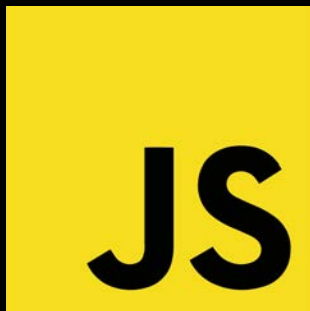


Supercharged production stacktraces with local variables

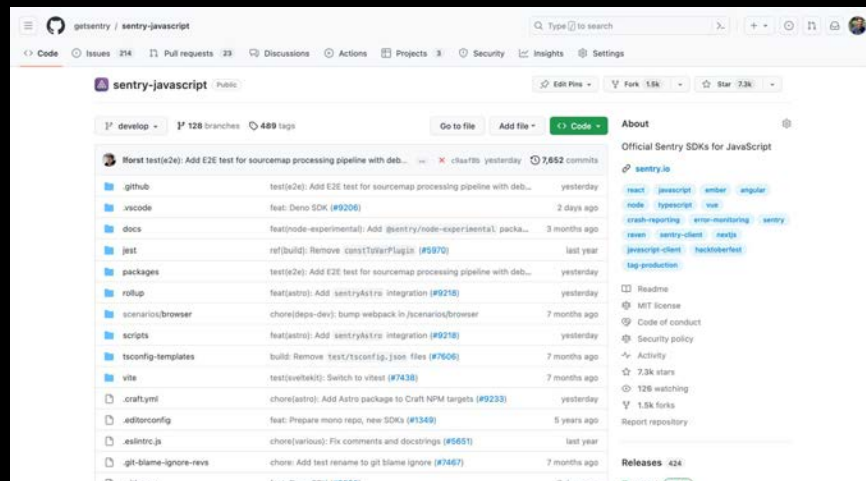
Abhijeet Prasad
Senior Software Engineer @ Sentry



Hey I'm Abhijeet (he/him)

I currently work at Sentry 

I help maintain **Sentry's JavaScript SDKs**





Error

```
at baz (filename.js:10:15)  
at bar (filename.js:6:3)  
at foo (filename.js:2:3)  
at filename.js:13:1
```

Stacktraces are vital for
debugging your JavaScript

At Sentry, we make it
easy to view
production
stacktraces

```
Stack Trace Most Relevant Full Stack Trace 🕒 Newest ⋮  
TypeError  
Cannot set property generateText of #<Object> which has only a getter  
mechanism onuncaughtexception handled false  
JS /Users/abhijeetprasad/sdk-demos/node/node-vercel-ai/src/instrumentation.ts in  
SentryVercelAIInstrumentation.patch at line 36:19 Set up Code Mapping In App ↔  
29 console.log(module);  
30 return module;  
31 }  
32  
33 private patch(moduleExports: any): unknown {  
34   this.patchIsActive = true;  
35  
36   moduleExports.generateText = new Proxy(moduleExports.generateText, {  
37     apply: (target, thisArg, argArray) => {  
38       console.log("generateText called");  
39       return target.apply(thisArg, argArray);  
40     },  
41   });  
42  
43   return moduleExports;  
  
Called from: /node-vercel-ai/node_modules/@opentelemetry/instrumentation/src/platform/node/instrumentation.ts in  
SentryVercelAIInstrumentation.onRequire Show 5 more frames  
  
/Users/abhijeetprasad/sdk-demos/node/node-vercel-ai/src/index.ts in dotenv at line 5:55 In App ↕  
/Users/abhijeetprasad/sdk-demos/node/node-vercel-ai/src/index.ts in Object.? at line 6:127 In App ↕  
  
Called from: node-internal/modules/cjs/loader in Module._compile
```



Error

```
at baz (filename.js:10:15)  
at bar (filename.js:6:3)  
at foo (filename.js:2:3)  
at filename.js:13:1
```

There's still a **vital** piece of
debugging context missing

What arguments or
state caused the
exception to occur?

Enter: Local Variables

The image shows a code editor interface with a dark theme. On the left, there is a 'RUN AND DEBUG' sidebar with a 'No Configuration!' dropdown and a gear icon. Below this, the 'VARIABLES' panel is expanded to show 'Local: run' with the following values: `hash = '3d8c3e934d10fd956c952e0b1b1fb61d'`, `param1 = 'potato'`, and `param2 = 'potato'`. Below these are 'this = global', 'Closure' with `md5 = f (message, options) {`, and 'Global' with various built-in objects like `AbortController`, `AbortSignal`, `atob`, `Blob`, `BroadcastChannel`, `btoa`, `Buffer`, and `ByteLengthQueuingStrategy`.

The main editor area shows two tabs: 'JS index.js' and 'JS test.js'. The 'JS index.js' tab is active and contains the following code:

```
1  const md5 = require("md5");
2
3  function run() {
4      const { param1, param2 } = { param1: "potato", param2: "potato" };
5      const hash = md5(`${param1}${param2}`);
6
7      if (!hash) {
8          throw new Error("An error occurred during hashing");
9      }
10 }
11
12 run();
13
```

A yellow highlight is under the `if (!hash) {` line, and a yellow arrow points to it from the left margin, indicating a debugger breakpoint.

Supported in Sentry via language features for:



```
index.py in make_error at line 28
23     var_obj = {"key": "value"}
24     var_bool = False
25     var_null = None
26     auth = "should-be-scrubbed"
27
28     raise ValueError("foo")
29
30
31     make_error()
```

auth	[Filtered]
large_array	> [10 items]
var_bool	False
var_class	<SomeClass var="value">
var_int	1000
var_null	None
var_obj	{ key: "value" }
var_str	"1000"

```
index.py in <module> at line 31
```


How can we do the
same for Node.js?

support Stack locals #990

[Edit](#)[New issue](#)Closed

grigored opened on Jul 4, 2017

...

Do you want to request a *feature* or report a *bug*?

feature

(If this is a *usage question*, please **do not post it here**—post it on forum.sentry.io instead. If this is not a "feature" or a "bug", or the phrase "How do I...?" applies, then it's probably a usage question.)

What is the **expected behavior**?

According to the sentry docs,

In Python and PHP, Sentry will display the values of local variables at the time of each error.

Why don't we have this in js as well, and preferably also in react native? I think with the `arguments` variable we could log at least the locals of the function generating the exception, which would still be immensely useful.

Create sub-issue



3



benvinegar on Jul 21, 2017

Member

...

Why don't we have this in js as well, and preferably also in react native? I think with the `arguments` variable we could log at least the locals of the function generating the exception, which would still be immensely useful.

Unfortunately this isn't possible today in browser JavaScript. The `arguments` object is lost by the time execution enters `try/catch` or the `onerror` global error handler.

We've long experimented with making this possible, but short of re-writing your code during a compilation step to expose every conceivable variable to Raven.js (e.g. using a babel plugin) – which would make all but trivial applications pretty much unusable – there's not much we can do.



8

Assignees



No one – [Assign yourself](#)

Labels



No labels

Type



No type

Projects



No projects

Milestone



No milestone

Relationships



None yet

Development



[Create a branch](#) for this issue or link a pull request.

Notifications



Unsubscribe

You're receiving notifications because you're subscribed to this thread.



<https://github.com/getsentry/sentry-javascript/issues/990>

Node.js

[About this documentation](#)

[Usage and example](#)

[Assertion testing](#)

[Asynchronous context tracking](#)

[Async hooks](#)

[Buffer](#)

[C++ addons](#)

[C/C++ addons with Node-API](#)

[C++ embedder API](#)

[Child processes](#)

[Cluster](#)

[Command-line options](#)

[Console](#)

[Corepack](#)

[Crypto](#)

[Debugger](#)

[Deprecated APIs](#)

[Diagnostics Channel](#)

[DNS](#)

[Domain](#)

Node.js v23.0.0 | [Table of contents](#) | [Index](#) | [Other versions](#) | [Options](#)

Inspector

#

Stability: 2 - Stable

Source Code: [lib/inspector.js](#)

The `node:inspector` module provides an API for interacting with the V8 inspector.

It can be accessed using:

```
const inspector = require('node:inspector/promises');
```

COPY

CJS ESM

or

```
const inspector = require('node:inspector');
```

COPY

CJS ESM

Promises API

#

Stability: 1 - Experimental

Added in: v19.0.0

Class: `inspector.Session`

#

- Extends: `<EventEmitter>`

The `inspector.Session` is used for dispatching messages to the V8 inspector back-end and receiving message responses and notifications.

```
new inspector.Session()
```

#

<https://nodejs.org/api/inspector.html>

We can use the
Node.js inspector in
production to grab
local variables!



```
async function startInspector(): Promise<void> {  
  const inspector = await import('node:inspector')  
  if (!inspector.url()) {  
    inspector.open(0)  
  }  
}
```



```
function startWorker(options: LocalVariablesWorkerArgs): void {  
  const worker = new Worker(new URL(`data:application/javascript;base64,${base64WorkerScript}`), {  
    workerData: options,  
    // We don't want any Node args to be passed to the worker  
    execArgv: [],  
  });  
  
  process.on('exit', () => {  
    worker.terminate();  
  });  
  
  worker.once('error', (err: Error) => {  
    log('Worker error', err);  
  });  
  
  worker.once('exit', (code: number) => {  
    log('Worker exit', code);  
  });  
  
  // Ensure this thread can't block app exit  
  worker.unref();  
}
```



```
import { Session } from 'node:inspector/promises';

async function startDebugger(): Promise<void> {
  const session = new Session();
  session.connectToMainThread();

  // more code
}
```




```
async function getLocalVariables(session: Session, objectId: string): Promise<Variables> {
  const properties: Runtime.GetPropertiesReturnType = await session.post('Runtime.getProperties', {
    objectId,
    ownProperties: true,
  });
  const variables = {};

  for (const prop of properties.result) {
    if (prop?.value?.objectId && prop?.value.className === 'Array') {
      const id = prop.value.objectId;
      await unrollArray(session, id, prop.name, variables);
    } else if (prop?.value?.objectId && prop?.value?.className === 'Object') {
      const id = prop.value.objectId;
      await unrollObject(session, id, prop.name, variables);
    } else if (prop?.value) {
      unrollOther(prop, variables);
    }
  }

  return variables;
}
```



```
function addLocalVariablesToEvent(event: Event, hint: EventHint): Event {
  if (
    hint.originalException &&
    typeof hint.originalException === 'object' &&
    LOCAL_VARIABLES_KEY in hint.originalException &&
    Array.isArray(hint.originalException[LOCAL_VARIABLES_KEY])
  ) {
    for (const exception of event.exception?.values || []) {
      addLocalVariablesToException(exception, hint.originalException[LOCAL_VARIABLES_KEY]);
    }

    hint.originalException[LOCAL_VARIABLES_KEY] = undefined;
  }

  return event;
}
```



```
function addLocalVariablesToEvent(event: Event, hint: EventHint): Event {
  if (
    hint.originalException &&
    typeof hint.originalException === 'object' &&
    LOCAL_VARIABLES_KEY in hint.originalException &&
    Array.isArray(hint.originalException[LOCAL_VARIABLES_KEY])
  ) {
    for (const exception of event.exception?.values || []) {
      addLocalVariablesToException(exception, hint.originalException[LOCAL_VARIABLES_KEY]);
    }

    hint.originalException[LOCAL_VARIABLES_KEY] = undefined;
  }

  return event;
}
```



```
function addLocalVariablesToEvent(event: Event, hint: EventHint): Event {
  if (
    hint.originalException &&
    typeof hint.originalException === 'object' &&
    LOCAL_VARIABLES_KEY in hint.originalException &&
    Array.isArray(hint.originalException[LOCAL_VARIABLES_KEY])
  ) {
    for (const exception of event.exception?.values || []) {
      addLocalVariablesToException(exception, hint.originalException[LOCAL_VARIABLES_KEY]);
    }

    hint.originalException[LOCAL_VARIABLES_KEY] = undefined;
  }

  return event;
}
```



```
Sentry.init({  
  includeLocalVariables: true,  
});
```

JS

/Users/abhijeetprasad/sdk-demos/node-express/esm-preload-example/index.mjs at line 32:9

Unminify Code

In App



```
25   const varBool = false;
26   const varClass = new Test();
27   const varNumber = 123;
28   const varNull = null;
29   const varObj = { key: "value" };
30   const varString = "string";
31
32   throw new Error("This is an error");
33 });
34
35 app.get("/http-req", function (req, res) {
36   http
37     .request("http://example.com", (httpRes) => {
38     let data = "";
39     httpRes.on("data", (d) => {
```

auth [Filtered]

largeArray > [10 items]

req <IncomingMessage>

res <ServerResponse>

varBool false

varClass <Test>

varNull null

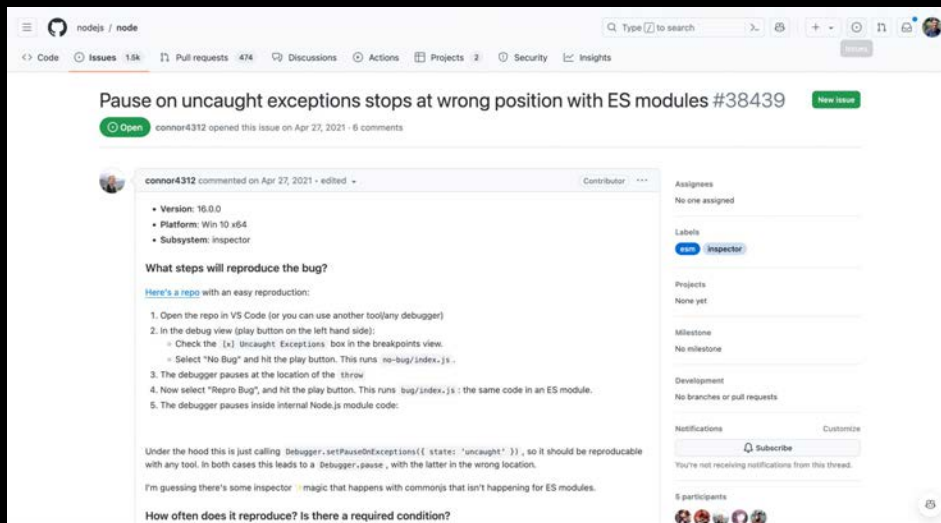
varNumber 123

varObj {
 key: value
}

varString string

Challenges

- Worker overhead
- Memory pressure
- Unhandled exceptions
- ESM vs. CJS
- Minified Variables
- Incompatibility with Debuggers
- Only Node.js support



<https://github.com/nodejs/node/issues/38439>

 Thank you!

Twitter: <https://twitter.com/imabhiprasad>

Bluesky:

<https://bsky.app/profile/abhiprasad.bsky.social>

GitHub: <https://github.com/abhiprasad>

Open Source JavaScript SDKs:

<https://github.com/getsentry/sentry-javascript>

Try Sentry



conf42js2024