javascript is good actually

hiiiiiiiiiiiiii

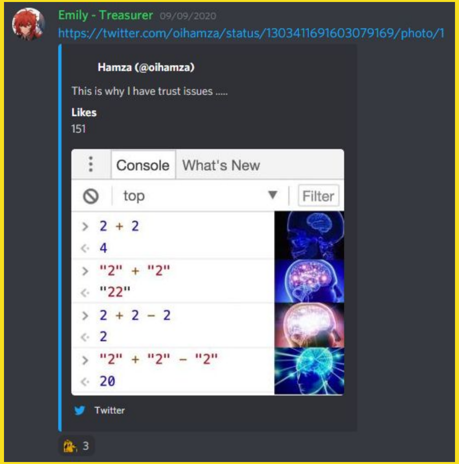
My name is Oisín

Studied compsci in ucd
Was involved with ucd netsoc
Graduated in 2020
Now i’m a software engineer in hubspot doing front-end fun

Disclaimer: I’m not an expert

(and 10 mins is really not a lot of time to talk about anything)

Everyone hates JavaScript



Emily - Treasurer 09/09/2020
https://twitter.com/oihamza/status/1303411691603079169/photo/1

Hamza (@oihamza)

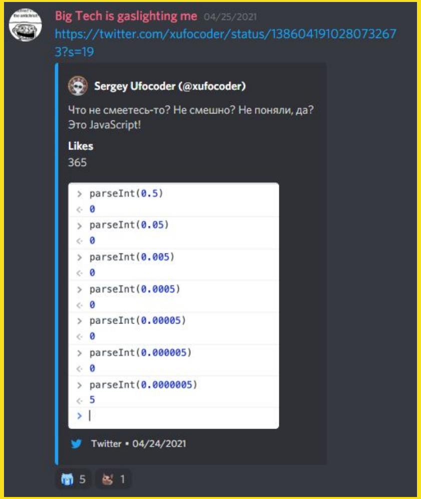
This is why I have trust issues .....

Likes
151

```
> 2 + 2
< 4
> "2" + "2"
< "22"
> 2 + 2 - 2
< 2
> "2" + "2" - "2"
< 20
```

Twitter

3



Big Tech is gaslighting me 04/25/2021
https://twitter.com/xufocoder/status/1386041910280732673?s=19

Sergey Ufocoder (@xufocoder)

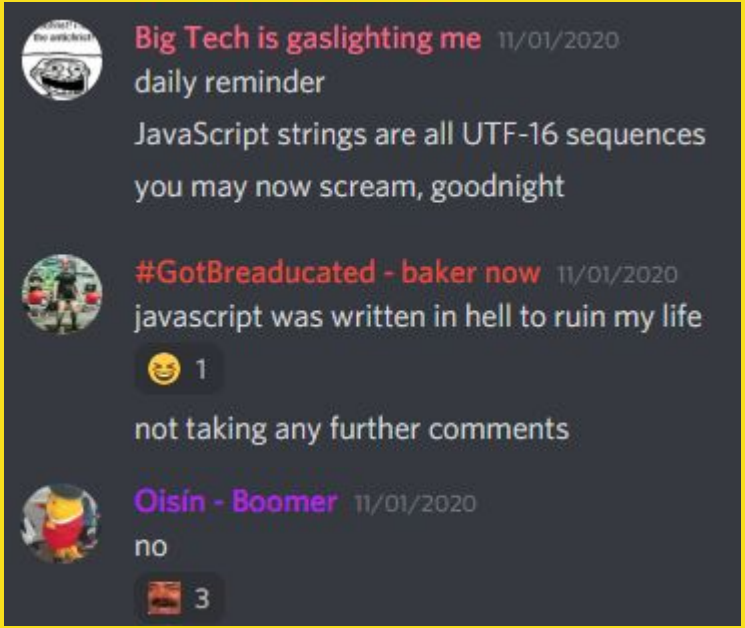
Что не смеетесь-то? Не смешно? Не поняли, да? Это JavaScript!

Likes
365

```
> parseInt(0.5)
< 0
> parseInt(0.05)
< 0
> parseInt(0.005)
< 0
> parseInt(0.0005)
< 0
> parseInt(0.00005)
< 0
> parseInt(0.000005)
< 0
> parseInt(0.0000005)
< 5
> |
```

Twitter • 04/24/2021

5 1



Big Tech is gaslighting me 11/01/2020

daily reminder

JavaScript strings are all UTF-16 sequences

you may now scream, goodnight

#GotBreaducated - baker now 11/01/2020

javascript was written in hell to ruin my life

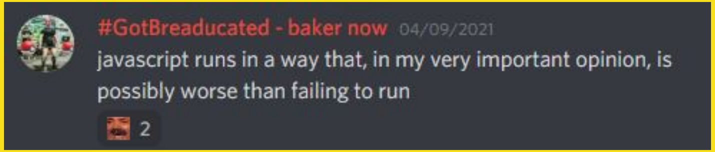
😆 1

not taking any further comments

Oisín - Boomer 11/01/2020

no

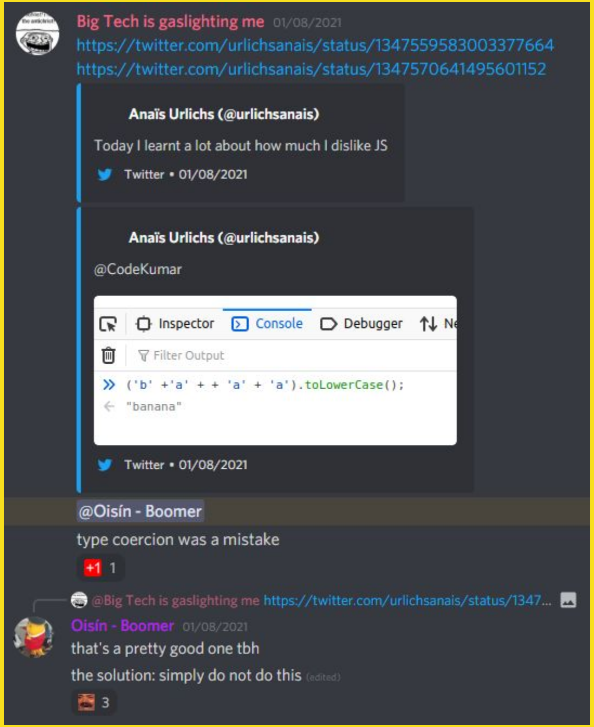
3



#GotBreaducated - baker now 04/09/2021

javascript runs in a way that, in my very important opinion, is possibly worse than failing to run

2



Big Tech is gaslighting me 01/08/2021
https://twitter.com/urlichsanais/status/1347559583003377664
https://twitter.com/urlichsanais/status/1347570641495601152

Anaïs Urlichs (@urlichsanais)

Today I learnt a lot about how much I dislike JS

Twitter • 01/08/2021

Anaïs Urlichs (@urlichsanais)

@CodeKumar

```
» ('b' +'a' + + 'a' + 'a').toLowerCase();
← "banana"
```

Twitter • 01/08/2021

@Oisín - Boomer

type coercion was a mistake

+1 1

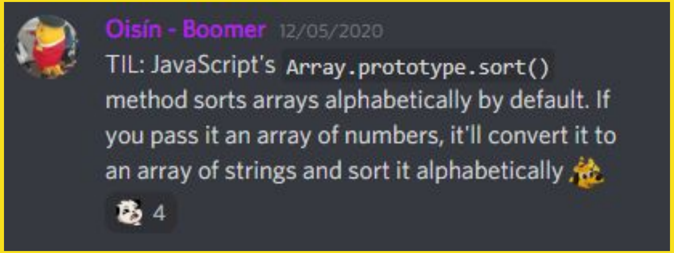
@Big Tech is gaslighting me https://twitter.com/urlichsanais/status/1347...

Oisín - Boomer 01/08/2021

that's a pretty good one tbh

the solution: simply do not do this (edited)

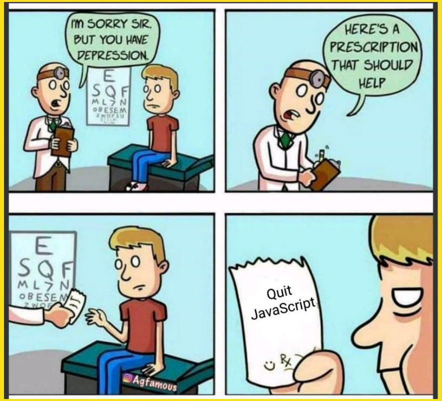
3

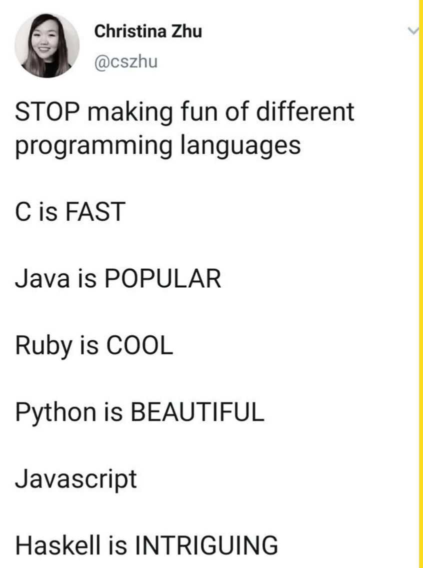


Oisín - Boomer 12/05/2020

TIL: JavaScript's `Array.prototype.sort()` method sorts arrays alphabetically by default. If you pass it an array of numbers, it'll convert it to an array of strings and sort it alphabetically

4


I'M SORRY SIR. BUT YOU HAVE DEPRESSION.
HERE'S A PRESCRIPTION THAT SHOULD HELP
Quit JavaScript
@Agfamous


Christina Zhu
@cszhu

STOP making fun of different programming languages

C is FAST

Java is POPULAR

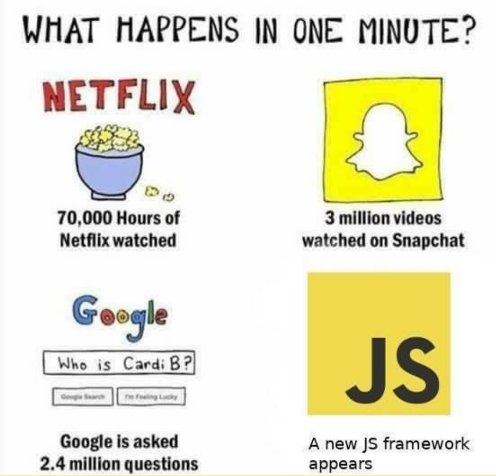
Ruby is COOL

Python is BEAUTIFUL

Javascript

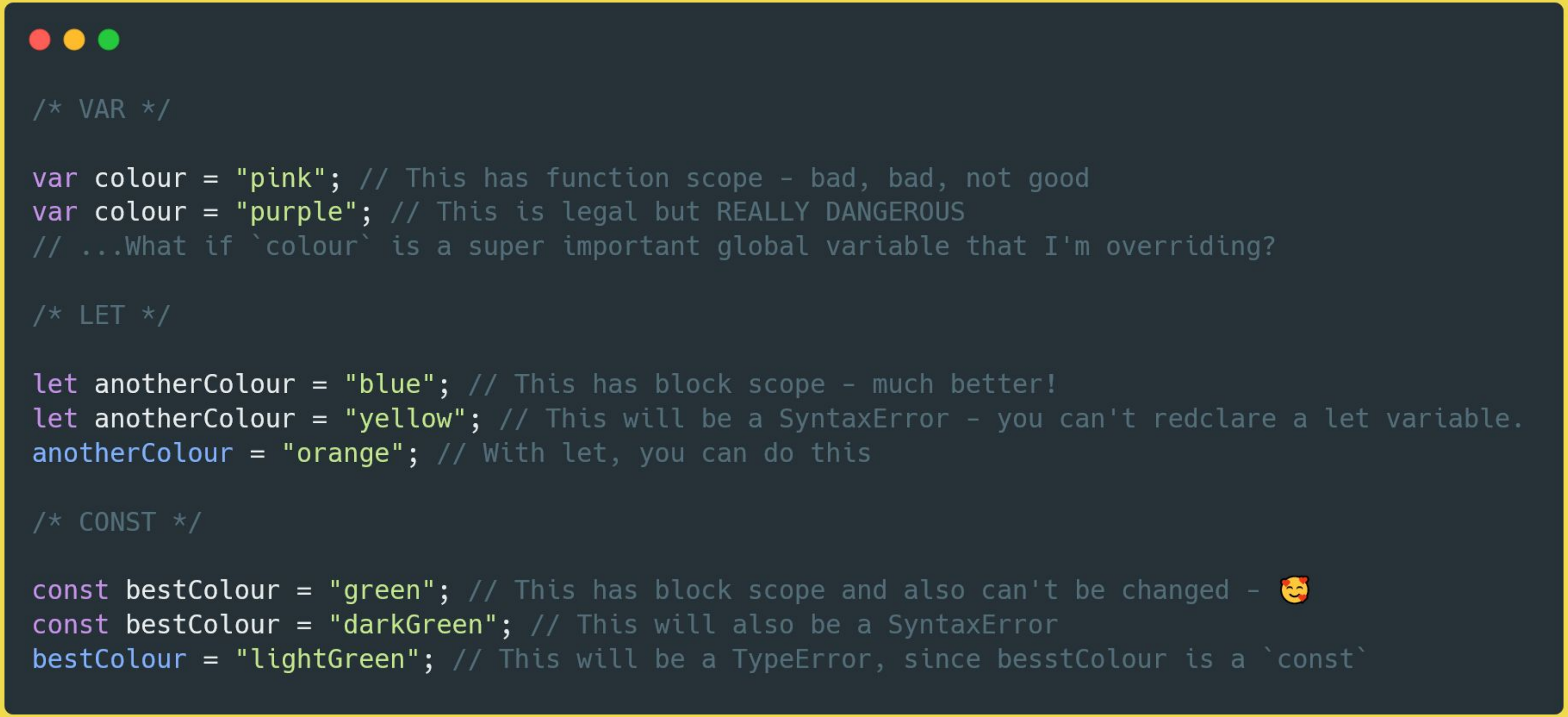
Haskell is INTRIGUING

Just because your language is widely used doesn't mean it is any good!
If those JavaScript developpers could read, they'd be very upset


WHAT HAPPENS IN ONE MINUTE?
NETFLIX
70,000 Hours of Netflix watched
3 million videos watched on Snapchat
Google
Who is Cardi B?
Google is asked 2.4 million questions
JS
A new JS framework appears

They are wrong

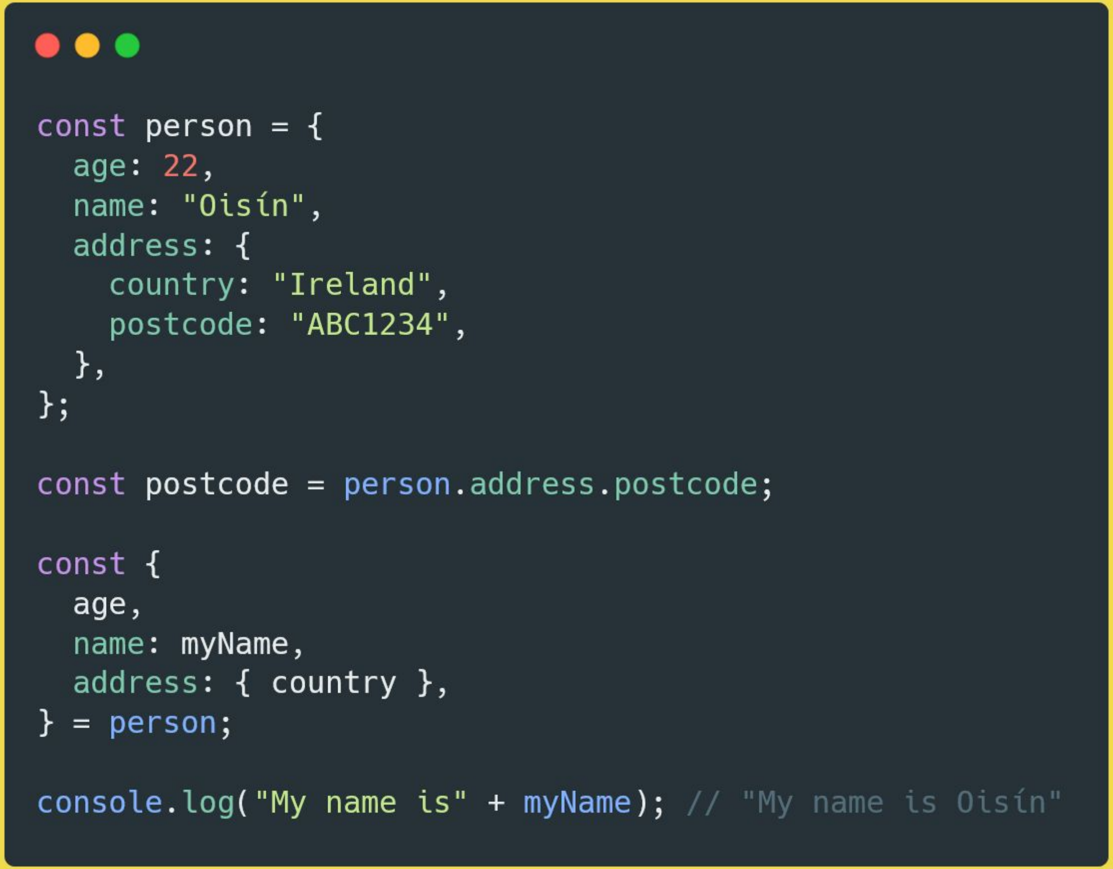
The syntax is really great



```
/* VAR */

var colour = "pink"; // This has function scope - bad, bad, not good
var colour = "purple"; // This is legal but REALLY DANGEROUS
// ...What if `colour` is a super important global variable that I'm overriding?

/* LET */

let anotherColour = "blue"; // This has block scope - much better!
let anotherColour = "yellow"; // This will be a SyntaxError - you can't redclare a let variable.
anotherColour = "orange"; // With let, you can do this

/* CONST */

const bestColour = "green"; // This has block scope and also can't be changed - 🥰
const bestColour = "darkGreen"; // This will also be a SyntaxError
bestColour = "lightGreen"; // This will be a TypeError, since besstColour is a `const`
```

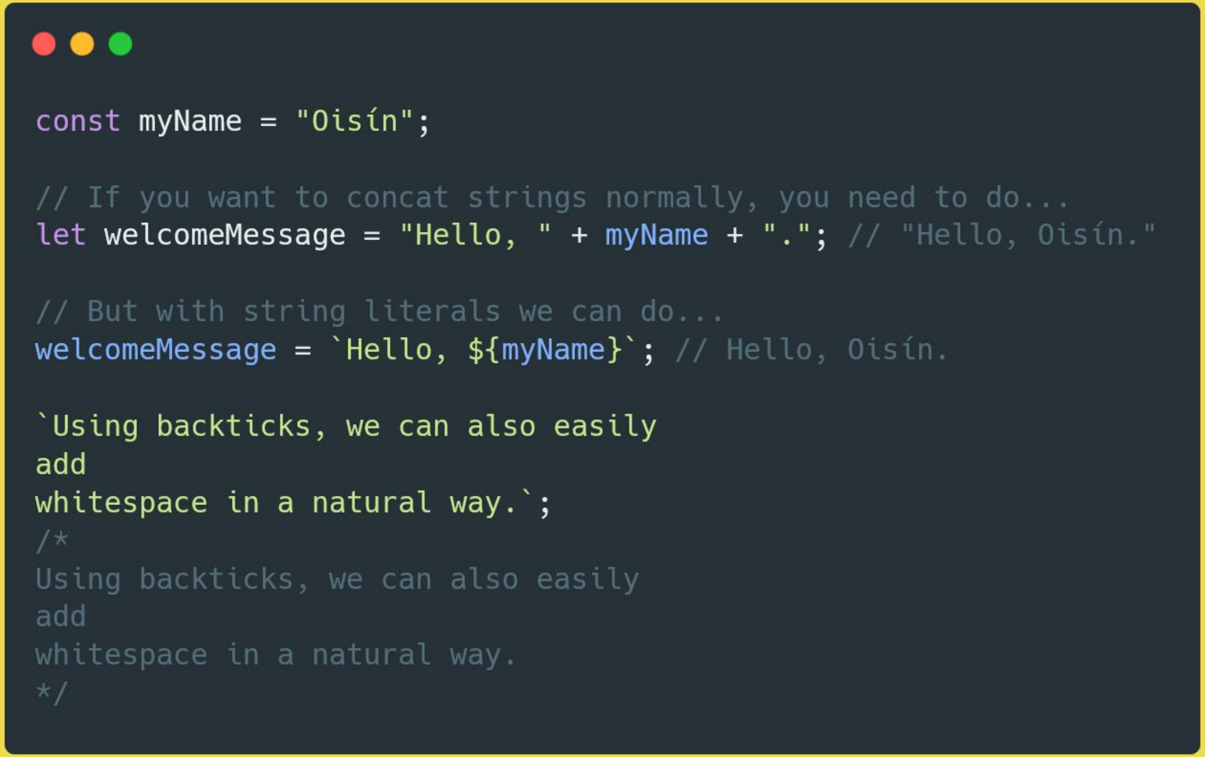
var vs let vs const

Objects and object destructuring!



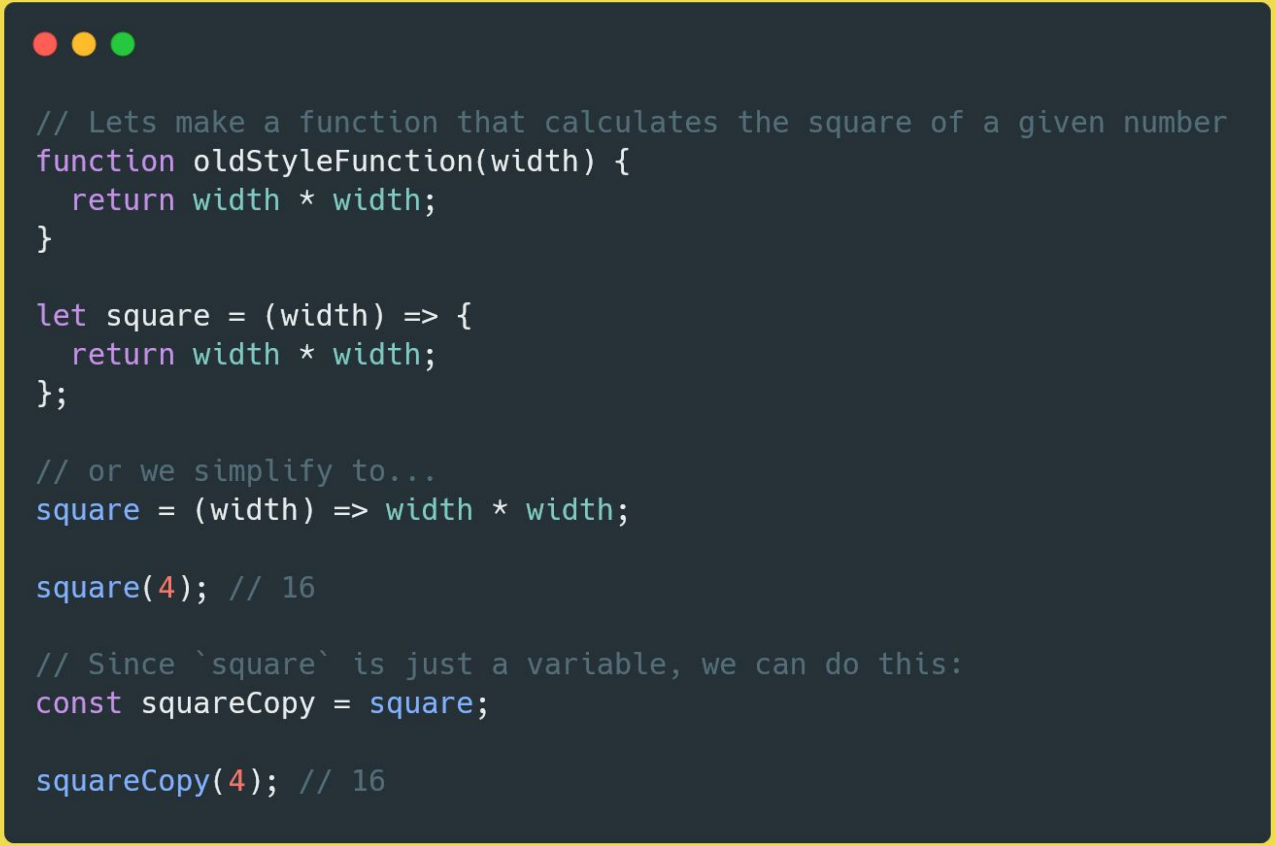
```
const person = {
  age: 22,
  name: "Oisín",
  address: {
    country: "Ireland",
    postcode: "ABC1234",
  },
};

const postcode = person.address.postcode;

const {
  age,
  name: myName,
  address: { country },
} = person;

console.log("My name is" + myName); // "My name is Oisín"
```

String literals!

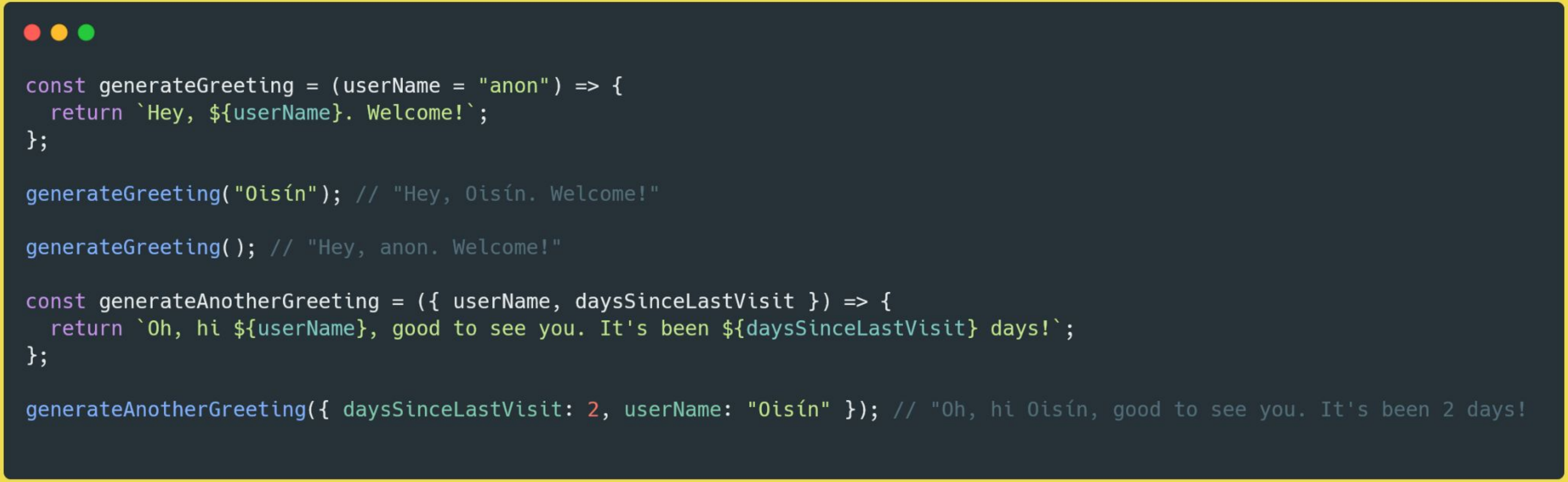


```
const myName = "Oisín";

// If you want to concat strings normally, you need to do...
let welcomeMessage = "Hello, " + myName + "."; // "Hello, Oisín."

// But with string literals we can do...
welcomeMessage = `Hello, ${myName}`; // Hello, Oisín.

`Using backticks, we can also easily
add
whitespace in a natural way.`;
/*
Using backticks, we can also easily
add
whitespace in a natural way.
*/
```

Arrow functions!

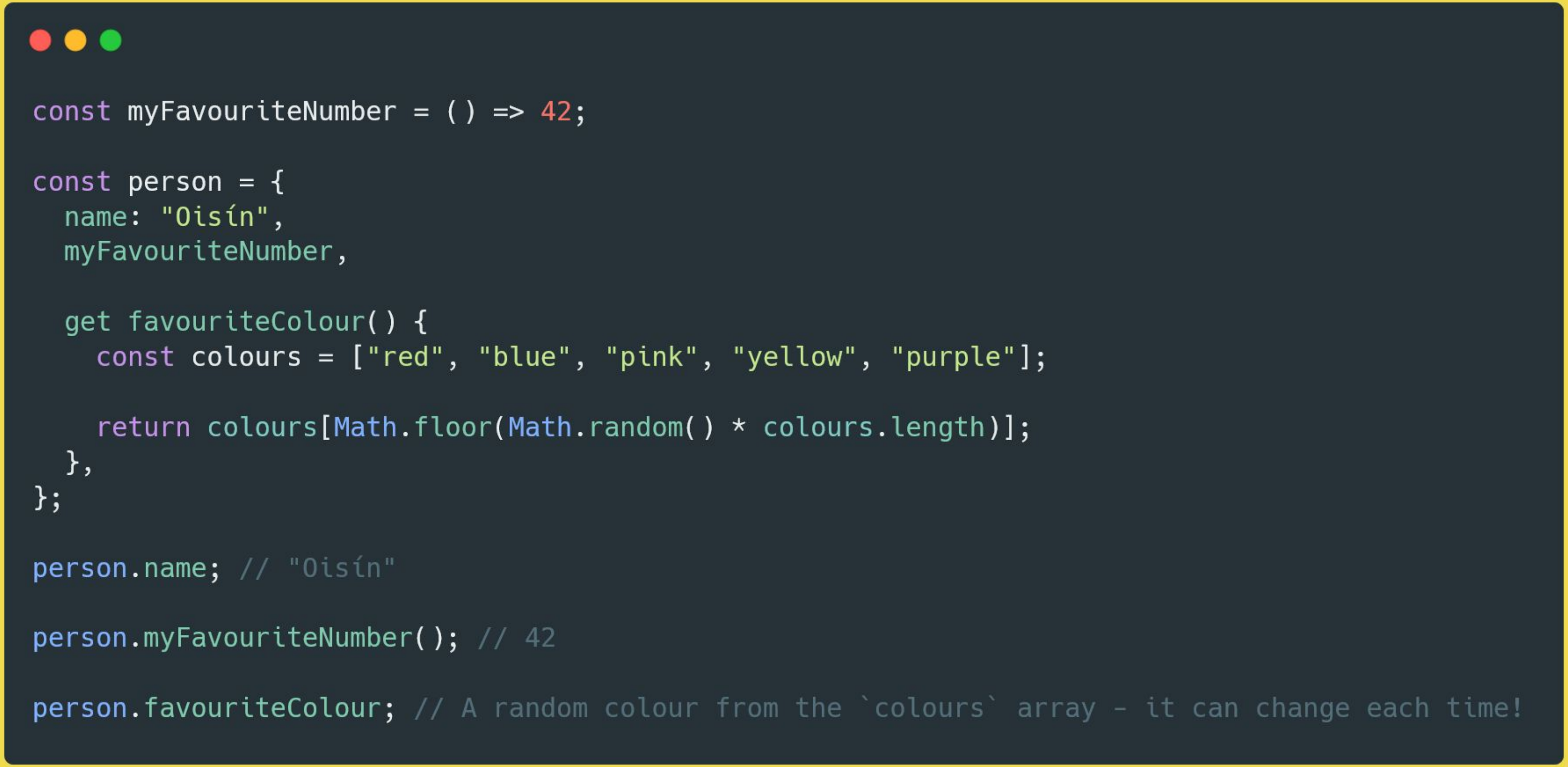


```javascript
// Lets make a function that calculates the square of a given number
function oldStyleFunction(width) {
  return width * width;
}

let square = (width) => {
  return width * width;
};

// or we simplify to...
square = (width) => width * width;

square(4); // 16

// Since `square` is just a variable, we can do this:
const squareCopy = square;

squareCopy(4); // 16
```



```
const generateGreeting = (userName = "anon") => {
  return `Hey, ${userName}. Welcome!`;
};

generateGreeting("Oisín"); // "Hey, Oisín. Welcome!"

generateGreeting(); // "Hey, anon. Welcome!"

const generateAnotherGreeting = ({ userName, daysSinceLastVisit }) => {
  return `Oh, hi ${userName}, good to see you. It's been ${daysSinceLastVisit} days!`;
};

generateAnotherGreeting({ daysSinceLastVisit: 2, userName: "Oisín" }); // "Oh, hi Oisín, good to see you. It's been 2 days!
```

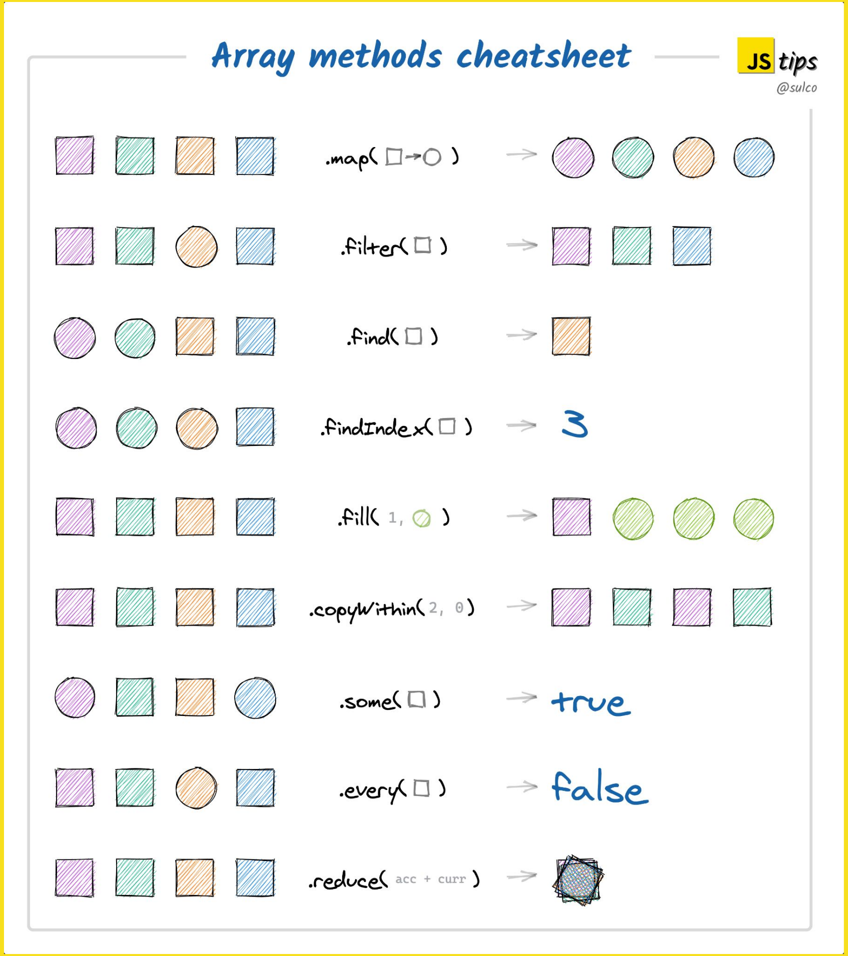
More cool function stuff...



```
const myFavouriteNumber = () => 42;

const person = {
  name: "Oisín",
  myFavouriteNumber,

  get favouriteColour() {
    const colours = ["red", "blue", "pink", "yellow", "purple"];

    return colours[Math.floor(Math.random() * colours.length)];
  },
};

person.name; // "Oisín"

person.myFavouriteNumber(); // 42

person.favouriteColour; // A random colour from the `colours` array - it can change each time!
```

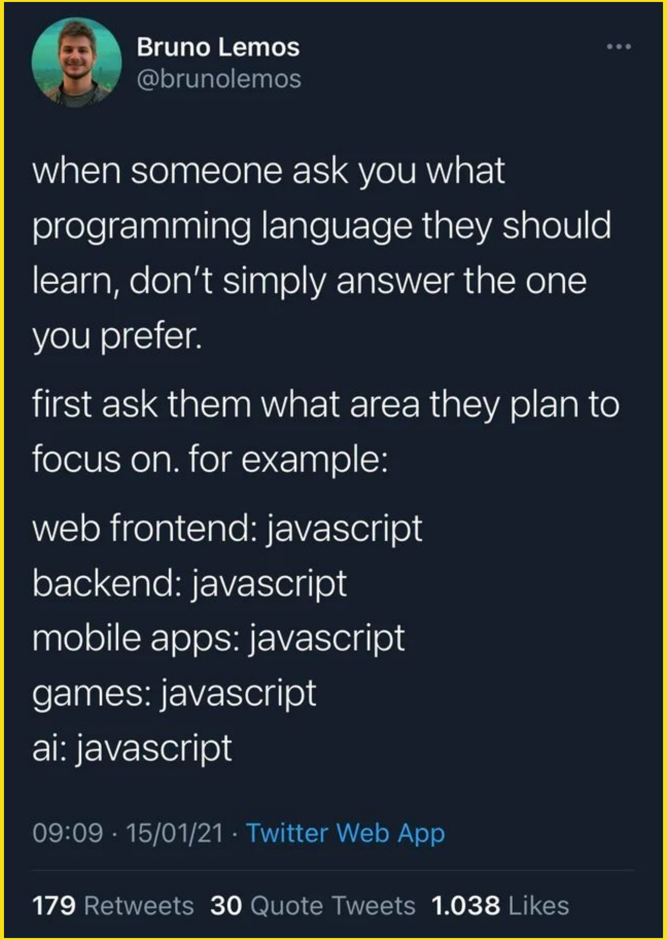
Objects can have getters and setters!

There are some really nice and simple ways for you to deal with arrays



You can pretty much do anything with JavaScript

...but that doesn't mean you should



Bruno Lemos
@brunolemos

when someone ask you what programming language they should learn, don't simply answer the one you prefer.

first ask them what area they plan to focus on. for example:

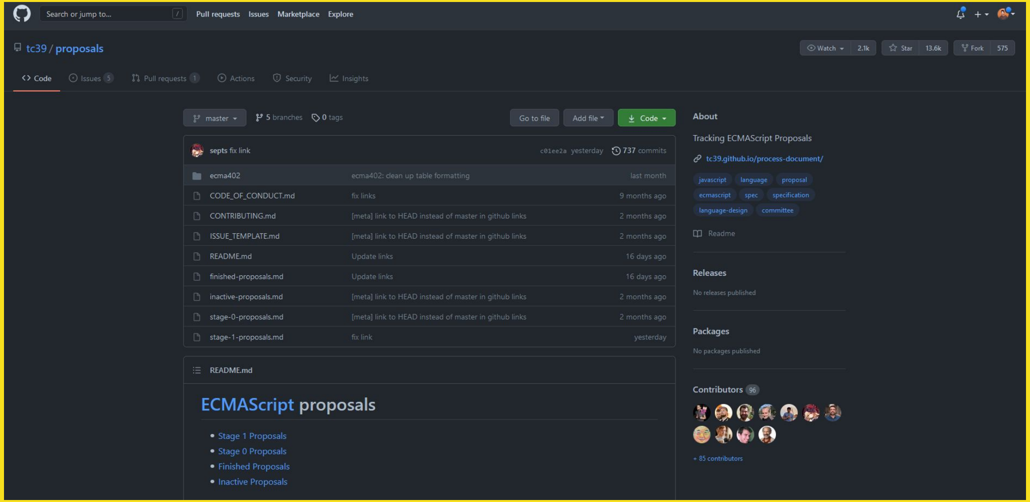
web frontend: javascript
backend: javascript
mobile apps: javascript
games: javascript
ai: javascript

09:09 · 15/01/21 · Twitter Web App

179 Retweets 30 Quote Tweets 1.038 Likes

New features are being developed always

...and anyone can pitch ideas!



septs fix link c01ee2a yesterday 737 commits

ecma402	ecma402: clean up table formatting	last month
CODE_OF_CONDUCT.md	fix links	9 months ago
CONTRIBUTING.md	[meta] link to HEAD instead of master in github links	2 months ago
ISSUE_TEMPLATE.md	[meta] link to HEAD instead of master in github links	2 months ago
README.md	Update links	16 days ago
finished-proposals.md	Update links	16 days ago
inactive-proposals.md	[meta] link to HEAD instead of master in github links	2 months ago
stage-0-proposals.md	[meta] link to HEAD instead of master in github links	2 months ago
stage-1-proposals.md	fix link	yesterday

README.md

ECMAScript proposals

- Stage 1 Proposals
- Stage 0 Proposals
- Finished Proposals
- Inactive Proposals

About

Tracking ECMAScript Proposals

tc39.github.io/process-document/

javascript language proposal ecmascript spec specification language-design committee

Readme

Releases

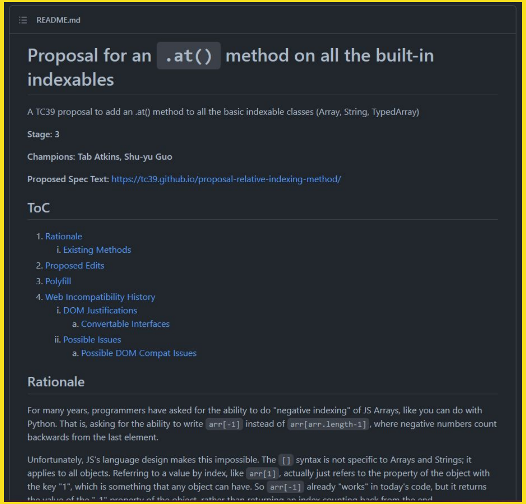
No releases published

Packages

No packages published

Contributors 96

+ 85 contributors


README.md

Proposal for an `.at()` method on all the built-in indexables

A TC39 proposal to add an .at() method to all the basic indexable classes (Array, String, TypedArray)

Stage: 3

Champions: Tab Atkins, Shu-yu Guo

Proposed Spec Text: https://tc39.github.io/proposal-relative-indexing-method/

ToC

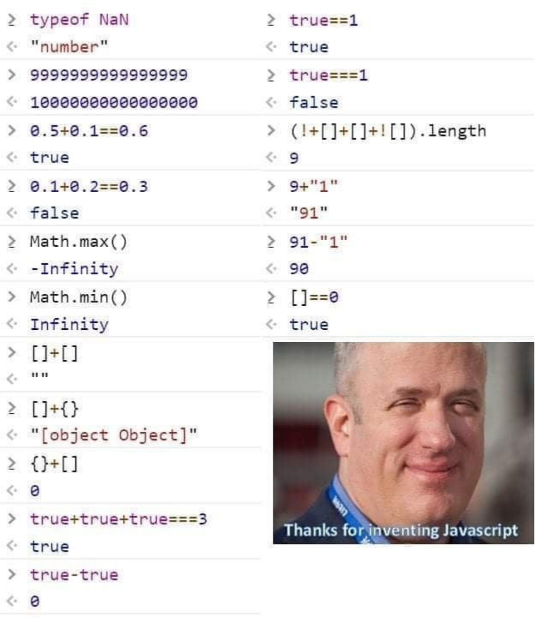
1. Rationale
   i. Existing Methods
2. Proposed Edits
3. Polyfill
4. Web Incompatibility History
   i. DOM Justifications
      a. Convertable Interfaces
   ii. Possible Issues
      a. Possible DOM Compat Issues

Rationale

For many years, programmers have asked for the ability to do "negative indexing" of JS Arrays, like you can do with Python. That is, asking for the ability to write `arr[-1]` instead of `arr[arr.length-1]`, where negative numbers count backwards from the last element.

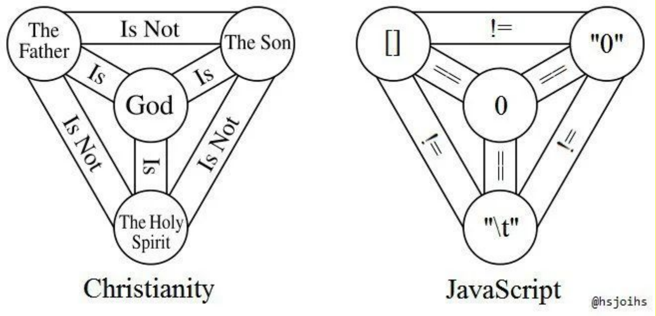
Unfortunately, JS's language design makes this impossible. The `[]` syntax is not specific to Arrays and Strings; it applies to all objects. Referring to a value by index, like `arr[1]`, actually just refers to the property of the object with the key "1", which is something that any object can have. So `arr[-1]` already "works" in today's code, but it returns the value of the "-1" property of the object, rather than returning an index counting back from the end.

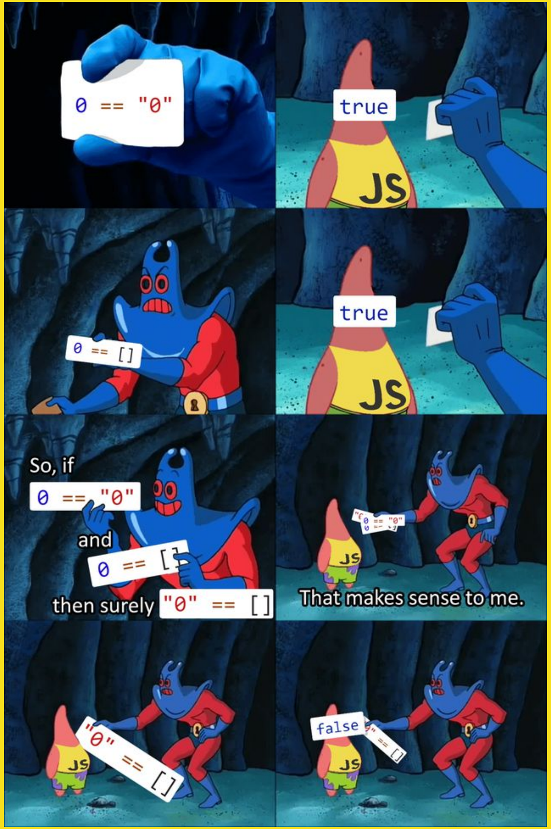
But sometimes JavaScript can get a bit chaotic...



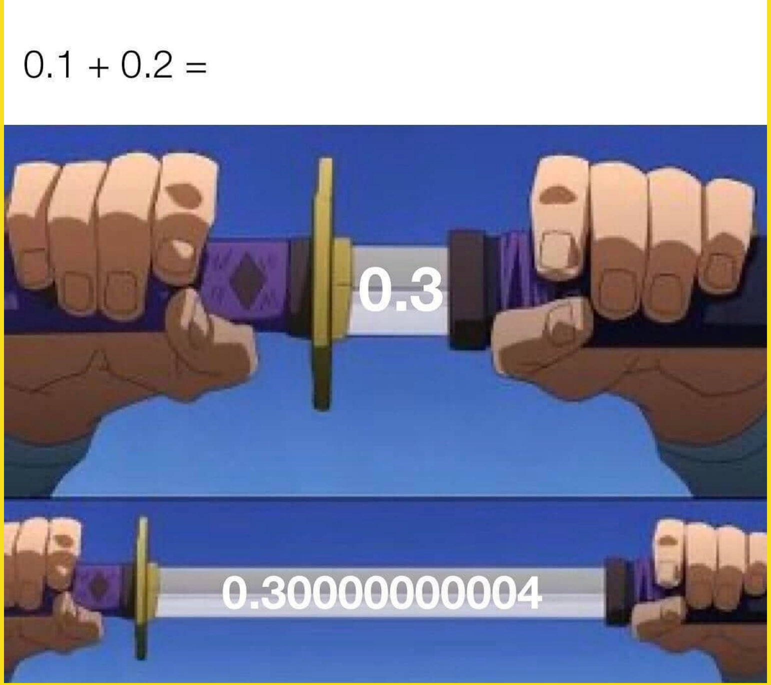
```
> typeof NaN
< "number"
> 9999999999999999
< 10000000000000000
> 0.5+0.1==0.6
< true
> 0.1+0.2==0.3
< false
> Math.max()
< -Infinity
> Math.min()
< Infinity
> []+[]
< ""
> []+{}
< "[object Object]"
> {}+[]
< 0
> true+true+true===3
< true
> true-true
< 0
```

```
> true==1
< true
> true===1
< false
> (!+[]+[]+![]).length
< 9
> 9+"1"
< "91"
> 91-"1"
< 90
> []==0
< true
```

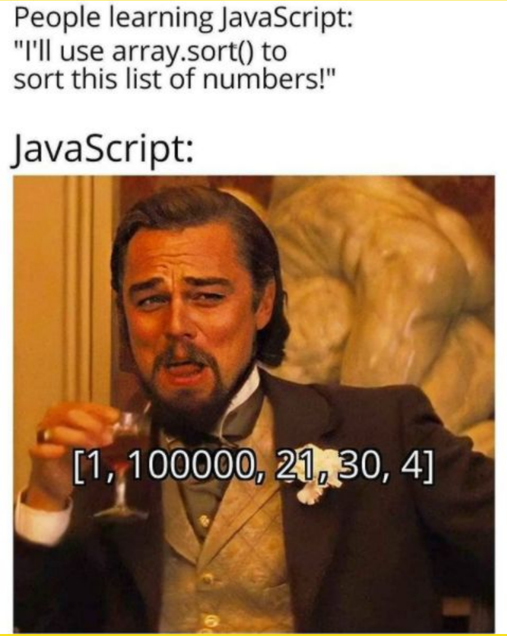




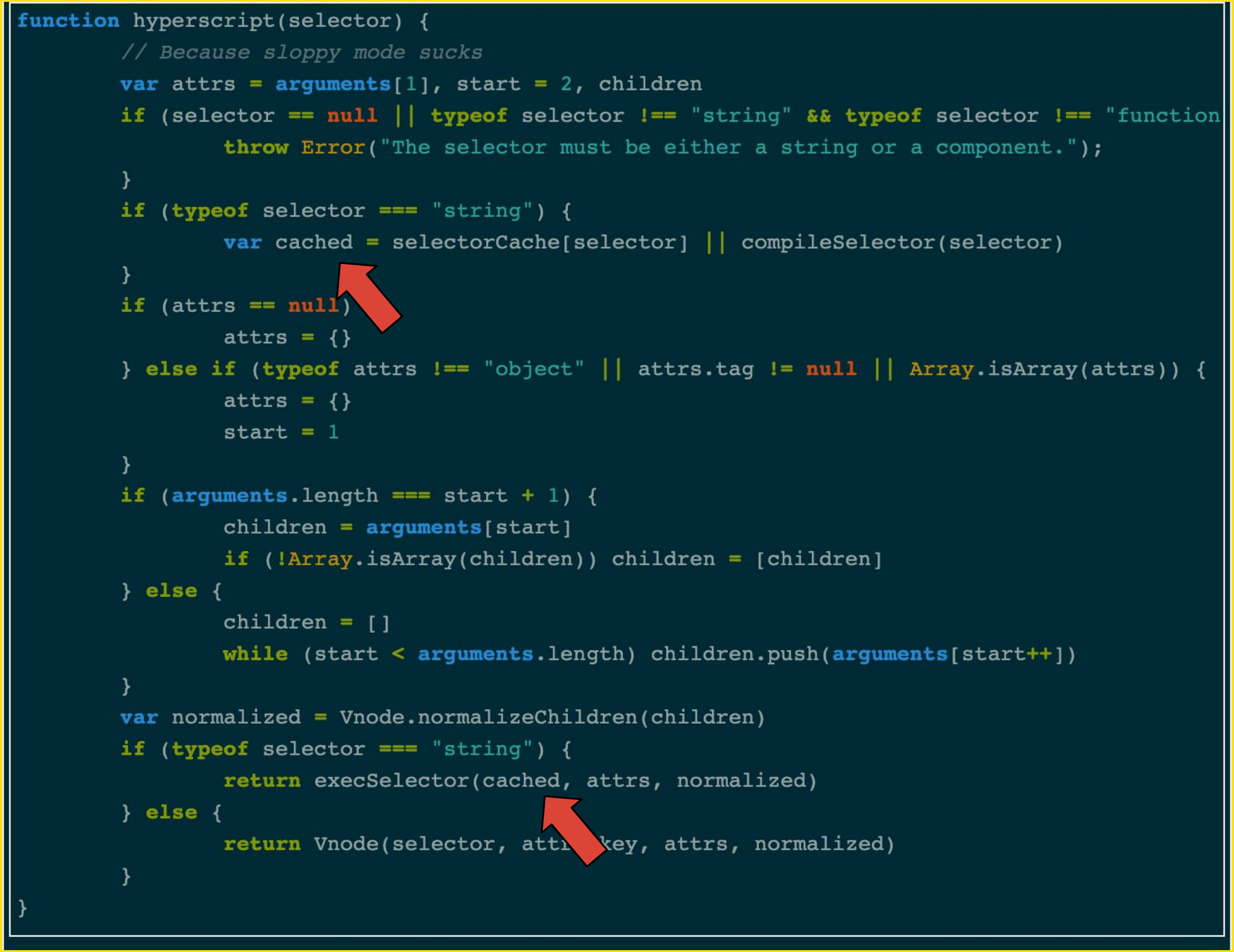
JavaScript interpolates if types mismatch



Numbers and maths are kinda broken


People learning JavaScript:
"I'll use array.sort() to
sort this list of numbers!"
JavaScript:
[1, 100000, 21, 30, 4]

Other random quirks too

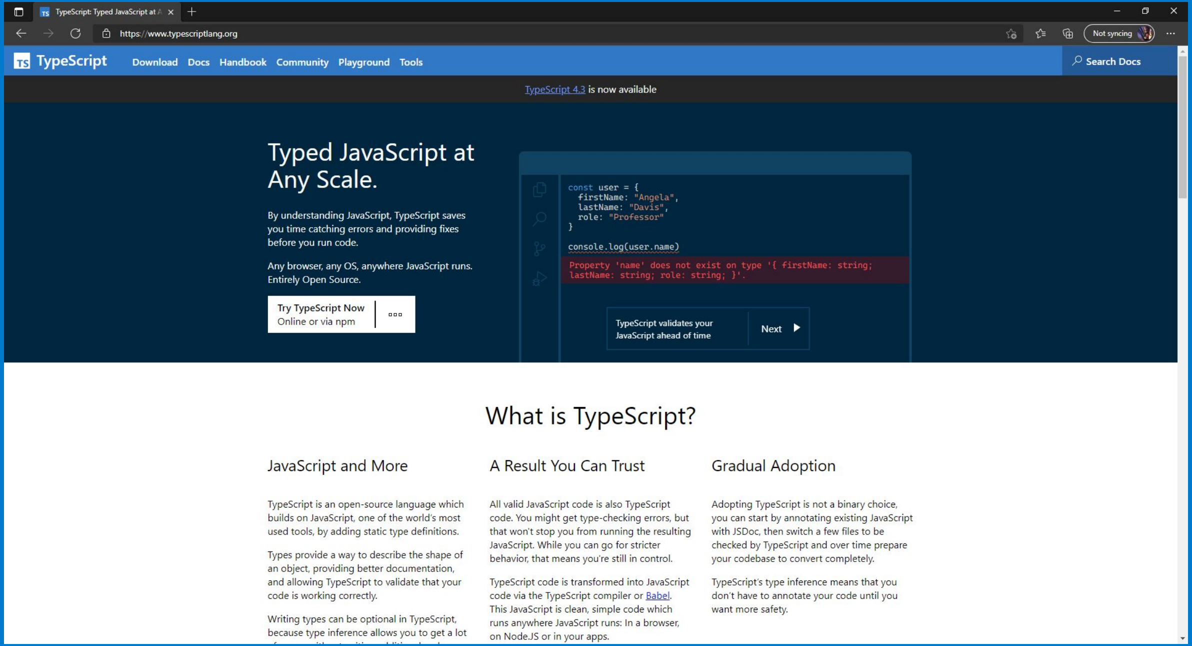


```
function hyperscript(selector) {
        // Because sloppy mode sucks
        var attrs = arguments[1], start = 2, children
        if (selector == null || typeof selector !== "string" && typeof selector !== "function
                throw Error("The selector must be either a string or a component.");
        }
        if (typeof selector === "string") {
                var cached = selectorCache[selector] || compileSelector(selector)
        }
        if (attrs == null) {
                attrs = {}
        } else if (typeof attrs !== "object" || attrs.tag != null || Array.isArray(attrs)) {
                attrs = {}
                start = 1
        }
        if (arguments.length === start + 1) {
                children = arguments[start]
                if (!Array.isArray(children)) children = [children]
        } else {
                children = []
                while (start < arguments.length) children.push(arguments[start++])
        }
        var normalized = Vnode.normalizeChildren(children)
        if (typeof selector === "string") {
                return execSelector(cached, attrs, normalized)
        } else {
                return Vnode(selector, attrs.key, attrs, normalized)
        }
}
```

But there’s a way to fix some of these annoying JavaScript bits...

Just add types!

TypeScript is really great



TypeScript 4.3 is now available

Typed JavaScript at Any Scale.

By understanding JavaScript, TypeScript saves you time catching errors and providing fixes before you run code.

Any browser, any OS, anywhere JavaScript runs. Entirely Open Source.

```
const user = {
  firstName: "Angela",
  lastName: "Davis",
  role: "Professor"
}

console.log(user.name)
Property 'name' does not exist on type '{ firstName: string;
lastName: string; role: string; }'.
```

TypeScript validates your JavaScript ahead of time

What is TypeScript?

JavaScript and More

TypeScript is an open-source language which builds on JavaScript, one of the world's most used tools, by adding static type definitions.

Types provide a way to describe the shape of an object, providing better documentation, and allowing TypeScript to validate that your code is working correctly.

Writing types can be optional in TypeScript, because type inference allows you to get a lot

A Result You Can Trust

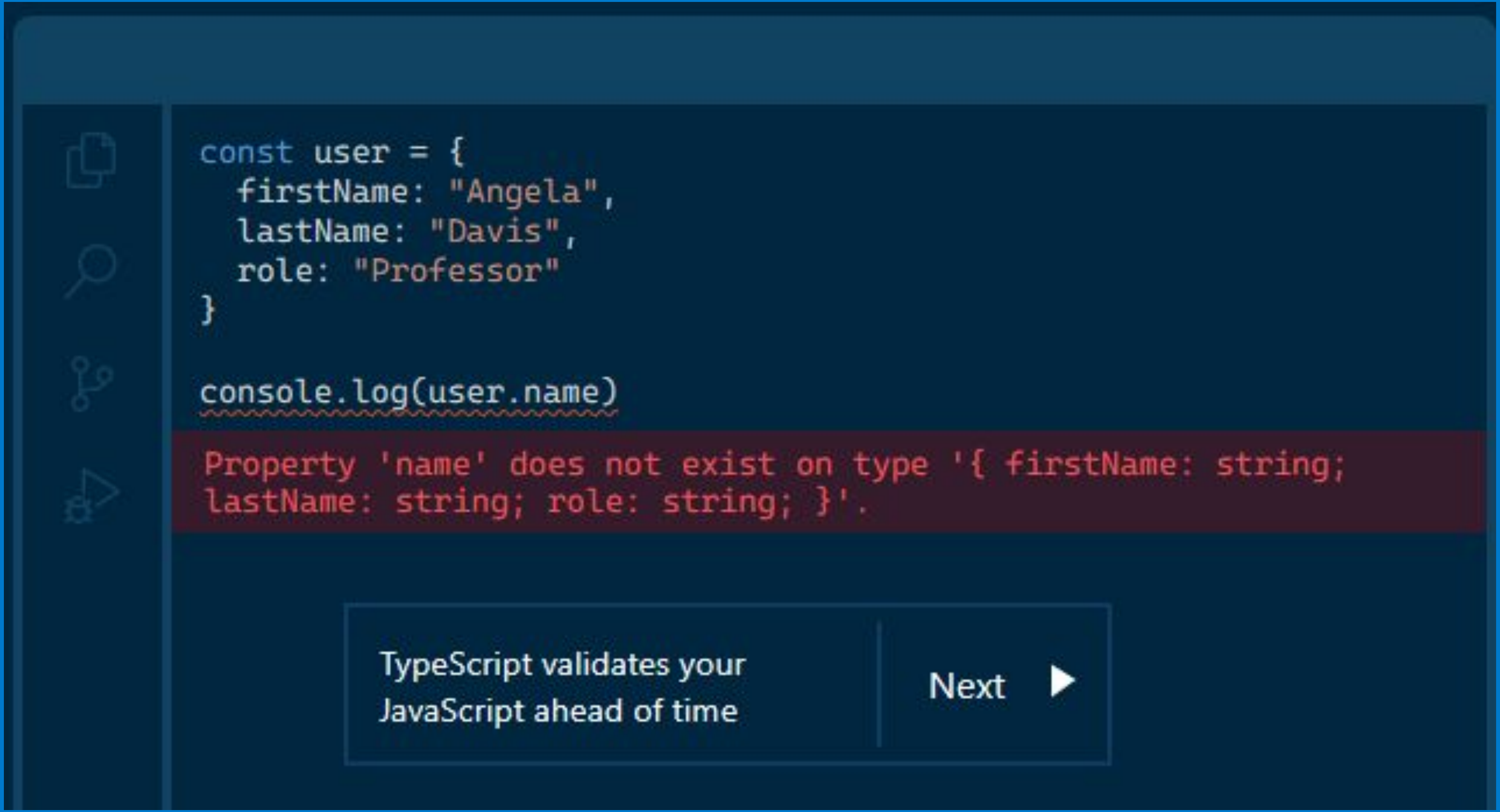
All valid JavaScript code is also TypeScript code. You might get type-checking errors, but that won't stop you from running the resulting JavaScript. While you can go for stricter behavior, that means you're still in control.

TypeScript code is transformed into JavaScript code via the TypeScript compiler or Babel. This JavaScript is clean, simple code which runs anywhere JavaScript runs: In a browser, on Node.JS or in your apps.

Gradual Adoption

Adopting TypeScript is not a binary choice, you can start by annotating existing JavaScript with JSDoc, then switch a few files to be checked by TypeScript and over time prepare your codebase to convert completely.

TypeScript's type inference means that you don't have to annotate your code until you want more safety.



```
const user = {
  firstName: "Angela",
  lastName: "Davis",
  role: "Professor"
}

console.log(user.name)
```

```
Property 'name' does not exist on type '{ firstName: string;
lastName: string; role: string; }'.
```

TypeScript validates your JavaScript ahead of time

Demo time.

JavaScript is good actually

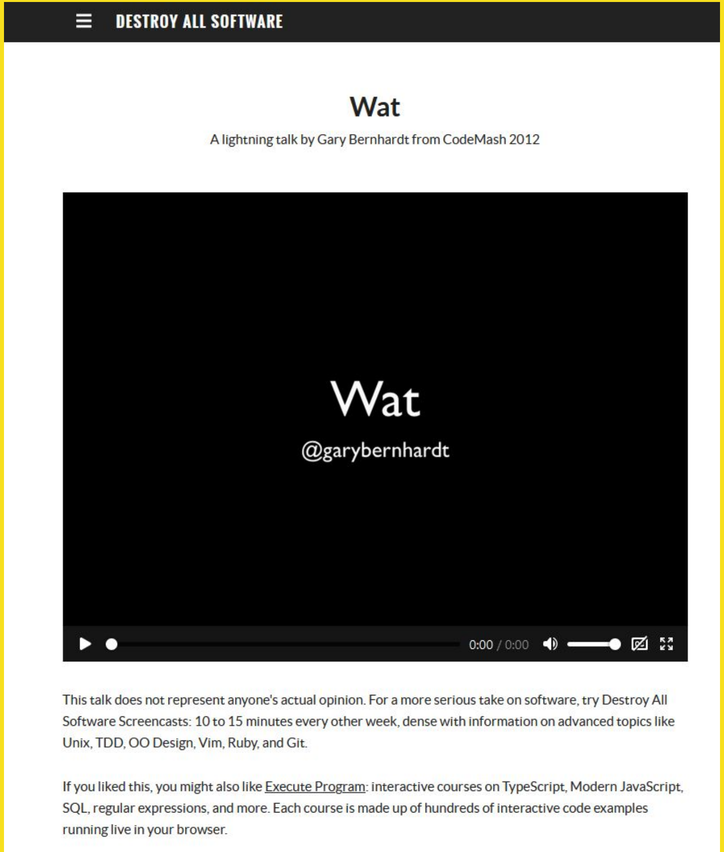
But TypeScript makes it even better

Just JavaScript

Explore the JavaScript Universe

Rebuild your mental model from the inside out.

https://justjavascript.com/



Wat

A lightning talk by Gary Bernhardt from CodeMash 2012

This talk does not represent anyone's actual opinion. For a more serious take on software, try Destroy All Software Screencasts: 10 to 15 minutes every other week, dense with information on advanced topics like Unix, TDD, OO Design, Vim, Ruby, and Git.

If you liked this, you might also like Execute Program: interactive courses on TypeScript, Modern JavaScript, SQL, regular expressions, and more. Each course is made up of hundreds of interactive code examples running live in your browser.

https://www.destroyallsoftware.com/talks/wat

Thanks for listening!

Code samples (in JS and TS) at github.com/oisinq/js-is-good

Slides at ois.im/js-is-good

I’m @oisinq_ on Twitter btw
and @oisinq on Polywork