# ****How JavaScript works: functional style and how it compares to other approaches****

In reality js is not object oriented language. It is prototype-based language and not a class-based language.

When you create an object using the class syntax it automatically gets the prototype.

In code below calling "new" keyword before function will changes it's context and "victor" is a new object and it can call speak().

let Person = function(name, age) {

this.name = name;

this.age = age;

}

Person.prototype.speak = function() {

return `Hello, my name is ${this.name} and I am ${this.age}`

}

let victor = new Person(`Victor`, 23)

console.log(victor.speak());

Extending our object just as we did in the class syntax above where **Work extends class Person**, can be donе with Prototypes like this:

let Work = function(name, age, work) {

Person.call(this, name, age);

this.work = work;

}

Functions regarder as values and can be passed to variables.

const getSum = function(num) {

return num + num;}

getSum(9);

cosnt addNum = getSum;

## Declarative vs. Imperative JavaScript

The imperative approach is more like stating all the steps you would need to achieve a problem. While the declarative approach just declares or says what you want to be done.

### The imperative approach

const filterArray = (array) => {

let filteredArray = [];

for(let i = 0; i < array.length; i++) {

if(array[i] > 5) {

filteredArray.push(array[i]);

}

}

return filteredArray;

}

const array = [1, 2, 3, 4, 5, 6, 7, 8]

filterArray(array)

Rather than tell the computer what we want we just give instructions in steps on what we want to achieve. Our steps include:

* Declare an empty array
* Loop through a given array
* if/else if each item is greater than 5
* Push each element that passes the test into the empty array declared earlier
* Display our new array

### ****The declarative approach in JavaScript:****

// Filter method to give us a new array

const filterArray = array => array.filter(x => x > 5);

const array = [1, 2, 3, 4, 5, 6, 7, 8];

console.log(filterArray(array)); // [6, 7, 8]

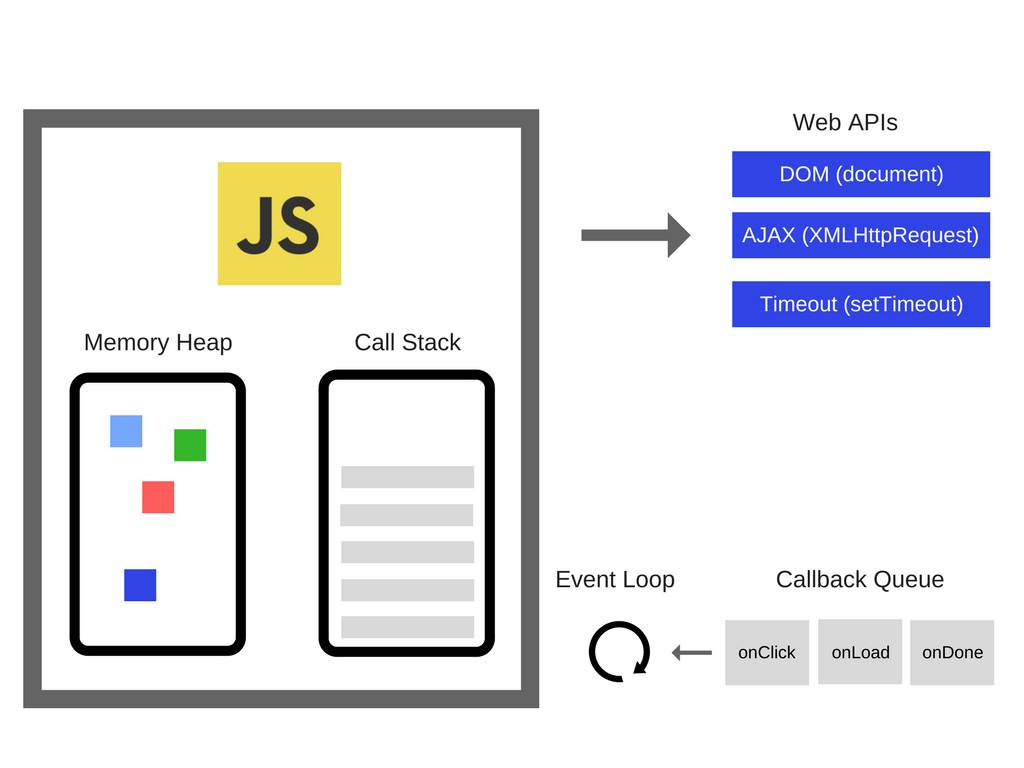
## Why is the functional approach preferred by most?

* This also offers code reusability where functions are composable (treated as components) just as is mostly done in React.
* Debugging is easier here
* Any developer can read and understand your code fast. Because you write what you think and not how the computer should think for you.

# How JavaScript works: an overview of the engine, the runtime, and the call stack

## ****The Runtime****

There are APIs in the browser that have been used by almost any JavaScript developer out there (e.g. “setTimeout”). Those APIs, however, are not provided by the Engine.



## The Call Stack

JavaScript is a single-threaded programming language, which means it has a single Call Stack. Therefore it can do one thing at a time.