Ingo Steinke
Posted on 26 Nov 2021 • Updated on 21 Dec 2021

Using JSDoc to write better JavaScript Code

#javascript #webdev #programming #typescript

Using TypeScript consistently offers a lot of benefits, especially for writing clean code to prevent unnecessary bugs and errors. But it takes some time and effort to learn, and you have to adapt every code snippet after copy-pasting from a tutorial or a StackOverflow example. Quite a challenge when re-using code for React, Node, Express and Mongoose like I did in my full-stack web app side project.

Alternatives to TypeScript

- Popular, but unhelpful, alternative: don't care at all.
- Use eslint, but that's not an alternative. With or without strong typing, you should lint your code anyway to benefit from (mostly) helpful hints and warnings.
- ReactPropTypes add some type checking to ECMAScript / JavaScript in React applications, but PropTypes are merely footnotes, placed far away from where they would be most useful, while still bloating your code.

And there are no PropTypes in Vanilla JS.

- enter **JSDoc**:

JSDoc

Often overlooked, maybe never even heard of until now, JSDoc deserves more attention, as it brings a lot of advantages out of some short lines of documentation.

Code Documentation

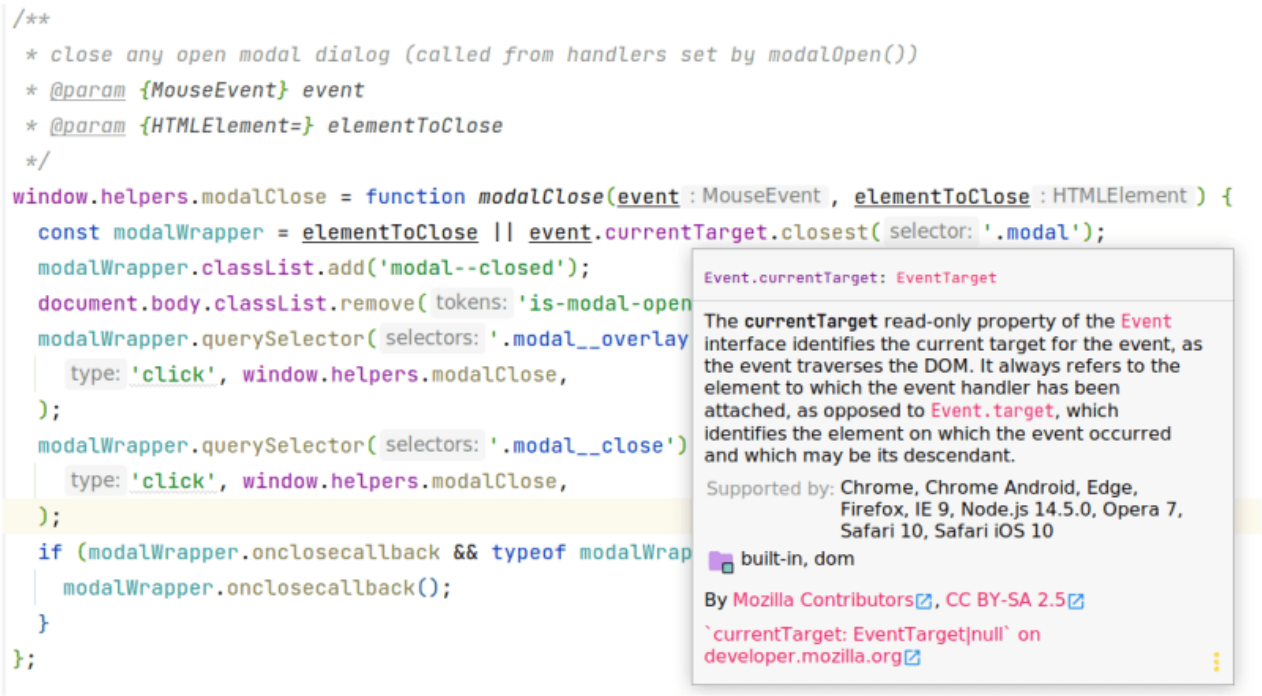
That's JSDoc's original purpose: generating a code / API documentation out of a few lines placed before variables, functions, and classes.

Similar approaches have been used with Java and PHP for a long time, and JSDoc follows established practice and is quite easy to learn.

Hints and Code Completion

Using JSDoc inside a modern IDE, you'll get another benefit: live code inspection, warnings, and proper code completion even for the most obscure DOM methods you never knew about before. Or well-known classics like `event.currentTarget` that still have some tricky pitfalls.

Here is a - seemingly simple - example:



I wanted to allow a modal dialog to be closed typing the Escape key. My first quick code-like-it's-1999-style script (not shown here) was frowned upon by eslint. ☹️

Writing Clean, Modern Vanilla JS Code

So I decided to write proper, modern code, but still plain "Vanilla JS" (that does not need a transpiler to produce working code, unlike TypeScript, which does not have native browser support, not even in Microsoft's Edge browser).

I wrote a function that takes two arguments: an event, and an optional DOM element so that we are able to close a specific modal dialog from outside without relying on the event context.

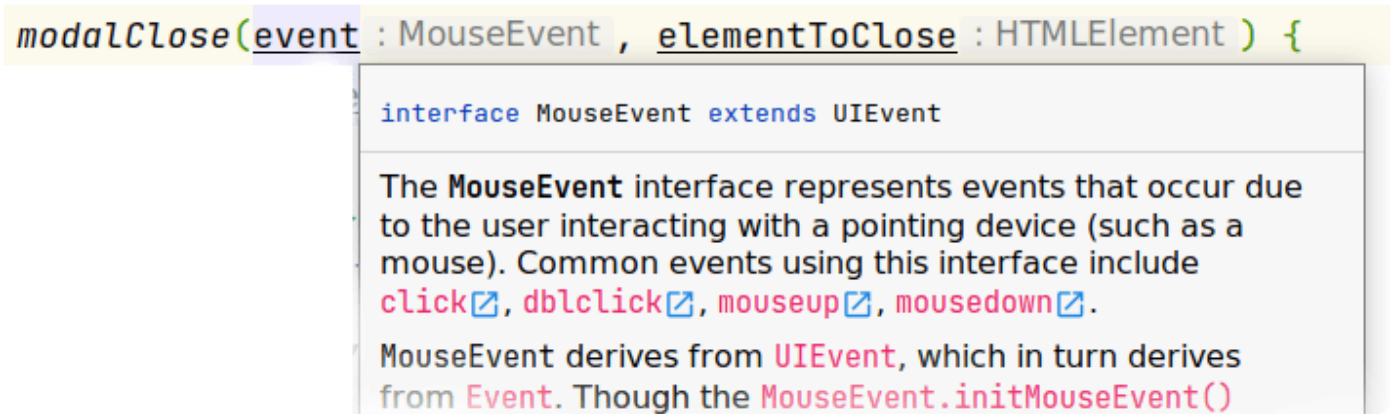
Adding a JSDoc comment before, it becomes

```
/**
 * close an open modal dialog
 * @param {MouseEvent} event
 * @param {HTMLElement=} elementToClose
 */
const modalClose = function modalClose(event, elementToClose) {
  // ...
};
```

telling my readers (of this code, and of a possible, automatically generated, documentation / API reference) what the function is supposed to do, and what arguments it expects:

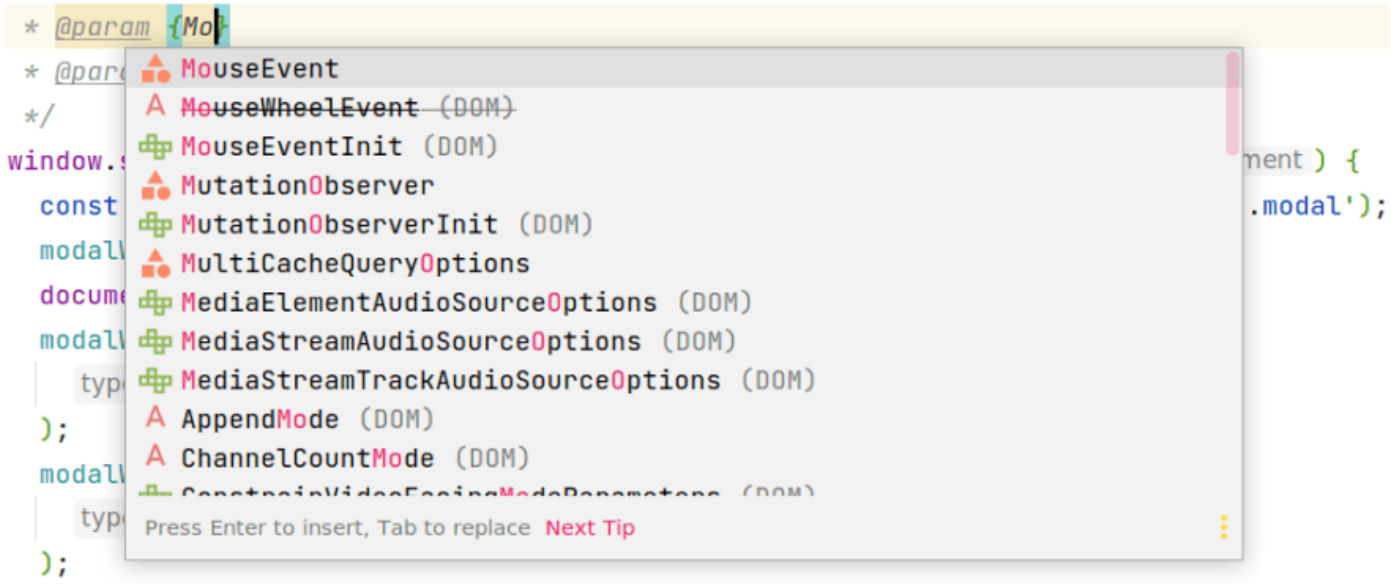
```
@param {MouseEvent} event
```

Now my IDE (PhpStorm) will show me helpful information:



I don't even have to look up the proper type string to write in the JSDoc comment!

When I start typing, PhpStorm has some suggestions for code completion even in this special kind of comment, suggesting `MouseEvent` on top of the list.



JSDoc Syntax

The basic syntax is rather simple.

Annotations blocks are special comments that start with a slash and a double asterisk `/**`

A parameter hint starts with an at sign, the word "param", and a type definition inside curly braces, followed by the parameter's name.

To mark an optional parameter, add an equals sign behind the type, like

```
@param {HTMLElement=} elementToClose
```

but to be more clear to human readers, we can also add anything behind the parameter's name, like

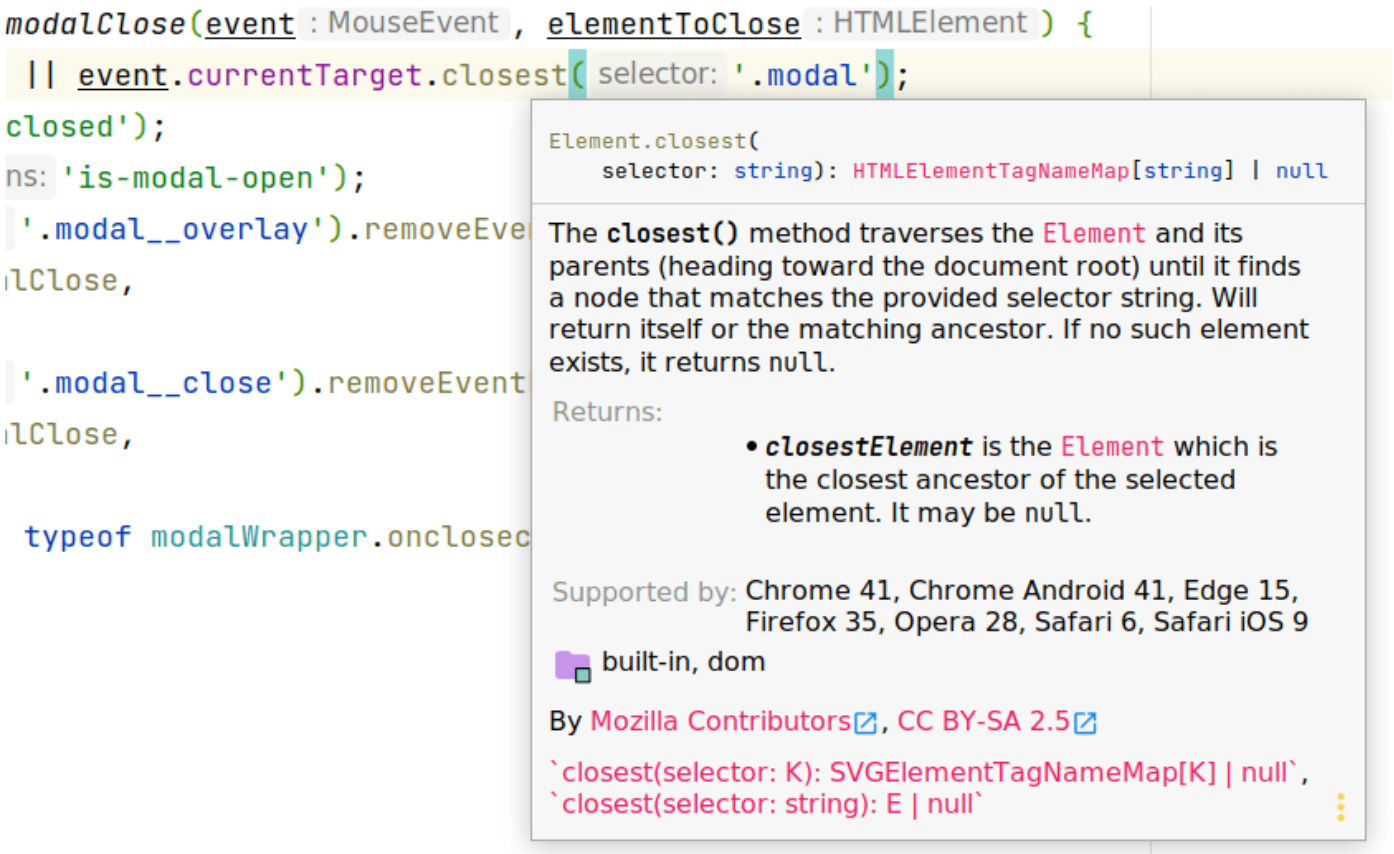
```
@param {HTMLElement=} elementToClose (optional) DOM element to receive .closed
```

Now my editor shows me type annotations, that are not part of my written code (unlike they would be in TypeScript) but rather implicitly follow from my code. So my actual code stays short and compact, while the implicit meaning is still more obvious than before.

Assistance for Lazy Developers

Not only do we see the additional hint `event: MouseEvent`, but when we start using the `event` in our code below, there are more code suggestions, helping us to choose methods and properties that are actually available and not deprecated.

More assistance as we continue: hints and documentation everywhere. We don't even have to visit MDN in our browser anymore!



Conclusion

JSDoc makes coding in JavaScript easier, helping us to code quickly while avoiding obvious errors, just by adding some lines of optional comments in our code.

Discussion (1)

Manuel Sommerhalder • Nov 28 '21

You get the type safety and autocompletion without using another language or changing the build pipeline with JSDocs. I really don't understand the hype for Typescript.

Ingo Steinke

Web Development, Sustainability, Art and Music, Nature and Travel, Sustainability

LOCATION
Germany

WORK
Creative Web Developer at Ingo Steinke

JOINED

21 Sep 2019

More from Ingo Steinke

Animated Gradient Text Color

#webdev #showdev #css #tutorial

Aspect ratio: no need for container units!

#css #html #webdev #todayilearned

CSS :has(.parent-selectors)

#css #webdev #todayilearned #javascript