TIMELESS SOFTWARE WISDOM

TWO YEARS OF LINKS FROM MY WEEKLY PINKLETTER

BY RICCARDO ODONE

Contents

Elm

Shaping Values with Types

by Josh Clayton

While there are only 110,000 valid four- and five-digit employee IDs, our data model for an employee ID uses the underlying type of String, which can represent an infinite number of values. Our data model does not reflect reality. By reducing the number of possible values captured in a type, it's less likely that an incorrect value sneaks in.

Running Out of Maps

by Joël Quenneville

Many Elm packages provide map2, map3, map4, etc functions. No matter how many of these the package author has provided, inevitably someone will end up needing a mapN larger than those included in the package. Perhaps that "someone" is you. How do you deal with a situation where you run out of maps?

Tail-Call Elimination

by Evan Czaplicki

First, a common technique for writing recursive functions is to create a function foo with the API you want and a helper function fooHelp that has a couple additional arguments for accumulating some result. This definitely can come in handy if you are working on some more advanced algorithms!

Second, it turns out that "Is recursion going to crash my program?" is not a practical concern in functional languages like Elm. With linear structures (like List) we can do tricks to take advantage of tail-call elimination. With branching structures (like Dict, Set, and Array) you are likely to run out of disk space before you run out of stack space. Those are the only two kinds of structures we have!

Performant Elm

by Ju Liu

You have written your first application in Elm. Congratulations my friend, the hard work has finally paid off. Now it's time to relax and enjoy the pure bliss that is maintaining and refactoring Elm code.

But wait a second, someone tells you the app feels a bit sluggish to use. Mh. I thought that writing pure functions would automatically make code fast. Okay. Let's try to stay calm.

elm-graphql

by Elm Radio Podcast

Elm Radio: Parse, Don't Validate

by Elm Radio Podcast

Making Impossible States Impossible

by Richard Feldman

Among the most time-consuming bugs to track down are the ones where we look at our application state and say "this shouldn't be possible."

We can use Elm's compiler to rule out many of these bugs in the first place—but only if we design our Models using the right techniques! This talk explores how.

Opaque Types Let You Think Locally

by Dillon Kearns

Elm's Opaque Types are a powerful tool for narrowing the surface area where you check a constraint. TypeScript's Branded Types give similar functionality but without closing outside use, so you can't be sure the constraints are enforced everywhere.

Haskell / PureScript

Elementary programming

by Michael Peyton Jones

So we have a dilemma: abstraction is great for the advanced users, but is inevitably going to make your code hard for less advanced users to understand. And if they can't understand your code, they certainly aren't going to be able to maintain it.

Getting acquainted with Lens

by Paweł Szulc

Programming Without Type Classes

by Berlin Functional Programming Group

Type classes have become a cornerstone of statically-typed functional programming, powering abstractions like monoid and monad. Yet, type classes often have generalized names, which don't reflect their purpose in specific domains; and they incur higher learning costs, especially when emulated in languages without them.

The Misunderstood Roots of FRP Can Save Programming

by Steve Krouse

For many years I been searching for the perfect paradigm for programming user interfaces. Like many others, I fell in love with FRP with the rise of ReactJS and spent a few years searching for the perfect reactive model. Eventually, I found my way back to the original work on FRP by Conal Elliott. It took me almost a year to make sense of it.

This essay attempts to make Conal's vision more understandable to less mathematically-oriented programmers, and also show how this perspective could be the foundation for a new era of programming, not just with user interfaces, but also multi-node computing, storage, machine learning, etc.

Anamorphisms aka Unfolds Explained

by Marty Stumpf

Unfolds can be thought of as the dual of folds. As Conal Elliott puts it: while folds contract a structure down to a value, unfolds expand a structure up from a value!

Denotational Design: From Meanings To Programs

by Conal Elliot

I'll share a methodology that I have applied many times over the last 20+ years when designing high-level libraries for functional programming. Functional libraries are usually organized around small collections of domain-specific data types together with operations for forming and combining values of those types. When done well, the result has the elegance and precision of algebra on numbers while capturing much larger and more interesting ideas.

Bind The Gap

by Kowainik

Bind The Gap is a monthly magazine about Haskell and Functional Programming in general.

It discusses hot news in the Haskell ecosystem, describes recent events, educates about FP concepts, and provides a view on diverse language areas from experienced Haskell users and active community members.

(((Wait a moment .) .) .) - Composing Functions with Multiple Arguments

by Ubik

The goal of the point-free style is the same as all other forms of abstraction: to express your intention in a concise (to write), clear (to read), and general (to change) way.

Composing predicates

by Christian Gill

Note that by compose I mean not function composition but to combine two predicates together to get a new one. Either by conjunction (&&) or disjunction (||).

PureScript by Example

by Phil Freeman

PureScript is a small strongly, statically typed programming language with expressive types, written in and inspired by Haskell, and compiling to Javascript.

Functional programming in JavaScript has seen quite a lot of popularity recently, but large-scale application development is hindered by the lack of a disciplined environment in which to write code. PureScript aims to solve that problem by bringing the power of strongly-typed functional programming to the world of JavaScript development.

This book will show you how to get started with the PureScript programming language, from the basics (setting up a development environment) to the advanced.

Tying Shoes with GADTs

by MorrowM

Do you put them both on and then tie them both? In what order? Do you put on one shoe, tie it and then do the other one? That might be a bit odd, but it's acceptable. Anyone who ties either of their shoes before putting them on is a no-go, though. And since my Haskell file is my world, and GHC is my enforcer, let's make this state of affairs unrepresentable.

Simpler And Safer API Design Using GADTs

by Chris Penner

A lot of the writing out there regarding GADTs is pretty high-level research and academia, in contrast, today I'm going to show off a relatively practical and simple use-case. In this post we'll take a look at a very real example where we can leveraged GADTs in a real-world Haskell library to build a simple and expressive end-user interface.

Free Course On Functional Programming in Haskell

by Nikos Vaggalis

Videos from an introductory course by Professor Graham Hutton from the University of Nottingham have been made freely available on YouTube. Designed for first year Computer Science students, they teach the basic principles of functional programming using Haskell.

Refactoring Recursion

by Harold Carr

Recursion is the fundamental looping mechanism in functional programming. This talk shows patterns of recursion using Haskell. It shows those patterns for list structure only. This makes it easier for beginners to understand recursion schemes by focusing on their operation with lists. We start by writing explicit recursive versions of sum, product, and length of lists, then factor them into fold functions. We proceed in a similar manner with other folds, unfolds, and refolds with many examples of the patterns in operation. We end by mentioning factoring recursion out of data.

Zippers—Learn You a Haskell for Great Good!

by Miran Lipovaca

One thing we can do is to remember a path from the root of the tree to the element that we want to change. We could say, take this tree, go left, go right and then left again and change the element that's there. While this works, it can be inefficient.

If we want to later change an element that's near the element that we previously changed, we have to walk all the way from the root of the tree to our element again!

In this chapter, we'll see how we can take some data structure and focus on a part of it in a way that makes changing its elements easy and walking around it efficient. Nice!

Time travel in Haskell for dummies

by Csongor Kiss

Browsing Hackage the other day, I came across the Tardis Monad. Reading its description, it turns out that the Tardis monad is capable of sending state back in time. Yep. Back in time.

Names are not type safety

by Alexis King

The Haskell school of program construction advocates "capturing invariants in the type system" and "making illegal states unrepresentable," both of which sound like compelling goals, but are rather vague on the techniques used to achieve them.

Parse, don't validate

by Alexis King

Now I have a single, snappy slogan that encapsulates what type-driven design means to me, and better yet, it's only three words long: Parse, don't validate.

Types as axioms, or: playing god with static types

by Alexis King

The real point of this blog post is that a type system does not have a well-defined list of things it "can prove" and "can't prove." Languages like TypeScript don't

really encourage this approach to data modeling, where you restructure your values in a certain way so as to guarantee certain properties. Rather, they prefer to add increasingly sophisticated constraints and type system features that can capture the properties people want to capture without having to change their data representation.

Haskell for the Elm Enthusiast

by NoRedInk

Rails has served us well and has supported amazing growth of our website, both in terms of the features it supports, and the number of students and teachers who use it. But we've come to miss some of the tools that make us so productive in Elm: Tools like custom types for modeling data, or the type checker and its helpful error messages, or the ease of writing (fast) tests.

A couple of years ago we started looking into Haskell as an alternative backend language that could bring to our backend some of the benefits we experience writing Elm in the frontend. Today some key parts of our backend code are written in Haskell. Over the years we've developed our style of writing Haskell, which can be described as very Elm-like (it's also still changing!).

Parser Combinators: a Walkthrough

by Antoine Leblanc

Today, I want to explore Parsec, and most specifically how Parsec works. Parsing is ubiquitous, and most Haskell programs will use Parsec or one of its variants (megaparsec or attoparsec). While it's possible to use those libraries without caring about how they work, I think it's fascinating to develop a mental model for their inner design; namely, monadic parser combinators, as it is a simple and elegant technique that is useful beyond the use case of parsing raw text.

Generalizing 'Jq' And Traversal Systems Using Optics And Standard Monads

by Chris Penner

Hi folks! Today I'll be chatting about Traversal Systems like jq and XPath; we're going to discover which properties make them useful, then see how we can replicate their most useful behaviours in Haskell.

JSON Parsing from Scratch in Haskell: Error Reporting

by Abhinav Sarkar

In the previous post we wrote a simple but correct JSON parser in Haskell. The parser was written very naively: if it failed, it returned nothing. You couldn't tell what the failure was or where it happened. That's OK for a toy parser but error reporting is an absolute must requirement for all good parsers. So in this post and next post, we'll add simple but useful error reporting capability to our JSON parser.

An introduction to typeclass metaprogramming

by Alexis King

Typeclass metaprogramming is a powerful technique available to Haskell programmers to automatically generate term-level code from static type information. It has been used to great effect in several popular Haskell libraries (such as the servant ecosystem), and it is the core mechanism used to implement generic programming via GHC generics. Despite this, remarkably little material exists that explains the technique, relegating it to folk knowledge known only to advanced Haskell programmers.

JavaScript / TypeScript

Statically Prevent 404s

by Gary Bernhardt

Most web applications have route patterns like "/courses/:courseId". Then they link to the routes by manually building path strings. But there's no guarantee that links actually point to valid routes. When we change or delete routes, we might forget to update some of the links, turning them into 404s.

PrinceJS

by Oliver Klemenz

Prince of Persia reimplementation written in HTML5 / JavaScript (MS-DOS version).

Hasty

by Mads B. Cordes

JavaScript snippet performance comparison tool

Beyond Console.log() – Level up Your Debugging Skills

by Chris Heilmann

You may have established a pattern of coding that utilizes a few key tools offered by your browser's console. But have you dug any deeper lately? There are some powerful tools available to you, and they might just revolutionize the way you work.

How To Build Resilient JavaScript UIs

by Callum Hart

Embracing the fragility of the web empowers us to build UIs capable of adapting to the functionality they can offer, whilst still providing value to users. This article explores how graceful degradation, defensive coding, observability, and a healthy attitude towards failures better equips us before, during, and after an error occurs.

Dynamic Static Typing In TypeScript

by Stefan Baumgartner

In this article, we look at some of the more advanced features of TypeScript, like union types, conditional types, template literal types, and generics. We want to formalize the most dynamic JavaScript behavior in a way that we can catch most bugs before they happen.

Strict null checking Visual Studio Code

by Matt Bierner

In this post, I'd like to share a major engineering effort that the VS Code team recently completed: enabling TypeScript's strict null checking in our codebase. We believe this work will allow us to both move faster and to ship a more stable product. Enabling strict null checking was motivated by understanding bugs not as isolated events but as symptoms of larger hazards in our source code.

Using strict null checking as a case study, I'm going to discuss what motivated our work, how we came up with an incremental approach to addressing the problem, and how we went about implementing the fix. This same general approach to identifying and reducing hazards can be applied to any software project.

The TypeScript Handbook

by Microsoft

The TypeScript Handbook is intended to be a comprehensive document that explains TypeScript to everyday programmers. You can read the handbook by going

from top to bottom in the left-hand navigation.

End-to-End TypeScript: Database, Backend, API, and Frontend

by Gary Bernhardt

This is a walk-through of a real commercial system written in TypeScript. Static types are used to ensure that the backend code uses the database's data correctly, that the backend sends the correct data over the API, and that the frontend consumes API data correctly.

Optional Chaining: The ?. Operator in TypeScript

by Marius Schulz

In this post, I will go over the following three optional chaining operators and explain why we might want to use them in our TypeScript or JavaScript code.

3 Useful TypeScript Patterns to Keep in Your Back Pocket

by Casey Falkowski

Using TypeScript with its most basic types alone adds plenty of safety to your code. Sometimes, however, you come across situations where you need a little bit more. Here are three useful TypeScript patterns that will kick your game up a notch.

10 bad TypeScript habits to break this year

by Daniel Bartholomae

TypeScript and JavaScript have steadily evolved over the last years, and some of the habits we built over the last decades have become obsolete. Some might never have been meaningful. Here's a list of 10 habits that we all should break.

Getting started with fp-ts

by Giulio Canti

In this blog series I will often talk about “type classes” and “instances”, let’s see what they are and how they are encoded in fp-ts.

Practical Guide to Fp-ts

by Ryan Lee

This post is an introduction to fp-ts, a functional programming library for Typescript. Why should you be learning fp-ts? The first reason is better type safety. Fp-ts allows you to make assertions about your data structures without writing user-defined type guards or using the as operator. The second reason is expressiveness and readability. Fp-ts gives you the tools necessary to elegantly model a sequence of operations that can fail. All in all, you should add fp-ts to your repertoire of tools because it will help you write better Typescript programs.

React

A Look at React Hooks

by Victoria Lo

In this series, we shall look at various React Hooks and learn how we can use them to implement in our React projects!

Useful React Hooks That You Can Use In Your Projects

by Ifeanyi Dike

We'll go through several code examples of each hook and also explore how you'd create a custom hook.

Tao of React - Software Design, Architecture & Best Practices

by Alex Kondov

While there are best practices on the micro level, most teams build their own "thing" when it comes to architecture.

Guideline from the 70's on how to split your React components

by João Forja

Deciding how to break a component into sub-components isn't easy and is a challenge that isn't specific to React. This is fortunate since it means we can go outside React's ecosystem and get some ideas on how to do it.

In this article, I'll present a guideline to validate ideas on splitting a React component to increase code reuse and reduce maintenance costs. This guideline comes from the paper "Designing Software for Ease of Extension and Contraction" written in 1979 by David Parnas.

The Algebraic Structure of Functions, Illustrated Using React Components

by James Sinclair

Let me be clear. I'm not suggesting anyone go and write all their React components using compose, map, and chain. I'm not even suggesting anyone include a Func library in their codebase. What I am hoping is that this gives you some tools to think differently about your React code. I'm also hoping that the algebraic structure of functions makes a little more sense now.

Algebraic effects, Fibers, Coroutines Oh my!

by Brandon Dail

React Fiber was a full re-write of React that will enable new and exciting patterns around control flow, which we've seen previewed with React Suspense.

But what is a fiber? How does it relate to a coroutine? What are algebraic effects, and why do I keep hearing about them?

This talk will go over these computer science topics in the context of React Fiber, to help shed some light on how React Fiber is implemented and the control flow concepts behind the new APIs.

SHOCKED BY THE QUALITY?!

SUBSCRIBE TO MY PINKLETTER

WANT TO SUPPORT MY WORK?

BUY ME A COFFEE (OR TWO)

Ruby

How to Make a Gem of a Gem

by Justin Searls

This presentation from RubyConf 2021 will show you every single step involved in creating and releasing a brand new gem—all in the first 8 minutes! The other 22 minutes will distill a decade of hard-fought lessons about how not to make a gem to help you ensure your next gem is a great one.

Hotwire - HTML Over The Wire

by Basecamp

Hotwire is an alternative approach to building modern web applications without using much JavaScript by sending HTML instead of JSON over the wire. This makes for fast first-load pages, keeps template rendering on the server, and allows for a simpler, more productive development experience in any programming language, without sacrificing any of the speed or responsiveness associated with a traditional single-page application.

Building HEY with Hotwire

by Full Stack Radio

In this episode Adam talks to DHH about using Hotwire to develop Basecamp's new email service HEY while shipping only 40kb of JavaScript to the client.

What is a reduction and why Fibers are the answer for Ruby concurrency

by Julik Live

In my view, for Ruby web apps to be viable concurrency is vastly more beneficial than parallelism. Ractors are, at this stage, problematic with most existing Ruby

programs. The fiber scheduling system added to Ruby 3.x is going to be extremely important to the story of Ruby on the server (and of UIs, if that ever reemerges).

Surgical Refactors

by Justin Searls

Is there anything we can do to make legacy Ruby more maintainable?

That question led me to this talk. In it, I introduces a new gem that we designed to help wrangle legacy refactors. It's called Suture, and along with providing some interesting functionality to make refactoring less mysterious and scary, it also prescribes a clear, careful, and repeatable workflow for increasing our confidence when changing legacy code.

Rust

Type-Driven API Design in Rust

by Will Crichton

In this talk, I will live-code the design of a simple Rust API. Through the evolution of the API, I will demonstrate how Rust's type system (especially traits) can be used to design interfaces that cleanly compose with existing code, and that help API clients catch mistakes at compile-time.

Take your first steps with Rust

by Microsoft

Interested in learning a new programming language that's growing in use and popularity? Start here! Lay the foundation of knowledge you need to build fast and effective programs in Rust.

In this learning path, you'll:

- Install the tools you need to write your first lines of Rust code.
- Learn basic concepts in Rust.
- Learn how to handle errors.
- Manage memory in Rust.
- Use generic types and traits.
- Set up modules for packages and crates.
- Write and run automated tests.
- Create a command-line program.

The Importance of Not Over-Optimizing in Rust

by Lily Mara

In this talk I will show that these optimizations are often unnecessary to get performance that beats highly dynamic languages like Python. For new Rust developers, breaking the temptation to over-optimize can lead to higher productivity and satisfaction with Rust.

Sql / Postgres

Why is it hard to automatically suggest what index to create?

by Depesz

Should I put it in index? Or not? What will I do, if it has 20% selectivity, but together with condition on d it will drop to 2%? Should I index it on (a,d)? (d,a)? (a) where d = 'done'?

Some Indexing Best Practices

by Michael Christofides

We all know that indexing is important, but it can be difficult to know where to start. In this post, my aim is to collate some of the best advice I've seen for indexing Postgres, and simplify it for folks just getting started.

Exploring PL/pgSQL: Strings, arrays, recursion, and parsing JSON

by Phil Eaton

PostgreSQL comes with a builtin imperative programming language called PL/pgSQL. I used to think this language was scary because it has a bit more adornment than your usual language does. But looking deeper, it's actually reasonably pleasant to program in.

Building Robust Systems with ACID and Constraints

by Brandur

In the last decade we've seen the emergence of a number of new flavors of data store that come with untraditional features like streaming changesets, JavaScript APIs, or nestable JSON documents. Most of them assume that the need for horizontal

partitioning is a given in this day and age and therefore ACID is put to the altar (this doesn't necessarily have to be the case, see below). Every decision comes with trade-offs, but trading away these powerful guarantees for the novelties du jour or an unexamined assumption that horizontal scaling will very soon be a critically required feature is as raw of a deal as you'll ever see in the technical world.

Using checksums to verify syncing 100M database records

by Simon Eskildsen

In this issue of napkin math, we look at implementing a solution to check whether A and B are in sync for 100M records in a few seconds. The key idea is to checksum an indexed updated_at column and use a binary search to drill down to the mismatching records.

Draw Entity-Relationship Diagrams, Painlessly

by holistics.io

A free, simple tool to draw ER diagrams by just writing code. Designed for developers and data analysts.

Speeding up SQL queries by orders of magnitude using UNION

by Ben Levy and Christian Charukiewicz

One of the most common cases where SQL query performance can degrade significantly is in a diamond shaped schema, where there are multiple ways of joining two tables together. In such a schema, a query is likely to use OR to join tables in more than one way, which eliminates the optimizer's ability to create an efficient query plan.

Advanced SQL course | SQL tutorial advanced

by Carnagie Mellon Database Group

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system. In this advance #SQL course you will learn how to execute complex queries.

Cut Out the Middle Tier: Generating JSON Directly from Postgres

by Paul Ramsey

PostgreSQL has built-in JSON generators that can be used to create structured JSON output right in the database, upping performance and radically simplifying web tiers.

PostgreSQL: Detecting Slow Queries Quickly

by Hans-Jürgen Schönig

I believe the best and most efficient way to detect performance problems is to make use of pg_stat_statement, which is an excellent extension shipped with PostgreSQL and is used to inspect query statistics in general. It helps you to instantly figure out which queries cause bad performance and how often they are executed.

Finding the Root Cause of Slow Postgres Queries Using EXPLAIN

by pganalyze

This situation might look familiar: Our application is slow, but we don't know why. We look at our monitoring dashboards and finally arrive at the center of our problem:

A Postgres query that is taking longer than it should.

Now where do we go from here? How can we understand better what our database does when it executes our slow query to retrieve our data?

Postgresql: How to write a trigger

by Hans-Jürgen Schönig

Just like in most databases, in PostgreSQL a trigger is a way to automatically respond to events. Maybe you want to run a function if data is inserted into a table. Maybe you want to audit the deletion of data, or simply respond to some UPDATE statement. That is exactly what a trigger is good for. This post is a general introduction to triggers in PostgreSQL. It is meant to be a simple tutorial for people who want to get started programming them.

Forget SQL vs NoSQL - Get the Best of Both Worlds with JSON in PostgreSQL

by Derek Xiao

Have you ever started a project and asked - "should I use a SQL or NoSQL database?"

It's a big decision. There are multiple horror stories of developers choosing a NoSQL database and later regretting it.

But now you can get the best of both worlds with JSON in PostgreSQL.

In this article I cover the benefits of using JSON, anti-patterns to avoid, and an example of how to use JSON in Postgres.

Using ON Versus WHERE Clauses to Combine and Filter Data in PostgreSQL Joins

by Jacek Trociński

In an SQL query, data can be filtered in the WHERE clause or the ON clause of a join. This guide will examine the difference between the two in PostgreSQL.

A simple example of LATERAL use

by Luca Ferrari

A few days ago I found a question by a user on Facebook: how to select events from a table where they are no more than 10 minutes one from another? My first answer was related to LATERAL, and this post I try to represent with an example how I understood and could solve the above question.

Saving a Tree in Postgres Using LTREE

by Pat Shaughnessy

LTREE allows me to save, query on and manipulate trees or hierarchical data structures using a relational database table. As we'll see, using LTREE I can count leaves, cut off branches, and climb up and down trees easily - all using SQL right inside my application's existing Postgres database!

Explaining Your Postgres Query Performance

by Kat Batuigas

In a previous post, I talked about pg_stat_statements as a tool for helping direct your query optimization efforts. Now let's say you've identified some queries you want to look into. The EXPLAIN command helps you look even closer into an individual query. If you're already proficient in EXPLAIN, great! Read on for an easy refresher. If you're less familiar with it, this will be a (hopefully) gentle introduction on what insights it might help provide.

Index Advisor for Postgres

by pganalyze

The pganalyze Index Advisor analyzes your query, determines how Postgres will scan the tables involved, and may suggest indexes to make those scans more efficient.

How to create (lots!) of sample time-series data with PostgreSQL generate_series()

by Ryan Booz

Generating sample time-series data with the PostgreSQL generate_series() function is a useful skill to have when evaluating new database features, creating demonstrations, or testing insert and query patterns. Learn what PostgreSQL `generate_series()` is and how to use it for basic data generation.

JSON in PostgreSQL: How to do it Right

by Laurenz Albe

The comprehensive JSON support in PostgreSQL is one of its best-loved features. Many people – particularly those with a stronger background in Javascript programming than in relational databases – use it extensively. However, my experience is that the vast majority of people don't use it correctly. That causes problems and unhappiness in the long run.

Postgres Window Magic

by Bruce Momjian

Modern data analysis with PostgreSQL – JSONB throws Window functions out the...

by Thomas Richter

In this blog post I would like to take a look at a classical data layout paradigm and how transactional database features, such as fast UPDATEs and JSON/JSONB types, can make analytics easier.

Fuzzy Name Matching in Postgres

by Paul Ramsey

A surprisingly common problem in both application development and analysis is: given an input name, find the database record it most likely refers to. It's common because databases of names and people are common, and it's a problem because names are a very irregular identifying token.

The Art Of PostgreSQL

by Dimitri Fontaine

Applications nowadays are written with the help of many programming languages. When the backend should implement user oriented workflows, it may rely on a RDBMS component to take care of the system's integrity.

PostgreSQL is the world's most advanced open source relational database, and is very good at taking care of your system's integrity. PostgreSQL also comes with a ton of data processing power, and in many cases a simple enough SQL statement may replace hundreds of lines of code written in Python, Java, PHP, Ruby, Javascript, you name it.

In this talk, we learn advanced SQL techniques and how to reason about which part of the backend code should be done in the database, and which part of the backend code is better written as an SQL query.

Test

Magic Test

by Andrew Culver and Adam Pallozzi

Magic Test allows you to write Rails system tests interactively through a combination of trial-and-error in a debugger session and also just simple clicking around in the application being tested, all without the slowness of constantly restarting the testing environment.

Testing Your Edge Cases

by Joël Quenneville

Do we have any tests that cover the nil case?

Domain Invariants & Property-Based Testing for the Masses

by Romeu Moura

Domain invariants are all around you. In every business rule your domain expert ever tried to give you. You should use them to guide your design.

You should also be testing them! Not only an example representing them: but testing the invariant itself.

You can do them with property-based tests (PBTs). If you think you cannot do PBTs on legacy codebases (or outside FP) this talk should show you otherwise.

Let's also use Property-based tests to reduce test-debt. Create smaller, fewer tests that: test more, are more readable & document the problem. Challenge your understanding of the domain and communication with domain experts.

Prefer Fakes Over Mocks

by Tyrrrz

In this article we will look at the differences between these two variants of test doubles, identify how using one over the other impacts test design, and why using fakes often results in more manageable test suites.

Cross-Branch Testing

by Hillel Wayne

There's a certain class of problems that's hard to test:

1. The output isn't obviously inferable from the input. The code isn't just solving a human-tractable problem really quickly, it's doing something where we don't know the answer without running the program.
2. The output doesn't have "mathematical" properties, like invertibility or commutativity.
3. Errors in the function can be "subtle": there can be a bug that affects only a small subset of possible inputs, so that a set of individual test examples would still be correct.

Performance

Why Averages Suck and Percentiles are Great

by Michael Kopp

Anyone that ever monitored or analyzed an application uses or has used averages. They are simple to understand and calculate. We tend to ignore just how wrong the picture is that averages paint of the world.

Speed is the killer feature

by Brad Dickason

Speed is a killer feature. Speed is a differentiator.

Yet teams consistently overlook speed. Instead, they add more features (which ironically make things slower). Products bloat over time and performance goes downhill.

New features might help your users accomplish something extra in your product. Latency stops your users from doing the job they already hire your product for.

Loading Third-Party JavaScript

by Addy Osmani and Arthur Evans

You've optimized all of your code, but your site still loads too slowly. Who's the culprit?

Often, performance problems slowing pages down are due to third-party scripts: ads, analytics, trackers, social-media buttons, and so on.

Third-party scripts provide a wide range of useful functionality, making the web more dynamic, interactive, and interconnected. These scripts may be crucial to your website's functionality or revenue stream. But third-party scripts also come with many risks that should be taken into consideration to minimize their impact while still providing value.

Reflections on software performance

by Nelson Elhage

I've really come to appreciate that performance isn't just some property of a tool independent from its functionality or its feature set. Performance — in particular, being notably fast — is a feature in and of its own right, which fundamentally alters how a tool is used and perceived.

DevOps

Write your first workflow with GitHub Actions and GitHub APIs

by Bassem Dghaidi

We will cover how the internals of GitHub Actions, workflow syntax, jobs, steps, functions, conditionals, community actions, log viewer, and I will be showing you step by step how you can use GitHub Actions to call GitHub's REST APIs to automate certain tasks!

The Unfulfilled Promise of Serverless

by Corey Quinn

I suggest that serverless computing, or "serverless" has hype that at this point has outpaced what the technology / philosophy / religion has been promising.

The Unfulfilled Potential of Serverless

by Jeremy Daly

[…] we've barely scratched the surface of what's possible with serverless technology.

Forget a server — bring your static website to life with AWS S3, Lambda, and API Gateway

by Jull.io

An entire backend server for something as simple as a contact form seems quite overkill, and, well — it is.

Why We Need More Chaos - Chaos Engineering, That Is

by Nora Jones

[Nora] discusses the benefits and challenges of establishing Chaos Engineering into your systems.

Failing over without falling over

by Adrian Cockcroft

You've gone through the motions and play-acted a disaster recovery scenario, but despite spending a lot on the production, it's not real. What you have is a fairy tale: "Once upon a time, in theory, if everything works perfectly, we have a plan to survive the disasters we thought of in advance." In practice, it's more likely to be a nightmare.

The NGINX Handbook

by Farhan Hasin Chowdhury

After going through the entire book, you should be able to:

- Understand configuration files generated by popular tools as well as those found in various documentation.
- Configure NGINX as a web server, a reverse proxy server, and a load balancer from scratch.
- Optimize NGINX to get maximum performance out of your server.

Amazon Web Services In Plain English

by Expedited Security

Hey, have you heard of the new AWS services: ContainerCache, ElastiCast and QR72? Of course not, I just made those up.

But with 50 plus opaquely named services, we decided that enough was enough and that some plain english descriptions were needed.

Introduction to Terraform - A Practical Approach

by Matthew Sanabria

This video introduces Terraform through a practical approach by walking through how a fictional company would use Terraform to manage their DigitalOcean infrastructure.

SHOCKED BY THE QUALITY?!

SUBSCRIBE TO MY PINKLETTER

WANT TO SUPPORT MY WORK?

BUY ME A COFFEE (OR TWO)

Legacy Code / Maintenance

The Path of Madness

by Brandur

The above is a non-exaggerated description of the life of a programmer in Oracle fixing a bug. Now imagine what horror it is going to be to develop a new feature. It takes 6 months to a year (sometimes two years!) to develop a single small feature.

Keep a Changelog

by Olivier Lacan and Tyler Fortune

Who needs a changelog?

People do. Whether consumers or developers, the end users of software are human beings who care about what's in the software. When the software changes, people want to know why and how.

When costs are nonlinear, keep it small

by Jessica Kerr

Batching work is more efficient … until cost rises nonlinearly with batch size. Then smaller batches are the most efficient. So don't delay maintenance!

Managing technical quality in a codebase

by Will Larson

If there's one thing that engineers, engineering managers, and technology executives are likely to agree on, it's that there's a crisis of technical quality. One diagnosis and cure is easy to identify: our engineers aren't prioritizing quality, and we need to hire better engineers or retrain the ones we have. Of course, you should feel free to replace "engineers" with "Product Managers" or "executives" if that feels more comfortable. It's a compelling narrative with a clear villain, and

it conveniently shifts blame away from engineering leadership. Still, like most narratives that move accountability towards the folks with the least power, it's both unhelpful and wrong.

The Wall of Technical Debt

by Mathias Verraes

The Wall of Technical Debt is a surface in your office where you visualize issues on sticky notes. It's easy to start and maintain, and yet it has a profound impact on how you choose to add, reduce, pay back, or ignore debt. It's by no means intended as a complete solution to scale the management of debt, but it works precisely because it requires no buy-in.

Web

How MDN's autocomplete search works

by Peter Bengtsson

Last month, Gregor Weber and I added an autocomplete search to MDN Web Docs, that allows you to quickly jump straight to the document you're looking for by typing parts of the document title. This is the story about how that's implemented. If you stick around to the end, I'll share an "easter egg" feature that, once you've learned it, will make you look really cool at dinner parties. Or, perhaps you just want to navigate MDN faster than mere mortals.

Have Single-Page Apps Ruined the Web?

by Rich Harris

The backlash to modern front end development is gaining steam, with good reason: single-page apps have ruined the web. Can we rescue it without going backwards? In this talk, Rich Harris presents a way to do just that.

JWT should not be your default for sessions

by Evert Pot

Using JWTs for tokens add some neat properties and make it possible in some cases for your services to be stateless, which can be desirable property in some architectures.

Adopting them comes with drawbacks. You either forego revocation, or you need to have infrastructure in place that be way more complex than simply adopting a session store and opaque tokens.

My point in all this is not to discourage the use of JWT in general, but be deliberate and careful when you do. Be aware of both the security and functionality trade-offs and pitfalls. Keep it out of your 'boilerplates' and templates, and don't make it the default choice.

The Future of Web Software Is HTML-over-WebSockets

by Matt E. Patterson

The future of web-based software architectures is already taking form, and this time it's server-rendered (again). Papa's got a brand new bag: HTML-over-WebSockets and broadcast everything all the time.

Getting started with WebRTC

by Google

The WebRTC standard covers, on a high level, two different technologies: media capture devices and peer-to-peer connectivity.

Media capture devices includes video cameras and microphones, but also screen capturing "devices". For cameras and microphones, we use navigator.mediaDevices.getUserMedia() to capture MediaStreams. For screen recording, we use navigator.mediaDevices.getDisplayMedia() instead.

The peer-to-peer connectivity is handled by the RTCPeerConnection interface. This is the central point for establishing and controlling the connection between two peers in WebRTC.

Deep dive in CORS: History, how it works, and best practices

by Ilija Eftimov

Learn the history and evolution of same-origin policy and CORS, understand CORS and the different types of cross-origin access in depth, and learn (some) best practices.

The State Of Web Workers In 2021

by Surma

The web is single-threaded. This makes it increasingly hard to write smooth and responsive apps. Workers have a bad rep, but can be an important and useful tool

in any web developer's toolbelt for these kinds of problems. Let's get up to speed on Workers on the Web!

A Look at Server-Sent Events

by Simon Prickett

Server Sent Events are a standard allowing browser clients to receive a stream of updates from a server over a HTTP connection without resorting to polling. Unlike WebSockets, Server Sent Events are a one way communications channel - events flow from server to client only.

A11y Coffee

by Amberley

Pick your serving size of web accessibility information.

Http

HTTPWTF

by Tim Perry

HTTP is fundamental to modern development, from frontend to backend to mobile. But like any widespread mature standard, it's got some funky skeletons in the closet.

Some of these skeletons are little-known but genuinely useful features, some of them are legacy oddities relied on by billions of connections daily, and some of them really shouldn't exist at all. Let's look behind the curtain:

The Idempotency-Key HTTP Header Field

by Sanjay Dalal and IETF

The HTTP Idempotency-Key request header field can be used to carry idempotency key in order to make non-idempotent HTTP methods such as "POST" or "PATCH" fault-tolerant.

The Long Road To HTTP/3

by Andrew Savchyn

While HTTP/3 specification is still in the draft stage, the latest version of the Chrome browser already supports it by default . With Chrome holding around 70% of browser market share, you could say HTTP/3 has gone mainstream.

How HTTPS works ...in a comic!

by Dnsimple

Have you ever wondered why a green lock icon appears on your browser URL bar? And why is it important? We did too, and this comic is for you!

REST / HTTP methods: POST vs. PUT vs. PATCH

by Michael Scharhag

In general we can say:

- POST requests create child resources at a server defined URI. POST is also used as general processing operation
- PUT requests create or replace the resource at the client defined URI
- PATCH requests update parts of the resource at the client defined URI

HTTP methods: Idempotency and Safety

by Michael Scharhag

Idempotency and safety are properties of HTTP methods. The HTTP RFC defines these properties and tells us which HTTP methods are safe and idempotent. Server application should make sure to implement the safe and idempotent semantic correctly as clients might expect it.

HTTP SEARCH Method

by Julian Reschke, Ashok Malhotra, James Snell

Many existing HTTP-based applications use the HTTP GET and POST methods in various ways to implement the functionality provided by SEARCH.

Software in General

No, dynamic type systems are not inherently more open

by Alexis King

Internet debates about typing disciplines continue to be plagued by a pervasive myth that dynamic type systems are inherently better at modeling "open world" domains. The argument usually goes like this: the goal of static typing is to pin everything down as much as possible, but in the real world, that just isn't practical. Real systems should be loosely coupled and worry about data representation as little as possible, so dynamic types lead to a more robust system in the large.

What Is Functional Programming?

by Kris Jenkins

This is my take on what functional programming really is, in a way that will make sense to a jobbing programmer just trying to Get Stuff Done.

Finding Service Boundaries

by Udi Dahan

Although Service-Oriented Architecture is a fairly well-known topic, there are very few good examples out there of the application of SOA principles to non-trivial domains so developers don't have much to learn from. In this presentation, Udi will show a case study from the healthcare domain resulting in services so autonomous they almost don't have to share any data with each other – whether that's through request/response, events, or via a shared data store.

Essays on programming I think about a lot

by Ben Kuhn

Every so often I read an essay that I end up thinking about, and citing in conversation, over and over again. Here's my index of all the ones of those I can remember!

Domain Modelling Made Functional

by Scott Wlaschin

Types can be used to represent the domain in a fine-grained, self documenting way. And in many cases, types can even be used to encode business rules so that you literally cannot create incorrect code. You can then use the static type checking almost as an instant unit test — making sure that your code is correct at compile time.

How we ship code faster and safer with feature flags

by Alberto Gimeno

At GitHub, we're continually working to improve existing features and shipping new ones all the time. From our launch of GitHub Discussions to the release of manual approvals for GitHub Actions—in order to ship new features and improvements faster while lowering the risk in our deployments, we have a simple but powerful tool: feature flags.

The Wrong Abstraction

by Sandi Metz

The moral of this story? Don't get trapped by the sunk cost fallacy. If you find yourself passing parameters and adding conditional paths through shared code, the abstraction is incorrect. It may have been right to begin with, but that day has passed. Once an abstraction is proved wrong the best strategy is to re-introduce duplication and let it show you what's right. Although it occasionally makes sense to accumulate a few conditionals to gain insight into what's going on, you'll suffer less pain if you abandon the wrong abstraction sooner rather than later.

Minimalism versus Types

by Hisham

We love minimalistic languages because they let us do so much with so little. But when we start doing a lot with them, often we start yearning for types to help us make sense of it all. Adding types to a minimalistic language (well, adding anything!) makes it larger. Is this worth the price? Is a rich type system antithetical to minimalism? Let's find out!

The data model behind Notion's flexibility

by Jake Teton-Landis

Everything you see in Notion is a block. Text, images, lists, a row in a database, even pages themselves — these are all blocks, dynamic units of information that can be transformed into other block types or moved freely within Notion

How to deal with money in software

by Tom Sydney Kerckhove

I recently did my accounting for my investment account. I noticed a significant amount of errors in the account statement that my broker gave me. After a bit of a rant on twitter, I want to explain what they likely did wrong and how to deal with money correctly.

Rethinking Best Practices

by Will Gallego

The concepts behind best practices may be sound, but as they are not ubiquitous, they should be up for debate. Our tendency to skip deeper investigation, to assume an answer is correct based on a label, makes use of the term "best practice" dangerous.

Breaking Up the Behemoth

by Sandi Metz

Sticking with procedures too long is just as bad as doing design too soon. If important classes in your domain change often, get bigger every time they change, and are accumulating conditionals, stop adding to them right now. Use every new request as an opportunity to do a bit of design. Implement the change using small, well-designed classes that collaborate with the existing object.

Falling Into The Pit of Success

by Jeff Atwood

Wouldn't it be nice to use a language designed to keep you from falling into the pit of despair? But avoiding horrific, trainwreck failure modes isn't a particularly laudable goal. Wouldn't it be even better if you used a language that let you effortlessly fall into The Pit of Success?

Team

Get your work recognized: write a brag document

by Julia Evans

There's this idea that, if you do great work at your job, people will (or should!) automatically recognize that work and reward you for it with promotions / increased pay. In practice, it's often more complicated than that – some kinds of important work are more visible/memorable than others. It's frustrating to have done something really important and later realize that you didn't get rewarded for it just because the people making the decision didn't understand or remember what you did. So I want to talk about a tactic that I and lots of people I work with have used!

Tidepool Employee Handbook

by Tidepool

Q: Why can't we just say "No politics at work?"

A: Our mission is inherently political. Also, one of our values is to "fight the default of exclusion."

Hyperproductive development

by Jessica Kerr

Let's talk about why some developers, in some situations, are ten times more productive than others.

hint: it isn't the developers, so much as the situation.

Bourdieu's social theory applied to tech

by Romeu Moura

Obscure though it may seems, the sociologist Bourdieu and his social theory tell us a lot about what is happening in the workplace and society around us. By understanding what he meant by "symbolic violence", "cultural capital" "hexis" etc, we see how each of us influences and is influenced by the people around us, in ways that we wouldn't expect. From this talk, a vulgarized and easy to understand version of Bourdieu's ideas, each of us can seek how to improve the ambience immediately around us.

Pushing through Friction

by Dan Na

As a senior-level engineering leader, experience tells you things could be better. You see the gaps. If only the company adopted policy A or dumped technology B, everyone would benefit. But there's so much inertia. The company has always used B. You are frustrated. Can you actually make a difference?

Sociotechnical Lenses into Software Systems

by Paul Osman

The systems we are building and operating are constantly being modified by different people, with different contexts, at different times, who may or may not speak to each other directly. This emergent collaboration can present unique challenges that are fun to navigate.

Meeting Resistance and Moving Forward

by Linda Rising

It's "those skeptical people" who are most annoying. They don't seem to listen to our ideas. They usually start raising objections before we have even finished describing what we are thinking. They have a counterargument for every argument. What's to be done with "those people"? In this presentation, Linda will pull patterns from the Fearless Change collection plus the latest research in neuroscience to help you in the challenges you face with resistance.

Team Norming Ceremony: Build Effective Remote Collaboration from Day One

by Hana Mohan

Tools such as Zoom and Slack have made remote collaboration not just manageable but downright efficient. One challenge still present is the human element of working together. Is a teammate a morning or night person? Do they prefer Slacking asynchronously or Zooming face-to-face? Are there differences in personal values amongst the team?

The best retrospective for beginners

by Corinna Baldauf

Are you new to facilitating retrospectives? Then you're probably wondering how to best get started. For what it's worth, here's my "Given that I know nothing about you or the team's situation here's my best shot at a multi-purpose, easy to facilitate retrospective plan".

Project Management

Do Things that Don't Scale

by Paul Graham

One of the most common types of advice we give at Y Combinator is to do things that don't scale.

Shape Up: Stop Running in Circles and Ship Work that Matters

by Basecamp

Shape Up is for product development teams who struggle to ship. If you've thought to yourself "Why can't we ship like we used to?" or "I never have enough time to think about strategy," then this book can help. You'll learn language and techniques to define focused projects, address unknowns, and increase collaboration and engagement within your team.

Overengineering can kill your product

by Simón Muñoz

The graveyard is filled with exquisitely designed startups and products to scale to millions of users who never got the slightest bit of traction. Don't become one of them.

Self Development

Write 5x more but write 5x less

by Mike Crittenden

The biggest networking opportunity you're missing out on

by Mayuko

Stop Looking For Mentors

by Stay SaaSy

So stop worrying about mentors and mentorship and mentoring. Those words psyche people out. They set expectations that are counter to the normal time and mechanics it takes to build a relationship.

25 lessons from 25 years of coding

by Swizec Teller

Here's 25 lessons I've learned about code. In no particular order.

Software development topics I've changed my mind on after 6 years in the industry

by Chris Kiehl

At some point, I realized I would've argued the exact opposite position on a lot of topics just a few years ago.

Why Do We Work Too Much?

by Cal Newport

Our tendency to work twenty per cent too much is neither arbitrary nor sinister: it's a side effect of the haphazard nature in which we allow our efforts to unfold. By thinking more intentionally about how work is identified, how it is prioritized, and how it is ultimately assigned, we can avoid some of the traps set by pure self-regulation.

20 Things I've Learned in my 20 Years as a Software Engineer

by Justin Etheredge

My experiences over the last 20 years have shaped how I view software, and have led me to some beliefs which I've tried to whittle down to a manageable list that I hope you find valuable.

Why Tacit Knowledge is More Important Than Deliberate Practice

by Cedric Chin

Once you understand that tacit knowledge exists, you will begin to see that big parts of any skill tree is tacit in nature, which means that you can go hunting for it, which in turn means you can start to ask the really useful question when it comes to expertise, which is: that person has it; that person is really good at it; how can I have it too?

An Easier Method for Extracting Tacit Knowledge

by Cedric Chin

Let's say that you're a junior employee at a company, and you find a senior person with a set of skills that you'd like to learn. You ask them how they do it, and they give you an incomprehensible answer — something like "oh, I just know what to do", or "oh, I just do what feels right". How can you learn their skills?

Expiring vs. Permanent Skills

by Morgan Housel

Robert Walter Weir was one of the most popular instructors at West Point in the mid-1800s. Which is odd at a military academy, because he taught painting and drawing.

How You Know

by Paul Graham

I've read Villehardouin's chronicle of the Fourth Crusade at least two times, maybe three. And yet if I had to write down everything I remember from it, I doubt it would amount to much more than a page. Multiply this times several hundred, and I get an uneasy feeling when I look at my bookshelves. What use is it to read all these books if I remember so little from them? (This made me feel better about myself.)

On being wrong

by Kathryn Schulz

Most of us will do anything to avoid being wrong. But what if we're wrong about that? "Wrongologist" Kathryn Schulz makes a compelling case for not just admitting but embracing our fallibility.

Life Audit

by Ximena Vengoechea

Life audit (n.): An exercise in self-reflection that helps you clear the cobwebs of noisy, external goals and current distractions, and revisit or uncover the real themes & core values that drive & inspire you. Also known as: spring-cleaning for the soul.

Personal values

by Julian Shapiro

So I'm publicly sharing my framework to help others come to similar realizations: What should you really be working on?

The paradox of choice

by Barry Schwartz

Psychologist Barry Schwartz takes aim at a central tenet of western societies: freedom of choice. In Schwartz's estimation, choice has made us not freer but more paralyzed, not happier but more dissatisfied.

Deep Questions with Cal Newport Ep. 39: DAVID EPSTEIN on Skills, Practice, and the Subtle Art of Cultivating a Meaningful Career

by Cal Newport

SHOCKED BY THE QUALITY?!

SUBSCRIBE TO MY PINKLETTER

WANT TO SUPPORT MY WORK?

BUY ME A COFFEE (OR TWO)

Stories

Half a Billion in Bitcoin, Lost in the Dump

by D.T. Max

For years, a Welshman who threw away the key to his cybercurrency stash has been fighting to excavate the local landfill.

The Art of Code - Dylan Beattie

by Dylan Beattie

But what about the code that only exists because somebody wanted to write it? Code created just to make people smile, laugh, maybe even dance? Maybe even code that does nothing at all, created just to see if it was possible?

Inventing on Principle

by Bret Victor

This is one of my favorite talks about programming. Bret Victor is a visionary and you should watch this talk.

The Future of Programming

by Bret Victor

I would make you a disservice if I didn't link the other one by Bret. This talk is from 1973, or is it?

A Brief Rant on the Future of Interaction Design

by Bret Victor

With an entire body at your command, do you seriously think the Future Of Interaction should be a single finger?

Edsger Dijkstra—The Man Who Carried Computer Science on His Shoulders

by Krzysztof R. Apt

A pioneer whose work shaped his field like few others.

The Cold War Bunker That Became Home to a Dark-Web Empire

by Ed Caesar

An eccentric Dutchman began living in a giant underground facility built by the German military—and ran a server farm beloved by cybercriminals.

Growing a Language

by Guy Steele

Guy Steele's keynote at the 1998 ACM OOPSLA conference on "Growing a Language" discusses the importance of and issues associated with designing a programming language that can be grown by its users.

I programmed some creatures. They Evolved

by Dave Miller

This is a report of a software project that created the conditions for evolution in an attempt to learn something about how evolution works in nature. This is for the programmer looking for ideas for interdisciplinary programming projects, or for anyone interested in how evolution and natural selection work.

Design

Reviewing your design portfolios!

by Charli Marie

I hope you enjoy getting a look at some awesome viewers work, and hearing the advice I give them! My hope with this series is that even if it's not your portfolio being reviewed you can get some useful insights that perhaps you could apply to your own design portfolio. I'm not critiquing the work in it, but rather the design and layout of your portfolio itself and looking critically at the type of work included to see if it aligns with your goals. I'd love to hear what you think of the series in the comments!

UX request: Tell, don't hide

by Jessica Kerr

Y'know how sometimes a particular logged-in user isn't authorized for some option, so you don't show it to them?

Don't do that to people. Tell me it's there, tell me I can't do it, tell me who can.

Pride Backgrounds

by Gosia Nowak

Happy Pride, folks!

Christopher Alexander: A Primer

by Ryan Singer

Christopher Alexander's work is hard to get into. He's written over 15 books, and there isn't one that serves as a general intro or overview for the rest. In this livestream, I gave an informal introduction to what I think are the most important ideas in his body of work.

Css

How to Find and Remove Dead CSS

by Justin Searls

Do you have a pile of old CSS styles that you're pretty sure are no longer referenced anywhere, but that you're nevertheless afraid to delete because you have no way to be sure that no musty corners of your site somehow depend on them to render correctly?

CSS Utility Classes and "Separation of Concerns"

by Adam Wathan

Why "separation of concerns" is the wrong way to think about CSS and why presentational classes scale better than semantic classes.

Flexbox Zombies

by Dave Geddes

Flexbox is incredibly powerful. But it's also crazy hard to learn well. So we all end up depending on a cheat sheet and some mad guessing in the dev tools.

This is an Educational Game. Each section unravels part of the plot, gives you expertise over a new flexbox concept, and presents zombie survival challenges that force you to solidify your new skills like your life depends on it.

Git

Git's Best And Most Unknown Feature

by ThePrimeagen

HOW HAVE I NOT HEARD OF GIT WORK TREES??? WHAT THE EFF. They are so incredible. You have to check them out!!! In this video I go over them briefly, assuming you are smart enough to understand them, and also show you my workflow with vim! Its fantastic!

13 Advanced (but useful) Git Techniques and Shortcuts

by Fireship

Productive programmers tend to be really good at Git. Take a look at 13 advanced git tips and tricks to supercharge your development workflow.

Oh My Git!

by bleeptrack and blinry

An open source game about learning Git!

Little Things I Like to Do with Git

by Harry Roberts

I thought I would note down some useful little Git snippets that I use the most frequently.

Strategies For Small, Focused Pull Requests

by Steve Hicks

A common suggestion for improving pull requests (PRs) is to “make your PR small and focused”. I myself gave this suggestion in a recent article on this very blog about including context in PRs.

Like most internet advice, this can feel like the “draw the rest of the owl” meme. Even if we’re in agreement that I should make a PR smaller…how do I do it? How do I avoid a big PR when there’s a lot of cross-cutting changes to make? How do I create small, focused units of work when I’m building a large feature? How can I overcome my perfectionism and submit a PR that feels incomplete to me because the edges aren’t all polished?

Tools

An Introduction to JQ

by Adam Gordon Bell

In this article, I'm going to go over the basics building blocks of jq in enough depth that you will be able to understand how jq works. Of course, you still might occasionally need to head to google to find a function name or check your syntax, but at least you'll have a firm grounding in the basics.

Developer Tools secrets that shouldn't be secrets

by Chris Heilmann

This is a talk that I've given at CityJS this September. I am a principal product manager for developer tools in Microsoft Edge and these are things I encountered during working on the tools, documenting them and going through user feedback.

ShellCheck

by Vidar Holen

Finds bugs in your shell scripts.

Remix

Remix is a full stack web framework that lets you focus on the user interface and work back through web fundamentals to deliver a fast, slick, and resilient user experience. People are gonna love using your stuff.

Ray.so

by Nichlas W. Andersen

Turn your code into beautiful images. Choose from a range of syntax colors, hide or show the background, and toggle between a dark and light window.

Visualizing a codebase

by Amelia Wattenberger

How can we "fingerprint" a codebase to see its structure at a glance? Let's explore ways to automatically visualize a GitHub repo, and how that could be useful.

My workflow using Vim, 2021

by ThePrimeagen

This is my workflow using vim. I wanted the kind of showcase how I jump around and how I think about things. this isn't meant to be a super technical video, more a video just describing how I approach using vim.

Spying on your programs with strace

by Julia Evans

strace is my favorite program. I think that it doesn't get enough attention from programmers, so I wrote a zine about it to teach more people about how to use it.

Understanding AWK

by Adam Gordon Bell

If you read through the article and maybe even try an example or two, you should have no problem writing Awk scripts by the end of it.

cheat.sh

by Igor Chubin

Unified access to the best community driven cheat sheets repositories of the world

ShortcutFoo

ShortcutFoo was created by programmers for programmers in an attempt to make learning your editor fun, easy, and effective. Akin to the days of first learning how to type on a keyboard, shortcutFoo aims to help programmers accomplish more in less time and with fewer keystrokes.

Fun Stuff

Proposal: Downward assignments

by Yusuke Endoh

Rightward assignments have been introduced since 3.0. To be honest, I'm not a big fan of the syntax because it does not add a new dimension to Ruby. Why don't we bring Ruby to the next dimension?

Blob Opera

by David Li

Create your own opera inspired song with Blob Opera - no music skills required! A machine learning experiment by David Li in collaboration with Google Arts & Culture

If PHP Were British

by Dave Child

PHP developers in Britain have been grumpy about this ever since. What was he thinking? And more importantly, how do we undo this travesty? How do we developers ensure the traditions of the British Empire continue to be upheld, even in the digital age?

Terms & Conditions Apply Game

by Wieden Kennedy, Jon Plackett, Alex Bellos, Adam Hunt

A mini-game about pop ups, and the deviousness of websites and apps EVIL CORP wants your data. It will use every trick in the book (and a few more, just for fun).

Misc

NFTs Weren't Supposed to End Like This

by Anil Dash

When we invented non-fungible tokens, we were trying to protect artists. But tech-world opportunism has struck again.

What designers NEED to know about IP, copyright & trademarks!

by Charli Marie

Intellectual property and your rights surrounding it can be a complex topic. In this video I'm breaking them down for you and giving you tips on things to watch out for.

What Makes Quantum Computing So Hard to Explain?

by Scott Aaronson

Quantum computers, you might have heard, are magical uber-machines that will soon cure cancer and global warming by trying all possible answers in different parallel universes. For 15 years, on my blog and elsewhere, I've railed against this cartoonish vision, trying to explain what I see as the subtler but ironically even more fascinating truth.

Two Million Years in Two Hours: A Conversation with Yuval Noah Harari

by Center for Humane Technology

Yuval Noah Harari is one of the rare historians who can give us a two-million-year perspective on today's headlines. In this wide-ranging conversation, Yuval explains

how technology and democracy have evolved together over the course of human history, from Paleolithic tribes to city states to kingdoms to nation states.

So where do we go from here? "In almost all the conversations I have," Yuval says, "we get stuck in dystopia and we never explore the no less problematic questions of what happens when we avoid dystopia." We push beyond dystopia and consider the nearly unimaginable alternatives in this special episode of Your Undivided Attention.

Patterns in confusing explanations

by Julia Evans

So why do I find all these explanations so confusing? I decided to try and find out! I came up with a list of 13 patterns that make explanations hard for me to understand. For each pattern I'll also explain what I like to do instead to avoid the issue.

Understanding by Design

by Grant Wiggins

Grant Wiggins introduces Understanding by Design (UbD), a framework for improving student achievement that helps teachers clarify learning goals, devise assessments that reveal student understanding, and craft effective learning activities.

Emoji under the hood

by Tonsky

For the past few weeks, I've been implementing emoji support for Skija. I thought it might be fun sharing a few nitty-gritty details of how this "biggest innovation in human communication since the invention of the letter A" works under the hood.

Are We Really Engineers?

by Hillel Wayne

Is software engineering "really" engineering? A lot of us call ourselves software engineers. Do we deserve that title? Are we mere pretenders to the idea of engineering?

Why Electron apps are fine

by Niels Leenheer

It is not difficult to find some incredibly shitty takes on Electron, and every time it boils down to: It's slow. Downloads are huge, and it uses a lot of memory. Electron apps are just websites. Developers that are using Electron are taking the lazy or easy approach to cross-platform development. Native apps are just better in every single way.

The Stubborn Optimist's Guide Revisited

by Center For Humane Technology

Internationally-recognized global leader on climate change Christiana Figueres argues that the battle against global threats like climate change begins in our own heads. She became the United Nations' top climate official after she watched the 2009 Copenhagen climate summit collapse "in blood, in screams, in tears."

In the wake of that debacle, Christiana began performing an act of emotional Aikido on herself, her team, and eventually delegates from 196 nations. She called it "stubborn optimism." It requires a clear and alluring vision of a future that can supplant the dystopian and discouraging vision of what will happen if the world fails to act.

It was stubborn optimism, she says, that convinced those nations to sign the first global climate framework, the Paris Agreement. In this episode, we explore how a similar shift in Silicon Valley's vision could lead three billion people to take action for the planet.

Tackling Concurrency Bugs with TLA+

by Hillel Wayne

Concurrency is hard. How do you test your system when it's spread across three services and four languages? Unit testing and type systems only take us so far. At some point we need new tools.

Enter TLA+. TLA+ is a specification language that describes your system and the properties you want. This makes it a fantastic complement to testing: not only can you check your code, you can check your design, too! TLA+ is especially effective for testing concurrency problems, like stalling, race conditions, and dropped messages.

Falsehoods programmers believe about time zones

by Zain Rizvi

I soon discovered just how wrong I was. One after another, I kept learning the falsehood of yet another "fact" that had seemed obviously true.

Learn to Code Blockchain DApps By Building Simple Games

by CleverFlare

CryptoZombies is an interactive school that teaches you all things technical about blockchains. Learn to make smart contracts in Solidity or Libra by making your own crypto-collectibles game.

Reverse Engineering the source code of the BioNTech/Pfizer SARS-CoV-2 Vaccine

by Bert Hubert

Welcome! In this post, we'll be taking a character-by-character look at the source code of the BioNTech/Pfizer SARS-CoV-2 mRNA vaccine.

Hands-Free Coding

by Joshua Comeau

Earlier this year, I developed Cubital Tunnel Syndrome, a repetitive-strain injury, in both of my elbows. As a result, I pretty much can't use a mouse or keyboard; after a few minutes, I get a burning pain shooting down my arms. Even if I try to limit my computer usage to 60-second bursts, I wind up inadvertently making the situation worse.

Automating my job by using GPT-3 to generate database-ready SQL to answer business questions

by Brian Kane

I want to be able to describe a question in plain English and have GPT-3 convert it into the SQL code that, if executed on my Postgres database, would answer the question.

Why Is Apple's M1 Chip So Fast?

by Erik Engheim

Real-world experience with the new M1 Macs has started ticking in. They are fast. Real fast. But why? What is the magic?

Github Copilot MAKES A CLI GAME IN GOLANG FROM SCRATCH?!?!

by ThePrimeagen

Yes the dang AI MAKES A GAME!!! I could not believe it and the ending just blew me away.

SHOCKED BY THE QUALITY?!

SUBSCRIBE TO MY PINKLETTER

WANT TO SUPPORT MY WORK?

BUY ME A COFFEE (OR TWO)

THANKS.

If your mind is empty, it is always ready for anything; it is open to everything.

In the beginner's mind there are many possibilities; in the expert's mind there are few.

– Zen Mind, Beginner's Mind