



October 4-6, 2017 | Vancouver, BC

# new debugger features

Yang Guo, Google, V8

[@hashseed](#)



October 4-6, 2017 | Vancouver, BC

**DevTools protocol**  
**code coverage**  
**type profile**  
**debug-evaluate**



# LEGACY V8 DEBUG API

JSON message API

debug context JavaScript API

C++ API

DebugCommandProcessor

--expose-debug-as, vm.runInDebugContext

v8/include/v8-debug.h



# LEGACY V8 DEBUG API

JSON message API	DebugCommandProcessor
debug context JavaScript API	--expose-debug-as, vm.runInDebugContext
C++ API	v8/include/v8-debug.h



# CURRENT V8 DEBUG API

JSON message API	DebugCommandProcessor
debug context JavaScript API	--expose-debug-as, vm.runInDebugContext
C++ API	v8/include/v8-debug.h
DevTools protocol	--inspect

```
yangguo@yangguo2: ~  
File Edit View Search Terminal Help  
[2017-09-29 10:53:16] yangguo@yangguo2:~$ ~/node/node --inspect  
Debugger listening on ws://127.0.0.1:9229/5ca793d5-3cd7-4408-8787-b3aaaded9fef  
For help see https://nodejs.org/en/docs/inspector  
> Debugger attached.  
> 
```



# DEVTOOLS PROTOCOL IN V8

Chrome with Chrome DevTools

F12, CTRL+SHIFT+J

Node.js with Chrome DevTools

--inspect

Chrome with Node.js

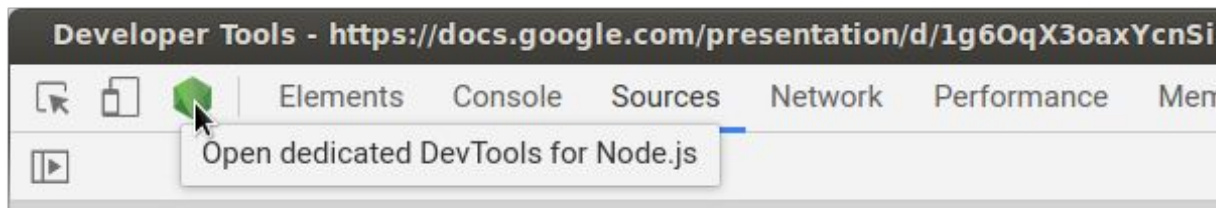
[github.com/GoogleChrome/puppeteer](https://github.com/GoogleChrome/puppeteer)

Node.js with Node.js

`require('inspector')`

Node.js with telnet

;-)



```

{
  "id": 0,
  "method": "Debugger.enable"
}
{
  "method": "Debugger.scriptParsed",
  "params": {
    "scriptId": "14",
    "url": "test/mjsunit/mjsunit.js",
    "startLine": 0,
    "startColumn": 0,
    "endLine": 826,
    "endColumn": 0,
    "executionContextId": 1,
    "hash":
      "8A11A530A7BFCC85AF929B31548DDBF56FF75175",
    "isLiveEdit": false,
    "sourceMapURL": "",
    "hasSourceURL": false,
    "isModule": false,
    "length": 27412,
    "stackTrace": {
      "callFrames": [
        {
          "functionName": "sendMessage",
          "scriptId": "15",
          "url":
            "test/debugger/test-api.js",
          "lineNumber": 360,
          "columnNumber": 4
        }
      ]
    }
  }
}
{
  "method": "Debugger.paused",
  "params": {
    "callFrames": [
      {
        "callFrameId":
          "{ \"ordinal\":0, \"injectedScriptId\":1 }",
        "functionName": "f",
        "functionLocation": {
          "scriptId": "16",
          "lineNumber": 24,
          "columnNumber": 10
        },
        "location": {
          "scriptId": "16",
          "lineNumber": 25,
          "columnNumber": 2
        },
        "url": "test.js",
        "scopeChain": [
          {
            "type": "local",
            "object": {
              "type": "object",
              "className": "Object",
              "description": "Object",
              "objectId": "{ \"injectedScriptId\":1, \""
            }
          },
          {
            "name": "f",
            "startLocation": {
              "scriptId": "16",
              "lineNumber": 24,
              "columnNumber": 10
            },
            "endLocation": {
              "scriptId": "16",
              "lineNumber": 26,
              "columnNumber": 1
            }
          }
        ]
      }
    ]
  }
}

```



# INSPECTOR IN NODE

```
const inspector = require('inspector');  
const session = new inspector.Session();  
session.connect();  
session.post(command, arguments, callback);  
session.disconnect();
```

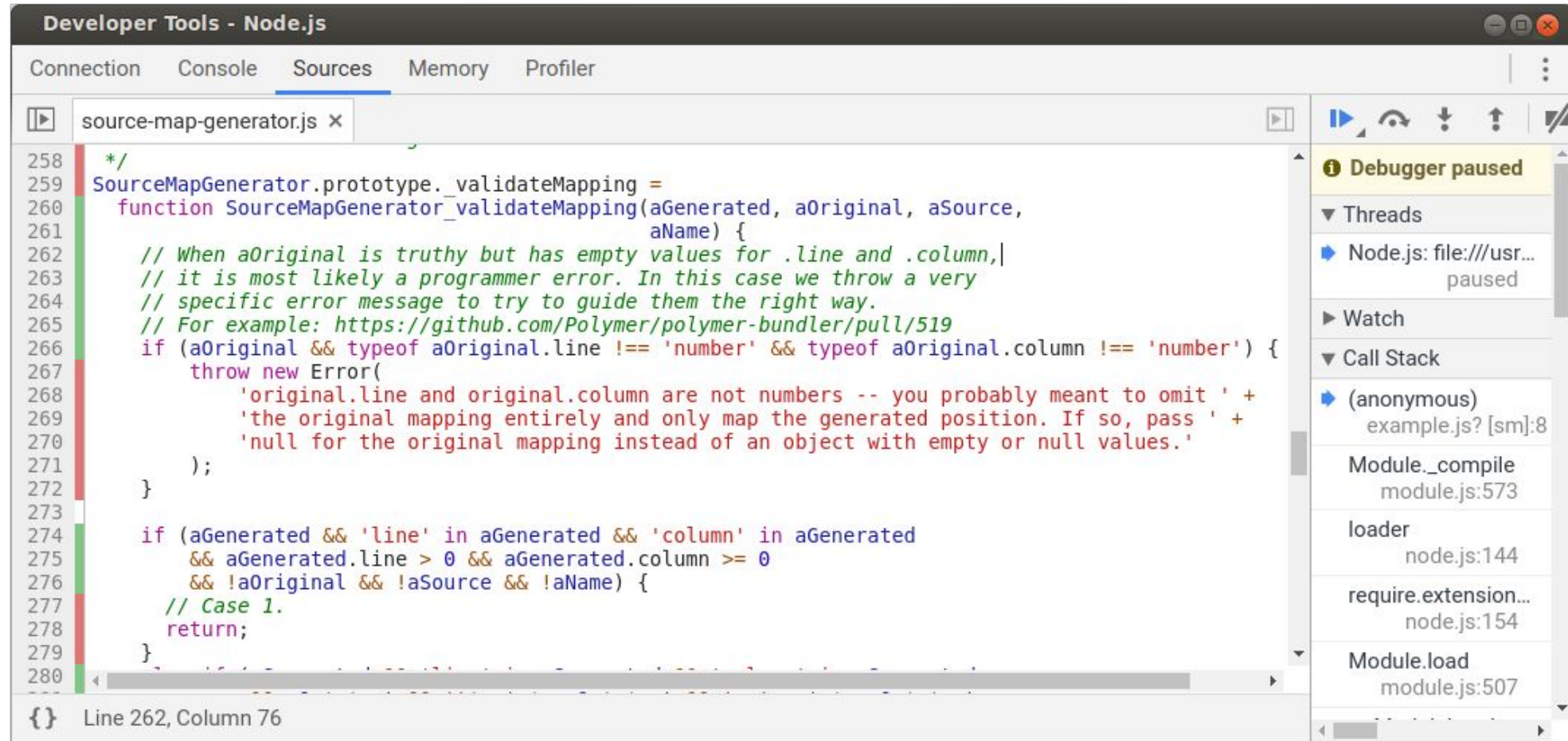




# CODE COVERAGE



# CODE COVERAGE



Developer Tools - Node.js

Connection Console Sources Memory Profiler

source-map-generator.js x

```
258 */
259 SourceMapGenerator.prototype.validateMapping =
260   function SourceMapGenerator_validateMapping(aGenerated, aOriginal, aSource,
261     aName) {
262     // When aOriginal is truthy but has empty values for .line and .column,|
263     // it is most likely a programmer error. In this case we throw a very
264     // specific error message to try to guide them the right way.
265     // For example: https://github.com/Polymer/polymer-bundler/pull/519
266     if (aOriginal && typeof aOriginal.line !== 'number' && typeof aOriginal.column !== 'number') {
267       throw new Error(
268         'original.line and original.column are not numbers -- you probably meant to omit ' +
269         'the original mapping entirely and only map the generated position. If so, pass ' +
270         'null for the original mapping instead of an object with empty or null values.'
271       );
272     }
273
274     if (aGenerated && 'line' in aGenerated && 'column' in aGenerated
275       && aGenerated.line > 0 && aGenerated.column >= 0
276       && !aOriginal && !aSource && !aName) {
277       // Case 1.
278       return;
279     }
280
```

Debugger paused

Threads

- Node.js: file:///usr... paused

Watch

Call Stack

- (anonymous) example.js? [sm]:8
- Module.\_compile module.js:573
- loader node.js:144
- require.extensions... node.js:154
- Module.load module.js:507

{ } Line 262, Column 76



# CODE COVERAGE

branch-based code coverage  
accurate call counts  
accessible via DevTools protocol  
can be translated to common  
coverage formats, e.g. for LCOV

	Line data	Source code
1	1	function thrower() {
2	0	throw new RangeError();
3	1	}
4	177	function fib(n) {
5	177	if (n == 0) {
6	34	return 0;
7	177	} else if (n == 1) {
8	55	return 1;
9	143	} else if (n > 1) {
10	88	return fib(n - 1) + fib(n - 2);
11	0	} else {
12	0	thrower();
13	177	}
14	177	}
15	1	console.log("fib(10):", fib(10));
16	1	

*Coverage report generated by LCOV*



DEMO



# CODE COVERAGE

## script

```
function fib(x) {  
  if (x < 2) {  
    return 1;  
  }  
  return fib(x-1) + fib(x-2);  
}  
  
function dead() {  
  unreachable;  
}  
  
var failed = false;  
try {  
  fib(8);  
} catch (e) {  
  failed = true;  
}
```

☒ Call count ☒ Detailed coverage

obtain coverage

## coverage

```
function fib(x) {  
  if (x < 2) {  
    return 1;  
  }  
  return fib(x-1) + fib(x-2);  
}  
  
function dead() {  
  unreachable;  
}  
  
var failed = false;  
try {  
  fib(8);  
} catch (e) {  
  failed = true;  
}
```



# PROTOCOL FOR COVERAGE

## **Profiler.startPreciseCoverage**

Enable precise code coverage. Coverage data for JavaScript executed before enabling precise code coverage may be incomplete. Enabling prevents running optimized code and resets execution counters. **EXPERIMENTAL**

### PARAMETERS

<b>callCount</b> optional	<b>boolean</b> Collect accurate call counts beyond simple 'covered' or 'not covered'.
<b>detailed</b> optional	<b>boolean</b> Collect block-based coverage.

---

## **Profiler.stopPreciseCoverage**

Disable precise code coverage. Disabling releases unnecessary execution count records and allows executing optimized code. **EXPERIMENTAL**

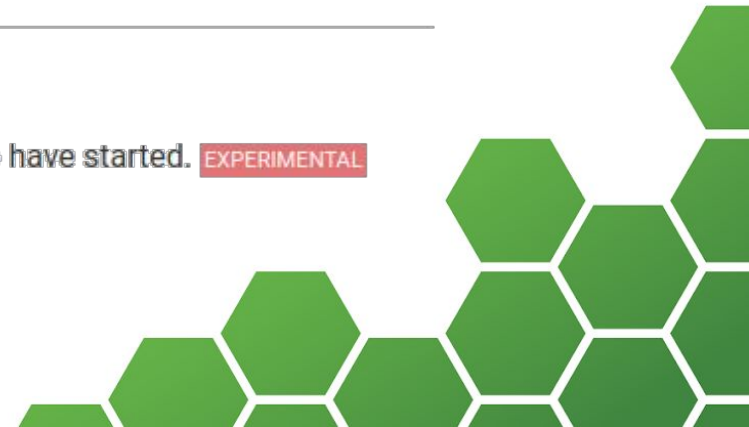
---

## **Profiler.takePreciseCoverage**

Collect coverage data for the current isolate, and resets execution counters. Precise code coverage needs to have started. **EXPERIMENTAL**

### RETURN OBJECT

<b>result</b>	<b>array [ <a href="#">ScriptCoverage</a> ]</b> Coverage data for the current isolate.
---------------	---



# PROTOCOL FOR COVERAGE

```
session.post("Profiler.enable", callback1);  
...  
session.post("Profiler.startPreciseCoverage",  
             { callCount: true, detailed: true },  
             callback2);  
...  
session.post("Profiler.takePreciseCoverage", callback3);
```



# TYPE PROFILE





# TYPE PROFILE

## script

```
function add(left,
              right) {
  return left + right;
}

class Potato {
  valueOf() { return 3; }
}
class Tomato {
  toString() { return "T"; }
}

console.log(add(1, 2));
console.log(add(1, "2"));
console.log(add(new Potato, new Tomato));
```

obtain type profile

## type profile

```
function add(number Potato left,
              number string Tomato right) {
  return left + right;
number string }

undefined class Potato {
  valueOf() { return 3; number }
}
undefined class Tomato {
  toString() { return "T"; string }
}

console.log(add(1, 2));
console.log(add(1, "2"));
console.log(add(new Potato, new Tomato));
```



# PROTOCOL FOR TYPE PROFILE

## **Profiler.startTypeProfile**

Enable type profile. **EXPERIMENTAL**

---

## **Profiler.stopTypeProfile**

Disable type profile. Disabling releases type profile data collected so far. **EXPERIMENTAL**

---

## **Profiler.takeTypeProfile**

Collect type profile. **EXPERIMENTAL**

RETURN OBJECT

result    **array [ [ScriptTypeProfile](#) ]**  
Type profile for all scripts since startTypeProfile() was turned on.



# PROTOCOL FOR TYPE PROFILE

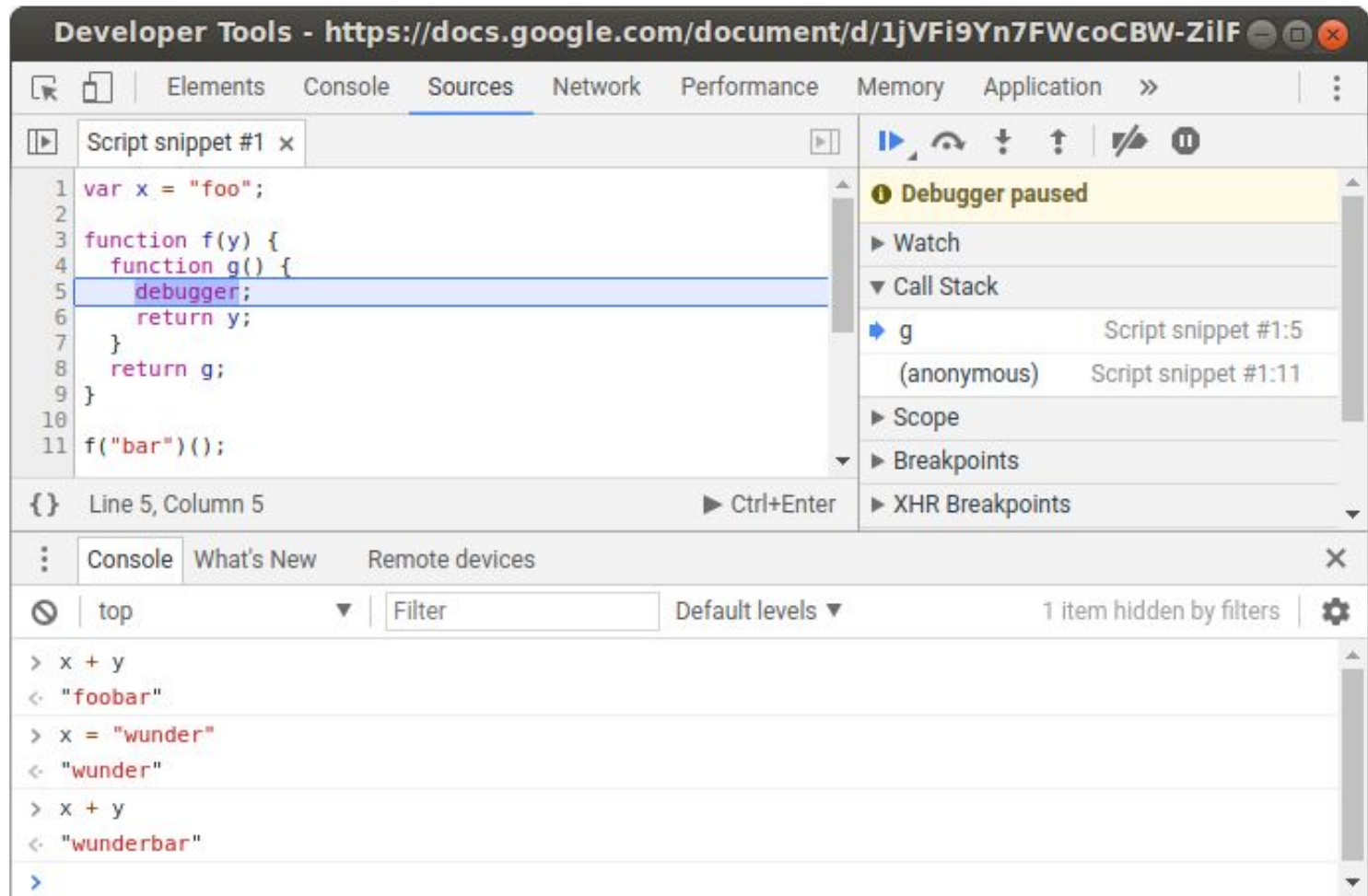
```
session.post("Profiler.enable", callback1);  
...  
session.post("Profiler.startTypeProfile", callback2);  
...  
session.post("Profiler.takeTypeProfile", callback3);
```



# DEBUG-EVALUATE



# DEBUG-EVALUATE



DEMO



# DEBUG-EVALUATE

## script

```
function f(x) {  
  return function g() {  
    debugger;  
    console.log(x);  
    return x;  
  }  
}  
  
f(2)();
```

evaluate at debugger statement

```
x = 3
```

run

☒ allow side effect

## result

3

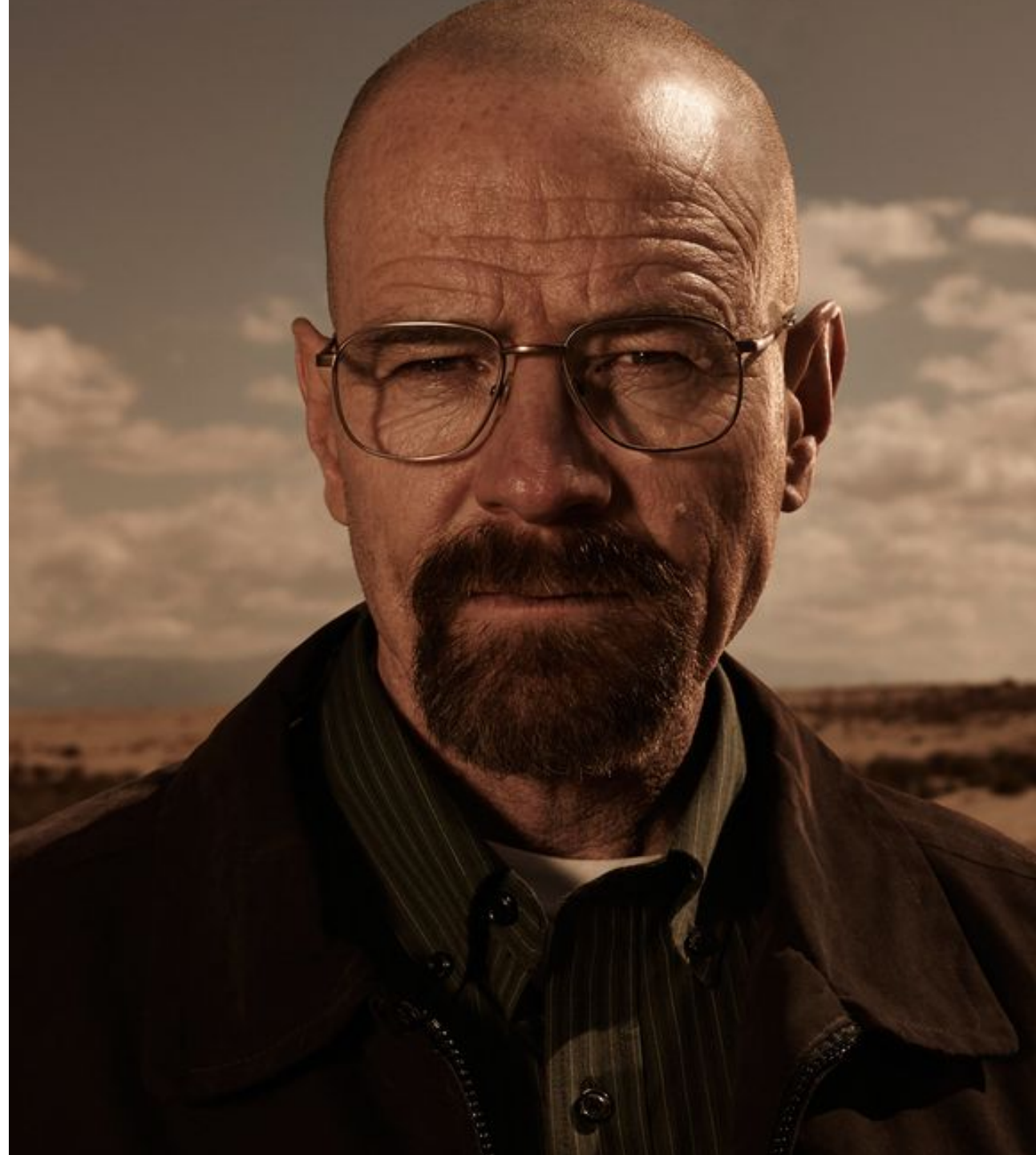
## console

console.log: 3



"Looking at something  
changes it"

- Werner Heisenberg





# DEBUG-EVALUATE W/O SIDE-EFFECTS

## script

```
function f(x) {  
  return function g() {  
    debugger;  
    console.log(x);  
    return x;  
  }  
}  
  
f(2)();
```

evaluate at debugger statement

```
x = 3
```

run

☐ allow side effect

## result

[exception]

## console

console.log: 2



# PROTOCOL FOR DEBUG-EVALUATE

## **Debugger.evaluateOnCallFrame**

Evaluates expression on a given call frame.

### PARAMETERS

<b>callFrameId</b>	<a href="#"><u>CallFrameId</u></a> Call frame identifier to evaluate on.
<b>expression</b>	<b>string</b> Expression to evaluate.
<b>objectGroup</b> <small>optional</small>	<b>string</b> String object group name to put result into (allows rapid releasing resulting object handles using <code>releaseObjectGroup</code> ).
<b>includeCommandLineAPI</b> <small>optional</small>	<b>boolean</b> Specifies whether command line API should be available to the evaluated expression, defaults to false.
<b>silent</b> <small>optional</small>	<b>boolean</b> In silent mode exceptions thrown during evaluation are not reported and do not pause execution. Overrides <code>setPauseOnException</code> state.
<b>returnByValue</b> <small>optional</small>	<b>boolean</b> Whether the result is expected to be a JSON object that should be sent by value.
<b>generatePreview</b> <small>optional</small>	<b>boolean</b> Whether preview should be generated for the result. <b>EXPERIMENTAL</b>
<b>throwOnSideEffect</b> <small>optional</small>	<b>boolean</b> Whether to throw an exception if side effect cannot be ruled out during evaluation. <b>EXPERIMENTAL</b>

### RETURN OBJECT

<b>result</b>	<a href="#"><u>Runtime.RemoteObject</u></a> Object wrapper for the evaluation result.
<b>exceptionDetails</b> <small>optional</small>	<a href="#"><u>Runtime.ExceptionDetails</u></a> Exception details.



# PROTOCOL FOR DEBUG-EVALUATE

`throwOnSideEffect` **boolean**  
optional Whether to throw an exception if side effect cannot be ruled out during evaluation. **EXPERIMENTAL**

```
session.post('Debugger.evaluateOnCallFrame',  
  { callFrameId,  
    expression,  
    throwOnSideEffect },  
  callback);
```



# LINKS

- Node.js with newest V8  
<https://github.com/v8/node/commits/vee-eight-lkgr>
- DevTools protocol documentation  
<https://chromedevtools.github.io/debugger-protocol-viewer/v8/>
- demo files  
<https://github.com/hashseed/node-coverage-demo>

