Functions

Hide implementation details, create generic building parts

Agenda

- Function
- Modularization
- Questions

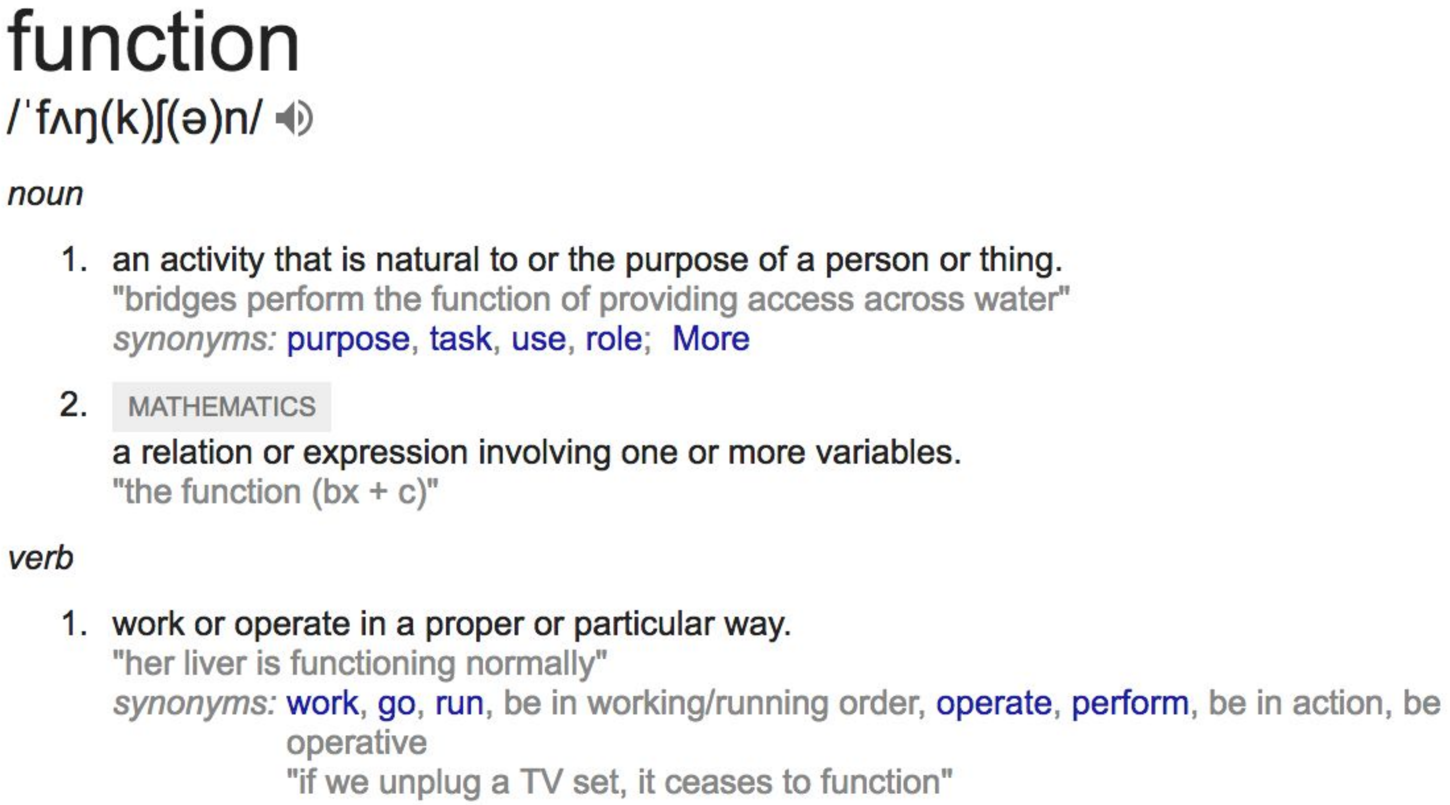
Learn Objectives

- Understand the reusability in context of the functions.
- You understand what modularization is.
- You understand what a function is.
- You know how to make generic processing parts
- You know how to promote a logic

Function

Abstracting a code block

- Do one operation/calculation, do only one job!
- A reusable part in an entire logic.
- Examples
  - turnLeft()
  - drinkWater()
  - fly()
  - cut()
- A group of recurring code lines.
- Naming convention (get, set, add, remove, ..)
- parameter, argument, return values, arguments parameter
- definition, call



function

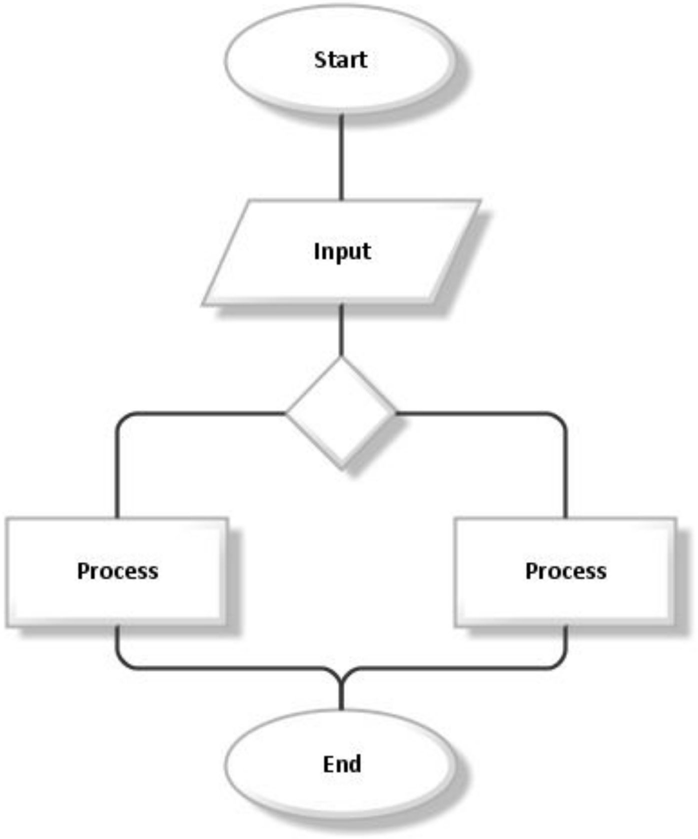
/ˈfʌŋ(k)ʃ(ə)n/

noun

1. an activity that is natural to or the purpose of a person or thing.
   "bridges perform the function of providing access across water"
   synonyms: purpose, task, use, role; More
2. MATHEMATICS
   a relation or expression involving one or more variables.
   "the function (bx + c)"

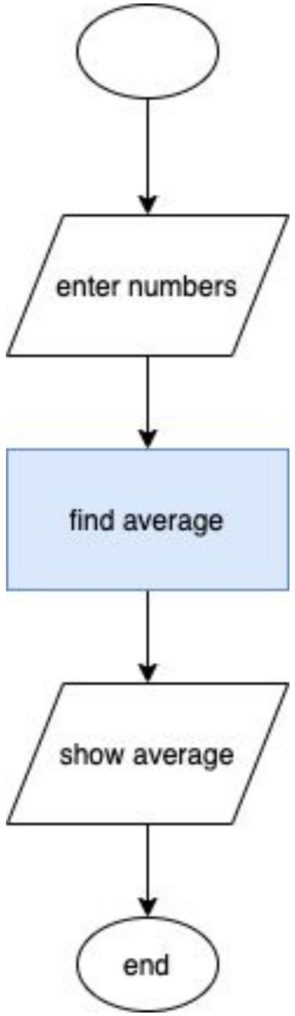
verb

1. work or operate in a proper or particular way.
   "her liver is functioning normally"
   synonyms: work, go, run, be in working/running order, operate, perform, be in action, be operative
   "if we unplug a TV set, it ceases to function"

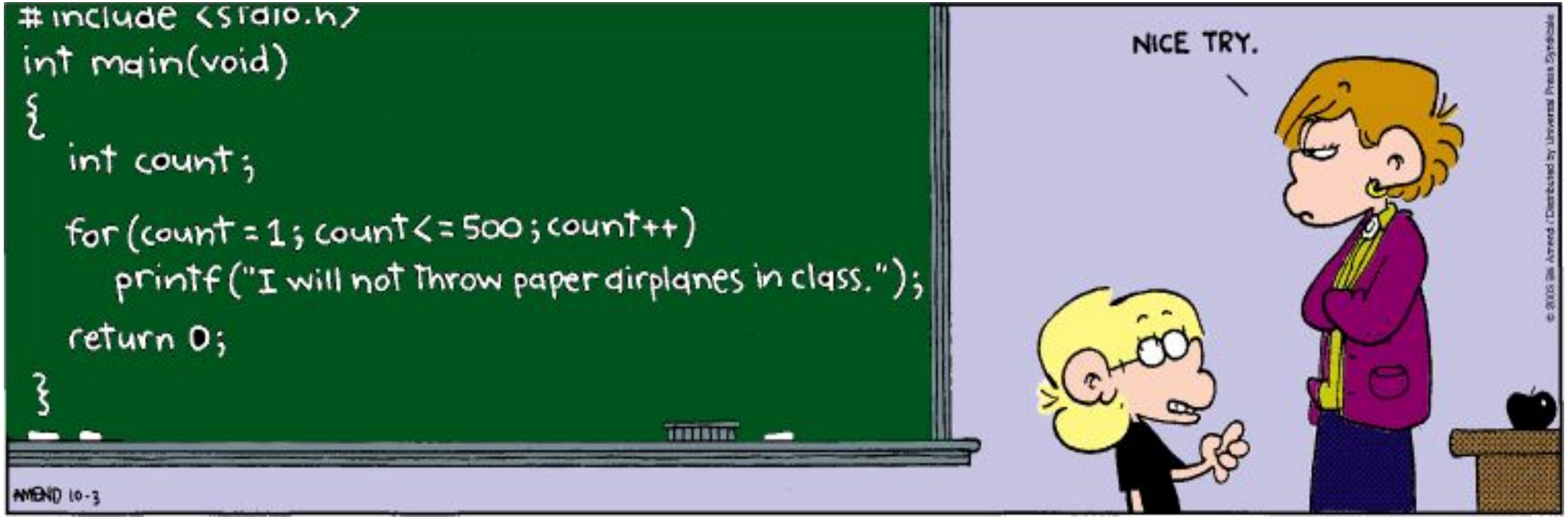


Is there any function here?

source: http://www.ariscommunity.com/flowchart



A function in a flowchart

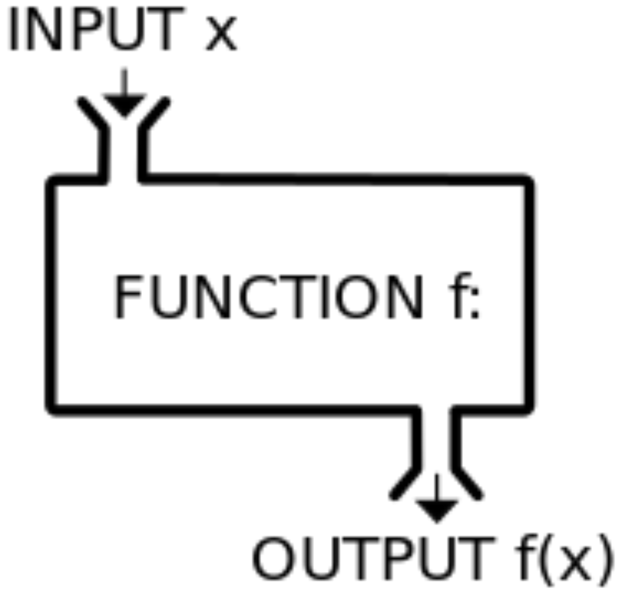


```
#include <stdio.h>
int main(void)
{
  int count;

  for(count = 1; count <= 500; count++)
     printf("I will not Throw paper airplanes in class.");

  return 0;
}
```

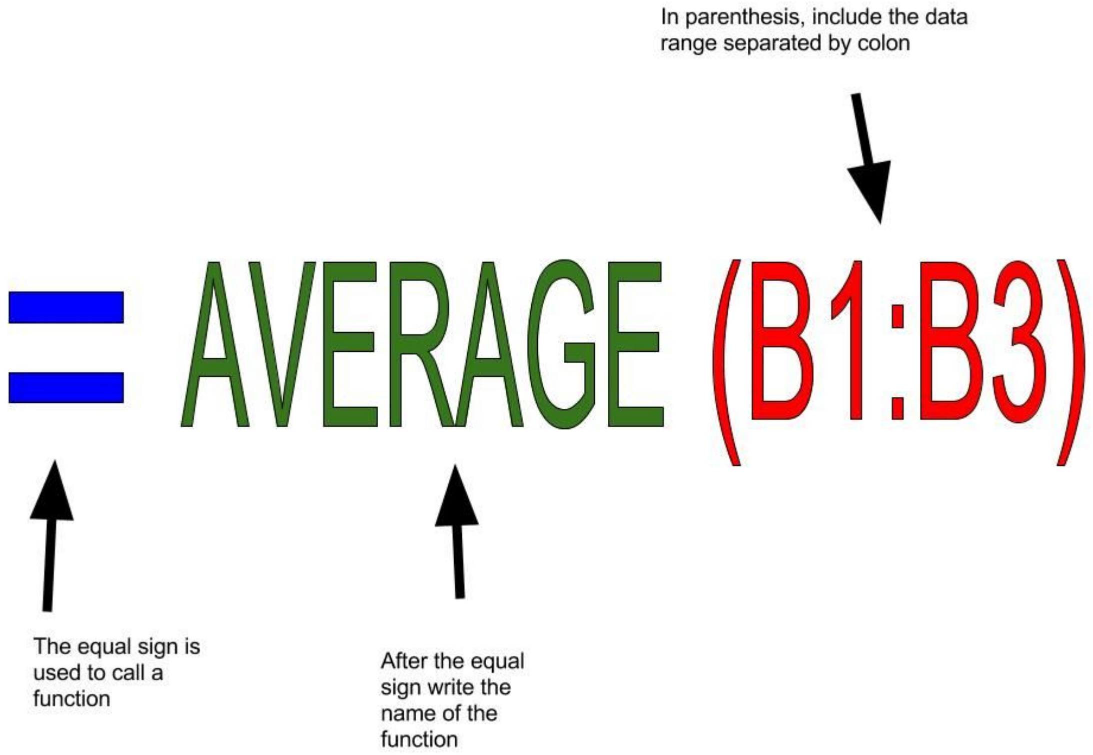
Code reusability in a nutshell!

Reasons

- Reusability
  - Code should be created/formed reusability in mind.
  - How can I reuse the code?
- Modularity
  - Building blocks.
  - Some code are generic enough to be used in another code
- Readability
- Information Hiding/Encapsulation
  - Nobody should know about what you code does

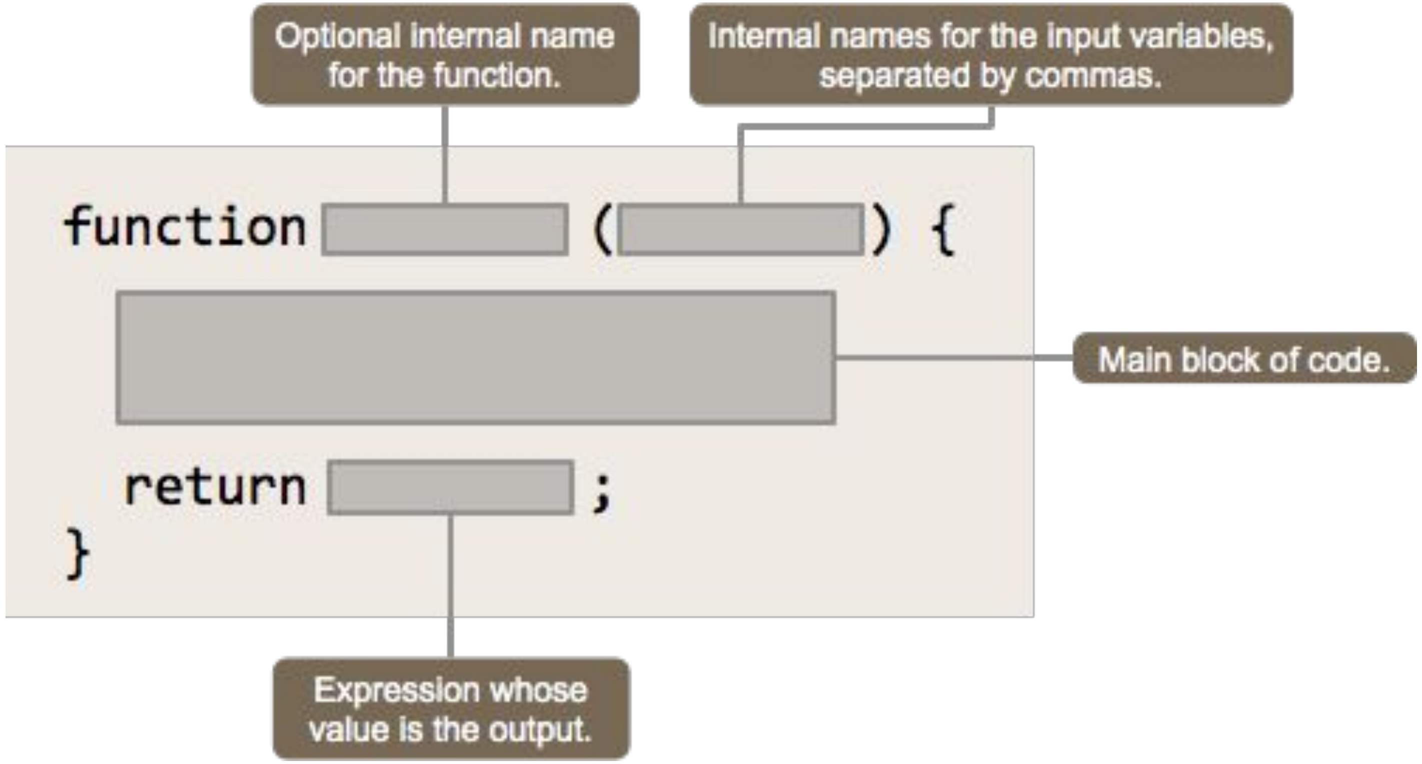


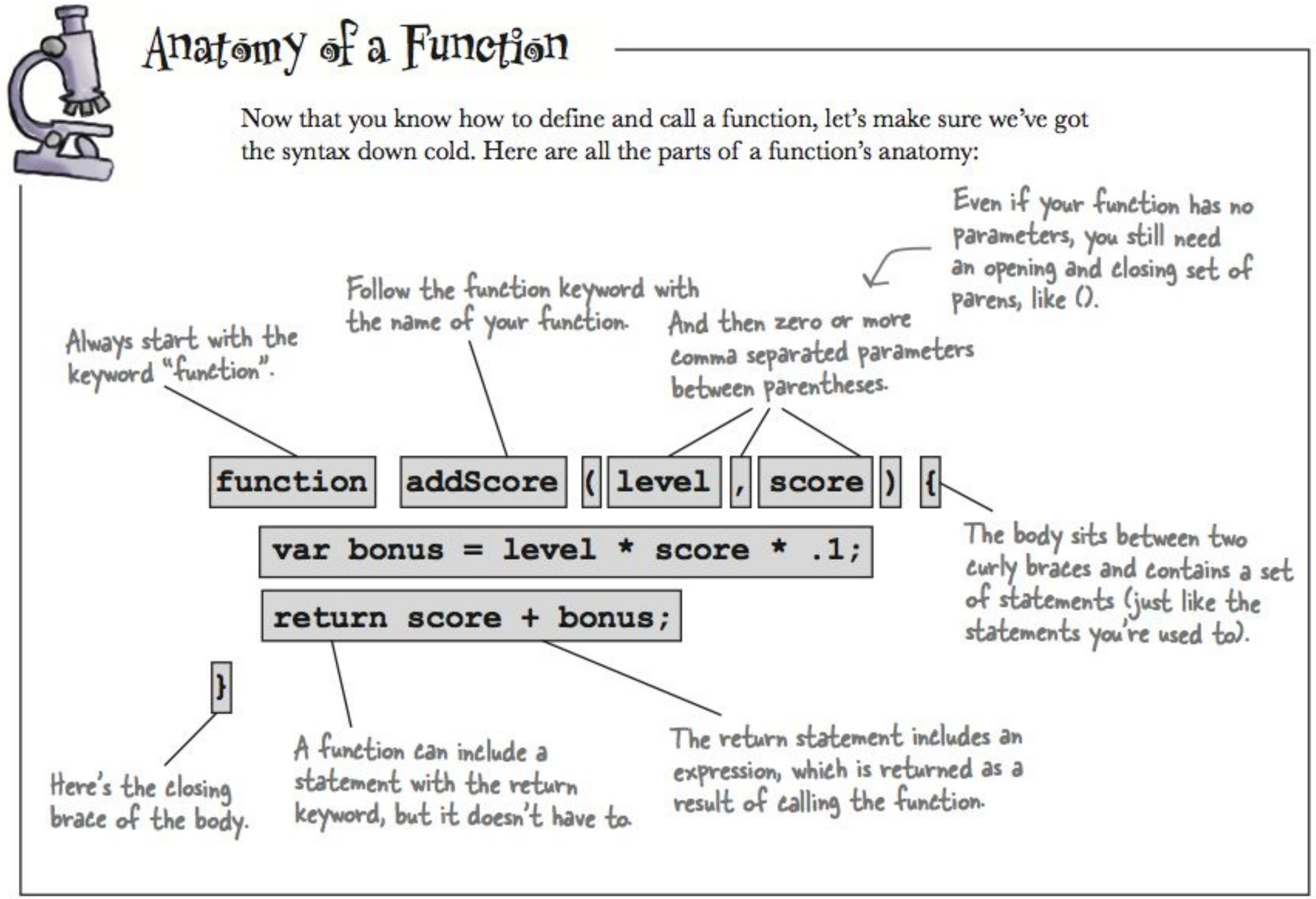
Function gets an input (not always) and returns an output (but not always).


In parenthesis, include the data range separated by colon
= AVERAGE (B1:B3)
The equal sign is used to call a function
After the equal sign write the name of the function

In the excel, you are using many functions

Function Syntax


Optional internal name for the function.
Internal names for the input variables, separated by commas.
function ( ) {
Main block of code.
return ;
}
Expression whose value is the output.



Anatomy of a Function

Now that you know how to define and call a function, let's make sure we've got the syntax down cold. Here are all the parts of a function's anatomy:

```
function addScore ( level , score ) {
    var bonus = level * score * .1;
    return score + bonus;
}
```

Always start with the keyword "function".

Follow the function keyword with the name of your function.

And then zero or more comma separated parameters between parentheses.

Even if your function has no parameters, you still need an opening and closing set of parens, like ().

The body sits between two curly braces and contains a set of statements (just like the statements you're used to).

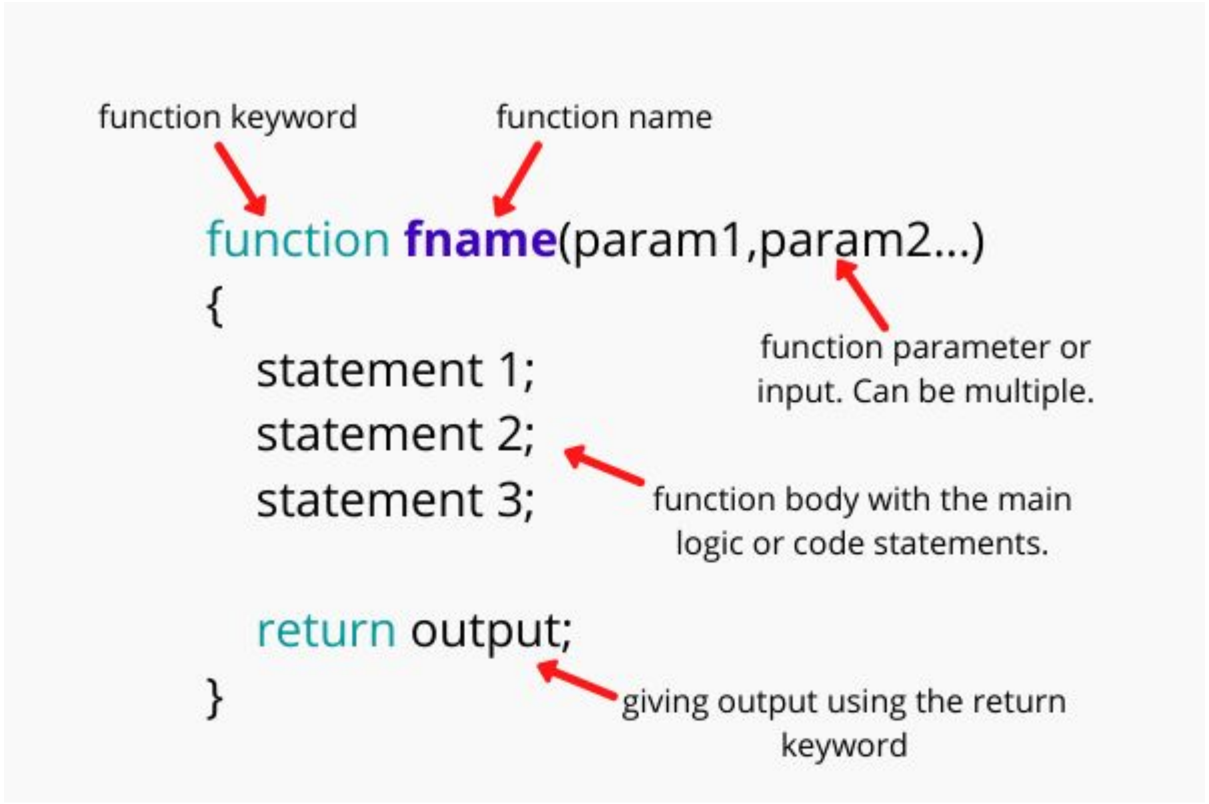
Here's the closing brace of the body.

A function can include a statement with the return keyword, but it doesn't have to.

The return statement includes an expression, which is returned as a result of calling the function.

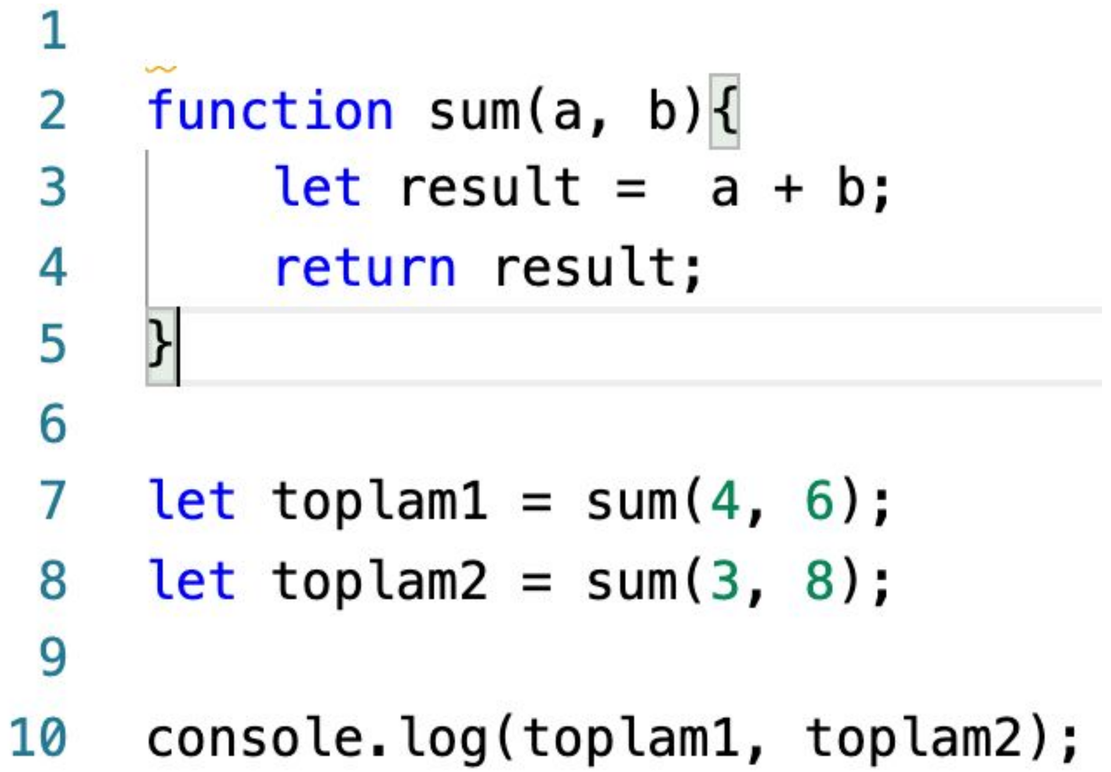
Anatomy of a function

source: https://cs.wellesley.edu/~cs110/lectures/L16/


```
function fname(param1,param2...)
{
   statement 1;
   statement 2;
   statement 3;

   return output;
}
```
function keyword
function name
function parameter or input. Can be multiple.
function body with the main logic or code statements.
giving output using the return keyword

```
function makeSandwich(🍖,🥬,🍞){
              🥪
     let sandwich = 🍖+🥬 + 🍞;
     return 🥪 ;
}
```

Get some parameters, create a result and return it.



```

function sum(a, b){
    let result =  a + b;
    return result;
}

let toplam1 = sum(4, 6);
let toplam2 = sum(3, 8);

console.log(toplam1, toplam2);
```

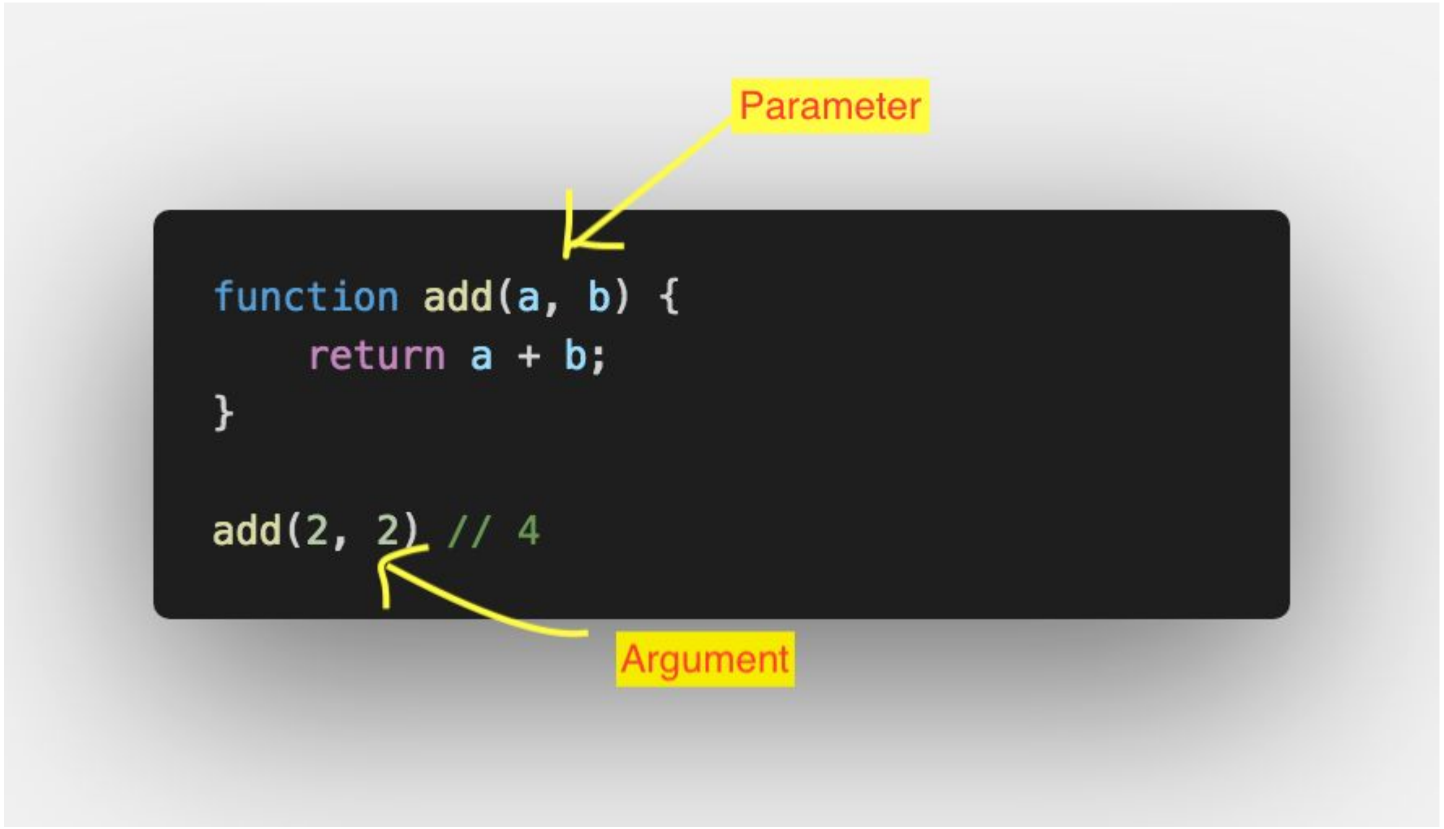
Function Definition vs Function Call (invoke)

let's try it!

Create a function, which takes two parameters and finds/returns an area of a rectangle.

function parameters & arguments

- Functions are/should be generic
- Functions can be given different values as input
- Function calls can be done by different arguments
- The parameters are the variables of the function call.
- Arguments are passed into functions as parameters



```
function add(a, b) {
    return a + b;
}

add(2, 2) // 4
```

Types of functions

- Named functions without parameters
- Named functions with one parameter
- Named functions with multiple parameters
- Functions Expressions
- IIFE
- Arrow Functions
- Function Constructor
- Anonymous Functions
- Callback functions

```
function showMessage() {
   alert( 'Hello everyone!' );
}
```

No parameter function

```
function checkAge(age) {
   if (age > 18) {
     return true;
   }
   return confirm('Did parents allow you?');
}
```

One parameter function

```
function sendMessage(from, to, cc, bcc, message) {
   const result = mailService(from, to, cc, bcc, message);
   if(result==true){
      console.log(“Your message has been sent”);
   }
   return result;
}
```

Function with multiple parameters

```
let isTruthy = function(value) {
   return !!value;
};

isThuthy(“1”);
```

Function expressions

```
const result = (function(pIsim) {
   const name = pIsim.toLowerCase();
   return name;
})(“HiCoders”);

// “hicoders”
```

IIFE (immediately invoked function expressions)

```
const absValue = (number) => {
    if (number < 0) {
        return -number;
    }
    return number;
}


absValue(-10); // => 10
absValue(5);  // => 5
```

```
let double = function(n) {
        return n * 2
}
let double = n => n * 2;

alert( double(3) ); // 6
```

Arrow Functions

```
var sum = new Function('a', 'b',
'return a + b');

console.log(sum(2, 6));
```

Function Constructor

```
setInterval(function() {
    console.log(“her bir saniyede göster bu testi”)
}, 1000);
```

Functions that get a function as parameter

```
function successCallback() {
// Do stuff if success message received
}
function completeCallback() {
// Do stuff upon completion
}
function errorCallback() {
// Do stuff if error received
}

function sendToServer(success, complete, error) {
// Do stuff if error received
     success();
     complete();
     error();
}

sendToServer(successCallback, completeCallback, errorCallback);
```

Functions that get multiple functions as parameters

```
function getFruit(type) {
    if(type===”apple”){
       return function(sayi) {
                return sayi + “ elma”;
             }
    }else if(type===”pear”){
       return function(sayi) {
                return sayi + “ armut”;
             }
    }

    return function() {
             return “zikkimin kökü”;
           }
}


getFruit(“apple”)(2); // 2 elma
getFruit(“pear”)(5); // 5 armut
getFruit()();  // zikkimin kökü
```

Functions that return a function as return value

```
function iAmCaller(callMeBack) {
    let x = 2;
    callMeBack(x);
}


iAmCaller(function(param){
    console.log("iki kati: ", param * 2);
});
```

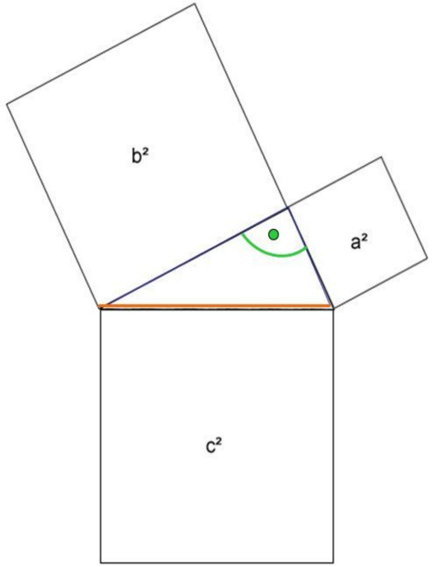
Callback/Anonymous Functions

let's try it! (10 min)

Write a function that tells whether a number is even or odd.

let's try it! (10 min)

Write a function, that finds the hypotenuse in a triangle.



Im Rechtwinkligen Dreieck gilt:

Die Summe der Flächen der Kathetenquadrate ist genau so groß wie die Fläche des Quadrates über der Hypotenuse.

Oder hier:

$$a^2 + b^2 = c^2$$

Questions?