In this thread I want to discuss the console API and what it is, how it is used, and how often it should be used in a JavaScript program. Let’s start with a definition of the console API. The console API “is the go-to debugging tool of millions of developers when working with JavaScript. You can access it using the **console** interface. It is accessible via the global **window** object. You can use this API to view object/variable values on runtime or log messages on your console.”*(Interactive Guide to Console API, 2022).*

All browsers will reveal a console that will allow us to interact “with the web platform APIs and give an inside look at the code by printing messages that are generated by your JavaScript code running in the page.” (Copes, 2019). You can use the console API by performing simple logging methods. The following is a short list of simple logging methods:

* console.clear() – clears the console
* console.debug(object1, object2, ..etc) – although it is preferred to use console.log() as it does the same thing.
* console.error(object1, object2, ..etc) – logs the parameters to the console. In browsers the logged content may be marked by an “error” icon and/ or include a stack trace or a link to the code.
* console.warn(object1, object2, ..etc) - logs the parameters to the console. In browsers the logged content may be marked by an “warning” icon and/ or include a stack trace or a link to the code.

(Rauschmayer, 2013).

We can also do counting and checking by performing the methods below:

* console.assert(expr, obj?) – If **expr** is false, log **obj** to the console and throw an exception. If it is **true**, do nothing.
* Console.count(label?) – counts how many times the line with this statement is executed with this label.

(Rauschmayer, 2013).

We can also do formatted logging by performing the methods below:

* console.dir(object) – Print a representation of the object to the console.
* Console.group(object1, object2 ..etc) – log the objects to the console and open a nested block that contains all future logged content.
* Console.groupCollapsed(object1, object2, ..etc) – works like console.group() but the block is initially collapsed.
* Console.groupEnd() – closes a group that has been opened by console.group() or consolegroupCollapsed().

(Rauschmayer, 2013).

We can also perform timing methods as listed below:

* Console.time(label) – start a timer whose label is label.
* Console.timeEnd(label) – stops the timer whose label is label and print the time that has elapsed since starting it.
* Console.timeStamp(label?) – Log a time stamp with the given label. May be logged to the console or a timeline.

(Rauschmayer, 2013)

In reference to how often the console API should be used in a JavaScript program, it is wise to consider that “as we develop and not only in JavaScript, it is regularly necessary to read the information contained in a variable or the execution result.” (Sulikowski, 2021). It is pertinent that as we our developing and writing our code to take advantage of the console API to minimize mistakes, to debug programs that are not running as they should, and to maximize the performance of the web pages that we are building.

In conclusion the console API in JavaScript can be a developer’s best friend as it is an invaluable tool to help develop and debug our apps. There are so many helpful methods that we can use to maximize the performance of our web pages and to display the messages that we need to get our programs to run efficiently and reliably. As we will use it often to log informational, warning, and error messages to the console, we can see why this tool is so important to help us as developers as we bring our creations to the world.

References:

*Interactive Guide to Console API*. (2022, May 30). RapidAPI GUIDES. Retrieved June 20,2022, from <https://rapidapi.com/guides/console-api>

Rauschmayer, A. (2013, October 5). *The JavaScript API*. 2ality – JavaScript and More. Retrieved June 20, 2022, from <https://2ality.com/2013/10/console-api.html>

Copes, F. (2019, September 26). *How to work with the DevTools Console and Console API: an overview.* freeCodeCamp.Org. Retrieved June 20, 2022, from <https://www.freecodecamp.org/news/working-with-the-devtools-console-and-console-api-an-overview-13cff6dc3db4/>

Sulikowski, S. (2021, December 9). *How to improve the use of the console in javascript*. DEV Community. Retrieved June 20, 2022, from <https://dev.to/suliste/how-to-improve-the-use-of-the-console-in-javascript-44ha>