**Console Logs, Errors, Warnings & More | JavaScript Tutorial In Hindi #2**

In this tutorial, we will learn how to run **JavaScript** in the Chrome Console. We can open our console in the web browser by using: ***Ctrl + Shift + K***or by Right-click on any webpage, click *Inspect*, and then we can see the innards of that site; its source code, the CSS that form its design, the JavaScript code that powers animations, and more. It has a console option as well, where we can run our JavaScript code.

In JavaScript, a **console** is an object which provides access to the browser debugging console. This object provides us with several different methods like **log(), error() ,table()** etc. Each method provides different functionalities. Following is the description of these methods along with examples.

**Console.log():-**

This method is used to log(print) the output to the console. We can put anything inside the log(). It can be an array, object, string, boolean, etc.

**Example:-**

console.log('CodeWithHarry');

console.log(1);

console.log(true);

console.log(null); ;

console.log([1, 2, 3]); // array inside log

console.log({name:"Harry", language:"JavaScript", tutorial:2}); // object inside log

Copy

**Output:-**

Graphical user interface, application

Description automatically generated with medium confidence

**Console.table ():-**

To generate a table inside a console, we use console.table() method. The input must be an array or an object which will be displayed as a table. In the example, we provide the object as an input.

**Example:-**

console.table({name:"Harry", language:"JavaScript", tutorial:2});

Copy

**Output:-**

Application

Description automatically generated with medium confidence

**Console.assert():-**

This method writes a message to the console that the assertion failed and the message we provide as a parameter, but only if an expression evaluates to *false*. If the expression is true, then nothing will happen.

**Example:-**

console.assert(0>1, "Expression is false")

Copy

**Output:-**



**Console.warn():-**

This method is used to log a warning message to the console. By default, the warning message will be highlighted with yellow color.

**Example:-**

console.warn("This is a warning");

Copy

**Output:-**



**Console.clear():-**

It is used to clear the console. The console will be cleared. In the case of Chrome, a simple overlayed text will be printed on the console.

**Example:-**

console.clear();

Copy

**Output:-**

Table

Description automatically generated

**Console.time() and Console.timeEnd():-**

With the help of console.time() and console.timeEnd() we can find the amount of time spend by a code on execution.

**Example:-**

console.time();

for (i = 0; i < 100; i++) {

// code

}

console.timeEnd();

Copy

**Output:-**

Text

Description automatically generated with medium confidence

**Console.error():-**

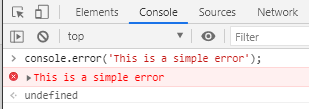
Used to log error message to the console. Useful in the testing of code. By default, the error message will be highlighted with red color.

**Example:-**

console.error("This is a simple error");

Copy

**Output:-**



**Console.count():-**

The console.count() method is used to count the number that the function hit by this counting method.

**Example:-**

for (i = 0; i<4; i++) {

console.count(i);

}

Copy

**Output:-**

Graphical user interface, application

Description automatically generated

**Console.group() and Console.groupEnd():-**

group() and groupEnd() methods of the console object allow us to group contents in a separate block, indented. Just like the time() and the timeEnd(), they also accept the label, again of the same value.

**Example:-**

console.group('simple');

console.log('Grouped');

console.groupEnd('simple');

console.log('new section');

Copy

**Output:-**

Graphical user interface, application, table

Description automatically generated

**Custom Console logs:-**

If the user has even a little idea about CSS, they can add Styling to the console logs to make logs Custom. The Syntax for it is to add the CSS styling as a parameter to the logs, which will replace %c in the logs as shown in the example below:

**Example:-**

const spacing = '8px';

const mystyle =

`padding: ${spacing}; background-color: white; color: blue ; font-style:

italic; border: 1px solid black dotted; font-size: 2em;`;

console.log('%cCode With Harry', mystyle);

Copy

**Output:-**



**Code index.html as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Tutorial on Js</title>

</head>

<body>

<h1>This is Js tutorial by Harry</h1>

</body>

<!-- <script src="js/tut2.js"></script> -->

<!-- <script src="js/tut3.js"></script> -->

<!-- <script src="js/tut4.js"></script> -->

<!-- <script src="js/tut5.js"></script> -->

<!-- <script src="js/tut6.js"></script> -->

<!-- <script src="js/tut7.js"></script> -->

<!-- <script src="js/tut8.js"></script> -->

<!-- <script src="js/tut9.js"></script> -->

<!-- <script src="js/tut10.js"></script> -->

<script src="js/tut11.js"></script>

</html>

Copy

**Js code as described/written in the video**

console.time('Your code Took');

console.log('Hello console');

console.log(4+34);

console.log(34);

console.log(true);

console.log([34,2,1,2]);

console.log({harry: 'this', marks:34});

console.table({harry: 'this', marks:34});

console.warn('This is a warning');

// console.clear();

console.timeEnd('Your code Took');

// console.assert(566<189, 'Age >189 is not possible')

// console.error('This is an error')

/\*

this

is a

multiline comment

\*/