Depurando JS con DevTools

UD1: Tecnologías para el desarrollo de interfaces

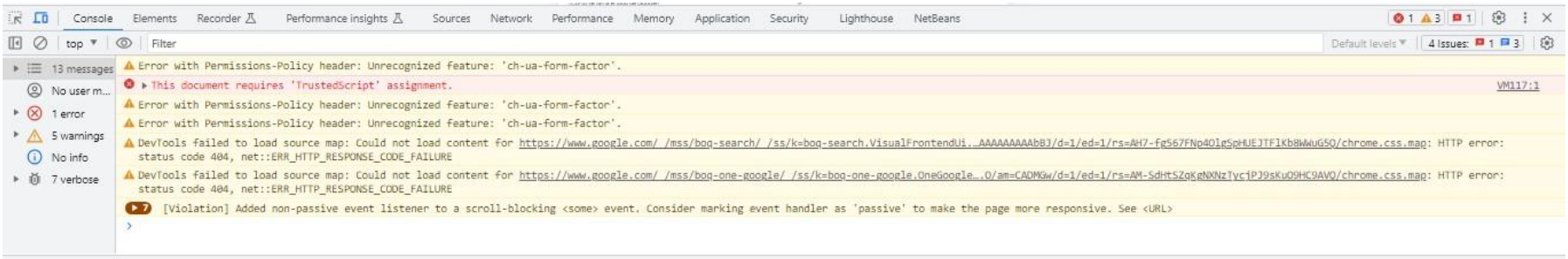
Objetivos de aprendizaje

- Aprender a usar las principales funcionalidades de la consola de Chrome para depurar JavaScript

DevTools

Shift+Ctrl+J o **F12**

- Herramientas de desarrollo de Chrome
- Facilitan la depuración y control de errores de las tecnologías de lado cliente



La consola

- En el menú superior “Console” (atajo con **esc**)

- **Muy útil para probar “sobre la marcha” el resultado de una expresión**

Opciones útiles de la consola

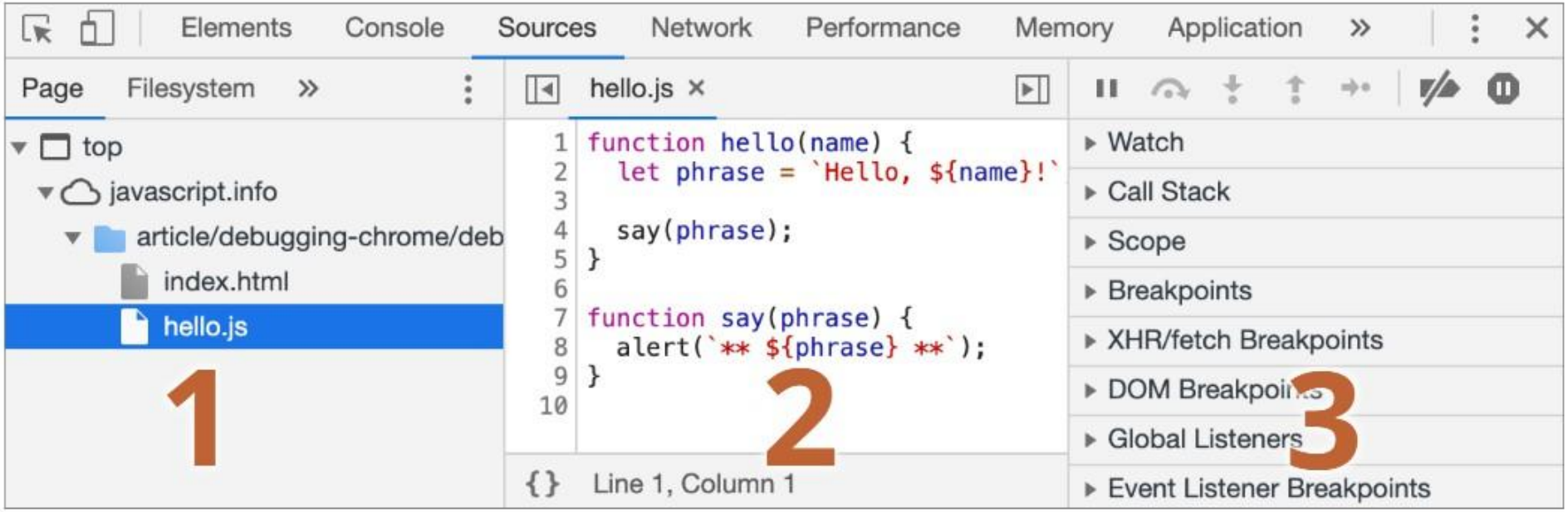
Función	Utilidad	Ejemplo
`console.log( )`	Mostrar mensajes de información	`console.log('Hola');` `console.log(variable);`

`console.warn( )`
`console.error( )`
`console.debug( )`
	Muestra mensajes de advertencia / error / debugging	`console.warn('Cuidado!');`
		`console.warn('Error!');`
		`console.deb('Registro guardado');`

`console.table( )`	Muestra información de manera tabular Se usa para tablas o para objetos	`console.table(["apples", "oranges", "bananas"]);`

Panel “Sources”

- Partes:
  - 1. Zona de recursos
  - 2. Código fuente de los archivos
  - 3. Zona con opciones de debugging



Breakpoints

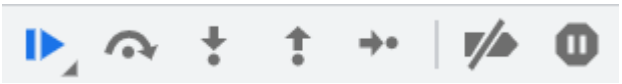
- Son puntos donde queremos que nuestro código se detenga en tiempo de ejecución
- Se insertan haciendo clic sobre el número de línea:

```
7
8          resultado += num + " * " + i + " = " +
9                       num* i + "\n";
10     }
```

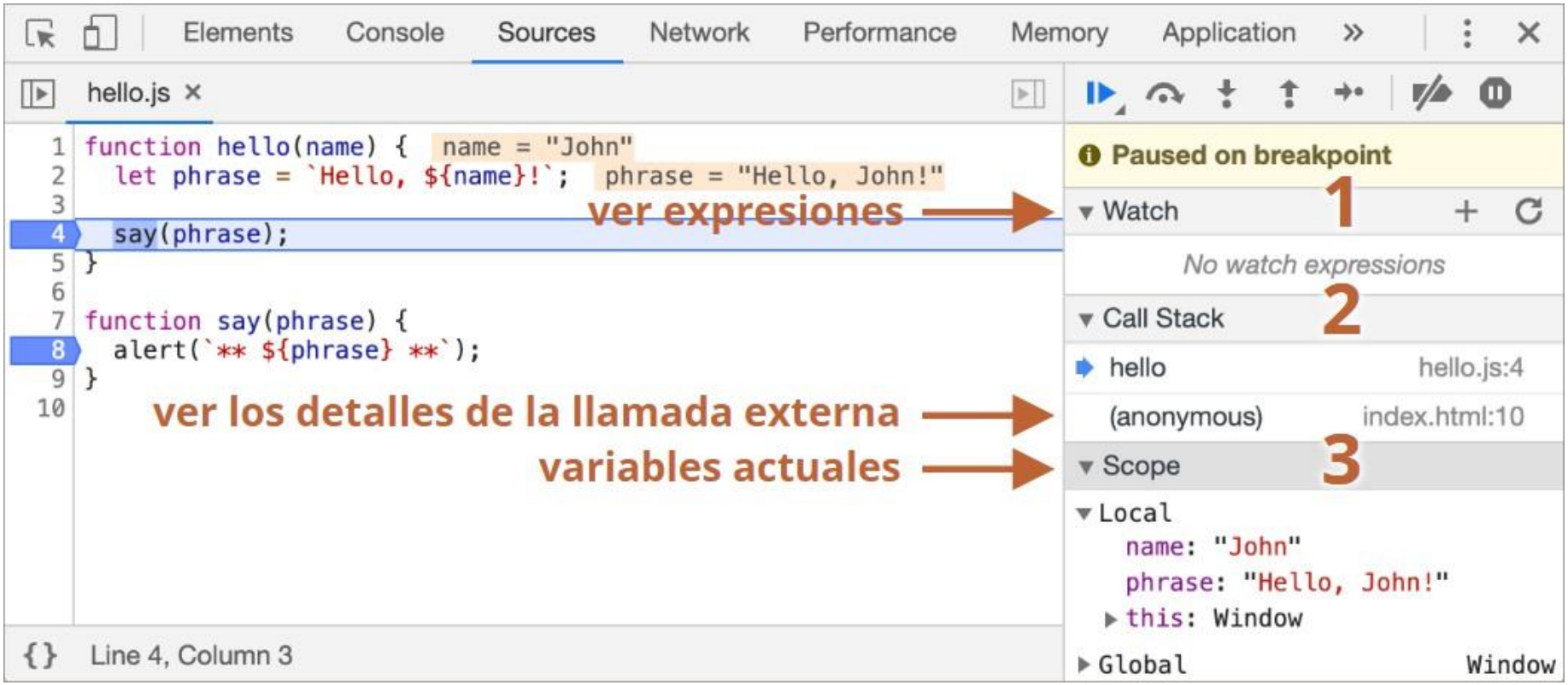
  - También se puede usar la instrucción **debugger;** en el código para forzar dicha detención
- Los siguientes botones permiten ir paso a paso decidiendo si entrar o no en el código interno de cada función:

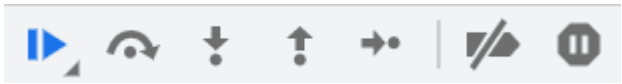
Opciones de ejecución

- Reanudar, continua siguiente instrucción (si no hay más breakpoints, termina la ejecución): F8
- Siguiente paso (ejecuta la siguiente sentencia): F9
- Saltar paso (siguiente sentencia, sin entrar en funciones): F10
- Continuar la ejecución hasta el final de la función actual: Shift + F11



Si hay un error lo vas a encontrar





¿Cómo?

1. La sección ***Watch*** permite mirar el valor y tipo de datos que van tomando las variables en tiempo de ejecución
   - El valor de las variables también se puede ver posando el ratón sobre la misma en el código
2. La sección ***Call Stack*** muestra las llamadas anidadas
3. En ***Scope*** están las variables activas, locales y globales

Ejemplo 1

- Realiza un script que pida un número entero por pantalla y a continuación genere su tabla de multiplicar del 1 al 10 (es decir, el resultado de multiplicar el número indicado sucesivamente por los números del 1 al 10
- El resultado se presentará en 1 mensaje por consola
- Para resolver el ejercicio crea el menos la siguiente función:
  - `tablaMultiplicar(entero):cadena`

Ejemplo 1 - BIS

- Introduce un error en el código, por ejemplo cambiando el operador de acumular por el de asignar:

```
resultado += num + " * " + i + " = " +  num* i + "\n";

resultado =  m + " * " + i + " = " +  num* i + "\n";
```

- Inserta un ***breakpoint*** en esa línea y traza el código, comprobando en todo momento que valores van tomando las variables `resultado, num e i`

Snippets

- En la consola, sección ***Source***, parte izquierda, existe la posibilidad de crear ***Snippets*** de código
  - Son fragmentos de código con una funcionalidad concreta
- Se pueden ejecutar con **Run** y llamarlos desde la consola para probar su funcionalidad

Tabla de multiplicar..

```
function tablaMultiplicar(num){
    let resultado = "\n";

    for(let i=1; i<=10; i++){
        resultado += num + " * " + i + " = " +
                     num* i + "\n";
    }
    return resultado;
}
```

Ejemplo de snippet

- Crea un **Snippet** "Tabla de multiplicar" con el código para generar la tabla de multiplicar de un número que se pasa como parámetro.
- Ejecútalo y comprueba que funciona generando la tabla del 4

```
console.log(tablaMultiplicar(4))
```

```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

Ejemplos 3 y 4

- Pon en práctica todo lo aprendido hoy para depurar los dos scripts `ejemplo3` y `ejemplo4` que contienen errores de diversa índole