DjangoCong
2016

Systematic: la toolchain JS simple

(yeah, de retour avec une lightning trop longue)

(mais j'ai la flemme de faire des prez sérieuses)

(il y aura toujours trop de texte par rapport à ce que je raconte)

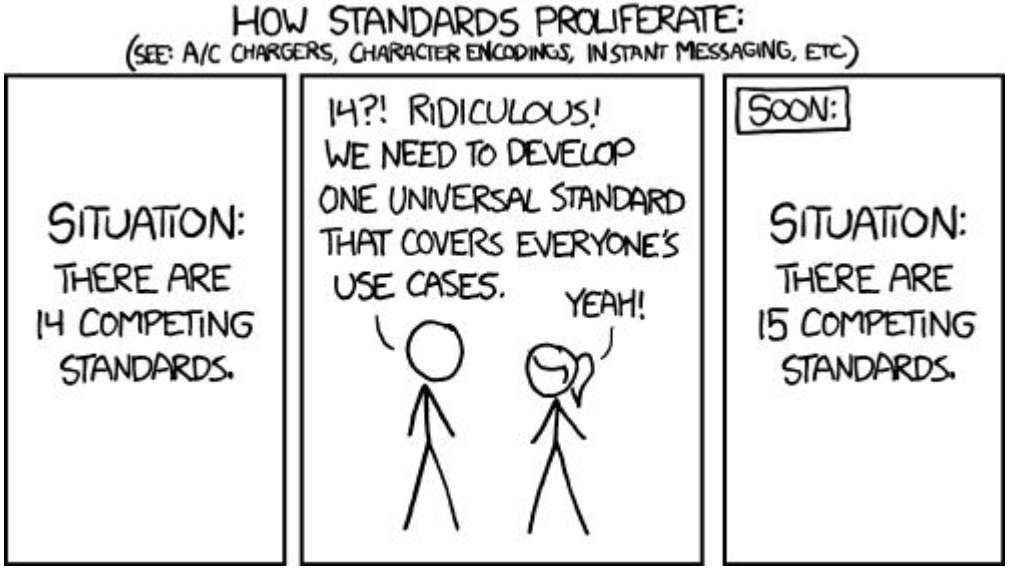
Polyconseil

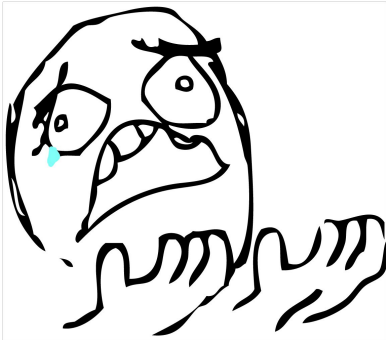
Intro rapide

- a passé l'année dernière chez **Polyconseil**
- ai écrit pas mal (trop peut-être déjà) de JS
- a testé joyeusement beaucoup parmi les toolchains JS existantes (grunt, brunch, broccoli, gulp)

Après tâtonnements et itérations, nous en sommes venus à écrire notre propre toolchain: **systematic**.

Je sais ce que vous pensez





Pourquoi Systematic ?

Les toolchains JS: FACTS

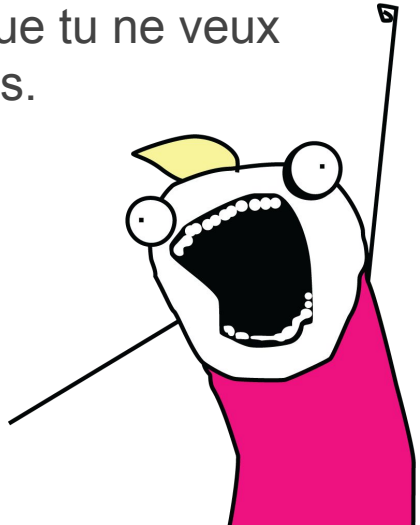
- Des outils écrits en JS pour essayer de refaire MAKE (mal).
- Des plugins codés sans tests, avec 8 grammes, en 2013.
- Une composabilité à tout va sans standard précis.
- *Ce ne sont pas des toolchains, mais des méta-toolchains*. Des générateurs de toolchains qui autorisent tout et n'importe quoi.

Généralement, on prend le dernier *grunt-bootstrap4-angular-jade-starter* sur Github, avec son Gruntfile buggé et illisible de 4000 lignes, qui en fait beaucoup trop et pas assez à la fois.

Ring a bell ?

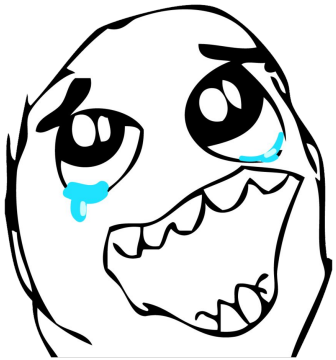
Ta toolchain idéale

- Tu veux que ce soit SIMPLE et savoir ce qui se passe.
- Tu veux utiliser `make test` et `make dist`, avec de la vraie CLI derrière.
- Tu veux partir du principe que tous les choix techniques sont déjà faits et encouragent les bonnes pratiques.
- Tu veux qu'il y ait une seule bonne façon de faire, parce que tu ne veux pas suivre 12 coding styles et 5 schémas d'applis différents.
- Tu veux que ce soit raisonnablement à la page (ES6, CSS Modules, Standard Style, …)



En fait, tu veux **Systematic**.

Sous le capot



- Un système basé sur **Make**.
- Un fichier de configuration **minimaliste**.
- Une sélection minimale de plugins **Webpack**.
- Une configuration **ready-to-use** dans la plupart des cas.
- Des **choix simples et clairs** de syntaxe, tests, cycle de vie applicatif, procédure de release.
- Open source, évidemment. https://github.com/Polyconseil/systematic

Ce qu'on y gagne

- Finies les discussions sur le codestyle.
- Finies les discussions sur l'organisation du code et sa compilation.
- Encourage l'indépendance de la toolchain vis-à-vis des "frameworks".
- Le local CSS (:local) et le namespacing JS via Webpack permettent de créer des modules intégrables directement dans nos pages Django, sans conflits de style ou de bibliothèques JS globales (en faisant attention)

```
<script src="{% static 'common_ui/js/base.min.js' %}"></script>
<script src="{% static 'external_js/componentA/dist/bundle.js' %}"></script>
<script src="{% static 'external_js/componentB/dist/bundle.js' %}"></script>
```

Bilan: un an d'efforts front-end

- On a cessé d'avoir 5 stacks différentes à moitié imbriquées avec Django, avec npm et gulp et grunt et un peu bower et github comme CDN et...
- 6 applications et bibliothèques entièrement Javascript ont été codées.
- Chacune suit un cycle de vie strict, unifié et documenté.
- Chaque nouvelle application est plus rapide et facile à coder que la précédente.
- Leur déploiement indépendant ou directement dans les pages existantes est maintenant trivial (ai-je parlé de django-npm ? ;)

Bref: on a gagné.