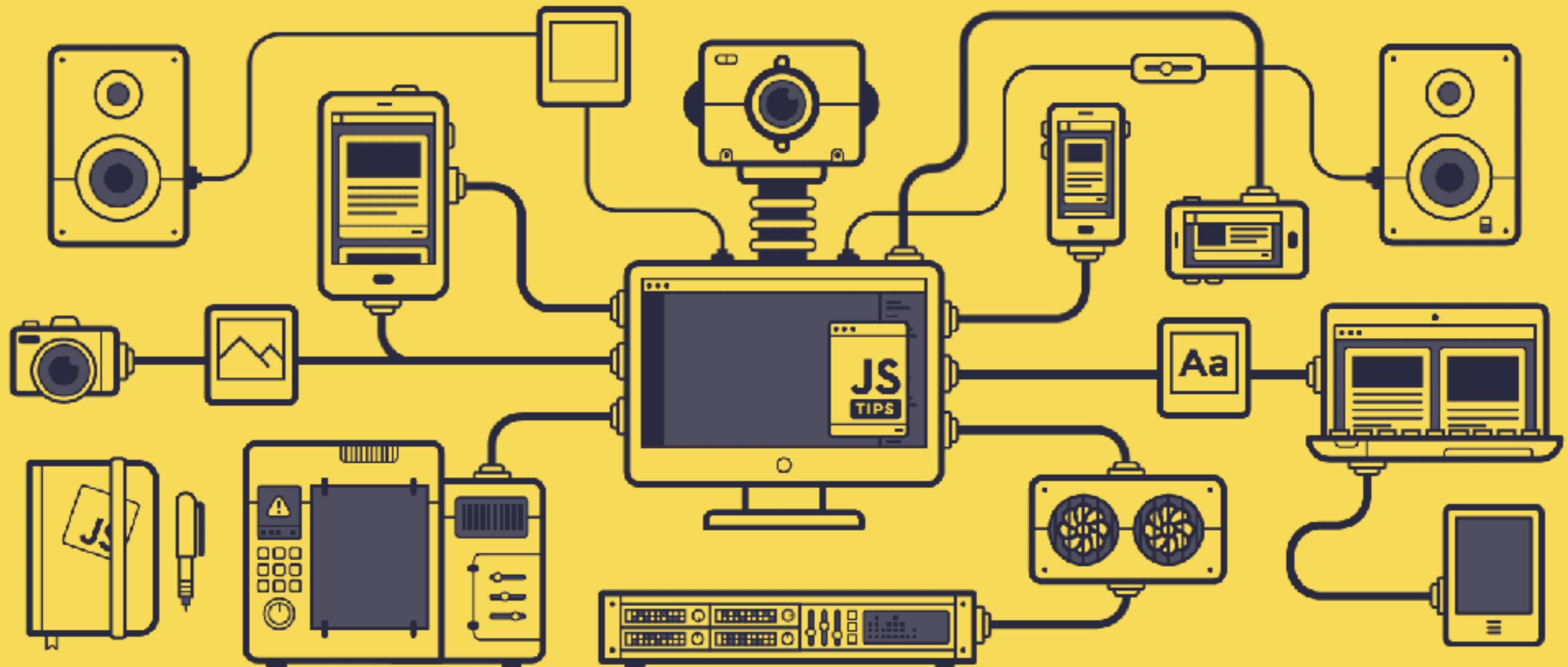


INTRO JAVASCRIPT



INTRODUCTION

SOMMAIRE

- Introduction - présentation
- Bases Javascript (syntaxe + Algorithme)
 - Type de valeur / assignation
 - Calcul
 - Chaine de caractères
 - Tableaux
 - Fonctions
 - Commentaires
 - Expressions de comparaison / opérateurs logiques
 - Condition IF ELSE
 - Les objets
 - Les Boucles while / For
- DOM
 - Manipuler les éléments du DOM
 - Réagir à différents évènements
- POO
 - Classes
 - Méthodes de classes

INTRODUCTION

JS

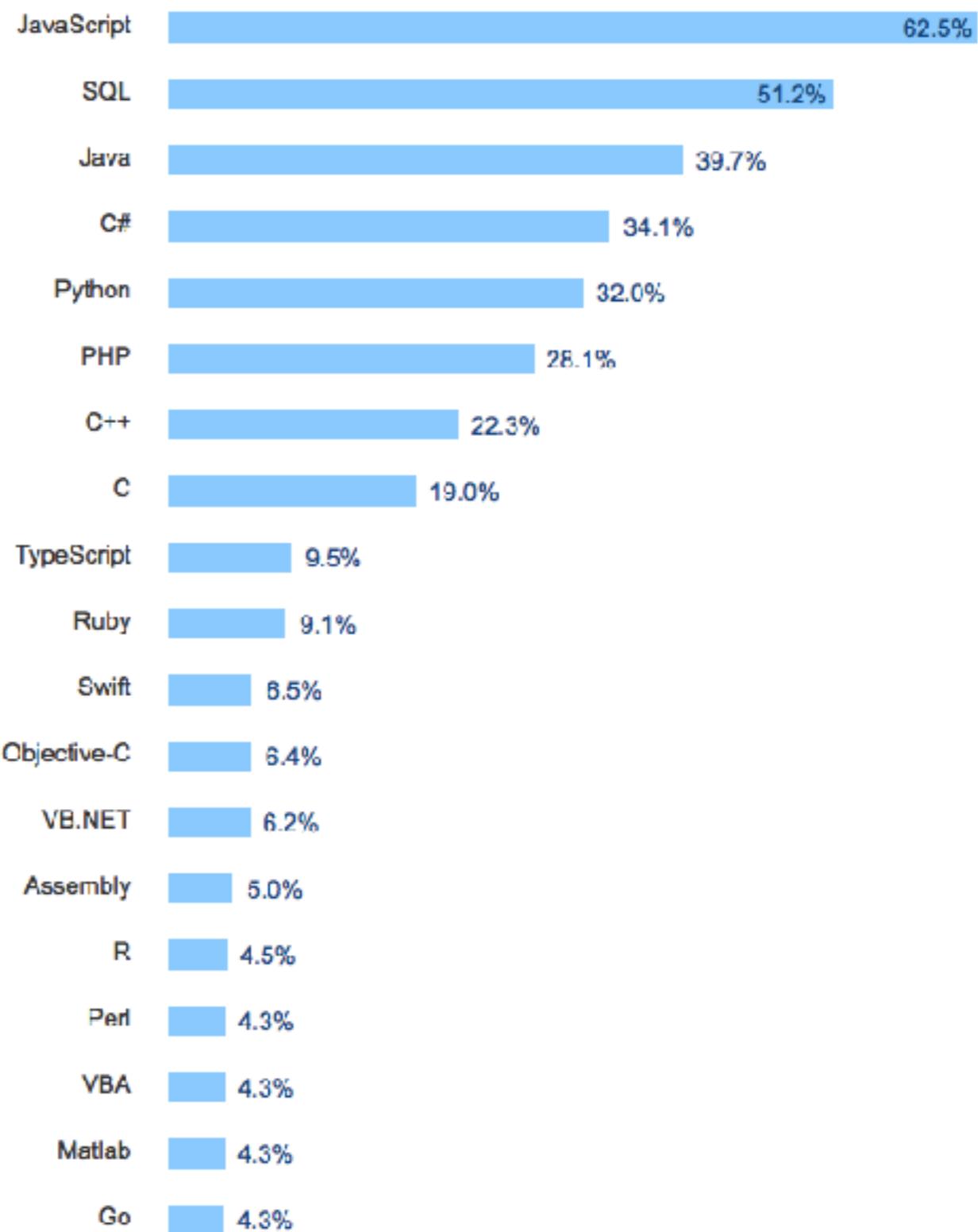
INTRODUCTION

HISTORIQUE

- 1995
- Brendan Eich
- 10 Jours
- Langage de programmation
 - Web dynamique
 - Orienté Objet
 - Multi Environnement



INTRODUCTION HISTORIQUE



OUTILS

JS

- Un navigateur web :
- Un éditeur de code :
(VS Code + live Server)

JS

BASE JAVASCRIPT



JS

BASE JAVASCRIPT

LOGIQUE DE JS

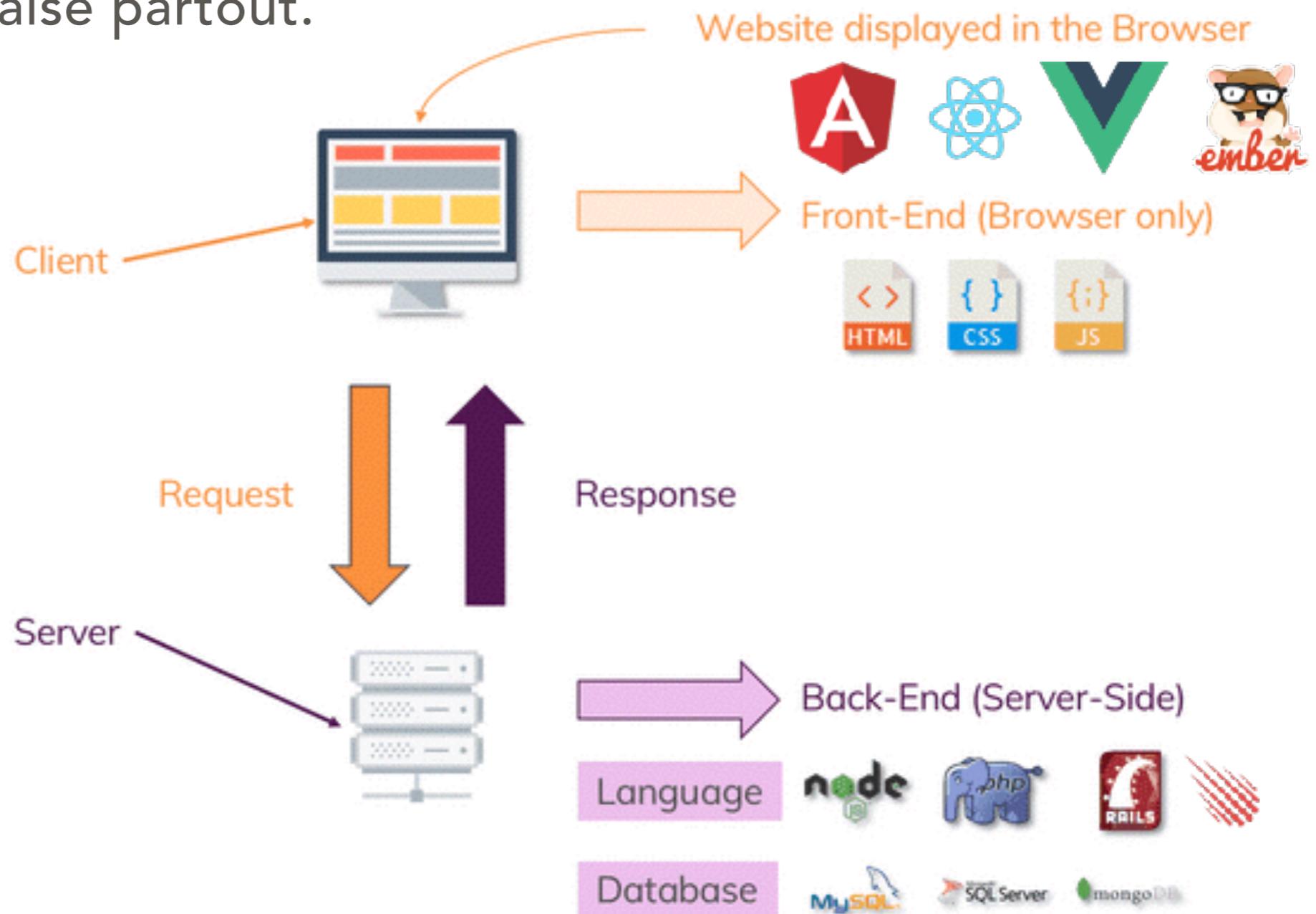
- Pendant longtemps JS était considéré comme un langage de second rang.
- Au début la plupart des développeur web l'utilisaient uniquement pour « bidouiller » quelques effets et des animations.
- Il arrive que l'on retrouve sur des sites ou des tutoriels « anciens » du code JS pas très propre.

BASE JAVASCRIPT

LOGIQUE DE JS



- L'idée c'est d'acquérir les bases et la logique du langage JS pour ensuite être à l'aise partout.





BASE JAVASCRIPT

LOGIQUE DE JS

- Objectif d'une application : interagir avec des données
- Ex : Nb Billet restant concert, Adresse Mail user, Num de commande etc ...
- Un programme utilise des variables pour enregistrer les données
- Variable = Contenant pour Enregistrer une données (ex Boite)
- Une donnée placée dans une variable = Valeur
- Variable = espace mémoire ou on stock une donnée



JS

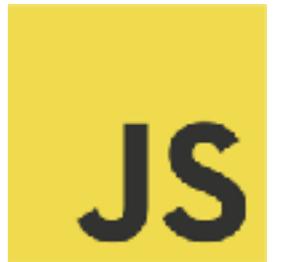
BASE JAVASCRIPT

LOGIQUE DE JS

- Une Variable est définie par 3 Composants :
- NOM (Pour identifier ce que c'est)
- TYPE (catégorie Nombres, textes etc...)
- VALEUR (Contenu de la variable)

BASE JAVASCRIPT

LOGIQUE DE JS

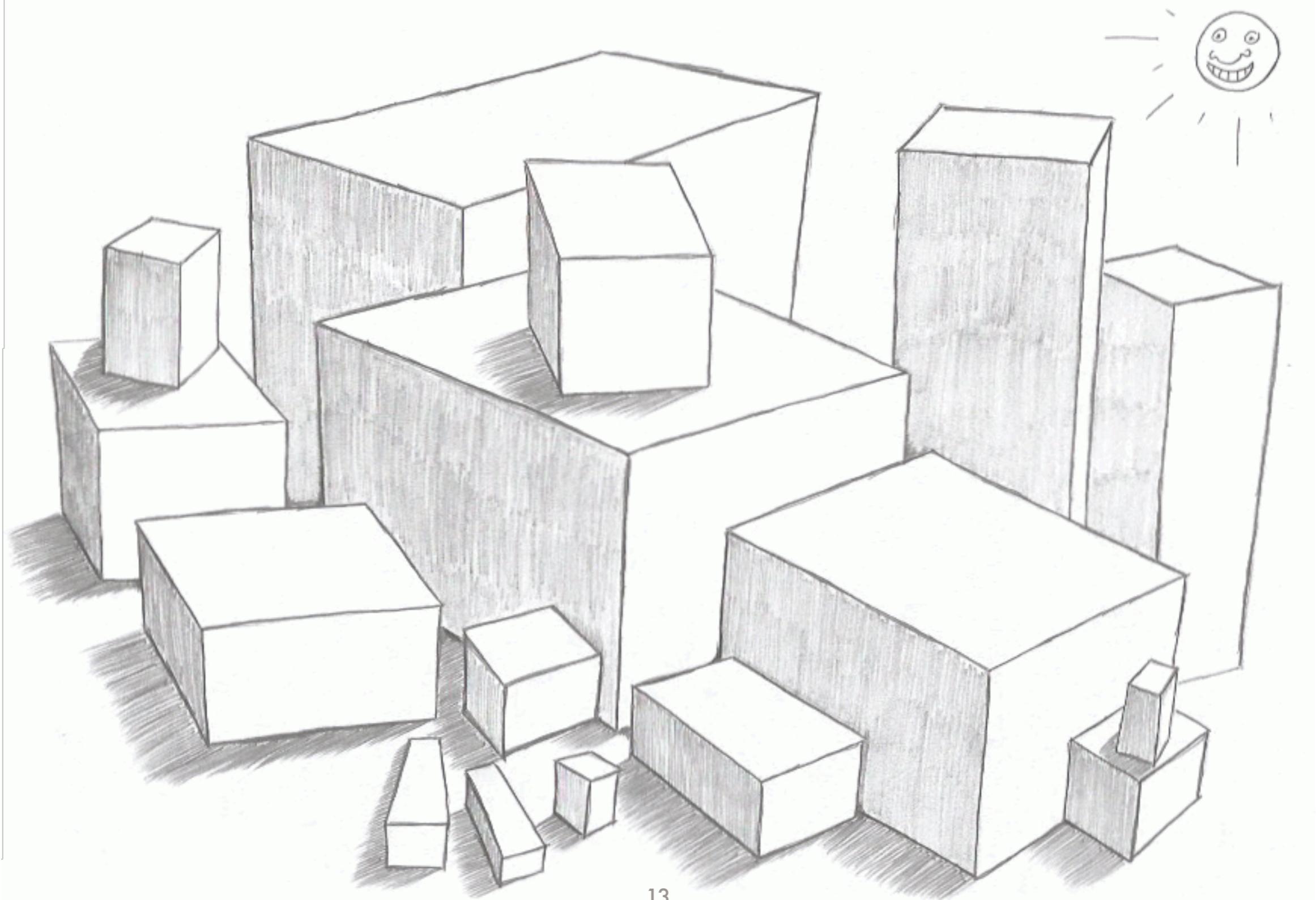


- Warning :
- Pour ASSIGNER une VALEUR à une VARIABLE on l'INITIALISE
- Crucial car ça permet à la VARIABLE d'être UTILISABLE par le programme
- Si on utilise une variable pas initialisée = BUG de l'appli / CRASH

BASE JAVASCRIPT

LOGIQUE DE JS

JS



BASE JAVASCRIPT

LOGIQUE DE JS

- Les framework
- Traduits littéralement, les frameworks sont « des cadres de travail ».
- c'est un ensemble de composants logiciels qui permettent de créer le squelette d'un logiciel ou d'une application. Il offre une architecture « prêt à l'emploi » afin de faciliter la vie des développeurs informatiques.
- un « framework » est la caisse à outils du développeur : ce dernier vient piocher les éléments dont il a besoin pour créer son support (application, site...).
- Un framework sera toujours associé à un langage de programmation (Java, PHP, JS,...).

BASE JAVASCRIPT

LOGIQUE DE JS

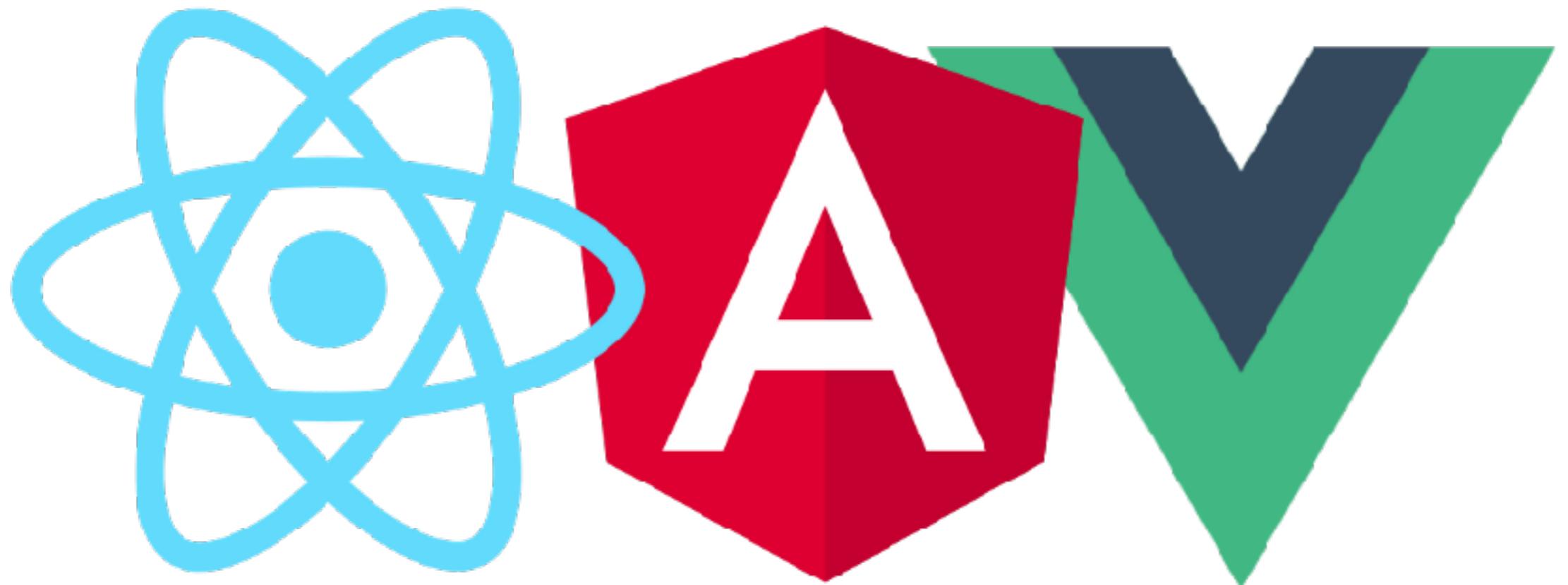
- Pourquoi utiliser Les frameworks
- gain de temps et d'efficacité pour le développeur, il lui sert de structure de base à chaque nouveau projet. Ainsi, ce dernier n'est pas contraint de développer le support de A à Z, il peut alors se concentrer sur la réalisation de tâches spécifiques.
- lorsqu'un développeur intègre une nouvelle équipe qui travaille sur un framework déjà établi et qu'il connaît, ce dernier prendra ses repères très rapidement, il sera donc d'autant plus efficace.
- Sa structure permet une maintenance simplifiée
- Les failles de sécurité seront déjà pré-protégées

BASE JAVASCRIPT

LOGIQUE DE JS

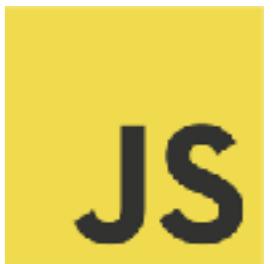


- Les frameworks



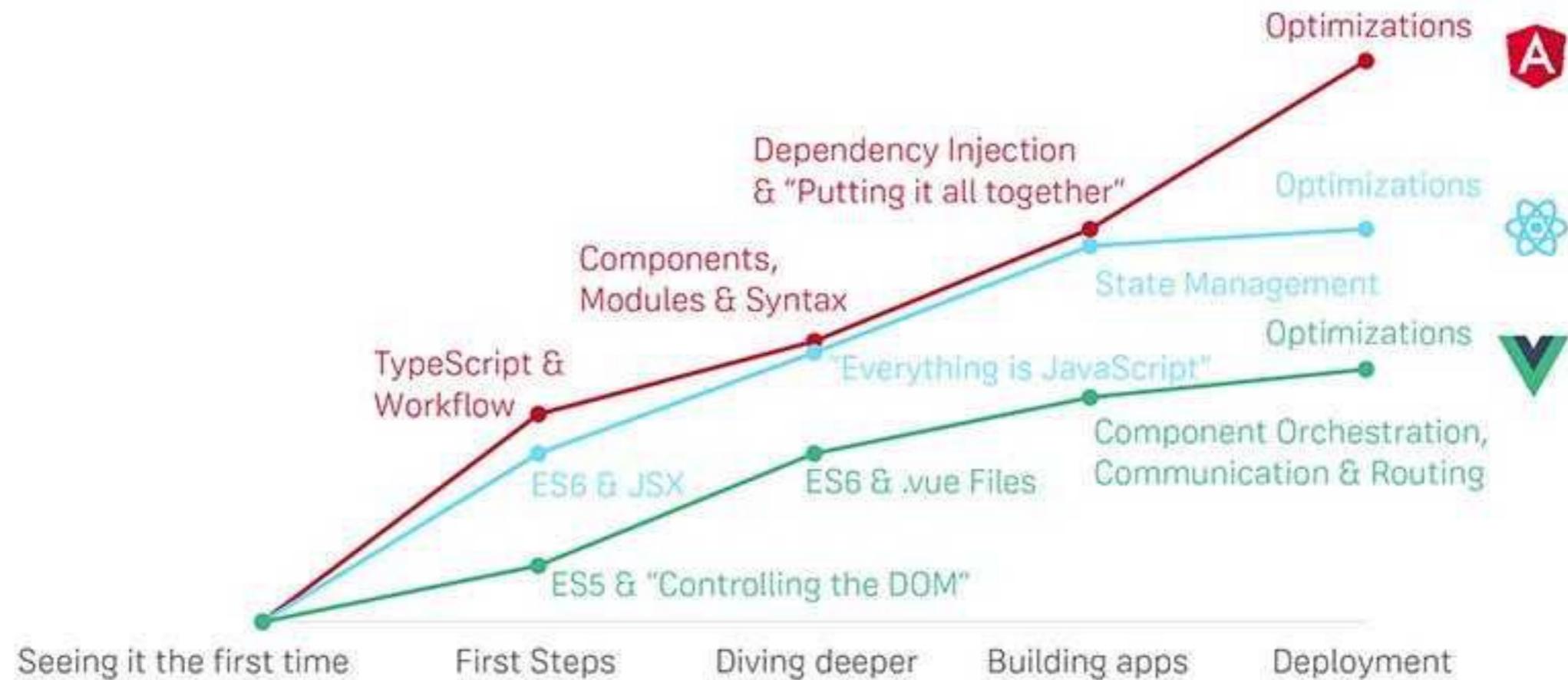
BASE JAVASCRIPT

LOGIQUE DE JS



- Les frameworks

(Possible) Learning Curve

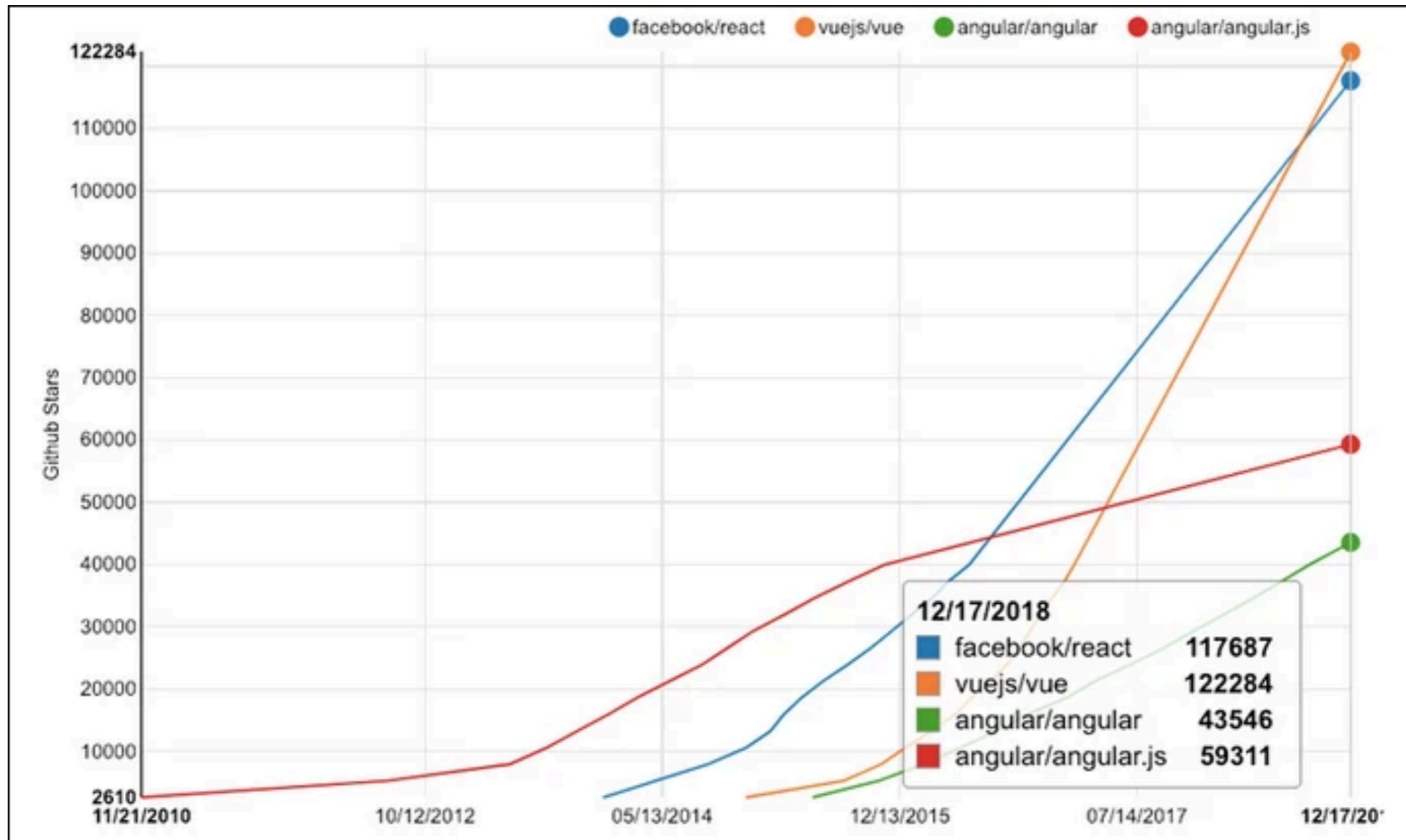


BASE JAVASCRIPT

LOGIQUE DE JS

JS

- Les frameworks



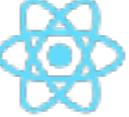
BASE JAVASCRIPT

LOGIQUE DE JS

- Les frameworks (Angular) 
- Publié en 2010 par Google
- Basé sur le langage TypeScript
- De 2010 à 2016 le framework s'appelait AngularJS.
- 2016 Google rebaptise le framework en Angular
(Cette version prend pour base Angular 2)
(L'architecture est différente mais mieux structurée.)

BASE JAVASCRIPT

LOGIQUE DE JS

- Les frameworks (React)
- Publié en 2013 par Facebook
- Il est principalement utilisé dans les sites Web à fort trafic.
- Il a été développé lorsque les publicités Facebook ont commencé à gagner en trafic et ont rencontré des problèmes de codage et de maintenance
- Ils ont été résolus avec la sortie de cette bibliothèque de Javascript.
- Il est très dynamique et offre un excellent support dans la création d'interfaces utilisateur interactives.
- Whatsapp, Instagram, Paypal, Glassdoor, BBC sont quelques-unes des entreprises populaires utilisant React.

BASE JAVASCRIPT

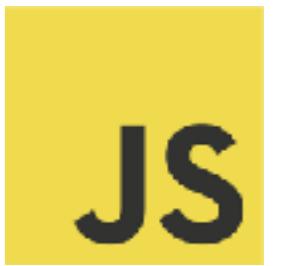
LOGIQUE DE JS



- Les frameworks (Vue)
- Publié en 2014 par Evan You (Un ancien ingénieur de Google)
- Un framework Js Progressif
- Accessible au débutants Productif pour les experts.
- Proto, widget petits projets Grandes applis complexes
- Il gagne en popularité
- Utilisé par Gitlab, Alibaba etc ...

BASE JAVASCRIPT

LOGIQUE DE JS

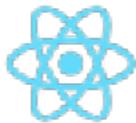


- Les frameworks (communauté) 
- Il a été introduit par le célèbre leader du marché Google.
- Utilisé par l'application Google Adwords (Le must pour le référencement sur le web).
- La longévité du framework assure une très large communauté de développeurs en plus du support des équipes de Google.

JS

BASE JAVASCRIPT

LOGIQUE DE JS

- Les frameworks (communauté) 
- Il a été introduit par Facebook
- Utilisé sur les applis de Facebook
- Communauté croissante appuyé par le support des équipes de Facebook.

JS

BASE JAVASCRIPT

LOGIQUE DE JS

- Les frameworks (communauté) 
- Contrairement à Angular ou React Vue n'est pas soutenu par une grande entreprise.
- Cela aurait du entraîné une baisse de popularité mais dans les fait c'est tout le contraire (surtout dans le monde de l'open source)

BASE JAVASCRIPT

TYPES DE VALEURS

- Dans tout langage de programmation on manipule des données.
- On manipule du texte, des nombres, des calculs, des dates...
- On est à l'ère du Big Data, l'avancée du numérique nous plonge dans un océan de données .
- En revanche pour un ordinateur il va stocker les données sous forme binaire (10000010001010101110111000111) il fonctionne comme ça.
- Pour nous êtres humains c'est illisible et peu pratique.
- Du coup en JS (et autre langage de code) on a créé des types de valeur pour stocker des données qui soient plus parlantes.

BASE JAVASCRIPT

TYPES DE VALEURS

- On retrouve 6 types de valeurs en JS :
 - Nombre : number
 - Chaines de caractère : string
 - Booléen (Vrai / Faux) : boolean
 - Objets : object
 - Fonction: function
 - ???: Undefined



BASE JAVASCRIPT

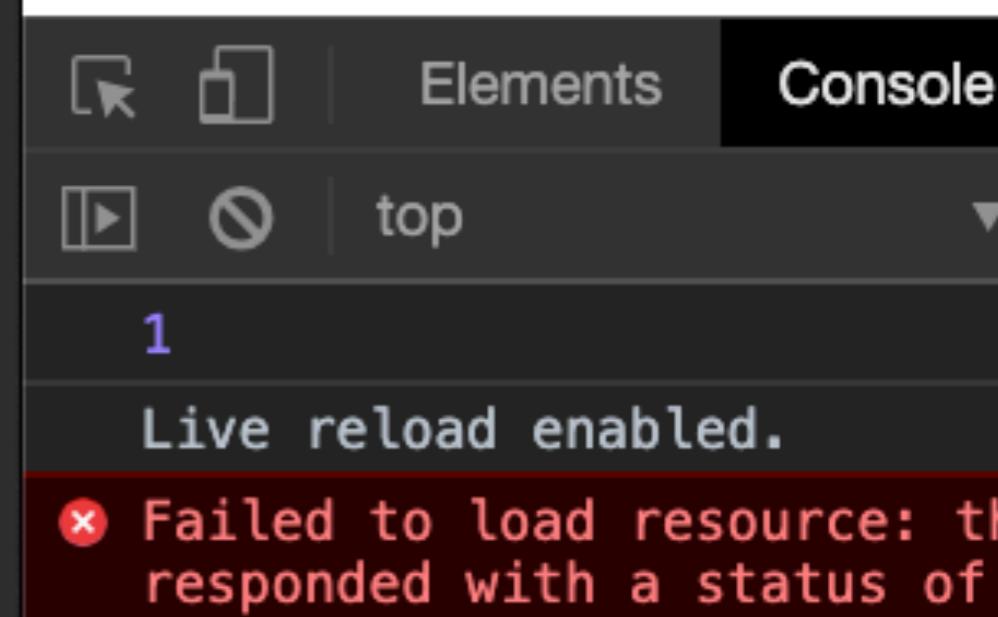
ASSIGNER UNE VALEUR

- Pour assigner des valeur on commence par créer une variable
- Et dans notre variable on peut stocker quelquechose (Valeur)
- Le = permet d'assigner une valeur à notre variable
- Le ; pour finir une instruction

BASE JAVASCRIPT

ASSIGNER UNE VALEUR

```
// Afficher un message dans la console  
// console.log("Coucou");  
  
// Assigner une valeur  
  
var num = 1;  
console.log(num);
```



BASE JAVASCRIPT

ASSIGNER UNE VALEUR

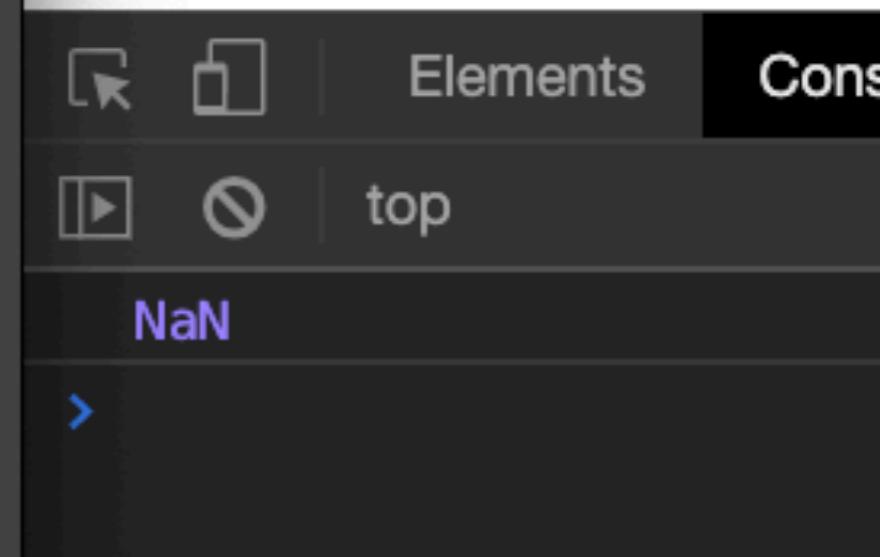
```
1 // Code JavaScript ici  
2  
3 // Afficher un message dans la console  
4 // console.log("Coucou");  
5  
6 // Assigner une valeur  
7  
8 var num = 1;  
9 var chiffre;  
10 console.log(chiffre);
```



BASE JAVASCRIPT

ASSIGNER UNE VALEUR

```
1 // Code JavaScript ici  
2  
3 // Afficher un message dans la console  
4 // console.log("Coucou");  
5  
6 // Assigner une valeur  
7  
8 var num = 1;  
9 var chiffre;  
10 console.log(chiffre + num);
```



BASE JAVASCRIPT

ASSIGNER UNE VALEUR

```
// Code JavaScript ici  
  
// Afficher un message dans la console  
// console.log("Coucou");  
  
// Assigner une valeur  
  
var num = 1;  
var chiffre = 0;  
console.log(chiffre + num);
```



@
ASSIGNER UNE VALEUR

```
2  
3 // Afficher un message dans la console  
4 // console.log("Coucou");  
5  
6 // Assigner une valeur  
7  
8 var num = 1;  
9 var chiffre = 0;  
10 console.log(chiffre + num);  
11 chiffre=30;  
12 console.log(chiffre + num);  
13
```



BASE JAVASCRIPT

CAMEL CASE NOTATION

- Complexe et simple à la fois
- Donner un nom à nom variables.
- Jusque là notre programme est ultra simple mais dans des appli beaucoup beaucoup plus complexes, on est amené à utiliser énormément de variables et on ne sait plus trop comment les nommer.
- On a besoin d'une convention de nommage de nos variable de manière à s'y retrouver facilement et efficacement.
- En JS on utilise un Camel Case légèrement modifié.



BASE JAVASCRIPT

CAMEL CASE NOTATION

- Camel Case
- L'idée est d'utiliser des noms pour nos variable qui soient à la fois COURT, ÉVOCATEUR, PRÉCIS.
- Cela permet une meilleur visibilité du code, qui le rend plus facile à maintenir.

JS

BASE JAVASCRIPT

CAMEL CASE NOTATION

main.js > ...

```
// Code JavaScript ici
```

```
var motdepasse = 1234567890;
var mot_de_passe = 1234567890;
var MotDePasse = 1234567890;
var motDePasse = 1234567890;
var motDePasseSecret = 1234567890;
var password = 12345;
```



JS

BASE JAVASCRIPT

EXO: CALCUL

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following code:

```
// Code JavaScript ici
console.log(20+33);
```

Browser: The browser's developer tools are open, specifically the Console tab. It shows the output of the console.log statement: **53**. The output is timestamped as **main.js:6**.

JS

BASE JAVASCRIPT

EXO: CALCUL

The screenshot shows a development environment with the following components:

- Code Editor:** A dark-themed code editor with tabs for "index.html" and "main.js". The "main.js" tab contains the following code:

```
1 // Code JavaScript ici
2
3
4
5
6 // console.log(20+33);
7 console.log(20*3987948798743);
```
- Browser:** A browser window showing the URL <http://127.0.0.1:5>. The page content is blank.
- Developer Tools:** An open developer tools panel with the "Console" tab selected. The console output shows:

```
79758975974860
main.js:7
```

BASE JAVASCRIPT

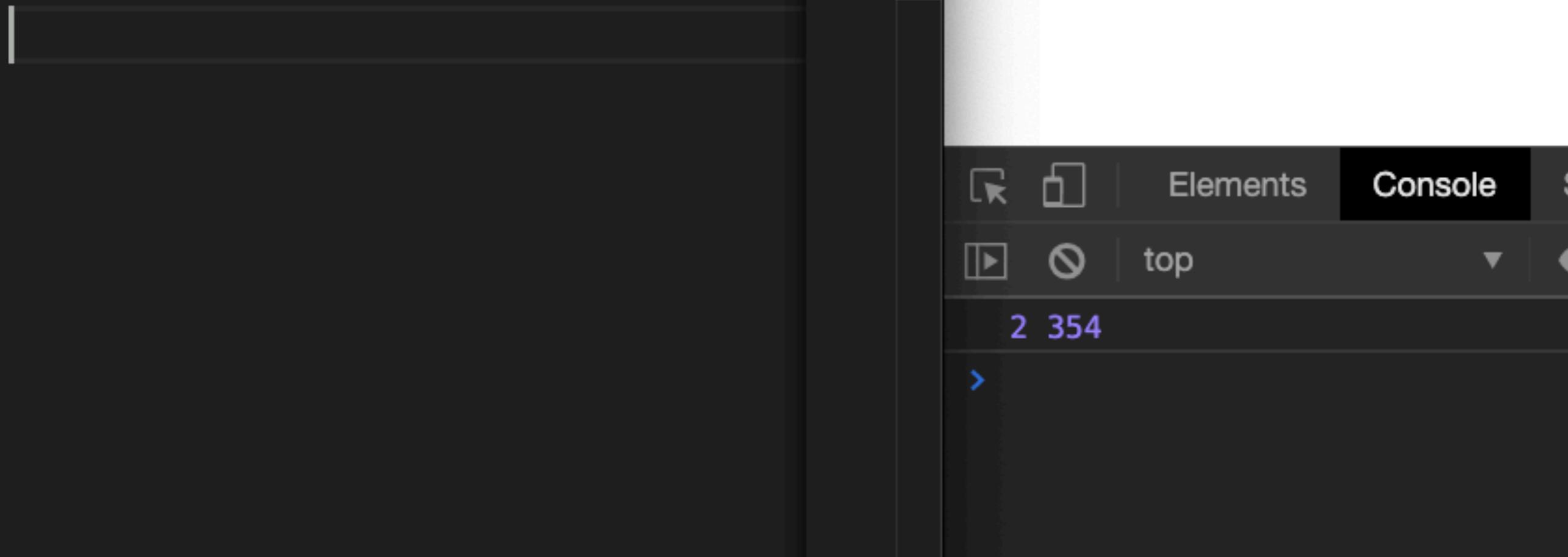
EXO: CALCUL

- On a plusieurs opérateurs pour faire des calculs
- + Additionner
- - Soustraire
- * Multiplier
- / Diviser

BASE JAVASCRIPT

NOMBRE DÉCIMAUX

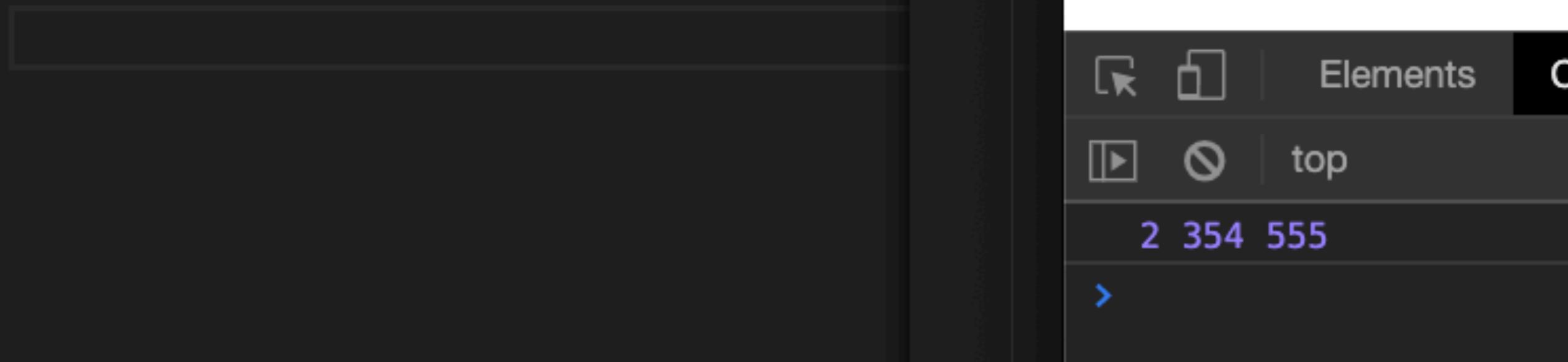
```
2  
3  
4  
5 console.log(2,342 + 12);  
6  
7  
8  
9  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```



BASE JAVASCRIPT

NOMBRE DÉCIMAUX

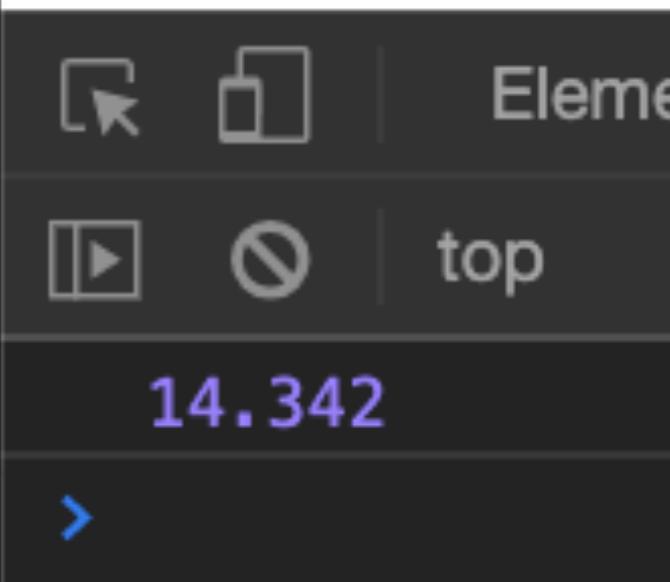
```
5 console.log(2,342 + 12,555);
```



BASE JAVASCRIPT

NOMBRE DÉCIMAUX

```
// console.log(2,342 + 12,555);
console.log(2.342 + 12);
```



BASE JAVASCRIPT

NOMBRE DÉCIMAUX

- Le « BUG » de JS avec les calculs décimaux.
- À la base quand JS a été créé les capacités mémoire des machines étaient inférieures comparé à aujourd'hui.
- Pour gérer les décimaux les concepteurs du langage ont fait le choix d'un certain algorithme de calcul
(Pour résoudre les Pb de mémoire)
- JS utilise une précision de virgule Flottante
(un système d'arrondi particulier)
- Pour faire du Jeux vidéo, des Apps Web ou Mobile, des animations cela ne pose pas de soucis.
- Pas pertinent si vous travaillez sur une app scientifique avec des données hyper précises.

JS

BASE JAVASCRIPT

NOMBRE DÉCIMAUX

The screenshot shows a code editor on the left and a browser window on the right. The code editor has tabs for 'index.html' and 'main.js'. The 'main.js' tab contains the following code:

```
// Code JavaScript ici
// console.log(2,342 + 12,555);
// console.log(2.342 + 12);
console.log(0.1 + 0.2);
```

The browser window shows the developer tools with the 'Console' tab selected. The output in the console is:

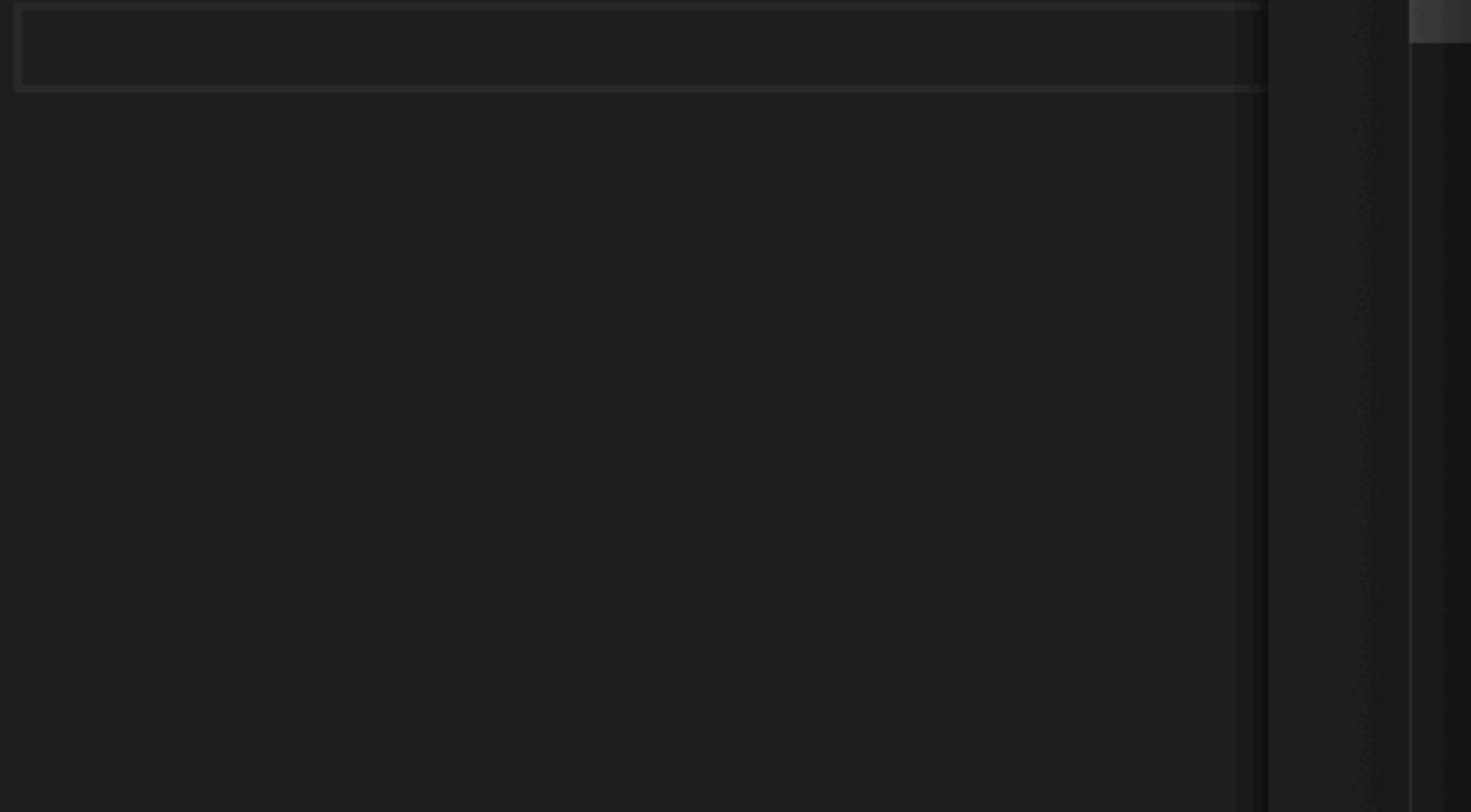
```
0.3000000000000004
```

main.js:7

BASE JAVASCRIPT

PRIORITÉS DE CALCUL

```
console.log(1 + 2 * 5);
```



BASE JAVASCRIPT

PRIORITÉS DE CALCUL

```
// console.log(1 + 2 * 5);
console.log((1 + 2) * 5);
```

The screenshot shows a browser's developer tools open to the 'Elements' tab, specifically the 'Console' section. Two lines of code are being executed:

```
// console.log(1 + 2 * 5);
console.log((1 + 2) * 5);
```

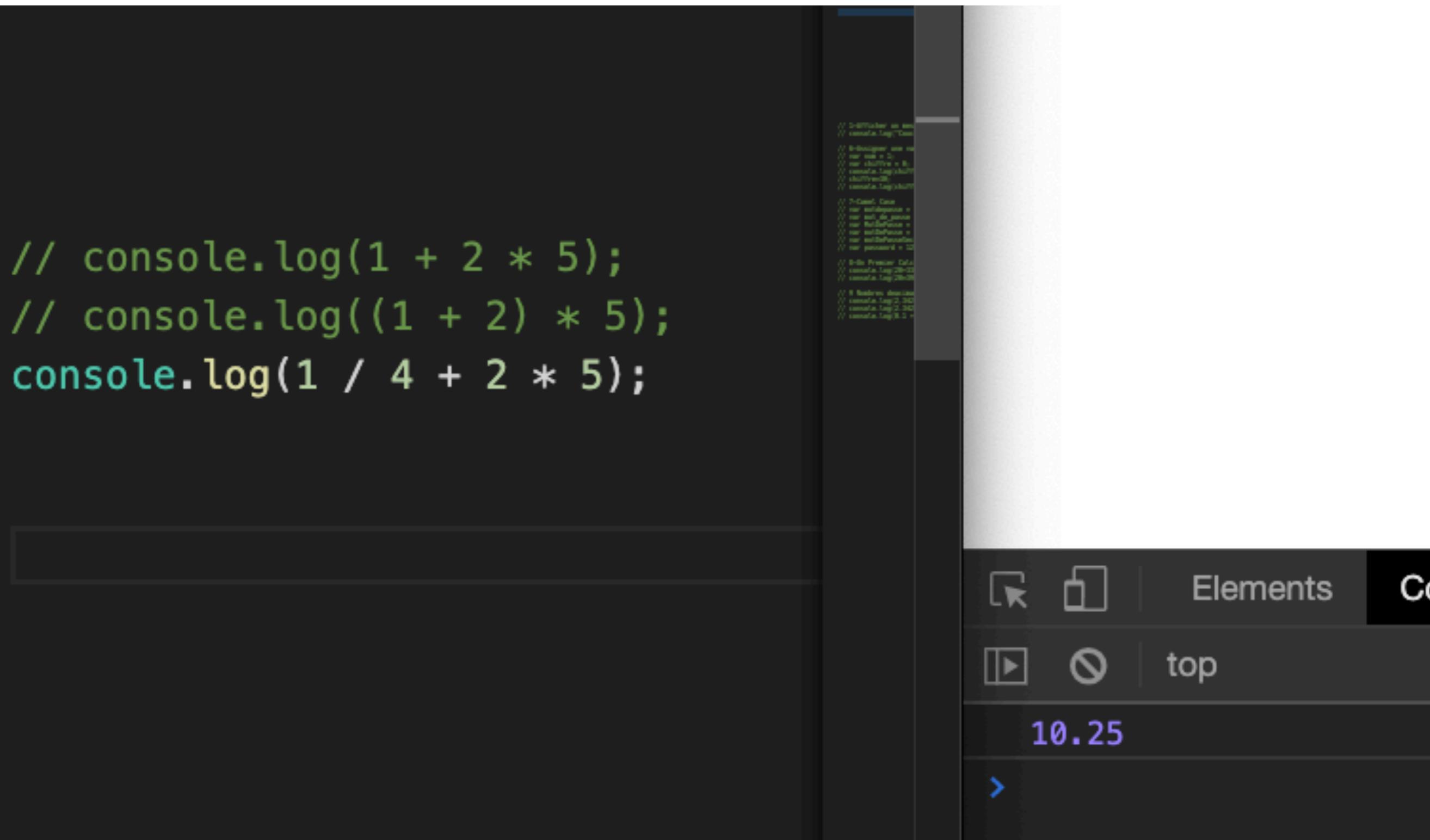
The first line, `// console.log(1 + 2 * 5);`, is shown with its execution path expanded. It starts with a call to `evalScript`, which then calls `eval`. Inside the `eval` frame, the expression `1 + 2 * 5` is evaluated. This expression goes through several stages of evaluation, including `parse`, `createTempObject`, `getTempObject`, `callEvalFunction`, `callEvalFunction`, `callEvalFunction`, and finally `processEval` where the result `11` is produced.

The second line, `console.log((1 + 2) * 5);`, is also shown with its execution path expanded. It follows a similar pattern, starting with `evalScript` and then `eval`. Inside the `eval` frame, the expression `(1 + 2) * 5` is evaluated. This expression goes through stages of `parse`, `createTempObject`, `getTempObject`, `callEvalFunction`, and `processEval` where the result `15` is produced.

BASE JAVASCRIPT

PRIORITÉS DE CALCUL

```
// console.log(1 + 2 * 5);
// console.log((1 + 2) * 5);
console.log(1 / 4 + 2 * 5);
```





JS

BASE JAVASCRIPT

EXO: OPÉRATION

- On a vu Presque tous les opérateur utiles pour faire des opérations.
- Sauf 1.
- Modulo

JS

BASE JAVASCRIPT

EXO: OPÉRATION

The screenshot shows a development environment with the following components:

- Code Editor:** A dark-themed code editor window titled "main.js — base". It contains the following JavaScript code:

```
1 // Code JavaScript ici
2
3 console.log(10/3);
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```
- Browser:** A browser window showing the URL <http://127.0.0.1:5>. The page content is blank.
- Developer Tools:** An open developer tools panel with the "Console" tab selected. The console output shows the result of the `console.log(10/3);` statement:

```
3.3333333333333335
>
```

JS

BASE JAVASCRIPT

EXO: OPÉRATION

The screenshot shows a development environment with the following components:

- Code Editor:** A dark-themed code editor window titled "main.js — base". It contains the following JavaScript code:

```
1 // Code JavaScript ici
2
3 console.log(10/3);
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```
- Browser:** A browser window showing the URL <http://127.0.0.1:5>. The page content is blank.
- Developer Tools:** An open developer tools panel with the "Console" tab selected. The console output shows the result of the `console.log(10/3);` statement:

```
3.3333333333333335
>
```

BASE JAVASCRIPT

EXO: OPÉRATION

Voici une division posée :

$$\begin{array}{r} 35 \\ - 24 \\ \hline 11 \end{array} \qquad \begin{array}{r} 12 \\ \hline 2 \end{array}$$

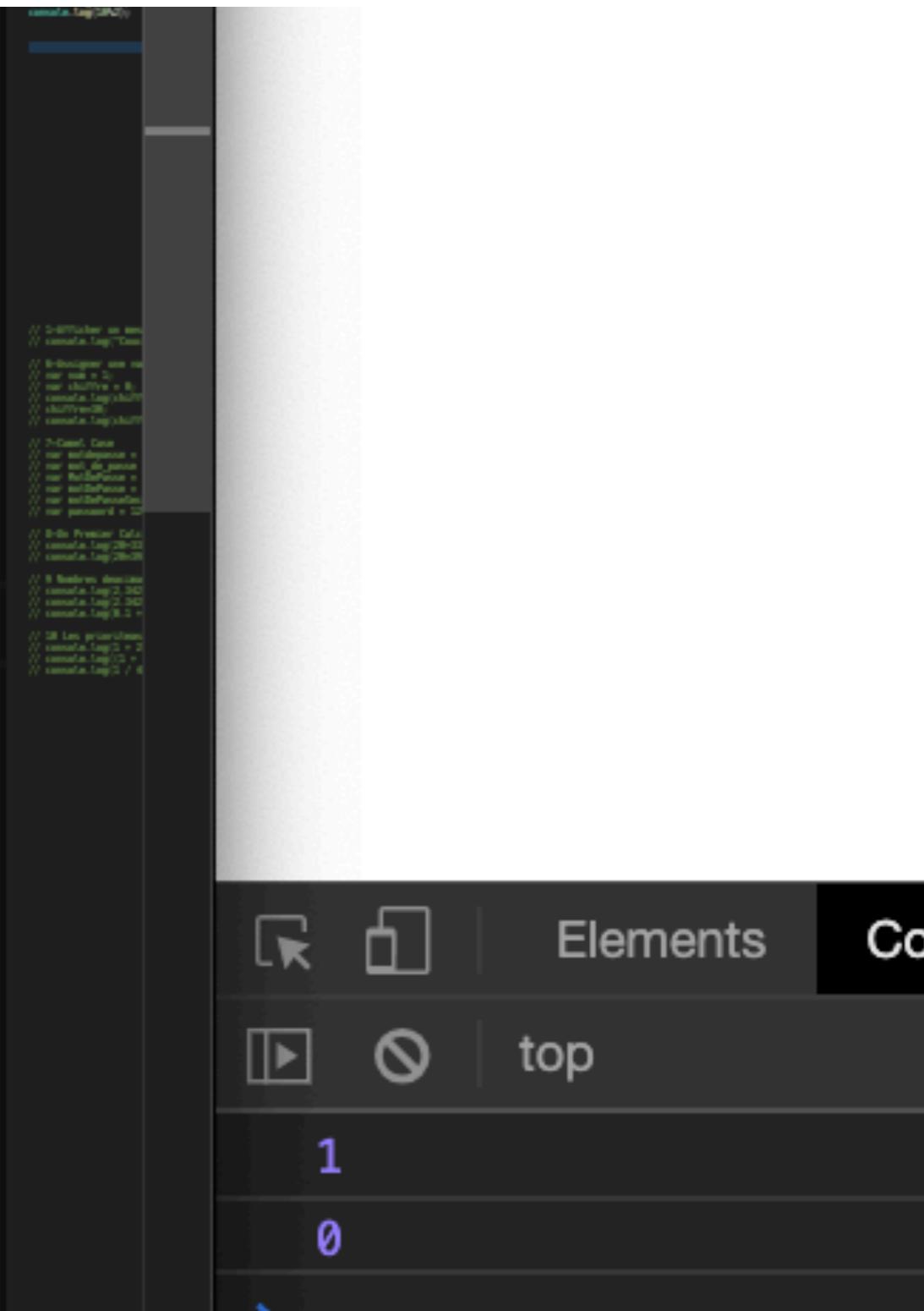
La division euclidienne correspondante va s'écrire :

$$35 = 12 \times 2 + 11$$

BASE JAVASCRIPT

EXO: OPÉRATION

```
1 // code JavaScript ici  
2  
3 // console.log(10/3);  
4 console.log(10%3);  
5 console.log(10%2);  
6  
7 |  
8  
9  
10  
11  
12  
13  
14  
15  
16
```



BASE JAVASCRIPT

INCRÉMENTATION



BASE JAVASCRIPT

INCREMENTATION

JS

```
var num = 0; var num: number  
console.log(num+1);
```

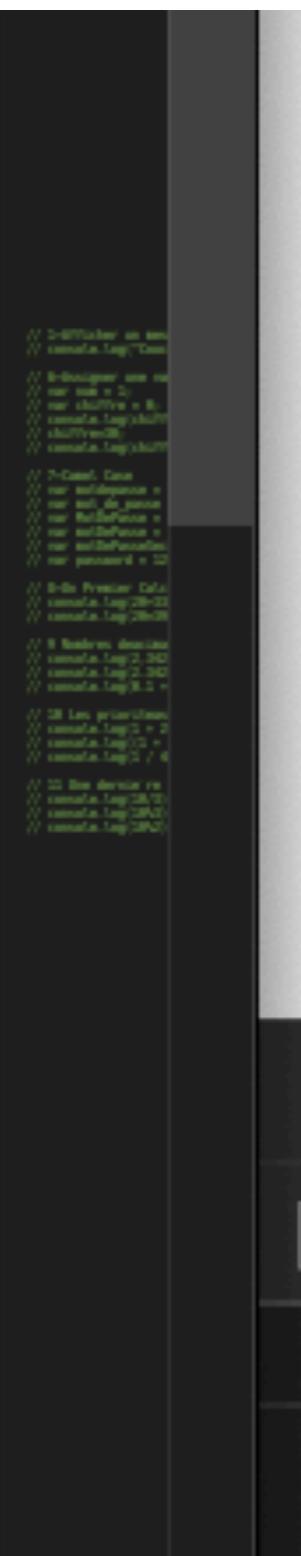
```
    3-Offensive or In  
    console.Log("Cone  
  
    3-Scouting our  
    our role = 3)  
    our.chillTime = 8;  
    console.Log(chillT  
    chillTime);  
    console.Log(chillT  
  
    3-Clean, Clean  
    our.missionStatus =  
    our.role = 3; //just  
    our.RollPower =  
    our.rollPower =  
    our.rollPower =  
    our.rollPower = 12  
  
    3-The Previews Data  
    console.Log("28-03")  
    console.Log("28-03")  
  
    3-Ready to cleanData  
    console.Log("3-2803")  
    console.Log("3-2803")  
    console.Log("3-2803")  
  
    38-Low priority Clean  
    console.Log("3-2803")  
    console.Log("3-2803")  
    console.Log("3-2803")  
  
    33-High priority Clean  
    console.Log("3-2803")  
    console.Log("3-2803")  
    console.Log("3-2803")
```

1

BASE JAVASCRIPT

INCRÉMENTATION

```
2  
3  
4  
5 var num = 0;  
6 num = num+1;  
7 console.log(num);  
8  
9  
10  
11  
12  
13  
14  
15  
16
```



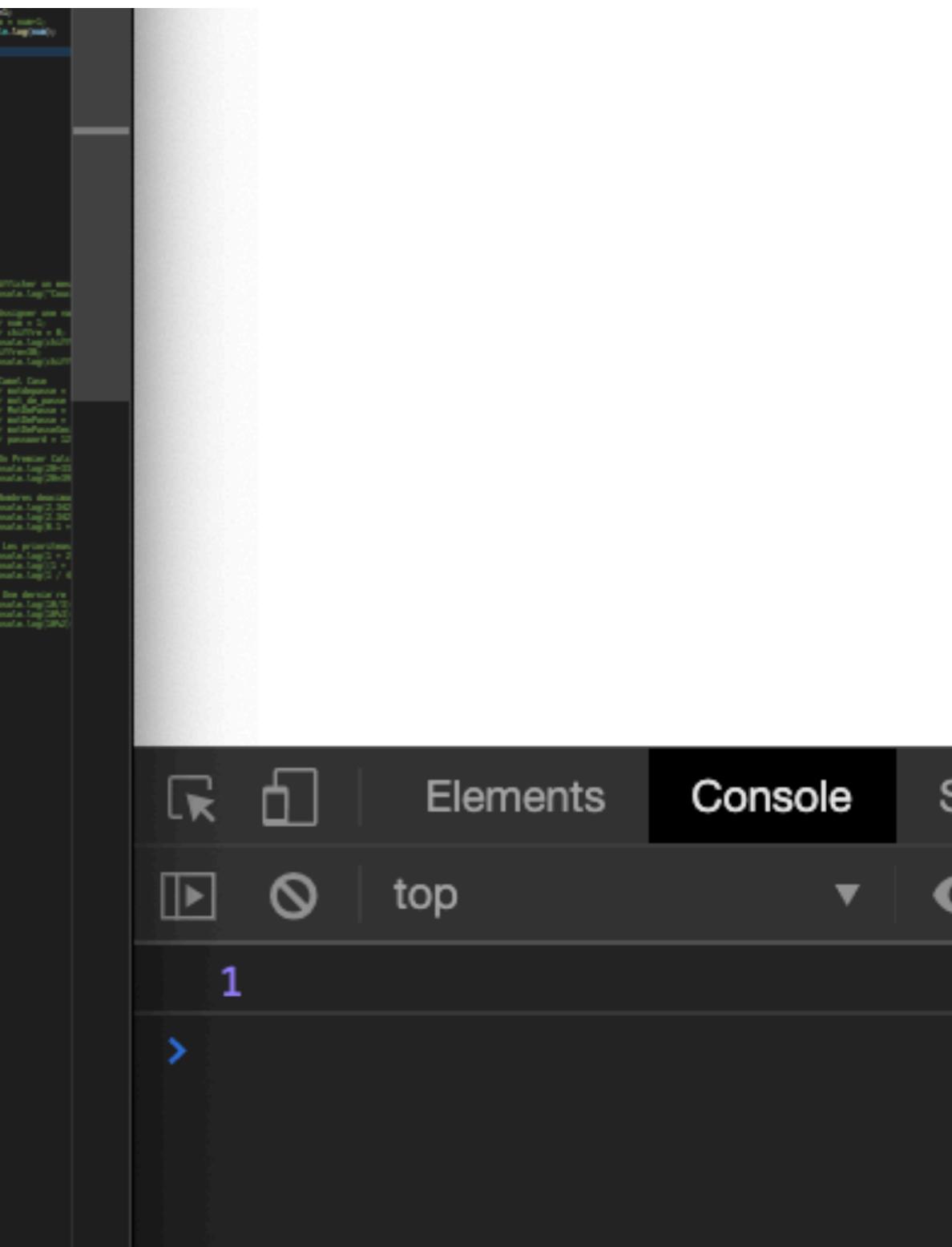
```
Elements  
top  
1  
>
```

BASE JAVASCRIPT

INCREMENTATION

JS

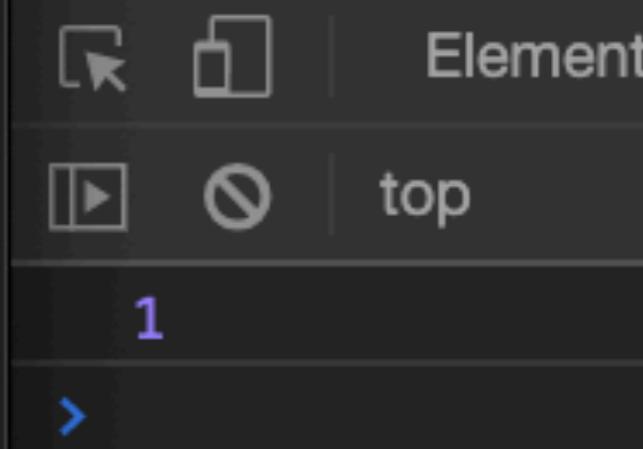
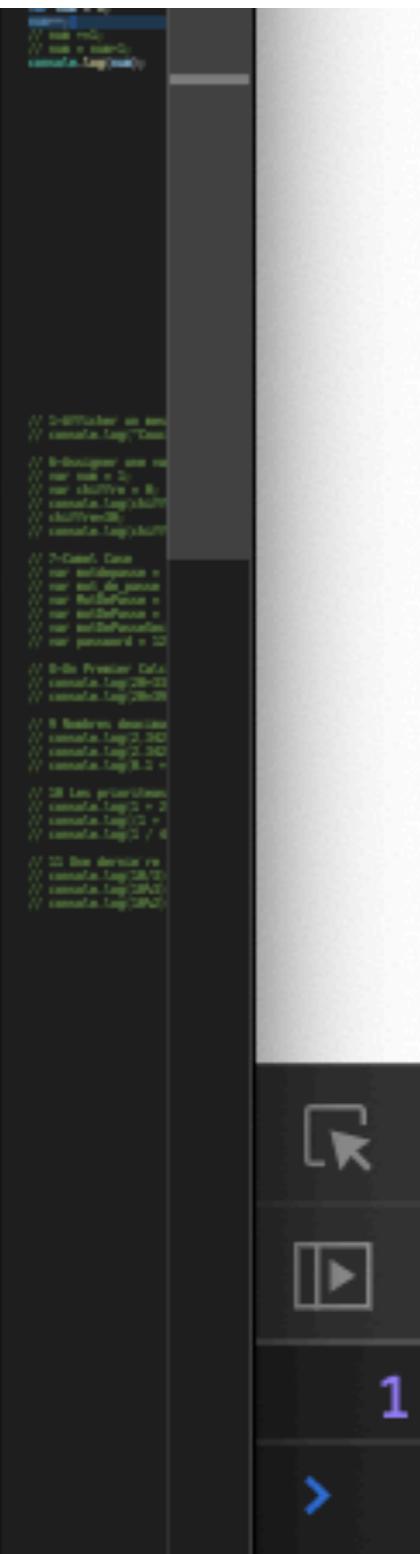
```
2
3
4
5     var num = 0;
6     num +=1;
7     // num = num+1;
8     console.log(num);
9
10
11
12
13
14
15
16
17
```



BASE JAVASCRIPT

INCRÉMENTATION

```
1 // code JavaScript ici
2
3
4
5 var num = 0;
6 num++;
7 // num +=1;
8 // num = num+1;
9 console.log(num);
10
11
12
13
14
15
```

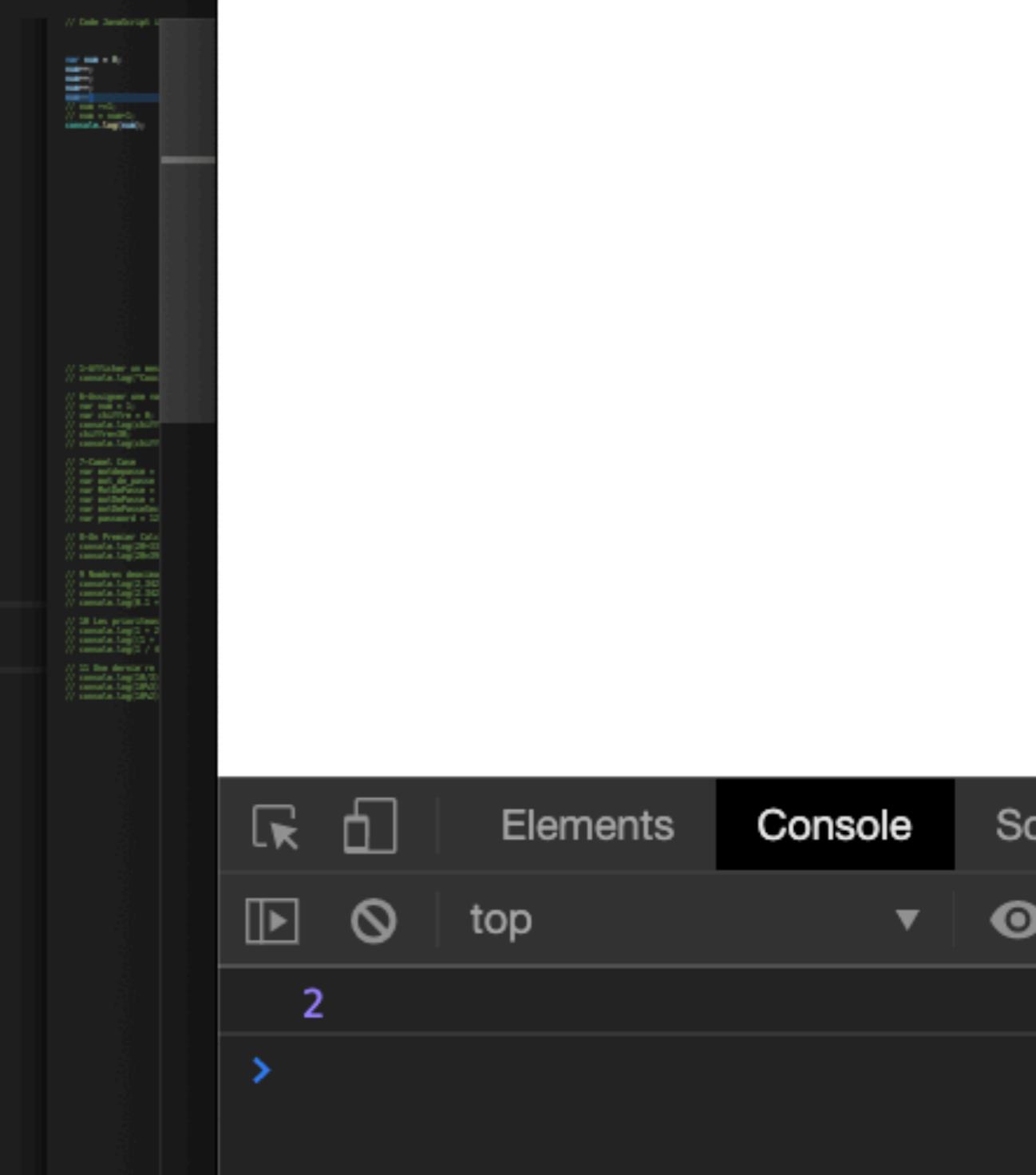


BASE JAVASCRIPT

INCRÉMENTATION

JS main.js > ...

```
1 // Code JavaScript ici
2
3
4
5 var num = 0;
6 num++;
7 num++;
8 num++;
9 num--;
10 // num +=1;
11 // num = num+1;
12 console.log(num);
13
14
15
16
```





BASE JAVASCRIPT

ASSIGNEMENT COMPOSÉ

- 2 en 1
- On donne une valeur et on fait une opération en même temps.
- Concrètement cela peut permettre d'effectuer des calculs avec un code plus réduit, plus lisible.
- On peut effectuer des calculs sans connaître la valeur de départ.
- Ex: Une remise de 20% sur tous les prix des produits d'une boutique.

BASE JAVASCRIPT

ASSIGNEMENT COMPOSÉ

JS

```
2
3
4
5
6
7 var num=10;
8 num +=5;
9 console.log(num);
10
11
12
13
14
15
16
17
```

BASE JAVASCRIPT

ASSIGNEMENT COMPOSÉ

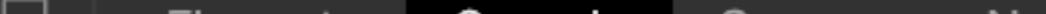
JS

III.js > ...

// Code JavaScript ici

```
var num=10;  
num /=5;  
// num +=5;  
console.log(num);
```





2

7

BASE JAVASCRIPT

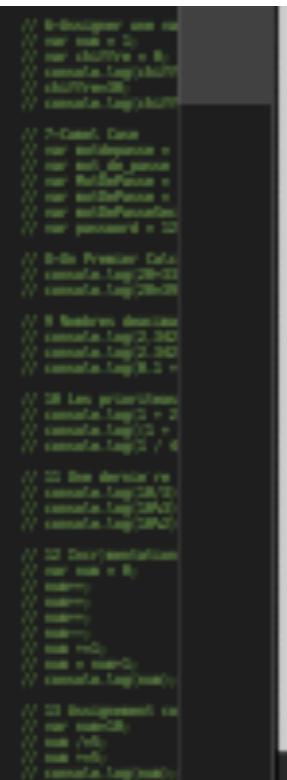
CHAÎNES DE CARACTÈRES

- Pour gérer du texte on utilise les guillemets " (double quote)
- Beaucoup de Dev JS utilisent le guillemet simple ' (quote)
- Il vaut mieux utiliser les double quote " pour être raccord avec d'autres fichier pour stocker des objets (le format JSON)
JavaScript Object Notation
- Bug quand on a déjà des guillemets " ou des apostrophe ' dans un texte.
- On utilise l'anti slash \ pour échapper le texte.

BASE JAVASCRIPT

CHAÎNES DE CARACTÈRES

```
var msg = "Bonjourno";
var slt = 'Salut !!';
console.log(msg);
console.log("msg");
console.log(slt);
```

A screenshot of a browser's developer tools console. The code in the editor window is:

```
var msg = "Bonjourno";
var slt = 'Salut !!';
console.log(msg);
console.log("msg");
console.log(slt);
```

The console output shows:

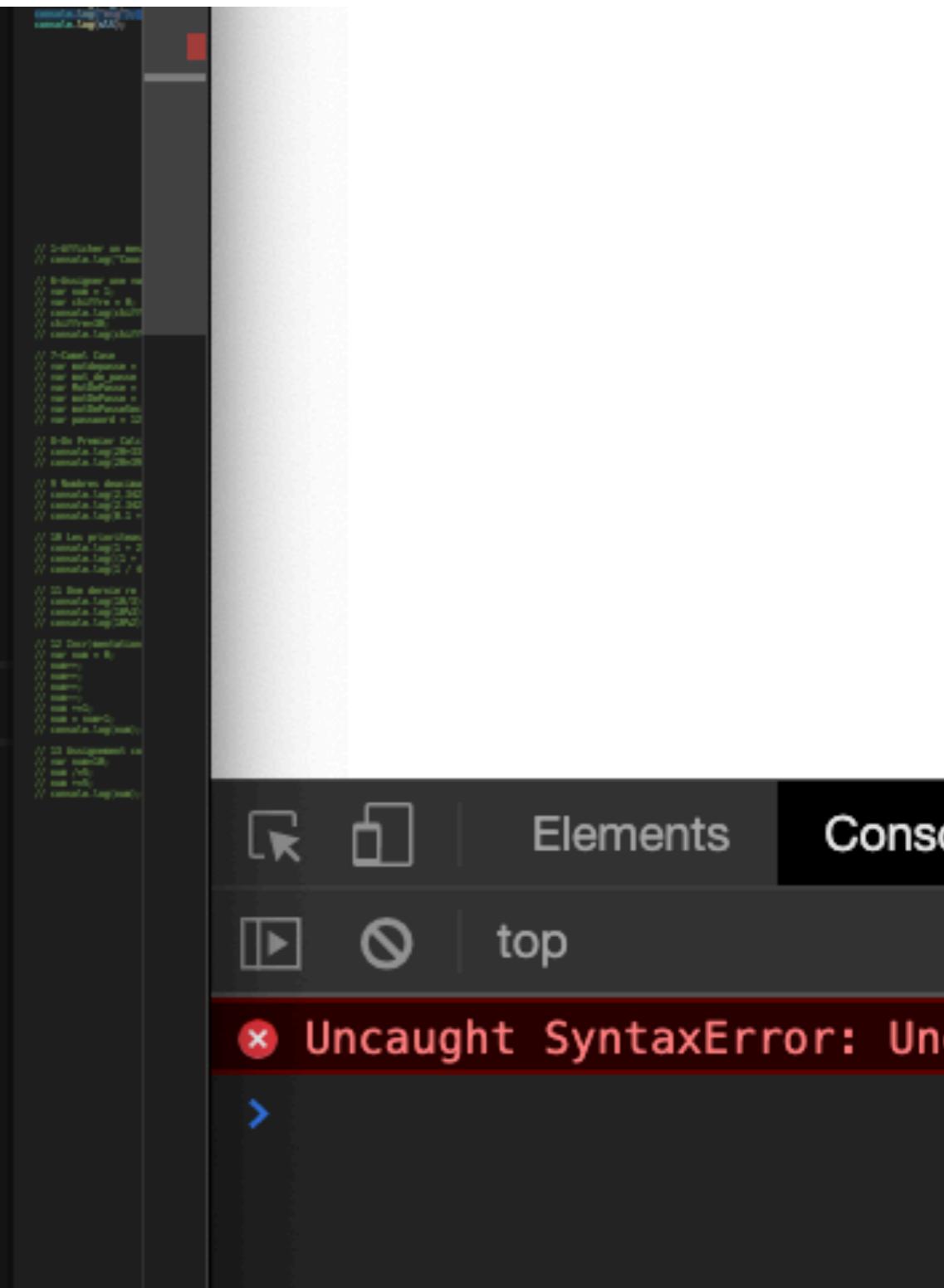
```
Bonjourno
msg
Salut !!
```

There are also many other log entries from the browser's internal processes.

BASE JAVASCRIPT

CHAÎNES DE CARACTÈRES

```
2  
3  
4  
5  
6  
7 var msg = "Bonjourno";  
8 var slt = 'Salut j'aime ceci !!!';  
9 console.log(msg);  
10 console.log("msg");  
11 console.log(slt);  
12  
13  
14  
15  
16  
17
```



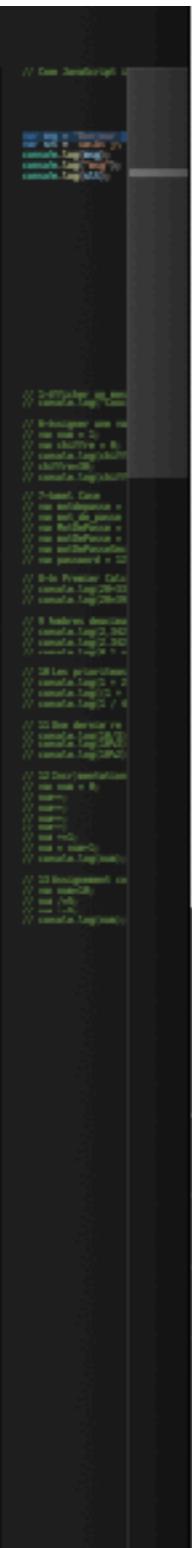
BASE JAVASCRIPT

CHAÎNES DE CARACTÈRES

ain.js > [msg]

```
// Code JavaScript ici
```

```
var msg = "Bonjour \"MR Gingle\"";  
var slt = 'Salut j\'aime ceci !!';  
var console: Console  
console.log("msg");  
console.log(slt);
```





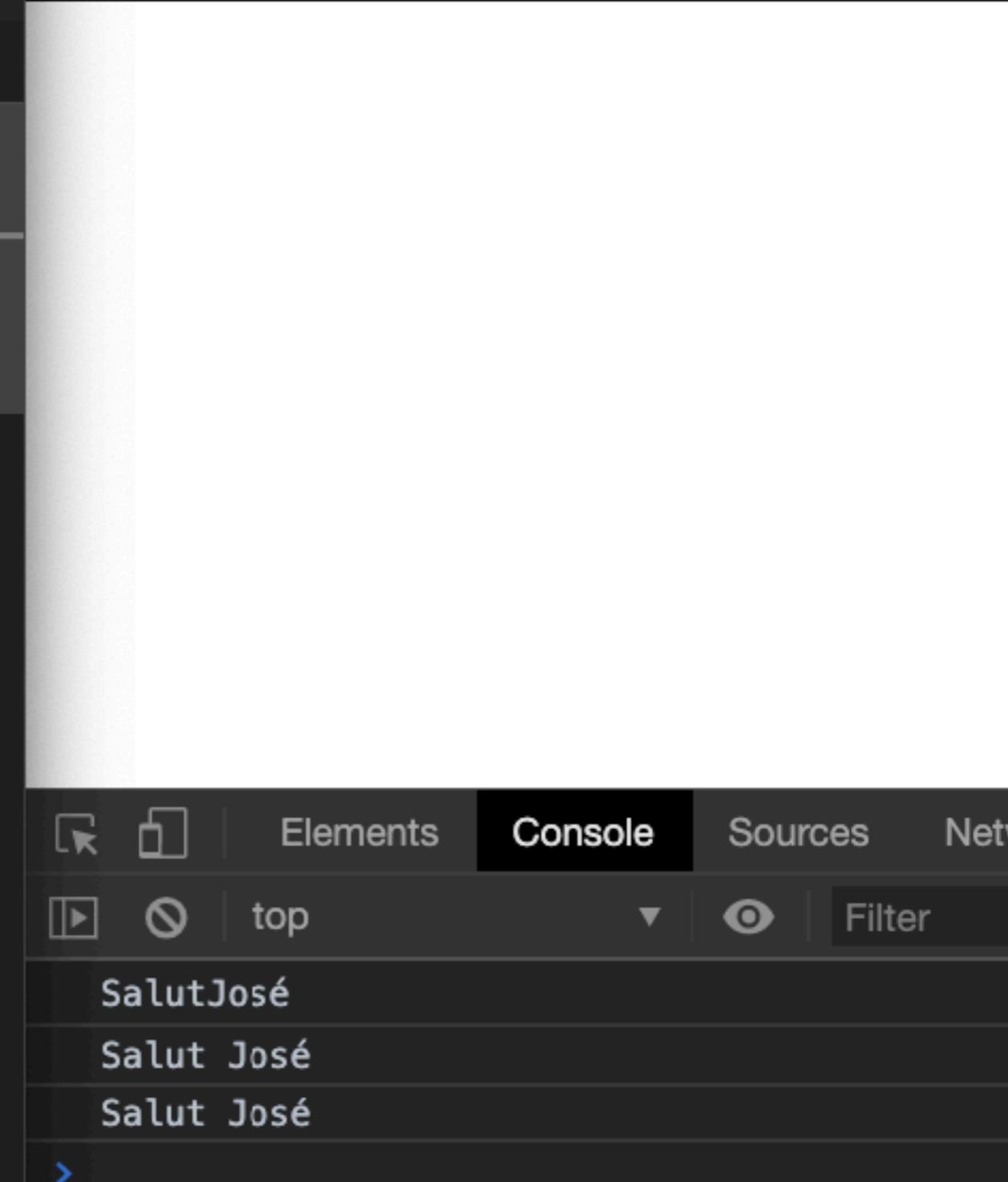
BASE JAVASCRIPT

CONCATÉNER DES CHAÎNES DE CARACTÈRES

- On peut additionner du texte.
- Avec des chaines de caractères on peut utiliser + pour les concaténer.

BASE JAVASCRIPT

CONCATÉNER DES CHAÎNES DE CARACTÈRES



index.html JS main.js × ⌂ ... < > C ⌂ | ⌂ http://127.0.0.1:5 | 📁 🗃 🔍 | 1 5

S main.js > ...

```
1 // Code JavaScript ici
2
3
4
5 var msg = "Salut";
6 var name = "José";
7 var space = " ";
8 console.log(msg + name);
9 console.log(msg + space + name);
10 console.log(msg + " " + name);
```

11
12
13
14 SalutJosé
15 Salut José
16 Salut José
17 >

BASE JAVASCRIPT

CONCATÉNER DES CHAÎNES DE CARACTÈRES

```
1 // Code JavaScript ici  
2  
3  
4  
5 var salut = "Salut";  
6 var name = "José";  
7 salut += " " +name;  
8 console.log(salut);  
9  
10  
11  
12  
13  
14  
15  
16
```

The screenshot shows a browser's developer tools open to the 'Console' tab. The code editor on the left contains the provided JavaScript code. The console output on the right shows the result of the `console.log` statement: 'Salut José'. The browser interface includes tabs for 'Elements' and 'Console'.

```
// Code JavaScript ici  
var salut = "Salut";  
var name = "José";  
salut += " " +name;  
console.log(salut);  
  
Salut José
```

BASE JAVASCRIPT

EXO: GÉNÉRER UNE PHRASE

- Créer 2 variables
 - Le nom (vous assignez un nom de votre choix)
 - L'âge (vous assignez un âge de votre choix)
- Créer une variable phrase de bienvenue :

« Bonjour Jean-Jacques (le nom) tu as 57(l'âge) ans aujourd'hui,
c'est la fiesta ! »
- Console log de phrase

JS

BASE JAVASCRIPT

EXO: GÉNÉRER UNE PHRASE

The screenshot shows a code editor with a dark theme and a browser window side-by-side.

Code Editor (main.js — base):

```
index.html    JS main.js  ×  
JS main.js > [o] phrase  
1 // Code JavaScript ici  
2  
3  
4 var name = "Sarah";  
5 var age = 22;  
6 var phrase = "Bonjour "  
7     + name  
8     + " tu as "  
9     + age  
10    + " aujourd'hui, c'est la fiesta !";  
11  
12 console.log(phrase);  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22
```

Browser:

- Address bar: `http://127.0.0.1:5`
- Console tab is selected in the developer tools.
- Console output:
Bonjou Sarah tu as 22 aujourd'hui, c'est la fiest!
!

BASE JAVASCRIPT



JS

TROUVER LA TAILLE D'UNE CHAÎNE DE CARACTÈRES

- Connaitre le nombre de lettre dans un mot
 - On va utiliser length
- Accéder à un caractère dans une chaîne de caractères.
 - On va utiliser les crochets []

BASE JAVASCRIPT

TROUVER LA TAILLE D'UNE CHAÎNE DE CARACTÈRES

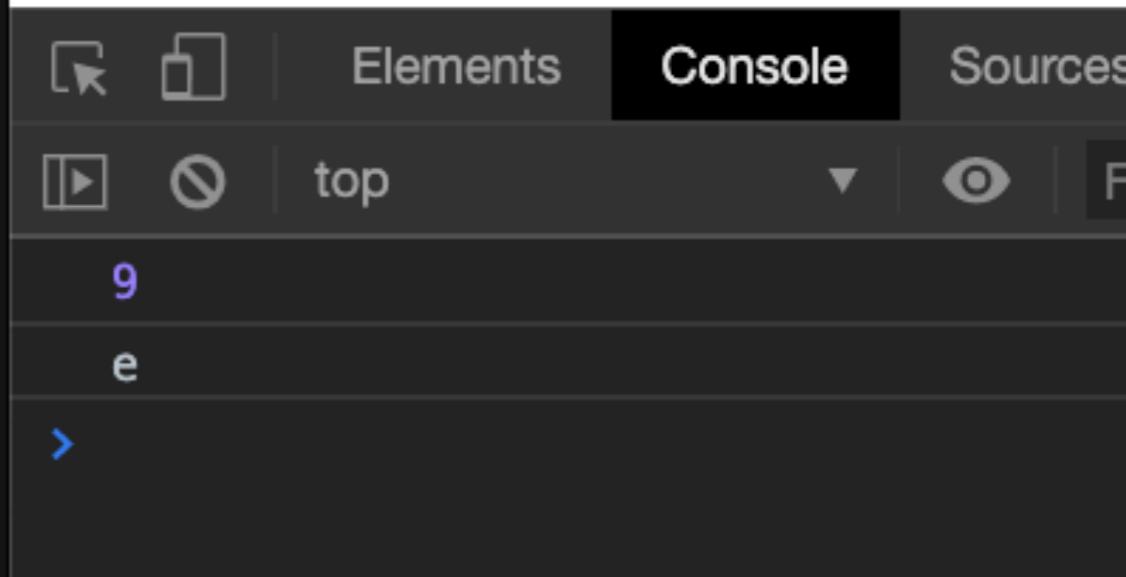
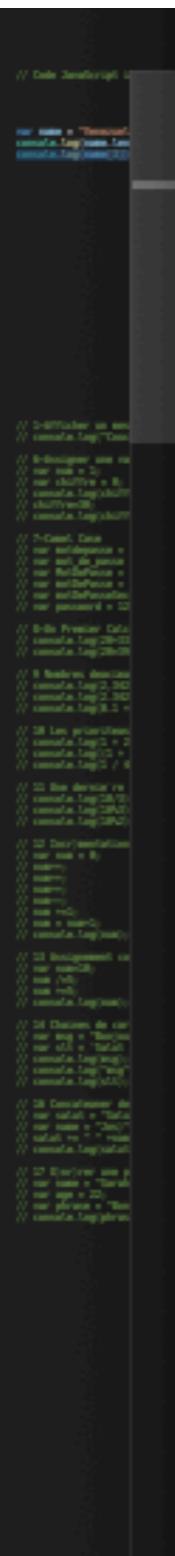
```
1  
2  
3  
4  
5  
6 var name = "Venezuela";  
7 console.log(name.length);  
8  
9  
10  
11  
12  
13  
14
```

BASE JAVASCRIPT

TROUVER LA TAILLE D'UNE CHAÎNE DE CARACTÈRES

▶ main.js > ...

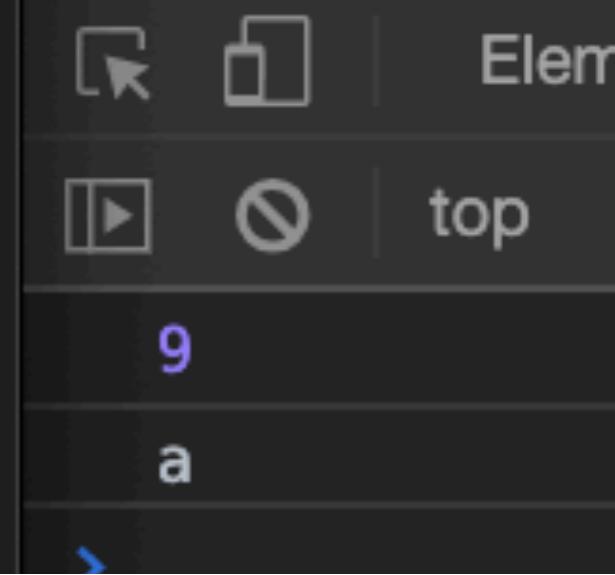
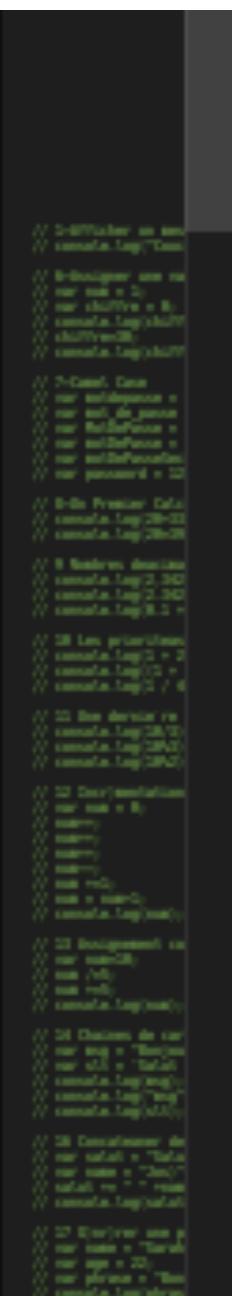
```
1 // Code JavaScript ici  
2  
3  
4  
5  
6 var name = "Venezuela";  
7 console.log(name.length);  
8 console.log(name[3]);  
9  
10  
11  
12  
13  
14  
15  
16  
17
```



BASE JAVASCRIPT

TROUVER LA TAILLE D'UNE CHAÎNE DE CARACTÈRES

```
3  
4  
5  
6 var name = "Venezuela";  
7 console.log(name.length);  
8 // console.log(name[3]);  
9 console.log(name[name.length -1]);  
10  
11  
12  
13  
14  
15  
16
```



BASE JAVASCRIPT

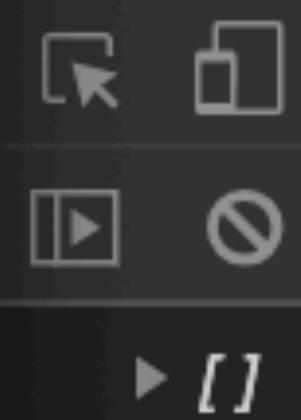
ARRAYS

- En JS on a un type d'objet particulier
- Les arrays, un objet pour stocker des données A LA MANIÈRE d'un tableau
- On peut y stocker plusieurs données, des nombres, des chaînes de caractères, on peut même stocker des tableaux dans des tableaux
- Cela permet de stocker plusieurs informations dans 1 Variable
- En JS on utilise les crochets [] que l'on assigne à une variable.
- On sépare chaque élément du tableau par une ,

BASE JAVASCRIPT

ARRAYS

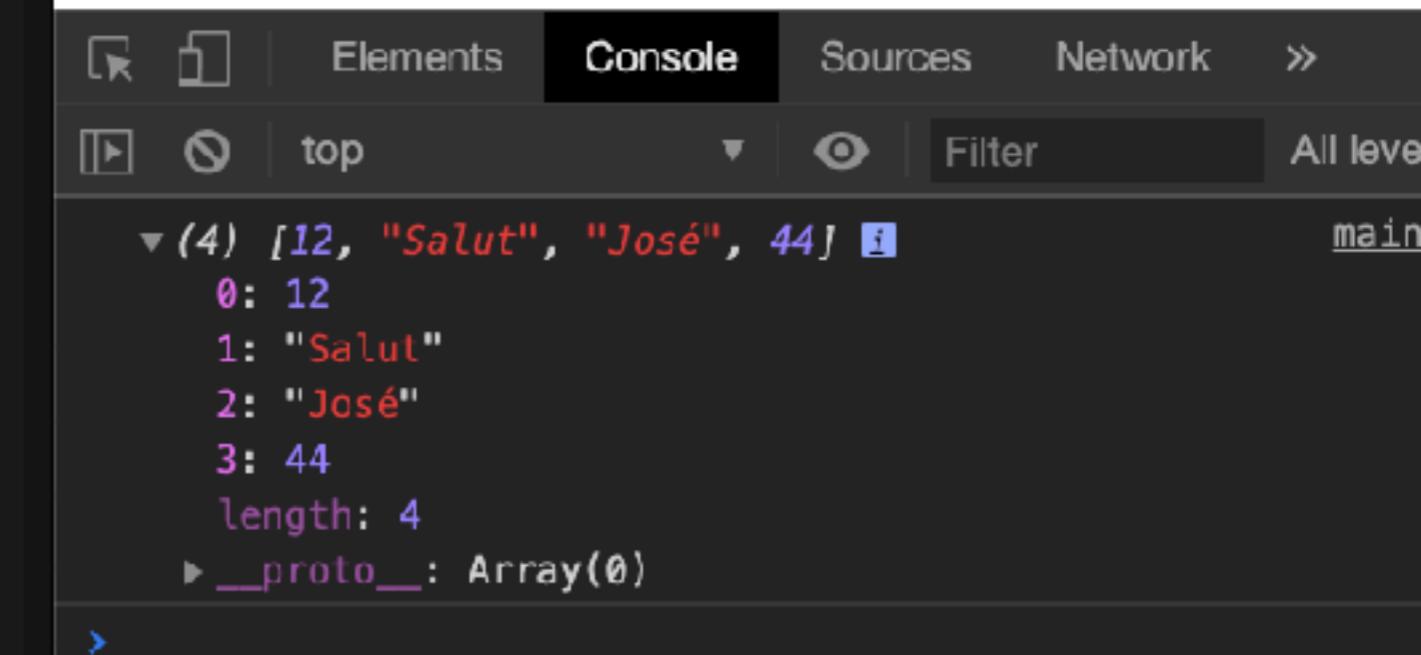
```
4  
5  
6  
7 var name = "José";  
8 var age = 44;  
9 var array = [];  
10 console.log(array);  
11  
12  
13  
14  
15  
16  
17  
18
```



BASE JAVASCRIPT

ARRAYS

```
5 main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 var name = "José";
8 var age = 44;
9 var array = [12, "Salut", name, age];
10 console.log(array);
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```



JS

BASE JAVASCRIPT ARRAYS

The screenshot shows a development environment with the following components:

- Code Editor:** A dark-themed code editor with tabs for "index.html" and "main.js". The "main.js" tab contains the following JavaScript code:

```
1 // Code JavaScript ici
2
3
4
5
6
7 var name = "José";
8 var age = 44;
9 var passions = ["Boxe", "Fleurs"];
10 var array = [name, age, passions];
11 // var array = [12, "Salut", name, age];
12 console.log(array);
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```
- Browser Preview:** A browser window showing the output of "index.html". The URL is "http://127.0.0.1:5".
- Developer Tools:** An open developer tools panel with the "Console" tab selected. It displays the output of the "console.log(array)" statement:

```
(3) ["José", 44, Array(2)] main.js:12
  0: "José"
  1: 44
  2: Array(2)
    0: "Boxe"
    1: "Fleurs"
    length: 2
  > __proto__: Array(0)
  length: 3
  > __proto__: Array(0)
```

BASE JAVASCRIPT

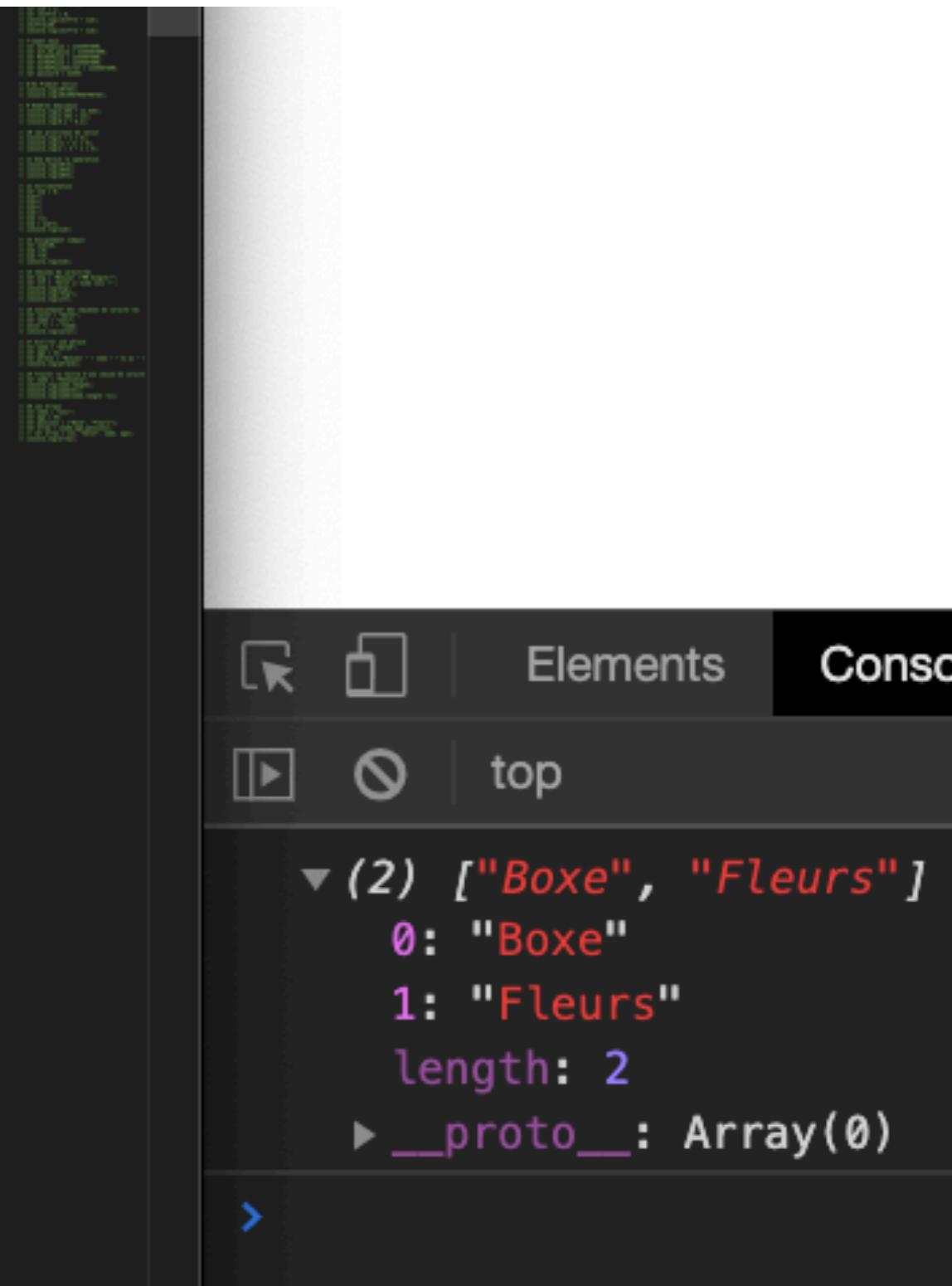
ARRAY : ACCÈS AUX DONNÉES

- On peut accéder spécifiquement à une données contenue dans un array.
- On utilise encore une fois les crochets [] et on renseigne un index [1]

BASE JAVASCRIPT

ARRAY : ACCÈS AUX DONNÉES

```
var name = "José";
var age = 44;
var passions = ["Boxe", "Fleurs"];
var array = [name, age, passions];
console.log(array[2]);
```

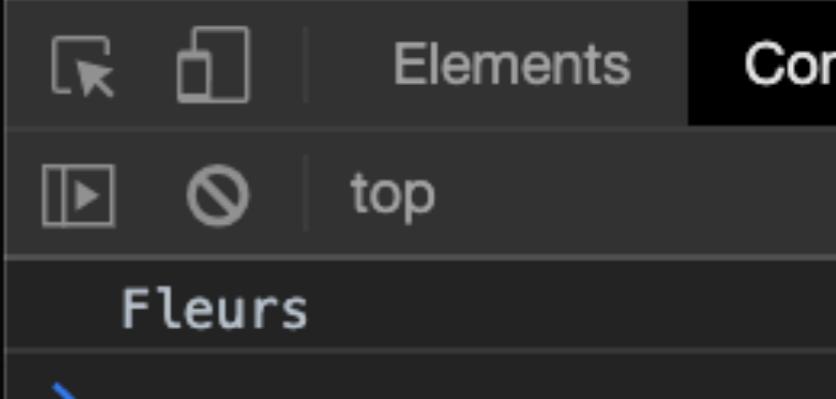


BASE JAVASCRIPT

ARRAY : ACCÈS AUX DONNÉES

main.js > ...

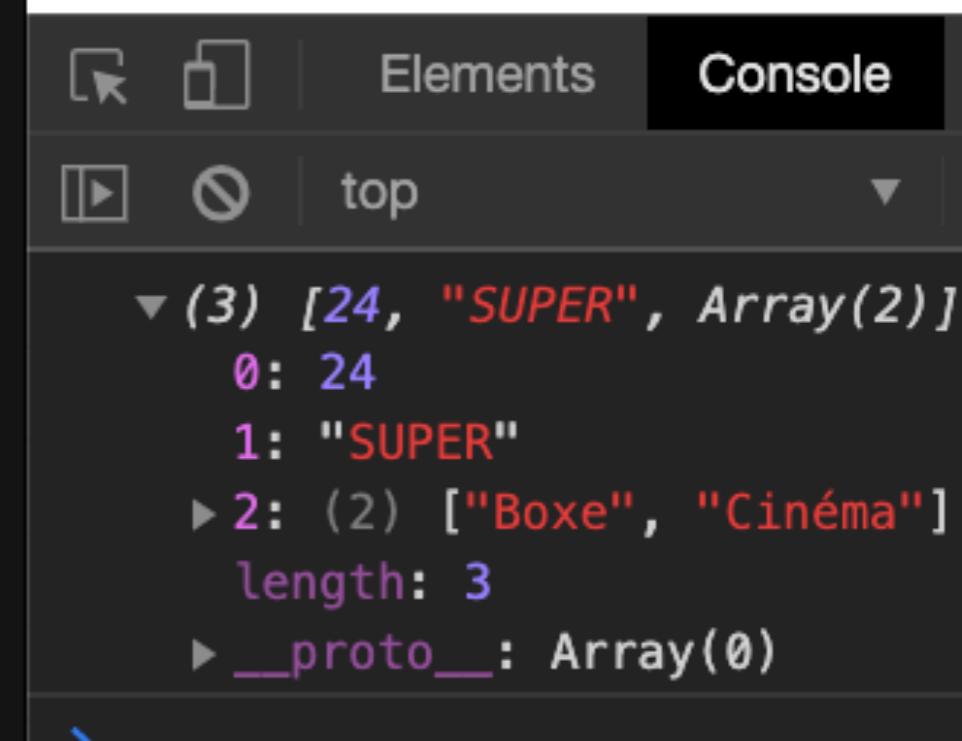
```
1 // Code JavaScript ici  
2  
3  
4  
5  
6 var name = "José";  
7 var age = 44;  
8 var passions = ["Boxe", "Fleurs"];  
9 var array = [name, age, passions];  
10 console.log(array[2][1]);  
11  
12  
13  
14  
15  
16  
17  
18
```



BASE JAVASCRIPT

ARRAY : ACCÈS AUX DONNÉES

```
3  
4  
5  
6 var name = "José";  
7 var age = 44;  
8 var passions = ["Boxe", "Fleurs"];  
9 var array = [name, age, passions];  
10 array[0] = 24;  
11 array[1] = "SUPER";  
12 array[2][1] = "Cinéma";  
13 console.log(array);
```



BASE JAVASCRIPT

ARRAY : TROUVER LA TAILLE

- On utilise `length`

BASE JAVASCRIPT

ARRAY : TROUVER LA TAILLE

The screenshot shows a browser window with the URL `http://127.0.0.1:8000`. In the bottom right corner of the browser, there is a yellow square containing the letters "JS". Below the browser is a dark-themed code editor or developer tools interface. On the left, a file tree shows a file named "main.js". The main pane displays the following JavaScript code:

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7
8 var name = "José";
9 var age = 44;
10 var passions = ["Boxe", "Fleurs"];
11 var array = [name, age, passions];
12 array[0] = 24;
13 array[1] = "SUPER";
14 array[2][1] = "Cinéma";
15 console.log(array[2].length);
16
17
18
```

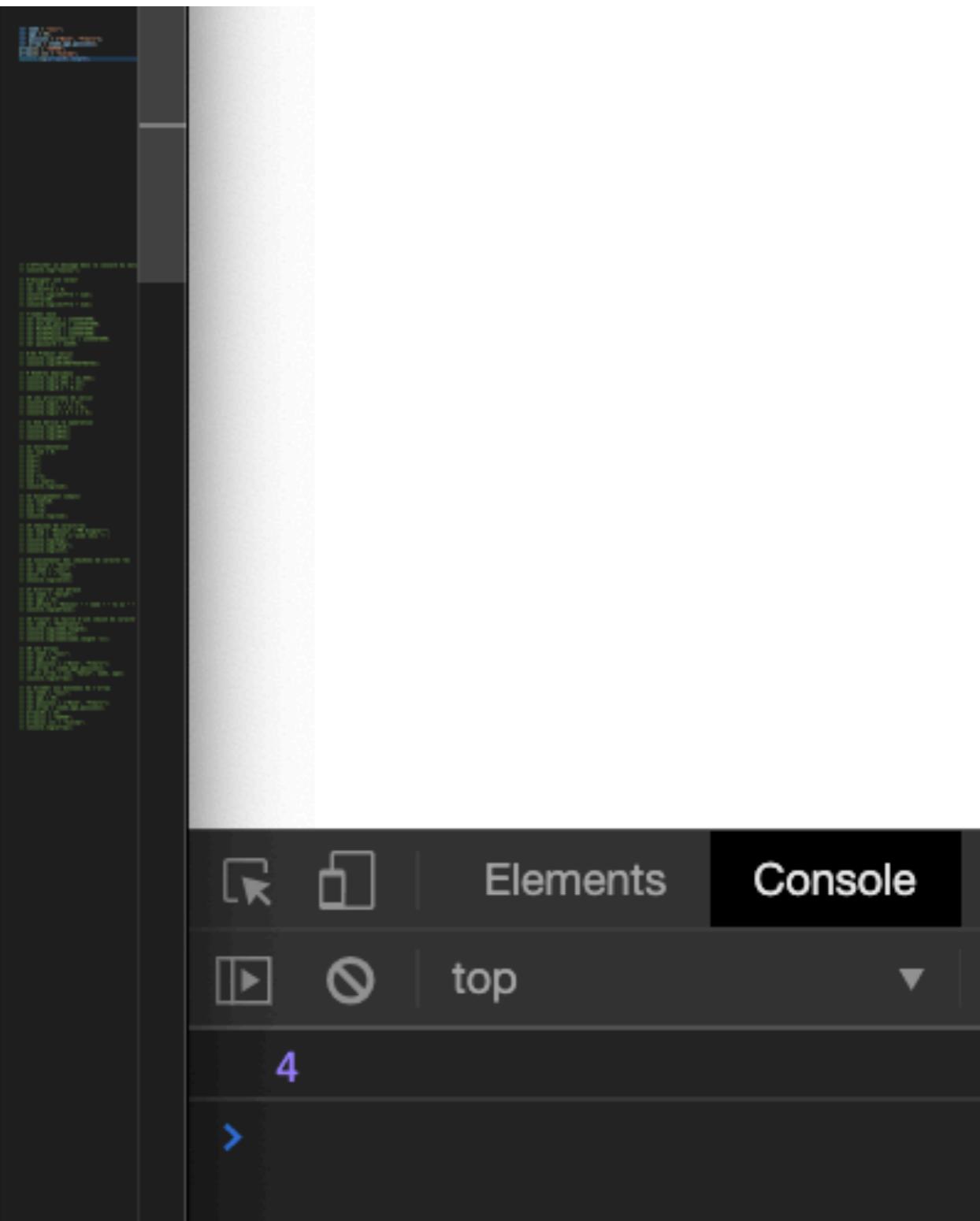
To the right of the code editor is a developer tools panel. At the top of this panel is a toolbar with icons for back, forward, and search. Below the toolbar, there are tabs labeled "Elements" and "Console". The "Console" tab is active, showing the output of the last command: `array[2].length`, which has a value of `2`.

BASE JAVASCRIPT

ARRAY : TROUVER LA TAILLE

// Code JavaScript ici

```
var name = "José";
var age = 44;
var passions = ["Boxe", "Fleurs"];
var array = [name, age, passions];
array[1] = "SUPER";
array[2][1] = "Cinéma";
console.log(array[0].length);
```



BASE JAVASCRIPT

ARRAY : AJOUTER/SUPPR DES DONNÉES

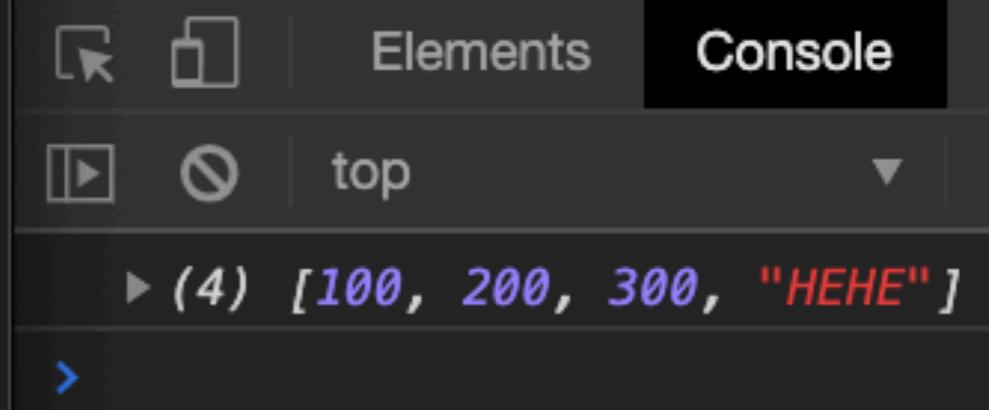
- On l'a vu les tableau permettent de stocker facilement plusieurs variables
- C'est utiles pour regrouper des variables qui ont le même contexte d'utilisation. Par exemple toutes les variable d'un utilisateur
(nom, prénom, âge, mail, mot de passe, etc...)
- Il peut s'avérer très utile de pouvoir ajouter ou supprimer des éléments d'un tableau.
(supprimer âge et ajouter date de naissance à la place)
- On va utiliser la fonction push()

BASE JAVASCRIPT

ARRAY : AJOUTER/SUPPR DES DONNÉES

5 main.js > ...

```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7 var monTab = [100, 200, 300];  
8 monTab.push("HEHE");  
9 console.log(monTab);  
10  
11  
12
```





BASE JAVASCRIPT

ARRAY : AJOUTER/SUPPR DES DONNÉES

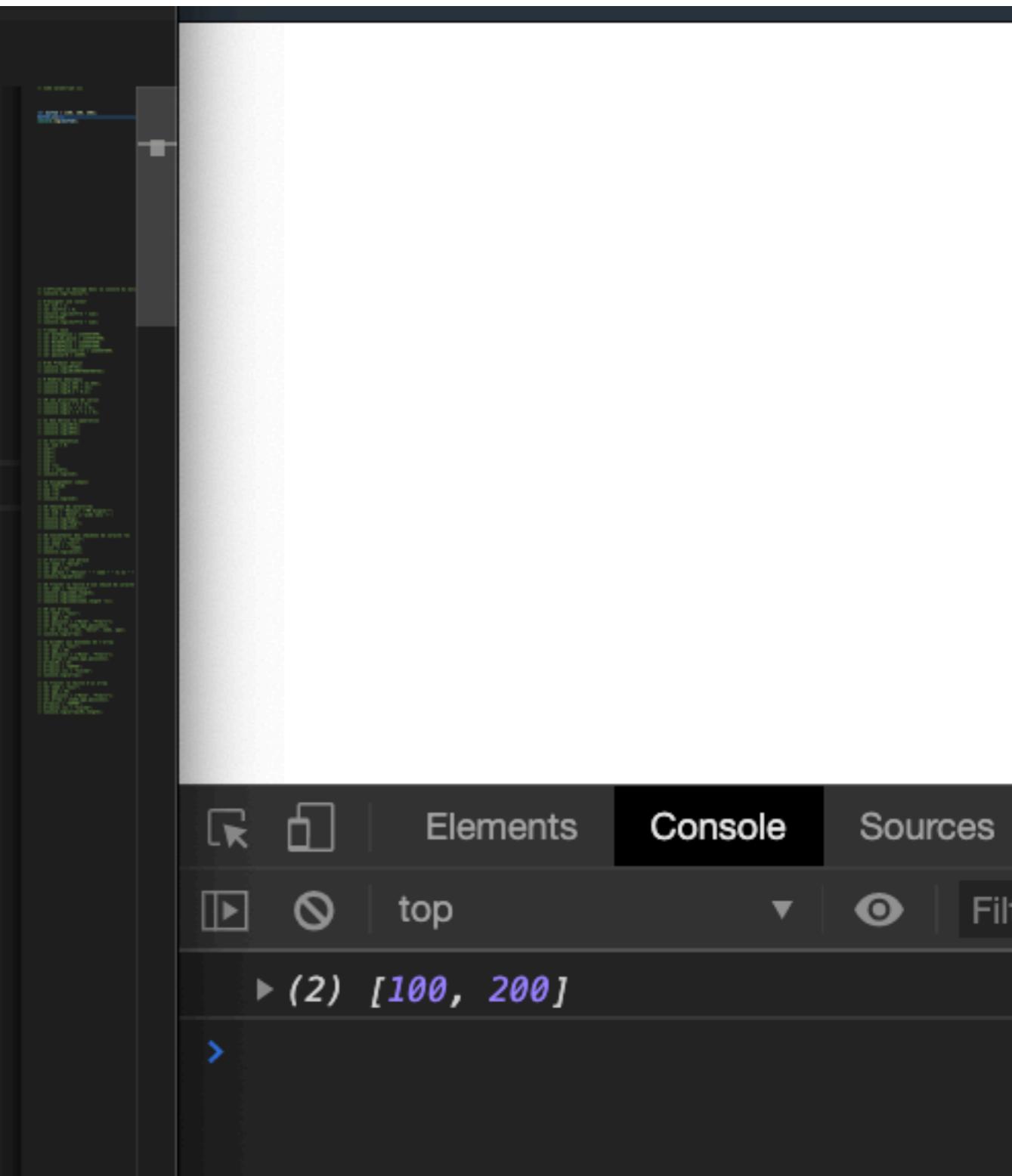
- On peut également supprimer des données
- On va utiliser la fonction Pop()
- Par défaut pop() va supprimer le dernier élément du tableau.

BASE JAVASCRIPT

ARRAY : AJOUTER/SUPPR DES DONNÉES

JS main.js > ...

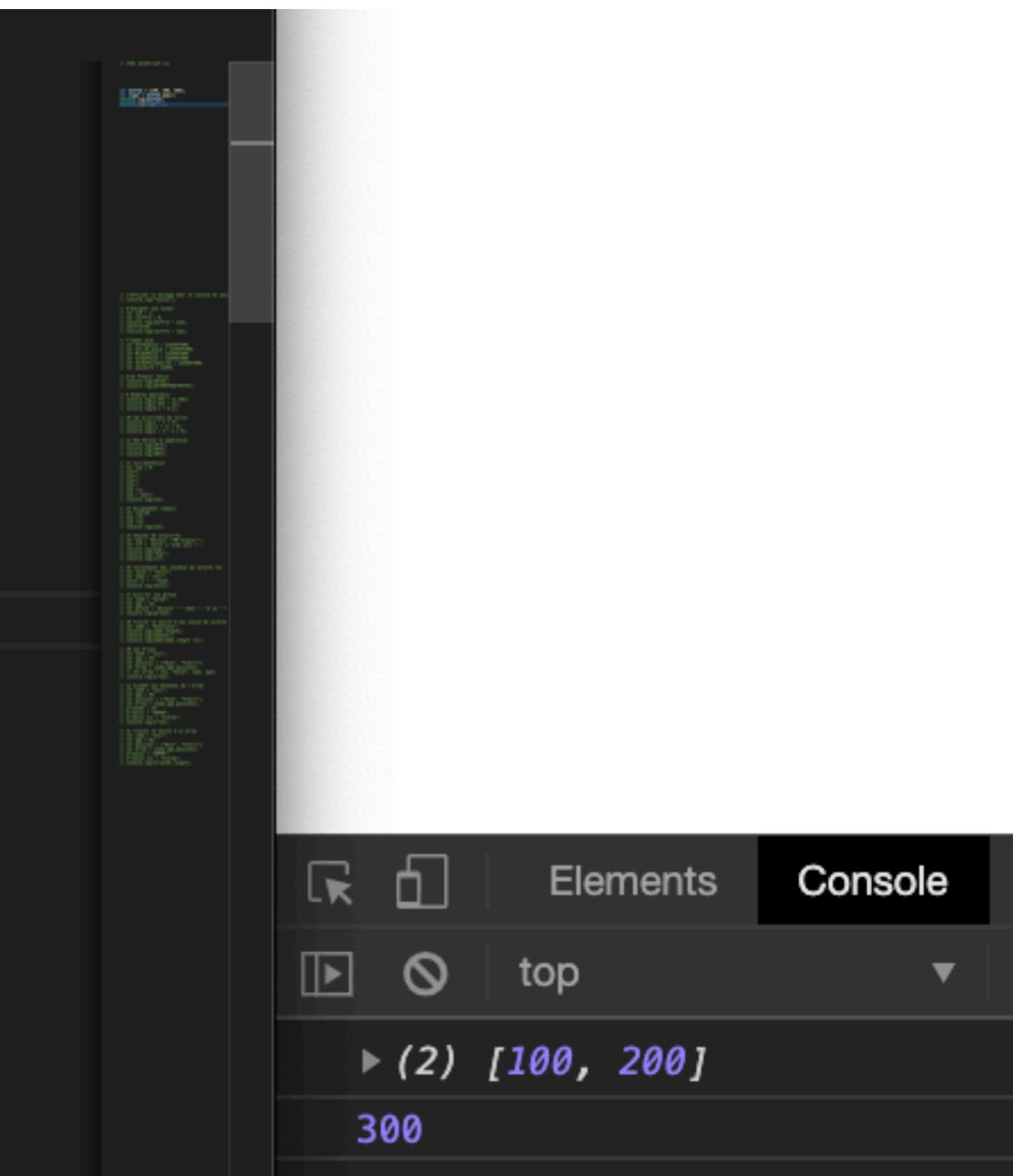
```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7 var monTab = [100, 200, 300];  
8 monTab.pop();  
9 console.log(monTab);  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21
```



BASE JAVASCRIPT

ARRAY : AJOUTER/SUPPR DES DONNÉES

```
main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 var monTab = [100, 200, 300];
8 var suppr = monTab.pop();
9 console.log(monTab);
10 console.log(suppr);
```



BASE JAVASCRIPT

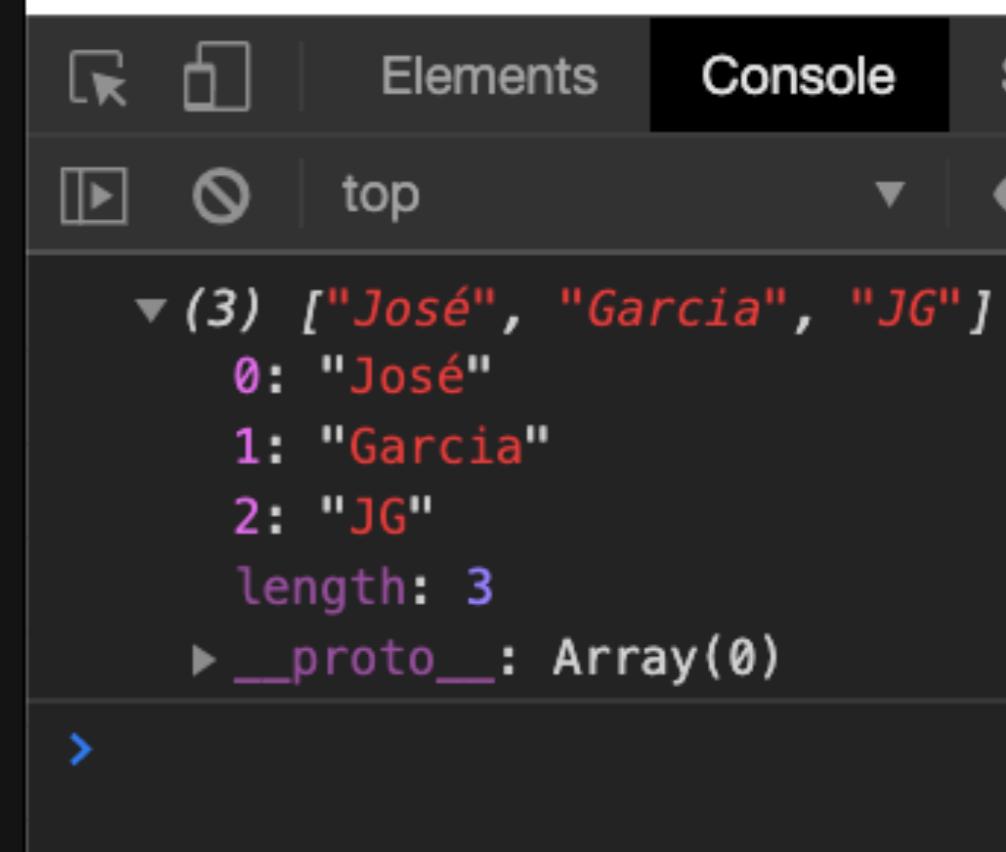
EXO: TROUVER LES INITIALES

- Créer 2 variables
 - nom
 - prénom
- Créer un tableau phrase et on y ajoute
 - Le nom
 - Le prénom
 - Les initiales
- Afficher le tableau dans la console
le nom le prénom et les initiales

BASE JAVASCRIPT

EXO: TROUVER LES INITIALES

```
5 var firstName = "José";
6 var name = "Garcia";
7 var initials = firstName[0] + name[0];
8
9 var phrase = [];
10
11 phrase.push(firstName);
12 phrase.push(name);
13 phrase.push(initials);
14
15 console.log(phrase);
16
17
18
19
20
21
22
23
24
```



JS

BASE JAVASCRIPT

EXO: TROUVER LES INITIALES

The screenshot shows a browser window with developer tools open. On the left, a code editor displays a JavaScript file named 'main.js' with the following code:

```
// Code JavaScript ici
var firstName = "José";
var name = "Garcia";
var initials = firstName[0] + name[0];
var phrase = [];
phrase.push(firstName, name, initials);
console.log(phrase);
```

On the right, the browser's developer tools are visible, specifically the 'Console' tab. The console output is:

```
(3) ["José", "Garcia", "JG"]
```



BASE JAVASCRIPT

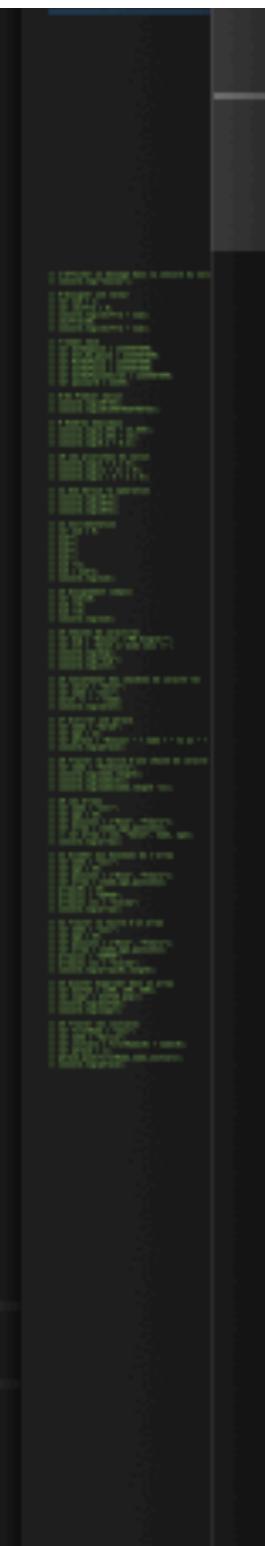
FONCTIONS

- Une fonction c'est un bout de code réutilisable
- C'est utile car en programmation on a souvent besoin de répéter certaines actions.
(Exemple: Enregistrer un nouvel utilisateur)
- On va utiliser function
- Une fonction va être interprétée comme un sous programme dans notre fichier.

BASE JAVASCRIPT FONCTIONS



```
2
3
4
5
6
7
8 function maSuperFonction() {
9     console.log("Hello");
10    console.log(33 + 42);
11 }
12
13 maSuperFonction();
14
15
16
17
18
19
```



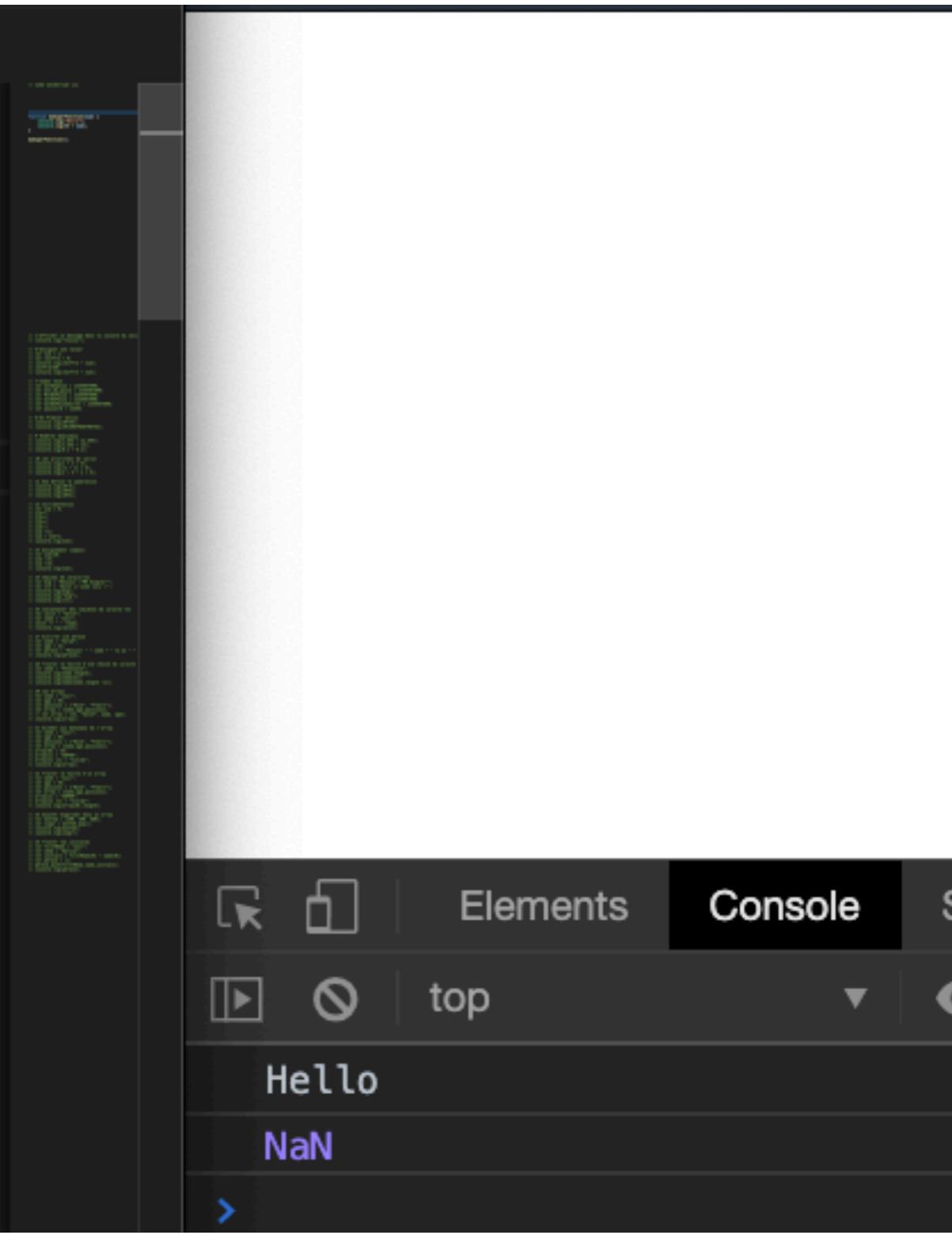
		Elements
		top
		Hello
		75

BASE JAVASCRIPT

FONCTIONS

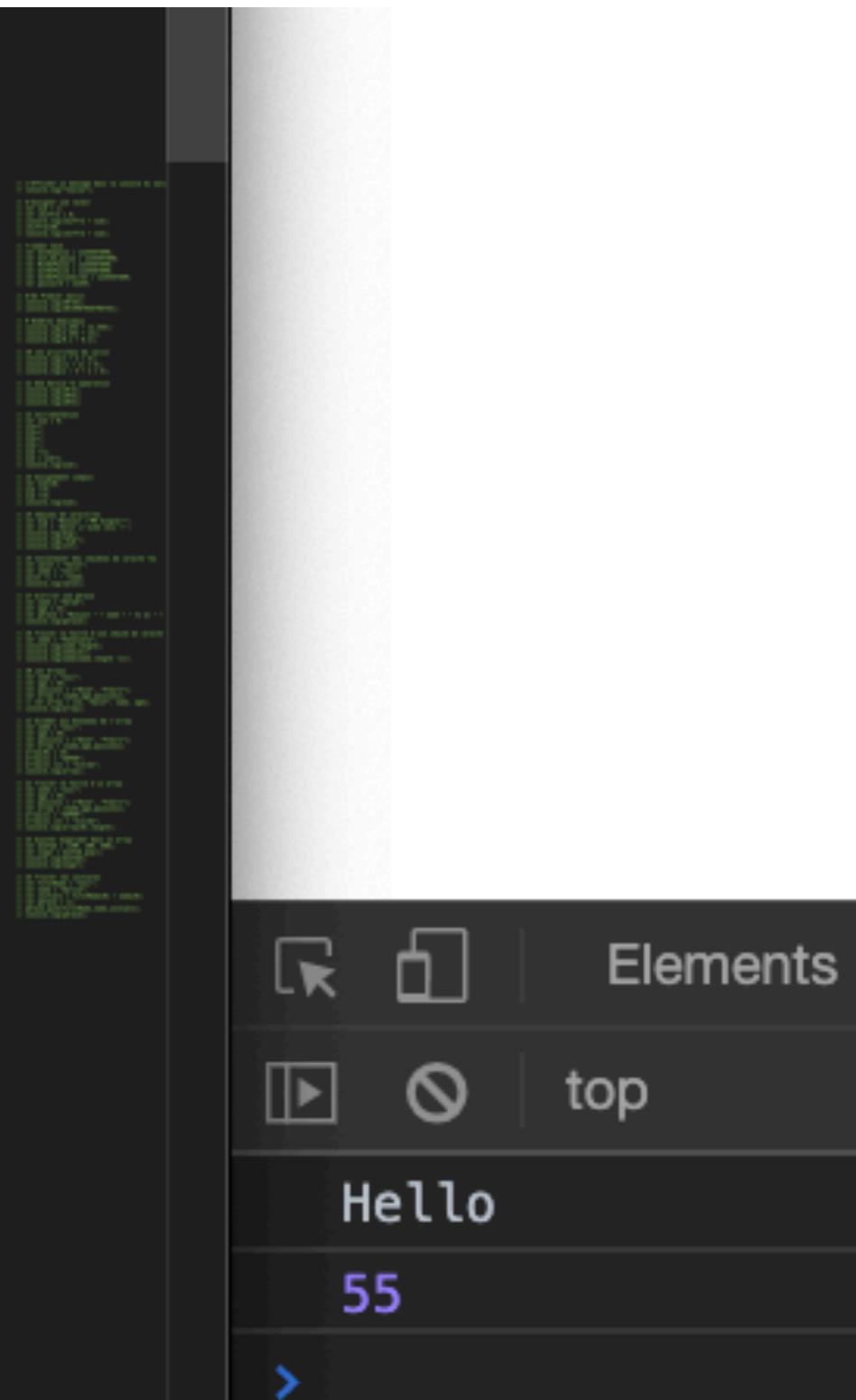
JS main.js > ...

```
1 // Code JavaScript ici
2
3
4
5
6
7
8 function maSuperFonction(num) {
9     console.log("Hello");
10    console.log(33 + num);
11 }
12
13 maSuperFonction();
14
15
16
17
18
19
20
```



BASE JAVASCRIPT FONCTIONS

```
3
4
5
6
7
8 function maSuperFonction(num) {
9     console.log("Hello");
10    console.log(33 + num);
11 }
12
13 maSuperFonction(22);
```



JS

BASE JAVASCRIPT FONCTIONS

The screenshot shows a browser window with developer tools open. On the left, there's a code editor with tabs for 'index.html' and 'main.js'. The 'main.js' tab is active, displaying the following JavaScript code:

```
// Code JavaScript ici
function moins(a, b) {
    console.log(a - b);
}
moins(99, 287);
```

To the right of the code editor is a browser window showing the result of the execution. At the bottom of the browser window is a developer tools panel with tabs for 'Elements', 'Console', and 'Sources'. The 'Console' tab is active, showing the output of the console.log statement: '-188'.



BASE JAVASCRIPT

SCOPE

- Notion de bloc de code
- On l'a vu les fonction agissent comme un sous programme à l'intérieur de notre programme.
- La notion de scope permet de se rendre compte JUSQU'OU une variable peut être accessible dans le code.
- Notion Parent - Enfant

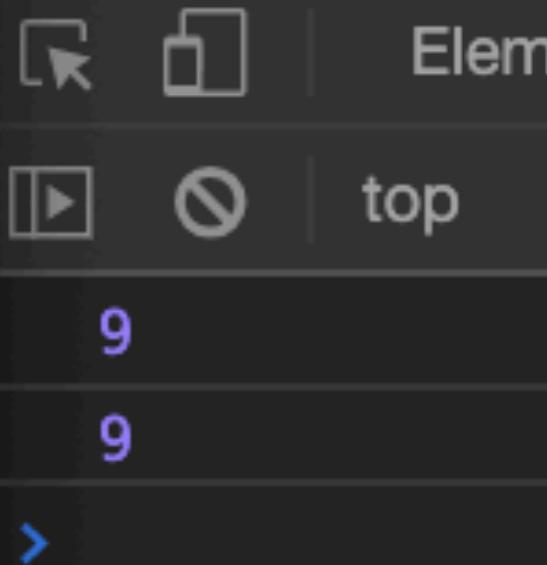
BASE JAVASCRIPT SCOPE

```
7
8
9  var num = 30;
10
11 function uneFonction(){
12     var num = 9;
13     console.log(num);
14 }
15
16 uneFonction();
17 console.log(num);
18
19
20
21
22
23
```



BASE JAVASCRIPT SCOPE

```
6
7
8
9  var num = 30;
10
11 function uneFonction(){
12     num = 9;
13     console.log(num);
14 }
15
16 uneFonction();
17 console.log(num);
```



JS

BASE JAVASCRIPT SCOPE

```
// var num = 30;

function uneFonction(){
    var num = 9;
    console.log(num);
}

uneFonction();
console.log(num);
```



BASE JAVASCRIPT SCOPE

```
7  
8  
9 var num = 30;  
0  
1 function uneFonction(){  
2     | var num = 9;  
3     | console.log(num);  
4     | console.log(test);  
5  
6     function autreFonction(){  
7         | var test = "Un Test";  
8     }  
9 }  
10  
11 uneFonction();  
12 console.log(num);  
13
```



BASE JAVASCRIPT SCOPE

```
5  
6  
7  
8  
9  var num = 30;  
10  
11 function uneFonction(){  
12     | var num = 9;  
13     | console.log(num);  
14     | autreFonction();  
15  
16     function autreFonction(){  
17         | var test = "Un Test";  
18         | console.log(test);  
19     }  
20 }  
21  
22 uneFonction();  
23 console.log(num);  
24
```





BASE JAVASCRIPT

RETOURNER UNE VALEUR

- En JS et dans la plupart des langage de programmation, quand on utilise des fonctions.
- Avec `return` on indique à une fonction qu'elle doit nous retourner ou renvoyer quelque chose.
- C'est utile si on veut stocker le résultat d'une fonction directement dans une variable.

BASE JAVASCRIPT

RETOURNER UNE VALEUR

```
3  
4  
5 function moins(a,b) {  
6     return a-b;  
7 }  
8 var total = moins(56,23);  
9  
10 console.log(total);  
11  
12  
13  
14  
15  
16  
17  
18
```

BASE JAVASCRIPT

EXO: CALCULER UNE MOYENNE

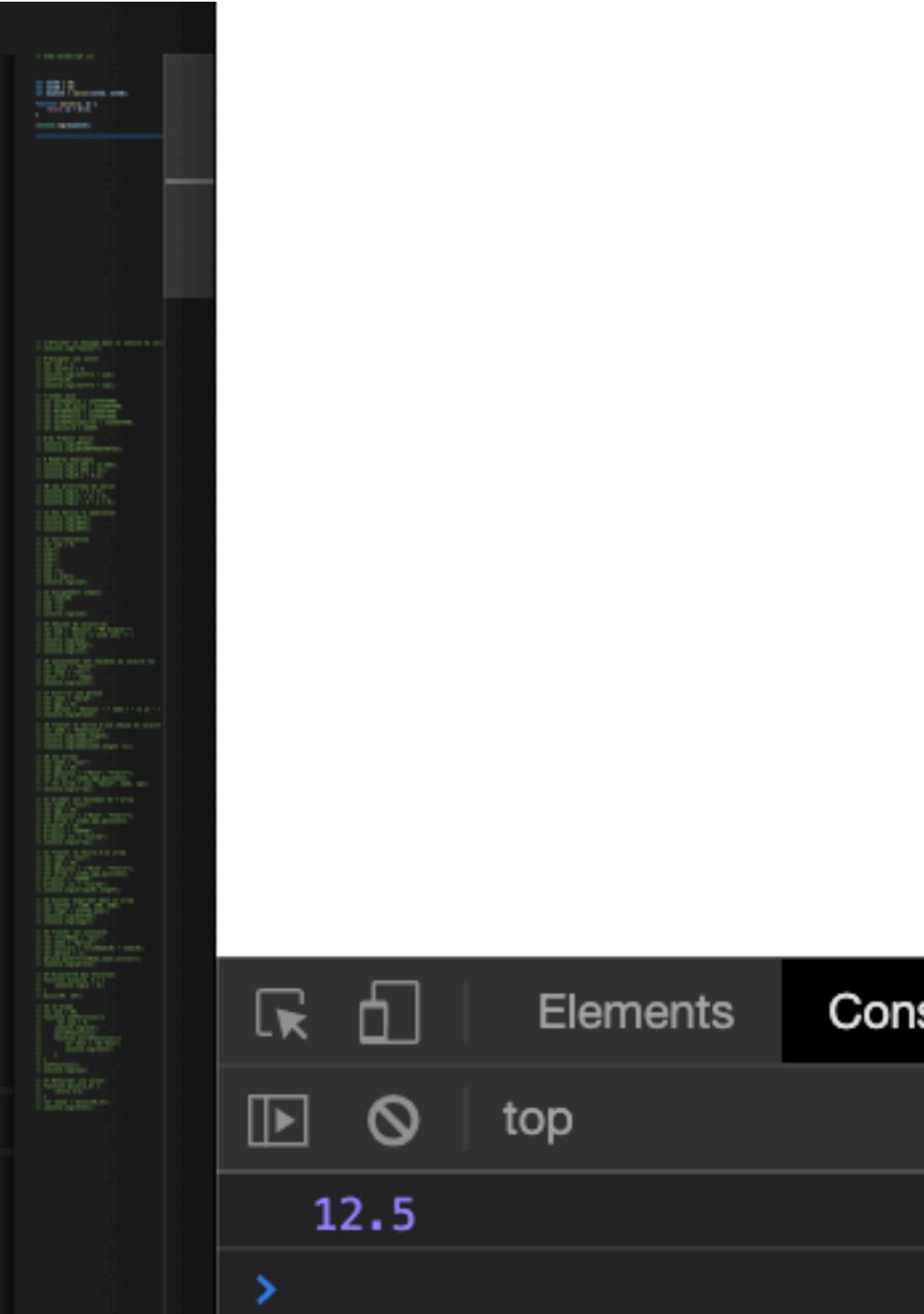
- Créer 2 variables
 - noteA
 - noteB
- Créer une fonction qui va calculer la moyenne et qui retourne le résultat

BASE JAVASCRIPT

EXO: CALCULER UNE MOYENNE

JS main.js > ...

```
1 // Code JavaScript ici
2
3
4
5
6 var noteA = 10;
7 var noteB = 15;
8 var moyenne = calcul(noteA, noteB);
9
10 function calcul(a, b) {
11     return (a + b)/2;
12 }
13
14 console.log(moyenne);
15
16
17
18
19
```



BASE JAVASCRIPT

LES COMMENTAIRES

- On a 2 manière d'écrire des commentaire en JS
- Soit en ligne (in line)
- Soit en bloc
- Quand on commence à avoir beaucoup de code dans notre application les commentaire permettent d'améliorer la visibilité du code et du coup sa maintenance et du coup sa performance.
- Utile également quand des nouveaux développeurs intègre l'équipe.
- Cela permet de mieux se comprendre nous même et entre dev.

BASE JAVASCRIPT

LES COMMENTAIRES

JS main.js

```
1 // Code JavaScript ici
2
3
4
5 /**
6      Un bloc de commentaire
7      on peut y décrire ce que
8      fait une fonction par exemple
9 */
10
11
12
13
14
15
```

BASE JAVASCRIPT

BOOLÉENS

- Utile pour faire des vérifications sur une des variables
- Les booléens n'ont que 2 états True ou False
- Pour effectuer ces vérifications on utilise souvent des opérateur de comparaison
 - `==`
 - `!=`

BASE JAVASCRIPT

BOOLÉENS

```
3  
4  
5  
6 var a = 11;  
7 var b = 99;  
8  
9 console.log(a == b);  
0 console.log(a != b);  
1  
2  
3  
4  
5  
6  
7  
8
```

A screenshot of a browser's developer tools console. The code from the previous slide is pasted into the input field. The output shows two lines: 'false' and 'true', corresponding to the results of the equality and inequality operators respectively.

```
false  
true
```

JS

BASE JAVASCRIPT

CONDITION TERNAIRE

A screenshot of a browser's developer tools console. The code being run is:

```
var age = 11;  
var majorite = 18;  
  
console.log(age == majorite ? "OK" : "NON");
```

The output of the log statement is "NON".

BASE JAVASCRIPT

OPÉRATEURS

- Expressions de comparaison
- < inférieur à
- <= inférieur ou égal à
- === égal à
- >= supérieur ou égal à
- > supérieur à
- != différent de

BASE JAVASCRIPT

OPÉRATEURS

- Égalité simple ou égalité stricte == VS ===
- Égalité simple vérifie la valeur mais pas le Type
- `2 == "2"` TRUE
- `2 === "2"` FALSE
- Innégalité
- != ou !==

BASE JAVASCRIPT

OPÉRATEURS

- Opérateurs Logique
- Pour combiner plusieurs conditions
- ET &&
- OU ||
- NON !

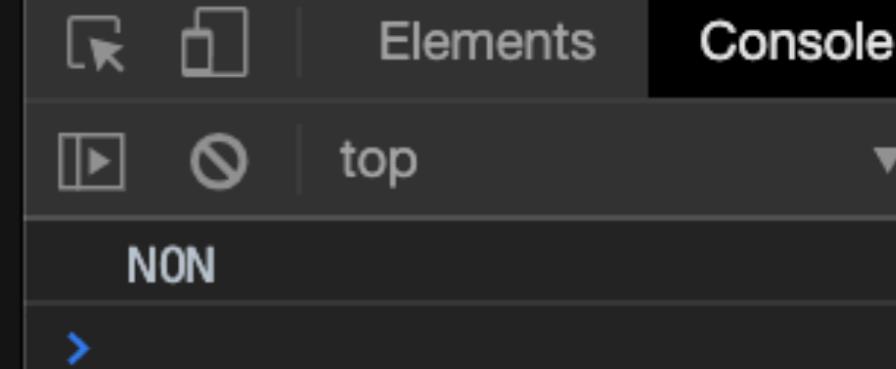
BASE JAVASCRIPT

OPÉRATEURS

JS

JS main.js > [e] age

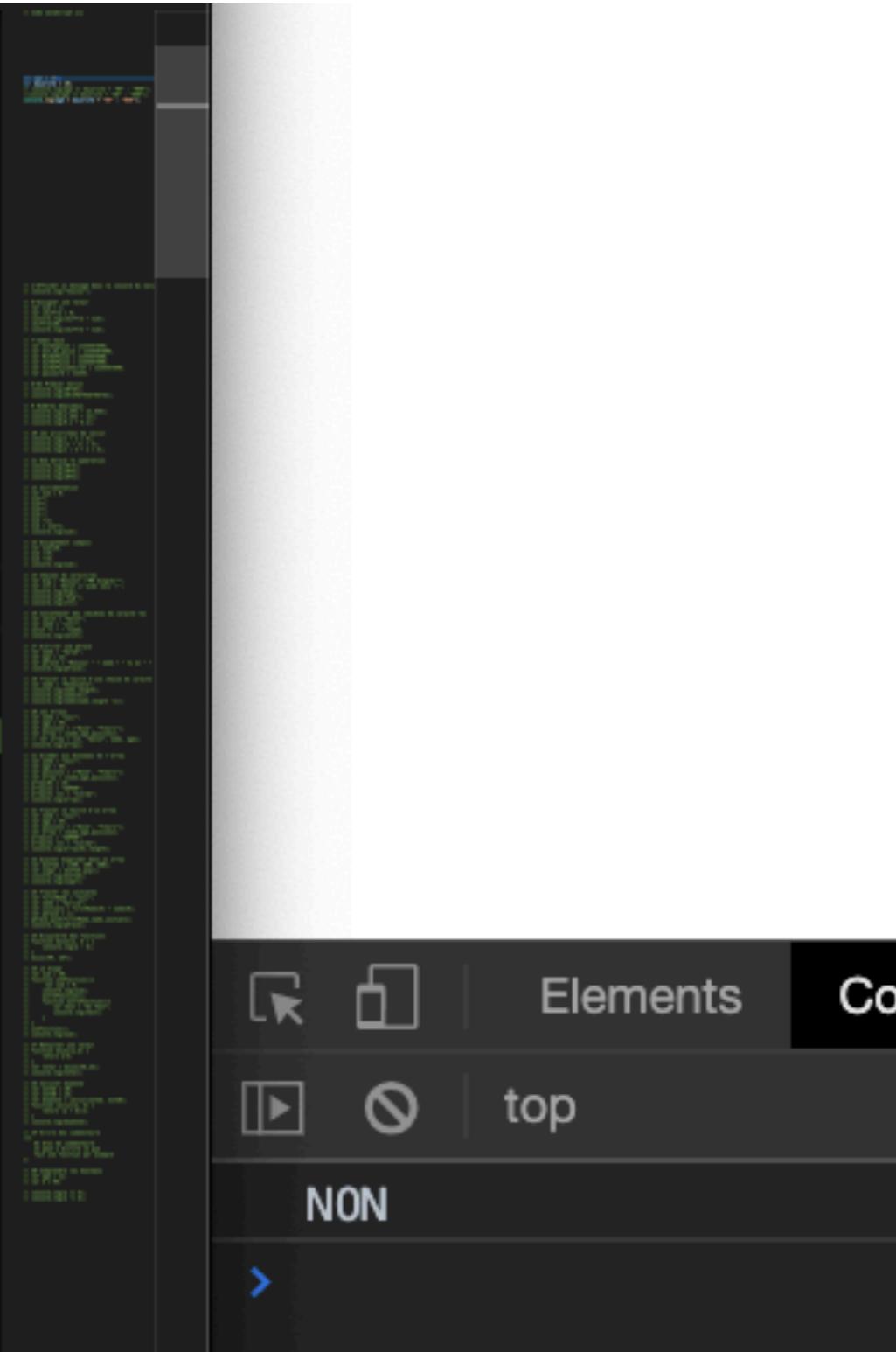
```
5  
6  
7  
8  
9  
10  
11  
12  
13 var age = 22;  
14 var majorite = 18;  
15  
16 console.log(age == majorite ? "OK" : "NON");  
17  
18  
19  
20  
21  
22  
23
```



BASE JAVASCRIPT

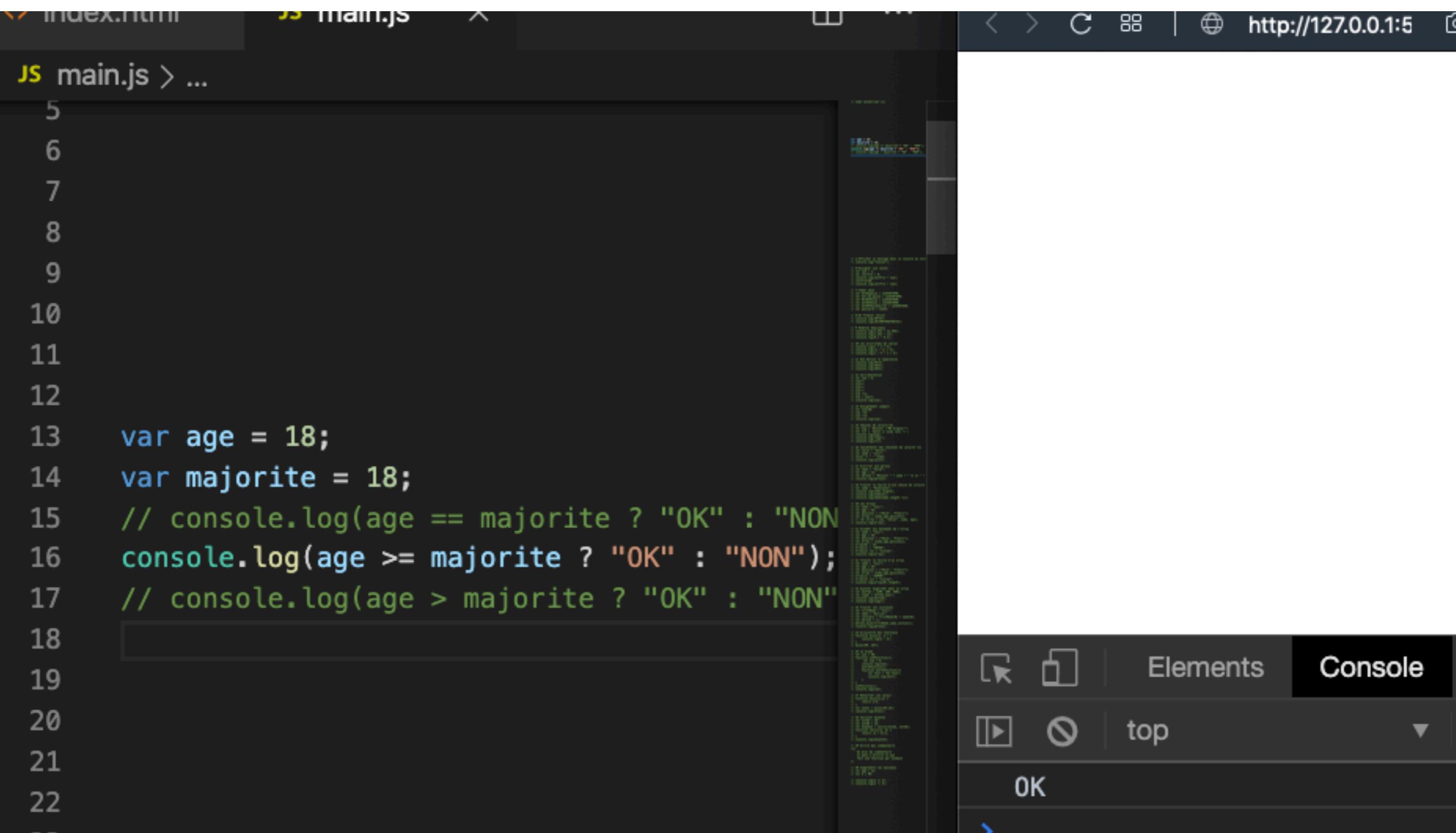
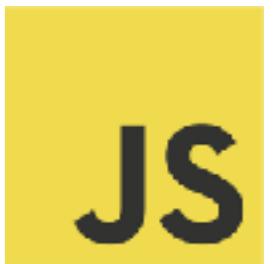
OPÉRATEURS

```
5  
6  
7  
8  
9  
10  
11  
12  
13 var age = 18;  
14 var majorite = 18;  
15 // console.log(age == majorite ? "OK" : "NON")  
16 //console.log(age >= majorite ? "OK" : "NON")  
17 console.log(age > majorite ? "OK" : "NON");  
18  
19  
20  
21  
22  
23
```



BASE JAVASCRIPT

OPÉRATEURS



The screenshot shows a browser window with developer tools open. The left pane displays a portion of a JavaScript file named 'main.js' containing code related to age and majority status. The right pane shows the browser's developer tools interface, specifically the 'Console' tab, which displays the output 'OK'.

```
JS main.js > ...
5
6
7
8
9
10
11
12
13 var age = 18;
14 var majorite = 18;
15 // console.log(age == majorite ? "OK" : "NON")
16 console.log(age >= majorite ? "OK" : "NON");
17 // console.log(age > majorite ? "OK" : "NON")
18
19
20
21
22
23
```

Elements Console

OK

BASE JAVASCRIPT

OPÉRATEURS

```
5  
6  
7  
8  
9  
10  
11  
12  
13 var age = 18;  
14 var majorite = 18;  
15 console.log(3 ==3 && 3<4);  
16 // console.log(age == majorite ? "OK" : "NON")  
17 // console.log(age >= majorite ? "OK" : "NON")  
18 // console.log(age > majorite ? "OK" : "NON")  
19  
20  
21  
22  
23
```



BASE JAVASCRIPT

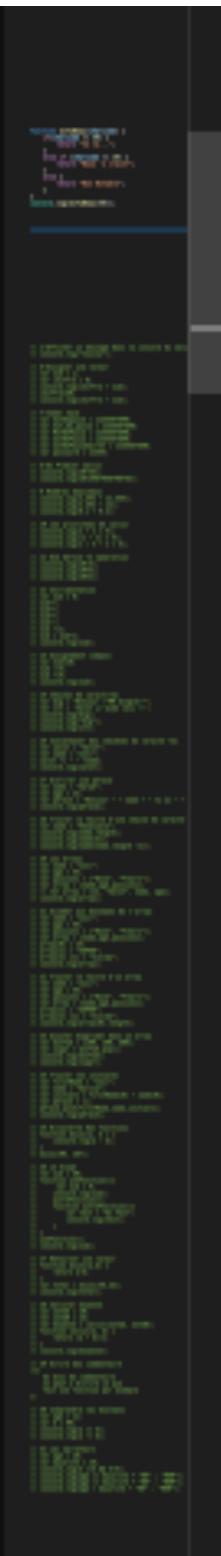
CONDITION IF ELSE

- On a vu comment faire des conditions avec les conditions ternaires
- Avec If et else on peut pousser le concept encore plus loin
- Si on veut établir des conditions plus précises et plus complexes.
- On déclare notre condition avec if
 - On exécute du code (si la condition est remplie)
- Avec Else on déclare le code qui s'exécute si la condition n'est pas remplie.
- On peut imbriquer des if dans des if

BASE JAVASCRIPT

CONDITION IF ELSE

```
17  
18  
19  
20  
21  
22  
23 function esTuBeau(charisme) {  
24     if(charisme >= 30) {  
25         return "Ca va...";  
26     }  
27     else if (charisme <= 10) {  
28         return "Waow ça craint";  
29     }  
30     else {  
31         return "Non Notable";  
32     }  
33 }  
34 console.log(esTuBeau(50));
```



```
Elements  
top  
Ca va...>
```

BASE JAVASCRIPT

EXO: CALCULER UNE MOYENNE

- Créer une fonction mention qui va recevoir en paramètre un tableau de 3 notes du BAC (Français, Math, Philo).
- Cette fonction a pour but de nous renvoyer du texte en fonction de la moyenne des 3 notes qu'on lui passe :
 - Si la moyenne est supérieure ou = à 15 on retourne Très Bien
 - Si la moyenne est supérieure ou = à 10 on retourne Assez Bien
 - Sinon on retourne Refus

BASE JAVASCRIPT

EXO: CALCULER UNE MOYENNE

```
function mention(notes){  
var moy = (notes[0] +notes[1]+ notes[2]) /notes.length;  
if (moy >= 16) {  
    return "Très Bien";  
}  
else if (moy >= 10) {  
    return "Assez Bien";  
}  
return "Niet"  
console.log(mention([13, 9, 10]));
```

BASE JAVASCRIPT

OBJETS

- On a vu comment stocker des données via les Arrays.
- Ensuite on stockait cet array dans une variable.
- C'est donc très pratique pour regrouper des variables.
- Mais on peut zapper une étapes
- En stockant directement un objet dans une variable.
- On a plusieurs manière d'accéder aux variables contenues dans un objet.

JS

BASE JAVASCRIPT

OBJETS

```
var voiture = {  
    "nbChevaux": 500,  
    "vitesseMax": 450,  
    "faiblesse": ["accélération", "adhérence"]  
};  
  
console.log(voiture.vitesseMax);
```

450

BASE JAVASCRIPT

OBJETS

```
5  
6  
7  
8 var voiture = {  
9     "nbChevaux": 500,  
10    "vitesseMax": 450,  
11    "faiblesse": ["accélération","adhérence"]  
12};  
13 var afficher = "vitesseMax";  
14  
15 console.log(voiture.vitesseMax);  
16 console.log(voiture["vitesseMax"]);  
17 console.log(voiture [afficher]);  
18 console.log(voiture.faiblesse[1]);  
19  
20  
21
```

BASE JAVASCRIPT

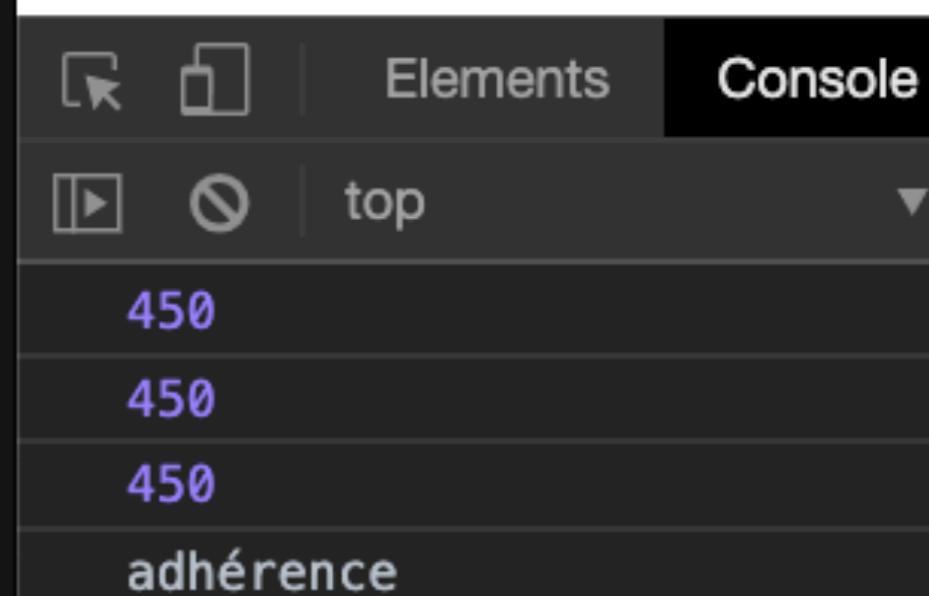
MANIPULER DES OBJETS

- Comment modifier, ajouter ou supprimer des propriétés d'un objet.
- On accède aux données contenus dans notre variable objet nomObjet.propriété
- Si la propriété n'existe pas, elle est créée, si elle existe déjà on peut la modifier.
- On peut supprimer une propriété d'un objet en utilisant delete.

BASE JAVASCRIPT

MANIPULER DES OBJETS

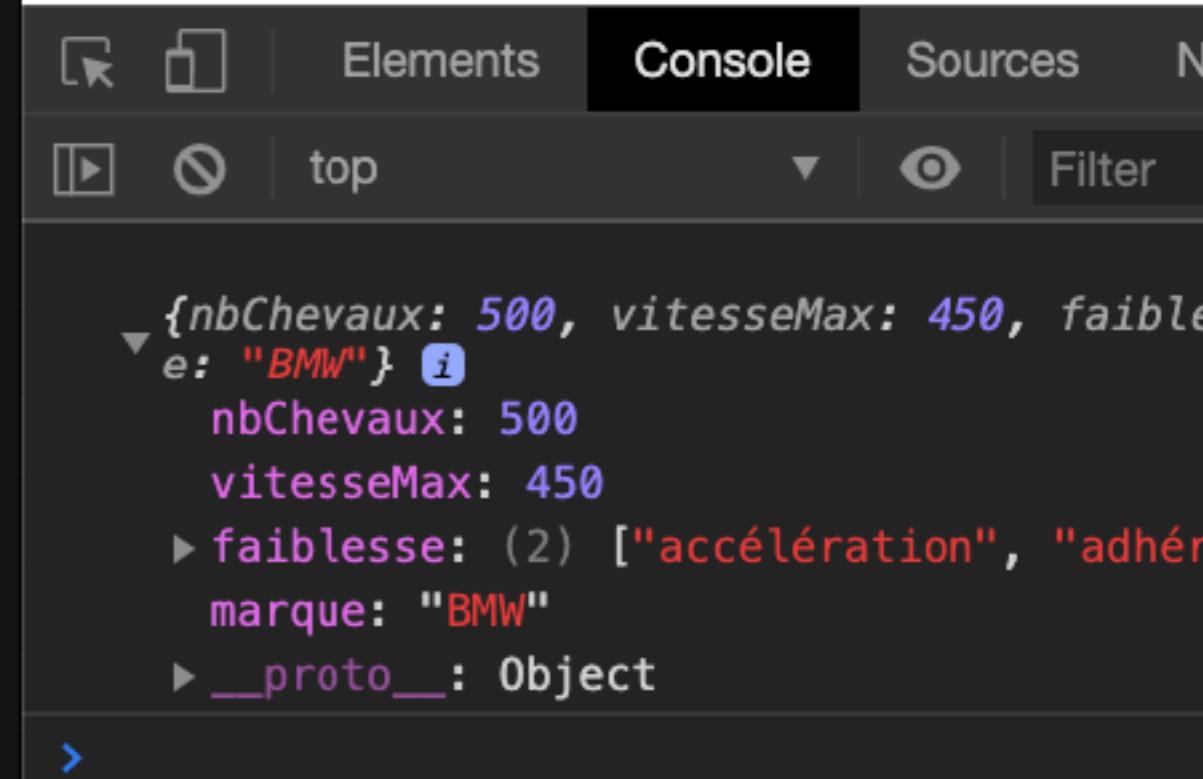
```
var voiture = {  
    "nbChevaux": 500,  
    "vitesseMax": 450,  
    "faiblesse": ["accélération", "adhérence"]  
};  
var afficher = "vitesseMax";  
  
console.log(voiture.vitesseMax);  
console.log(voiture["vitesseMax"]);  
console.log(voiture [afficher]);  
console.log(voiture.faiblesse[1]);
```



BASE JAVASCRIPT

MANIPULER DES OBJETS

```
var voiture = {  
    "nbChevaux": 500,  
    "vitesseMax": 450,  
    "faiblesse": ["accélération", "adhérence"]  
};  
var afficher = "vitesseMax";  
voiture.marque = "BMW";  
console.log(voiture);  
// console.log(voiture.vitesseMax);  
// console.log(voiture["vitesseMax"]);  
// console.log(voiture [afficher]);  
// console.log(voiture.faiblesse[1]);
```



BASE JAVASCRIPT

MANIPULER DES OBJETS

```
// Code JavaScript ici
```

```
var voiture = {
    "nbChevaux": 500,
    "vitesseMax": 450,
    "faiblesse": ["accélération", "adhérence"]
};
var afficher = "vitesseMax";
voiture.marque = "BMW";
delete voiture.marque;
console.log(voiture);
// console.log(voiture.vitesseMax);
// console.log(voiture["vitesseMax"]);
// console.log(voiture [afficher]);
// console.log(voiture.faiblesse[1]);
```



BASE JAVASCRIPT

OBJETS (BONUS)

- Comment connaitre les propriétés d'un objet.
- Dans le cas ou l'on a beaucoup de code.
- On veut utiliser une propriété d'un objet mais on ne sait pas si cette dernière existe.
- On va utiliser une fonction qui renvoi true ou false si l'objet possède ou non la propriété.
`hasOwnProperty()`

BASE JAVASCRIPT

OBJETS (BONUS)

- On peut aussi utiliser le format JSON
Javascript Object Notation
- JSON va être un gros objet JS qui fonctionne un peu comme une base de données.
- Il va représenter nos données sous forme d'objet JS
- On peut même stocker 1 Objet dans 1 objet.

BASE JAVASCRIPT

OBJETS (BONUS)

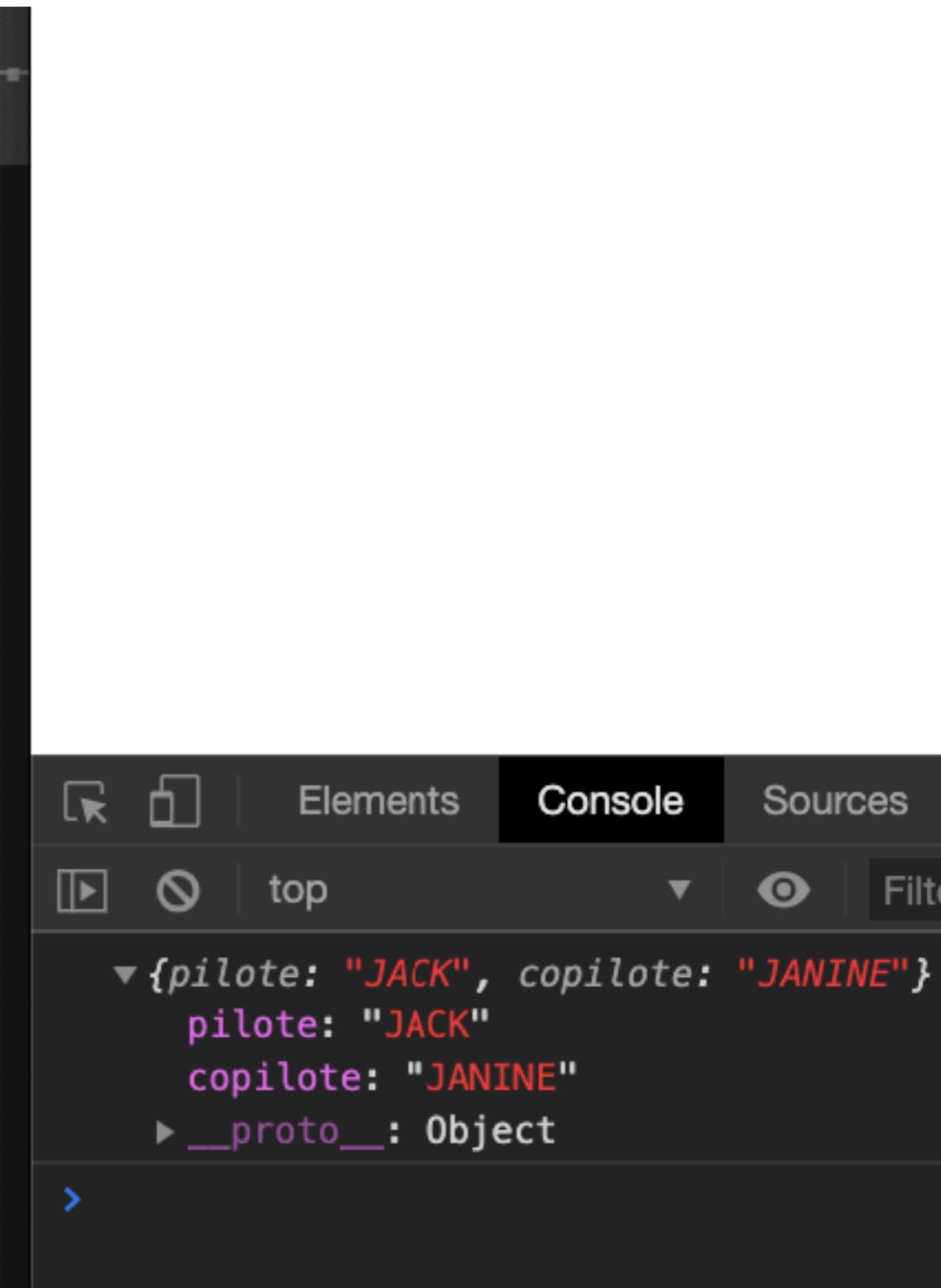
```
4  
5  
6  
7 var voiture = {  
8     "nbChevaux": 500,  
9     "vitesseMax": 450,  
10    "faiblesse": ["accélération", "adhérence"]  
11};  
12 var afficher = "vitesseMax";  
13 voiture.marque = "BMW";  
14 delete voiture.marque;  
15 console.log(voiture.hasOwnProperty("vitesseMax"));  
16  
17  
18  
19  
20  
21  
22
```



BASE JAVASCRIPT

OBJETS (BONUS)

```
2  
3  
4  
5  
6  
7 var voiture = {  
8     "nbChevaux": 500,  
9     "vitesseMax": 450,  
10    "faiblesse": ["accélération","adhérence"],  
11    "pilotes": {  
12        "pilote" : "JACK",  
13        "copilote" : "JANINE"  
14    }  
15};  
16 var afficher = "vitesseMax";  
17 voiture.marque = "BMW";  
18 delete voiture.marque;  
19 console.log(voiture.pilotes);  
20  
21  
22  
23  
24  
25  
26  
27  
28
```



BASE JAVASCRIPT

OBJETS (BONUS)

```
5  
6  
7 var voiture = {  
8     "nbChevaux": 500,  
9     "vitesseMax": 450,  
10    "faiblesse": ["accélération","adhérence"],  
11    "pilotes": {  
12        "pilote" : "JACK",  
13        "copilote" : "JANINE"  
14    }  
15};  
16 var afficher = "vitesseMax";  
17 voiture.marque = "BMW";  
18 delete voiture.marque;  
19 console.log(voiture.pilotes.copilote);  
20  
21  
22  
23  
24  
25
```





BASE JAVASCRIPT

BOUCLE WHILE

- En programmation il arrive très souvent d'avoir des tâches répétitives à faire.
- Pour faire des boucles on a plusieurs manière
- La boucle while
- On décrit une condition
- Tant que la condition est remplie
- On exécute le code.
- On se sert d'une valeur index pour l'itération.
- Attention aux boucles infinies.

JS

BASE JAVASCRIPT

BOUCLE WHILE

The screenshot shows a browser window with developer tools open. On the left, the code editor displays a file named 'main.js' with the following content:

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 var i =0;
8
9 while (i < 10) {
10     i++;
11     console.log(i);
12 }
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

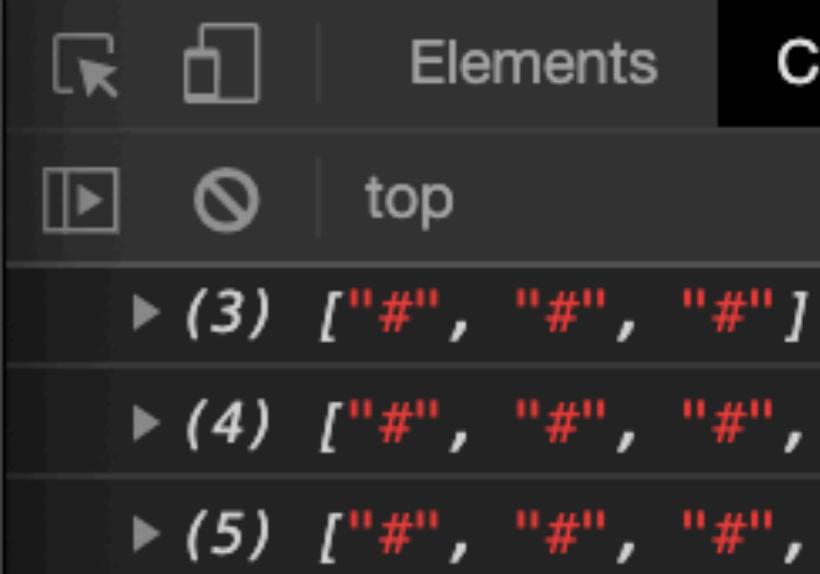
The right side of the image shows the browser's developer tools interface, specifically the 'Console' tab. The console output window displays the numbers 1 through 9, each preceded by a purple timestamp and followed by the file name 'main.js:11'. This indicates that the loop was executed 10 times, starting from 0 and ending at 9.

```
1 main.js:11
2 main.js:11
3 main.js:11
4 main.js:11
5 main.js:11
6 main.js:11
7 main.js:11
8 main.js:11
9 main.js:11
```

BASE JAVASCRIPT

BOUCLE WHILE

```
var i = 0;  
var dessin = [];  
  
while (i < 10) {  
    i++;  
    dessin.push("#");  
    console.log(dessin);  
}
```



BASE JAVASCRIPT

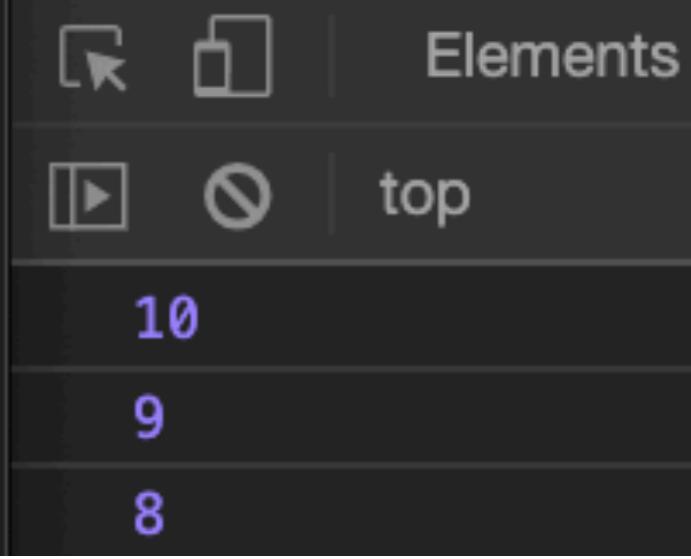
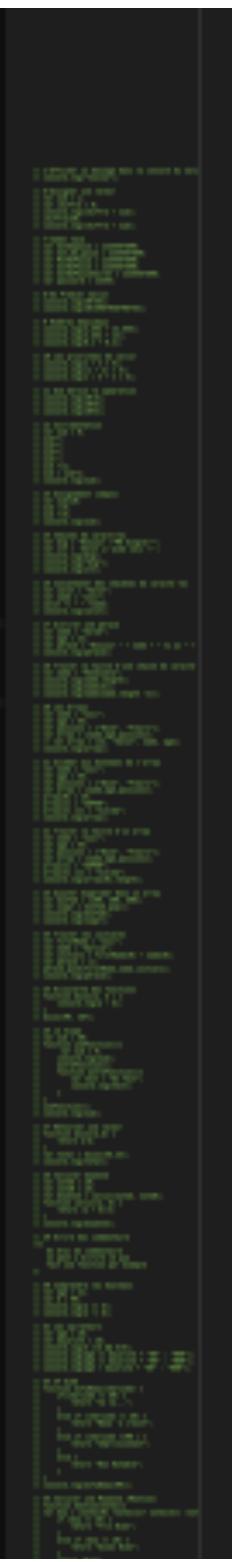
BOUCLE FOR

- Il existe une autre manière d'utiliser des boucles
- For
- On passe à la fonction for 3 arguments
 - On crée l'index (compter le nombre de fois que la boucle va être exécutée)
 - On donne la condition selon laquelle on reste dans la boucle (Si $i <$ nombre de tables)
 - Itération de l'index pour suivre le nombre d'exécutions dans la boucle

BASE JAVASCRIPT

BOUCLE FOR

```
3  
4  
5  
6  
7 var dessin=[];  
8  
9 for(var i = 10; i > 0; i--){  
10     console.log(i);  
11 }  
12  
13  
14  
15  
16  
17  
18  
19  
20
```



BASE JAVASCRIPT

BOUCLE FOR

```
2  
3  
4  
5  
6  
7 var dessin=[];  
8  
9 for(var i = 10; i > 0; i -=2){  
10    dessin.push("#");  
11    console.log(dessin);  
12 }  
13  
14  
15  
16  
17  
18  
19  
20  
21
```





BASE JAVASCRIPT

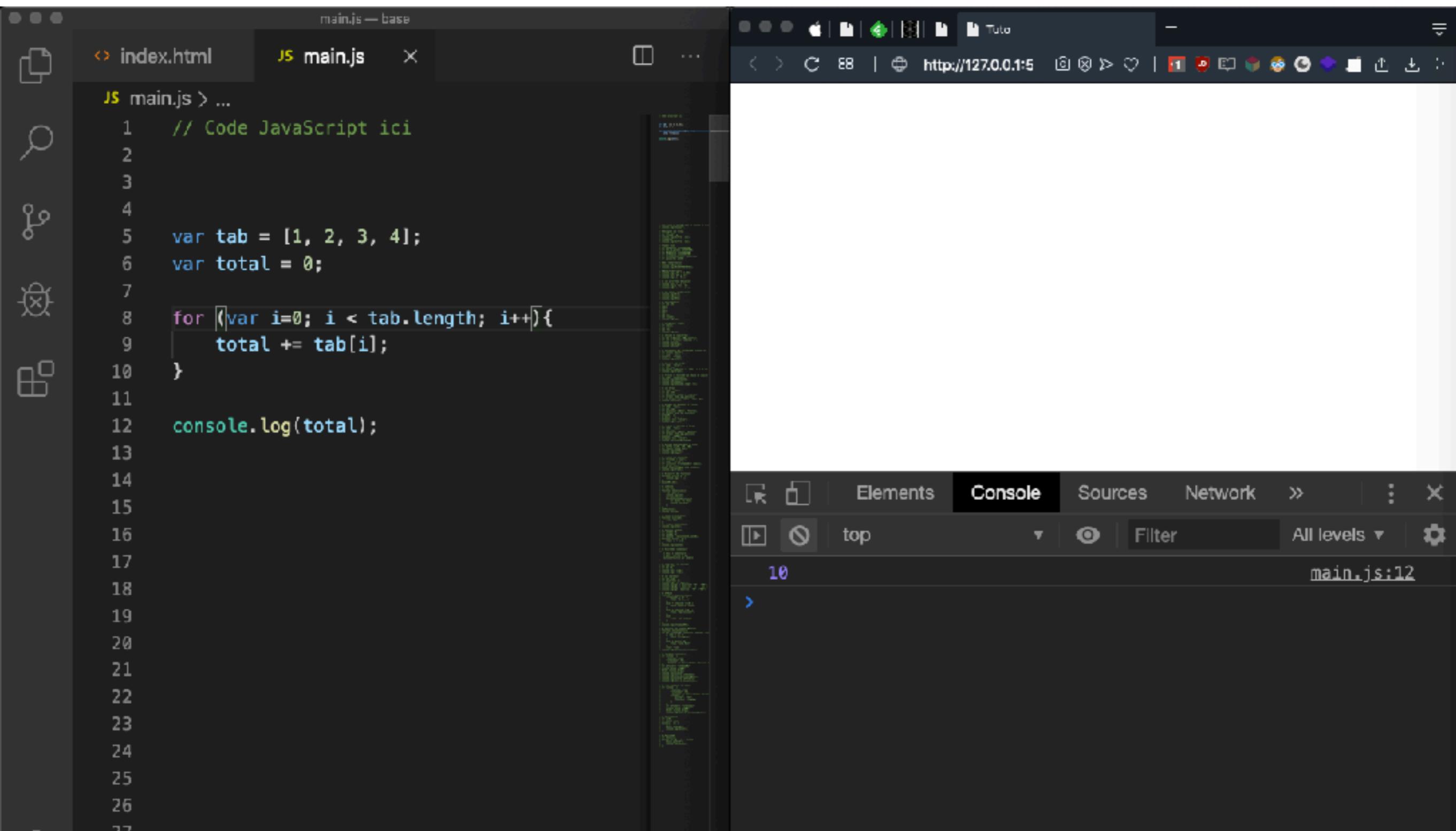
EXEMPLE BOUCLE FOR

- Comment faire une boucle en fonction du nombre de données si on ne connaît pas le nombre de données ?
- On a beaucoup de notes du Bac et on veut faire un total
- Parfois on a 2 notes, parfois 4, parfois 7 etc ...

 JS

BASE JAVASCRIPT

EXEMPLE BOUCLE FOR



The screenshot shows a development environment with two main windows. On the left is a code editor with tabs for 'index.html' and 'main.js'. The 'main.js' tab contains the following JavaScript code:

```
// Code JavaScript ici
var tab = [1, 2, 3, 4];
var total = 0;

for (var i=0; i < tab.length; i++){
    total += tab[i];
}

console.log(total);
```

On the right is a browser window showing the result of running the script. The address bar shows 'http://127.0.0.1:5'. Below it is a developer tools console tab labeled 'Console'. The console output shows the value '10' at line 12 of 'main.js'.

BASE JAVASCRIPT

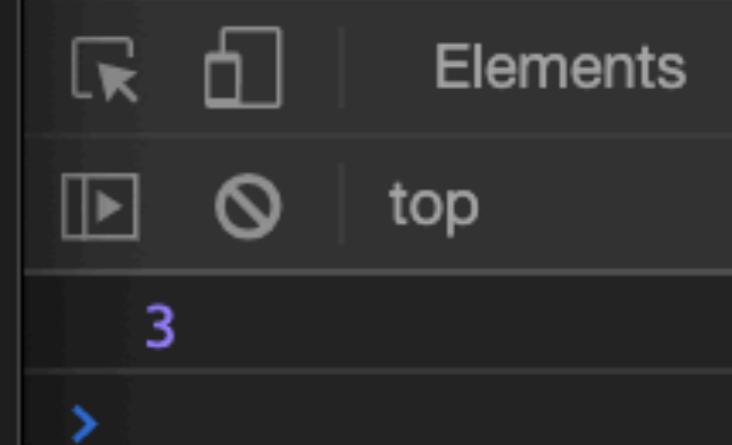
EXO: CALCULER MOYENNE

- 1 fonction moyenne qui prend en paramètre un array de plusieurs notes et elle retourne la moyenne calculée.
- Dans cette fonction
On créer une boucle pour faciliter le calcul de la moyenne
- On console log un appel à la fonction moyenne en lui passant un tableau de plusieurs notes.

BASE JAVASCRIPT

EXO: CALCULER MOYENNE

```
3  
4  
5  
6  
7  
8 function moyenne (tab) {  
9     var total = 0;  
10    // on Boucle pour tout additionner  
11    for(var i =0; i < tab.length; i++){  
12        total += tab[i];  
13    }  
14    // On fait la Division  
15    return total / tab.length;  
16}  
17 console.log(moyenne([1,2,3,4,5]));  
18  
19  
20
```



BASE JAVASCRIPT

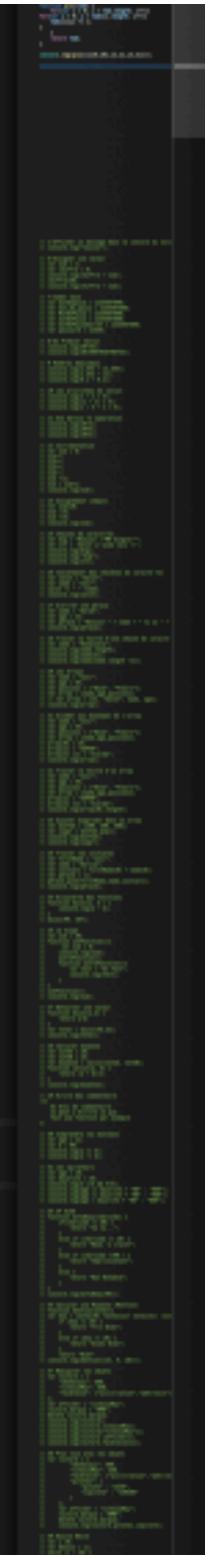
BOUCLE DANS UNE BOUCLE

- Quand nos applications se complexifient
- On peut avoir besoin d'utiliser des boucles imbriquées (1 boucle à l'intérieur d'une boucle).
- Par exemple si j'ai 1 tableau qui contient plusieurs tableaux de chiffres.
- Je veux ajouter +1 à toutes les cases du second Tableau.

BASE JAVASCRIPT

BOUCLE DANS UNE BOUCLE

```
1 // Code JavaScript ici
2
3
4
5
6 function plus(tab) {
7   for(var i = 0; i < tab.length; i++){
8     for(var j = 0; j < tab[i].length; j++){
9       tab[i][j] += 1;
10    }
11  }
12  return tab;
13 }
14
15 console.log(plus([[10,10],[2,2],[3,3]]));
16
17
18
19
20
21
22
```



```
Elements Console Sources
top
▼ (3) [Array(2), Array(2), Array(2)]
▶ 0: (2) [11, 11]
▶ 1: (2) [3, 3]
▶ 2: (2) [4, 4]
  length: 3
▶ __proto__: Array(0)
```

BASE JAVASCRIPT

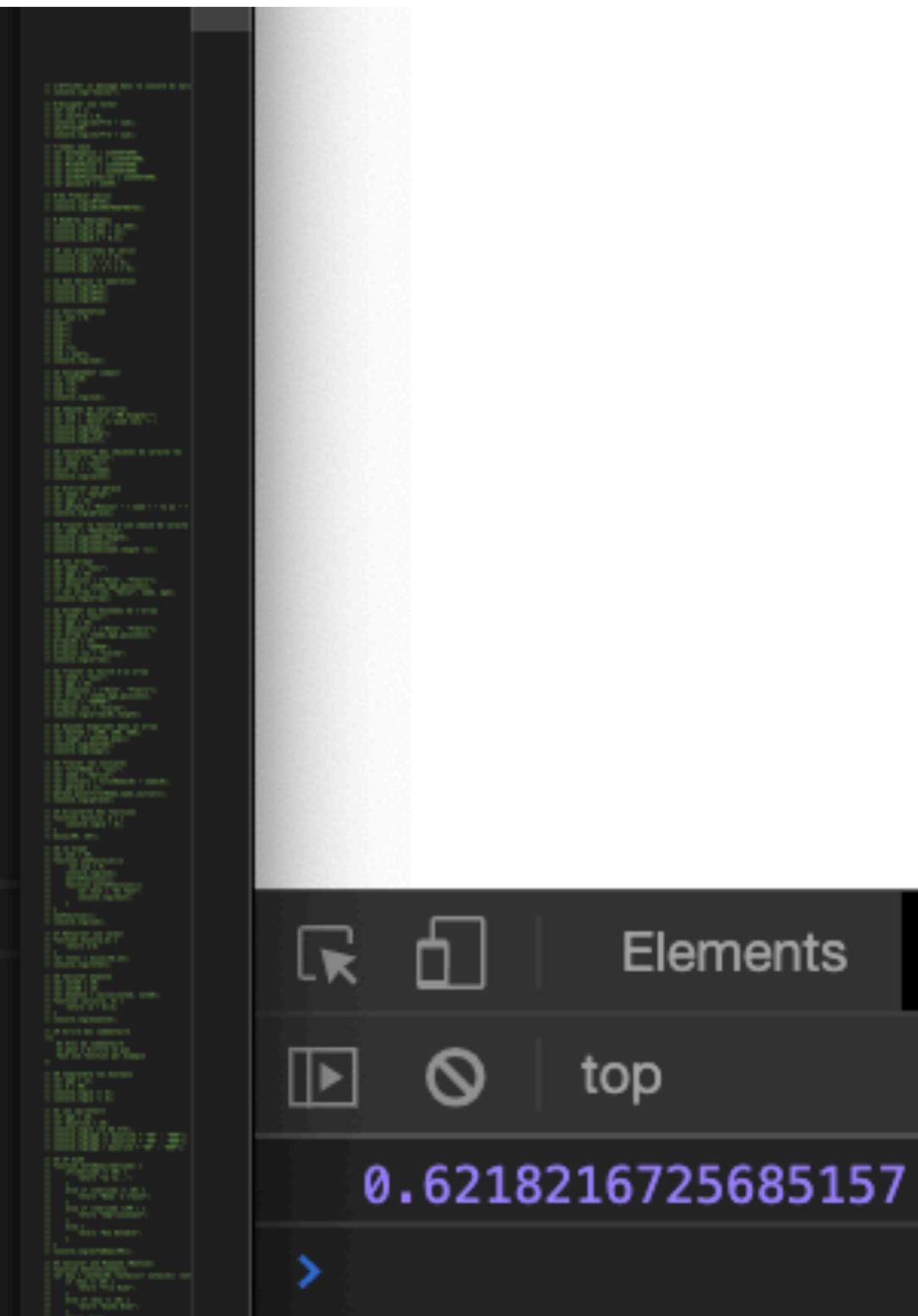
GÉNÉRER UN NOMBRE ALÉATOIRE

- Gestion du hasard.
- En JS on va utiliser l'objet Math
(C'est un objet Natif de JS du coup on l'appelle Math)
- Gérer l'aléatoire peut être utile dans certains cas
(Jeux de hasard, loterie, exécuter des tests ...)
- On peut arrondir au dessous le résultat d'un chiffre aléatoire avec
`Math.floor()`

BASE JAVASCRIPT

GÉNÉRER UN NOMBRE ALÉATOIRE

```
var random = Math.random();  
  
console.log(random);
```



JS

BASE JAVASCRIPT

GÉNÉRER UN NOMBRE ALÉATOIRE

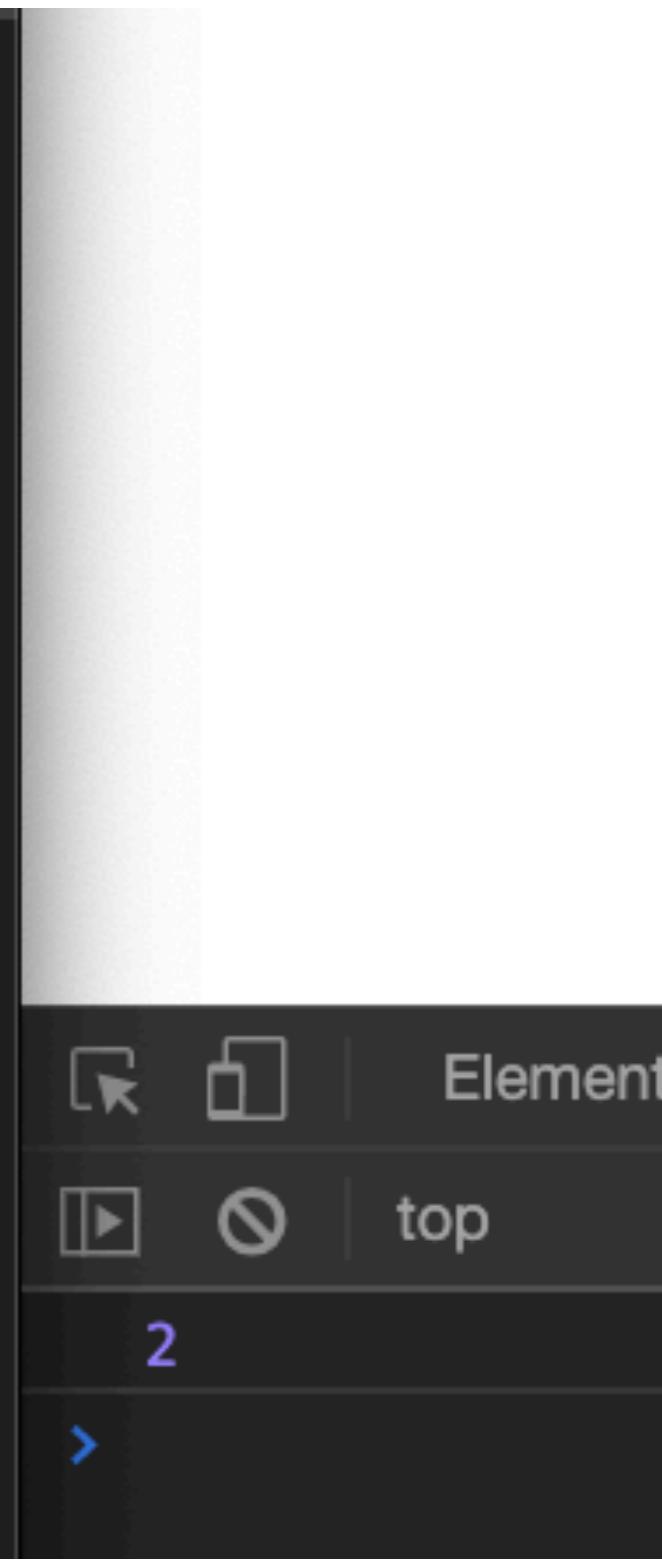
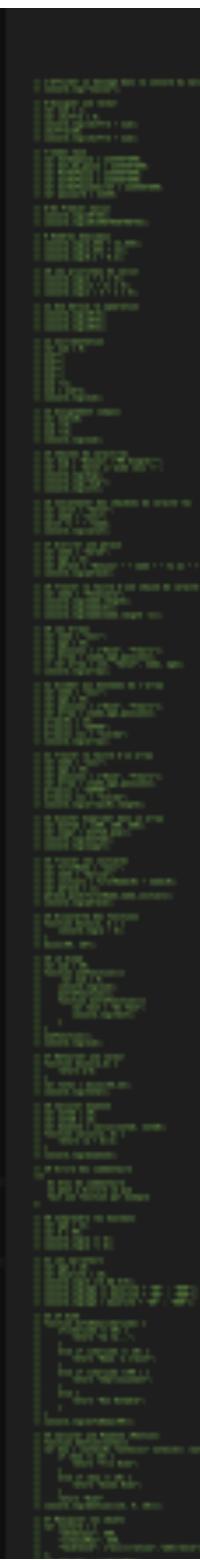
```
var random = Math.random() * 6;  
  
console.log(random);
```



BASE JAVASCRIPT

GÉNÉRER UN NOMBRE ALÉATOIRE

```
3  
4  
5  
6  
7  
8  
9 var random = Math.floor(Math.random() * 6);  
10  
11 console.log(random);  
12  
13  
14  
15  
16 |  
17  
18  
19
```



BASE JAVASCRIPT

GÉNÉRER UN NOMBRE ALÉATOIRE

```
var random = Math.floor(Math.random() * 6+1);  
  
console.log(random);
```



The screenshot shows a browser's developer tools open to the console tab. A code snippet is being run:

```
var random = Math.floor(Math.random() * 6+1);  
  
console.log(random);
```

The output of the code is displayed below the console input area. The output shows the number 6, indicating that the random number generated falls within the specified range (0 to 6). The browser interface includes standard navigation and search controls at the top.

BASE JAVASCRIPT

EXO: MÉLANGER UN ARRAY

- On créer une fonction mélange qui reçoit un tableau de plusieurs chiffres en paramètre
- La fonction a pour but de modifier l'ordre des chiffre dans le tableau
- On console log la fonction en lui passant un tableau de chiffres

JS

BASE JAVASCRIPT

EXO: MÉLANGER UN ARRAY

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following JavaScript code:

```
// Code JavaScript ici
function melange(tab) {
    //on crée une variable temporaire
    var temp = 0;
    //on crée une var random pour stocker un nb aléatoire
    var random = 0;

    for(var i = 0; i < tab.length; i++) {
        random = Math.floor(Math.random() * tab.length);
        temp = tab[i];
        tab[i] = tab[random];
        tab[random] = temp;
    }

    return tab;
}

console.log(melange([1,2,3,4,5,6]));
```

Browser Console: The browser window shows the output of the console.log statement:

```
(6) [5, 2, 3, 2, 2, 5] main.js:26
```

BASE JAVASCRIPT

EXO: MÉLANGER UN ARRAY

```
function melange(tab) {  
    //on crée une variable temporaire  
    var temp = 0;  
    //on crée une var random pour stocker un nb aléatoire  
    var random = 0;  
    for(var i = 0; i < tab.length; i++) {  
        //random va nous donner un nb entre 0 et 5  
        random = Math.floor(Math.random() * tab.length);  
        //On parcourt le tableau et chaque valeur est stockée  
        // dans la variable temporaire provisoirement  
        temp = tab[i];  
        // on remplace la valeur dans le tableau par  
        // La valeur random  
        tab[i] = tab[random];  
        //On remplace la valeur random par la variable qu'on  
        //a cait stocké (on oubli pas d'intervertir)  
        tab[random] = temp;  
    }  
    return tab;  
}  
  
console.log(melange([1,2,3,4,5,6]));
```



BASE JAVASCRIPT

MÉTHODE MAP

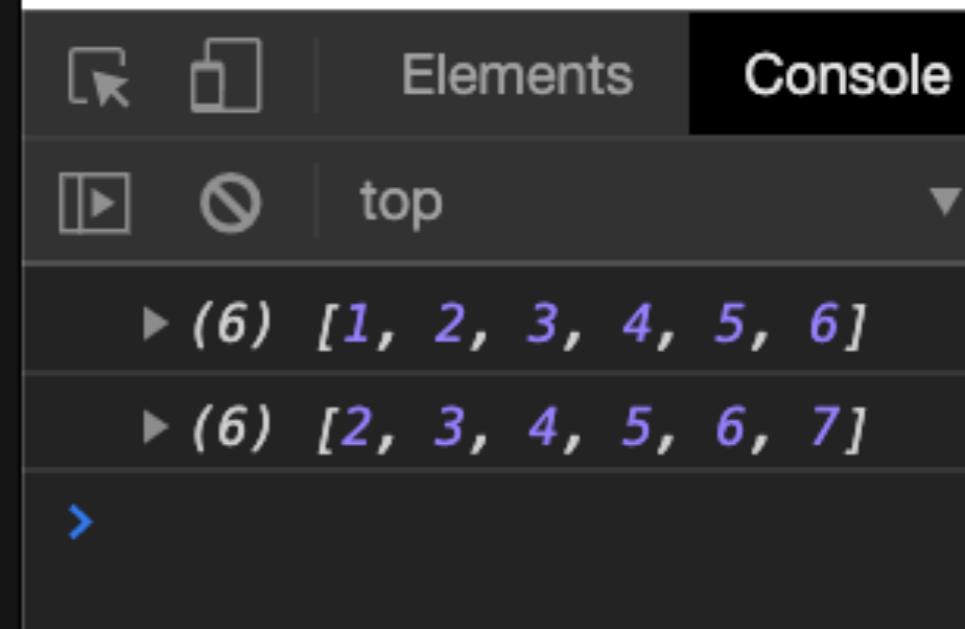
- Pour manipuler des tableaux on a besoin de créer une boucle pour parcourir ce dernier et ainsi manipuler ses éléments.
- Pour se faciliter les vie avec les tableaux, on a la méthode map()
- Map facilite le parcours d'un tableau avec une syntaxe plus allégée.
- Exemple : Je veux ajouter +1 à tous les éléments de mon tableau.

BASE JAVASCRIPT

MÉTHODE MAP

```
var tab = [1,2,3,4,5,6];

var ajouterUn = tab.map(function(nombre){
    return nombre +1;
});
console.log(tab);
console.log(ajouterUn);
```



NORMES ES6 & ES7

JS

ES6 and JS

ECMAScript 6 and Javascript

NORMES ES6 & ES7

NOUVEAUTÉS EN JS

- Les normes récentes de JS
- JS se base sur la norme ECMA Script (ES)
- ES1, ES2, ES3, ES4, ES5, ES6, ES7, ES8, ES9, ES10, ES11
- ES6 version majeure qui a popularisé JS (Maturité Totale)
- Ce n'est plus un « petit langage » pour bidouiller des éléments Front-End (HTML+CSS)
- On peut aussi l'utiliser en Back-End (côté serveur)
 - Relier à une base de données
 - Traiter des données

NORMES ES6 & ES7

NOUVEAUTÉS EN JS

- Chaque année des nouveautés sont ajoutées à JS
- Ces nouveautés sont mise en place selon une norme ECMA Script
- 2015 ES6
- 2016 ES7
- 2017 ES8
- 2018 ES9
- 2019 ES10
- 2020 ES11

NORMES ES6 & ES7

NOUVEAUTÉS EN JS

- JS est un langage qui se développe par itérations
- On garde la base du langage et on ne fait QUE y ajouter de nouvelles fonctionnalités
- On ne peut pas se permettre dans une nouvelle version de retirer des fonctionnalités.
(Problèmes de compatibilité avec des sites ou apps dit « vieux »)

NORMES ES6 & ES7

VAR PROBLEM

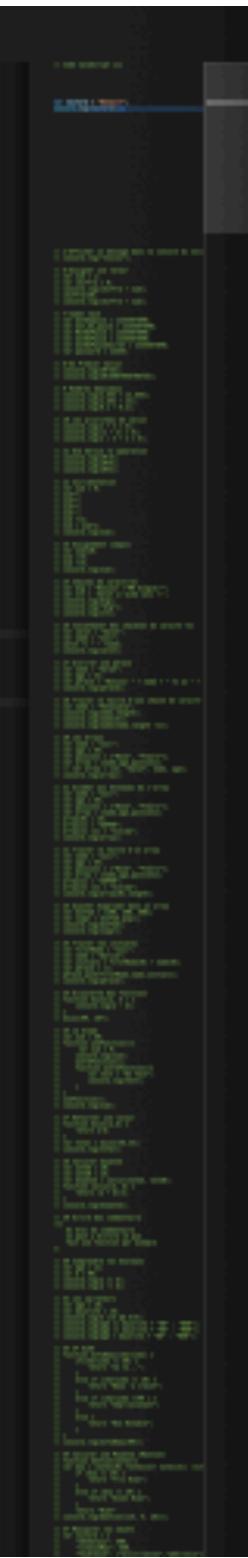
- Var let const
- D'habitude on déclare nos variable avec var
- On peut aussi en déclarer avec let et const
- Var peut poser problème au niveau du scope.
(il a un scope de fonction)
- Rappel scope : Jusqu'ou notre variable va être disponible
- Avec let et const on gère le scope en fonction des blocs dans lesquels on se situe.
(scope de bloc)
- C'est plus de contraintes mais ça permet de garder une logique et un code plus propre

NORMES ES6 & ES7

VAR PROBLEM

js main.js > ...

```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7 var voiture = "Renault";  
8 console.log(voiture);  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```



JS

NORMES ES6 & ES7

VAR PROBLEM

The screenshot shows a browser developer tools interface. On the left, there is a code editor window titled "main.js" containing the following JavaScript code:

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 var voiture = "Renault";
8 console.log(voiture);
9 var voiture = "BMW";
10 console.log(voiture);
11
12
13
14
15
16
17
18
19
```

To the right of the code editor is a panel with several tabs: "Elements", "Console", "Network", "Sources", and "Performance". The "Console" tab is active, showing the output of the console.log statements:

```
Renault
BMW
```

JS

NORMES ES6 & ES7

VAR PROBLEM

JS main.js > ...

```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7  
8 console.log(voiture);  
9 var voiture = "BMW";  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

The screenshot shows a browser's developer tools open to the 'Console' tab. The console output is as follows:

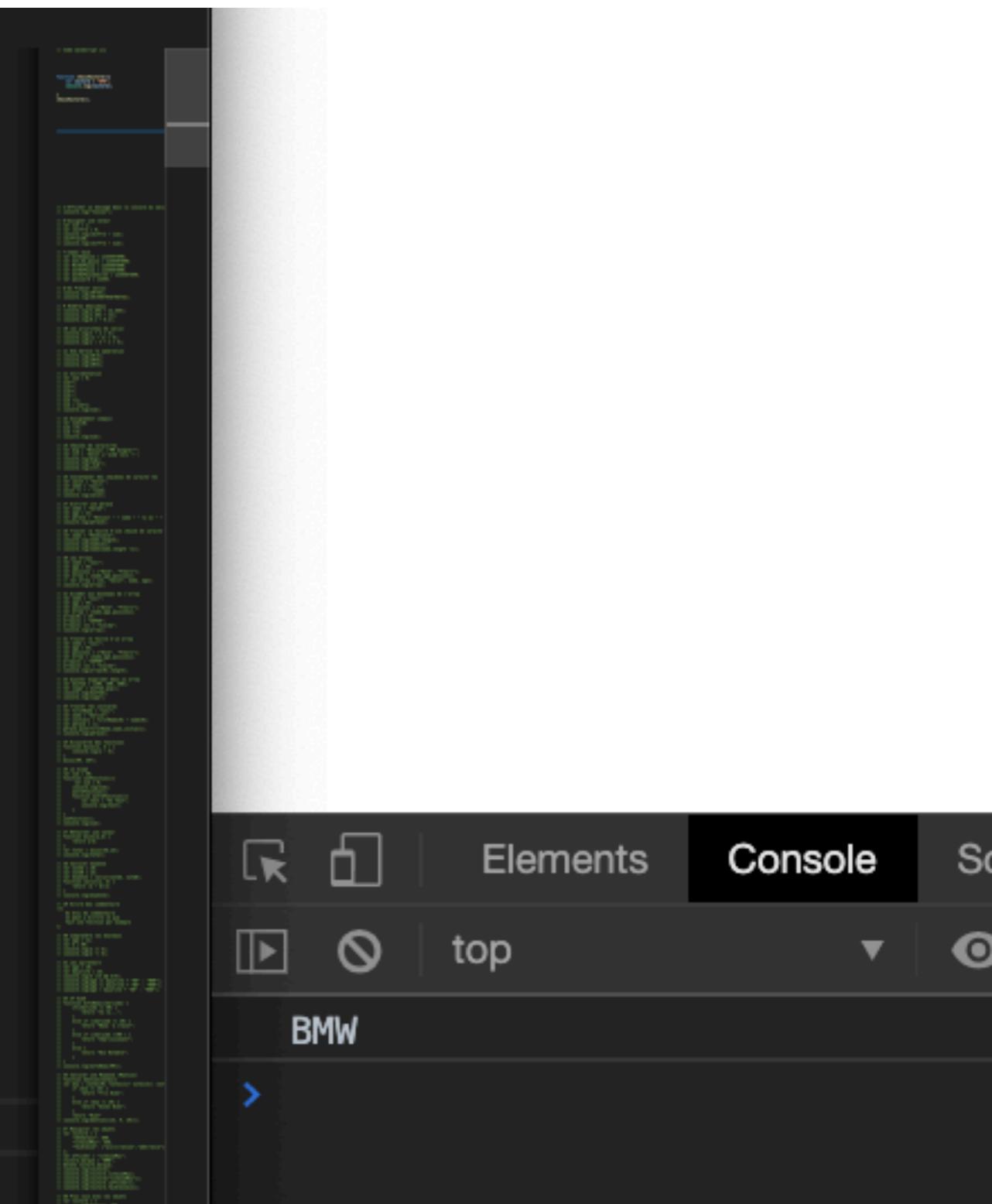
```
undefined
```

NORMES ES6 & ES7

VAR PROBLEM

main.js > ...

```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7 function choixVoiture(){  
8     var voiture = "BMW";  
9     console.log(voiture);  
10 }  
11 choixVoiture();  
12  
13  
14  
15  
16  
17  
18  
19  
20
```



JS

NORMES ES6 & ES7

VAR PROBLEM

The screenshot shows a code editor on the left and a browser window on the right.

Code Editor (main.js):

```
main.js — base
index.html JS main.js ×
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 function choixVoiture(){
8     var voiture = "BMW";
9 }
10 choixVoiture();
11 console.log(voiture);
12
13
14
15
16
17
18
19
20
21
```

Browser Console:

Elements Console Sources »

top Filter

✖ ▶ Uncaught ReferenceError: voiture is not defined
at main.js:12

JS

NORMES ES6 & ES7

VAR PROBLEM

The screenshot shows a browser window with developer tools open. The code editor on the left contains a file named 'main.js' with the following content:

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6 var voiture = "BMW";
7
8 function choixVoiture(){
9     console.log(voiture);
10
11 }
12 choixVoiture();
13
14
15
16
17
18
```

The browser's address bar shows the URL <http://127.0.0.1:8000>. The developer tools panel at the bottom has tabs for 'Elements' and 'Console'. The 'Console' tab is active, showing the output of the script: 'BMW'.

JS

NORMES ES6 & ES7

VAR PROBLEM

The screenshot shows a browser window with developer tools open. The code editor on the left contains a file named 'main.js' with the following content:

```
// Code JavaScript ici
var voiture = "BMW";
if (voiture == "BMW") {
    var vitesse = 800;
}
console.log(vitesse);
```

The browser's address bar shows the URL `http://127.0.0.1:5`. The developer tools console tab is selected, displaying the output:

```
800
```

NORMES ES6 & ES7

VAR PROBLEM

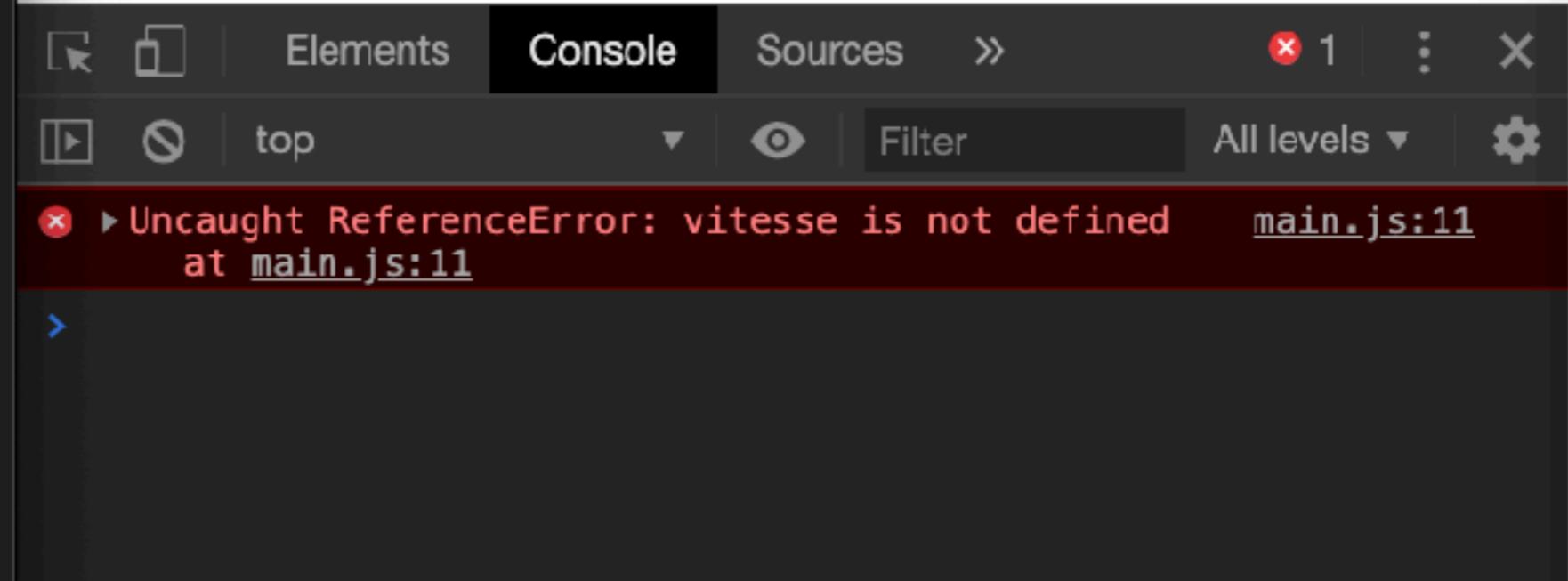
JS

ipt ici

BMW®

= 800;

esse);



JS

NORMES ES6 & ES7

VAR PROBLEM

The screenshot shows a code editor on the left and a browser window on the right.

Code Editor (main.js):

```
// Code JavaScript ici
console.log(voiture);
let voiture = "BMW";
```

Browser:

- Address bar: `http://127.0.0.1:5`
- Console tab is selected.
- Error message: `Uncaught ReferenceError: Cannot access 'voiture' before initialization` at `main.js:7`

NORMES ES6 & ES7

LET & CONST

- Let et const sont 2 autres manières de déclarer des variables
- Leurs scope ont un fonctionnement en mode bloc. { }
- Const permet de déclarer une variable dont la valeur ne changera pas ?

JS

NORMES ES6 & ES7

LET & CONST

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following code:

```
// Code JavaScript ici
let voiture = "BMW";
const model = "Sport";

let voiture = "Citroen";

if (voiture == "BMW") {
    const vitesse = 800;
}

console.log(voiture);
```

Browser Console: The browser's developer tools show a syntax error:

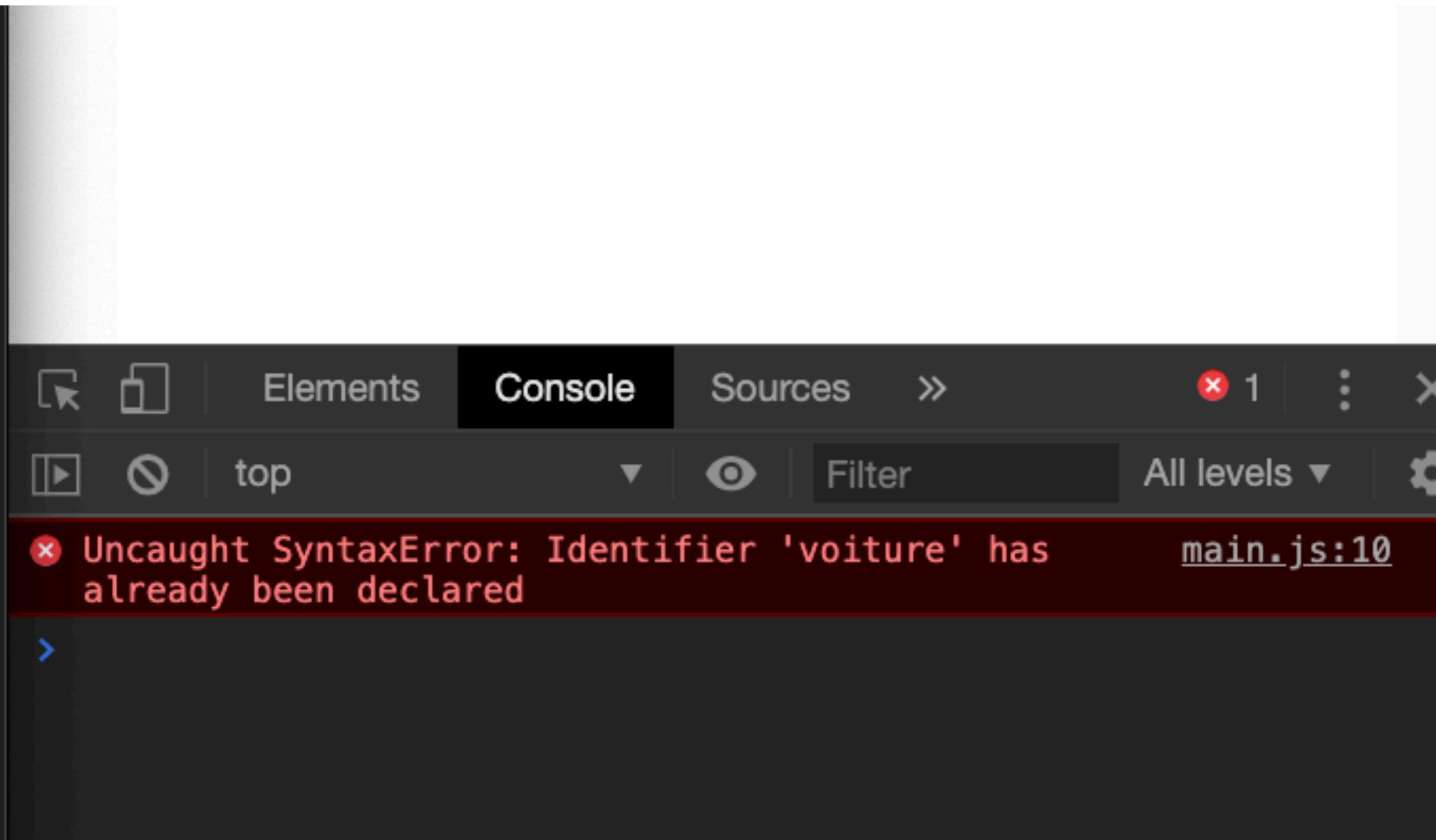
```
Uncaught SyntaxError: Identifier 'voiture' has already been declared
```

Page Address: http://127.0.0.1:5

JS

NORMES ES6 & ES7

LET & CONST



NORMES ES6 & ES7

LET & CONST

```
2  
3  
4  
5  
6  
7 let voiture = "BMW";  
8 const model = "Sport";  
9  
10  
11 if (voiture == "BMW") {  
12     const vitesse = 800;  
13     let voiture = "Citroen";  
14 }  
15  
16 console.log(voiture);  
17  
18  
19
```



NORMES ES6 & ES7

LET & CONST

```
let voiture = "BMW";
const model = "Sport";

if (voiture == "BMW") {
    const vitesse = 800;
    let voiture = "Citroen";
    console.log(voiture);

}

console.log(voiture);
```



JS

NORMES ES6 & ES7

LET & CONST

The screenshot shows a code editor with a dark theme and a browser window side-by-side.

Code Editor (main.js):

```
// Code JavaScript ici
const voiture = "BMW";
voiture = "Citroen";

const model = "Sport";

if (voiture == "BMW") {
    const vitesse = 800;

}
console.log(voiture);
```

Browser Console:

Elements Console Sources main.js:8

Uncaught TypeError: Assignment to constant variable.
at main.js:8

JS

NORMES ES6 & ES7

LET & CONST

The screenshot shows a code editor and a browser window. The code editor has tabs for 'index.html' and 'main.js'. The 'main.js' tab is active, displaying the following code:

```
1 // Code JavaScript ici
2
3
4
5
6
7 const voiture = "BMW";
8 voiture = "Citroen";
9
10 const uneVoiture = {
11     marque : "Mitsubishi",
12     model : "Sport 2",
13     vitesse : 999
14 };
15
16 uneVoiture = {
17     marque : "Mitsubishi",
18     model : "Sport 2",
19     vitesse : 300
20 };
21
22 const model = "Sport";
23
24 if (voiture == "BMW") {
25     const vitesse = 800;
26 }
```

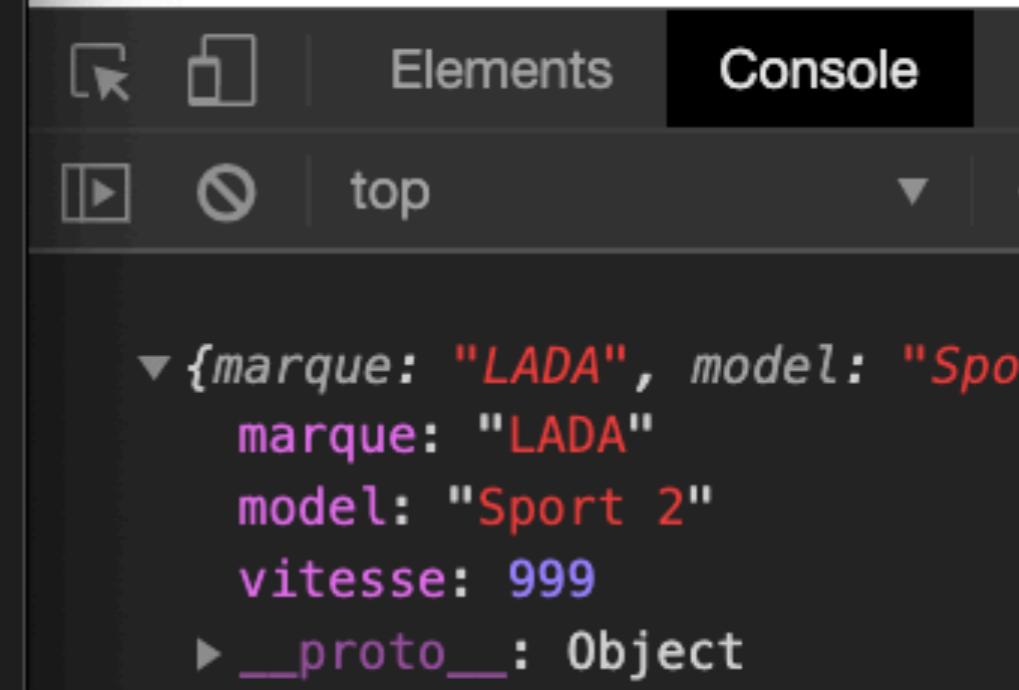
The browser window shows a yellow header bar with the text 'JS'. Below it is a toolbar with various icons. The address bar shows 'http://127.0.0.1:5'. The main content area is blank. At the bottom, the developer tools console is open, showing the following error message:

```
Uncaught TypeError: Assignment to constant variable.
    at main.js:8
```

NORMES ES6 & ES7

LET & CONST

```
7 let voiture = "BMW";
8 voiture = "Citroen";
9
10 const uneVoiture = {
11   marque : "Mitsubishi",
12   model : "Sport 2",
13   vitesse : 999
14 };
15
16 uneVoiture.marque = "LADA"
17
18 const model = "Sport";
19
20 if (voiture == "BMW") {
21   const vitesse = 800;
22 }
23
24 console.log(uneVoiture);
```



NORMES ES6 & ES7

LET & CONST COMMENT CHOISIR

- C'est toujours la guerre sur les forum pour savoir quelle est la meilleure méthode.
- 1 - Privilégier le let et le const, utilisation de var dans « certains » cas pour des if ou des boucles.
- 2- Ne plus utiliser Var privilégier le Const quand on sait que des variable ne se mettront pas à jour et utilisation du let pour tout le reste (+ de contraintes mais meilleur code, logique).

NORMES ES6 & ES7

TEMPLATE STRINGS

- On a déjà vu la concaténation de plusieurs chaînes de caractères.
- Une autre manière de faire c'est d'utiliser le template strings
- `` backticks sur le clavier
- L'intérêt est de simplifier le code
- Et pouvoir accéder à des données à l'intérieur d'une chaîne de caractères.

JS

NORMES ES6 & ES7

TEMPLATE STRINGS

The screenshot shows a browser window with developer tools open. The code editor on the left contains the following JavaScript:

```
ex.html  JS main.js  ×
main.js > ...
// Code JavaScript ici

const uneVoiture = {
    marque : "Mitsubishi",
    model : "Sport 2",
    vitesse : 999
};

const desc = uneVoiture.marque + "," +
uneVoiture.model + " a une vitesse de :" +
uneVoiture.vitesse;

console.log(desc);
```

The browser's address bar shows the URL `http://127.0.0.1:5`. The developer tools' console tab is active, displaying the output of the `console.log` statement:

```
Elements  Console  Sources  Network
top
Mitsubishi,Sport 2 a une vitesse de :999
```

NORMES ES6 & ES7

TEMPLATE STRINGS

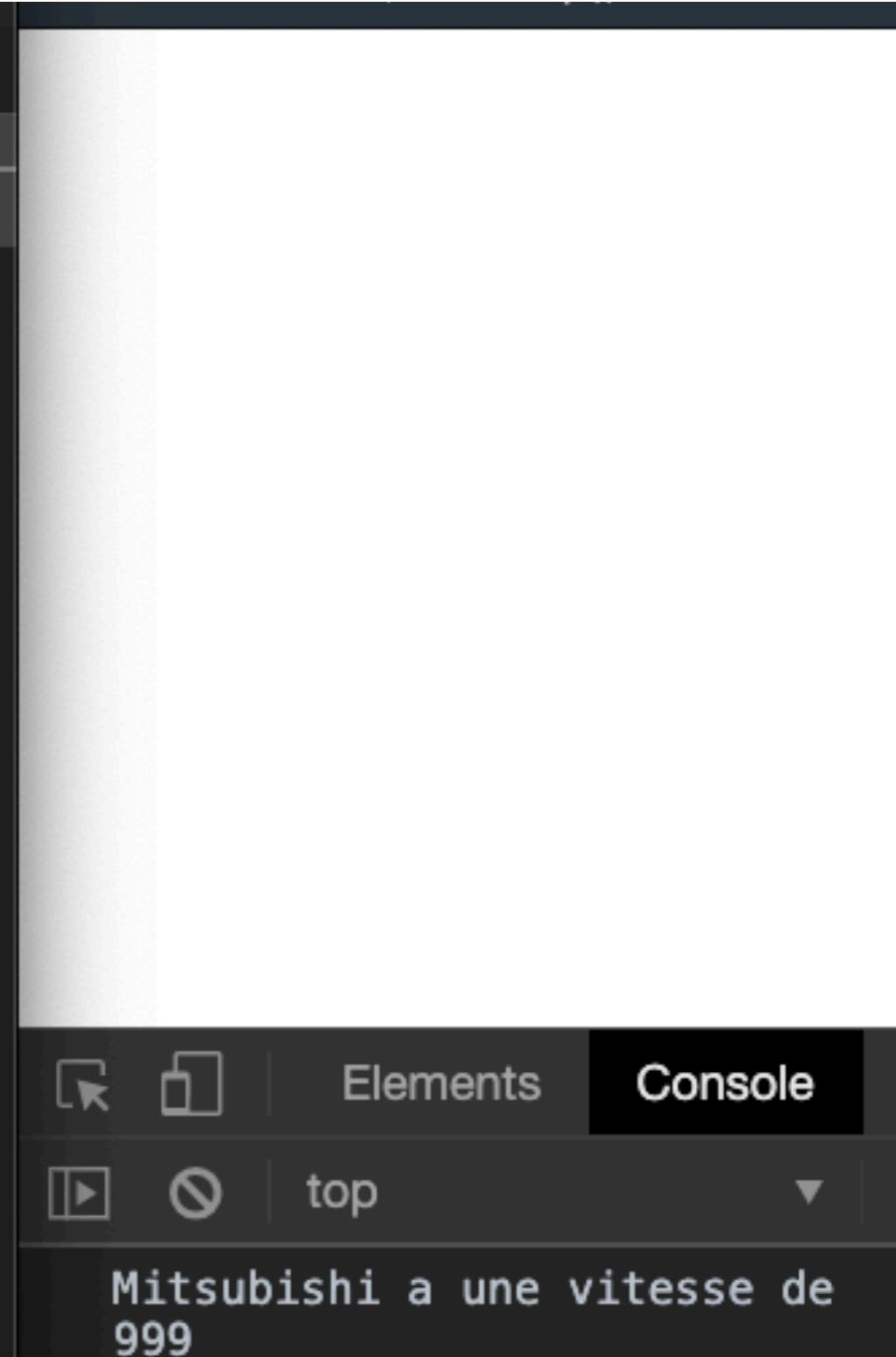
main.js > [e] desc

```
// Code JavaScript ici
```

```
const uneVoiture = {
    marque : "Mitsubishi",
    model : "Sport 2",
    vitesse : 999
};
```

```
const desc =
` ${uneVoiture.marque} a une vitesse de
${uneVoiture.vitesse}`;
```

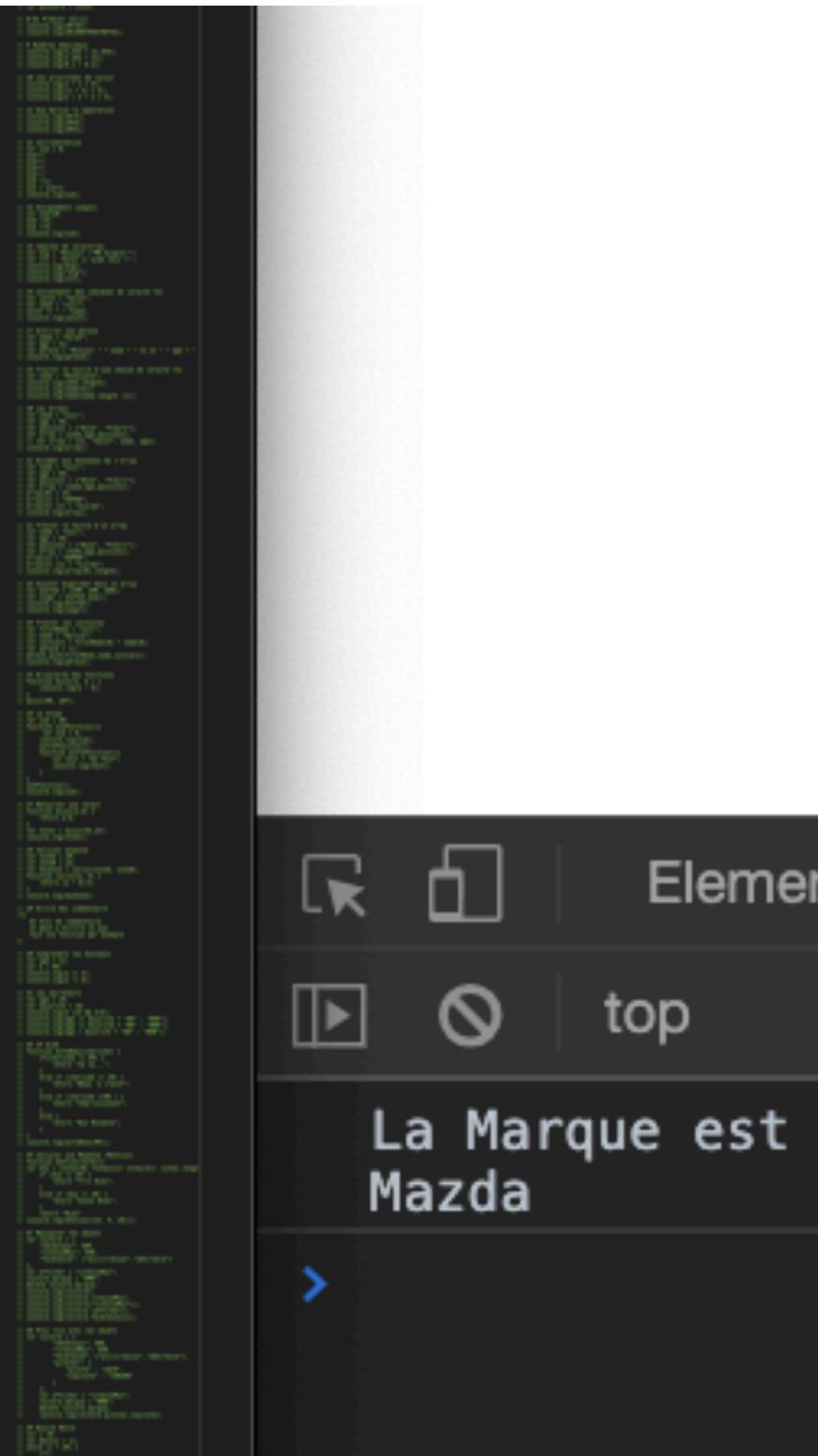
```
console.log(desc);
```



NORMES ES6 & ES7

EXO: TEMPLATE STRINGS

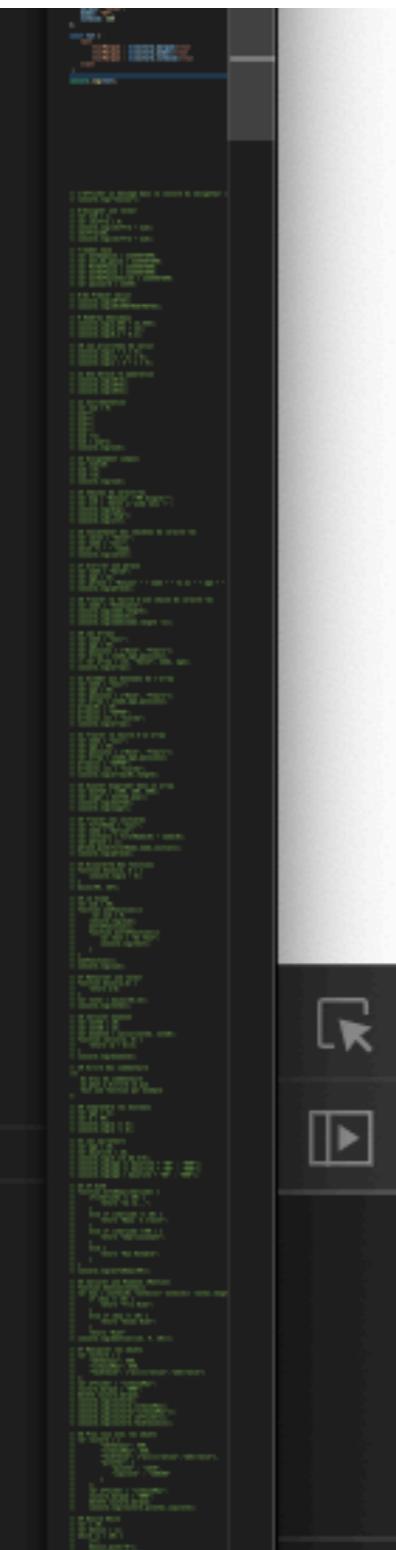
```
6  
7 const voiture = {  
8     marque: "Mazda",  
9     model: "GT",  
10    vitesse: 280  
11};  
12  
13 const txt =  
14 `La Marque est  
15 ${voiture.marque ? voiture.marque : "Aucun"}`;  
16  
17 console.log(txt);  
18  
19  
20  
21  
22  
23  
24  
25  
26
```



NORMES ES6 & ES7

EXO: TEMPLATE STRINGS

```
const voiture = {  
    marque: "Mazda",  
    model: "GT",  
    vitesse: 280  
};  
  
const txt = `  
    <ul>  
        <li>Marque : ${voiture.marque}</li>  
        <li>Marque : ${voiture.model}</li>  
        <li>Marque : ${voiture.vitesse}</li>  
    </ul>  
`;  
  
console.log(txt);
```



NORMES ES6 & ES7

FONCTIONS FLÉCHÉES

- Une nouvelle fonctionnalités depuis ES6
- Un manière d'écrire des fonctions qui est plus courte et plus lisible.
- On remplace notre fonction par =>
- La fonction fléchée retourne automatiquement ce qu'on lui indique après le =>
- Une fonction fléchés est anonyme par défaut (pas de nom)
(Sauf si on la stock dans une variable 😊)

NORMES ES6 & ES7

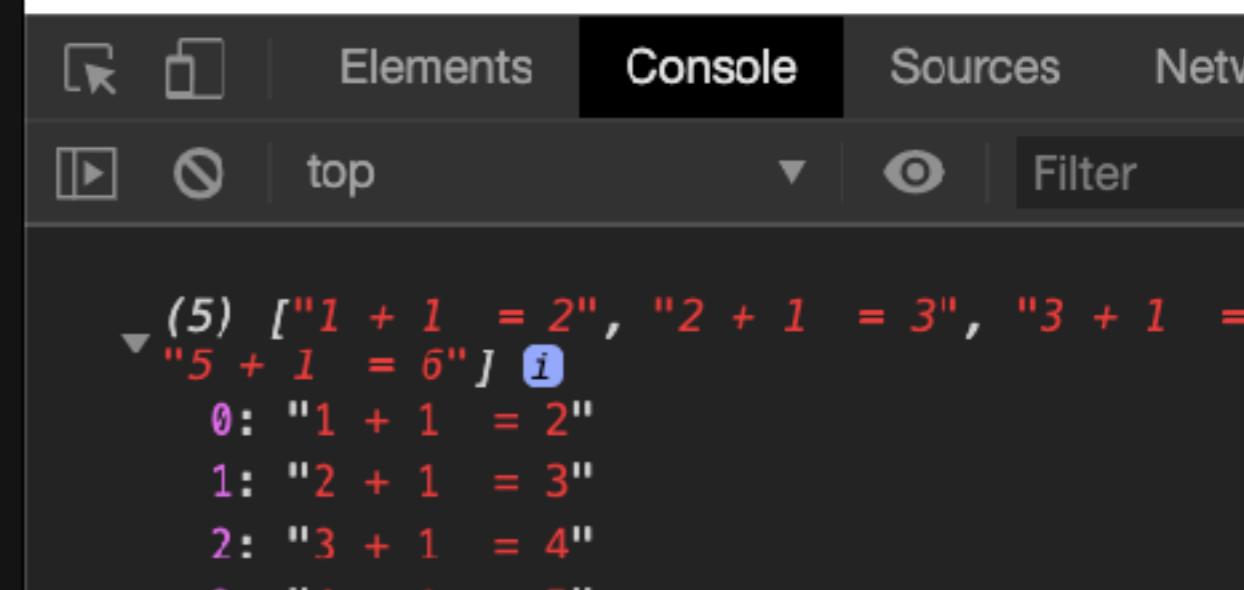
FONCTIONS FLÉCHÉES

```
main.js > ...
// Code JavaScript ici
```

```
const tab = [1,2,3,4,5];

const plus = tab.map(function(nombre) {
  return `${nombre} + 1 = ${nombre + 1}`;
});

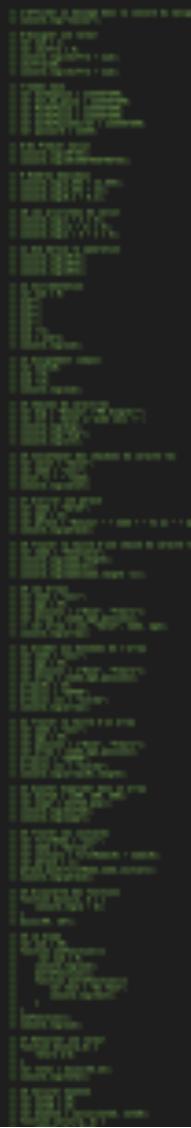
console.log(plus);
```



NORMES ES6 & ES7

FONCTIONS FLÉCHÉES

```
3  
4  
5  
6  
7 const tab = [1,2,3,4,5];  
8  
9 const plus = tab.map(nombre) => {  
10 |   return `${nombre} + 1 = ${nombre + 1}`;  
11 };  
12  
13 // const plus = tab.map(function(nombre) {  
14 //   return `${nombre} + 1 = ${nombre + 1}`;  
15 // });  
16  
17 console.log(plus);  
18  
19  
20  
21  
22 |
```



NORMES ES6 & ES7

FONCTIONS FLÉCHÉES

```
/ const tab = [1,2,3,4,5];
8
9 const plus = tab.map(nombre => `${nombre} + 1 = ${nombre + 1}`);
0
1 // const plus = tab.map((nombre) => {
2 //   return `${nombre} + 1 = ${nombre + 1}`;
3 // });
4
5 // const plus = tab.map(function(nombre) {
6 //   return `${nombre} + 1 = ${nombre + 1}`;
7 // });
8
9 console.log(plus);
0
1
2
3
4
5
```

NORMES ES6 & ES7

FONCTIONS FLÉCHÉES

```
const tab = [1,2,3,4,5];

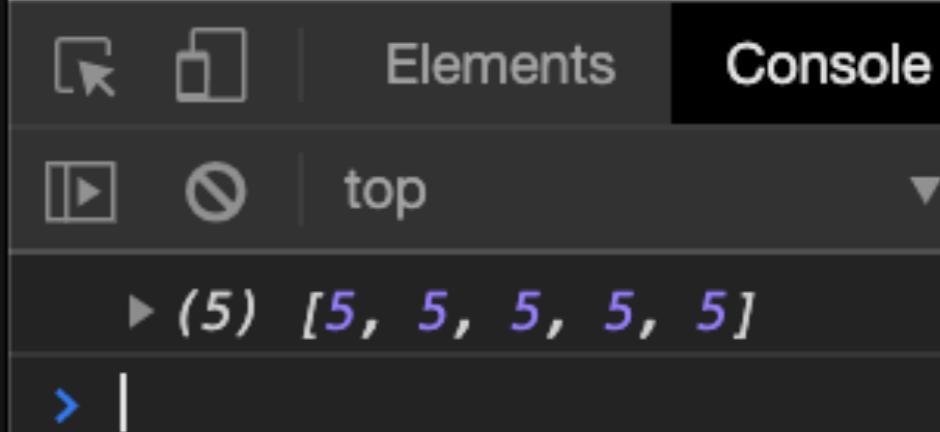
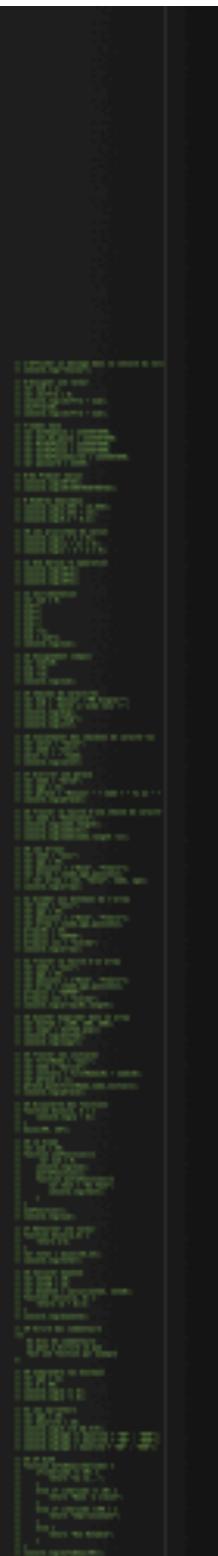
const plus = tab.map(() => 5);

// const plus = tab.map(nombre => `${nombre}`)

// const plus = tab.map((nombre) => {
//     return `${nombre} + 1` = ${nombre + 1}
// });

// const plus = tab.map(function(nombre) {
//     return `${nombre} + 1` = ${nombre + 1}
// });

console.log(plus);
```



NORMES ES6 & ES7

FONCTIONS FLÉCHÉES

```
const tab = [1,2,3,4,5];

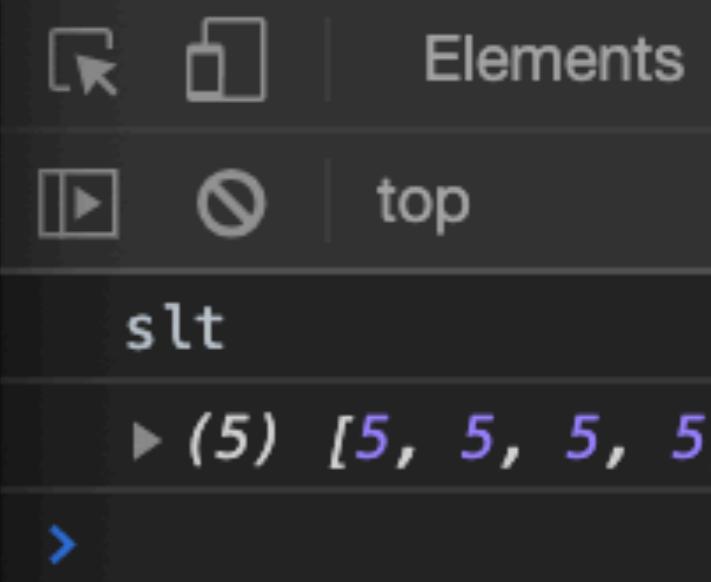
const plus = tab.map(() => 5);

const maFonction = () => console.log("slt");

maFonction();
// const plus = tab.map(nombre => `${nombre}`)

// const plus = tab.map((nombre) => {
//   return `${nombre} + 1` = ${nombre + 1}
// });

// const plus = tab.map(function(nombre) {
```



NORMES ES6 & ES7

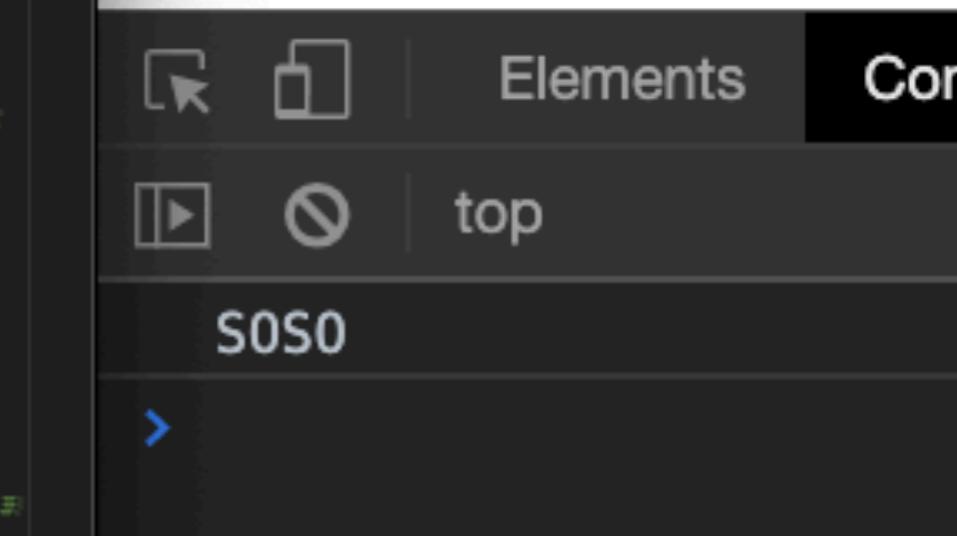
PARAMÈTRE PAR DÉFAUT DANS 1 FONCTION

- Nouveautés depuis ES6
- On a vu comment se servir des fonctions.
- Quand une fonction attend qu'on lui passe un paramètre pour nous retourner un résultat.
- Que faire si elle ne reçoit pas de paramètre ?

NORMES ES6 & ES7

PARAMÈTRE PAR DÉFAUT DANS 1 FONCTION

```
function disNom(nom) {  
    console.log(nom);  
}  
  
disNom("SOSO");
```

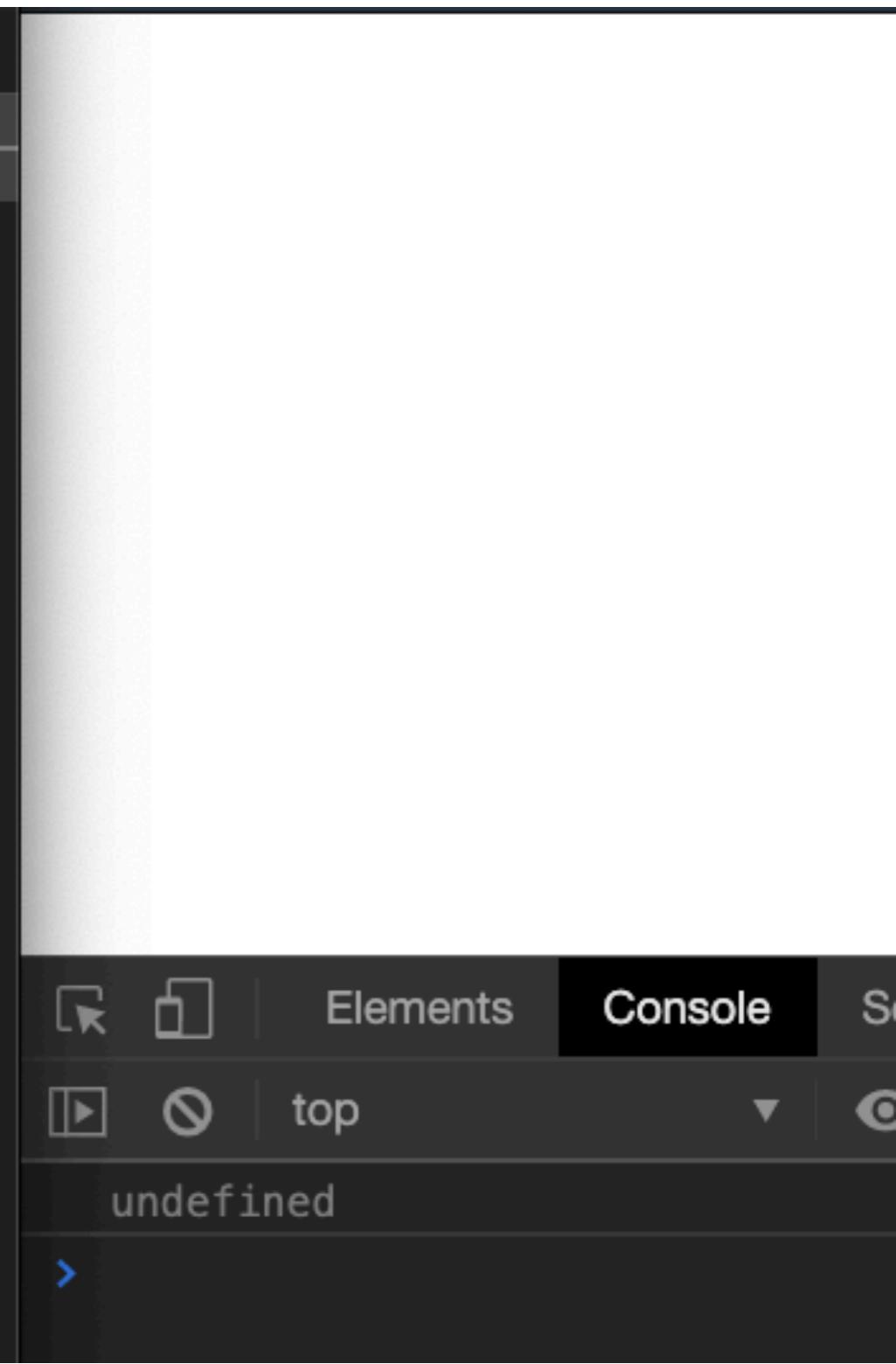


NORMES ES6 & ES7

PARAMÈTRE PAR DÉFAUT DANS 1 FONCTION

JS main.js > ...

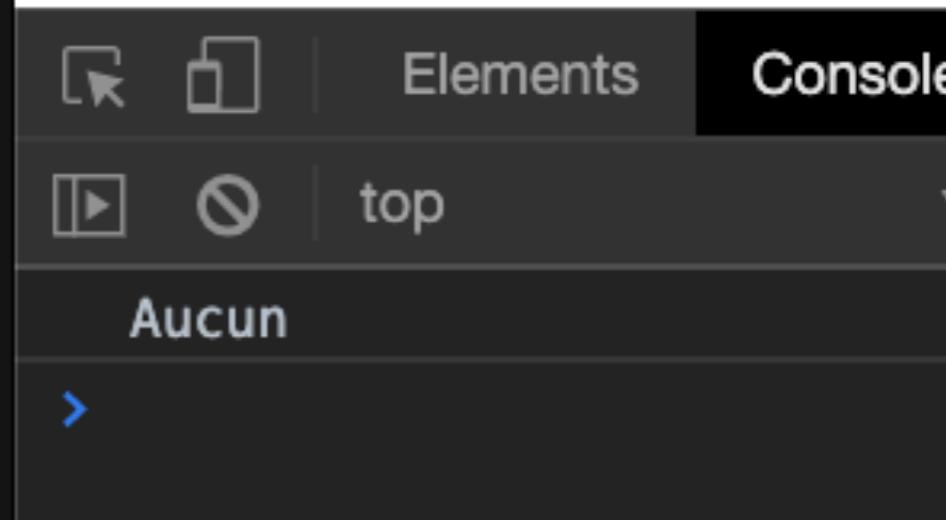
```
1 // Code JavaScript ici
2
3
4
5
6
7
8
9 function disNom(nom) {
10   console.log(nom);
11 }
12
13 disNom();
14
15
16
17
18
19
20
```



NORMES ES6 & ES7

PARAMÈTRE PAR DÉFAUT DANS 1 FONCTION

```
function disNom(nom = "Aucun") {  
    console.log(nom);  
}  
  
disNom();
```



NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

- Permet de créer des variables à la volée.
- Quand on manipule des objets qui contiennent énormément d'éléments.
- { }

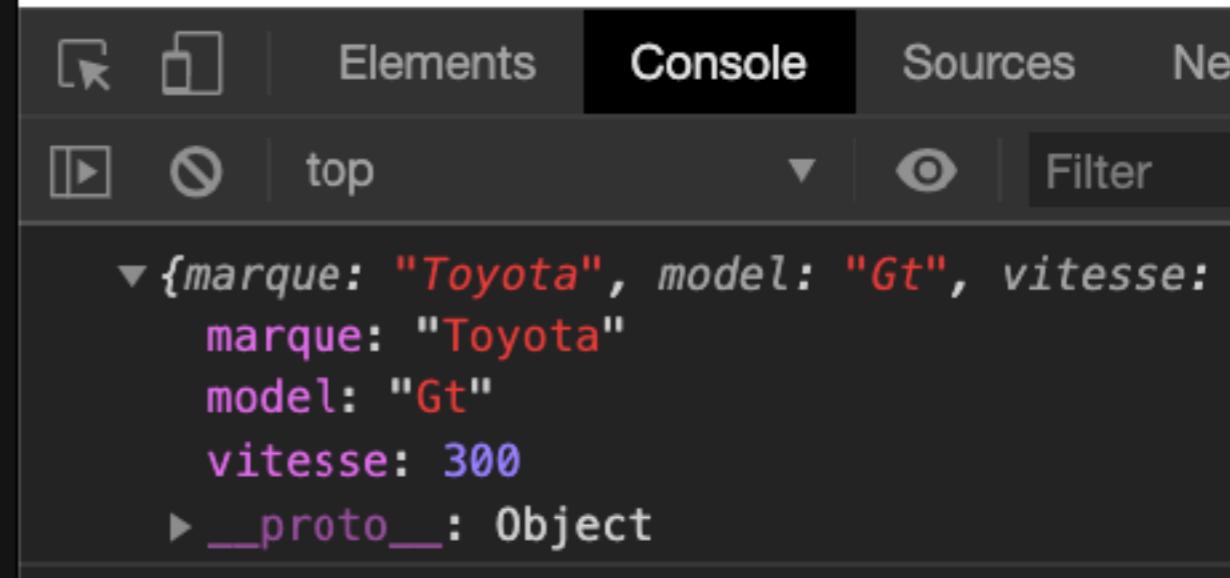
NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

main.js > ...

```
// Code JavaScript ici
```

```
const voiture = {  
    marque: "Toyota",  
    model: "Gt",  
    vitesse: 300  
}  
console.log(voiture);
```



NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

The screenshot shows a browser window with developer tools open. The code editor on the left contains the following JavaScript:

```
// Code JavaScript ici  
  
const voiture = {  
    marque: "Toyota",  
    model: "Gt",  
    vitesse: 300  
}  
console.log(model);
```

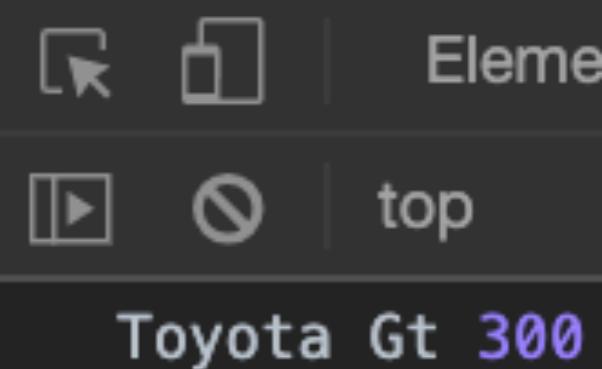
The browser's address bar shows the URL `http://127.0.0.1:5`. The developer tools console tab is selected, displaying the error message:

Uncaught ReferenceError: model is not defined
at main.js:10

NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

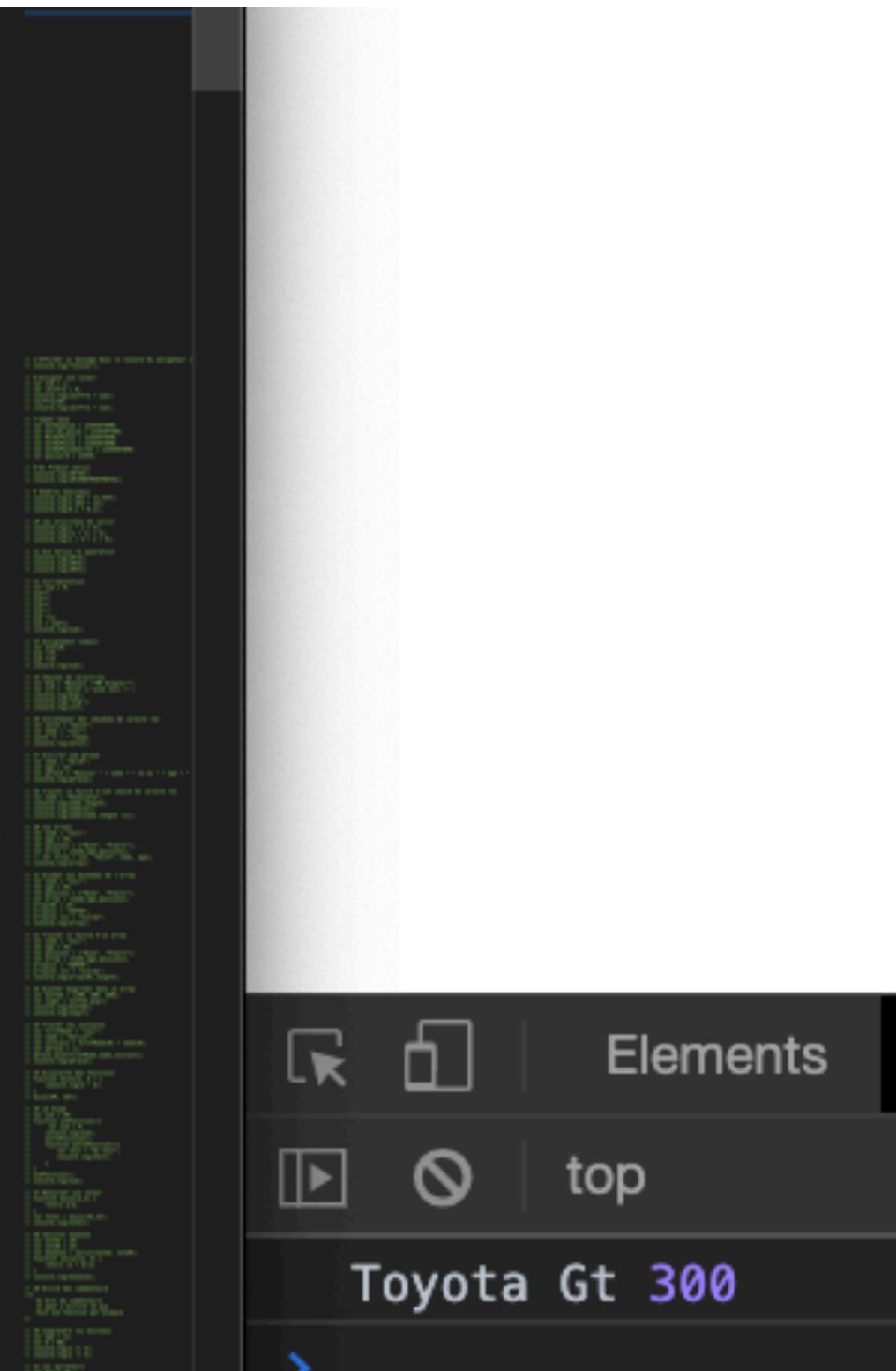
```
const voiture = {  
    marque: "Toyota",  
    model: "Gt",  
    vitesse: 300  
}  
  
const { marque, model, vitesse } = voiture;  
console.log(marque, model, vitesse);
```



NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

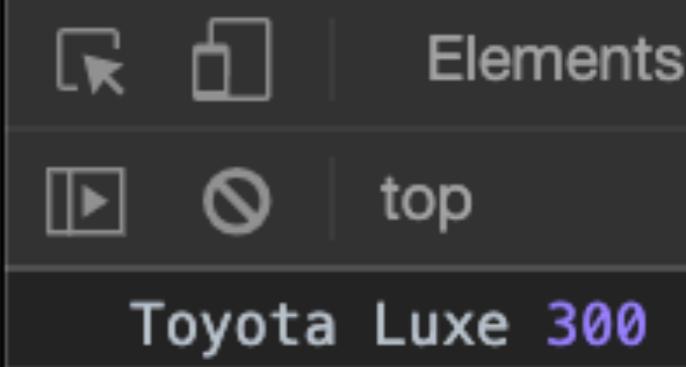
```
const voiture = {  
    marque: "Toyota",  
    model:"Gt",  
    vitesse: 300  
}  
  
const { marque, model: prix, vitesse } = voiture;  
console.log(marque, prix, vitesse);
```



NORMES ES6 & ES7

DESTRUCTURING D'UN OBJET

```
const voiture = {  
    marque: "Toyota",  
    // model:"Gt",  
    vitesse: 300  
}  
  
const { marque, model = "Luxe", vitesse } = voiture;  
console.log(marque, model, vitesse);
```



NORMES ES6 & ES7

DESTRUCTURING D'UN ARRAY

- Parfois les données ne sont pas toujours sous forme d'objets, mais sous forme de tableaux.
- On peut aussi utiliser le destructuring avec les tableaux pour faciliter la manipulation des données.
- []

NORMES ES6 & ES7

DESTRUCTURING D'UN ARRAY

// Code JavaScript ici

```
const notes = [12, 17, 8, 9];

const [ français, philo, Lv1, Lv2] = notes;

console.log(français);
```

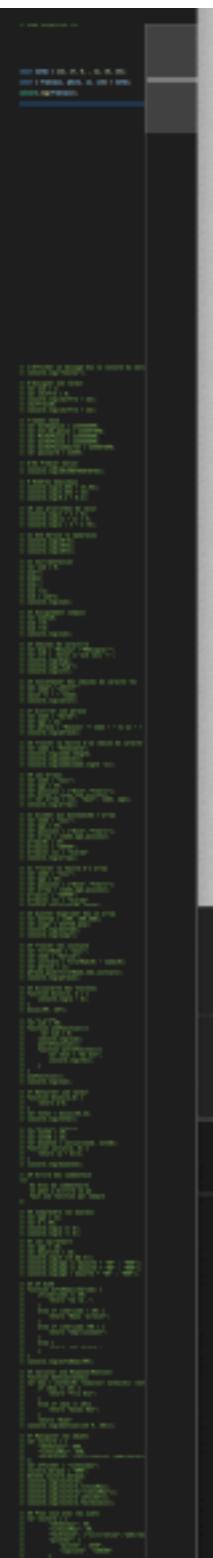


NORMES ES6 & ES7

DESTRUCTURING D'UN ARRAY

// Code JavaScript ici

```
const notes = [12, 17, 8, 9, 11, 15, 14];  
  
const [ français, philo, Lv1, Lv2] = notes;  
  
console.log(français);
```



NORMES ES6 & ES7

INTERVERTIR DES VALEURS

- On a déjà vu comment intervertir des valeurs dans un tableau
- En utilisant une variable temporaire pour gérer l'échange

NORMES ES6 & ES7

INTERVERTIR DES VALEURS

```
1 // Code JavaScript ici  
2  
3  
4  
5  
6  
7  
8 let maCarte = "Ronflex";  
9 let taCarte = "Magmar";  
10  
11 let temp = maCarte;  
12 maCarte = taCarte;  
13 taCarte = temp;  
14  
15 console.log(maCarte, taCarte);  
16  
17  
18
```



↶	⤵	Elements
⤶	🚫	top
Magmar Ronflex		

NORMES ES6 & ES7

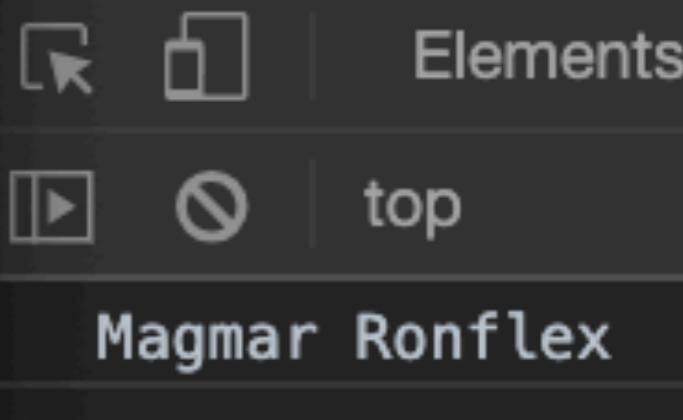
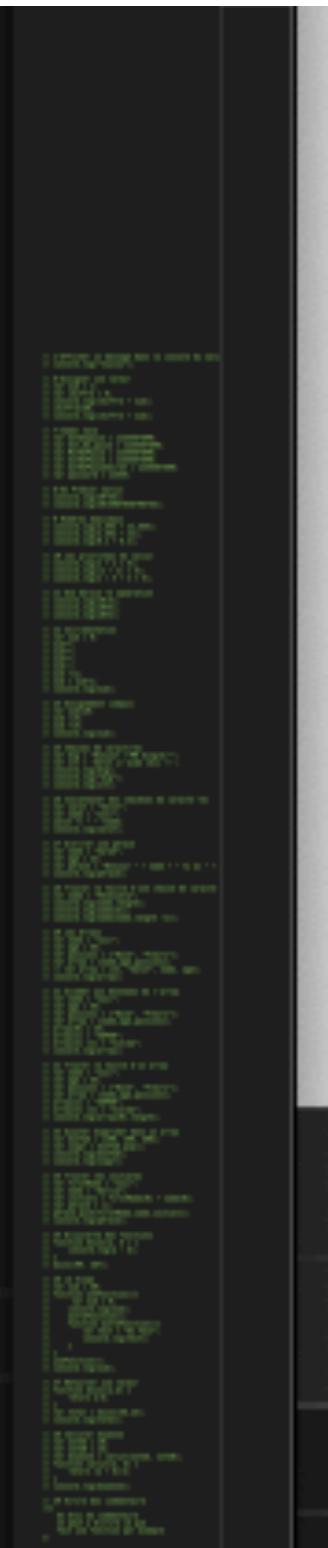
INTERVERTIR DES VALEURS

```
let maCarte = "Ronflex";
let taCarte = "Magmar";

[maCarte, taCarte] = [taCarte, maCarte];

// let temp = maCarte;
// maCarte = taCarte;
// taCarte = temp;

console.log(maCarte, taCarte);
```



NORMES ES6 & ES7

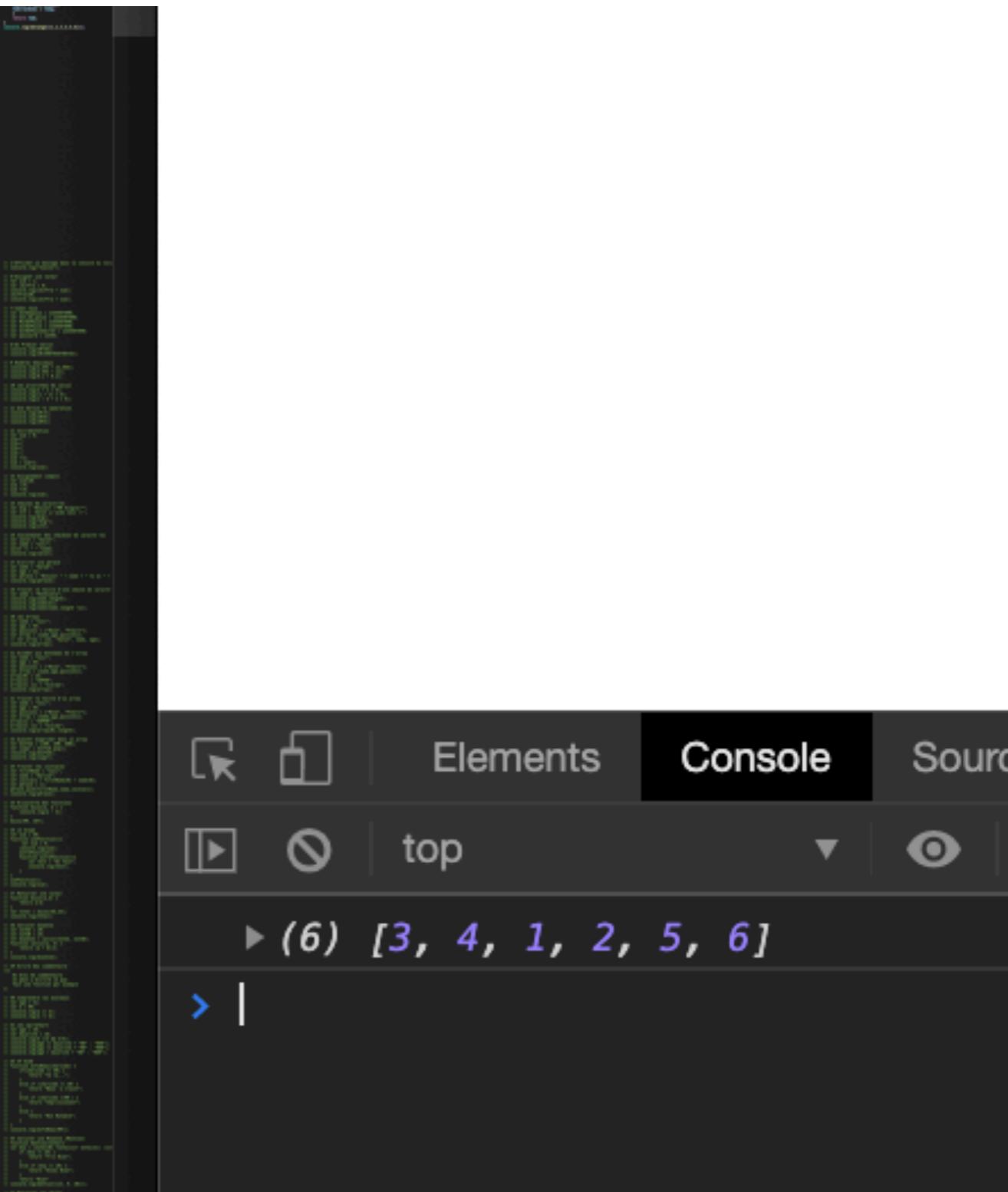
EXO: MÉLANGER UN ARRAY (V.ES6)

- Retransformer le code utilisé pour mélanger des tableaux
- Avec les nouvelles méthodes ES6
- Un code + simple avec moins de lignes
- Un code + lisible.

NORMES ES6 & ES7

EXO: MÉLANGER UN ARRAY (V.ES6)

```
//46 Mélanger un array
function melange(tab) {
    var temp = 0;
    var random = 0;
    for(var i = 0; i < tab.length; i++) {
        random = Math.floor(Math.random() * tab.
        temp = tab[i];
        tab[i] = tab[random];
        tab[random] = temp;
    }
    return tab;
}
console.log(melange([1,2,3,4,5,6]));
```



NORMES ES6 & ES7

EXO: MÉLANGER UN ARRAY (V.ES6)

```
//46 Mélanger un array ES6
function melange(tab) {
    for(var i = 0; i < tab.length; i++) {
        const random = Math.floor(Math.random() * tab.length);
        [tab[i], tab[random]] = [tab[random], tab[i]];
    }
    return tab;
}
console.log(melange([1,2,3,4,5,6]));
```

NORMES ES6 & ES7

SPREAD OPERATOR

- Depuis ES on a un nouvel opérateur (en plus de * + - /)
- Spread Operator on peut l'utiliser sur n'importe quel élément qui est ittéable (ou l'on peut ajouter des chose) ex : liste, tableaux, chaîne de caractères, objets etc...
- ...
- Sert à stocker et manipuler tout ce qu'il trouve dans le paramètre qu'on va lui passer.

NORMES ES6 & ES7

SPREAD OPERATOR

The screenshot shows a development environment with two tabs: 'index.html' and 'main.js'. The 'main.js' tab contains the following code:

```
// Code JavaScript ici
const groupe1 = ["José", "Hélène", "Nat"];
const groupe2 = ["Chris", "Nico", "Joana"];

console.log(...groupe1, ...groupe2);
console.log([...groupe1, ...groupe2]);
```

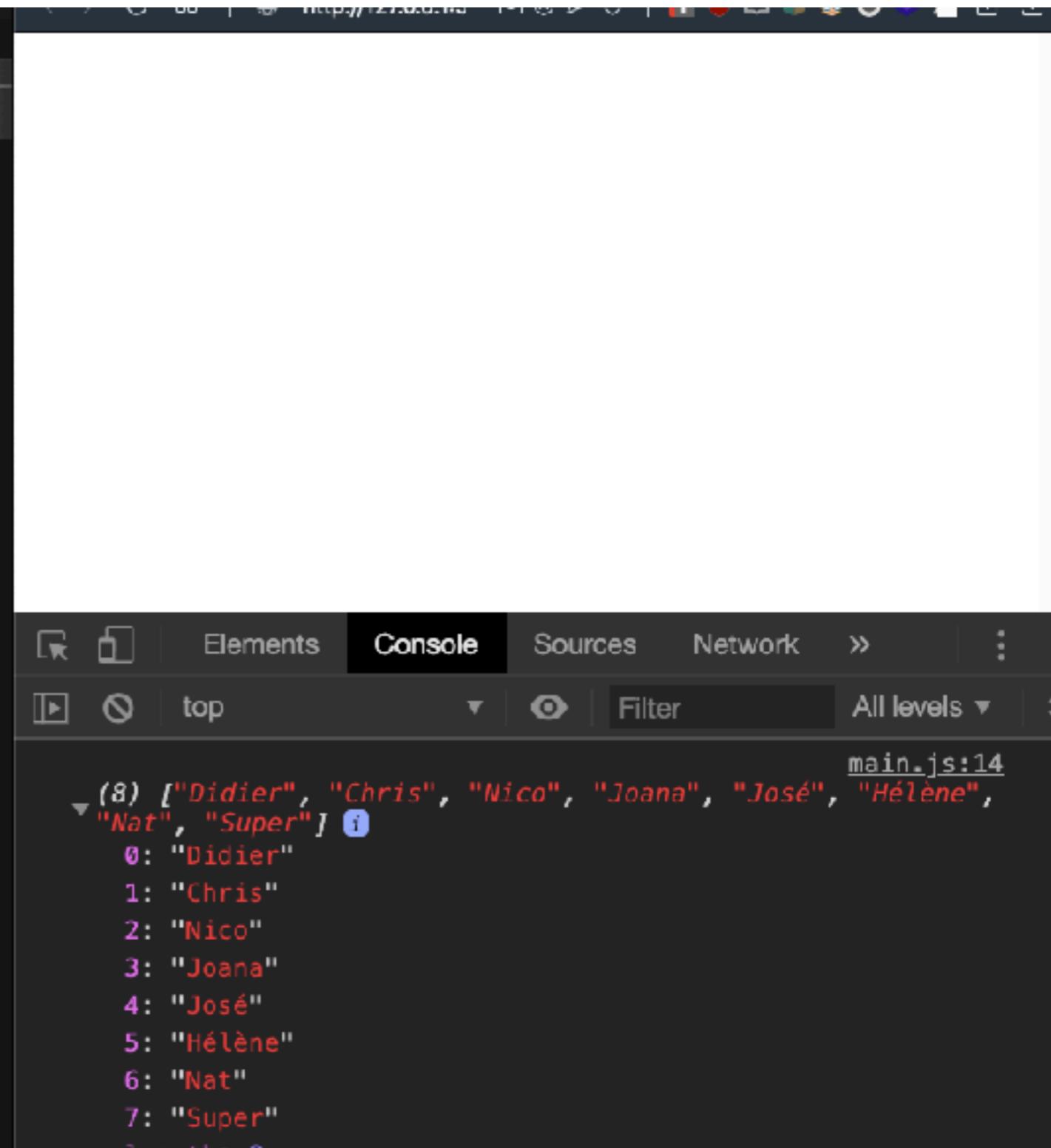
To the right, a browser window displays the output of the code. The console tab is active, showing the following results:

```
José Hélène Nat Chris Nico Joana
main.js:10
main.js:11
▼ (6) ["José", "Hélène", "Nat", "Chris", "Nico", "Joana"]
  0: "José"
  1: "Hélène"
  2: "Nat"
  3: "Chris"
  4: "Nico"
  5: "Joana"
  length: 6
  ▶ __proto__: Array(0)
```

NORMES ES6 & ES7

SPREAD OPERATOR

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6
7 const groupe1 = ["José", "Hélène", "Nat"];
8 const groupe2 = ["Chris", "Nico", "Joana"];
9
10 const liste = ["Didier", ...groupe2, ...groupe1];
11 const liste2 = liste;
12 liste2.push("Super");
13
14 console.log(liste);
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```



JS

NORMES ES6 & ES7

SPREAD OPERATOR

The screenshot shows a development environment with two tabs: 'index.html' and 'main.js'. The 'main.js' tab contains the following code:

```
// Code JavaScript ici
const groupe1 = ["José", "Hélène", "Nat"];
const groupe2 = ["Chris", "Nico", "Joana"];

const liste = ["Didier", ...groupe2, ...groupe1];
const liste2 = [...liste];
liste2.push("Super");

console.log(liste);
```

To the right, a browser window displays the output of the code. The developer tools' 'Console' tab is active, showing the result of the `console.log(liste);` statement:

```
main.js:14
(7) ["Didier", "Chris", "Nico", "Joana", "José", "Hélène", "Nat"]
  0: "Didier"
  1: "Chris"
  2: "Nico"
  3: "Joana"
  4: "José"
  5: "Hélène"
  6: "Nat"
length: 7
```

JS

NORMES ES6 & ES7

SPREAD OPERATOR

The screenshot shows a development environment with two tabs: 'index.html' and 'main.js'. The 'main.js' tab contains the following code:

```
// Code JavaScript ici
const groupe1 = ["José", "Hélène", "Nat"];
const groupe2 = ["Chris", "Nico", "Joana"];

const liste = ["Didier", ...groupe2, ...groupe1];
const liste2 = [...liste];
liste2.push("Super");

console.log(liste2);
```

To the right, a browser window displays the output of the code. The address bar shows 'http://127.0.0.1:5'. The browser's developer tools are open, specifically the 'Console' tab, which shows the following output:

```
(8) ["Didier", "Chris", "Nico", "Joana", "José", "Hélène",
  "Nat", "Super"] i
  0: "Didier"
  1: "Chris"
  2: "Nico"
  3: "Joana"
  4: "José"
  5: "Hélène"
  6: "Nat"
  7: "Super"
length: 8
```

The 'main.js:14' reference in the console output indicates the line number where the 'console.log' statement is located.

JS

NORMES ES6 & ES7

EXEMPLE: SPREAD OPERATOR

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The file `main.js` contains the following JavaScript code:

```
// Code JavaScript ici
const groupe1 = ["José", "Hélène", "Nat"];
const groupe2 = ["Chris", "Nico", "Joana"];

const liste = [...groupe2, "Didier"];
liste.push(groupe1);

console.log(liste);
```

Browser Console: The browser's developer tools show the output of the `console.log` statement. The output is an array of 5 elements:

```
main.js:13
(5) ["Chris", "Nico", "Joana", "Didier", Array(3)]
  0: "Chris"
  1: "Nico"
  2: "Joana"
  3: "Didier"
  4: Array(3)
    0: "José"
    1: "Hélène"
    2: "Nat"
    length: 3
```

JS

NORMES ES6 & ES7

EXEMPLE: SPREAD OPERATOR

The screenshot shows a code editor on the left and a browser window on the right.

Code Editor (main.js):

```
// Code JavaScript ici
const groupe1 = ["José", "Hélène", "Nat"];
const groupe2 = ["Chris", "Nico", "Joana"];

const liste = [...groupe2,"Didier"];
liste.push(...groupe1);

console.log(liste);
```

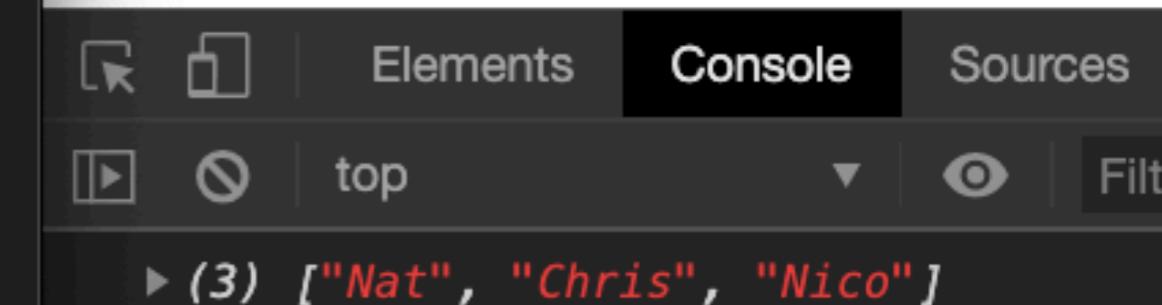
Browser Console Output:

```
main.js:13
(7) ["Chris", "Nico", "Joana", "Didier", "José", "Hélène",
"Nat"]
  0: "Chris"
  1: "Nico"
  2: "Joana"
  3: "Didier"
  4: "José"
  5: "Hélène"
  6: "Nat"
length: 7
```

NORMES ES6 & ES7

EXAMPLE: SPREAD OPERATOR

```
const liste =  
["José", "Hélène", "Nat", "Chris", "Nico"];  
  
const [ami1, ami2, ...lesGensPasCool] = liste;  
  
console.log(lesGensPasCool);
```



NORMES ES6 & ES7

ES7

- Nouvel opérateur pour calculer des puissances
`8 ** 9`
- La fonction `includes` qui permet de vérifier si une variable est contenue dans un objet. (cela nous renvoi true ou false)

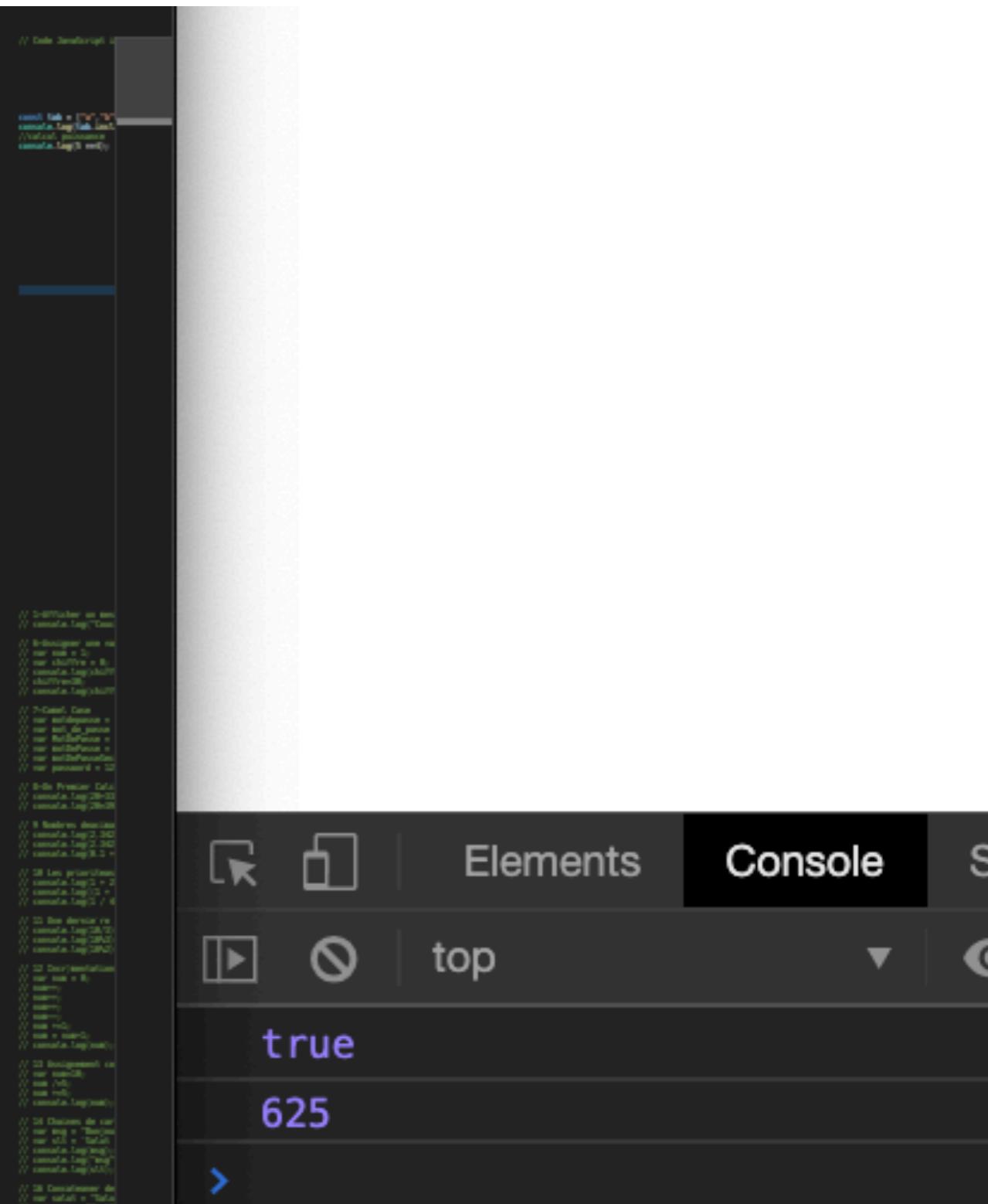
NORMES ES6 & ES7

ES7

JS

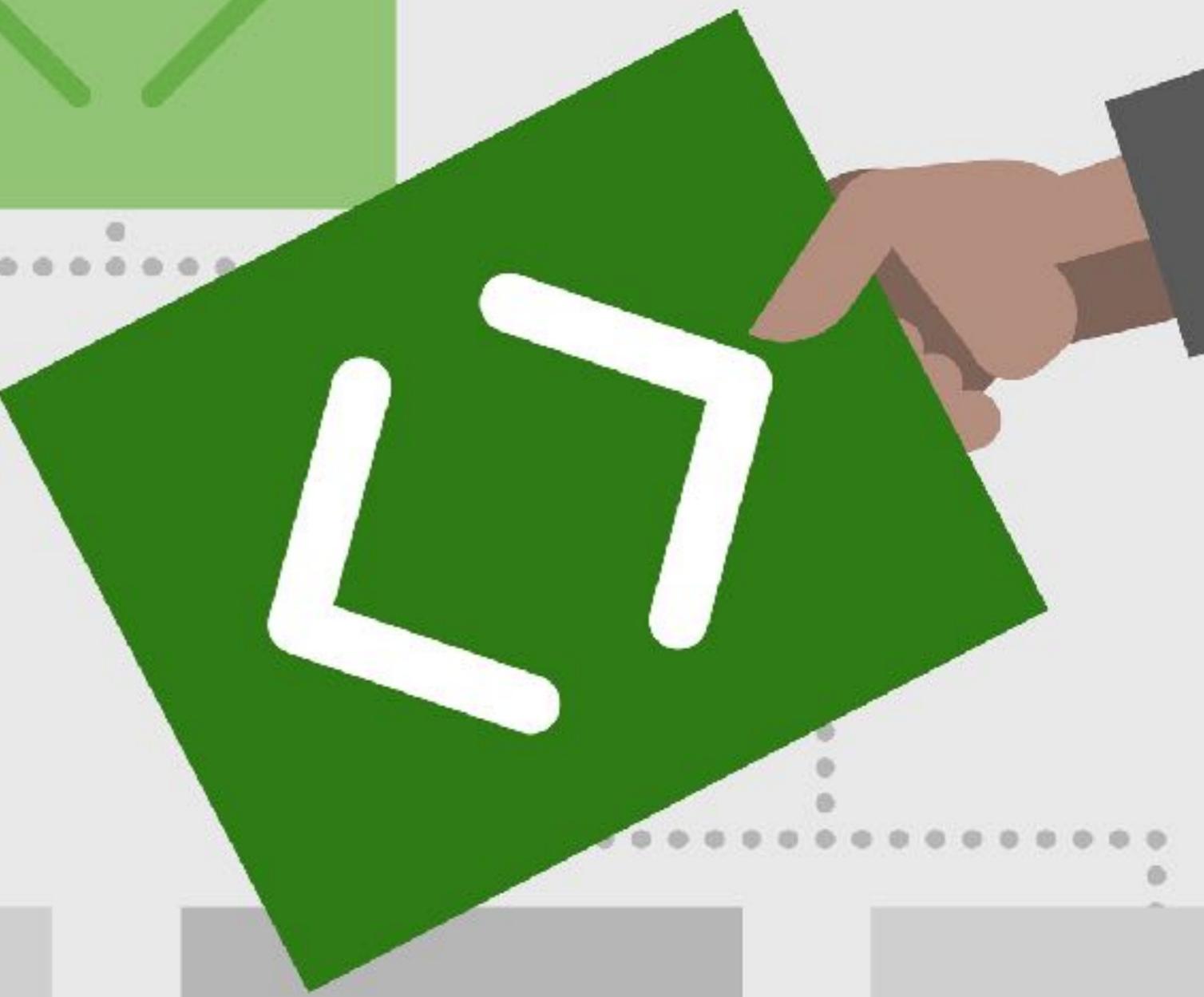
```
// Code JavaScript ici
```

```
const tab = ["a","b","c"];
console.log(tab.includes("a"));
//calcul puissance
console.log(5 **4);
```



DOM

JS



DOM

MODIFICATION DE PAGES WEB



DOM

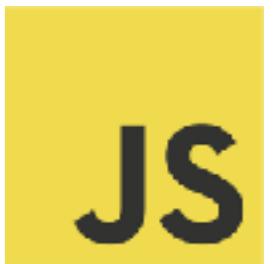
ORGANISER LE CODE JS



- Où on code ?

DOM

ORGANISER LE CODE JS



index.html × JS main.js

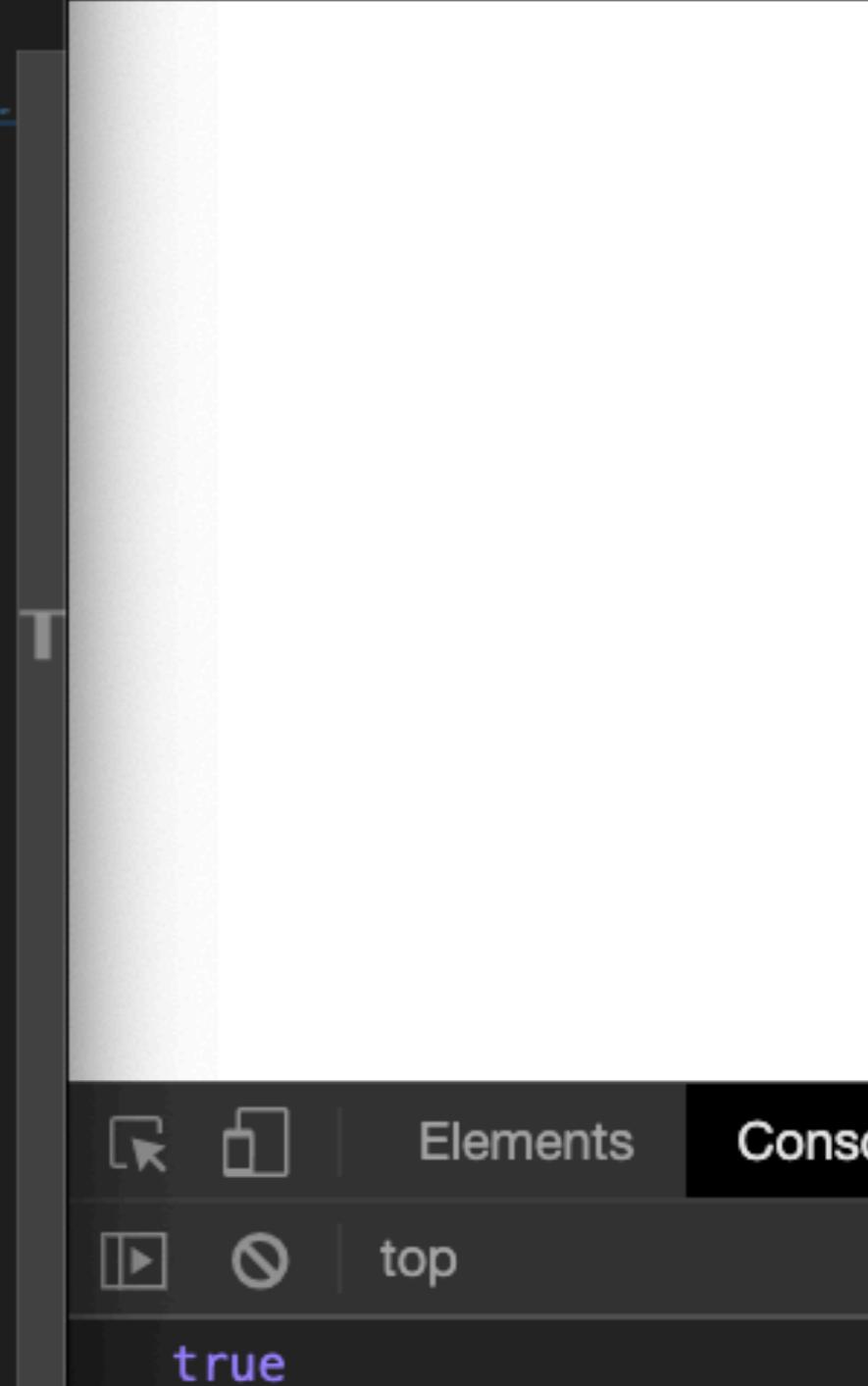
```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
</head>
<body>

<!-- Code HTML -->

<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

...

< > C 88 | 127.0.0.1



DOM

ORGANISER LE CODE JS

JS

The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the file `index.html` with the following content:

```
<> index.html <
<> index.html > ⚒ html > ⚒ body > ⚒ script
1   <!DOCTYPE html>
2   <html lang="fr">
3   <head>
4       <meta charset="UTF-8">
5       <title>Tuto</title>
6   </head>
7   <body>
8
9   <!-- Code HTML -->
10
11  <!-- <script type="text/javascript" src="main.js"></scr
12      <script>
13          alert("Salut on se marre");
14      </script>
15
16  </body>
17  </html>
```

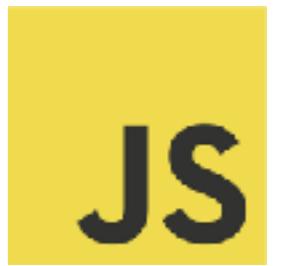
The browser window shows the URL `http://127.0.0.1:550`. The page content is:

127.0.0.1:5500 indique
Salut on se marre

The browser's developer tools are visible at the bottom, showing the `Console` tab which contains the message `Salut on se marre`.

DOM

DOCUMENT MODELING OBJECT



- Le navigateur va lire notre document HTML
- Il en fait une « traduction » pour l'afficher d'une certaine manière.
- JS va pouvoir interagir avec ce Document créé par le navigateur
 - On va pouvoir sélectionner, modifier des éléments, en ajouter, en supprimer...

JS

DOM

DOCUMENT MODELING OBJECT

The screenshot illustrates the Document Modeling Object (DOM) in action. On the left, the `index.html` file contains standard HTML structure and a placeholder for a script. On the right, the `main.js` file contains a single line of JavaScript code: `// alert("Salut");`. In the browser preview, the word "Hello" is displayed above a block of placeholder text. The browser's developer tools are open, showing the DOM tree where the `<body>` element is expanded, revealing its contents: the `<h1>Hello</h1>` heading and the `<p>Lorem ipsum dolor sit amet` paragraph.

```
index.html > html
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <title>Tuto</title>
6  </head>
7  <body>
8
9  <!-- Code HTML -->
10 <h1>Hello</h1>
11 <p>Lorem ipsum dolor sit amet
12     consectetur, adipisicing elit.
13     Mollitia, magnam quibusdam
14     reprehenderit cum, sint aspernatur
15     dolore eveniet quidem, praesentium
16     enim accusantium odit rem ad earum
17     tempore illo cumque saepe quos.</p>
18 <script type="text/javascript" src="main.j
19
20
21
22
23
24
25
26
27
28
29
30

JS main.js
1  // Code JavaScript ici
2
3
4  // alert("Salut");
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

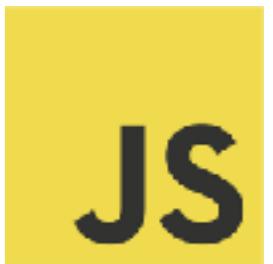
Hello

Lorem ipsum dolor sit amet consectetur, adipisicing elit.
magnam quibusdam reprehenderit cum, sint aspernatur
quidem, praesentium enim accusantium odit rem ad earum
illo cumque saepe quos.

Elements Console
▶ top
> document
< ▶ #document
    <!doctype html>
    <html lang="fr">
        <head>...</head>
        <body cz-shortcut-listen="true">
            <!-- Code HTML -->
            <h1>Hello</h1>
            <p>Lorem ipsum dolor sit amet
                consectetur, adipisicing elit.
                Mollitia, magnam quibusdam
                reprehenderit cum, sint aspernatur
                dolore eveniet quidem, praesentium
                enim accusantium odit rem ad earum
                tempore illo cumque saepe quos.</p>
        </body>
    </html>
```

DOM

DOCUMENT MODELING OBJECT



The image shows a development environment with two code editors and a browser window.

Code Editors:

- index.html:** Contains standard HTML structure including a title, head, body, and a paragraph with placeholder Latin text.
- main.js:** Contains a single line of JavaScript code: `// alert("Salut");`.

Browser Preview:

- The browser displays the word "Hello" in a large font.
- The page content includes a paragraph of Latin text: "Lorem ipsum dolor sit amet consectetur, adipisicing elit. Mollitia, magnam quibusdam reprehenderit cum, sint aspernatur dolore eveniet quidem, praesentium enim accusantium odit rem ad earum tempore illo cumque saepe quo".

Console:

```
> document.body
< -<body cz-shortcut-listen="true">
    <!-- Code HTML -->
    <h1>Hello</h1>
    <p>...</p>
    <script type="text/javascript" src="main.js"></script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
```

DOM

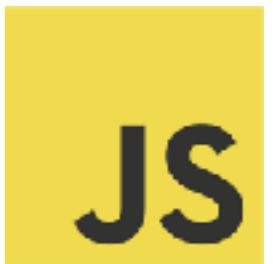
TROUVER UN ÉLÉMENT



- Comment trouver un élément dans notre code HTML via JS ?
- On va stocker un élément du DOM dans une variable.
- On va utiliser `getElementsByName ()`
(récupérer un élément selon le nom de son tag)
(C'est le nom des balises html p, h1, body, script etc ...)
- Par défaut `getElementsByName ()` récupère des données et les stocke sous formes de tableau.
(même vide si il ne trouve rien)

DOM

TROUVER UN ÉLÉMENT



The image shows a development environment with a code editor and a browser's developer tools open.

Code Editor: The file `main.js` contains the following JavaScript code:

```
// Code JavaScript ici
const titre = document.body.getElementsByTagName("h1");
console.log(titre);
```

Browser Dev Tools - Elements Tab: The browser window displays the word "Hello" in large bold letters. The developer tools show the DOM structure:

- Elements Tab:** Shows the `h1` element with attributes: align, title, lang, translate, dir, hidden, and accessKey.
- Console Tab:** The output of the console log command is shown as an `HTMLCollection [h1]` object with a length of 1 and one item at index 0.

Output: The browser displays the text "Hello". Below it is a block of Latin placeholder text (Lorem ipsum).

DOM

TROUVER UN ÉLÉMENT



- Comment trouver un élément dans notre code HTML via JS ?
- On peut aussi chercher un élément html par son id
- On va utiliser getElementById ()
(récupérer un élément selon le nom de son tag)
(C'est le nom de l'id des balises html)
- getElementById () renvoi directement les données elles mêmes et non pas sous forme de tableau.

JS

DOM

TROUVER UN ÉLÉMENT

The screenshot shows a developer environment with the following components:

- Code Editor:** A dark-themed code editor with tabs for "index.html" and "main.js". The "main.js" tab is active, displaying the following code:

```
1 // Code JavaScript ici
2
3
4 const titre = document.getElementById("titre");
5 // const titre = document.body.getElementsByTagName("h1");
6
7 console.log(titre);
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```
- Browser Preview:** A window showing a simple page with a large "Hello" heading and some placeholder text.
- Developer Tools:** An open "Elements" panel in the bottom right corner, showing the DOM tree with an entry for the "titre" element.

DOM

DÉPLACER UN ÉLÉMENT



- Maintenant que l'on sait comment trouver un élément du DOM
- On va pouvoir en plus le sélectionner et le déplacer dans notre page.
- `insertBefore()`

DOM

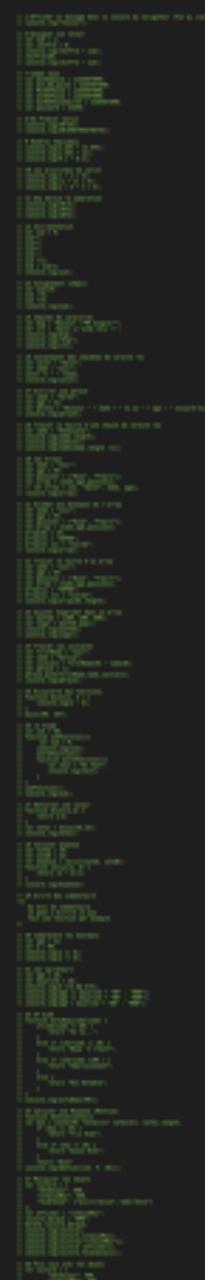
DÉPLACER UN ÉLÉMENT

quidem, praesentium enim a
illo cumque saepe quos.

Hello

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");

document.body.insertBefore(txt[0], titre);
console.log(titre);
```



DOM

DÉPLACER UN ÉLÉMENT

JS

- Avec la fonction appendChild()
- On va pouvoir ajouter un élément à la fin.

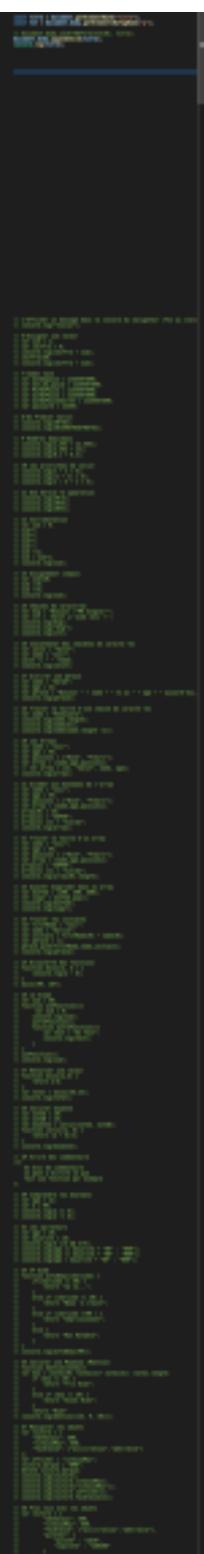
DOM

DÉPLACER UN ÉLÉMENT

```
// Code JavaScript ici
```

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
```

```
// document.body.insertBefore(txt[0], titre);
document.body.appendChild(titre);
console.log(titre);
```



magnam quibusdam reprehenderit
quidem, praesentium enim accusa
illo cumque saepe quo.

Hello

JS

DOM

SUPPRIMER UN ÉLÉMENT

- Avec la fonction `removeChild()`
- On va pouvoir supprimer un élément

DOM

SUPPRIMER UN ÉLÉMENT

Illo cumque s

```
3
4 const titre = document.getElementById("titre");
5 const txt = document.body.getElementsByTagName("p");
6
7 // document.body.insertBefore(txt[0], titre);
8 // document.body.appendChild(titre);
9 document.body.removeChild(titre);
10 console.log(document.body);
11
12
13
14
15
16
17
18
19
20
21
22
```



JS

DOM

CRÉER UN ÉLÉMENT (TEXTE)

- On peut créer des éléments texte
- Avec la fonction `createTextNode()` on créer du texte supplémentaire
- Mais il faut le placer.

DOM

CRÉER UN ÉLÉMENT (TEXTE)

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const newTxt = document.createTextNode("Bienvenue !");

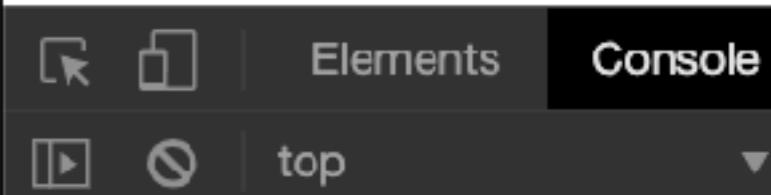
document.body.appendChild(newTxt);

console.log(document.body);
```



Lorem ipsum dolor sit amet consectetur, adipisci magnam quibusdam reprehenderit cum, sint a quidem, praesentium enim accusantium odit re illo cumque saepe quos.

Bienvenue !



```
▼<body cz-shortcut-listen=
  <!-- Code HTML -->
  <h1 id="titre">Hello</h1>
  ▶<p>...</p>
  <script type="text/javascript"
  src="main.js"></script>
  "Bienvenue !"
```

DOM

CRÉER UN ÉLÉMENT (TEXTE)

- On peut remplacer du texte par un. Autre
- Avec la fonction replaceChild()
- On indique Le nouvel élément
- Et l' ancien élément qui sera remplacé.

DOM

CRÉER UN ÉLÉMENT (TEXTE)

// Code JavaScript ici

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const newTxt = document.createTextNode("Bienvenue !");

document.body.appendChild(newTxt);
document.body.replaceChild(newTxt, txt[0]);

console.log(document.body);
```



DOM

CRÉER TOUS LES TYPES D'ÉLÉMENTS HTML

- On peut générer n'importe que élément HTML à partir de JS.
- On va voir innerHTML pour chercher directement le code HTML de quelque chose.
- On va voir createElement() pour créer un nouvel élément HTML.

DOM

CRÉER TOUS LES TYPES D'ÉLÉMENTS

The screenshot shows a browser window with developer tools open. The DOM tree on the left lists various elements like ``, `<div>`, and `<p>`. On the right, a list of created elements is shown:

- Hello
- Lorem ipsum dolor sit amet
magnam quibusdam reprehenderit
quidem, praesentium enim a
illo cumque saepe quo.
- JOSé : AZERTYUIOP
- sarah : wxcvbn
- Michel : Super
- Lala : POIUYTREZA
- Tom : mlkjhgfdsq

At the bottom, the developer tools footer shows icons for back, forward, and search, along with tabs for "Elements" and "top".

```
main.js > ajoutTexte
// Code JavaScript ici

const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");

function ajoutTexte(pseudo, duTexte) {
    const newTxt = document.createElement("p");

    newTxt.innerHTML = `<strong>${pseudo}</strong> : ${duTexte}`;
    document.body.appendChild(newTxt);
}

ajoutTexte("JOSé", "AZERTYUIOP");
ajoutTexte("sarah", "wxcvbn");
ajoutTexte("Michel", "Super");
ajoutTexte("Lala", "POIUYTREZA");
ajoutTexte("Tom", "mlkjhgfdsq");
```

DOM

MODIFIER LES ATTRIBUTS



- Sur chaque élément HTML on retrouve des attributs
- <h1 id="titre">Titre</h1>
(Les id, les class, le type, le src etc ...)
- En JS on peut récupérer les attributs d'un élément avec la fonction `getAttribute()`
- Et on peut les modifier avec `setAttribute()`
on indique ce qu'on veut remplacer et par quoi on le remplace.

DOM

MODIFIER LES ATTRIBUTS

```
1<html lang="fr">
2  <head>
3    <meta charset="UTF-8">
4    <title>Tuto</title>
5  </head>
6  <body>
7    <!-- Code HTML -->
8    <h1 id = "titre" >Hello</h1 id = "titre" >
9    <p>Lorem ipsum dolor sit amet
10       consectetur, adipisicing elit.
11       Mollitia, magnam quibusdam
12       reprehenderit cum, sint aspernatur
13       dolore eveniet quidem, praesentium
14       enim accusantium odit rem ad earum
15       tempore illo cumque saepe quos.</p>
16    <p>Lorem ipsum dolor sit amet
17       consectetur, adipisicing elit.
18       Mollitia, magnam quibusdam
19       <a href="#">MON LIEN</a>
20       reprehenderit cum, sint aspernatur
21       dolore eveniet quidem, praesentium
22       enim accusantium odit rem ad earum
23       tempore illo cumque saepe quos.</p>
24    <script type="text/javascript" src="main.js"></script>
25  </body>
```



Lorem ipsum dolor sit amet consectetur, adip
magnam quibusdam reprehenderit cum, sint a
quidem, praesentium enim accusantium odit i
illo cumque saepe quos.

Lorem ipsum dolor sit amet consectetur, adip
magnam quibusdam MON LIEN reprehender
dolore eveniet quidem, praesentium enim acc
earum tempore illo cumque saepe quos.

DOM

MODIFIER LES ATTRIBUTS

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
```

```
const lien = document.body.getElementsByTagName("a")[0];
```

```
console.log(lien.getAttribute("href"));
console.log(titre.getAttribute("id"));
```

DOM

MODIFIER LES ATTRIBUTS

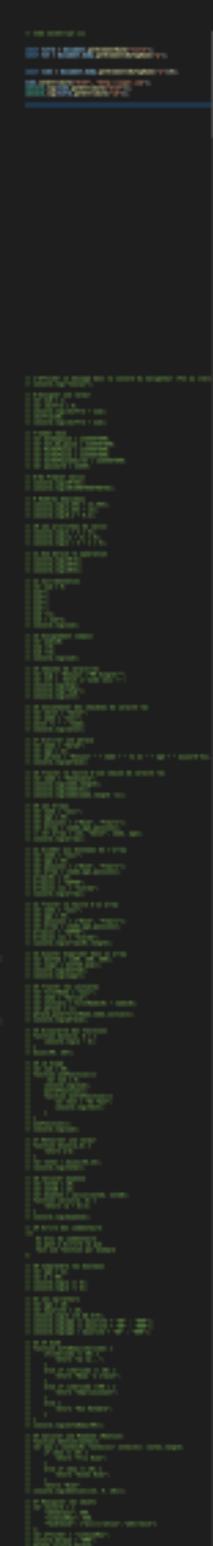
JS

```
// Code JavaScript ici
```

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
```

```
const lien = document.body.getElementsByTagName("a")[0];
```

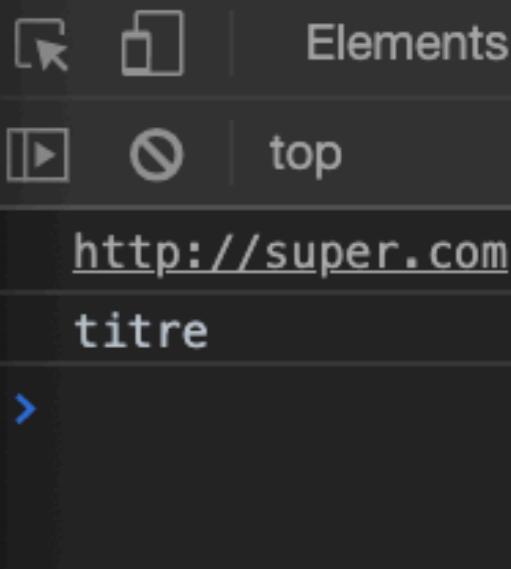
```
lien.setAttribute("href", "http://super.com");
console.log(lien.getAttribute("href"));
console.log(titre.getAttribute("id"));
```



Hello

Lorem ipsum dolor sit amet con
magnam quibusdam reprehende
quidem, praesentium enim accu
illo cumque saepe quos.

Lorem ipsum dolor sit amet con
magnam quibusdam MON LIE
dolore eveniet quidem, praesent
earum tempore illo cumque sae





DOM

CHANGER UNE LISTE D'ÉLÉMENT EN ARRAY

- Si on veut changer tous les paragraphes d'une page
- On a déjà vu Map avec les tableau ou on lui passe des entrées et il les remplace.
- Problème : les éléments HTML on beaucoup d'attributs mais ils n'ont pas .map()
- En JS on a une méthode simple pour transformer quelque chose en tableau
la méthode Array.from()

DOM

CHANGER UNE LISTE D'ÉLÉMENT EN ARRAY

main.js > ...

```
// Code JavaScript ici
```

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
```

```
const lien = document.body.getElementsByTagName("a")[0];
```

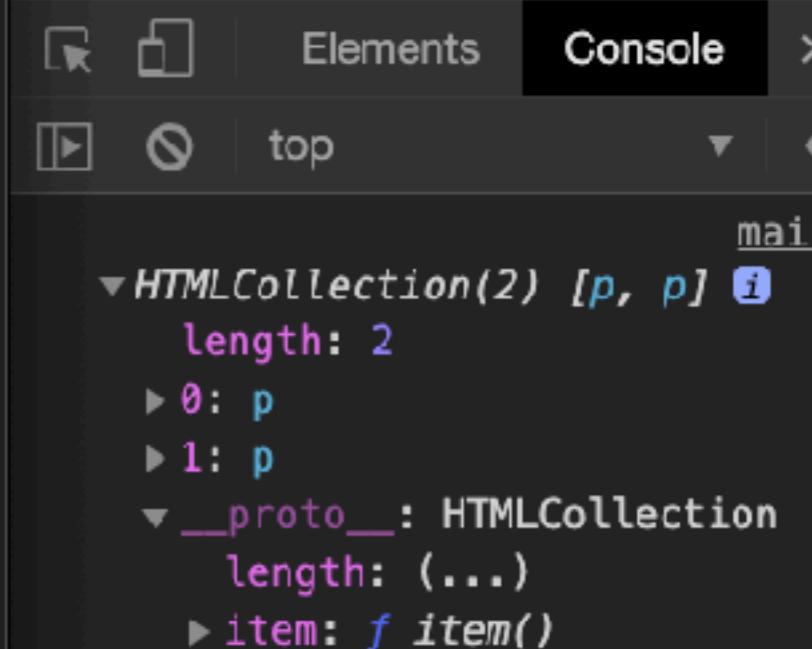
```
console.log(txt);
```



Hello

Lorem ipsum dolor sit amet consectetur, adipisci
magnam quibusdam reprehenderit cum, sint aspernatur
quidem, praesentium enim accusantium odit rem ad
illo cumque saepe quos.

Dolore ipsum dolor sit amet consectetur, adipisci
magnam quibusdam MON LIEN reprehenderit cum
dolore eveniet quidem, praesentium enim accusantium
earum tempore illo cumque saepe quos.



DOM

CHANGER UNE LISTE D'ÉLÉMENT EN ARRAY

The screenshot shows a browser window with developer tools open. On the left, the code editor displays `main.js` with the following content:

```
// Code JavaScript ici
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");

const lien = document.body.getElementsByTagName("a")[0];

console.log([1,2,3]);
```

The browser's main content area shows the word "Hello" above two paragraphs of placeholder text. The first paragraph contains a blue link labeled "MON LIEN". The JavaScript console at the bottom right lists several array methods:

- ▶ flat: *f flat()*
- ▶ flatMap: *f flatMap()*
- ▶ map: *f map()*
- ▶ every: *f every()*
- ▶ some: *f some()*
- ▶ reduce: *f reduce()*
- ▶ reduceRight: *f reduceRight()*
- ▶ toLocaleString: *f toLocaleString()*
- ▶ toString: *f toString()*
- ▶ Symbol(Symbol.iterator): *f valueOf()*
- ▶ Symbol(Symbol.unscopables): {com...

JS

DOM

CHANGER UNE LISTE D'ÉLÉMENT EN ARRAY

The screenshot shows a browser window with developer tools open. On the left, the code editor displays `main.js` with the following script:

```
// Code JavaScript ici
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const lien = document.body.getElementsByTagName("a")[0];
const texteTab = Array.from(txt);
console.log(texteTab);
```

The browser's main content area shows the word "Hello" above two paragraphs of placeholder text. The first paragraph contains a link labeled "MON LIEN".

The developer tools' Elements tab shows the DOM structure with two `p` elements and one `a` element.

The developer tools' Console tab shows the output of the `console.log` statement:

```
main.js:10
(2) [p, p]
▶ 0: p
▶ 1: p
  length: 2
  __proto__: Array(0)
    length: 0
    constructor: f Array()
    concat: f concat()
    copyWithin: f copyWithin()
    fill: f fill()
```

DOM

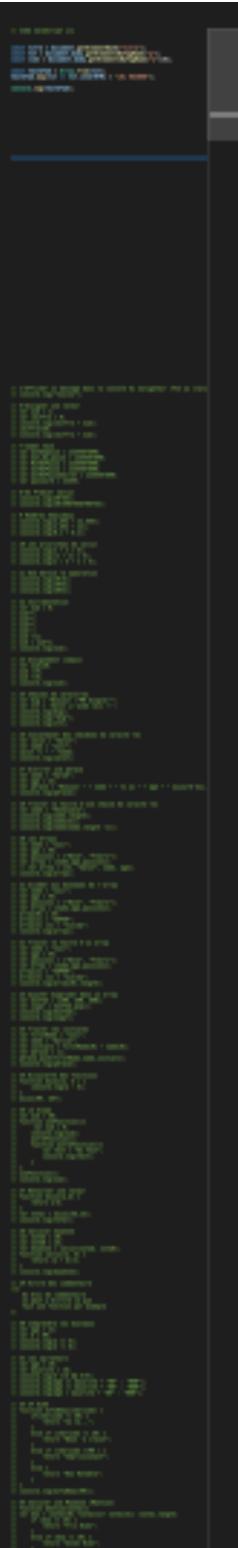
CHANGER UNE LISTE D'ÉLÉMENT EN ARRAY

```
// Code JavaScript ici
```

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const lien = document.body.getElementsByTagName("a")[0];

const texteTab = Array.from(txt);
texteTab.map(txt => txt.innerHTML = "LOL HACKED");

console.log(texteTab);
```



DOM

OBTENIR LA TAILLE D'UN ÉLÉMENT



- Récupérer la taille en hauteur et en largeur d'un élément.
- Utile pour ensuite le modifier avec précision (le déplacer, le réduire etc...)
- 2 type de calcul pour JS
- offset on calcule la taille de l'élément dans sa globalité.
(bordure comprises)
 - offsetHeight et offsetWidth
- client on calcule la taille de ce qu'il ya à l'intérieur de l'élément.
(il ne compte pas les bordures etc...)
 - clientHeight et clientWidth

DOM

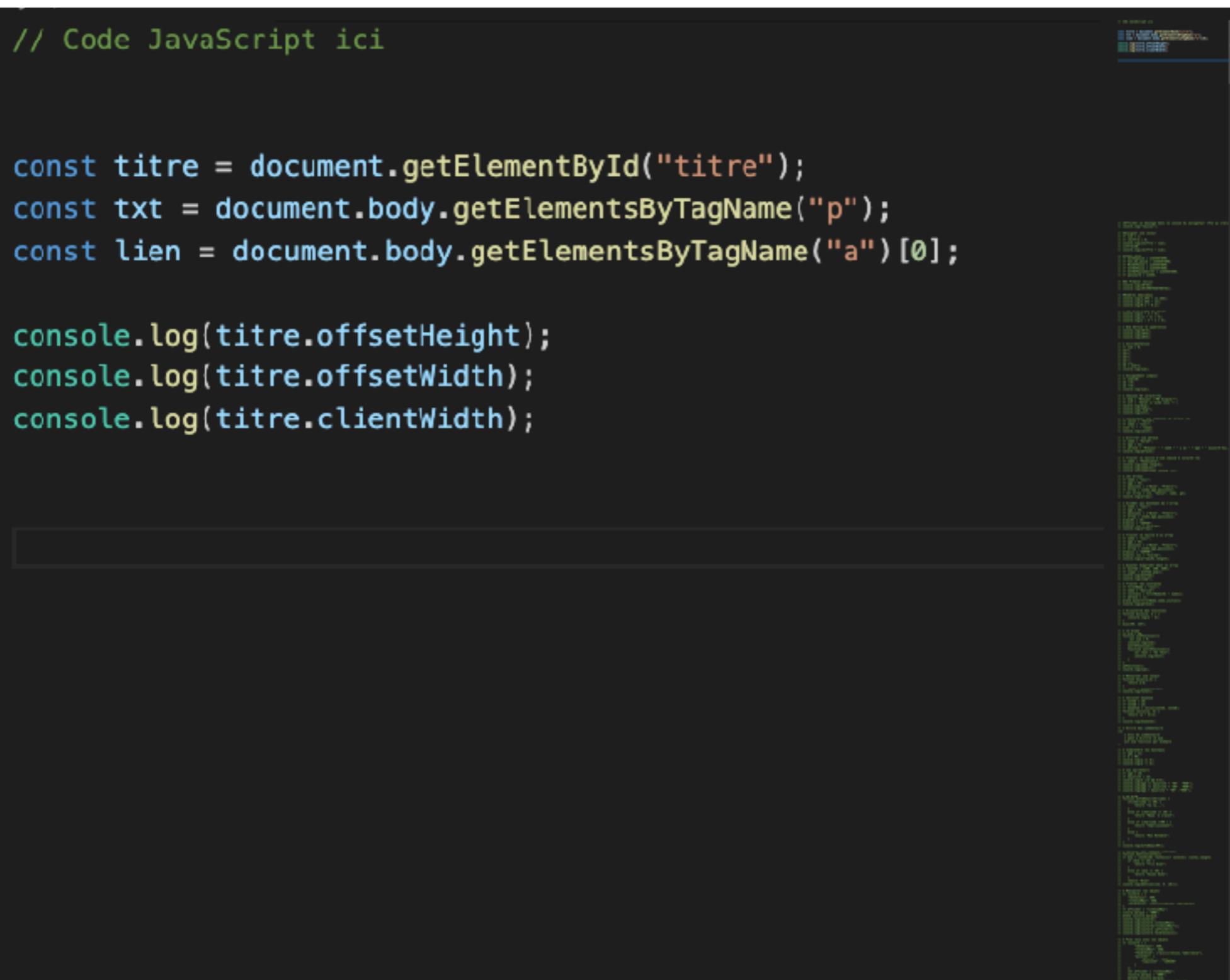
OBTENIR LA TAILLE D'UN ÉLÉMENT

JS

// Code JavaScript ici

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const lien = document.body.getElementsByTagName("a")[0];

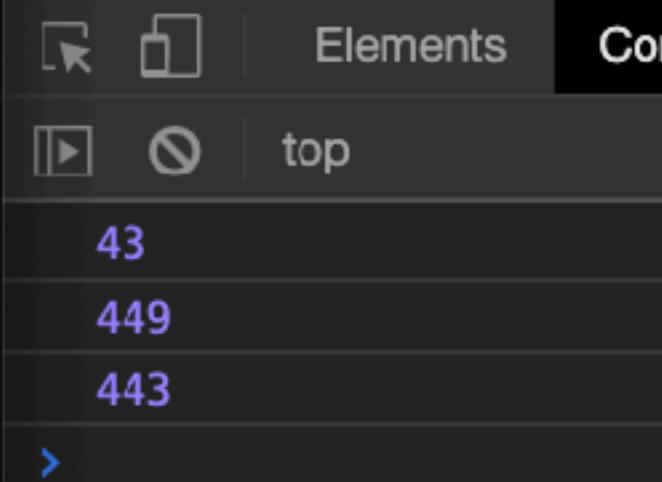
console.log(titre.offsetHeight);
console.log(titre.offsetWidth);
console.log(titre.clientWidth);
```



Hello

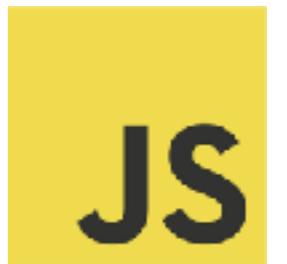
Lore ipsum dolor sit amet consectetur
magnam quibusdam reprehenderit cum
quidem, praesentium enim accusantium
illo cumque saepe quos.

Lore ipsum dolor sit amet consectetur
magnam quibusdam MON LIEN repre
dolore eveniet quidem, praesentium en
carum tempore illo cumque saepe quos



DOM

MODIFIER LE STYLE



- On a déjà vu comment intégrer du code css à l'intérieur de notre code HTML. Avec les balises `<style> </style>`
- Du coup en JS on peut manipuler n'importe quel élément HTML
- On va pouvoir modifier très simplement du CSS.

DOM

MODIFIER LE STYLE

JS

The screenshot shows a browser's developer tools open. On the left, the JavaScript pane displays the following code:

```
// Code JavaScript ici  
  
const titre = document.getElementById("titre");  
const txt = document.body.getElementsByTagName("p");  
const lien = document.body.getElementsByTagName("a")[0];  
  
titre.style.color = "red";
```

In the center, the DOM tree shows a structure with nodes like "Document", "Body", "P", and "A". To the right, the style editor is used to change the color of the title element to red, resulting in the word "Hello" being displayed in red. The bottom right corner of the slide features a yellow square containing the letters "JS".

DOM

MODIFIER LE STYLE

JS

main.js > ...

```
// Code JavaScript ici
```

```
const titre = document.getElementById("titre");
const txt = document.body.getElementsByTagName("p");
const lien = document.body.getElementsByTagName("a")[0];

titre.style.color = "red";
titre.style.fontFamily = "Impact";
titre.style.boxShadow = "2px 2px 20px green";
```

Hello

Lore ipsum dolor sit amet conse
magnam quibusdam reprehenderi
quidem, praesentium enim accusa
illo cumque saepe quos.

Lore ipsum dolor sit amet conse
magnam quibusdam MON LIEN
dolore eveniet quidem, praesentiu
earum tempore illo cumque saepe

Elements

top

DOM

OBTENIR LA TAILLE D'UN ÉLÉMENT (ALTERNATIVE)

JS

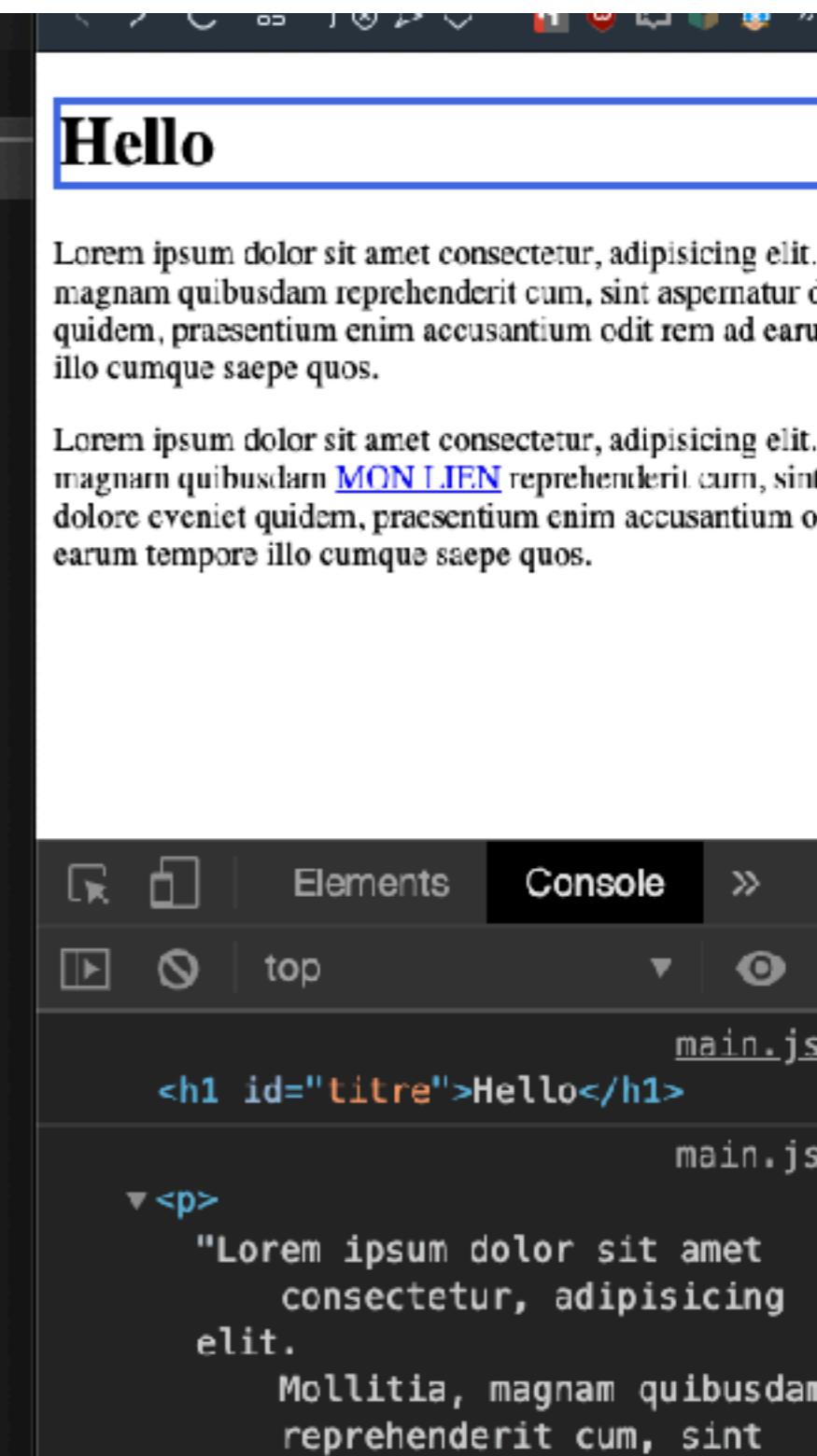
- CSS utilise déjà un système de sélecteur très précis
- On peut en profiter en JS
- Avec `querySelector()`, on a 2 types :
 - Simple `querySelector()`
il va sélectionner l'élément qu'on lui donne à chercher.
dès qu'il trouve il s'arrête de chercher.
 - `querySelectorAll()`
Qui nous renvoi tout sous forme de `NodeList`

DOM

JS

OBTENIR LA TAILLE D'UN ÉLÉMENT (ALTERNATIVE)

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4 const titre = document.getElementById("titre");
5 const txt = document.body.getElementsByTagName("p");
6 const lien = document.body.getElementsByTagName("a")[0];
7
8 const qqch = document.querySelector("#titre");
9 const truc = document.querySelector("p");
10
11 // const qqch = document.querySelector(".leNomDeMaClassCss");
12
13 console.log(qqch);
14 console.log(truc);
15
16
17
18
19
20
21
22
23
24
25
```

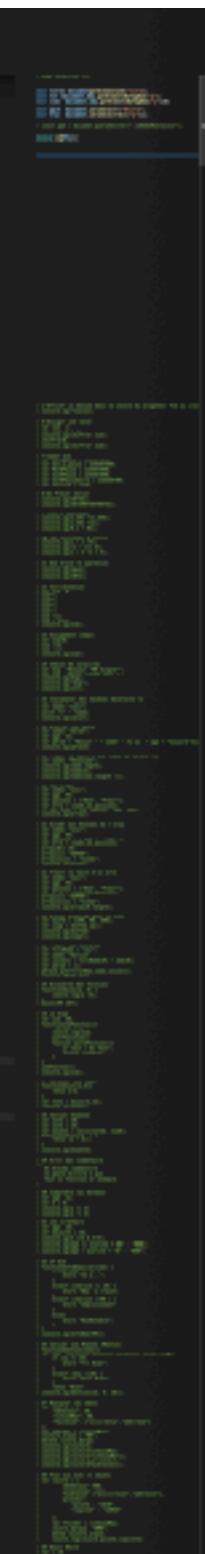


DOM

OBTENIR LA TAILLE D'UN ÉLÉMENT (ALTERNATIVE)

JS

```
JS main.js > ...
1 // Code JavaScript ici
2
3
4 const titre = document.getElementById("titre");
5 const txt = document.body.getElementsByTagName("p");
6 const lien = document.body.getElementsByTagName("a")[0];
7
8 const qqch = document.querySelector("#titre");
9 const truc = document.querySelectorAll("p")[1];
10
11 // const qqch = document.querySelector(".leNomDeMaClassCss");
12
13 console.log(qqch);
14 console.log(truc);
15
16
17
18
19
20
21
22
23
24
```



Hello

Lore ipsum dolor sit amet consectetur, adipisci
magnam quibusdam reprehenderit cum, sint
quidem, praesentium enim accusantium odit
illo cumque saepe quos.

2Lore ipsum dolor sit amet consectetur, adipisci
magnam quibusdam [MON LIEN](#) reprehenderit
dolore eveniet quidem, praesentium enim ac
earum tempore illo cumque saepe quos.

Elements Console
top

```
<h1 id="titre">Hello</h1>
▼<p>
  "2Lore ipsum dolor sit amet consectetur, adipisci
  elit.
  Mollitia, magni, et, natus, et, natus, et, natus,
```



DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

- `classList` nous renvoi la liste de toutes les classes CSS qui sont appliquées sur un élément HTML.
- Avec `classList.add("leNomdeMaClasse")` on ajoute une nouvelle classe
- Avec `classList.remove("leNomdeMaClasse")` on supprime une classe
- Avec `classList.toggle("leNomdeMaClasse")` on active ou non la classe.

JS

DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

The screenshot shows a development environment with three main panels:

- HTML Panel:** Contains the `index.html` file. It includes a header with meta information and a body section. In the body, there is an `<h1 class="test">Hello</h1>` element and three anchor tags (`Ajouter`, `Suppr`, `On / Off`) which likely correspond to the buttons in the browser preview.
- CSS Panel:** Contains the `style.css` file. It defines three styles:
 - `#style.css > h1{ border: 3px solid royalblue; }`
 - `#style.css > .test{ color: red; }`
 - `#style.css > .maCouleur{ color: violet; }`
- Browser Preview Panel:** Shows a browser window with the word "Hello" in red text inside a blue-bordered box. Below the browser window, the text "Ajouter Suppr On / Off" is displayed, indicating the state of the buttons.

JS

DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

The screenshot shows a browser window with developer tools open. On the left, there's a sidebar with various icons. The main area has tabs for 'index.html', '# style.css', and 'JS main.js'. The 'JS main.js' tab is active, displaying the following code:

```
// Code JavaScript ici
const titre = document.querySelector("h1");
const liens = document.querySelectorAll("a");

liens[0].addEventListener("click", function() {
    console.log(titre.classList);
});
```

To the right of the code editor, the browser window displays the word "Hello" in red text inside a blue-bordered box. Below the browser window, the developer tools interface is visible, featuring the 'Elements' tab (which shows a tree view of the page structure), the 'Console' tab (which is currently selected and shows the output of the console.log statement), and the 'Elements' tab again.

In the 'Console' tab, the output is:

```
main.js:8 DOMTokenList ["test", value: "test"]
i
length: 1
value: "test"
0: "test"
▶ __proto__: DOMTokenList
```

JS

DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

The screenshot shows a browser developer tools interface. On the left, there is a code editor window titled "main.js — base" containing the following JavaScript code:

```
// Code JavaScript ici
const titre = document.querySelector("h1");
const liens = document.querySelectorAll("a");

liens[0].addEventListener("click", function() {
    // console.log(titre.classList);
    titre.classList.add("maCouleur");
});
```

To the right of the code editor is a preview area showing a red "Hello" text in a blue-bordered box. Below the preview is a toolbar with buttons labeled "Ajouter", "Suppr", "On / Off". At the bottom of the preview area, there is a link "Ajouter Suppr On / Off".

At the bottom of the interface is a console window with tabs "Elements" and "Console" selected. The console tab shows the command "top".

JS

DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

The screenshot shows a browser developer tools interface. On the left, there's a file tree with 'index.html' and 'main.js'. The 'main.js' tab is active, displaying the following code:

```
main.js — base
index.html    JS main.js  X
JS main.js > ⚡ addEventListener("click") callback
1 // Code JavaScript ici
2
3
4 const titre = document.querySelector("h1");
5 const liens = document.querySelectorAll("a");
6
7 liens[0].addEventListener("click", function() {
8     titre.classList.add("maCouleur");
9 });
10
11 liens[1].addEventListener("click", function() {
12     titre.classList.remove("maCouleur");
13 });
14
15 liens[2].addEventListener("click", function() {
16     titre.classList.toggle("maCouleur");
17 });
18
19
20
21
22
```

The browser preview window shows a blue header bar with the text "Hello". Below it, a button labeled "Ajouter Suppr On / Off" is visible.

At the bottom, the developer tools console tab is open, showing the command:

```
>
```

JS

DOM

AJOUTER / SUPPRIMER DES CLASSES CSS

The image shows a development environment with a code editor and a browser window.

Code Editor (main.js):

```
// Code JavaScript ici
const titre = document.querySelector("h1");
const liens = document.querySelectorAll("a");

liens[0].addEventListener("click", function() {
    titre.classList.add("maCouleur");
});

liens[1].addEventListener("click", function() {
    titre.classList.remove("maCouleur");
});

liens[2].addEventListener("click", function() {
    titre.classList.toggle("maCouleur");
});
```

Browser Screenshot:

The browser displays the word "Hello" in red text inside a blue-bordered box. Below the box is a link labeled "Ajouter Suppr On / Off". The browser's developer tools are open, showing the "Elements" tab selected. The page source code is visible in the bottom left of the browser window.

EXO: SCROLL DE PAGE PRÉCIS

- Coder une fonction qui permet d'adapter le scroll de la page pour aller de titres en titres.
- `offsetTop` qui est une propriété d'un élément qui nous dis à quelle distance il est dans notre page par rapport au haut de la page.
(px)
- L'objet `window` de JS qui permet d'accéder à tous les élément de notre page (Tout le DOM)
- La méthode `ScrollTo(x, y)` on lui passe des coordonnées X Y
(on peut lui passer des options (behaviour smooth par exemple))

DOM

JS

EXO: SCROLL DE PAGE PRÉCIS

The screenshot shows a dark-themed code editor with a sidebar on the left containing icons for file, search, navigation, and help. The main area displays an HTML file named 'index.html' with the following content:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
<div>
    <h1>Titre 01</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipi
    <p>Officia sequi explicabo laboriosam fuga! Sunt
    <p>Totam iure quam hic! Accusamus pariatur, aspe
    <h1>Titre 02</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipi
    <p>Molestiae amet commodi ratione expedita. Vita
    <p>Provident amet a minus praesentium, id eliger
    <h1>Titre 03</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipi
    <p>Ipsa at debitis, animi incidentum rerum veritat
    <p>Cumque nam voluptate nisi explicabo ut unde
```

A horizontal bar with several icons and text labels. From left to right, it includes: three black dots, a white apple icon, a white document icon, a green diamond icon, a white square icon, a white document icon, a white document icon, the word "Tuto", a blue Google logo with the word "works", a white document icon, the word "Tuto", a minus sign, an equals sign, a left arrow, a right arrow, a double left arrow, a double right arrow, a double left arrow with a circle, a double right arrow with a circle, a double left arrow with a dot, a double right arrow with a dot, a double left arrow with a plus, a double right arrow with a plus, a double left arrow with a minus, a double right arrow with a minus, a double left arrow with a divide, a double right arrow with a multiply, a double left arrow with a plus minus, a double right arrow with a plus minus, a double left arrow with a divide plus, a double right arrow with a divide plus, a double left arrow with a divide minus, a double right arrow with a divide minus, a double left arrow with a multiply plus, a double right arrow with a multiply plus, a double left arrow with a multiply minus, a double right arrow with a multiply minus, a double left arrow with a divide plus minus, a double right arrow with a divide plus minus, a double left arrow with a multiply plus minus, a double right arrow with a multiply plus minus, a double left arrow with a divide plus minus plus, a double right arrow with a divide plus minus plus, a double left arrow with a multiply plus minus plus, a double right arrow with a multiply plus minus plus, a double left arrow with a divide plus minus plus plus, a double right arrow with a divide plus minus plus plus, a double left arrow with a multiply plus minus plus plus, a double right arrow with a multiply plus minus plus plus.

Titre 01

Consectetur adipisci velit at optio molestias accusamus ipsam quia rem. Quos nam quaerat molestias commodi voluptatum consectetur cum, rerum qui provident.

Officia sequi explicabo laboriosam fugit. Sunt molestiae, reprehenderit, repellat natus dignissimos cupiditate iure odio ad. Pariatur placeat facere voluptatum qui, optio cumque eum, quibusdam reiciendis esse voluptate consequatur, consectetur omnis!

Totam iure quam hic! Accusamus parialur, aspernatur ea dolorum exercitationem illum expedita veritatis vitae, error reiciendis possimus rerum numquam dolorem, suscipit animi adipisci reprehenderit non ducimus cumque. Dicta ab, explicabo.

Titre 02

Molestiae amet commodi ratione expedita. Vitae nisi aspernatur odio impedit rem cupiditate adipisci libero consequatur totam quia delectus, necessitatibus, ut laborineam a commodi voluntate vel a quo nocturnum fuisse.

DOM

JS

EXO: SCROLL DE PAGE PRÉCIS

The screenshot shows a code editor interface with several tabs: index.html, main.js, and style.css. The style.css tab is active, displaying the following CSS code:

```
# style.css > ⚡ div
1  h1{
2      border: 3px solid #royalblue;
3  }
4
5  .test{
6      color: #red;
7  }
8
9  .maCouleur {
10     color: #violet;
11 }
12
13 div {
14     max-width: 500px;
15     margin: auto;
16     font-size: 18px;
17     text-align: justify;
18 }
19
```

To the right of the editor is a browser window showing a page with two sections. The first section has a blue border and contains the text "Titre 01". Below it is a large block of Latin placeholder text. The second section also has a blue border and contains the text "Titre 02". Below it is another large block of Latin placeholder text.

Titre 01

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Perspiciatis adipisci velit iusto at optio molestias accusamus ipsam quia rem. Quos nam quaerat molestias commodi voluptatum consectetur cum, rerum qui provident.

Officia sequi explicabo laboriosam fuga! Sunt molestiae, reprehenderit, repellat natus dignissimos cupiditate iure odio ad. Pariatur placeat facere voluptatum qui, optio cumque eum, quibusdam reiciendis esse voluptate consequatur, consectetur omnis!

Totam iure quam hic! Accusamus parialur, aspernatur ea dolorum exercitationem illum expedita veritatis vitae, error reiciendis possimus rerum numquam dolorem, suscipit animi adipisci reprehenderit non ducimus cumque. Dicta ab, explicabo.

Titre 02

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Suscipit temporibus id ipsam dignissimos libero, autem porro hic quibusdam similique ad nihil non, numquam totam expedita est perspiciatis rem praesentium veniam.

Molestiae amet commodi ratione expedita. Vitae nisi aspernatur odio impedit rem cupiditate adipisci libero consequatur totam quia delectus, necessitatibus, ut laboriosam a commodi volumata vel quae nostrum faciat

DOM

EXO: SCROLL DE PAGE PRÉCIS

JS

The image shows a code editor window and a browser screenshot side-by-side.

Code Editor (main.js — base):

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5 const titres = document.querySelectorAll("h1");
6
7 function goTitre(titre) {
8     const dist = titre.offsetTop;
9     window.scrollTo(0, dist);
10}
11
12 goTitre(titres[3]);
13
14
15
16
17
18
19
20
21
22
```

Browser Screenshot:

- Title Bar:** Titre 04
- Content Area:** A large

Titre 04

 heading is visible at the top of the page. Below it is a block of Latin placeholder text (Lorem ipsum...).
- Console:** The developer tools console shows the command `goTitre(titres[3]);` being run, followed by `< undefined`.

DOM

EXO: REBONDIR UN ÉLÉMENT



- `querySelector`
- `clientHeight` et `clientWidth`
- Accès aux propriétés HTML d'un élément.
- Gérer le déplacement d'un élément px par px avec une vitesse ou force
- `requestAnimationFrame`

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The screenshot shows a browser developer tools window with the following components:

- DOM Tree:** On the left, the DOM tree for "index.html" is displayed. It includes nodes for "index.html", "main.js", and "style.css". The "index.html" node contains the HTML structure:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <div>
        <h1>Mon texte</h1>
    </div>
</body>
</html>
```
- Preview:** In the center, a preview of the web page shows a red-bordered

Mon texte

 element.
- Console:** At the bottom, the JavaScript console shows the following output:

```
500 500
main.js:9
```

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The screenshot shows a code editor interface with several files open:

- EXPLORER**: Shows files like `index.html`, `main.js`, and `# style.css`.
- OPEN EDITORS**: Shows `index.html`, `main.js`, and `# style.css`.
- BASE**: Shows files like `exoScroll.html`, `exoWindows.html`, and `index.html`.

The `# style.css` editor contains the following CSS code:

```
/* * {margin:0;} */

body {
    display: flex;
    justify-content: center;
    min-height: 100vh;
    align-items: center;
}

div {
    width: 500px;
    height: 500px;
    border: 10px solid teal;
}

h1 {
    width: 200px;
    height: 40px;
    border: 5px solid royalblue;
    color: red;
    margin: 0;
    text-align: center;
}
```

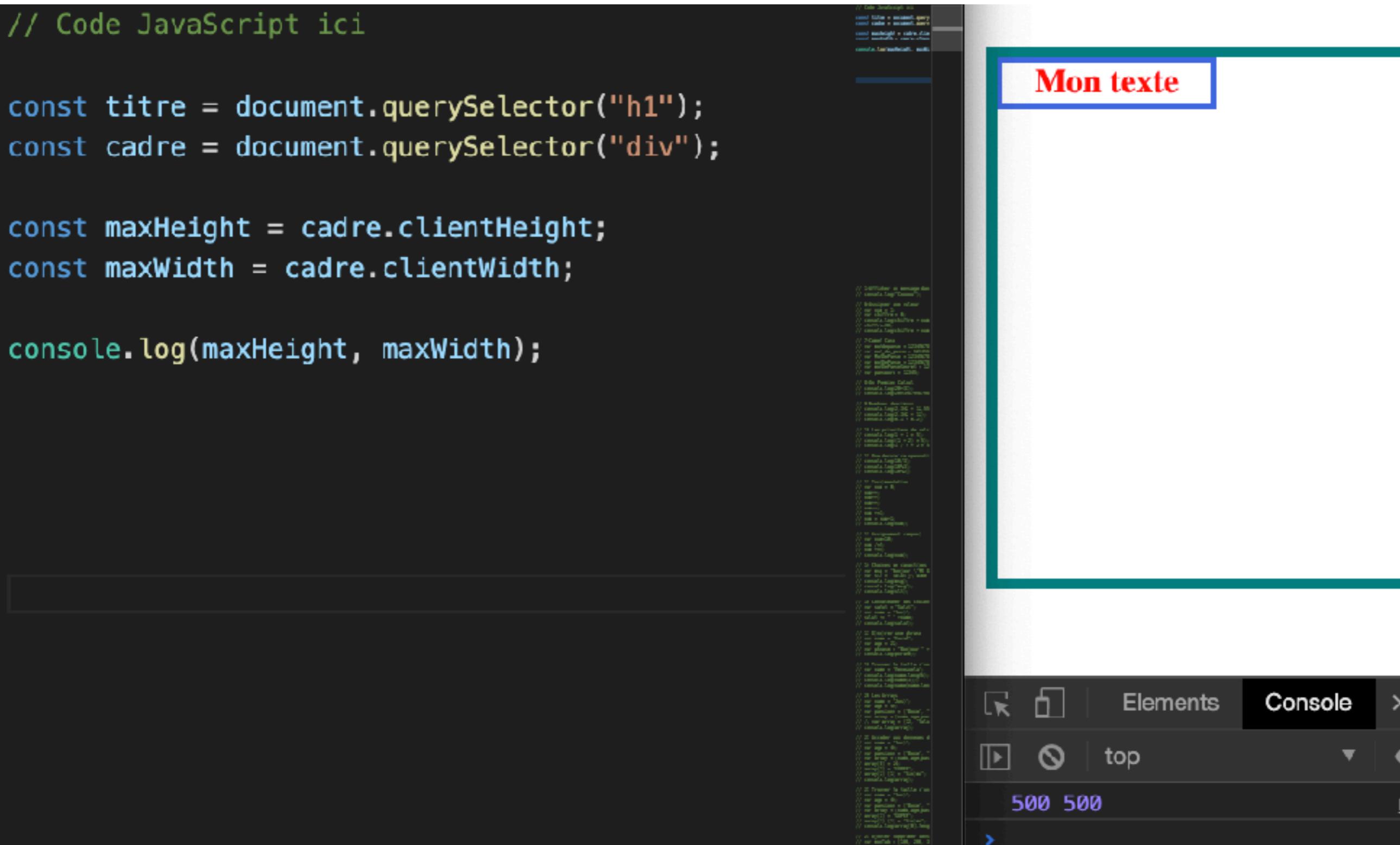
The browser preview window shows a centered teal-bordered div containing a red-bordered h1 element with the text "Mon texte".

The browser's developer tools are visible at the bottom, showing the element's dimensions as `500 500` and the file `main.js:9`.

DOM

EXO: REBONDIR UN ÉLÉMENT

```
1 // Code JavaScript ici  
2  
3 const titre = document.querySelector("h1");  
4 const cadre = document.querySelector("div");  
5  
6 const maxHeight = cadre.clientHeight;  
7 const maxWidth = cadre.clientWidth;  
8  
9 console.log(maxHeight, maxWidth);  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21
```



DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The screenshot shows a browser developer tools interface. On the left, there is a code editor window titled "main.js — base" containing the following JavaScript code:

```
// Code JavaScript ici
const titre = document.querySelector("h1");
const cadre = document.querySelector("div");

const maxHeight = cadre.clientHeight;
const maxWidth = cadre.clientWidth;

titre.style.position = "relative";
let topPos = 0;
let leftPos = 0;
let forceH = -2;
let forceW = -2;

console.log(maxHeight, maxWidth);
```

To the right of the code editor is a preview area showing a red-bordered box labeled "Mon texte". Below the preview is a browser's address bar with the URL "Tuto" and a "Console" tab in the bottom right corner of the developer tools.

The browser's status bar at the bottom shows the coordinates "500 500" and the file "main.js:15".

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The screenshot shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following JavaScript code:

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 const cadre = document.querySelector("div");
5 const maxHeight = cadre.clientHeight;
6 const maxWidth = cadre.clientWidth;
7 titre.style.position = "relative";
8 let topPos = 0;
9 let leftPos = 0;
10 let forceH = -2;
11 let forceW = -2;
12
13 function anim() {
14     if(topPos == 0) { forceH *= -1}
15     else if (topPos == maxHeight) {forceH *= -1}
16 }
17 console.log(maxHeight, maxWidth);
18
19
20
21
22
```

Browser Preview: A red box highlights the title bar of a browser window. Inside the browser, a blue-bordered box contains the text "Mon texte".

Console: The browser's developer tools show the output of the console.log statement:

```
Elements    Console    >
top          500 500    main.js:17
```

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The image shows a code editor window and a browser window side-by-side.

Code Editor (main.js):

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 const cadre = document.querySelector("div");
5 const maxHeight = cadre.clientHeight;
6 const maxWidth = cadre.clientWidth;
7 titre.style.position = "relative";
8 let topPos = 0;
9 let leftPos = 0;
10 let forceH = -2;
11 let forceW = -2;
12
13 function anim() {
14     if(topPos == 0) { forceH *= -1}
15     else if (topPos == maxHeight) {forceH *= -1}
16
17     topPos += forceH;
18     titre.style.top = topPos + "px";
19 }
20 requestAnimationFrame(anim);
21
22
```

Browser Preview:

A simple web page with a single

element containing the text "Mon texte". The element is styled with a red border and a blue background.

Developer Tools:

The browser's developer tools are open at the bottom, showing the "Elements" tab selected. The "Console" tab is also visible.

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The image shows a development environment with a code editor and a browser window.

Code Editor: The file `main.js` contains the following JavaScript code:

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > anim
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 const cadre = document.querySelector("div");
5 const maxHeight = cadre.clientHeight;
6 const maxWidth = cadre.clientWidth;
7 titre.style.position = "relative";
8 let topPos = 0;
9 let leftPos = 0;
10 let forceH = -2;
11 let forceW = -2;
12
13 function anim() {
14     if(topPos == 0) { forceH *= -1}
15     else if (topPos == maxHeight) {forceH *= -1}
16
17     topPos += forceH;
18     titre.style.top = topPos + "px";
19     requestAnimationFrame(anim);
20 }
21
22 requestAnimationFrame(anim);
```

Browser Screenshot: A screenshot of a web browser showing a page with a red box containing the text "Mon texte". The browser's developer tools are open, showing the "Elements" tab selected. The console tab shows the command `top`.

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The image shows a code editor window and a browser developer tools window side-by-side.

Code Editor (left):

- File menu: Fichier, Nouveau, Ouvrir, Sauvegarder, Fermer, Propriétés, Aide, Aide en ligne, Aide de l'application.
- Editor tabs: index.html, JS main.js, # style.css.
- Code content:

```
main.js — base
JS main.js > ...
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 const cadre = document.querySelector("div");
5 const maxHeight = cadre.clientHeight;
6 const maxWidth = cadre.clientWidth;
7 titre.style.position = "relative";
8 let topPos = 0;
9 let leftPos = 0;
10 let forceH = -2;
11 let forceW = -2;
12
13 function anim() {
14     if(topPos == 0) { forceH *= -1}
15     else if (topPos == maxHeight - titre.offsetHeight)
16         {forceH *= -1}
17
18     topPos += forceH;
19     titre.style.top = topPos + "px";
20     requestAnimationFrame(anim);
21 }
22 requestAnimationFrame(anim);
23
24
25
26
27
28
29
```

Browser Developer Tools (right):

- Address bar: Mon texte
- Elements tab: Shows a single element with a red border.
- Console tab: Shows the command "top" and its output.

DOM

JS

EXO: REBONDIR UN ÉLÉMENT

The image shows a development environment with a code editor on the left and a browser window on the right.

Code Editor (main.js):

```
main.js — base
index.html JS main.js # style.css
JS main.js > ...
1 // Code JavaScript ici
2 const titre = document.querySelector("h1");
3 const cadre = document.querySelector("div");
4 const maxHeight = cadre.clientHeight;
5 const maxWidth = cadre.clientWidth;
6 titre.style.position = "relative";
7 let topPos = 0;
8 let leftPos = 0;
9 let forceH = -2;
10 let forceW = -2;
11
12 function anim() {
13     if(topPos == 0) { forceH *= -1}
14     else if (topPos == maxHeight - titre.offsetHeight)
15         {forceH *= -1}
16
17     if(leftPos == 0) { forceW *= -1}
18     else if (leftPos == maxWidth - titre.offsetWidth)
19         {forceW *= -1}
20
21     topPos += forceH;
22     leftPos += forceW;
23
24     titre.style.top = topPos + "px";
25     titre.style.left = leftPos + "px";
26
27     requestAnimationFrame(anim);
28 }
29 requestAnimationFrame(anim);
```

Browser Preview:

A large red-bordered box contains the text "Mon texte".

Browser DevTools:

- Elements tab selected.
- Console tab.
- Console output:

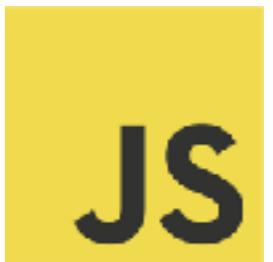
```
top
```



DOM ÉVÈNEMENTS

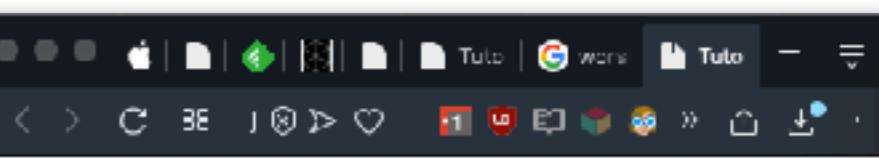
- Cela va permettre d'avoir des actions déclenchées par l'utilisateur.
- Au click d'un lien
- Les events

DOM ÉVÈNEMENTS

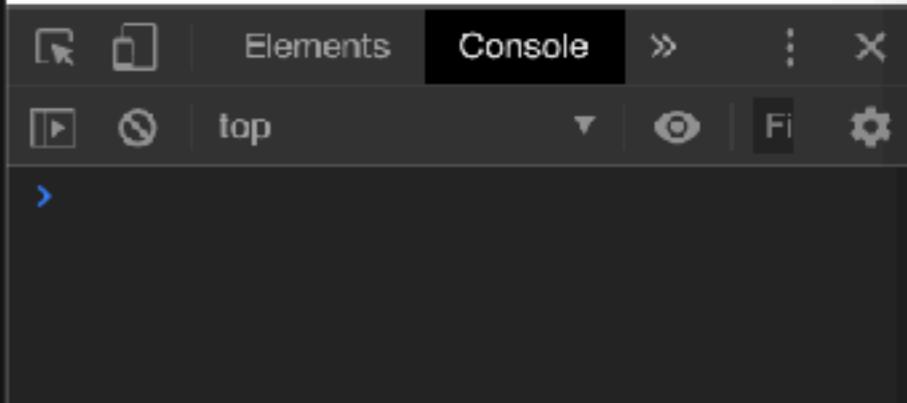


```
index.html — base
index.html × JS main.js # style.css

< index.html > ⏷ html > ⏷ body > ⏷ div > ⏷ a
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Tuto</title>
6  |   <link rel="stylesheet" href="style.css">
7  </head>
8  <body>
9  <!-- Code HTML -->
10 <div>
11 |   <!-- <h1>Mon texte</h1> -->
12 |   <a href = "#">UN LIEN</a>
13 </div>
14
15 <script type="text/javascript" src="main.js"></script>
16 </body>
17 </html>
```



UN LIEN



JS

DOM ÉVÈNEMENTS

The screenshot shows a browser developer tools interface with several panels:

- Top Bar:** Shows tabs for "index.html", "main.js", and "# style.css".
- Code Editor:** Displays the "main.js" file with the following code:

```
// Code JavaScript ici
const link = document.querySelector("a");
link.addEventListener("click", function() {
    console.log("Bonjour")
})
```
- Console Panel:** Shows the output of the console log statement: "19 Bonjour main.js:7".
- Bottom Status Bar:** Shows various icons and the text "Tuto" twice.

UN LIEN

DOM

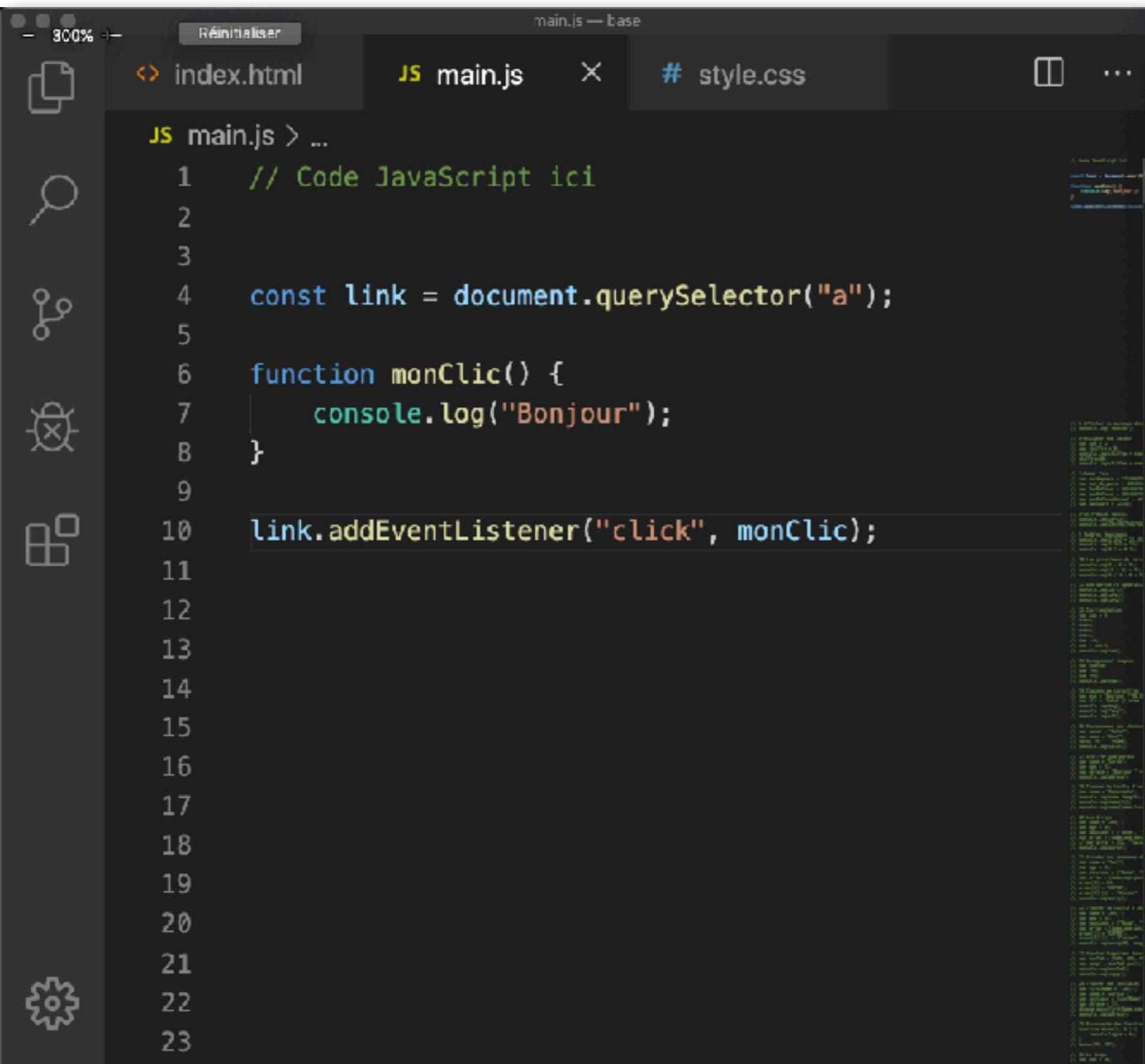
SUPPRIMER UN ÉVÈNEMENT

- Parfois on a créé un event
- Mais on veut qu'il soit valable qu'une seule fois il ne se relance pas.
- Une fois l'action déclenchée on arrête d'écouter la page.

JS

DOM

SUPPRIMER UN ÉVÈNEMENT



The screenshot shows a code editor with a dark theme. On the left, there are several icons: a file, a magnifying glass, a gear, a lightbulb, a grid, and a flower. The main area displays the following JavaScript code:

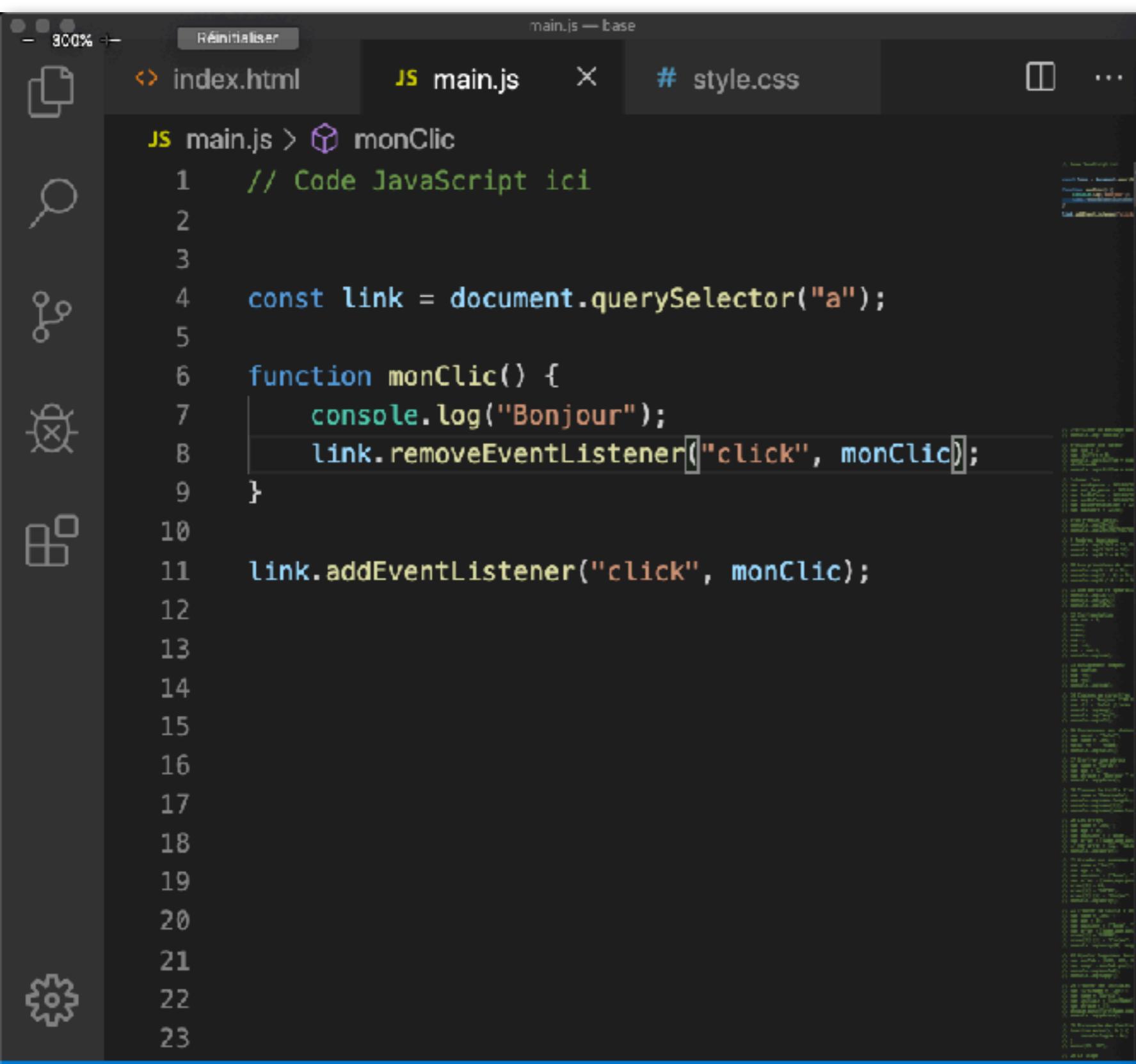
```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3
4 const link = document.querySelector("a");
5
6 function monClic() {
7     console.log("Bonjour");
8 }
9
10 link.addEventListener("click", monClic);
11
12
13
14
15
16
17
18
19
20
21
22
23
```

To the right of the code editor is a browser window showing a single purple link labeled "UN LIEN". At the bottom of the browser window is the developer tools' Console tab, which contains the following output:

```
Elements    Console    >
top
6 Bonjour    main.js:7
>
```

DOM

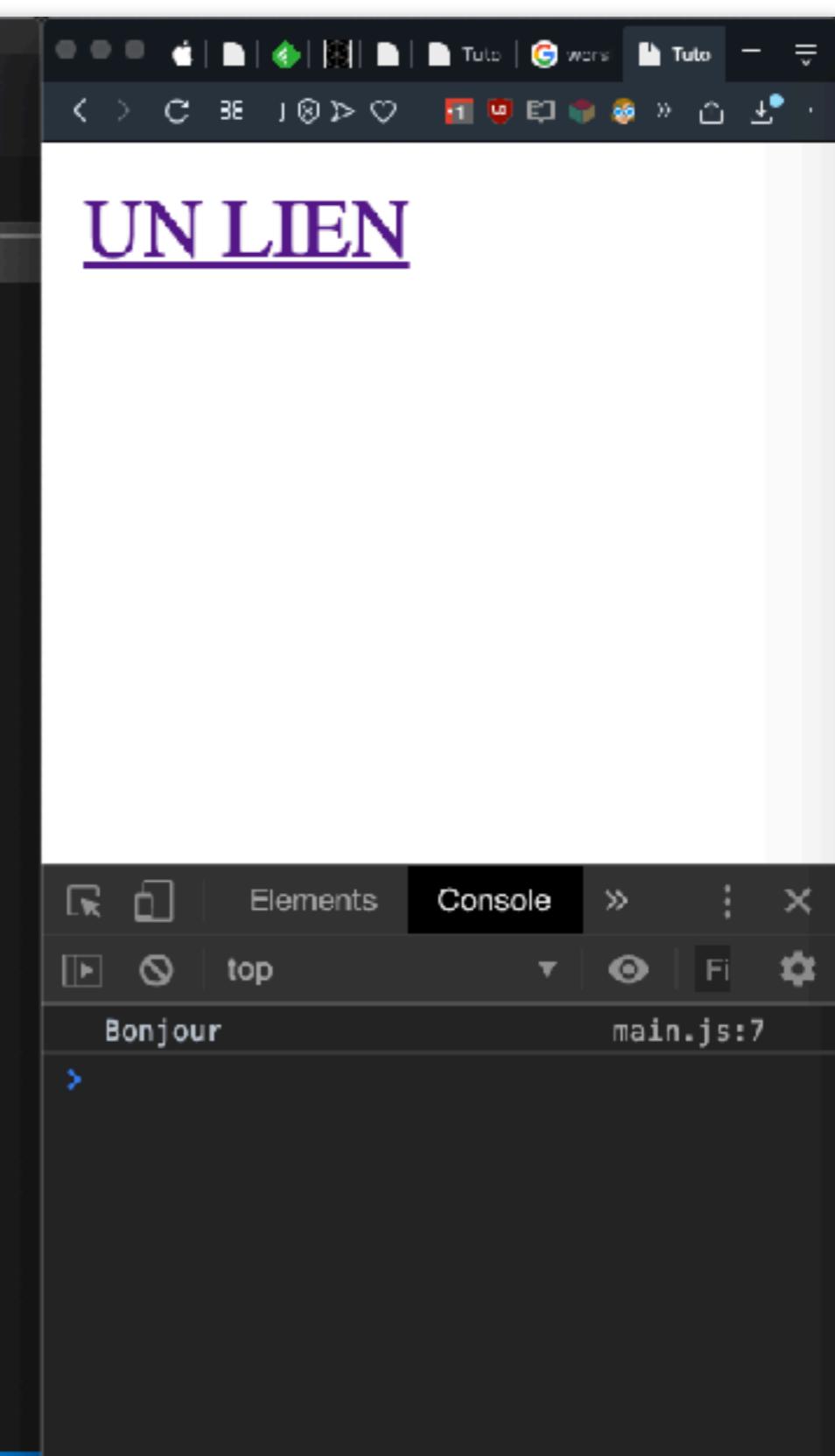
SUPPRIMER UN ÉVÈNEMENT



The screenshot shows a browser developer tools interface. On the left, the DOM tree is visible, with a tooltip pointing to a link element. The main code editor window contains the following JavaScript code:

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ⚡ monClic
1 // Code JavaScript ici
2
3
4 const link = document.querySelector("a");
5
6 function monClic() {
7     console.log("Bonjour");
8     link.removeEventListener("click", monClic);
9 }
10
11 link.addEventListener("click", monClic);
12
13
14
15
16
17
18
19
20
21
22
23
```

The code defines a function `monClic` that logs "Bonjour" to the console and removes its own event listener. It then adds a new event listener to the same link element.



The browser's developer tools console shows the output of the `console.log` statement:

```
Elements    Console    main.js:7
top
> Bonjour
```

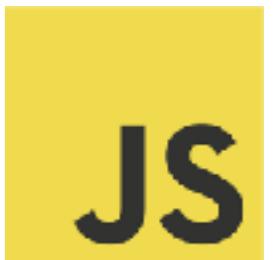
JS

DOM SURVEILLER LE CLAVIER

- On va utiliser addEventListener sur toute notre page
- On va surveiller keypress

DOM

SURVEILLER LE CLAVIER



The image shows a code editor window and a browser window side-by-side.

Code Editor (main.js):

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3
4 const link = document.querySelector("a");
5
6 function monClic() {
7     console.log("Bonjour");
8     link.removeEventListener("click", monClic);
9 }
10 link.addEventListener("click", monClic);
11
12 addEventListener("keypress", function(event) {
13     console.log(event);
14 })
15
16
17
18
19
20
21
22
23
```

Browser Screenshot:

A browser window displays the text "UN LIEN" in purple. The browser's developer tools are open, showing the "Console" tab with the following log output:

```
main.js:13
KeyboardEvent {isTrusted: true, key: "r", code: "KeyR", location: 0, ctrlKey: false, ...}
  isTrusted: true
  key: "r"
  code: "KeyR"
  location: 0
  ctrlKey: false
  shiftKey: false
  altKey: false
  metaKey: false
  repeat: false
  isComposing: false
  charCode: 114
  keyCode: 114
  view: Window {parent: Window, open...
```

DOM

SURVEILLER LE CLAVIER

JS

```
main.js — base
index.html JS main.js # style.css

JS main.js > ...
1 // Code JavaScript ici
2
3
4 const link = document.querySelector("a");
5
6 function monClic() {
7     console.log("Bonjour");
8     link.removeEventListener("click", monClic);
9 }
10 link.addEventListener("click", monClic);
11
12 addEventListener("keypress", function(event) {
13     console.log(event.key);
14 })
15
16
17
18
19
20
21
22
23
```

UN LIEN

Elements Console

top

h
e
2 l
o
>

main.js:13
main.js:13
main.js:13
main.js:13

DOM



JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

- On va coder un script pour afficher une image à l'endroit où l'on click.

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT



The screenshot shows a code editor interface with three tabs: `index.html`, `main.js`, and `style.css`. The `index.html` tab contains the following code:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
</body>
<script type="text/javascript" src="main.js"></script>
</html>
```

The `main.js` file is currently empty. The `style.css` file is also empty. In the bottom right corner, there is a browser preview window showing a blank white page.

JS

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
// Code JavaScript ici
addEventListener("click", function() {
    console.log("1 Clic");
});
```

Developer Tools - Console:

```
1 Clic
main.js:4
```

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT

JS

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor: The main.js file contains the following code:

```
// Code JavaScript ici
addEventListner("click", function(unEvent) {
    console.log(unEvent);
});
```

Developer Tools - Console Tab: The console output shows the result of the log statement:

```
MouseEvent {isTrusted: true, screenX: 1241, screenY: 370, clientX: 103, clientY: 91, ...}
  isTrusted: true
  screenX: 1241
  screenY: 370
  clientX: 103
  clientY: 91
  ctrlKey: false
  shiftKey: false
  altKey: false
  metaKey: false
  button: 0
```

A red oval highlights the event object in the console output.

JS

DOM

EXO: AFFICHER UNE IMAGE EN CLIQUANT

The screenshot shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
// Code JavaScript ici
addEventListener("click", function(unEvent) {
    console.log(unEvent.x, unEvent.y);
});
```

Browser Developer Tools - Console:

Line	Log	Source
6	6	main.js:4
7	15	main.js:4
8	22	main.js:4
9	26	main.js:4
10	76	main.js:4
11	76	main.js:4
12	55	main.js:4
13	41	main.js:4
14	43	main.js:4
15	84	main.js:4
16	47	main.js:4
17	52	main.js:4

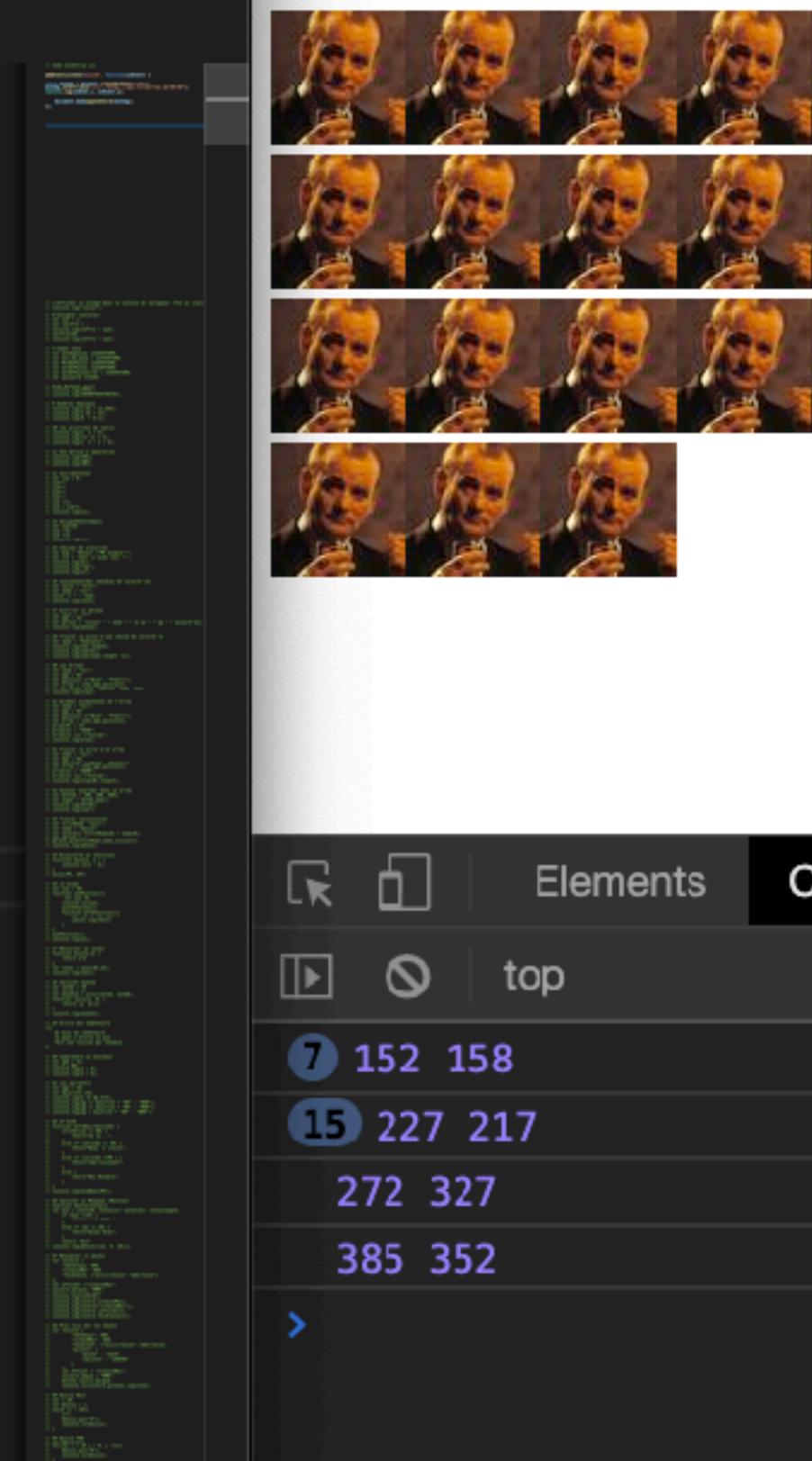
DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT

main.js > ...

```
1 // Code JavaScript ici  
2  
3 addEventListener("click", function(unEvent) {  
4  
5     const monImg = document.createElement("img");  
6     monImg.setAttribute("src", "https://www.fillmurray.com/58/58");  
7     console.log(unEvent.x, unEvent.y);  
8  
9     document.body.appendChild(monImg);  
0});  
1  
2  
3  
4  
5  
6  
7  
8  
9  
0  
1  
2  
3
```



DOM

JS

EXO : AFFICHER UNE IMAGE EN CLICQUANT

JS main.js > addEventListener("click") callback

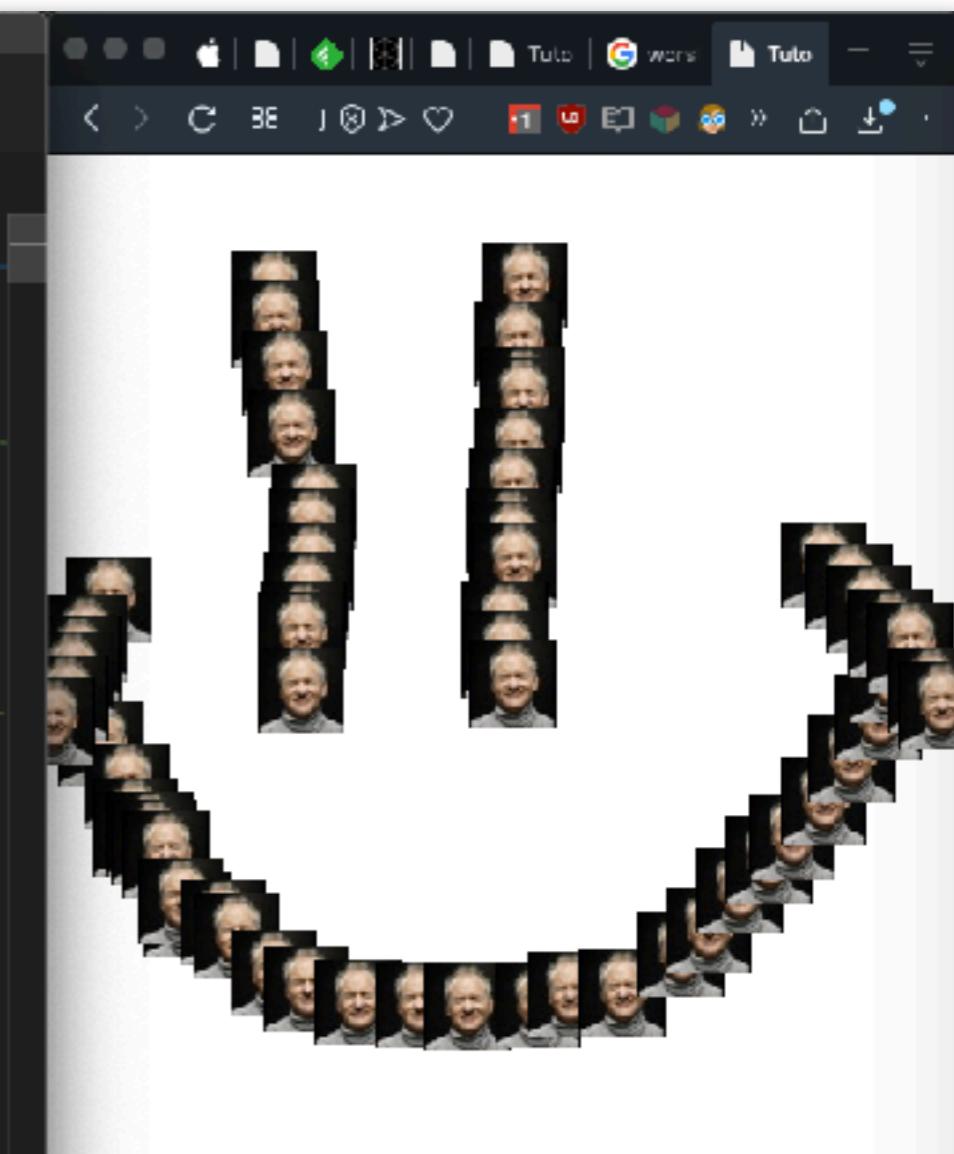
```
1 // Code JavaScript ici
2
3 addEventListener("click", function(unEvent) {
4
5 const monImg = document.createElement("img");
6 monImg.setAttribute("src","https://www.fillmurray.com/58/58");
7 // console.log(unEvent.x, unEvent.y);
8
9 monImg.style.position = "absolute";
10 monImg.style.left = event.x + "px";
11 monImg.style.top = event.y + "px";
12
13
14
15 document.body.appendChild(monImg);
16 });
17
18
19
20
21
```



DOM

JS

EXO: AFFICHER UNE IMAGE EN CLIQUANT



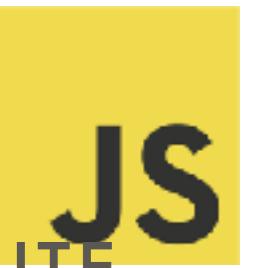
The screenshot shows a browser window with the address bar containing "Tuto" and "wors". The page content displays a grid of small, overlapping images of a man's face, which is a common visual effect used to demonstrate how multiple DOM elements are rendered.

```
main.js — base
index.html    JS main.js    # style.css

JS main.js > addEventListener("click") callback
1 // Code JavaScript ici
2
3 addEventListener("click", function(unEvent) {
4
5 const monImg = document.createElement("img");
6 const taille = 44;
7 monImg.setAttribute
8 ("src", "https://www.fillmurray.com/${taille}/${taille}/");
9
10 monImg.style.position = "absolute";
11 monImg.style.left = event.x - taille /2 + "px";
12 monImg.style.top = event.y - taille /2 + "px";
13
14
15 document.body.appendChild(monImg);
16
17 });
18
19
20
21
22
23
24
25
26
27
28
29
```

The code in the main.js file creates a new image element, sets its source to a URL that generates a square image of a man's face based on a size parameter, and then appends it to the body of the document. It also adds an event listener to the document that triggers this action whenever a click event occurs.

DOM



EXO: RETENIR L'UTILISATEUR QUAND IL QUITTE LE SITE

- Technique que l'on voit souvent sur des site de commerce par exemple
- Dès que la souris sors de window alors on vous affiche un pop up, une mailing list, un questionnaire, une réduction etc ...

DOM

JS

EXO: RETENIR L'UTILISATEUR QUAND IL QUITTE LE SITE

The screenshot shows a code editor and a browser window. The code editor has tabs for index.html, main.js, and style.css. The index.html file contains an H1 element with the text "Génial ce titre !". A script tag includes a reference to main.js. The browser window shows a simple page with the same H1 title.

```
index.html — base
index.html × JS main.js # style.css

< index.html > ⚡ html > ⚡ body > ⚡ h1
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Tuto</title>
6  |   <link rel="stylesheet" href="style.css">
7  </head>
8  <body>
9  |   <!-- Code HTML -->
10 |   <h1>Génial ce titre !</h1>
11
12 <script type="text/javascript" src="main.js"></script>
13 </body>
14 </html>
```

The browser window shows the following content:

```
<h1>Génial ce titre !</h1>
```

Console output:

```
>
```

DOM

JS

EXO: RETENIR L'UTILISATEUR QUAND IL QUITTE LE SITE

The screenshot shows a code editor interface with several tabs:

- index.html
- JS main.js
- # style.css

The # style.css tab is active, displaying the following CSS code:

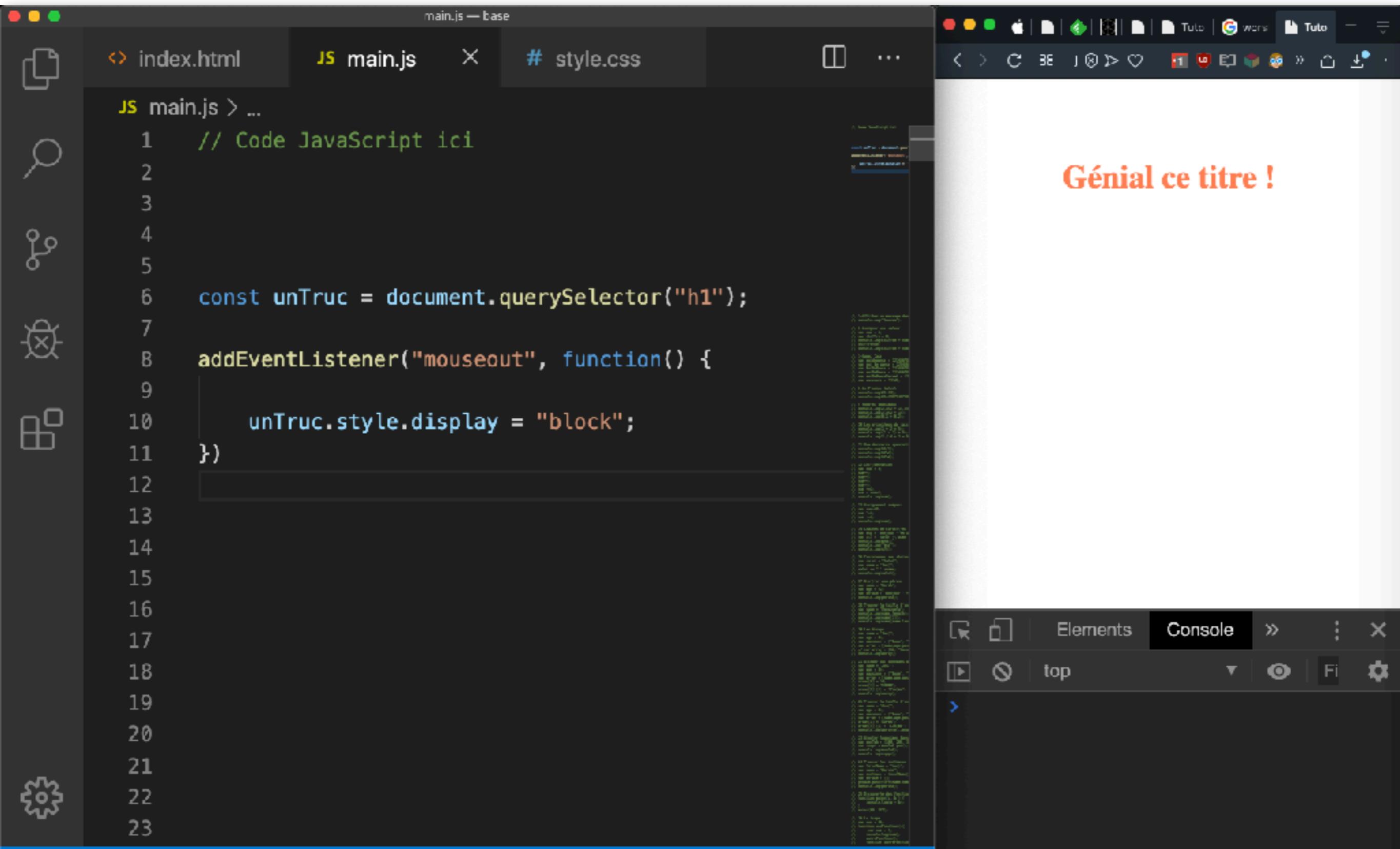
```
/* * {margin:0;} */
h1 {
    display: none;
    text-align: center;
    margin-top: 80px;
    color: coral;
}
```

To the right of the code editor is a browser window. The browser's address bar shows "localhost:3000". The browser's developer tools are open, specifically the "Elements" tab under the "Console" tab. The console log shows the following message:> top

DOM

JS

EXO: RETENIR L'UTILISATEUR QUAND IL QUITTE LE SITE



The image shows a code editor interface with a dark theme. On the left, there are several icons: a file, a magnifying glass, a gear, and a grid. The top bar shows tabs for "index.html", "main.js", and "# style.css". The main area contains the following JavaScript code:

```
main.js — base
index.html      JS main.js      # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3
4
5
6 const unTruc = document.querySelector("h1");
7
8 addEventListener("mouseout", function() {
9
10    unTruc.style.display = "block";
11})
12
13
14
15
16
17
18
19
20
21
22
23
```

To the right, a browser window is open, displaying a yellow header with the text "JS" and a white page with the text "Génial ce titre !". Below the browser is a developer tools panel with tabs for "Elements", "Console", and "Network". The "Console" tab is active, showing the command "top".

JS

DOM

EXO: INDIQUER LE % DU SCROLL

DOM

EXO: INDIQUER LE % DU SCROLL

JS

The image shows a development environment with a code editor and a browser preview.

Code Editor (index.html - base):

```
index.html — base
JS main.js    <> index.html X   # style.css
<> index.html > ⚡ html > ⚡ body > ⚡ h1
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="UTF-8">
5      <title>Tuto</title>
6      <link rel="stylesheet" href="style.css">
7  </head>
8  <body>
9  <!-- Code HTML -->
10 <div class="boxBar">
11     <div class="bar"></div>
12 </div>
13 <h1>Super Scroll Ultra</h1>
14
15 <script type="text/javascript" src='main.js'></script>
16 </body>
17 </html>
```

Browser Preview:

The browser window displays the text "Super Scroll Ultra" in a large, bold font. The title bar of the browser says "Tuto".

DOM

EXO: INDIQUER LE % DU SCROLL

JS

The image shows a code editor interface with several tabs: 'main.js', 'index.html', and '# style.css'. The '# style.css' tab is active, displaying the following CSS code:

```
/* * {margin:0;} */

body {min-height: 5000px;
      h1 {text-align: center; margin: 60px;}

      .boxBar {
          width: 100%;
          border: 5px solid royalblue;
          position: fixed;
          top: 0;
          left: 0;
      }

      .bar {
          height: 20px;
          background: royalblue;
          width: 0%;
```

To the right of the code editor is a browser window showing a page titled 'Super Scroll Ultra'. The page has a large blue header bar at the top and a blue scroll bar on the right side.

DOM

EXO: INDIQUER LE % DU SCROLL

JS

The screenshot shows a browser window with developer tools open. On the left, the code editor displays `main.js` with the following JavaScript code:

```
// Code JavaScript ici
const bar = document.querySelector(".bar");
addEventlistener("scroll", function() {
    console.log(`Hauteur page : ${document.body.scrollHeight}
    Hauteur affichage : ${innerHeight}
    Scroll Position : ${pageYOffset}`);
})
```

The browser's address bar shows `http://localhost:3000/Tuto`. The developer tools sidebar includes icons for file, search, and settings.

In the bottom right corner, the browser's developer tools console tab is active, showing the output of the `console.log` statements. The logs are as follows:

- Log 1: `Hauteur page : 365
Hauteur affichage : 365
Scroll Position : 748` (main.js:6)
- Log 2: `Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 749` (main.js:6)
- Log 3: `Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 750` (main.js:6)
- Log 4: `Hauteur page : 5000
Hauteur affichage : 365
Scroll Position : 751` (main.js:6)

DOM

JS

EXO: INDIQUER LE % DU SCROLL

main.js > addEventListener("scroll") callback

```
// Code JavaScript ici
```

```
const bar = document.querySelector(".bar");
```

```
addEventListener("scroll", function() {
```

```
    const scrollMax = document.body.scrollHeight - innerHeight;
```

```
    const onEstOu = pageYOffset / scrollMax * 100;
```

```
    bar.style.width = onEstOu + "%";
```

```
    console.log(`  
        Hauteur page : ${document.body.scrollHeight}  
        Hauteur affichage : ${innerHeight}  
        Scroll Position : ${pageYOffset}`);
```

Elements Console
top

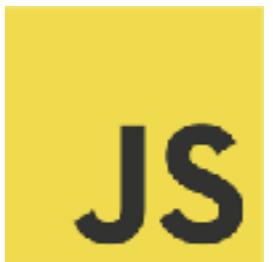
```
Hauteur page : 5000  
Hauteur affichage : 36  
Scroll Position : 3620  
  
Hauteur page : 5000  
Hauteur affichage : 36  
Scroll Position : 3622  
  
Hauteur page : 5000  
Hauteur affichage : 36  
Scroll Position : 3623
```

JS

DOM FOCUS & BLUR

- J'ai juste un input
- Quand je click dedans ça change la couleur à l'intérieur
- quand je click en dehors la couleur va s'en aller.

DOM FOCUS & BLUR



The image shows a development environment with a code editor and a browser. The code editor has tabs for 'main.js', 'index.html', 'style.css', and 'focusBlur.html'. The 'index.html' tab is active, displaying the following code:

```
index.html — base
JS main.js      index.html ×  # style.css      focusBlur.html
< index.html > ⏷ html > ⏷ body
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Tuto</title>
6  |   <link rel="stylesheet" href="style.css">
7  </head>
8  <body>
9  |   <!-- Code HTML -->
10 <input type="text" />
11
12
13
14 <script type="text/javascript" src="main.js"></script>
15 </body>
16 </html>
```

The browser window shows a simple page with a single text input field. The developer tools are open, showing the 'Elements' tab selected. The console tab is also visible, showing a single line of text: '> |'.

JS

DOM FOCUS & BLUR

The screenshot shows a code editor interface with several tabs open. The active tab is 'style.css' containing the following CSS code:

```
# style.css > ...
1  /* * {margin:0;} */
2
3
4
5  input {
6    width: 300px;
7    display: block;
8    margin: 120px auto;
9    font-size: 24px;
10   padding: 10px;
11 }
```

To the right of the code editor is a browser window displaying a simple input field. Below the browser window is the developer tools console, which is currently empty.

JS

DOM FOCUS & BLUR

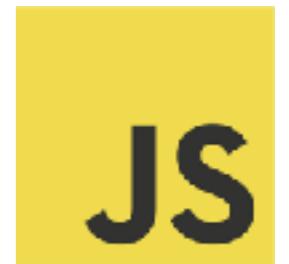
The screenshot shows a code editor interface with several tabs at the top: "main.js — base", "main.js", "index.html", "# style.css", and "focus". The "main.js" tab is active, displaying the following code:

```
1 // Code JavaScript ici
2
3
4
5 const leInput = document.querySelector("input");
6
7 leInput.addEventListener("focus", function() {
8     leInput.style.backgroundColor = "royalBlue";
9 });
10
11 leInput.addEventListener("blur", function() {
12     leInput.style.backgroundColor = "white";
13 });
14
15
16
17
18
19
20
21
22
23
```

To the right of the code editor is a browser window showing a single input field. The input field has a blue background color, indicating it is currently focused. At the bottom of the browser window is the developer tools console, which is currently active and shows the command "top".

DOM

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT



- Un nouvel event
- load
- Permet de faire une action uniquement après qu'un élément est chargé.
 - On met le load sur un élément en particulier
 - ex : images trop lourdes sur un site on veut pouvoir utiliser la page avant que les images soit totalement chargées.
 - On met le load sur l'ensemble de la page pour exécuter notre script une fois que toute la page soit chargée

DOM

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT

```
<meta charset="UTF-8">
<title>Tuto</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<!-- Code HTML -->












<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

DOM

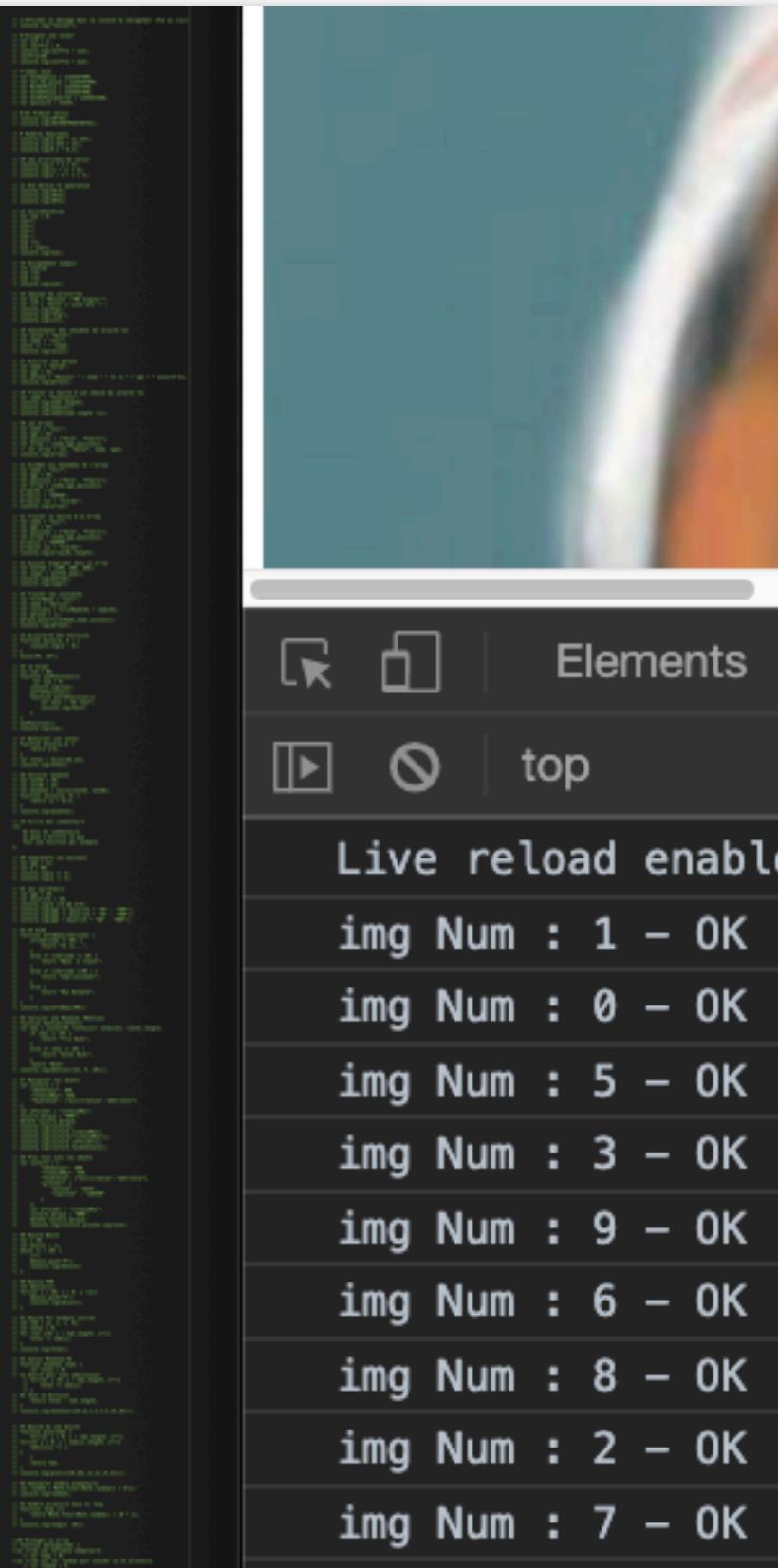
JS

ATTENTE AU CHARGEMENT D'UN ÉLÉMENT

```
const images = document.querySelectorAll("img");

const tabImg = Array.from(images);

tabImg.map((image,i) => image.addEventListener("load",
function(){
    console.log(`img Num : ${i} - OK`);
}))
```





DOM

EXO: DÉCLENCHER UN SCRIPT APRÈS X SEC

- Comment exécuter un script après un temps donnée
- milliseconde
- setTimeout

DOM

JS

EXO: DÉCLENCHER UN SCRIPT APRÈS X SEC

The screenshot shows a code editor and a browser developer tools interface.

Code Editor:

- File: index.html — base
- JS: main.js
- index.html
- # style.css

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <h1></h1>
<script type="text/javascript" src="main.js"></script>
</body>
</html>
```

Browser Developer Tools:

- Elements tab (selected)
- Console tab (selected)
- top scope

The console output shows a single line: >

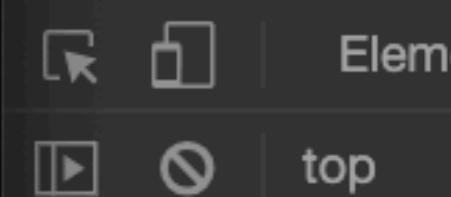
DOM

JS

EXO: DÉCLENCHER UN SCRIPT APRÈS X SEC

style.css > ...

```
1  
2  
3  
4  
5  
6 * {  
7     transition: 1s;  
8 }  
9 h1 {  
10    text-align: center;  
11    margin-top: 120px;  
12    opacity: 0;  
13 }  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23
```



JS

DOM

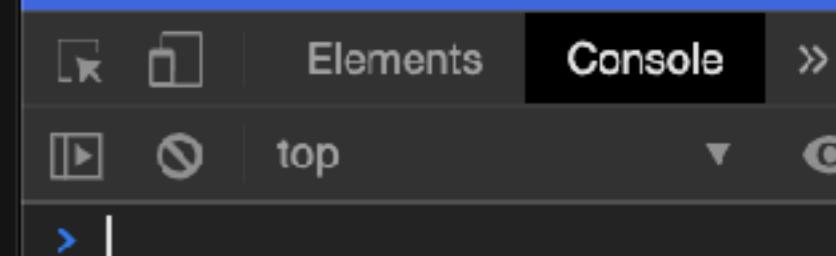
EXO: DÉCLENCHER UN SCRIPT APRÈS X SEC

JS main.js > ...

```
1 // Code JavaScript ici
2
3
4
5 const titre = document.querySelector("h1");
6
7 const txt = setTimeout(function() {
8     titre.textContent = "SALUT C'EST COOL";
9     titre.style.opacity = 1;
10    document.body.style.background = "royalblue";
11 }, 2000)
12
13
14
15
16
17
18
19
20
21
```



SALUT C'EST COOL



DOM



JS

EXO: DÉCLENCHEZ UN SCRIPT TOUT LES X SEC

- Un peu dans le même fonctionnement on a
- setInterval, un code qui va s'exécuter toutes les X secondes.
- Et clearInterval pour le stopper.

DOM

JS

EXO: DÉCLENCHEUR UN SCRIPT TOUT LES X SEC

The screenshot shows a development environment with the following components:

- Code Editor:** The left pane displays the `index.html` file with the following content:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <h1>C'est Parti, Allons Y !</h1>
    <script type="text/javascript" src="main.js"></script>
</body>
</html>
```
- Console:** The bottom right pane shows the browser's developer tools with the "Console" tab selected, displaying the message: `C'est Parti, Allons Y !`.
- Browser Preview:** The right side of the interface shows a preview of the browser window containing the rendered HTML with the heading "C'est Parti, Allons Y !".

DOM

EXO: DÉCLENCHER UN SCRIPT TOUT LES X SEC



The image shows a development environment with the following components:

- Code Editor:** Displays three tabs: `main.js`, `index.html`, and `# style.css`. The `style.css` tab contains the following CSS:

```
h1 {  
    text-align: center;  
    margin-top: 120px;  
    cursor: pointer;  
}
```
- Browser Preview:** Shows the output of `index.html`, which is a simple page with a centered `C'est Parti, Allons Y !` heading.
- Console:** Located at the bottom right, the browser's developer tools show the console tab is active, displaying the message `C'est Parti, Allons Y !`.

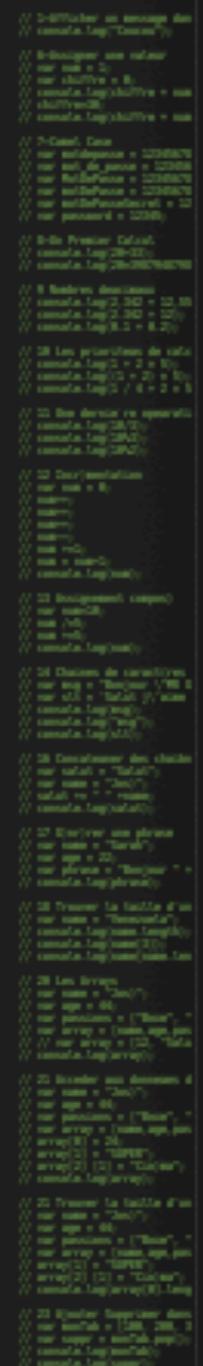
DOM

JS

EXO: DÉCLENCHER UN SCRIPT TOUT LES X SEC

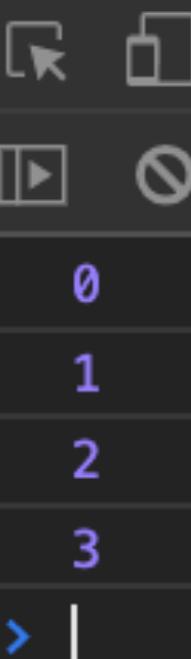
```
let timer = 0;
```

```
const countDown = setInterval(function() {
    console.log(timer);
    timer++;
}, 2000);
```



A vertical list of log entries from the browser's developer tools timeline. The entries show the value of 'timer' increasing from 0 to 3 over time, with each entry timestamped at approximately 2-second intervals.

- 0: 0-Début de l'exécution des fonctions
- 1: 1-Initialiser une valeur pour la variable 'timer' à 0.
- 2: 2-Initialiser une valeur pour la variable 'timer' à 1.
- 3: 3-Initialiser une valeur pour la variable 'timer' à 2.
- 4: 4-Initialiser une valeur pour la variable 'timer' à 3.



A vertical list of log entries from the browser's developer tools timeline. The entries show the value of 'timer' increasing from 0 to 3 over time, with each entry timestamped at approximately 2-second intervals.

- 0: 0-Beginning of script execution.
- 1: 1-Assigning a value to the variable 'timer'.
- 2: 2-Assigning a value to the variable 'timer'.
- 3: 3-Assigning a value to the variable 'timer'.

C

DOM

JS

EXO: DÉCLENCHEZ UN SCRIPT TOUT LES X SEC

The screenshot shows a code editor with a dark theme and a browser window side-by-side.

Code Editor (main.js):

```
main.js — base
JS main.js    X  index.html  # style.css
JS main.js > titre.addEventListener("click") callback > [o] countDown > set
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 let timer = 3;
5
6 titre.addEventListener("click", function(){
7
8     const countDown = setInterval(function() {
9         if(timer > 0) {
10             titre.textContent = timer;
11         } else{
12             titre.textContent = "GO GO GO";
13         }
14         timer--;
15         console.log(timer);
16     }, 2000);
17
18 })
19
20
21
22
23
```

Browser Console:

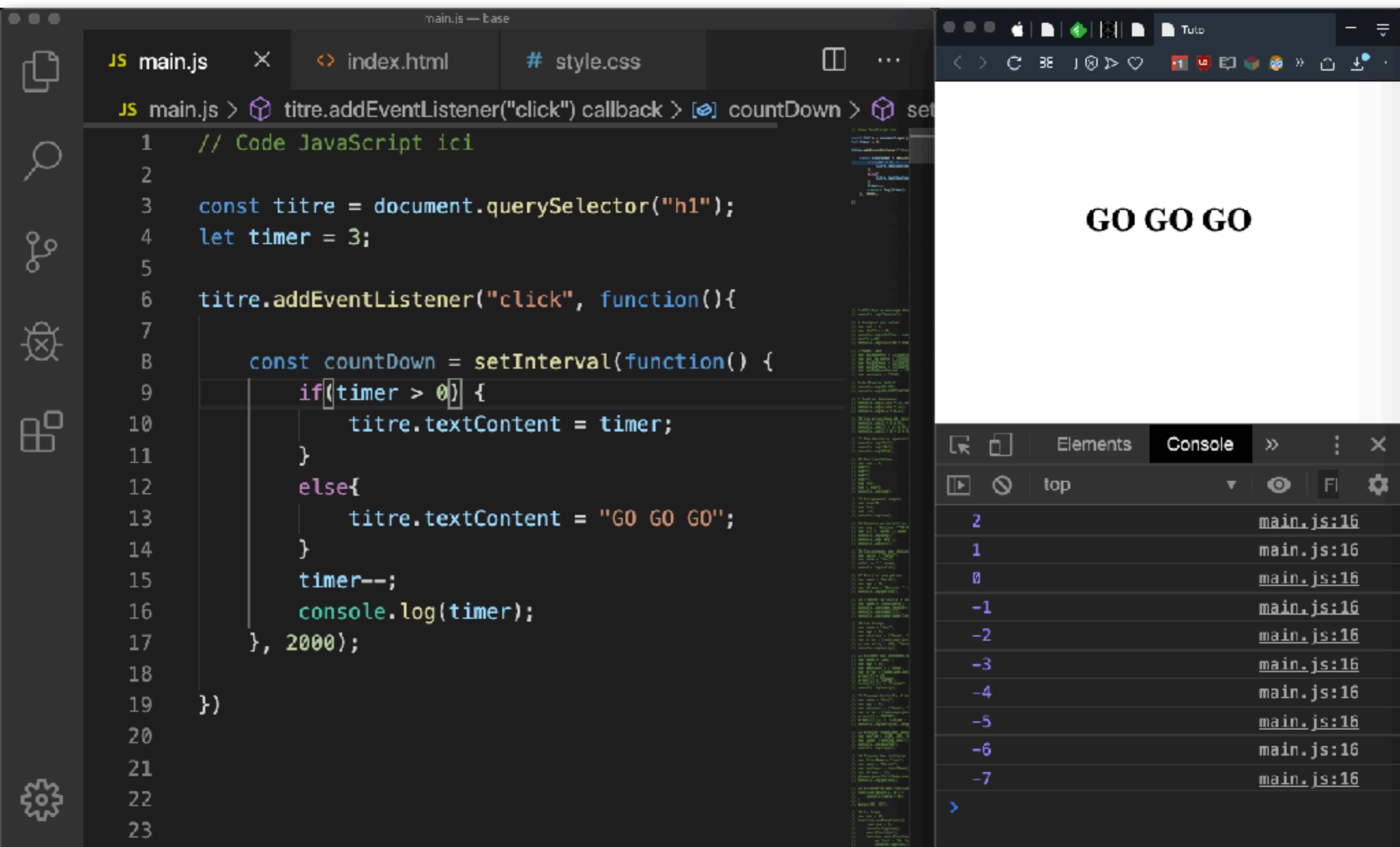
Output	Line Number
2	main.js:16
1	main.js:16
0	main.js:16

A large number '1' is displayed on the right side of the browser interface.

DOM

JS

EXO: DÉCLENCHEZ UN SCRIPT TOUT LES X SEC

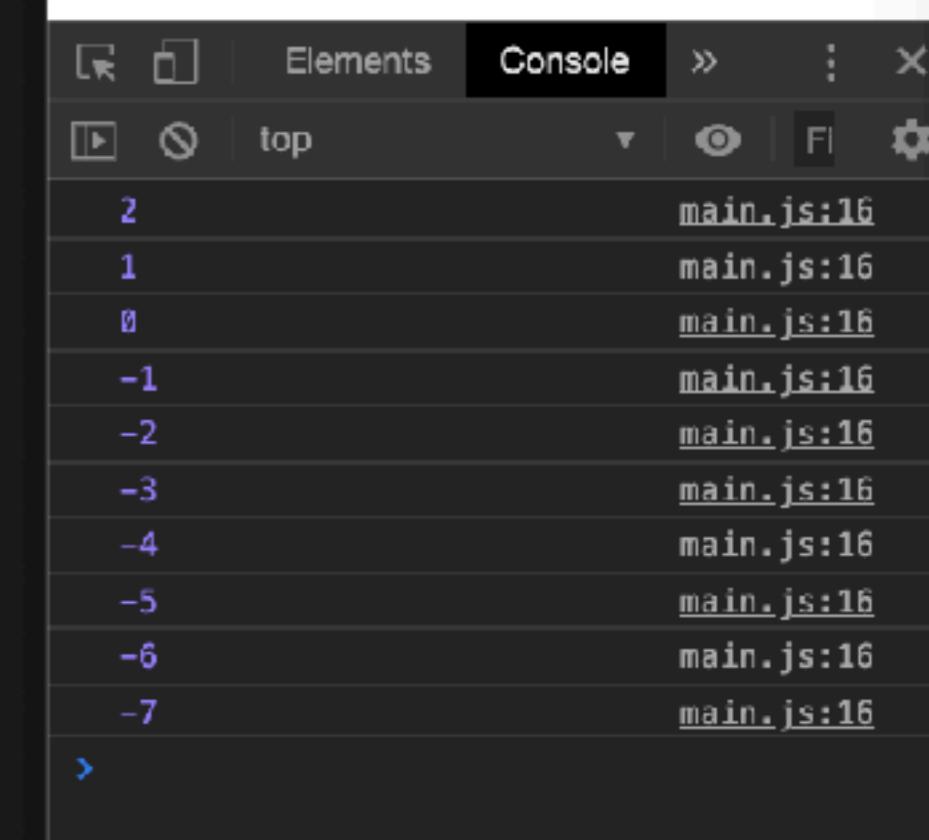


```
main.js — base
JS main.js    X  index.html  # style.css  ...
```

```
JS main.js > titre.addEventListener("click") callback > countDown > set
```

```
1 // Code JavaScript ici
2
3 const titre = document.querySelector("h1");
4 let timer = 3;
5
6 titre.addEventListener("click", function(){
7
8     const countDown = setInterval(function() {
9         if(timer > 0) {
10             titre.textContent = timer;
11         } else{
12             titre.textContent = "GO GO GO";
13         }
14         timer--;
15         console.log(timer);
16     }, 2000);
17 })
18
19
20
21
22
23
```

GO GO GO



Log Entry	Source
2	main.js:16
1	main.js:16
0	main.js:16
-1	main.js:16
-2	main.js:16
-3	main.js:16
-4	main.js:16
-5	main.js:16
-6	main.js:16
-7	main.js:16

DOM

JS

EXO: DÉCLENCHEZ UN SCRIPT TOUT LES X SEC

The image shows a development environment with a code editor and a browser. The code editor on the left has tabs for 'main.js — base', 'main.js', 'index.html', and '# style.css'. The 'main.js' tab is active, displaying the following code:

```
// Code JavaScript ici
const titre = document.querySelector("h1");
let timer = 3;

titre.addEventListener("click", function(){

    const countDown = setInterval(function() {
        if(timer > 0) {
            titre.textContent = timer;
        } else{
            titre.textContent = "GO GO GO";
            clearInterval(countDown);
        }
        console.log(timer);
        timer--;
    }, 2000);
})
```

The browser window on the right shows a yellow header with the text 'JS'. Below it, the title bar says 'Tuto'. The main content area of the browser shows the number '3'.

GO GO GO

The browser's developer tools are open, showing the 'Console' tab with the following log entries:

Time	Message
main.js:16	3
main.js:16	2
main.js:16	1
main.js:16	0



DOM

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

- L'utilisateur tape du texte
- On le récupère
- Et on le console.log sans les voyelles.
- Array includes
- Array join

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a code editor interface with several tabs:

- index.html — base
- main.js
- index.html
- style.css

The index.html tab contains the following code:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <textarea cols="50" rows="10"></textarea>
    <script type="text/javascript" src="main.js"></script>
</body>
</html>
```

The main.js file contains the following script:

```
function removeVowels(text) {
    return text.replace(/[aeiouAEIOU]/g, '');
}

const textarea = document.querySelector('textarea');
const output = document.querySelector('p');

textarea.addEventListener('input', () => {
    const text = textarea.value;
    output.textContent = removeVowels(text);
});
```

The browser window shows the result of running the script. The text area contains "bonjour tout le monde". The output paragraph below it also contains "bonjour tout le monde", but with all vowels removed, resulting in "bnnjr tt l mnd".

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a code editor interface with several tabs: 'main.js', 'index.html', and '# style.css'. The '# style.css' tab is active, displaying the following CSS code:

```
# style.css > ...
1
2
3  textarea {
4      display: block;
5      margin: 60px auto;
6  }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

To the right of the code editor is a browser window with the address bar showing 'Tuto'. The browser's developer tools are open, with the 'Console' tab selected. The console output is as follows:

```
Elements Console >
top
```

JS

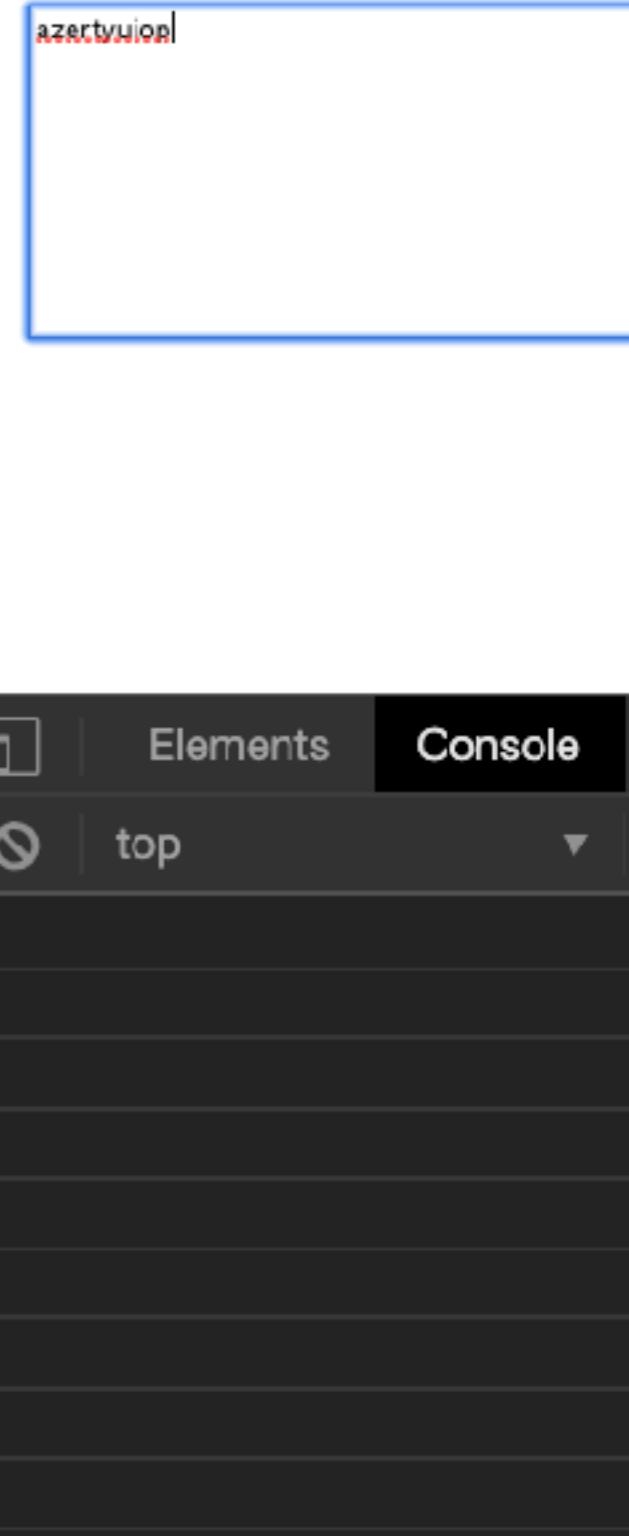
DOM

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

```
main.js > ...
// Code JavaScript ici

const leTexte = document.querySelector("textarea");
const txt = [];
const voyelles = ["a","e","i","o","u","y"];

leTexte.addEventListener("keypress", function(unEvent){
    const touche = unEvent.key;
    console.log(touche);
});
```



azertyuiop

```
Elements | Console
top
a
z
e
r
t
y
u
i
o
```

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The image shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following code:

```
main.js — base
JS main.js    X  index.html  # style.css
JS main.js > leTexte.addEventListener("keypress") callback
1 // Code JavaScript ici
2
3 const leTexte = document.querySelector("textarea");
4 const txt = [];
5 const voyelles = ["a","e","i","o","u","y"];
6
7 leTexte.addEventListener("keypress", function(unEvent){
8     const touche = unEvent.key;
9
10    if(voyelles.includes(touche)){
11        txt.push(touche);
12    }
13    console.log(txt);
14});
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Browser: The browser window shows an alert dialog with the message "alert('voyelles')". The developer tools console shows the following log entries:

Log	File	Line
▶ ["a"]	main.js:13	
▶ ["a"]	main.js:13	
▶ (2) ["a", "e"]	main.js:13	
▶ (2) ["a", "e"]	main.js:13	
▶ (2) ["a", "e"]	main.js:13	
▶ (3) ["a", "e", "y"]	main.js:13	
▶ (4) ["a", "e", "y", "u"]	main.js:13	
▶ (5) ["a", "e", "y", "u", "i"]	main.js:13	
▶ (6) ["a", "e", "y", "u", "i", "o"]	main.js:13	

DOM

JS

EXO: SUPPRIMER LES VOYELLES DU CLAVIER

The screenshot shows a code editor on the left and a browser window on the right.

Code Editor (main.js):

```
main.js — base
JS main.js  X  index.html  # style.css
JS main.js > leTexte.addEventListener("keypress") callback
1 // Code JavaScript ici
2
3 const leTexte = document.querySelector("textarea");
4 const txt = [];
5 const voyelles = ["a","e","i","o","u","y"];
6
7 leTexte.addEventListener("keypress", function(unEvent){
8     const touche = unEvent.key;
9
10    if(!voyelles.includes(touche)) {
11        txt.push(touche);
12    }
13    console.log(txt);
14});
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Browser Window:

- Address bar: `http://127.0.0.1:5500/Tuto`
- Alert dialog: `zertyuiop`
- Console tab in developer tools:
 - Output:

```
▶ []
▶ ["z"]
▶ ["z"]
▶ (2) ["z", "r"]
▶ (3) ["z", "r", "t"]
▶ (4) ["z", "r", "t", "p"]
```

JS

DOM

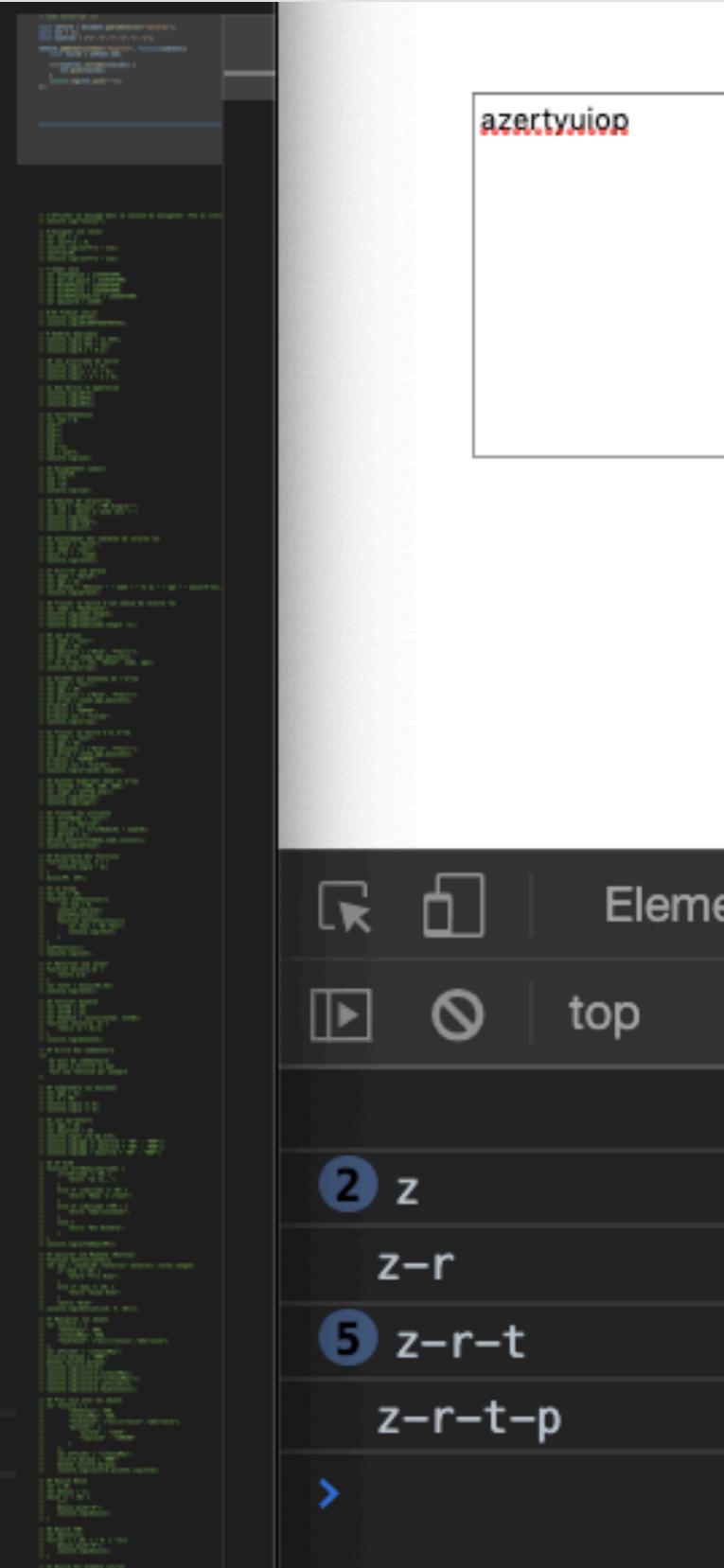
EXO: SUPPRIMER LES VOYELLES DU CLAVIER

// Code JavaScript ici

```
const leTexte = document.querySelector("textarea");
const txt = [];
const voyelles = ["a","e","i","o","u","y"];

leTexte.addEventListener("keypress", function(unEvent){
    const touche = unEvent.key;

    if(!voyelles.includes(touche)) {
        txt.push(touche);
    }
    console.log(txt.join("-"));
});
```



DOM

EXO: DÉSACTIVER UN BOUTON

JS

- Une astuce classique mais très utile.
- On va avoir une zone de texte et un bouton send
- On désactive le bouton si le message est trop long.

DOM

JS

EXO: DÉSACTIVER UN BOUTON

The screenshot shows a code editor interface with three tabs: `index.html`, `main.js`, and `# style.css`. The `index.html` tab contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <title>Tuto</title>
6   <link rel="stylesheet" href="style.css">
7 </head>
8 <body>
9   <!-- Code HTML -->
10  <form>
11    <textarea cols="30" rows="5"></textarea>
12    <button>Send</button>
13  </form>
14
15  <script type="text/javascript" src="main.js"></script>
16 </body>
17 </html>
```

The browser window shows a simple form with a text area and a button labeled "Send". The button is currently disabled, as indicated by its gray color and lack of a click effect.

JS

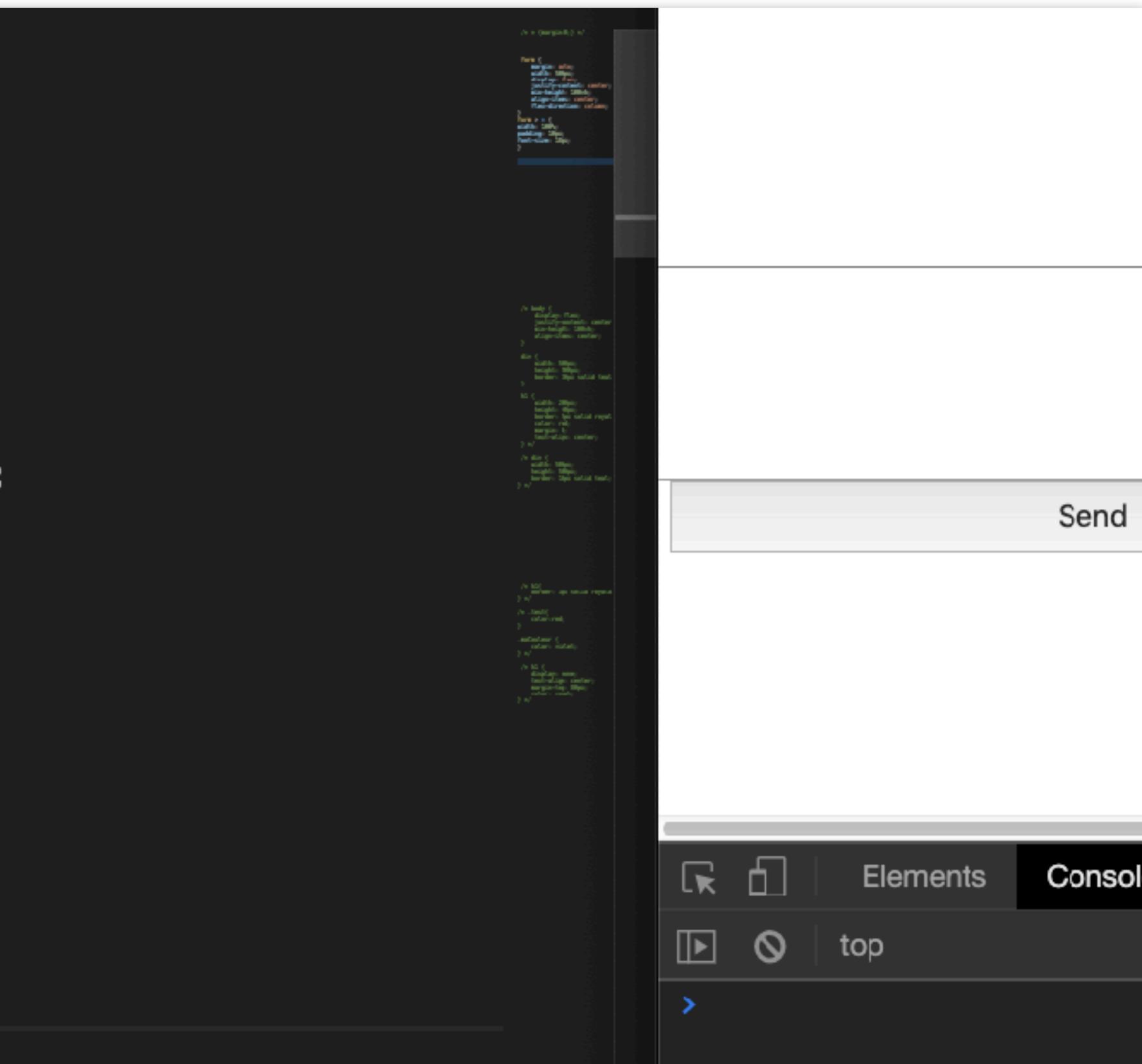
DOM

EXO: DÉSACTIVER UN BOUTON

```
/* * {margin:0;} */

form {
    margin: auto;
    width: 500px;
    display: flex;
    justify-content: center;
    min-height: 100vh;
    align-items: center;
    flex-direction: column;
}

form > * {
    width: 100%;
    padding: 10px;
    font-size: 18px;
}
```



DOM

JS

EXO: DÉSACTIVER UN BOUTON

The screenshot shows a code editor and a browser window side-by-side.

Code Editor: The main.js file contains the following code:

```
main.js — base
index.html    JS main.js    # style.css
JS main.js > txt.addEventListener("keyup") callback
1 // Code JavaScript ici
2
3
4 const txt = document.querySelector("textarea");
5 const btn = document.querySelector("button");
6
7 txt.addEventListener("keyup", function() {
8     btn.disabled = true;
9 })
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Browser: The browser shows a simple form with a text area and a button. The button is currently disabled, as indicated by its grayed-out appearance.

DOM

JS

EXO: DÉSACTIVER UN BOUTON

The image shows a development environment with a code editor and a browser window.

Code Editor: The main.js file contains the following code:

```
// Code JavaScript ici
const txt = document.querySelector("textarea");
const btn = document.querySelector("button");

txt.addEventListener("keyup", function() {
  btn.disabled= txt.value.length >= 5 ? true : false;
})
```

Browser Window: A simple form with a text area and a button is displayed. The button is initially enabled. As soon as the user types 5 or more characters into the text area, the button becomes disabled (grayed out).

DOM PREVENTDEFAULT

JS

JS

DOM PREVENTDEFAULT

The screenshot shows a development environment with a code editor and a browser window.

Code Editor:

- index.html — base**:
Content:

```
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Tuto</title>
6  |   <link rel="stylesheet" href="style.css">
7  </head>
8  <body>
9  <!-- Code HTML -->
10 <form>
11 |   <input type="text" placeholder=" Votre nom"/>
12 |   <button type="submit">Connexion</button>
13 </form>
14
15 <script type="text/javascript" src="main.js"></script>
16 </body>
17 </html>
```
- main.js**:
Content:

```
>
```

Browser Window:

- Title Bar:** Tuto
- Form Fields:** Votre nom, Connexion
- Console Tab:** Elements, Console, top

DOM PREVENTDEFAULT

JS

The image shows a development environment with a code editor and a browser. On the left, a code editor displays two files: `style.css` and `index.html`. The `style.css` file contains the following CSS:

```
1
2  form {
3      margin: auto;
4      width: 500px;
5      display: flex;
6      justify-content: center;
7      min-height: 100vh;
8      align-items: center;
9      flex-direction: column;
10 }
11 form > * {
12     width: 100%;
13     padding: 10px;
14     font-size: 18px;
15 }
```

The `index.html` file contains a simple HTML form:

```
<form>
    Votre nom
    Connexion
</form>
```

To the right, a browser window shows a login interface with fields for "Votre nom" and "Connexion". At the bottom, the browser's developer tools are visible, specifically the Elements and Console tabs.

JS

DOM PREVENTDEFAULT

The screenshot shows a browser developer tools interface. On the left, there is a code editor window titled "main.js — base" containing the following JavaScript code:

```
JS main.js    X  index.html  # style.css
JS main.js >  monForm.addEventListener("submit") callback
1 // Code JavaScript ici
2
3
4 const monForm = document.querySelector("form");
5
6 monForm.addEventListener("submit", function(){
7     console.log("OK");
8 })
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

To the right of the code editor is a browser window displaying a simple form. The form has a text input field labeled "Votre nom" and a button labeled "Connexion". At the bottom of the browser window, the developer tools' "Console" tab is active, showing the message "OK" in the output area.

JS

DOM PREVENTDEFAULT

The image shows a development environment with a code editor and a browser window.

Code Editor (main.js):

```
main.js — base
JS main.js    X  index.html  # style.css
JS main.js > ...
1 // Code JavaScript ici
2
3
4 const monForm = document.querySelector("form");
5
6 monForm.addEventListener("submit", function(event){
7     event.preventDefault();
8     console.log("OK");
9 })
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Browser Screenshot:

A browser window displays a form with a single input field containing "azertyuiop". A button labeled "Connexion" is visible below the input field. The browser's developer tools are open, showing the "Elements" tab selected. In the bottom right corner of the browser window, the word "OK" is displayed, indicating that the preventDefault() method was successful in stopping the default form submission behavior.

JS

DOM PREVENTDEFAULT

The screenshot shows a browser developer tools interface with several panels:

- Code Editor:** A dark-themed code editor window titled "main.js — base". It contains the following JavaScript code:

```
JS main.js    X  index.html  # style.css
JS main.js > ⚡ monForm.addEventListener("submit") callback
1 // Code JavaScript ici
2
3
4 const monForm = document.querySelector("form");
5
6 monForm.addEventListener("submit", function(event){
7     event.preventDefault();
8     console.log("OK");
9     monForm.reset();
10})
11
12
13
14
15
16
17
18
19
20
21
22
23
```
- Preview:** A browser window showing a simple form with a text input field labeled "Votre nom" and a button labeled "Connexion".
- Console:** A panel at the bottom showing the output of the code execution. It displays the message "OK" followed by the file path "main.js:8".

DOM

EXO: CRÉER UN ÉDITEUR DE TEXTE

- Enregistrer sur le navigateur de l'utilisateur des données.
- Et les ré utiliser par la suite même si il ferme son navigateur, éteint l'ordinateur, il reviens 4j plus tard
- Les infos qu'il aura rentré seront toujours là et on pourra les réutiliser
- Markdown

DOM

JS

EXO: CRÉER UN ÉDITEUR DE TEXTE

The image shows a development environment with two code editors and a browser window.

Code Editors:

- index.html — base**:
Content:

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <title>Tuto</title>
6   <link rel="stylesheet" href="style.css">
7 </head>
8 <body>
9 <!-- Code HTML -->
10 <textarea cols="50" rows="10"></textarea>
11 <div></div>
12
13 <script type="text/javascript" src="main.js"></script>
14 </body>
15 </html>
```
- main.js**: (File icon)

Browser Window:

- Title bar: Tuto
- Address bar: C BE J @ D
- Content area: An empty text area.
- Console tab: Elements (selected), Console, top, F1, settings.

DOM

EXO: CRÉER UN ÉDITEUR DE TEXTE

JS

The screenshot shows a code editor interface with several tabs: 'main.js', 'index.html', and '# style.css'. The '# style.css' tab is active, displaying the following CSS code:

```
# style.css — base
# style.css >  div
1
2
3
4  textarea {
5    display: block;
6    margin: 20px auto;
7    padding: 10px;
8  }
9  div {
10   width: 500px;
11   margin: auto;
12 }
```

To the right of the code editor is a browser window showing a simple page with a single empty text area.

Below the browser window, the developer tools are open, showing the 'Elements' tab selected. The 'Console' tab is also visible, showing the output 'top'.

JS

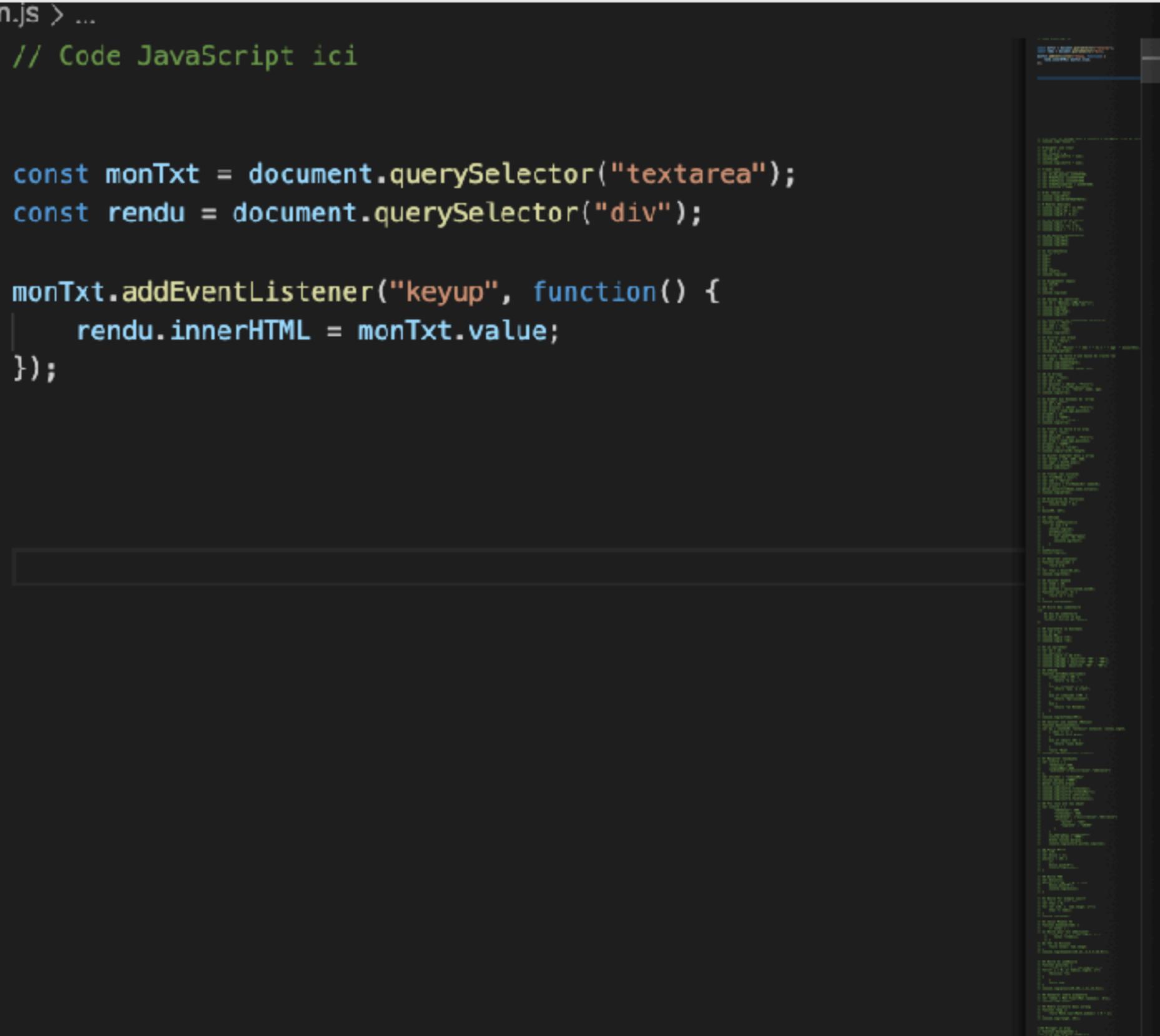
DOM

EXO: CRÉER UN ÉDITEUR DE TEXTE

```
main.js > ...
// Code JavaScript ici

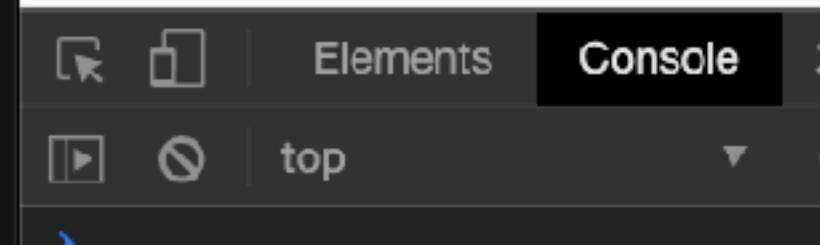
const monTxt = document.querySelector("textarea");
const rendu = document.querySelector("div");

monTxt.addEventListener("keyup", function() {
  rendu.innerHTML = monTxt.value;
});
```



bonjour je suis Philéas un chat de 432 ans |

bonjour je suis Philéas un chat de 432 ans



JS

DOM

EXO: CRÉER UN ÉDITEUR DE TEXTE

The screenshot shows a web browser window with the URL <https://cdnjs.com/libraries/marked>. The page title is "marked". The top navigation bar includes links for "Community", "About", "API", "Browse Libraries", "Support us!", "git stats", and "Network & Status". On the left, there's a sidebar with "CDNJS on GitHub" and a "cdnjs" logo. The main content area displays the library details: "marked" version 0.8.0, 398 commits, 20389 stars, and 2633 forks. It provides links to GitHub (<https://github.com/markedsjs/marked.git>), commits/history, cdnjs API, repository, and current license MIT. A brief description states it's a markdown parser built for speed, supporting *markdown, markup, html*. Below this, there's a dropdown for "Version" set to 0.8.0, a "CDN provider" dropdown set to "Cloudflare", and a "Copy" button for the URLs. The URLs listed are:

- <https://cdnjs.cloudflare.com/ajax/libs/marked/0.8.0/marked.js> (Copy, Copy Url, Copy SRI)
- <https://cdnjs.cloudflare.com/ajax/libs/marked/0.8.0/marked.min.js> (Copy Script Tag, Copy Script Tag without SF)
- <https://cdnjs.cloudflare.com/ajax/libs/marked/0.8.0/marked.min.js.map> (Copy Script Tag, Copy Script Tag without SF)

Related Tutorials

Write your own awesome web development tutorials for the libraries on cdnjs! [Submit your community driven tutorials now!](#)

DOM

JS

EXO: CRÉER UN ÉDITEUR DE TEXTE

The screenshot shows a development environment with the following components:

- Code Editor:** On the left, there is a code editor with two tabs: "index.html" and "main.js".
 - index.html:** Contains the following HTML code:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Tuto</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/marked/0.3.6/marked.min.js"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Code HTML -->
    <textarea cols="50" rows="10"></textarea>
    <div></div>
</body>
</html>
```
 - main.js:** Contains the following JavaScript code:

```
<script type="text/javascript" src="main.js"></script>
```
- Browser Preview:** On the right, a browser window titled "Tuto" is displayed, showing a blank white page.
- Console:** At the bottom, a developer console is open, showing the "Console" tab selected. It contains the following output:

```
>
```

DOM

EXO: CRÉER UN ÉDITEUR DE TEXTE

JS

main.js > ...

```
// Code JavaScript ici
```

```
const monTxt = document.querySelector("textarea");
const rendu = document.querySelector("div");
```

```
monTxt.addEventListener("keyup", function() {
    rendu.innerHTML = marked(monTxt.value);
});
```

```
[REDACTED]
```



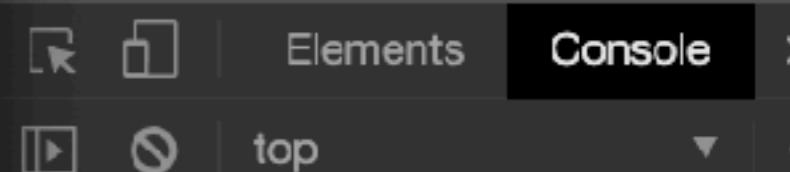
```
# tyuiuytre
1. azertyujk.nbvcxwx bgfdxsd
2.vfexcvfdfef

sdfghjkfdgf
```

tyuiuytre

```
1. azertyujk.nbvcxwx bgfdxsd
2. vfexcvfdfef
```

sdfghjkfdgf



DOM LOCAL STORAGE

- On peut sauvegarder des données sur le navigateur de l'utilisateur.
- Avec le local Storage (un peu comme les cookies)
- Local storage fonctionne avec 2 actions.
 - `setItem` pour enregistrer les informations
 - `getItem` pour récupérer les informations
- Tout est enregistrer sous forme d'une grande chaîne de caractère.

JS

@

LOCAL STORAGE

The image shows a code editor on the left and a browser developer tools interface on the right.

Code Editor (main.js):

```
main.js — base
JS main.js  X  index.html  # style.css
JS main.js >  monTxt.addEventListener("keyup") callback
1 // Code JavaScript ici
2
3
4 const monTxt = document.querySelector("textarea");
5 const rendu = document.querySelector("div");
6
7 monTxt.addEventListener("keyup", function() {
8     localStorage.setItem("duTexte", monTxt.value);
9     rendu.innerHTML = marked(monTxt.value);
10});
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Browser Developer Tools (Application tab):

Key	Value
duTexte	oiuodiudioudou

Text Overlay:

Select a value to preview

JS

DOM LOCAL STORAGE

The image shows a code editor window and a browser developer tools window side-by-side.

Code Editor (left):

- File: main.js
- Content:

```
main.js — base
JS main.js  X  index.html  # style.css
JS main.js >  monTxt.addEventListener("keyup") callback
1 // Code JavaScript ici
2
3
4 const monTxt = document.querySelector("textarea");
5 const rendu = document.querySelector("div");
6
7 monTxt.addEventListener("keyup", function() {
8     localStorage.setItem("duTexte", monTxt.value);
9     rendu.innerHTML = marked(monTxt.value);
10});
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Browser Developer Tools (right):

- Elements tab is active.
- Application tab is selected.
- LocalStorage table:

Key	Value
duTexte	oiuoduioduiodou
- A message at the bottom says "Select a value to preview".

JS

DOM LOCAL STORAGE

```
// Code JavaScript ici

const monTxt = document.querySelector("textarea");
const rendu = document.querySelector("div");

monTxt.value = localStorage.getItem("duTexte");

monTxt.addEventListener("keyup", function() {
  localStorage.setItem("duTexte", monTxt.value);
  rendu.innerHTML = marked(monTxt.value);
});
```

oiuodiudoiuadu

The screenshot shows the Chrome DevTools interface with the Application tab selected. In the main pane, there is a 'Manifest' icon, a 'Service Workers' icon, and a 'Clear storage' button. Below these, under the 'Storage' section, is a 'Local Storage' entry for the URL 'http://127.0.0.1:5'. The 'duTexte' key is listed under this entry with the value 'oiuodiudoiuadu'. On the right side, there is a 'Filter' input field and a 'Key' dropdown menu.

JS

DOM LOCAL STORAGE

The image shows a development environment with a code editor and a browser window.

Code Editor: The left side shows a code editor with tabs for `main.js`, `index.html`, and `# style.css`. The `main.js` tab contains the following JavaScript code:

```
// Code JavaScript ici

const monTxt = document.querySelector("textarea");
const rendu = document.querySelector("div");

monTxt.value = localStorage.getItem("duTexte");

if (monTxt.value) {
    rendu.innerHTML = marked(localStorage.getItem("duTexte"));
}

monTxt.addEventListener("keyup", function() {
    localStorage.setItem("duTexte", monTxt.value);
    rendu.innerHTML = marked(monTxt.value);
});
```

Browser Preview: The right side shows a browser window displaying the result of the code execution. The page content includes:

oiuodiudioudo
SALUT
1. YO
2. ULTRA Cool
3. BIG UP

Developer Tools: At the bottom, the browser's developer tools are open, specifically the Application tab under Storage. It shows the following local storage items:

Key	Value
duTexte	oiuodiudioudo...

A message at the bottom right of the developer tools panel says: "Select a value to preview".

POO

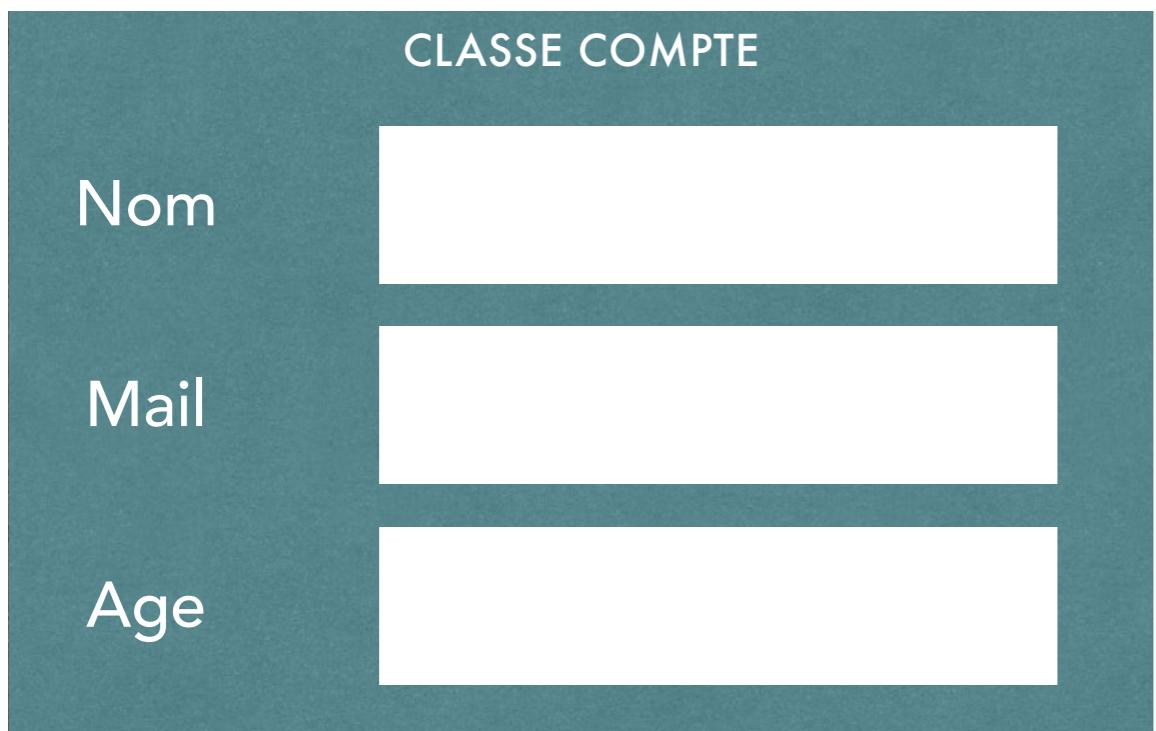
CLASSES

- On a déjà vu les objets comment les créer et y accéder
- Mais si on a besoin de créer plusieurs objets du même type ?
- Une classe est un modèle pour pouvoir créer plusieurs objets de même type.
- + facile, + rapide, + fiable.

POO

CLASSES

JS



INSTANCE	
Nom	José Garcia
Mail	<u>jgarcia@truc.com</u>
Age	45

INSTANCE	
Nom	Sarah
Mail	<u>sarah@gmail.com</u>
Age	29

POO
CLASSES

```
class Film {  
    constructor(title, author, duration){  
        this.title = title;  
        this.author = author;  
        this.duration = duration;  
    }  
}
```



POO

CLASSES

- Mot clé class
- constructor d'une classe = fonction appelée quand on crée une nouvelle instance de cette classe
- This fait référence à la nouvelle instance qui sera créée
- Notation dot

POO CLASSES

```
let myFilm = new Film("Taxi 3", "Luc Besson", 300);
// On va créer l'objet :
{
  title: "Taxi 3",
  author: "Luc Besson",
  duration: 300
}
```

POO

MÉTHODE D'INSTANCE

- Fonctions associées à une classe, agit sur une instance de cette classe.

POO

MÉTHODE D'INSTANCE

```
class Car = {  
    constructor(){  
        this.engineIsRunning = false;  
    }  
    startEngine(){  
        this.engineIsRunning = true;  
    }  
  
let brandNewCar = new Car;  
brandNewCar.startEngine();  
  
let blueCar = new Car;  
blueCar.startEngine();  
  
let redCar = new Car;  
redCar.startEngine();
```

JS

