JSFUNCTIONS

WHAT

Javascript functions **are block of code** (so, a set of statements) who can be **used and reused many times**, each time passing different arguments to reach different results. Like every code blocks, functions' statements are **enclosed in curly brackets { }**.

A FUNCTION IS...

EXAMPLE

DEFINED → A Javascript function is defined with the **function** keyword, followed by **a name**, followed by **parentheses ( )** →

```
function sayHello() {
   // block of code / statements
}
```

INVOKED → A Javascript **function needs to be invoked**, in order to be executed. Defining it, put the function in a sort of limbo. A function can exit the limbo, only when **someone calls for her**! →

```
sayHello();
```

EXECUTED → When executed, a Javascript function **return** an output / value. The return statement, **will stop the function**. →

```
function sayHello() {
   return 'Hello World!';
}
```

SCOPE

Scope is **the visibility** of an entity. Entities are variables, and variables task is to store values. Scope can be **Global**, **Local**, or (with ES6+) **Block**. Functions have **local scope**. Every variable declared inside the function, lives only inside that function, and **cannot be accessed from outside**.

```
// here code can't use hello

function sayHello() {
   var hello = 'Ciao!'
   // here code can use hello
}
```

HOISTING

Hoisting is Javascript default behavior of **moving declarations to the top** of the current scope. A function (and a variable) can be **used before been declared**. For this reason, is common best practice to declare functions all together at the bottom of the script. Javascript will enter the script, look for every declared function, and 'parking' it to the top...waiting for the function to be invoked.

```
myfunction(2);

function myfunction(x) {
   return x + x;
}
```

annalisacecchini.com | annalisacecchini.dev@gmail.com | (+39) 349.251.23.35