



Jon Pither

Malcolm Sparks

goto;

JUXT

A GRID DYNAMICS
COMPANY

Dynamic & Proud

Malcolm Sparks - GOTO Copenhagen 2024

Statically typed

C++

Java

C#

Haskell

Scala

Kotlin

Go

Rust

OCaml

TypeScript

vs.

Dynamically typed

Lisp

Scheme

Erlang

Python

Perl

Ruby

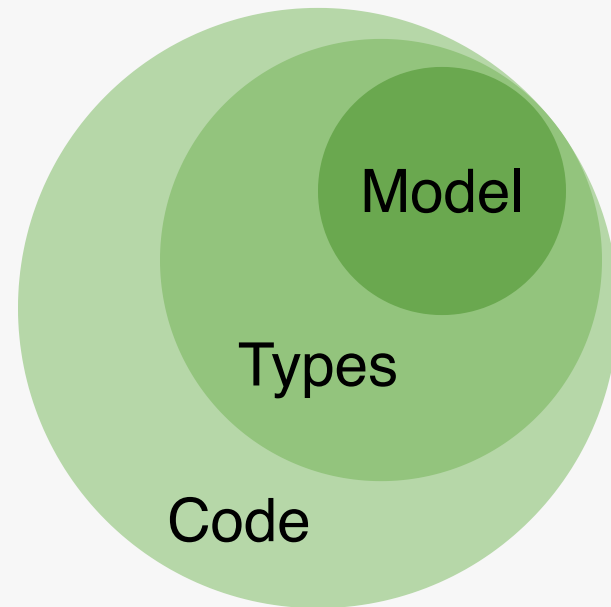
PHP

Lua

Clojure

JavaScript

1. Compilers catch bugs
2. Performance
3. IDE suggestions
4. Refactoring





"Type Systems - The Good, Bad and Ugly"
by Paul Snively and Amanda Laucher



Is it even ethical to use
a dynamically typed
language?

Are there *any* reasons
to adopt a dynamically-
typed language?

Benefits of dynamically typed

- Rapid development
- Easier to learn
- Metaprogramming
- Interactive development

Is there more to dynamic languages?



Nathan Marz
@nathanmarz



Static typing is better for large codebases if your development team is mediocre or weak. Dynamic typing is far superior for a strong team. If you think static typing is better, then there are huge improvements you can make in how you structure your code and how you test.

9:49 PM · Aug 13, 2024 · **193.4K** Views



Static or Dynamic?

It depends. What do we
want to *fix*, and what do
we want to *flex*?

What do we mean by
type?

Primitive Types

Boolean Float
Integer Character

'Atomic'
Types

Compound Types

String Date Money
UUID Point URL

Aggregate Types

Arrays Lists
Sets Stacks
Maps Queues

Record Types

Customer Employee
Address Account
Product Trade

Record types

```
{  
  "conf": "GOTO Copenhagen 2024",  
  "title": "Dynamic and Proud",  
  "speaker": "Malcolm Sparks"  
}
```

```
public record TalkSummary (  
  Conference conf,  
  String title,  
  Speaker speaker) {  
}
```

Why does a domain evolve?

Ontological Change

- Market changes
- New product launches
- New regulations

External

Epistemological Change

- New information
- Improved modelling
- Correction of mistakes

Internal

Common dev practices of many static language projects

TDD

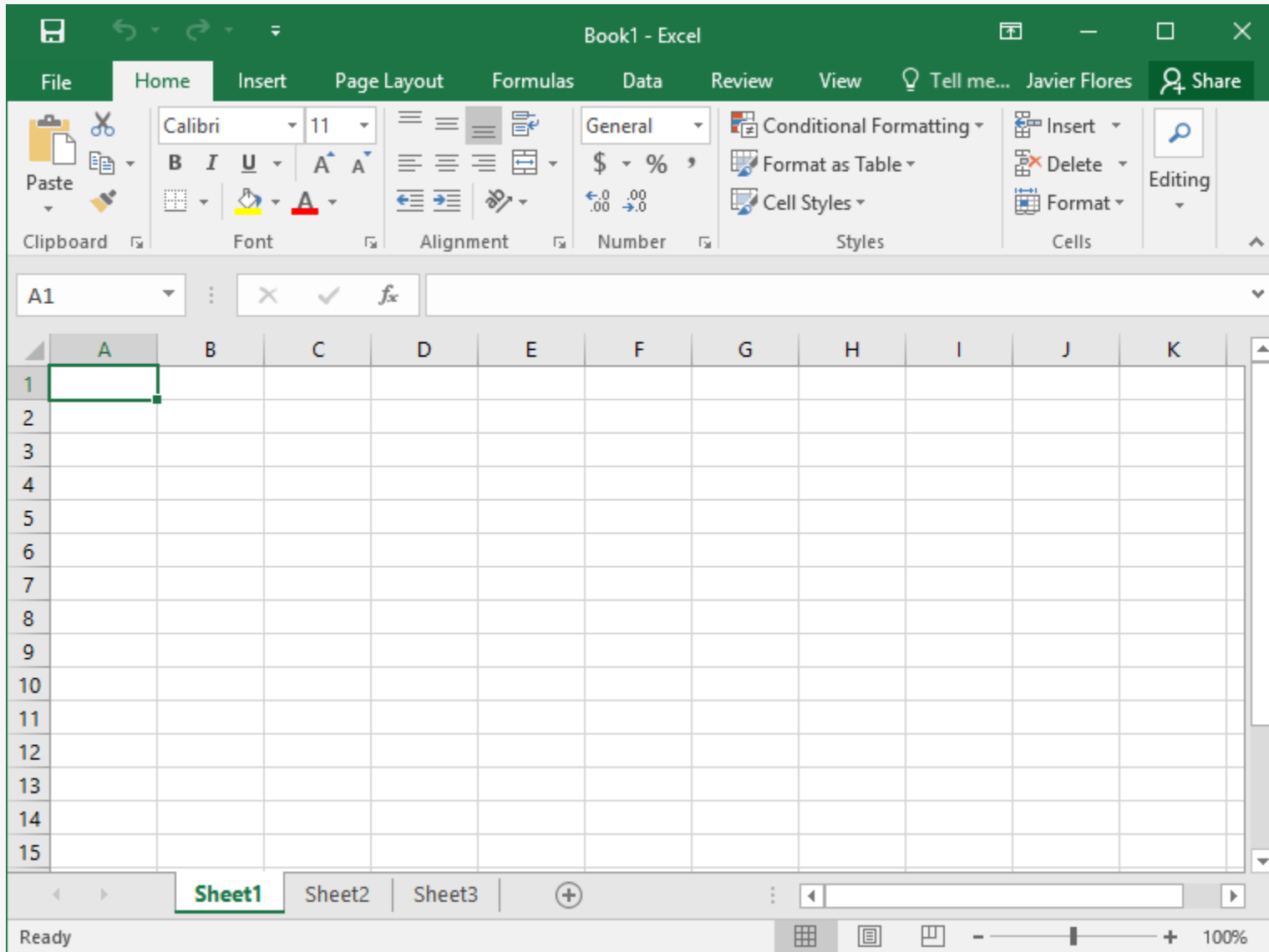
Domain records

Refactoring

Incremental development

CI/CD

Continual change



Statically Typed System

Primitive Types

Compound Types

Aggregate Types

Record Types

vs.

Dynamically Typed System

Primitive Types

Compound Types

Aggregate Types



Introspection

Schema

JSON Schema

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "type": "object",
4   "properties": {
5     "conf": {
6       "type": "string",
7       "description": "The name of the conference."
8     },
9     "talk": {
10      "type": "string",
11      "description": "The title of the talk."
12    },
13    "speaker": {
14      "type": "string",
15      "description": "The name of the speaker."
16    },
17    "bio": {
18      "type": "string",
19      "description": "An optional biography of the speaker."
20    }
21  },
22  "required": ["conf", "talk", "speaker"],
23  "additionalProperties": false
24 }
```



Other kinds of schema

1. RDF/SHACL
2. GraphQL Schema
3. XML Schema
4. Database Schema

Form

Schemas

Business Processes

Rules

Domain Language and Types

Ontologies

= Evolving Organisational Knowledge

An architectural 'stack' for dynamic systems

Code

keep code general, write compilers and interpreters

Form

describe your domain as schema

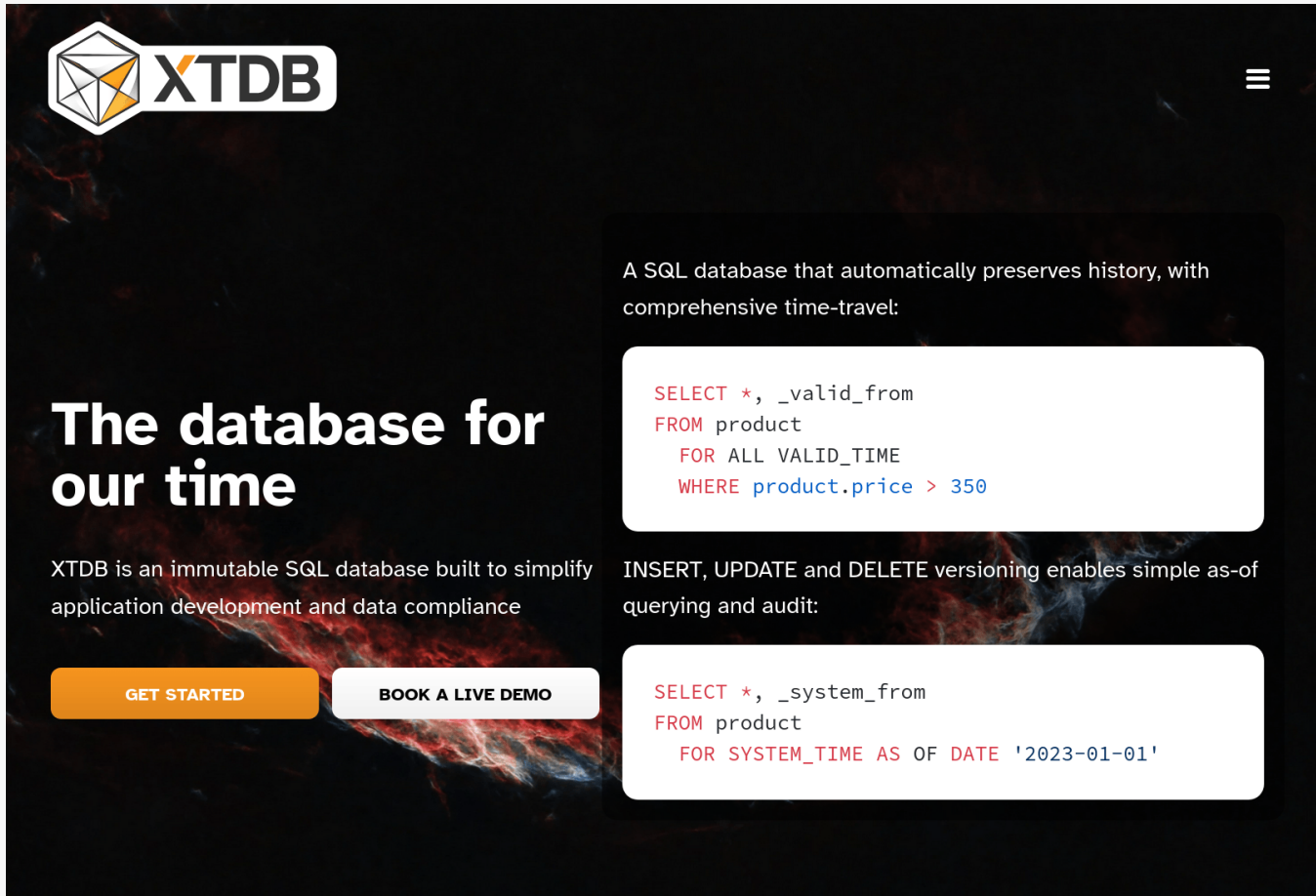
Data

store schema as data

Time

track all versions of data, across time, immutably

XTDB

The hero section of the XTDDB website features a dark background with a vibrant, colorful nebula or galaxy image. The XTDDB logo is in the top left, and a hamburger menu is in the top right. The main headline is 'The database for our time'. Below it, a paragraph describes XTDDB as an immutable SQL database. Two buttons, 'GET STARTED' and 'BOOK A LIVE DEMO', are positioned below the paragraph. To the right, two code snippets are displayed in white boxes, demonstrating time-travel queries. The first snippet shows a query for all valid time records where the price is greater than 350. The second snippet shows a query for system time as of a specific date.

 **XTDB**



The database for our time

XTDB is an immutable SQL database built to simplify application development and data compliance

[GET STARTED](#) [BOOK A LIVE DEMO](#)

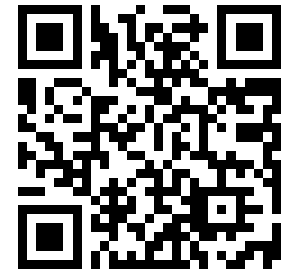
A SQL database that automatically preserves history, with comprehensive time-travel:

```
SELECT *, _valid_from
FROM product
  FOR ALL VALID_TIME
 WHERE product.price > 350
```

INSERT, UPDATE and DELETE versioning enables simple as-of querying and audit:

```
SELECT *, _system_from
FROM product
  FOR SYSTEM_TIME AS OF DATE '2023-01-01'
```

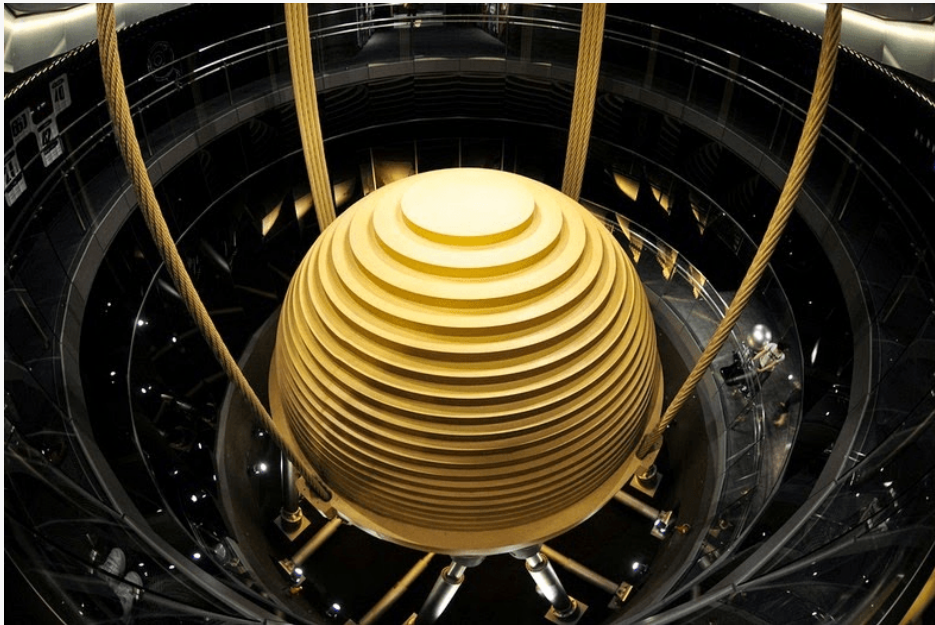
JUXTCast on this topic



<https://www.youtube.com/watch?v=E6ilWUa0N9U>

Conclusion

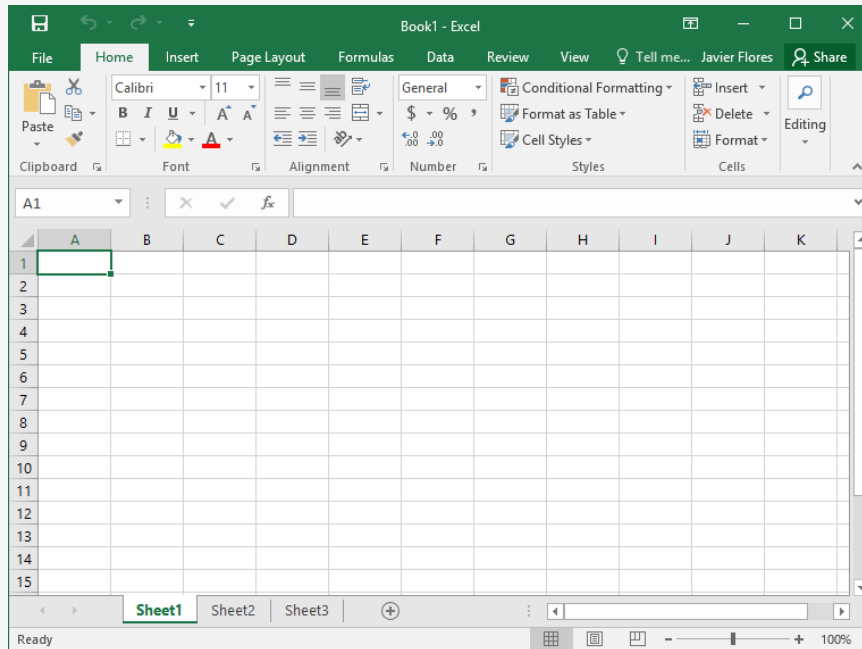
Let's design systems that can flex and adapt to change



Let's design systems that survive their development phase



Let's build dynamic systems. Dynamic languages can help.



Q & A



Jon Pither
jon@juxt.pro
@jonpither

Malcolm Sparks
mal@juxt.pro
@malcolmsparks