JavaScript: The Game

Ahmed Moawad

Course Prerequisites

JS

Course Objectives

Learn about JavaScript, its uses and really understand it.

Learn how to build dynamic and interactive websites.

Make you fall in love with JavaScript

JS

JavaScript History

Netscape wanted to make the web more Dynamic

making a scripting language

Netscape hired **Brendan Eich** for that reason

Defend Idea by do a prototype

Eich do a prototype in 10 days

JavaScript first name was **mocha**

JavaScript Second name was **LiveScript**

And finally be **JavaScript**

JS

Fact #1

“There are two types of people, One who writes it “Java Script” and the other who writes it “JavaScript”. First one has no idea about what JavaScript is.”

level

JavaScript Language Core

HELLO WORLD!

```
alert('Hello World');

document.write('Hello World');

console.log('Hello World');
```

WHERE TO?

```
<html>
        <head>
          <script>
                alert('hello world');
          </script>
          <script  src="myscript.js"></script>
        </head>
         <body>
        <p>Hello</p>

          <script>
                alert('hello world');
          </script>
          <script  src="script.js"></script>
         </body>
<html>
```

index.html

```
alert('hello world');
```

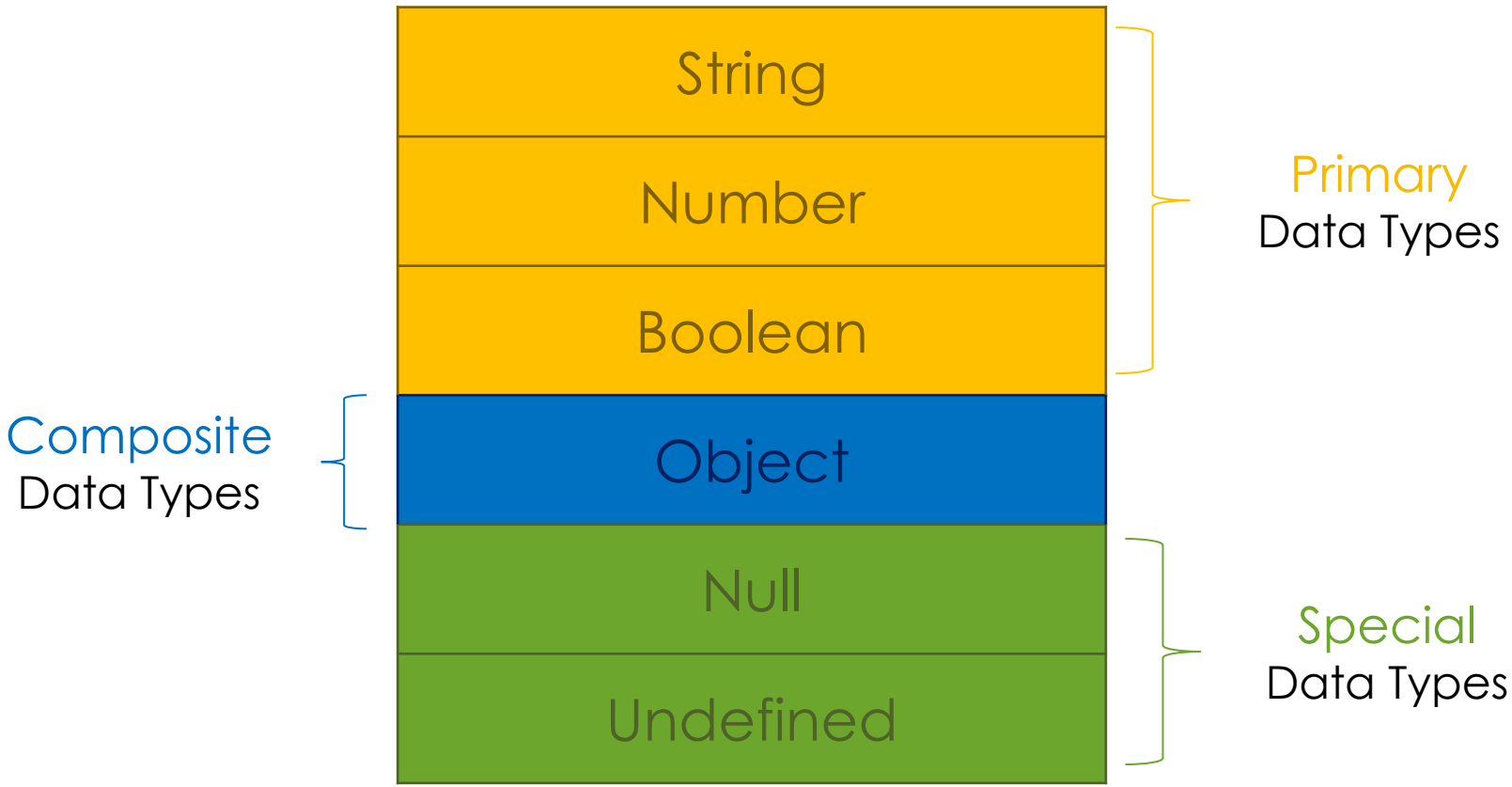
script.js

JAVASCRIPT SYNTAX

- JavaScript is case sensitive Var **is not equal to** **var**
- JavaScript statements are separated by **semicolons** (;) (optional But Best Practice).
- Variable Names follows this rules:
  - the first character must be a letter, an underscore (_), or a dollar sign ($). $dollar **(√)** _underScore **(√)** name **(√)** 12twelve **(x)**
  - Subsequent characters may be letters, digits, underscores, or dollar signs. $do22ar twelve12

```
var name;

var name, age, email;

var name, age=12;
```

DATA TYPES


String
Number
Boolean
Object
Null
Undefined
Primary
Data Types
Composite
Data Types
Special
Data Types

String

Any character array or text quoted

```
var str1="hello JS";
var str2='11.26';
var str3='false';
```

Number

Any Numeric value but **not** quoted

```
var num1= 8;
var num2= 11.26;
```

Boolean

Has only two values **true** or **false**

```
var isBool= true;
var isStr= false;
```

Null

This describes the no valid value , And has only one value **null**

```
var thisIsNull = null;
```

undefined

The **undefined** value is returned when you declare a variable that has never had a value assigned to it.

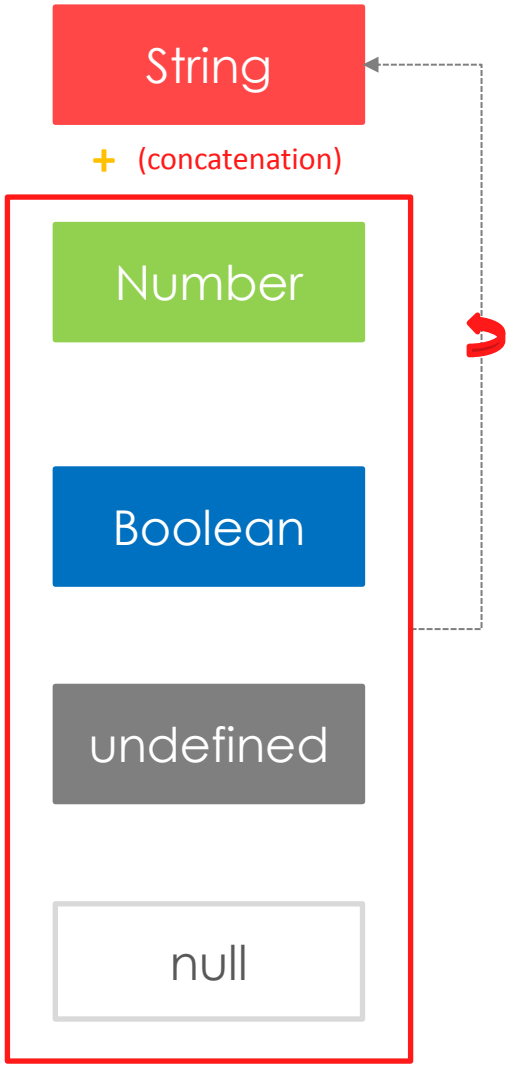
```
var num1; //num1 is undefined
```

```
var var1;                    //its type is undefined

var1 =“ahmed”;               //its type is string

var1 =12;                    //its type is number

var1 = true;                 //its type is boolean
```

```
var name;

typeof name; //undefined

name ="ahmed";

typeof name; //string

name = null;

typeof name;  //object How??

typeof name == 'object'; //true
```

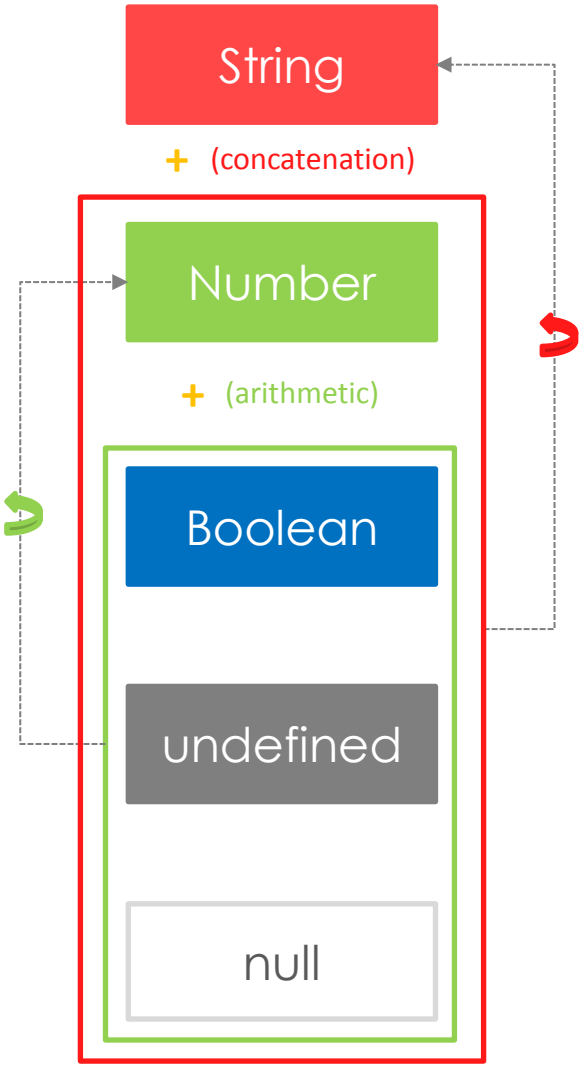
Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

OPERATORS | + operator



input	output
`"ITI" + " OS"`	ITI OS
`20 + "17"`	2017
`"is Exist " + true`	is Exist true
`"not " + undefined`	not undefined

OPERATORS | + operator



input	output
`"ITI" + " OS"`	ITI OS
`20 + "17"`	2017
`"is Exist " + true`	is Exist true
`"not " + undefined`	not undefined
`37 + null`	37
`37 + undefined`	NaN
`true + false`	1
`true + undefined`	NaN

a <op> **b**

Operator	Description
==	Return true if value of **a** *equal* to value of **b**.
===	Return true if value and type of **a** *equal* to value and type of **b**.
!=	Return true if value of **a** *not equal* to value of **b**.
!==	Return true if value and type of **a** *not equal* to value and type of **b**.
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Fight No#1

JS

==

It compares only the variable value

===

It compares the variable type and value

OPERATORS | == operator

1 | ● == ● ------> ● === ●

2 |  == ------> Number() === Number()

3 |  == null or undefined ------> false

4 | null == undefined ------> true

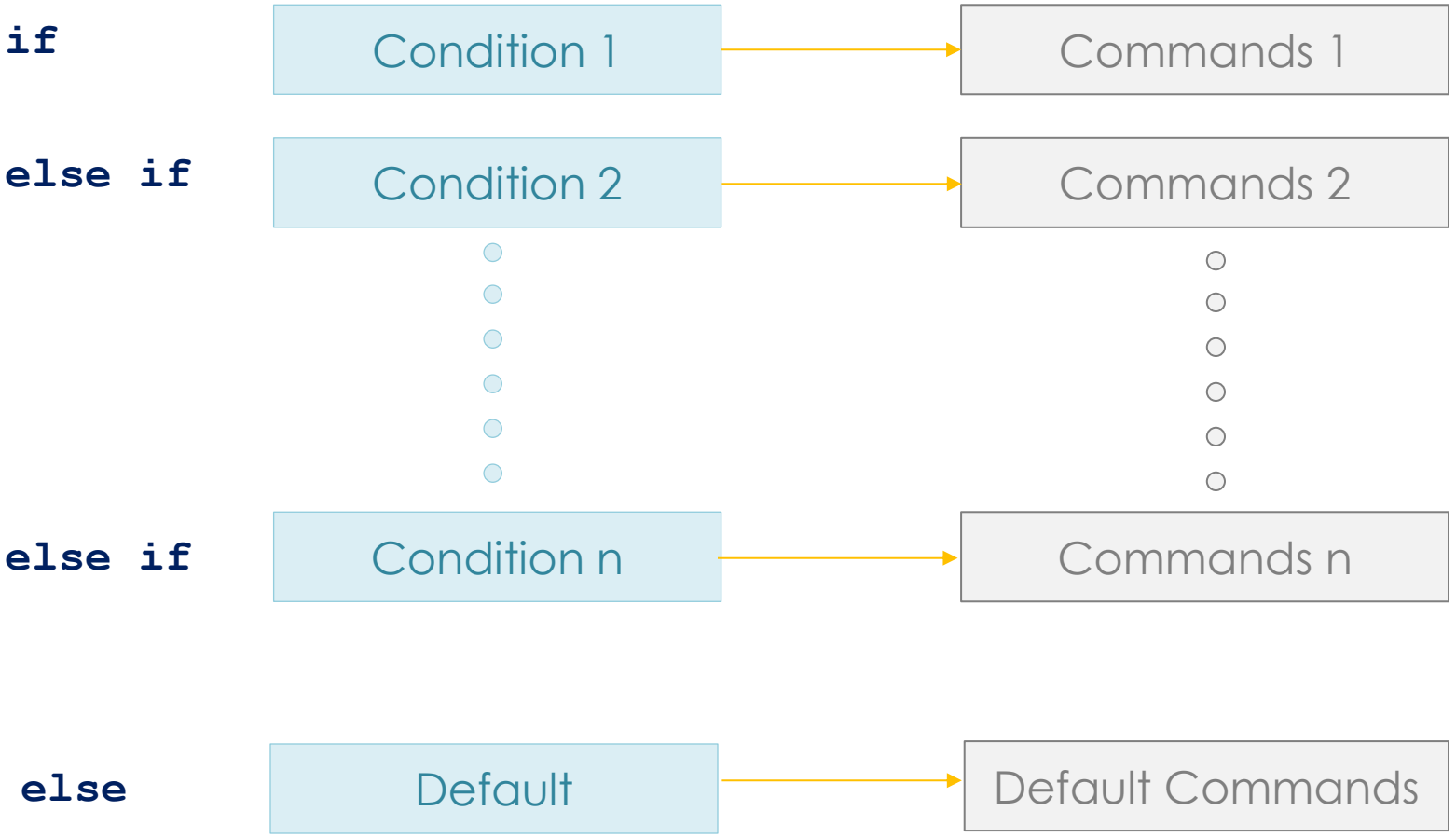
input	output
"20" == 20	true
0 == null	false
"true" == true	false
NaN == NaN	false
undefined == null	true

input	output
true == 1	true
true == 4	false
false == 0	true
NaN == undefined	false

Operator	Description
&&	**and** Gate
||	**or** Gate
!	**not** Gate

input	output
`(0 == null) && (true == 1)`	`false`
`(0 == "") || ("2" == 2)`	`true`
`!(null == undefined)`	`false`

CONDITIONS

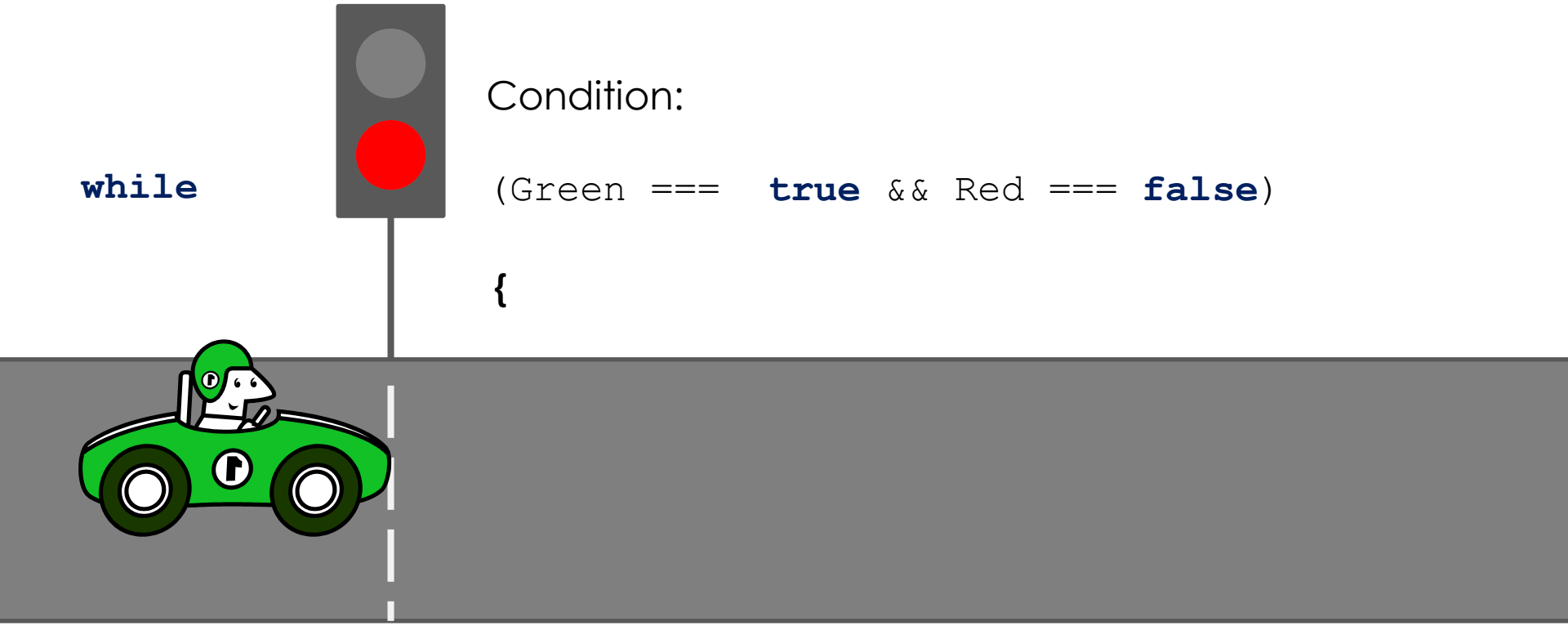

if
Condition 1
Commands 1
else if
Condition 2
Commands 2
else if
Condition n
Commands n
else
Default
Default Commands

```
if (name) {
        alert("hi");
}else{
        alert("Bye");
}
```

If name has falsy value it will execute the code in the **Else** statement

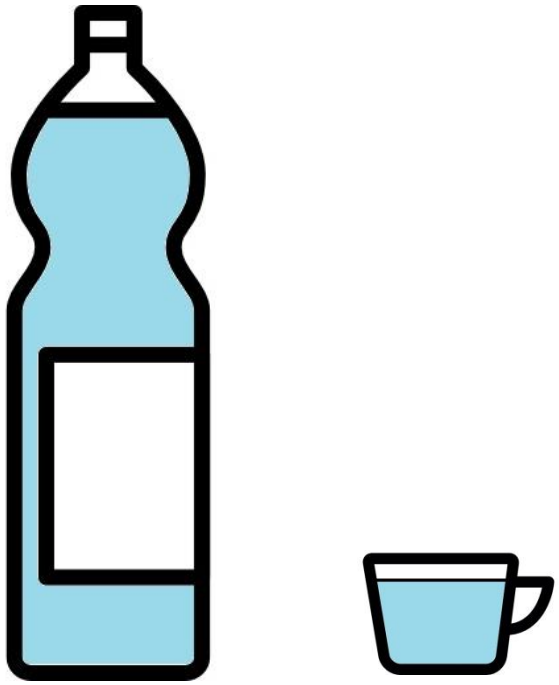
So what is the **falsy values** :

0 , **false, null ,undefined,** "", **NaN**



```
while (Green ===  true && Red === false)
{
    Command:

    Car.move();

}
```

```
for(var i=0;i<5;i++){
    Bottle.fill(cup,water);
}
```



i=5 = 5

For loop will finish executing here

Fight No#2

JS

continue

It makes program skip the current iteration of loop without completing it

break

It makes program exit loop without completing the remaining iterations.

```
alert(text);
```

Return : Doesn't Return any value

```
alert("Hello JavaScript!");
```

OR

```
var greetings = "Hello JavaScript!";
alert(greetings);
```

```
prompt(text, default return value);
```

Return : String

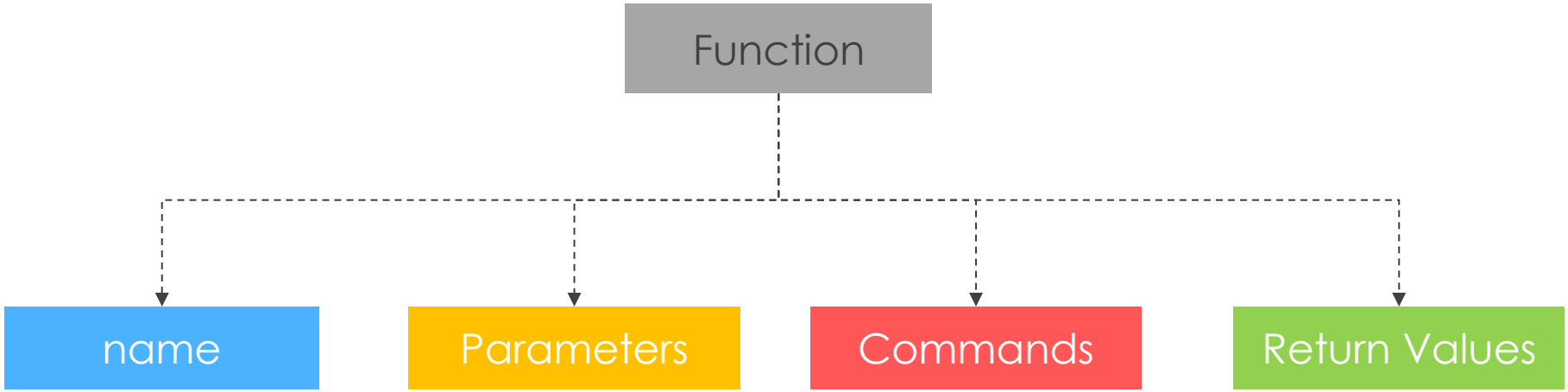
```
var person = prompt("Please enter your name", "Ahmed");

console.log(person) //person = Ahmed
```

`confirm(message)`

Return : Boolean

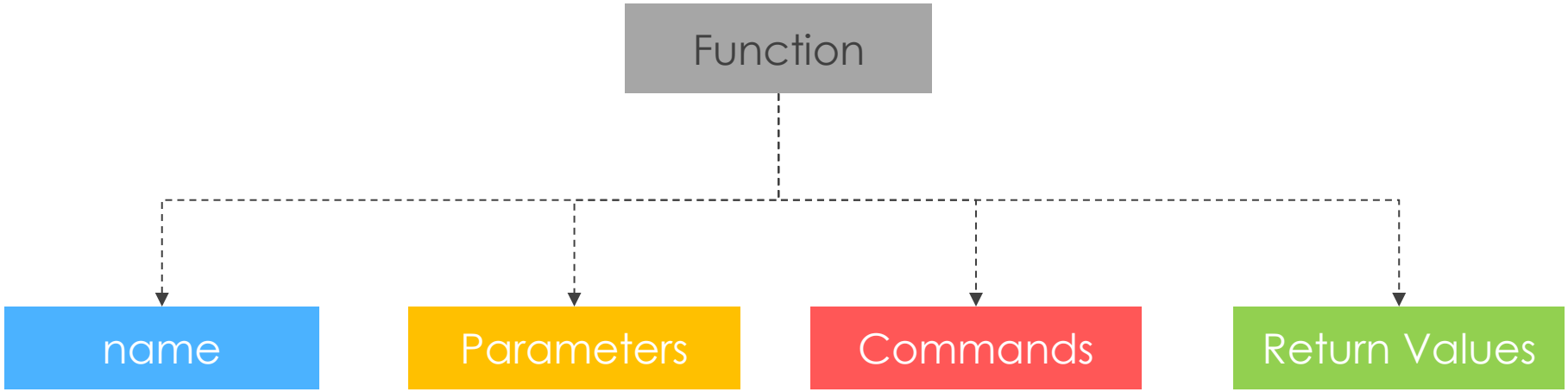
```
var isReady = confirm("Are you ready?");

if(isReady){
        alert("Yes");
}else{
        alert("No");
}
```

FUNCTIONS




```
function name(parameter1, parameter2, parameter3)
{
    code to be executed
    return true;
}
```

FUNCTIONS




```
function multiply(num1,num2) {
    var result = num1 * num2;
    return result;
}
```

Calling it:

```
var result = multiply(3,4);
alert(result); // result = 12
```

Global Scope

```
var  globalVar = 0;
```

function1

```
{
 var funcOneVar = 1;
 globalVar++;
 console.log(globalVar);
 console.log(funcTwoVar);
}
```

function2

```
{
 var funcTwoVar = 4;
 globalVar++;
 console.log(funcTwoVar);
 console.log(funcOneVar);
}
```

```
function1();
function2();
console.log(globalVar);
console.log(funcOneVar);
```

Result

```
1
undefined
4
undefined
2
undefined
```

Everything in JavaScript is an Object

JS

OBJECT .

Bike is an Object

It has characteristics

Properties

```
Bike.color
Bike.height
Bike.width
Bike.weight
```

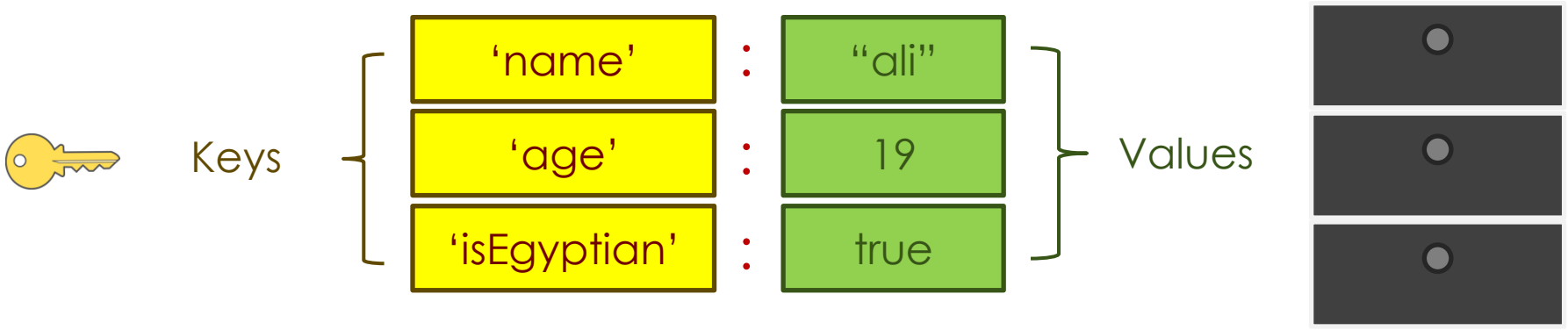
```
Bike.move()
Bike.stop()
Bike.brake()
Bike.jump()
```

Methods

It does some actions

OBJECT . .

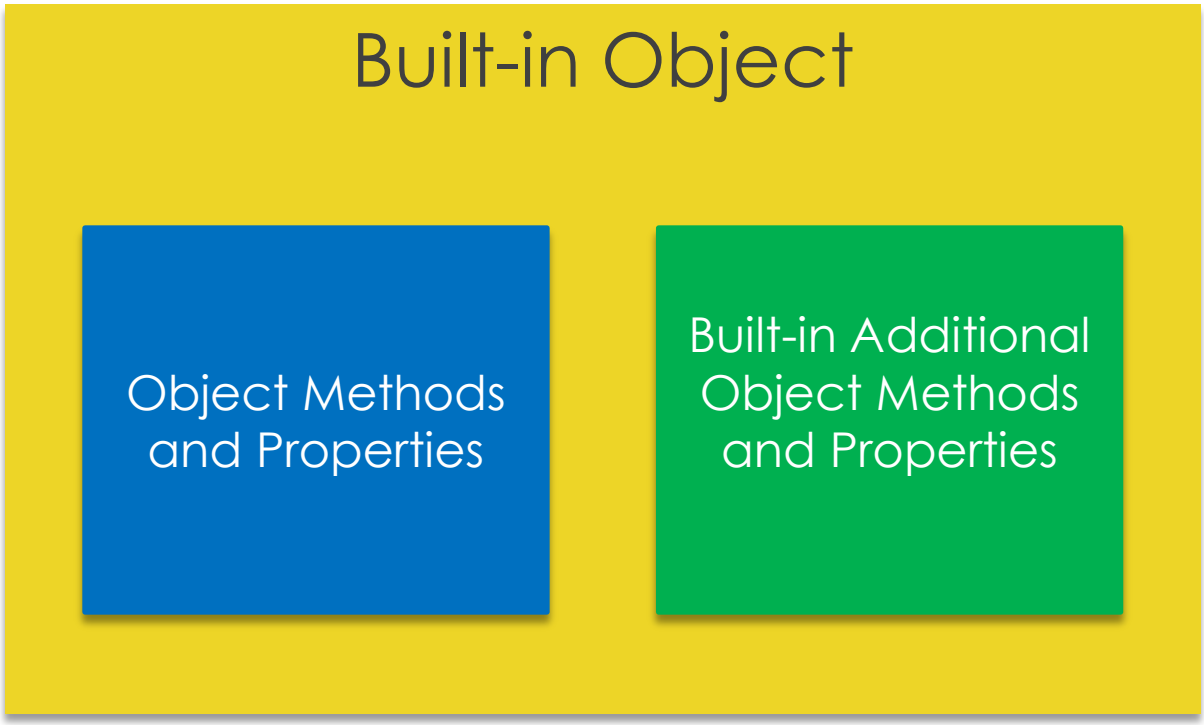
```
{
        name: 'ali',
        age: 19,
        isEgyptian: true
}
```



That's Enough for now !

BUILT-IN OBJECTS

They are helper objects that wrap some methods and properties about something like Date, Mathematical Operations, etc.



STRINGS

```
var  message = "this is string"
```

input	output
`message.toUpperCase()`	`THIS IS STRING`
`message.slice(5,7)`	`is`
`message.replace("is", "was")`	`thwas is string`
`message.charAt(2)`	`i`
`message.indexOf("is")`	`2`
`message.lastIndexOf("is")`	`5`

NUMBERS

```
var  num = 15.528
```

input	output
`num.toString()`	"15.528"
`num.toFixed(2)`	"15.53"
`num.toPrecision(3)`	"15.5"
`num.toPrecision(2)`	"16"
`parseInt(num)`	15

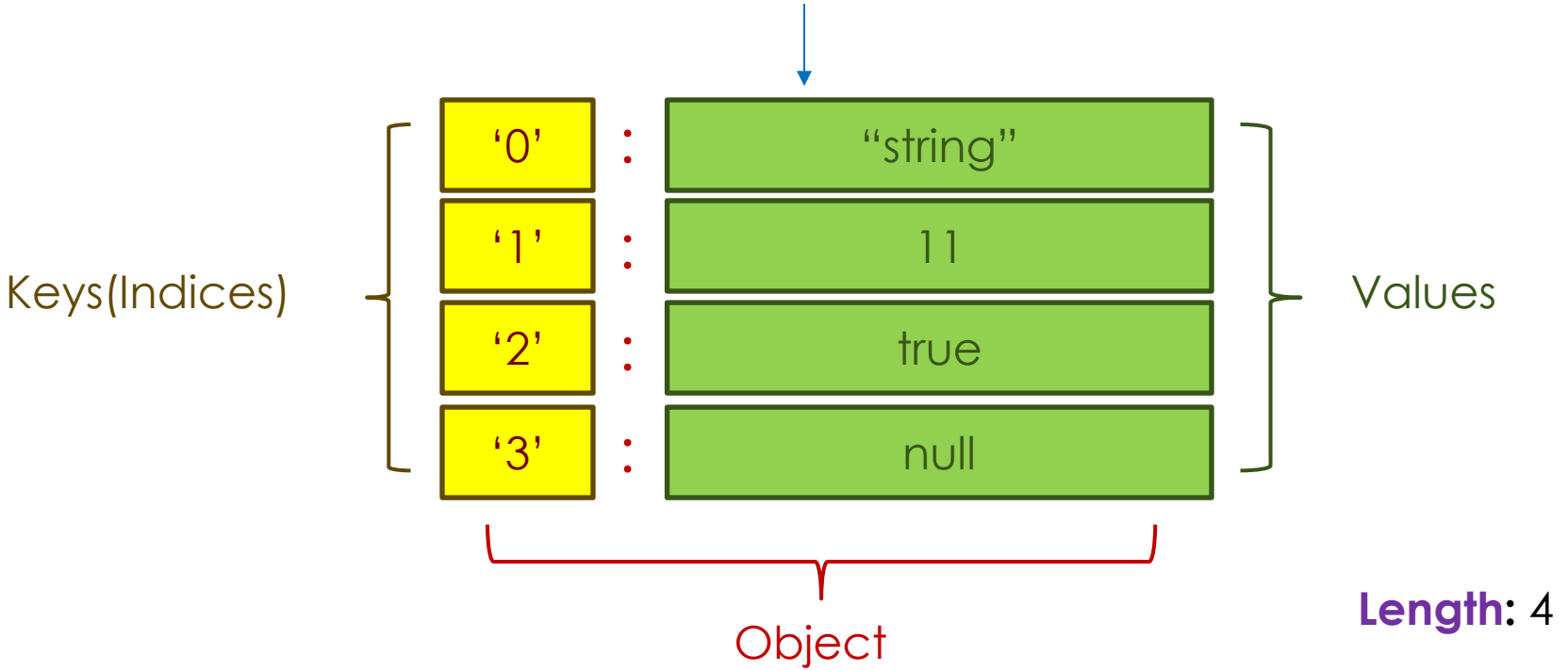
Arrays

Arrays are a special kind of objects, with numbered indexes.

JS

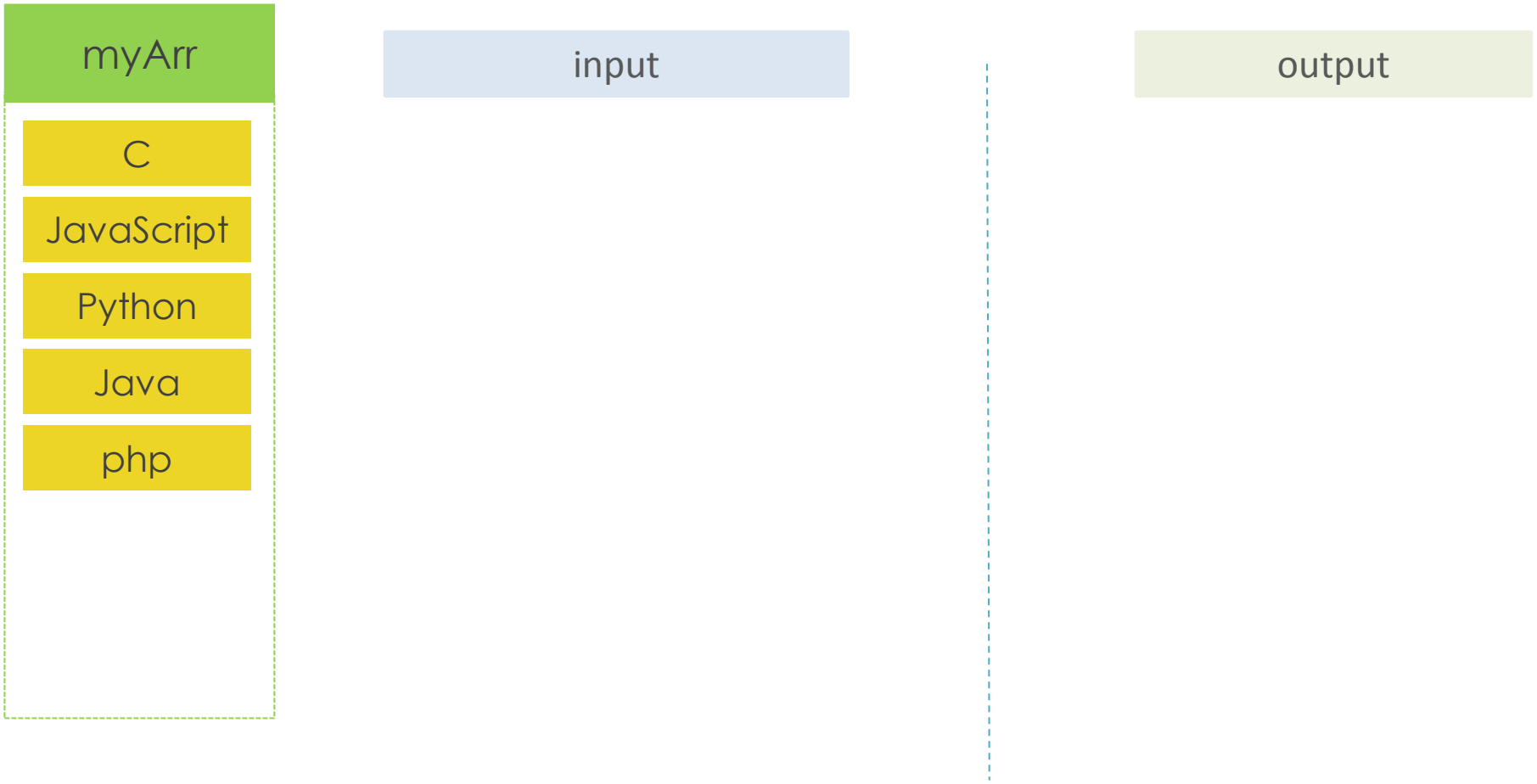
ARRAYS

```
var  myArr = ["string", 11, true, null];
```




```
myArr[50] = "no50";
```

```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```



```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```

input

```
myArr.pop()
```

output

```
php
```

```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```

input	output
`myArr.pop()`	`php`
`myArr.push("go")`	`4`

```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```

input	output
`myArr.pop()`	`php`
`myArr.push("go")`	`4`
`myArr.shift()`	`C`

```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```

myArr

- C++
- JavaScript
- Python
- Java
- go

input	output
`myArr.pop()`	php
`myArr.push("go")`	4
`myArr.shift()`	C
`myArr.unshift("C++")`	0

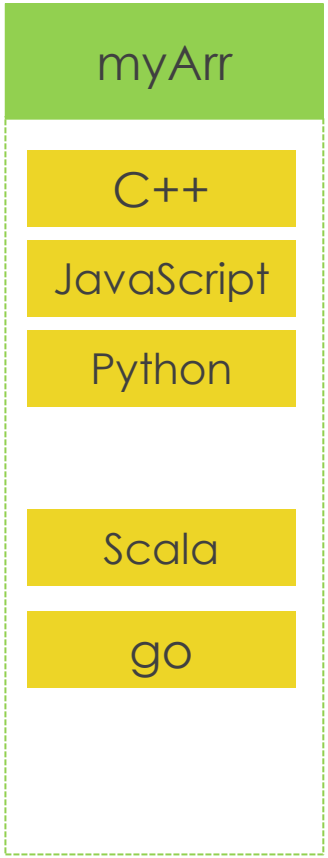
```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```

myArr

- C++
- JavaScript
- Python
- Java
- Scala
- go

input	output
`myArr.pop()`	php
`myArr.push("go")`	4
`myArr.shift()`	C
`myArr.unshift("C++")`	0
`myArr.splice(4,0,"Scala")`	[]

```
var  myArr = ["C", "JavaScript", "Python", "Java", "php"];
```



input	output
`myArr.pop()`	`php`
`myArr.push("go")`	`4`
`myArr.shift()`	`C`
`myArr.unshift("C++")`	`0`
`myArr.splice(4,0,"Scala")`	`[]`
`delete myArr[3]`	`true`

MATH

The Math object allows you to perform mathematical tasks

input	output
`Math.PI`	3.14
`Math.sqrt(25)`	5
`Math.abs(-1)`	1
`Math.floor(1.6)`	1
`Math.ceil(1.4)`	2
`Math.round(1.5)`	2

```
var d = new Date()
```

Try
exploring its Methods and Properties

THANK YOU

ahmedmowd@gmail.com