

Debugging JS with Chrome

>The little power tools



Igor Zalutsky



Async call stacks

```
setTimeout(function () {  
  $.ajax({  
    url: 'http://echo.jsonstest.com/foo/bar'  
  }).done(function(response) {  
    debugger;  
  });  
}, 500);
```



▼ Call Stack		<input type="checkbox"/> Async
(anonymous function)		(Index):26
jQuery.Callbacks.fire	jquery-git2.js:3073	
jQuery.Callbacks.self.fireWith	jquery-git2.js:3185	
done	jquery-git2.js:8245	
jQuery.ajaxTransport.send.callback	jquery-git2.js:8592	

▼ Call Stack		<input checked="" type="checkbox"/> Async
(anonymous function)		(Index):26
jQuery.Callbacks.fire	jquery-git2.js:3073	
jQuery.Callbacks.self.fireWith	jquery-git2.js:3185	
done	jquery-git2.js:8245	
jQuery.ajaxTransport.send.callback	jquery-git2.js:8592	
XMLHttpRequest.send (async)		
jQuery.ajaxTransport.send	jquery-git2.js:8617	
jQuery.extend.ajax	jquery-git2.js:8146	
(anonymous function)	(index):23	
setTimeout (async)		
(anonymous function)	(index):22	
jQuery.event.dispatch	jquery-git2.js:4405	
jQuery.event.add.elemData.handle	jquery-git2.js:4091	
DOMWindow.addEventListener("load") (async)		
jQuery.event.add	jquery-git2.js:4137	
(anonymous function)	jquery-git2.js:4831	
jQuery.extend.each	jquery-git2.js:375	
jQuery.fn.jquery.each	jquery-git2.js:139	
jQuery.fn.extend.on	jquery-git2.js:4830	
jQuery.each.jQuery.fn.(anonymous function)	jquery-git2.js:7454	
jQuery.fn.load	jquery-git2.js:8822	
(anonymous function)	(index):21	

Console API: the basics

```
var obj = {  
  prop: 1, child: {  
    prop: 2  
  }  
};  
console.log(obj);  
console.log('%O\n%o', document.body, document.body);  
console.warn('something unexpected happened');  
console.error('something went wrong');  
console.assert("str" instanceof String, 'gotcha!');
```

```
▼ Object {prop: 1, child: Object} ⓘ  
  ► child: Object  
    prop: 1  
    __proto__: Object  
  ► body.two-column.docs  
    ► <body class="two-column docs">...</body>  
  ⚠ something unexpected happened  
  ✖ ▼ something went wrong  
    (anonymous function)  
    InjectedScript._evaluateOn  
    InjectedScript._evaluateAndWrap  
    InjectedScript.evaluate  
  ✖ ► Assertion failed: gotcha!
```

Console API: time() and count()

```
function fibonacci(n) {  
  return (n > 1) ? fibonacci(n - 2) + fibonacci(n - 1) : n;  
}  
console.time('fibonacci');  
console.log(fibonacci(30));  
console.timeEnd('fibonacci');  
  
document.addEventListener('keydown', function(event) {  
  console.count(String.fromCharCode(event.keyCode));  
});
```

```
832040  
fibonacci: 14.798ms  
< undefined  
W: 1  
W: 2  
W: 3  
A: 1  
S: 1  
D: 1  
D: 2
```

Console from the inside

`console.dir(console);`

```
▼ Console ⓘ
  ▼ memory: MemoryInfo
    jsHeapSizeLimit: 793000000
    totalJSHeapSize: 97400000
    usedJSHeapSize: 86400000
  ▼ __proto__: MemoryInfo
    ► constructor: function MemoryInfo() { [native code] }
    ► __proto__: Object
  ▼ __proto__: Console
    ► constructor: function Console() { [native code] }
    ► __proto__: ConsoleBase
```



```
► assert: function assert() { [native code] }
► clear: function clear() { [native code] }
► constructor: function ConsoleBase() { [native code] }
► count: function count() { [native code] }
► debug: function debug() { [native code] }
► dir: function dir() { [native code] }
► dirxml: function dirxml() { [native code] }
► error: function error() { [native code] }
► group: function group() { [native code] }
► groupCollapsed: function groupCollapsed() { [native code] }
► groupEnd: function groupEnd() { [native code] }
► info: function info() { [native code] }
► log: function log() { [native code] }
► markTimeline: function markTimeline() { [native code] }
► profile: function profile() { [native code] }
► profileEnd: function profileEnd() { [native code] }
► table: function table() { [native code] }
► time: function time() { [native code] }
► timeEnd: function timeEnd() { [native code] }
► timeStamp: function timeStamp() { [native code] }
► timeline: function timeline() { [native code] }
► timelineEnd: function timelineEnd() { [native code] }
► trace: function trace() { [native code] }
► warn: function warn() { [native code] }
```

Console API: table()

```
var people = [  
  {  
    name: 'Hopper', surname: 'Herring'  
  }, {  
    name: 'Sampson', surname: 'Douglas'  
  }, {  
    name: 'Carmella', surname: 'Vincent'  
  }  
];  
console.table(people);  
console.table(  
  document.querySelectorAll('a'),  
  ['href', 'text']  
);
```

(index) ▲	name	surname
0	"Hopper"	"Herring"
1	"Sampson"	"Douglas"
2	"Carmella"	"Vincent"

VM3826:11

(index)	href	text ▲
132	"https://developers.g...	" "
gc-logo	"https://developers.g...	""
0	"https://developers.g...	""
68	"https://developers.g...	"0.1"
58	"https://developers.g...	"1.0"
47	"https://developers.g...	"1.1"
130	"http://www.apache.or...	"Apache 2.0 License"
13	"https://developers.g...	"Authoring and Develo...
96	"https://developers.g...	"Blog posts"
136	"https://developers.g...	"Careers"

Console API: other methods

<code>clear()</code>	remove all previous output
<code>debug()</code> , <code>info()</code>	aliases for <code>log()</code>
<code>group()</code> , <code>groupEnd()</code>	grouped output
<code>profile()</code> , <code>profileEnd()</code>	last evaluated expression
<code>timeStamp()</code>	marking the timeline
<code>trace()</code>	stack trace

Digging out Command Line API

```
(function() {  
  debugger;  
})();
```

```
with (__commandLineAPI || { __proto__: null }) {  
  (function() {  
    debugger;  
  })();  
}
```

```
var members = Object.keys(__commandLineAPI)  
  .filter(function(key) {  
    return !(key in console);  
  }).reduce(function(obj, key) {  
    obj[key] = __commandLineAPI[key];  
    return obj;  
  }, {});  
console.dir(members);
```


Command Line API: overview

▼ Object ⓘ

- ▶ `$0`: `div#gc-wrapper`
- ▶ `$1`: `div#needAuth.msgDialogContent`
- ▶ `$2`: `div#gc-appbar`
- ▶ `$3`: `li`
- ▶ `$4`: `img`
- ▶ `$$`: `function $(selector, [startNode]) { [Command Line API] }`
- ▶ `$_`: `Object`
- ▶ `$x`: `function $x(xpath, [startNode]) { [Command Line API] }`
- ▶ `copy`: `function copy(object) { [Command Line API] }`
- ▶ `getEventListeners`: `function getEventListeners(node) { [Command Line API] }`
- ▶ `inspect`: `function inspect(object) { [Command Line API] }`
- ▶ `keys`: `function keys(object) { [Command Line API] }`
- ▶ `monitor`: `function monitor(fn) { [Command Line API] }`
- ▶ `monitorEvents`: `function monitorEvents(object, [types]) { [Command Line API] }`
- ▶ `undebug`: `function undebug(fn) { [Command Line API] }`
- ▶ `unmonitor`: `function unmonitor(fn) { [Command Line API] }`
- ▶ `unmonitorEvents`: `function unmonitorEvents(object, [types]) { [Command Line API] }`
- ▶ `values`: `function values(object) { [Command Line API] }`

Command Line API: shortcuts

<code>\$(selector)</code>	<code>document.querySelector()</code>
<code>\$\$ (selector)</code>	<code>document.querySelectorAll()</code>
<code>\$_</code>	last evaluated expression
<code>\$0 - \$4</code>	last 5 inspected DOM nodes or heap entries
<code>\$x(path)</code>	XPath query

Command Line API: monitoring events

```
> monitorEvents(document, 'mousedown');
undefined
mousedown ► MouseEvent {dataTransfer: null, toElement: div#gc-wrapper, fromElement: null, y: 116, x: 171...}
mousedown ► MouseEvent {dataTransfer: null, toElement: th, fromElement: null, y: 74, x: 394...}
> monitorEvents(document, 'key');
undefined
mousedown ► MouseEvent {dataTransfer: null, toElement: div#gc-wrapper, fromElement: null, y: 130, x: 189...}
keydown ► KeyboardEvent {altGraphKey: false, repeat: false, metaKey: false, altKey: false, shiftKey: false...}
keypress ► KeyboardEvent {altGraphKey: false, repeat: false, metaKey: false, altKey: false, shiftKey: false...}
keyup ► KeyboardEvent {altGraphKey: false, repeat: false, metaKey: false, altKey: false, shiftKey: false...}
> unmonitorEvents(document, ['keyup', 'keydown']);
undefined
mousedown ► MouseEvent {dataTransfer: null, toElement: td, fromElement: null, y: 147, x: 345...}
keypress ► KeyboardEvent {altGraphKey: false, repeat: false, metaKey: false, altKey: false, shiftKey: false...}
>
```

Command Line API: hidden hooks

```
> function f() {};  
monitor(f);  
f(1); f(2); f(3);
```

```
function f called with arguments: 1
```

```
function f called with arguments: 2
```

```
function f called with arguments: 3
```

```
< undefined
```

```
> debug(f);  
f(4);
```

```
function f called with arguments: 4
```

```
< undefined
```

```
> undebug(f);  
unmonitor(f);
```

```
1 with ( __commandLineAPI || { __proto__: null }) {  
2   function f() {};  
3   monitor(f);  
4   f(1); f(2); f(3);  
5 }
```

Command Line API: other methods

<code>copy(object)</code>	copies string representation to clipboard
<code>inspect(object)</code>	shows object in DOM inspector or profiler
<code>getEventListeners(object)</code>	returns hash of arrays of listeners
<code>profile(), profileEnd()</code>	last evaluated expression
<code>keys(object)</code>	same as <code>Object.keys(object)</code>
<code>values(object)</code>	returns array of object's values

Thanks!

Useful links

- > [Google Chrome Console API docs](#)
- > [Google Chrome Command Line API docs](#)
- > [“Lesser-Known JavaScript Debugging Techniques”](#) by Amjad Masad
- > [“Advanced JavaScript Debugging with console.table\(\)”](#) by Marius Schulz

Follow me!



@igorzij



github.com/zij



linkedin.com/in/izalutsky