getfirebug.com

Console API

Firebug adds a global variable named "console" to all web pages loaded in Firefox. This object contains many methods that allow you to write to the Firebug console to expose information that is flowing through your scripts.

- 1 Commands
  - 1.1 console.log(object[, object, ...])
  - 1.2 console.debug(object[, object, ...])
  - 1.3 console.info(object[, object, ...])
  - 1.4 console.warn(object[, object, ...])
  - 1.5 console.error(object[, object, ...])
  - 1.6 console.assert(expression[, object, ...])
  - 1.7 console.clear()
  - 1.8 console.dir(object)
  - 1.9 console.dirxml(node)
  - 1.10 console.trace()
  - 1.11 console.group(object[, object, ...])
  - 1.12 console.groupCollapsed(object[, object, ...])
  - 1.13 console.groupEnd()
  - 1.14 console.time(name)
  - 1.15 console.timeEnd(name)
  - 1.16 console.profile([title])
  - 1.17 console.profileEnd()
  - 1.18 console.count([title])
  - 1.19 console.exception(error-object[, object, ...])
  - 1.20 console.table(data[, columns])
- 2 Implementation Notes
  - 2.1 Firebug 1.4
  - 2.2 Firebug 1.3
  - 2.3 Firebug 1.2
  - 2.4 Firebug 1.1 and earlier
- 3 See also

[edit] Commands

[edit] console.log(object[, object, ...])

Writes a message to the console. You may pass as many arguments as you'd like, and they will be joined together in a space-delimited line.

The first argument to `log` may be a string containing printf-like string substitution patterns. For example:

```
console.log("The %s jumped over %d tall buildings", animal, count);
```

The example above can be re-written without string substitution to achieve the same result:

```
console.log("The", animal, "jumped over", count, "tall buildings");
```

These two techniques can be combined. If you use string substitution but provide more arguments than there are substitution patterns, the remaining arguments will be appended in a space-delimited line, like so:

```
console.log("I am %s and I have:", myName, thing1, thing2, thing3);
```

If objects are logged, they will be written not as static text, but as interactive hyperlinks that can be clicked to inspect the object in Firebug's HTML, CSS, Script, or DOM tabs. You may also use the %o pattern to substitute a hyperlink in a string.

You may also use the %c pattern to use the second argument as a style formatting parameter. For example:

```
console.log('%cThis is red text on a green background', 'color:red; background-color:green');
```

Here is the complete set of patterns that you may use for string substitution:

Pattern	Type
%s	String
%d, %i	Integer (numeric formatting is not yet supported)
%f	Floating point number (numeric formatting is not yet supported)
%o	Object hyperlink
%c	Style formatting

[edit] console.debug(object[, object, ...])

Writes a message to the console, including a hyperlink to the line where it was called.

[edit] console.info(object[, object, ...])

Writes a message to the console with the visual "info" icon and color coding and a hyperlink to the line where it was called.

[edit] console.warn(object[, object, ...])

Writes a message to the console with the visual "warning" icon and color coding and a hyperlink to the line where it was called.

[edit] console.error(object[, object, ...])

Writes a message to the console with the visual "error" icon and color coding and a hyperlink to the line where it was called.

[edit] console.assert(expression[, object, ...])

Tests that an expression is true. If not, it will write a message to the console and throw an exception.

[edit] console.clear()

Clears the console.

[edit] console.dir(object)

Prints an interactive listing of all properties of the object. This looks identical to the view that you would see in the DOM tab.

[edit] console.dirxml(node)

Prints the XML source tree of an HTML or XML element. This looks identical to the view that you would see in the HTML tab. You can click on any node to inspect it in the HTML tab.

[edit] console.trace()

Prints an interactive stack trace of JavaScript execution at the point where it is called.

The stack trace details the functions on the stack, as well as the values that were passed as arguments to each function. You can click each function to take you to its source in the Script tab, and click each argument value to inspect it in the DOM or HTML tabs.

[edit] console.group(object[, object, ...])

Writes a message to the console and opens a nested block to indent all future messages sent to the console. Call `console.groupEnd()` to close the block.

[edit] console.groupCollapsed(object[, object, ...])

Like `console.group()`, but the block is initially collapsed.

[edit] console.groupEnd()

Closes the most recently opened block created by a call to `console.group()` or `console.groupCollapsed()`

[edit] console.time(name)

Creates a new timer under the given name. Call `console.timeEnd(name)` with the same name to stop the timer and print the time elapsed..

[edit] console.timeEnd(name)

Stops a timer created by a call to `console.time(name)` and writes the time elapsed.

[edit] console.profile([title])

Turns on the JavaScript profiler. The optional argument `title` would contain the text to be printed in the header of the profile report.

[edit] console.profileEnd()

Turns off the JavaScript profiler and prints its report.

[edit] console.count([title])

Writes the number of times that the line of code where `count` was called was executed. The optional argument `title` will print a message in addition to the number of the count.

[edit] console.exception(error-object[, object, ...])

Prints an error message together with an interactive stack trace of JavaScript execution at the point where the exception occurred.

[edit] console.table(data[, columns])

Allows to log provided data using tabular layout. The method takes one required parameter that represents table like data (array of arrays or list of objects). The other optional parameter can be used to specify columns and/or properties to be logged (see more here).

[edit] Implementation Notes

The `console` is an object attached to the `window` object in the web page. In Firebug for Firefox the object is attached only if the Console panel is enabled. In Firebug lite, the console is attached if Lite is installed in the page.

[edit] Firebug 1.4

The console is implemented by adding a `div` element and a `script` tag to the web page just before the first Javascript script tag is run. So the first script tag is compiled, then the console is injected, then the outer function code of the script tag is executed.

[edit] Firebug 1.3

As in Firebug 1.4

[edit] Firebug 1.2

The code and tags are added on document load event.

[edit] Firebug 1.1 and earlier

The console is implemented with an insecure technique

[edit] See also

Original URL:

http://getfirebug.com/wiki/index.php/Console_API