Free email newsletter: “ES.next News”

2016-04-12

Tracking unhandled rejected Promises

Labels: async, dev, esnext, javascript, promises

In Promise-based asynchronous code, rejections are used for error handling. One risk is that rejections may get lost, leading to silent failures. For example:

```
function main() {
    asyncFunc()
    .then(···)
    .then(() => console.log('Done!'));
}
```

If `asyncFunc()` rejects the Promise it returns then that rejection will never be handled anywhere.

Let’s look at how you can track unhandled rejections in browsers and in Node.js.

1. Unhandled rejections in browsers

Some browsers (only Chrome at the moment) report unhandled rejections.

1.1. `unhandledrejection`

Before a rejection is reported, an event is dispatched that you can listen to:

```
window.addEventListener('unhandledrejection', event => ···);
```

The event is an instance of `PromiseRejectionEvent` whose two most important properties are:

- `promise`: the Promise that was rejected
- `reason`: the value with which the Promise was rejected

The following example demonstrates how this event works:

```
window.addEventListener('unhandledrejection', event => {
    // Prevent error output on the console:
    event.preventDefault();
    console.log('Reason: ' + event.reason);
});

function foo() {
    Promise.reject('abc');
}
foo();
```

The output of this code is:

```
Reason: abc
```

1.2. `rejectionhandled`

If a rejection is initially unhandled, but taken care of later then `rejectionhandled` is dispatched. You listen to it as follows:

Most popular (last 30 days)

ECMAScript 2017: the final feature set
ECMAScript 6 modules: the final syntax
ES proposal: import() – dynamically importing ES modules
Making transpiled ES modules more spec-compliant
ES proposal: Shared memory and atomics
Classes in ECMAScript 6 (final semantics)
Communicating between Web Workers via MessageChannel

Most popular (all time)

ECMAScript 6 modules: the final syntax
Classes in ECMAScript 6 (final semantics)
Iterating over arrays and objects in JavaScript
ECMAScript 6’s new array methods
The final feature set of ECMAScript 2016 (ES7)
WebAssembly: a binary format for the web
Basic JavaScript for the impatient programmer
Google Dart to “ultimately ... replace JavaScript”
Six nifty ES6 tricks
Google’s Polymer and the future of web UI frameworks

Blog archive

► 2017 (4)
▼ 2016 (38)

Dr. Axel Rauschmayer

```
window.addEventListener('rejectionhandled', event => ···);
```

event is also an instance of `PromiseRejectionEvent`.

The following code demonstrates `rejectionhandled`:

```
window.addEventListener('unhandledrejection', event => {
    // Prevent error output on the console:
    event.preventDefault();
    console.log('Reason: ' + event.reason);
});
window.addEventListener('rejectionhandled', event => {
    console.log('REJECTIONHANDLED');
});


function foo() {
    return Promise.reject('abc');
}
var r = foo();
setTimeout(() => {
    r.catch(e => {});
}, 0);
```

This code outputs:

```
Reason: abc
REJECTIONHANDLED
```

1.3. Further reading

The Chrome Platform Status site links to a "Promise Rejection Events Sample" that contains an explanation and code.

2. Unhandled rejections in Node.js

Node.js does not report unhandled rejections, but it emits events for them. You can register an event listener like this:

```
process.on('unhandledRejection', (reason, promise) => ···);
```

The following code demonstrates how the event works:

```
process.on('unhandledRejection', (reason) => {
    console.log('Reason: ' + reason);
});
function foo() {
    Promise.reject('abc');
}
foo();
```

Note: Node.js v6.6.0+ reports unhandled rejections by default – you don't need to register an event listener, anymore. Hat tip: @PetrHurtak

2.1. Further reading

The Node.js documentation has more information on the Event `unhandledRejection`.

13 Comments   The 2ality blog   1 Login

Recommend 3   Share   Sort by Best

Join the discussion…

Alexander Biryukov • 10 months ago

Great thing! Don't know about it. Have added to the project. Thanks.

- December (1)
- November (4)
- October (4)
- September (5)
- August (1)
- June (1)
- May (2)
- April (2)
  - Trees of Promises in ES6
  - Tracking unhandled rejected Promises
- March (2)
- February (8)
- January (8)
- 2015 (65)
- 2014 (55)
- 2013 (98)
- 2012 (177)
- 2011 (380)
- 2010 (174)
- 2009 (68)
- 2008 (46)
- 2007 (12)
- 2006 (1)
- 2005 (2)

Labels

dev (592)
javascript (400)
computers (316)
life (194)
jslang (179)
esnext (156)
apple (107)
webdev (95)
mobile (83)
scitech (50)
hack (49)
mac (47)
google (39)
java (37)
ios (33)
business (32)
video (32)
clientjs (31)
hci (27)
entertainment (26)
nodejs (26)
society (26)
browser (25)

Tweets by @rauschma

Axel Rauschm… @rauschma

Enjoying "Troll Hunters":
– "Let's call him 'Gnome Chomsky'"
– "Juliet dies in this? Nooo!"

9h

Axel Rauschmayer Retweeted

Jonathan Creamer @jcreamer898

A nice little shout out to @rauschma... infoworld.com/article/316483 … #es2017 #async

JavaScr… ECMASc… infoworld…

13h

Axel Rauschmayer @rauschma

Not a physics book! twitter.com/ManningBooks/s …

02 Feb

Axel Rauschmayer Retweeted

Jordan Harband @ljharb

Making our React components forbid extra props has caught SO many bugs. I highly recommend it.

npmjs.com/airbnb-prop-ty…

npm: air… Custom … npmjs.com

02 Feb

Axel Rauschmayer Retweeted

Seth Petry-Johnson @spetryjohnson

Open strong. Be bold. Tell a story. Do something to get me interested. Opening w/ the "obligatory 'about me' slide" puts me to sleep :)

02 Feb

Google

firefox (25)
html5 (24)
ipad (24)
movie (23)
psychology (22)
2ality (18)
tv (18)
android (17)
social (17)
chrome (16)
fun (16)
jsmodules (16)
tablet (16)
humor (15)
politics (15)
web (15)
cloud (14)
hardware (14)
microsoft (14)
software engineering (13)
blogging (12)
gaming (12)
eclipse (11)
gwt (11)
numbers (11)
programming languages (11)
app store (10)
media (10)
nature (10)
security (10)
semantic web (10)
software (10)
twitter (10)
webos (10)
12quirks (9)
education (9)
jstools (9)
photo (9)
webcomponents (9)
windows 8 (9)
async (8)
idea (8)
iphone (8)
itunes (8)
scifi-fantasy (8)
app (7)
babel (7)
bookmarklet (7)
chromeos (7)
english (7)
es proposal (7)
fringe (7)
html (7)

3 ^ | v • Reply • Share ›

Benjamin Gruenbaum • 10 months ago

These tools are invaluable and in my opinion everyone using promises should be acquainted with them. They are also supported in Node by most userland promise libraries like Q, bluebird, When and the core-js shim. In browsers, Bluebird and native promises are the only two variants that supports these events (the native events take their names and behavior from bluebird).

It's worth mentioning that browsers show a console error by default and not doesn't. Which is why it is very important to install a `unhandledRejection` handler. There is discussion about changing this default in https://github.com/nodejs/prom... .

Here is a talk I gave last year that discusses adding these events to NodeJS https://www.youtube.com/watch?... for those who want some historical context.

3 ^ | v • Reply • Share ›

m3l • 15 days ago

I'm quite confused on this.
Node default behaviour is to report a warning which hasn't even a trace of the error.

An unhandled rejection is an async version of throw.
Shouldn't it be logged with trace and cause the node app to crash?

^ | v • Reply • Share ›

Marcos Cáceres • 4 months ago

I'm worried that you don't reject with a `new Error()` in the example. I'm worried people will copy/paste your example and start writing code that rejects with strings (which will equate to much sadness in practice because no stack).

^ | v • Reply • Share ›

Mark Volkmann • 4 months ago

A minor thing perhaps, but I recommend using console.error instead of console.log to log error messages.

^ | v • Reply • Share ›

> **Dmitriy Ievlev** ➦ Mark Volkmann • a month ago
>
> Moreover, I would recommend using comma instead of string concatenation:
> console.error('Reason: ', reason)
> In this case you are getting the stack trace along with the error message.
>
> ^ | v • Reply • Share ›
>
> **Axel Rauschmayer** Mod ➦ Mark Volkmann • 3 months ago
>
> Very good point! I've started to switch.
>
> ^ | v • Reply • Share ›

Bruno Scopelliti • 10 months ago

Didn't know about unhandledrejection/rejectionhandled events...

@Axel Rauschmayer do you have some real example where using such events is preferable in respect to attaching a `catch` on the promise?

^ | v • Reply • Share ›

> **Aleksandrs Ulme** ➦ Bruno Scopelliti • 10 months ago
>
> I believe the intention here is to check for missing `catch`es rather than replace them.
>
> 3 ^ | v • Reply • Share ›

jsint (7)
jsshell (7)
thunderbolt (7)
webapp (7)
advancedjs (6)
blogger (6)
crowdsourcing (6)
latex (6)
lion (6)
promises (6)
ted (6)
book (5)
environment (5)
gadget (5)
googleio (5)
intel (5)
jsarrays (5)
jshistory (5)
layout (5)
light peak (5)
michael j. fox (5)
music (5)
pdf (5)
polymer (5)
shell (5)
tc39 (5)
template literals (5)
underscorejs (5)
vlc (5)
__proto__ (4)
coffeescript (4)
concurrency (4)
dart (4)
facebook (4)
gimp (4)
googleplus (4)
health (4)
howto (4)
hp (4)
javafx (4)
kindle (4)
leopard (4)
macbook (4)
motorola (4)
münchen (4)
occupy (4)
pl fundamentals (4)
presenting (4)
publishing (4)
series (4)
textbook (4)
web design (4)
amazon (3)
asmjs (3)

Axel Rauschmayer Mod ➦ Aleksandrs Ulme • 10 months ago

Yes!

1 ^ | ˅ • Reply • Share ›

Barney Carroll ➦ Bruno Scopelliti • 10 months ago

Reading between the lines of Axel's recent postings here and on Twitter, I think this is a harness for debugging — not for application development. Is that fair to say Axel?

This 'catch' is fundamentally different in scope and purpose to the Promise catch method: the intended purpose and error recovery strategy for any given Promise is ultimately the responsibility of the calling context. Thus you might write an interface for a 3rd party API and decide that you need to be able to fallback from remote server errors. In theory, Promises can serve as an abstract interface for all manner of remote processes where the calling context may depend upon catching failures and taking measures. But in practice, errors aren't expected and an application full of catches is an oddity — writing catches for every promise called in an application is a bit of an anti pattern because it belies an expectancy of failure!

Having said that, mistakes do happen, and there's a temptation to write catch or finally on every promise you have simply to be able to get a lock on where async processes are failing you. But that's a mess, because you end up having to write a significant volume of flow control code — worse, being comprehensive in this approach often requires significant changes to application logic itself (in the case of chained promises, a catch or finally breaks the chain).

This API allows a programmatic way of logging and debugging promise rejections without touching the fundamental application logic.

1 ^ | ˅ • Reply • Share ›

dSebastien ➦ Barney Carroll • 9 months ago

This is great for frameworks trying to avoid bad UX due to sloppy programming :)

^ | ˅ • Reply • Share ›

Benjamin Gruenbaum ➦ Barney Carroll • 10 months ago

Precisely. It is mainly for debugging purposes and to make sure no errors slip. Like Node's `uncaughtException` or `window.addEventListener("error"`. It also ensures errors are never "swallowed".

^ | ˅ • Reply • Share ›

Subscribe **Add Disqus to your site Add Disqus Add** **Privacy**

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

back to the future (3)

bitwise_ops (3)

css (3)

es2016 (3)

flattr (3)

fluentconf (3)

food (3)

foreign languages (3)

house (3)

icloud (3)

info mgmt (3)

jsfuture (3)

jsstyle (3)

linux (3)

mozilla (3)

python (3)

regexp (3)

samsung (3)

tizen (3)

traffic (3)

typedjs (3)

unix (3)

adobe (2)

angry birds (2)

angularjs (2)

astronomy (2)

audio (2)

comic (2)

design (2)

dom (2)

ecommerce (2)

eval (2)

exploring es6 (2)

facebook flow (2)

facets (2)

flash (2)

free (2)

futurama (2)

guide (2)

history (2)

hyena (2)

internet explorer (2)

iteration (2)

journalism (2)

jquery (2)

jsengine (2)

jslib (2)

law (2)

lightning (2)

markdown (2)

math (2)

meego (2)

month (2)

nike (2)

nokia (2)

npm (2)

programming (2)

raffle (2)

repl (2)

servo (2)

sponsor (2)

steve jobs (2)

travel (2)

typescript (2)

usb (2)

winphone (2)

wwdc (2)

airbender (1)

amdefine (1)

aol (1)

app urls (1)

architecture (1)

atscript (1)

basic income (1)

biology (1)

blink (1)

bluetooth (1)

canada (1)

clip (1)

coding (1)

community (1)

cross-platform (1)

deutsch (1)

diaspora (1)

distributed-social-network (1)

dsl (1)

dvd (1)

dzone (1)

emacs (1)

emberjs (1)

energy (1)

esnext news (1)

esprop (1)

example (1)

facetator (1)

feedback (1)

firefly (1)

firefoxos (1)

fritzbox (1)

german (1)

git (1)

guest (1)

guice (1)

h.264 (1)

home entertainment (1)

hosting (1)

htc (1)

ical (1)

jsdom (1)
jsmyth (1)
library (1)
location (1)
marketing (1)
mars (1)
meta-data (1)
middle east (1)
mpaa (1)
msl (1)
mssurface (1)
netflix (1)
nsa (1)
obama (1)
openoffice (1)
opinion (1)
oracle (1)
organizing (1)
philosophy (1)
pixar (1)
pnacl (1)
prism (1)
privacy (1)
proxies (1)
puzzle (1)
raspberry pi (1)
read (1)
rodney (1)
rust (1)
safari (1)
sponsoring (1)
star trek (1)
static generation (1)
talk (1)
technique (1)
theora (1)
thunderbird (1)
typography (1)
unicode (1)
v8 (1)
voice control (1)
webassembly (1)
webkit (1)
webm (1)
webpack (1)
yahoo (1)