1. Principles of JavaScript

• Thread of execution & Memory.

When JS code runs it:
- Goes through the code line-by-line and ~~runs/executes each line~~ : this is the thread of execution.

- Saves data (strings, array ..) so we can use that data later : this is the memory

• Functions & Execution context.

Function : Code we save ('define') functions & can use (call/invoke/execute/run) later with the function's name & ()

Execution context : Created to run the code of a function has two parts: — thread of execution — ~~memory~~ (local memory)

Example :

identifier (when the thread of execution arrive to this line)

```
const num = 3 ;
function multiplyBy 2 (inputNumber) {
    const result = inputNumber * 2 ;
    return result ;
}


const output = multiplyBy 2 (num) ;
const newOutput = multiplyBy 2 (10) ;
```

parameter

argument

can't change

function call (command)

global execution context

label   value

output = multiplyBy 2 (num) ;

Local Memory
inputNumber : 3
result : 6

return result

a new execution context for the function

In JavaScript, only one thing at a time. One thread of execution.

newOutput = multiplyBy 2(10)

new execution context

Local Memory
inputNumber : 10
result : 20

return result

Memory:

num : 3

multiplyBy 2 : → 8 →

output : 6

newOutput : 20

all the "strings" which compose the function are stored

can't store the function call. Only store data