



Débogage

```
CONSOLE.LOG('TOTAL : ' + THIS.AMOUNT);
```

Les outils de débogage



Utiliser la console



Utiliser Angular dev Tools

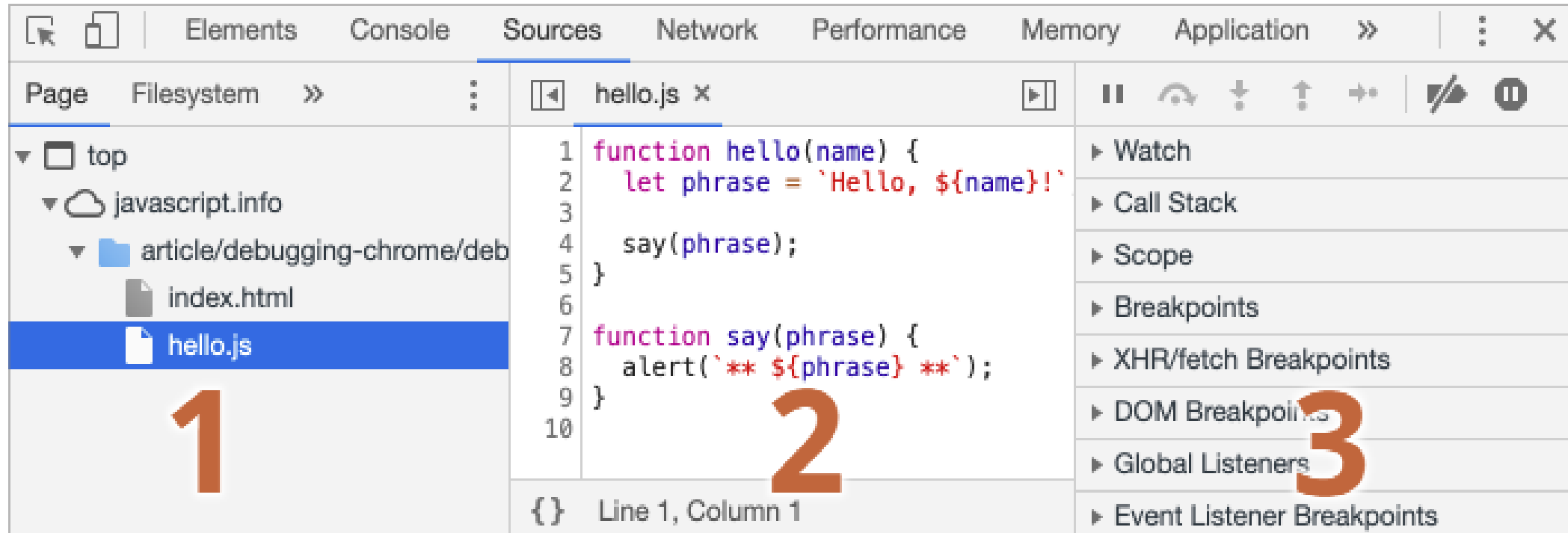
Débogage dans le navigateur

Le Debugging est le processus de recherche et de correction des erreurs dans un script.

Tous les navigateurs modernes et la plupart des autres environnements prennent en charge les outils de débogage.



Le volet “Sources”



Explorateur de fichiers

Editeur de code

Débugueur Javascript

NB : Activer les outils de développement avec F12

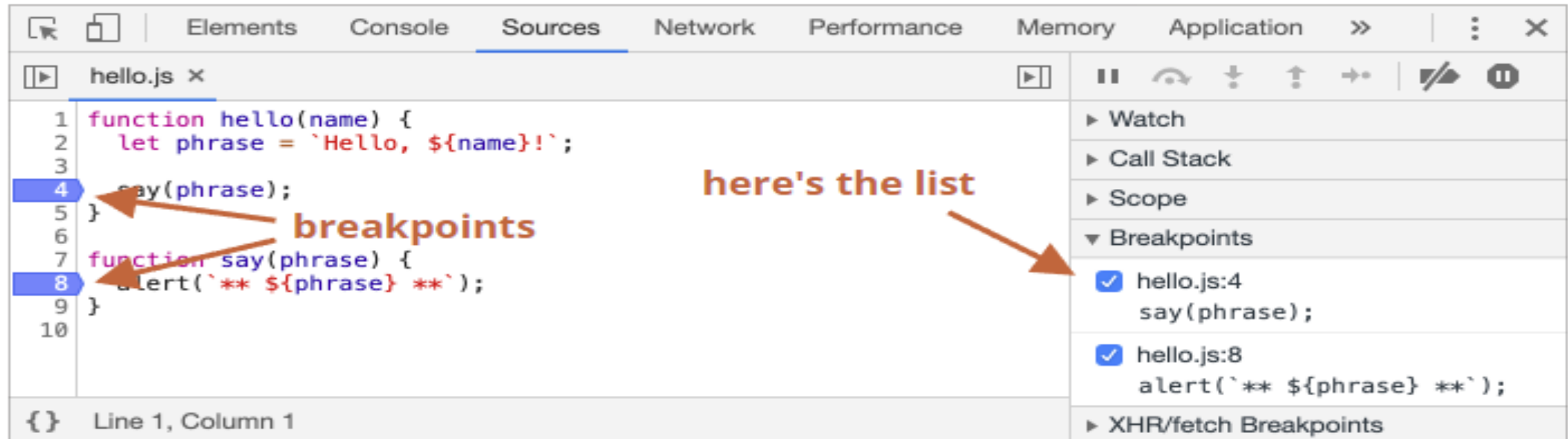
Console

The screenshot displays a web browser's developer console. The top toolbar includes icons for a bug, a cursor, and navigation arrows, followed by a dropdown menu set to 'Console'. To the right of the dropdown are tabs for 'HTML', 'CSS', 'Script', 'DOM', 'Net', and 'Cookies'. A search bar on the far right is labeled 'Search within Console panel'. Below the toolbar, a secondary bar contains a status icon (two vertical bars), and buttons for 'Clear', 'Persist', 'Profile', and 'All'. To the right of these are filters for 'Errors', 'Warnings', 'Info', 'Debug Info', and 'Cookies'. Further right are buttons for 'Run', 'Clear', 'Copy', 'Pretty Print', and 'History'. The main console area shows a list of messages: a log message '120', a debug message '130', a warning message '140' (highlighted in yellow), an info message '150' (highlighted in light blue), and an error message '160' (highlighted in light red). To the right of the messages, the corresponding JavaScript code is displayed, numbered 1 through 11. The code defines five player variables and then logs their IDs using console.log, console.debug, console.warn, console.info, and console.error.

```
> var playerOne = 120; var playerTwo = 130; var pla...nsole.error("Console.error" + " " +  
  playerFive);  
  
Console.log 120  
Console.debug 130  
! Console.warn 140  
i Console.info 150  
x Console.error 160
```

```
1 var playerOne = 120;  
2 var playerTwo = 130;  
3 var playerThree = 140;  
4 var playerFour = 150;  
5 var playerFive = 160;  
6  
7 console.log("Console.log" + " " + playerOne);  
8 console.debug("Console.debug" + " " + playerTwo);  
9 console.warn("Console.warn" + " " + playerThree);  
10 console.info("Console.info" + " " + playerFour);  
11 console.error("Console.error" + " " + playerFive);
```

BreakPoints



Un *breakpoint* est un point dans le code où le débogueur mettra automatiquement en pause l'exécution de JavaScript.

hello.js x

```
1 function hello(name) { name = "John"
2   let phrase = `Hello, ${name}!`; phrase = "Hello, John!"
3
4   say(phrase);
5 }
6
7 function say(phrase) {
8   alert(`** ${phrase} **`);
9 }
10
```

Paused on breakpoint

- 1 Watch: No watch expressions
- 2 Call Stack
 - hello hello.js:4
 - (anonymous) index.html:10
- 3 Scope
 - Local
 - name: "John"
 - phrase: "Hello, John!"
 - this: Window
 - Global Window

Line 4, Column 3

► Reprendre : continue l'exécution

➡ Step : lance la commande suivante

↻ Step over : lance la commande suivante, mais *n'entre pas dans une fonction*

⬆ Step out : continuer l'exécution jusqu'à la fin de la fonction en cours

hello.js x

```
1 function hello(name) {
2   let phrase = `Hello, ${name}!`;
3
4   say(phrase);
5 }
6
7 function say(phrase) { phrase = "Hello, John!"
8   alert(`** ${phrase} **`);
9 }
10
```

Paused on breakpoint

- Call Stack (nested calls)
 - say hello.js:8
 - hello hello.js:4
 - (anonymous) index.html:10
- Scope
 - Local
 - phrase: "Hello, John!"
 - this: Window
 - Global Window

Line 8, Column 3

La commande 'debugger'

Nous pouvons également suspendre le code en utilisant la commande `debugger`

```
1 function hello(name) {  
2   let phrase = `Hello, ${name}!`;  
3  
4   debugger; // <-- le débogueur s'arrête ici  
5  
6   say(phrase);  
7 }
```

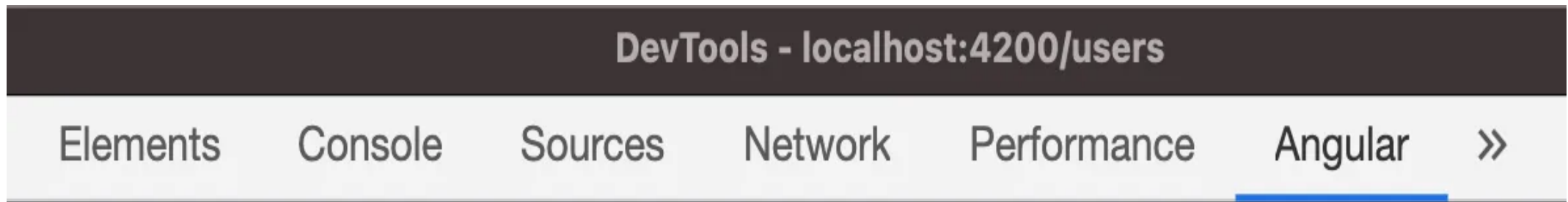
Une telle commande ne fonctionne que lorsque les outils de développement sont ouverts.

Angular Dev Tools

Une extension qui permet de visualiser la disposition d'une application angular dans une structure arborescente.

On peut cliquer sur chaque composant et voir ses propriétés, et voir exactement où se trouve ce composant sur la page.

De plus, il existe un profileur pour analyser les cycles des changements dans l'application.



ElementsConsoleSourcesNetworkPerformanceMemoryApplicationAngular>>

Angular version: 13.0.3 | DevTools SHA: be21eaa6

ComponentsProfiler

Search components

app-root

- a[RouterLinkWithHref]
- a[RouterLinkWithHref]
- router-outlet[RouterOutlet]
- app-trainings == \$ng0
 - input[NumberValueAccessor, DefaultValueAccessor, NgControlStatus, NgModel]
 - input[NumberValueAccessor, DefaultValueAccessor, NgControlStatus, NgModel]
 - input[NumberValueAccessor, DefaultValueAccessor, NgControlStatus, NgModel]

app-trainings

app-trainings

Properties

cartServices :

- customer :
 - address: ""
 - email: ""
 - firstName: ""
 - lastName: ""
 - phone: ""
- customerListTrainings :
 - > 0 : {...}
 - > 1 : {...}
 - length: 2

listTrainings :

- 0 :
 - description: Formation Java SE 8 5 jours
 - id: 1
 - name: Java
 - price: 1500
 - quantity: 1
- > 1 : {...}
- > 2 : {...}
- length: 3

router : {...}

Elements Console Sources Network Performance Memory Application Security Angular 1

Components Profiler Angular version: 13.0.3 | DevTools SHA: be21ea

Search components

app-root

- a[RouterLinkWithHref]
- a[RouterLinkWithHref]
- router-outlet[RouterOutlet]

app-customer == \$ng0

- form[ɵNgNoValidate, NgControlStatusGroup, NgForm]
 - input[DefaultValueAccessor, RequiredValidator, NgControlStatus, NgModel]
 - input[DefaultValueAccessor, NgControlStatus, NgModel]
 - input[DefaultValueAccessor, NgControlStatus, NgModel]
 - input[DefaultValueAccessor, NgControlStatus, NgModel]
 - input[DefaultValueAccessor, RequiredValidator, NgControlStatus, NgModel]

app-customer

Properties

cartService :

customer :

address: rue de la caaataemail: toto@yomail.frfirstName: ayyoublastName: mehdiouiphone: 0548784597

> customerListTrainings : Array(0)

Nom :

mehdioui

Prénom :

ayyoub

adresse :

rue de la caaata

Tel :

0548784597

Email :

toto@yomail.fr

Next

🔍 📄

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder 🦾 Performance insights 🦾 Angular >> 1 ⚙️

🖼️ ⚙️ ⓘ Components Profiler Angular version: 13.0.3 | DevTools SHA: be21ea

🔴 Interact to preview change detection. Clicking stop ends this Profiler recording.

🔍 📄

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder 🦾 Performance insights 🦾 Angular >> 1 ⚙️

🔄 Click Save Profile to save your recording or click refresh to clear the current recording.


Filter

Time spent: 1.2 ms
Source: `HTMLFormElement.addEventListener.submit`

Save Profile

Bar chart

21 / 21



< >

CustomerComponent

NgModel

CustomerComponent details

Total time spent: 0.2 ms

Directive	Method	Time
CustomerComponent	changes	0.2 ms
CustomerComponent	form-ngSubmit	0.3 ms
CustomerComponent	submit-HostBindingHandler	0.6 ms

Parent Hierarchy

- AppComponent

Ressources

<https://fr.javascript.info/debugging-chrome>

<https://medium.com/ngconf/introducing-the-angular-dev-tools-a71ea0ba65a7>