

# 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

≈ 750 Million (2016)





**BUT EXCEL IS  
NOT PROGRAMMING**

# Oh, but it is

2001Financialstatements.xls - Microsoft Excel						
	A	B	C	D	E	F
1	Consolidated Statements of Shareholders' Equity					
2	(DOLLARS IN THOUSANDS)					
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						
39						
40						
41						
42						
43						
44						
45						
46						
47						
48						
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						
60						
61						
62						
63						
64						
65						
66						
67						
68						
69						
70						
71						
72						
73						
74						
75						
76						
77						
78						
79						
80						
81						
82						
83						
84						
85						
86						
87						
88						
89						
90						
91						
92						
93						
94						
95						
96						
97						
98						
99						
100						
101						
102						
103						
104						
105						
106						
107						
108						
109						
110						
111						
112						
113						
114						
115						
116						
117						
118						
119						
120						
121						
122						
123						
124						
125						
126						
127						
128						
129						
130						
131						
132						
133						
134						
135						
136						
137						
138						
139						
140						
141						
142						
143						
144						
145						
146						
147						
148						
149						
150						
151						
152						
153						
154						
155						
156						
157						
158						
159						
160						
161						
162						
163						
164						
165						
166						
167						
168						
169						
170						
171						
172						
173						
174						
175						
176						
177						
178						
179						
180						
181						
182						
183						
184						
185						
186						
187						
188						
189						
190						
191						
192						
193						
194						
195						
196						
197						
198						
199						
200						
201						
202						
203						
204						
205						
206						
207						
208						
209						
210						
211						
212						
213						
214						
215						
216						
217						
218						
219						
220						
221						
222						
223						
224						
225						
226						
227						
228						
229						
230						
231						
232						
233						
234						
235						
236						
237						
238						
239						
240						
241						
242						
243						
244						
245						
246						
247						
248						
249						
250						
251						
252						
253						
254						
255						
256						
257						
258						
259						
260						
261						
262						
263						
264						
265						
266						
267						
268						
269						
270						
271						
272						
273						
274						
275						
276						
277						
278						
279						
280						
281						
282						
283						
284						
285						
286						
287						
288						
289						
290						
291						
292						
293						
294						
295						
296						
297						
298						
299						
300						
301						
302						
303						
304						
305						
306						
307						
308						
309						
310						
311						
312						
313						
314						
315						
316						
317						
318						
319						
320						
321						
322						
323						
324						
325						
326						
327						
328						
329						
330						
331						
332						
333						
334						

## A little bit of Spreadsheet bashing

---

## Seeing the code is difficult

The screenshot shows a spreadsheet interface with a formula bar at the top and a table below it.

**Formula Bar:**

- Cell reference: B1
- Icon: A dropdown arrow icon.
- Icon: A small icon with three dots and a yellow star.
- Icon: A Greek letter Σ (Sigma).
- Icon: An equals sign (=).
- Text: =A1 \* 13.37

**Table:**

	A	B	C
1	10	133.7	
2	14	187.18	
3			

## Dynamic Types

	A	B	C
1	Firstname	Lastname	Age
2	John	Doe	34
3	Hans	Mustermann	twenty-five
4			

## Dynamic Types

D4	A	B	C
1	Firstname	Lastname	Age
2	John	Doe	34
3	Hans	Mustermann	25

## Dynamic Types

The screenshot shows a spreadsheet interface with a toolbar at the top and a data table below. The toolbar includes a dropdown menu, a formula editor icon, a sum icon ( $\Sigma$ ), an equals sign icon (=), and a value '25'. The table has columns labeled A, B, and C. Row 1 contains column headers: '1' under A, 'Firstname' under B, and 'Lastname' under C. Row 2 contains data: '2' under A, 'John' under B, and '34' under C. Row 3 contains data: '3' under A, 'Hans' under B, and 'Mustermann' under C. The cell containing 'Mustermann' is currently selected.

C3	A	B	C
1	Firstname	Lastname	Age
2	John	Doe	34
3	Hans	Mustermann	25



**git blame**

git blame

**git blame**

git blame

lets not go there

we will just become sad

## Code refactoring

- =SUM(1,2)

## 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

- =SUM(1,2)
- equal, identifier, lparen, int(1), comma, int(2), rparen
- set Excel locale to de\_DE
- =SUM(1,2)
- equal, identifier, lparen, float(1.2), rparen

## Bits and Pieces

- 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?

## (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

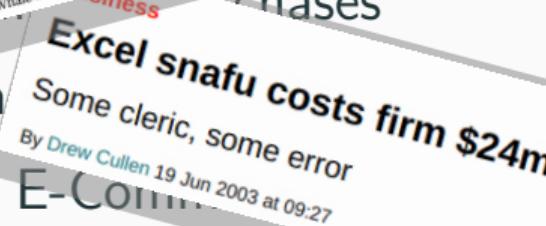


# (Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to business partners
3. Use spreadsheet to make purchases
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce



The trader known as the London Whale lost at least \$6.2 billion for a bank



Some cleric, some error  
By Drew Cullen 19 Jun 2003 at 09:27

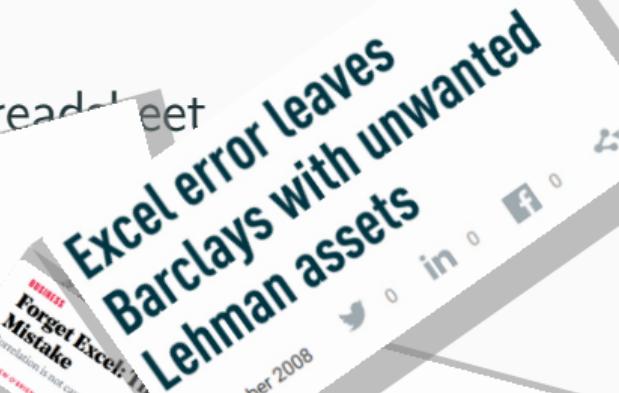
# (Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to business partners
3. Use spreadsheet to make large purchases
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce



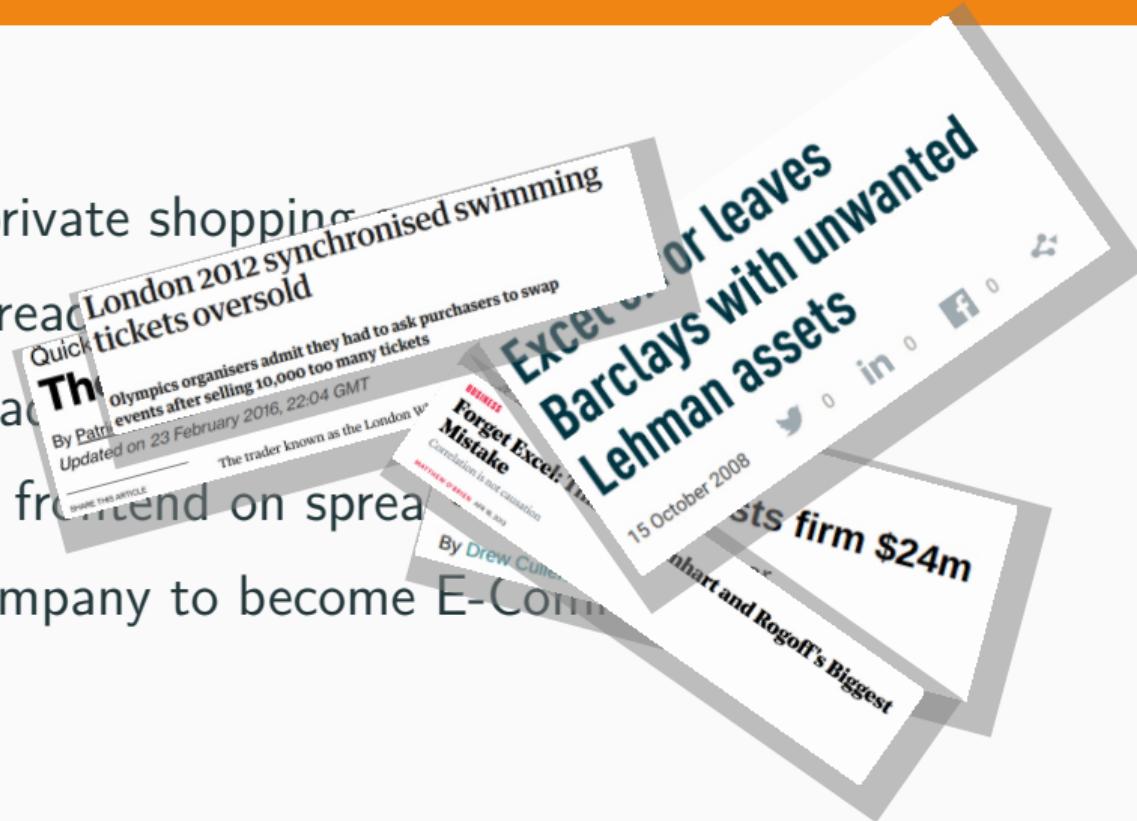
# (Typical) Spreadsheet Lifecycle

1. Create private shopping spreadsheet
2. Show spreadsheet to colleagues
3. Use spreadsheet to make money
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce



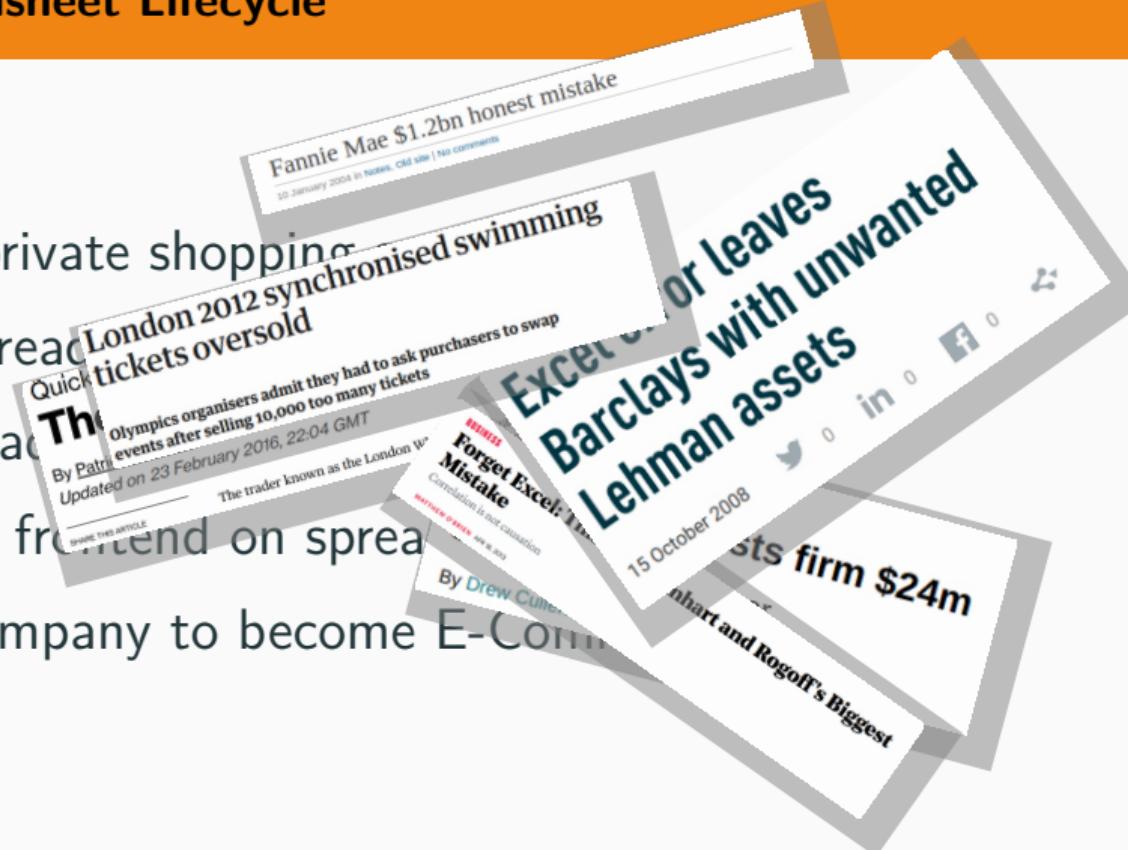
# (Typical) Spreadsheet Lifecycle

1. Create private shopping list
2. Show spreadsheet to friends
3. Use spreadsheet to calculate
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce



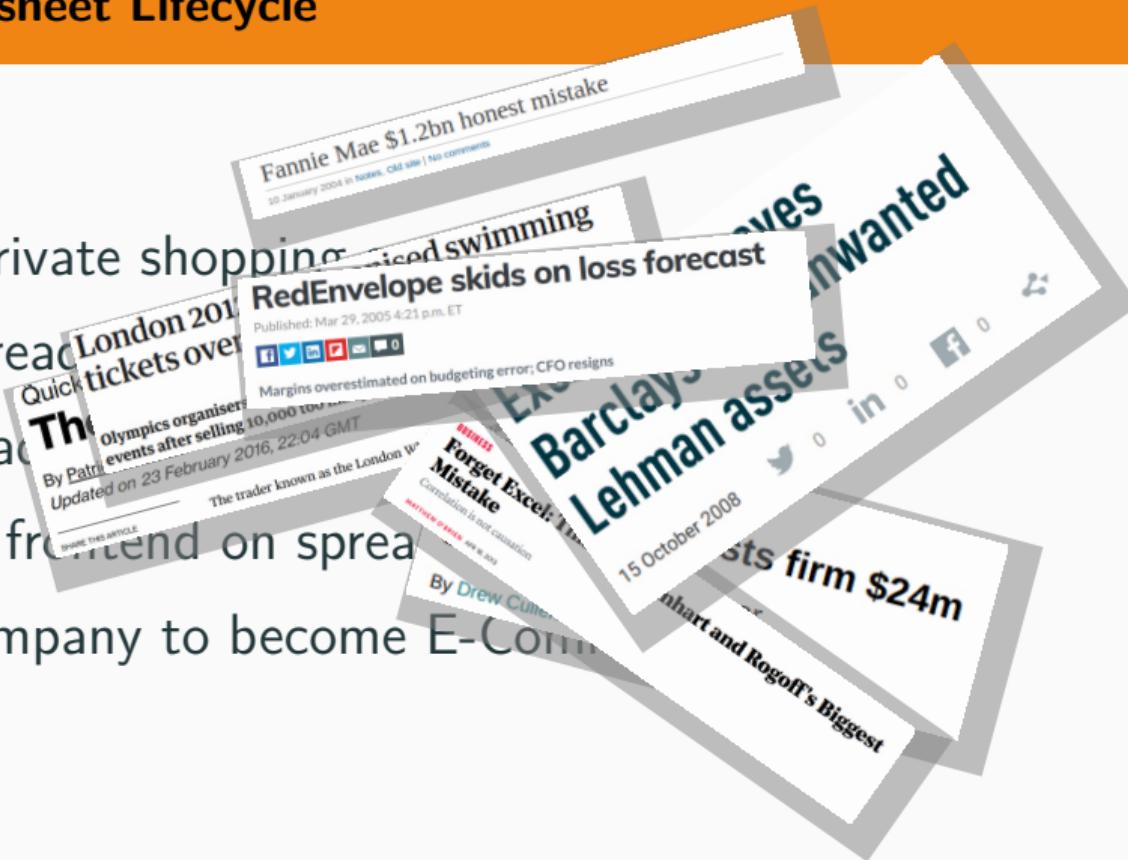
# (Typical) Spreadsheet Lifecycle

1. Create private shopping list
2. Show spreadsheet to friends
3. Use spreadsheet to calculate
4. Put web frontend on spreadsheet
5. Pivot company to become E-Commerce



# (Typical) Spreadsheet Lifecycle

1. Create private shopping list
2. Show spreadsheets to investors
3. Use spreadsheets to manage risk
4. Put web frontend on spreadsheets
5. Pivot company to become E-Commerce



# (Typical) Spreadsheet Lifecycle

1. Create private shopping list
  2. Show spreadsheet to your boss
  3. Use spreadsheet to calculate ROI
  4. Put web frontend on it
  5. Pivot company to become E-commerce
-

# (Typical) Spreadsheet Lifecycle

1. Create private spreadsheet
  2. Show spreadsheets to investors
  3. Use spreadsheets to manage business causes
  4. Put web frontend on spreadsheets
  5. Pivot company to become E-commerce
-

# (Typical) Spreadsheet Lifecycle

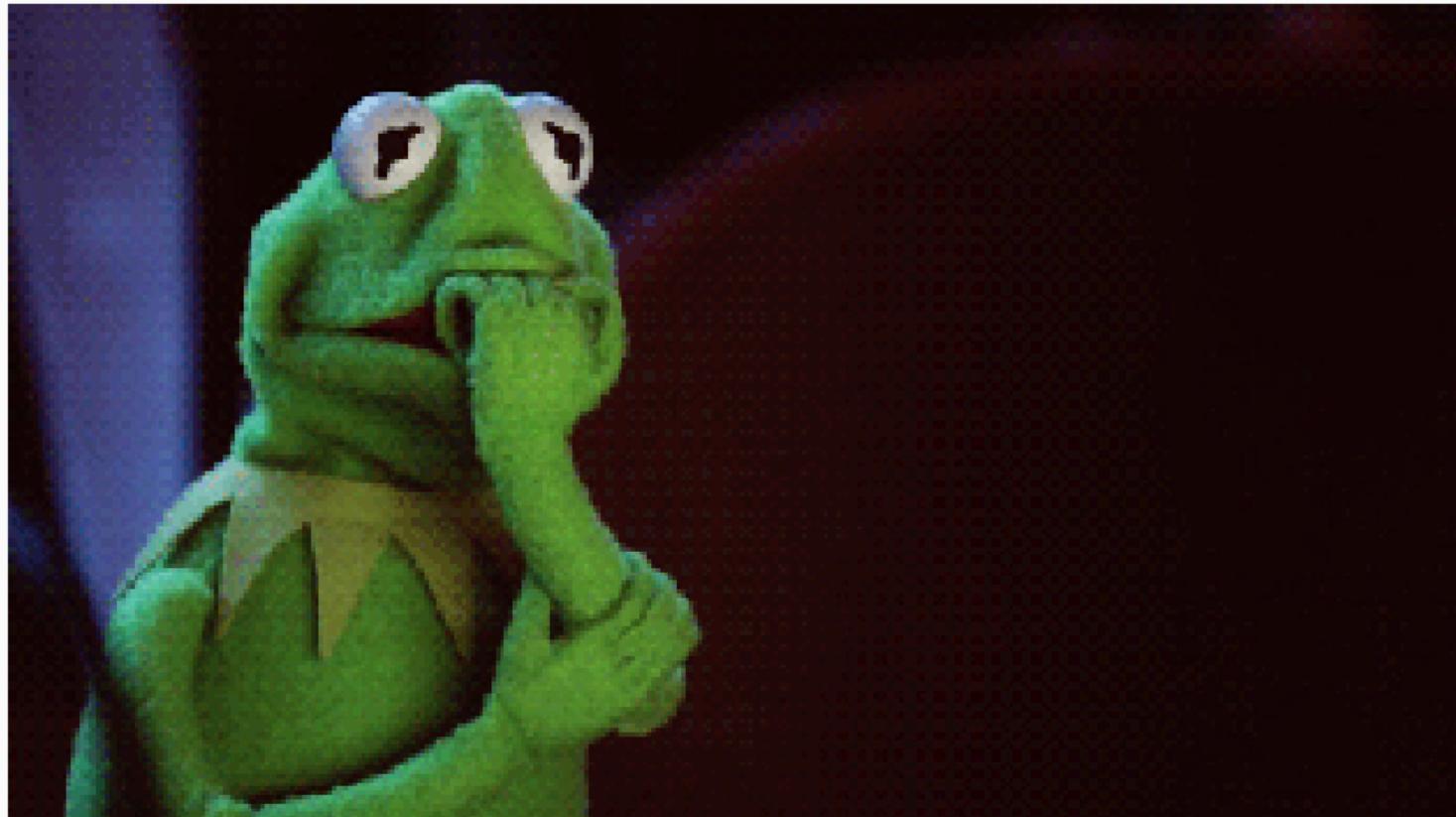
1. Create problem
2. Show spreadsheet errors
3. Use spreadsheets
4. Put web pages online
5. Pivot control

The collage includes:

- A screenshot of the EuSpRIG homepage featuring a banner about a £2bn honest mistake.
- A screenshot of the "Original Horror Stories" section, showing a story about calculating percentage change.
- A screenshot of the "Scoring an own goal" section, showing a story about the Office of Government Commerce's spreadsheet error.
- A screenshot of the "in" social media page.
- A screenshot of the "Facebook" social media page.
- A screenshot of the "Twitter" social media page.

# **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!

**Lets draw up a battle plan**

---

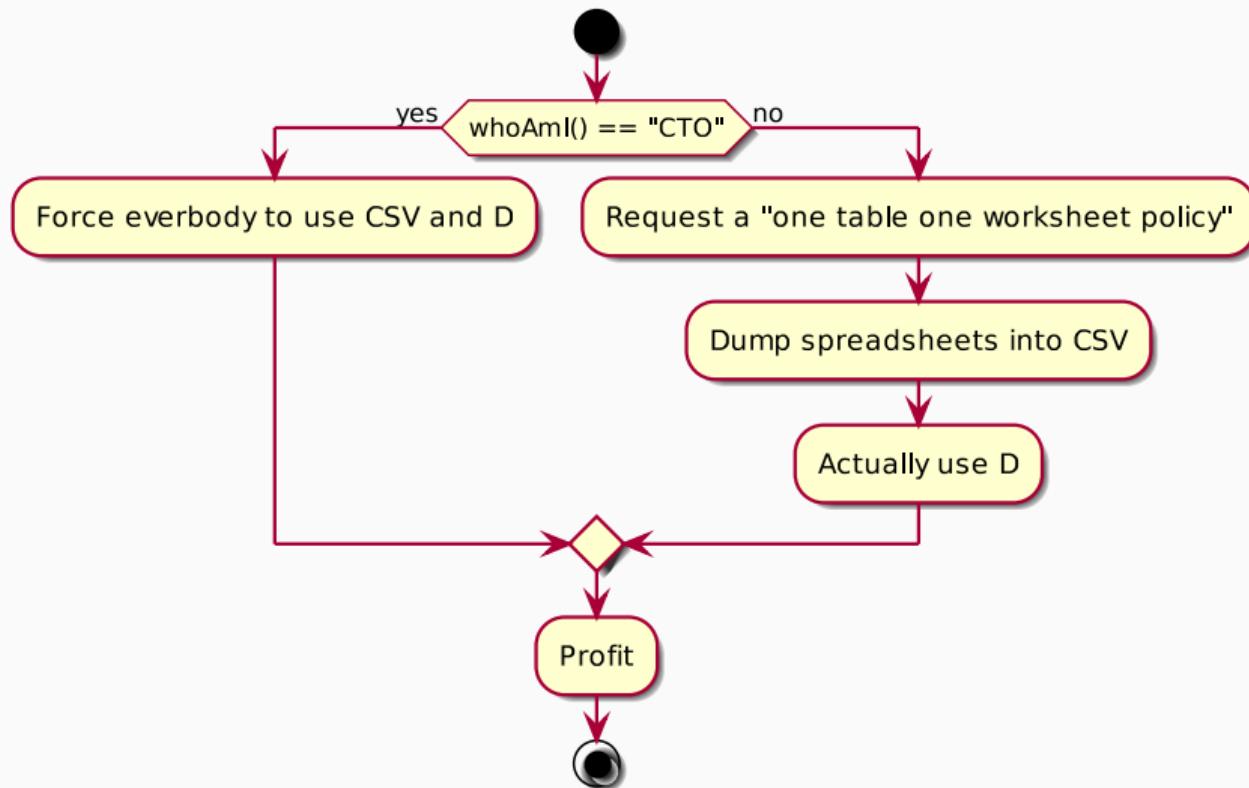
## Lets take stock of what we have

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

## Lets 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



## **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

## dpp output

```
1 void chart_axis_set_name(lxw_chart_axis*, const(char)*)  
2 void chart_axis_set_name_font(lxw_chart_axis*, lxw_chart_font*)  
3 void chart_axis_set_num_font(lxw_chart_axis*, lxw_chart_font*)  
4 void chart_axis_set_num_format(lxw_chart_axis*, const(char)*)  
5 void chart_axis_set_line(lxw_chart_axis*, lxw_chart_line*)  
6 void chart_axis_set_fill(lxw_chart_axis*, lxw_chart_fill*)  
7 ...
```

## Semi-automatic refactoring

```
1 struct ChartAxis {
2     lxw_chart_axis* handle;
3
4     void setName(string name) {
5         chart_axis_set_name(this.handle, toStringz(name));
6     }
7
8     void setNameRange(string n, lxw_row_t row,
9                         lxw_col_t col)
10    {
11        chart_axis_set_name(this.handle, toStringz(n), row, col);
12    }
13    ...
14 }
```

## Creating fake data

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

- Name
- Address
- i18n
- ...

Cre



# faker.js

- +160 attributes
- 39 languages

## faker.js

```
1 module["exports"] = [
2     "#{prefix} #{first_name} #{last_name}",
3     "#{first_name} #{nobility_title_prefix} #{last_name}",
4     "#{first_name} #{last_name}",
5     "#{first_name} #{last_name}",
6     "#{first_name} #{last_name}",
7     "#{first_name} #{last_name}"
8 ];
```

Listing 1: locales/de/name/name.js

```
1  override string nameName() {
2      switch(uniform(0, 6, this.rnd)) {
3          case 0:
4              return format! "%s %s %s"(namePrefix(), nameFirstName(),
5                  nameLastName());
6          case 1:
7              return format! "%s %s %s"(nameFirstName(), nameNobilityTitlePrefix(),
8                  nameLastName());
9          case 2:
10             return format! "%s %s"(nameFirstName(), nameLastName());
11         case 3:
12             return format! "%s %s"(nameFirstName(), nameLastName());
13         case 4:
14             return format! "%s %s"(nameFirstName(), nameLastName());
15         case 5:
16             return format! "%s %s"(nameFirstName(), nameLastName());
17         default: assert(false);
18     }
```

# FakeD

```
1 import faked;
2
3 auto f = new Faker(1337);
4 writeln(f.nameName());
5
6 // localized to german
7 f = new Faker_de(1338);
8 writeln(f.nameName());
```

- Input:
  - Parser and Generator  $\approx$  1500 lines of D
  - A day of boring work

- Input:
  - Parser and Generator  $\approx$  1500 lines of D
  - A day of boring work
- Output:
  - Output feature equivalent  $\approx$  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

---

## The Wanted Output: Salary Table

Firstname	Lastname	Amount	Currency	CreatedBy
Hans	Meier	73331	USD	Ruth Ember
John	Doe	83431	GPB	Ruth Ember
Ruth	Ember	103431	EUR	Hans Meier

## The starting point

```
1  class Employee {          17  class Salary {           32  class Currency {  
2    long id;              18  long id;             33  long id;  
3    DateTime createdAt;   19  Employee createdBy;  34  string name;  
4    EmployeeInfo info;    20  long createdById;   35  }  
5    EmployeeInfo info;    21  
6    long infoId;          22  CurrencyAmount amount;  
7  }                      23  long amountId;  
8  
9  class EmployeeInfo {    24  }  
10  long id;               25  
11  string firstname;     26  class CurrencyAmount {  
12  string lastname;       27  long id;  
13  
14  Salary salary;        28  double amount;  
15  long salaryId;        29  
16  }                      30  Currency currency;  
                           31  long currencyId;  
                           32  }
```

## The vibe.d REST interface

```
1 interface Backend {  
2     Employee[] getAllEmployees();  
3     Employee getEmployee(long empId);  
4     EmployeeInfo getEmployeeInfo(long empInfoId);  
5     Salary getSalary(long salaryId);  
6     CurrencyAmount getCurrencyAmount(long amountId);  
7     Currency getCurrency(long currencyId);  
8 }
```

## The Frontend Code: Types

```
1  interface Employee {          17  interface Salary {  
2    id: number;                18    id: number;  
3    createdAt: number;         19    createdBy?: Employee;  
4                                20    createdById: number;  
5    info?: EmployeeInfo;       21  
6    infoId: number;           22    amount?: CurrencyAmount;  
7  }                           23    amountId: number;  
8                                24 }  
9  interface EmployeeInfo {      25  
10   id: number;                26  interface CurrencyAmount {  
11   firstname: string;         27   id: number;  
12   lastname: string;          28   amount: number;  
13                                29  
14   salary?: Salary;           30   currency?: Currency;  
15   salaryId: number;          31   currencyId: number;  
16 }                           32 }
```

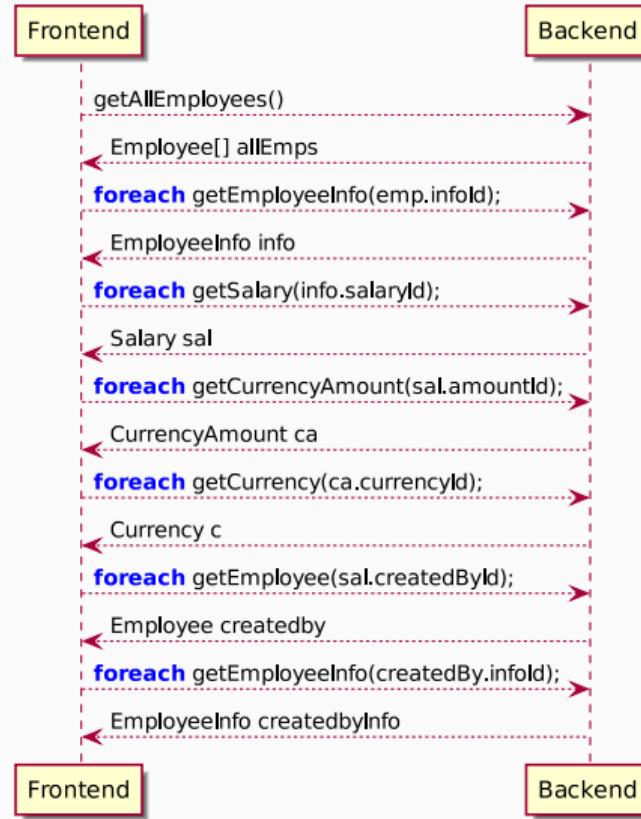
## The Frontend Code: Backend Service

```
1 import {  
2     Employee, EmployeeInfo, Salary, CurrencyAmount, Currency  
3 } from "model";  
  
4  
5 class Backend {  
6     getAllEmployees(): Employee[] { ... }  
7     getEmployee(empId: number): Employee { ... }  
8     getEmployeeInfo(empInfoId: number): EmployeeInfo { ... }  
9     getSalary(salaryId: number): Salary { ... }  
10    getCurrencyAmount(amountId: number): CurrencyAmount { ... }  
11    getCurrency(currencyId: number): Currency { ... }  
12 }
```

## The Frontend Code: Calling the Backend

```
1  this.backend.getAllEmployees().pipe(  
2    mergeMap((emps: Employee[]) => {  
3      const obs = [];  
4      for(const emp of emps) {  
5        obs.push(this.backend.getEmployeeInfo(emp.infoId)  
6          .pipe(map((empInfo: EmployeeInfo) => {  
7            const ne: Employee = {...emp, info : empInfo};  
8            return ne;  
9          }))  
10         )  
11       );  
12     }  
13     return forkJoin(obs);  
14   },  
15   mergeMap((emps: Employee[]) => {  
16     const obs = [];
```

# The Communication



## The Takeaways

- Clearly, this is unworkable
- Not plastic at all
- Just a lot of boring work

## The real Takeaway



MAKE GIFS AT GIFSOUP.COM

## So, what do we/I want?

Declare how we want data to get, when we're asking for it.

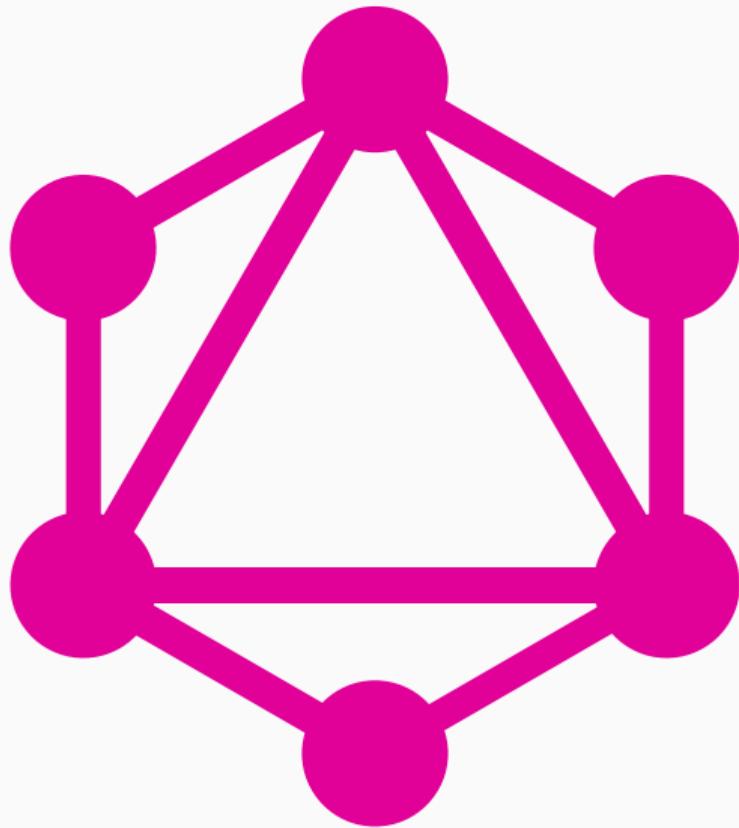
## Why can't we write this?

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

## Why can't we write this?

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

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



# GraphQL

```
1 schema {  
2   query: Backend  
3 }  
4  
5 type Backend {  
6   getAllEmployees: [Employee]  
7 }  
8  
9 type Employee {  
10   id: number!;  
11   createdAt: number!;  
12  
13   info: EmployeeInfo;  
14   infoId: number!;  
15 }  
16  
17 type EmployeeInfo {  
18   id: number!;  
19   firstname: String!;  
20   lastname: String!;  
21   salaryId: number!;  
22   salary: Salary;  
23 }  
24  
25 type Salary {  
26   id: number!;  
27   createdBy: Employee;  
28   amountId: number!;  
29   amount: CurrencyAmount;  
30 }  
31  
32 type CurrencyAmount {  
33   id: number!;  
34   amount: number!;  
35   currencyId: number!;  
36   currency: Currency;  
37 }
```

## Less code is better

```
1 query one {  
2   allEmployees {  
3     ...deep  
4   }  
5 }  
6  
7 fragment names on EmployeeInfo {  
8   firstname  
9   lastname  
10 }  
11  
12 fragment empInfo on Employee {  
13   info {  
14     ...names  
15   }  
16 }  
  
1   fragment deep on Employee {  
2     info {  
3       ...names  
4       salary {  
5         amount  
6         currency {  
7           name  
8         }  
9         createdBy {  
10           ...empInfo  
11         }  
12       }  
13     }  
14   }
```

# Introspecting Types

```
1  {
2      __type(name: "Employee") {
3          name
4          fields {
5              name
6              type {
7                  name
8                  kind
9                  ofType {
10                     name
11                 }
12             }
13         }
14     }
15 }
```

```
1  {
2      "data": {
3          "__type": {
4              "name": "Employee",
5              "fields": [
6                  {
7                      "name": "id",
8                      "type": {
9                          "name": null,
10                         "kind": "NON_NULL",
11                         "ofType": {
12                             "name": "INT"
13                         }
14                     }
15                 },
16                 {
17                     "name": "info",
18                     "type": {
19                         "name": "EmployeeInfo",
20                         "kind": "OBJECT"
21                     }
22                 }
23             ]
24         }
25     }
26 }
```

# D's GraphQL Story

[Packages](#)[Documentation ▾](#)[About ▾](#)[Download](#)[Log in](#)

Search results for: *graphql*

Package	Latest version
---------	----------------

Found 0 packages.

JUST DO IT



```
1 import graphqld;
2
3 interface Query {
4     Employee[] getAllEmployees();
5 }
6
7 class Schema {
8     Query queryType;
9 }
```

```
1 import graphqld;
2
3 interface Query {
4     Employee[] getAllEmployees();
5 }
6
7 class Schema {
8     Query queryType;
9 }
```

```
1     auto graphqld = new GraphQL!(Schema)();
2
3     graphqld.setResolver(
4         "queryType", "getAllEmployees",
5         delegate(string name, Json parent,
6                  Json args, ref Context context) @safe
7         {
8             Employee[] employees = getAllEmployees();
9             Json ret = Json.emptyObject();
10            ret["data"] = toGraphqlJson(employees);
11            return ret;
12        });
13 }
```

## GraphQLD

```
1 void graphqlEndpoint(HTTPSRequest req,
2     HTTPSResponse res)
3 {
4     string toParse = extractQuery(req);
5     auto p = Parser(Lexer(toParse));
6
7     Document d = p.parseDocument();
8     auto fv = new QueryValidator(d);
9     auto sv = new SchemaValidator!Schema(d, graphqld.schema);
10    fv.accept(d);
11    sv.accept(d);
12
13    Context con = buildContext(req);
14    Json ret = graphqld.execute(d, extractVariables(req), con);
15    res.writeJsonBody(ret);
16 }
```

# GraphQL

```
1 graphQL.setResolver("Employee", "info",
2     delegate(string name, Json parent, Json args,
3             ref Context context)
4 {
5     const id = parent["infoId"].get!long();
6     EmployeeInfo ei = getEmployeeInfo(id);
7     Json ret = Json.emptyObject();
8     ret["data"] = toGraphqlJson(ei);
9     return ret;
10 });
```

- Mostly feature complete
- Some validations are missing
- ≈ 17000 lines
- ≈ 9000 lines are generated by parser
- ready for use now

# Homework

---

## 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://blog.jetbrains.com/clion/2015/07/infographics-cpp-facts-before-clion/>. (Accessed on 04/08/2019).
- [2] Developer Economics. *Developer Economics: State of the Developer Nation 15th Edition*. <https://www.developereconomics.com/reports/state-of-the-developer-nation-15th-edition>. (Accessed on 04/08/2019).
- [3] Microsoft. *Build 2016 Keynote (Day 2)*.  
<https://www.youtube.com/watch?v=bf0Rr81is6U>. (Accessed on 04/08/2019).
- [4] Irish Tech News. *Seven reasons why Excel is still used by half a billion people worldwide*. <https://irishtechnews.ie/seven-reasons-why-excel-is-still-used-by-half-a-billion-people-worldwide/>. (Accessed on 04/08/2019).

- [5] Felienne Hermans. *GOTO 2016: Pure Functional Programming in Excel by Felienne Hermans*. <https://www.youtube.com/watch?v=0yKf8TrLU0w>. (Accessed on 04/08/2019).
- [6] Marak/faker.js: generate massive amounts of realistic fake data in Node.js and the browser. <https://github.com/marak/Faker.js/>. (Accessed on 05/03/2019).
- [7] kaleidicassociates/faked: D library to create real fake data.  
<https://github.com/kaleidicassociates/faked>. (Accessed on 05/03/2019).
- [8] GraphQL | A query language for your API. <https://graphql.org/>. (Accessed on 05/03/2019).

- [9] *burner/graphqld: A vibe.d library to handle the GraphQL Protocol written in the D Programming Language.* <https://github.com/burner/graphqld>. (Accessed on 05/03/2019).
- [10] *burner/Darser: LL1 Parser Generator for D.* <https://github.com/burner/Darser>. (Accessed on 05/03/2019).

**Encore**

---

## Reappearing UDA Pattern

```
1 struct Employee {  
2     @GQLD(  
3         Description("The social security number of an employee"),  
4         Deprecated(IsDeprecated.yes, "To complex")  
5     )  
6     SocialSecurityNumber number;  
7 }
```

## Reappearing UDA Pattern

```
1 struct Employee {
2     @GQLD(
3         Description("The social security number of an employee"),
4         Deprecated(IsDeprecated.yes, "To complex")
5     )
6     SocialSecurityNumber number;
7 }
8
9 enum IsDeprecated {
10     undefined,
11     no,
12     yes
13 }
14
15 struct GQLDData {
16     Description desc;
17     Deprecated depre;
18 }
```

## Reappearing UDA Pattern

```
1 struct GQLDData {  
2     Description desc;  
3     Deprecated depre;  
4 }  
5  
6 GQLDData GQLD(Args...)(Args args) {  
7     GQLDData ret;  
8     static foreach(mem; __traits(allMembers, GQLDData)) {  
9         static foreach(arg; args) {  
10             static if(is(typeof(__traits(getMember, ret, mem)) ==  
11                         typeof(arg)))  
12             {  
13                 __traits(getMember, ret, mem) = arg;  
14             }  
15         }  
16     }  
17     return ret;  
18 }
```

## Static Foreach Switch Case

```
1  Json ret = Json.emptyObject();
2  string typename = ...;
3  l: switch(typename) {
4      static foreach(type; collectTypes!(T)) {{
5          case typeToTypeName!(type): {
6              ret["data"] = typeToJson!(type)();
7              break l;
8          }
9      }}
10     default: break;
11 }
12 return ret;
```

## Collecting all Referenced Types

```
1 alias allTypes = collectTypes!Schema;
2
3 template collectTypesImpl(Type) {
4     import graphql.uda;
5     static if(is(Type : GQLDCustomLeaf!F, F)) {
6         alias collectTypes Impl= AliasSeq!(Type);
7     } else static if(is(Type == interface)) {
8         alias RetTypes = AliasSeq!(collectReturnType!(Type,
9             __traits(allMembers, Type)));
10        alias ArgTypes = AliasSeq!(collectParameterTypes!(Type,
11            __traits(allMembers, Type)));
12        alias collectTypesImpl = AliasSeq!(Type, RetTypes,
13            ArgTypes, InterfacesTuple!Type);
14    ....
15 } else static if(is(Type == union)) {
16     alias collectTypesImpl = AliasSeq!(Type, InheritedClasses!Type);
17 } else static if(is(Type : Nullable!F, F)) {
18     alias collectTypesImpl = .collectTypesImpl!(F);
19 }
```

## Type trinary expression

```
1 template InheritedClass(T) {
2     import std.meta : staticMap, AliasSeq, NoDuplicates;
3     import std.traits : Select;
4
5     alias getInheritedFields() = staticMap!(.InheritedClass, FieldTypeTuple!T);
6     alias ftt = Select!(is(T == union), getInheritedFields, AliasSeq);
7
8     alias getBaseTuple() = staticMap!(.InheritedClass, BaseClassesTuple!T);
9     alias clss = Select!(is(T == class), getBaseTuple, AliasSeq);
10
11    alias getInter() = staticMap!(.InheritedClass, InterfacesTuple!T);
12    alias inter = Select!(is(T == class) || is(T == interface),
13                          getInter,
14                          AliasSeq
15    );
16
17    alias InheritedClass = NoDuplicates!(AliasSeq!(fft!(), clss!(), inter!()));
18 }
```

## Compile-Time are long as

- $\approx$  7000 lines
- $\approx$  8 seconds build time

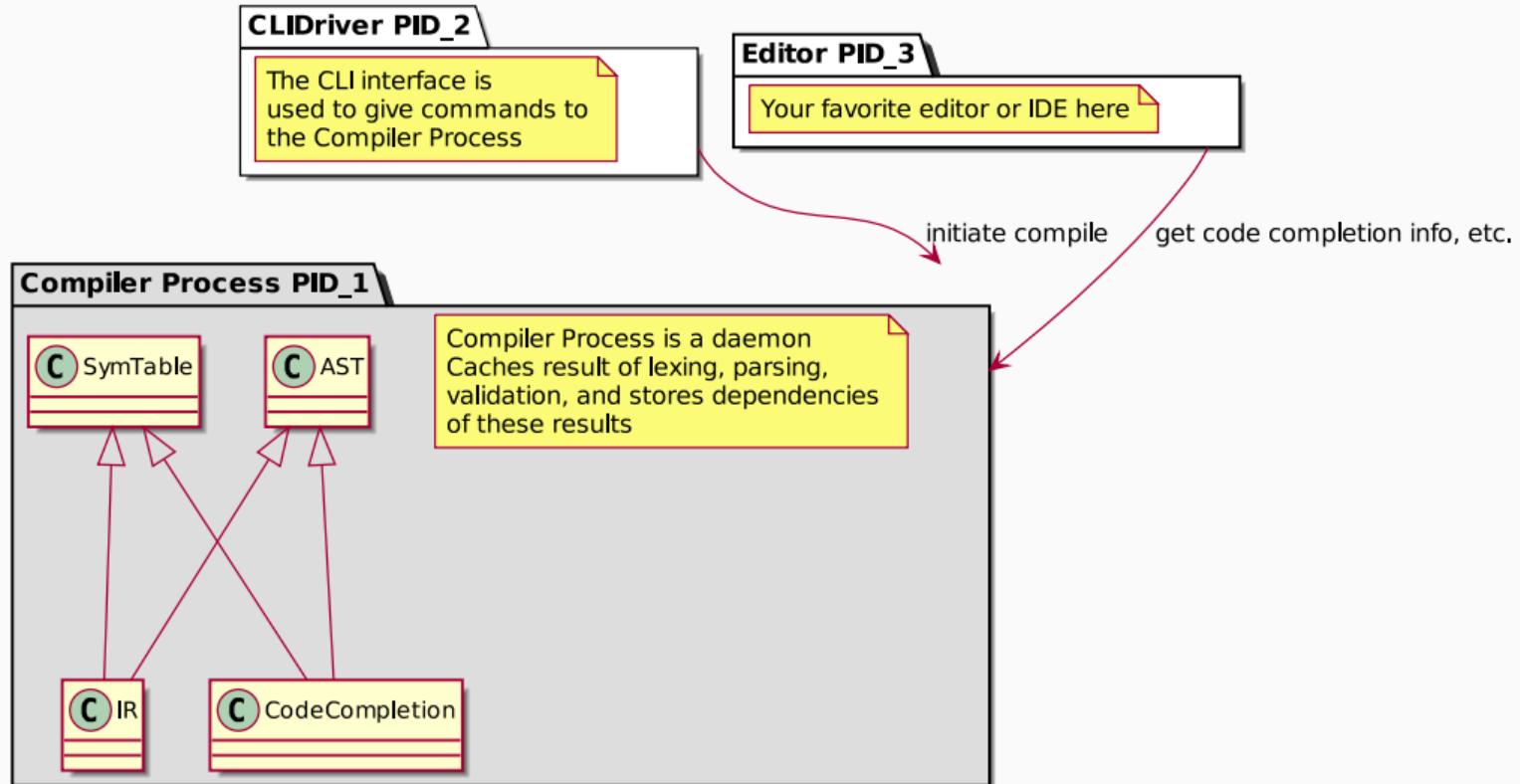
## Overall Legacy Architecture

- 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



- 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**

---