**Nombres et calculs**

Table des matières

[Nombres et Calculs JS 3](#_Toc68872015)

[L’opérateur modulo 6](#_Toc68872016)

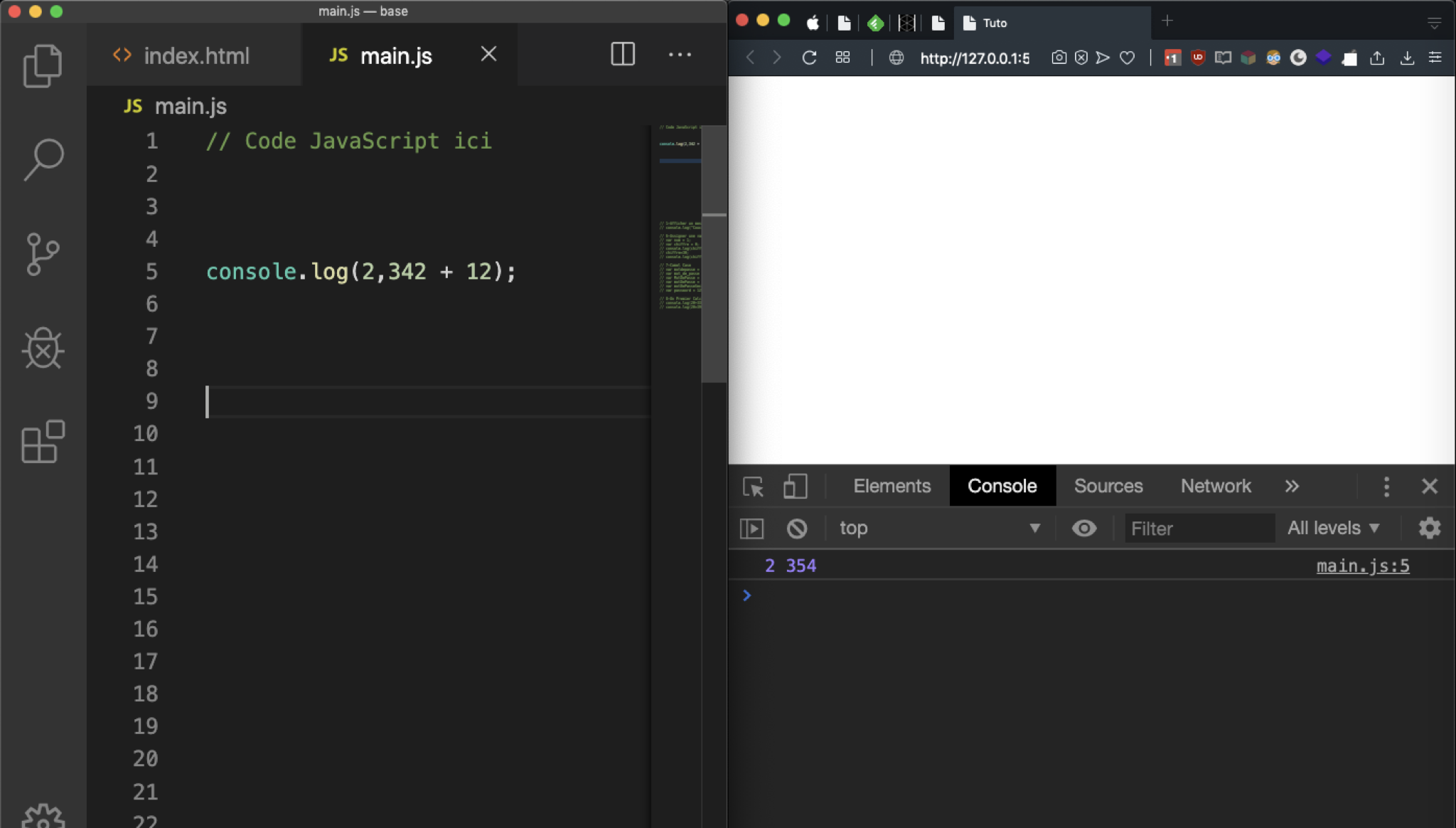
[Incrémentation 7](#_Toc68872017)

[Assignement Composé 9](#_Toc68872018)

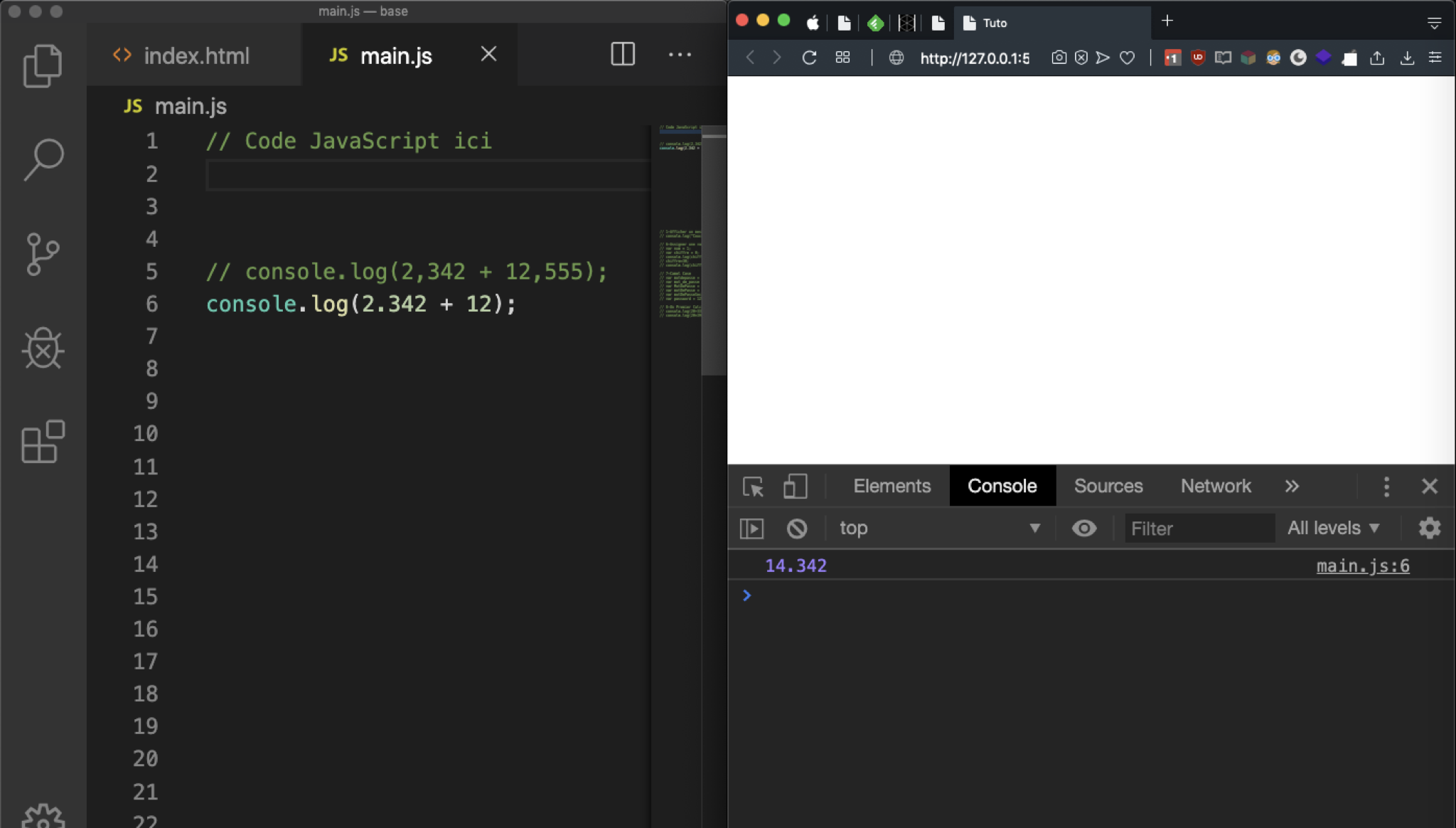
# Nombres et Calculs JS

On a plusieurs opérateurs pour faire des calculs

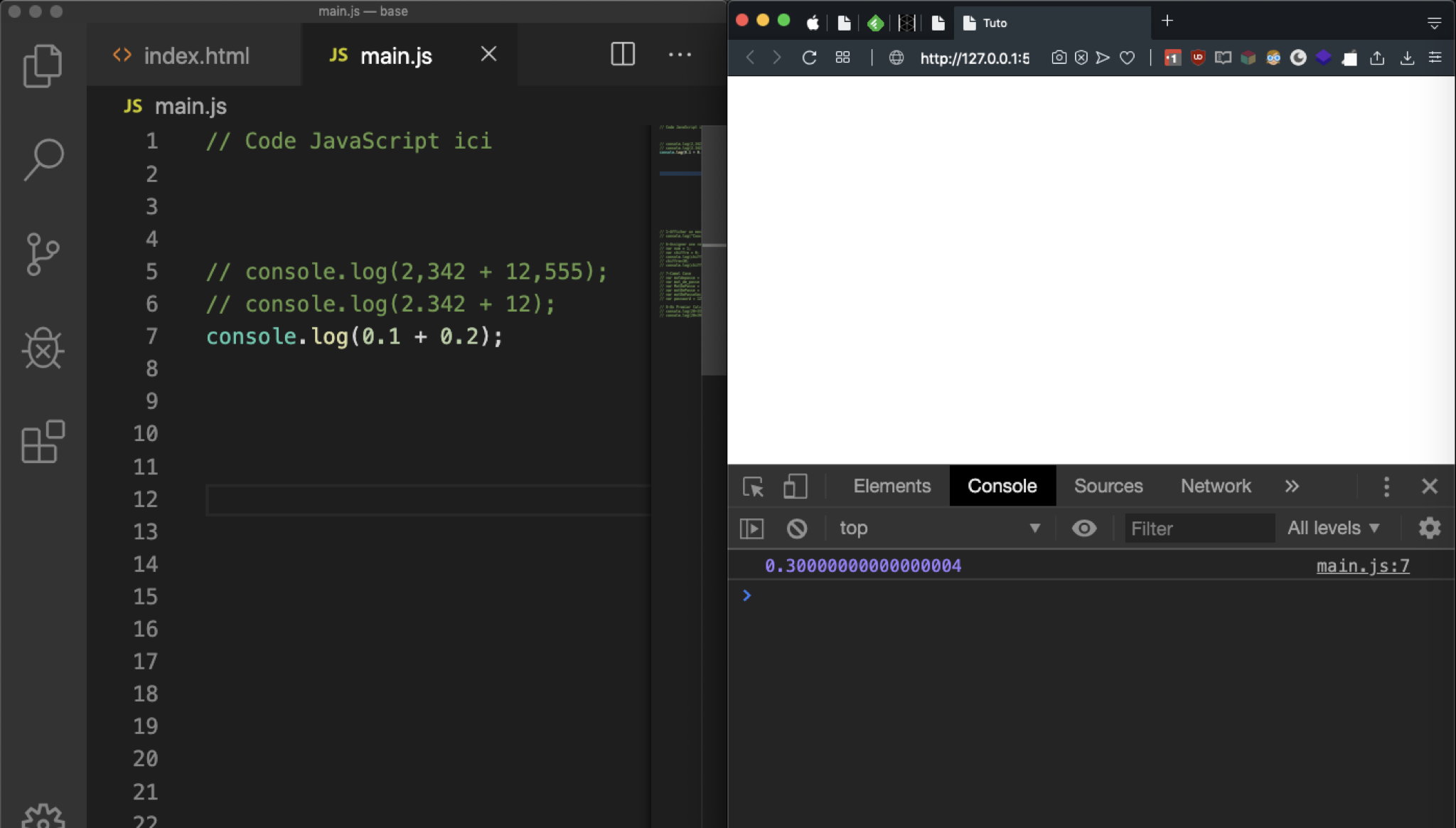
* + Additionner
* - Soustraire
* \* Multiplier
* / Diviser



En français on écrit les décimales avec la « , » mais en US on écrit les décimales avec un « . » Ici le console.log affiche 2 et le calcul (342+12).



Pour les nombres décimaux à virgule on utilise le point



Warning : Le « BUG » de JS avec les calculs décimaux.

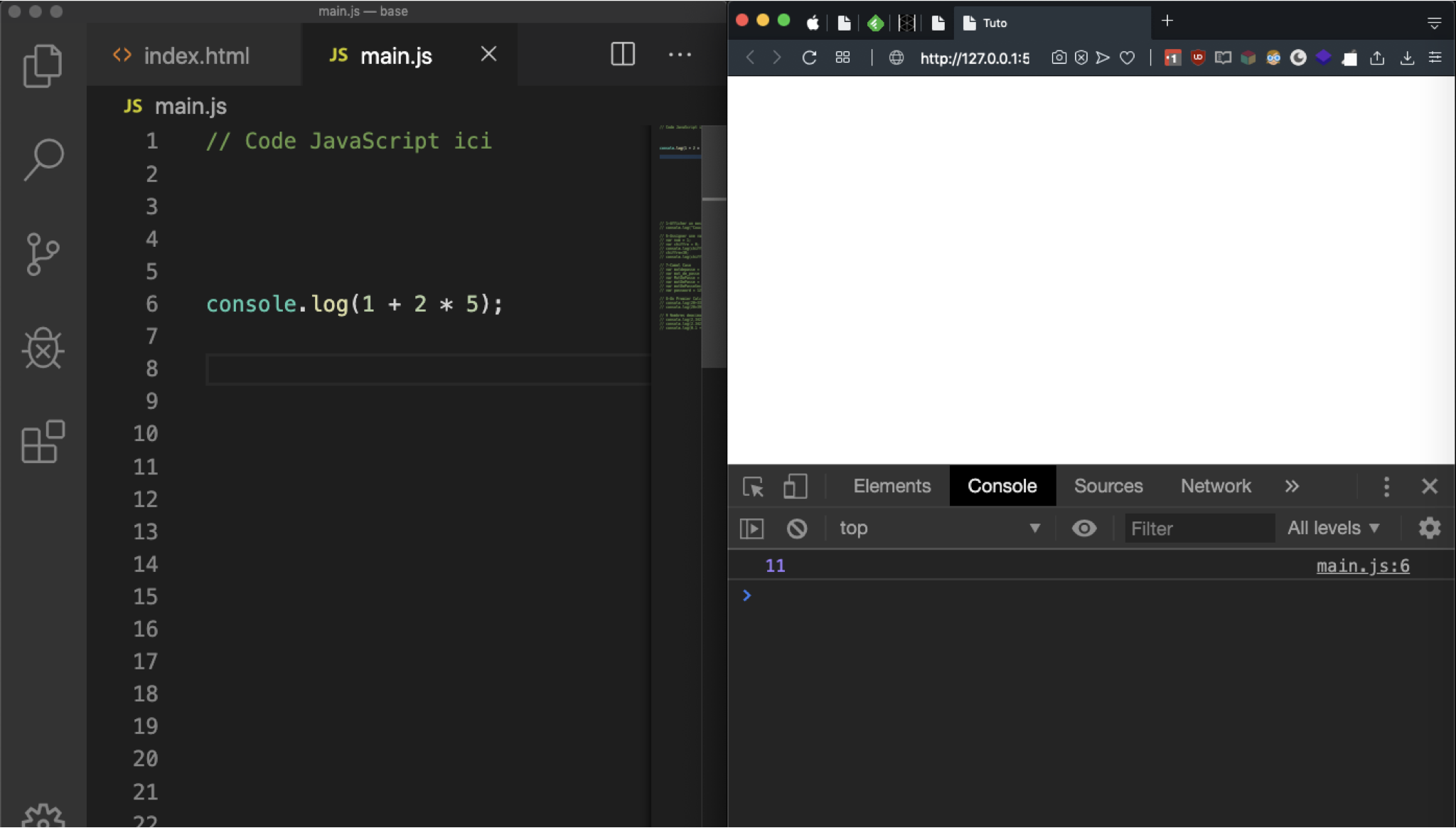
À la base quand JS a été créé les capacités mémoire des machines étaient inférieures comparé à aujourd’hui. Pour gérer les décimaux, les concepteurs du langage on fait le choix d’un certain algorithme de calcul (Pour résoudre les Pb de mémoire)

JS utilise une précision de virgule Flottante (Un système d’arrondi particulier)

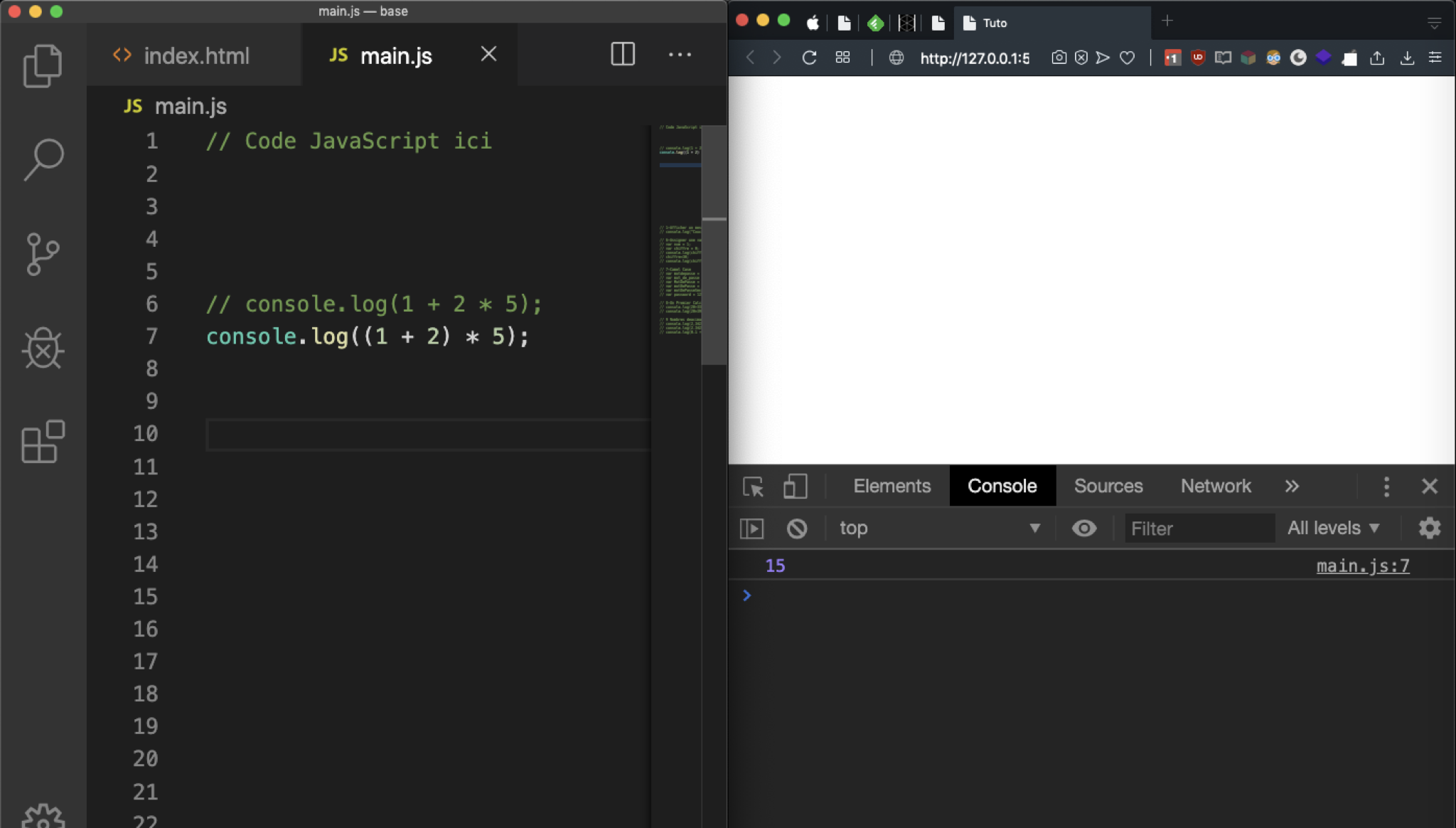
Pour faire du Jeux vidéo, des Apps Web ou Mobile, des animations cela ne pose pas de soucis.

Pas pertinent sauf si vous travaillez sur une app scientifique avec des données hyper précises.

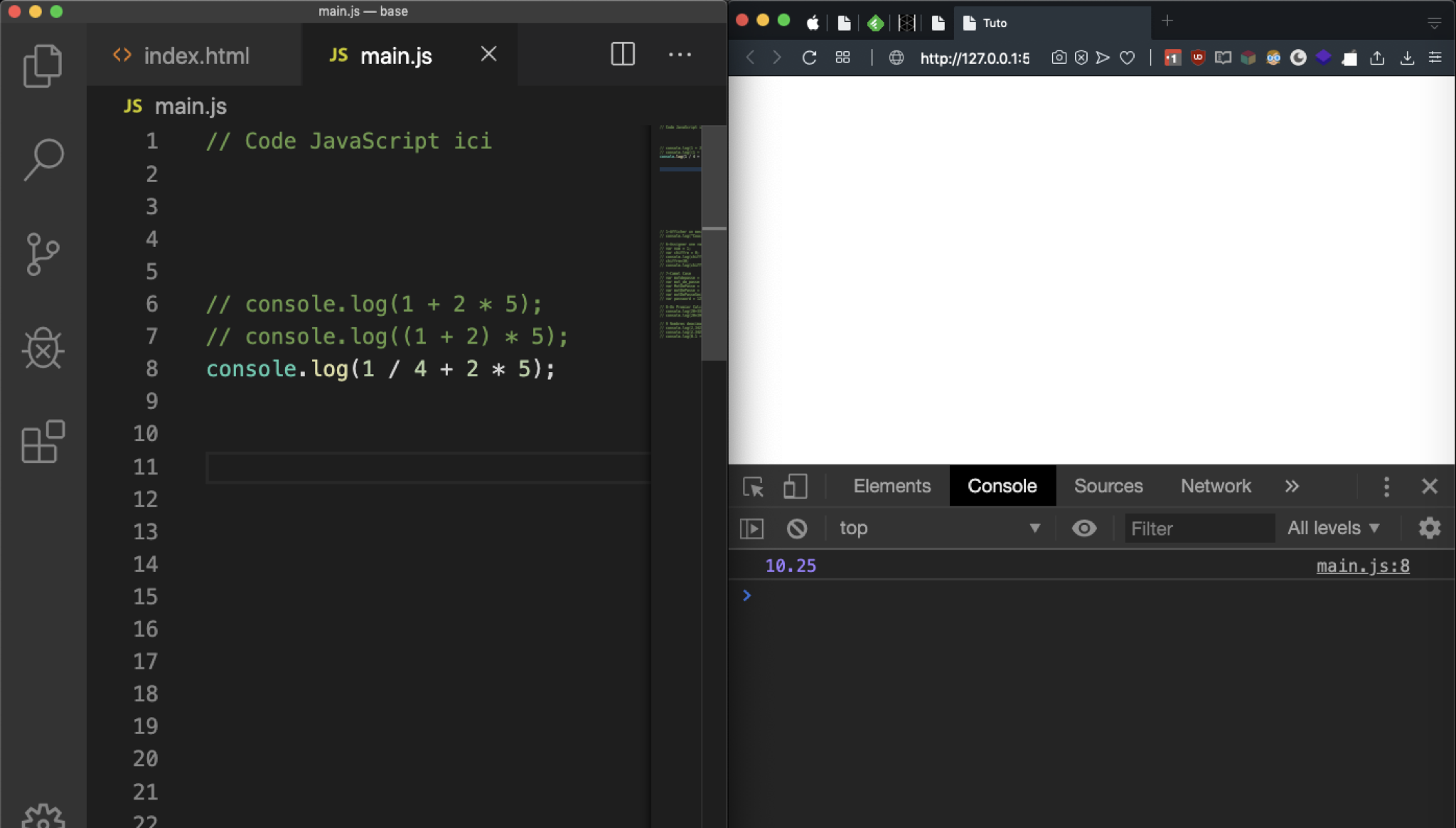
On peut bien entendu gérer les priorités de calculs (Comme en Maths)



Ordre du calcul : 2 \* 5, puis + 1



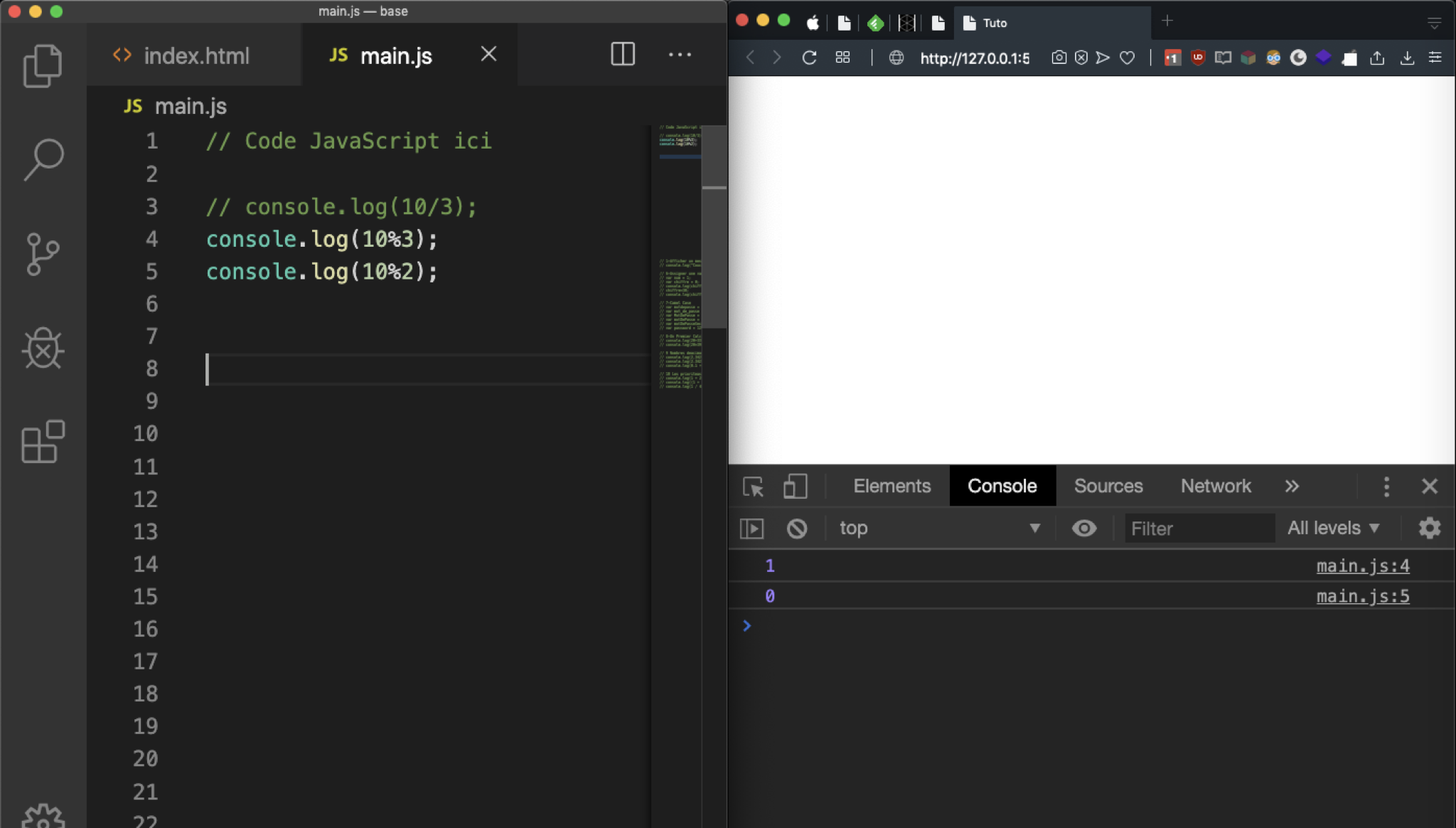
On peut modifier l’ordre de calcul avec des parenthèses ()



Ordre du calcul : ¼, puis multiplication 2 \*5, puis la somme

## L’opérateur modulo

Permet de calculer le reste d’une division.



Quand on veut calculer le reste d’une division, modulo 2 est utile pour savoir si un résultat est pair ou impair.

Si le calcul renvoie 0, c’est une division parfaite, on a un nombre pair

Si ≠ 0 : impair

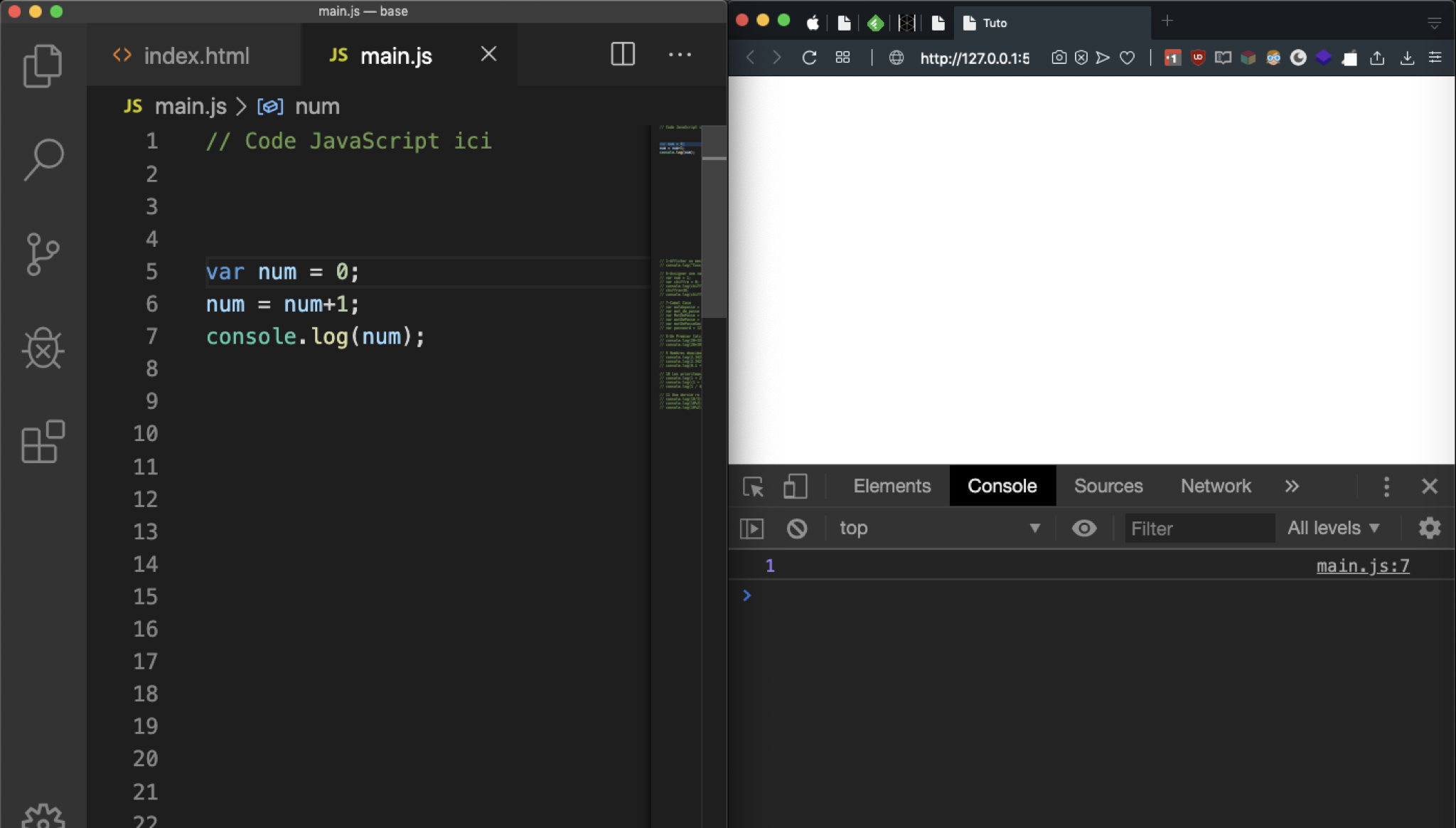
Autre exemple si on développe un jeu tour par tour (échec, jeux de carte) Le J1 pourra jouer quand le nombre de tour est pair et le J2 pour les tours impairs

## Incrémentation

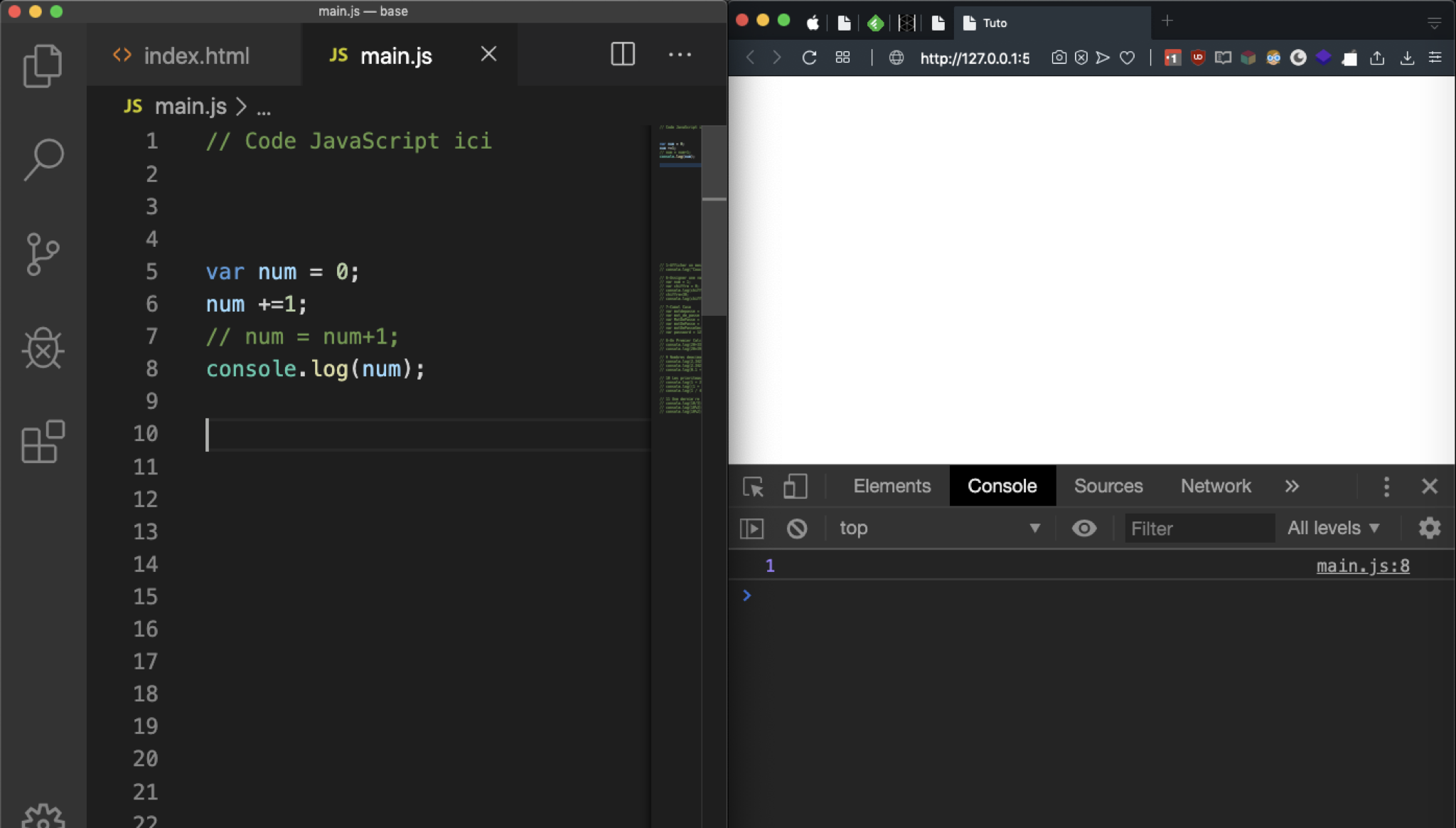
Dans le cadre d’une variable qui serait un nombre, l’incrémentation permet de faire +1 à cette variable.

(Idéal pour compter les nombres de visites sur une page, un nombre de commentaires ou de likes.)

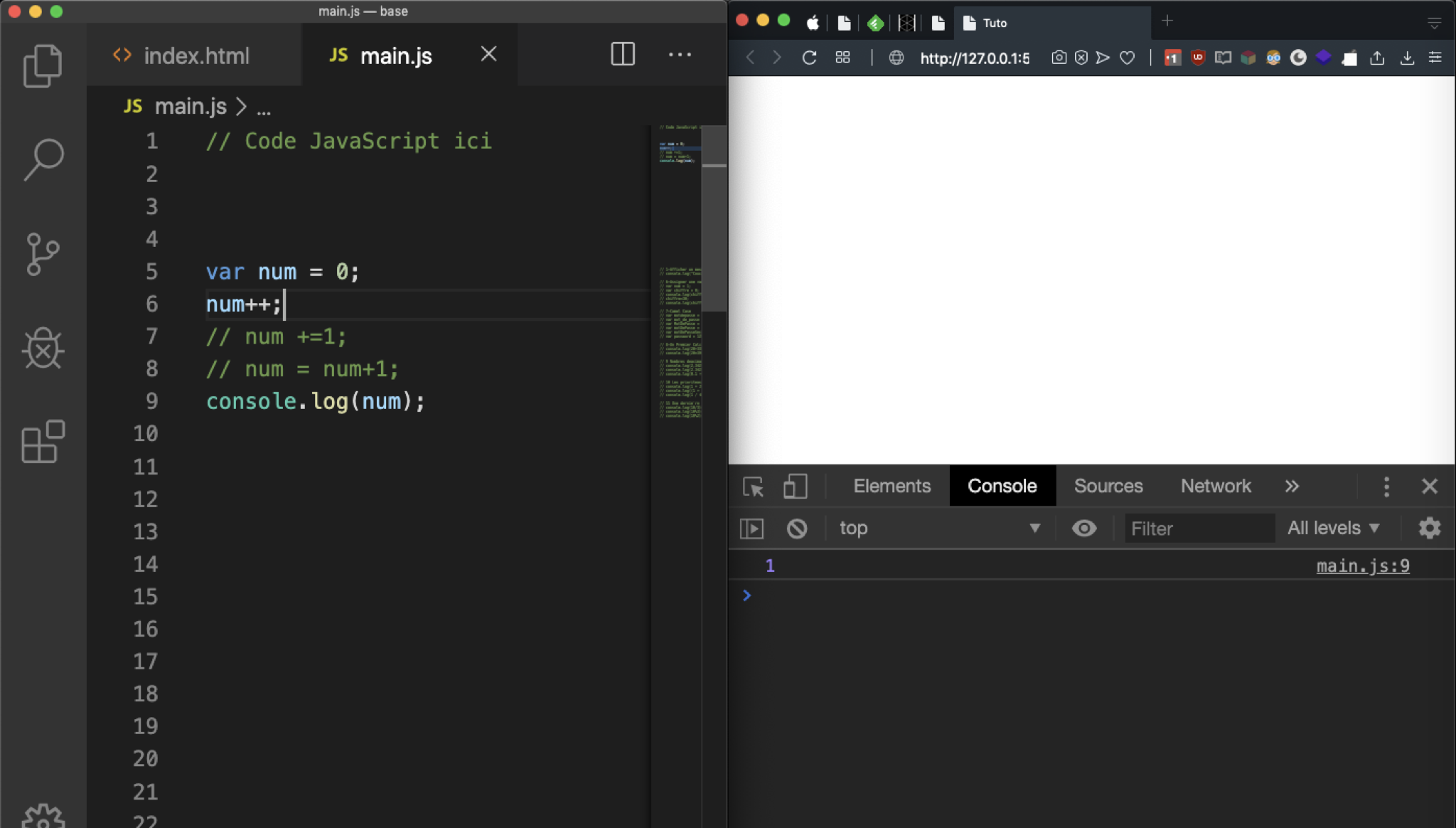
On peut directement écrire num+1



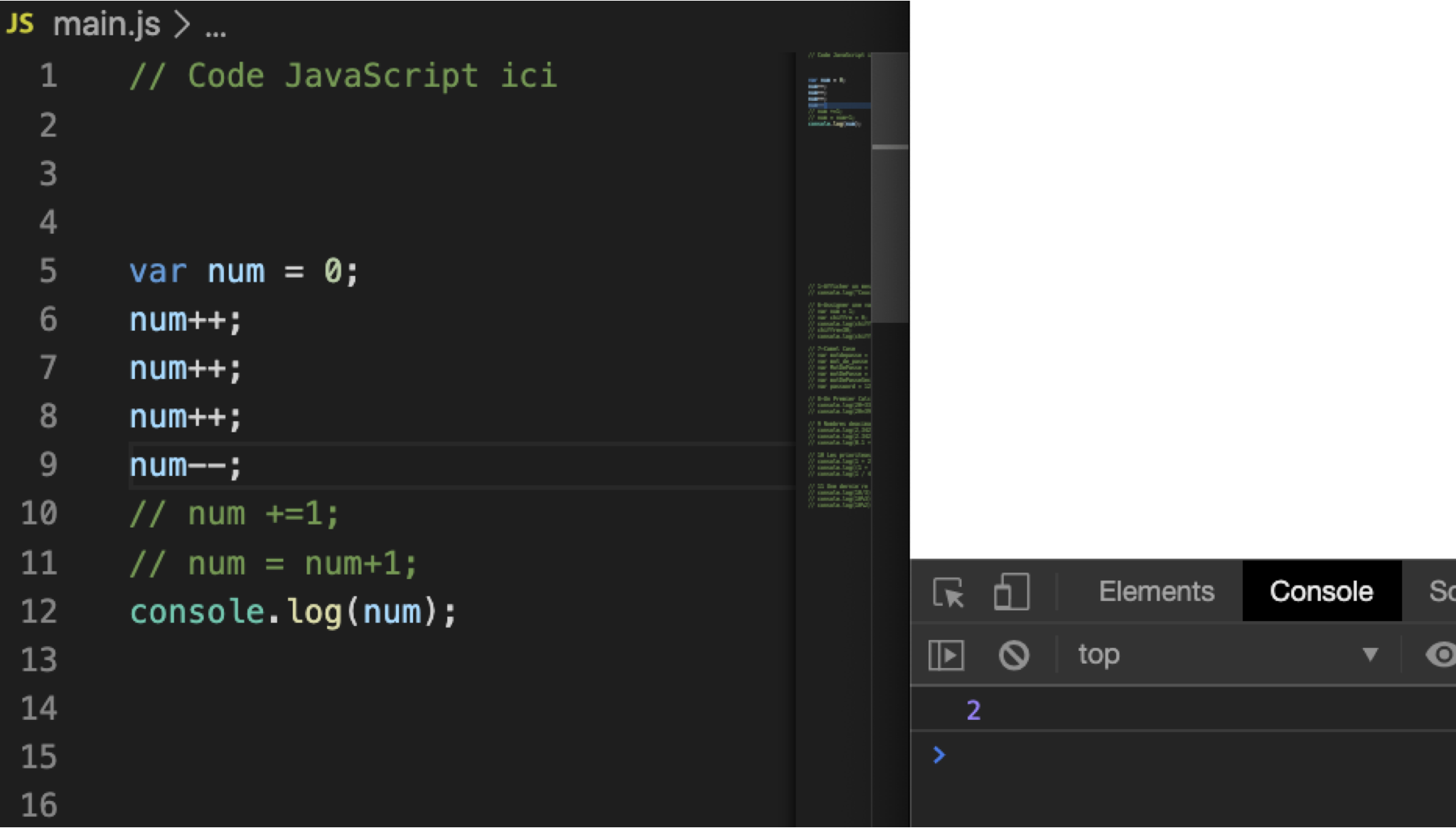
Ici pour créer un cumul, on assigne à la variable num, la variable num+1



Une autre notation pour créer un cumul : num+=1



Une autre notation pour directement faire +1 sur une variable : num++



On peut incrémenter et décrémenter.

Ici num est initialisée à 0 puis on fait 3 fois +1 et on fait -1 (L9)

## Affectation Composée

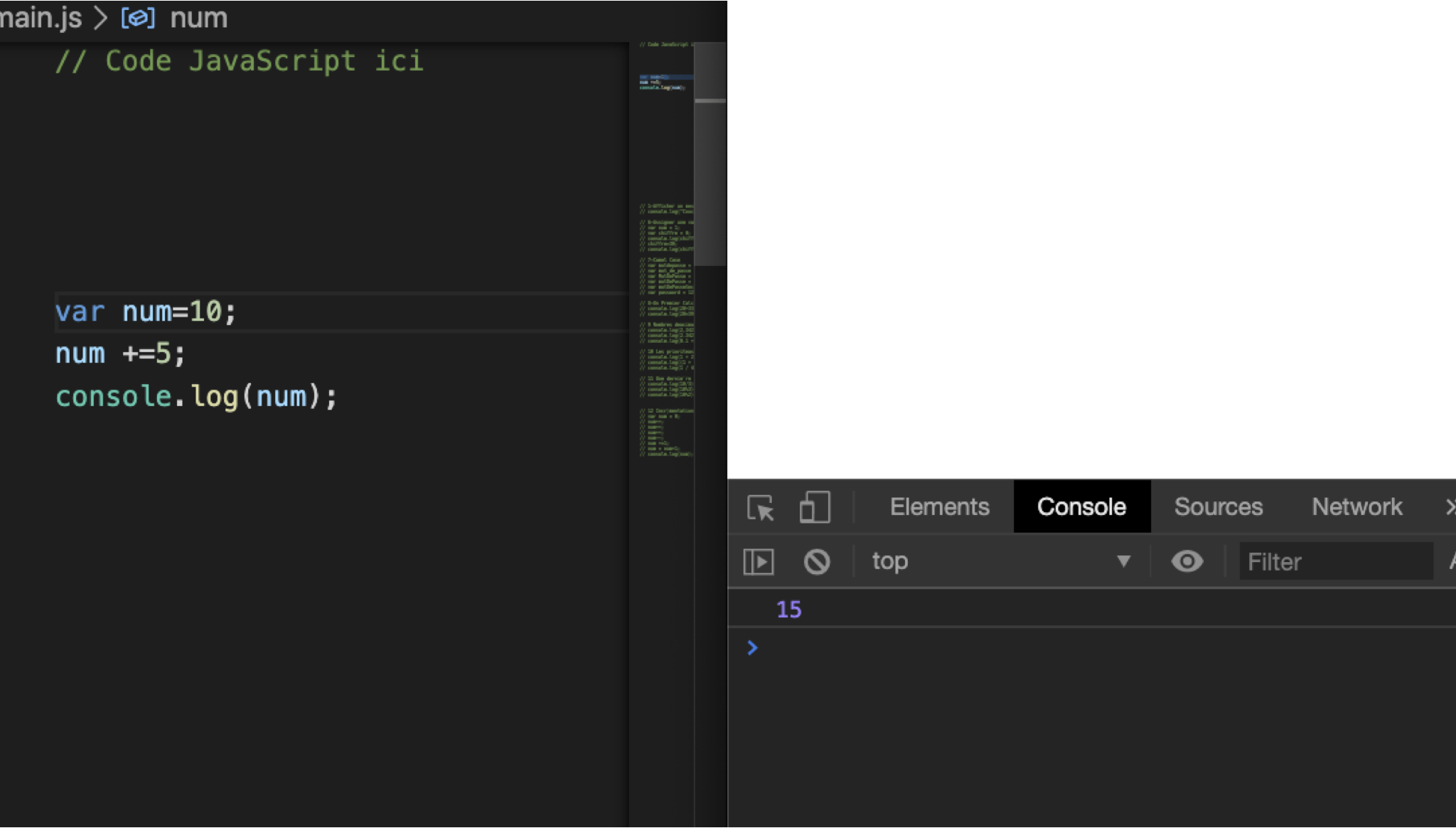
2 en 1

On donne une valeur et on fait une opération en même temps.

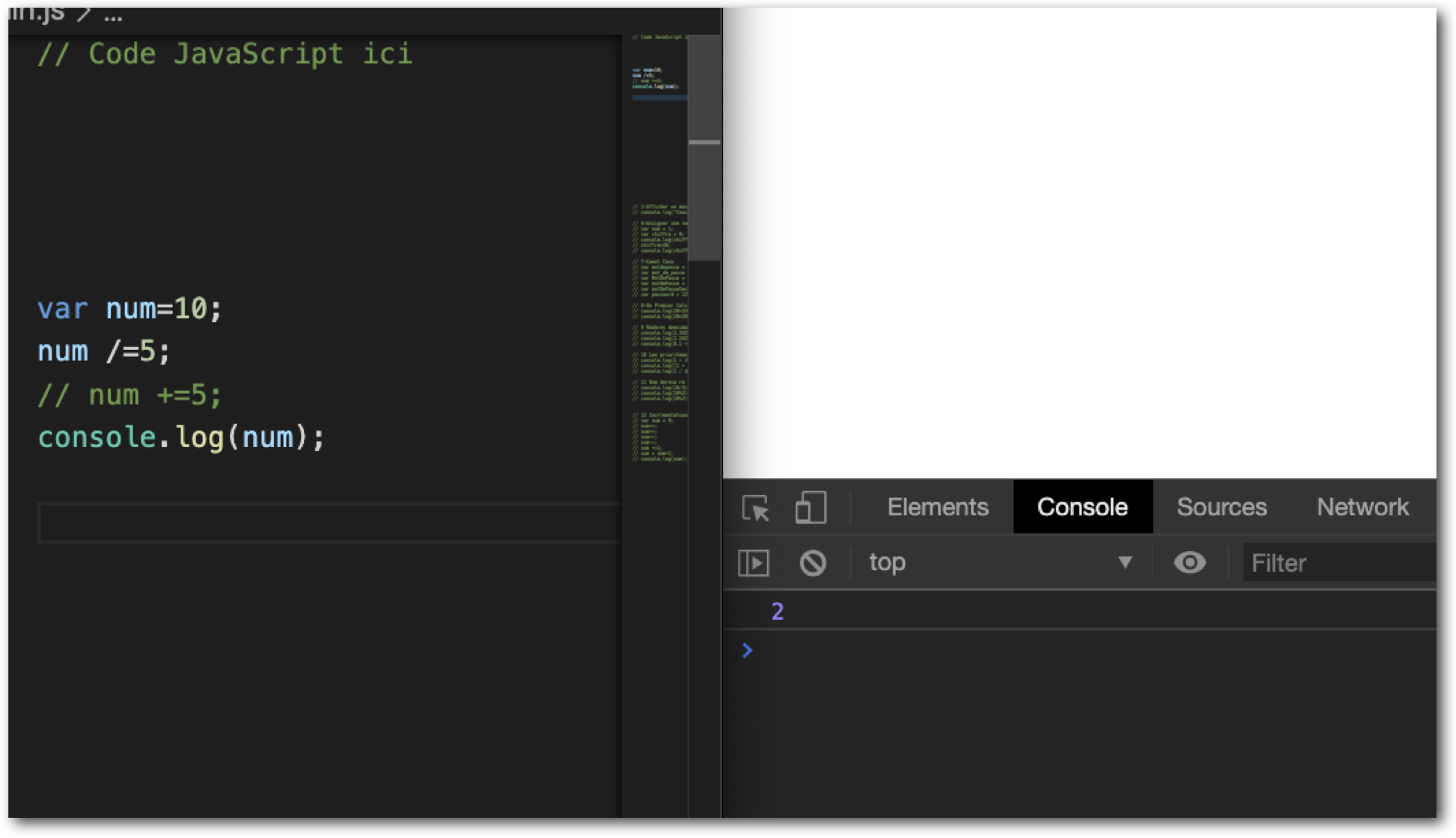
Concrètement cela peut permettre d’effectuer des calculs avec un code plus réduit, plus lisible.

On peut effectuer des calculs sans connaitre la valeur de départ.

Ex: Une remise de 20% sur tous les prix des produits d’une boutique.



La variable num est initialisée à 10, puis on fait 10+5



L’affectation composée fonctionne avec tous les opérateurs.