JavaScript, un langage web?

BARASCUT Jérémy

25 mai 2013

Résumé

Le résumé (abstract en anglais) de mon article.

Table des matières

Chapitre 1

Introduction

Le développement JavaScript a nettement changé depuis sa conception. Il est facile d'oublier la mise en œuvre de JavaScript dans le navigateur Netscape, jusqu'au navigateur d'aujourd'hui avec de puissants moteurs, tels que le moteur V8 de Google.

Cela a été un chemin rocailleux impliquant renommage, fusion et normalisation éventuelle comme ECMAScript. Les capacités de JavaScript que nous avons aujourd'hui sont au delà des rêves les plus fous de ses concepteurs.

Malgré son succès et sa popularité, JavaScript est encore largement méconnu. Peu de gens savent que JavaScript est un langage puissant et orienté objet. Ils sont surpris d'en apprendre davantage sur certaines de ses fonctionnalités les plus avancées, telles que l'héritage, les modules et les espaces de noms. Alors, pourquoi JavaScript est il si mal compris ?

Un premier élément de réponse s'explique par précédentes implémentations de JavaScript boguées, et le second élément de réponse proviens du nom du préfixe Java suggérant que JavaScript est en quelque sorte lié à Java. En réalité la raison de cette incompréhension est totalement différente. La véritable raison est la façon dont la plupart des développeurs sont initiés à ce langage. Avec d'autres langages, tels que Python ou Ruby, les développeurs font généralement un effort concerté pour apprendre le langage avec l'aide de livres, screencasts et tutoriaux. Jusqu'à récemment, les développeurs recevaient des demandes pour ajouter un peu de validation de formulaire, peut être ajouter un album ou une galerie photo avec du code tout prêt ou un calendrier. Ils utilisaient des scripts qu'ils trouvaient sur Internet, appelé avec peu de compréhension du langage derrière le script. Après cette phase, certains développeurs ajoutent JavaScript à leurs CV.

Récement les moteurs JavaScript et les navigateurs sont devenus si puissant que la construction complète d'applications riches en JavaScript est non seulement faisable mais aussi de

plus en plus populaire. Les applications tels que Gmail et Google Maps ont ouvert la voie à une manière complètement différente de penser les applications Web, et les utilisateurs en réclament plus. Les entreprises embauchent pour répondre à la demande des développeurs JavaScript à temps plein. Ce n'est plus un sous-langage relégué à des scripts simples et un peu de validation de formulaire, il est désormais un langage autonome de son propre droit, avec un sérieux potentiel.

Cet afflux de popularité signifie qu'un grand nombre de nouvelles applications JavaScript sont en cours de construction, surtout avec l'émergence de toujours plus de périphériques autonomes comme les smartphones, les tablettes etc

Malheureusement, et peut être en raison de l'histoire de ce langage, beaucoup d'applications JavaScript sont mal conçus. Peu importe la raison, les modèles reconnus et les meilleures pratiques partent à la poubelle. Les développeurs ignorent les modèles architecturaux tels que le Modèle-Vue-Controleur (MVC), mêlant à la place un désordre de HTML et JavaScript à leurs applications .

Mon mémoire va plutôt présenter des bibliothèques, plateformes et frameworks pour structurer et construire vos applications complexe entièrement en JavaScript. Que se soit la partie serveur, les clients légers comme les navigateurs, les clients lourds avec les applications Windows 8 ou Gnome 3, les périphériques mobiles comme les smartphones, tablettes, tv connectés et liseuses ainsi que le stockage des données avec l'utilisation de bases de données. Ce mémoire inclut également la présentation d'outils utilisé pour créer une application de qualité avec l'inclusion des tests, de la documentation ...

Pour ceux qui veule en apprendre plus sur le langage JavaScript de nombreux livres sont disponibles pour apprendre à sa syntaxe et sa structure.

1.1 Histoire de JavaScript

1.1.1 Présentation

Javascript (js pour les intimes) est un langage de script créé en 1995 par Brendan Eich pour le compte de Netscape Communications.

A cet époque Netscape Communications domine le marché du web. La firme fournie le navigateur web le plus populaire Netcape Navigator mais aussi des logiciels serveurs.

Netcape Communications lançait un partenariat avec Sun Microsystems pour exploiter leur nouveau langage ainsi que sa VM multi-plateforme, Java. Le but étant d'utiliser Java au sein du navigateur du coté serveur afin de fournir des UI riche portable ainsi qu'un moyen d'accéder à des applications via le web à travers un simple navigateur.

Cependant, Java est perçu par Sun et Netscape comme un langage peu adapté à une utilisation simple applicable à la page web car trop professionnel et contraignant. Java est à l'époque en concurrence avec Visual C++ de Microsoft. Il faut donc un langage plus simple à écrire, plus facile à utiliser pour les développeurs débutant et à prendre en main.

Netscape et Sun décide de créer un nouveau langage répondant à cette demande et en confie la création a Brendan Eich, gourou technique chez Netscape. La seule condition est que le langage doit être "dans le style java" mais en moins puissant et clairement différenciés de java par les potentiels développeurs.

Comme l'a expliqué Brendan Eich dans une interview :

> JS devait « ressembler à Java », mais en moins avancé, [il devait] être le petit frère simplet de Java, son partenaire-otage. Et par-dessus le marché, je n'avais que dix jours pour pondre ça, ou on se retrouverait avec un truc pire que JS.

Javascript utilise la même syntaxe que Java (JS 1.0, mot-clés réservé et convention du JDK) mais il a du s'abstenir d'utiliser la syntaxe orientée-objet du langage, à une époque où la POO (Porgrammation Orientée Objet) était encore considérée comme un sujet réservé aux professionnels...

Brendan ne voulant pas écrire un langage diminué, il du trouver des ruses pour y glisser assez de puissance sans que celle-ci ne soit immédiatement visible aux profanes. Le langage devait rester simple et léger en apparence, tout en ayant assez de sophistication pour que des développeurs avancés soient à même d'en tirer des applications puissantes.

Même si Javascript et Java appartiennent à une famille syntaxique "de type C" (syntaxe des identifiants, accolades, opérateurs principaux, structures de contrôle...), ils ont des sémantiques

extrêmement différentes. Javascript à une philosophie très fortement inspirée de langage objet ou fonctionnels “purs” au premier rang desquels Scheme et Self, mais aussi certains aspects de LISP et SmallTalk.

Java est un langage statique (chaque variable est typée), compilée et dotée d'un typage fort, là ou JavaScript est dynamique, interprété avec un typage plus léger.

Java utilise un modèle d'héritage “classique”, mono-parent, basé sur l'héritage de classes. JavaScript s'appuie également sur les prototypes et autorise plusieurs paradigmes de programmation notamment les types impératif, fonctionnel et orienté-objet.

Les deux langages sont donc extrêment différents, à un niveau philosophique, fondamental et pratique.

1.1.2 Nom de code Mocha

JavaScript est développé sous le nom de code Mocha. Son nom officiel étant LiveScript. Dans les deux premières versions beta de Netscape Navigator 2.0, en septembre 1995, on trouve en effet LiveScript. A cet époque Miscrosoft n'avait pas encore collé une connotation négative à “Live”.

Cependant les marketeux ont voulu insister sur le rôle “collaboratif” de ca langage et tenter de récupérer un peu du prestige issu du marketing de Sun autour de Java. Du coup Brendan a du renommer LiveScript en JavaScript.

Peu de temps après (1996-1997), Netscape voulut faire bénéficier à JavaScript d'un processus formel de standardisation. ISO, IETF et le jeune W3C posaient chacun des problèmes distincts dans cet standardisation. L'ECMA (European Computer Manufacturers Association), un organisme de standardisation européen, en récupéra le bébé. Ainsi est sorti la première édition du standard ECMA-262 Ed. 1 : ECMAScript est le nom de la norme officielle, JavaScript étant la plus connue des implémentations. ActionScript 3 est une autre implémentation bien connue de ECMAScript, avec des extensions.

Au fil du temps, il était clair cependant que Microsoft n'avait pas l'intention de coopérer ou de mettre en oeuvre correctement JS dans Internet Explorer même si Microsoft n'avait pas de proposition concurrente et avaient une mise en oeuvre partielle (et divergent à ce point) sur le coté serveur avec .NET.

Ainsi en 2003, le travail de JS2/original-ES4 à été mis en sommeil.

Le prochain évènement a eu lieu en 2005. Il s'agit en fait de deux évènements majeurs de l'histoire de JavaScript. Tout d'abord Brendan Eich et Mozilla rejoignent Ecma en tant

que membres non-lucratif. Ont commencé alors les travaux sur E4X ECMA-357 en travaillant conjointement avec Macromedia.

Ainsi, avec Macromedia (racheté par Adobe), le travail a redémarré sur ECMAScript 4, dans le but de normaliser ce qui était en AS4 et sa mise en oeuvre dans SpiderMonkey.

Hélas, en 2007, Doug Crokfort puis Yahoo ont uni leurs forces avec Microsoft pour s'opposer à ECMAScript 4, ce qui a conduit à la spécification ECMAScript 3.1 effort.

Pendant que les géant s'affrontaient, la communauté de développeurs open source s'est mise au travail pour révolutionner ce que l'on pouvait faire avec JavaScript. Cet effort collectif a été déclenché en 2005, lorsque Jesse James Garrett a publié un livre blanc dans lequel il inventa le terme Ajax. Dans ce livre blanc, il décrit un ensemble de technologies, donc l'épine dorsale est JavaScript. La technologie est utilisée pour créer des applications web où les données peuvent être chargées en arrière-plan, en évitant la nécessité de recharger la page entière et aboutissant à des applications plus dynamiques. Suite à ce livre blanc, il en a résulté une période de renaissance de l'utilisation de JavaScript dirigée par les bibliothèques open source et les communautés qui se sont formées autour d'elles. Ainsi sont nées des bibliothèques telles que Prototype, JQuery, Dojo, Motools et bien d'autres.

En juillet 2008, les parties conflictuelles se sont réunies à Oslo. Cela a conduit début 2009, à l'accord final de renommer ECMAScript 2.1 à ECMAScript 5 et à conduire le langage vers l'avant avec la future norme connue sous le nom de "Harmony".

La 3ème édition (ES3) consitue le socle de la version la plus utilisée/répandue du langage. La 4ème est morte-née, et la 5ème (ES5), désormais implémentée dans tous les navigateurs modernes, sert de socle aux applications web modernes ainsi qu'à JavaScript côté serveur (avec notament Node.js).

Tout cela nous amène à aujourd'hui, JavaScript entre dans un cycle complètement nouveau et passionnant dans son évolution, son innovation et sa normalisation, avec de nouveaux développements tels que Node.js, permettant d'utiliser JavaScript coté serveur. L'arrivé de HTML5 dans le monde du web va également transformer l'évolution de JavaScript grâce aux HTML5 APIs, ces dernières vont permettre de controler les navigateurs coté-client, d'utiliser les web-sockets pour toujours plus de communications, obtenir des données sur des fonctionnalités tels que l'accéléromètre, la localisation GPS, et bien plus encore.

Actuellement JavaScript est disponible de base sur davantage de périphériques et de plates-formes que Java.

Même si l'explosion d'Android a fortement relancé le déploiement de Java qui s'enorgueillit de " plusieurs milliards de périphériques installés ", Java n'est pas tellement déployé sur d'autres

plates-formes mobiles, et n'est pas non plus automatiquement présent sur toutes les plates-formes desktop.

Allez trouver un seul desktop, laptop, smartphone, tablette ou liseuse qui n'ait pas une runtime JavaScript installé et qui ne s'en serve pas intensivement !

Dans certains cas, comme webOs, Firefox Mobile, JavaScript est au cœur-même de la plate-forme, constituant sa clé de voûte.

iOS, Android, Windows Phone et Blackberry ont une tendance forte à utiliser des web apps reposant très lourdement sur JavaScript. Petit à petit, les technologies collectivement appelées "HTML5" remplacent ce pourquoi on avait encore recours aux applets ou, plus souvent, à Flash.

Ainsi commence la révolution JavaScript.

1.2 EcmaScript

1.2.1 Présentation

ECMASCript est un langage de programmation de type script. Il est standardisé par Ecma International dans le cadre de la spécification ECMA-262. Il s'agit d'un standard, donc les spécifications sont mises en œuvre dans différents langages comme JavaScript ou ActionScript.

JavaScript évolue donc en fonction de l'avancement des standard de Ecma-International.

Comme vu précédemment JavaScript a vu le jour en décembre 1995 par Sun et Netscape.

En mars 1996, Netscape implémente le moteur JavaScript dans son navigateur web Netscape Navigator 2.0. Le succès de ce navigateur contribue à l'adoption rapide de JavaScript dans le développement web orienté client. Microsoft qui était à l'époque le seul concurrent a réagit en développant JScript, qu'il inclut ensuite dans Internet Explorer 3.0 en août 1996 pour la sortie de son navigateur.

Netscape soumet alors JavaScript à l'ECMA pour le faire standardiser. Les travaux débutent en novembre 1996, et se terminent en juin 1997 par l'adoption du nouveau standart ECMAScript. Les spécifications sont rédigées dans le document Standard ECMA-262.

ECMAScript existe à ce jour en 5 versions du standart ECMA-262.

1.3 JSON

1.3.1 Présentation

JSON est un format d'échange de données qui est un sous ensemble de la notation objet littérale en JavaScript. Il a beaucoup gagné en popularité ces derniers temps comme une alternative légère au format XML, en particulier dans les applications AJAX.

Pourquoi ça ?

En raison de la capacité de JavaScript pour l'analyse rapide de l'information à l'aide la fonction eval(). JSON ne nécessite pas cependant JavaScript, et il est possible de l'utiliser comme un format d'échange simple pour n'importe quel langage de script.

Voici un exemple de ce que JSON est :

'détails' :

'id' : 1,

'type' : 'mémoire',

'auteur' : 'Anthony T. Holdener III',

'titre' : 'JavaScript, un langage web ?',

'detail' :

'pages' : 120,

'extra' : 22,

'price' :

'eu' : 00.00,

'us' : 00.00

Voici à titre de comparaison sont équivalent en XML

<détails id="1" type="mémoire">

<auteur>BARASCUT Jérémy</auteur>

<title>Ajax : The Definitive Guide</title>

<detail>

<pages extra="20">960</pages>

<isbn>0596528388</isbn>

<price us="49.99" ca="49.99" />

</detail>

</details>

Certains développeurs pensent que JSON est une façon plus élégante d'écrire les données. D'autres apprécient sa simplicité. D'autres encore soutiennent qu'il est plus léger.

Peut importe, si JSON est si populaire c'est surtout grace à la technologie AJAX.

1.4 Ajax

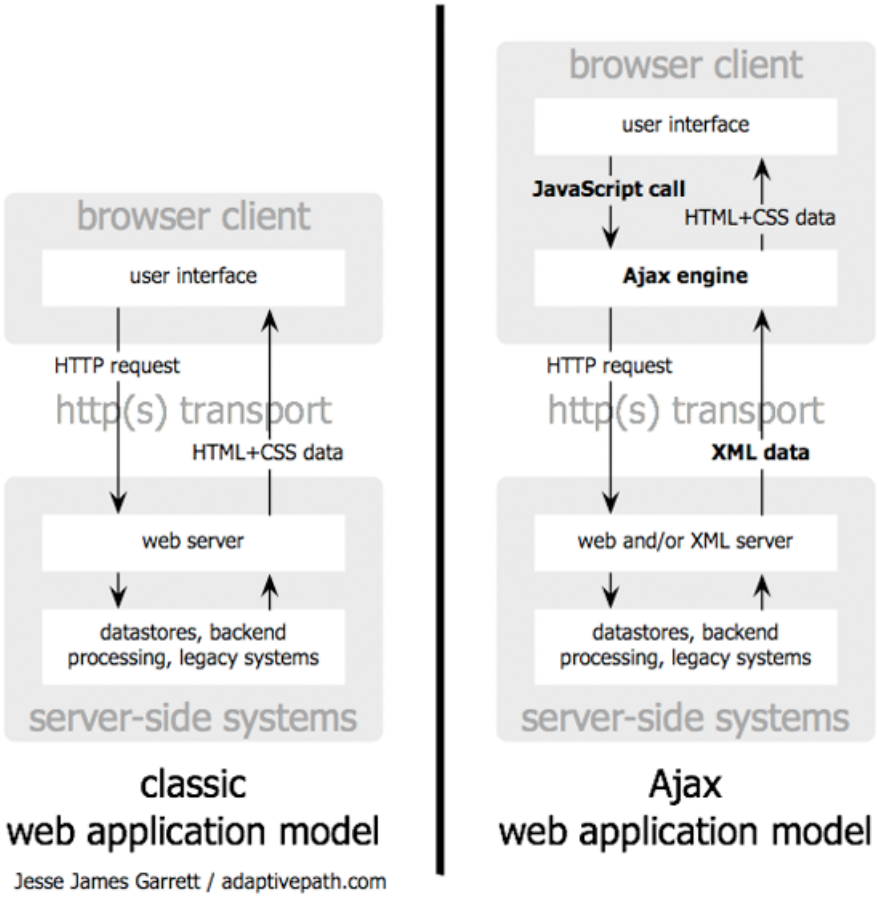
1.4.1 Présentation

AJAX (Asynchronous JavaScript And XML) est une architecture informatique permettant de construire des sites web dynamiques interactifs et des applications Web en se servant de différentes technologies présentées précédemment.

Ajax combine JavaScript, XML, CSS, le DOM et XMLHttpRequest afin d'améliorer la maniabilité et le confort d'utilisation des Applications Internet Riches (RIA).

Le terme Ajax a été introduit par Jesse James Garrett, le 18 février 2005, dans un article sur le site Web Adaptive Path. Ajax a créé une petite révolution dans les navigateurs.

En utilisant Ajax, le dialogue entre le navigateur et le serveur se déroule la plupart du temps de la manière suivante :



Les échanges de données entre le client et le serveur peuvent utiliser d'autres formats notamment le format JSON.

Ajax a permis à JavaScript de gagner en popularité et de ne plus être le vilain petit canard des langages de programmation. Ainsi avec AJAX JavaScript n'est plus un langage qui se contente d'afficher des popup intempestives ou de vérifier des formulaires.

Google a marqué les esprits avec Google Maps. Maps n'aurait jamais pu exister sans Ajax. L'utilisation de JavaScript par Google a permis à JavaScript de montrer son potentiel. Depuis JavaScript s'est désinhibé, et de véritables logiciels sont apparus dans nos navigateurs.

Google est un très gros consommateurs de JavaScript notamment avec Drive ou Gmail.

JavaScript est donc passé du petit langage d'agrément pour pages web à un langage de développement d'applications réseau supporté par tout les navigateurs quel que soit le système d'exploitation. Et le navigateur, pour un grand nombre d'utilisateurs, est la porte d'entrée de l'ordinateur et du réseau.

Avant AJAX : la page web est le support pour de petites applications JavaScript, le plus souvent d'agrément et facultatives pour utiliser le contenu.

Avec AJAX : JavaScript deviens le cœur du site. Il génère du contenu HTML dont il a la maîtrise. Avec un autre langage coté serveur, JavaScript coté client s'appuie sur le moteur graphique du navigateur pour générer l'interface de l'application, plus pratique que n'importe quel "toolkit".

Cette mutation a pris du temps. Le navigateur est passé d'un logiciel pour consulter des pages web à un logiciel permettant d'exécuter d'autres logiciels (en somme comme un système d'exploitation).

Ajax est donc plébiscité parce qu'il donne naissance à des applications inédites accessible à travers notre navigateur. Avec AJAX une application comme Google Earth pourrait aussi bien exister de façon autonome.

Malgré toutes ces avancées, JavaScript était encore confronté a des obstacles majeurs qui ont freiné son adoption.

JavaScript étant exécuté par le navigateur, ses performances de l'époque étaient extrêmement faibles. Actuellement la bataille des navigateurs se fait sur de meilleurs support tels que HTML5/CSS3 mais aussi sur les performances des moteurs JavaScript. L'arrivée de Google Chrome et des énormes besoins de Google en performances JavaScript ont poussé tout les acteurs du marché à améliorer les moteurs pour toujours plus de performances.

Le second problème de JavaScript est l'exécution du script par le moteur. En fonction du navigateur utilisé, le code JavaScript doit être différent, chaque éditeur étant libre d'intégrer JavaScript comme il le souhaite. Ainsi un code JavaScript peut fonctionner correctement sur Firefox ou Chrome mais pas sur Internet Explorer.

Le Javascript est aussi un langage qui nécessite des efforts importants et dont le développement en AJAX pur peut être extrêmement coûteux.

Afin de résoudre ce problème de développement, John Resig à sortie un framework révolu-

tionnant la façon d'écrire du code JavaScript. Le nom de ce framework : jQuery.

1.5 JQuery

1.5.1 Introduction

Au cours des dernières années, JavaScript a subi une transformation remarquable. A partir du moment où JavaScript a pu reléguer son image de langage jouet au second plan, il a alors pu s'affirmer comme l'un des langage de programmation les plus importants dans le monde.

Avec l'importance grandissante du développement basé sur Ajax, la montée des bibliothèques JavaScript augmentent et la stigmatisation entourant JavaScript a pratiquement disparu.

Il faut reconnaître que la bibliothèque la plus populaire et user-friendly, jQuery est en partie responsable de ce progrès.

jQuery est plus qu'un simple choix de débutant, il est en réalité utilisé par certaines des plus grandes organisations dans le monde, ajoutant de l'interactivité à des milliards de pages vues chaque mois. Google, Microsoft, Amazon, IBM, Twitter, NBC, Best Buy et Dell ne sont que quelques une des entreprises utilisant jQuery en production. Avec une très grande utilisation dans le monde, il n'est donc pas surprenant que jQuery évolue à grande vitesse.

jQuery continue de s'épanouir et les développeurs du monde entier contribuent aux corrections des bugs, des plugins et des travaux sur des projets connexes comme jQuery UI et QUnit. Ce regain d'activité assure que jQuery représente une option complète pour tout développeur cherchant à faire du développement JavaScript de catégorie mondiale.

Cela est vrai quelle que soit la philosophie ou la technique de développement utilisée : jQuery est utilisé et ce peu importe le langage coté serveur tel que : Java/Spring, PHP, .NET, Ruby on Rails, Python/Django par exemple.

jQuery est une bibliothèque JavaScript qui porte sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et a pour but de simplifier les commandes communes de JavaScript. Jquery se caractérise par un ensemble de fonctions qui permettent d'offrir une alternative à la programmation JavaScript de façon uniforme sur les navigateurs les plus courants et permet par exemple de manipuler aisément le DOM, de créer des animations etc... mais surtout de gagner du temps dans le développement des applications : « write less, do more ».

La librairie est sous licence GPL et MIT, et donc complètement réutilisable sur des travaux professionnels. De plus la librairie à l'avantage d'être compatible avec d'autres librairies JavaScript.

- C'est une bibliothèque puissante. Le système jQuery réalise toutes sortes de tâches impressionnantes pour simplifier l'écriture du code JavaScript.
- Elle est légère. Il faut inclure une référence à votre bibliothèque dans chaque fichier

qui l'utilise. La bibliothèque jQuery fait 26 ko (dans sa version compressée), une taille inférieure à certains fichiers image. Elle n'a donc aucun impact significatif sur la vitesse de chargement.

- Elle prend en charge un mécanisme de sélection flexible. jQuery simplifie et développe le mécanisme document.getElementById, essentiel pour la manipulation du DOM.
- Elle dispose d'un excellent support d'animation. Vous pouvez utiliser jQuery pour afficher et masquer, déplacer et glisser des éléments.
- Elle rend les requêtes AJAX évidentes. Vous allez être surpris par la facilité d'utiliser AJAX avec jQuery.
- Elle possède un mécanisme d'évènement amélioré. JavaScript dispose d'un support très limité pour les évènements. jQuery offre un outil très puissant pour ajouter un gestionnaire d'évènement à presque tous les éléments.
- Elle fournit un support multiplateforme. La bibliothèque jQuery tente de gérer des questions de compatibilité de navigateur pour vous. Ainsi, vous n'avez pas à vous soucier d'éventuels problèmes de navigateurs.
- Elle prend en charge les composants d'interface utilisateur. jQuery propose une bibliothèque d'interfaces utilisateur puissante qui compte des outils que HTML n'a pas, comme les contrôles glisser-déposer, les sliders et les calendriers.
- Elle est évolutive. jQuery possède une bibliothèque d'extension qui accepte tous types de fonctionnalités optionnelles, y compris de nouveaux composants et outils tels que l'intégration de son, les galeries d'images, les menus, etc.
- Elle introduit de nouvelles idées de programmation. jQuery est l'outil idéal pour découvrir des idées intéressantes comme la programmation fonctionnelle et les objets chaînables.
- Elle est gratuite et open-source. jQuery est disponible en licence open-source, ce qui signifie que son utilisation est gratuite et que vous pouvez la consulter et la modifier si vous le souhaitez.
- Elle est tout de même classique. Si vous décidez d'utiliser une autre bibliothèque AJAX, vous pourrez y exploiter les enseignements acquis dans jQuery.

1.5.2 Document Object Model (DOM)

Quand vous regardez un site web, vous voyez beaucoup d'éléments regroupés et assemblés pour former ce qui est en face de vous. Pour pouvoir accéder à ces éléments pour supprimer, ajouter et manipuler ces éléments, vous avez besoin d'une interface appropriée, d'une représen-

tation des éléments dans une page structuré et qui suit un ensemble de règle sur la manière de les modéliser. C'est ce qu'on appelle le DOM. Le DOM nous permet aussi de capturer dans le navigateur un événement comme lorsqu'un utilisateur clique sur un lien, soumet un formulaire, ou fait défiler la page.

Dans les premiers jours du web et des navigateurs, les normes en matière de mise en œuvre de JavaScript n'étaient pas efficaces. Cela a conduit les navigateurs à intégrer leur propre mise en œuvre de JavaScript créant ainsi des caractéristiques d'applications différentes et causant des problèmes aux développeurs d'applications. Ainsi pour développer une application JavaScript, il faut la coder pour chaque navigateur, ceux-ci ayant des implémentations différentes principalement entre Netscape et Internet Explorer.

Heureusement, les choses ont progressé, les navigateurs optent actuellement pour les mêmes normes et les choses se sont stabilisées. Toutefois, le niveau auquel les navigateurs supportent le DOM peut encore poser des problèmes aujourd'hui. En particulier avec les versions d'Internet Explorer 6, 7 et 8 qui ne supportent pas le DOM au même niveau que les navigateurs les plus modernes.

C'est une des raison pour lesquelles jQuery est si précieux : tout ce qu'il offre fonctionne aussi bien dans une version antérieure d'Internet Explorer que dans la dernière version de Google Chrome ou Mozilla Firefox.

Afin d'avoir des bases solides pour la suite, je vais prendre la peine de présenter la façon dont le DOM est mis en œuvre.

Quand une page est chargée, le navigateur génère une représentation de ce qui est sur la page, et pour chaque élément il génère un ou plusieurs nœuds qui le représentent. Lorsque vous travaillez avec JavaScript, le DOM (suivant les implémentation différentes des navigateurs) peut provoquer des problèmes et vous amener à passer beaucoup de temps sur des solutions de contournement, mais l'efficacité d'un framework comme jQuery permet de régler ce soucis. Lorsqu'un navigateur forme une représentation de la page en cours avec le DOM, chaque élément est un nœud. Prenons l'exemple d'un paragraphe avec du texte un l'intérieur, tels que :

Hello World

Ce n'est pas un, mais deux nœuds. Il y'a un nœud de type texte qui contient "Hello World" et un nœud de type élément qui est la paragraphe. Le nœud de type texte est un enfant du nœud de type élément parce qu'il réside en son sein. Dans une page type, il existe beaucoup de nœuds imbriqués. Un div avec deux paragraphes composé de texte à l'intérieur s'articule comme ceci :

div element node – paragraph element node —- text node – paragraph element node —-

text node

Les deux paragraphes de cette instance sont frères parce qu'ils ont le même nœud parent. Les paragraphes sont enfants de la div, mais les nœuds de type texte ne sont pas des nœuds enfants parce qu'ils ne sont pas descendants directs de l'élément div. Ils sont les nœuds enfants des nœuds de type paragraphe. Il existe trois principaux types de nœuds que vous devez connaître : élément, texte et attribut.

1.5.3 Les autres forces de jQuery

Pour finir avec jQuery j'ajouterai quelques autres détails qui permettrons d'expliquer comment jQuery à permis à JavaScript de prendre son envol.

- La documentation officielle est très fournie et de grande qualité ;
- La communauté qui gravite autour de jQuery est en perpétuelle expansion et elle fournit un support de qualité ;
- De nombreux acteurs de premier plan du Web (Microsoft, Google, Amazon, Twitter, Mozilla, etc.) utilisent jQuery ;
- Une multitude de plugins est disponible afin d'augmenter les possibilités de base de jQuery.

1.6 Restfull

1.6.1 Présenation

En 2000, Roy Fleding, l'un des principaux contributeurs aux protocoles HTTP et URI, a codifié l'architecture du Web dans sa thèse de doctorat intitulée "Architectural Styles and the Design of Network-Based Software Architectures."

Dan cette thèse, il a introduit une architecture connu sous le nom "Representational State Transfer (REST)". Ce modèle, en termes abstraits, décrit les fondations du World Wide Web (WWW). Les technologies qui composent ces fondations comprennent le Hypertext Transfer Protocol (HTTP), Uniform Ressource Identifier (URI), les langages de balisages tels que HTML et XML et formats adaptés au Web comme JSON.

REST est un style d'architecture pour les applications en réseau. Il se compose de plusieurs contraintes pour assurer la visibilité, la fiabilité, l'évolutivité etc.

Cela rend REST attrayant pour construire des applications client/serveur distribué et décentralisé dans l'infrastructure du Web. Le déploiement de services Web sur cette infrastructure vous permet de profiter d'un large éventail d'infrastructures existantes qui comprennent les serveurs web, les clients, les bibliothèques, les serveurs proxy, les caches, les firewalls, etc.

HTTP est un protocole de niveau applications qui définit des opérations de transfert entre les clients et les serveurs. Dans ce protocole, les méthodes telles que GET, POST, PUT et DELETE sont des opérations génériques agissant sur les ressources. Ce protocole élimine le besoin d'inventer des opérations spécifiques à l'application tels que CreateOrder, getStatus, updateStatus, etc. Les bénéfices que vous pouvez tirer de l'infrastructure HTTP dépénd de comment vous utiliser le protocole HTTP. Cependant, un certain nombre de techniques, y compris SOAP et certains frameworks web Ajax utilisent HTTP comme protocole pour transporter des messages.

1.7 MVC

1.7.1 Une structure

Le secret pour faire de grandes applications JavaScript n'est pas d'en faire une volumineuse. Il s'agit plutôt de dissocier votre application dans une série de composants relativement indépendants. Les développeurs font bien souvent l'erreur de créer des applications avec beaucoup d'interdépendances, avec d'énormes fichiers JavaScript linéaires générant une flopée de balises HTML. Ces applications sont difficiles à maintenir et à étendre, et par conséquent devraient être évitées à tout prix.

Faire un peu attention à la structure de l'application lorsque vous commencez à la construire peut faire une grande différence sur le résultat final. Ignorez toutes les idées préconçues que vous avez sur JavaScript et traitez le comme un langage orienté object. Utilisez les classes, l'héritage, les objets et les modèles de la même façon que vous le feriez avec un autre langage, comme Python ou Ruby. L'architecture est essentielle du coté serveur alors pourquoi ne pas faire la même chose coté client ? L'architecture retenue dans ce mémoire comme dans la plupart des frameworks présentés est l'architecture MVC, une architecture permettant de maintenir et étendre vos applications. Ce modèle s'applique particulièrement bien aux applications JavaScript.

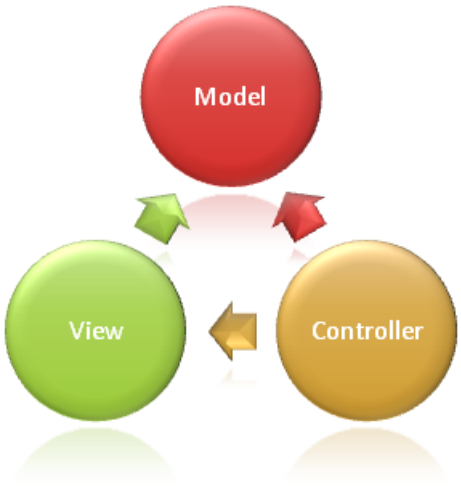
1.7.2 Qu'est ce que MVC ?

MVC est un modèle de conception qui découpe une application en trois parties : les données (Modèle), la couche de présentation (Vue), et la couche d'interaction de l'utilisation (Contrôleur).

En d'autres termes, le flux dévénement est comme ceci :

1. L'utilisateur interagit avec l'application
2. Le gestionnaire d'évenements déclenche le Controleur
3. Le Controleur demande les données du modèle, qui est envoyé à la Vue.
4. La Vue présente les données à l'utilisateur.

Ou en représentation graphique



Le modèle architectural MVC peut même être mis en oeuvre sans bibliothèques ni framework. La clé est de répartir les responsabilités des composants MVC en sections de code définis, en les gardant découpés. Cela permet de rendre indépendant le développement, les tests et la maintenances de chaque composants.

1.8 Conclusion

1.8.1 conclusion

blabla

Chapitre 2

Client

2.1 Introduction

2.2 Ember

2.2.1 Présentation

Ember.js est un framework frontend MVC écrit en JavaScript qui s'éxécute dans le navigateur. Il est destiné aux développeurs qui cherchent à construire des applications web avec autant de puissance et de fonctionnalités que les applications natives concurrentes.

Ember.js a été créé à partir de concepts introduits par les applications natives telles que Cocoa. Ember.js vous aide à créer une grand expérience pour l'utilisateur. Il vous aidera à organiser toutes les interactions directes qu'un utilisateur peut effectuer sur votre site/applications.

Quand vous pensez que votre code JavaScript va devenir complexe, lorsque le code deviens long et compliquer, lorsque le refactoring du code va être compliquer alors Ember.js vous évite cela. Grave à ça structure MVC (Modèle-vue-controleur) il est facile de faire des modifications ou refactoring de code de n'importe quelle partie de votre code. Il vous permettra également d'adhérer aux principes du DRY (Don't Repeat Yourself). Le modèle associé aux vues et aux contrôleurs exécute du CRUD (Create, Read, Update, Delete) pour l'informer d'un changement d'état. Il peut envoyer également une demande à la vue pour modifier la façon dont la vue représente le modèle actuel. La vue recevra alors des informations du modèle pour créer un rendu graphique.

Ember.js découpe les zones problématiques de votre interface vous permettant de vous concentrer sur une zone à la fois sans le souci d'affecter d'autre parties de votre application. Pour vous donner un exemple de certains domaines de Ember.js, jeter un oeil à la liste suivante :

- Navigation : le routeur d'Ember s'occupe de la navigation de l'application
- Mise à jour automatique des modèles : La vue d'Ember sont automatiquement mise à jour lorsqu'il y'a une modification. Ce qui signifie qu'ember mettre à jour automatiquement les données sous-jacentes si il y'a un changement.
- Manipulaton des données : Chaque objet créé sera un objet Ember, héritant ainsi de toutes les méthodes Ember.object.
- Comportement asynchrone : Les liaisons et les propriété utilisé avec Ember aides à gérer l'asynchrone.

Ember.js est plus qu'une bibliothèque. Ember.js prévoit de construire une bonne partie du frontend autour de méthodes et d'une architecture carré, créant une solide architecture une fois que vous avez terminé votre application. C'est la principale différence entre Ember et un framework comme Angular.js. Augular s'autorise à être incorporé dans une application existante

alors que Ember doit être utilisé avec sa propre architecture et sa philosophie. Backbone.js serait un autre exemple de bibliothèque pouvant être facilement inséré dans des projets JavaScript existants.

Ember.js est un excellent framework pour la gestion complexe des interactions réalisées par les utilisateurs de votre application. Si vous pensez que Ember.js est un framework difficile à apprendre, c'est totalement faux. La seule difficulté pour les développeurs réside dans la compréhension des concepts que Ember.js cherche à mettre en oeuvre. Ember.js favorise convention plutôt que configuration.

2.3 Backbone.js

2.3.1 Introduction

Backbone.js est un framework JavaScript permettant de structurer une application web non pas comme une suite d'instruction jQuery, mais comme un ensemble de vues autonomes les unes des autres.

Backbone est un bon choix pour les applications dites 'single page application".

Single page application (Application Web Monopage) c'est à dire pour simplifier, une page principale avec un nombre important d'interactions utilisateur. Plutôt que d"avoir une navigation par page (le serveur envoie une page à chaque URL), la navigation, l'envoi de formulaire et toutes les actions classiques se gèrent en javascript.

Avant, lorsque l'on souhaitait faire ce genre de choses, on écrivait son propre JavaScript, sans aucune convention, à grand coup d'AJAX et de script Jquery. On se retrouvais vite avec un code complexe à maintenir et à tester.

Backbone.js permet de cadrer cela en définissant modèle, vue et collections afin de structurer notre code. On va pouvoir définir des événements sur des changements de valeurs de nos modèles et ainsi rafraîchir automatiquement nos vues.

Backbone.js possède son routeur. On peut faire correspondre des actions à des URL.

Il se veut non-contraignant par rapport à ses rivaux, ce qui lui coûte d'être présenté souvent comme moins complet. Son point fort est d'être facilement utilisable avec d'autres libraires ou frameworks.

Par contre Backbone.js est uniquement une couche client. Il ne gère pas la persistance des données sur un serveur. Heureusement cela se gère très bien en utilisant l'architecture REST. Il suffit d'indiquer à nos modèles une URL et d'utiliser une API JSON Restful pour être capable de réaliser du CRUD très facilement.

2.3.2 Dépendances

Backbone.js est basé sur la blirairie Undercore.js. Cette dernière propose des fonctionnalités de manipulation d'objets, de collections et de tableaux assez poussées. Comme toute librairie populaire, elle se veut cross-browser et par héritage, Backbone.js l'est aussi.

Pour tout ce qui est manipulation DOM, plutôt que réinventer la roue, l'équipe en charge de Backbone.JS a décidé de sous traiter cette tâche à jQuery. Il est possible de remplacer jQuery, par exemple avec Zepto, du moment qu'elle respecte l'api jQuery-compatible.

2.3.3 Composants

Backbone.js fournit des composants logiciels pouvant être utilisés librement, que ce soit avec les autres composants Backbone.js ou avec une autre librairie. Les trois plus importants sont le Model, La Vue et la Collection.

Les composants Router et Sync fournissent des fonctionnalités très intéressantes.

2.3.4 Les données

La classe Backbone.Model est utilisé pour gérer du contenu sous la forme d'un objet JavaScript, en l'encapsulant et en proposant des méthodes accesseurs.

La classe Backbone.Collection permet de manipuler des collections de Model. On retrouve les méthodes habituelles : push, pop, shift, unshift, add, remove, get, sort ou encore length. On retrouve également les fonctions provenant de Underscore.js

Ces deux classes permettent de manipuler des données. Mais Sync permet de le faire plus simplement.

Sync est le composant permettant de synchroniser les objets à travers une API du type RESTful JSON. Pour cela, il suffit de lier les objets Model et Collection à une ressource grâce à l'attribut url.

2.3.5 La vue

Toutes les manipulations DOM se font à travers le plugin jQuery qui est de loin, le mieux armé pour ces opérations.

Chaque objet Backbone.View est lié à un nœud DOM (el) et pourra le générer à nouveau, à n'importe quel moment. Le but est alors de découper le document en une multitude de vues que l'on pourra régénérer à souhait et individuellement.

2.3.6 Asynchronous Module Definition API

Il est possible d'utiliser Backbone.JS avec un AMD Loader comme RequireJS ou Curl. Ni Backbone.JS, ni Underscore ne supportent officiellement AMD. Toutefois, il est possible d'utiliser un fork implémentant de cette API. On trouve sur Github des projets templates (boilerplate) qui facilitent la mise en place d'un environnement couplant RequireJS et Backbone.

2.3.7 Plugins

Backbone.JS se veut léger : il ne fournit que des composants essentiels. Un des gros manquements de Backbone.JS par rapport à ses concurrents est l'absence de la fonctionnalité de DataBinding. Mais qu'à cela ne tienne, de nombreux plugins sont présents en libre accès sur GitHub, pour implémenter des fonctionnalités.

2.4 AngularJS

2.4.1 Introduction

AngularJs est un framework structurel pour les applications web dynamiques. Il est écrit en JavaScript et est sous Licence MIT.

C'est un framework open-source au même titre que MooTools, Prototype, Dojo ou Jquery.

AngularJs est développé par Google et sa communauté.

AngularJS permet d'utiliser le HTML comme langage de template et vous permet d'étendre la syntaxe du HTML pour exprimer les composants de votre application claire et succincte.

Il a pour but de simplifier l'écriture du JavaScript en simplifiant la syntaxe et en comblant les faiblesses de ce langage en lui ajoutant de nouvelles fonctionnalités. Le but d'AngularJs est de faciliter la réalisation d'applications web monopages.

AngularJs peut être utiliser avec ou sans jQuery pour la manipulation du DOM.

Le framework adapte et étend le HTML traditionnel pour servir le contenu dynamique de façon améliorée grâce à un data-binding bidirectionnel qui permet la synchronisation automatique des modèles et des vues. En conséquence AngularJS minore l'importance des manipulations DOM et améliore la testabilité du code.

AngularJs essaie d'être une solution complète pour créer une application web. Il est livré avec tout ce que vous avez besoin pour construire une application CRUD de façon cohérente : liaison des données, directives de bases pour les templates, la validation du formulaire, le routage, le deep-linking, la réutilisation de composants et l'injection des dépendances. AngularJs permet l'écriture de tests unitaire, cas de test, mocks etc.

2.4.2 Objectifs de conception du framework

- Découper les manipulations du DOM de la logique métier. Cela améliore la testabilité du code.
- Guider les développeurs pendant toute la durée de la construction d'une application : de la conception de l'interface utilisateur, en passant par l'écriture de la logique métier, jusqu'au test de l'application.
- Considérer le test d'une application aussi important que l'écriture de l'application elle-même. La difficulté de la phase de test est considérablement impactée par la façon dont le code est structuré.
- Découper les côtés client et serveur d'une application. Cela permet au développement

logiciel des côtés client et serveur de progresser en parallèle, et permet la réutilisation du code de chaque côtés.

- Rendre les tâches faciles évidentes et les tâches difficiles possibles.

AngularJS est un framework MVC et encourage le couplage faible entre la présentation, les données, et les composants métiers. En utilisant l'injection de dépendances, AngularJS apporte aux applications web coté client les services traditionnellement apportés coté serveur, comme les contrôleurs de vues. En conséquence, une bonne partie du fardeau supporté par le back-end est supprimée, ce qui conduit à des applications web beaucoup plus légères et maintenables.

2.5 CanJS

2.5.1 Présentation

Le framework CanJS est issu du framework Javascript MVC. Il s'agit en fait de l'extraction du noyau et il est décomposé en modules pour coller exactement aux besoins de l'application sans embarquer des Kilos octects de fonctionnalités inutiles. La nouvelle version de Javascript MVC se basera sur CanJS pour assurer l'inter-compatibilité des deux frameworks.

Le grand intérêt de CanJS est de fournir des objets « Model » pour stocker nos données, y associer des fonctions de mise à jour coté serveur personnalisables et des événements auxquels attacher des actions. Mieux, il gère automatiquement, grâce à son système de templates, la mise à jour de toutes les vues associées à un modèle lorsque ce dernier est modifié.

CanJS est composé de :

- can.Construct - hérite des fonctions du constructeur
- can.Observe - clé valeur contraignante
- can.Model - permet de connecter une interface JSON-REST
- can.View - chargement du modèle, mise en cache
- can.EJS - modèles contraignant direct
- can.Control - liaisons d'évnements déclaratif
- can.Route - support du bouton précédent et des onglets

Il prend également en charge un riche ensemble d'extensions et plugins.

2.5.2 Caractéristiques vs Poids

En plus de jQuery, CanJS fait 8.5kb. A titre d'exemple voici quelques autres bibliothèques MVC (compressé avec gzip) :

- 8.97kb pour Backbone (avec Underscore.js)
- 24kb pour AngularJS
- 13kb pour Knockout
- 37kb pour Ember
- 15kb pour Batman

Pour être juste, le poids est trompeur, car chaque bibliothèque possède un ensemble différent de fonctionnalités. Cependant, CanJS fournit tout ce dont vous avec besoin pour créer une applications coté client riche, avec la bibliothèque la plus légère parmi celle comparé. Par comparaison Backbone.js est livré le micro template Underscore.js, mais ceux-ci ne se compare

pas à la puissance de EJS. La plupart des applications réseau de base comprennent également un moteur de templates qui ajoute du poids à la bibliothèque.

2.5.3 Facilité d'utilisation

CanJS bénéficie d'une courbe d'apprentissage plus facile que n'importe quelles autres biblothèques. CanJS bénéficie d'une bonne documentation. Il permet de se faire la main avec la page de présentation, puis de plonger plus profondément en lisant chaque méthodes et classe dans la page de documentation. CanJS permet de voir comment les applications sont construites en parcourant les exemples, en lisant les annotations et en regardant les jeux de tests. Un blog permet d'en savoir toujours plus sur CanJS, et il est possible de poser des questions sur le forum. Twitter ou d'obtenir un support premium, de formations ou de conseil.

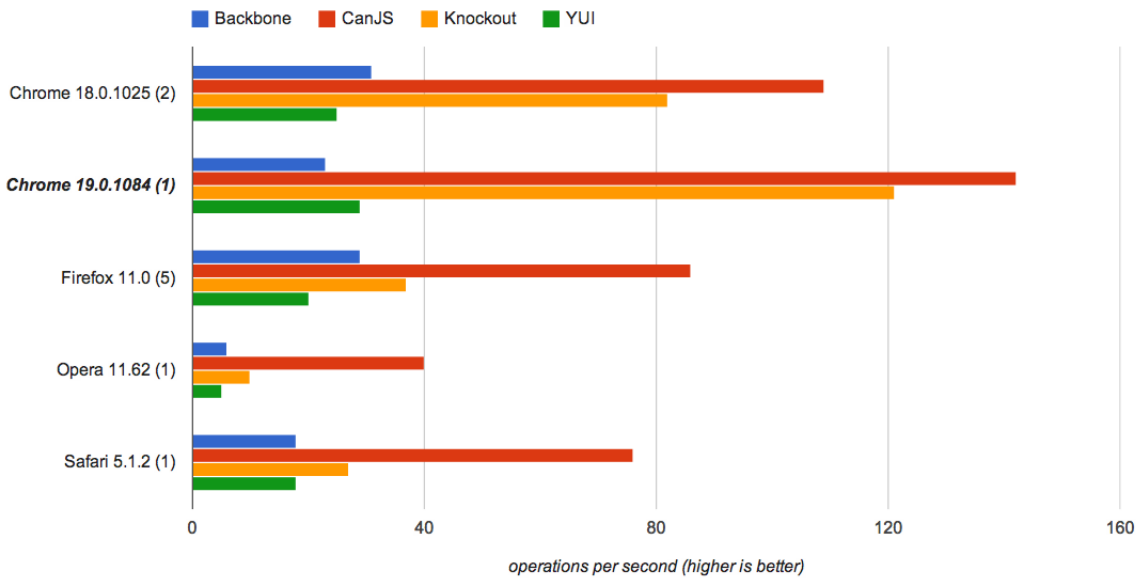
2.5.4 Prévention des fuites mémoire

CanJS empêche les fuites de mémoire qu'un développeur ne connait problablement pas et a dans son application. Les applications JavaScript ont communément des fuites mémoire de deux sources : les gestionnaires d'évenements et les objets inutilisées. Autant dire qu'il s'agit d'un problème critique pour les client MVC. CanJS gère ces fuites automatiquement ce qui rend presque impossible une fuite mémoire dans l'application.

2.5.5 Performance

CanJS est optimisé pour la performance dans des domaines clés. Par exemple la liaison en temps réel est optimisé pour les performances en modifiant directement et exactement ce qui doit être mis à jour, plutôt que le modèle en entier.

Voici un schéma mesurant les performance de CanJS en fonction d'autres frameworks.



2.5.6 Support de librairie

CanJS permet d'intègrer cinq des bibliothèques les plus couramment utilisés pour l'accès au DOM :

Cela donne la possibilité de choisir la bibliothèque préférée ou passer même facilement par une autre bibliothèque sans avoir a réécrire la couche MVC de votre application.

Il y'a une intégration profonde avec chaque bibliothèque afin de permettre une utilisation complète peut importe la librairie choisie.

2.6 Conclusion

Au vue de la présentation des différents framework client, il peut être extrêmement difficile de choisir un framework MVC JavaScript pour développer notre client web. Il y'a tellement de facteurs à considérer et tellement d'options à étudier que la sélection d'un framework peut être hardu. Pour avoir une idée de toutes les alternatives possibles il y'a le site TodoMVC qui permet de comparer l'utilisation de plusieurs framework codé afin de réaliser une liste de choses à faire (TODO)

Je vous ai présenter 4 frameworks : Angulare, BackBone, CanJS et Ember. J'ai donc décidé de réaliser une comparaison pour ce mémoire pour aider à la décision du framework à utiliser. Je vais passer par plusieurs facteurs que vous pourriez envisager pour en choisir un.

Pour chaque facteur, j'ai attribué une note comprise entre 1 et 5. Où 1 est pauvre et 5 est riche. J'ai essayé d'être impartial dans mon mémoire, mais bien sûr mon objectivité est fortement compromise parce que mes sources sont uniquement basé sur les exemples du site TodoMVC.

2.6.1 Fonctionnalités

Il y'a des éléments vraiment important qu'un framework doit fournir pour avoir les bases nécéssaire à la réalisation d'une application utiles. Fait elle une liaison entre le modèle et la vue ? une liaison bidirectionnelle entre le modèle et la vue ? l'utilisation de filtres ? les propriétés sont être calculé ? Y'a t'il une validation des formulaires ? etc. Cela peut être une très longue liste. Voici une comparaison de ce que je considère comme des caractéristiques vraiment importantes dans un framework MVC.

Fonctionnalités	Angular	Backbone	CanJS	Ember
Observables	O	O	O	O
Routage	O	O	O	O
Liaison modèle-vue	O	-	O	O
Liaison modèle-vue bidirectionnelle	O	-	-	O
Vue partielle	O	-	O	O
Vue filtrée	O	-	O	O

Observables : Objets qui peuvent être observés lors des changements.

Routage : Ecoute les changements d'URL du navigateur et permet d'agir en conséquence.

Liaison modèle-vue : Utilisation d'objets observables dans les vues, les vues ayant un rafraîchissement automatique lorsque le changement de l'objet est observable.

Liaison bidirectionnelle entre le modèle et la vue : Permet à la vue de pousser la modification de l'objet observation automatiquement, par exemple, un formulaire de saisie.

Vue partielle : Vues qui comprennent d'autres vue

Vue filtrée : Avec des vues qui affichent des objets filtrés par un cetain nombre de critères.

Résultat

Sur la base de ces caractéristiques les scores sont les suivants :

Angular	Backbone	CanJS	Ember
5	2	4	5

Il est important de noter que Backbone peut faire la plupart de ces choses avec beaucoup d'écriture de code manuel ou à l'aide de plug-in. Mais je n'ai considéré que les fonctions disponibles de base dans les framework.

2.6.2 Flexibilité

Il existe des centaines de plugin et bibliothèques qui font des choses spécilalisés. Ils le font habituellement mieux que ce qui vient du framework. Il est donc important d'être en mesure de pouvoir intégrer ces bibliothèques dans le framework MVC choisi.

Backbone est le framework le plus souple, car il est celui qui à le moins de convention et de normes. Vous êtes obligé de prendre beaucoup de décision lors du développement.

CanJS est presque aussi souple que Backbone car il permet d'intégrer facilement d'autres bibliothèques avec un minimum d'effort. Avec CanJS il est possible également d'utiliser un moteur de rendu totalement différent. Par exemple l'utilisation du moteur de rendu "Rivets" ne pose aucun problème. Mais il est vivement recommandé d'utiliser le moteur de rendu du framework.

Ember et **Angular** sont aussi des framework souple dans une certaine mesure, mais il se peut que l'utilisation d'autres biliothèques soit difficile à mettre en place en complément du framework.

Résultat

Angular	Backbone	CanJS	Ember
3	5	4	3

2.6.3 La documentation et la courbe d'apprentissage

Angular

Angular est un framework qui lorsqu'on le découvre procure un effet "wow". Il peut faire des choses étonnantes, comme la liaison bidirectionnelles, sans avoir à apprendre beaucoup de choses. Il semble assez facile à première vue. Mais une fois les bases appris, la courbe d'apprentissage devient à partir de là abrupte. Il s'agit d'un framework complexe avec beaucoup de particularités. La lecture de la documentation n'est pas facile car il y'a beaucoup de jardon spécifique à Angular et un manque important d'exemple.

Backbone

Les bases de Backbone sont assez facile à apprendre. Mais une fois appris, l'on se rend compte que la documentation ne donne pas assez de conseil sur la façon de structurer son code. Il est alors obligatoire de regarder ou lire des tutoriels pour apprendre certaines des meilleures pratiques de backbone. Il se peut également que l'apprentissage d'une autre bibliothèque soit obligatoire pour faire avancer votre développement (par exemple Marionette ou Thorax). Au final, l'apprentissage de Backbone n'est pas plus facile.

CanJS

CanJS en comparaison est le plus facile à apprendre. Juste en lisant le site internet du framework (http ://canjs.us), l'essentiel de ce que l'on a besoin de savoir est présenté pour être productif. Il y'a bien sûr encore beaucoup à apprendre, mais la nécessité de recourir à l'aide survint à de rares occasions (tutoriels, forum, irc).

Ember

Ember a aussi une courbe d'aprentissage abrupte comme Angular, mais l'apprentissage de Ember est un peut plus facile que Angular. Il nécessite par contre un investissement important au début pour acquérir les bases. Angular en revanche permet de faire des choses incroyables sans en avoir trop à apprendre. Ember manque de cet effet "wow"

Résultat

Angular	Backbone	CanJS	Ember
2	4	5	3

2.6.4 La productivité des développeurs

Après avoirr appris correctement le framework ce qui importe le plus est de savoir comment le framework permet d'avoir une productivité performante en mixant conventions, magie et conception la plus rapide possible.

Angular

Une fois que l'on maîtrise Angular correctement, l'on devient très productif. Il n'y a aucun doute à ce sujet. Il n'a cependant pas le score le plus élevé parce que Ember franchi une étape supplémentaire dans cette catégorie.

Backbone

Backbone oblige à écrire beaucoup de code réutilisable, ce qui est pour certains projet totalement inutile. C'est un avis de certain développeur qui pense que c'est une contrainte importante allant à l'encontre de la productivité des développeurs.

CanJS

CanJS ne brille pas mais ne déçoit pas non plus dans ce domaine. En raison de la faible courbe d'apprentissage vous pouvez être très productif avec lui très tôt.

Ember

Ember brille particulièrement ici. Parce qu'il est plein de conventions, il fait beaucoup de choses à votre place. Tout ce qu'il reste à faire est d'apprendre et appliquer les conventions et Ember s'occupe du reste.

Résultat

Angular	Backbone	CanJS	Ember
4	2	4	5

2.6.5 Communauté

Est-il facile de trouver de l'aide, des tutoriels et des experts ?

La communauté **Backbone** est énorme, il n'y a aucun doute à ce sujet. L'on trouve des dizaines de tutoriaux sur Backbone et une communauté très active sur StackOverflow et IRC.

Les communautés **Angular** et **Ember** sont assez grandes aussi. Il y'a également beaucoup de tutoriel et d'activité sur StackOverflow et IRC, mais pas autant que sur Backbone.

La communauté **CanJS** est assez faible en comparaison, mais heureusement, très active et utile. La taille de la communauté n'est par conséquent pas un frein important.

Résultat

Angular	Backbone	CanJS	Ember
4	5	3	4

2.6.6 Ecosystème

Y'a t'il un écosystème de plugin et de bibliothèques ?

Là encore **Backbone** surpasse les autres. Il y'a des tonnes de plugin disponible. L'écosystème d'**Angular** devient assez interessant avec des choses comme l'interface utilisateur. L'écosystème d'**Ember** est moins développé mais il devrait s'améliorer en raison de sa popularité importante.**CanJS** en comparaison à le plus petit écosystème.

Résultat

Angular	Backbone	CanJS	Ember
4	5	2	4

2.6.7 Poid

Cela peut être un facteur important, surtout pour du développement mobile.

Poid des frameworks seul (sans aucune dépendances)

Angular	Backbone	CanJS	Ember
80ko	61ko	57ko	269ko

Backbone est le plus léger et les développeurs soulignent souvent ce fait. Mais ce n'est pas le cas si l'on compte les dépendances.

Poids des frameworks avec dépendances

Angular est le seul framework du groupe qui ne nécessite pas de bibliothèques supplémentaires pour travailler.

Tout les autres ont besoins de dépendances pour fonctionner.

Backbone a au moins besoin de Underscore et Zepto. Il est possible d'utiliser les mini-templates de Underscore pour le rendu des vues, mais la plupart du temps, l'utilisation d'un véritable moteur de template est appréciable comme Moustache par exemple. Au final le poids est de 61ko.

CanJS a besoin d'au moins Zepto. 57ko.

Ember a besoin de jQuery et de Handlebars. 269ko.

Résultat

Angular	Backbone	CanJS	Ember
4	5	5	2

2.6.8 Performance

La performance n'est pas forcément un facteur déterminant dans le choix d'un framework car ils sont tous assez performant pour la plupart des cas d'utilisation. Mais cela dépend bien sur de l'utilisation du framework. Si il est utilisé pour la création d'un jeu alors les performances deviennent importantes.

Il y'a sur internet de nombreux tests de performance avec ces frameworks. Par contre la fiabilité des tests ne sont pas une valeur sure car il est impossible de vérifier si ces tests ont été réalisé de dans bonne condition, ni avec un code de qualité.

Cependant, au vu des tests, il semblerait que **CanJS** ait un avantage quand il s'agit de performance, spécialement dans l'affichage de la vue. D'autre part, **Angular** est le moins performant.

Résultat

Angular	Backbone	CanJS	Ember
3	4	5	4

2.6.9 Maturité

Est-ce un framework mature, est-ce prouvé en production, est-ce que de nombreux application web l'utilise ?

Backbone a une tonne de sites construit avec lui. Sa base de code n'a pas eu de grands changements dans les deux dernières années ce qui est une bonne chose du point de vue de la maturité.

Bien que **Ember** n'est pas nouveau, il a eu de profond changements en cours de route, transformant tout pour attendre une forme stable dans les trois derniers mois. Par conséquent il ne s'agit pas d'un framework mature.

Angular semble plus stable et éprouvé que Ember. Mais pas autant que Backbone.

CanJS est une extraction de JavaScriptMVC, une bibliothèque qui a été lancé en 2008 et possède un important retour d'expériences et d'application construite avec.

Résultat

Angular	Backbone	CanJS	Ember
4	5	4	3

2.6.10 Sécurité des fuites mémoire

Il s'agit d'une considération importante si l'on construit des applications en single page, qui sont destinés à rester ouvert pendant de longue période. Si l'on ne veut pas que l'application est une fuite mémoire, cela peut être un réel problème. Malheureusement les fuites peuvent se faire assez facilement, surtout si l'on crée soit même des écoutes pour les évènements DOM.

Angular, **CanJS** et **Ember** traiteront de façon efficace et aussi longtemps les fuites mémoire du moment que l'on suit les meilleures pratiques. **Backbone** en revanche oblige à faire ce travail manuellement dans une méthode de désallocation mémoire.

Résultat

Angular	Backbone	CanJS	Ember
5	3	5	5

2.6.11 Testabilité

Est-il facile de tester le code ?

Les clés pour avoir un grand code testable sont la modularité (avoir de petits morceaux qui peuvent être testés séparément) et l'injection de dépendance (être capable de changer les dépendances dans les tests).

Il est possible de le faire avec la plupart des framewoks du moment que l'on utilise les bons modèles, mais ce n'est pas facile et ils obligent le changement des habitudes d'applications.

La modularité et l'injection de dépendance sont des caractéristiques fortes de **Angular**, en décourageant activement de faire les choses d'une autre manière que Angular l'a prévu. Cela

conduit généralement à un code plus facile à tester. Pour cette raison, Angular à un avantage dans ce domaine.

Résultat

Angular	Backbone	CanJS	Ember
5	4	4	4

2.6.12 Goût personnel

C'est probablement l'un des facteurs les plus importants lors du choix d'un framework. Mais il n'est pas possible de noter ce sujet et reste à l'entière appréciation du développeur.

2.6.13 Total

En rassemblant les résultats on arrive à un palmarès. Ceci n'est qu'une opinion forgé sur TodoMVC et de lecture de sites internet. Il ne représente pas un résultat fiable et sert juste à dégager un point de vue.

Total général	Angular	Backbone	CanJS	Ember
Fonctionnalités	5	2	4	4
Flexibilité	3	5	4	3
La documentation et la courbe d'apprentissage	2	4	5	3
La productivité des développeurs	4	2	4	5
Communauté	4	5	3	4
Écosystème	4	5	2	4
Poid	4	5	5	2
Performance	3	4	5	4
Maturité	4	5	4	3
Sécurité des fuites mémoire	5	3	5	5
Testabilité	5	4	4	4
Total	43	44	45	42

Si le développeur accorde autant de poid à chaque facteur, il s'agit d'un concours serré, ll n'y a ni gagnants ni perdants. Donc je suppose que tout revient au goût personnel du développeur ou à l'affection de poids différents pour chaque catégories.

Chapitre 3

Smartphone

Introduction

Dans votre poche se trouve un appareil qui a changé la vie de milliards de personnes partout dans le monde. Le troisième écran personnel (après la télévision et l'ordinateur) est le plus personnelle de tous et est la priorité clés des entreprises de cette décennies afin de proposer de nouveaux services.

Cependant le développement mobile est une activité plus difficile que le développement sur Pc. Les plateformes mobiles sont très fragmentés et les développeurs doivent travailler avec un minimum de ressources. Heureusement le web-mobile permet de traiter plus facilement cette fragmentation en permettant aux développeurs de créer des applications qui s'éxecutent sur plus de plateformes qu'en natif.

Les statistiques les plus récentes au moment de la rédaction de ce mémoire indique que Android est en tête du classement des smartphones avec environ de 70% de toutes les ventes au dernier trimestres 2012, iOS est à environ 20% dans la même période. Blackberry un très grand nom dans le monde des smartphone ainsi que Windows Phone, Bada et Symbian avec d'autre plateformes plus ou moins connus, se partagent le reste des pourcentages restant.

Global Smartphone OS Shipments (Millions of Units)	Q4 '11	2011	Q4 '12	2012
Android	80.6	238.9	152.1	479.0
Apple iOS	37.0	93.0	47.8	135.8
Others	39.4	158.6	17.1	85.3
Total	157.0	490.5	217.0	700.1

Global Smartphone OS Marketshare %	Q4 '11	2011	Q4 '12	2012
Android	51.3%	48.7%	70.1%	68.4%
Apple iOS	23.6%	19.0%	22.0%	19.4%
Others	25.1%	32.3%	7.9%	12.2%
Total	100.0%	100.0%	100.0%	100.0%

Total Growth Year-over-Year %	55.9%	63.8%	38.2%	42.7%

Ces chiffres montrent clairement que le marché des smartphones est très différent du marché du marché des PC, il n'y a pas vraiment de gagnant et les societés voulant profiter ce nouveau canal de communication ont à faire d'importants investissement afin d'être présents dans autant de poches que possible. Beaucoup d'applications doivent être écrite dans au moins deux ou trois plateformes (généralement iOS, Android et Blackberry) pour atteindre une tranche non négligeable du marché.

Actuellement les smartphone ont conquéri le marché des téléphones portables ces dernières années.

Beaucoup de choses ont changé depuis 2007, mais comme comme pour son homologue de bureau, le Web apparaît comme la solution multiplateforme la plus importante à la dispositions des développeurs d'aujourd'hui.

3.0.14 La croissance du Web mobiles

Une des avancées de cette nouvelle génération d'appareils mobiles est la disponibilité d'utiliser pleinement un véritable navigateur Web mobiles, compatible avec la plupart des normes actuelles telles que HTML5, CSS, JavaScript et de nombreux autres standard.

Beaucoup d'entre nous se souviennent regarder Steve Jobs présentant les capacités du navigateur Safari mobile dans le premier iPhone, tout en reconnaissant qu'une nouvelle ère avait commencé précisément ce jour-là.

Les navigateurs mobiles n'étaient pas seulement aussi capable que leurs homologues de bureau, ils étaient meilleurs que les pc , ils étaient rapide et étaient entièrement conformes aux normes.

La montée en puissance de l'internet mobile a apporté de nouvelles possibilités, en particulier dans les pays à faibles pénétration technologique comme l'Amérique latine ou l'Afrique. Les smartphone apparaissent alors comme un bien beaucoup moins cher pour accéder aux informations et services en ligne.

Exemple, en 2010, plus de 30% de tout les accès web à partir de l'Afrique était fait à partir d'un smartphone.

Dans le monde, on estime que plus de 50% de tout les requêtes web viendront d'appareils mobile d'ici 2015.

3.0.15 Nouveaux paradigmes

Tout cela représente un énorme changement dans nos habitudes de développer des logiciels.

Un changement radical indiquant que le web mobile est en train de devenir le nouveau canal principal de la présence du web. L'utilisation de la bande passante à partir d'un pc va être inférieur à celle du mobile.

Mais cette nouvelle perspective pose quelques questions :

- Combien de plateforme je dois tester pour mon site web ?
- Dois-je me soucier des téléphones mobiles bas de gamme ?
- Quel bibliothèques puis-je utiliser pour accélérer mon développement ?
- Quel est le niveau de support des standard dans les principaux navigateurs mobiles.

Pour ce faire, nous allons étudier les technologies suivantes, qui sont actuellement très prometteuses et qui ont une feuille de route très interessantes

- PhoneGap
- Sencha Touch
- Jquery Mobile

Il y'a bien sur d'autres technologies intéressantes mais je vais me limiter à l'étude de ces 3 frameworks.

3.1 Phonegap

3.1.1 Introduction

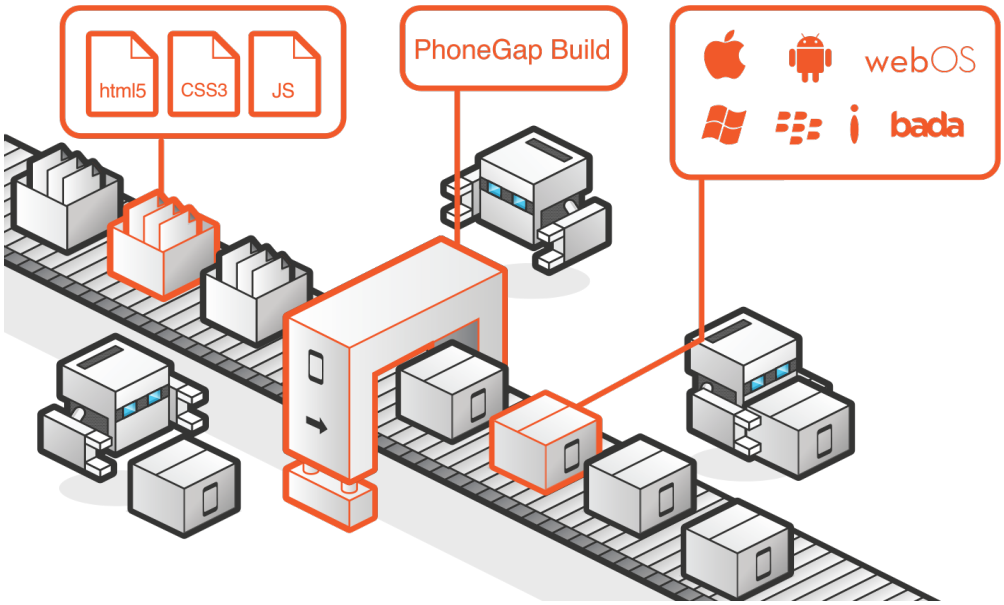
Développer des applications pour les appareils mobiles peut être fait en utilisant différentes approches et langages. La plupart des applications sont développées en native, ce qui signifie qu'ils sont développés en Java, Objective C, ou un autre langage compris par le SDK disponible dans l'appareil.

Alors que le développement natif permet une plus grande flexibilité et de meilleure performance, le problème se pose lorsque vous souhaitez déplacer une application d'une plateforme à l'autre. Tout d'un coup vous écrivez l'application presque à partir de zéro. Alors si vous voulez passer à une autre plateforme la même chose se produit.

Il doit y'avoir une meilleure façon de faire !

Toutes les plateformes mobiles actuels supportent l'utilisation d'application web. Ce sont des applications codés entièrement en HTML, JavaScript. Pour des applications simples, ou pour des applications qui n'ont pas besoin d'interagir avec les cappacités de l'appareil, cela fonctionne très bien. Mais au moment où vous avez besoin d'accéder au système de fichiers, travailler avec la caméra, l'accéléromètre ou d'autre composants de l'appareil, vous commencez à avoir un plus grand accès à l'appareil.

C'est la qu'intervient PhoneGap.



Lorsque vous utilisez l'API de PhoneGap, vous pouvez écrire votre application sans écrire de code natif (Java. Objectif-C, etc). Au lieu de cela, les technologies web sont utilisés, et sont

hébérgés en local dans l'application elle même. Et parce que ces API JavaScript sont compatibles sur toutes les plateformes mobiles et construites sur les standards du Web, l'application doit être portable sur les plateformes des appareils avec peu ou pas de modifications.

PhoneGap enveloppe votre HTML et votre JavaScript avec juste assez de code natif pour que votre application web se sente plus à l'aise sur l'appareil. Cette enveloppe est différente pour chaque plateforme, mais elle expose ses capacités commune de manière cohérente. Cela vous permet d'écrire moins de code sur de multiples plateformes.

PhoneGap est disponible sur les plateformes suivantes : iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada et Symbian.

Depuis que PhoneGap enveloppe votre code HTML et JavaScript dans une application native, vous gagnez la possibilité de soumettre votre application au store de l'appareil. Chose que vous ne pouvez pas faire avec une application web simple. Gardez à l'esprit, cependant, que la plupart des stores veulent que votre application est le style d'une application native et que certains store sont plus strict que d'autres.

Voici une liste des composants supporté par PhoneGap.

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	X	✓	✓	X	✓
Contacts	✓	✓	✓	✓	✓	X	✓	✓	✓
File	✓	✓	✓	✓	✓	X	✓	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	X	X	✓	X	X
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓	X

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

3.2 SenchaTouch

3.2.1 Qu'est ce que Sencha Touch ?

Sencha Touch est un framework MVC spécialement conçu pour créer des applications web mobile pour les appareils tactile. Sencha Touch permet aux développeurs de créer des applications pour les plateformes mobiles qui disposent de navigateurs supportant les derniers standards, comme le moteur de rendu Webkit.

Au moment d'écrire ces lignes, la dernière version disponible de Sencha Touch étail la version 2.2.0

Sencha Touch est un gros framework, qui peut sembler intimidant pour les développeurs JavaScript habitués à utiliser des petites bibliothèques telles que jQuery ou Prototype.

Sencha Touch est conçu comme un tout, y compris la plupart des fonctions des services offerts par d'autres frameworks, et il peut être facilement étendu de nombreuses et différentes façon pour s'adapter aux besoins des développeurs de différents domaines d'expertises

Vous n'avez généralement pas besoin d'utiliser d'autres bibliothèques que Sencha Touch dans votre projet, si vous avez besoin d'une fonction particulière vous êtes sûr que le framework l'embarque par défaut.

Le choix explicite de Webkit est intéressante, l'équipe de Sencha Touch a pris une décision délibéré de ne pas soutenir d'autres moteurs de navigateurs mobile, tels que Gecko (Firefox), Presto (Opéra), ou Trident (Internet Explorer). Le soutient exclusif des navigateurs modernes permet à Sencha Touch d'utiliser de nombreuses technologies Web les plus avancées.

Ce choix influe également sur l'expérience du développeur, parce que Safari ou Google Chrome peuvent être utilisé pour déboguer les applications Sencha Touch sur un environnement de bureau comme Linux, Windows ou OS X.

L'équipe de Sencha a récemment annoncé le support d'Internet Explorer 10 pour Windows Phone 8.

Quel type d'application peut-on écrire avec Sencha Touch.

Apple, dans l'une des premières version de son guide sur les directives de design pour IOS a énoncé qu'il existe trois grands types d'applications moblles qui peuvent être créé pour l'iPhone.

- les applications utilitaires, comme la météo ou les informations d'informations de stock.
- Les applications de productions, comme les applications d'affaires ou orientés document.
- Les applications immersives comme les jeux vidéo.

Suite à cette taxonomie simple, Sencha Touch est plus apapté pour des applications délivrant

les deux premiers types. Bien qu'il est certainement possible de créer des jeux ou d'autres types d'applications mettant en vedette des expériences utilisateurs complexe, ce mémoire ne couvre que la problématique des applications des deux premiers types.

3.2.2 Un peu d'histoire

Retour en 2005, le mouvement web 2.0 est en train de transformer radicalement la notion de contenu web. Les sites en AJAX comme Gmail ont montré au public qu'un nouveau type d'interaction était possible, qu'un nouveau type de contenu a pu être proposé dans les pages web classiques sans l'aide d'extension propriétaires. Douglas Crockford expliquait que JavaScript était un grand langage incompris par beaucoup, et les bibliothèques comme Script.aculo.us et Prototype offrait aux développeurs de concrètes et solides raison pour développer des applications cross-plateformes.

Au milieu de toute cette agitation , Yahoo a publié la première version de sa bibliothèque YUI, permettant de développer des applications complexes à la "desktop-like" à travers les systèmes d'exploitations et navigateurs. YUI peut être considéré comme une œuvre précurseur, après quoi plusieurs autres bibliothèques sont apparus au fil des ans.

Pendant ce temps, Jack Slocum a commencé à travailler sur un ensemble d'extensions pour YUI appelé YUI-Ext. Après quelques versions, l'interêt pour sa bibliothèques a tellement augmenté qu'il à supprimé l'obligation d'utiliser YUI en complément, rendant la bibliothèque en mesure d'utiliser Prototype et/ou YUI pour un niveau de compatiblité cross-plateformes.

3.2.3 Ext JS est né

Pendant des années, Ext JS a établi la norme en terme de compatibilité cross-browser et de conception, permettant aux développeurs de créer des applications de navigation complexe en une fraction de temps, et sans avoir à se soucier des problèmes de compatibilité entre navigateurs. En 2009, la société derrière Ext JS incorporé comme Sencha Inc, dont le siège se trouve à Redwood City, en Californie.

En 2009, la hausse des martphone à écran tactile et, plus tard, 'liPad, a incité l'équipe d' Ext JS à créer une version du framework orienté exclusivement pour ces nouveaux dispositifs : le résultat de leurs efforts est Sencha Touch, sortie en version1.0 à la fin 2010.

La première version de Sencha Touch nétait pas totalement compatible avec le version courante d'Ext JS, et il a aussi été critiqué pour ses relatives faibles performance, en particulier sur les anciens appareils tels que l'iPhone 3G. Pour répondre à ces question, Sencha Touch 2 a

été publié en Mars 2012 offrant un tout nouveau moteur de rendu basé à 100% sur Cascading Style Sheet (CSS), et un nouveau système de classe compatible avec Ext JS 4

3.2.4 Caractéristiques principales

Sencha Touch est plus que juste un framework complet orienté vers la création de services et d'applications orienté productivité, il est en fait un système web complet d'entreprise pour application cross-plateforme, avec les caractéristiques suivantes :

- Nombreux widget disponibles, largement inspiré par iOS, tant dans la conception que dans les fonctionnalités.
- Moteur de rendu rapide basé sur CSS, qui peut être accéléré par le matériel dès les smartphones moderne.
- Une architecture bien définie, Sencha Touch utilise une architecture MVC.
- Des connecteurs intégrés pour les serveurs de transfert de données réseau, telles que les services web REST et le soutiens aux application web offline mobile.
- Un système de construction de ligne de commande, la gestion de la fusion et de la minification du code de l'application, ainsi que la création d'applications natives pour Android et iOS.

Une documentation complète est disponible comme un ensemble de pages HTML dynamiques, y compris la recherche et le filtrage des fonctionnalités sans nécessiter d'infrastructure serveur.

Sencha Touch peut être considéré comme un framework "tout en un", y compris toutes les API et outils nécessaire pour créer vos applications mobiles.

3.2.5 Supprt appareils et navigateurs

Sencha Touch au moment d'écrire ces lignes ne supporte que les plateformes suivantes :

- iOS depuis la version 3
- Android depuis la version 2.3
- BlackBerry OS depuis la version 6 (uniquement pour les appareils équipés de WebKit)
- Windows Phone 8

Sencha Touch est un framework basé à 100% sur le navigateurs, et vous pouvez déployer vos applications Sencha Touch en utilisant n'importe quelle technologie coté serveur, à l'instar de PHP, Java, Ruby on Rails, .Net ou tout autre langage de votre choix.

3.2.6 Licence

Sencha Touch est disponible sous plusieurs licences :

Pour des projets open-source :

- Si vous prévoyer de distribuer votre application en divulgant pleinement le code source, il y'a une version de Sencha Touch distribué sous licence GPLv3
- Si vous ne souhaitez pas utiliser la licence GPLv3, vous pouvez également utiliser la licence FLOSS.

Pour des projets commerciaux :

- Vous pouvez utiliser Sencha Touch gratuitement, sans aucun frais que se soit par applications, par utilisateur ou par développeur
- Pour les applications embarquées, vous pouvez utiliser Sencha Touch gratuitement jusqu'à 5000 installations.

Enfin, une licence OEM commercial est disponible aussi bien pour les entreprises désireuses de distribuer Sencha Touch comme une partie de leurs propres applications ou services commerciales.

3.3 JQuery Mobile

3.3.1 Introduction

jQuery mobile est un framework open source JavaScript UI construit sur la populaire bibliothèque jQuery, créé par John Resig au cours de la dernière décennie.

Le développement de jQuery mobile a commencé mi-2010, et est rapidement devenu l'un des frameworks JavaScript les plus populaires. Aujourd'hui jQuery Mobile est utilisé dans plus d'application web mobile que n'importe quel autres framework.

jQuery Mobile est un projet open source, hébergé sur Github et avec un site internet très complet, énormément de documentation, d'éxemples et de références à des applications créées avec ce framework. Au moment d'écrire ces ligne, la version actuelle de jQuery Mobile est la version 1.3.1.

3.3.2 Plateformes supportés

jQuery Mobile fonctionne sur la grande majorité des pc moderne, smartphone et tablette et des plateformes eReader.

En outre, les téléphones et les navigateurs plus ancien sont également supportés en raison d'une évolution progressive. C'est probablement l'une des caractéristiques les plus importes de jQuery Mobile

3.3.3 Compatibilité

Les utilisateurs des navigateurs mobiles les plus avancés peuvent profiter de l'une expérience plus complète, avec des transitions de pages animées basées sur Ajax. Au moment d'écrire ces lignes cette liste comprend les systèmes / navigateurs suivant

- iOS depuis la version 3.2
- Android depuis la version 2.1
- Windows Phone depuis la version 7
- BlackBerry depuis la version 6, notament Playbook
- Palm WebOS depuis la version 1.4
- Firefox mobile depuis le 10 bêta
- Skyfire depuis la version 4.1
- Meego depuis la version 1.2
- Samsung Bada depuis la version 2.0

- Navigateur UC
- Kindle et Kindle Fire
- Nook Color depuis la version 1.4.1

Une liste impressionante ! Toutes les plateformes importantes de smartphone à écran tactile sont aujourd'hui disponilbe et sont représentée et soutenue par jQuery Mobile.

Sur les plateformes de bureau, jQuery Mobile est compatible avec Windows, Linux et Mac OS X sur les versions des navigateurs suivants :

- Firefox depuis la version 4
- Chrome depuis la version 11
- Safari deuis la version 4
- Internet Explorer depuis la version 7
- Opéra depuis la version 10

Un des plus grands avantages en regardant les listes ci-dessus, c'est que jQuery Mobile est l'un des frameworks les plus largement compatibles aujourd'hui. Même mieux, son grand suport des navigateurs de bureau permettent aux développeurs d'utiliser différentes plateformes pour construire et tester leurs applications.

Etant donné que les versions les plus récentes de ces navigateurs intègrent des outils de développement, elle augmentent aussi sont attrait auprès des développeurs.

3.3.4 Compatibilités avec les anciennes plateformes mobiles

Mais que faire si nos utilisateurs ou les exigences précisent certaines anciennes plateforme ?

Est-ce que jQuery Mobile peut nous aider dans ce cas ?

Les applications jQuery Mobile sont construit avec une dégration progessive par défaut. Les anciennes plateformes, incapable d'afficher les dernières normes CSS et JavaScript, vont tranquillement affichés par défaut la structure HTML des applications, ce qui pourrait ou non être la solution idéale, mais est au moins une réponse par défaut. Par exemple, les navigateurs suivants ont une expérience améliorée, à l'execption d'Ajax pour la navigation :

- BlackBerry 5.0
- Opera Mini 5.0 à 6.5
- Nokia Synbian 3

Et quelques autres navigateurs ne peuvent profiter que d'une expérience de base en HTML, non amélioré :

- BlackBerry 4.x

- Windows Mobile 6 et plus.
- Plateformes de smartphone plus anciennes, y compris les téléphones.

3.3.5 Principales caractéristiques

Une liste succincte des principales caractéristiques de jQuery Mobile

- Construit sur la syntaxe de jQuery afin d'avoir une syntaxe familière, cohérente et une courbe d'apprentissage minimale
- Compatible avec toutes les principales plateformes de bureau et mobiles : iOS, Android, BlackBerry, Palm WebOS, Nokia/Symbian, Windows Mobile, Opera Mobile/Mini, Firefox Mobile, et tout les navigateurs de bureau récent.
- Léger (environ 20ko une fois compressé avec toutes les fonctionnalités mobile) et minimale
- Utilisation du HTML5 conduit à un développement des pages rapides et à un minimum de scripting requis.
- L'utilisation de l'amélioration progressive apporte un contenu de base et de fonctionnalité à tout mobile, tablette ou navigateur avec une expérience riche.
- Initialisation automatique des widget jQuery en utilisant des attributs HTML dans le code HTML.
- Des fonctions d'accessibilité telles que WAI-ARIA sont également inclus afin de s'assurer que les pages puissent travailler avec les lecteurs d'écran (par exemple, VoiceOver dans iOS) et d'autres technologies d'assistance.
- Prise en charge des événements tactile et de la souris en rationalisant le processus du support tactile.

La chose la plus importante à savoir sur jQuery Mobile est qu'il est une bibliothèque d'interface utilisateur, pas un plugin de jQuery. C'est une bibliothèque qui aura des balises HTML en entrée et utilis des styles prédéfnis en les adaptant aux capacités des navigateurs actuel. Ce n'est pas un framework complet comme .NET, Java, ou encore Sencha Touch qui fournissent des services de niveau inférieur, comme la sérialisation, le stockage ou le réseau.

jQuery mobile s'appuie sur JavaScript et les fonctionnalités du HTML5 prises en charge par le navigateur utilisé.

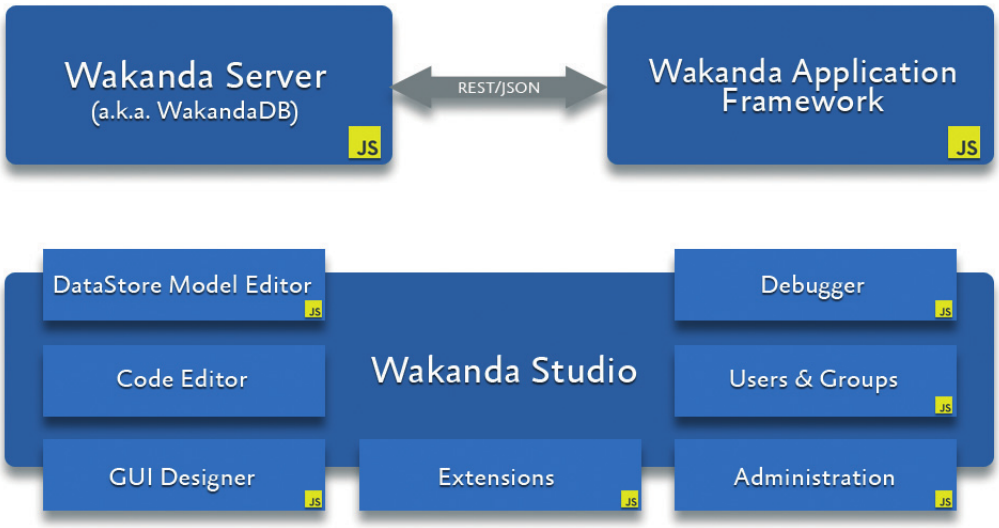
Cette première caractéristique détermine pourquoi le support des navigateurs mobiles de jQuery Mobile est important comparé aux développeurs Java devant déployer leur propre code pour mettre en œuvre des comportements complexe comme le stockage ou d'intérargir avec le matériel exposés par le navigateur de l'hotes (géolocalisation, boussole, etc)

Une autre caractéristique importante de jQuery Mobile est qu'elle n'impose aucune sorte de structure du code JavaScript dans votre application, la principale composante de l'application étant les fichiers HTML qui défnissent la sémantique de l'interface utilisateur, mais pas son look. En général, les développeurs s'appliqueront à utiliser le comportement de jQuery en utilisant sa norme, sa syntaxe comme avec n'importe qu'elle page web ordinaire.

3.4 Wakanda

3.4.1 Introduction

Wakanda est une plateforme Open Source qui propose une dual licence, avec des capacités de développement strictement identique dans les deux cas. La création d'une communauté d'utilisateurs et contributeurs est clairement l'ambition première affichée par l'éditeur de Wakanda, dans le but de garantir la pérennité des projets et de s'adapter au foisonnement d'idées, de frameworks, d'appareils et d'usages qui se déroule sous nos yeux.



Au coeur de la plateforme se niche WakandaDBn un datastore objet NoSQL ultra-performant, qui, joint au moteur d'exécution JavaScript, basé sur Webkit, et à un serveur HTTP communiquant via JSON/REST constitue Wakanda Server, le back-end multi-plateformes (Linux, Mac, Windows) qui permettra d'héberger une solution Wakanda sur un serveur dédié, sur un Cloud privé ou public, en mode SaaS, etc...

Le framework Wakanda est automatiquement déployé vers les clients HTML et embarque un Data Provider qui se comporte comme un proxy des différentes DataClasses générées coté serveur. Ainsi de façon transparente, tous les objets JavaScript générés à l'aide du modèle de données sont exploitables via les différents widgets fournis avec le framework, mais également au travers de requêtes REST pour potentiellement connecter tout type d'interface, par ligne de commande ou à l'aide d'adaptateurs que pourront fournir des tierces parties, la communauté, ou l'éditeur 4D lui-même. Enfin, le studio Wakanda vient compléter la suite de développement en proposant un IDE dédié capable de contrôler, via une application Desktop Mac ou Windows, aussi bien le modèle, les widgets, et l'adaptation cross-device de chaque écran coté client.

Le studio fait la part belle à la gestion graphique du projet, et fait en sorte d'économiser à chaque étape l'écriture de code «technique». Le développeur peut dès lors se concentrer sur ses propres scripts et méthodes qui contiendront l'intelligence métier qui fera la valeur de son application. Outre les différents éditeurs (modèles, code, UI, utilisateurs et groupes) le studio propose une fenêtre d'administration des solutions et projets et un débuggeur qui permet de tracer le code JavaScript tournant sur le serveur. L'installation, tout comme le déploiement, s'effectuent par drag-and- drop d'un package sans aucune configuration. L'hébergement d'une solution Wakanda peut s'effectuer sur un serveur dédié, sur un VPS (serveur dédié virtuel, ou même sur une plateforme IaaS Cloud comme Amazon EC2 ou Rackspace. Côté mobile, Wakanda permet de réaliser des applications Web ou Hybrides, c'est-à-dire embarquées dans un runtime natif généré par PhoneGap. Il est également possible de créer des applications mobiles natives et d'accéder au Serveur via ses API REST et JSON-RPC.

Index