



FORMATION

IT - Digital - Management

**CURSUS JAVA SOPRA JUIN
2023**



m2iinformation.fr



MODULE JAVASCRIPT

Les fondamentaux du Langage Javascript

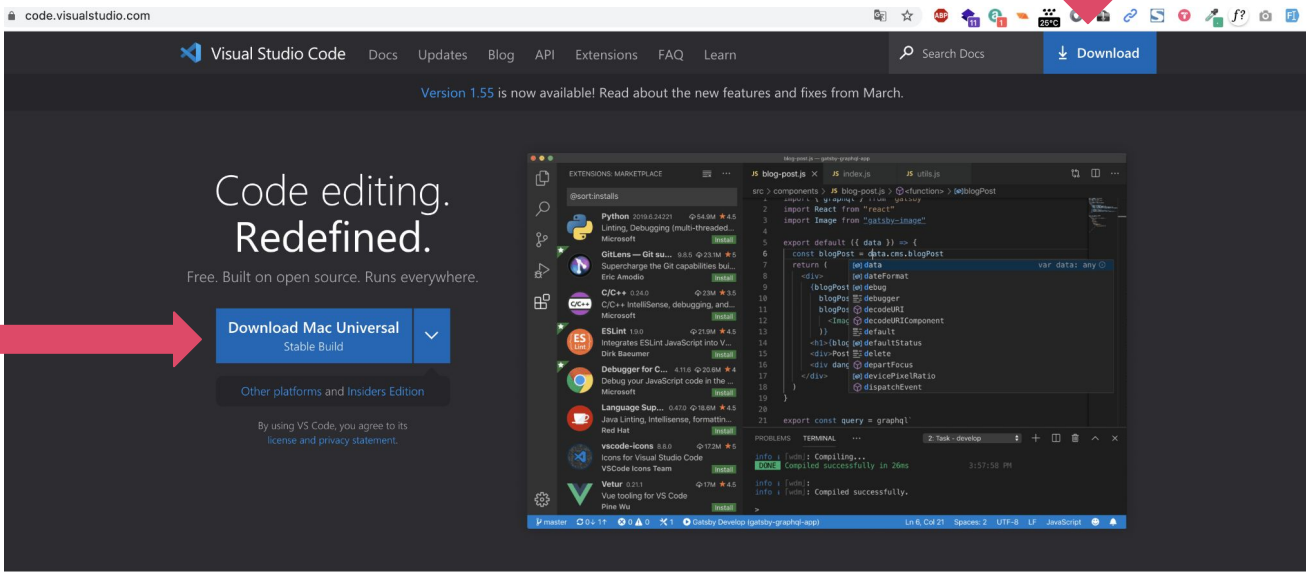
Formateur : Christian Lisangola

1.

Préparons
l'environnement

Install de vscode

<https://code.visualstudio.com/>



The screenshot shows the Visual Studio Code website. A red arrow points to the 'Download' button in the top navigation bar. The main content area features the text 'Code editing. Redefined.' and 'Free. Built on open source. Runs everywhere.' Below this is a 'Download Mac Universal Stable Build' button. To the right, a preview of the Visual Studio Code interface is shown, displaying the Extensions Marketplace, a code editor with a React component, and a terminal window. At the bottom, four icons represent key features: IntelliSense, Run and Debug, Built-in Git, and Extensions.

code.visualstudio.com

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Version 1.55 is now available! Read about the new features and fixes from March.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download Mac Universal Stable Build

Other platforms and Insiders Edition

By using VS Code, you agree to its license and privacy statement.

EXTENSIONS: MARKETPLACE

@sortInstalls

- Python 2020.8.24.2021 54.8M 4.5 Microsoft
- GitLens — Git au... 8.8.5 23.1M 4.5 Supercharge the Git capabilities bu... Eric Azevedo
- C/C++ 0.24.0 23M 4.5 C/C++ IntelliSense, debugging, and... Microsoft
- ESLint 1.9.0 21.9M 4.5 Integrates ESLint JavaScript into V... Dirk Baumeier
- Debugger for C... 4.11.8 20.8M 4.4 Debug your JavaScript code in the... Microsoft
- Language Sup... 0.47.0 18.6M 4.2 Java Linting, Intellisense, formatin... Red Hat
- vscode-icons 8.8.0 17.2M 4.5 Icons for Visual Studio Code VSCode Icons Team
- Vetur 0.21.1 17M 4.5 Vue tooling for VS Code Pine Wu

blog-post.js src > components > A blog-post.js < <function> < @blogPost

```
1 import { useState } from 'react'
2 import React from "react"
3 import Image from "gatsby-image"
4
5 export default ({ data }) => {
6   const blogPost = data.cms.blogPost
7   return (
8     <div>
9       <blogPost>
10         <blogPost>
11         <blogPost>
12         <blogPost>
13         <blogPost>
14         <blogPost>
15         <blogPost>
16         <blogPost>
17         <blogPost>
18         <blogPost>
19         <blogPost>
20         <blogPost>
21         <blogPost>
22         <blogPost>
23         <blogPost>
24         <blogPost>
25         <blogPost>
26         <blogPost>
27         <blogPost>
28         <blogPost>
29         <blogPost>
30         <blogPost>
31         <blogPost>
32         <blogPost>
33         <blogPost>
34         <blogPost>
35         <blogPost>
36         <blogPost>
37         <blogPost>
38         <blogPost>
39         <blogPost>
40         <blogPost>
41         <blogPost>
42         <blogPost>
43         <blogPost>
44         <blogPost>
45         <blogPost>
46         <blogPost>
47         <blogPost>
48         <blogPost>
49         <blogPost>
50         <blogPost>
51         <blogPost>
52         <blogPost>
53         <blogPost>
54         <blogPost>
55         <blogPost>
56         <blogPost>
57         <blogPost>
58         <blogPost>
59         <blogPost>
60         <blogPost>
61         <blogPost>
62         <blogPost>
63         <blogPost>
64         <blogPost>
65         <blogPost>
66         <blogPost>
67         <blogPost>
68         <blogPost>
69         <blogPost>
70         <blogPost>
71         <blogPost>
72         <blogPost>
73         <blogPost>
74         <blogPost>
75         <blogPost>
76         <blogPost>
77         <blogPost>
78         <blogPost>
79         <blogPost>
80         <blogPost>
81         <blogPost>
82         <blogPost>
83         <blogPost>
84         <blogPost>
85         <blogPost>
86         <blogPost>
87         <blogPost>
88         <blogPost>
89         <blogPost>
90         <blogPost>
91         <blogPost>
92         <blogPost>
93         <blogPost>
94         <blogPost>
95         <blogPost>
96         <blogPost>
97         <blogPost>
98         <blogPost>
99         <blogPost>
100        <blogPost>
101      </blogPost>
102    </div>
103  )
104}
```

PROBLEMS TERMINAL

Task: develop

```
info [vite]: Compiling...
00% Compiled successfully in 26ms
3:57:58 PM
info [vite]:
info [vite]: Compiled successfully.
```

IntelliSense Run and Debug Built-in Git Extensions

Plugins VS Code

VsCode

- ❑ Emmet
- ❑ Prettier
- ❑ Auto Rename Tag
- ❑ npm Intellisense
- ❑ Live server
- ❑ Tout ce que vous voulez 😊 ...

2.

Introduction JS

Qu'est ce que le JavaScript?

- ▶ **JavaScript**, c'est quoi ?
 - ▷ JavaScript n'est pas Java !
 - ▷ Langage de **script** permettant d'interagir avec une page Web
 - ▷ JS permet de créer des pages **dynamiques** / des **interactions**
 - ▷ JS fonctionne avec HTML et CSS
 - ▷ Standardisé par **EcmaScript**
- ▶ Implémentations possibles :
 - ▷ Balises: `<script></script>`
 - ▷ Fichier externe: `<script type="module" src="file.js"></script>`

Syntaxe

- ▶ Une instruction JS se termine toujours par un point virgule ;
- ▶ Les déclarations de variables doivent être précises - Attention à la casse !
 - ▷ `myVar` et `Myvar` sont 2 variables différentes.
- ▶ Les commentaires s'écrivent de deux façons :
 - ▷ `//` commentaires
 - ▷ `/*` commentaires

sur plusieurs

lignes

`*/`

Les variables

- ▶ **Déclarer / Manipuler** une variable
 - ▷ Mot clé **var** / **let**
 - ▷ Déclarer une variable: **let varName;**
 - ▷ Initialiser une variable : **varName = 1;**
 - ▷ Additionner: **varName = varName + 4; / varName += 5**
 - ▷ Soustraire: **varName = 6-1; / varName -=4**
 - ▷ Multiplier / Diviser: **varName = 6*5; / varName = 4/2;**
 - ▷ Incrément / Décrément: **varName++; / varName--;**
- ▶ Les constantes:
 - ▷ Variables **non modifiables**
 - ▷ Mot clé **const**
 - ▷ Déclarer une constante: **const varName = 0;**

Les types de variable

- ▶ On dit que JavaScript à un **typage faible**
- ▶ Il existe **7 types primitifs** en JavaScript :

String	<code>let ville = "Toulouse";</code>
Number	<code>let prix = 55;</code>
Boolean	<code>let existe = true; / let existe = false;</code>
Null	<code>let varNull = null;</code>
Undefined	<code>let ville;</code>
Object	<code>let ville = { nom: "Toulouse", cp:"31100" }</code>

Outils et méthodes utiles

- ▶ Afficher une alerte

- ▷ `alert("mon message"); / alert(maVariable);`

- ▶ Afficher la console du navigateur :

- ▷ `console.log("mon message"); / console.log(maVariable, "1");`

- ▶ Concaténation :

- ▷ `let a = "Je"; / let b = "suis"; / let c = a + b;`

- ▶ Conversion de type:

- ▷ String en nombre (ou float) : `parseInt()` - `parseFloat()`

- ▷ All en string: `string()`

Opérateurs logiques

- Les conditions ont besoin d'opérateurs logiques, en voici quelques uns :

Egalité	==
Eglité stricte	===
Différent de	!=
Différent stricte de	!==
Plus grand / petit	> / < / >= / <=
ET / OU / NON	&& / / !

Structures conditionnelles

- Condition IF - ELSE

```
if (condition1) {  
    // Traitement IF  
} else if (condition2) {  
    // Traitement ELSE IF  
} else {  
    // Traitement ELSE  
}
```

- Condition ternaire :

- condition ? if true : if false;

```
function getFee(isMember) {  
    return (isMember ? '$2.00' : '$10.00');  
}
```

Les boucles

- ▶ FOR : `for (let i=0; i<10; i++) { ... };`
- ▶ FOR ... IN : `for (index in tab) { tab[index] };`
- ▶ FOR ... OF : `for (index of tab) { index };`
- ▶ WHILE : `while (condition) { ... };`
- ▶ DO ... WHILE : `do { ... } while {condition};`
- ▶ Le mot clé **break** permet de sortir d'une boucle prématurément.

Tableaux : Déclaration et accès

Déclaration d'un tableau :

- ❑ `const monTableau = []`

- ❑ `const monTableau = new Array()`

Remplissage d'un tableau :

- ❑ `monTableau[0] = 4` : Mettre 4 à la première position 0

- ❑ `console.log(monTableau[4])` : Accès à l'élément situé à l'index 2

Tableaux : Méthodes

Voici les méthodes couramment utilisés des tableaux :

- ❑ `monTableau.push(valeur)`: push permet d'ajouter une valeur à la fin du tableau
- ❑ `monTableau.pop()` : pop permet de retirer un élément à la fin du tableau
- ❑ `monTableau.shift()` : Retirer un élément au début du tableau
- ❑ `monTableau.unshift(valeur)` : Ajouter un élément au début du tableau
- ❑ `monTableau.length` : Attribut qui permet de connaître la taille du tableau

Tableaux : Tableaux à n dimensions

Il est possible en Javascript de créer des tableaux à plusieurs dimensions:

Pour par exemple créer un tableau à 2 dimensions, il suffit de créer un tableau qui contient un autre tableau.

❑ `const matrice=[[2 , 3 , 4] , [5 , 9 , 8] , [2 , 4 , 1]]`: Va créer une matrice carrée 3x3

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 9 & 8 \\ 2 & 4 & 1 \end{bmatrix}$$

Fonctions en Javascript

```
function maFonctionSansValeurDeRetour(param1, param2 ,...){  
    Instruction 1  
    Instruction 2  
    Instruction 3  
}
```

```
function maFonctionAvecValeurDeRetour(param1,param2,...){  
    Instruction 1  
    Instruction 2  
    return valeur  
}
```

Objets en Javascript

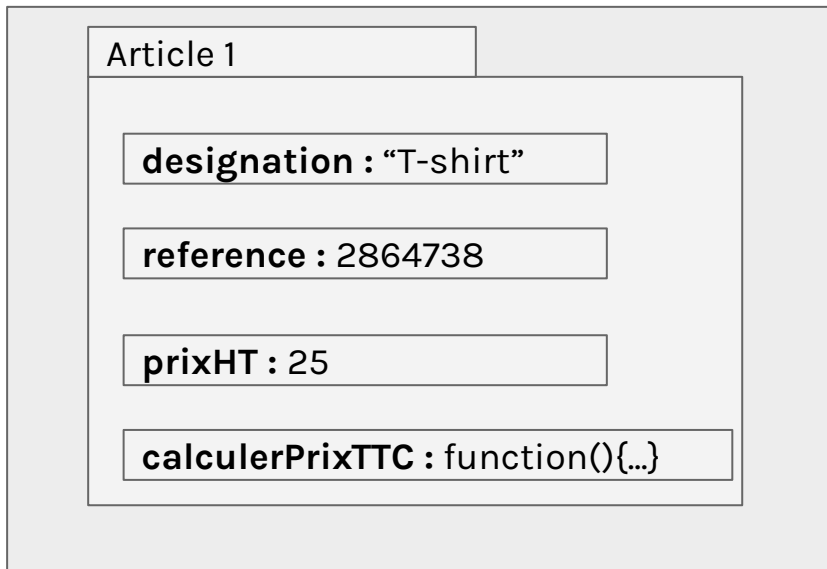
On va prendre un exemple concret. Supposons que l'on veuille gérer une boutique en ligne d'articles de sport.

- ❑ Tous les articles vont avoir des **caractéristiques communes**. Ils vont tous avoir un **nom de produit**, une **référence** et un **prix hors taxes**.
- ❑ Ils auront aussi une fonctionnalité commune car pour tous les articles on aura besoin de **calculer le prix toutes taxes comprises**.

Objets en Javascript

- ❑ Au niveau du code que l'on va écrire pour gérer cette boutique on voit bien que pour **chaque article** on va avoir **besoin d'une variable produit** pour stocker le **nom du produit**, **d'une autre variable référence** pour stocker la **référence** et d'une **dernière variable prix HT** pour stocker le **prix hors taxes**.
- ❑ **Ces variables sont liées entre elles** car elles **correspondent à un article**. Et ces variables ont une valeur spécifique pour chacun de ces articles.
- ❑ Une idée peut consister à regrouper ces **variables** et de les **encapsuler dans une "boîte"**. Cette boîte **on va lui donner un nom** et **pour ça on va se servir d'une variable**. On va l'appeler article 1 par exemple.
- ❑ Cette boîte est un **objet**.

Objets en Javascript



Objets en Javascript : Syntaxe

```
const article1 = {  
  designation : "T-shirt",  
  reference : 2864738,  
  prixHT : 25  
  calculerPrixTTC : function(){ return this.prixHT*... }  
}
```

Pour accéder au attribut ou méthodes d'un dans l'objet, on y fait référence via le mot clé **this**.

Pour y accéder aux attributs:

- ❑ `article.designation`
- ❑ `article.calculerPrixTTC()`

Objets en Javascript : Constructor function

```
function Article(id,designation,reference,prixHT){  
    this.id=id;  
    this.designation=designation  
    this.reference=reference  
    this.prixHT=prixHT  
}
```

```
Article.prototype.calculerPrixTT=function(){...}
```

Objets en Javascript : Classes ES6

```
class Article{  
    constructor(id,designation,reference,prixHT){  
        this.id=id;  
        this.designation=designation  
        this.reference=reference  
        this.prixHT=prixHT  
    }  
    calculerPrixTT(){...}  
}
```


2.

API DU DOM

DOM

Le DOM est une API qui permet d'interagir avec un document HTML ou XML avec un script(Javascript).

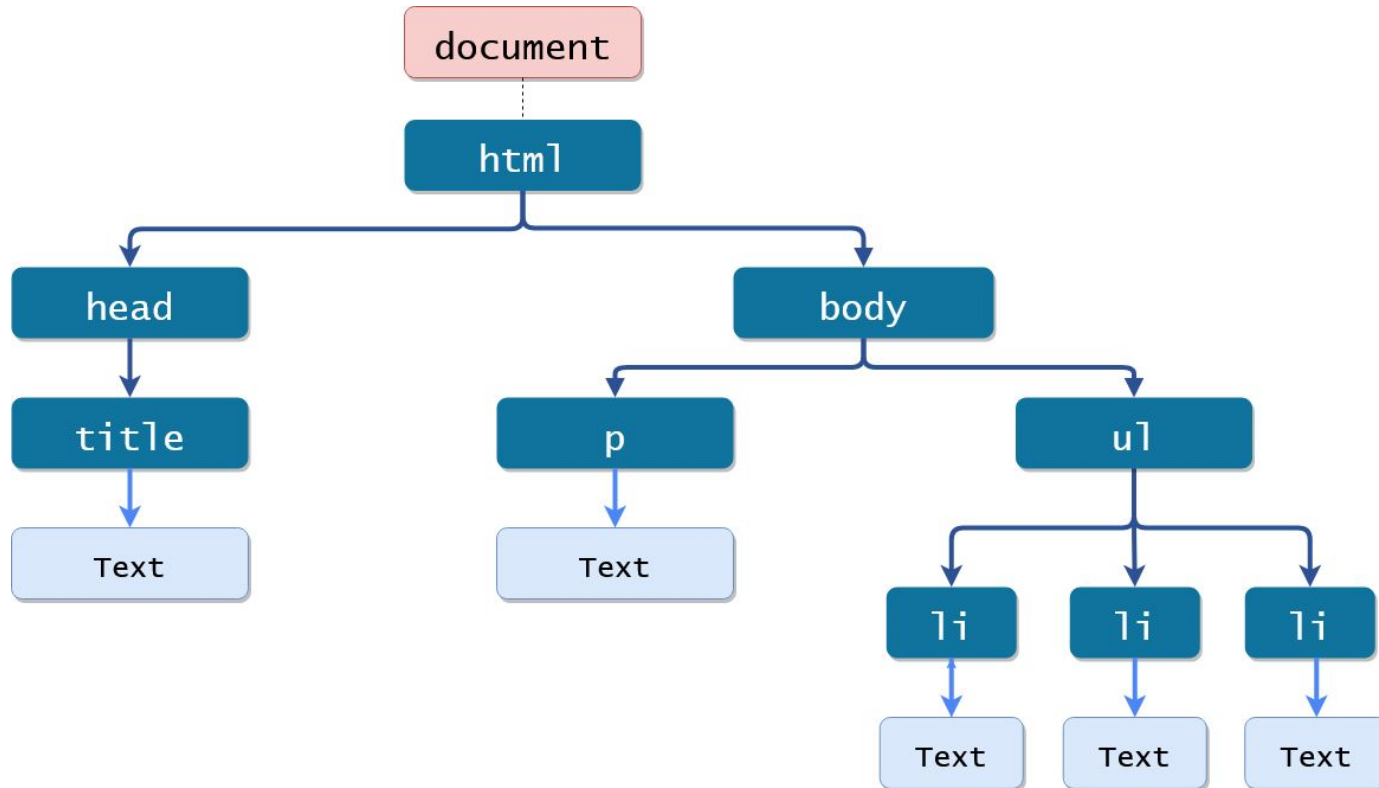
C'est aussi un modèle fidèle d'un document HTML ou XML en forme d'un arbre où chaque élément HTML est représenté par un objet possédant de propriétés, de méthodes et des événements.

DOM

Les actions qu'on peut faire à travers le DOM:

- ❑ Accédez aux éléments du DOM(leur attributs ainsi que leur contenu)
- ❑ Modifiez le DOM
- ❑ Écoutez des événements

DOM



DOM

L'interface Document permet de récupérer:

- ❑ **`document.querySelector(selecteur)`** : Retourne la première occurrence trouvé
- ❑ **`document.querySelectorAll(selecteur)`** : Retourne une collection de tous les élément de l'arbre du DOM qui match le sélecteur.

Exemple d'utilisation

- ❑ **`document.querySelector(".btn")`** : Cible la première occurrence de l'élément possédant la classe `.btn`
- ❑ **`document.querySelectorAll(".btn")`** : retourne tous les éléments possédant la classe `.btn`.

DOM

Accéder aux attributs des éléments:

- ❑ **document.querySelector(selecteur).value** : Cibler un champ input d'une forme
- ❑ **document.querySelector(selecteur).innerText** : lire contenu textuel d'un élément HTML

Modifier:

- ❑ **document.querySelector(selecteur).value=valeur** : modifier un champ input d'une forme
- ❑ **document.querySelector(selecteur).innerText=valeur** : Modifier/Insérer le contenu textuel d'un élément HTML
- ❑ **document.querySelector(selecteur).innerHTML=valeur** : Modifier/Insérer le contenu qui sera interprété comme de l'HTML par le navigateur

DOM

Manipulation du DOM:

- ❑ **`document.querySelector(selecteur).style.color=value`** : Modifier la couleur d'un élément ciblé
- ❑ **`document.querySelector(selecteur).classList.add(class1,...,classN)`** : Ajouter dynamiquement des classes aux éléments

Gestion des events:

- ❑ **`<objetDom>.addEventListener(event, callback)`**: modifier un champ input d'une formation
 - ❑ **`click`** : event click
 - ❑ **`submit`** : Soumission du formulaire
 - ❑ **`input`** : Saisi dans un champ de formulaire
 - ❑ ...

TP

Réaliser un programme qui permet de calculer les dimensions du cercle à partir du rayon.

Calcul des dimensions du Cercle

Rayon :

Diamètre :

Circonférence :

Aire :

Calculer

A dark blue diagonal shape that starts from the top right corner and extends towards the bottom left, creating a split background effect.

3.

APPELS AJAX

Ajax

Les appels Ajax sont un moyen pour permettre au développeur de récupérer les données de manière asynchrone depuis une API.

```
fetch(URL).then((response)=>{  
    return response.json()  
}).then(data=>{  
    //Votre traitement  
})
```

TP Final

Vous trouverez la maquette du projet à réaliser dans ce dossier :

<https://bit.ly/2Yr81WU>

Requirements

- ☐ Pour le stockage, utiliser localStorage.
- ☐ Le TP est à faire 16 h
- ☐ Possibilité d'utiliser bootstrap
- ☐ Le rendu final doit être un travail d'équipe