You might not need TypeScript

Oliver Mckee

23

ypeScript

ctor

Pain

tures

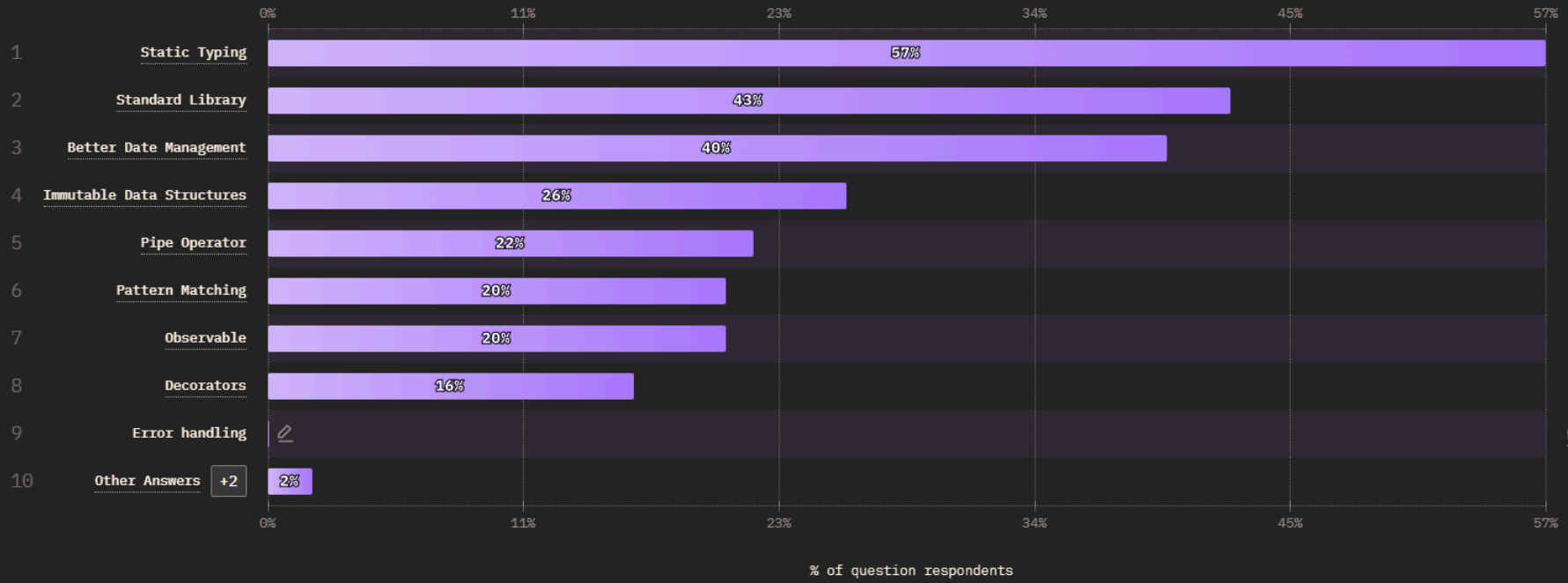
Pain

tures

ogies

MISSING FEATURES

All Respondents | Add Filters…



16,737 | % 71%

Share Chart… Expor

LIBRARY TIER LIST

Front-end Frameworks | Meta-Frameworks | Testing | Mobile & Desktop | Build Tools | Monorepo Tools

S	90%	Vite 98%, Vitest 97%, Playwright 95%, pnpm 92%, Mock Serv… 92%, esbuild 91%, SWC 90%, Testing L… 90%, Astro 90%
A	80%	SvelteKit 89%, Svelte 86%, Turborepo 83%, tsc CLI 82%, Storybook 81%, Turbopack 80%, Rollup 80%
B	60%	Jest 79%, Nuxt 76%, Next.js 76%, React 76%, Nx 75%, Vue.js 75%, Expo 73%, npm Works… 72%, Puppeteer 72%, Cypress 71%, React Nat… 69%, Electron 67%, Preact 66%, Capacitor 65%, Yarn Work… 60%
C		Parcel 56.(, Mocha 52%, webpack 46%, Angular 42%, Ionic 39%, Lerna 36%, Selenium 30%, Gatsby 21%, Cordova 15%

ECMA TC39 Working Group - Futures list, as of 1999.03.30

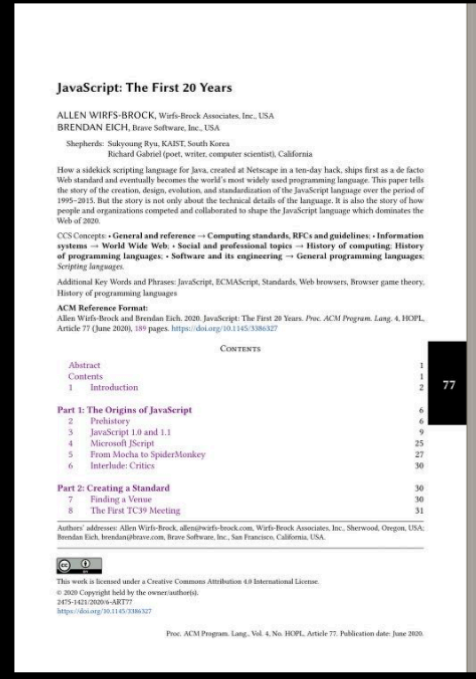
This list records the working group's current list of work items (major topics) for ECMA 262, 4th edition (E4), and beyond.

Provisionally agreed items for 4th Edition

- Modularity enhancements: classes, types, modules, libraries, packages, *etc.*
- Internationalization (I18N) items:
  - Non-normative description of Internationalization library [possibly as a separate ECMA technical report]
- Exceptions and runtime errors
  - Catch guards
- Decimal arithmetic (enhanced or alternative Number object)
- Atomic (thread-safe) operations discussion/definition (possibly non-normative)
- Miscellaneous enhancements [other than modularity]:
  - Declaration qualifiers
  - Extensible syntax (*e.g.*, use of # for RGB values)
  - Units syntax and arithmetic library

Archives

ECMA-262, 1st edition, June 1997 - PDF file	Download
ECMA-262, 2nd edition, August 1998 - PDF file	Download
ECMA-262, 3rd edition, December 1999 - PDF file	Download
ECMA-262, 4th edition (not existing)	No file available
ECMA-262, 5th edition, December 2009 - PDF file	Download
ECMA-262, 5.1th edition, June 2011 - PDF file	Download
ECMA-262, 6th edition, June 2015 - PDF file	Download
ECMA-262, 7th edition, June 2016 - PDF file	Download
ECMA-262, 8th edition, June 2017 - PDF file	Download



JavaScript: The First 20 Years

ALLEN WIRFS-BROCK, Wirfs-Brock Associates, Inc., USA
BRENDAN EICH, Brave Software, Inc., USA

Shepherds: Sukyoung Ryu, KAIST, South Korea
Richard Gabriel (poet, writer, computer scientist), California

How a sidekick scripting language for Java, created at Netscape in a ten-day hack, ships first as a de facto Web standard and eventually becomes the world's most widely used programming language. This paper tells the story of the creation, design, evolution, and standardization of the JavaScript language over the period of 1995–2015. But the story is not only about the technical details of the language. It is also the story of how people and organizations competed and collaborated to shape the JavaScript language which dominates the Web of 2020.

CCS Concepts: • **General and reference → Computing standards, RFCs and guidelines**; • **Information systems → World Wide Web**; • **Social and professional topics → History of computing**; **History of programming languages**; • **Software and its engineering → General programming languages**; *Scripting languages*.

Additional Key Words and Phrases: JavaScript, ECMAScript, Standards, Web browsers, Browser game theory, History of programming languages

ACM Reference Format:
Allen Wirfs-Brock and Brendan Eich. 2020. JavaScript: The First 20 Years. *Proc. ACM Program. Lang.* 4, HOPL, Article 77 (June 2020), 189 pages. https://doi.org/10.1145/3386327

Contents

Authors' addresses: Allen Wirfs-Brock, allen@wirfs-brock.com, Wirfs-Brock Associates, Inc., Sherwood, Oregon, USA; Brendan Eich, brendan@brave.com, Brave Software, Inc., San Francisco, California, USA.

2475-1421/2020/6-ART77
https://doi.org/10.1145/3386327

What did ES4 types look like?

```
// Interface
interface I {
  function f(): String;
}

// Class
class C implements I {
  function f() {
    return "foo";
  }
}

// Function
function add(a: int, b: int): int {
  return a + b;
}

// Union
type StringOrInt = (String,Int)
```

Find out more at https://evertpot.com/ecmascript-4-the-missing-version

proposal-type-annotations

Public

Watch 217 Fork 46 Star 4.2k

main 29 Branches 0 Tags Go to file t Add file Code

orta Merge pull request #219 from someone-aka-sum1/patch-1 2c43464 · 2 months ago 241 Commits

.github/workflows	Update Deploy.yml	3 years ago
site	Update links with changed repository name	2 years ago
syntax	Make updates from feedback	9 months ago
CODEOWNERS	Add Romulo to CODEOWNERS	3 years ago
README.md	Update README.md	2 months ago

README Code of conduct Security

> ⚠️ *This document has not been updated regularly. See TC39 meeting notes from 2022 (March 29, March 31) and 2023 (March 22) for the most up-to-date information.*

ECMAScript proposal: Type Annotations

This proposal aims to enable developers to add type annotations to their JavaScript code, allowing those annotations to be checked by a type checker that is *external to JavaScript*. At runtime, a JavaScript engine ignores them, treating the types as comments.

The aim of this proposal is to enable developers to run programs written in TypeScript, Flow, and other static typing supersets of JavaScript without any need for transpilation, if they stick within a certain reasonably large subset of the language.

About

ECMAScript proposal for type syntax that is erased - Stage 1

tc39.es/proposal-type-annotations/

Readme
Code of conduct
Security policy
Activity
Custom properties
4.2k stars
217 watching
46 forks
Report repository

Contributors 30

+ 16 contributors

Deployments 30

github-pages last year

+ 29 deployments

[–] **pftbest** **180 points** 7 years ago

can you please explain this go syntax to me?

```
type ImmutableTreeList<ElementT> struct {
```

I thought go doesn't have generics.

permalink source embed save save-RES report reply hide child comments

> [–] **Uncaffeinated** [S] **439 points** 7 years ago
>
> It doesn't. That's just a "template" file, which I use search and replace in order to generate the three monomorphized go files.
>
> If you look closely, those aren't angle brackets, they're characters from the Canadian Aboriginal Syllabics block, which are allowed in Go identifiers. From Go's perspective, that's just one long identifier.
>
> permalink source embed save save-RES parent report reply hide child comments
>
> > [–] **pcopley** **360 points** 7 years ago
> >
> > > they're characters from the Canadian Aboriginal Syllabics block
> >
> > Oh my god
> >
> > permalink source embed save save-RES parent report reply hide child comments

What can we do today?


JSDoc

JSDoc relatives

JSDoc for JavaScript

```
/**
 * This is a simple JSDoc comment
 * @param {string} name - The name of the person
 * @returns {string} - The greeting
 */
```

Javadoc for Java

```
/**
 * This is a simple JavaDoc comment
 * @param {string} name - The name of the person
 * @returns {string} - The greeting
 */
```

PHPDoc for PHP

```
/**
 * This is a simple PHPDoc comment
 * @param string $name - The name of the person
 * @return string - The greeting
 */
```


TS

JSDoc Reference

The list below outlines which constructs are currently supported when using JSDoc annotations to provide type information in JavaScript files.

Note:

- Any tags which are not explicitly listed below (such as `@async`) are not yet supported.
- Only documentation tags are supported in TypeScript files. The rest of the tags are only supported in JavaScript files.

Types

- @type
- @param (or @arg or @argument)
- @returns (or @return)
- @typedef
- @callback
- @template
- @satisfies

Classes

- Property Modifiers `@public`, `@private`, `@protected`, `@readonly`
- @override
- @extends (or @augments)
- @implements
- @class (or @constructor)

This repository has been archived by the owner on Aug 1, 2024. It is now read-only.

closure-library

Public archive

Watch 209 Fork 1k Star 4.9k

master 8 Branches 85 Tags Go to file t Code

shicks Update README.md b312823 · 3 months ago 10,301 Commits

.github	Bump actions/checkout from 4.1.0 to 4.1.1	last year
closure-deps	Bump Closure Library version to 20230802.0.0.	last year
closure	[webchannel][js] Fix `useFetchStreams` duplicate message disp...	last year
doc	RELNOTES: Change variable declarations from var to let/const.	2 years ago
scripts	Skip drafting release notes for versions older than the latest ...	last year
third_party/closure/goog/mochikit	Document a surprising behavior of `goog.async.Deferred.STR`...	2 years ago
.gitignore	Move Doc generation from Travis to GitHub Actions.	3 years ago
.npmignore	Remove checked-in demos/editor/deps.js, and generate dep...	3 years ago
AUTHORS	modified goog.net.WebSocket.EventType.CLOSED to include...	3 years ago
CONTRIBUTING	Fix typo in javascript/closure/CONTRIBUTING	8 years ago
LICENSE	Disabling tests that are failing in new browsers. There are bu...	13 years ago
README.md	Update README.md	3 months ago
WORKSPACE	Use closure_base_js_library for closure:base and setup WOR...	3 years ago
alltests.html	Generate alltests.js on `npm run serve_tests`.	3 years ago

About

Google's common JavaScript library

code.google.com/closure/library/

javascript google closure closure-library goog

Readme
Apache-2.0 license
Code of conduct
Security policy
Activity
Custom properties
4.9k stars
209 watching
1k forks
Report repository

Releases 85

Closure Library v20230802 Latest
on Aug 10, 2023

+ 84 releases

Packages


TS

That's why TypeScript now supports a new `@import` comment tag that has the same syntax as ECMAScript imports.

Copy

```
/** @import { SomeType } from "some-module" */

/**
 * @param {SomeType} myValue
 */
function doSomething(myValue) {
    // ...
}
```

`/** @satisfies */` can catch type mismatches:

Copy

```
// @ts-check

/**
 * @typedef CompilerOptions
 * @prop {boolean} [strict]
 * @prop {string} [outDir]
 */

/**
 * @satisfies {CompilerOptions}
 */
let myCompilerOptions = {
    outdir: "../lib",
//  ~~~~~~ oops! we meant outDir
};
```

JSDoc in action

Enabling JSDoc type checking

1. Enable via comment

```
// @ts-check
```

2. Enable via tsconfig.json

```
{
  "compilerOptions": {
    "allowJs": true,
    "checkJs": true,
    "noEmit": true,
  }
}
```

3. Enable globally in VS Code

```
{
  "js/ts.implicitProjectConfig.checkJs": true,
}
```

Importing

```
// TypeScript

import type { MyObject } from "my-types"
```

Defining an Object

```
// TypeScript

/**
 * A user object
 */
interface User {
  /**
   * The name of the user
   */
  name: string;
  /**
   * The age of the user
   */
  age: number;
  /**
   * Whether the user has full access
   */
  fullAccess: boolean;
}

let myUser: User;
```

Extending an Object

```
// TypeScript

interface Foo {
  /**
   * The A string
   */
  a: string;
}

interface Bar extends Foo {
  b: number;
}
```

Typing a variable

```
// TypeScript

const names: string[] = ["Alice", "Bob", "Eve"];
```

Defining a constant

```
// TypeScript

const myObject = {
  foo: true,
  bar: false
} as const;
```

Function

```
// TypeScript

/**
 * A function that adds two numbers
 *
 * @param a The first number
 * @param b The second number
 * @returns The sum of the two numbers
 */
export function add(a: number, b: number): number {
  return a + b;
}
```

Function Overloading

```ts
// TypeScript

function foo(a: string): void;
function foo(a: number): void;
function foo(a: string | number): void {}
```

Defining a callback

```
// TypeScript

/**
 * The function foo
 */
type Foo = (a: string, b: boolean) ⇒ void
```

Assigning `this`

```
// TypeScript

interface Foo {
  /**
   * The value
   */
  value: string;
  /**
   * The method
   */
  method(): void;
}

function bar(this: Foo) {
  this.method();
}
```

Classes

```
class Foo {
  protected readonly a: string;

  private b: number;

  public constructor(a: string, b: number) {
    this.a = a;
    this.b = b;
  }
}
```

Implements

```
// TypeScript

interface Bar {
  /**
   * The fuzz method
   */
  fuzz(): void;
}

class Foo implements Bar {
  fuzz() {}
}
```

Casting

```
// TypeScript

const mystery = undefined as unknown

const notMystery = mystery as string
```

Modifiers

```
// TypeScript

type Bar = {
  stringKey: string
  optionalNumber?: number
  optionalNumber2: number | undefined
  nullableString: string | null
  nullableOptionalString: string | null | undefined
  union: string | number
  nested: {
    stringKey: string
  }
}
```

Type Guards

```
type LiteralString = "bar" | "foo"

function guard(value: LiteralString): value is LiteralString {
  return value === "foo" || value === "bar";
}

function asserts(value: LiteralString): asserts value is LiteralString {
  if (!guard(value)) {
    throw new Error("Invalid value");
  }
}
```

Generics

```
// TypeScript

type Foo<T> = {
  /**
   * The value
   */
  value: T;
  /**
   * The promise
   */
  promise: Promise<T>;
}
```

Generic with base type and default

```
type Bar = {
  foo: string
};

type FooBar = {
  foo: string
  bar: string
}

type Foo<T extends Bar = FooBar> = {
  value: T
}
```

Comment Syntax

```
/**
 * Logs the amount to the console, throws a {@link RangeError} if the amount is less than 0
 *
 * @param {number} amount - The amount to log
 * @returns {void}
 * @throws {RangeError} If the amount is less than 0, a {@link RangeError} is thrown
 */
function logAmount(amount) {
  if (amount > 0) {
    console.info(`The amount is ${amount}`);
  } else {
    throw new RangeError("amount cannot be less than 0");
  }
}
```

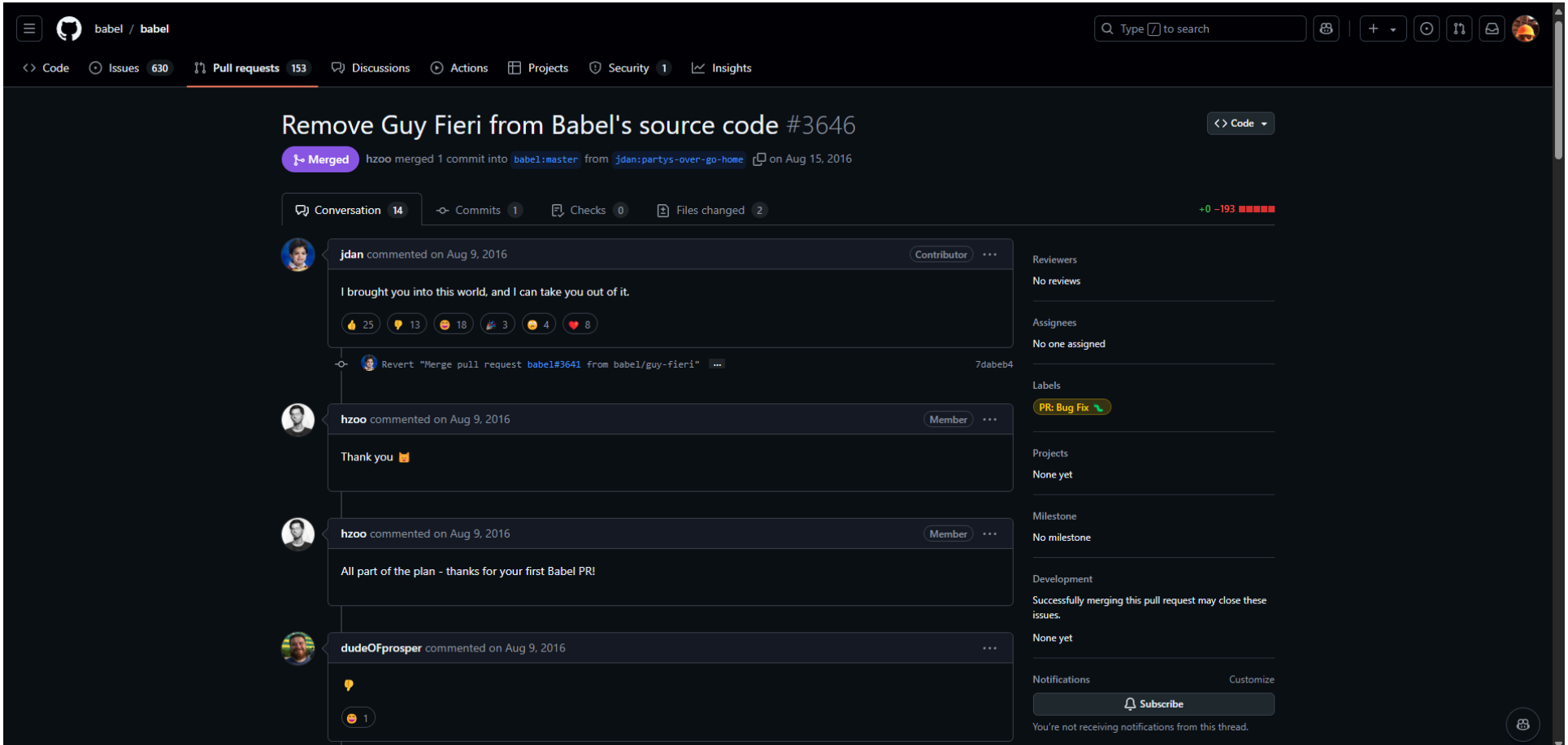
Comment Syntax

```
const STRING_CONSTANT = "Hello, World!";

/**
 * @typedef {object} Foo
 * @property {string} bar - [A string constant](#STRING_CONSTANT)
 * @property {string} baz - ![Fine](http://localhost:3030/assets/fine.webp)
 */

/**
 * @type {Foo}
 */
const foo = {};

foo.bar
foo.baz
```


babel / babel
Type / to search
Code
Issues 630
Pull requests 153
Discussions
Actions
Projects
Security 1
Insights
Remove Guy Fieri from Babel's source code #3646
Code
Merged
hzoo merged 1 commit into babel:master from jdan:partys-over-go-home on Aug 15, 2016
Conversation 14
Commits 1
Checks 0
Files changed 2
+0 −193
jdan commented on Aug 9, 2016
Contributor
I brought you into this world, and I can take you out of it.
25
13
18
3
4
8
Revert "Merge pull request babel#3641 from babel/guy-fieri"
7dabeb4
hzoo commented on Aug 9, 2016
Member
Thank you
hzoo commented on Aug 9, 2016
Member
All part of the plan - thanks for your first Babel PR!
dudeOFprosper commented on Aug 9, 2016
1
Reviewers
No reviews
Assignees
No one assigned
Labels
PR: Bug Fix
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Subscribe
You're not receiving notifications from this thread.

What’s next for JSDoc?

Coming soon to a JSDoc near you

New JSDocs tags

@nonnull

```
// TypeScript

let items = [1, 2, 3, 4, 5];

let item = items.pop()!;
```

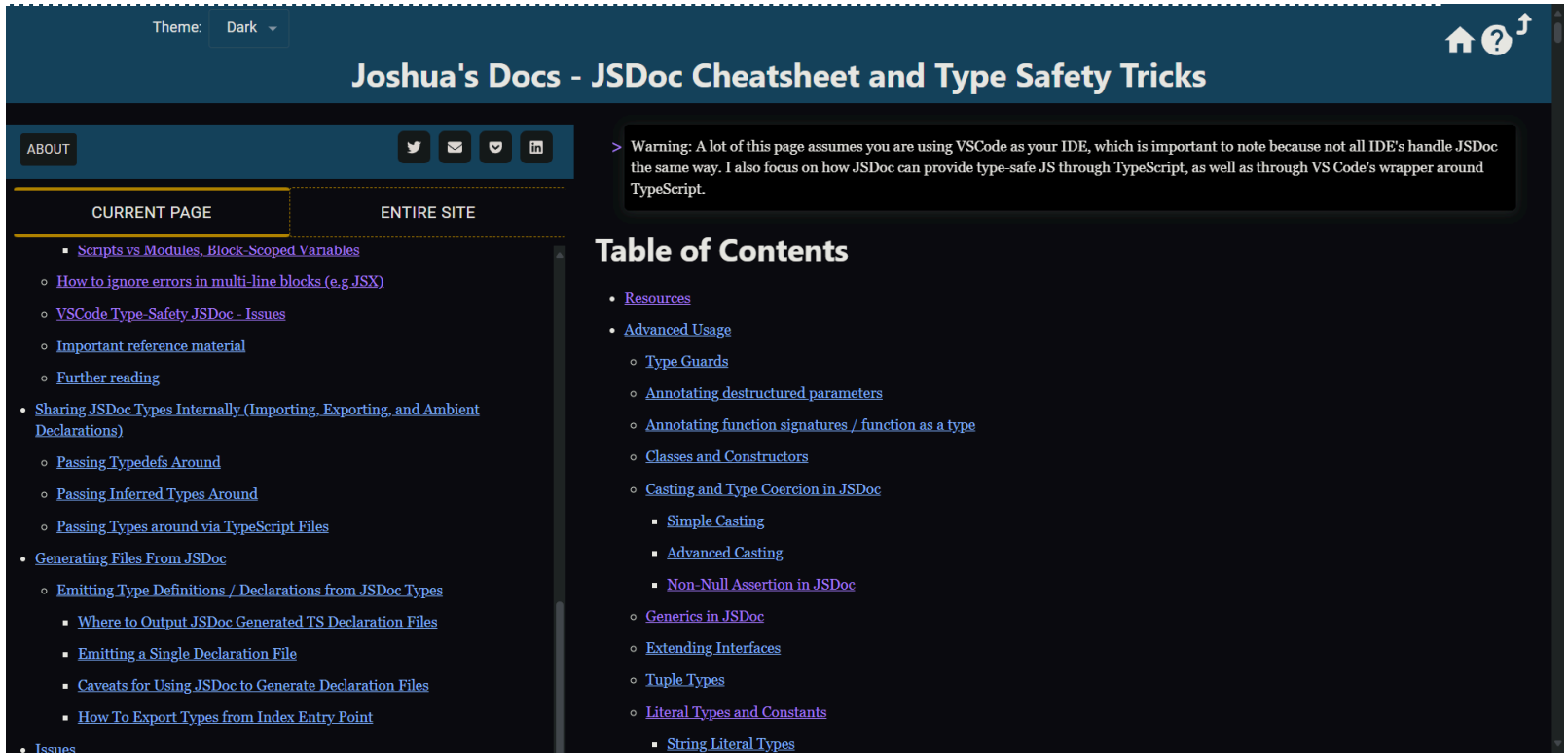
Coming soon to a JSDoc near you

New JSDocs tags

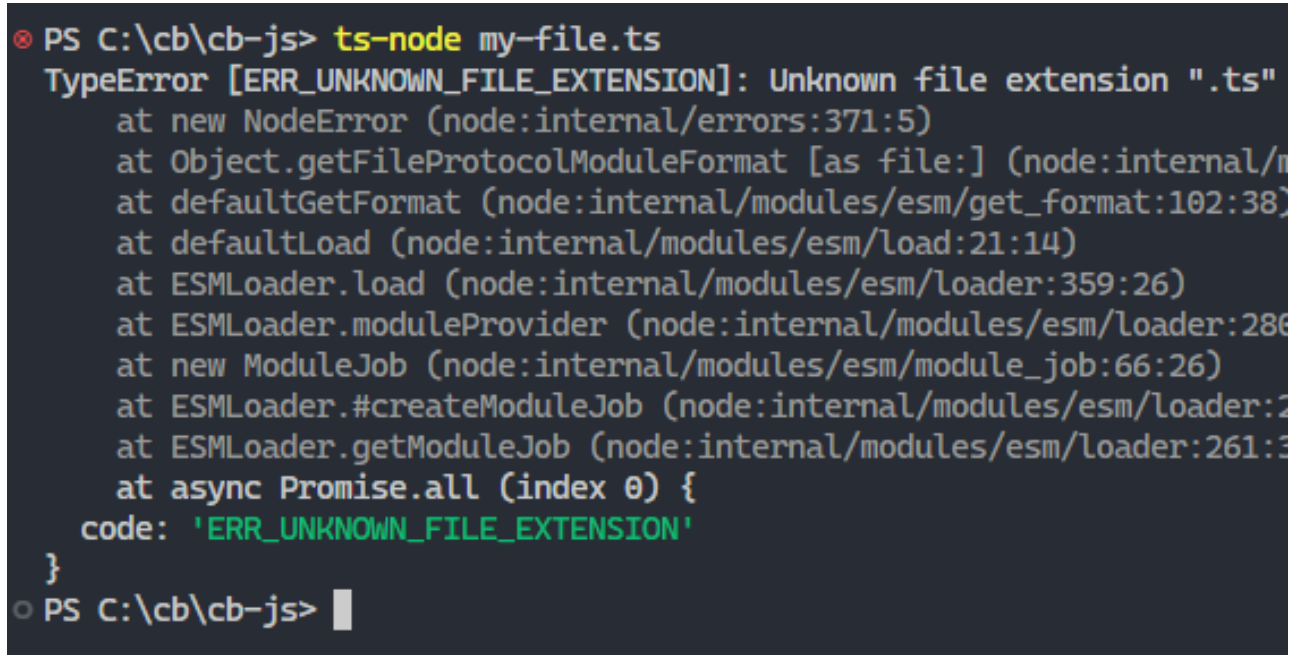
@specialize

```
// TypeScript

const [optionalString, setOptionalString] = useState<string | null>(null);
```

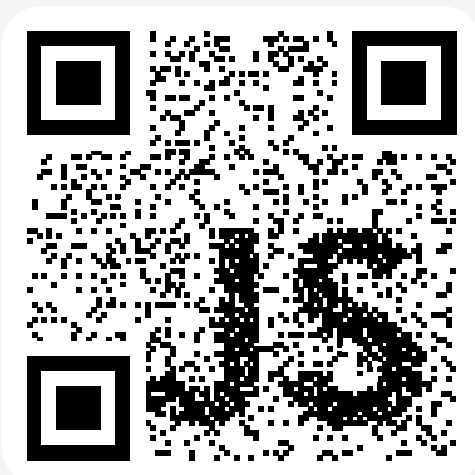

Theme: Dark
Joshua's Docs - JSDoc Cheatsheet and Type Safety Tricks
ABOUT
CURRENT PAGE
ENTIRE SITE
Scripts vs Modules, Block-Scoped Variables
How to ignore errors in multi-line blocks (e.g JSX)
VSCode Type-Safety JSDoc - Issues
Important reference material
Further reading
Sharing JSDoc Types Internally (Importing, Exporting, and Ambient Declarations)
Passing Typedefs Around
Passing Inferred Types Around
Passing Types around via TypeScript Files
Generating Files From JSDoc
Emitting Type Definitions / Declarations from JSDoc Types
Where to Output JSDoc Generated TS Declaration Files
Emitting a Single Declaration File
Caveats for Using JSDoc to Generate Declaration Files
How To Export Types from Index Entry Point
Issues
> Warning: A lot of this page assumes you are using VSCode as your IDE, which is important to note because not all IDE's handle JSDoc the same way. I also focus on how JSDoc can provide type-safe JS through TypeScript, as well as through VS Code's wrapper around TypeScript.
Table of Contents
Resources
Advanced Usage
Type Guards
Annotating destructured parameters
Annotating function signatures / function as a type
Classes and Constructors
Casting and Type Coercion in JSDoc
Simple Casting
Advanced Casting
Non-Null Assertion in JSDoc
Generics in JSDoc
Extending Interfaces
Tuple Types
Literal Types and Constants
String Literal Types

All of this to avoid…


PS C:\cb\cb-js> ts-node my-file.ts
TypeError [ERR_UNKNOWN_FILE_EXTENSION]: Unknown file extension ".ts"
at new NodeError (node:internal/errors:371:5)
at Object.getFileProtocolModuleFormat [as file:] (node:internal/
at defaultGetFormat (node:internal/modules/esm/get_format:102:38)
at defaultLoad (node:internal/modules/esm/load:21:14)
at ESMLoader.load (node:internal/modules/esm/loader:359:26)
at ESMLoader.moduleProvider (node:internal/modules/esm/loader:28
at new ModuleJob (node:internal/modules/esm/module_job:66:26)
at ESMLoader.#createModuleJob (node:internal/modules/esm/loader:
at ESMLoader.getModuleJob (node:internal/modules/esm/loader:261:
at async Promise.all (index 0) {
code: 'ERR_UNKNOWN_FILE_EXTENSION'
}
PS C:\cb\cb-js>

All of this to avoid…

```
                                    }
                                  }
                                }
                              }
                            }
                          // When I wrote this, only God and I understood what I was doing
                          // Now, God only knows
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      } catch {
        return null;
      }
    }
  }
}
```



https://github.com/MrBazlow/youmightnotneedtypescript