DEBUGGER!

WHAT IS IT GOOD FOR?

Stanislav Párnický @ stylers.cloud

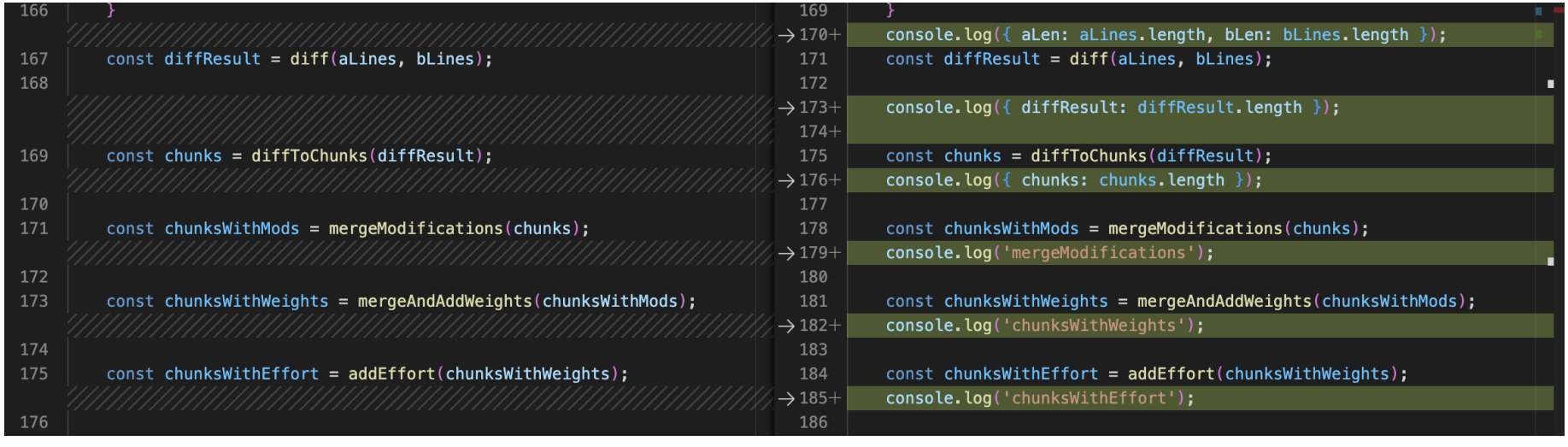
DEBUGGER

- inpect code execution
- show call stack
- read variables
- REPL

POLL: HOW DO YOU DEBUG?

1. temporary `console.log()`
2. write more tests
3. staring at code and thinking really hard
4. use debugger

1. TEMPORARY `console.log`

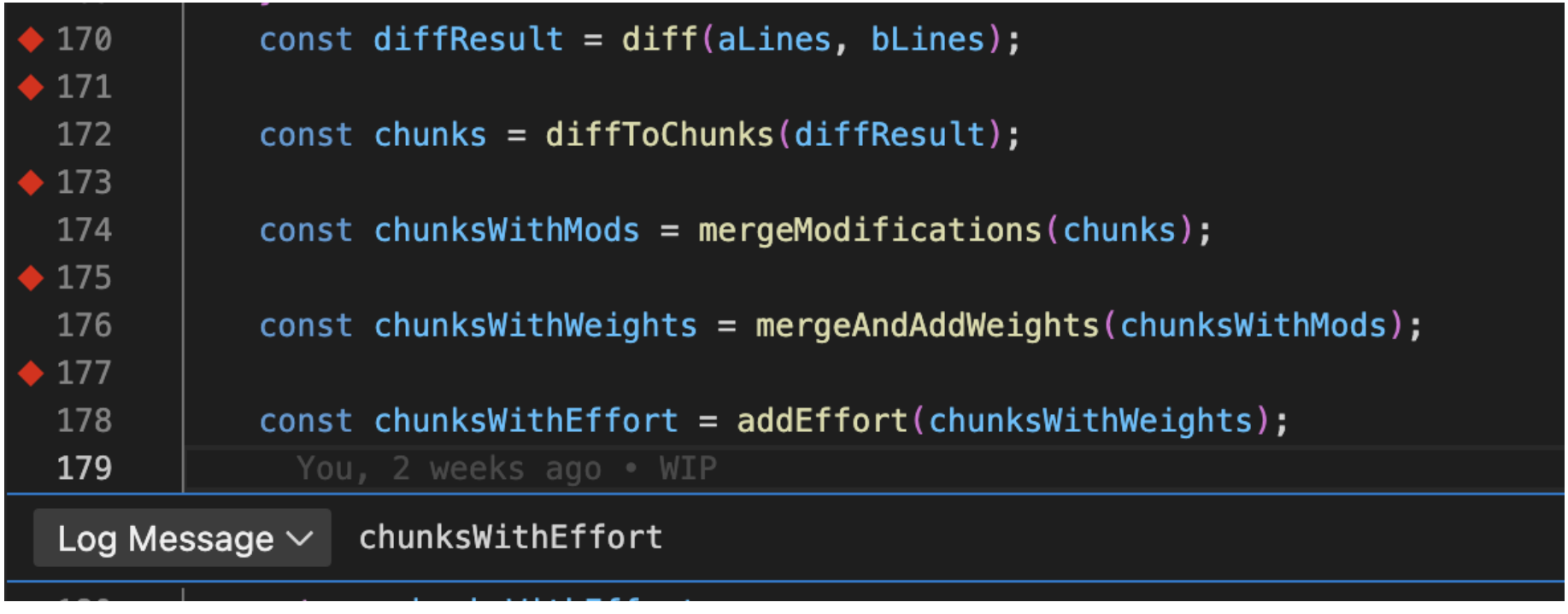


```
166     }
167     const diffResult = diff(aLines, bLines);
168
169     const chunks = diffToChunks(diffResult);
170
171     const chunksWithMods = mergeModifications(chunks);
172
173     const chunksWithWeights = mergeAndAddWeights(chunksWithMods);
174
175     const chunksWithEffort = addEffort(chunksWithWeights);
176
```

```
169     }
170+    console.log({ aLen: aLines.length, bLen: bLines.length });
171     const diffResult = diff(aLines, bLines);
172
173+    console.log({ diffResult: diffResult.length });
174+
175     const chunks = diffToChunks(diffResult);
176+    console.log({ chunks: chunks.length });
177
178     const chunksWithMods = mergeModifications(chunks);
179+    console.log('mergeModifications');
180
181     const chunksWithWeights = mergeAndAddWeights(chunksWithMods);
182+    console.log('chunksWithWeights');
183
184     const chunksWithEffort = addEffort(chunksWithWeights);
185+    console.log('chunksWithEffort');
186
```

I will just add few console.logs...

LOGPOINTS

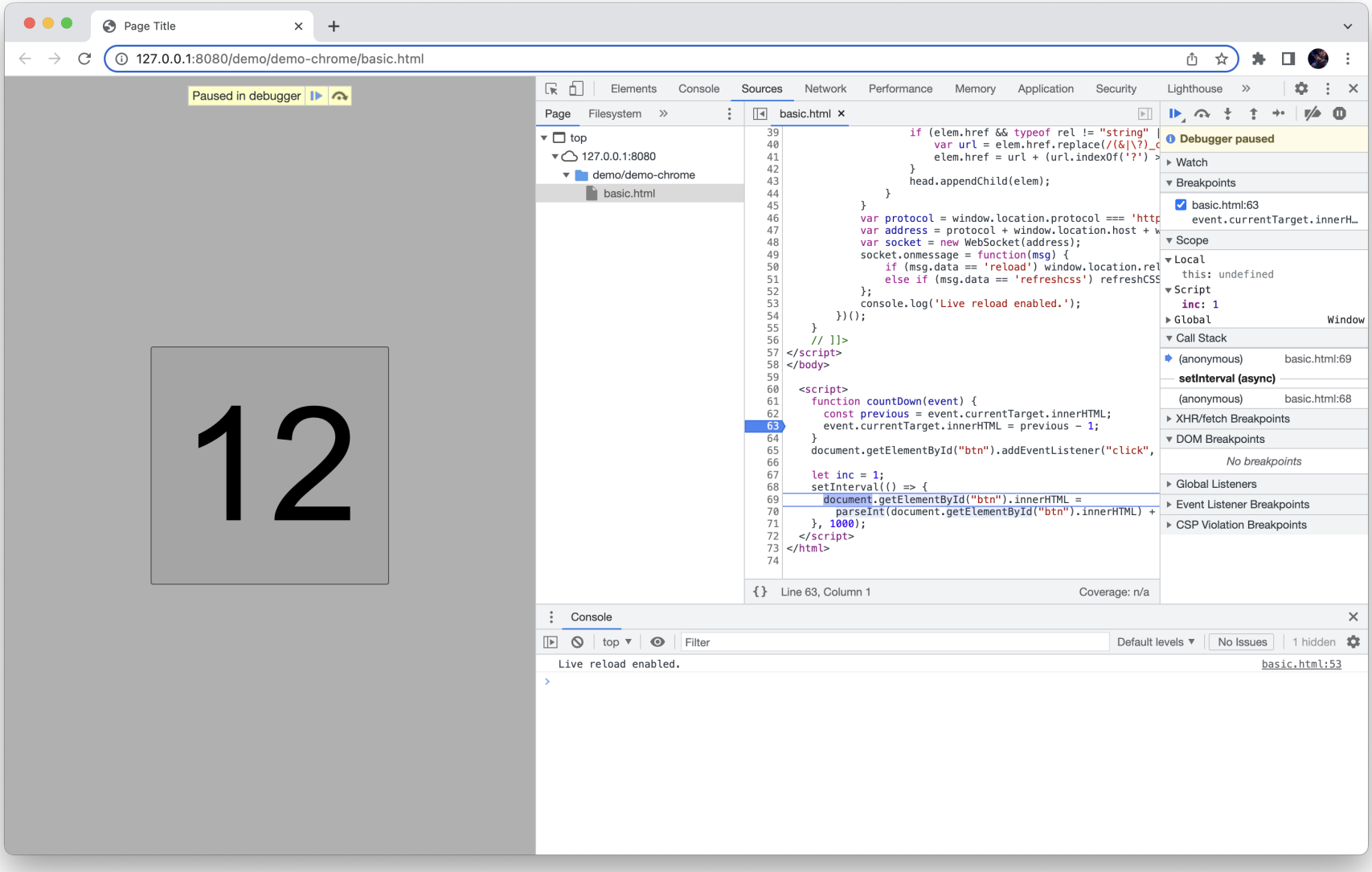


Easy to add, easy to remove

SO HOW DO I USE THIS DEBUGGER THING?

1. Open DevTools Sources tab
2. ???
3. Profit!!!

Demo in browser


Page Title
127.0.0.1:8080/demo/demo-chrome/basic.html
Paused in debugger
12
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Page
Filesystem
top
127.0.0.1:8080
demo/demo-chrome
basic.html
basic.html ×
39 if (elem.href && typeof rel != "string" |
40 var url = elem.href.replace(/(&|\?)_c
41 elem.href = url + (url.indexOf('?') >
42 }
43 head.appendChild(elem);
44 }
45 }
46 var protocol = window.location.protocol === 'http
47 var address = protocol + window.location.host + w
48 var socket = new WebSocket(address);
49 socket.onmessage = function(msg) {
50 if (msg.data == 'reload') window.location.rel
51 else if (msg.data == 'refreshcss') refreshCSS
52 };
53 console.log('Live reload enabled.');
54 })();
55 }
56 //]]>
57 </script>
58 </body>
59
60 <script>
61 function countDown(event) {
62 const previous = event.currentTarget.innerHTML;
63 event.currentTarget.innerHTML = previous - 1;
64 }
65 document.getElementById("btn").addEventListener("click",
66
67 let inc = 1;
68 setInterval(() => {
69 document.getElementById("btn").innerHTML =
70 parseInt(document.getElementById("btn").innerHTML) +
71 }, 1000);
72 </script>
73 </html>
74
{} Line 63, Column 1
Coverage: n/a
Debugger paused
Watch
Breakpoints
basic.html:63
event.currentTarget.innerH…
Scope
Local
this: undefined
Script
inc: 1
Global
Window
Call Stack
(anonymous)
basic.html:69
setInterval (async)
(anonymous)
basic.html:68
XHR/fetch Breakpoints
DOM Breakpoints
No breakpoints
Global Listeners
Event Listener Breakpoints
CSP Violation Breakpoints
Console
top
Filter
Default levels
No Issues
1 hidden
Live reload enabled.
basic.html:53

2. WRITE MORE TESTS

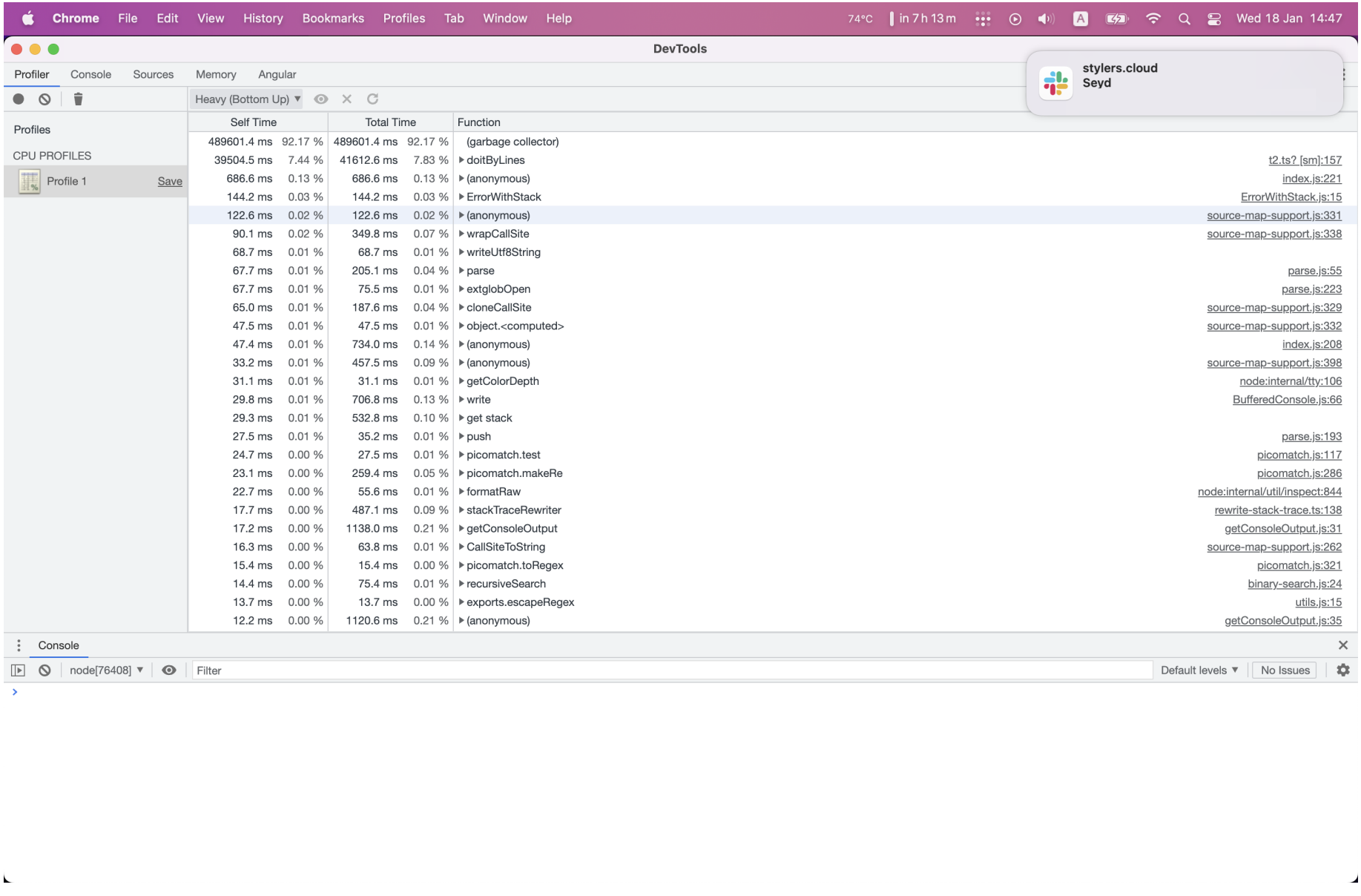
VSCODE DEBUG

demo-jest

- config launch.json
- run debug

3. STARING AT CODE AND THINKING REALLY HARD

PROFILER



Sometimes the bug isn't inside the code

HOW IT WORKS?

- runtime
- debugger "server"
- debugger "client"

DEBUGGING MOBILE

- on mobile
  - allow USB debugging
  - open chrome
- on computer
  - `chrome://inspect`

KEY POINTS

- Debuggers are easy to set up
- Logpoints are temporary console.logs
- Breakpoints help you poke around running app
- `debugger;` works when sourcemaps fail
- Debugger has REPL
- Profilers exist

END

Programmer is happy that he caught the bug