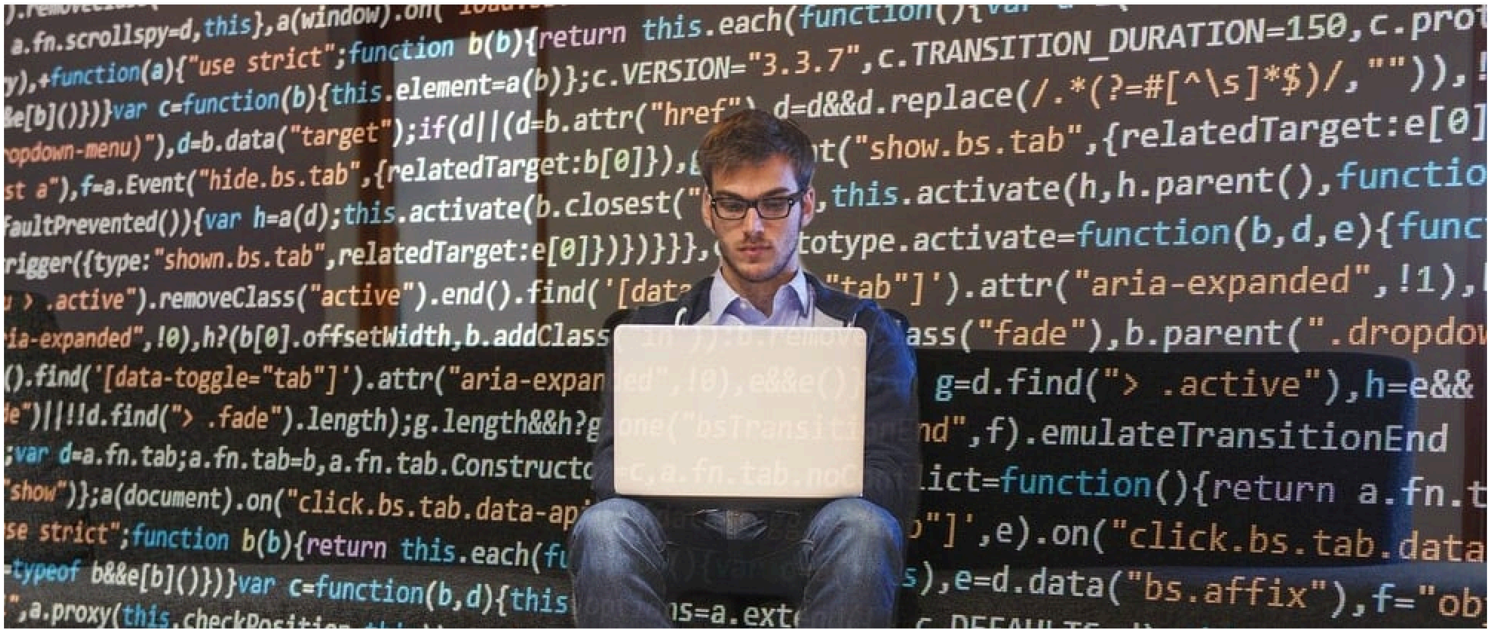


Bola Adebesin
Posted on Feb 23

❤️ 4 🦄 1 🤯 1 🙌 1 🔥 1

Writing Human Readable Code: To Name or Not To Name

#javascript #programming #webdev

I waffled back and forth about the title of this post, but in the end, I couldn't resist the Hamlet reference.

Today's topic is about writing clear, human-readable code. I care about this for a few reasons. When code is easier to read/understand, it's easier to:

- Maintain
- Refactor
- Discuss (with clients, colleagues, your rubber duck etc).

There are so many instances where I've come across old code that I don't understand.

The Developer
@vishal7arya

When your read some incredibly bad code , thinking “What moron wrote this...” , but halfway through it starts to become familiar.

This frustrating experience has ignited my interest in learning how to craft "better" code.

The good news is, there are a bunch of blog posts, forums, books, and articles about how to write better code. A few tips I've seen:

- Add comments to your code; Comments that explain "why" are better than comments that explain "what"
- Write tests
- Name your variables clearly
- The single responsibility principle: every function should do one job really well
- Avoid complicated conditionals
- Keep files to 250 lines of code or less

I coud go on, but there is one piece of advice that really struck me and I want to discuss it. The advice was: "Use traditional function declarations and opt for named functions over anonymous ones."

This advice caught my attention because my code is littered with anonymous functions. I can't count how many unnamed functions I've passed into `.map`, `.forEach`, and `.reduce` methods.

I don't think I'm alone in this. I've read time and time again that arrow functions and anonymous functions help to make code more concise and that this brevity makes for smaller files, which is part of writing better code, right? Well, it turns out that concise and anonymous does not necessarily mean better or more readable.

The more I think about this, the more it makes sense. "Well-named variables" is one of the bullet items for achieving clearer code. Variables with clear names help us to understand the data contained within them. This should also be true for functions which not only contain data, but often manipulate that data in some way too.

Let's look at a small example pulled from the Frontendmasters course, "Deep JavaScript Foundations V3" by Kyle Simpson, author of the "You don't know JS" series.

I am going to share two different implementations I wrote for a function called `printRecords`. One uses named functions and the other does not.

The function, `printRecords` should:

- Accept an array of numbers (these numbers represent student Ids)
- Retrieve each student record using the student Id
- Sort these records alphabetically by the students' names
- Print each record to the console in the following format: `Name (student Id): Paid (or Not Paid)`

```
var currentEnrollment = [410, 105, 664, 375]
var studentRecords = [
    { id: 313, name: "Frank", paid: true, },
    { id: 410, name: "Suzy", paid: true, },
    { id: 709, name: "Brian", paid: false, },
    { id: 105, name: "Henry", paid: false, },
    { id: 502, name: "Mary", paid: true, },
    { id: 664, name: "Bob", paid: false, },
    { id: 250, name: "Peter", paid: true, },
    { id: 375, name: "Sarah", paid: true, },
    { id: 867, name: "Greg", paid: false, },
];
```

```
// Named Functions Version:
function printRecords(recordIds){
  function findStudentRecords(recordId){
     function getStudentRecordById(record){
       return recordId == record.id;
     }
     return studentRecords.find(getStudentRecordbyId);
  }
  function alphabetizeRecordsByName(a,b){
    return a.name > b.name ? 1: -1;
  }
  function printEachRecord(record){
    console.log(
      `${record.name} (${record.id}): ${record.paid ? 'Paid': 'Not Paid'}`
     );
  }
  recordIds.map(findStudentRecords)
  .sort(alphabetizeRecordsByName)
  .forEach(printEachRecord);
}

//Anonymous Functions Version:

function printRecords(recordIds){
    recordIds
        .map((recordId) =>
            studentRecords.find((studentRecord) => studentRecord.id == recordId
        )
        .sort((a,b) => a.name > b.name ? 1: -1)
        .forEach((student) => {
            console.log(
                `${student.name} (${student.id}): ${student.paid ? 'Paid': 'Not
            )
        })
}


printRecords(currentEnrollment);


// Console:
/*
 Bob (664): Not Paid
 Henry (105): Not Paid
 Sarah (375): Paid
```

```
  Suzy (410): Paid
*/
```

Looking at the two implementaions side-by-side in my editor, I can see that the named version of `printRecords` spans lines 1-23, while the anonymous version spans lines 1-12. Some of this is a result of the automated code formatter I use, but right away it's clear that the anonymous version is more concise.

Still, when I look at the named version, I understand what each function in the `.map`, `.sort`, and `.forEach` method is doing. I don't have to read the code line by line. It's all right there in the names: find the student records, sort them alphabetically by name, print each record. By contrast, the anonymous version requires me to look at the code in the body of each function to understand what is happening.

In this case, I would rather take the extra 11 lines of code for a better experience modifying this code down the road.

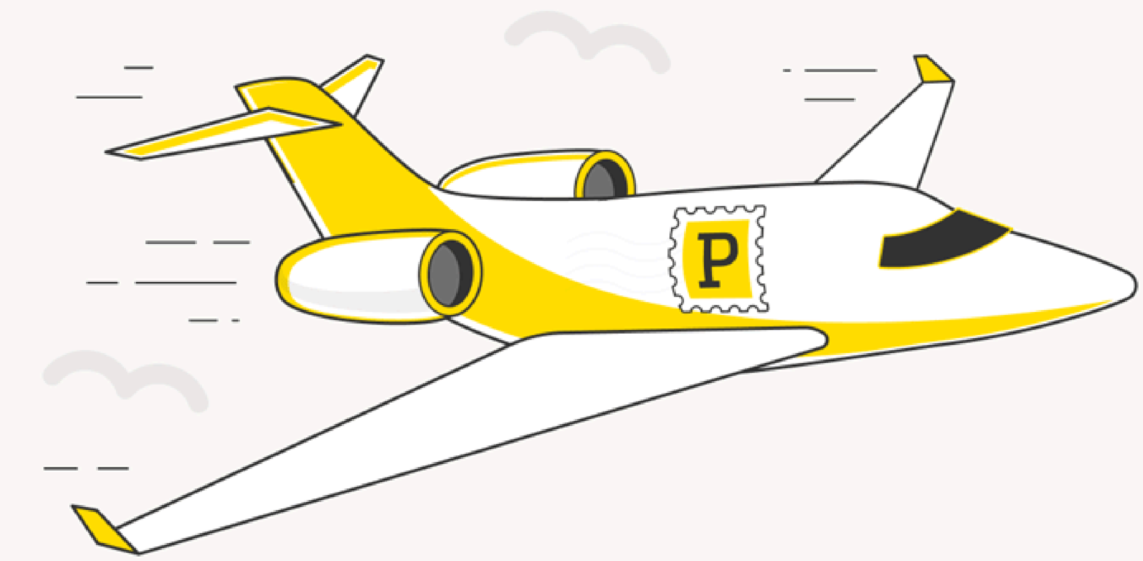
In his lecture, Kyle Simpson posits that there are at least three reasons to choose named functions over anonymous ones:

1. Reliable self-reference (e.g. If a function has a name, it can do things like call itself for a recursive solution)
2. More debuggable Stack Trace (if the function has a name, the stack trace will provide it)
3. More self documenting code

What do you think? Would you give up anonymous functions if it meant more human readable code? Do you use the number of lines of code as an indicator of whether a piece of code needs to be rewritten? How do ensure your code is readable 2 months down the road and beyond?

** Image by Tumisu from Pixabay

ActiveCampaign >
Postmark



Bootstrapping your startup?

Get $75 to help with your

Read More

Top comments (6)

Paco Luna • Feb 24

As a matter of fact, you can use named functions to create a very useful library to your project. Just use anonymous functions that you certainly know that are very simple or one in a million in your project.

Bola Adebesin • Feb 24

@devpacoluna Thanks for your comment! I hadn't thought of using named functions for building a library, is that something you've done?

Paco Luna • Feb 24

yep, I work with react. So i create a folder `lib/` and each file has some functions with the same purpose example date.ts, car.ts, etc...

Ben Sinclair • Mar 4

I like this and agree mostly, but I think the examples you've given add to the confusion rather than making it clearer:

Reading the following, I'd be confused expecting `getStudentRecordById` to take an ID as a parameter and to return a `record` object, when in fact it takes a record object and returns a boolean. That's suitable for use in the `find` method, but it doesn't make sense in itself:

```
function findStudentRecords(recordId){
   function getStudentRecordById(record){
     return recordId == record.id;
   }
   return studentRecords.find(getStudentRecordbyId);
}
```

If I was reading this one, `printEachRecord` would imply to me that it was a function which took some kind of iterable, rather than a single record:

```
function printEachRecord(record){
  console.log(
    `${record.name} (${record.id}): ${record.paid ? 'Paid': 'Not Paid'
    );
  }
```

David • Feb 24

As a software engineer with decades of experience I found the anonymous function version easier to read. I also point out that English has a problems with being imprecise and having different meanings for different people in different contexts. I do appreciate good variable names and comments about why, but somehow shorter is quicker and easier to digest.

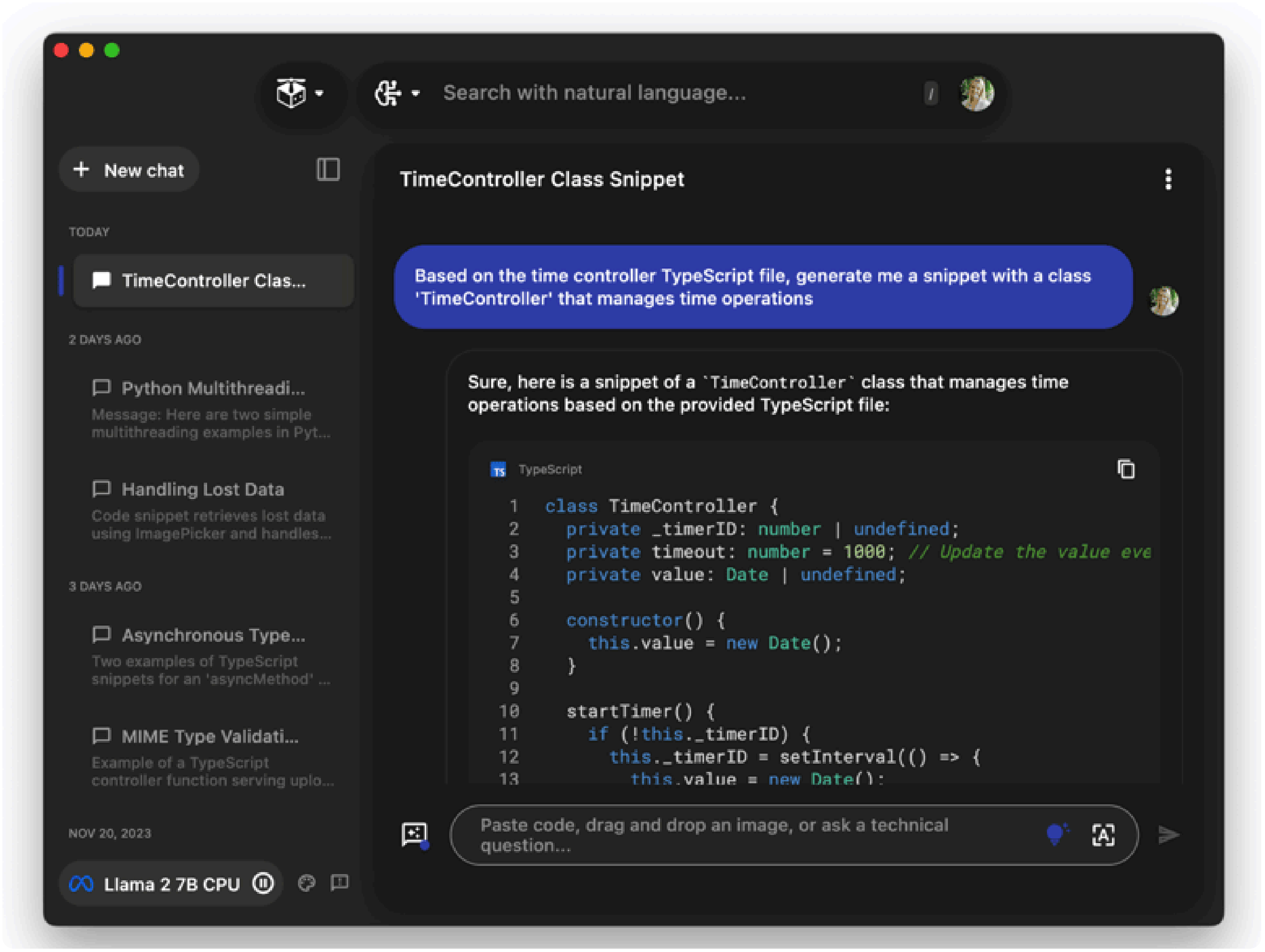
Bola Adebesin • Feb 24

@ddfridley Thanks for your comment! You're right, English can be imprecise, and I hadn't considered the confusion that might arise from that. Maybe it's be worth it to let the code, speak for itself.

Code of Conduct • Report abuse

Pieces.app PROMOTED

A Workflow Copilot. Tailored to You.


Search with natural language...
New chat
TODAY
TimeController Clas...
2 DAYS AGO
Python Multithreadi...
Message: Here are two simple multithreading examples in Pyt...
Handling Lost Data
Code snippet retrieves lost data using ImagePicker and handles...
3 DAYS AGO
Asynchronous Type...
Two examples of TypeScript snippets for an 'asyncMethod' ...
MIME Type Validati...
Example of a TypeScript controller function serving uplo...
NOV 20, 2023
Llama 2 7B CPU
Paste code, drag and drop an image, or ask a technical question...

TimeController Class Snippet

Based on the time controller TypeScript file, generate me a snippet with a class 'TimeController' that manages time operations

Sure, here is a snippet of a `TimeController` class that manages time operations based on the provided TypeScript file:

TypeScript

```
class TimeController {
  private _timerID: number | undefined;
  private timeout: number = 1000; // Update the value eve
  private value: Date | undefined;

  constructor() {
    this.value = new Date();
  }

  startTimer() {
    if (!this._timerID) {
      this._timerID = setInterval(() => {
        this.value = new Date();
```

Our desktop app, with its intelligent copilot, streamlines coding by generating snippets, extracting code from screenshots, and accelerating problem-solving.

Read the docs

Bola Adebesin

While my approach to programming is both analytical and methodical, my passion for technology and discovery drives me to pursue creative solutions.

LOCATION
Baltimore, MD

WORK
Booz Allen Hamilton

JOINED

Nov 30, 2020

More from Bola Adebesin

JavaScript Types & Coercion Corner Cases

#webdev #programming #javascript

DEV The DEV Team PROMOTED ...

Check out DEV++