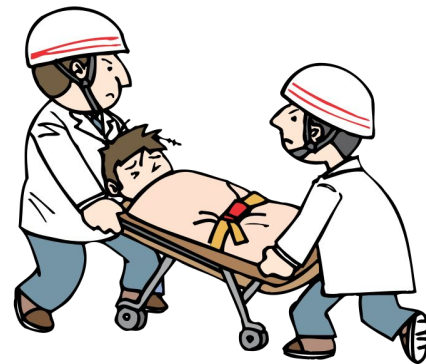
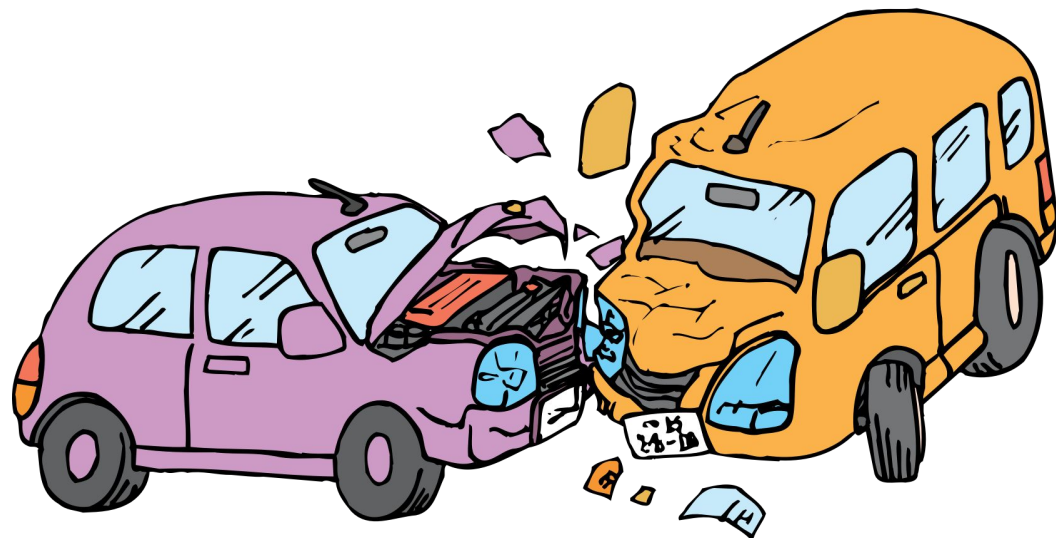


Big forms with JSON schemas and Transcript

November 15th, 2018
Philippe Entzmann



Reinsurance of car insurers



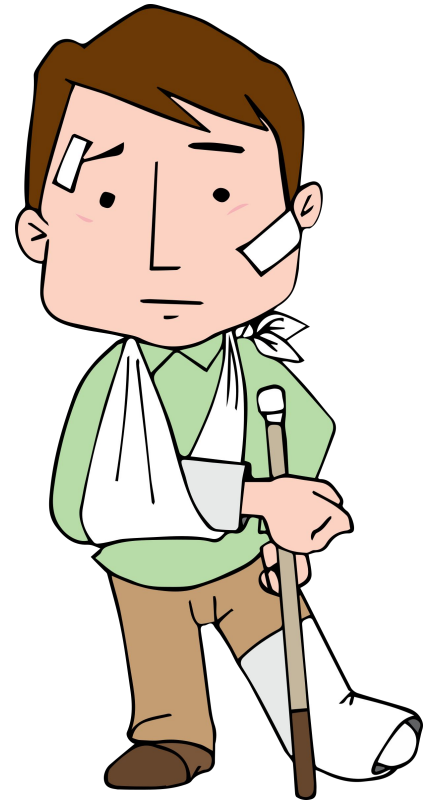
Victim's injuries follow-up

Yearly evaluation over lifespan.

Detailed expenses tracking of physical and non-physical injuries to the victim and its relatives.

Reference to mortality tables and currency rate.

A lot of differently structured data to collect.



From written forms to a database

Written big forms from different sources and different shapes consolidated in a single database.

The data schema will highly evolve over time. Our experts have to manage the data schema themselves :

- add fields, nested fields, list, set properties, ...
- split the whole schema in reusable parts
- define simple but usefull formulas

Each form may use 30 reusable sub-form parts leading to 300 base fields per form for a filled document of more than 1000 fields.

From schema to web form

We choose the excellent [json-editor](#) library :

“JSON Editor takes a [JSON Schema](#) and uses it to generate an HTML form.”



Write the business
rules in javascript

It worked

Acw IDB - Injuries Data Base Cedantes Dossier visite Restitution + Connecté en tant que ADMIN ADMIN

Table de capitalisation Taux d'intérêt (0.00) Fractionnement
0.76 trimestriel

Terme Service
échu viager

III. Préjudices patrimoniaux temporaires (avant consolidation)

III. A. Dépenses de santé actuelles (DSA)

Frais médicaux et pharmaceutiques jusqu'à la consolidation

III. A. 1. Hospitalisation

Hospitalisations	Nombre de jours	Coût journalier	Montant (Calculé)	Montant (Saisi)	Date de règlement	
Autre	1				jj/mm/aaaa	X ↓
IME	70				jj/mm/aaaa	X ↑ ↓
IME	2730				jj/mm/aaaa	X ↑

+ Hospitalisation X Last Hospitalisation X All

TOTAL Hospitalisations

III. A. 2. 1er appareillage achat initial

+ 1er appareillage achat initial

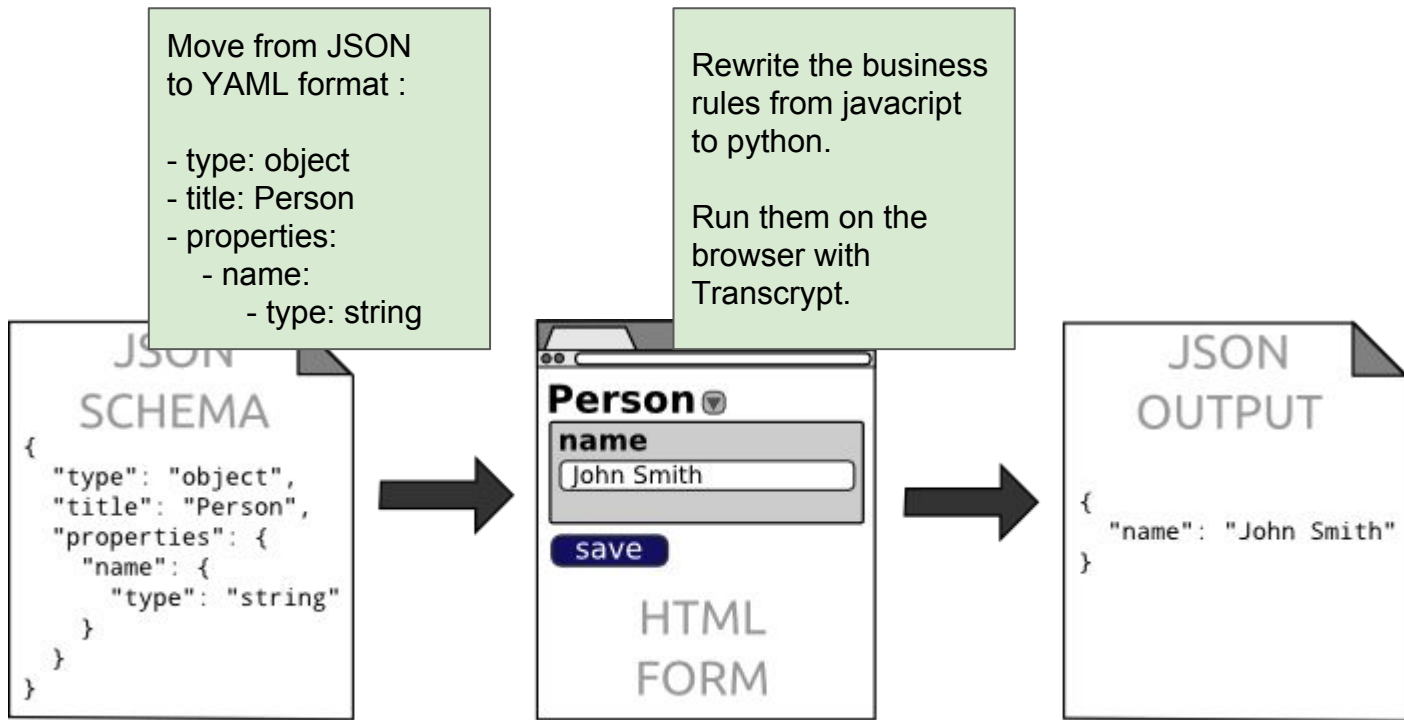
It worked but at a cost ...

```
1329 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
1330 // LISTE DES FONCTIONS DES TOTAUX - PREJUDICES PATRIMONIAUX TEMPORAIRES
1331 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
1332
1333 //Fonction de calcul du sous fomulaire Dépenses de Santé
1334 function calcul_sous_form_dsa_tot() {
1335     console.log('calcul_sous_form_dsa_tot: ');
1336     // Lecture des sous totaux
1337     var dsa_frai_medi = get_editor_float(editor, 'root.prej_patri_temp.depe_sant_actu.dsa_frai_medi')
1338     var dsa_tota_hosp = get_editor_float(editor, 'root.prej_patri_temp.depe_sant_actu.dsa_tota_hosp')
1339     var dsa_tota_prem_appa = get_editor_float(editor, 'root.prej_patri_temp.depe_sant_actu.dsa_tota_prem_appa')
1340
1341
1342     // Calcul du montant total
1343     var cumul = dsa_frai_medi + dsa_tota_hosp + dsa_tota_prem_appa
1344
1345     // Ecriture du résultat
1346     set_editor_value(editor, 'root.prej_patri_temp.depe_sant_actu.dsa_tota', cumul)
1347 }
1348
1349 //Fonction de calcul du sous fomulaire ATP ACTIVE CAPITAL
1350 function calcul_sous_form_atp_tota_tpa() {
1351     console.log('calcul sous form atp tota tpa: ');
```


Don't ask non-dev to mess up with javascript

```
15 // à déclencher à la fin du chargement de la page ...
16 $(document).ready(function() {
17     console.log('ready2 ...')
18
19
20 // à déclencher à la fin du chargement du formulaire ...
21 editor.on('ready', function() {
22
23     ////////////////////////////////////////
24     // Recopie des totaux dans
25     ////////////////////////////////////////
26
27     //root.prej_patri_temp.tota
28     //prej_patri_temp.tota_prej
29
30     //Préjudices patrimoniaux t
31     recopie_montant_json_schema
32
33     //Préjudices patrimoniaux t
34     recopie_montant_json_schema
35     //
36     //set all type_provi
37     call_maj_auto_type_provi()
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
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Hiding the javascript quirks with python



Better, stronger, faster, shorter

json-schema in JSON

```
1  {
2  |  "title": "Person",
3  |  "type": "object",
4  |  "properties": {
5  |  |  "A": {
6  |  |  |  "type": "integer"
7  |  |  |  },
8  |  |  "B": {
9  |  |  |  "type": "integer"
10 |  |  |  },
11 |  |  "C": {
12 |  |  |  "type": "integer",
13 |  |  |  "formula": "A+B"
14 |  |  |  }
15 |  |  }
16 |  }
```

json-schema in YAML

```
1  title: Person
2  type: object
3  properties:
4  |  A:
5  |  |  type: integer
6  |  B:
7  |  |  type: integer
8  |  C:
9  |  |  type: integer
10 |  |  formula: A+B
```

Better, stronger, faster, shorter

```
title: Tierce Personne
type: object
properties:
  nomb_heur:
    title: Nb d'heures
    type: number
    propertyOrder: 1
    field_array: ATPA_K_HORS_ARRE_FD.NBRHR
  nomb:
    title: Nombre
    type: number
    propertyOrder: 2
    field_array: ATPA_K_HORS_ARRE_FD.NBRE
  jour_sema:
    title: Jours/Semaine
    type: string
    propertyOrder: 3
    field_array: ATPA_K_HORS_ARRE_FD.UNIT
    enum:
      - Jour
      - Semaine
  cout_tier_pers:
    title: Cout
    type: number
    propertyOrder: 4
    field_array: ATPA_K_HORS_ARRE_FD.COUT
  mont_tota_calc:
    readonly: true
    title: Montant (calculé)
    type: number
    propertyOrder: 5
    field_array: ATPA_K_HORS_ARRE_FD.MT_CALC
    calc: ATPA_K_HORS_ARRE_FD.NBRHR * ATPA_K_HORS_ARRE_FD.COUT * ATPA_K_HORS_ARRE_FD.NBRE
  mont_tota_sais:
```

```
title: Calcul des indemnités
type: object
properties:
  atp_assi_temp_tier_pers_annu:
    title: Annuité
    type: number
    propertyOrder: 1
    field_id: R_ATP_DET_ANNUITE
    recopy: TOTAL_R_AVEC_ARRE_ATP
  atp_assi_temp_tier_pers_per:
    title: PER
    type: number
    propertyOrder: 2
    field_id: R_ATP_DET_PER
  mont_tota_calc:
    title: MT Calculé
    type: number
    propertyOrder: 3
    readonly: true
    field_id: R_ATP_DET_MT_CALC
    calc: R_ATP_DET_ANNUITE * R_ATP_DET_PER
  mont_tota_sais:
    title: MT Saisi
    type: number
    propertyOrder: 4
    field_id: R_ATP_DET_MT_SAIS
    recopy: R_ATP_DET_MT_CALC
  date_regl:
    format: date
    title: Date de règlement
    type: string
    propertyOrder: 5
    field_id: R_ATP_DET_DTE_REGL
```

Feedback of our Transcript experience

1. Easy Transcript setup
2. Accessing DOM and JS objects
3. Calling JS from python and python from JS
4. `eval()` missing
5. Python object overloading
6. Example formulas
7. Unit tests with `pytest`
8. End-to-end tests with `pytest/splinter/selenium`
9. Debugging with or without `sourcemap`
10. Watch files for transpilation
11. Transcript overhead
12. Transcript alternatives

Easy Transcript setup

```
$ pip install transcript
```

<< install (+java for clojure minification)

```
$ transcript hello
```

<< transpile hello.py to javascript

```
$ python3 -m http.server
```

<< serve static content

Accessing DOM and JS objects

```
<script type="module">
  import * as hello from './__target__/hello.js';
  window.hello = hello;
</script>

<h2>Hello pyparis</h2>

<div id = "greet">...</div>
<button onclick="hello.solarSystem.greet ()">
  Click me repeatedly!
</button>

<div id = "explain">...</div>
<button onclick="hello.solarSystem.explain ()">
  And click me repeatedly too!
</button>
```

```
from itertools import chain

class SolarSystem:
    planets = [list (chain (planet, (index + 1,))) for index, planet
                ('Mercury', 'hot', 2240),
                ('Venus', 'sulphurous', 6052),
                ('Earth', 'fertile', 6378),
                ('Mars', 'reddish', 3397),
                ('Jupiter', 'stormy', 71492),

                ('Saturn', 'ringed', 60268),
                ('Uranus', 'cold', 25559),
                ('Neptune', 'very cold', 24766)
                )]]

    lines = (
        '{} is a {} planet',
        'The radius of {} is {} km',
        '{} is planet nr. {} counting from the sun'
    )

    def __init__ (self):
        self.lineIndex = 0

    def greet (self):
        self.planet = self.planets [int (Math.random () * len (self.
        document.getElementById ('greet') .innerHTML = 'Hello {}'.fo
        self.explain ()

    def explain (self):
        document.getElementById ('explain') .innerHTML = (
            self.lines [self.lineIndex] .format (self.planet [0], se
        )
        self.lineIndex = (self.lineIndex + 1) % 3

solarSystem = SolarSystem ()
```

Calling JS from python and python from JS

```
def value(self, path):  
    # Get a reference to a node within the editor  
    node = self.jsoneditor.getEditor(path)
```

```
});  
jsoneditor.on('ready',function() {  
    // Now the api methods will be available  
    formula.editor.bind(jsoneditor);  
});  
};
```


eval() missing

Evaluating our formulas is easy with eval()

The single disappointment in our experiment :

eval() is not implemented in Transcript

You must use the transpiler server-side only.

So we had to parse and evaluate our formulas in python.

Python object overloading

Transcrypt is very close to Python regarding subclassing, overloading, compositing objects.

We implemented a simple formula parser and a schema/document walker.

All the python tricks we needed worked :

`__get__`, `__missing__`, `__setitem__`, `__iter__`, `__contains__`, ...

Example formulas

Simple formula : $AMOUNT * QTY$
referring to nearby fields

Dot notation formula : $sum(HOSP.NB * HOSP.AMOUNT)$
referring to array and doing matrix operation

Custom function : $my_special_pricer(x, y, z)$
defined in Python (Transcrypt)

```
26 field_array: ATPA_K_HORS_ARRE_FD.COUT
27 mont_tota_calc:
28   readonly: true
29   title: Montant (calculé)
30   type: number
31   propertyOrder: 5
32   field_array: ATPA_K_HORS_ARRE_FD.MT_CALC
33   calc: ATPA_K_HORS_ARRE_FD.NBRHR * ATPA_K_HORS_ARRE_FD.COUT * ATPA_K_HORS_ARRE_FD.NBRE
34 mont_tota_sais:
35   title: Montant (saisi)
```

Relative json pointer support

We plan to support the draft proposal [relative-json-pointer](#) that will help for some complex cases of relative references.

Example :

```
{
  "foo": ["bar", "baz"],
  "highly": {
    "nested": {
      "objects": true
    }
  }
}
```

Starting from the value `{"objects":true}` (corresponding to the member key "nested"), the following JSON strings evaluate to the accompanying values:

<code>"0/objects"</code>	<code>true</code>
<code>"1/nested/objects"</code>	<code>true</code>
<code>"2/foo/0"</code>	<code>"bar"</code>
<code>"0#"</code>	<code>"nested"</code>
<code>"1#"</code>	<code>"highly"</code>

Unit tests with pytest

Formulas in schema are tested.

Dozens of tests, easily readable and writable by the business experts.

Run on CPython.

/schema/simple.yaml

```
1 title: Person
2 type: object
3 properties:
4   A:
5     type: integer
6   B:
7     type: integer
8   C:
9     type: integer
10    formula: A+B
```

```
def test_simple():
    form = Form(schema='/schema/simple.json',
                data={'A':3, 'B':2})

    assert form.C == 3 + 2

    form.A = 4
    assert form.C == 4 + 2

    form.B = 6
    assert form.C == 4 + 6
```

End-to-end tests with selenium

Same tests !

Automatically transpiled to Javascript
and run on a real browser with selenium, splinter
and pytest.

Chrome headless mode for running on CI jobs.

/schema/simple.yaml

```
1 title: Person
2 type: object
3 properties:
4   A:
5     type: integer
6   B:
7     type: integer
8   C:
9     type: integer
10    formula: A+B
```

```
def test_simple():
    form = Form(schema='/schema/simple.json',
                data={'A':3, 'B':2})

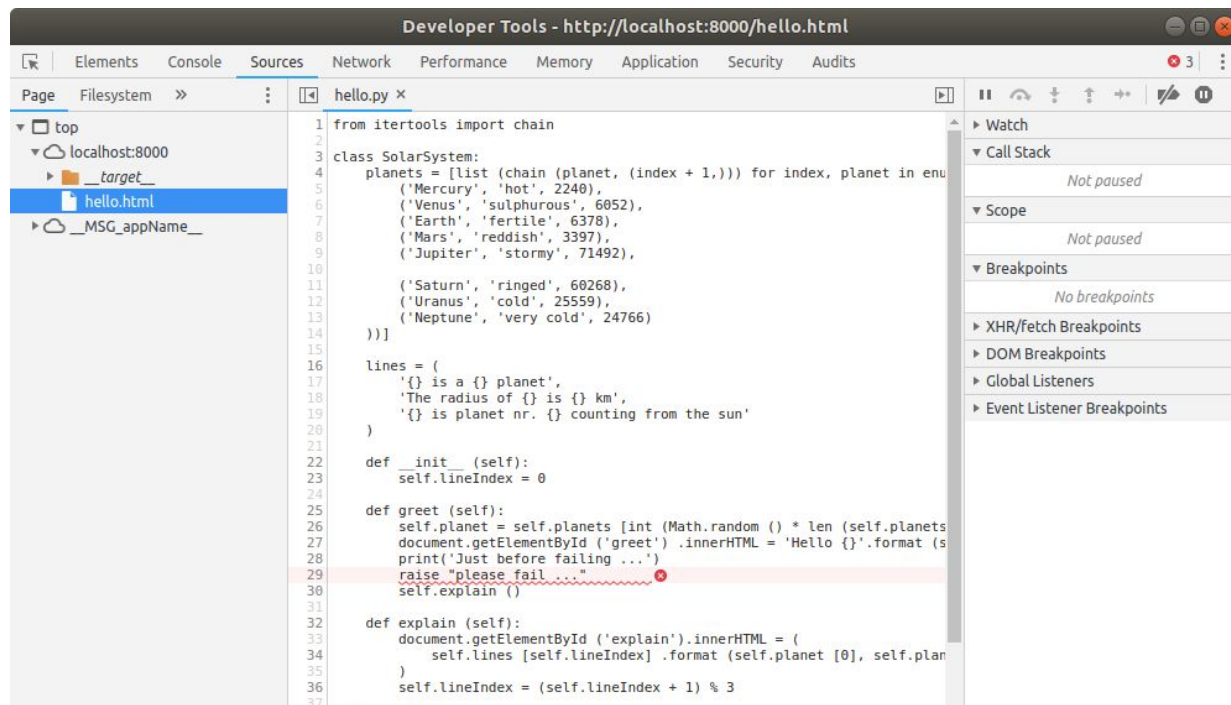
    assert form.C == 3 + 2

    form.A = 4
    assert form.C == 4 + 2

    form.B = 6
    assert form.C == 4 + 6
```

Sourcemaps debugging

transcript -m hello



Developer Tools - http://localhost:8000/hello.html

Page Filesystem » hello.py x

```
1 from itertools import chain
2
3 class SolarSystem:
4     planets = [list(chain(planet, (index + 1,))) for index, planet in enumerate(
5         ('Mercury', 'hot', 2240),
6         ('Venus', 'sulphurous', 6052),
7         ('Earth', 'fertile', 6378),
8         ('Mars', 'reddish', 3397),
9         ('Jupiter', 'stormy', 71492),
10
11         ('Saturn', 'ringed', 60268),
12         ('Uranus', 'cold', 25559),
13         ('Neptune', 'very cold', 24766)
14     )]]
15
16     lines = (
17         '{} is a {} planet',
18         'The radius of {} is {} km',
19         '{} is planet nr. {} counting from the sun'
20     )
21
22     def __init__(self):
23         self.lineIndex = 0
24
25     def greet(self):
26         self.planet = self.planets[int(Math.random() * len(self.planets)
27             )]
28         document.getElementById('greet').innerHTML = 'Hello {}'.format(self.planet)
29         print('Just before failing ...')
30         raise "please fail ..."
31         self.explain()
32
33     def explain(self):
34         document.getElementById('explain').innerHTML = (
35             self.lines[self.lineIndex].format(self.planet[0], self.planet[1])
36         )
37         self.lineIndex = (self.lineIndex + 1) % 3
```

Watch
Call Stack
Not paused
Scope
Not paused
Breakpoints
No breakpoints
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints

Debugging without sourcemap

transcript -a -n hello

Annotated target code for hello.py

```
// ===== Source: D:/activ_tosh/geatec/transcript/transcript/demos/hello/hello.py =====  
  
/* 000001 */ (function () {  
/* 000001 */     var chain = __init__ (__world__.itertools).chain;  
/* 000003 */     var SolarSystem = __class__ ('SolarSystem', [object], {  
/* 000021 */         get __init__ () {return __get__ (this, function (self) {  
/* 000022 */             self.lineIndex = 0;  
/* 000022 */         });},  
/* 000024 */         get greet () {return __get__ (this, function (self) {  
/* 000025 */             self.planet = self.planets [int (Math.random () * len (self.planets))]  
/* 000026 */             document.getElementById ('greet').innerHTML = 'Hello {}'.format (self.  
/* 000027 */             self.explain ());  
/* 000027 */         });},  
/* 000029 */         get explain () {return __get__ (this, function (self) {  
/* 000031 */             document.getElementById ('explain').innerHTML = self.lines [self.lineI  
/* 000033 */             self.lineIndex = (self.lineIndex + 1) % 3;  
/* 000033 */         });}  
/* 000033 */     });  
/* 000004 */     SolarSystem.planets = function () {  
/* 000004 */         var __accu0__ = [];  
/* 000004 */         var __iter0__ = enumerate (tuple ([tuple (['Mercury', 'hot', 2240]), tuple  
/* 000004 */         for (var __index0__ = 0; __index0__ < __iter0__.length; __index0__++) {  
/* 000012 */             var __left0__ = __iter0__ [__index0__];  
/* 000012 */             var index = __left0__ [0];  
/* 000012 */             var planet = __left0__ [1];
```


Watch file for transpilation

Static transpilation not an option in our case since users can change python source (schema formula)

Watch and transpile : run transcript again on any file change
entr or inotify tools

Transcrypt overhead

The minified JavaScript code for each of your own modules is roughly just as large as the Python source code. On top of that there's a one time overhead of 20kB for Transcrypt's core and built-ins. Should you use the JavaScript 5 to 6 translator, that adds an extra 10kB. For larger projects, the overhead becomes negligible. A project with a Python source of say 600kB tends to result in a download of about equal size. Moreover Python sourcecode for a certain application tends to be smaller than handwritten JavaScript source code for the same problem, due to language constructs like list comprehensions, but also due to facilities like class based OO and multiple inheritance. As far as speed is concerned, in most cases it is roughly equal to the speed of hand-written JavaScript. [..]

Transcrypt alternatives

- [transcrypt](#) : transpiler, partial python support, numpy port
- [rapidscript](#) : transpiler, support eval() !
- [brython](#) : full python interpreter
- [pyodide](#) : WASM based
- [batavia](#) : python VM, run python bytecode, not source !

- [pyjs](#) : full python interpreter ?
- [pypyjs](#) : full python interpreter, emscripten/ASM based
- [jiphy](#) : transpiler, too limited

Being able to push Python in the browser helps us to add features to our automatically generated big forms.

Transcrypt saves us from the two languages pitfall in a critical part of our project. The overhead induced is negligible in our case.

We are closing gaps between the front-end and back-end development by sharing the same languages and test framework.

Python everywhere, really ?

So we get python on the browser and we're happy with it.

Lonely trick or real trend ?

We think it's a bold move for a good reason :

All the technologies are moving faster ...

... but our brain is not !!



You can't master many programming languages.

Non-developer can only learn a single trivial programming language.

Developers and non-dev. must have a common programming language.

The toolset shared among the team must be as light as possible.

The Python language and eco-system is the best fit today.



Split the stack



Injury Data Bank

Visible to anyone :

jupyter > YAML > pytest > xlwings
> python > json-schema > plotly > pandas

Must stay hidden except for devops :

transcript, cpython, anaconda, mongo, docker,
kubernetes, flask, swagger, angular, bootstrap, caddy,
docker-compose, pyinstaller, git, dash, python-pptx,
dramatiq, secretary, gitlab, javascript, ...

Thank you !

philippe@phec.net

Image credits :

- Car top view by [qubodup](#)
- Motor vehicle accident illustration by [oksmith](#)
- Injured illustration by [oksmith](#)
- Head with brain silhouette illustration by [monstara](#), [GDJ](#)