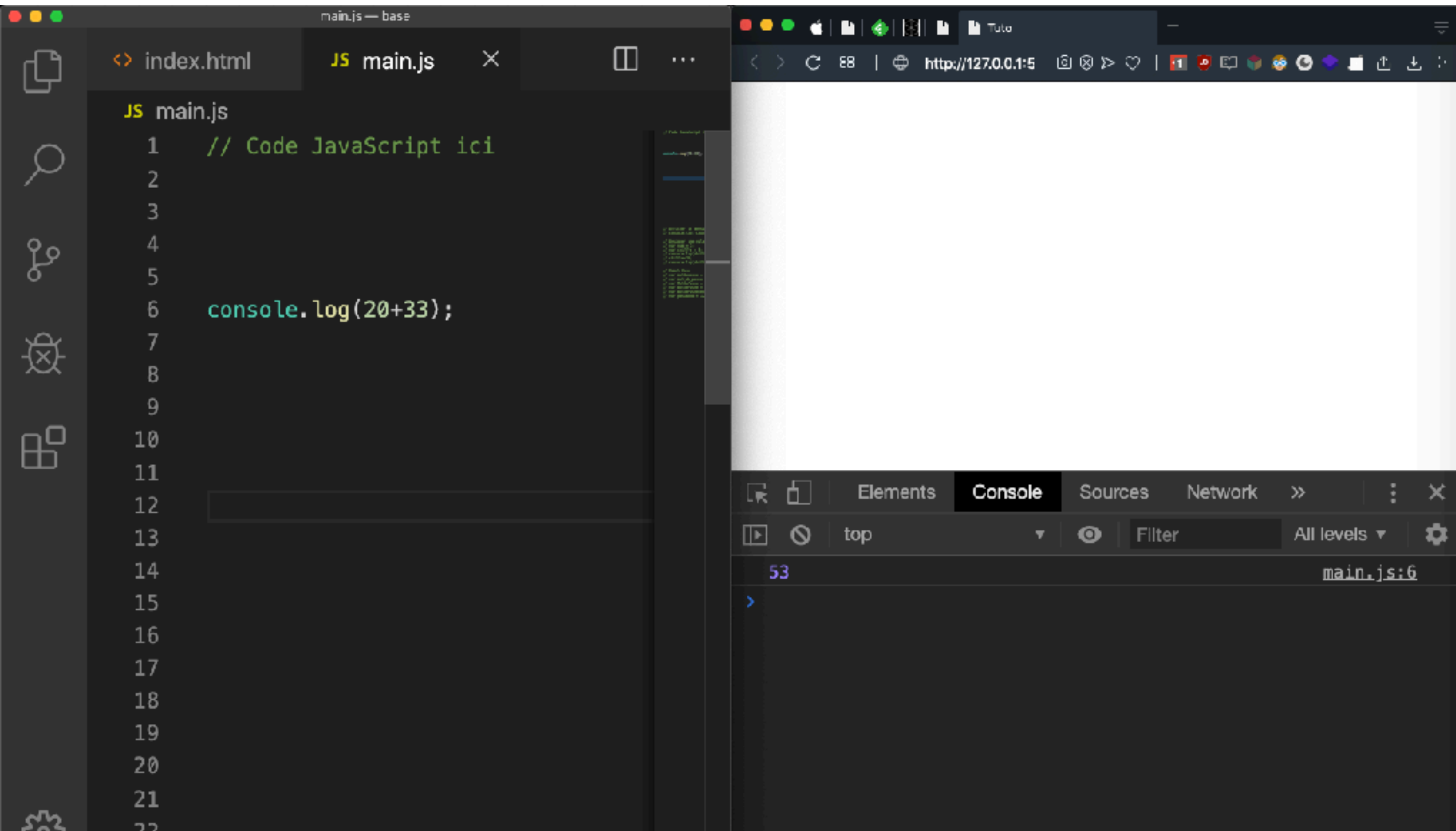


BASE JAVASCRIPT

EXO: CALCUL



BASE JAVASCRIPT

EXO: CALCUL



The image shows a web browser window with a dark theme. The address bar displays `http://127.0.0.1:5`. The browser's developer tools are open, showing the **Console** tab. The console displays the output of the JavaScript code: `79758975974860` and `main.js:7`. The code editor on the left shows the following JavaScript code in `main.js`:

```
1 // Code JavaScript ici
2
3
4
5
6 // console.log(20+33);
7 console.log(20*3987948798743);
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

BASE JAVASCRIPT

EXO: CALCUL



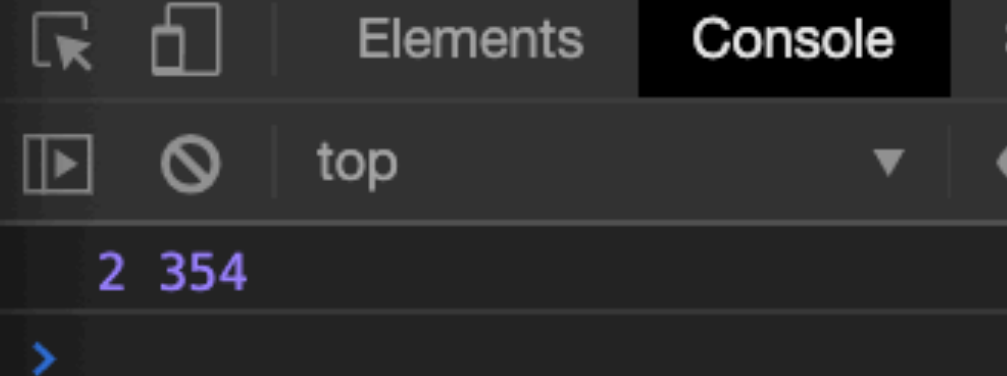
- On a plusieurs opérateurs pour faire des calculs
- + Additionner
- - Soustraire
- * Multiplier
- / Diviser

BASE JAVASCRIPT

NOMBRE DÉCIMAUX



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



BASE JAVASCRIPT

NOMBRE DÉCIMAUX



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



Elements



top

2 354 555



BASE JAVASCRIPT

NOMBRE DÉCIMAUX



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

```
// 5-Clicker on the  
console.log("Click")  
  
// 6-Designing our ra  
var mae = 0;  
var shuffle = 0;  
console.log(shuffle)  
shuffle+=1;  
console.log(shuffle)  
  
// 7-Click, Click  
var antiference =  
var antiference =  
var antiference =  
var antiference =  
var antiference =  
var password = 12  
  
// 8-6- Premier Data  
console.log(20-10)  
console.log(20-10)
```



Element



top

14.342



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 on 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.

BASE JAVASCRIPT

NOMBRE DÉCIMAUX



The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the following JavaScript code in `main.js`:

```
1 // Code JavaScript ici
2
3
4
5 // console.log(2,342 + 12,555);
6 // console.log(2.342 + 12);
7 console.log(0.1 + 0.2);
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

The browser window on the right shows the console output for the code. The console displays the result of the calculation `0.1 + 0.2` as `0.30000000000000004`, which is the expected floating-point precision error in JavaScript. The console also shows the source file `main.js` and line `7`.

BASE JAVASCRIPT

PRIORITÉS DE CALCUL



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

```
// 2-0774368 on 2019-01-01  
// console.log("Case")  
  
// 8-0000000 on 2019-01-01  
// var x = 10;  
// console.log(x/10);  
// console.log(x/10);  
// console.log(x/10);  
  
// 7-0000000 on 2019-01-01  
// var x = 10;  
// var y = 10;  
// console.log(x/10);  
// console.log(x/10);  
// console.log(x/10);  
// console.log(x/10);  
  
// 6-0000000 on 2019-01-01  
// console.log(10/10);  
// console.log(10/10);  
  
// 5-0000000 on 2019-01-01  
// console.log(10/10);  
// console.log(10/10);  
// console.log(10/10);
```



Elements

Con



top

11



BASE JAVASCRIPT

PRIORITÉS DE CALCUL



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

```
// 1- Afficher un message  
console.log("Coucou");  
  
// 2- Définir une variable  
var nom = "John";  
var age = 30;  
console.log(nom);  
console.log(age);  
  
// 3- Calculer la somme de deux nombres  
var x = 10;  
var y = 5;  
var somme = x + y;  
console.log(somme);  
  
// 4- Afficher le résultat du calcul  
console.log(somme);
```



Elements



top

15



BASE JAVASCRIPT

PRIORITÉS DE CALCUL



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

```
// 1- Afficher un message  
console.log("Coucou");  
  
// 2- Définir une variable  
var nom = "John";  
var prenom = "Doe";  
console.log(nom + " " + prenom);  
  
// 3- Calculer la somme de deux nombres  
var a = 10;  
var b = 5;  
var somme = a + b;  
console.log(somme);  
  
// 4- Afficher la date du jour  
console.log(new Date());  
  
// 5- Afficher le résultat d'une opération  
console.log(2 + 3 * 4);  
console.log(2 * 3 + 4);
```



Elements

Co



top

10.25



BASE JAVASCRIPT

EXO: OPÉRATION



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

BASE JAVASCRIPT

EXO: OPÉRATION



The screenshot displays a web browser window with a dark theme. The address bar shows the URL `http://127.0.0.1:5`. The browser's developer tools are open, with the 'Console' tab selected. The console shows a log message `3.3333333333333335` from `main.js:3`. The code editor on the left shows the following JavaScript code in `main.js`:

```
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
```

BASE JAVASCRIPT

EXO: OPÉRATION



The screenshot displays a web browser window with a dark theme. The address bar shows the URL `http://127.0.0.1:5`. The browser's developer tools are open, with the 'Console' tab selected. A single log entry is visible, showing the output of the JavaScript code: `3.3333333333333335`, with the source file `main.js:3` indicated. On the left, a code editor window titled 'main.js — base' shows the following code:

```
JS main.js
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
```

BASE JAVASCRIPT

EXO: OPÉRATION



Voici une division posée :

$$\begin{array}{r|l} 35 & 12 \\ -24 & \\ \hline 11 & 2 \end{array}$$

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
```

Debugger console output:

- 6-00000000 console.log: 10%3 = 1
- 6-00000001 console.log: 10%2 = 0

Debugger UI:

- Elements: top
- 1
- 0