Firebug Cheat Sheet Version 1.2

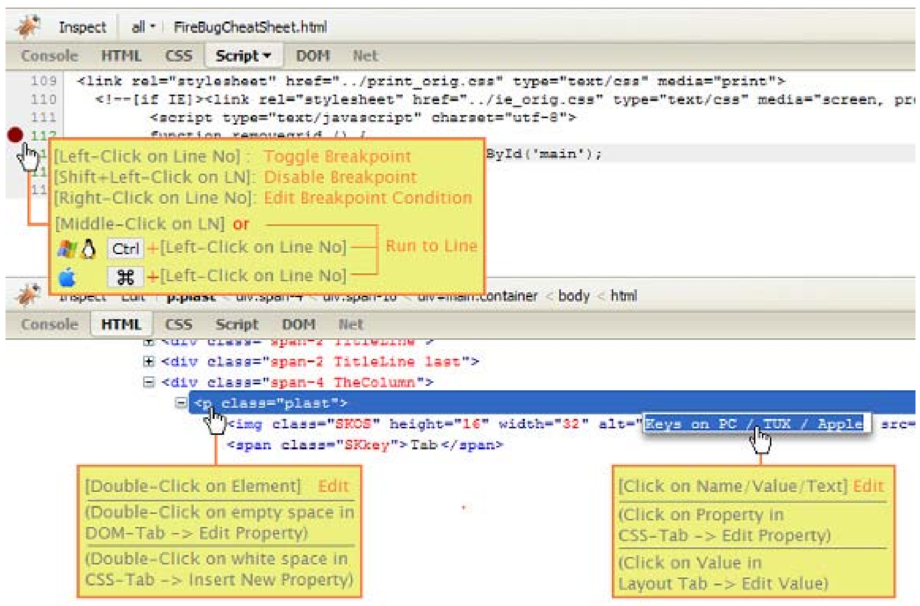
KEYBOARD SHORTCUTS / MOUSE ACTIONS

OS / KEY COMBINATION	HTML	SCRIPT	DOM	CSS	LAYOUT	CONSOLE
Tab	Advance to Next Field E		Auto-complete Next Property E	Advance to Next Field E	Advance to Next Field E	Auto-complete Next Property S
Shift + Tab	Advance to Previous Field E		Auto-complete Prev. Property E	Advance to Previous Field E	Advance to Previous Field E	Auto-complete Prev. Property S
Esc	Cancel Editing E Cancel Inspect I		Cancel Editing E	Cancel Editing E	Cancel Editing E	
Return	Finish Editing E		Finish Editing E	Finish Editing E	Finish Editing E	Execute S
Shift + Return						Inspect Results S
🪟🐧 Ctrl + Return 🍎 ⌘ + Return						Execute L Open Context-Menu: Results S
Up				Increase Number by One / Auto-complete Next Keyword E	Increase Number by One E	
🪟🐧 Ctrl + Up 🍎 ⌘ + Up	Inspect Parent I					
Down				Decrease Number by One / Auto-complete Previous Keyword E	Decrease Number by One E	
🪟🐧 Ctrl + Down 🍎 ⌘ + Down	Inspect Child I					
Page Up				Increase Number by Ten E	Increase Number by Ten E	
Page Down				Decrease Number by Ten E	Decrease Number by Ten E	
🪟🐧 F5 or Ctrl + / 🍎 ⌘ + /		Continue T				
🪟🐧 F10 or Ctrl + ' 🍎 ⌘ + '		Step Over T				
🪟🐧 F11 or Ctrl + ; 🍎 ⌘ + ;		Step Into T				
🪟🐧 Shift + F11 or Ctrl + Shift + ; 🍎 ⌘ + Shift + ;		Step Out T				
Ctrl + .	Next Node in Path T	Next Function on Stack T	Next Object in Stack T			
Ctrl + ,	Previous Node in Path T	Prev. Function on Stack T	Previous Object in Stack T			
🪟🐧 Ctrl + Space 🍎 ⌘ + Shift + Space		Focus Menu of Scripts T		Focus Menu of Style Scripts T		
🪟🐧 Ctrl + Shift + N 🍎 ⌘ + Shift + N		Focus Watch Editor T				

E Editor for this Tab T In the Tab I Inspect S Small Console L Large Console

OS / KEY COMBINATION	GLOBAL
F12	Open / Close Firebug Panel
🪟🐧 Ctrl + F12 🍎 ⌘ + F12	Open Firebug in Window
🪟🐧 Ctrl + ` 🍎 Option + Tab	Switch to Previous Tab
🪟🐧 Ctrl + Shift + L 🍎 ⌘ + Shift + L	Focus Command Line
🪟🐧 Ctrl + Shift + K 🍎 ⌘ + Shift + K	Focus Search Box
🪟🐧 Ctrl + Shift + C 🍎 ⌘ + Shift + C	Toggle Inspect Mode
🪟🐧 Ctrl + Shift + P 🍎 ⌘ + Shift + P	Toggle JavaScript Profiler
Ctrl + Shift + E	Re-Execute Last Command Line

MOUSE SHORTCUTS



```
<link rel="stylesheet" href="../print_orig.css" type="text/css" media="print">
  <!--[if IE]><link rel="stylesheet" href="../ie_orig.css" type="text/css" media="screen, pr
        <script type="text/javascript" charset="utf-8">
```

[Left-Click on Line No] : Toggle Breakpoint
[Shift+Left-Click on LN]: Disable Breakpoint
[Right-Click on Line No]: Edit Breakpoint Condition
[Middle-Click on LN] or
🪟🐧 Ctrl +[Left-Click on Line No]
🍎 ⌘ +[Left-Click on Line No]
Run to Line

p.plast < div.span-4 < div.span-10 < div#main.container < body < html
Console
HTML
CSS
Script
DOM
Net

```
<div class="span-2 TitleLine last">
<div class="span-4 TheColumn">
  <p class="plast">
    <img class="SKOS" height="16" width="32" alt="Keys on PC / TUX / Apple src=
    <span class="SKkey">Tab</span>
```

[Double-Click on Element] Edit
(Double-Click on empty space in DOM-Tab -> Edit Property)
(Double-Click on white space in CSS-Tab -> Insert New Property)

[Click on Name/Value/Text] Edit
(Click on Property in CSS-Tab -> Edit Property)
(Click on Value in Layout Tab -> Edit Value)

Firebug Cheat Sheet Version 1.2

COMMAND LINE API

A list of the available command line options.

$(id)	Returns a single element with the given id
$$(selector)	Returns an array of elements that match the given CSS selector.
$x(xpath)	Returns an array of elements that match the given XPath expression.
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.
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.
cd(window)	By default, command line expressions are relative to the top-level window of the page. cd() allows you to use the window of a frame in the page instead.
clear()	BClears the console.
inspect(object[, tabName])	Inspects an object in the most suitable tab, or the tab identified by the optional argument tabName. The available tab names are "html", "css", "script", and "dom".
keys(object)	Returns an array containing the names of all properties of the object.
values(object)	Returns an array containing the values of all properties of the object.
debug(fn)	Adds a breakpoint on the first line of a function.
monitor(fn)	Turns on logging for all calls to a function.
unmonitor(fn)	Turns off logging for all calls to a function.
monitorEvents(object[, types])	Turns on logging for all events dispatched to an object. The optional argument types may specify a specific family of events to log. The most commonly used values for types are "mouse" and "key". The full list of available types includes "composition", "contextmenu", "drag", "focus", "form", "key", "load", "mouse", "mutation", "paint", "scroll", "text", "ui", and "xul".
unmonitorEvents(object[, types])	Turns off logging for all events dispatched to an object.
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.
profileEnd()	Turns off the JavaScript profiler and prints its report.
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.
console.count([title])	Writes the number of times that the line of code where "count" was was called has been executed. The optional argument title will print a message in addition to the number of the count.<

CONSOLE API

A list of methods of the console object that Firebug adds to the loaded web page(s). Check the Firebug documentation pages at http://getfirebug.com/ for further information.

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.
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. (Available substring patterns: "%s" String - "%d, %i" Integer - "%f" Floating point number) - "%o" Object hyperlink
console.debug(object[, object, ...])	Writes a message to the console, including a hyperlink to the line where it was called.
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.
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.
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.
console.assert(expression[, object, ...])	Tests that an expression is true. If not, it will write a message to the console and throw an exception.
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.
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.
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.
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.
console.groupEnd()	Closes the most recently opened block created by a call to "console.group."
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..
console.timeEnd(name)	Stops a timer created by a call to "console.time(name)" and writes the time elapsed.
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.
console.profileEnd()	Turns off the JavaScript profiler and prints its report.
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.