Big forms with JSON schemas and Transcrypt

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 useful 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](https://jsoneditor.org/) library:

“JSON Editor takes a [JSON Schema](https://json-schema.org/) and uses it to generate an HTML form.”

Write the business rules in javascript.
It worked
It worked but at a cost ...
3000 LOC of clumsy javascript business rules

```javascript
var path_table = [
  "root.prej_patri_perm.depe_sant_futu.dsfs_appa_aide_tech.dsfs_appa_aide_tech_crit_eval.fd_refe_tabl",
  "root.prej_patri_perm.pert_gain_prof_futu.pgpf_crit_eval.fd_refe_tabl",
  "root.prej_patri_perm.asss_tier_pers.fd_refe_tabl_rent",
  "root.prej_patri_perm.depe_sant_futu.dsfs_plac_viaq.dsfs_plac_viaq_crit_eval.fd_refe_tabl"
];

var path_taux = [
  "root.prej_patri_perm.depe_sant_futu.dsfs_appa_aide_tech.dsfs_appa_aide_tech_crit_eval.taux_inte",
  "root.prej_patri_perm.pert_gain_prof_futu.pgpf_crit_eval.taux_inte",
  "root.prej_patri_perm.asss_tier_pers.taux_inte",
  "root.prej_patri_perm.depe_sant_futu.dsfs_plac_viaq.dsfs_plac_viaq_crit_eval.taux_inte"
];

var path_age = [
  "root.prej_patri_perm.depe_sant_futu.dsfs_appa_aide_tech.dsfs_appa_aide_tech_crit_eval.age_ouve_droi",
  "root.prej_patri_perm.pert_gain_prof_futu.pgpf_crit_eval.age_ouve_droi",
  "root.prej_patri_perm.asss_tier_pers.age_ouve_droi",
  "root.prej_patri_perm.depe_sant_futu.dsfs_plac_viaq.dsfs_plac_viaq_crit_eval.age_ouve_droi"
];

path_taux.forEach(function(path) {
  for (var i = 0; i < path.provi.length; i++) {
    maj_all_type_provi(path.provii[i], path_table[i], path_taux[i], path_age[i]);
  }
});
```
Don’t ask non-dev to mess up with javascript
Hiding the javascript quirks with python

Move from JSON to YAML format:
- type: object
- title: Person
- properties:
  - name:
    - type: string

Rewrite the business rules from javascript to python.
Run them on the browser with Transcrypt.
Better, stronger, faster, shorter

json-schema in JSON

```json
{
  "title": "Person",
  "type": "object",
  "properties": {
    "A": {
      "type": "integer"
    },
    "B": {
      "type": "integer"
    },
    "C": {
      "type": "integer",
      "formula": "A+B"
    }
  }
}
```

json-schema in YAML

```yaml
title: Person
type: object
properties:
  A:
    type: integer
  B:
    type: integer
  C:
    type: integer
    formula: A+B
```
Better, stronger, faster, shorter
Feedback of our Transcrypt experience

1. Easy Transcrypt 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. Transcrypt overhead
12. Transcrypt alternatives
Easy Transcrypt setup

```
$ pip install transcrypt  << install (+java for clojure minification)
$ transcrypt hello  << transpile hello.py to javascript
$ python3 -m http.server  << serve static content
```
Accessing DOM and JS objects

```html
<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>
```

```python
class SolarSystem:
    planets = [list (chain (planet, (index + 1,))) for index, planet in enumerate(["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)])]

    def __init__ (self):
        self.lineIndex = 0

    def greet (self):
        self.plant = self.planets [int (Math.random () * len (self.plants))]
        self.lineIndex = (self.lineIndex + 1) % 3
        document.getElementById ("greet").innerHTML = "{} is a {} planet, the radius of {} is {} km, {} is planet nr. {} counting from the sun".format (self.plant [0], self.plant [1], self.plant [2], self.plant [3], self.plant [4], self.lineIndex)
        self.lineIndex = (self.lineIndex + 1) % 3

    def explain (self):
        document.getElementById ("explain").innerHTML = (self.lines [self.lineIndex] .format (self.plant [0], self.plant [1], self.plant [2], self.lineIndex))
        self.lineIndex = (self.lineIndex + 1) % 3

solarSystem = SolarSystem ()
```
Calling JS from python and python from JS

```python
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 Transcrypt**

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
refering to nearby fields

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

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

```
"0/objects": true
"1/nested/objects": true
"2/foo/0": "bar"
"0#: "nested"
"1#: "highly"
```
Unit tests with pytest

Formulas in schema are tested.

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

Run on CPython.
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.
Sourcemap debugging

transcrypt -m hello
Debugging without sourcemap

transcrypt -a -n hello
Watch file for transpilation

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

Watch and transpile : run transcrypt 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-developper can only learn a single trivial programming language. Developpers 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

Visible to anyone:

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

Must stay hidden except for devops:

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

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](#)