Développement SMW (web & mobiles)

Tux the Penguin by Carlos Pardo

Le JavaScript c'est comme la coke : c'est mieux pur. Ben

Javascript est le seul langage pour lequel un institut de recherche a été créé, afin d'essayer de comprendre comment il marche : « The JSCert project aims to really understand JavaScript. » http://jscert.org. Sad

Le navigateur Web est devenu le système et le système son driver.
Le nouveau langage système est donc Javascript. NumberSix

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Indice	Validation	Objet	
1	03/05/16	Édition initiale	sr
9	06/05/16	Première refonte	sr
13	08/05/16	Seconde refonte	sr
16	10/05/16	Version 1	sr
20	12/05/16	Début d'installation sur OSX	sr
27	15/05/16	Typos et suppression césure, ajout sous-chapitre « Bonnes pratiques » - Rémi (bar@ovh)	rj
29	17/05/16	Typos, ajout « La compression », sommaire avec hyperliens - Rémi (bar@ovh)	rj
30			

- Positionner le curseur sur l'avant dernière ligne du tableau (celle au dessus de « Édition courante ») ;
- Créer une nouvelle ligne dans le tableau ;
- Sélectionner et copier la dernière ligne, de « Validation » à « Email » (tout sauf la première colonne) ;
- Positionner le curseur sur l'avant dernière ligne, dans la colonne « Validation » ;
- Coller ;

Reporter l'indice de la dernière ligne dans la nouvelle ligne.

Impression	12/05/16 - 09:36
Édition	- 30:13:12

« Be seeing you » Number six, The prisoner - « I have been studying how I may compare this prison where I live unto the world » - Richard II, Act V

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Table des matières

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Ressources retenues

Environnement

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Généralités

1 Problématique

Une entreprise de services « sans bureaux » va être créée. Elle sera animée par des personnes :

- Statiques et sans contraintes géographiques (direction, administration, recherche & développement) : *leur bureau est leur domicile* ;
- Mobiles et avec contraintes géographiques (commerciaux, typiquement un commercial pour deux départements) : *leur bureau est leur voiture*.

La communication :

- Formelle se fait par applications Web et Mobiles ;
- Informelle se fait par visioconférence.

En fonction des métiers, des réunions physiques périodiques sont prévues.

L'entreprise comprend un service informatique « producteur de solutions » au service des « producteurs de revenus », dans ce cas les services marketing et commerciaux.

Cette mission comprend, entre autres, la mise en place de solutions informatiques adaptées :

- Certaines prises sur étagères ;
- D'autres développées spécifiquement, au fur et à mesure des besoins et des possibilités.

Dans ce second cas, il faudra développer *à façon*, avec la *meilleure productivité possible*, des applications pour le Web et les Mobiles.

2 Choix initiaux

NumberSix law
Le mille feuilles de tous les sous-langages, pseudos concepts et outils pour *gibbons* qui sont imposés aux développeurs Web nécessite la maîtrise d'une somme phénoménale de connaissances *stupides* pour un résultat extraordinairement *pitoyable*.

Tout changement de paradigme entraîne une période difficile pour les développeurs. Par exemple :

- Au début des années 1992, quand les applications sont devenues graphiques ;
- Au début des années 2000, quand le Web est devenu dynamique.

Dans ce dernier cas, les concepts étant déficients et les navigateurs non standardisés, certaines technologies de compromis et de compensation sont apparues, comme Flash.

Il a fallu attendre les années 2010 pour que cette situation évolue, avec HTML5, CSS3, et une version de Javascript (ES5) standardisés, quel que soit le navigateur.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Nous sommes désormais dans une période de créativité débridée, avec la possibilité d'utiliser de nouveaux concepts, à travers un paradigme acceptable et un langage unique, Javascript, certes mal né, mais unique et évoluant dans le bon sens.

Si l'on se rappelle que NeWS[1] (Network extensible Window System), au milieu des années 80, utilisait également un langage unique (un sur-ensemble de Postscript également dénommé NeWS) :

- A la place de Javascript pour la programmation ;
- A la place de DHTML et de CSS pour le rendu ;
- A la place de XML and JSON pour la représentation des données.

On se demande juste si 30 (trente) années n'auraient pas été perdues...

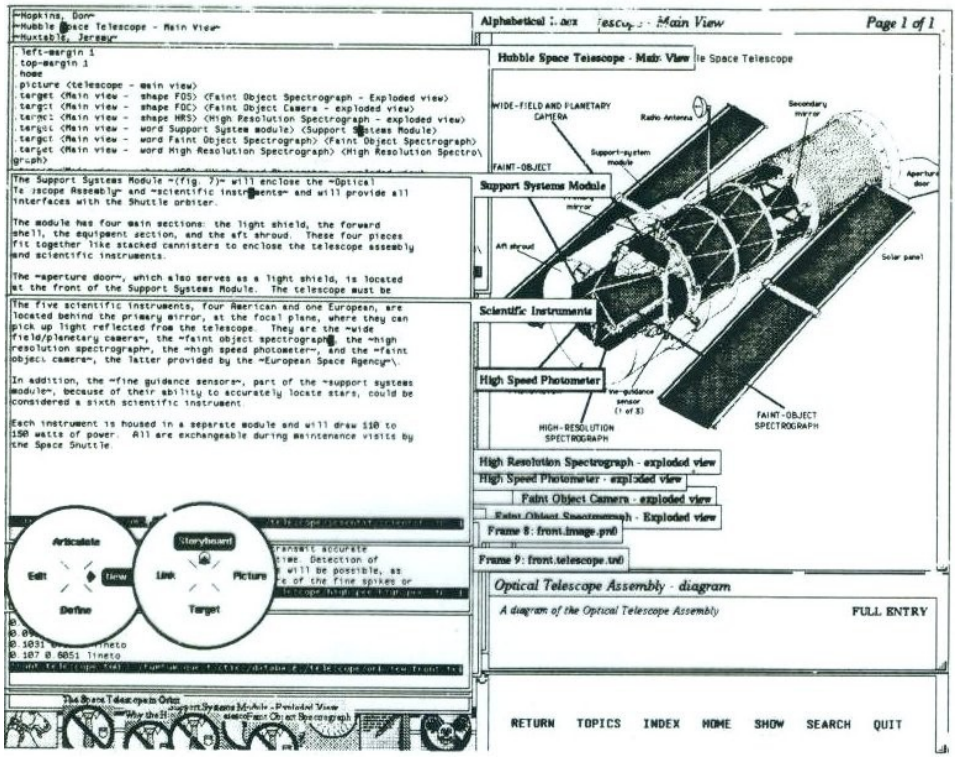

Illustration 1 : NeWS : Emacs authoring tool & navigateur Hyperties (milieu des années 80)

Afin d'optimiser les recherches de collaborateurs (compétences standard, technologies connues, etc...) et de simplifier le développement, il est envisagé d'utiliser :

- Un langage unique ;
- Un environnement unique.

[1] NeWS : https://en.wikipedia.org/wiki/NeWS

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

3 Recherches initiales

3.1 Conseils - Daniel Caillibaud (bar@ovh)

La stack de dev de node.js est un peu pénible à maintenir, parce que npm (gestionnaire de paquets) reste assez stupide et que la chaîne de dépendances peut être énorme avec des trucs qui bougent beaucoup, surtout si l'on part sur des technos fraîches en pleine évolution.

Si ça dérange (il y a de quoi), on est pas obligé de se convertir à la « single page application », le web à l'ancienne ça marche toujours, même avec node.Js qui génère du html statique ;-)

Exemple de ToolsChain pour du front :

- React/Redux ;
- Express.js ;
- Babel pour coder en es6-7 parce que c'est nettement plus clair ;
- WebPack pour faire des bundles complets et optimisés ;
- Mocha + Chai + Enzyme pour les tests.

Ça marche bien mais à condition de figer toutes les versions et ne plus y toucher (attention au réglage npm par défaut en « semver » : il faut remplacer ^ par =). De temps en temps prendre une journée pour mettre à jour tout le monde, voir que ça marche plus, recommencer en mettant à jour partiellement, re-tester, et trouver la bonne combinaison (des fois ça marche du premier coup, mais pas toujours).

Lectures :

- Axel Rauschmayer : http://www.2ality.com ou http://exploringjs.com/es6 ;
- Dan Abramov : http://redux.js.org déjà cité, mais plein d'autres contrib ;
- http://www.js-attitude.fr/2014/11/05/veille-techno-utile ;
- https://gist.github.com/tdd/0841b940f9adc59740a9
- http://youdontknowjs.com ;
- https://ericelliottjs.com/product/programming-javascript-applications-paper-ebook-bundle .

3.2 Recherches personnelles

Daniel a raison : l'écosystème node.js est vraiment bien *et* c'est vraiment un foutoir.

Daniel a également conseillé les packages et les outils communément admis comme étant les plus recommandés. On verra que les choix retenus diffèrent légèrement mais sa vision et son raisonnement général sont bons.

Monter un environnement de développement node.js qui tient la route n'est effectivement pas une opération facile, et le maintenir non plus. Ca bouge dans tous les sens, ça forke plus vite que son ombre, la technologie est en train de partir dans tous les sens et (imho) c'est bien : c'est juste toute la splendeur du Darwinisme en action. La technologie est jeune et tout le monde fonce dessus.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Rien n'est stable, même pas le langage, ES5, ES6, Babel ou Scala ?, Babel et Typescript ? Imaginons alors des dépendances en ES5, certaines Babelisées, d'autres Typescriptées et un dev principal en Scala. C'est cool ? Oui ? Non ? Ca semble dément ? Tout est normal. Et en plus, demain, ça sera mieux (et passablement différent) !

Bienvenue chez les fous qui font des choses incroyables avec l'un des langages le plus incompris et le plus (initialement) mal implémenté jamais créé et devenu, à l'insu de son plein gré, le plus répandu au monde.

> NumberSix law
> **Devenu incontournable, mais incompris car hybride, les améliorations pullulèrent, factorisant ainsi toutes les possibilités de l'employer, pour le « bonheur de la plus grande confusion ».**

L'incohérence provient probablement du hiatus entre la syntaxe (style C) et l'essence de Javascript (langage hybride, entre la programmation impérative et la programmation fonctionnelle).

Il faut enfin noter qu'avec TypeScript, qui intègre Javascript ES6, le paradigme est sur la bonne voie.

Critiques générales

C'est le « Hater Corner » du document. Deux liens pris au hasard. On doit pouvoir trouver mieux.

http://linuxfr.org/news/et-si-javascript-allait-droit-dans-le-mur
https://medium.com/@wob/the-sad-state-of-web-development-1603a861d29f

4 Recherches approfondies

L'écosystème Javascript est en ébullition, les concepts pullulent et avec eux les nouveaux termes : « Packageur », « Single Page Application », framework et application « Isomorphique », sans oublier mes termes préférés : « Transpiler » et « Front Riche » (Plize donte forguette to praonaounce Transpaïlleur et Fronte Rrriche, because rien qu'à les dire comme ça, on se sent tellement mieux).

J'ai cherché à comprendre, en partant de zéro et, au bout d'une semaine de recherches, le « grand schéma » est apparu. J'ai pu alors laisser tomber une bonne partie des trucs « à la mode », divisant par deux mes besoins de formation et, probablement par autant, mes emmerdes futures.

4.1 Démarche

Précisons que cette méfiance envers « la mode » n'implique pas le refus de l'innovation (bien au contraire), ni la réticence à employer de nouveaux outils.

Le but de la démarche est de créer un environnement de développement « raisonnablement standard », avec des ressources « pratiques et pragmatiques », ayant fait leurs preuves, et avec une bonne communauté et donc une pérennité raisonnable.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

4.2 Trois points clés

Si vous respectez ces trois points, vous êtes certainement partis pour une bonne expérience.

❑ Do It Yourself

Utiliser Node.js et Javascript implique de se prendre en main et de concevoir son environnement de développement selon ses choix. Rien n'est standard, tout est ouvert : on peut se retrouver avec des tonnes de dépendances mouvantes ou au contraire contenir ses dépendances et ses outils à l'essentiel.

❑ Javascript est un Lisp habillé en C

Après avoir lu cette phrase, j'ai compris la « logique » de Javascript, langage en partie fonctionnel.

Référence : http://www.crockford.com/javascript/javascript.html
Programmation fonctionnelle en JS : http://fr.eloquentjavascript.net/chapter6.html
http://www.24joursdeweb.fr/2014/un-peu-de-programmation-fonctionnelle-en-javascript

❑ Tri sélectif et Darwinisme

Il faut oublier les premières implémentations et versions de Javascript, regarder du coté de Type-Script et d'ES6, et penser à Javascript comme un langage fonctionnel et hybride.

Il faut résister à « la mode » et laisser le temps à Darwin de faire le tri. Choisir soigneusement les quelques bijoux incontournables et laisser de coté les concepts inutiles et les composants bouffis.

4.3 Single Page Application & Framework Isomorphique

NumberSix law
Toute simplification apparente implique une complexité cachée.
Immanquablement, la complexité cachée se vengera.

❑ Introduction

Le besoin d'application « Isomorphique » découle du concept de « Single Page Application », qui découle du concept de « Front riche », afin d'obtenir une réactivité proche d'une application « Client lourd » à partir d'un navigateur Web.

Pour faire simple, utiliser au mieux le potentiel d'HTML5, CSS3, ES5 et DOM3 pour proposer une expérience utilisateur aussi agréable qu'avec une application classique.

Cette expérience utilisateur existe, le web actuel en offre plein d'exemples et c'est simplement merveilleux de pouvoir enfin réaliser des tâches complexes avec un simple navigateur.

La discussion concerne les concepts de « Single Page Application » et de framework « Isomorphique » qui sont une des voies pour y parvenir, et certainement la voie « à la mode ».

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

❑ Single Page Application

Quand on charge une SPA, on charge une page HTML, du CSS, des images, des scripts standards, au moins un script de framework, et des scripts spécifiques représentant le code de l'application (et c'est long à charger, même avec du lazy loading) et puis on se met à exécuter « tout ça » (et ce « tout ça » n'est pas simple).

Évidemment, c'est lourd et ça prends du temps, quelques secondes a minima. C'est acceptable pour du « Back Office », nettement moins pour du « Front Office ». Et, comme nous ne sommes pas dans un jour charitable, on va alors évoquer les problèmes :

- Des terminaux mobiles qui disposent d'une connectivité limitée ;
- Du référencement (même si Google indexe les SPA), ce qui nécessite de déployer des « rustines », comme pretender.io, selon un principe qui rappelle le bon vieux temps de Flash.

Donc le concept de SPA engendre des problèmes de performance et de référencement. Pour les résoudre, Facebook, à travers React, a proposé une nouvelle façon de déployer une SPA.

❑ Framework Isomorphique

Au lieu d'un framework MVC fondé sur le DOM, React propose un framework V fondé sur un Virtual DOM. Cette architecture permet d'utiliser les composants aussi bien coté client que coté serveur (caractérisant ainsi l'approche Isomorphique).

Quand on charge une SPA isomorphique, la première page est rendue par le serveur, puis l'application est chargée dans un « package » généré par un outil spécifique, tandis que les rendus suivants sont effectués par le navigateur.

Cette démarche solutionne à la fois :

-Le temps de chargement initial ;
- Le problème de référencement.

Mais on créé alors d'autres problématiques, qui nécessitent d'introduire encore de nouveaux outils et concepts, avec en particulier :

- La mise en place d'un système de packaging, tels WebPack ou Browserify ;
- Un nouveau paradigme d'architecture Web, avec un modèle qui peut être :

- Action > Sélecteur > Dépôt > Composant pour Flux (proposé initialement comme compagnon à React par Facebook) ;
- Action > Réducteur> Dépôt > Composant pour Redux (solution plus récente et plus élégante).

Ce qui implique de maîtriser encore de nouveaux concepts et outils.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Conclusion

Le développement de SPA isomorphiques ne cadre pas avec un objectif de simplicité et le gain attendu ne semble pas au rendez-vous ou n'est pas pertinent pour nos objectifs. Il est possible que, dans le cadre d'une application très modulaire, ce type de développement ait un intérêt.

Développer des applications réactives, performantes et maintenables ne nécessite pas de SPA isomorphique. Donc maîtriser React, Redux et WebPack n'est qu'une voie parmi d'autres.

On pourrait même dire qu'il est préférable d'éviter les SPA isomorphiques pour simplifier le développement et la maintenance. Mais si l'architecture SPA est retenue, alors pour être « Mobile Friendly » et référençable, elle devra forcément être isomorphique.

Il y a une contradiction entre l'objectif d'une expérience réactive, de type « Client lourd », typique d'une application dynamique et le référencement, qui n'est pertinent que pour du contenu statique.

Des tests démontrent que tant que les volumes de données ne sont pas énormes[2], le rendu par le client ou le serveur n'a pas d'importance et le critère de décision sera plutôt le délai avant le début d'affichage (rendu par le serveur - pour mieux faire patienter l'utilisateur) ou la fin d'affichage (rendu par le client - quand elle est réellement fonctionnelle).

Les pratiques « à la mode » rendent les choses inutilement complexes et finalement non efficientes.

Toutes ces recherches auront permis de distinguer l'excellent concept de DOM virtuel, qui peut être simplement mis en œuvre par Mithril, un framework ultra compact et très rapide.

Liens

https://medium.com/@oleg008/isomorphic-rendering-d3e39c3ed073
http://tech.m6web.fr/isomorphic-single-page-app-parfaite-react-flux
https://pragmaticpatterns.com/2015/12/12/redux-un-nouveau-paradigme-darchitecture-web

[2] http://www.onebigfluke.com/2015/01/experimentally-verified-why-client-side.html?m=1

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Ressources non retenues

NumberSix law
Les vastes technologies qui découlent de Javascript s'étendent à perte de vue, dans la plaine immense où reposent déjà tant de cadavres.

1 Introduction

La liste ci-dessous ne stigmatise aucun composant. Elle est le produit d'une analyse théorique réalisée par un incompétent notoire. Picorez avec modération et vérifiez par vous-mêmes.

2 Meteor

Dans l'optique de réduire le problème des dépendances et de l'environnement, en tenant compte de la problématique initiale, la solution Meteor a retenu l'attention.

Meteor est un Framework qui intègre Node.js, avec son propre outil de build et permettant le dev web front et back, ainsi que le dev Android et iPhone, via Phonegap.

Conséquence de ce choix :

- Délégation d'une partie de la gestion de l'environnement de développement à Meteor ;
- Délégation d'une grosse partie de la « glue logique » à Meteor ;
- Développement multi-plateformes Web et Mobiles unifiés ;
- Environnement Meteor réputé pour sa courbe d'apprentissage aisée et son usage *dev friendly*.

Bien sûr, il faut rappeler que Meteor est un framework spécialisé dans la réalisation d'applications métier, optimisé pour « traiter des données dynamiques ». Il n'est pas approprié pour servir des « données statiques » en volume, les solutions web classiques le font très bien.

2.1 Limites

Après lecture de l'ouvrage « Meteor in action », des doutes persistent sur l'apport réel de ce Framework. Meteor propose un développement plus simple pour le concepteur et une expérience utilisateur plus réactive et donc proche de celle d'un client lourd.

Meteor ajoute un niveau d'abstraction mais la simplification obtenue n'est pas celle attendue, car l'approche de React est supérieure à celle de Meteor/Blaze, la version de Node.js utilisée est très ancienne et l'attrait d'une expérience utilisateur « temps-réel », selon l'architecture retenue est discutable :

- Cette fonctionnalité n'est pas toujours utile ;
- Elle peut-être obtenue autrement ;
- Elle limite actuellement le choix de la base de données à (principalement) MongoDB ;
- Elle délègue systématiquement une grosse partie des traitements sur le poste client.

Ce dernier point appelle d'autres critiques :

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

- Une partie du processus métier se retrouve alors, de facto, en « open source » pour l'utilisateur ;
- Les navigateurs sont tellement inefficaces qu'une généralisation de cette architecture est susceptible de les ralentir encore plus (le miniMongoDB local est chargé en mémoire) ;
- Les applications métiers habituelles ne sont pas si consommatrices de ressources qu'une application bien conçue, installée sur un serveur approprié, ne puisse les gérer, même pour quelques centaines d'utilisateurs simultanés.

Pour toutes ces raisons, Meteor n'a pas été retenu.

D'autres frameworks similaires auraient pu être envisagés, comme Meatier ou DoneJS mais cela revient toujours à remplacer un cadre contraint par un autre.

Au lieu de frameworks, nous avons besoin de bons concepts et de bons composants simples.

2.2 Liens

http://www.sitepoint.com/7-reasons-develop-next-web-app-meteor
https://www.quora.com/What-are-the-short-term-and-long-term-pros-and-cons-of-React-vs-Meteor-for-developing-a-webapp
http://marmelab.com/blog/2015/11/27/meteor-webpack-react-redux.html
http://www.slant.co/topics/389/viewpoints/10/~node-js-web-frameworks~meteor

Exemples

https://www.quora.com/Whats-the-most-impressive-Meteor-example-w-source
https://github.com/SachaG/Telescope
http://www.telescopeapp.org
Meteor repository, sidebar menu : https://atmospherejs.com/xiaoyuer/gnmenu

Formation

https://bulletproofmeteor.com
https://www.quora.com/Where-can-good-Meteor-tutorials-be-found-other-than-the-official-ones
http://www.gajotres.net/top-five-meteor-js-video-tutorials
https://www.youtube.com/watch?v=hgjyr6BPAtA
https://www.toptal.com/meteor/building-real-time-web-applications-with-meteor
http://www.meteorpedia.com/read/Tutorials
http://blog.dataflows.io/technology/2015/04/29/meteor_typescript.html

3 CofeeScript

Croire qu'une sémantique telle que :

```
# Condition
number = -42 if opposite

# Object
math =
  root:   Math.sqrt
  square: square
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

```
  cube:   (x) -> x * square x
```

Est plus lisible que :

```
// Condition
if (opposite) {
  number = -42;
}

// Object
math = {
  root: Math.sqrt,
  square: square,
  cube: function(x) {
    return x * square(x);
  }
};
```

Mériterait, selon cette logique, une *reconversion* en WhiteSpace, version de BrainFuck où les instructions sont des tabulations, des espaces et des retours chariot, ce qui rend le code invisible dans un éditeur, et donc aussi les bugs, et puisqu'on ne les vois pas, c'est qu'il n'y en a pas.

4 React

React est une lame de fond qui redéfinit le développement des interfaces utilisateurs par l'utilisation d'un DOM virtuel. Il permet également le développement d'applications SPA isomorphiques.

http://facebook.github.io/react/docs/getting-started.html

4.1 Limites

La courbe d'apprentissage est réputée assez facile, les performances en rendu client sont moyennes, mais c'est sans importance en développement SPA isomorphique, et l'implémentation est assez complexe. React prend tout son sens dans le développement d'applications SPA isomorphiques, deux concepts qui ont été rejetés.

React aurait été notre choix si nous n'avions pas trouvé Mithril, qui implémente également un DOM Virtuel, avec d'autres avantages plutôt décisifs.

4.2 Liens React

http://www.universalmind.com/blog/technology/react-js-a-designers-perspective
http://putaindecode.io/fr/articles/js/react
http://blog.andrewray.me/reactjs-for-stupid-people
http://blog.andrewray.me/flux-for-stupid-people

Pourquoi React n'a pas de templates : https://www.youtube.com/watch?v=DgVS-zXgMTk

Intégration de React dans Meteor par l'utilisation standard de NPM (ce que ne permet pas par défaut Metor) : https://github.com/jedwards1211/meteor-webpack-react

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Pour comprendre le choix React vs Blaze : https://kadira.io/blog/meteor/introducing-blaze-plus-the-next-generation-of-blaze

4.3 Liens React avec JSX

JSX est un port de XHP (en PHP) qui fut introduit pour prévenir des injections XSS.

https://facebook.github.io/react/docs/jsx-in-depth.html
https://medium.com/modern-user-interfaces/dear-templating-sincerely-jsx-part-1-1df99c599001
https://medium.com/javascript-scene/jsx-looks-like-an-abomination-1c1ec351a918

4.4 Liens React sans JSX

❑ Liens raw React

Learning RR, part 1 : http://jamesknelson.com/learn-raw-react-no-jsx-flux-es6-webpack
Learning RR, part 2 : http://jamesknelson.com/learn-raw-react-ridiculously-simple-forms
Learning RR, part 3 : http://jamesknelson.com/routing-with-raw-react

❑ Liens concepts

http://jamesknelson.com/structuring-react-applications-higher-order-components
http://jamesknelson.com/react-js-by-example-interacting-with-the-dom
http://jamesknelson.com/taming-css-globals-with-react-without-webpack-or-inline-style
http://jamesknelson.com/push-state-vs-hash-based-routing-with-react-js

https://github.com/jamesknelson
http://unicornstandard.com/packages/uniloc.html

https://www.packtpub.com/books/content/using-reactjs-without-jsx

Avec JSX :

```
var Hello = React.createClass({
    render: function() {
        return <div>Hello {this.props.name}</div>;
    }
});

React.render(<Hello name="World" />, document.body);
```

Sans JSX :

```
var Hello = React.createClass({
    render: function() {
        return React.DOM.div({}, 'Hello ' + this.props.name);
    }
});
Hello = React.createFactory(Hello);

React.render(Hello({name: 'World'}), document.body);
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

5 Enzyme

Enzyme est une bibliothèque de tests pour React. Il n'est pas retenu puisque React n'est pas retenu.

5.1 Liens

Enzyme : https://github.com/airbnb/enzyme
https://medium.com/airbnb-engineering/enzyme-javascript-testing-utilities-for-react-a417e5e5090f

6 Redux

Redux est un paradigme d'architecture couplé à React dans le cadre d'applications SPA isomorphiques. Il n'est pas retenu puisque ni React, ni ce modèle d'architecture n'ont été retenus.

6.1 Liens

Redux : http://redux.js.org
https://egghead.io/series/getting-started-with-redux
http://www.jchapron.com/2015/08/14/getting-started-with-redux
https://medium.com/modern-user-interfaces/how-we-redux-part-1-introduction-18a24c3b7efe
https://lainelouis.wordpress.com/2016/02/02/redux-explique-a-ma-mere
http://teropa.info/blog/2015/09/10/full-stack-redux-tutorial.html

7 Browserify, Grunt, Gulp

Ce sont des outils de « packaging » (plus ou moins) comme WebPack, concept obligatoire dans le cadre d'applications SPA isomorphiques. Aucun n'est retenu dans cette optique puisque cette architecture n'a pas été retenue.

Un seul composant de packaging est retenu, pour le build et pour l'expérimentation : WebPack. En effet, ce n'est pas parce que le concept SPA ne m'intéresse globalement pas qu'il faut se priver d'expérimenter ou de refuser cette solution dans un cas particulier où elle apparaitrait souhaitable.

❑ Liens

http://stackoverflow.com/questions/35062852/npm-vs-bower-vs-browserify-vs-gulp-vs-grunt-vs-webpack
https://npmcompare.com/compare/browserify,grunt,gulp,webpack
http://callmenick.com/post/an-introduction-to-gulp

8 Express.js

Express.js est un micro-framework généralement associé à Node.js. Il permet de gérer les Templates et les Routes, fonctionnalités que nous n'utiliseront pas, grâce à Mithril.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Depuis sa version 4, le sous-ensemble accédant à connect.js a été supprimé et l'on doit désormais utiliser directement, si nécessaire, connect.js, ou mieux, se tourner vers KoaJS, une alternative à la fois plus compacte et plus moderne. Pour toutes ces raisons, Express.js n'a pas été retenu.

9 DB NoSQL

Comparaison des principales bases NoSQL :
http://zohararad.github.io/presentations/outgrowing-mongodb

9.1 MongoDB

MongoDB est probablement la base NoSQL phare. MongoDB est écrit en C++, il est très bien documenté, le langage de base est JS, le clustering est assez complexe, et il peut être utilisé comme système de fichier via GridFS.

Site : https://www.mongodb.com
Documentation : https://docs.mongodb.com/manual
MongoDB manager : https://robomongo.org

9.2 SQL ou NoSQL ?

http://erthalion.info/2015/12/29/json-benchmarks
https://www.quora.com/Which-database-should-I-use-for-a-killer-web-application-MongoDB-PostgreSQL-or-MySQL
https://speakerdeck.com/mitsuhiko/a-year-of-mongodb

9.3 Conclusion

Après recherches, il est confirmé que PostgreSQL est techniquement très largement supérieur à toute solution fondée sur une base NoSQL et ce malgré la surcouche d'un moteur SQL, ce qui démontre la qualité et l'avancée technologique de PostgreSQL ou la médiocrité technique des solutions NoSQL.

Par ailleurs, les fonctionnalités JSON et JSONB de PostgreSQL, mais également l'existence du serveur Postrest font de PostgreSQL le meilleur choix objectif pour l'écosystème node.js. Cette conclusion est assez iconoclaste, dans le monde de la MEAN[3] stack.

PostgreSQL offre le meilleur des deux mondes, avec la performance et la fiabilité en plus.

[3] MongoDB Express.js AngularJS Node.js

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Ressources à étudier

1 Outils

1.1 Outils de déploiement et de packaging

En cas de nécessité.

❑ Build simple via transpile-watch & chokidar

A adapter pour rajouter l'étape TypeScript.
https://github.com/ArthurClemens/transpile-watch
https://github.com/paulmillr/chokidar

❑ Déploiement intermédiaire

Afin de limiter les outils et simplifier l'apprentissage, le déploiement peut être réalisé par de simples scripts utilisant la commande NPM, pour lancer des outils tiers à travers leur API. Éventuellement, l'aide de Webpack et de Browser-sync peut être un plus.

Exemple : https://github.com/kriasoft/react-starter-kit :

- react-starter-kit-master\tools\deploy.js ;
- react-starter-kit-master\docs\recipes\using-npm-and-webpack-as-a-build-tool.md ;
- Autres recherches par les mots clés npm, webpack et browser-sync.

❑ Alternatives aux outils de packaging

http://www.echojs.com/news/17367
https://www.pluralsight.com/courses/javascript-systemjs-jspm
http://stealjs.com
http://nervosax.com/2015/08/05/why-not-try-jspm-and-systemjs

1.2 Générateur de documentation

Générateur de documentation Javascript en Javascript.
https://esdoc.org
https://github.com/esdoc/esdoc

2 Composants généraux

2.1 KoaJS

Nouveau gestionnaire de middleware très compact et qui remplace avantageusement connect.js, sous ensemble middleware d'Express.js

Site : http://koajs.in
Source : https://github.com/koajs/koa

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

3 Composants spécifiques

3.1 Authentification

https://scotch.io/tutorials/easy-node-authentication-setup-and-local
https://truongtx.me/2014/03/29/authentication-in-nodejs-and-expressjs-with-passportjs-part-1

3.2 Générateur d'états

❑ Jsreport

En Javascript, sorties en PDF ou Excel, sous la forme d'un serveur indépendant fondé sur Node.js et contrôlé par une API REST.

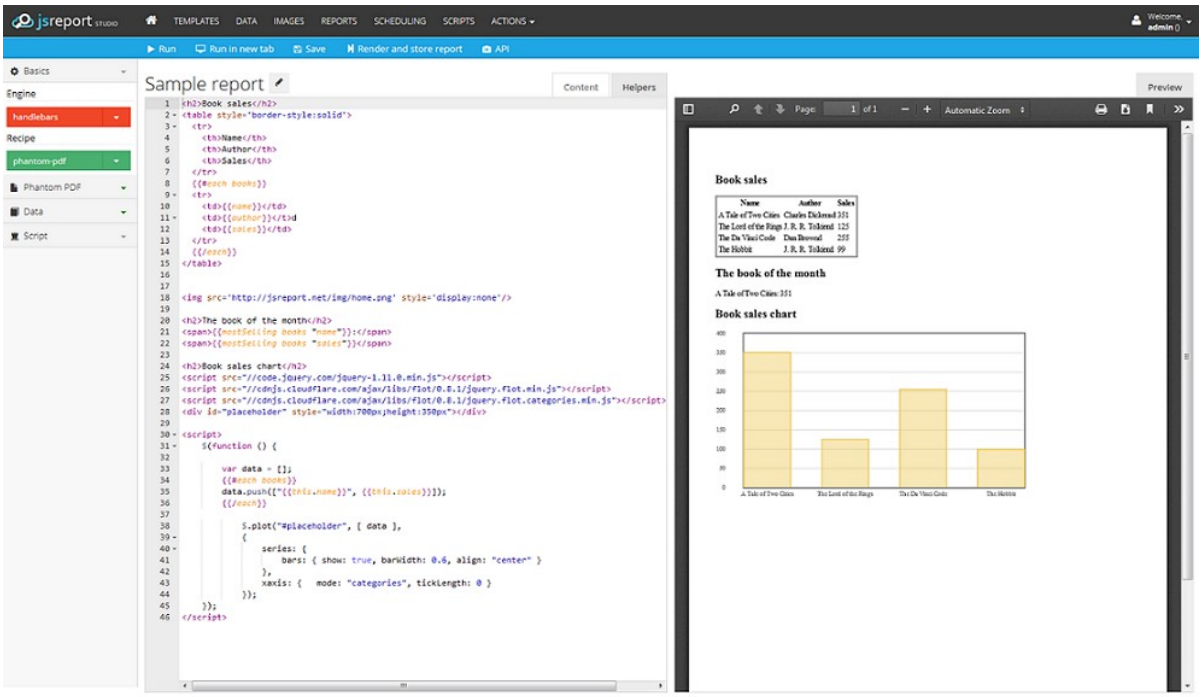

Illustration 2: JsReport Studio pour la conception des états

Site : http://jsreport.net
Sources : https://github.com/jsreport
Tutorial : http://jsreport.net/learn/get-started

3.3 Graphiques

Principalement fondées sur D3 : NDV3, C3, Metricsgraphics, DC, dimple, Epoch, D3xter, Charts, Sparklines, Taucharts, Plottablejs : http://blog.webkid.io/javascript-chart-libraries

Sans dépendances : https://github.com/andreaferretti/paths-js - voir .../paths-js-demo

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

3.4 i18n

i18n : https://github.com/musterknabe/translate.js

3.5 PDF

❑ pdfmake

Site : http://pdfmake.org
Code : https://github.com/bpampuch/pdfmake
Playground : http://pdfmake.org/playground.html

3.6 Sidebar menus

menu sidebar : https://github.com/CodyHouse/responsive-sidebar-navigation
menu sidebar off-canvas : https://github.com/sjardim/responsive-sidebar-navigation

3.7 Tests

Mocha : https://mochajs.org - Tests async debug node.js
Chai : http://chaijs.com - Tests BDD/TDD

4 Mobile

En développement mobile Javascript, la solution traditionnelle est Apache Cordova et ses compléments comme Adobe PhoneGap. L'application résultante est un hybride, avec une expérience utilisateur dégradée.

Une alternative existe, conceptuellement dérivée de React : React Native. Comme son nom l'indique, React Native produit des applications natives Android et IPhone. Cette voie sera à explorer le moment venu.

4.1 Liens

Cordova : https://cordova.apache.org
PhoneGap : http://www.phonegap.com
Native : https://www.npmjs.com/package/nativescript
React-native : https://facebook.github.io/react-native

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Ressources retenues

> NumberSix law
> Avec tous ces beaux outils, on a fait une belle application, et puis on se rappelle qu'elle doit faire quelque chose. Et là on cherche comment faire des formulaires.

La liste des ressources retenues n'est pas limitative et évoluera au fur et à mesure de l'expérience.

1 Javascript

Si l'on veux utiliser un langage unique, il n'y a aucun choix à faire, puisque le seul langage disponible dans les navigateurs est Javascript[4].

Résoudre le problème revient à trouver une solution permettant d'utiliser Javascript coté serveur avec une productivité et une fiabilité correctes.

2 Node.js

Pour Javascript, la BaseStack serveur la plus performante et la plus adoptée est Node.js.

Node.js est un environnement d'exécution multi-plateforme pour Javascript. Javascript n'a jamais été pour le concepteur de Node.js la finalité de son environnement.

Les prémices de Node.js étaient programmables en C, puis en LUA, mais le concepteur de Node.js était toujours insatisfait car ces langages possèdent des bibliothèques aux E/S bloquantes.

Finalement, il intégra le moteur Javascript de Google (V8 engine) car ce langage possédait enfin les caractéristiques qu'il recherchait (closures, first-class functions, absence -sic- de bibliothèques standards) pour mettre en valeur la raison d'être de Node.js : être un environnement d'exécution fondé sur un gestionnaire événementiel et non bloquant d'E/S, afin d'augmenter les performances dans un environnement réseau transactionnel.

Le livre « Hands-on Node.js » est très éclairant sur tous ces points.

Toutefois, le choix de Node.js n'est pas suffisant et doit s'accompagner d'autres choix moins évidents, d'autant plus que la problématique est placée assez haut (cf. début du document).

2.1 Liens

https://nodejs.org

[4] Javascript est le nom officieux d'ECMAScript, répondant à la norme ECMA-262. Ecma international est une association européenne de standardisation dans le domaine des systèmes d'information. Javascript s'appelait à l'origine Moka, puis LiveScript, ou encore JScript. Javascript a été initialement conçu par Netscape. Selon le bon mot de Douglas Crockford, l'un de ses développeurs initiaux, Javascript est un « Lisp habillé en C ». http://javascript.crockford.com/javascript.html

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

2.2 Liens formation

http://sdz.tdct.org/sdz/des-applications-ultra-rapides-avec-node-js.html

3 Typescript, Babel & Source Map

3.1 Typescript

Typescript est un sur-ensemble de Javascript, co-créé par Anders Hejlberg[5], une personne extrêmement respectable dans le domaine des langages.

Typescript, qui intègre ES6, permet de transformer Javascript en un langage de programmation plus acceptable pour des projets conséquents, en délivrant un code d'exécution de niveau ES6, avec l'annotation et l'inférence[6] de type, les énumérations et les apports d'ES6 comme les classes, les modules et les interfaces.

Typescript est un langage complet et dispose d'un compilateur open source. Tout code ou bibliothèque Javascript sont acceptés sans modification par Typescript. Les apports de Typescript (typage, classe, etc...) sont toujours optionnels et ces additions disparaissent à l'exécution, afin de ne pas abaisser les performances.

TypeScript ne pose donc pas de problème de compatibilité. Il peut transpiler directement en ES5.

❑ Liens

https://www.typescriptlang.org

Excellente vidéo de présentation, par son créateur :
https://channel9.msdn.com/posts/Anders-Hejlsberg-Introducing-TypeScript.

❑ Lectures

TypeScript deep dive : https://github.com/basarat/typescript-book
TypeScript Language Specification - 1.8 :

3.2 Babel

Les buts de Babel et de TypeScript ne sont pas similaires. Babel permet d'écrire en ES6. Babel peut d'ailleurs être utilisé en parallèle avec Typescript. Les objectifs de ce dernier, qui intègre également ES6 dans sa définition, nous semblent supérieurs à Babel, qui génère un code Javascript nettement moins lisible que Typescript et ne permet pas l'annotation de type.

Babel est retenu pour ses meilleures qualités de transpiling ES5, incluant une meilleure couverture de polyfills, avec Typescript désormais en front, afin d'offrir le meilleur des deux mondes

---

[5] Créateur de Turbo Pascal, co-fondateur de Borland, concepteur de Delphi, de J++, des WFC, de .NET et responsable du développement de C#.
[6] Recherche automatique des types associés à des expressions.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

❑ Liens

https://www.quora.com/Should-I-go-with-Babel-or-TypeScript-for-writing-web-apps
http://www.thinkingincrowd.me/2015/12/26/TypeScript-vs-Babel
https://medium.com/@dferber90/es6-in-meteor-5e088c998e4a#.ld4lmy6ty

3.3 Source Map

http://blog.getsentry.com/2015/10/29/debuggable-javascript-with-source-maps.html

3.4 Chaîne des transpilers

TypeScript > ES6 > Babel > ES5 > Source maps TypeScript Source pour le débuggage.

❑ Liens pour WebPack

https://www.npmjs.com/package/awesome-typescript-loader
http://jbavari.github.io/blog/2015/10/20/windows-and-webpack-with-typescript-and-babel

4 Webpack

Quoique l'usage d'un « Packager » ne soit pas obligatoire dans l'environnement envisagé, on peut considérer WebPack comme, entre autres, une alternative à des scripts de build.

Compte tenu de son emploi courant, connaître WebPack, afin de maîtriser au moins un « Packager », est un « plus » en terme de formation (on sent l'absence d'enthousiasme).

En particulier, pour créer un environnement acceptant simultanément TypeScript, Javascript ES6 et générant des Sourcemap pour le debuggage, WebPack est probablement une solution pertinente.

4.1 Liens

Webpack : https://webpack.github.io

5 Mithril

Mithril est conceptuellement similaire à React (DOM virtuel) mais avec une couverture plus vaste, une API réduite et une documentation très abondante, dans une bibliothèque minuscule (14 fonctions, moins de 1000 lignes de code), très performante[7] et sans dépendance.

Mithril est un framework MVC client, hiérarchique par composants et sûr par défaut. Sa rapidité est obtenue par un DOM virtuel différentiel couplé à un système automatique de rafraîchissement.

> « Compared to React, you are getting an order of magnitude faster performance with an order of magnitude less code. » [8]

[7] http://lhorie.github.io/mithril/benchmarks.html
[8] https://medium.com/@l1ambda/mithril-vs-angular-vs-react-d0d659c24bae

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

De plus, Mithril supporte TypeScript via un fichier de définition de types déjà créé (se reporter à la page des outils pour l'installation) et dispose d'un module de routes, qui peuvent être traversées par une liste, par programmation ou par des liens.

5.1 Liens

http://mithril.js.org
https://lhorie.github.io/mithril
http://lhorie.github.io/mithril-blog
https://lhorie.github.io/mithril/tools.html

Qui utilise Mithril : https://github.com/lhorie/mithril.js/wiki/Who-Uses-Mithril

5.2 Liens Formation

http://lhorie.github.io/mithril/getting-started.html

http://ratfactor.com/daves-guide-to-mithril-js

http://gilbert.ghost.io/mithril-js-tutorial-1
http://gilbert.ghost.io/mithril-js-tutorial-2
http://mindeavor.github.io/blog-post-examples/index.html (demos du tutorial)

Discussion concernant React et le virtual DOM avec les exemples traduits pour Mithril par l'auteur de Mithril, avec une seconde discussion encore plus intéressante :
http://jlongster.com/Removing-User-Interface-Complexity,-or-Why-React-is-Awesome
http://lhorie.github.io/mithril-blog/an-exercise-in-awesomeness.html
Article en relation : http://dailyjs.com/2014/12/26/json-resume-jsnox

5.3 Liens Composants

https://github.com/lhorie/mithril.js/wiki
https://github.com/lhorie/mithril.js/wiki/Recipes-and-Snippets

Form model : https://github.com/ludbek/mithril-form
Form validator : https://github.com/Nijikokun/mithril-validator

Animations avec Mithril : http://jsguy.github.io/mithril.animate
Two way automatic bindings for mithril : https://github.com/jsguy/mithril.bindings

Dependency injector : https://gist.github.com/ilsenem/11345055

The templating system brings together the ability to add templates in the controller, inside the dom or loaded via ajax - all able to be nested : https://github.com/jsguy/mithril.templates

Sugar tags for mithril : https://github.com/jsguy/mithril.sugartags

https://github.com/blanchg/mithril-webpack

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

https://github.com/devel-pa/mithril-mantle
https://github.com/barneycarroll/sm-pagination
https://github.com/barneycarroll/modulator
https://github.com/farzher/mithril-livescript-todomvc
https://github.com/ng-vu/mithril-bootstrap
https://github.com/barneycarroll/mithril.attributes
https://github.com/barneycarroll/mithril.exitable.js
https://github.com/dontwork/mithril-select
https://github.com/barneycarroll/mithril.observer.js

5.4 Liens Material Design Specification

http://www.google.com/design/spec/material-design/introduction.html

❑ Polymer

https://github.com/ArthurClemens/mithril-polymer-demo/pull/1
https://groups.google.com/forum/#!topic/mithriljs/PRD3RlcsLDc

❑ Polythene

La tuerie à maîtriser d'urgence.

Core : https://github.com/ArthurClemens/Polythene
Exemples : https://github.com/ArthurClemens/Polythene-examples

Documentation : http://polythene.js.org
Démo : http://arthurclemens.github.io/Polythene-examples

⇨ Tout le repository de l'auteur est à examiner pour les utilitaires Polythene et les composants Mithril :https://github.com/ArthurClemens?tab=repositories

❑ Material Design Lite

Mithril google material design lite components : https://github.com/jsguy/mithril.component.mdl
Composants MDL : https://getmdl.io
Référence MDL : https://getmdl.io/components/index.html

5.5 Liens Custom Elements

Explications :http://knockoutjs.com/documentation/component-custom-elements.html
Composable custom element types : https://github.com/jsguy/mithril.elements
Mithril.Elements Starter Kit : https://github.com/philtoms/mithril-starter-kit

http://blog.caplin.com/2014/04/02/experimenting-with-custom-elements

Composable apps : https://github.com/kopa-app/mithril-app

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

5.6 Liens Postgrest

https://github.com/catarse/mithril.postgrest

http://postgrest.com
https://github.com/begriffs/postgrest
https://github.com/begriffs/postgrest/wiki/Performance-and-Scaling

5.7 Liens Outils

JSX pour Mithril : https://github.com/insin/msx
Mithril-query : https://github.com/StephanHoyer/mithril-query
Find where DOM node was created : https://github.com/StephanHoyer/mithril-source-hint
Inspecting & debugging : https://github.com/barneycarroll/mithril-debug

5.8 Liens applications

PNI application : https://github.com/pdfernhout/narrafirma (spa)

Welcome to ze Bank : https://github.com/curiousercreative/bankDemo-mithril <- excellent
Demo : http://curiousercreative.com/demos/bankDemo-mithril

Serveur de jeux d'échecs : https://github.com/ornicar/lila (http://fr.lichess.org)
Playground de jeu d'échec : https://github.com/ornicar/chessground
Dashboard : https://github.com/martinp/jarvis2 (python backend)
Presentation engine : https://github.com/matthiasak/mithril-slide-engine
Programming Blog : https://github.com/jon49/programming-blog
Chat with expressJS et socket.io : https://github.com/Bondifrench/Node-Mithril-Chat
Real time trader desktop : https://github.com/Bondifrench/mithril-trader

5.9 Liens Applications Isomorphiques

Présentation : https://gist.github.com/StephanHoyer/bddccd9e159828867d2a
Server side rendering : https://github.com/jsguy/mithril-node-render
Exemple : https://github.com/matthiasak/mithril-isomorphic-example

https://github.com/matthiasak/mithril-ssr-test

Exploring universal/isomorphic es6/babel/express with Mithril :
https://github.com/matthiasak/wrinklefree-mithril A React Resolveresque Higher Order Component for the Mithril VDOM library to write universal / isomorphic lazy-loading views :
https://github.com/matthiasak/mithril-resolver

Lichess mobile : https://github.com/veloce/lichobile
High order resolver : https://github.com/matthiasak/mithril-resolver
dstllry.co : https://github.com/matthiasak/dstllry.co

Isomorphic framwework using mithril : https://github.com/jsguy/misojs
Exemples et applications, chercher « misojs- » dans : https://github.com

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

5.10 Liens Applications Mobiles

https://github.com/Bondifrench/mithril-employee-directory

5.11 Liens Comparaison

Riot : https://news.ycombinator.com/item?id=8928433 (search Riot apparently requires)
Angular, React : https://medium.com/@1lambda/mithril-vs-angular-vs-react-d0d659c24bae
React-Lite : https://github.com/Lucifier129/react-lite
Pract : https://github.com/developit/preact
Preact-Compat : https://github.com/developit/preact-compat
Deku : https://github.com/dekujs/deku

6 J2C

Dans la logique de Mithril, une petite bibliothèque d'intégration du CSS dans Javascript et contenant un sous-ensemble de SASS (Syntactically Awesome Style Sheets).

Une intégration de j2c dans Mithril existe, ainsi qu'un convertisseur de fichiers CSS et une implémentation de pocketgrid, parfait pour des formulaires. Des opérations mathématiques sont également possibles sur les longueurs CSS

6.1 Liens

Site : http://j2c.py.gy
Sources : https://github.com/j2css/j2c
Intégration de j2c dans Mithril : https://github.com/j2css/mithril-j2c
j2c-pocketgrid : https://github.com/j2css/j2c-pocketgrid

pocketgrid : https://github.com/j2css/j2c-pocketgrid
sass : http://sass-lang.com

7 DB SQL

> **NumberSix law**
> **SQL est non standard, fait sale dans le code, rend complexe ce qui était simple et, grâce au trou noir du moteur et de sa grammaire, retourne une interprétation au lieu d'un résultat.**

Ceci ne veux pas dire que toutes les bases de données relationnelles sont à jeter, bien au contraire.

7.1 PostgreSQL

Si on peux dépolluer le code source des requêtes SQL pour leur substituer une interface JSON naturelle en Javascript, alors pourquoi ne pas profiter des performances et de la fiabilité de PostgreSQL, puisqu'elle gère désormais JSONB et devient donc, de facto, une base de donnée NoSQL ?

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

PostgREST

Serveur web autonome qui transforme PostgrSQL en une RESTful API.

http://postgrest.com
https://github.com/begriffs/postgrest

RESTful for PostgreSQL

Serveur web autonome qui semble similaire à PostgREST, mais en plus léger.

https://github.com/QBisConsult/psql-api
https://www.npmjs.com/package/psql-api

Massive.js

http://massive-js.readthedocs.io/en/latest

Massive permet d'échanger les données en JSONB afin d'avoir une gestion des données transparente au niveau du code (pas de langage de requête, les données sont des instances d'objets).

On obtient alors un code aussi propre que si l'on utilisait une base NoSQL, tout en pouvant utiliser SQL, si nécessité. Massive permet donc aussi bien d'éliminer le SQL...

```
$ db.users.find({email: 'jane.doe@gmail.com'}, function(err, res){console.log(res)});
$ db.users.find({'created_at >': '2015-09-24'}, function(err, res){console.log(res)});
```

... que de l'utiliser en l'éjectant du code source :

```
$ echo 'SELECT * FROM users WHERE email = $1' > ./db/user_lookup.sql
$ db.user_lookup(['jane.doe@gmail.com'], function(err,res){console.log(res);});
```

http://www.craigkerstiens.com/2015/12/08/massive-json
http://www.craigkerstiens.com/2015/11/30/massive-node-postgres-an-overview-and-intro
http://blog.j0.hn/post/48591247017/using-json-in-postgres-and-nodejs

Autres solutions

Moins radicales et pour certaines plus portables :

https://github.com/brianc/node-postgres
https://github.com/vitaly-t/pg-promise
http://vincit.github.io/objection.js/#introduction

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Environnement

1 Introduction

1.1 Environnement local

Compte tenu de l'usage de TypeScript, un des critères sera l'assistance fournie par l'éditeur à ce langage, en particulier via une aide de type IntelliSense (Visual Studio, Eclipse, GPS, etc...), mais également la facilité d'intégration de l'écosystème node.js, avec un bon support Git et bien sûr open-source et multi-plateforme (Windows, Linux et Mac).

1.2 Environnement Cloud

A partir d'une entreprise « sans bureaux », avec ses outils « dans le Cloud », il est également apparu un besoin « d'environnement de développement collaboratif », afin de proposer :

- Une environnement de développement commun et standardisé ;
- Des outils de collaboration spécifiques aux développeurs ;
- La possibilité de développer à partir de n'importe quel terminal Web (option à explorer).

A la recherche d'un environnement de développement unique, collaboratif, fondé sur le Web et le Cloud, plusieurs solutions ont été examinées.

https://iprodev.com/20-best-code-editors-for-real-time-collaboration
https://en.wikipedia.org/wiki/List_of_collaborative_software
https://en.wikipedia.org/wiki/Collaborative_real-time_editor

Cloud9 est généralement plébiscité, Codebox semble mort, Codiad n'est pas loin de l'agonie, d'autres solutions sont finalement assez limitées, les solutions existantes « prêtes à l'emploi », « dans le Cloud » et réellement libres sont plus que rares.

La plupart de ces solutions utilisent l'éditeur ACE https://github.com/ajaxorg/ace, qui provient d'un projet défunt des Mozilla Labs : Skywriter, anciennement connu sous le nom de Bespin. ACE est désormais maintenu par Cloud9.

2 Environnement local : Visual Studio Code

D'une façon assez surprenante, un produit Microsoft remplit tous ces critères : Visual Studio Code !

Visual Studio Code n'est pas une version dégradée de Visual Studio mais un autre produit, fondé sur l'éditeur d'Atom, le code de la console Chromium et Node.js. A noter que Visual Studio Community aurait pu convenir si il n'avait été limité pour un usage professionnel et disponible sur une autre plateforme que Windows. Microsoft fait du bon libre avec du libre... dans quelle étagère ?

2.1 Utilisation

L'utilisation est immédiate et la documentation est excellente.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Le mode IntelliSense fait des merveilles avec TypeScript.

Présentation : http://johnpapa.net/getting-started-with-visual-studio-code

2.2 Personnalisation

https://code.visualstudio.com/Docs/languages/javascript

❑ TypeScript

https://code.visualstudio.com/Docs/languages/typescript
https://blogs.msdn.microsoft.com/typescript/2015/04/30/using-typescript-in-visual-studio-code
http://blog.wolksoftware.com/setting-up-your-typescript-vs-code-development-environment

❑ Linting

ESlint est disponible en extension

https://github.com/eslint/typescript-eslint-parser
https://github.com/Microsoft/vscode-eslint/issues/5
http://www.sitepoint.com/comparison-javascript-linting-tools

❑ Extensions

https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome
[Ctrl+P] ext install debugger-for-chrome [Enter]
https://marketplace.visualstudio.com/items?itemName=cssho.vscode-svgviewer
[Ctrl+P] ext install SVG Viewer [Enter]
https://marketplace.visualstudio.com/items?itemName=Shan.code-settings-sync
[Ctrl+P] ext install code-settings-sync [Enter]

❑ Suppression de la télémétrie

File > Preferences > User Settings > "telemetry.enableTelemetry": false

2.3 Liens

Site : https://code.visualstudio.com
Documentation : https://code.visualstudio.com/docs
Extensions : https://marketplace.visualstudio.com/vscode

3 Environnement local : Atom

Visual Studio Code est plus rapide qu'Atom et possède IntelliSense mais Atom dispose du package atom-typescript, qui semble même disposer de fonctionnalités supérieures.

Atom possède une bibliothèque de packages bien plus vaste.
La gestion des paramètres est également plus simple et plus complète.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

3.1 Personnalisation

❑ Packages

https://atom.io/packages

Atom > Préférences > Install > Install Packages
- atom-typescript
- minimap
- joe-keybinds (bindings wordstar)
- export-html (impression en pdf)
- fold-functions
- fold-lines

❑ Divers

Atom > Préférences > Settings > Editor Settings > Font Size
- 12 (au lieu de 14)

3.2 Liens

https://atom.io
http://flight-manual.atom.io

https://github.com/TypeStrong/atom-typescript

La vidéo présentant la version 1.0 (un bijou) : https://www.youtube.com/watch?v=Y7aEiVwBAdk

4 Environnement Cloud : Cloud9

Cloud9, produit phare de cette famille, est désormais en GPL V3, les contributions sont très actives, et deux possibilités d'utilisation existent :

- Une plate-forme commerciale, avec une instance gratuite, ou à des coûts très faibles pour une équipe de développement, incluant une instance système de développement (via docker) ;
- Installer Cloud9 dans sa propre infrastructure. Il n'a pas encore été déterminé si la version disponible sur GitHub est aussi avancée que la version payante mais cette alternative est rassurante.

Dans sa version commerciale, Cloud9 propose plusieurs environnements courants.

Une instance gratuite Cloud9 comprend une VM Ubuntu, avec 1 Cpu, 768 Mo de Ram et 2 Go de disque. Les instances payantes vont de $19/mois pour un individuel à $29/mois pour des équipes avec dans les deux cas un nombre d'instances illimitées et un nombre suffisant d'instances « hot » (toujours allumées, afin de gagner du temps à la reconnexion). On peut également « inviter » des collaborateurs sur une instance.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Utilisation

Dans le cadre de l'évaluation, une instance gratuite Cloud9 a été créée sur https://c9.io :

La première impression très est bonne. L'expérience utilisateur est bien là. La console est à portée de main. L'ensemble dégage une excellente impression.

L'intégration TypeScript est réalisée par un simple « npm install typescript » dans la console.

La fonction IntelliSense manque. En cas d'erreur de syntaxe sur du code un peu lourd, ou pour assister les débutants et les distraits, c'est bien dommage. Je ne suis pas le seul à remarquer cette lacune, peut-être peut-on espérer une amélioration de ce coté.

Liens

site : https://github.com/c9/core
setup : http://blog.viniciusprado.org/how-to-install-and-configure-cloud9-in-your-own-server
conf nginx : https://gist.github.com/wake/5852204
nginx, Php7fpm on Cloud9 workspaces : https://github.com/GabrielGil/c9-lemp
Pros & Cons de Cloud9 : http://www.slant.co/topics/713/viewpoints/1/~cloud-ides~cloud9

5 Git

<<<TODO>>>

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Installation

1 Installation Windows

Sur Windows 7 pro 64 bits

<<<TODO>>>

2 Installation Mac

Sur Mac OSX 10.9.5 Mavericks

2.1 Visual Studio Code

Version v1.1, installation à partir de l'installeur du site, dans :
- /Applications

Créer un raccourci sur le bureau.
Personnaliser.

2.2 Atom

Version 1.7.3, installation à partir de l'installeur du site, dans :
- /Applications

Créer un raccourci sur le bureau.
Personnaliser.

2.3 Node.js

Version v4.4.4 LTS, installation, à partir de l'installeur du site, dans :
- /usr/local/bin/node
- /usr/local/bin/npm

❑ Mise à jour des droits

```
root@system npm config get prefix

/usr/local

// La maj des droits n'impactera pas le système
// i.e. npm n'est pas dans /usr ou /usr/bin
// Donc on peut procéder à la modification :

root@system sudo chown -R $(whoami) $(npm config get prefix)/{lib/node_modules,bin,share}
```

2.4 TypeScript

Installation des packages :

```
// command line tool => installation globale => -g option
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

```
root@system npm install -g typescript

/usr/local/bin/tsc -> /usr/local/lib/node_modules/typescript/bin/tsc
/usr/local/bin/tsserver -> /usr/local/lib/node_modules/typescript/bin/tsserver
typescript@1.8.10 /usr/local/lib/node_modules/typescript

// Vérification

root@system tsc

Version 1.8.10
Syntax:   tsc [options] [file ...]

.
.
.

// TypeScript est bien installé
```

2.5 WebPack, Babel & awesome-typescript-loader

Comme nous aurons des sources TypeScript ou des sources ES6, il nous faut deux points d'entrées, en paramétrant TypeScript pour générer de l'ES6, et non pas directement de l'ES5 :

- Sources ES6 > Babel > Sources ES5 > Web
- Sources TypeScript > Sources E6 > Babel > Sources ES5 > Web

Il faut également générer les Sourcemap pour le debug.

Installation des packages. Compter 5 bonnes minutes, les sorties sont laissées pour donner une idée des dépendances :

```
// Webpack
root@system npm install -g webpack


>fsevents@1.0.12
install
/usr/local/lib/node_modules/webpack/node_modules/watchpack/node_modules/chokidar/node_modules/fseve
nts
> node-pre-gyp install --fallback-to-build

[fsevents] Success:
"/usr/local/lib/node_modules/webpack/node_modules/watchpack/node_modules/chokidar/node_modules/fsev
ents/lib/binding/Release/node-v46-darwin-x64/fse.node" is installed via remote
/usr/local/bin/webpack -> /usr/local/lib/node_modules/webpack/bin/webpack.js
webpack@1.13.0 /usr/local/lib/node_modules/webpack
├── interpret@0.6.6
├── tapable@0.1.10
├── async@1.5.2
├── clone@1.0.2
├── loader-utils@0.2.14 (object-assign@4.1.0, big.js@3.1.3, json5@0.5.0, emojis-list@1.0.2)
├── enhanced-resolve@0.9.1 (graceful-fs@4.1.4, memory-fs@0.2.0)
├── acorn@3.1.0
├── mkdirp@0.5.1 (minimist@0.0.8)
├── supports-color@3.1.2 (has-flag@1.0.0)
├── optimist@0.6.1 (wordwrap@0.0.3, minimist@0.0.10)
├── memory-fs@0.3.0 (errno@0.1.4, readable-stream@2.1.2)
├── webpack-core@0.6.8 (source-list-map@0.1.6, source-map@0.4.4)
├── uglify-js@2.6.2 (async@0.2.10, uglify-to-browserify@1.0.2, source-map@0.5.6, yargs@3.10.0)
├── node-libs-browser@0.5.3 (https-browserify@0.0.0, tty-browserify@0.0.0, path-browserify@0.0.0,
constants-browserify@0.0.1, punycode@1.4.1, string_decoder@0.10.31, os-browserify@0.1.2,
process@0.11.3, assert@1.3.0, domain-browser@1.1.7, querystring-es3@0.2.1, timers-browserify@1.4.2,
stream-browserify@1.0.0, events@1.1.0, util@0.10.3, vm-browserify@0.0.4, readable-stream@1.1.14,
console-browserify@1.1.0, url@0.10.3, buffer@3.6.0, http-browserify@1.7.0, browserify-zlib@0.1.4,
crypto-browserify@3.2.8)
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

```
└── watchpack@0.2.9 (graceful-fs@4.1.4, async@0.9.2, chokidar@1.5.0)


// Ligne de commande pour utiliser directement babel (tests)
root@system npm install -g babel-cli


> fsevents@1.0.12
 install /usr/local/lib/node_modules/babel-cli/node_modules/chokidar/node_modules/fsevents
> node-pre-gyp install --fallback-to-build

[fsevents] Success: "/usr/local/lib/node_modules/babel-
cli/node_modules/chokidar/node_modules/fsevents/lib/binding/Release/node-v46-darwin-x64/fse.node"
is installed via remote
/usr/local/bin/babel-doctor -> /usr/local/lib/node_modules/babel-cli/bin/babel-doctor.js
/usr/local/bin/babel -> /usr/local/lib/node_modules/babel-cli/bin/babel.js
/usr/local/bin/babel-node -> /usr/local/lib/node_modules/babel-cli/bin/babel-node.js
/usr/local/bin/babel-external-helpers -> /usr/local/lib/node_modules/babel-cli/bin/babel-external-
helpers.js
babel-cli@6.8.0 /usr/local/lib/node_modules/babel-cli
├── path-is-absolute@1.0.0
├── slash@1.0.0
├── path-exists@1.0.0
├── fs-readdir-recursive@0.1.2
├── log-symbols@1.0.2
├── convert-source-map@1.2.0
├── commander@2.9.0 (graceful-readlink@1.0.1)
├── v8flags@2.0.11 (user-home@1.1.1)
├── source-map@0.5.6
├── chalk@1.1.1 (escape-string-regexp@1.0.5, ansi-styles@2.2.1, supports-color@2.0.0, has-
ansi@2.0.0, strip-ansi@3.0.1)
├── glob@5.0.15 (inherits@2.0.1, once@1.3.3, inflight@1.0.4, minimatch@3.0.0)
├── output-file-sync@1.1.1 (xtend@4.0.1, mkdirp@0.5.1)
├── request@2.72.0 (oauth-sign@0.8.2, aws-sign2@0.6.0, tunnel-agent@0.4.3, forever-agent@0.6.1, is-
typedarray@1.0.0, caseless@0.11.0, stringstream@0.0.5, aws4@1.4.1, isstream@0.1.2, json-stringify-
safe@5.0.1, extend@3.0.0, tough-cookie@2.2.2, node-uuid@1.4.7, qs@6.1.0, combined-stream@1.0.5,
mime-types@2.1.11, form-data@1.0.0-rc4, bl@1.1.2, hawk@3.1.3, http-signature@1.1.1, har-
validator@2.0.6)
├── babel-core@6.8.0 (babel-messages@6.8.0, babel-template@6.8.0, shebang-regex@1.0.0, babel-
helpers@6.8.0, private@0.1.6, babel-code-frame@6.8.0, debug@2.2.0, minimatch@2.0.10, babylon@6.8.0,
babel-types@6.8.1, babel-generator@6.8.0, babel-traverse@6.8.0, json5@0.4.0)
├── bin-version-check@2.1.0 (minimist@1.2.0, semver@4.3.6, semver-truncate@1.1.0, bin-
version@1.0.4)
├── lodash@3.10.1
├── babel-register@6.8.0 (home-or-tmp@1.0.0, mkdirp@0.5.1, source-map-support@0.2.10, core-
js@2.4.0)
├── babel-polyfill@6.8.0 (babel-regenerator-runtime@6.5.0, core-js@2.4.0)
├── babel-runtime@6.6.1 (core-js@2.4.0)
└── chokidar@1.5.0 (inherits@2.0.1, glob-parent@2.0.0, async-each@1.0.0, is-glob@2.0.1, is-binary-
path@1.0.1, readdirp@2.0.0, anymatch@1.3.0, fsevents@1.0.12)


// Transpiler ES6 > ES5
root@system npm install -g babel-preset-es2015


babel-preset-es2015@6.6.0 /usr/local/lib/node_modules/babel-preset-es2015
├── babel-plugin-transform-es2015-destructuring@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-typeof-symbol@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-block-scoping@6.8.0 (babel-types@6.8.1, babel-template@6.8.0,
babel-traverse@6.8.0, lodash@3.10.1, babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-for-of@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-arrow-functions@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-literals@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-duplicate-keys@6.8.0 (babel-types@6.8.1, babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-computed-properties@6.8.0 (babel-helper-define-map@6.8.0, babel-
template@6.8.0, babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-template-literals@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-spread@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-check-es2015-constants@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-function-name@6.8.0 (babel-types@6.8.1, babel-helper-function-
name@6.8.0, babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-block-scoped-functions@6.8.0 (babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-modules-commonjs@6.8.0 (babel-plugin-transform-strict-mode@6.8.0,
babel-template@6.8.0, babel-types@6.8.1, babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-unicode-regex@6.8.0 (regexpu-core@1.0.0, babel-helper-
regex@6.8.0, babel-runtime@6.6.1)
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

```
├── babel-plugin-transform-es2015-shorthand-properties@6.8.0 (babel-types@6.8.1, babel-
runtime@6.6.1)
├── babel-plugin-transform-es2015-sticky-regex@6.8.0 (babel-helper-regex@6.8.0, babel-types@6.8.1,
babel-runtime@6.6.1)
├── babel-plugin-transform-es2015-parameters@6.8.0 (babel-helper-get-function-arity@6.8.0, babel-
helper-call-delegate@6.8.0, babel-template@6.8.0, babel-types@6.8.1, babel-traverse@6.8.0, babel-
runtime@6.6.1)
├── babel-plugin-transform-es2015-object-super@6.8.0 (babel-helper-replace-supers@6.8.0, babel-
runtime@6.6.1)
├── babel-plugin-transform-es2015-classes@6.8.0 (babel-messages@6.8.0, babel-helper-replace-
supers@6.8.0, babel-helper-optimise-call-expression@6.8.0, babel-helper-function-name@6.8.0, babel-
template@6.8.0, babel-helper-define-map@6.8.0, babel-types@6.8.1, babel-traverse@6.8.0, babel-
runtime@6.6.1)
└── babel-plugin-transform-regenerator@6.8.0 (babel-plugin-syntax-async-functions@6.8.0,
private@0.1.6, babylon@6.8.0, babel-types@6.8.1, babel-traverse@6.8.0, babel-core@6.8.0, babel-
runtime@6.6.1)


// Babel Webpack
root@system npm install -g babel-core


babel-core@6.8.0 /usr/local/lib/node_modules/babel-core
├── slash@1.0.0
├── babel-messages@6.8.0
├── babel-template@6.8.0
├── path-exists@1.0.0
├── shebang-regex@1.0.0
├── path-is-absolute@1.0.0
├── babel-helpers@6.8.0
├── private@0.1.6
├── convert-source-map@1.2.0
├── debug@2.2.0 (ms@0.7.1)
├── source-map@0.5.6
├── babylon@6.8.0
├── babel-types@6.8.1 (to-fast-properties@1.0.2, esutils@2.0.2)
├── minimatch@2.0.10 (brace-expansion@1.1.4)
├── babel-code-frame@6.8.0 (js-tokens@1.0.3, esutils@2.0.2, chalk@1.1.3)
├── babel-traverse@6.8.0 (globals@8.18.0, invariant@2.2.1, repeating@1.1.3)
├── babel-generator@6.8.0 (trim-right@1.0.1, detect-indent@3.0.1, is-integer@1.0.6,
repeating@1.1.3)
├── json5@0.4.0
├── lodash@3.10.1
├── babel-register@6.8.0 (home-or-tmp@1.0.0, mkdirp@0.5.1, source-map-support@0.2.10, core-
js@2.4.0)
└── babel-runtime@6.6.1 (core-js@2.4.0)


// Babel loader pour Webpack
root@system npm install -g babel-loader


npm WARN peerDependencies The peer dependency babel-core@^6.0.0 included from babel-loader will no
npm WARN peerDependencies longer be automatically installed to fulfill the peerDependency
npm WARN peerDependencies in npm 3+. Your application will need to depend on it explicitly.
npm WARN peerDependencies The peer dependency webpack@1 || ^2.1.0-beta included from babel-loader
will no
npm WARN peerDependencies longer be automatically installed to fulfill the peerDependency
npm WARN peerDependencies in npm 3+. Your application will need to depend on it explicitly.
babel-loader@6.2.4 /usr/local/lib/node_modules/babel-loader
├── object-assign@4.1.0
├── loader-utils@0.2.14 (big.js@3.1.3, json5@0.5.0, emojis-list@1.0.2)
└── mkdirp@0.5.1 (minimist@0.0.8)

// Typescript loader pour Webpack
root@system npm install --save-dev awesome-typescript-loader

awesome-typescript-loader@0.17.0 node_modules/awesome-typescript-loader
├── object-assign@2.1.1
├── strip-json-comments@2.0.1
├── strip-bom@2.0.0 (is-utf8@0.2.1)
├── colors@1.1.2
├── loader-utils@0.2.14 (object-assign@4.1.0, big.js@3.1.3, json5@0.5.0, emojis-list@1.0.2)
├── es6-promisify@3.0.0 (es6-promise@3.1.2)
├── enhanced-resolve@0.9.1 (tapable@0.1.10, graceful-fs@4.1.4, memory-fs@0.2.0)
├── parse-json@2.2.0 (error-ex@1.3.0)
├── source-map-support@0.4.0 (source-map@0.1.32)
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

```
├── tsconfig@2.2.0 (array-uniq@1.0.2, xtend@4.0.1, pinkie-promise@2.0.1, globby@4.0.0)
├── lodash@3.10.1
└── babel-polyfill@6.8.0 (babel-regenerator-runtime@6.5.0, babel-runtime@6.6.1, core-js@2.4.0)
```

2.6 Git

<<<TODO>>>

❑ Liens

https://git-scm.com

3 Installation Linux

Sur Debian Jessie

<<<TODO>>>

4 Mini projet pour valider l'environnement

<<<TODO>>>

Typescript + Babel + WebBack > minify.js + sourcemaps, avec jeu de test genre test_ts.ts, test_es6.js, test_es5.js + précédure de test de debug.

https://anybox.fr/blog/developpement-front-end-en-javascript-outils-de-base
http://devlog.disco.zone/2016/04/28/manygolf-typescript
http://www.pgbovine.net/react-babel-webpack-javascript-es6-setup.htm

windows typescript awesome-typescript-loader babel webpack
http://jbavari.github.io/blog/2015/10/20/windows-and-webpack-with-typescript-and-babel

avec ts-loader
http://www.jbrantly.com/es6-modules-with-typescript-and-webpack

avec typescript sans babel avec sourcemaps
http://www.jbrantly.com/typescript-and-webpack

https://www.npmjs.com/package/awesome-typescript-loader
https://github.com/andreypopp/typescript-loader

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

NumberSix law
L'objectif doit être : « Learn Once - Write everywhere ».

1 Généralités

Le plan de « formation initiale » est adapté à un développeur « junior ». Une formation de « remise à niveau » pour un développeur « senior » pourra être élaborée à partir de ces ressources. Ce plan inclus la formation aux langages, aux composants et aux concepts et méthodes de l'entreprise.

2 Livres

CSS3 for Dummies - 2014
HTML5 Programming with JavaScript For Dummies - 2013

Typescript Specifications 1.8
TypeScript dive book : https://github.com/basarat/typescript-book

Node.js the right way - 2013
Hands on Node.js - 2015
The node beginner book - 2015
The node craftsman book - 2015

High performance browser networking - 2013
WebRTC : APIs and protocoles of the HTML5 Real-Time Web - 3rd edition - 2013

3 Ressources

https://github.com/matthiasak/frontend-dev-bookmarks
http://jsbooks.revolunet.com
https://github.com/vioan/nodejs-learning-resources
https://leanpub.com <- excellent

3.1 Bases de données

http://blog.ragingflame.co.za/2014/7/21/using-nodejs-with-mysql
http://mherman.org/blog/2015/02/12/postgresql-and-nodejs

3.2 Debug

http://blog.johnnyreilly.com/2015/12/es6-typescript-babel-react-flux-karma.html
http://elm-lang.org/blog/time-travel-made-easy

3.3 Emscripten

https://github.com/kripken/emscripten/wiki

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

3.4 Jeux et animations

http://blog.sklambert.com/html5-game-tutorial-game-ui-canvas-vs-dom

https://fr.wikipedia.org/wiki/WebGL
https://github.com/mrdoob/three.js
https://mrdoob.github.io/three.js/examples/webgl_materials_cars.html

3.5 Node.js

http://justinklemm.com/node-js-async-tutorial

4 Roadmap de formation

<<<TODO>>>

5 Bonnes pratiques

5.1 Le cache - Rémi Jolin (bar@ovh)

La taille du code (html + css + js) qui est transmis au navigateur, même s'il est « énorme », est (presque) un faux problème :

- Tout le monde supporte la compression et le texte ascii la supporte très bien. Il faut juste penser à le mettre en œuvre sur le frontal (en même temps que le https).
- Si l'application le gère bien, les navigateurs modernes peuvent cacher les pages, mais aussi stocker de la donnée locale. Pour gérer le « Caching » (le forcer), il y a la notion de « cache manifest » où sont décrits les éléments à cacher (ou pas) et leur version.

Voir par exemple http://www.html5rocks.com/en/tutorials/appcache/beginner

Le stockage local de données peut aussi permettre d'avoir des applications html qui fonctionnent offline (utile pour le mobile) sans avoir à faire une application native.

A ce sujet : https://mobile.angular.io

5.2 La compression - Rémi Jolin (bar@ovh)

La compression vient en plus de toute les actions préalables de minification/inlining à faire sur le code "de production" et qui fontaussi baisser sérieusement la taille du code (et accessoirement, ça rend un peu plus compliqué le reverse engineering sur le code).

Sur l'inlining, il faut être plus circonspect, surtout avec l'arrivée de HTTP/2. Les quelques tests réalisés avec des serveurs web qui le supportent montrent des gains très importants sur les temps de chargement d'un grand nombre de fichiers (site avec beaucoup de petites images), dû au multiplexage des chargements entre le navigateur et le serveur (une seule socket pour plusieurs requêtes lancées en parallèle plutôt que successivement ou sur plusieurs sockets).

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Conclusions

1 Le contexte

Souhaiter l'utilisation d'un langage unique n'est pas une fin en soit et il faut aussi considérer l'intégralité de l'écosystème nécessaire. Sans précautions, un environnement de développement Javascript va très vite ressembler à ça :

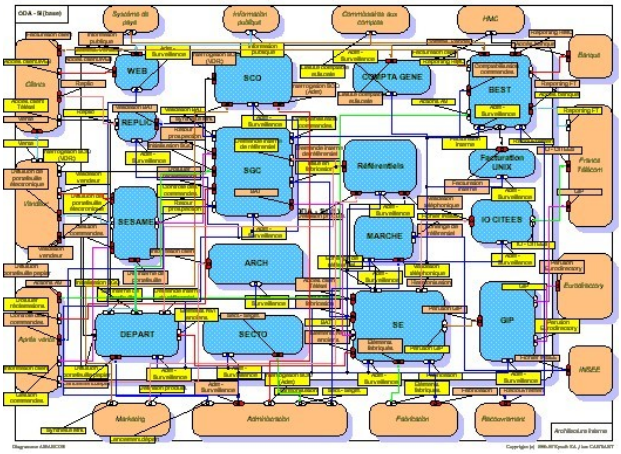

Illustration 3: Pourquoi faire simple ?

NumberSix law
En développement Web, le ratio du volume et de la complexité du code par rapport aux fonctionnalités obtenues est, la plupart du temps, incroyablement mauvais.

Les environnements de développement Javascript sont devenus énormes. Ils peuvent tout faire, mais à quel coût ?

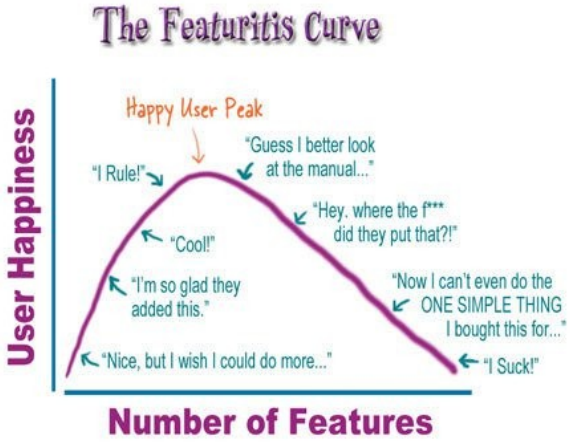

Illustration 4: Pourquoi se limiter au nécessaire ?...

Car s'investir dans leur maîtrise n'est pas récompensé puisque le développement informatique est une activité où règne la *mode*.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

> NumberSix law
> Les artisans font laborieusement des trucs qui tombent en marchent. Les « génies » font des trucs *super cools*, mais qui ne le sont pas, et qui ne marchent pas bien, mais juste assez pour qu'on mette beaucoup trop de temps à admettre, qu'en fait, c'était de *la merde*.

Et à chaque *mode* - qui, par essence, change souvent - ses nouveaux outils, ses nouveaux environnements, en fonction d'une évolution illogique et perverse, les meilleurs choix étant trop souvent délaissés au profit des options les plus naturellement stupides pour le commun des développeurs les plus médiocres.

De ce fait, un développeur devra régulièrement renouveler l'ensemble de ses connaissances, sans pour autant que la nouveauté soit justifiée par un gain de productivité.

2 Un (début de) solution

> Leonard de Vinci, Antoine de Saint-Exupéry & Albert Einstein laws
> La simplicité est la sophistication suprême. La perfection est atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher. Tout devrait être fait de manière simple mais non simpliste.

Le plus gros travail aura été de simplifier en ne gardant que l'essentiel.

> NumberSix law
> Un environnement de développement *normal* devrait être assez simple pour être appréhendé par une seule personne, dans un temps raisonnable, de quelques jours à quelques semaines.

La solution, orientée développeur et non pas designer web, est purement Javascript, sans HTML, sans CSS[9], sans JSX, sans templates.

Un seul langage, coté client, coté serveur, du JSON et rien d'autre.

Compacte, réactive et éprouvée, elle comporte quelques choix politiquement incorrects, par rapport aux tendances de l'écosystème.

Fondée sur un DOM Virtuel, mais sans React, elle n'adhère pas à la mode de la SPA isomorphique[10], qui reste toutefois possible.

---

[9] The worst things about CSS are the « Cascading » and the « Sheets » - Jed Schmidt
https://css-tricks.com/the-debate-around-do-we-even-need-css-anymore

[10] ...qui restera possible, quoiqu'elle n'apporte, au prix d'une complexité accrue, aucun bénéfice concret, voire des effets négatifs, comme l'a rapporté la discussion au premier chapitre.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

La solution est essentiellement composée de :

- Javascript ;
- TypeScript & Babel ;
- Node.js
- Mithril ;
- J2c ;
- WebPack ;
- PostgreSQL en mode JSONB (à la NoSQL) ;

3 Modules applicatifs

La recherche de la simplicité impose de sélectionner les vrais besoins, pour se limiter au fonctionnalités suffisantes : en d'autres termes, dans notre contexte, une application Webavec :

- Une UI de type « Material Design » : www.google.com/design/spec/material-design
- Des écrans avec menu side bar off-canvas
- Module d'authentification ;
- Modules standards CRUDS (Create, Read, Update, Delete, Search) connectés à une BD ;
- Module administrateur (gestion des utilisateurs, etc...) ;
- Générateur d'états ;
- États au format PDF.

3.1 Objectif d'UI

Interface fondée sur « Material Design » et sur une disposition similaire au back office de Wordpress ou d'OVH (sidebar à gauche et barre d'état et de connexion en haut) :

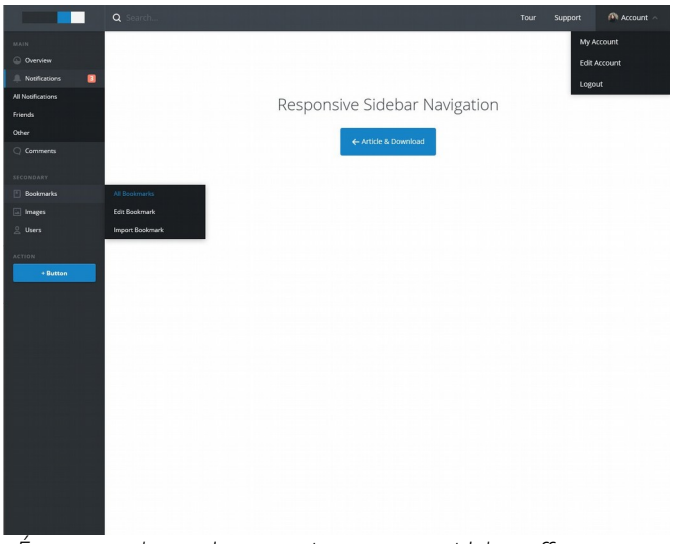

Illustration 5: Écran avec barre de connexion et menu sidebar off-canvas avec sous-menus

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Annexes

1 Mots Clés de recherche

real time collaborative open source
form in mithril

2 Lexique

❑ AMD, RequireJS et Common JS

Asynchronous Module Definition. est une API permettant de charger en parallèle et en asynchrone des modules Javascript, dans leur propre scope, donc sans conflit, en même temps que leur dépendances.

RequireJS est une implémentation d'AMD et offre également un wrapper à CommonJS.

CommonJS, anciennement ServerJS, est un groupe de définition de spécifications modules) Javascript quand ce langage est utilisé en dehors d'un navigateur. par exemple, Node.js implémente CommonJS.

❑ Asset

En développement Web, ressource basique devant être traitée par un navigateur Web, comme des fichiers images, multimedia ou du texte.

❑ BDD

BDD : https://fr.wikipedia.org/wiki/Behavior_driven_development

Mode de développement impliquant toutes les parties prenantes, et en particulier les utilisateurs, et générant une valeur commerciale par la qualité du code résultant et finalement l'acceptation du produit fini par les utilisateurs. Ce mode autorise un cycle de vie du produit techniquement et fonctionnellement apaisé. Également lié à l'Extreme Programming.

http://philippe.poumaroux.free.fr/index.php?post/2012/02/06/Introduction-au-Behaviour-Driven-Developement

http://behaviourdriven.org

❑ FRP

Functional Reactive Programming : https://en.wikipedia.org/wiki/Functional_reactive_programming

❑ Isomorphique (application)

Application écrite en Javascript aussi bien coté client que coté serveur.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

NPM

node package manager

Installé par défaut dans node.js

Off-Canvas (design)

Principe d'interface utilisateur, plus particulièrement utilisée sur les plateformes mobiles, où certains éléments d'interface sont en dehors de l'écran et deviennent accessible, généralement par la droite ou la gauche de l'écran, par un « glisser » du doigt.

Ce principe est très utilisé pour les menus adaptés aux mobiles, par exemple.

https://developer.mozilla.org/fr/docs/Tutoriel_canvas
https://developers.google.com/web/fundamentals/design-and-ui/responsive/patterns/off-canvas

Polyfill

En programmation Web, c'est un ensemble de fonctions en Javascript permettant de combler des incompatibilités provenant de navigateurs anciens.

Le nom provient d'une marque de résine « à tout faire » britannique : Polyfilla.

https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/les-polyfills-et-les-wrappers

Raw React

Utilisation de React *sans* JSX.

SPA

Single Page Application

Application web accessible via une page web unique. Le but est d'éviter le chargement d'une nouvelle page à chaque action demandée, et de fluidifier ainsi l'expérience utilisateur.
https://fr.wikipedia.org/wiki/Application_web_monopage

TDD

TDD : https://en.wikipedia.org/wiki/Test-driven_development

Mode de développement lié à la répétition de courts cycles de développements, lié au concept d'Extreme Programming.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

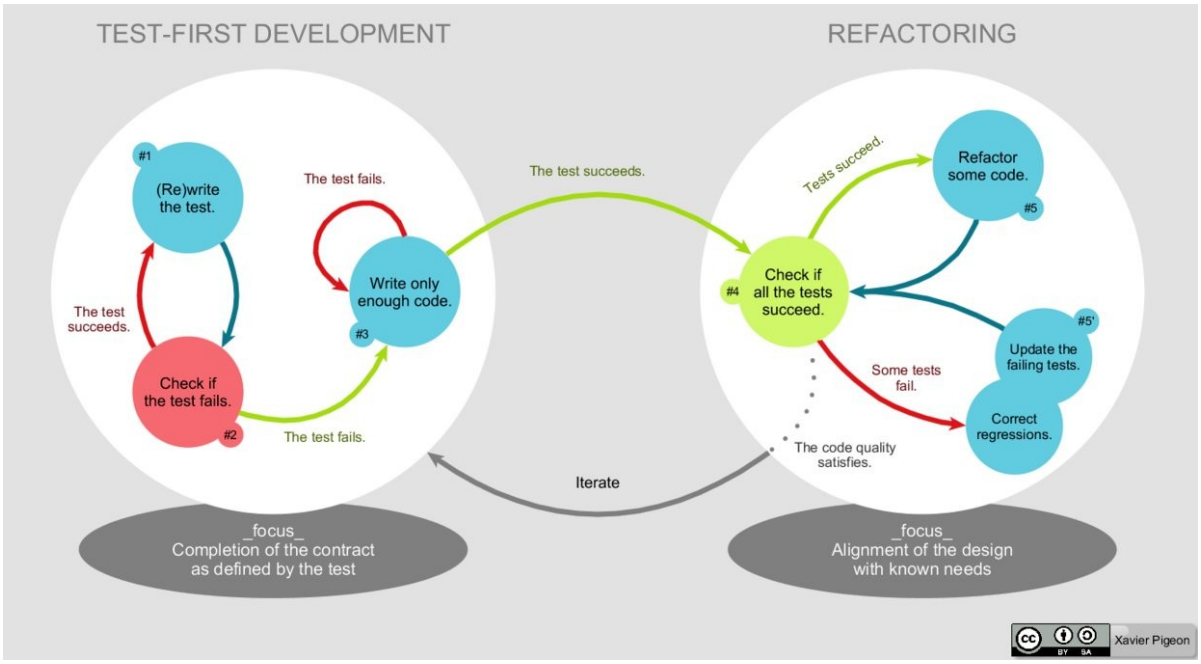

Illustration 6: Test Driven Development

❑ Transpiler

Convertisseur de code d'un langage à l'autre ou d'une version de langage à une autre.

Exemples dans la "galaxie explosée" de javascript :

- scala : https://www.scala-js.org
- typescript : https://www.typescriptlang.org (typage fort)
- babel : https://babeljs.io

❑ Wrapper

En programmation Web, permet l'ajout de propriétés ou de méthodes aux objets, en créant un objet dérivé. Principe très utilisé pour réutiliser en les modifiant des objets natifs sans les altérer.

https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/les-polyfills-et-les-wrappers

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

1 Binding Joe-Keybinds

```
atom-workspace

  ctrl-k b: joe:block-start
  ctrl-k k: joe:block-end
  ctrl-k ctrl-b: joe:block-start
  ctrl-k ctrl-k: joe:block-end
  ctrl-k c: joe:block-copy
  ctrl-k ctrl-c: joe:block-copy
  ctrl-k m: joe:block-move
  ctrl-k ctrl-m: joe:block-move

atom-text-editor

  ctrl-k: unset!
  ctrl-u: core:page-up
  ctrl-v: core:page-down
  ctrl-k u: core:move-to-top
  ctrl-k v: core:move-to-bottom
  ctrl-k ctrl-u: core:move-to-top
  ctrl-k ctrl-v: core:move-to-bottom
  ctrl-z: editor:move-to-beginning-of-word
  ctrl-x: editor:move-to-end-of-word
  ctrl-l: find-and-replace:find-next
  ctrl-w: editor:delete-to-end-of-word
  ctrl-o: editor:delete-to-beginning-of-word
  ctrl-j: editor:delete-to-end-of-line
  ctrl-c: core:close

body

  ctrl-k s: core:save
  ctrl-k ctrl-s: core:save
  ctrl-_: core:undo
  ctrl-^: core:redo
  ctrl-k e: application:open
  ctrl-k ctrl-e: application:open
  ctrl-k d: core:save-as
  ctrl-k ctrl-d: core:save-as
  ctrl-k o: pane:split-down
  ctrl-k ctrl-o: pane:split-down
  ctrl-k i: pane:close-other-items
  ctrl-k ctrl-i: pane:close-other-items

platform-darwin win32 linux
  ctrl-k f: find-and-replace:show
  ctrl-k ctrl-f: find-and-replace:show
  ctrl-k l: go-to-line:toggle
  ctrl-k ctrl-l: go-to-line:toggle
  ctrl-c: core:cancel

atom-text-editor:not([mini])
  ctrl-y: editor:delete-line

atom-workspace atom-text-editor:not([mini])

  ctrl-k .: editor:indent-selected-rows
  ctrl-k ,: editor:outdent-selected-rows

platform-darwin atom-text-editor:

  ctrl-k j: autoflow:reflow-selection
  ctrl-k ctrl-j: autoflow:reflow-selection
```


CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.

Les versions suivantes aborderont :

- Exemples de setup d'environnement pour Windows, Mac et Linux.
- Exemples réalistes... et non pas la classique démo de ToDoList en RAM (sans connexion BD) qui ne fait quasiment rien et n'apporte rien.
- Analyse des composants de la solution (internals et philosophie).

Il y a aura une suite d'ailleurs, avec un exemple d'application de gestion (certainement un gestionnaire de rapports d'activité avec un admin pour la gestion des users), histoire d'être dans le concret.

CC-by-nc-sa : Paternité, pas d'utilisation commerciale, partage des conditions initiales à l'identique.