What's new with JavaScript in GNOME

The 2020 edition

Philip Chimento • pchimento • ptomato • @therealptomato

GUADEC Online, July 25, 2020

Introduction: What we will talk about

- Same thing we talk about every year!
- **What's new** in GJS?
  - Including a bit about a lesser-known corner: `Intl`
- **What's next** in GJS?
- We need some **help!** Here's what you can do

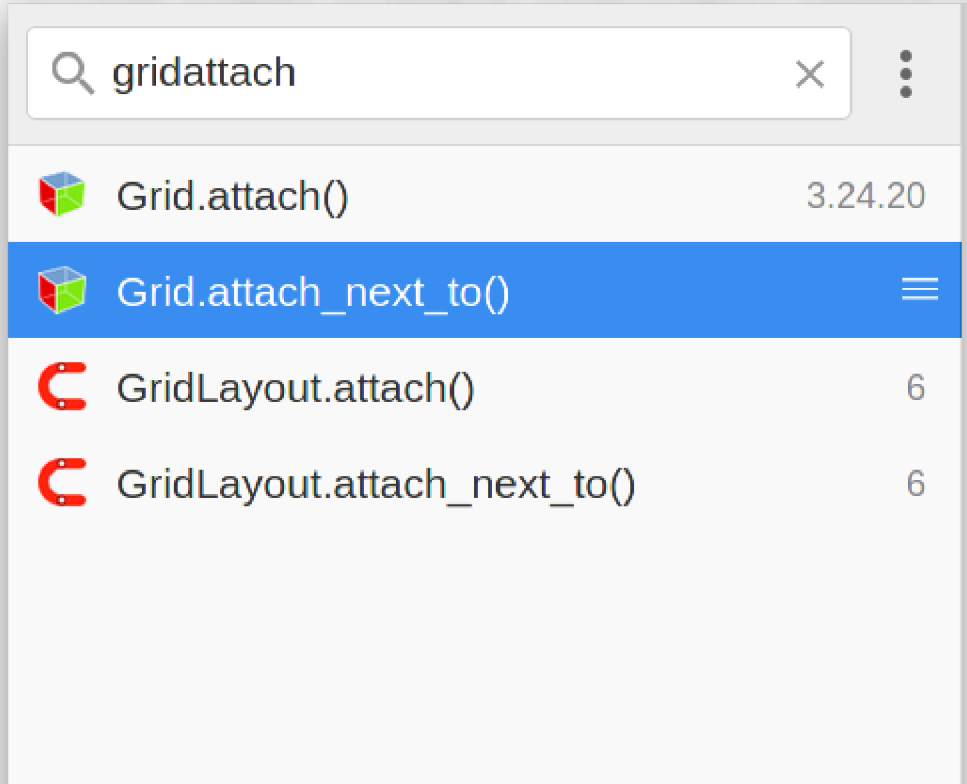
Introduction

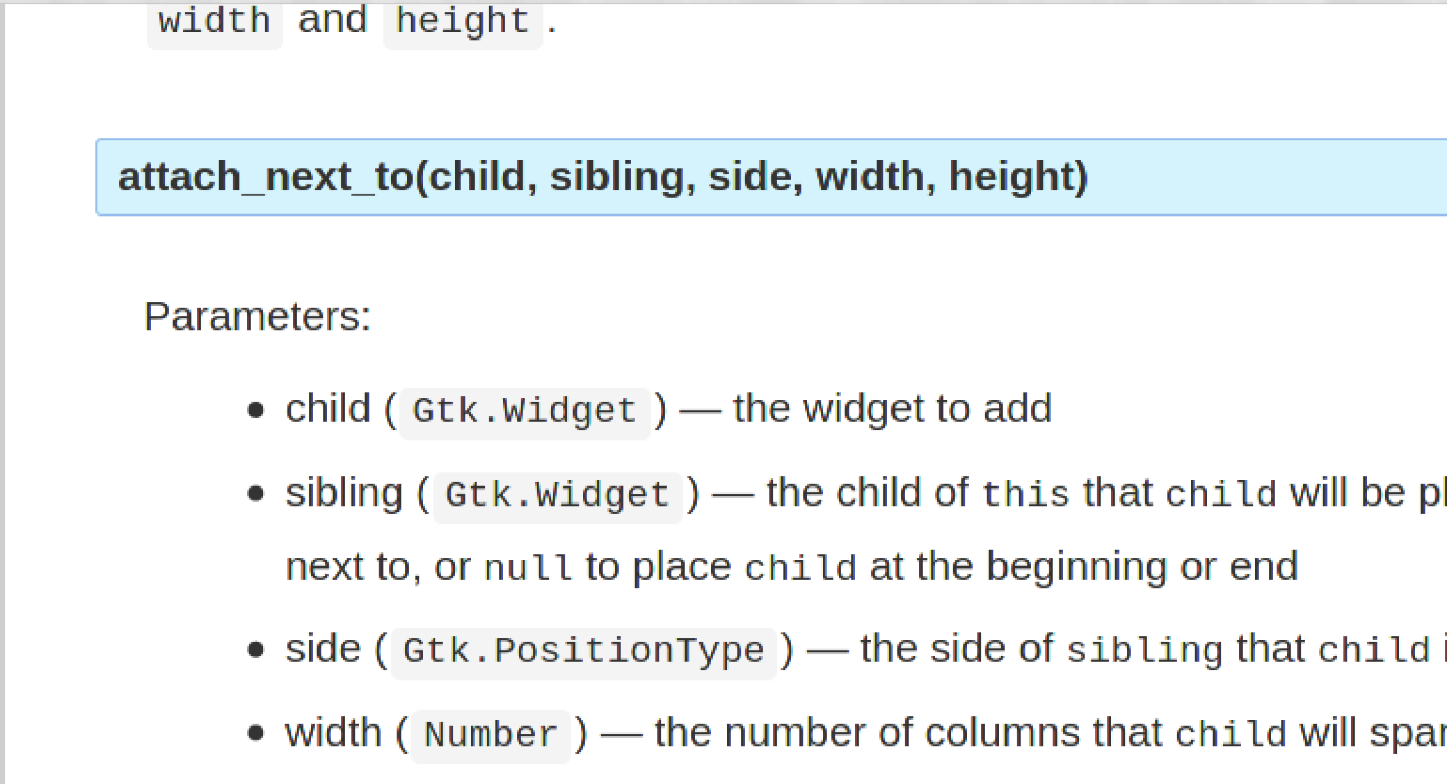
- 🗣 **Shout-outs** are a thank-you to someone who contributed a feature
- Presentation is full of links if you want to click through later
- If you want to follow along: **ptomato.name/guadec2020**

What's new in JavaScript land for GNOME 3.36 and 3.38?

Developer experience

- gjs-docs is now way easier to maintain and contribute to 🗣 **Meg Ford**
- Documentation added for GNOME Shell internals 🗣 **Andy Holmes**





Crash fixes

It shouldn't be possible to crash GNOME Shell just by writing JS code.

Crash fixes

- 9 crashing bugs still open
- 6 fixed, 5 new since last GUADEC 📈
- Improvements to type safety 🗣 **Marco Trevisan**

Performance fixes

- Looking up properties that may come from C:
  - e.g. `myWindow.present()`
  - Don't look up `myWindow.customProperty`
  - Negative lookup cache 🗣 **Daniel Van Vugt**

Performance fixes

- Calling C functions
  - Translating arguments is slow
  - e.g. `void gtk_c_func(int arg1, double arg2)` → `Gtk.c_func(number, number)`
  - Argument cache 🗣 **Giovanni Campagna**

GObject properties

- Adding GObject properties to your class was always really verbose
- Easy to do notifications incorrectly, for example
- Now you can leave out the boring part:

```
// Just delete all this!
get my_property() {
  return this._myProperty;
}
set my_property(value) {
  if (this._myProperty === value)
    return;
  this._myProperty = value;
  this.notify('my-property');
}
```

New edition of JS language

- JavaScript is gaining features every year
- No longer the language that everyone loved to hate
- Which of these new features can you use in GJS?

New JS language features in 3.36

- `String.trimStart()`, `String.trimEnd()`
- `String.matchAll()`
- `Array.flat()`
- `Array.flatMap()`
- `Object.fromEntries()`
- `globalThis`
- `BigInt`, `BigInt64Array`, `BigUint64Array`
- String generics removed
- New `Intl` features

String generics removed

- Nonstandard way of calling String methods on a non-string object
- Use moz68tool to scan your code for them
- Don't use array generics either, Mozilla will remove them soon

👎

```
let num = 15;
String.endsWith(num, 5);
```

👍

```
String.prototype.endsWith.call(num, 5);
String(num).endsWith(5);
`${num}`.endsWith(5);
// ⇒ true
```

New `Intl` features

`Intl.RelativeTimeFormat`

- Useful for user interfaces

```
rtf = new Intl.RelativeTimeFormat('en',
  {style: 'narrow'})
rtf.format(-1, 'day')
// ⇒ "1 day ago"

rtf = new Intl.RelativeTimeFormat('es',
  {numeric: 'auto'})
rtf.format(2, 'day')
// ⇒ "pasado mañana"
```

A small digression about `Intl`

`Intl`

- A global object
- Does internationalization stuff for you
- Some things awkward to do with `gettext`. `Intl` makes them easy
- Some things `gettext` does well. `Intl` mostly doesn't do those

`Intl`

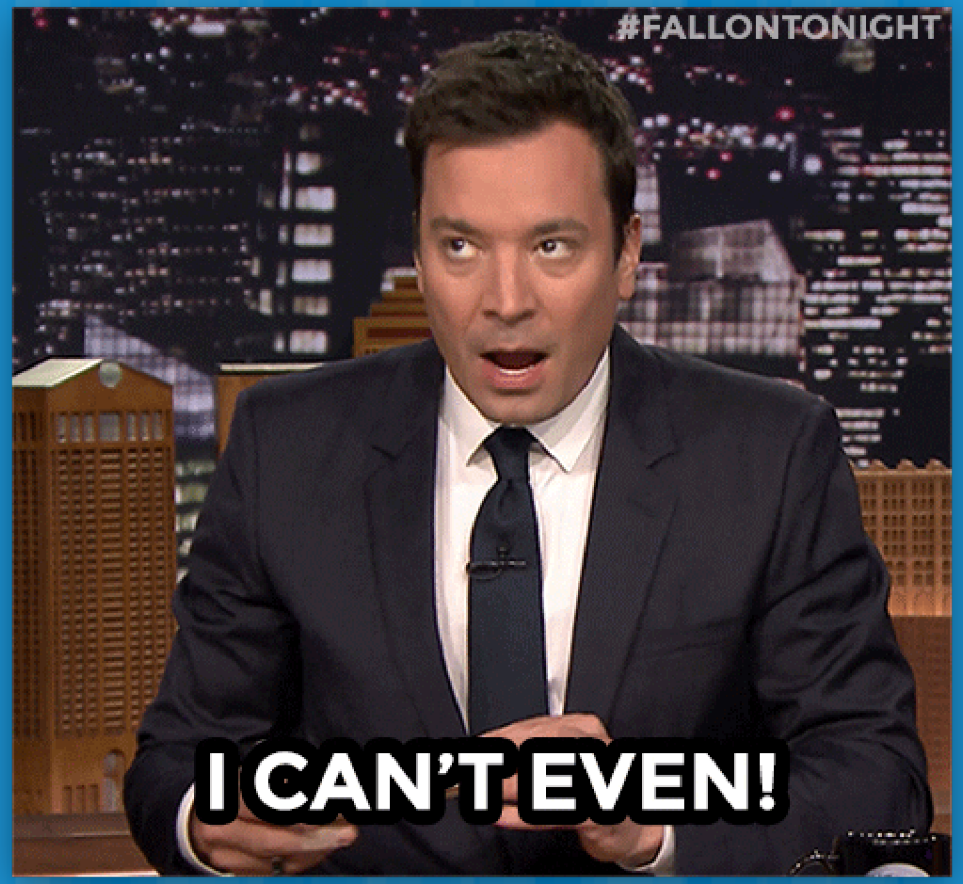
- `Intl.Collator` - locale-sensitive sorting
- `Intl.DateTimeFormat` - printing dates and times (`"Fri Jul 17"`)
- `Intl.NumberFormat` - printing numbers (`"€ 0,00"`)
- `Intl.RelativeTimeFormat` - printing relative times (`"3 days ago"`)
- More in future editions of JS!

Why you should use `Intl`

```
/* Translators: this is the month name and day number
 followed by a time string in 24h format.
 i.e. "May 25, 14:30" */
format = N_("%B %-d, %H\u2236%M");
```

- How does the translator know what to put there for their locale?

“Well why don't they just read the manual for strftime?”


#FALLONTONIGHT
I CAN'T EVEN!

Why you should use `Intl`

```
#. Translators: this is the month name and day number
#. followed by a time string in 24h format.
#. i.e. "May 25, 14:30"
#: js/misc/util.js:255
#, no-c-format
msgid "%B %-d, %H:%M"
msgstr "%j, %R %p"  😱😱😱
```

Why you should use `Intl`

- More readable, even if longer
- Less burden on GNOME translators

```
format = new Intl.DateTimeFormat(myLocale, {
  month: 'short',
  day: 'numeric',
  hour: '2-digit',
  minute: '2-digit',
  hourCycle: 'h24',
})
format.format(Date.now())
// ⇒ "Jul. 17, 18:01" in my locale
```

Why you should use `Intl`

- Who *does* translate those, anyway?
  - ICU project
  - Unicode Common Locale Data Repository (CLDR)
  - They have the power of major browser vendors behind them

Why you should use `Intl`

- What if I don't like the ICU/CLDR's translations in my UI?
  - You can still customize them using `formatToParts()`
  - Supported by most `Intl` formatters

Why you should use `Intl`

- What if I don't like the ICU/CLDR's translations in my UI?
  - You can still customize them using `formatToParts()`
  - Supported by most `Intl` formatters

12:00 12:00

What's next in GJS?

What's upcoming in 2020–2021?

- Progress towards removing the Big Hammer
- Progress towards ES Modules
- Following edition of JS language (based on Firefox 78)

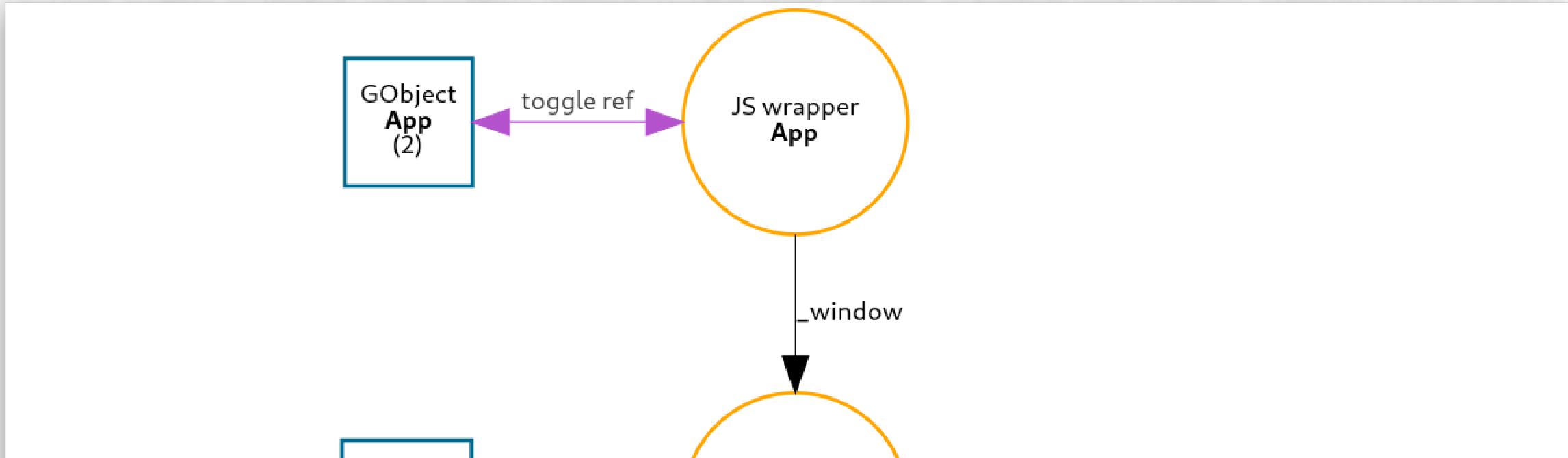
Big Hammer

- What is the Big Hammer?
- Why does it need to be removed?
- Why isn't it removed yet?


PEOPLE'S
CHOICE
The secret is to hit the computer with the hammer.

What is the Big Hammer?

- I gave a talk about this at GUADEC last year
- Some objects accidentally survive a garbage collection
- The “Big Hammer” detects this and runs an extra garbage collection



Why does it need to be removed?

- When memory is constantly being used and discarded, extra garbage collections happen quite often
- Bad for performance

Why isn't it removed yet?

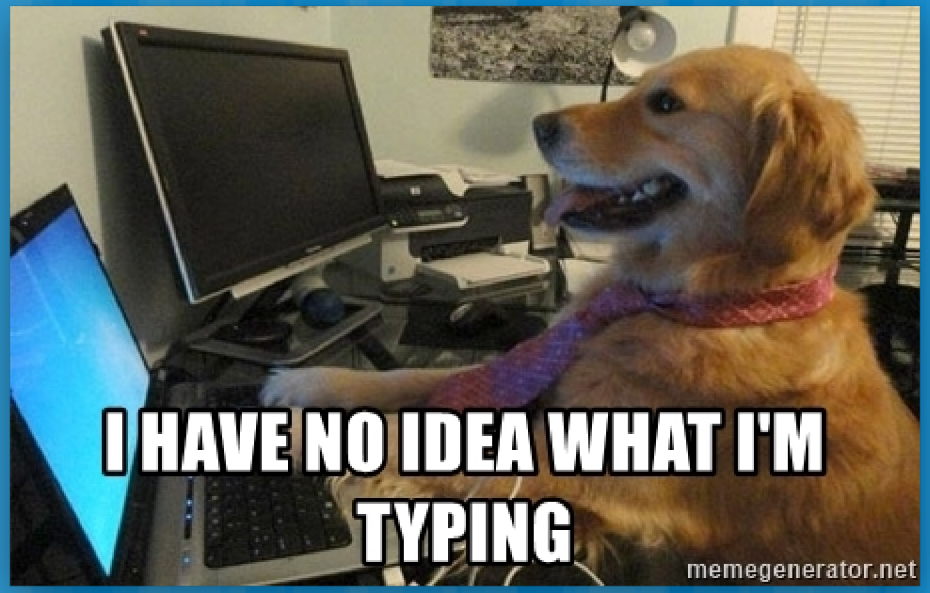
- We have a fix and it (mostly) works
- Except for a few strange cases... 🗣 **Evan Welsh**

Why isn't it removed yet?

- Ironically, the fix leads to higher average memory usage
- Public relations catastrophe! 😬

Why isn't it removed yet?

- We need to tune the garbage collector to work better for our use case
- We need help quantifying acceptable memory usage


I HAVE NO IDEA WHAT I'M TYPING
memegenerator.net

Why isn't it removed yet?

- We need to tune the garbage collector to work better for our use case
- We need help quantifying acceptable memory usage
- We need to figure out some solution that fits:
  - Shell
  - Apps
  - Random scripts that don't even run a main loop

ES Modules

- What are ES modules?
- When can we use them?

“The typical JavaScript program is only one line.”

“The typical JavaScript program is only one line.”

```
<input type="text" id="phone-number"
       onchange="if ($('phone-number').val().length !== 10) alert('Invalid phone number!')">
```

What are ES modules?

- GJS had its own module system: `imports`

👎

```
const System = imports.system;  // loads built-in System module
const {Gtk} = imports.gi;  // loads binding for Gtk
imports.searchPath.unshift('.');
const MyModule = imports.src.myModule;  // loads ./src/myModule.js
```

What are ES modules?

- Now we have one solution for this across JavaScript platforms
- Standardized in ECMAScript 2015: "ES Modules"

👍

```
import System from 'system';
import {Gtk} from 'gi';
import MyModule from './src/myModule.js';
```

When can we use ES modules?

- Currently working on a way to use the two module systems side by side
  🗣 **Evan Welsh**
- Technically, ES modules are a slightly different syntax than regular JavaScript

Next edition of JS

- Based on Firefox 78 🗣 **Evan Welsh**

Next edition of JS

- `??` operator
- `?.` operator
- Static class fields
- Numeric separators (`1_000_000`)
- `String.replaceAll()`
- `Promise.allSettled()`
- `Intl.Locale`
- `Intl.ListFormat`

Help!

Improving the developer experience

- For GNOME Shell
- For apps
- For shell extensions

Developer experience: shell extensions

- Shell extensions seen as "separate" from GNOME
- One of the biggest sources of community unhappiness
- Sri gave a talk on Friday on this topic
- Join the unconference session on Sunday 18:00 UTC!

Developer experience: Shell and apps

- Despite some improvements, developer tools are still not very good
- Here are some ideas
- These projects are not too large
- Can be picked up without knowledge of all of the internals of GJS

Improve console output

```
gjs> ({a:5, b:3, foo:true})
[object Object]

gjs> [undefined, null]
,

gjs> 😬😬😬
```

Issue #104 — Better console interpreter output

Improve debugger

- Debugger is quite bare-bones
- Doesn't handle multiple source files very well
- Most code is spread over multiple source files
- Issue #207 — Source locations and `list` command
- Issue #208 — `backtrace full` command

What else do we need help with?

- Become a code reviewer for GJS
- Create a good workflow for people who want to get started contributing to GJS
- Create a good process for measuring performance and memory usage in GNOME Shell

Let's hack!

Thanks

- **🗣 GJS contributors from 3.36 and 3.38**
- Background pattern by Randy Merrill — MIT license

License

Questions?

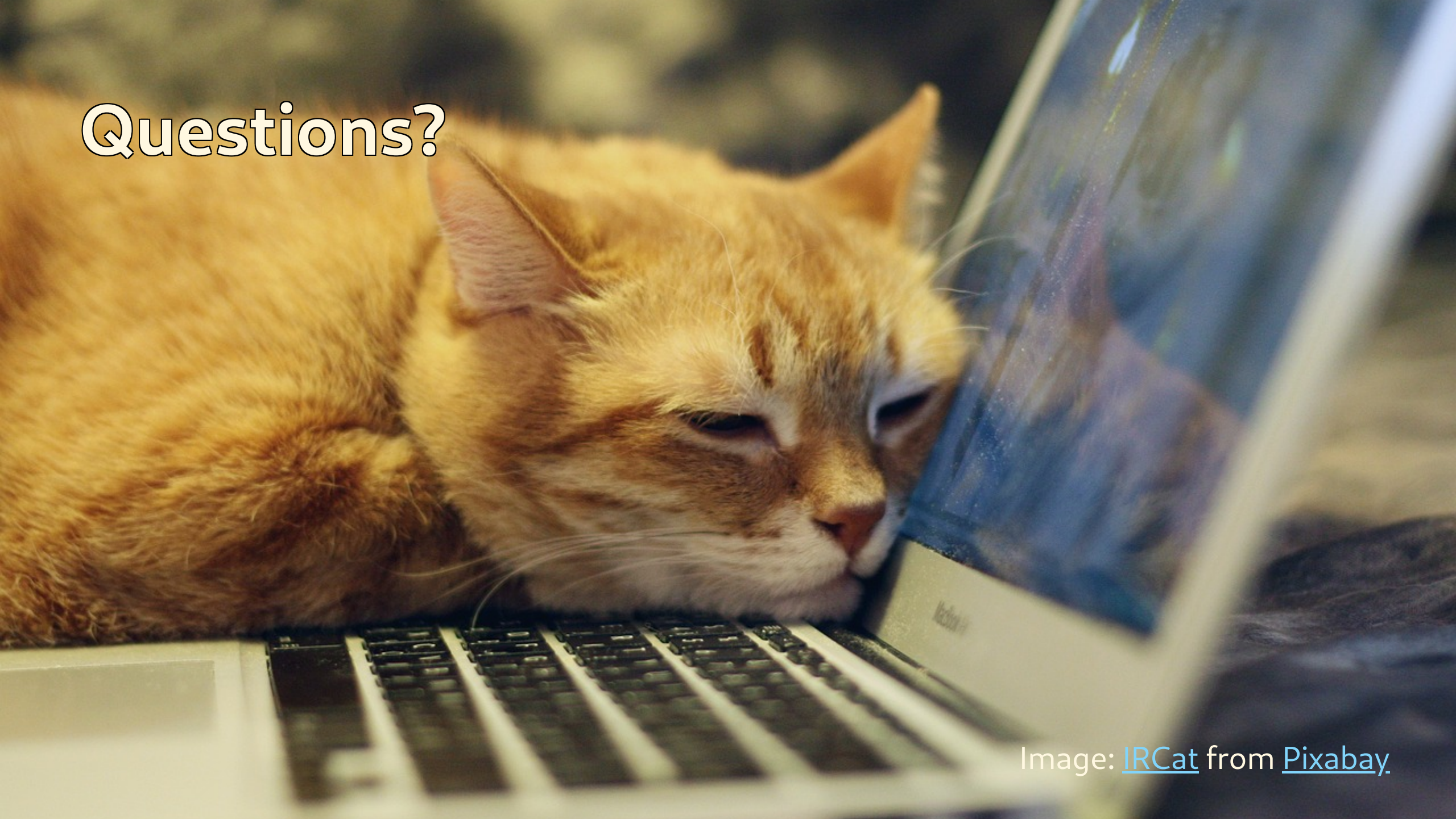


Image: IRCat from Pixabay

Appendix

New String features

`String.trimStart()`,
`String.trimEnd()`

- New standardized methods for trimming whitespace
- replaces the old nonstandard `trimLeft`, `trimRight`

👍

```
'   foo  '.trimStart()
// ⇒ "foo  "
'   foo  '.trimEnd()
// ⇒ "   foo"
```

👎

```
'   foo  '.trimLeft()
'   foo  '.trimRight()
```

New String features

`String.matchAll()`

- Easier access to regular expression capture groups
- Easier iterating through multiple matches in a string

```
const s = 'GNOME 3.36 and 3.38';
const pat = /(\d+)\.(\d+)/g;
for (let [v, major, minor] of s.matchAll(pat)) {
  // On first iteration:
  //   v = '3.36'
  //   major = '3'
  //   minor = '36'
  // On second iteration:
  //   v = '3.38'
  //   major = '3'
  //   minor = '38'
}
```

New Array features

`Array.flat()`

- Well known from lodash and friends

```
[1, 2, [3, 4]].flat()
// ⇒ [1, 2, 3, 4]
```

New Array features

`Array.flatMap()`

- A sort of reverse `Array.filter()`
- You can add elements while iterating functional-style

```
function dashComponents(strings) {
  return strings.flatMap(s => s.split('-'));
}

dashComponents(['foo-bar', 'a-b-c'])
// ⇒ ['foo', 'bar', 'a', 'b', 'c']
```

New Object features

`Object.fromEntries()`

- Create an object from iterable key-value pairs

```
o = Object.fromEntries(['a', 1], ['b', 2])

o.a
// ⇒ 1
o.b
// ⇒ 2
```

New environment features

`globalThis`

- Ever find it weird that GJS's global object is named `window`?

👍

```
if (typeof globalThis.foo === 'undefined')
```

👎

```
if (typeof window.foo === 'undefined')
```

BigInt

- Large numbers in JS are inexact
- BigInt is a new type in JS, arbitrary-precision integers
- BigInt literals are a number suffixed with `n`

```
Number.MAX_SAFE_INTEGER
// ⇒ 9007199254740991
9007199254740992 === 9007199254740993
// ⇒ true
BigInt(Number.MAX_SAFE_INTEGER)
// ⇒ 9007199254740991n
9007199254740992n === 9007199254740993n
// ⇒ false
```

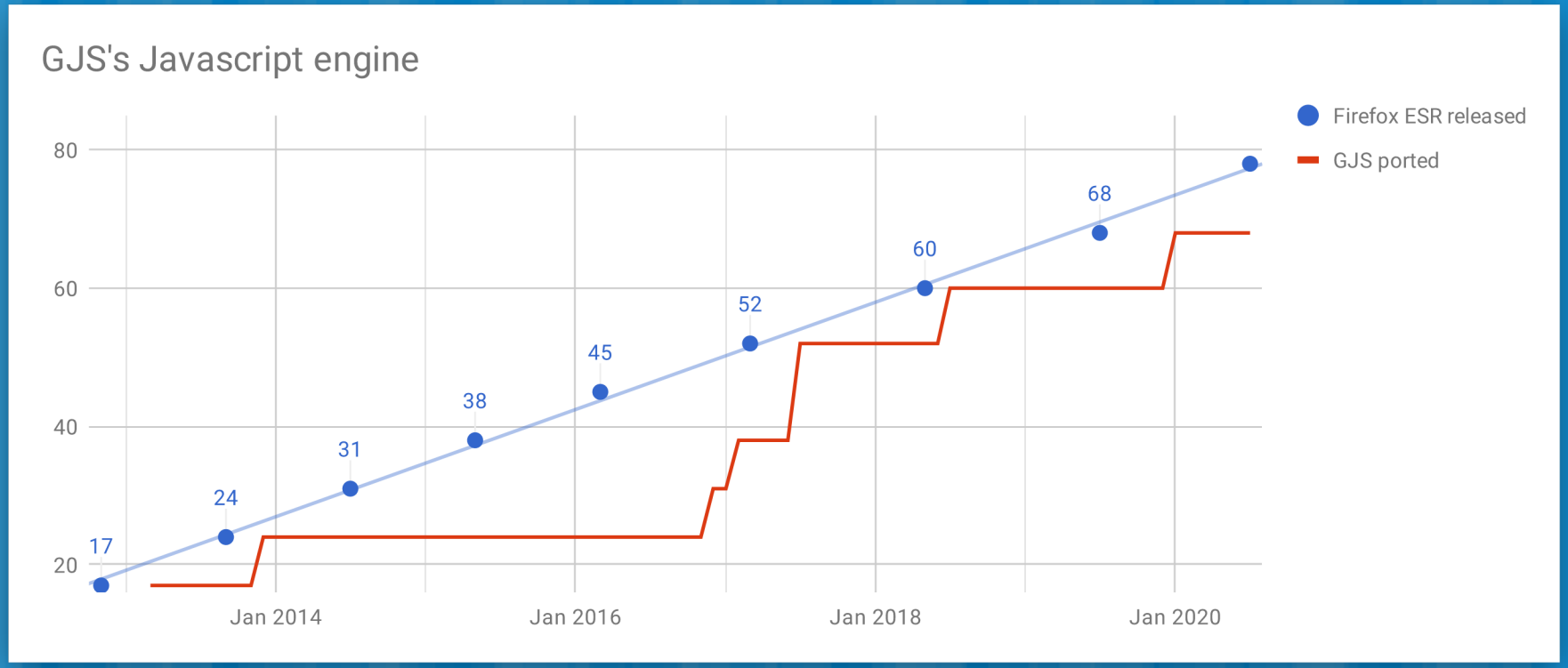
When you should use BigInt

- If you didn't already need to use it? Probably ignore it for now
- In the future, 64-bit APIs will accept BigInt as well as numbers
- Looking for a way to change the type of constants like `GLib.MAXINT64` without breaking existing code 🗣 **Evan Welsh**

BigInt

- Also two new typed array types for BigInts that fit in 64 bits

```
let s = new BigInt64Array([-1n, -2n]);
let u = new BigUint64Array([1n, 2n]);
```


GJS's Javascript engine
Firefox ESR released
GJS ported
80
60
40
20
17
24
31
38
45
52
60
68
Jan 2014
Jan 2016
Jan 2018
Jan 2020