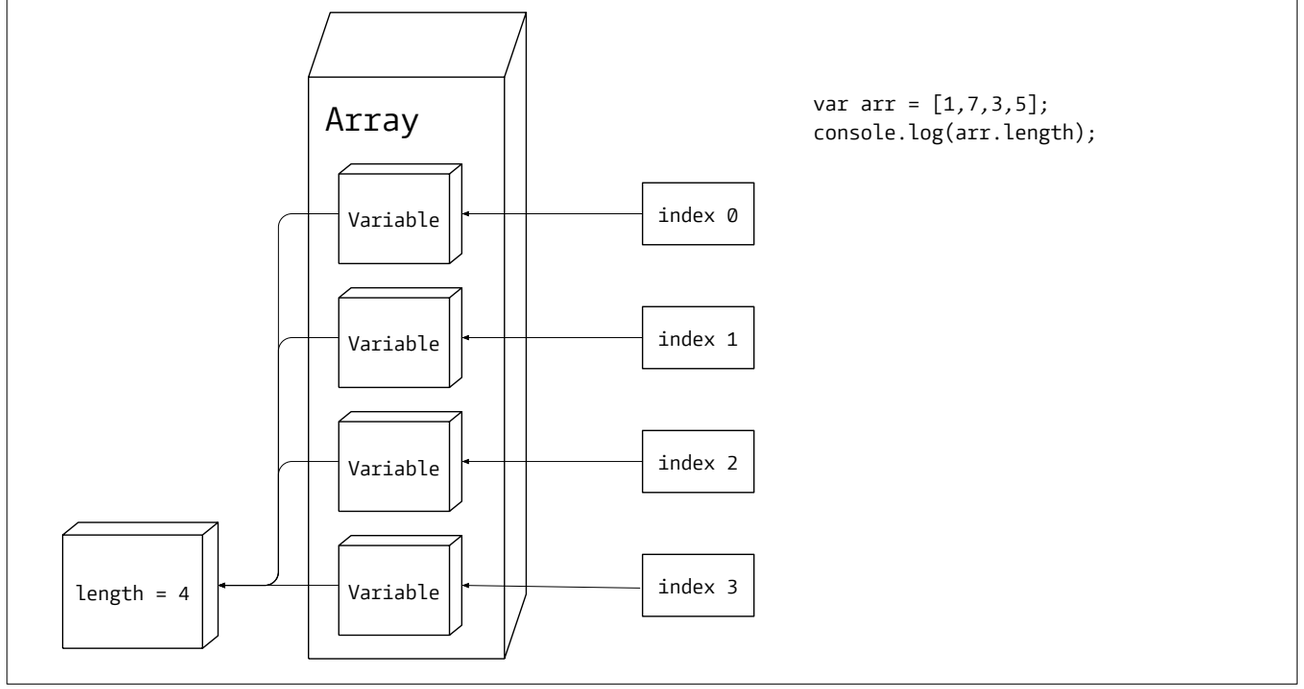

```
var arr = [1,7,3,5];
console.log(arr[2]);
> 3
```

und diese Inhalte dann über einen index abrufen

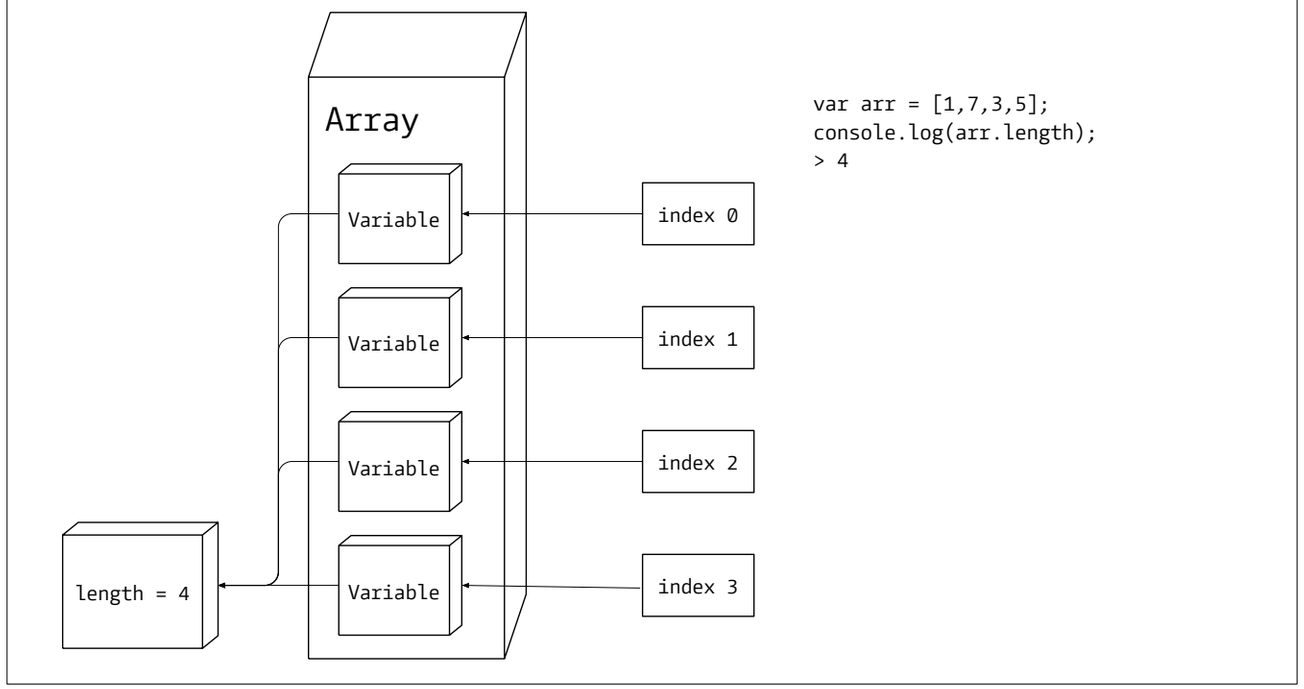



```
var arr = [];

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

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

Wir können das array auch nachträglich befüllen

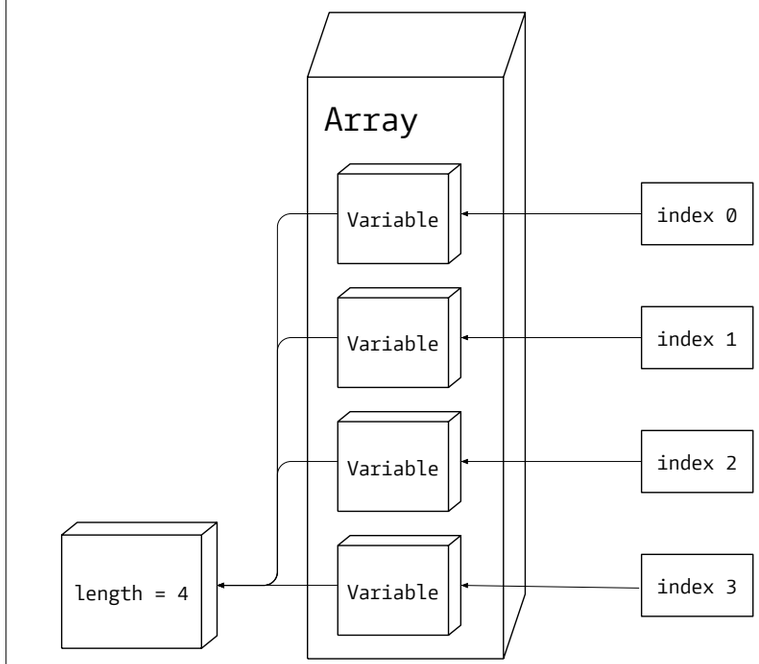


und so ein array hat auch wie der String Eigenschaften um damit zu arbeiten zB die length. Wie ist die Length von diesem Array?



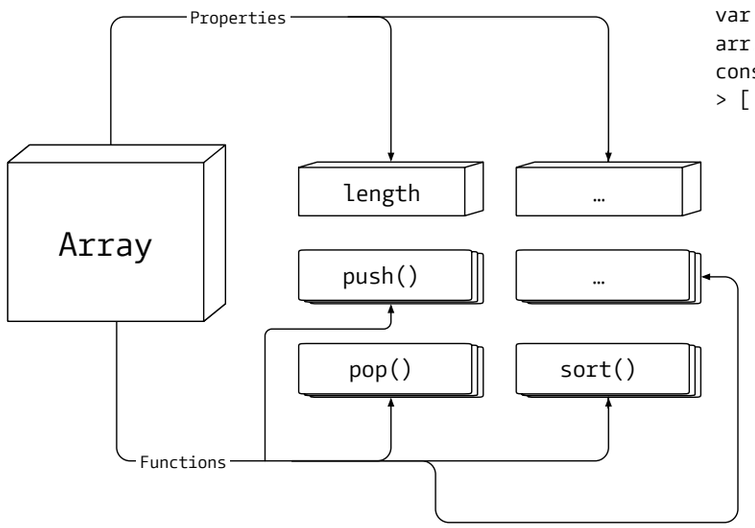

```
var arr = [1,7,3,5];
console.log(arr.length);
> 4
```

und so ein array hat auch wie der String Eigenschaften um damit zu arbeiten zB die length. Wie ist die Length von diesem Array?




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

4. dH die length ist immer einen grösser als der höchste index da der index bei 0 beginnt.



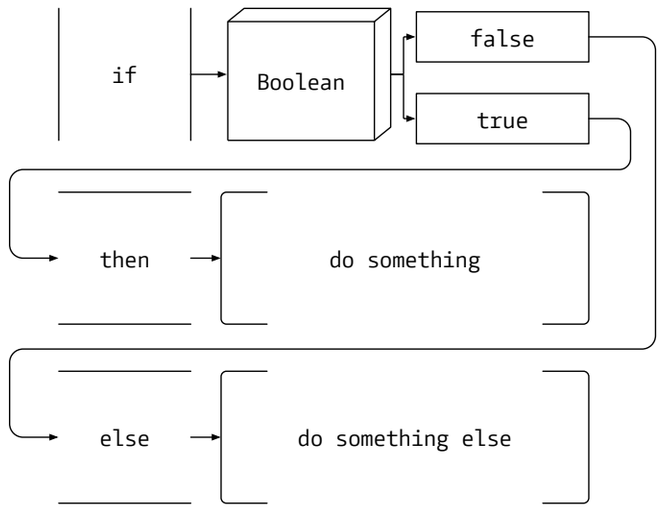

```
var arr = [1,7,3,5];
arr.push(23);
console.log(arr);
> [ 1, 7, 3, 5, 23 ]
```

es gibt noch andere Funktionen an einem Array. zB push fügt am ende eines Array ein Element an.

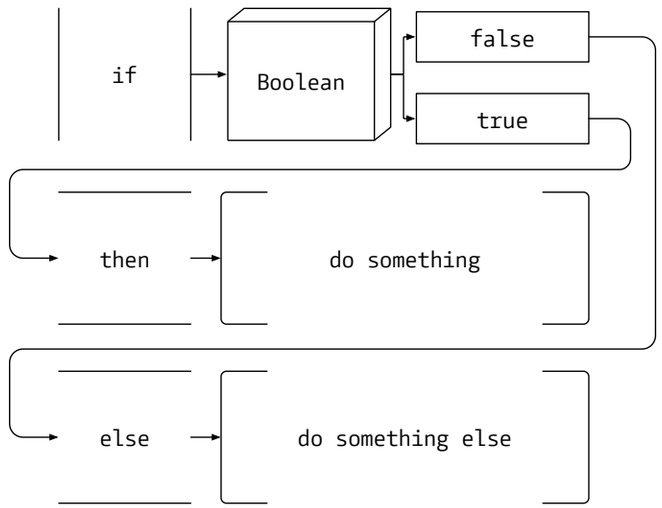
7 BASIC THINGS IN PROGRAMMING

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

?Hands on!



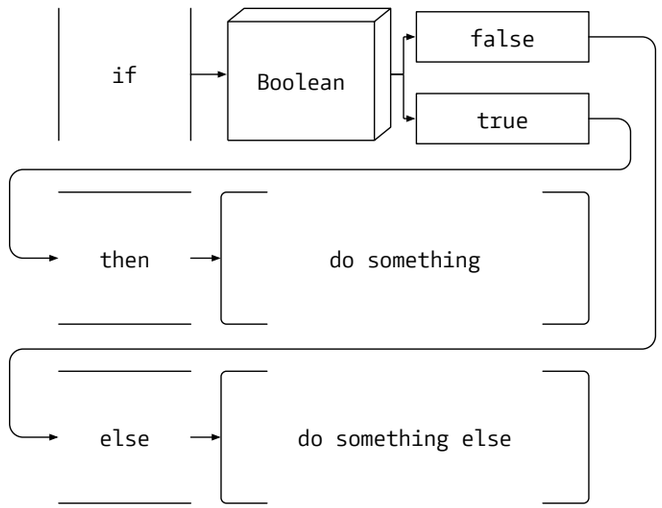
Kondition. Sind Fragen die testen ob eine Aussage wahr oder falsch ist.




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

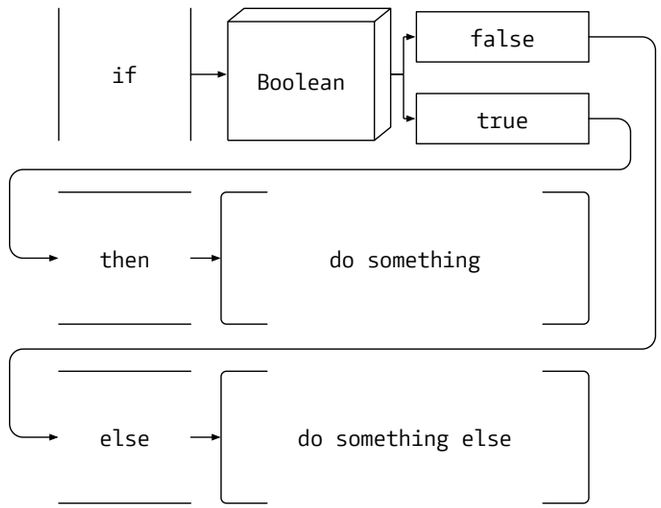
Wenn der Hund die Katze jagt dann öffne die Tür.

Hierbei wird geprüft ob ein Zustand wahr oder falsch ist und entsprechend reagiert.




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


// and, or, not
```

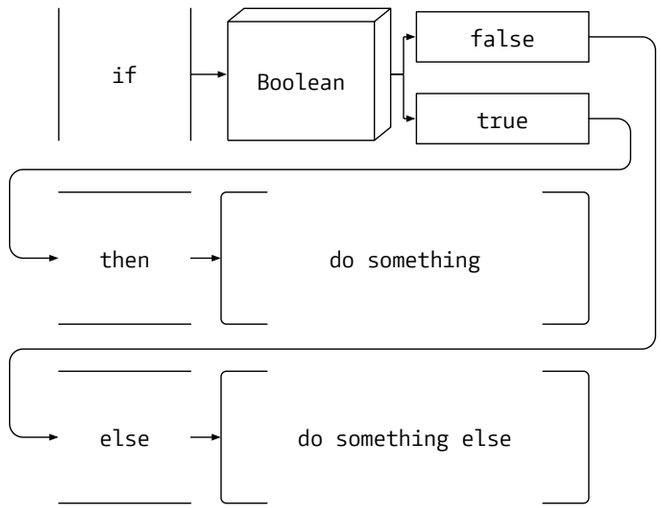
Es Kann natürlich auch geprüft werden ob mehrere Zustände zutreffen. Dann wird geprüft ob das gesamt Ergebnis war oder falsch ist. Also zB nur wenn sie sich jagen und drinnen sind.




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

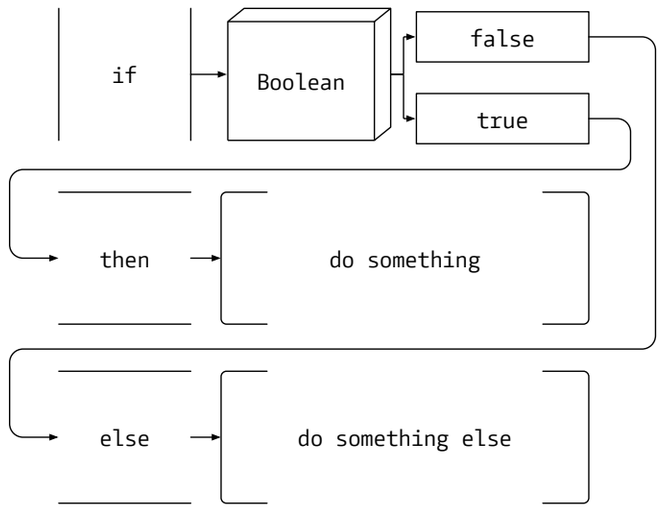
Das ganze geht auch mit mathematischen vergleichen.

Ist x kleiner als y? Dann mach das.




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

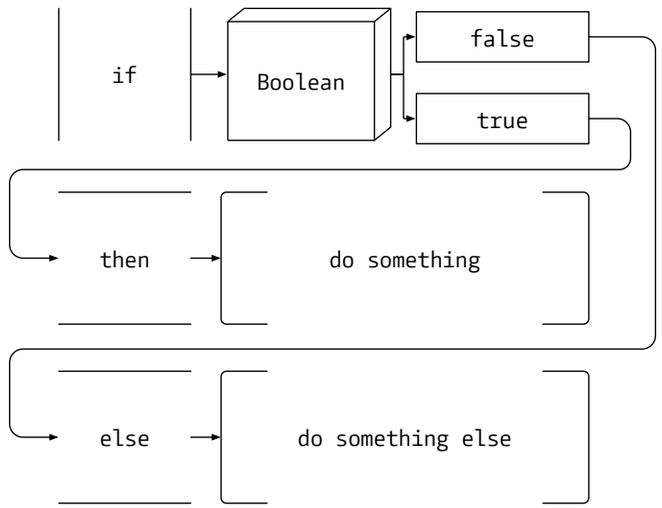
Mathematisch geschrieben sieht das dann so aus.




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

Wir kenn ebenfalls nicht nur agieren wenn etwas zutrifft sondern auch alle anderen Zustande greifen.

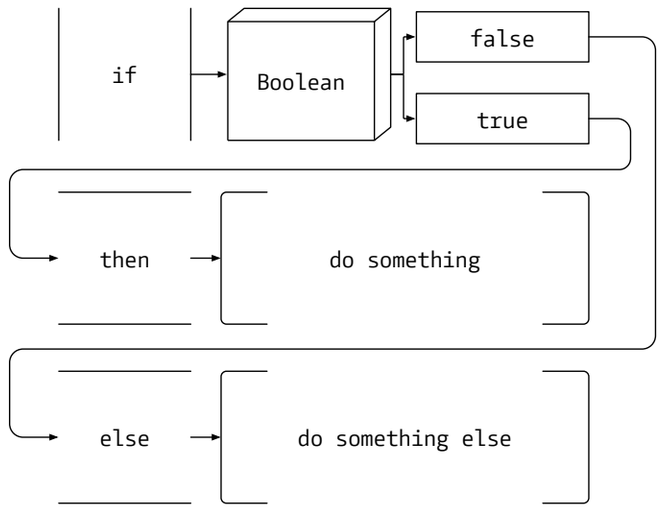
Also was ist wenn…




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

Wir kenn ebenfalls nicht nur agieren wenn etwas zutrifft sondern auch alle anderen Zustande greifen.

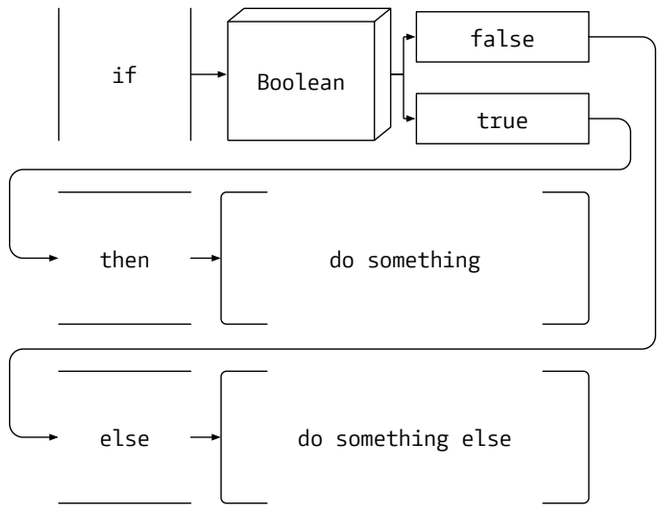
Also was ist wenn…




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

Wir kenn ebenfalls nicht nur agieren wenn etwas zutrifft sondern auch alle anderen Zustande greifen.

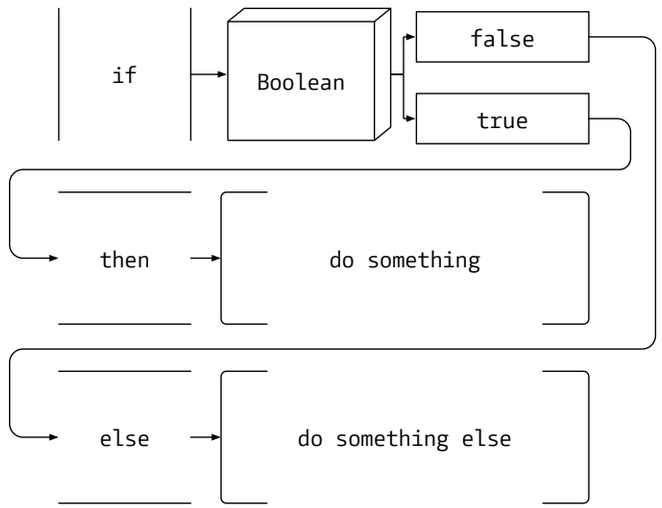
Also was ist wenn…




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

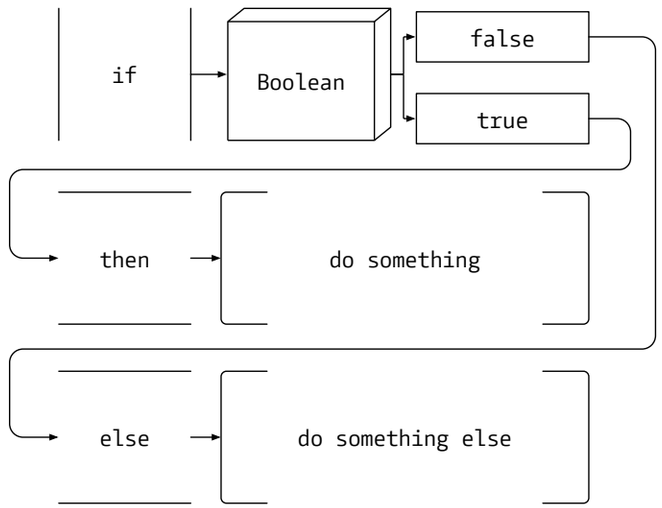
Wir kenn ebenfalls nicht nur agieren wenn etwas zutrifft sondern auch alle anderen Zustande greifen.

Also was ist wenn…

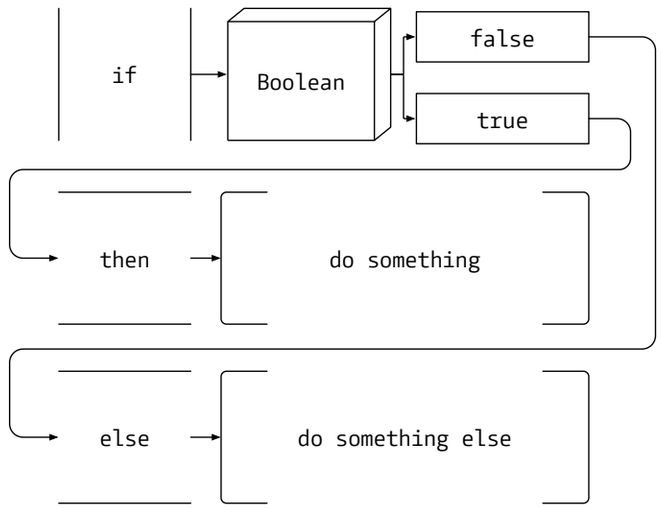



```
if
    x is smaller than y
    and
    x is smaller then 10
        do this
```

Wie im Katzen und Hunde Beispiel können wir auch mehrere mathematische Abfragen verknüpfen

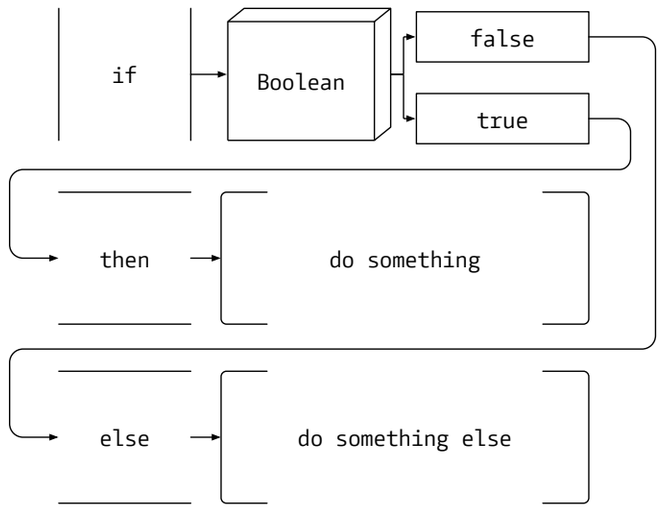



```
if
    x < y && x < 10
        do this
```




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

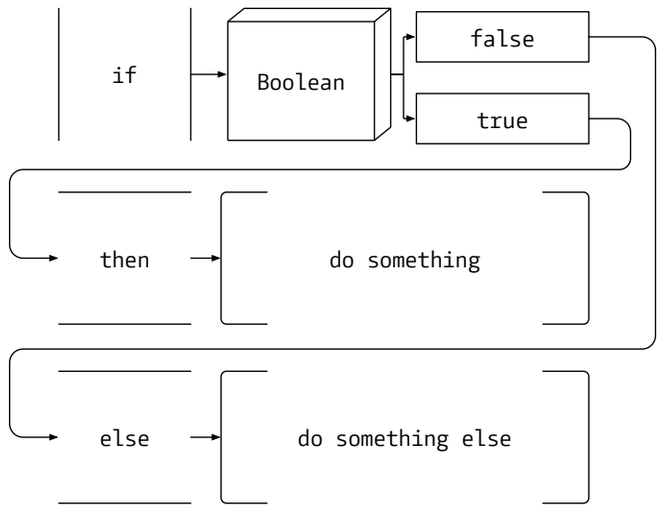
auch hier können wir verknüpfen.




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

auch hier können wir verknüpfen. Was ist das Ergebnis?

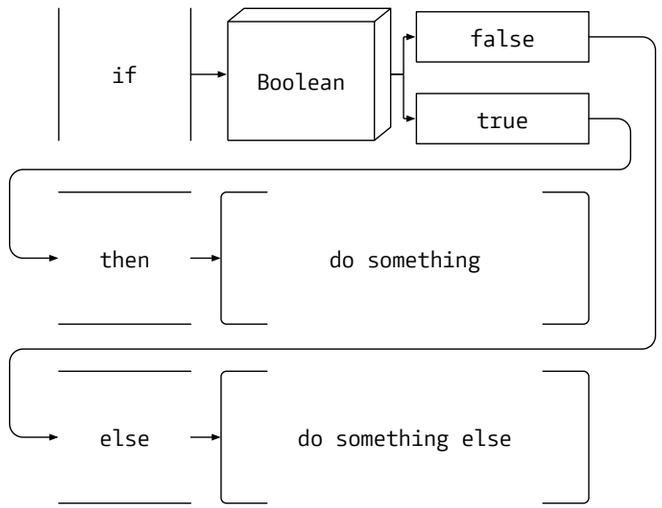
do that




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

???

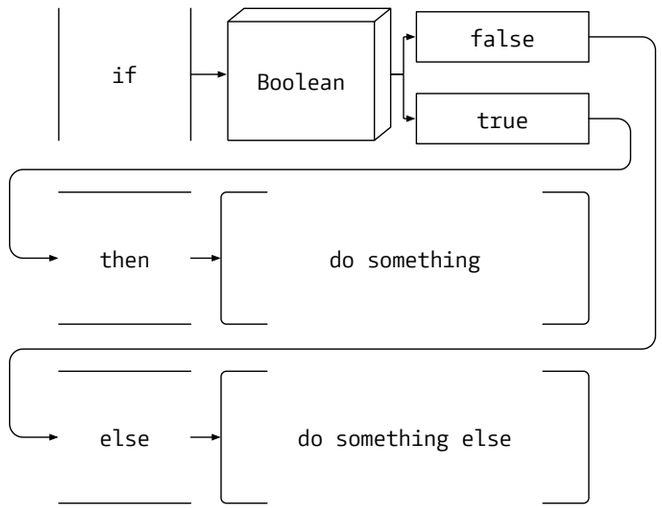
do that




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

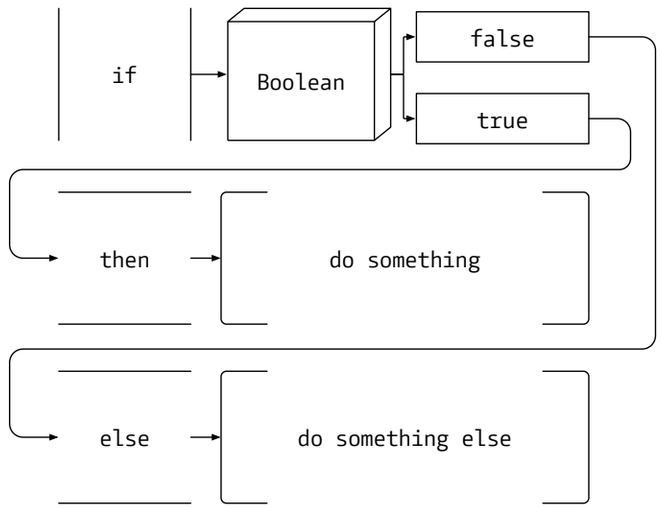
???

do this




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

else
        do something
```

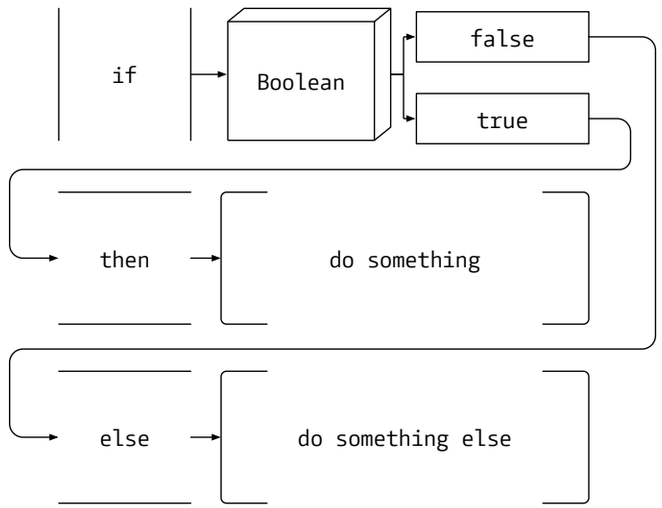
Es gibt noch einen weiter Möglichkeit zu verknüpfen. Mit Else if.




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

else
        do something
```

Es gibt noch einen weiter Möglichkeit zu verknüpfen. Mit Else if

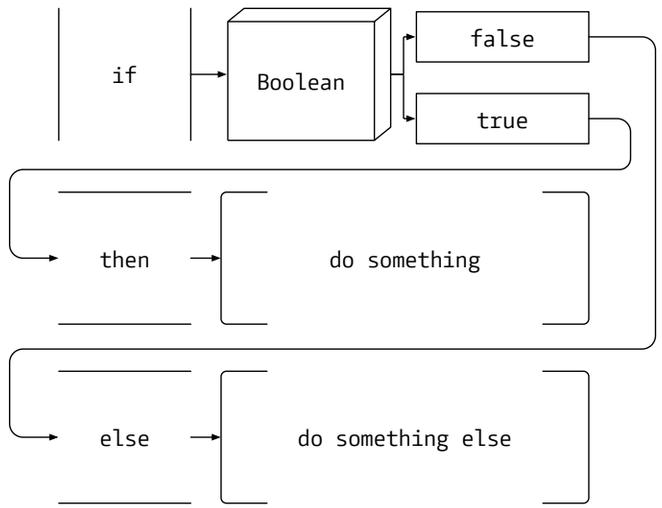
do that




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

else
        do something
```

Es gibt noch einen weiter Möglichkeit zu verknüpfen. Mit Else if

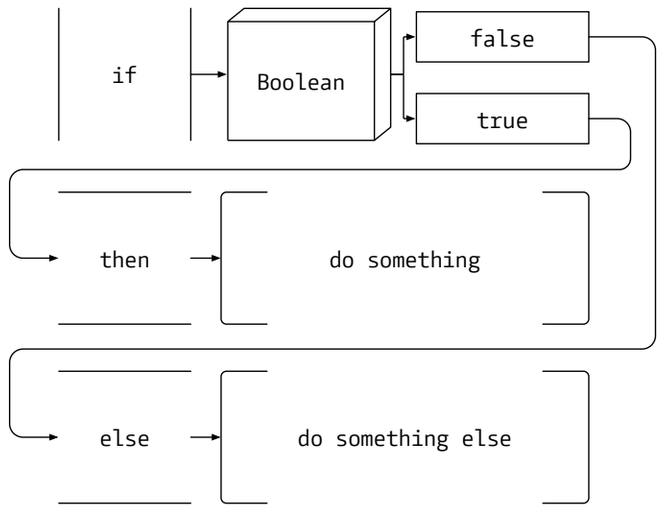
do this




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

else
        do something
```

Es gibt noch einen weiter Möglichkeit zu verknüpfen. Mit Else if

do something



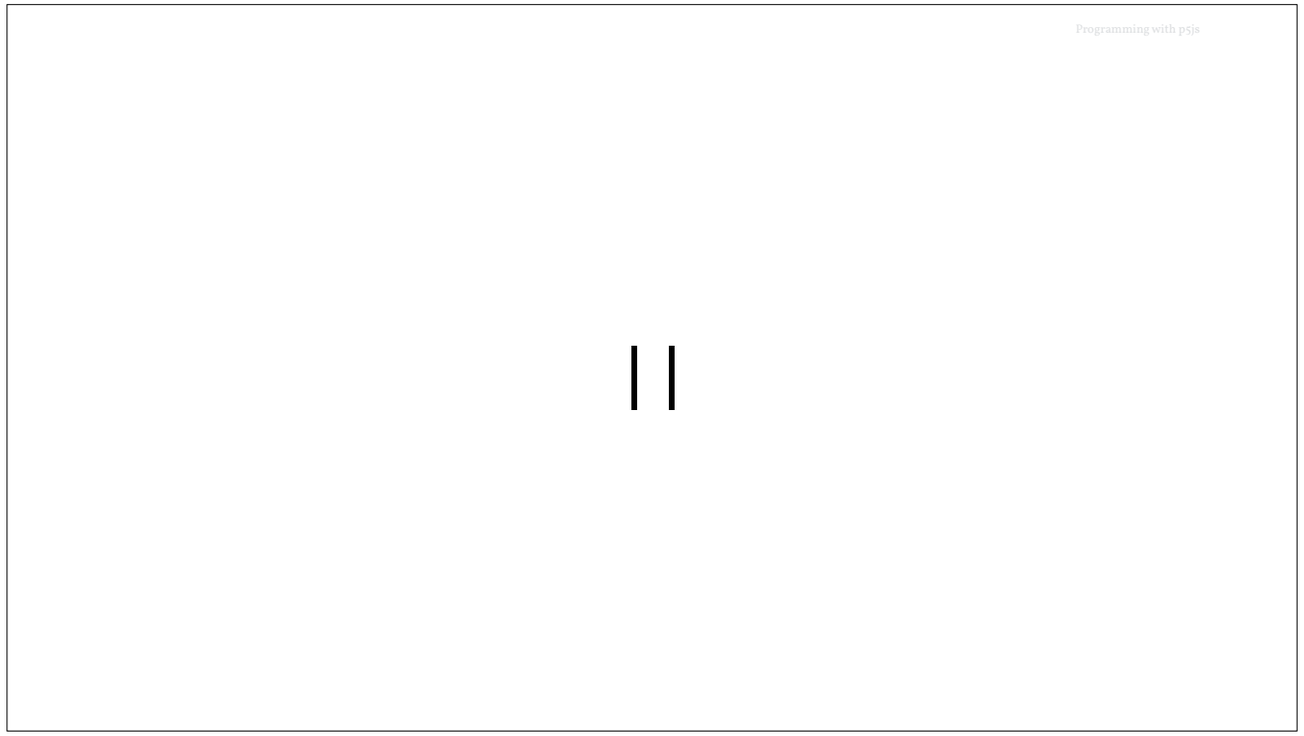

```
if(x < y && x < 10){
        // do this

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

}else{
        // do something

}
```

Komplett in Code ausgeschrieben wäre das dann so

Das war alles nur und also

&&

und



oder

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

und

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

oder

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

oder

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

oder

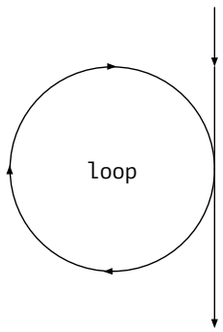
```
x !== 5 || 10 == "10"
```

oder

7 BASIC THINGS IN PROGRAMMING

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

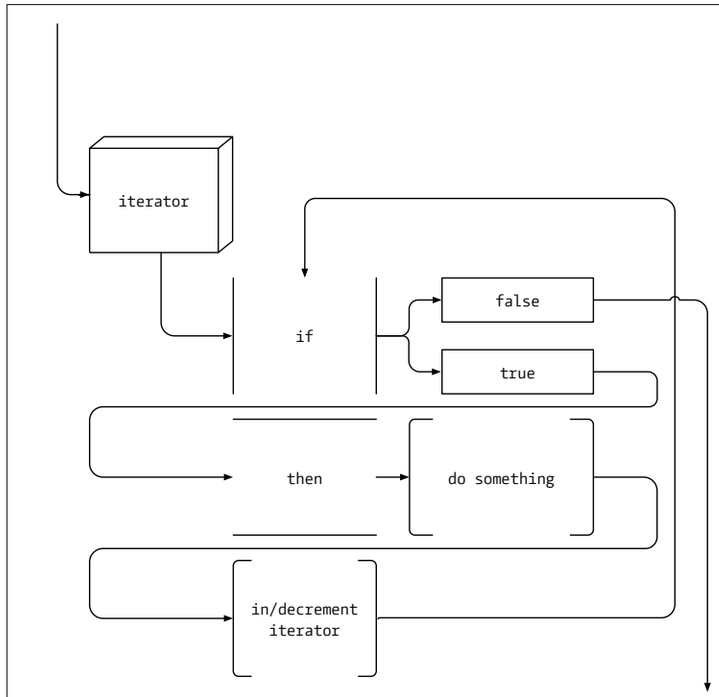
Hands on!

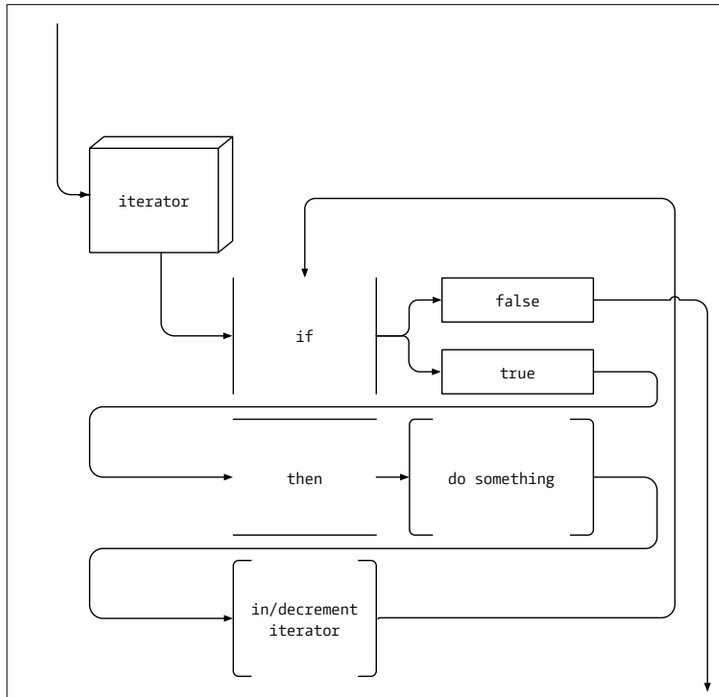


Es gibt 3 unterschiedliche Arten von loops. Jeder mit einer bestimmten Einsatzfeld.

Ranges

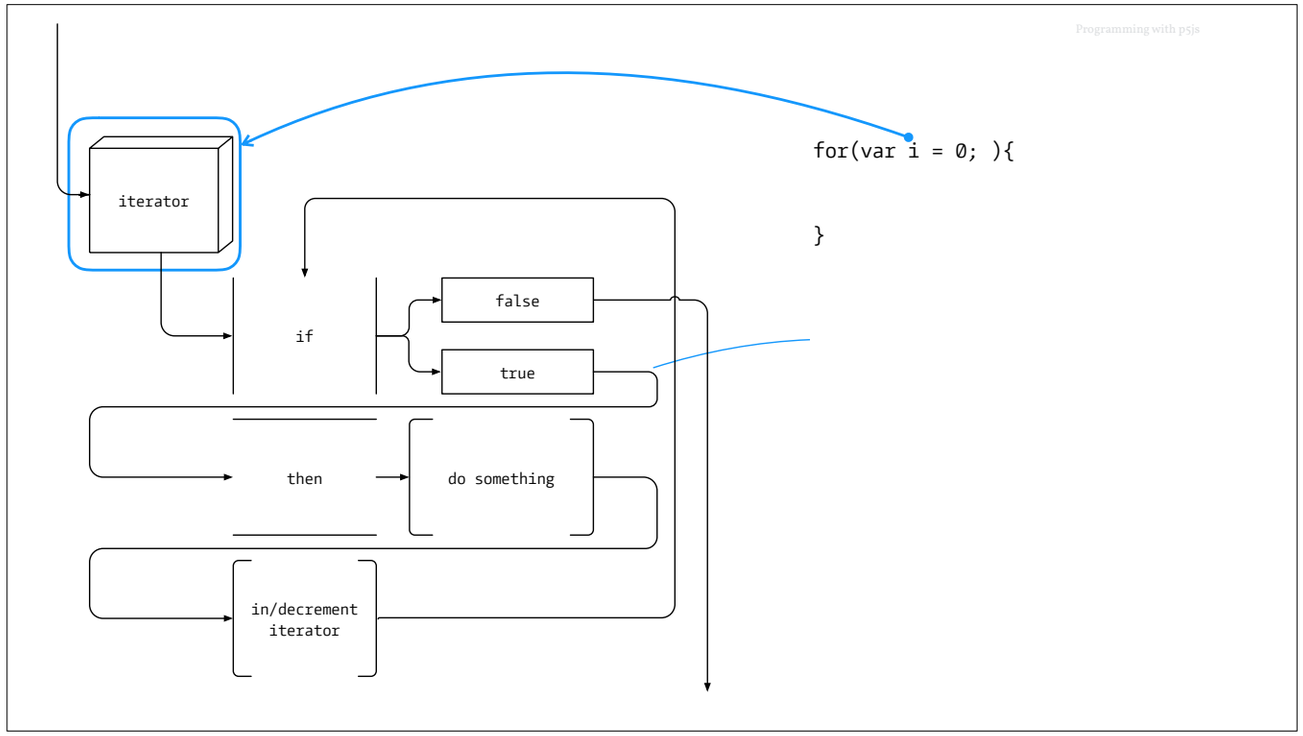
Erstmal ein loop für Wertebereiche. Der eignet sich besonders für Arrays


iterator
if
false
true
then
do something
in/decrement
iterator

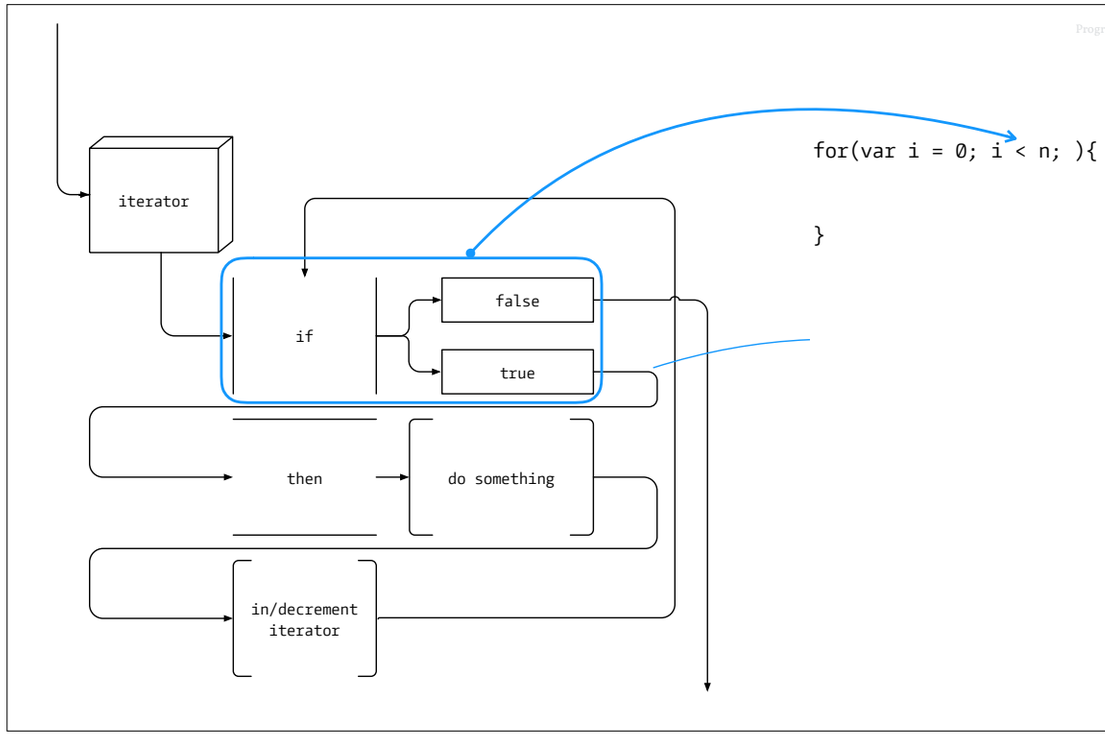



```
for(){

}
```

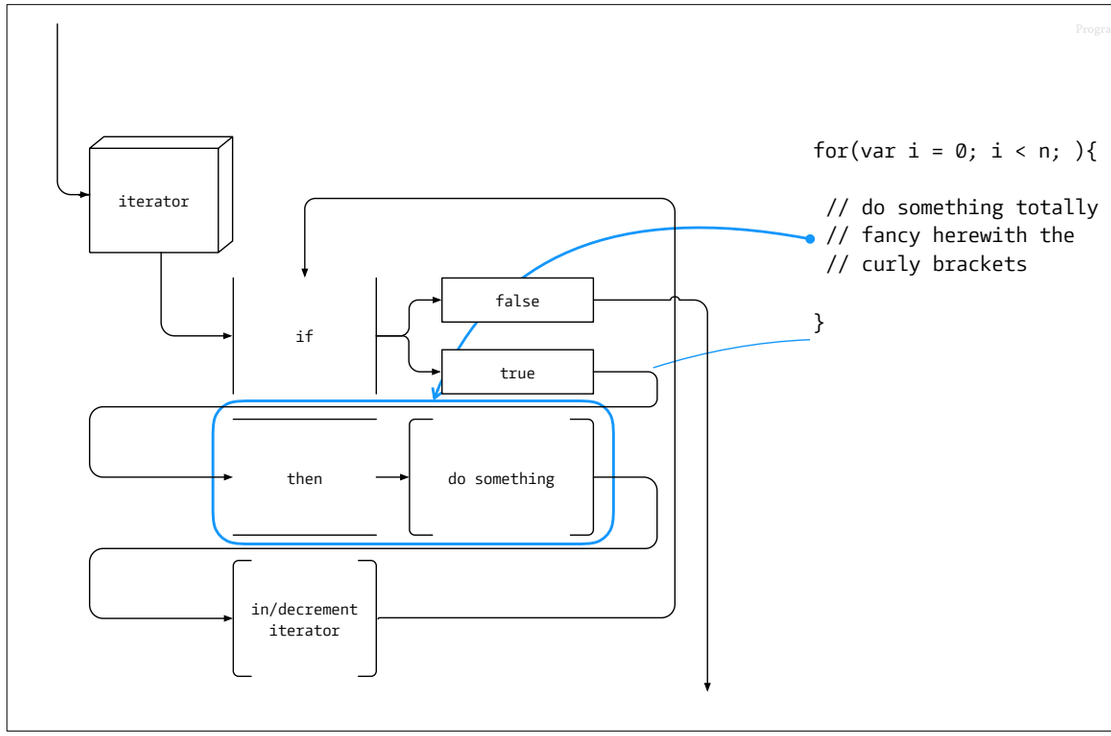
Das ist das Grundkonstrukt.



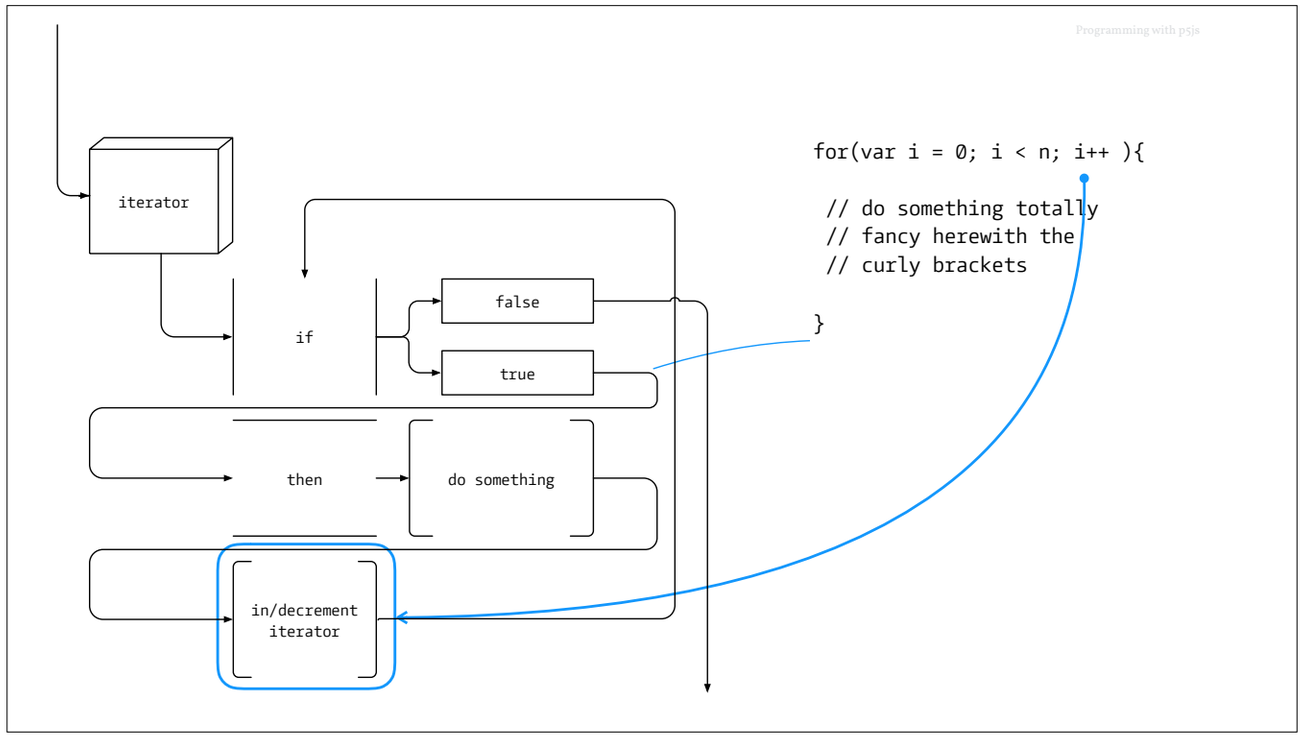
hinzukommt der Iterator. Eine lokale Variable die nur in diesem Loop existiert. Sie "i" zu nennen ist nur eine Konvention. Wir können sie auch Klaus oder Paul oder index nennen.



Als nächstes kommt eine Kondition. Diese bestimmt wie lange der Loop läuft. Die Kondition sollte sich immer auf den Iterator beziehen.



Also nächstes kommt der Block der ausgeführt wird solange unser Kondition zutrifft.



Also letztes inkrementieren oder dekrementieren wir den Iterator. Wobei die Schreibweise i++

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

i++ ist nur eine Schreibweise für i = i+1 oder i+=1

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

genauso i minus minus

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

die Veränderung des Iterators muss nicht um 1 sein

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

Somit können wir über einen Wertebereich loopen.

hier von 0 bis n - 1

0,5,10, 15, 20, 25, 30, 35, 40, 45

0 × 5 = 0 usw

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

wie ist hier der Wertebereich?

von 100 bis n runter in schritten von 5

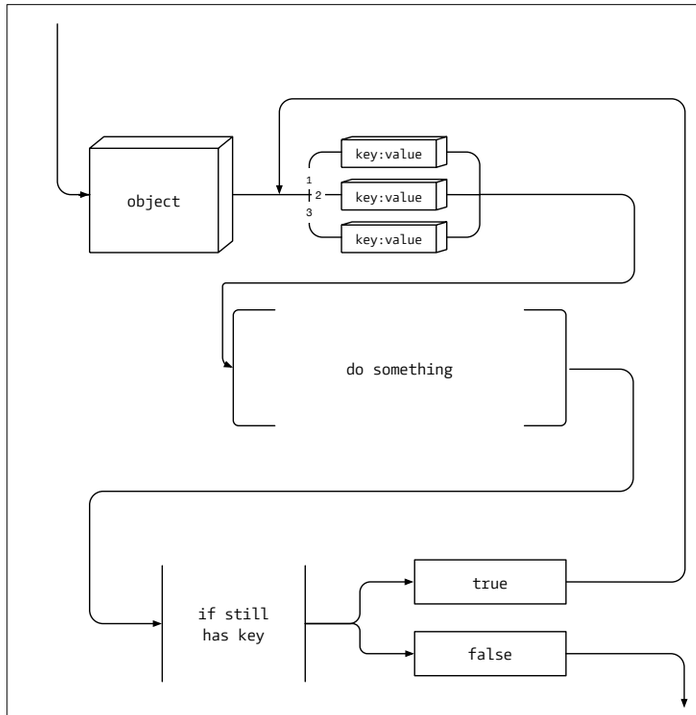
100, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10

```
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]);
}
```

Was ist hier das Ergebnis? a, c, e

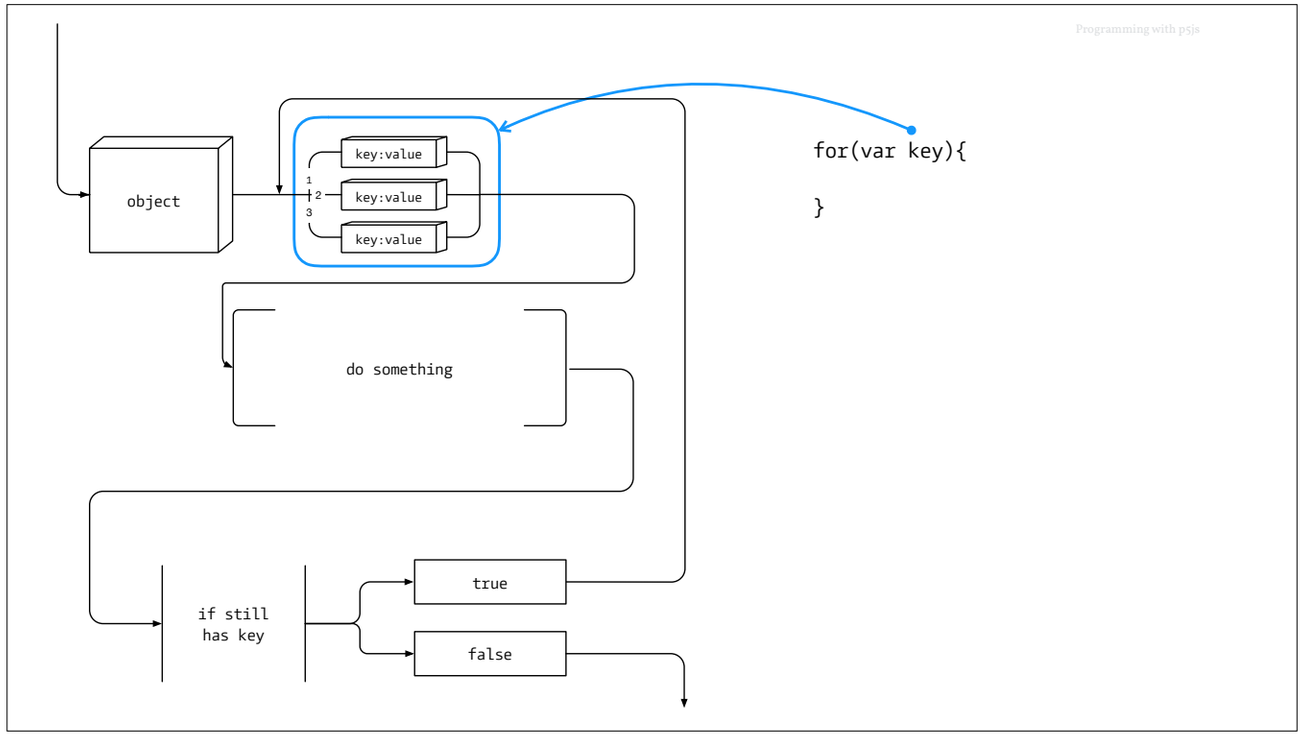
Was ist der Wertebereich? 0 bis 5

Object

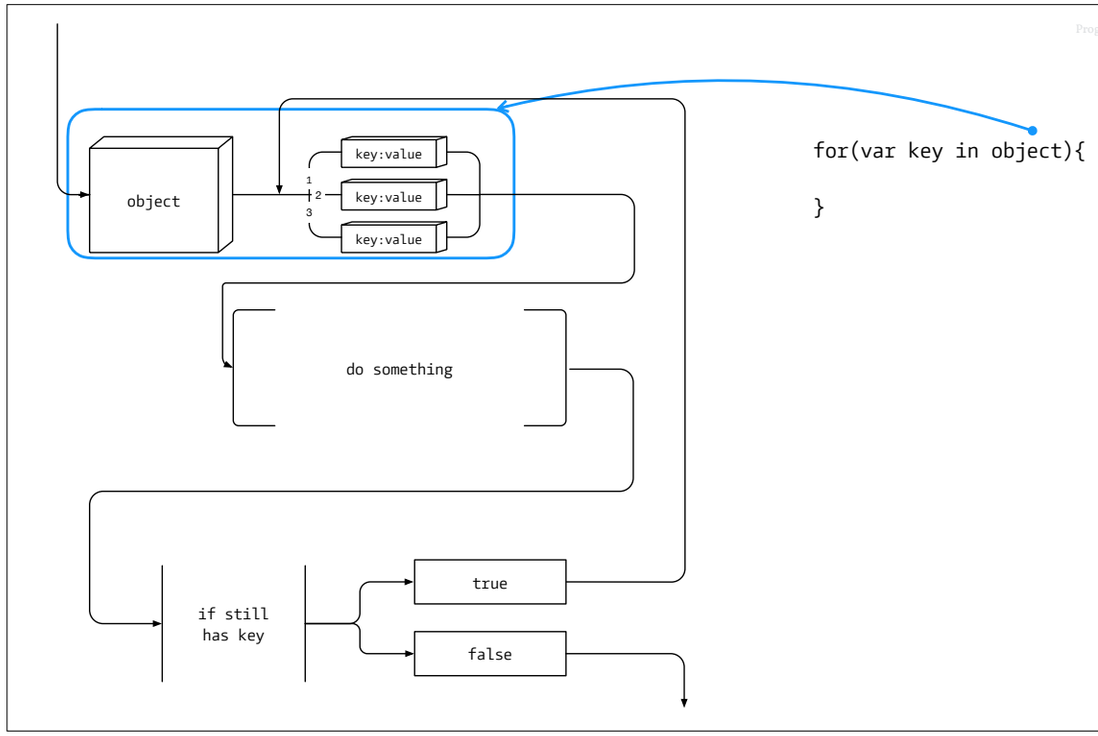



```
for(){

}
```

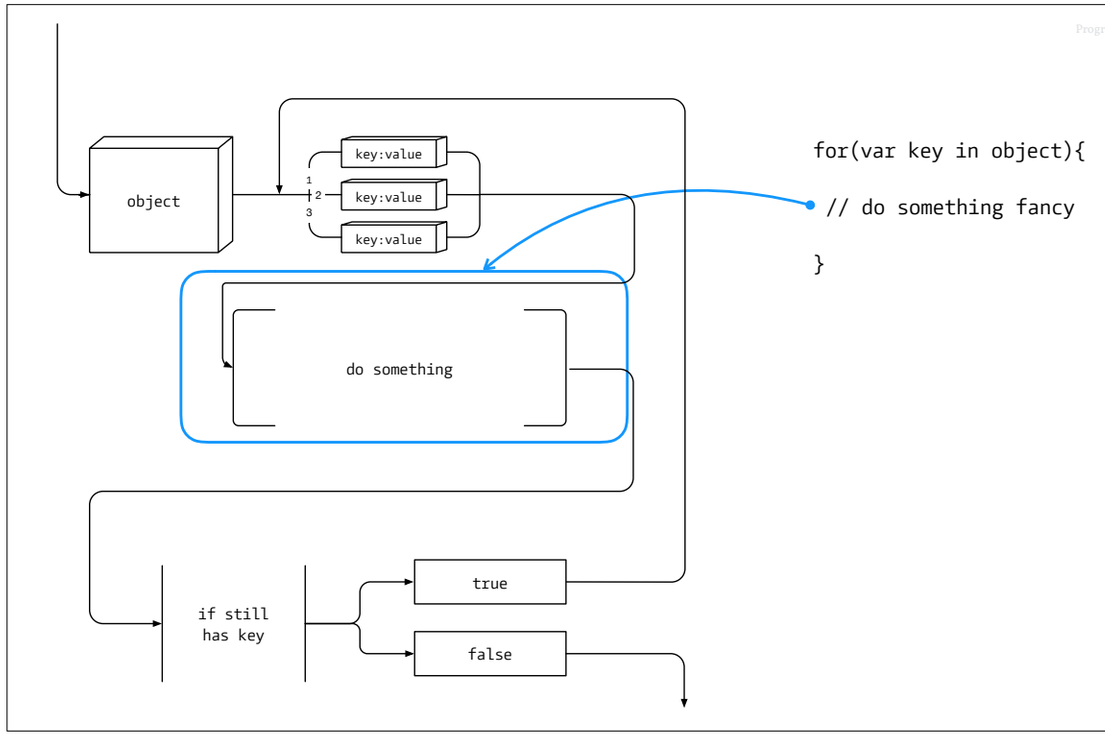
Auch hier ist das Grundkonstrukt das for



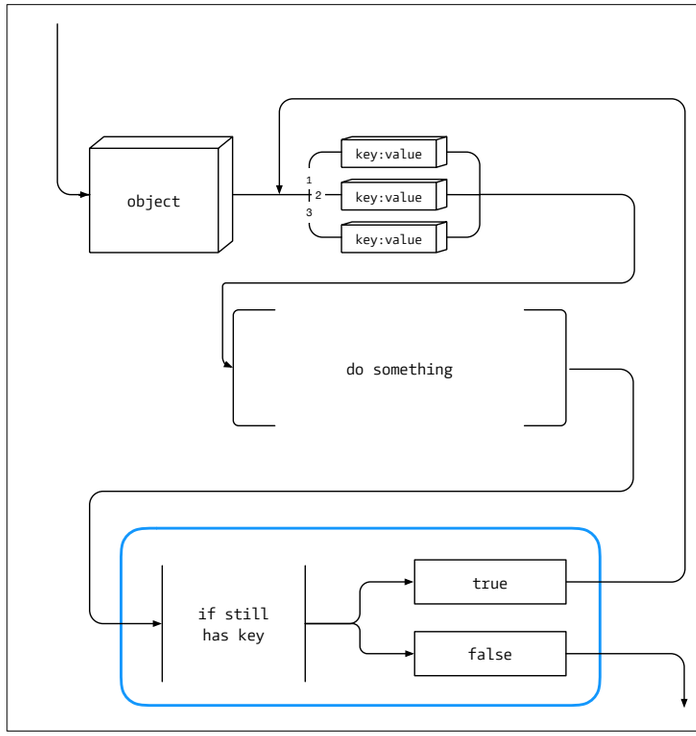
Dann kommt auch hier eine lokale Variable hinzu. Wobei key der Name ist und auch frei wählbar ist. wir greifen also unsre einzelne key:value Paare ab.



Das mache wir für alle Paare in unserem Objekt.



Also haben wir für jedes key:value Paar in unserem Objekt einen Block den wir ausführen.




```
for(var key in object){

 // do something fancy

}
```

Das ganze passiert solange wir key:value Paare in unserem Objekt haben.

```
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]);
}
```

Angewandt sieht das dann so aus. Was spuckt uns die Konsole aus?

The key is x, The value is 10, The key is y, The value is 20, The key is z, The value is -20

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

Was gibt die Konsole aus?

a, b

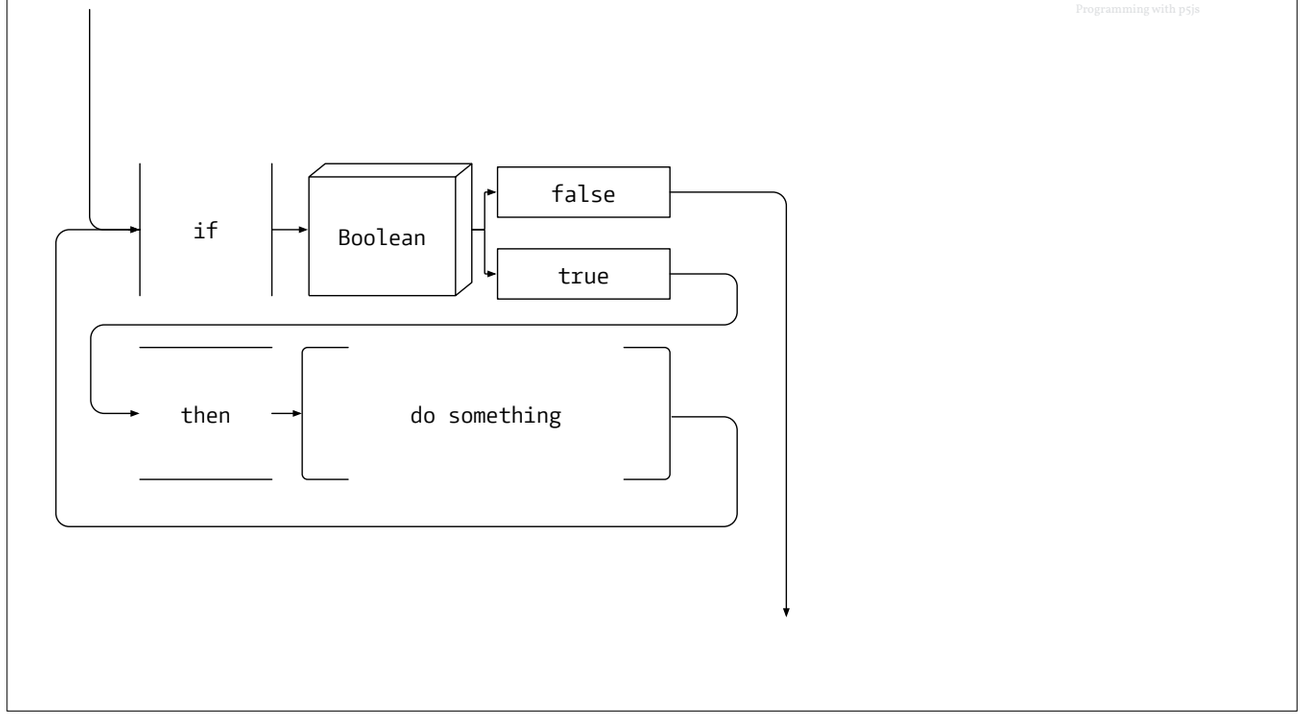
```
var obj = {"a":"Hello","b":"World"};
for(var key in obj){
 console.log(obj[key]);
}
```

Was gibt die Konsole aus?

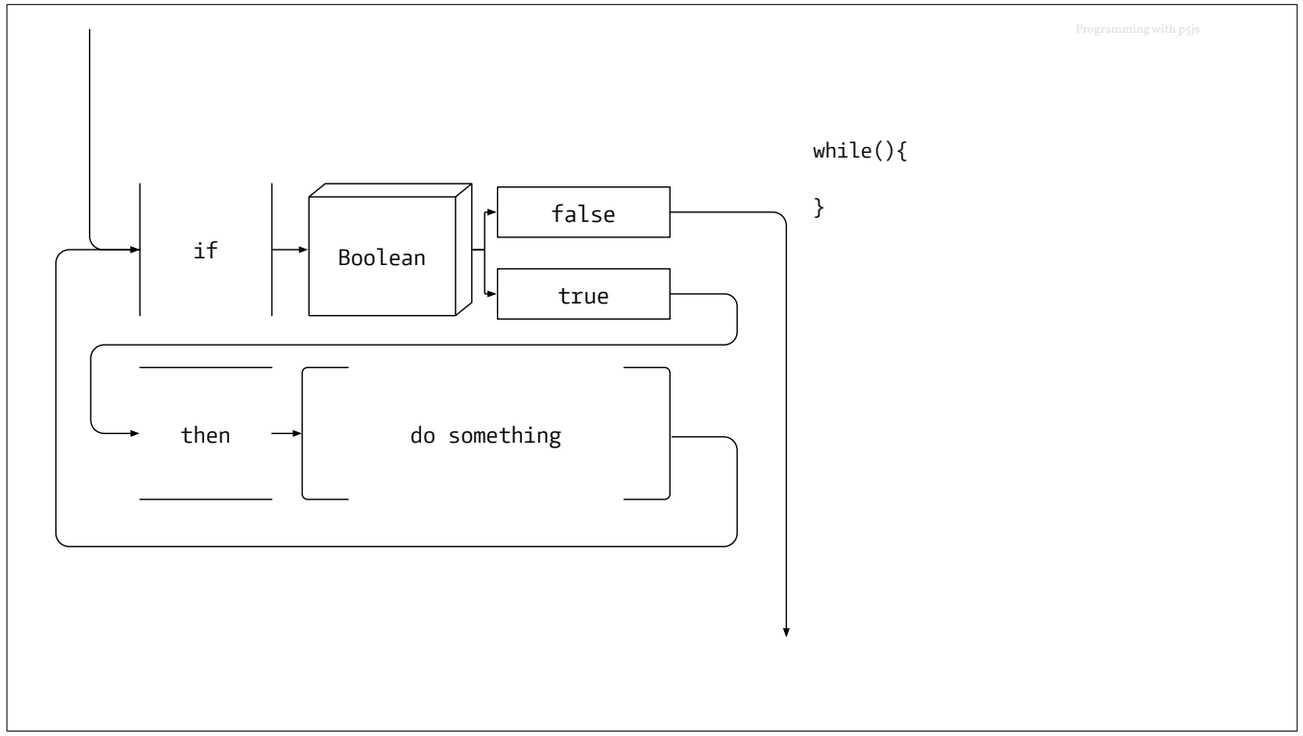
"Hello", "World"

Condition

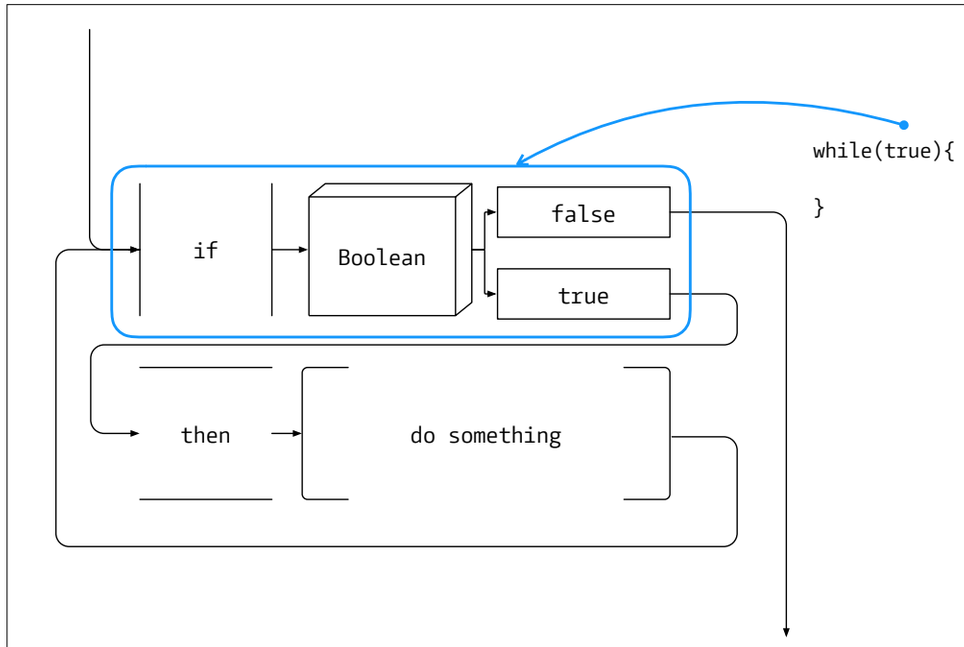
Als letzten Loop gibt es den While loop. Den ich als Konditional Loop bezeichnen würde.



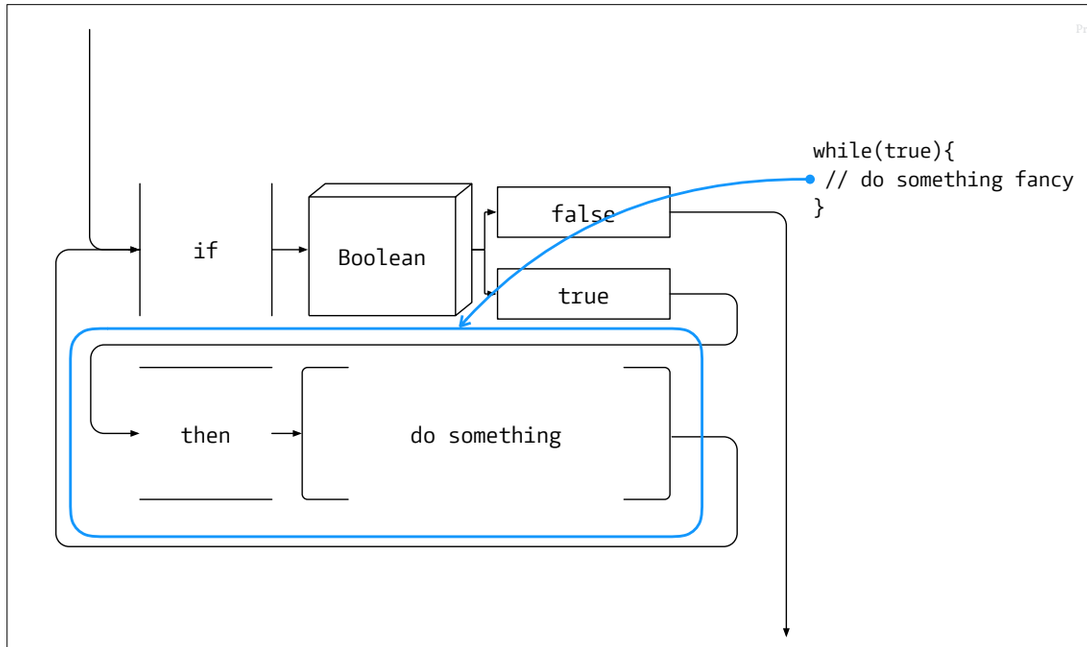
Logisch sieht das ungefähr so aus.



Das ist erstmal das Grundgerüst.



Wenn die Kondition true als Ergebnis zurück gibt.



Der Block wird solange ausgeführt bis das Ergebnis nicht mehr true ist.

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

Hie Fragen wir konditional. Wir erzeugen Zufallszahlen. Sobald eine Zufallszahl größer als 2.5 ist brechen wir ab

Hierbei ist nicht voraussehbar wieviele Wiederholungen wir bekommen.

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

Auch hier fragen wir eine Wertebeirech ab. wir loben von

0 bis 4

```
while(true){

}
```

ACHTUNG! das ist ein unendlicher Loop in den ich damit den Rechner schicke würde. Jeder while loop bracht ein Punkt an dem er verlassen wird.

break && continue

Es gibt für loops noch zwei extra Befehle break and continue;

break bricht eine loop ab. Continue springt zur nächsten Iteration

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

Was passiert hier?

```
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);
}
```

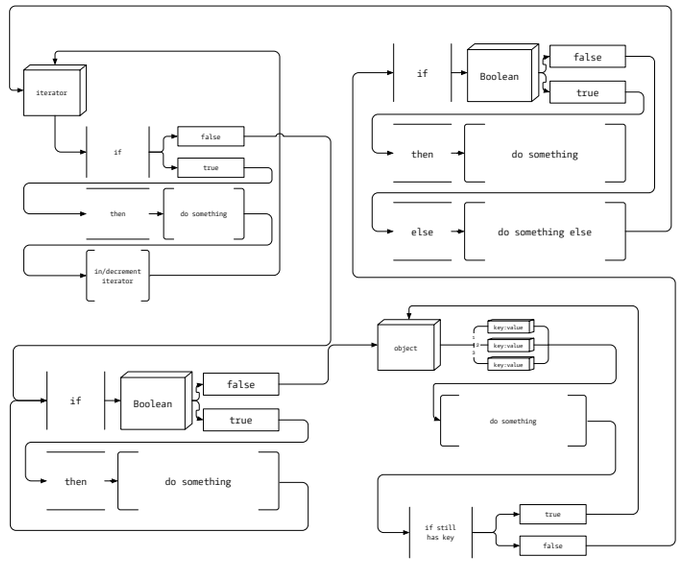
Was passiert hier? Was ist das Ergebnis 1,3,5

7 BASIC THINGS IN PROGRAMMING

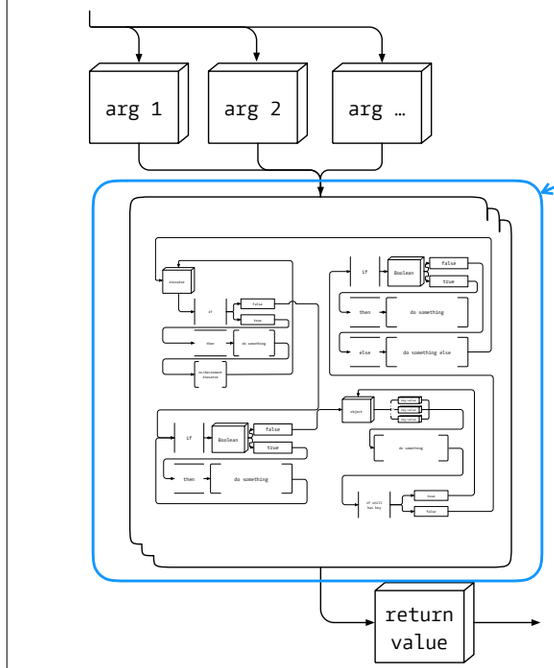
1. Variablen ✓
2. Objekte ✓
3. Arrays ✓
4. Konditionen ✓
5. Schleifen ✓
6. Funktionen
7. Algorithmus

? Fragen. Hands on!

function

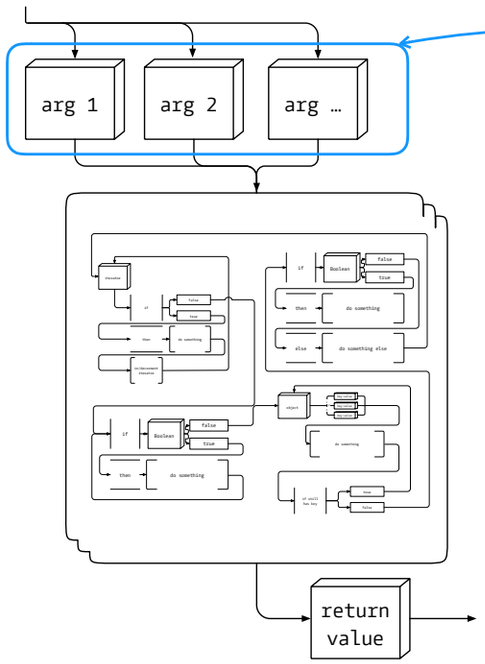


Wenn wir so einen Komplexen Programmteil haben. Wollen wir sowas natürlich nicht immer wieder neu schreiben müssen wenn wir es mehrfach benutzen wollen.



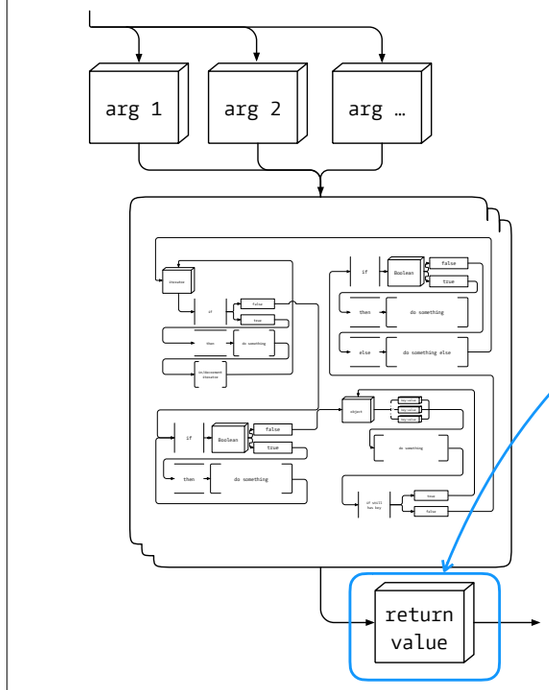

```
function name(){
 // lots of complex code
  var c = 5;
}

name();
```

Dafür verpacken wir unseren ganzen komplexen Code in eine function. Der Name ist frei gewählt!




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

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

Wir können an unser function Werte übergeben. Das nennt man Argumente




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

Und eine Funktion kann auch werte zurück geben. Den return value

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

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

Ergebnis 50, 6, 56088

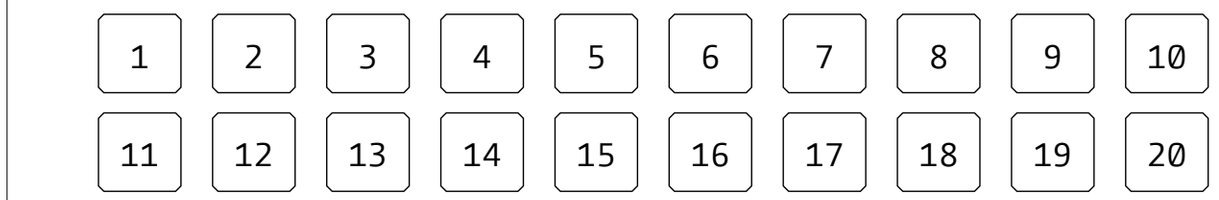
7 BASIC THINGS IN PROGRAMMING

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

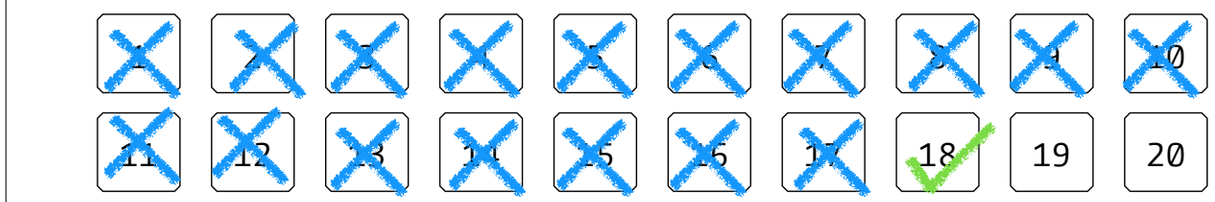
? Zu guter letzt der Algorithmus no hands on

BINÄRE SUCHE

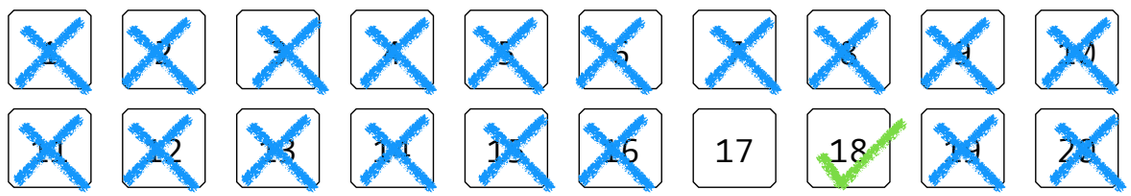
zB eine Binäre suche



Ich denke mir eine Zahl zwischen 1 und 20. Ich kann auch wenn ihr mich nach einer Zahl fragt nur sagen ob es stimmt ob sie grösser oder kleiner ist. Wie würdet ihr suchen?



Brute Force?



Binary Search Algorithm

Sagen wir mein Zahl ist wieder 18.

1. Dann frage ich ist es 10? Nein es ist größer
2. ist es 16? Nein es ist größer
3. ist es 19? Nein es ist kleiner
4. ist es 18? Ja woohoo

```
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);
```

In code sieht der dann so aus.

LICHT ALGORITHMUS

Oder der 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
```

Der Fabian-Licht-Algorithmus in Pseudocode

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

Der Fabian-Licht-Algorithmus in Pseudocode - abstrakter

DRY-CODE

(Don’t Repeat Yourself)

Aber wir sind faul und wollen nicht zufiel schreiben. Deshalb DONT REPEAT YOURSELF

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

Der Fabian-Licht-Algorithmus in Pseudocode - abstrakter -dry (don’t repeat yourself) zum Schalter muss ich so oder so gehen. das kann man auch nur einmal aufrufen.

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

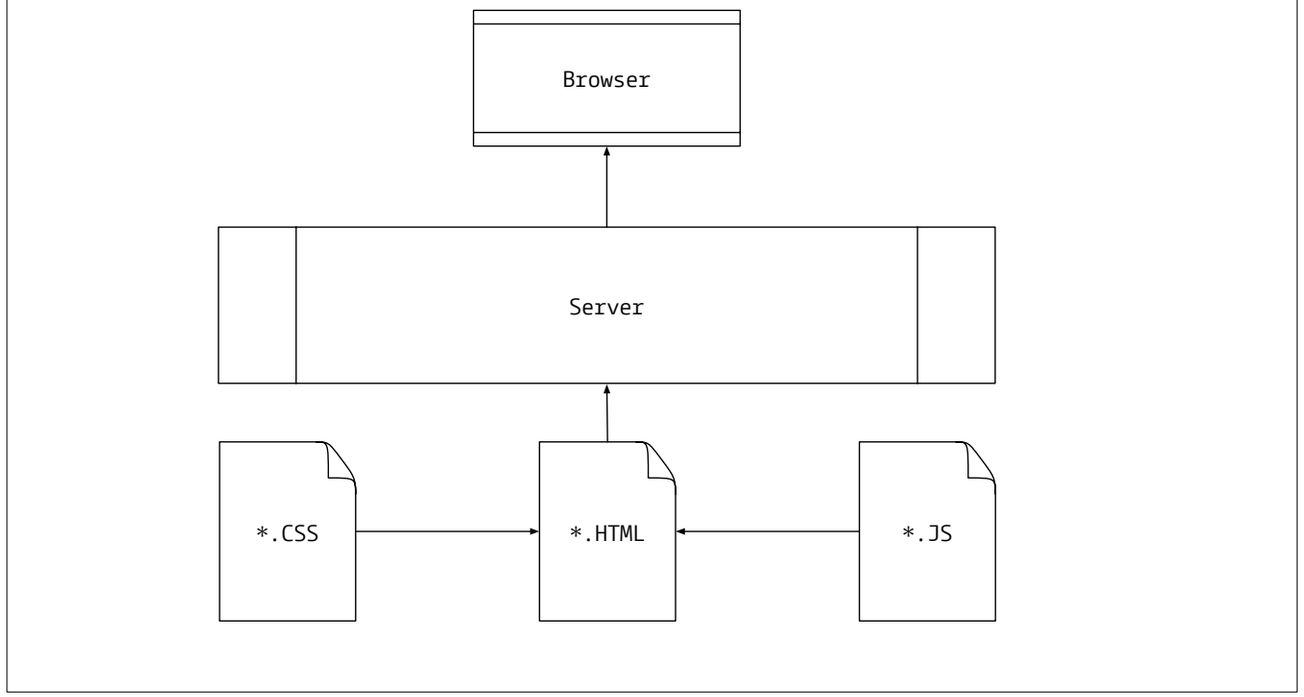
Der Fabian-Licht-Algorithmus in Pseudocode - noch abstrakter - dry (don’t repeat yourself)

ich setzte den Schalter auf das Gegenteil des aktuellen Zustands.

0. SETUP

HTML + CSS + JS + Server

Wie spielen html css und js zusammen?



Unser setup besteht meist aus folgenden 5 Bestandteilen.

1. CSS dort findet ihr alle style Beschreibungen. Wie gross ist ein spalte usw.
2. JS da schreiben wir die Funktion rein in unserem Fall was visuell passieren soll
3. HTML dort steht die Sruktur. Das Layout. sozusagen.
4. Der Server. Damit niemand mit JS Unfug auf eurem Computer betriebt sollte ein Server die Dateien bereitstellen.
5. Der Browser der den Server aufruft und das HTML+CSS+JS darstellt.

Ab jetzt Terminal + server Dateien erstellen und einbinden.

```
npm install reload -g
```

installiert einen lokalen server

```
sudo npm install reload -g
```

installiert als super user

1. SHAPES

point, line, ellipse, rect, vertex

Shapes Zeichnen.

Aufgabe: Zeichne einen Menschen mit primitiven Formen

AUFGABE

Zeichne einen Menschen mit primitiven Formen!

Shapes Zeichnen.

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

Shapes Zeichnen.

Aufgabe: Erzeuge eine spannende Farbkomposition/Farbreihe!

AUFGABE

Erzeuge eine spannende Farbkomposition/Farbreihe!

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

Shapes Zeichnen.

Aufgabe: Erzeuge eine spannende Farbkomposition/Farbreihe!

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

Interaction action loop.

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

AUFGABE

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

Shapes Zeichnen.

Aufgabe: Erzeuge eine spannende Farbkomposition/Farbreihe!

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

Interaction action loop.

Aufgabe: Schreibe ein Programm das basierend auf einem Maus oder Tastendruck Parameter wie Farbe oder Forma ändert.

AUFGABE

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

Interaction action loop.

Aufgabe: Schreibe ein Programm das basierend auf einem Maus oder Tastendruck Parameter wie Farbe oder Forma 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

Loops the power of programming.

Aufgabe: Fülle die Zeichenfläche mit primitiven Formen indem du einen Loop verwendest.

AUFGABE

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

Loops the power of programming.

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.