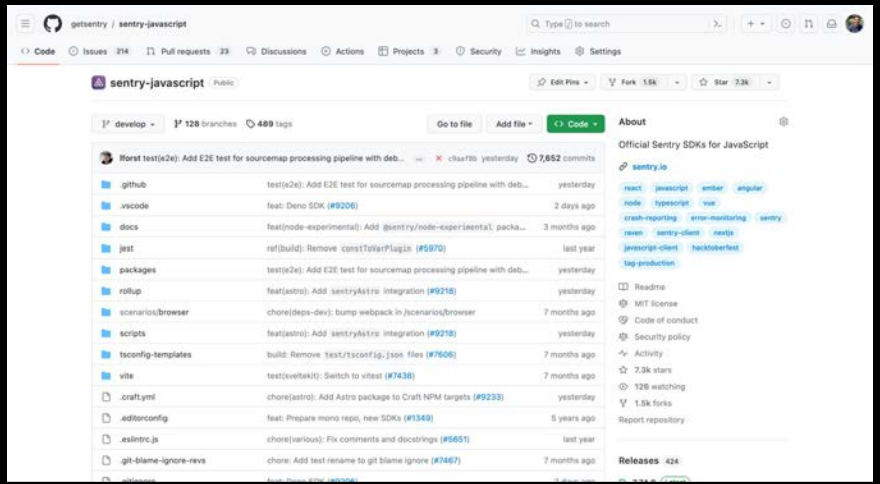
Supercharged production stacktraces with local variables

Abhijeet Prasad
Senior Software Engineer @ Sentry

Hey I’m Abhijeet (he/him)

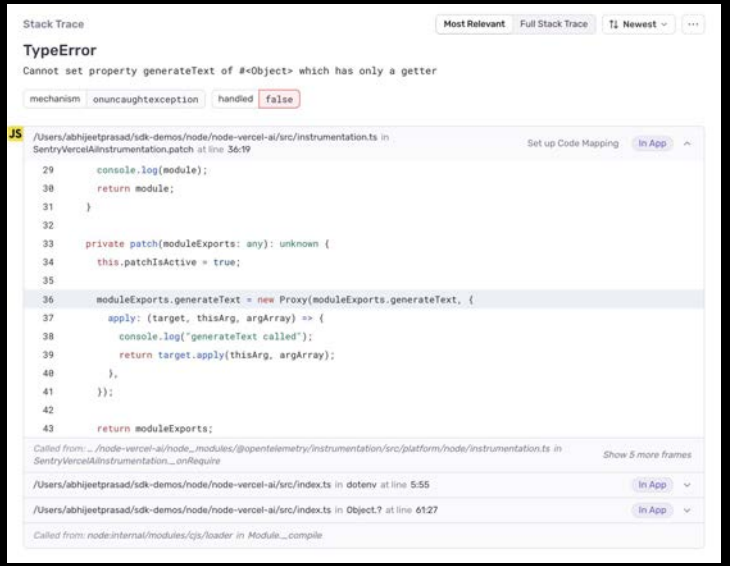
I currently work at Sentry

I help maintain Sentry’s JavaScript SDKs



Stacktraces are vital for debugging your JavaScript

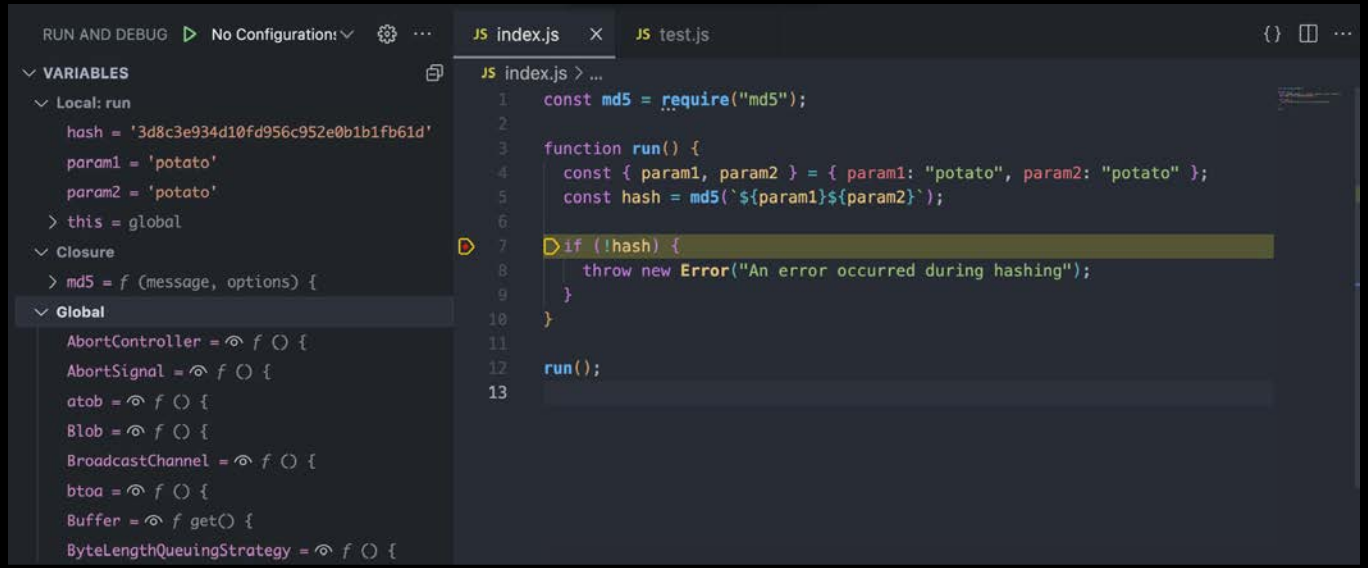
At Sentry, we make it easy to view production stacktraces



There’s still a vital piece of debugging context missing

What arguments or state caused the exception to occur?

Enter: Local Variables



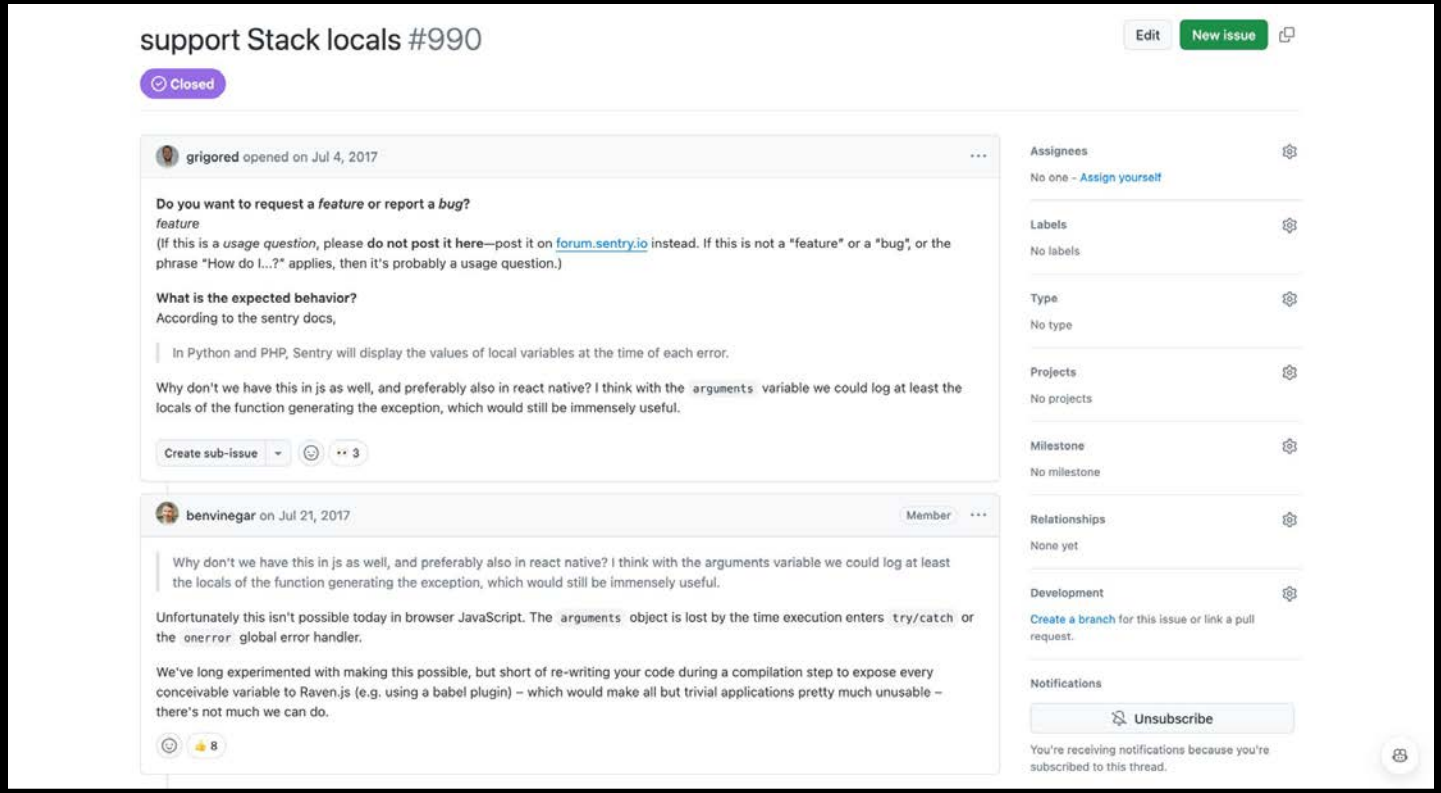
Supported in Sentry via language features for:

Python

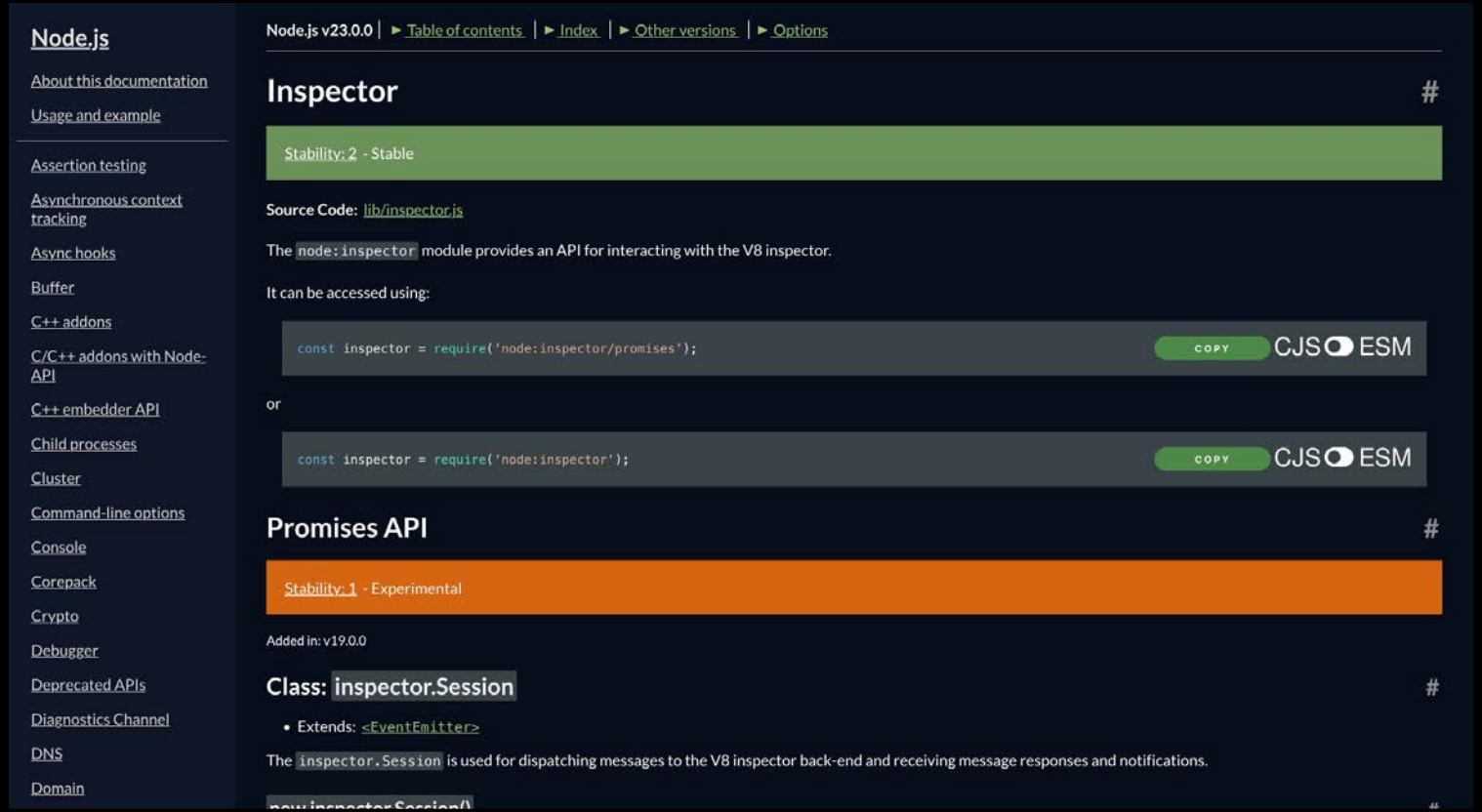
PHP

Ruby

How can we do the same for Node.js?



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



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

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

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

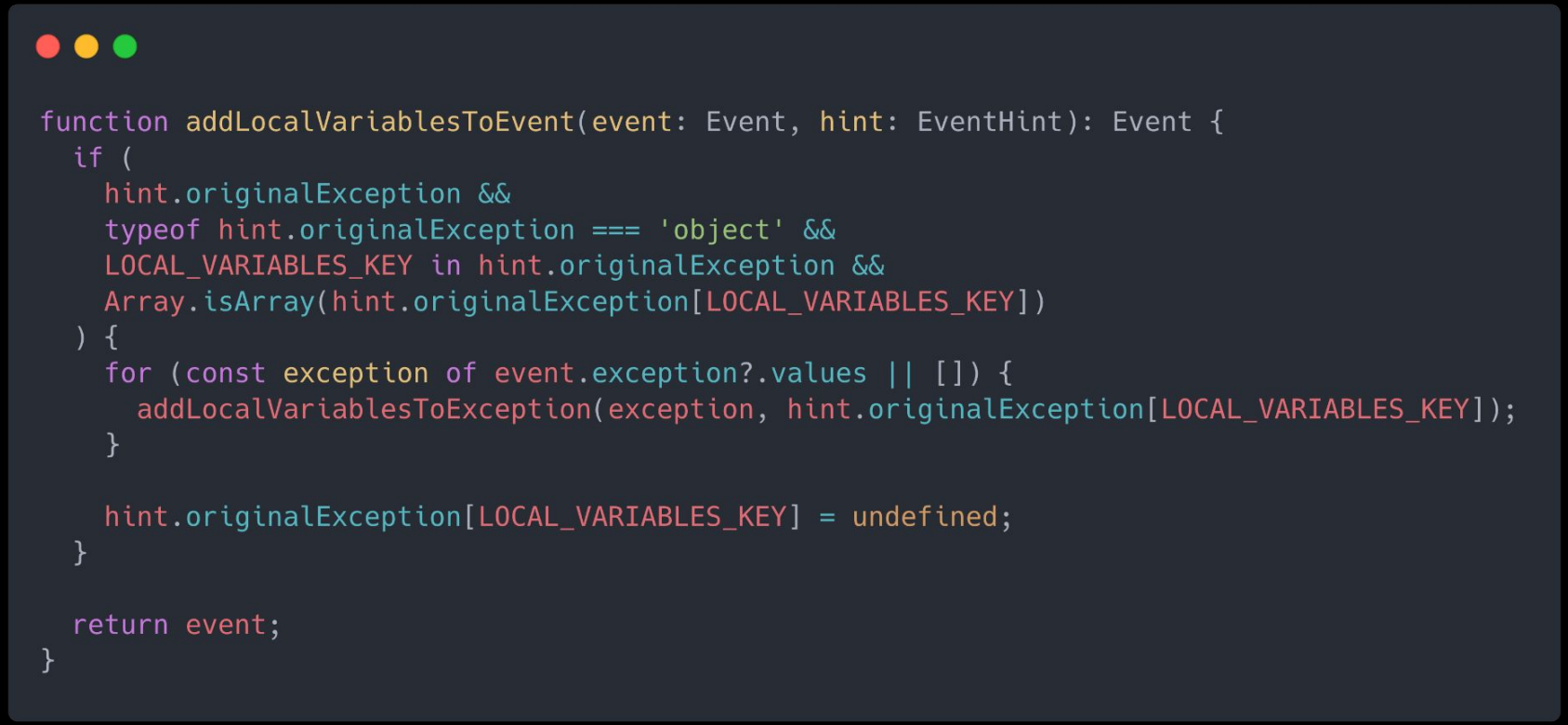
```ts
function startWorker(options: LocalVariablesWorkerArgs): void {
  const worker = new Worker(new URL(`data:application/javascript;base64,${base64WorkerScript}`), {
    workerData: options,
    // We don't want any Node args to be passed to the worker
    execArgv: [],
  });

  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
}

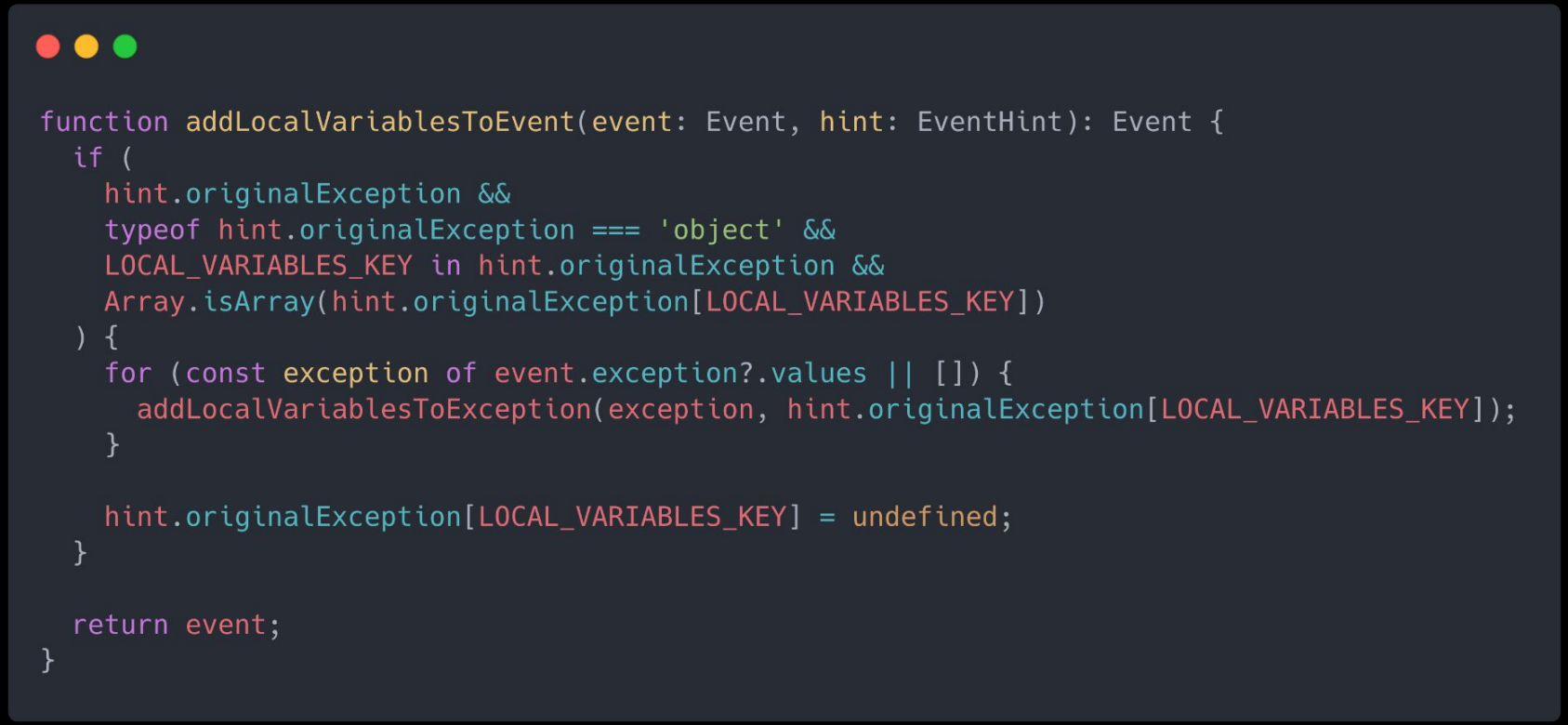
```
session.on('Debugger.resumed', () => {
  isPaused = false;
});

session.on('Debugger.paused', (event: InspectorNotification<Debugger.PausedEventDataType>) => {
  isPaused = true;

  handlePaused(session, event.params as PausedExceptionEvent).then(
    async () => {
      // After the pause work is complete, resume execution!
      if (isPaused) {
        await session.post('Debugger.resume');
      }
    },
    async _ => {
      if (isPaused) {
        await session.post('Debugger.resume');
      }
    },
  );
});

await session.post('Debugger.enable');

const captureAll = options.captureAllExceptions !== false;
await session.post('Debugger.setPauseOnExceptions', { state: captureAll ? 'all' : 'uncaught' });
```

```
async function handlePaused(
  session: Session,
  { reason, data: { objectId }, callFrames }: PausedExceptionEvent,
): Promise<void> {
  if (isInvalidReason(reason)) {
    return;
  }

  invokeRateLimiter();

  const frames = [];

  // Logic simplified
  for (let i = 0; i < callFrames.length; i++) {
    const frame = callFrames[i];

    const localScope = getLocalScope(frame);

    const fn = getFunctionName(frame)
    cost vars = await getLocalVariables(session, localScope.object.objectId);
    frames[i] = { function: fn, vars };
  }

  await session.post('Runtime.callFunctionOn', {
    functionDeclaration: `function() { this.${LOCAL_VARIABLES_KEY} = this.${LOCAL_VARIABLES_KEY} ||
${JSON.stringify(
      frames,
    )}; }`,
    silent: true,
    objectId,
  });

  await session.post('Runtime.releaseObject', { objectId });
}
```

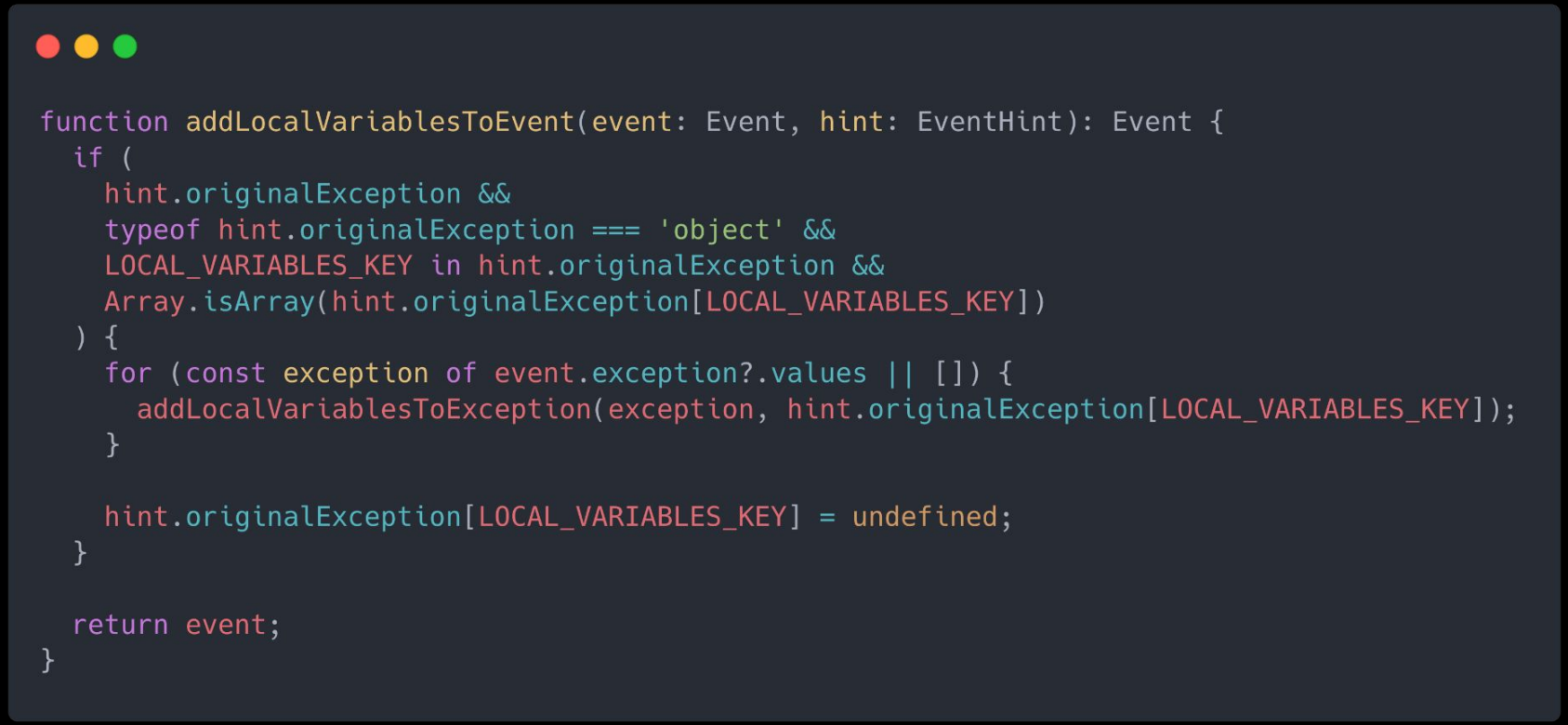
```
async function getLocalVariables(session: Session, objectId: string): Promise<Variables> {
  const properties: Runtime.GetPropertiesReturnType = await session.post('Runtime.getProperties', {
    objectId,
    ownProperties: true,
  });
  const variables = {};

  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,
});
```

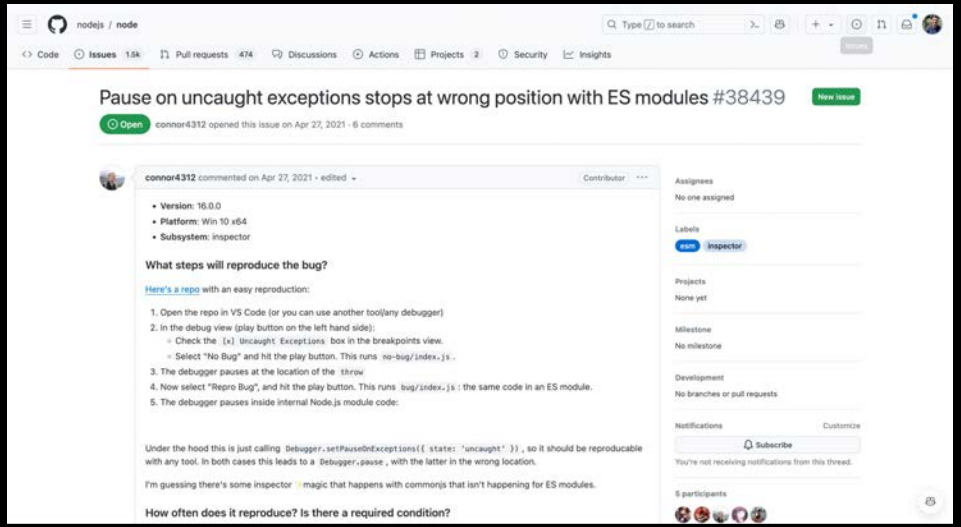
JS /Users/abhijeetprasad/sdk-demos/node-express/esm-preload-example/index.mjs at line 32:9 Unminify Code In App

```
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!

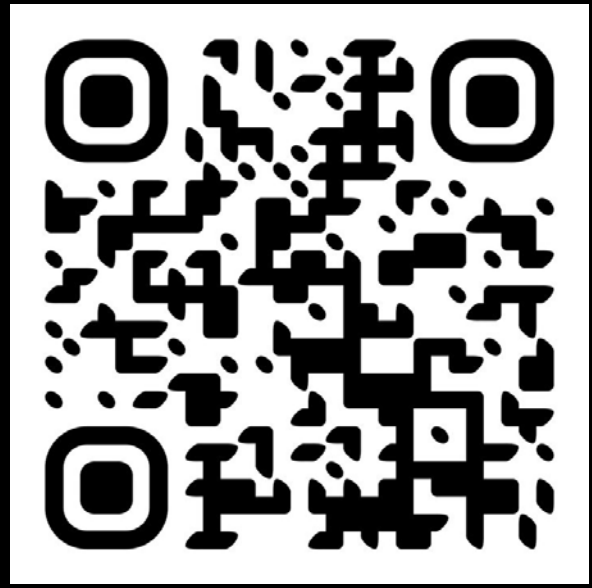
Twitter: https://twitter.com/imabhiprasad

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