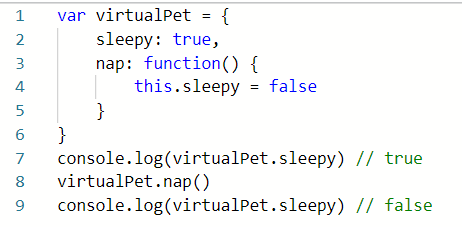
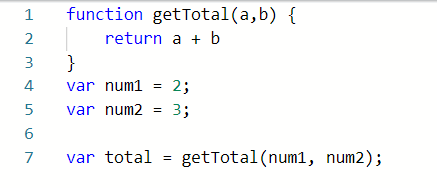
**JavaScript**

**Programming style :**

There are actually several styles of coding, also known as **paradigms**. A common style is called **functional programming**, or FP for short. In functional programming, we use a lot of functions and variables.

Another style is **object-oriented programming (OOP)**. In this style, we group data and functionality as properties and methods inside objects.



**Scope and hoisting:**

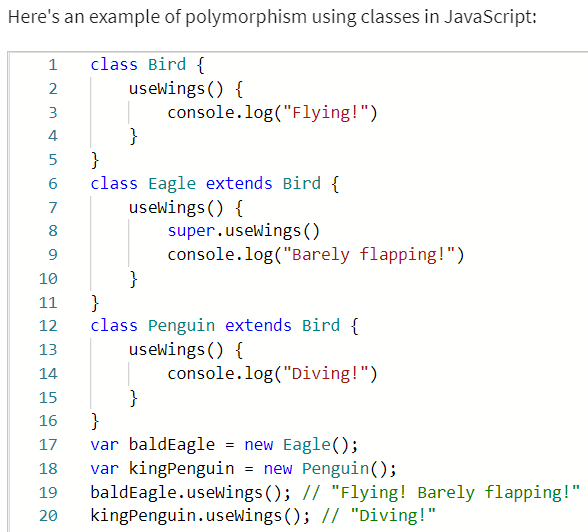
Scope determines the accessibility of variables, object and functions. A variable has 3 types of scope (block, function and global). Hoisting is a behavior in which a function or variable can be used before declaration (like **var**).

After ES6, we can use **let** and **const**, `**const**` is a signal that the identifier won't be reassigned. `**let**` is a signal that the variable may be reassigned.

**Benefits of OOP:**

Allows you to write modular code, makes your code more flexible and, makes your code reusable.

The four fundamental OOP principles are inheritance, encapsulation, abstraction and polymorphism:

****

* ENCAPSULATION has to do with making a code

implementation "hidden" from other users,

in the sense that they don't have to know how

my code works in order to "consume" the code.

* ABSTRACTION is all about writing code in a way

that will make it more generalized.

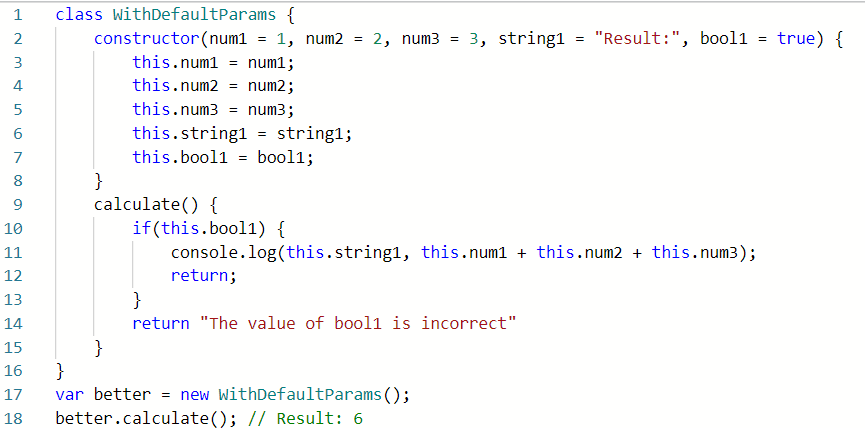
* POLYMORPHISM is a word derived from the

Greek language meaning "multiple forms".

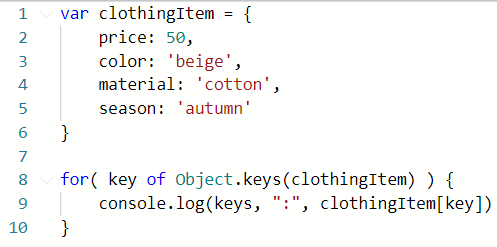
An alternative translation might be:

"something that can take on many shapes".

A useful a ES6 feature allows me to set a **default parameter** inside a function definition First



# For of loops and objects :

FOR of loop cannot work on an object directly, since **an object is not iterable**. Contrary to objects, arrays *are* iterable.

*You can extract object's keys, values,*

*or both, using the Object.keys(),*

*Object.values() and Object.entries()  :*

**For...in:**

The for...in**statement** iterates over all [enumerable properties](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Enumerability_and_ownership_of_properties) of an object that are keyed by strings (ignoring ones keyed by [Symbols](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol)), including inherited enumerable properties.

const object = { a: 1, b: 2, c: 3 };

for (const property in object) {

console.log(`${property}: ${object[property]}`);

}

// expected output:

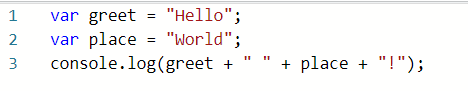
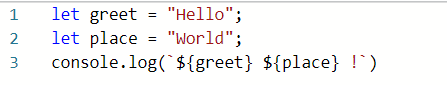
// "a: 1"

// "b: 2"

// "c: 3"

# Differences between a template and regular string :

Template literals are an alternative way of working with strings, which was introduced in the ES6 addition to the JavaScript language **(ex :** `Hello, World!`**).** Template literals allow for multi-line strings - something that simply isn't possible with string literals:



# Data Structures :

# Arrays in JavaScript come with a handy method that allows you to loop over each of their members (forEach). Another very useful method on the array is the filter() method. It filtersyour arrays based on a specific test. Finally, there's a very useful map method.

## 2. Working with Objects in JavaScript : 3. Working with Maps :

let bestBoxers = new Map();

bestBoxers.set(1, "The Champion");

bestBoxers.set(2, "The Runner-up");

bestBoxers.set(3, "The third place");

console.log(bestBoxers);

bestBoxers.get(1); // 'The Champion'

🡺 Map(3) {1 => 'The Champion', 2 => 'The Runner-up', 3 => 'The third place'}

const result = [];

const drone = {

    speed: 100,

    color: 'yellow'

}

const droneKeys = Object.keys(drone);

droneKeys.forEach( function(key) {

    result.push(key, drone[key])

})

console.log(result)

🡺 ['speed',100,'color','yellow']

4. Working with Set:

const repetitiveFruits = ['apple','pear','apple','pear','plum', 'apple'];

const uniqueFruits = new Set(repetitiveFruits);

console.log(uniqueFruits);

* {'apple', 'pear', 'plum'}

# Using Spread and Rest:

# Using the spread operator, it's easy to concatenate arrays, and also easy to join objects:

const fruits = ['apple', 'pear', 'plum']

const berries = ['blueberry', 'strawberry']

const fruitsAndBerries = [...fruits, ...berries] // concatenate

console.log(fruitsAndBerries); // outputs a single array

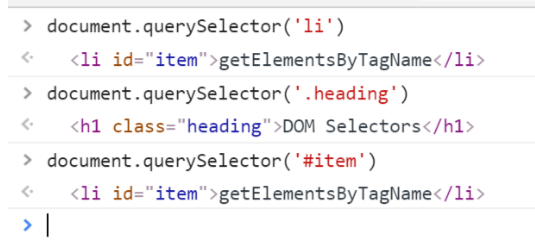
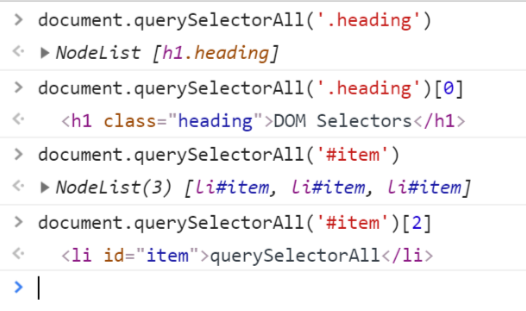
**Document Object Model:**

DOM stands for Document Object Model. It is a programming interface that allows us to create, change, or remove elements from the document. We can also add events to these elements to make our page more dynamic.

The DOM views an HTML document as a tree of nodes. A node represents an HTML element.

# DOM Selectors:

# DOM Selectors, as the name suggests is used to select HTML elements within a document using JavaScript. There are 5 ways in which you can select elements in a DOM using selectors.

* getElementsByTagName()
* getElementsByClassName()
* getElementById()
* querySelector()
* querySelectorAll()

**Unit testing:**

Unit testing revolves around the idea of having separate, small pieces of code that are easy to test.

function subtract(a, b) {

return a - b;

}

expect(subtract(10, 4)).toBe(6);

**End-to-end testing:**

End-to-end testing tries to imitate how a user might interact with your application. It can be performed in a web browser without writing code.

**Code Coverage:**

Code coverage shows what percentage of your code is covered by tests and where more testing may be required.

**Mocking:**

Functions of jest, is the recommended way to separate the code that you are testing from its related dependencies.