

Bootcamp Java Developer


Fase 2 - Java Web Developer
Módulo 17



Detección de errores

Detección de errores

Programar es estar en constante proceso de resolución de errores. De hecho, **la velocidad de la retroalimentación es una de las características propias de este trabajo**. Dicho de forma más simple: en este trabajo es posible tener una retroalimentación inmediata, por lo que puedes enterarte muy rápidamente (casi instantáneamente en determinados casos) si estás haciendo las cosas bien o mal.

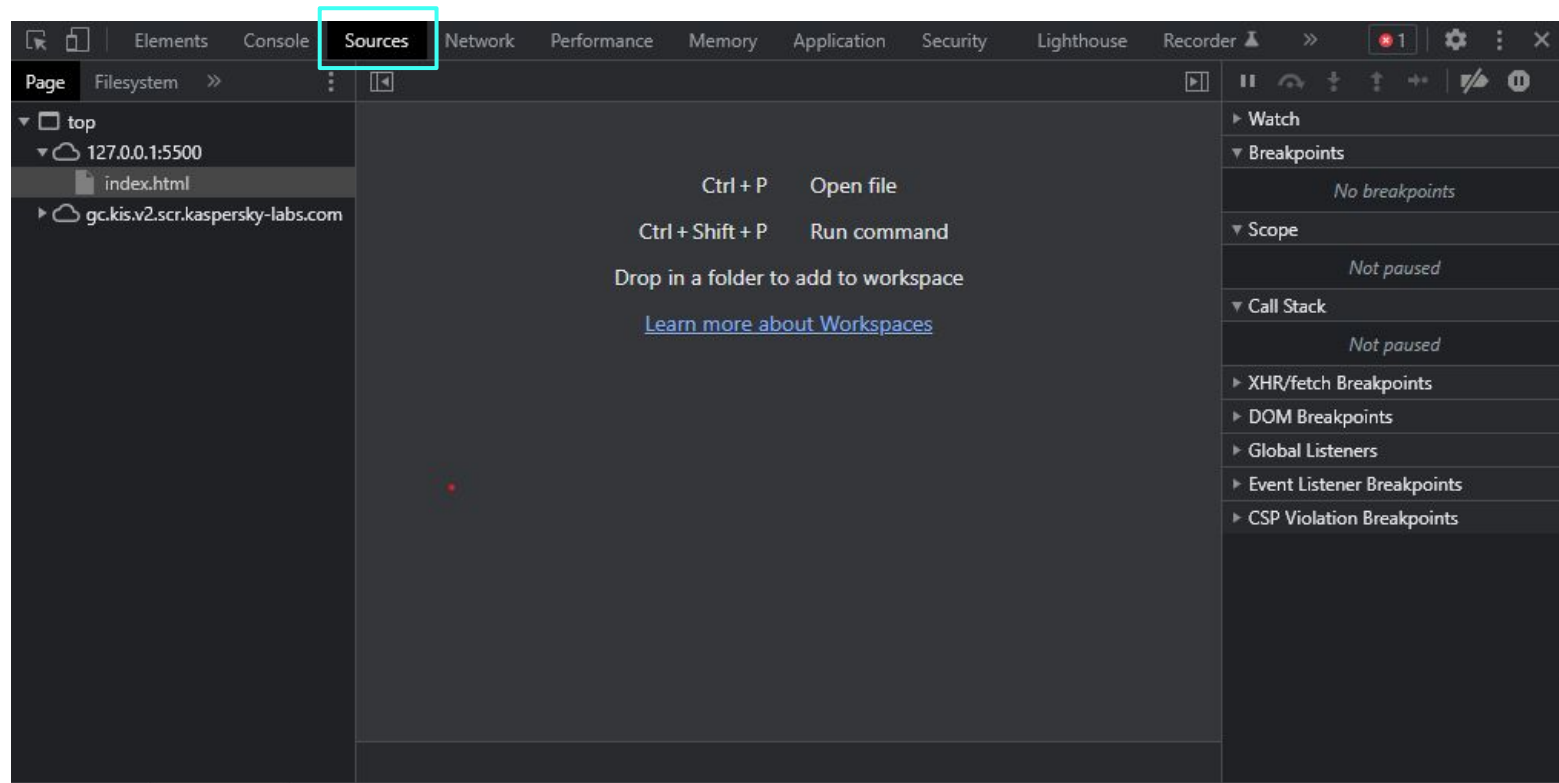


Si bien hay errores más difíciles de detectar y que requieren un juicio más pulido, **hay muchísimos errores que pueden ser detectados mientras estamos escribiendo código**.

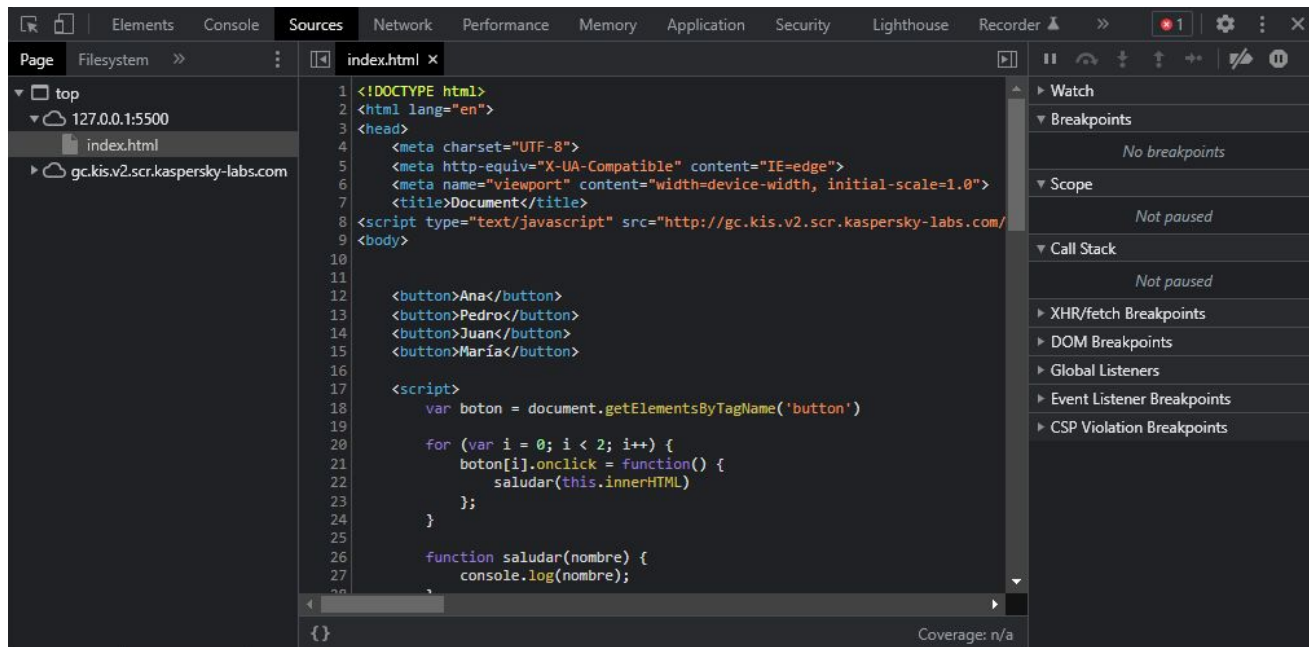
Para detectar esos errores **contamos con herramientas, como por ejemplo, el editor de texto**. Veamos otra herramienta que nos ayuda a detectar errores: el depurador de código

Depurador de código

1. Abre tu navegador favorito.
2. Haz click derecho en cualquier parte de la página.
3. Haz click en ***Inspeccionar***.
4. En el panel que se abra, haz click en **Fuentes** (se puede llamar “Código” o “Sources” también).
5. Ahí tendrás tres paneles:
 - a. A la **izquierda**, verás tus archivos.
 - b. Al **centro**, el archivo que estás evaluando.
 - c. A la **derecha**, las herramientas para evaluarlo.



6. Si haces click en un archivo, podrás ver su código.
Este es el **código fuente**.



The screenshot shows a web browser's developer tools interface. The 'Sources' panel is active, displaying the source code of 'index.html'. The file system on the left shows the file is located at 'gc.kis.v2.scr.kaspersky-labs.com'. The code is an HTML document with a JavaScript script that adds click events to buttons labeled 'Ana', 'Pedro', 'Juan', and 'Maria'. The script uses a loop to assign an onclick function to each button, which calls a 'saludar' function with the button's innerHTML. The 'saludar' function logs the name to the console. The right-hand sidebar shows various debugging panels like Watch, Breakpoints, Scope, Call Stack, and XHR/fetch Breakpoints, all of which are currently empty or show 'Not paused'.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-labs.com/
9 <body>
10
11
12   <button>Ana</button>
13   <button>Pedro</button>
14   <button>Juan</button>
15   <button>Maria</button>
16
17   <script>
18     var boton = document.getElementsByTagName('button')
19
20     for (var i = 0; i < 2; i++) {
21       boton[i].onclick = function() {
22         saludar(this.innerHTML)
23       };
24     }
25
26     function saludar(nombre) {
27       console.log(nombre);
28     }
29   </script>
30 </body>
31 </html>
```

Breakpoints

Si haces clic en un **número de línea** del archivo abierto, verás que cambia de color.

Acabas de crear un **punto de interrupción** (*Breakpoint*). El punto de interrupción indica que **el código se ejecutará hasta esa línea y luego quedará en pausa.**

Al quedarse en pausa, te otorga el control como desarrollador sobre la ejecución y podrás ir para adelante o atrás a voluntad.

Veámoslo en la pantalla a continuación.



The screenshot shows the Chrome DevTools Sources panel with the file `index.html` open. The file content is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-labs.com/
9 </body>
10
11
12   <button>Ana</button>
13   <button>Pedro</button>
14   <button>Juan</button>
15   <button>María</button>
16
17   <script>
18     var boton = document.getElementsByTagName('button')
19
20     for (var i = 0; i < 2; i++) {
21       boton[i].onclick = function() {
22         saludar(this.innerHTML)
23       };
24     }
25
26     function saludar(nombre) {
27       console.log(nombre);
28     }
29   </script>
30 </html>
```

A breakpoint is set on line 21, which is highlighted with a red box. The breakpoint is named `index.html:21` and has the condition `boton[i].onclick = function...`. The right sidebar shows the `Breakpoints` panel with the breakpoint listed. The `Scope` and `Call Stack` panels show `Not paused`. The `XHR/fetch Breakpoints`, `DOM Breakpoints`, `Global Listeners`, `Event Listener Breakpoints`, and `CSP Violation Breakpoints` panels are also visible.

Refresca la página (F5) con el breakpoint creado. Al hacerlo, verás que la página se queda estancada en ese punto.

Felicidades, has interrumpido la ejecución. Ahora puedes avanzar por el código según tu parecer.

Los breakpoints son útiles para examinar qué está haciendo el código en un momento dado.



The screenshot shows the Chrome DevTools interface with the Sources panel open. The file `index.html` is selected in the Filesystem view. The code editor displays the HTML and JavaScript of the file. A breakpoint is set on line 21, which is highlighted in green. The right sidebar shows the 'Paused on breakpoint' message and the 'Breakpoints' list, where the breakpoint on line 21 is checked. The 'Scope' panel shows the current scope as 'Global' and 'Window'. The 'Call Stack' panel shows the current call stack with the breakpoint location.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-labs.com/
9 </body>
10
11
12   <button>Ana</button>
13   <button>Pedro</button>
14   <button>Juan</button>
15   <button>María</button>
16
17   <script>
18     var boton = document.getElementsByTagName('button')
19
20     for (var i = 0; i < 2; i++) {
21       boton[i].onclick = function() {
22         saludar(this.innerHTML)
23       };
24     }
25
26     function saludar(nombre) {
27       console.log(nombre);
28     }
29   }
30 }
```

Line 21, Column 13

Coverage: n/a

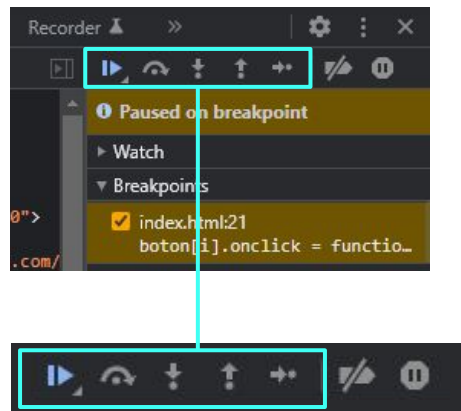
Control de ejecución

De izquierda a derecha, estos botones cumplen las siguientes funciones:

1. Continuar el código normalmente (F8).
2. Saltar la función actual*.
3. Entrar a la función actual*.
4. Salir de la función actual*.
5. Avanzar a la línea siguiente (F9).

Por ahora solo usaremos el 1 y el 5.

Con estas herramientas podemos ir paso a paso para examinar dónde está el error en el código.



* Nota: veremos funciones más adelante.

Inspección de variables

Es posible ver el valor actual de una variable posicionando el cursor encima de ella.

Al hacer eso, el navegador nos ofrecerá esa información.

Recuerda que **las variables son herramientas clave para el procesamiento de información.**

Veámoslo en la siguiente diapositiva.

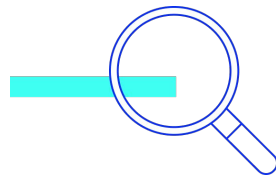


The screenshot shows a web browser's developer tools interface. The 'Sources' tab is active, displaying the code for 'index.html'. The file system on the left shows the file is located at 'gc.kis.v2.scr.kaspersky-labs.com'. The code in the center includes an HTML document with a head section containing meta tags for charset, http-equiv, and viewport, and a title 'Document'. The body contains five buttons with labels 'Ana', 'Pedro', 'Juan', and 'María'. A JavaScript script follows, which gets all buttons by their tag name and iterates over the first two. A breakpoint is set on line 21, which is highlighted in green. The breakpoint is located on the line: `boton[i].onclick = function() {`. The right-hand sidebar shows the 'Paused on breakpoint' panel, indicating the breakpoint is active at 'index.html:21'. Below this, the 'Scope' panel shows the current context as '(anonymous)' in 'index.html:21'. The status bar at the bottom indicates 'Line 20, Column 19' and 'Coverage: n/a'.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-labs.com/
9 <body>
10
11   <button>Ana</button>
12   <button>Pedro</button>
13   <button>Juan</button>
14   <button>María</button>
15
16
17   <script>
18     var botones = document.getElementsByTagName('button')
19
20     for (var i = 0; i < 2; i++) {
21       boton[i].onclick = function() {
22         saludar(this.innerHTML)
23       };
24     }
25
26     function saludar(nombre) {
27       console.log(nombre);
28     }
29   }
30 }
```

Revisión

- Aprendimos sobre el **uso del depurador**.
- Creamos nuestros propios **puntos de interrupción**.
- Utilizamos los controles para avanzar o continuar el **flujo de ejecución**.
- Utilizamos la **inspección de variables** para revisar el estado actual del código.



**¡Sigamos
trabajando!**

