

The background features a light gray illustration of a telescope on a tripod, angled upwards. Scattered around the telescope are various celestial bodies: a crescent moon, a ringed planet like Saturn, a striped planet, and a globe showing continents. Numerous small white dots of varying sizes are scattered throughout, representing stars or distant galaxies.

JavaScript

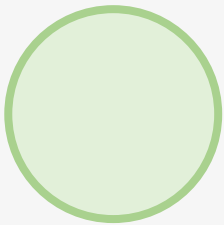
Attention

- Ce cours concerne **JavaScript** et non **Java**
- Les deux sont des langages de programmation différents



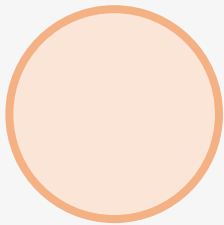
C'est quoi JavaScript ?

- Langage de programmation multi-paradigme



Application Web/Mobile

facebook

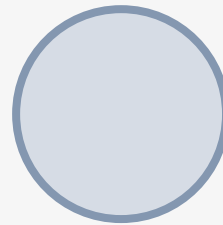


**Application en
temps réel**



WhatsApp

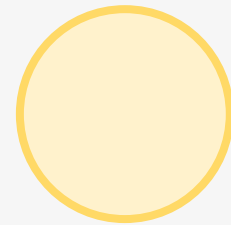
v



**Programme de ligne
de commande**



GitHub



Jeux



Candy Crush

ECMAScript **vs** JavaScript

- **ECMA script** est une norme
- Le **JavaScript** est un langage de programmation

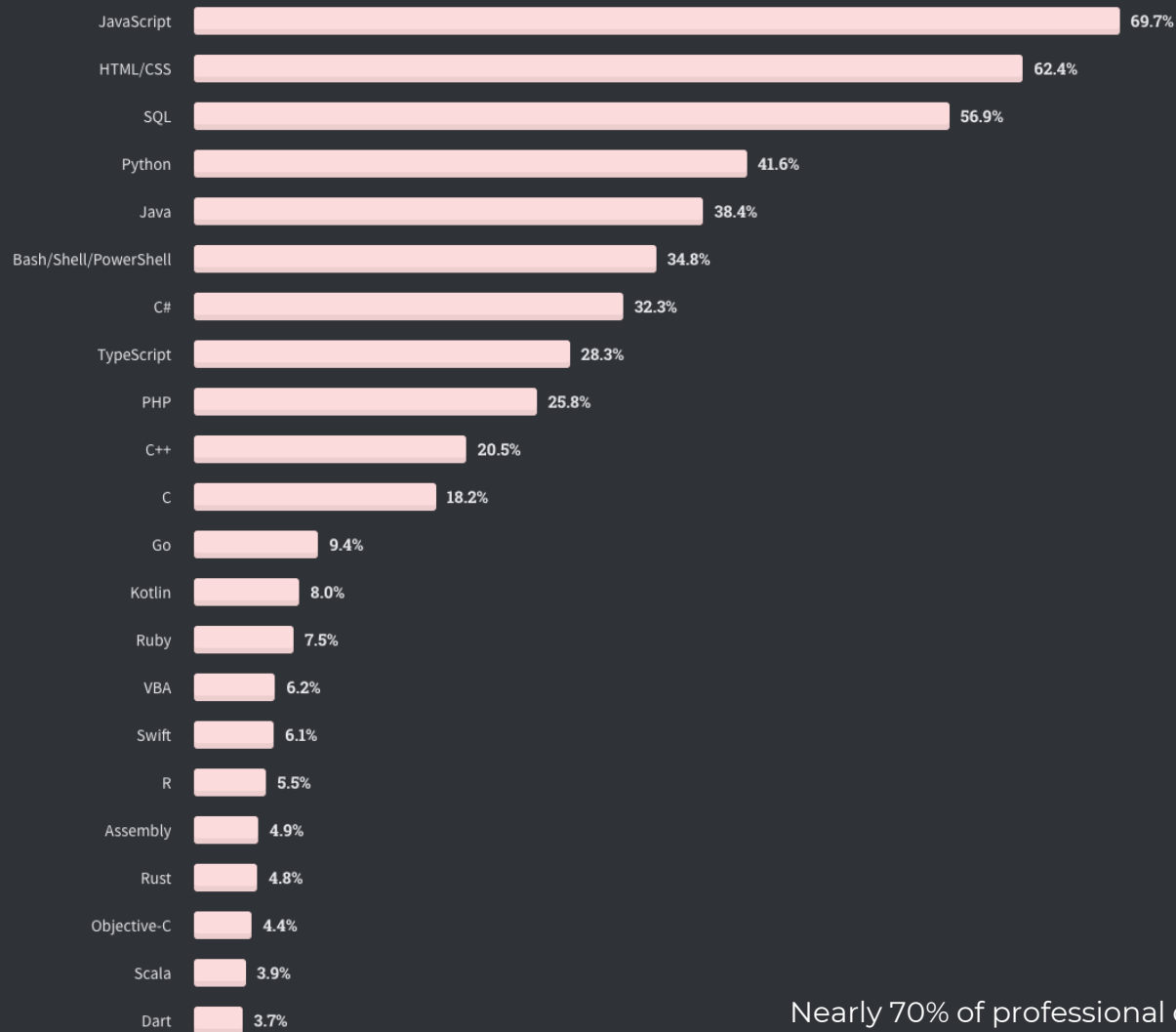
▼

ES6
ECMAScript6

JS
JavaScript

- [ECMA-262 - Ecma International \(ecma-international.org\)](http://ecma-international.org)

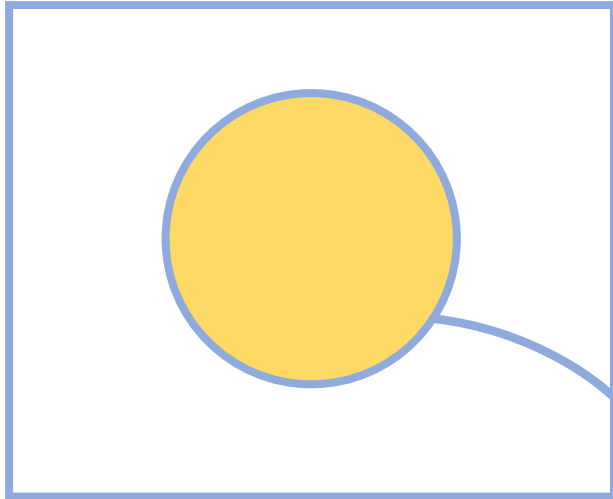
Langage de programmation très utilisé



Nearly 70% of professional developers who responded to the 2020 Stack Overflow survey coded in JavaScript.

Où se lance le JavaScript ?

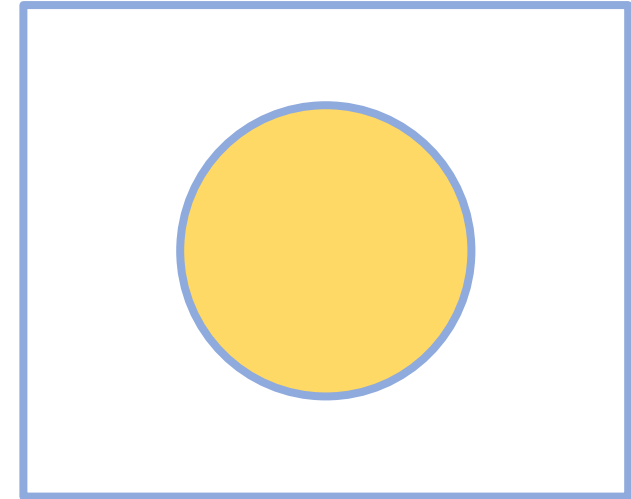
Navigateur



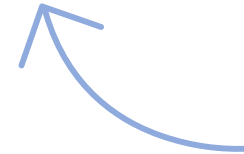
JavaScript Engine (Moteur)

Firefox : SpiderMonkey
Chrome : v8

Node.js



C++



Méthodologie d'apprentissage

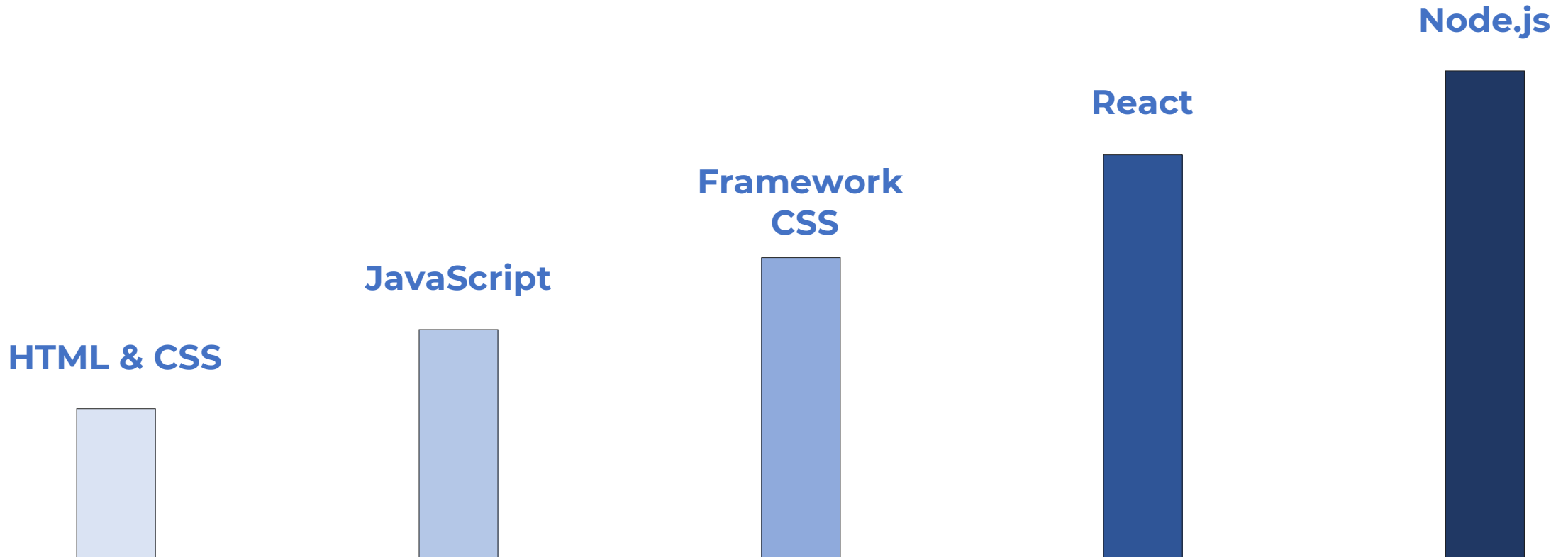
- Faut-il étudier le code ?

Non !

- Le plus important => **Comprendre les concepts !**
- Pour le reste il y a Google

Courbe d'apprentissage

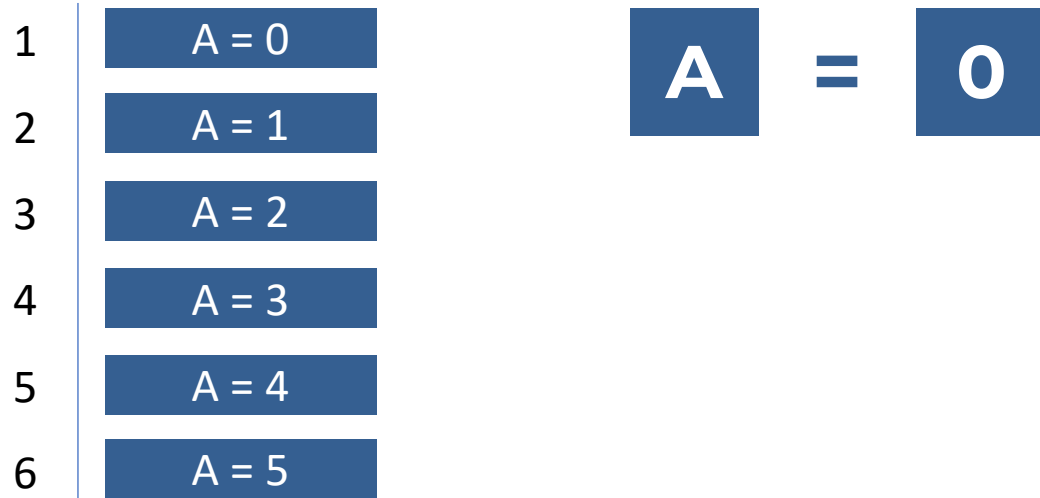
- **Exemple de framework css** : Bootstrap, tailwindcss, milligram, etc...
- **Il n'y a pas de bon ou de mauvais framework CSS**, ce sont des outils qui ont le même objectif !



Les variables

Qu'est ce qu'une variable ?

- Une variable peut se traduire par "**Ce qui varie**"
- En JavaScript, **une variable sert à stocker des données** qui peuvent être changées par la suite



Comment déclarer une variable ?

- Deux façons de faire ;

Mot-clé
(Keyword)

```
{ • let nomDeLaVariable = 1;  
  • const nomDeLaVariable = 2;
```

- Ancienne façon de faire, à ne plus utiliser ! ;

```
var nomDeLaVariable = 3;
```

Qu'est-ce qu'un mot-clé ?

- Les mots-clés sont **intégrés à JavaScript**, ils font partie des noms réservés
- *Exemple;*
 - let
 - const
 - return
 - typeof
- [List of JavaScript Reserved Words \(w3schools.com\)](https://www.w3schools.com/js/js_reserved.asp)

LET et CONST

let

La valeur de la variable **peut** être changée

```
JS scripts.js > ...
1  let x = 10;
2  x = 5;
3  console.log(x);
4
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\monsi\OneDrive\Bureau\JavaScriptIndex> node scripts.js
5
```

LET

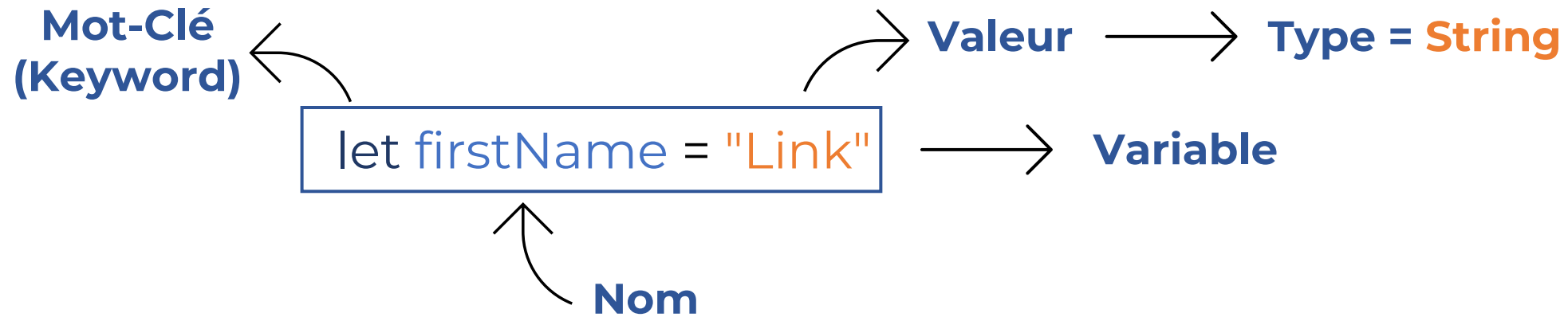
const

La valeur de la variable **ne peut** être changée

```
JS scripts.js > ...
1  const x = 10;
2  x = 5;
3  console.log(x);
4
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\monsi\OneDrive\Bureau\JavaScriptIndex> node scripts.js
C:\Users\monsi\OneDrive\Bureau\JavaScriptIndex\scripts.js:2
x = 5;
^
TypeError: Assignment to constant variable.
```

CONST

Déclaration de variable



Les types primitif de variable

- **Number** : 25
- **String** : "Sonic"
- **Boolean** : true ou false
- **Undefined** : non définie
- **Null** : inexistant ou introuvable

Les types de variables

Type primitif

- Number
- String
- Boolean
- Undefined
- Null


Type référence

- Object
- Function

La portée des variables

- Les accolades `{ }` représentent ce que l'on appelle le "**block-scope**"
- C'est qu'à **l'extérieur** de ces accolades, **les variables ne sont pas connues (undefined)**
- Sauf pour **VAR** 🐼

Les opérateurs

- Addition : +
 - Soustraction : -
 - Multiplication : *
 - Division : /
- 
- Ordre de priorité : **PEMDAS**
- Est plus grand que : >
 - Est plus grand que ou égale : >=
 - Incrémentation : ++ (+1)
 - Décrémentement : -- (-1)

Les opérateurs d'assignation

- **=** (Assignation)
- **+=** (Assignation additionné)

```
1 let age = 25;  
2 age += 5;  
3 console.log(age);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\Cours\Exercices JS\objet> node scripts.js  
30
```

Assignation additionné

```
1 let age = 25;  
2 console.log(age);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\Cours\Exercices JS\objet> node scripts.js  
25
```

Assignation

Les opérateurs logiques

- **&&** (ET)
- **||** (OU)
- **!** (NOT) (Non égale)
- **!=** (Inégalité)
- **!==** (Inégalité stricte)
- **===** (Égalité stricte)
- **==** (Égalité)

```
1 console.log(("5" === 5))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\Cours\Exercices JS\objet> node scripts.js  
false
```

Égalité
stricte

```
1 console.log(("5" == 5))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\Cours\Exercices JS\objet> node scripts.js  
true
```

Égalité

Truthy et Falsy

Concept JS présent dans un contexte Boolean

Tout est considéré comme

Truthy (**true**) sauf ;

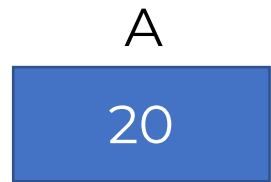
- false
- 0
- ""
- null
- undefined
- NaN

```
1 console.log(Boolean(false));
2 console.log(Boolean(0));
3 console.log(Boolean(""));
4 console.log(Boolean(null));
5 console.log(Boolean(undefined));
6 console.log(Boolean(NaN));
```

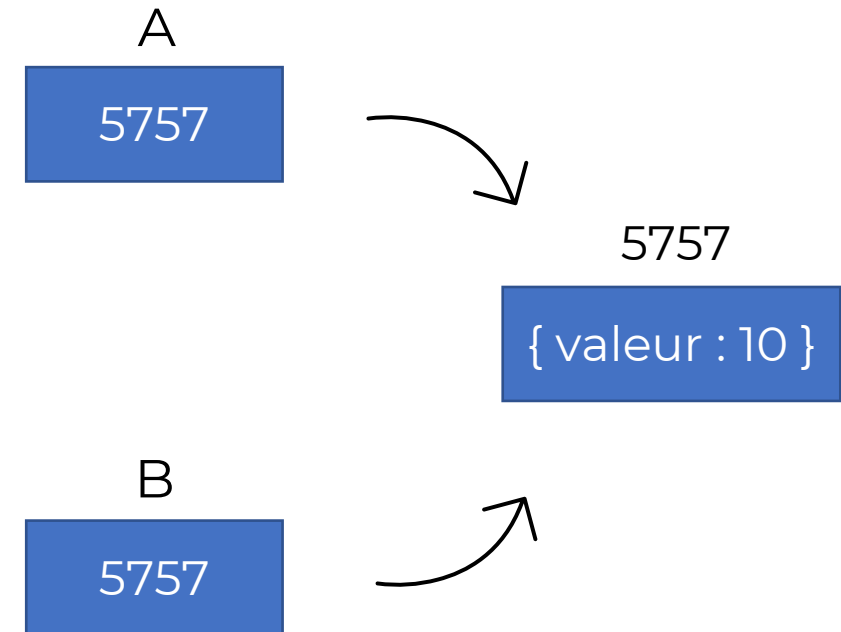
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\Cours\Exercices JS\objet> node scripts.js
false
false
false
false
false
false
false
```

Primitive



Référence



Conclusion du concept primitive/référence

- Les **primitives** sont copiées par leurs **valeurs**
- Les **objets** sont copiés par leur **référence**


Typage dynamique

- **Les langages à typage dynamique** sont ceux (comme [JavaScript](#)) dont l'interpréteur attribue aux [variables](#) un [type](#) lors de l'exécution en fonction de la [valeur](#) qu'elles possèdent à ce moment

JS scripts.js X

String

```
objet > JS scripts.js > [e] firstName  
1 let firstName = "Link"
```



PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

PS P:\Cours\Exercices JS\objet> node scripts

Typage implicite

TS scripts.ts X

```
objet > TS scripts.ts > ...  
1 let firstname: String = "Link";
```

Typage explicite

Les tableaux

Les tableaux

- Un tableau est semblable à une liste et a comme type object
- Type object => type référence

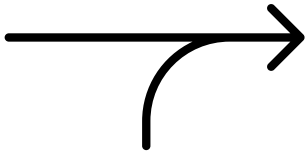
Syntaxe d'un tableau;

```
const monTableau = [ 5, "Hello", false, { name: "Hello World" } ]
```

- Ils peuvent contenir **tout type de variables**

Fonctionnement d'un tableau

- **L'index** d'un tableau commence à **0**
- Afficher une valeur précise dans un tableau ;
- **monTableau[1]** => Affiche l'index numéro 1 qui contient => 1

Index 

monTableau = ['a', 1, 3, 'b', 5]

Les tableaux

- Les tableaux sont utilisées dans plusieurs cas ;



Base de données



Liste d'éléments



Projet spécifique

- Ils peuvent contenir **tout type de variables**

Les objets

Les objets

- Un objet est une structure contenant des données et des instructions en rapport avec ces données

Syntaxe d'un objet;

```
const nomDeMonObjet = {  
  name : "Bob",  
  age: 25  
}
```

Fonctionnement d'un objet

- Un objet est une structure contenant des données et des instructions en rapport avec ces données

Syntaxe d'un objet;

```
const nomDeMonObjet = {  
  name : "Bob",  
  age: 25  
}
```

```
console.log(nomDeMonObjet.name) => Bob  
console.log(nomDeMonObjet.age) => 25
```

Les fonctions

Les fonctions

- Une fonction est une portion de code qui peut être appelée par d'autres codes (fonction) ou par elle-même

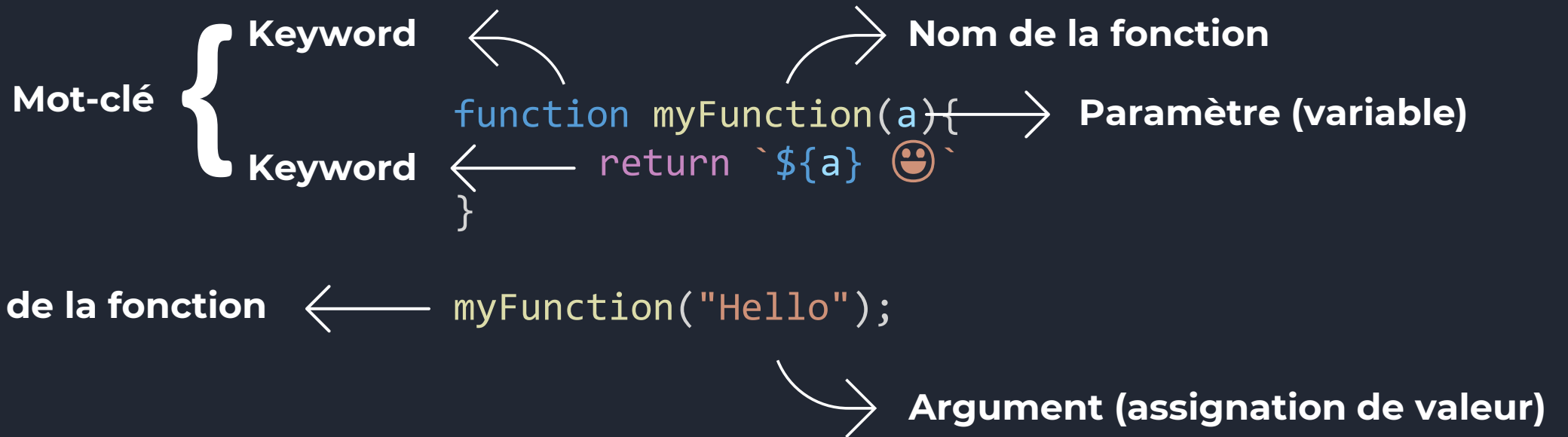
Syntaxe d'une fonction ;

```
function nomDeLaFonction(parametre){  
    return parametre  
}
```

Comment appeler la fonction ;

```
nomDeLaFonction('Hello')
```

Les fonctions en détails



La variable déclarée en paramètre a pour valeur l'argument passé à la fonction lors de son appel

Les méthodes

Les méthodes

- Une méthode est une fonction qui est une propriété d'un objet
- Elles sont intégrées à JavaScript

Une méthode est appelée de la sorte;

```
const nomDeMonObjet = {  
  name : "Bob",  
  age: 25  
}
```

```
nomDeMonObjet.nomDeLaMethode()
```

↑
point

Les méthodes de tableaux indispensables

- Ces méthodes fonctionnent uniquement avec les tableaux
- push()
- filter()
- map()
- find()
- indexOf()
- forEach()
- some()
- every()
- includes()
- [Array - JavaScript | MDN \(mozilla.org\)](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Array)

Le DOM

Découverte du DOM

- Document Object Model
- Le DOM est un modèle de document chargé dans le [navigateur](#). La représentation du document est un arbre nodal. Chaque nœud représente une partie du document

HTML document

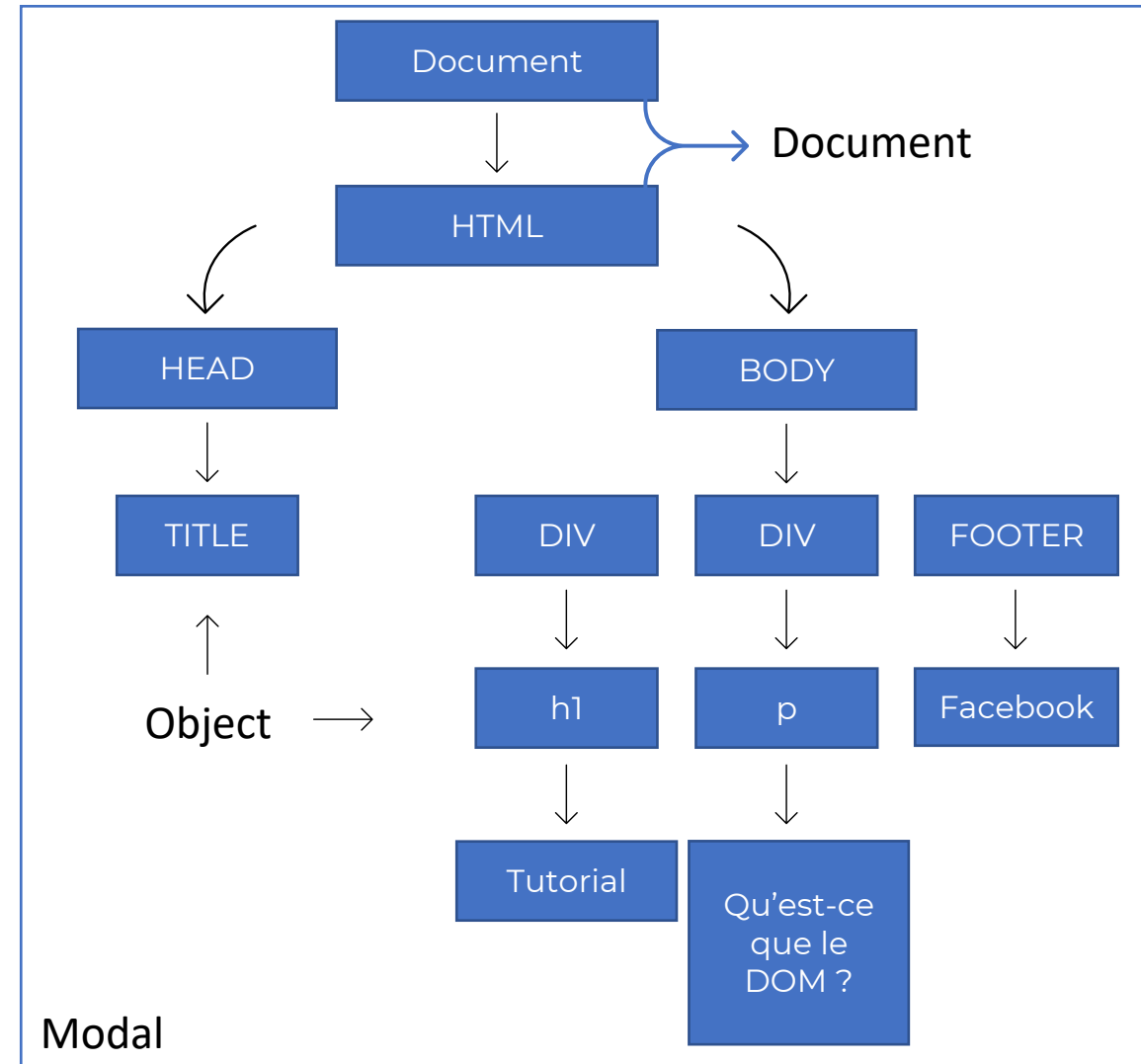
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Cours de JS</title>
  </head>

  <body>
    <div>
      <h1 id="title">Tutorial</h1>
    </div>

    <div id="paragraphe">
      <p>Qu'est-ce que le DOM ?</p>
    </div>

    <footer class="footer">Facebook</footer>
    <script src="scripts.js"></script>
  </body>
</html>
```

DOM – Vue par JS



Les méthodes de manipulation du DOM

Les méthodes de sélection des ID et classes

- document.querySelector(".nomDeLaClasse")
- document.querySelector("#nomDeLid")
- Une fois l'élément sélectionné on le stock dans une **variable** ;
- const **testClasse** = document.querySelector(".nomDeLaClasse")
- const **testId** = document.querySelector("#nomDeLid")

Les méthodes de manipulation du DOM

Les méthodes de création de contenu

- Après avoir sélectionné un élément et l'avoir assigné à une variable
- `const testId = document.querySelector("#nomDeLid")`
- `testId.innerText = "Hello World"`
- `testId.textContent = "Hello World"`