Ce message a été republié sur Blog de Ludovic Feltz à 01:55:39, le 21/11/2014

Bien choisir son Frameworks JavaScript: AngularJS, Backbone, Ember, Knockout

Compte Blog de Ludovic Feltz

Choisir son Framework front end n’est jamais une chose facile, ce choix impactera le temps de chargement, la vitesse de développement et la maintenabilité du code source de votre site web. Il en existe des dizaines, dans cet article nous allons nous restreindre aux 4 plus fameux afin de vous aider à faire le bon choix.

Tous ces Frameworks permettent d’organiser votre code et possèdent le concept de vue, d’évènements, de modèle de donnée, routage ? Plusieurs critères sont à prendre en compte dans le choix d’un Framework, son poids par exemple définira le temps de chargement des pages ce qui est un critère important pour les applications mobiles, sa facilité d’utilisation, sa documentation et sa communauté active permettent de réduire le temps de développement. Nous allons commencer par un bref historique puis nous allons plonger dans le vif du sujet avec un exemple simple que nous allons reproduire en utilisant chacun de ces Frameworks

TODO : On gère pas connexion a la base de données, Single page application

Why backbone: <http://backbonetutorials.com/why-would-you-use-backbone/>

Compare frameworks : <http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/>

# Histoire :

**AngularJS**: Apparu en 2009 sous le nom de GetAngular il est utilisé par Misko Hevery un des ingénieurs qui l’a développé pour recréer une application web qui représentait plus de 17 milles lignes de code. En 3 semaines il est parvenu à réduire ce nombre à seulement 1000 lignes ce qui a convaincu Google de sponsoriser ce projet ce qui a créé sa renommée. Son but est de simplifier le développement et les tests de site web MVC en ajoutant du vocabulaire au code HTML de votre application.

**Backbone :** Crée en 2010 par Jeremy Ashkenas qui à aussi participer au développement de Coffee Script, il est très léger ce qui lui a permis de se faire un nom parmi les autres Frameworks JavaScript. Il repose principalement sur le modèle MVP (model view presenter). (key value binding and custom events, collection, connect to existing API with RESTful JSon interface)

**Ember :** Développé en 2007 par SproutIt puis par Apple il est finalement forké en 2007 par Yehuda Katz, un des principaux contributeurs de JQuery et Ruby. Il est utilisé par des grands noms tels que Yahoo, Groupon et ZenDesk. Il est basé sur une architecture MVC. Son avantage est de permettre le développement d’applications d’une seule page grâce à son système de route et de template (view, controller, models, router)

**Knockout :** Né en 2010 et maintenu comme un projet open source par Steve Sanderson, employé chez Microsoft. Simplifie l’utilisation du JavaScript en appliquant le pattern MVVM.

# Les chiffres

Le temps de chargement d’une page web est crucial pour sa réussite. Les utilisateurs ne montrent pas beaucoup de patience quant au temps de chargement d’une page web, c’est pourquoi il essentiel de prendre en compte le temps de chargement et d’initialisation d’une librairie. Malgré le nombre de ligne de code plus important que ses concurrent **AngularJS** est plus légère car elle n’a besoin d’aucunes dépendances. **Backbone** nécessite l’utilisation de **underscore** ce qui lui fait prendre du poids malgré son nombre de lignes de codes moins important (verifier qu’il a pas besoin de JQuery en plus, sinon il est intégré). **Ember** a besoin de **JQuery** et de **Handlebar** pour fonctionner, d’où son poids très important par rapport à ses concurrents. Enfin **Knockout** ne nécessite aucune autre librairie ce qui en fait la plus légère de ce comparatif. La taille de votre site impactera aussi, si un Framework est lourd mais son utilisation produit peu de ligne de code il pourra tout de même être avantageux.

Le nombre de contributeur on s’en fou en vrai, non ?

# Les concepts clefs

# Exemple similaire

# Exemple des fonctionnalités propres

# Développer son propre Framework ?

Utiliser un Framework connu à l’avantage d’offrir au développeur de la rapidité sur un Framework qu’il connait déjà et de faciliter le transfert de la maintenance du code source à un client ou à une équipe tierce. Mais alors pourquoi développer son propre Framework et réinventer la route ? Tout d’abord cela permet d’avoir plus de souplesse, on peut le modifier, l’adapter et le faire évoluer selon les besoins tout en restant plus léger qu’un Framework existant dont on n’utilisera probablement pas toutes les fonctions. On décide de l’architecture à adopter et on n’est pas restreint à celle imposé par le Framework que l’on utilise.

Malgré ces avantages réinventer la roue n’est pas toujours la bonne solution, même si les composants développé seront réutilisables au fils des projets un nouveau venu aura du mal en en comprendre les subtilités et mettra donc du temps à s’adapter. De plus il ne pourra pas forcement s’aider d’internet ou de documentations pour avancer. De plus Il faudra gérer vous-même la maintenance, les évolutions et les adaptations aux nouveautés des langages sur lesquels il repose. La compatibilité entre les navigateur par exemple, est un problème qui est …

Cette solution … **TODO**

Si l’envi vous prenais de développer votre propre Framework n’hésitez pas à visiter ce site : <http://blogs.infinitesquare.com/b/jonathan/archives/mon-framework-mvvm-a-moi>

# Utiliser un CDN ou héberger la librairie ? (Supprimer ça ?)

CDN : Content Delivery Network

Avantages : Réduit la latence, augmente le parralélisme, meilleur caching (dépend du CDN utilisé)

Latence : Quand un utilisateur essaye Personne éloignés de votre serveur mettent.

Economie bande passante : pas sur votre serveur

Oui pour un grosse lib (JQuery, AngularJS), probablement non sinon

Src : <http://encosia.com/3-reasons-why-you-should-let-google-host-jquery-for-you/>

# Pas de vainqueurs…

Même si **AngularJS** semble légèrement devant ses concurrents il n’en est pas le vainqueur à l’unanimité. En effet chacun se démarque par son approche différente du sujet. **Knockout** est un bon concurrent grâce à sa légèreté et la concision du code produit. **Amber** avec son système de routage peut être très utile. **TODO**

Il n’y a donc pas de réponse catégorique quant au choix d’un Framework ou d’utiliser son propre Framework, il dépendra avant tout du besoin, du temps et de l’équipe qui participera au projet.