All Spreadsheets must Die

Robert Schadek
May 8, 2019
Getting started
A random list of languages we love to hate
C++ 4.4 Million (2015)
C 1.9 Million (2015)
Java 9 Million (2009)
JS 10 Million (2018)
These are all small fish
These are all small fish

Excel

\[ \approx 750 \text{ Million (2016)} \]
BUT EXCEL IS NOT PROGRAMMING
Oh, but it is

### Functional programming in Excel

**Felienne Hermans**

[@Felienne](https://twitter.com/Felienne)
A little bit of Spreadsheet bashing
Seeing the code is difficult

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>133.7</td>
<td>=A1 * 13.37</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>187.18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>---</td>
<td>------------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>Firstname</td>
<td>Lastname</td>
<td>Age</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>Hans</td>
<td>Mustermann</td>
<td>twenty-five</td>
</tr>
</tbody>
</table>
## Dynamic Types

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Firstname</td>
<td>Lastname</td>
<td>Age</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>Hans</td>
<td>Mustermann</td>
<td>25</td>
</tr>
</tbody>
</table>
### Dynamic Types

<table>
<thead>
<tr>
<th></th>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Doe</td>
<td>34</td>
</tr>
<tr>
<td>2</td>
<td>Hans</td>
<td>Mustermann</td>
<td>25</td>
</tr>
</tbody>
</table>
git blame

let's not go there
we will just become sad
git blame

lets not go there

we will just become sad
Code refactoring

- \( \text{SUM}(1,2) \)

- Set Excel locale to de_DE
Code refactoring

- =SUM(1,2)
- equal, identifier, lparen, int(1), comma, int(2), rparen
Code refactoring

- =SUM(1,2)

- equal, identifier, lparen, int(1), comma, int(2), rparen

- set Excel locale to de_DE
Code refactoring

- =SUM(1,2)
- equal, identifier, lparen, int(1), comma, int(2), rparen
- set Excel locale to de_DE
- =SUM(1,2)
Code refactoring

- \(=\text{SUM}(1,2)\)
- \(=\text{SUM}(1,2)\)
- set Excel locale to de\_DE
- \(=\text{SUM}(1,2)\)
- equal, identifier, lparen, float(1.2), rparen
- Knowledge silos
- Slow
- No separation between data and code
- Access management . . .
Bits and Pieces

- Knowledge silos
- Slow
- No separation between data and code
- Access management . . . anybody?
1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce
1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
(Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce
(Typical) Spreadsheet Lifecycle

1. Create private spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
1. Create private shopping spreadsheet
2. Show spreadsheet to college
3. Use spreadsheet for all company purchases
4. Put web frontend on spreadsheet backend
5. Pivot company to become E-Commerce company
Spreadsheets rule the world!
How you should be feeling right now
Two assumptions going forward

1. You believe that spreadsheets rule the world.
2. You want D to rule the world instead.
How are we going to win this?
How are we going to win this?

We are not!
Let's draw up a battle plan
Let's take stock of what we have

- Too many spreadsheets
- Too many tasks
- Too little man-power
Let's take stock of what we have

- Too many spreadsheets
- Too many tasks
- Too little man-power

- Millions of lines of source in different languages
- D
Possible Attack Vectors

1. Check if `whoAmI() == "CTO"`:
   - Yes: Force everybody to use CSV and D
   - No: Request a "one table one worksheet policy"

2. If requested "one table one worksheet policy":
   - Dump spreadsheets into CSV
   - Actually use D

3. Profit
How to work with limited man-power
Leveraging existing libraries

Writing data to spreadsheets
Leveraging existing libraries

Writing data to spreadsheets

- It is required, people will ask for that
- Writing a somewhat feature complete xlsx writer is a huge task
Leveraging existing libraries

Writing data to spreadsheets

- It is required, people will ask for that
- Writing a somewhat feature complete xlsx writer is a huge task

- libxlsxwriter is a feature rich xlsx writer
- Wrapping it by hand,
Leveraging existing libraries

Writing data to spreadsheets

- It is required, people will ask for that
- Writing a somewhat feature complete xlsx writer is a huge task

- libxlsxwriter is a feature rich xlsx writer
- Wrapping it by hand, no way (+78000 lines of structs, enums and functions)
Leveraging existing libraries

Writing data to spreadsheets

- It is required, people will ask for that
- Writing a somewhat feature complete xlsx writer is a huge task

- libxlsxwriter is a feature rich xlsx writer
- Wrapping it by hand, no way (+78000 lines of structs, enums and functions)

- dpp to the rescue
- libxlsxwriter.d (+4000 lines)
- But it is still a C api
void chart_axis_set_name(lxw_chart_axis*, const (char)*)
void chart_axis_set_name_font(lxw_chart_axis*, lxw_chart_font*)
void chart_axis_set_num_font(lxw_chart_axis*, lxw_chart_font*)
void chart_axis_set_num_format(lxw_chart_axis*, const (char)*)
void chart_axis_set_line(lxw_chart_axis*, lxw_chart_line*)
void chart_axis_set_fill(lxw_chart_axis*, lxw_chart_fill*)
...

Semi-automatic refactoring

```c
struct ChartAxis {
    lxw_chart_axis* handle;

    void setName(string name) {
        chart_axis_set_name(this.handle, toStringz(name));
    }

    void setNameRange(string n, lxw_row_t row,
        lxw_col_t col)
    {
        chart_axis_set_name(this.handle, toStringz(n), row, col);
    }
    ...
};
```
Problem to solve: We needed fake data with a variety of attributes.

- Name
- Address
- i18n
- ...
Creating fake data

Problem to solve: We needed fake data with a variety of attributes.

• Name
• Address
• i18n
• ...

25
faker.js

- +160 attributes
- 39 languages
```javascript
module["exports"] = [
    "#{prefix} #{first_name} #{last_name}",
    "#{first_name} #{nobility_title_prefix} #{last_name}",
    "#{first_name} #{last_name}",
    "#{first_name} #{last_name}",
    "#{first_name} #{last_name}",
    "#{first_name} #{last_name}"
];
```

Listing 1: locales/de/name/name.js
```java
override string nameName() {
    switch(uniform(0, 6, this.rnd)) {
        case 0:
            return format!"%s %s %s"(namePrefix(), nameFirstName(),
                nameLastName());
        case 1:
            return format!"%s %s %s"(nameFirstName(), nameNobilityTitlePrefix(),
                nameLastName());
        case 2:
            return format!"%s %s"(nameFirstName(), nameLastName());
        case 3:
            return format!"%s %s"(nameFirstName(), nameLastName());
        case 4:
            return format!"%s %s"(nameFirstName(), nameLastName());
        case 5:
            return format!"%s %s"(nameFirstName(), nameLastName());
        default: assert(false);
    }
}
```
```cpp
import faked;

auto f = new Faker(1337);
writeln(f.nameName());

// localized to german
f = new Faker_de(1338);
writeln(f.nameName());
```
- **Input:**
  - Parser and Generator \(\approx 1500\) lines of D
  - A day of boring work
FakeD

- **Input:**
  - Parser and Generator ≈ 1500 lines of D
  - A day of boring work

- **Output:**
  - Output feature equivalent ≈ 70000 lines faker.js clone
  - Most changes in faker.js just require a rerun of the tool to update
- **Input:**
  - Parser and Generator ≈ 1500 lines of D
  - A day of boring work

- **Output:**
  - Output feature equivalent ≈ 70000 lines faker.js clone
  - Most changes in faker.js just require a rerun of the tool to update

- **Bonus:**
  - Created two PRs to faker.js fixing wrong template expansion
Taking a step back
<table>
<thead>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Amount</th>
<th>Currency</th>
<th>CreatedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hans</td>
<td>Meier</td>
<td>73331</td>
<td>USD</td>
<td>Ruth Ember</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>83431</td>
<td>GPB</td>
<td>Ruth Ember</td>
</tr>
<tr>
<td>Ruth</td>
<td>Ember</td>
<td>103431</td>
<td>EUR</td>
<td>Hans Meier</td>
</tr>
</tbody>
</table>
The starting point

class Employee {
    long id;
    DateTime createdAt;
    EmployeeInfo info;
    long infoId;
}

class EmployeeInfo {
    long id;
    string firstname;
    string lastname;
    Salary salary;
    long salaryId;
}

class Salary {
    long id;
    Employee createdBy;
    long createdById;
    CurrencyAmount amount;
    long amountId;
}

class Currency {
    long id;
    string name;
}

class CurrencyAmount {
    long id;
    double amount;
    Currency currency;
    long currencyId;
}

The vibe.d REST interface

```d
interface Backend {
    Employee[] getAllEmployees();
    Employee getEmployee(long empId);
    EmployeeInfo getEmployeeInfo(long empInfoId);
    Salary getSalary(long salaryId);
    CurrencyAmount getCurrencyAmount(long amountId);
    Currency getCurrency(long currencyId);
}
```
interface Employee {
  id: number;
  createdAt: number;
  info?: EmployeeInfo;
  infoId: number;
}

interface EmployeeInfo {
  id: number;
  firstname: string;
  lastname: string;
  salary?: Salary;
  salaryId: number;
}

interface Salary {
  id: number;
  createdBy?: Employee;
  createdById: number;
  amount?: CurrencyAmount;
  amountId: number;
}

interface CurrencyAmount {
  id: number;
  amount: number;
  currency?: Currency;
  currencyId: number;
}
The Frontend Code: Backend Service

```typescript
import {
    Employee, EmployeeInfo, Salary, CurrencyAmount, Currency
} from "model";

class Backend {
    getAllEmployees() : Employee[] { ... }
    getEmployee(empId: number): Employee { ... }
    getEmployeeInfo(empInfoId: number): EmployeeInfo { ... }
    getSalary(salaryId: number): Salary { ... }
    getCurrencyAmount(amountId: number): CurrencyAmount { ... }
    getCurrency(currencyId: number): Currency { ... }
}
```
The Frontend Code: Calling the Backend

```javascript
this.backend.getAllEmployees().pipe(
  mergeMap((emps: Employee[]) => {
    const obs = [];
    for(const emp of emps) {
      obs.push(this.backend.getEmployeeInfo(emp.infoId)
        .pipe(map((empInfo: EmployeeInfo) => {
          const ne: Employee = {...emp, info : empInfo};
          return ne;
        }));
    }
    return forkJoin(obs);
  }),
  mergeMap((emps: Employee[]) => {
    const obs = [];
    for(const emp of emps) {
      obs.push(this.backend.getSalary(emp.info.salaryId)
        .pipe(map((sal: Salary) => {
          const ne: Employee = {...emp};
          ne.info.salary = sal;
          return ne;
        }));
    }
    return forkJoin(obs);
  }));
```

The Communication
The Takeaways

- Clearly, this is unworkable
- Not plastic at all
- Just a lot of boring work
The real Takeway
So, what do we/I want?

Declare how we want data to get, when we’re asking for it.
Why can't we write this?

```json
{
    allEmployees {
        info {
            firstname
            lastname
            salary {
                amount
                currency {
                    name
                }
            }
        }
    }
    createdBy {
        info {
            firstname
            lastname
        }
    }
}
```
Why can't we write this?

```javascript

1 {  
2   allEmployees {  
3     info {  
4       firstname  
5       lastname  
6       salary {  
7         amount  
8         currency {  
9           name  
10          }  
11         }  
12       createdBy {  
13         info {  
14           firstname  
15           lastname  
16          }  
17         }  
18     }  
19   }  
20 }
```

```javascript

1 {  
2   allEmployees: [ {  
3     info: {  
4       firstname: "Hans" ,  
5       lastname: "Meier" ,  
6       salary: {  
7         amount: 73331,  
8         currency: {  
9           name: "USD"  
10          }  
11         }  
12       createdBy: {  
13         info: {  
14           firstname: "Ruth" ,  
15           lastname: " Ember"  
16          }  
17         }  
18     }  
19   },  
20   ...  
21 ]
```
GraphQL
```graphql
schema {
  query: Backend
}

type Backend {
  getAllEmployees: [Employee]
}

type Employee {
  id: number!
  createdAt: number!
  info: EmployeeInfo
  infoId: number!
}

type EmployeeInfo {
  id: number!
  firstname: String!
  lastname: String!
  salaryId: number!
  salary: Salary
}

type Salary {
  id: number!
  createdBy: Employee
  amountId: number!
  amount: CurrencyAmount
}

type CurrencyAmount {
  id: number!
  amount: number!
  currencyId: number!
  currency: Currency
}

type Currency {
  id: number!
  name: string!
}
```
query one {
  allEmployees {
    ...deep
  }
}

fragment names on EmployeeInfo {
  firstname
  lastname
}

fragment empInfo on Employee {
  info {
    ...names
  }
}

fragment deep on Employee {
  info {
    ...names
    salary {
      amount
      currency {
        name
      }
      createdBy {
        ...empInfo
      }
    }
  }
}

Less code is better
Introspecting Types

```json
{
  __type(name: "Employee") {
    name
    fields {
      name
type {
      name
kind
      ofType {
        name
      }
    }
  }
}

"data": {
  "__type": {
"name": "Employee",
"fields": [
{
  "name": "id",
  "type": {
    "name": null,
    "kind": "NON_NULL",
    "ofType": {
      "name": "INT"
    }
  }
},
{
  "name": "info",
  "type": {
    "name": "EmployeeInfo",
    "kind": "OBJECT"
  }
}
]
}
```
Search results for: `graphql`

<table>
<thead>
<tr>
<th>Package</th>
<th>Latest version</th>
</tr>
</thead>
</table>

Found 0 packages.
WHAT ARE YOU
import graphqld;

interface Query {
    Employee[] getAllEmployees();
}

class Schema {
    Query queryType;
}

auto graphqld = new GraphQLD !( Schema )();

graphqld . setResolver (
    "queryType", "getAllEmployees",
    delegate ( string name , Json parent ,
    Json args , ref Context context ) @safe
    {
        Employee[] employees = getAllEmployees ();
        Json ret = Json . emptyObject ();
        ret [" data "] = toGraphqlJson ( employees );
        return ret ;
    });
GraphQLD

```csharp
import graphqld;

interface Query {
    Employee[] getAllEmployees();
}

class Schema {
    Query queryType;
}

auto graphqld = new GraphQLD!(Schema)();

graphql.setResolver(
    "queryType", "getAllEmployees",
    delegate(string name, Json parent,
              Json args, ref Context context) @safe
    {
        Employee[] employees = getAllEmployees();
        Json ret = Json.emptyObject();
        ret["data"] = toGraphqlJson(employees);
        return ret;
    });
```
void graphqlEndpoint(HTTPServerRequest req, HTTPServerResponse res)
{
    string toParse = extractQuery(req);
    auto p = Parser(Lexer(toParse));

    Document d = p.parseDocument();
    auto fv = new QueryValidator(d);
    auto sv = new SchemaValidator!Schema(d, graphqld.schema);
    fv.accept(d);
    sv.accept(d);

    Context con = buildContext(req);
    Json ret = graphqld.execute(d, extractVariables(req), con);
    res.writeJsonBody(ret);
}
graphqld.setResolver("Employee", "info",
    delegate(string name, Json parent, Json args,
        ref Context context)
    {
        const id = parent["infoId"].get!long();
        EmployeeInfo ei = getEmployeeInfo(id);
        Json ret = Json.emptyObject();
        ret["data"] = toGraphqlJson(ei);
        return ret;
    });
- Mostly feature complete
- Some validations are missing
- $\approx 17000$ lines
- $\approx 9000$ lines are generated by darser
- ready for use now
Homework
Write a GraphQL backend that uses an excel spreadsheet as a database.
Conclusion
Conclusion

- Spreadsheets are a terrible programming language.
- C++ and Rust are not our main competition on our path to world domination.
Conclusion

- Spreadsheets are a terrible programming language.
- C++ and Rust are not our main competition on our path to world domination.

- We need to learn to use what is there.
- Use D to work smart not hard.
- Do not write the code, write the code that writes the code.
- Look at JS for inspiration.
- GraphQL 👍
The End
[1] *Infographic: C/C++ facts we learned before going ahead with CLion.*


https://www.youtube.com/watch?v=bf0Rr81is6U. (Accessed on 04/08/2019).


[9] *burner/graphqld*: A *vibe.d* library to handle the *GraphQL* Protocol written in the

Encore
struct Employee {
  @GQLD(
    Description("The social security number of an employee"),
    Deprecated(IsDeprecated.yes, "To complex")
  )
  SocialSecurityNumber number;
}
Reappearing UDA Pattern

```
struct Employee {
  @GQLD(
    Description("The social security number of an employee"),
    Deprecated(IsDeprecated.yes, "To complex")
  )
  SocialSecurityNumber number;
}

enum IsDeprecation {
  undefined,
  no,
  yes
}

struct GQLDData {
  Description desc;
  Deprecated depre;
}
```
Reappearing UDA Pattern

```cpp
struct GQLDData {
    Description desc;
    Deprecated depre;
};

GQLDData GQLD(Args...)(Args args) {
    GQLDData ret;
    static foreach (mem; __traits(allMembers, GQLDData)) {
        static foreach (arg; args) {
            static if(is(typeof(__traits(getMember, ret, mem)) ==
                typeof(arg)))
            {
                __traits(getMember, ret, mem) = arg;
            }
        }
    }
    return ret;
}
```
Json ret = Json.emptyObject();

string typename = ...;

l: switch typename { 
  static foreach type; collectTypes!(T) {{
    case typeToTypeName!(type): {
      ret["data"] = typeToJson!(type)();
      break l;
    }
  }}}} 

  default: break;
}

return ret;
Collecting all Referenced Types

```cpp
alias allTypes = collectTypes!Schema;

template collectTypesImpl(Type) {
  import graphql.uda;
  static if(is(Type : GQLDCustomLeaf!F, F)) {
    alias collectTypesImpl = AliasSeq!(Type);
  } else static if(is(Type == interface)) {
    alias RetTypes = AliasSeq!(collectReturnType!(Type,
      __traits(allMembers, Type)));
    alias ArgTypes = AliasSeq!(collectParameterTypes!(Type,
      __traits(allMembers, Type)));
    alias collectTypesImpl = AliasSeq!(Type, RetTypes,
      ArgTypes, InterfacesTuple!Type);
  } else static if(is(Type == union)) {
    alias collectTypesImpl = AliasSeq!(Type, InheritedClasses!Type);
  } else static if(is(Type : Nullable!F, F)) {
    alias collectTypesImpl = .collectTypesImpl!(F);
  } ...
```
template InheritedClass(T) {
    import std.meta : staticMap, AliasSeq, NoDuplicates;
    import std.traits : Select;

    alias getInheritedFields() = staticMap!(.InheritedClass, FieldTypeTuple!T);
    alias ftt = Select!(is(T == union), getInheritedFields, AliasSeq);

    alias getBaseTuple() = staticMap!(.InheritedClass, BaseClassesTuple!T);
    alias clss = Select!(is(T == class), getBaseTuple, AliasSeq);

    alias getInter() = staticMap!(.InheritedClass, InterfacesTuple!T);
    alias inter = Select!(is(T == class) || is(T == interface),
                           getInter,
                           AliasSeq
                          );

    alias InheritedClass = NoDuplicates!(AliasSeq!(ftt!(), clss!(), inter!()));
}
Compile-Time are long as

- \( \approx 7000 \) lines
- \( \approx 8 \) seconds build time
Overall Legacy Architecture

- Traditional compiler pipeline design is dated
Overall Legacy Architecture

- Traditional compiler pipeline design is dated
- We have practically unlimited memory
- Recreating the AST, IR, and ASM on every compile is extremely wasteful
- Why does code-completion and the compiler use different frontends
Compiler Re-arch idea

**CLI Driver PID_2**

The CLI interface is used to give commands to the Compiler Process

**Editor PID_3**

Your favorite editor or IDE here

**Compiler Process PID_1**

- SymTable
- AST
- IR
- CodeCompletion

- Compiler Process is a daemon
  Caches result of lexing, parsing, validation, and stores dependencies of these results

Initiate compile
Get code completion info, etc.
Darser

- Darser is a recursive descent parser generator for LL(1) grammars
- It also generates the AST and a default Visitor
- Not at CT, but as a pre-build step
Darser is a recursive descent parser generator for LL(1) grammars.

It also generates the AST and a default Visitor.

Not at CT, but as a pre-build step.

It generates “good” error messages.

Not just a generic Node types, but names that reflect the grammar.

Inheriting from the default Visitor is trivial and powerful.

Used right now by GraphQLD.
I am out of Slides