Format and style messages in the Console

Jecelyn Yeen

This guide shows you how to format and style messages in the Chrome DevTools Console. See Get Started With Logging Messages to learn how to log messages to the Console.

This guide assumes that you understand the fundamentals of web development, such as how to use JavaScript to add interactivity to a page.

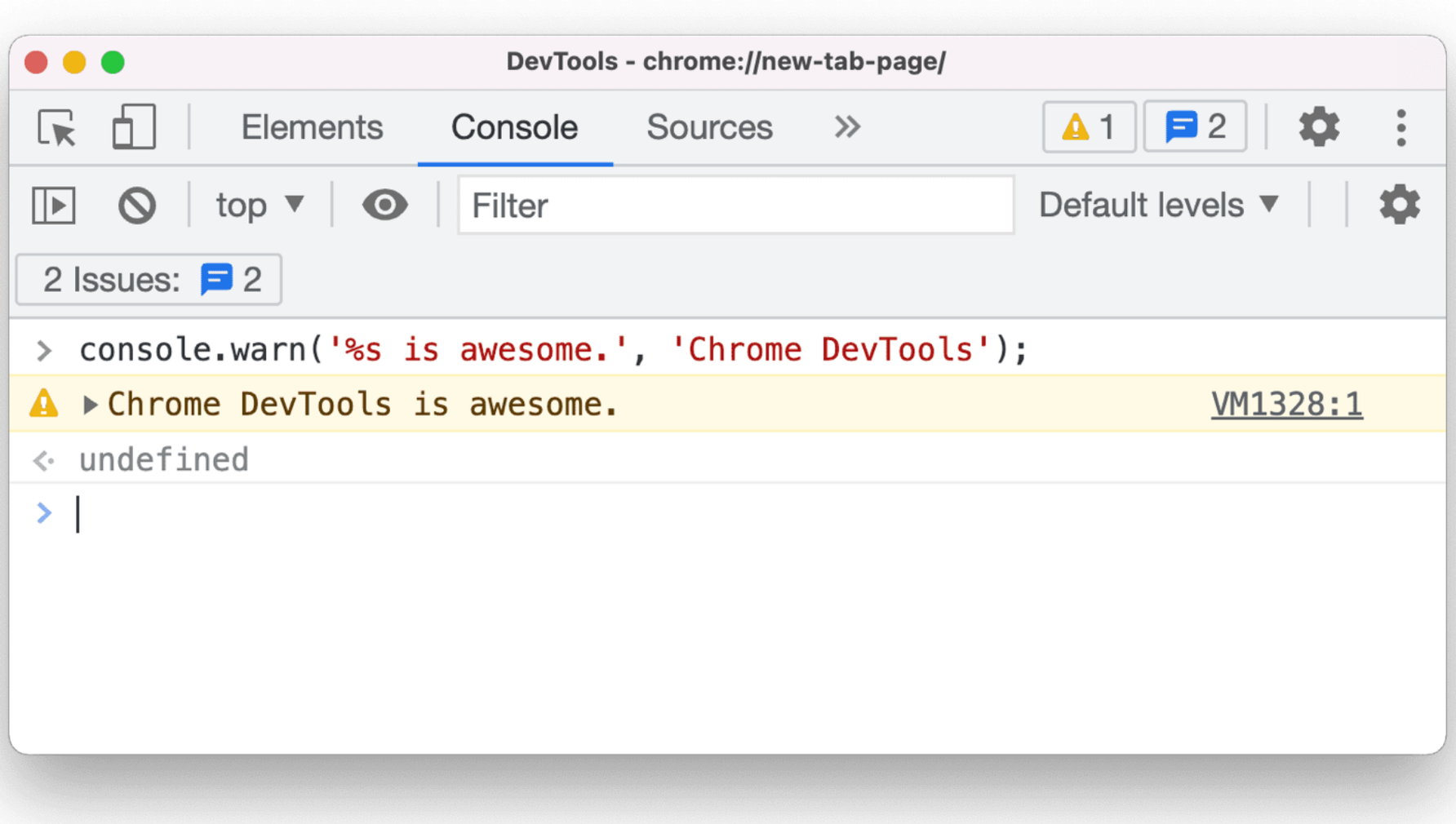
Format console messages

You can use the format specifiers to format the console messages.

Format specifiers begin with a percent character (%) and terminate with a "type character" which indicates the type of data (integer, float, etc.).

For example,

1. Open the Console
2. Enter the following console command. `js const tools = 'Chrome DevTools'; console.warn('%s is awesome.', tools);`
3. The command above produces `Chrome DevTools is awesome.` message.



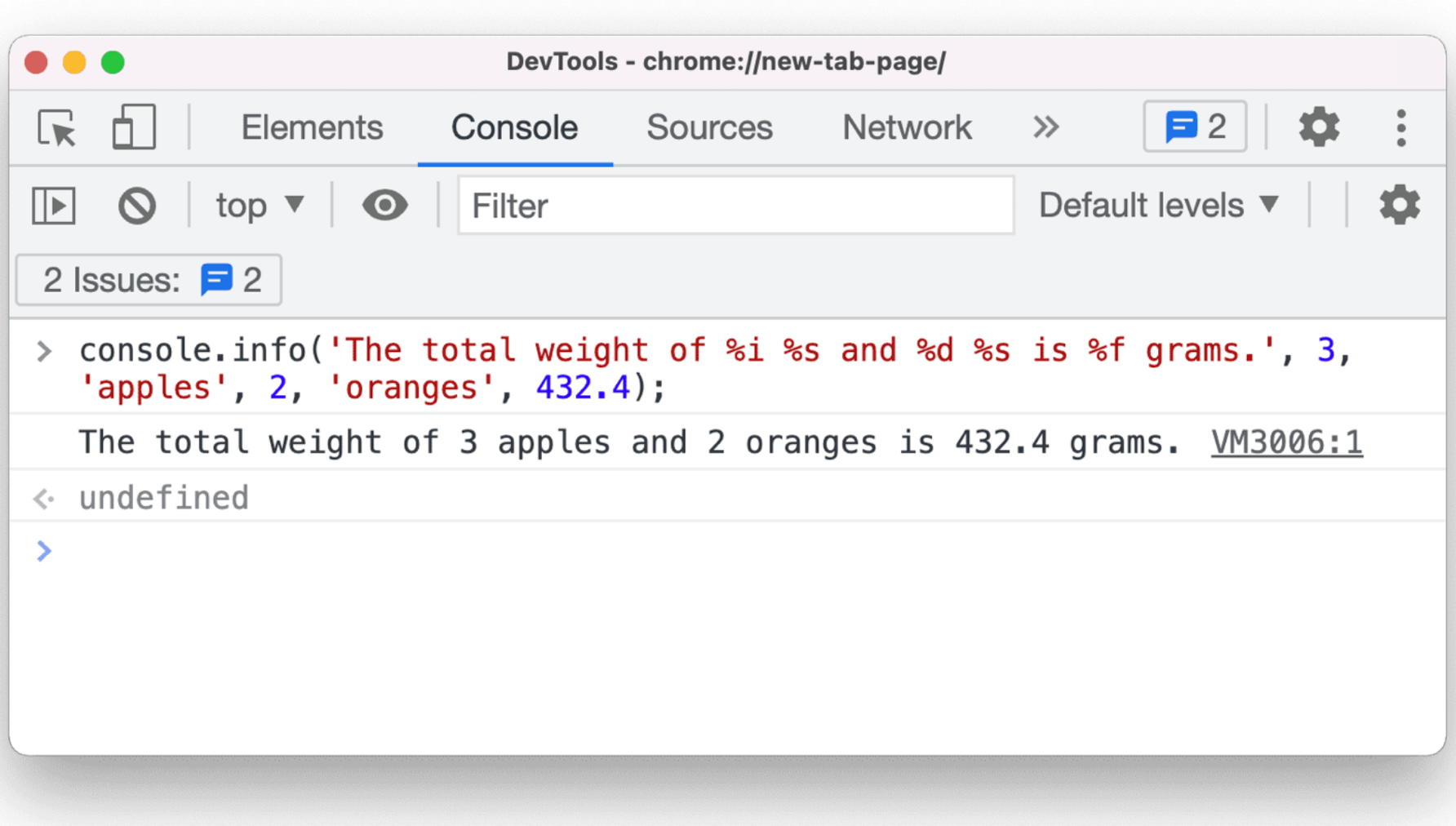
Here is the list of format specifiers Chrome DevTools support currently.

Specifier	Output
`%s`	Formats the value as a string
`%i` or `%d`	Formats the value as an integer
`%f`	Formats the value as a floating point value
`%o`	Formats the value as an expandable DOM element
`%O`	Formats the value as an expandable JavaScript object
`%c`	Applies CSS style rules to the output string as specified by the second parameter

Apply multiple format specifiers

You can use more than one format specifier in a message.

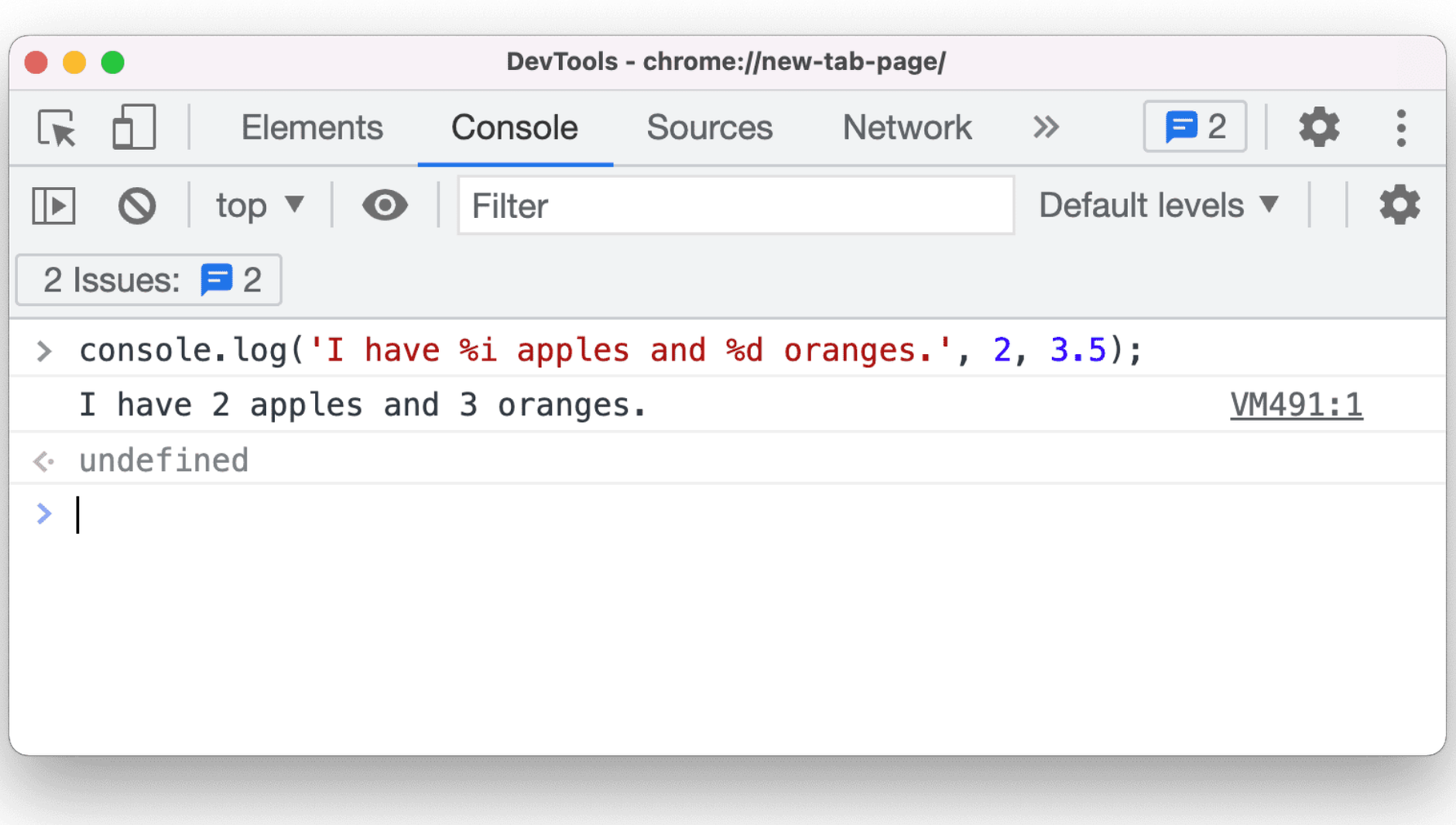
1. Enter the following console command. `js console.info('The total weight of %i %s and %d %s is %f grams.', 3, 'apples', 2, 'oranges', 432.4);`
2. The command above produces `The total weight of 3 apples and 2 oranges is 432.4 grams.` message.



Understand type conversions

The output message will be converted according to the format specifier.

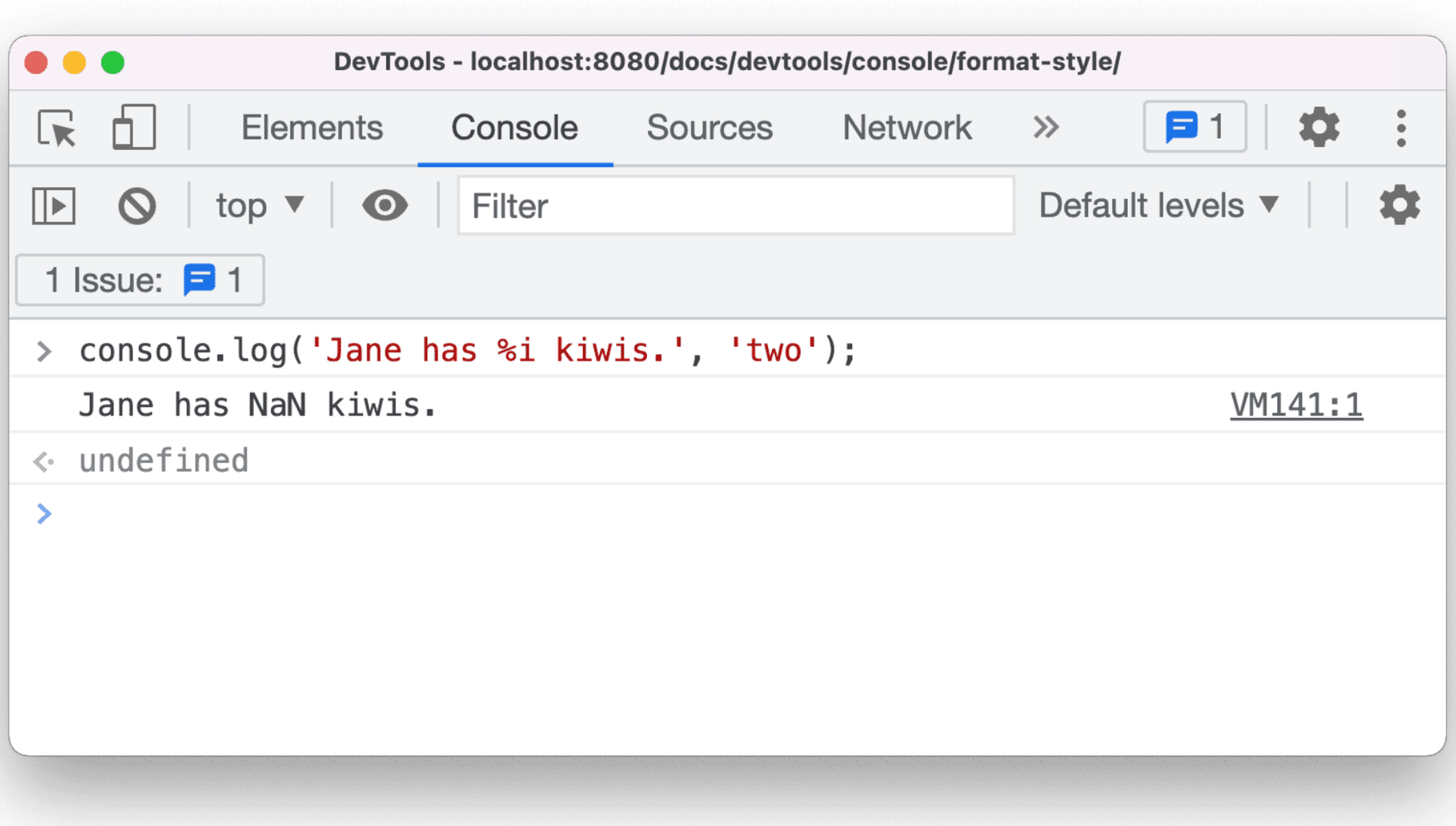
1. Enter the following console command. `js console.log('I have %i apples and %d oranges.', 2, 3.5);`
2. The command above produces `I have 2 apples and 3 oranges.` message.



3. Instead of logging `3.5 oranges`, the output is `3 oranges`. The `%d` indicates that the value should/will be converted to an integer.

Here is an example of what happens if the type conversion is invalid.

1. Enter the following console command. `js console.log('Jane has %i kiwis.', 'two');`
2. The command above produces `Jane has NaN kiwis.` message.



3. The `%i` indicates that the value should/will be converted to an integer, but the argument is a string. Thus it returns NaN (Not-A-Number).

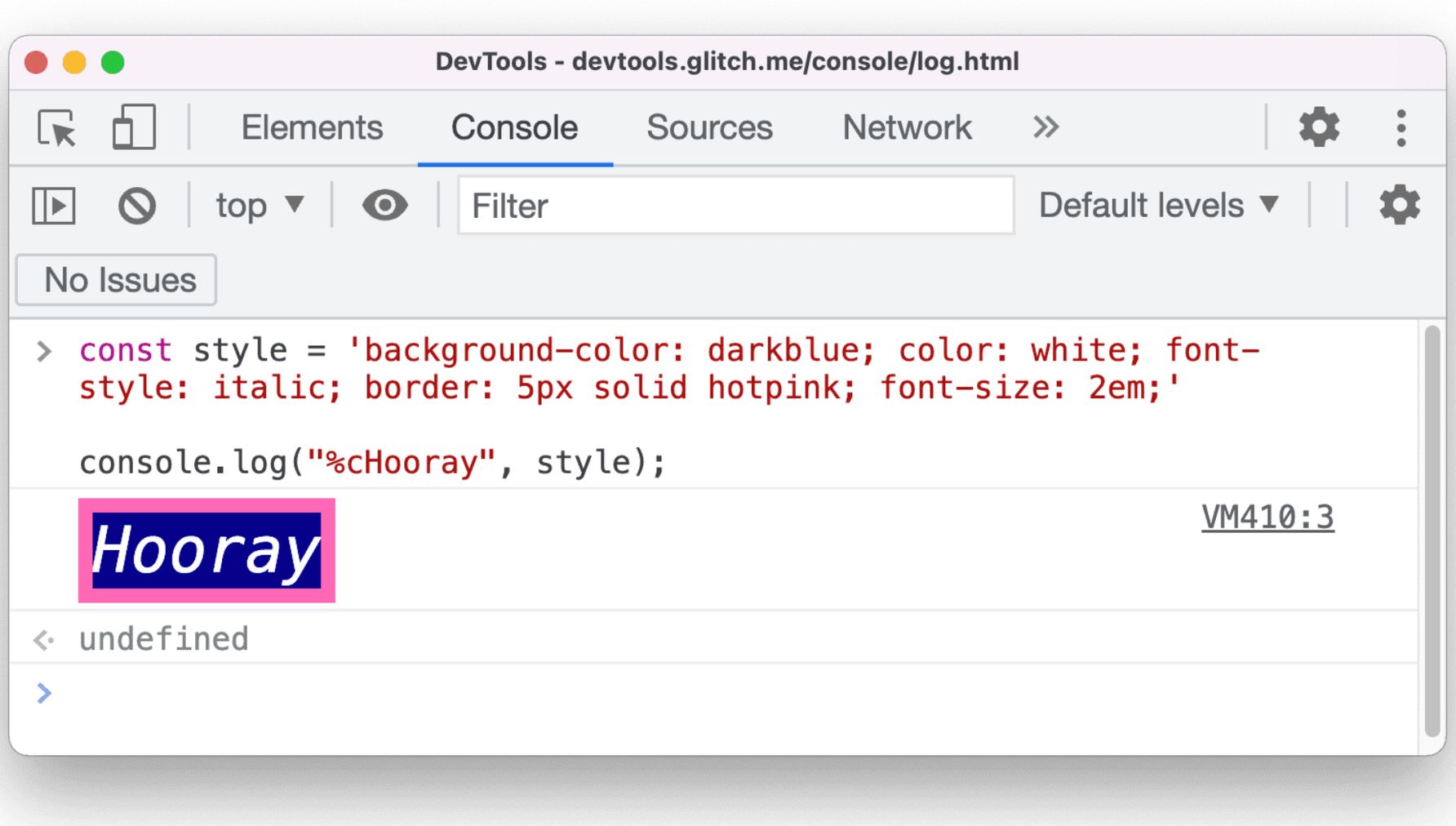
Style console messages

There are two ways to style console messages in DevTools.

Style with format specifier

You can use the `%c` format specifier to style the console messages with CSS.

1. Enter the following console command. `js const style = 'background-color: darkblue; color: white; font-style: italic; border: 5px solid hotpink; font-size: 2em;' console.log("%cHooray", style);`
2. The command above produces `Hooray` with CSS styles applied.



Key point: To prevent data leaks and bypasses of security policies, in this format, the `url()` CSS function supports only the `data:` URL schema. For example, you can set a background image in the following way: `css background: url(…);` Where `iVBORw…` is a base64-encoded PNG image.

Style with ANSI escape sequences

You can use the ANSI escape sequences to style console messages.

It is common for Node.js developers to colorize log messages via ANSI escape sequences, often with the help of some styling libraries like chalk, colors, ansi-colors, kleur.

Nevertheless, you can style the message with ANSI escape sequences without using any libraries. Here is the syntax:

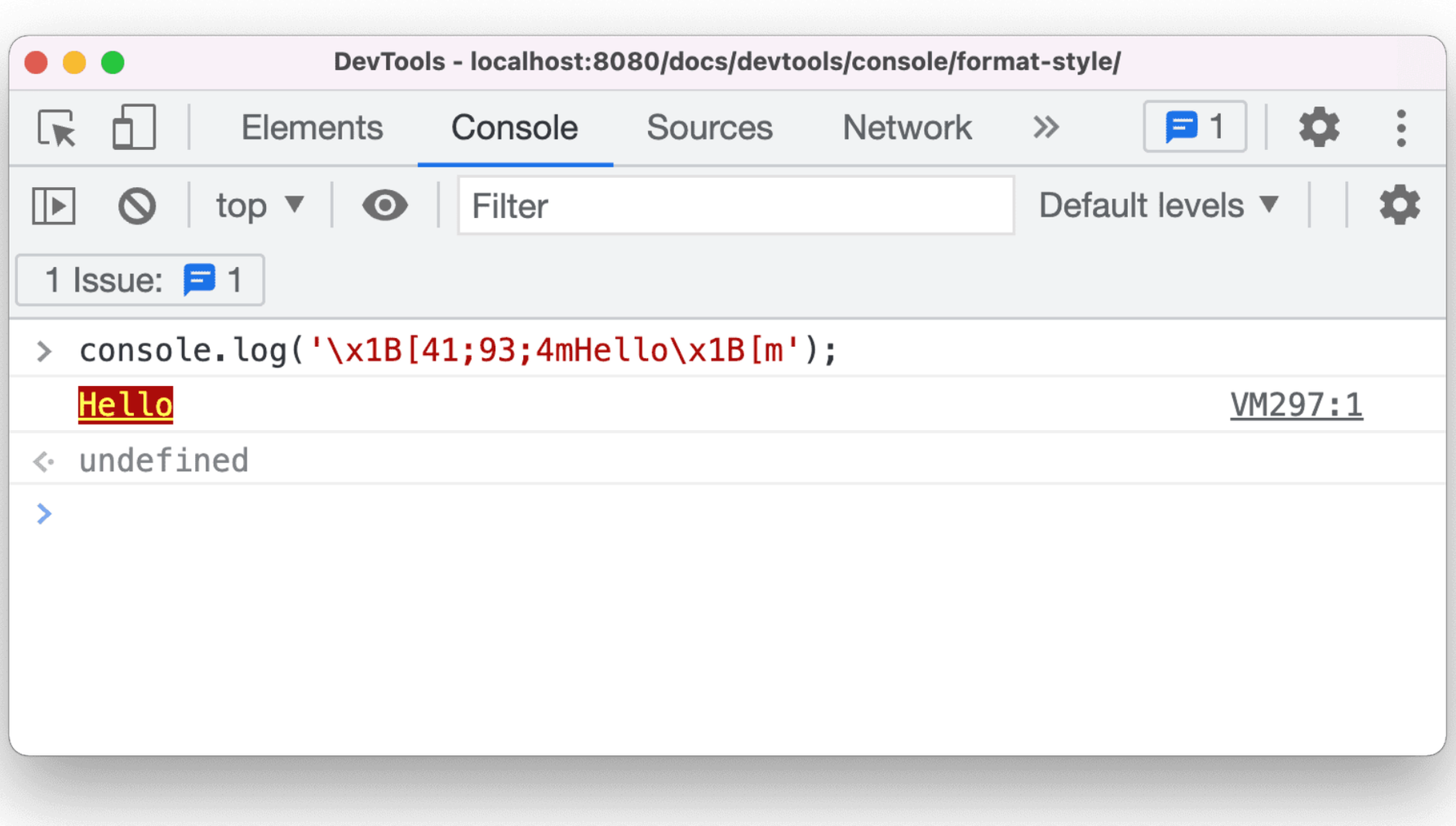
```
\x1B[P1;…;Pnm
```

Where,

- `P1` to `Pn` are valid subsequences of SGR (Select Graphic Rendition) parameters.
- Any of the parameters `P1` to `Pn` can be omitted, in which case its value is assumed to be zero.
- `\x1B[m` is the shorthand for `\x1B[0m`, in which the display attribute will be reset.

For example,

1. Enter the following console command. `js console.log('\x1B[41;93;4mHello\x1B[m');`
2. The command above produces a `Hello` message with red background, yellow text and underlined.



Here is a list of color codes supported in DevTools.

Foreground	Background	Light theme	Dark theme
30	40	#00000	#00000
31	41	#AA0000	#ed4e4c
32	42	#00AA00	#01c800
33	43	#AA5500	#d2c057
34	44	#0000AA	#2774f0
35	45	#AA00AA	#a142f4
36	46	#00AAAA	#12b5cb
37	47	#AAAAAA	#cfd0d0
90	100	#555555	#898989
91	101	#FF5555	#f28b82
92	102	#55FF55	#01c801
93	103	#FFFF55	#ddfb55
94	104	#5555FF	#669df6
95	105	#FF55FF	#d670d6
96	106	#55FFFF	#84f0ff
97	107	#FFFFFF	#FFFFFF

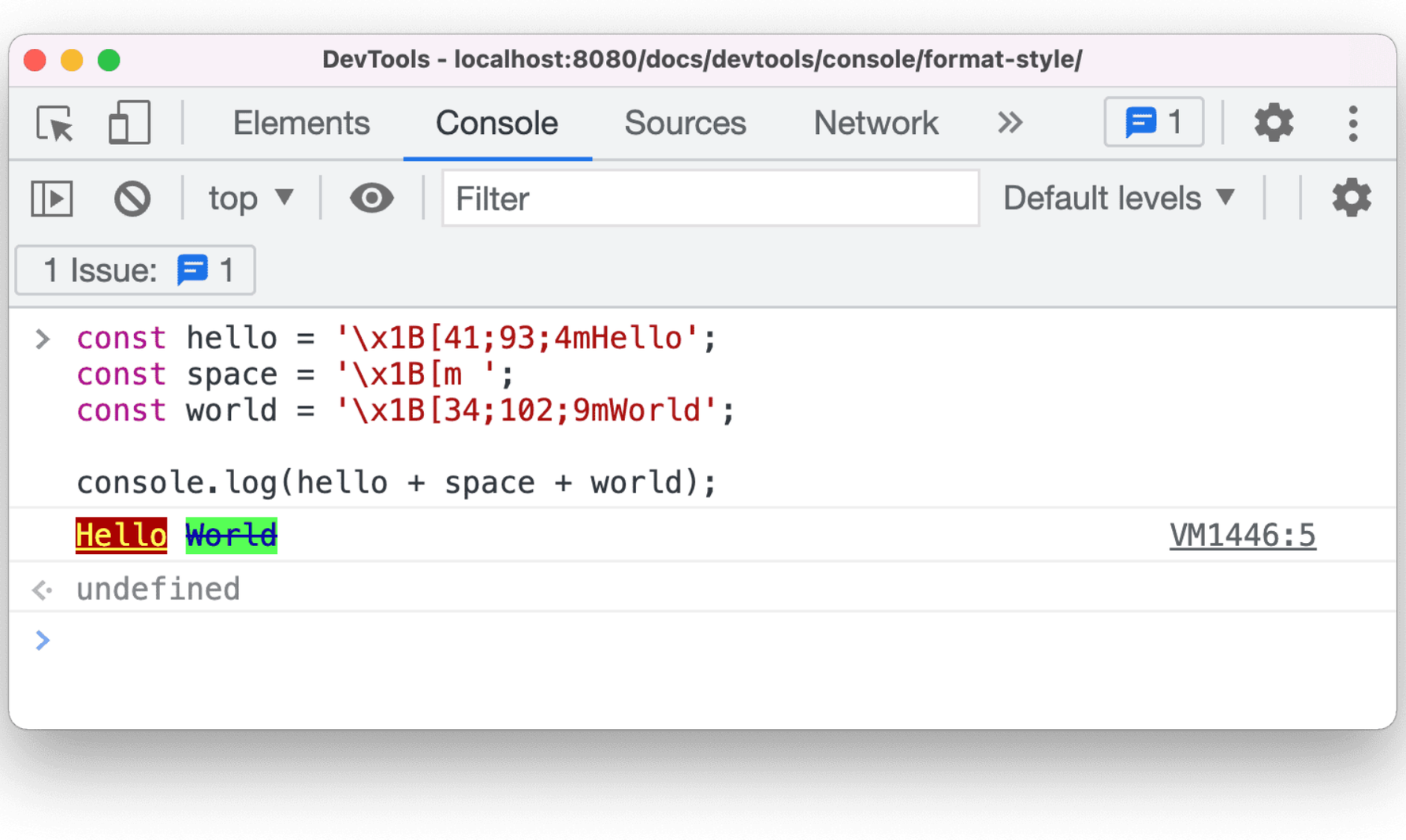
Here is a list of styling code supported in DevTools.

Parameter(s)	Meaning
0	Reset all display attributes
1	`font-weight: bold`
2	`font-weight: lighter`
3	`font-style: italic`
4	Add `underline` to `text-decoration` property
9	Add `line-through` to `text-decoration` property
22	Reset `font-weight` property
23	Reset `font-style` property
24	Remove `underline` from `text-decoration` property
29	Remove `line-through` from `text-decoration` property
38;2;R;G;B	`color: rgb(R,G,B)`
39	Reset `color property`
48;2;R;G;B	`background: rgb(R,G,B)`
49	Reset `background property`
53	Add `overline` to `text-decoration` property
55	Remove `overline` from `text-decoration` property

Here is another more complex example with multiple stylings.

1. Enter the following console command. ```js const hello = '\x1B[41;93;4mHello'; const space = '\x1B[m '; const world = '\x1B[34;102;9mWorld';

   console.log(hello + space + world); ```
2. The command above produces a `Hello World` message with 3 diffrent styles.



Was this helpful?

Last updated 2021-02-21 UTC.