JavaScript Developer Tools

Mohamed Emary

August 21, 2023

Console overview

The effect of the `console` methods may be different in different browsers. The following table shows the behavior in Chrome. The behavior also may not appear in windows terminal as expected so you better test these in browser dev tools not inside VSCode. Example console methods: `console.log()`, `console.warn()`, `console.error()`, ...etc.

The main difference between all these methods is how they display the data that you're logging. But you can also make things like changing page content from console. Example `document.querySelector('h1').textContent = 'Hello World'` will change the text of the first `h1` element in the page to `Hello World`. Console also allows you to try out new things and test your code.

There is even a function that lets you debug your code from the console. Example `debug(hello)` will pause the execution of the function `hello` and let you debug it.

Methods of `console.`	Description
`warn()`	Logs warning messages to the console.
`error()`	Logs error messages to the console.
`clear()`	Clears the console. You can also clear messages with the Clear button
`table()`	Displays data in a tabular format.
`dir()`	Displays an interactive list of the properties of the specified JavaScript object.
`group()`	Creates a new inline group, indenting all following output by another level. To move back out a level, call `groupEnd()`.
`groupCollapsed()`	Creates a new inline group, indenting all following output by another level. However, unlike `group()`, starts with the inline group collapsed requiring the use of a disclosure button to expand it. To move back out a level, call `groupEnd()`.
`count()`	Logs the number of times that this particular call to `count()` has been called.
`trace()`	Outputs a stack trace to the console.
`time()`	Starts a timer (can track how long an operation takes).

Example Usage

```
console.log("Hello World");
console.warn("Hello World");
console.error("Hello World");
```

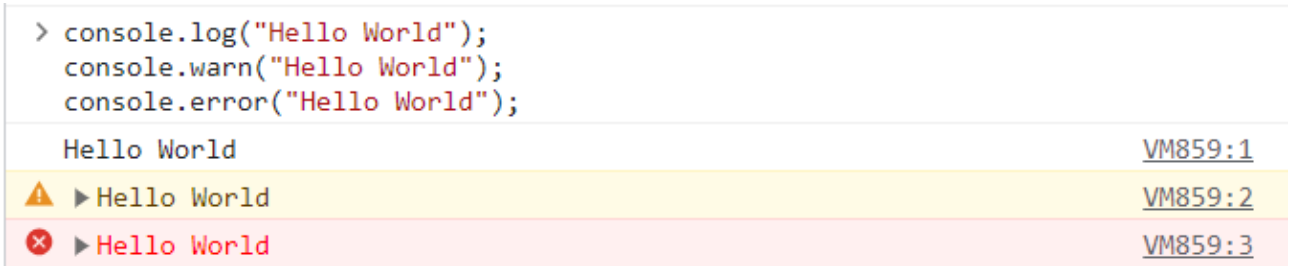

Figure 1: Output 1

```
console.clear(); // Just cleaars the console
console.table(["Hello", "World"]);
console.table([
  { name: "Mohamed", age: 25, job: "Software Engineer" },
  { name: "Ahmed", age: 30, job: "Doctor" },
  { name: "Ali", age: 20, job: "Student" },
]);
console.dir(["Hello", "World"]);
```

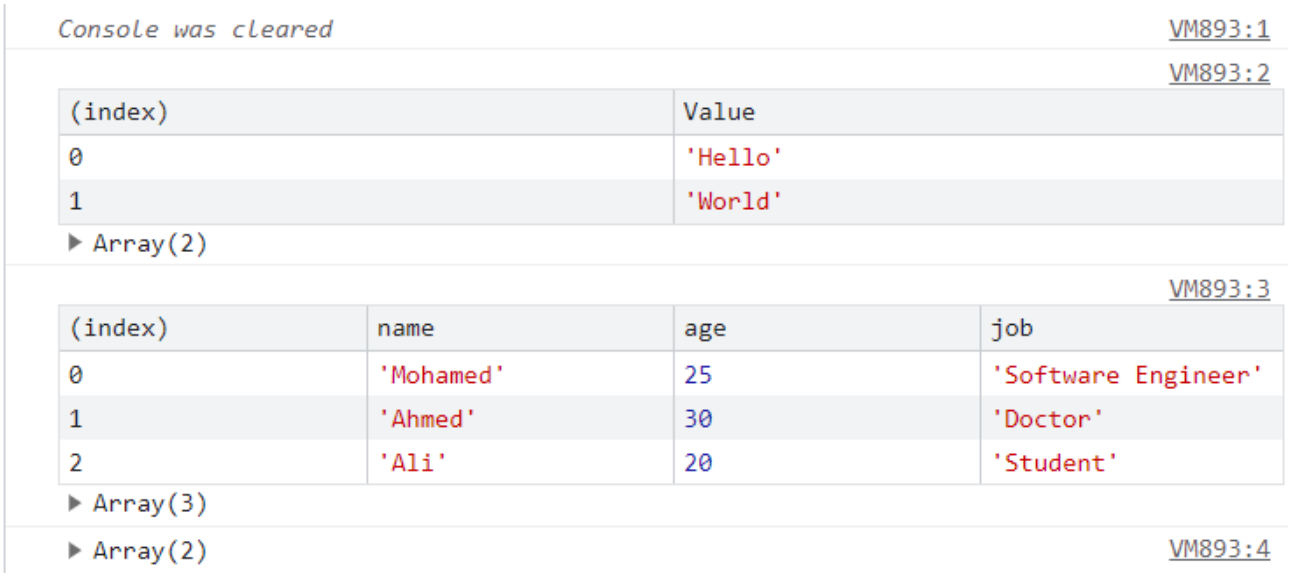

Figure 2: Output 2

Notice that the input lines doesn't appear in the image above because we used `console.clear()`.

```
console.group('Group 1');
console.log('Message 1');
console.log('Message 2');
```

```
console.groupEnd();

console.group('Group 2');
console.log('Message 3');
console.log('Message 4');
console.groupEnd();
```

```
> console.group('Group 1');
  console.log('Message 1');
  console.log('Message 2');
  console.groupEnd();

  console.group('Group 2');
  console.log('Message 3');
  console.log('Message 4');
  console.groupEnd();
```

```
▼ Group 1                                   VM989:1
    Message 1                               VM989:2
    Message 2                               VM989:3
▼ Group 2                                   VM989:6
    Message 3                               VM989:7
    Message 4                               VM989:8
```

Figure 3: Output 3

```
console.groupCollapsed('Group 1');
console.log('Message 1');
console.log('Message 2');
console.groupEnd();

console.groupCollapsed('Group 2');
console.log('Message 3');
console.log('Message 4');
console.groupEnd();
```

```
> console.groupCollapsed('Group 1');
  console.log('Message 1');
  console.log('Message 2');
  console.groupEnd();

  console.groupCollapsed('Group 2');
  console.log('Message 3');
  console.log('Message 4');
  console.groupEnd();
```

```
▶ Group 1                                   VM993:1
▶ Group 2                                   VM993:6
```

Figure 4: Output 4

```
console.count('Label 1');
console.count('Label 2');
console.count('Label 1');
console.count('Label 1');
console.count('Label 2');
```

```
> console.count('Label 1');
  console.count('Label 2');
  console.count('Label 1');
  console.count('Label 1');
  console.count('Label 2');
  Label 1: 1                                    VM997:1
  Label 2: 1                                    VM997:2
  Label 1: 2                                    VM997:3
  Label 1: 3                                    VM997:4
  Label 2: 2                                    VM997:5
```

Figure 5: Output 5

```
function foo() {
  console.trace();
}

function bar() {
  foo();
}

bar();
```

```
> function foo() {
    console.trace();
  }

  function bar() {
    foo();
  }

  bar();
  ▼console.trace                                VM1003:2
    foo          @ VM1003:2
    bar          @ VM1003:6
    (anonymous) @ VM1003:9
```

Figure 6: Output 6

```
console.time("Timer 1");

for (let i = 0; i < 1000000; i++) {
```

```
4    if (i === 5000) {
5      console.log("i = 5000");
6    }
7  }
8
9  console.timeEnd("Timer 1");
```

```
> console.time("Timer 1");

  for (let i = 0; i < 1000000; i++) {
      if (i === 5000) {
          console.log("i = 5000");
      }
  }

  console.timeEnd("Timer 1");
  i = 5000                                                    VM1009:5
  Timer 1: 7.236083984375 ms                                  VM1009:9
```

Figure 7: Output 7