LE JAVASCRIPT CÔTÉ SERVEUR

LA REVOLUTION DU WEB

Le JavaScript côté serveur sera-t-il aussi pérenne que le Java et le PHP ?

Mohamed Aharchi

Master II – Expert en Ingénierie

Informatique Mémoire de fin d'étude

2017-2018

campus sciences-U LYON

Résumé

De nos jours, les entreprises attendent des développeurs qu'ils puissent maîtriser plusieurs langages informatiques. Il est courant de voir des fiches de poste du type « Java / Angular (JavaScript) ». Alors on se demande d'où provient cette double compétence requise ?

Il faut savoir que cela n'a pas toujours été ainsi. De base on recrutait des développeurs Java pure pour réaliser des applications logicielles avec un seul langage. Les navigateurs n'étaient pas assez puissants pour supporter de grosses applications type client lourd. Mais ce temps est révolu !

En effet avec la montée en puissance des applications web grâce notamment grâce à une amélioration de performances des navigateurs et des serveurs web, la majorité des entreprises tend maintenant à développer leur pôle applicatif web.

Jusque-là, pas de problèmes. Seul bémol les langages populaires comme Java ou encore PHP sont des langages serveurs. C'est-à-dire que pour une application web donnée, lors d'une requête d'un navigateur web pour accéder à l'application, ses fichiers sont exécutés depuis le serveur d'hébergement de l'application. Ensuite, les données sont retranscrites en HTML et renvoyé au navigateur. Ainsi une fois la page web chargé, plus aucune possibilité d'exécuter du code serveur sans devoir recharger la page. On obtient donc une application web avec les données que nous souhaitons, mais qu'en est-t-il du design ? Le design s'exécute côté client, les langages serveurs ne peuvent donc pas y accéder. C'est là qu'intervient JavaScript, un langage côté client de base qui permet de modifier le design d'un site web. Voilà pourquoi il est nécessaire de maîtriser plusieurs langages afin de réaliser une application web complète. Alors comment faire pour réaliser une application web à l'aide d'un seul langage afin de palier au problème du multiple langage ?

Dans les premières versions de JavaScript, le langage était uniquement côté client. Mais depuis quelques années, JavaScript à intégrer le côté serveur grâce au serveur « Node.js ». Ce dernier a été créé en 2008 et a fait beaucoup parler de lui dû aux nombreuses possibilités qu'il apportait et qu'il apporte toujours dans le développement d'application web. Ainsi il est aujourd'hui possible de mettre en place une application full JavaScript qui allie côté client et côté serveur, on appelle cela le JavaScript FullStack.

Pour cette raison, j'ai souhaité faire une analyse sur cette technologie. Quel est le potentiel du JavaScript server-side ? Deviendra-t-elle une technologie majeure dans le développement web ?

Qu'en est-t-il des applications JavaScript FullStack ? Node.js possède l'une des communautés les plus dynamiques si ce n'est la communauté la plus importante de tous les langages. « NPM » son gestionnaire de paquets officiel est l'une des bibliothèques de modules les plus importantes. De plus, la communauté Node.js crée actuellement le plus grand nombre de modules par jour. Plusieurs frameworks ont été mis en place et cette technologie connait une montée en puissance faramineuse.

Node.js doit cette popularité à une légèreté et une performance sans précédent. Mais aussi grâce à de nouvelles fonctionnalités, comme le temps réel (rendu possible grâce au module Socket.io).

Les langages comme Java et PHP restent préférés à JavaScript dans de nombreuses entreprises car ils ont su rester productif depuis plusieurs années contrairement à d'autres technologies qui ont coulés alors qu'elles étaient annoncées comme étant le futur standard. Est-ce vraiment une question de performances ou simplement ces entreprises ne prennent aucun risque à se lancer sur une nouvelle technologie ?

Quoi qu'il en soit JavaScript côté serveur à nettement confirmé ce qu'on attendait de lui ce n'est pas pour rien qu'il connait une croissance positive depuis plusieurs années

Ainsi, la question **« Le Javascript côté serveur sera-t-il aussi pérenne que le Java et le PHP ? »** ne se pose plus, car cette technologie est déjà en place depuis plusieurs années et deviendra probablement un standard du web.

Malgré cela, de nombreuses sociétés ainsi que de nombreux développeurs ne souhaitent pas prendre le risque de se lancer dans une telle technologie et préfèrent rester sur leurs acquis par précaution. Il est, par conséquent, important de montrer le potentiel de cette technologie, car elle est nettement plus avantageuse que d'autres sur certains cas d'utilisation et c'est la raison même de ce mémoire.

Mots clés : langages serveurs, fullstack, Node.js, Java, PHP

Abstract

Today, companies expect from developers to master several languages. It is common to see job descriptions of the type "Java / Angular (JavaScript)". So, we come to wonder where does this dual competence come from?

It must be known that it has not always been so. Basically, companies were recruiting pure Java developers to realize software applications with a single language. Browsers were not powerful enough to support heavy-client applications. But that time is over!

Indeed, with the rise of web applications thanks to improved performance of browsers and web servers, most of companies now tend to develop their web application.

Until then, no problems. Only downside popular languages like Java or PHP are server-side languages. In other words, we can say that for a web application, during a request from a browser to access to this application, its files are executed from the hosting server of the application. Then, the data is transcribed in HTML and returned to the browser. Once the web page loaded, no more possibility to run server script without having to reload the page. So, we get a web application with the data we want, but what about the design? The design runs on the client side, so server languages cannot access it. That's where JavaScript comes in, a client-side language at the origin that can modify the design of a website. Therefore, it is necessary to master several languages in order to create a full web application. So how to make a web application using a single language to overcome the problem of multiple languages?

Since few years, the client-side JavaScript language has integrated the server side with JavaScript server "Node.js". The latter was created in 2008 and has been much talked about due to the many possibilities it brings in the development of web application. So, it is now possible to set up a full JavaScript application on the client side and server side, this is called the Full Stack JavaScript.

For this reason, I wanted to do an analysis on this technology. What is the potential of the JavaScript server-side? Will it become a major technology in web development? What about Full Stack JavaScript applications? Node.js has one of the most dynamic communities. "NPM" its official package manager is one of the most important module libraries. In addition, the Node.js community is creating the largest number of modules per day. Several frameworks have been put in place and this technology knows a tremendous rise in power.

Node.js owes this popularity to such a big lightness and performance. But also, thanks to new features, such as real time (made possible thanks to the Socket.io module).

Languages such as Java and PHP are still preferred to JavaScript in many companies because they have been able to remain productive for several years unlike other technologies that have flowed when it was announced as the future standard. Is it really a question of performance or are these companies taking no risk to embark on a new technology?

Anyway, JavaScript server side has clearly confirmed what was expected of him. It is not for nothing that it knows a growth for several years

So, the question "Will server-side JavaScript be as durable as Java and PHP?" Is no longer an issue because this technology has been effective for many years and will probably become a web standard.

Despite this, many companies and many developers do not wish to take the risk of embarking on such technology and prefer to stay on their gains as a precaution. It is therefore important to show the potential of this technology because it is much more advantageous than others in some use cases and this is the purpose of this memory.

Key words: server-side, full stack, Node.js, Java, PHP

Table des matières

Remerciement

Je tiens tout d'abord à remercier mon tuteur en entreprise, Christophe Cannie, sans qui je n'aurais pas eu la possibilité de mener à bien ce mémoire. Il a su se comporter comme un manager de grande qualité, sa flexibilité, son support m'ont permis de gérer mon stress et fournir ce travail, c'est pourquoi je lui suis reconnaissant.

Ensuite je tiens à remercier tout le corps enseignant de Sciences-U qui m'ont montré la voie à suivre pour bien le structurer. Merci aussi à Mr Noureddine Bouiche et Mr Guillaume Noëlaux pour m'avoir guidé et orienté sur le sujet de mon mémoire.
Merci plus particulièrement à Mr Noël BARON pour son suivi attentionné et ses conseils avisés quant aux recherches à effectuer. Merci d'avance pour son investissement concernant la lecture de mon mémoire.

Je remercie également mes collègues, mes amis ainsi que ma famille qui ont lu et relu mon mémoire pour m'aider à corriger les imperfections qui y subsistaient.

Sans oublier ceux qui m'ont soutenu dans ce sujet malgré ma faible connaissance de base du monde JavaScript, mener à bien ce mémoire est un bon moyen de clôturer la fin de mes études.

Pour finir je remercie tous les rédacteurs journalistiques, les journalistes, sociologues et internautes qui ont fournis des statistiques, des faits, des études ainsi que des schémas qui m'ont permis d'imager et d'appuyer de manière factuelle les écrits que j'ai intégré dans ce document.

Introduction

La technologie JavaScript côté serveur, fait beaucoup parler d'elle depuis quelques années, j'ai décidé donc de faire un point sur cette technologie, notamment avec la plateforme de développement d'application web Node.js qui a permis d'ouvrir la porte au JavaScript Full Stack.

Node.js connaît actuellement une ascension fulgurante mais va-t-elle le rester pour les prochaines années ? En effet plusieurs technologies ont vu le jour à travers le temps et étaient annoncés comme des futurs standards de leur domaine. Pour beaucoup d'entre elles, leur début a été fracassant et était la mode de leur époque mais elles n'ont pas forcément duré dans le temps hormis quelques exceptions qui continue de perdurer dans le temps comme les langages Java et PHP. Plusieurs questions viennent à se poser, la plateforme Node.js va-t-elle perdurer ? Sera-t-elle une technologie aussi pérenne que le Java ou encore le PHP ? Va-t-elle dans les années à venir, devenir un standard du web, ou disparaîtra-t-elle aussi vite qu'elle est apparue ?

Node.js a maintenant presque dix ans et elle a su rester au premier plan. On peut donc dire qu'elle a quasiment assurée sa pérennité. Mais elle reste tout de même un choix secondaire face à Java et PHP hormis pour certains cas spécifiques. Alors quelles sont les raisons de cette seconde place ? Est-ce que le monde informatique connaît vraiment le JavaScript côté serveur ou bien est-ce que les entreprises ne souhaitent pas prendre le risque de migrer vers cette technologie ?

Je vais donc ici, faire un point sur l'état actuel du JavaScript serveur, afin de déterminer son avenir et le mettre en comparaisons avec la concurrence. Ceci permettra d'avoir un regard plus détaillé et de choisir Node.js pour les cas d'utilisations appropriés. Ce mémoire aura pour but finale de proposer la technologie JavaScript serveur entre autres Node.js en tant que solution.

Je n'ai pas vraiment rencontré de difficultés particulières dans la rédaction de ce mémoire. La seule réelle difficulté a été de rendre ce mémoire attractif en mettant en avant cette technologie notamment en la vendant en tant que solution de projet web.

Pour répondre à cette problématique nous allons devoir exposer dans un premier temps comment l'environnement de la programmation a évolué et où elles en sont aujourd'hui, notamment au niveau :

- **Des paradigmes serveurs.**

  Cela représente les différentes méthodologies courantes de programmation ainsi que le prochain paradigme possible pour la suite à savoir, la métaprogrammation.

- **Des stack holders.**

  Qui sont les acteurs principaux des technologies de programmation. En effet, les technologies sont souvent issues d'entreprises de renommée comme IBM qui ont permis l'évolution de l'information dans sa globalité, j'appelle cela les **créateurs de technologies**. Mais une fois que les technologies de programmation sont créées, il faut qu'elles puissent être mis en avant et pour cela rien de mieux que leurs utilisations par de grandes entreprises qui agissent ainsi en tant que **précurseurs de la programmation**.

- **Des applicatifs serveurs.**

  Afin de pouvoir vendre l'idée du JavaScript Serveur, il faut le mettre en comparaisons direct avec ses concurrents les plus populaires à savoir les langages Java et PHP. C'est pourquoi j'ai décidé de rédiger une présentation détaillée de ces langages.

Et par la suite, mettre en évidence le sujet principal et d'où il tire sa source, notamment :

- **Qu'est-ce que le JavaScript ?**

  Une présentation détaillée de JavaScript et de ses fonctionnalités phares permettra dans un premier temps d'en savoir plus sur la technologie principale de ce mémoire qui est la source du Node.js.

- **L'intégration de JavaScript côté serveur.**

  Après avoir eu une vision plus claire de JavaScript, c'est le moment d'entrer dans le vif du sujet à savoir la technologie JavaScript côté serveur notamment avec la plateforme Node.js. J'estime qu'il est important de savoir pour quelle raison a-t-on commencé à utiliser le JavaScript côté serveur et comment s'est déroulé son intégration.

- **La frameworktisation de JavaScript côté serveur.**
  Node.Js est une plateforme très ouverte, c'est-à-dire qu'on peut l'utiliser pour créer divers framework. Avant de proposer une solution finale, j'ai rédigé un listing des principaux framework Node.js qui ont chacun leur point fort selon le cas d'utilisation.

Pour finir nous proposerons Node.js en tant que solution pour différents cas d'utilisation en séparant cela en trois parties à savoir :

- **Les cas d'utilisation de JavaScript côté serveur (Node.js).**
  Node.js excelle dans certains domaines, nous listerons les principaux cas d'utilisation pour lesquelles il est recommandé d'utiliser Node.js.
- **Les cas où il ne faut pas utiliser Node.js.**
  Il serait merveilleux de savoir que Node.js est approprié à tout type de projet mais ce type de technologies n'existe pas. L'idée de ce mémoire n'est pas de vendre a tout pris Node.js sans être objectif. Nous listerons donc les différents cas d'utilisation où il est recommandé de ne pas utiliser Node.js.
- **JavaScript Serveur correspondra-t-il à votre projet de demain ?**
  Dernier point, nous proposerons Node.js comme solution possible pour votre projet à l'aide de comparatifs entre Node.js et les technologies Ruby, Java et PHP.

Environnements de la programmation

I) Evolution des paradigmes serveurs

A. Introduction

On se pose toujours plusieurs questions, quand est-ce que la programmation fonctionnelle est devenue une chose énorme que tous les langages informatiques implémentent ? Qu'est-ce qui a conduit à sa popularité ? Et je pense que beaucoup de développeurs se sont déjà demandé : maintenant que nous avons des programmes fonctionnels dans les langues dominantes, quelle est la prochaine étape ? Eh bien, voici une supposition éclairée à ce sujet. Mais, d'abord, il est nécessaire d'avoir une leçon d'histoire sur les différents paradigmes de programmation et pourquoi ils ont été mis en œuvre dans les langages populaires que les entreprises utilisent chaque jour.

Comme d'habitude, pratiquement rien en informatique n'est "nouveau". Il y avait beaucoup d'expérimentations dans les années soixante et soixante-dix avec des langages de programmation différents et donc des paradigmes différents. Tout ce qui a trait à la langue a été essayé au moins une fois pendant cette période. Voici une introduction aux paradigmes et les langages grand public qui les utilisent.

Remarques : les explications qui vont suivre sont uniquement des suppositions éclairées qui ne sont pas basées sur des preuves. C'est à peu près comme répondre "pourquoi Pokémon est devenu populaire". Aucune preuve n'existe, mais nous pouvons faire des suppositions pour savoir pourquoi.

B. Le premier paradigme : la programmation procédurale

Tout d'abord il y a eu la programmation procédurale. Cela a été particulièrement bien marqué avec la création et la montée du langage C. La raison pour laquelle elle est devenue populaire était qu'on pouvait écrire du code qui pourrait traduire de très près le langage d'assemblage, donc un langage très bas niveau. Les ressources étaient rares, mais écrire directement dans le langage de l'assemblée commençait à être impraticable. Maintenant, on peut se demander, pourquoi les autres paradigmes ne sont-ils pas devenus populaires à ce stade ?

Voici une énumération des problèmes de chaque paradigme :

La programmation fonctionnelle était présente avec la famille de langage Lisp qui est d'ailleurs la plus ancienne famille de langages informatique. Cependant le garbage collector (collecte des ordures :

sous-système informatique de gestion automatique de la mémoire) est nécessaire. Avec les ressources étant rares, ce n'était pas la meilleure voie à suivre. En outre, à ce stade, la plupart des gens connaissaient le langage d'assemblage et étaient toujours familiers avec les détails de bas niveau comme les cartes perforées. Mettre une grande quantité d'abstraction au-dessus de ce concept signifiait qu'il serait difficile d'apprendre.

La programmation axée sur la pile était quant à elle utilisée par le langage Forth. Cela ne nécessitait pas de garbage collector, mais contenait une énorme couche d'abstraction au-dessus du jeu d'instructions réel. En revanche, cela ne voulait pas dire que c'était plus lent. En effet, c'était plus difficile pour les programmeurs qualifiés de s'y adapter.

On voit maintenant que la programmation procédurale est la meilleure avancée du codage de l'assemblage parce que l'assemblage est essentiellement une programmation procédurale. Cependant, le langage C introduit beaucoup de fonctionnalités. La plus importante est de rendre la compilation croisée réalisable. Le langage étant assez simple, ce qui a facilité la tâche des nouveaux compilateurs. En regardant tout cela en détail, on peut se demander pourquoi le langage Forth n'est pas devenu plus populaire que le C.

C. La programmation orientée objet

Suivant sur la liste des grands changements du paradigme : la programmation orientée objet. Cette méthodologie est devenue populaire bien après sa création. Le tournant qui l'a vraiment rendu populaire a été la mise en place de l'interface graphique. Les objets correspondent naturellement aux éléments de l'interface graphique. En effet avant les interfaces, la programmation orientée objet existait mais son utilité était moindre car cela compliquait les compilateurs et exiger plusieurs passages entre les appels et le code. Finalement, les compilateurs ont rattrapé le retard et C++ a remplacé C en tant que langage de programmation principal. Cette idée de la programmation orientée objet a vraiment été lancée dans le courant dominant avec la popularité de Java.

Voici un historique de la programmation objet :

- 1963 - Sketchpad d'Ivan Sutherland est considéré comme un travail de pionnier dans l'orientation des objets et les interfaces graphiques.
- 1967 Simula apparaît et on ne sait toujours pas qui de Simula ou Smalltak est considérée comme le premier langage orienté objet.
- 1969 - Dennis Ritchie commence à développer le langage C.

- 1972 - Smalltalk, une langue fortement influencée par Simula, apparaît. C'est l'idée originale d'Alan Kay, qui est souvent considéré comme l'inventeur du terme « orientation d'objet ».
- 1979 - Bjarne Stroustrup commence à travailler sur l'ajout des Classes dans le langage C, le précurseur de C++.
- 1983 - Objective C apparaît et est essentiellement un effort pour ajouter la saveur de Smalltalk de l'orientation objet au langage C.
- 1985 - Object Pascal apparaît et est presque immédiatement popularisé par Turbo Pascal 5.5.
- 1986 - Début du travail sur CLOS, un effort pour amener l'orientation objet au Lisp commun.
- 1991 - Visual Basic est publié.
- 1995 - Java est publié.
- 1995 - Delphi est libéré.

D. Garbage collection

Alors, quel est le prochain paradigme ? C'est probablement le plus gros, à savoir le Garbage collection littéralement collection des ordures en français. La mémoire et les bons algorithmes de collecte étaient finalement assez bon marché pour laisser les programmeurs oublier la gestion de la mémoire. Bien sûr, cela existait bien avant qu'il ne devienne courant, mais la plupart des programmeurs le considéraient comme lent et inutile (ce qui était sans doute le cas à l'époque). La principale raison pour laquelle le garbage collection est devenu si répandu est parce que la programmation informatique avait finalement atteint un point de bascule où le temps de calcul et les ressources étaient moins gourmands que le temps de programmation.

E. Génériques

Les génériques sont le paradigme suivant. Il permet de typer statiquement un objet qui peut prendre plus d'un seul type. C++ a été le premier à utiliser ce paradigme. Ada a utilisé des génériques depuis sa création dans les années soixante-dix. La principale raison pour laquelle les génériques sont devenus si courants est que la programmation orientée objet est devenue courante. Faire de la POO d'une manière statique est assez lourd sans génériques. Les développeurs commençaient à se rendre compte que dupliquer le code et utiliser des tonnes de distributions explicites était vraiment une mauvaise pratique.

F. Programmation fonctionnelle

Le prochain paradigme est en train de devenir le favori de plus en plus de développeurs, ce n'est autre que la programmation fonctionnelle. Effectivement cette tendance tant à se développer. La programmation fonctionnelle a vraiment semblé toucher le grand public avec le support de .Net, même si JavaScript est fonctionnel depuis 1995. JavaScript n'est pas vraiment utilisé jusqu'au début des années 2000 avec l'avènement des navigateurs modernes et le respect des normes (le début de la haine intense envers Internet Explorer 6). Donc, la programmation fonctionnelle ne sera pas vraiment reconnue comme devenue courante tant que tous les langages favoris n'auront pas commencé à ajouter des aspects fonctionnels. La principale raison motrice de la programmation fonctionnelle est liée à l'implémentation des processeurs dual-core sur les nouveaux ordinateurs. Soudainement, la programmation concurrente était quelque chose qui préoccupait tout le monde. La programmation fonctionnelle convient parfaitement aux tâches simultanées. Les programmes fonctionnels n'ont aucun état et permettent le multitâches. Cela a également vu la popularité de nombreuses langues accroître aussi bien. Haskell avait commencé à être considéré comme un vrai langage et non pas seulement comme un langage de recherche. Mais aussi F# est effectivement utilisé dans certains produits de production, Scala semble être là où tous les programmeurs JVM modernes sont. JavaScript voit maintenant une énorme quantité d'utilisation, ce qui nécessite naturellement une programmation fonctionnelle pour être correcte.

Alors, quel est le prochain paradigme ? Rien est encore sûr, mais on entend de plus en plus parler de la **Méta programmation**.

G. Et la suite ? : Méta programmation

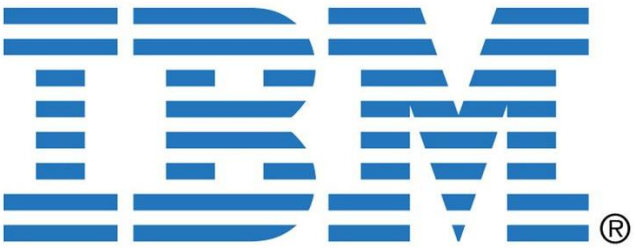
L'utilisation de T4 se fait de plus en plus. C'est une façon de prendre du code fastidieux et de le transformer en quelque chose qui fonctionne correctement et ce ne serait pas possible par d'autres moyens. C'est vraiment juste un pas envers la méta programmation. Écrire des programmes qui s'écrivent eux-mêmes. Et bien sûr, la réflexion avec .Net et Java est déjà commune. Cependant, il n'y a pas beaucoup de langages courants à ce stade qui rendent la méta programmation particulièrement facile. Cependant, il a déjà été constaté une augmentation dans ce domaine avec des langages grand public comme Ruby et Python. Là où la méta programmation brille vraiment sont les langages qui ne contiennent pas beaucoup de support facile à utiliser autre que des API assez basiques. Bien que ces langages ne soient pas statiquement typés. Le T4 est une exception, mais même T4 n'a certainement pas fait l'objet d'une utilisation grand public.

Alors, pourquoi la méta programmation n'est-elle pas déjà à la mode ? La raison la plus probable vient sûrement de la complexité du compilateur. Il peut être extrêmement difficile d'implémenter un interpréteur dans un compilateur. À part cela, c'est juste une question de temps. C'est pourquoi une veille technologique permettra d'avoir un coup d'avance sur le futur car la méta programmation va s'étendre rapidement.

II) Les stacks holders du développement informatique

A. Créateurs de technologies

Tout livre à son auteur tout comme toute technologie à son fondateur. Ces créateurs de technologies sont la racine de ce que nous utilisons aujourd'hui couramment. D'ailleurs beaucoup de technologies se basent sur d'autres technologies déjà existantes. Notamment en programmation informatique les langages comme le Java, C#, PHP tirent leur noyau du langage C. Les créateurs de technologies sont donc nécessaires à l'évolution dans chaque domaine. Généralement ces inventions proviennent de grandes entreprises et de consortium.



IBM est l'un des créateurs emblématiques des technologies. En effet, en 1954 John Backus ingénieur en radiophonie chez IBM ainsi que son équipe ont mis en place le compilateur Fortran (FORmula TRANslator). C'est un langage de programmation qui combine les paradigmes suivants : impératif (procédural, structuré, orienté objet), générique. Depuis sa création et sa sortie commerciale en 1957 comme l'ancêtre du logiciel, Fortran est devenu le premier standard de langage informatique, Il a permis d'ouvrir la porte à l'informatique moderne et pourrait bien être le langage le plus influent dans l'histoire. Fortran a libéré des ordinateurs du domaine exclusif des programmeurs et les a ouverts à presque tout le monde. Il est toujours utilisé plus de 50 ans après sa création.

Fortran a rendu le code compréhensible aux personnes ayant une expertise dans des domaines autres que l'informatique, ouvrant la programmation aux mathématiciens et aux scientifiques. Une personne connaissant l'algèbre avec un niveau lycéen mais rien au sujet des ordinateurs pourrait probablement comprendre des déclarations de Fortran. Fortran a commencé le processus

d'abstraction des logiciels à partir du matériel sur lequel il était exécuté. Les programmes de langage machine précédents devaient être écrits pour un ordinateur spécifique, tandis qu'un programme Fortran pouvait être exécuté sur n'importe quel système avec un compilateur Fortran.

Ce qui était autrefois une tâche laborieuse consistant à saisir manuellement jusqu'à 1 000 instructions de programme pour un problème donné pouvait maintenant être traduit, automatisé et réduit à seulement 47 lignes dans Fortran.

Le développeur du système d'exploitation UNIX (Ken Thompson chez Bell Labs en 1969) rappelle que « 95 % des personnes qui ont programmé dans les premières années n'auraient jamais pu le faire sans Fortran ». Le programme est en substance un compilateur. Un programmeur utilisant Fortran n'écrit que 5 % de toutes les instructions et le programme génère (compile) les 95 % restants pour l'ordinateur.

Voici un exemple de code Fortran :

```
      PROGRAM DEGRAD
!
! Imprime une table de conversion degrés -> radians
! ================================================
!
! Déclaration des variables
      INTEGER DEG
      REAL RAD, COEFF
!
! En-tête de programme
      WRITE (*, 10)
   10 FORMAT (' ',20('*') /
     &        ' * Degres * Radians *' /
     &        ' ', 20('*') )
!
! Corps de programme
      COEFF = (2.0 * 3.1416) / 360.0
      DO DEG = 0, 90
         RAD = DEG * COEFF
         WRITE (*, 20) DEG, RAD
   20 FORMAT (' *  ',I4,'  * ',F7.5,' *')
      END DO
!
! Fin du tableau
      WRITE ( *, 30)
   30 FORMAT (' ',20('*') )
!
! Fin de programme
      STOP
      END PROGRAM DEGRAD
```

IBM a été également amené à combiner certaines technologies semi-externes avec ses propres créations. Notamment avec la machine IBM AS400 et le langage COBOL qui a été développé par un consortium de six personnes.

Facebook est une société américaine créée en 2004 par Mark Zuckerberg. Initialement concentrée sur le réseau social Facebook, la compagnie a racheté Instagram en 2012, WhatsApp et Oculus VR en 2014. Le réseau social est codé en PHP et ReactJs. Ce qui nous intéresse ici est surtout le ReactJS : une création de la société Facebook.

Qu'est-ce que React JS ?

React a été créé par Jordan Walke, un ingénieur logiciel travaillant pour Facebook. ReactJS est essentiellement une bibliothèque JavaScript open source qui est utilisée pour créer des interfaces utilisateur spécifiques pour les applications d'une seule page. Il est utilisé pour gérer la couche de vue pour les applications Web et mobiles. React nous permet également de créer des composants UI réutilisables. React a d'abord été déployé sur le fil d'actualité de Facebook en 2011 et sur Instagram.com en 2012.

Voici un exemple de code React JS :

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

React permet aux développeurs de créer de grandes applications Web qui peuvent changer les données sans recharger la page. Le but principal de React est d'être rapide, évolutif et simple. Cela ne fonctionne que sur les interfaces utilisateur dans l'application. Cela correspond à l'affichage dans le modèle MVC. Il peut être utilisé avec une combinaison d'autres bibliothèques ou frameworks JavaScript, tels que Angular JS dans MVC.

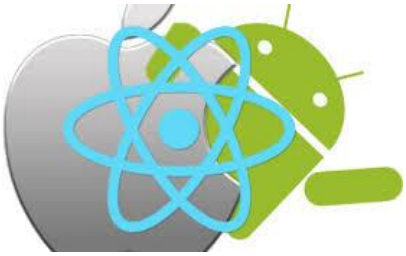
Quelles sont les fonctionnalités de ReactJS ?

Examinons de plus près certaines caractéristiques importantes de React.

- **JSX**

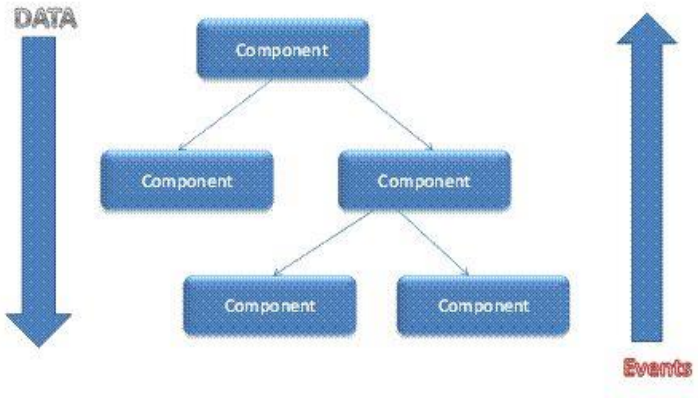
React JS n'utilise pas le JavaScript standard pour modéliser ses pages, il utilise JSX. JSX est un langage JavaScript simple qui permet la citation HTML et utilise une syntaxe de balise HTML. La syntaxe HTML est traitée dans les appels JavaScript de React Framework. Nous pouvons aussi écrire en pur JavaScript.

- **React Native**



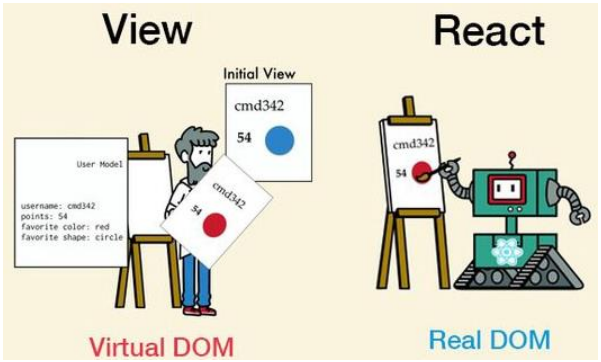
React dispose de bibliothèques natives qui ont été mises en place par Facebook en 2015. React Native fournit l'architecture nécessaire aux applications mobile natives comme IOS et Android.

- **Single-Way data flow**



Avec React, un ensemble de valeurs immuables sont transmises au composant de rendu en tant que propriétés dans ses balises HTML. Le composant ne peut pas directement modifier les propriétés mais peut passer une fonction de rappel à l'aide de laquelle nous pouvons faire des modifications.

- **Virtual Document Object Model**



React crée un cache de structure de données en mémoire qui calcule les modifications effectuées, puis met à jour le navigateur. Cela permet une fonctionnalité spéciale qui permet au programmeur de coder comme si la page entière était rendue à chaque changement, alors que la bibliothèque de réaction ne restitue que les composants qui changent réellement.

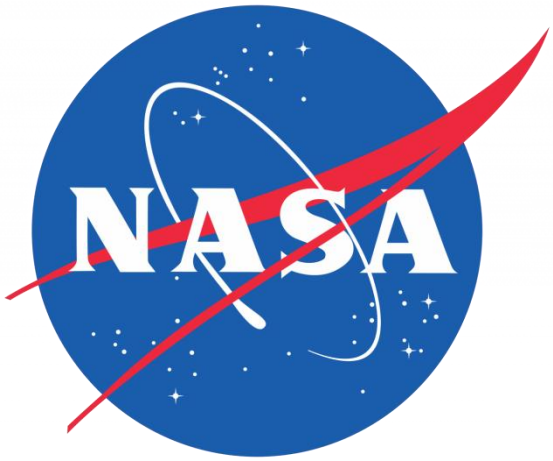
C'était donc une présentation de React et ses fonctionnalités. C'est une des inventions phares de Facebook. Actuellement React est de plus en plus utilisé notamment React Native pour les applications mobiles. Nous en entendrons sûrement parler davantage dans les prochaines années.

B. Les précurseurs de la programmation

Au fil des années, de nombreux langages informatiques ont vu le jour, mais seulement certains d'entre eux ont su se démarquer et faire partie des cadors. Comme on le sait, le monde informatique est en perpétuelle évolution. Ainsi ces cadors sont constamment en concurrence. Parmi ces langages certains ont su conserver leur rang mais d'autres s'éteignent petit à petit.

Ce qui a permis à ces langages de monter sur les devants de la scène, ce sont surtout les influenceurs. Alors quand je dis influenceurs, cela n'a complètement rien à voir avec les influenceurs YouTube et Instagram. Quand je parle d'influenceurs je pense notamment à ces grandes entreprises qui ont créé et utiliser ces langages dans le cadre de divers projets à grande échelle. Ainsi ils ont en quelque sorte servi de cobaye et de précurseur pour démonter la capacité et la puissance de ces différents langages.

Parmi eux, on peut distinguer de très grands groupes comme la NASA, Apple, Google, Microsoft, Amazon, Facebook et bien d'autres. Voici un découpage de ces différentes technologiques par précurseurs.



La National Aeronautics and Space Administration plus communément appeler NASA est connue mondialement donc inutile de démontrer l'influence qu'elle peut avoir. Cette agence gouvernementale dispose sûrement de beaucoup de projets d'ingénierie qui nécessite forcément des machines informatiques pour accompagner leurs recherches.

De 1957-1959 la NASA a été un précurseur majeur des langages ci-dessous :

- FORTRAN (FORMULA TRANSLATION)
- LISP (LIST PROCESSOR)

- COBOL (COMMON BUSINESS-ORIENTED LANGAGE)

Ces langages sont considérés comme les plus anciens langages encore utilisés de nos jours.
Ils ont été créés à des fins scientifiques, mathématiques et gestion des données informatiques. On retrouve également souvent ces langages dans le secteur bancaire.

Apple nous a tous marqué avec son design révolutionnaire et l'arrivée des iPhones. Pour faire simple, Appel est une entreprise multinationale américaine qui conçoit et commercialise des produits électroniques grand public, des ordinateurs et des logiciels. En tant que leader mondial des smartphones et réputé pour la qualité de ses ordinateurs, il en va de soi qu'Apple peut influencer énormément de monde. Comme on peut le deviner, cette société a permis de mettre en avant certains langages informatiques et certains paradigmes de développement en commençant par l'ordinateur Lisa ci-dessous.

Apple Lisa est un ordinateur personnel lancé par Apple en 1983. Il s'agit de l'un des premiers ordinateurs personnels à posséder une souris et une interface graphique. Le système de cet ordinateur a été développé en PASCAL. Sa commercialisation a permis de démontrer la puissance du langage PASCAL qui sera également utilisé 20 ans plus tard par l'application Skype (2003).

Le PASCAL est un langage de type programmation structuré mixé avec des données structurées qui a été propulsé dans les années soixante-dix. On retrouve ce type de programmation (mélange de structure + database) avec des langages propriétaire comme C/AL et C/SIDE utilisés sur les ERP Microsoft Dynamics NAV et Microsoft Dynamics AX. Ce type de programmation permet d'utiliser les tables de la base de données en tant qu'objet de code.

Passons maintenant aux deux systèmes d'opérations d'Apple, à savoir OS X pour Mac et iOS pour smartphone et tablette. Ces deux OS sont codés en Objective-c littéralement OBJECT-ORIENTED EXTENSION OF C. Ce langage est un dérivé du langage de programmation impérative C. En effet il se base sur le langage C mais c'est un langage orienté objet. Pour rappel, plusieurs langages réputés aujourd'hui reprennent des aspects de C. On peut citer par exemple : C++, C#, Java ou encore PHP. On en vient à se demander qu'est-ce que la programmation impérative. Au même titre que la programmation orientée objet, cette méthode est un paradigme de programmation. Sa particularité consiste à décrire des opérations en séquences d'instructions exécutées par un ordinateur pour modifier l'état d'un programme interne. C'est à ce jour le paradigme le plus répandu parmi l'ensemble des langages de programmation. Entre autres la programmation orientée objet tire sa source de ce paradigme on peut dire que c'est un sous-ensemble de la programmation impérative.

Pour en revenir à l'Objective-C, il a été mis en place en 1983. Il a la particularité de distribuer dynamiquement des messages. Il est connu aussi pour sa réactivité et son chargement dynamique. En revanche il ne permet pas l'héritage multiple comme le C++. Au jour d'aujourd'hui ce langage est essentiellement utilisé pour des OS Apple ou des applications Apple. C'est en quelque sorte un langage propriétaire et privé.

Que dire de Google, on peut juste constater l'immensité de ce groupe et sa technologie à point et à jour. Google partie d'un simple moteur de recherche est devenue une figure emblématique de

l'informatique. Ils ne cessent de s'implanter dans les nouveautés pour tout type de domaine lié à l'informatique. On entend également beaucoup parler de la méthode de travail Google, ses locaux uniques et l'importance qui est accordée à ses employés. Quoi qu'il en soit Google est l'un des plus grands influenceurs de la programmation informatique et a permis de mettre en avant beaucoup de langages. À commencer avec Google Search son programme phare depuis toujours.

Google Search est un moteur révolutionnaire qui a guidé notre façon d'utilisation d'internet. En effet, sans moteur de recherche il serait nécessaire de connaître les adresses URL spécifique à chaque site internet pour y accéder. Ce moteur permet de nous guider à l'aide de mots-clefs. Il se repose principalement sur la technologie PageRank qui est un algorithme qui permet d'analyser des liens concourant au système de classement des pages Web. Cet algorithme mesure la popularité d'une page web pour effectuer son propre classement. Google s'est beaucoup servi des technologies Open Source qu'il a contribué à améliorer en retour. Entre autres, son moteur de recherche est basé sur le langage Python qui est un langage de programmation orienté objet, multiparadigme et multiplateforme. C'est un mélange de programmation impérative structurée, fonctionnelle et orienté objet. Python possède une gestion automatique de la mémoire par ramasse-miettes ou plus communément appeler Garbage Collector qui permet de recycler de la mémoire préalablement allouée. Ce langage possède également une gestion d'exceptions qu'on peut retrouver sur des langages comme Java et C# qui permet de capter une exception et éviter de faire planter un programme. Alors pourquoi Python ? En comparaison à PHP par exemple, Python est plus rapide, plus performant et plus flexible en termes de recherches de données. Des sites comme YouTube n'ont pas hésité à passer de PHP à Python. Grosso modo, Python est plus adapté pour le traitement de données massives et la recherche d'informations réactive.

Ce qui nous intéresse ici est de constater que le Géant Google utilise Python pour son moteur de recherche. Cela fait de lui un précurseur de ce langage et de ses paradigmes. Forcément cela motive d'autres sociétés à utiliser ce langage pour des projets similaires.

Chrome est un navigateur web propriétaire développé par Google basé sur le projet libre Chromium fonctionnant sous Windows, Mac, Linux, Android et iOS. L'objectif initial de Google en développant Chrome était de fournir aux internautes un nouveau navigateur plus rapide et proposant plusieurs innovations par rapport à Mozilla Firefox ou Microsoft Internet Explorer. Comme la plupart des navigateurs, Chrome utilise le langage C++.

C++ est un langage de programmation orienté objet (POO) à usage général, développé par Bjarne Stroustrup et c'est une extension du langage C. Il est donc possible de coder C++ dans un "style C" ou un "style orienté objet". Dans certains scénarios, il peut être codé dans les deux cas et constitue donc un exemple efficace de langage hybride.

C++ est considéré comme un langage de niveau intermédiaire, car il encapsule à la fois des fonctionnalités de langage de haut niveau et de bas niveau. Initialement, le langage était appelé "C avec classes" car il avait toutes les propriétés du langage C avec un concept supplémentaire de "classes". Cependant, il a été renommé C++ en 1983.

C++ est l'un des langages les plus populaires principalement utilisée pour les logiciels système / d'application, les pilotes, les applications client-serveur et les micrologiciels intégrés.

Le point fort de C++ est la collection de classes prédéfinies, qui sont des types de données qui peuvent être instanciés plusieurs fois. Le langage facilite également la déclaration des classes définies par l'utilisateur. Les classes peuvent en outre accueillir des fonctions membres pour implémenter des fonctionnalités spécifiques. Plusieurs objets d'une classe particulière peuvent être définis pour implémenter les fonctions dans la classe. Les objets peuvent être définis comme des instances créées lors de l'exécution. Ces classes peuvent également être héritées par d'autres nouvelles classes qui prennent en charge les fonctionnalités publiques et protégées par défaut.

C++ inclus plusieurs opérateurs tels que la comparaison, l'arithmétique, la manipulation de bits et les opérateurs logiques. L'une des caractéristiques les plus attrayantes de C++ est qu'il permet la surcharge de certains opérateurs tels que l'addition.

Quelques-uns des concepts essentiels du langage de programmation C++ incluent le polymorphisme, les fonctions, les modèles, les espaces de noms et les pointeurs.

Exemple de code C++ :

```
#include<iostream>

int main()
{
    std::cout << "Hello, new world!\n";
}
```

Google Chrome n'a pas forcément contribué à l'augmentation de l'utilisation de C++ mais il reste tout de même une référence de renommée. Cela prouve la puissance de ce langage plus de 30 ans après son apparition.

III) Applicatifs serveurs

A. Java

Création

Java est un langage de programmation orienté objet développé par James Gosling et ses collègues chez Sun Microsystems au début des années 1990. Contrairement aux langages conventionnels qui sont généralement conçus pour être compilés en code natif (machine), ou pour être interprétés à partir du code source au moment de l'exécution, Java est destinée à être compilé en un byte code, qui est ensuite exécuté (généralement en compilant JIT) Machine virtuelle Java.

Le langage lui-même emprunte beaucoup de syntaxe de C et C++ mais a un modèle d'objet plus simple. Java n'est apparenté que de manière lointaine à JavaScript, bien qu'ils aient des noms similaires et partagent une syntaxe semblable à C.

Histoire

Java a été lancé en tant que projet "Oak" par James Gosling en juin 1991. Les objectifs de Gosling étaient d'implémenter une machine virtuelle et un langage qui avait une notation C familière mais avec une plus grande uniformité et simplicité que C / C ++. La première implémentation publique a été Java 1.0 en 1995. Elle a fait la promesse de "Write Once, Run Anywhere", avec des runtimes gratuites sur les plates-formes populaires. Il était assez sécurisé et sa sécurité était configurable, ce qui permettait de limiter l'accès au réseau et aux fichiers. Les principaux navigateurs Web l'ont

intégré dans leur configuration standard dans une configuration « applet » sécurisée. Java est devenue populaire très rapidement. Les nouvelles versions pour les grandes et petites plates-formes (J2EE et J2ME) ont été conçues avec l'avènement de "Java 2". Sun n'a pas annoncé de plans pour un "Java 3".

En 1997, Sun a approché l'organisme de normalisation ISO / IEC JTC1 et plus tard l'Ecma International pour officialiser Java, mais il s'est retiré du processus. Java reste une norme de fait propriétaire contrôlée par Java Community Process. Sun met à disposition gratuitement la plupart de ses implémentations Java, avec des revenus générés par des produits spécialisés tels que Java Enterprise System. Sun fait la distinction entre son kit de développement logiciel (SDK) et son environnement d'exécution (JRE) qui est un sous-ensemble du SDK, la principale distinction étant que dans le JRE, le compilateur n'est pas présent.

En 2009, La société Oracle rachète Sun Microsystems et redonne un coup de boost au langage Java.

Philosophie

Il y avait cinq objectifs principaux dans la création du langage Java :

- Utiliser la programmation orientée objet.
- Permettre à un programme d'être exécuté sur plusieurs systèmes d'exploitation.
- Contenir un support built-in pour l'utilisation de réseaux informatiques.
- Conçu pour exécuter du code à partir de sources distantes en toute sécurité.
- Facile à utiliser en sélectionnant ce qui était considéré comme les bonnes parties des autres langages orientés objet.

Pour atteindre les objectifs de prise en charge réseau et d'exécution de code à distance, les programmeurs Java trouvent parfois nécessaire d'utiliser des extensions telles que CORBA ou OSGi.

Orientation objet

La première caractéristique, l'orientation de l'objet ("OO"), fait référence à une méthode de programmation et la conception de la langue. Bien qu'il existe de nombreuses interprétations de OO, une idée distinctive est de concevoir un logiciel de sorte que les différents types de données qu'il manipule sont combinés avec leurs opérations pertinentes. Ainsi, les données et le code sont combinés en entités appelées objets. Un objet peut être considéré comme autonome faisceau de comportement (code) et état (données). Le principe est de séparer les choses qui changent des

choses qui restent les mêmes ; souvent, un changement à une structure de données nécessite un changement correspondant au code qui fonctionne sur ces données, ou vice versa.
Cette séparation en objets cohérents fournit une base plus stable pour la conception d'un système logiciel. L'objectif est de faciliter la gestion de grands projets logiciels, améliorer la qualité et réduire le nombre de projets ayant échoué.

Un autre objectif principal de la programmation OO est de développer des objets plus génériques de sorte que le logiciel peut devenir plus réutilisable entre les projets. Un objet "client" générique, par exemple, devrait avoir à peu près le même comportement entre différents projets logiciels, en particulier lorsque ces projets se chevauchent à un niveau fondamental. Ils le font souvent dans de grandes organisations. Dans ce sens, les objets logiciels peuvent être vus plus comme des composants enfichables, aider l'industrie du logiciel à construire des projets en grande partie à partir de pièces existantes et éprouvées, entraînant ainsi une réduction les temps de développement. L'utilisabilité logicielle a rencontré des résultats pratiques mitigés, avec deux principales difficultés : la conception d'objets vraiment génériques est mal comprise et un Il n'existe pas de méthodologie pour une large diffusion des opportunités de réutilisation. Certains ouverts les communautés sources veulent aider à faciliter le problème de la réutilisation, en fournissant aux auteurs façons de diffuser des informations sur les objets généralement réutilisables et les bibliothèques d'objets.

Indépendance de la plate-forme

La deuxième caractéristique, à savoir l'indépendance de la plate-forme, signifie que les programmes écrits en langage Java doivent fonctionner de manière similaire sur un matériel divers. On devrait être capable d'écrire un programme une fois et l'exécuter n'importe où.

Ceci est réalisé par la plupart des compilateurs Java en compilant le code de langage Java "à mi-chemin" en byte code (en particulier le byte code Java) avec des instructions machine simplifiées spécifiques à la plate-forme Java. Le code est ensuite exécuté sur une machine virtuelle (VM), un programme écrit en code natif sur le matériel hôte qui interprète et exécute le byte code Java générique. En outre, des bibliothèques standardisées sont fournies pour permettre l'accès aux fonctions des machines hôtes (telles que les graphiques, le filetage et la mise en réseau) de manière unifiée. Il faut savoir que bien qu'il existe une étape de compilation explicite, à un moment donné, le byte code Java est interprété ou converti en instructions machine natives par le compilateur JIT.

Mais Java est bien plus qu'un simple langage : c'est la somme de nombreux outils, connus sous le nom de Java Platform. Cet environnement de développement robuste et open source comprend des

bibliothèques, des frameworks, des API, Java Runtime Environment, des plug-ins Java et la machine virtuelle Java (JVM). En bref, les développeurs ont tout ce dont ils ont besoin à portée de main pour créer des applications et des systèmes Web Java.

Dans quels cas faut-il utiliser Java ?

Java est populaire pour les applications de niveau entreprises et idéal pour les sites à fort trafic qui ont besoin d'espace pour se développer. Il s'agit également d'un langage de base du développement d'applications mobiles Android de Google, qui l'a considérablement ramené sur les radars des développeurs.

Java est de « retour ». Bien sûr, beaucoup de développeurs n'ont jamais cessé d'utiliser Java en premier lieu. Mais là où la sélection de plateformes était facultative, surtout parmi les startups et les développeurs web, les plateformes alternatives ont offert plus d'agilité et d'expressivité au cours des 10 dernières années. Les goûts de PHP et de Ruby on Rails permettent aux développeurs de faire beaucoup plus avec moins.

Les temps changent et 16 ans plus tard, ce n'est plus le Java 1.0.

Voici plusieurs raisons pour lesquelles Java est très utilisé.

- Les IDE éliminent la douleur

Eclipse et NetBeans sont des outils incroyablement puissants et peuvent masquer certaines des horreurs admises qu'est la soupe API Java. Sans aucun IDE et toutes les fonctionnalités qui vont avec comme l'auto-complétion et autre le Java serait beaucoup plus complexe.

- Support de la langue

Il n'est pas nécessaire d'écrire Java pour utiliser Java. On peut obtenir tous les avantages de l'exécution de la JVM portable, mais gribouiller dans les environs familiers de Ruby ou Python. Cela peut aussi être plus rapide. Sans oublier les langages plus récents destinés à la programmation moderne tels que Scala, Groovy ou Clojure.

- Android

Qui peut résister au petit robot vert ? Google a fait un choix judicieux en choisissant le langage Java pour alimenter ce qui devient la principale plate-forme de téléphonie mobile au monde. Donc, peut-

être qu'Oracle a un milliard de dollars de bœuf avec cela, mais les programmeurs - nouveaux et chevronnés - prennent Java pour les applications mobiles car il est bien parti pour devenir l'avenir des logiciels grand public.

- Grosse communauté

L'écosystème de Java est une mine d'or. Il y a quasiment une bibliothèque pour tout ce qu'on veut mettre en place. En fait, il y a probablement un projet Apache pour ça. Très souvent, la manière la plus basse de communiquer avec le reste du monde est via une API Java.

- Évolution lente

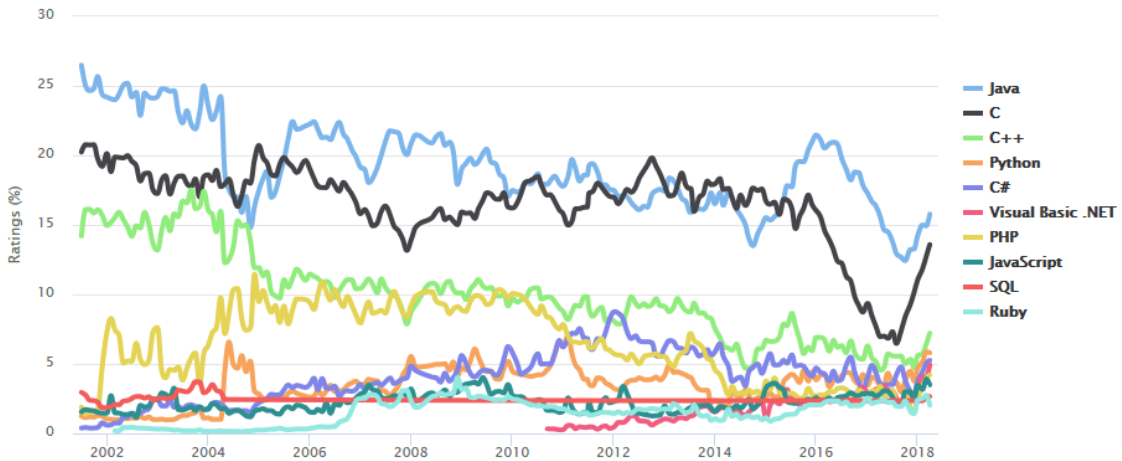
Après la sortie de Java 7 en 2011, Java 8 en 2014 et enfin Java 9 en 2017. On peut dire que le Java n'évolue pas aussi vite que des langages comme JavaScript. Il n'y a pas beaucoup de choses qui ont changé à travers les différentes versions. Si on vérifie régulièrement sur le site Hacker News, on peut vite s'apercevoir que tout logiciel dont la durée de vie dépasse un an devient difficile à maintenir lorsque la plate-forme sous-jacente change constamment.

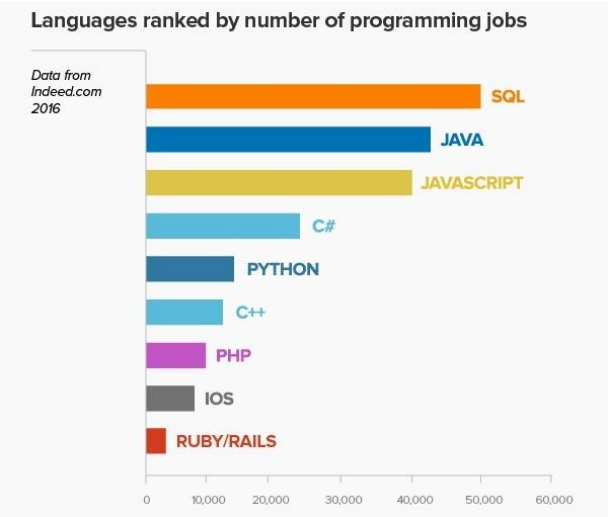
- On finit par l'utiliser de toute façon

La majorité des entreprises viennent à utiliser Java dans le cadre de certains projets. De plus avec la montée en puissance du déploiement et de la programmation vers le cloud, Java excelle dans ce domaine. Twitter l'a découvert et en parlera à OSCON Java. La plateforme cloud de SAP prône également les applications Java.

Popularité

D'après les statistiques TIOBE le Java est leader en popularité depuis 2016 et deuxième en termes d'offres d'emploi.





B. PHP

PHP : Qu'est-ce que c'est ?

PHP Hypertext Preprocessor est un langage de script côté serveur qui est utilisé pour développer des sites Web statiques, dynamiques et des applications Web. Il est considéré comme une des bases de la création des sites web. Les scripts PHP ne peuvent être interprétés que sur un serveur sur lequel PHP est installé. Côté client, les utilisateurs accédant aux scripts PHP nécessitent uniquement un navigateur Web. Un fichier PHP contient des balises PHP et se termine par l'extension ".php".

Qu'est-ce qu'un langage de script ?

Un script est un ensemble d'instructions de programmation qui est interprété lors de l'exécution. Un langage de script est un langage qui interprète les scripts lors de l'exécution. Les scripts sont généralement intégrés dans d'autres environnements logiciels.

Le but des scripts est généralement d'améliorer les performances ou d'effectuer des tâches de routine pour une application.

Les scripts côté serveur sont interprétés sur le serveur tandis que les scripts côté client sont interprétés par l'application cliente.

PHP est un script côté serveur interprété sur le serveur tandis que **JavaScript** est un exemple de script côté client interprété par le navigateur client. PHP et JavaScript peuvent être intégrés dans des pages HTML simultanément.

Script vs Langages de programmation

Langages de programmation	Langages de script
Possède toutes les fonctionnalités nécessaires pour développer des applications complètes.	Principalement utilisé pour les tâches de routine
Le code doit être compilé avant de pouvoir être exécuté	Le code est généralement exécuté sans compilation
N'a pas besoin d'être intégré dans d'autres langues	Est généralement intégré dans d'autres environnements logiciels.

Que signifie PHP ?

PHP signifie - Personal Home Page, mais il désigne maintenant l'acronyme récursif PHP : Hypertext Preprocessor. Le code PHP peut être intégré au code HTML ou il peut être utilisé en combinaison avec divers systèmes de modèles Web, système de gestion de contenu Web et frameworks Web.

Syntax PHP

```php
<?php
    echo 'Hello World';
?>
```

Un fichier PHP peut également contenir des balises telles que HTML et des scripts côté client tels que JavaScript.

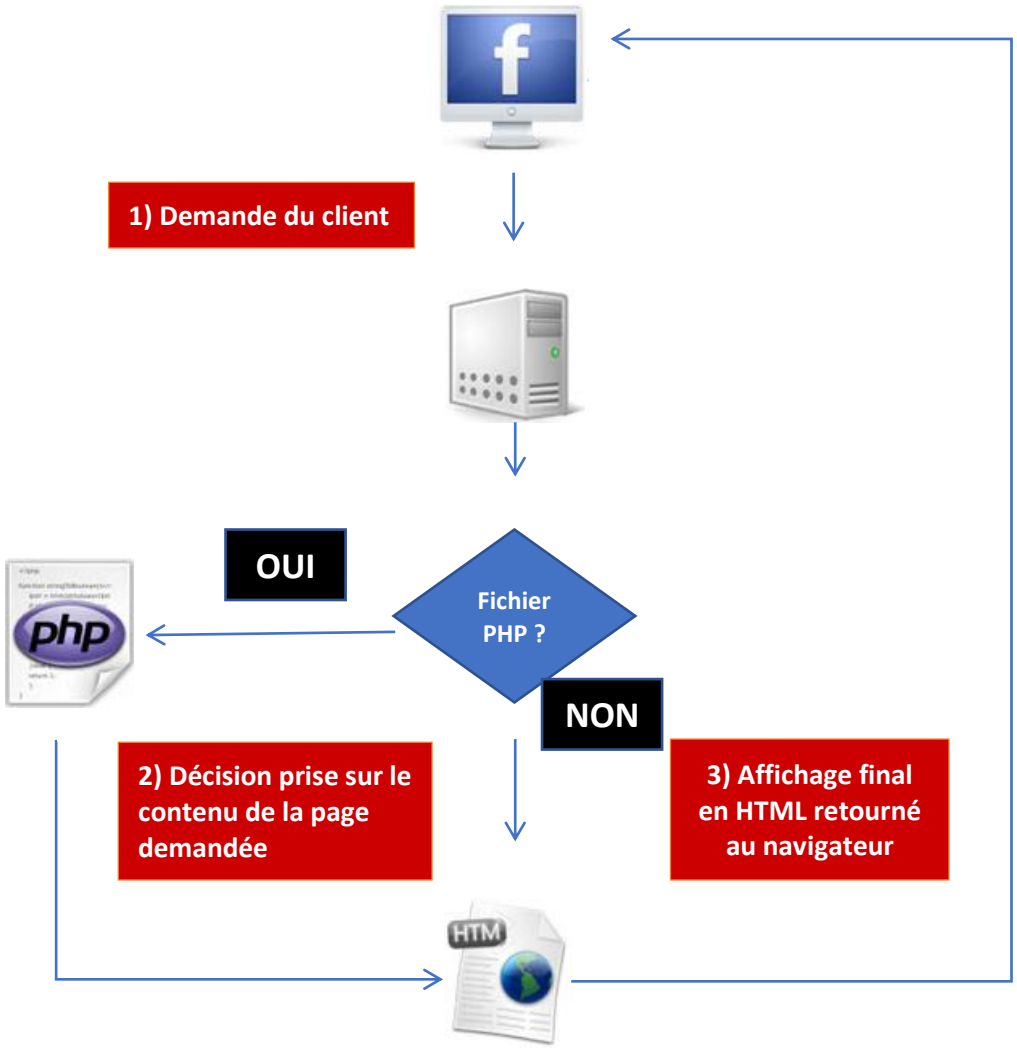
• HTML est un avantage supplémentaire lors de l'apprentissage du langage PHP. Il est possible d'apprendre le PHP sans connaître le HTML mais il est recommandé de connaître au moins les bases du HTML.

• Il est important également de connaître les systèmes de gestion de base de données (SGBD) pour les applications alimentées par base de données.

• Pour les sujets plus avancés tels que les applications interactives et les services Web, il est

nécessaire de se familiariser avec JavaScript et XML.

Le schéma ci-dessous illustre l'architecture de base d'une application Web PHP et la manière dont le serveur gère les requêtes.



Pourquoi utilise-t-on PHP ?

Voici quelques-unes des raisons pour lesquelles PHP est utilisé :

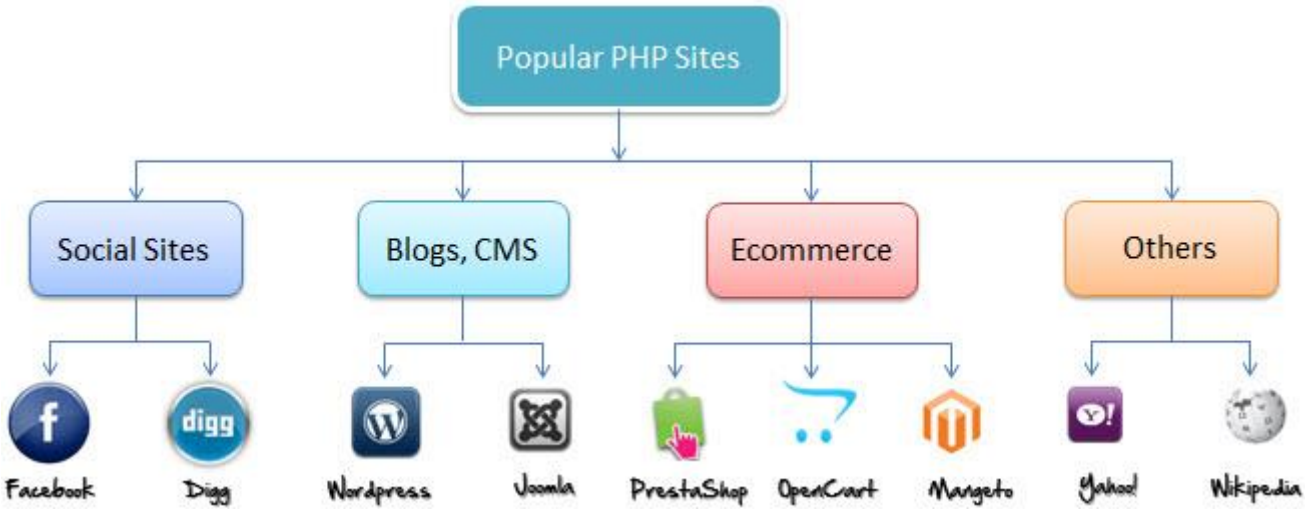
- PHP est open source et gratuit.
- Courbe d'apprentissage courte comparée à d'autres langages tels que JSP, ASP etc.
- PHP possède une grande communauté
- La plupart des serveurs d'hébergement Web supportent PHP par défaut contrairement à d'autres langages tels que ASP qui ont besoin d'IIS (Internet Information Service). Cela fait de PHP un choix rentable.

- PHP est régulièrement mis à jour pour se tenir au courant des dernières tendances technologiques.
- L'autre avantage de PHP est qu'il s'agit d'un langage de script côté serveur. Cela signifie que vous devez seulement l'installer sur le serveur et que les ordinateurs clients demandant des ressources du serveur n'ont pas besoin d'installer PHP ; seul un navigateur web suffirait.
- PHP a construit un support pour travailler main dans la main avec MySQL. Néanmoins vous pouvez utiliser PHP avec d'autres systèmes de gestion de base de données comme :
    - Postgres
    - Oracle
    - Microsoft SQL Server
    - ODBC etc.
- PHP est une plateforme croisée. Cela signifie que vous pouvez déployer votre application sur un certain nombre de systèmes d'exploitation différents tels que Windows, Linux, Mac OSetc.

Dans quels cas utilise-t-on PHP et quelles sont ses parts de marché ?

En termes de part de marché, il existe plus de 20 millions de sites Web et d'applications sur Internet développés en langage PHP.

Voici certains sites populaires qui utilisent PHP：



PHP vs Asp.Net VS JSP

ASP – Active Server Pages, JSP – Java Server Pages

FONCTIONNALITÉ	PHP	ASP	JSP
Courbe d'apprentissage	Courte	Plus long que PHP	Plus long que PHP
Hébergement Web	Pris en charge par presque tous les serveurs d'hébergement	Nécessite un serveur dédié	Assez soutenu
Open source	Oui	Non	Oui
Support des services Web	Aucun framework ou bibliothèques requises	Utilise le framework .NET	Utilise des bibliothèques
Intégration avec HTML	Facile	Assez complexe	Assez complexe
Support de MySQL	Aucun pilote nécessaire	A besoin de pilotes tiers	A besoin de pilotes tiers
Facilement étendu par d'autres langues	Oui	Non	Étendu en utilisant des classes et des bibliothèques Java.

Extensions Fichiers PHP

File extension and Tags In order for Pour qu'un serveur puisse identifier des fichiers PHP, ils doivent être enregistrés avec l'extension "php".

Les anciennes extensions des fichiers PHP comprennent :

- .phtml
- .php3
- .php4
- .php5
- .phps

PHP a été choisi pour travailler avec HTML, il est donc possible d'inclure du code PHP au sein d'un code HTML.

Mais vous pouvez créer des fichiers PHP sans balises html et c'est ce qu'on appelle un fichier PHP pur. Le serveur interprète le code PHP et affiche les résultats sous forme de code HTML dans les navigateurs Web. Afin que le serveur puisse identifier le code PHP à partir du code HTML, nous devons toujours inclure le code PHP dans les balises PHP. Une balise PHP commence par le symbole "<" suivi du point d'interrogation, puis des mots "php". PHP est un langage sensible à la casse, "VAR" n'est pas la même chose que "var". Les balises PHP ne sont pas sensibles à la casse, mais il est fortement recommandé d'utiliser des lettres minuscules.

Ce qu'il faut retenir :

- PHP signifie Hypertext pre-processor
- PHP est un langage de script côté serveur. Les applications clients n'ont pas besoin d'installer PHP.
- Les fichiers PHP sont enregistrés avec l'extension ".php" et le code PHP est inclus au sein de balises.
- PHP est open source et il est multiplateforme.

C. Java & PHP : Manque dans l'écosystème

Malgré la richesse des langages comme Java et PHP il reste tout de même un gros manque dans leur écosystème. Ceux qui font d'eux des langages « incomplets » ou non fullstack.

En effet Java et PHP sont des langages serveurs utilisés uniquement pour gérer le back-end d'une application. Ces langages côté serveur permettent de gérer les informations affichées sur un écran. Ils fonctionnent en convertissant les fichiers HTML dans le serveur en informations utilisables pour le navigateur.

Chaque fois que vous visitez un site Web, votre navigateur fait une demande au serveur qui contient le contenu du site Web. La requête ne prend généralement que quelques millisecondes, mais cela dépend en fin de compte d'une multitude de facteurs (connexion internet, localisation du serveur, etc.).

Le principal manque dans ces langages est la partie côté client. Afin de pouvoir modifier le design d'un site internet une fois la page web chargé il faut pouvoir accéder au DOM (Document Object Model) que nous verrons un peu plus bas. Grosso modo le DOM permet de modifier la structure d'une page web (changement de couleurs, animations, etc.). Actuellement Java et PHP sont incapables d'accéder au côté client. C'est pourquoi il est nécessaire de les combiner avec le JavaScript.

Le JavaScript a été spécialement conçu pour maîtriser le DOM, c'est donc le spécialiste en la matière. L'inconvénient de cette combinaison (Langage serveur + JavaScript) réside dans le fait de devoir maîtriser plusieurs langages afin de réaliser une application web complète.

JavaScript intervient donc pour combler le manque des langages comme Java et PHP. Mais ces dernières années JavaScript, tends de plus en plus à devenir un langage Fullstack (client-side + server-side) notamment avec la plateforme Node.js qui est un serveur d'exécution JavaScript. Elle permet d'exécuter une multitude de framerworks Node.js (Meteor.js, Express.js, Sails.js, …).

L'avantage majeur du FullStack JavaScript est que malgré les spécificités de chaque framework, le langage de base reste le même. Il n'est donc plus nécessaire de maîtriser plusieurs langages pour avoir un site web complet.

Une question se pose, le fullStack JavaScript va-t-il prendre l'ascendant sur le Java et le PHP ? Pour le moment Java reste leader mais en se basant sur la montée en puissance du JavaScript et la bataille des différents framework qui favorisent l'innovation on peut dire que le Java a du souci à se faire.

JavaScript : combleur de manque

I) JavaScript : Qu'est-ce que c'est ?

A. Introduction

JavaScript est un langage de programmation apparu en 1995 pour donner aux navigateurs web leur premier type de fonctionnalité intelligente. À l'époque, les navigateurs Web avaient un besoin urgent d'un langage de programmation, car sans cela, les navigateurs agissaient comme des consoles « idiotes » qui devaient constamment communiquer avec les serveurs Web distants pour modifier leur flux de travail. Avec la prise en charge du langage de programmation, il est devenu possible aux navigateurs d'exécuter des actions localement et dynamiquement - sur un clic d'utilisateur, un survol de la souris ou un raccourci clavier - sans avoir besoin de communiquer constamment avec un serveur Web distant. Ainsi, la première génération de JavaScript est née.

B. Web page Events & DHTML

Une caractéristique distinctive de cette première génération de JavaScript était les événements sur les pages Web. À l'aube du Web, tout ce qui était livré aux navigateurs était constitué de pages Web structurées en HTML (Hypertext Markup Language), ce qui ne laissait aucune possibilité d'effectuer d'autres activités une fois la page Web chargée. JavaScript a ouvert la porte à une percée majeure dans le comportement dynamique, car une page Web pouvait désormais être accompagnée d'une logique de programmation exécutée sur le navigateur d'un utilisateur.
Avec JavaScript, il est devenu possible pour un utilisateur de charger une page, survoler une image, cliquer sur un titre pour déclencher une logique supplémentaire comme présenter un message, changer l'image réelle, changer le texte sous-jacent, etc. Le tout, sans besoin de contacter le serveur d'origine d'où provient la page web (c'est-à-dire que tout a été fait localement).

Le premier évènement JavaScript qui est déclenché lors de l'ouverture d'une page web est **« onload »** associé au chargement de la page - déclaré dans l'élément HTML <body>.

Ensuite, il existe d'autres évènements sans ordre particulier comme les événements JavaScript **« onmouseout »** et **« onmouseover »** qu'on peut associer à un élément HTML <img>. L'événement **« onmouseover »** est déclenché lorsqu'un utilisateur passe son pointeur de la souris sur l'élément <img>, tandis que l'événement **« onmouseout »** est déclenché lorsqu'un utilisateur déplace le pointeur de la souris hors de l'élément <img>. Dans les deux cas, les événements peuvent être liés à des méthodes JavaScript personnalisées en bas de page ou dans un fichier JavaScript. En résumé, ces

événements et la logique JavaScript modifient la source d'un élément HTML lorsqu'un utilisateur déplace le pointeur de la souris sur l'élément en question.

Il existe également l'évènement JavaScript **« onclick »** qui est déclenché lorsqu'un utilisateur clique sur un élément, généralement sur des boutons.

Il y a donc la possibilité de mélanger et de faire correspondre les événements avec la logique de programmation JavaScript qui a donné lieu à un riche mélange de possibilités pour les pages Web. Il est devenu possible d'effectuer des calculs mathématiques, des effets visuels et d'offrir une meilleure expérience utilisateur globale, car il n'était plus nécessaire de contacter un serveur Web pour modifier l'état d'un navigateur.

L'un des premiers noms formels reçus par ces techniques JavaScript était DHTML (Dynamic HTML). DHTML a été classé comme tout ce qui combinait HTML, JavaScript et CSS (Cascading Style Sheets) dans le but de générer des effets dynamiques (exemples : texte animé, boutons de survol, affichage / masquage d'éléments) en fonction des actions de l'utilisateur (par exemple, la souris sur les éléments). En fait, à ce jour, il existe encore des sites qui exécutent et distribuent même DHTML et sont parfaitement compatibles avec les dernières versions de navigateurs.

Bien que DHTML ait été un pas en avant sur les pages Web sans JavaScript, les attentes en matière de pages Web ont augmenté, ce qui est devenu loin d'être idéal. Le plus gros défaut de DHTML était que la charge utile entière de la page Web (c'est-à-dire la logique de programmation JavaScript et les données structurées) soit livrée en une seule étape, avec toute charge supplémentaire nécessaire pour obtenir à nouveau une page Web nouvellement générée.

Avec le développement du Web - avec le commerce électronique, le contenu interactif et les médias vidéo - les attentes sur les navigateurs ont également augmenté et avec celles de JavaScript. Soudainement, la première génération JavaScript qui fournissait des fonctionnalités « modernes » pour l'époque, a commencé à apparaître un peu rude. JavaScript a donc changé, le langage JavaScript lui-même a subi un lifting comme d'autres langages web (HTML 5, CSS 3), d'autres techniques sont apparues pour rendre JavaScript plus puissant et des bibliothèques JavaScript ont été créées pour éviter de réinventer les tâches JavaScript.

C. AJAX & DOM

À partir de 2004, l'utilisation de JavaScript a subi un changement majeur avec une technique appelée AJAX, un acronyme pour « Asynchronous JavaScript and XML ». AJAX a apporté une nouvelle ère de pages Web capables de se mettre à jour dynamiquement sans aucune communication apparente avec leur site Web d'origine. Bien sûr, en arrière-plan, un navigateur a de nouveau établi une communication avec son site Web d'origine, sauf qu'il était maintenant fait de manière asynchrone via JavaScript. De cette manière, le serveur d'origine a répondu avec de nouvelles données en XML ou JSON (JavaScript Object Notation) qui ont ensuite été intégrées dans la page Web de manière transparente (c'est-à-dire sans actualisation de page complète).

AJAX est devenu la norme pour les pages Web gourmandes en données (par exemple, Maps et Boîte mail) où il était impossible de livrer toute la charge utile en une seule étape ou pour les pages Web où les actualisations en pleine page entraînaient des expériences utilisateur terribles. Mais la large utilisation d'AJAX dans toutes les pages Web, sauf les plus simples, a mis en lumière une série de problèmes qui ont affecté les pages Web et JavaScript depuis leur création.

Le premier problème était JavaScript lui-même car la langue était très fragmentée, principalement en raison des guerres de navigateurs. La guerre des navigateurs a été une période de concurrence féroce entre les créateurs de navigateurs pour dominer le marché grâce à des fonctionnalités de navigateur plus performantes. Cela a conduit à des fonctionnalités JavaScript à peine cuites ou innovantes qui ne fonctionnaient qu'avec certains navigateurs. Cette fragmentation signifiait que même la logique JavaScript ne fonctionnait pas nécessairement comme prévu sur tous les navigateurs, ce qui conduisait à des conceptions JavaScript de plus en plus complexes, nécessaires pour que les pages Web fonctionnent de manière égale entre plusieurs navigateurs.

Le second problème était JavaScript basé sur le DOM (Document Object Model) pour accéder et naviguer dans la structure des pages Web. Il n'y a vraiment aucun moyen rapide d'expliquer le DOM. Tous les événements de page Web (onload, onclick) sont techniquement connus en tant qu'événements DOM et les éléments HTML d'une page Web (<img>, <h1>) sont techniquement connus d'un navigateur en tant que nœuds DOM possédant des méthodes et propriétés DOM (innerHTML). L'écriture de la logique JavaScript nécessitait donc une compréhension approfondie du schéma complexe des conventions et des méthodes utilisées par le DOM. Avec AJAX devenant d'autant plus commun pour les pages Web, le DOM représentait simplement un autre obstacle d'adoption au-dessus de la fragmentation JavaScript.

Pour remédier à ces derniers problèmes, les pages Web faisant un usage intensif de JavaScript s'appuyaient sur des bibliothèques pour normaliser le comportement JavaScript dans les navigateurs et faciliter la mise à jour des pages Web sans utiliser directement le DOM. Dans ce domaine, la librairie JavaScript jQuery - créée en 2006 - s'est imposée comme le choix le plus populaire à cet effet, qui à ce jour fait partie des bibliothèques JavaScript les plus utilisées.

D. jQuery

La bibliothèque jQuery JavaScript a atteint une masse critique, car elle a résolu un point important des problèmes JavaScript. Il garantissait que la même logique JavaScript pouvait fonctionner dans tous les principaux navigateurs et fournissait également un moyen plus naturel d'accéder à la structure des pages Web et de la parcourir par rapport au DOM. De plus, grâce à l'adoption généralisée de jQuery, elle a également créé un écosystème florissant de plug-ins et de sous-projets (par exemple jQuery-UI) basés sur jQuery, ce qui a rendu sa proposition de valeur encore plus attrayante.

Mais aussi explosive que l'utilisation de jQuery à ce jour, jQuery est toujours ancrée dans les bases des premières pratiques de conception de pages Web.

Il existe une conception du code jQuery plutôt répandu qui consiste à déclarer les éléments HTML (par exemple <h4>, <button>,) séparément des événements DOM ou des actions logiques JavaScript. Ainsi on n'utilise pas forcément les événements en ligne (onload, onclick) ou les méthodes JavaScript déclarées directement sur les éléments HTML.

Cette conception repose sur le fait que les éléments HTML ont été conçus à des fins de présentation (comment afficher une page Web) et non sur la façon de réagir aux actions dynamiques (par exemple, un utilisateur qui clique ou survole un élément). JavaScript et HTML sont séparés. Cette séparation nécessite l'utilisation de références pour interconnecter des éléments HTML avec leur logique JavaScript. Le principe consiste à renseigner sur les divers éléments HTML un attribut id ou une classe CSS dans l'attribut class, qui sont utilisés à des fins d'identification.

Une fois que les éléments HTML ont des identifiants, il est possible d'y accéder à des fins de manipulation à l'aide du JavaScript. Cela peut être réalisé en ajoutant directement des balises <script> de JavaScript au sein du fichier HTML ou bien de créer un fichier JavaScript et inclure sa référence sur le fichier HTML.

Voici un exemple :
PS : Ce fichier de code provient de la source suivante :

https://plnkr.co/edit/7KABCwV45m7JeT7dh6is?p=info

```
<!DOCTYPE html>
<html>
  <head>
    <title>jQuery - Explication - Mohamed AHARCHI</title>
    <script data-require="jquery@2.2.0" data-semver="2.2.0" src="
    https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
  </head>
  <body>
    <h4>
      <button class="continue" id="button-action">Blank</button>
    </h4>
    <div id="banner-message">Jquery 101!</div>
    <h4>
      <button id="ajax-call">Click to update message via AJAX</button>
    </h4>
    <script>
      $(document).ready(function() {
        console.log("ready!");
        var bannerMessage = $("#banner-message");
        // DOM Traversal and manipulation
        $("button.continue").text("Click to hide " + bannerMessage.html());
        // Event handling
        $("#button-action").on( "click", function(event) {
          bannerMessage.toggle(function() {
            if ($(this).is(":hidden")) {
              $("#button-action").text("Click to show " + bannerMessage.html());
            } else {
              $("#button-action").text("Click to hide " + bannerMessage.html());
            }
          });
        });
        // AJAX
        $("#ajax-call").on("click", function(event) {
          $.ajax({
           url: "https://jsonplaceholder.herokuapp.com/posts/1",
           success: function(data) {
            console.log(data)
              bannerMessage.html(data.title);
           }
          });
        });
      });
    </script>
  </body>

</html>
```

Nous pouvons voir sur cet exemple, la syntaxe jQuery **<element_type>.<css_class>** utilisée pour accéder aux éléments HTML. Une fois les références jQuery établies pour les éléments HTML, il est

possible d'associer des événements (par exemple, cliquer, basculer) à des éléments qui déclenchent des actions logiques (par exemple mettre à jour du texte, effectuer un appel AJAX).

On remarque également comment toutes les actions JavaScript de l'exemple mettent à jour l'élément lorsque quelque chose se produit dans d'autres éléments HTML. En outre, il faut regarder la dernière déclaration JavaScript qui est un appel AJAX fournie par la méthode intégrée « .ajax » de jQuery qui vit également séparément de son élément HTML et s'appuie sur de nombreuses références pour faire son travail.

Maintenant, avec un peu de recul on peut très vite le bon et le mauvais côté de ce design. La bonne partie est la conception qui est plus simple et plus propre que l'utilisation du DOM. Avec jQuery, les gestionnaires d'événements et la logique ne sont plus dispersés avec des éléments HTML, mais sont clairement séparés. Un autre aspect positif de jQuery est comment il accède aux éléments HTML. Au lieu de travailler directement avec les méthodes et les propriétés DOM (par exemple getElementById(), parentNode(), childNodes) difficiles à maîtriser, jQuery prend en charge des méthodes plus simples et plus puissantes pour accéder aux éléments HTML (par exemple valeur, par valeur d'attribut de classe CSS).

La mauvaise partie de l'approche de jQuery est qu'il peut devenir trop complexe pour déchiffrer et déboguer des pages Web avec des dizaines ou des centaines d'éléments et d'actions. Ainsi, par exemple, ce n'est pas l'utilisation d'un ou deux appels AJAX dans ce format qui pose des problèmes, mais plutôt l'utilisation dans des pages Web qui nécessitent des dizaines ou des centaines d'interactions AJAX, où la gestion des références, des mises à jour et le traitement des erreurs de cette manière peut rapidement devenir difficile à comprendre.

E. Les composants JavaScript

Les composants sont un ancien concept d'ingénierie logicielle basé sur des promesses de logiciels réutilisables qui encapsulent la logique et les données. La POO (Programmation orientée objet), utilise également de concept de composants. Les deux concepts (POO, JavaScript Components) partagent des caractéristiques de conception communes (par exemple, l'utilisabilité, l'encapsulation). Alors qu'il était possible de créer des composants JavaScript depuis les premières années de la langue, la pratique n'a jamais vraiment décollé tant qu'une série de frameworks n'a pas facilité et banalisé l'utilisation des composants JavaScript.

Le principal avantage des composants JavaScript est qu'ils permettent d'intégrer le HTML et le JavaScript dans la même unité et de ne pas les utiliser séparément, contrairement au processus illustré dans l'exemple jQuery précédent. Avec les composants JavaScript, il devient également possible de réutiliser la même unité plusieurs fois dans la même application ou dans plusieurs projets. De plus, les composants JavaScript, offrent également la possibilité de créer des relations parent-enfant entre les composants afin que les actions et les données soient partagées. Enfin, avec les composants JavaScript, il devient plus facile de tester le comportement de l'interface utilisateur, puisque la présentation et la logique sont encapsulées dans la même unité.

Bien que le but des composants JavaScript soit clair, les composants JavaScript de mise en œuvre varie parfois largement en fonction de la bibliothèque ou du cadre sous-jacent d'où ils proviennent. Tout comme la sélection d'autres technologies Web, les composants JavaScript, nécessitent de consacrer du temps à l'apprentissage des subtilités qui ne fonctionnent que dans le contexte de certaines bibliothèques ou frameworks. Les approches les plus populaires des composants JavaScript sont les suivantes :

- **React :** Développé par Facebook, React est l'un des leaders incontestés dans ce domaine, principalement parce qu'il est conçu uniquement dans le but de créer des interfaces utilisateur. React complémente davantage de structures JavaScript riches en fonctionnalités (par exemple Angular, Node.js).

- **Angular :** Développé par Google, Angular est un autre concurrent sérieux dans cet espace. Cependant, contrairement à React, Angular est un framework MVC à part entière, ce qui signifie qu'en plus d'avoir ses propres composants JavaScript, Angular offre également une large gamme d'add-ons comme des contrôleurs et des liaisons de service pour les composants.

- **Web Components (http://webcomponents.org/) :** Bien que n'étant pas un type de composant JavaScript, il s'agit d'une norme émergente qui vise à unifier les différences entre les composants JavaScript. Il vaut la peine d’y jeter un oeil principalement à cause de ce qu'il vise à réaliser.

F. JavaScript MVC

Avec la montée en puissance d'AJAX, de nombreux concepteurs ont compris qu'il était possible de fournir une seule page Web et de simuler la fonctionnalité de dizaines de pages Web via AJAX, offrant ainsi à l'utilisateur final une expérience plus épurée plutôt que de passer de la page Web à la page Web. Cela a ouvert la voie à des applications de page unique ou connu aussi par leur acronyme SPA (Single Page Applications).

Comme d'autres demandes, SPA a vraiment repoussé les limites de ce qui était possible avec JavaScript. En reprenant l'exemple jQuery précédent qui est une page Web bac à sable qui utilise un couple de gestionnaires d'événements JavaScript. Imaginons maintenant une page Web réelle pour afficher un catalogue de produits, si une page Web bac à sable a quelques actions JavaScript, une page Web reflétant un processus de la vie réelle peut avoir des dizaines ou plus d'actions JavaScript. La gestion de cette quantité d'actions JavaScript est assez faisable avec jQuery, même si vous êtes susceptible de ressentir certaines des contraintes de conception comme la difficulté de débogage ou encore la gestion des erreurs.

Si on part du principe que nous réalisons un SPA. Cette page Web unique de catalogue de produits avec une douzaine d'actions JavaScript sera maintenant regroupée avec une page Web de vente qui a également une douzaine d'actions JavaScript ou plus, ainsi qu'une page web de caisse avec une douzaine d'actions JavaScript ou plus. À la fin, on est susceptible d'avoir le scénario que nous avons évoqué plus haut quand on fait référence aux limites de jQuery : des centaines d'éléments et d'actions tous regroupés dans une seule page.

Alors, comment peut-on gérer de façon réaliste des centaines d'éléments et d'actions tous regroupés dans une seule page avec JavaScript ? En principe, c’est plutôt simple. Il suffit de tout organiser avec une approche éprouvée. Il s'est avéré que depuis l'avènement des sites Internet, des piles de serveurs étaient utilisées dans des langages tels que PHP, Java et Python, pour concevoir des logiciels comme des pièces interconnectées qui favorisent la maintenance et la réutilisation.

Ces modèles de conception comprenant des éléments tels que le routage des pages Web (séquence de pages à suivre pour une tâche donnée), l'exécution de la logique métier et la validation des données sont connues sous le nom d'architecture Model-View-Controller (MVC).

Face aux complexités du SPA, JavaScript a ainsi pris les devants et adopté des fonctionnalités telles que le routage et l'injection de dépendances qui étaient autrefois réservées aux langages côté

serveur comme PHP, Java & Python. Et juste comme ça, les bibliothèques JavaScript sont rapidement devenues des frameworks JavaScript à part entière capables de gérer entièrement l'architecture MVC sur le navigateur.

Les frameworks MVC JavaScript côté client les plus populaires sont :

- **Angular :** Développé par Google, Angular est l'un des leaders incontestés dans ce domaine. Angular a été initialement publié en 2010, ce qui en fait l'un des premiers frameworks MVC JavaScript. En outre, Angular est maintenant dans sa deuxième version majeure - Angular 2+ incorporant un grand nombre de « leçons apprises » de sa version initiale appelée AngularJS. Notez que les versions Angular 2+ sont simplement appelées Angular que ce soit Angular 2.0 ou Angular 4.0.

- **Ember :** Bien que pas aussi populaire que Angular, Ember est une autre alternative de framework MVC JavaScript. Semblable à Angular, Ember a ses propres composants JavaScript, contrôleurs et équipements de modèle.

II) L'intégration de JavaScript côté serveur (Node.js)

A. Introduction

De toutes les nouveautés JavaScript potentielles, celle qui peut être la plus surprenante pour les vétérans du développement web est l'idée d'exécuter JavaScript sur un serveur. Même si certains développeurs ne sont pas d'accord avec l'idée des composants JavaScript ou JavaScript MVC, ils ne peuvent pas nier que les deux sont toujours liés au navigateur où JavaScript a toujours vécu et dominé.

Alors, pourquoi quelqu'un voudrait-il exécuter JavaScript sur un serveur avec autant de choix de langage de serveur solide comme PHP, Java et Python ? Il s'avère que quelqu'un a rencontré une interface utilisateur qui ne répondait pas lors du téléchargement d'un fichier et a fini par exécuter JavaScript sur un serveur pour obtenir des mises à jour en temps réel. Plusieurs années après cette expérience, JavaScript côté serveur est maintenant monnaie courante.

B. Concept

JavaScript sur la partie serveur est dominé par Node.js, une plate-forme basée sur le même moteur V8 utilisé par le navigateur Chrome de Google. Ce qui est intéressant à propos de Node.js, ce n'est

pas sa capacité à exécuter JavaScript côté serveur, mais plutôt sa capacité à fournir un comportement asynchrone et non bloquant, ce qui en fait un choix idéal pour les applications web en temps réel.

On ne peut pas définir Node.js comme un Framework car il possède plusieurs modules. Node.js se définit lui-même comme une plateforme, ce qui en fait beaucoup de choses. Node.js est un environnement d'exécution pour les applications serveur JavaScript, c'est-à-dire comme un serveur web classique, il utilise simplement une conception de boucle d'événements qui le rend différent et idéal pour les applications en temps réel. Node.js a également sa propre API pour des fonctionnalités comme les requêtes HTTP, les tampons de traitement et les requêtes DNS, entre autres, cela signifie que c'est aussi un Framework. De plus, Node.js est également livré avec un gestionnaire de paquets appelé **npm**, ce qui signifie qu'il contient beaucoup de modules, bibliothèques et frameworks. Plusieurs milliers de paquets sont écrits pour Node.js.

C. NPM (Node Packaged Modules)

Les développeurs Node.js remarquent rapidement, que Node.js seul est assez limité. Si l'on souhaite accéder à d'autres fonctionnalités, il va falloir installer des modules. Car la force de Node.js provient en réalité de sa multitude de module développée par la communauté et du gestionnaire de paquets officiel de Node.js « NPM ».

NPM est l'un des points clefs qui fait le succès de Node.js. Il est directement intégré à la plateforme Node.js et permet en quelques lignes de commande d'installer une multitude de modules. Il est disponible sur https://www.npmjs.org/

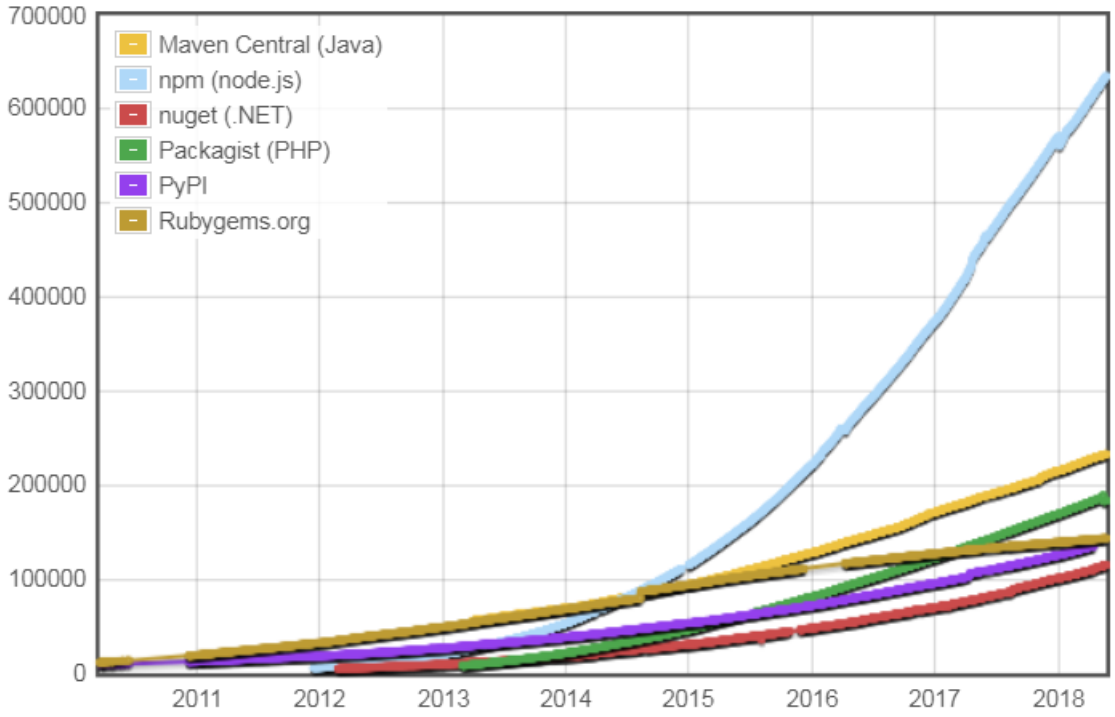
D. Nombre de module et évolutivité

Le nombre et la diversité des modules pour une plateforme à une certaine importance, car cela va déterminer la quantité de fonctionnalités qui n'auront pas à être développées pour un projet. C'est d'autant plus important pour une technologie comme Node.js qui est assez limité sans cela.

Mais afin de pouvoir juger si le nombre de modules disponible pour Node.js est notable, il faut pouvoir comparer cela avec d'autres bibliothèques.

Voici le récapitulatif des données récupérées sur « modulecounts.com », site qui enregistre chaque jour l'évolution du nombre de modules des bibliothèques de modules les plus importantes :

Source : http://www.modulecounts.com/



Gestionnaire de paquets	Maven Central (Java)	npm (node.js)	nuget (.NET)	Packagist (PHP)	PyPI (Python)	Rubygems (Ruby)
Nb module 2015	Env. 90 000	Env. 110000	Env. 30000	Env. 50000	Env. 50000	Env. 90000
Nb module 2016	Env. 120000	Env. 215000	Env. 45000	Env. 90000	Env. 80000	Env. 110000
Nb module 2017	Env. 170000	Env. 380000	Env. 80000	Env. 120000	Env. 98000	Env. 125000
Moyenne de module par jour : 05/18	139/jour	530/jour	90/jour	69/jour	93/jour	23/jour

Après une petite analyse de ce tableau, on constate d'une part que le nombre de module développé pour Node.js en 2017 est loin devant les bibliothèques les plus importantes qui sont respectivement Maven Central pour le langage Java et Ruby pour Ruby. On constate aussi dans un second temps que l'évolution du nombre de module est de loin la plus importante sur Node.js.

Cela étant, l'ajout d'un nouveau module dans NPM est extrêmement simple et pourrait expliquer le nombre aussi important de modules proposaient par la communauté Node.js.

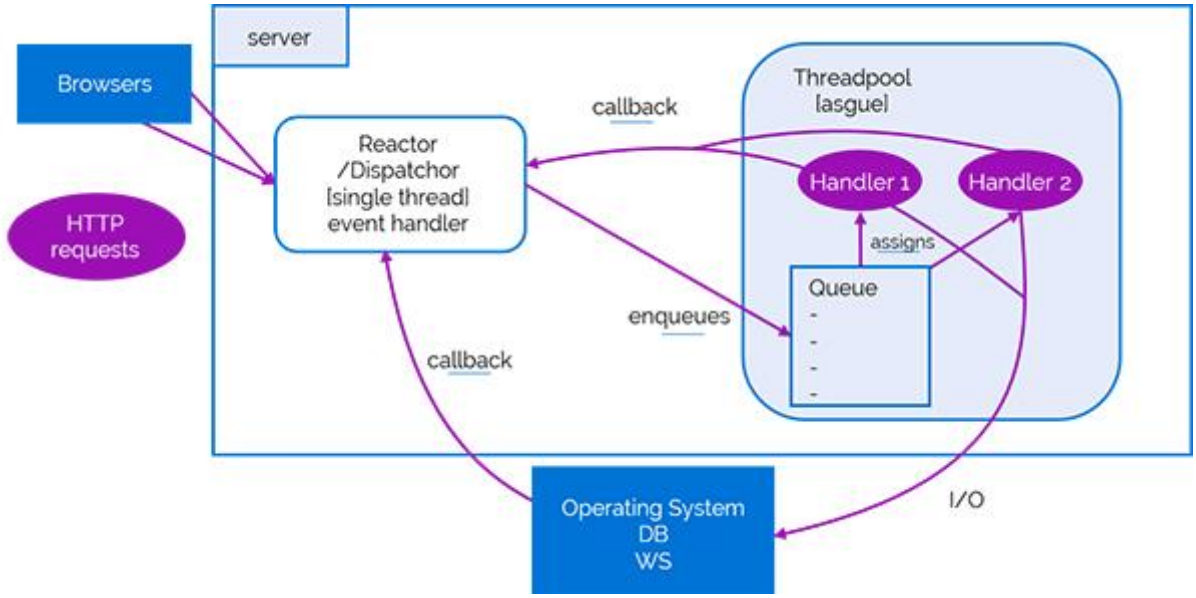
De plus, on se limite ici au gestionnaire de paquets, de nombreux sites pour les différents langages que nous venons de comparer utilisent d'autres types de diffusion, surtout pour les plus gros frameworks qui possèdent leur propre moyen de diffusion.

E. Aspect Technique

Introduction

Comme il a déjà été dit, Node.js est une plateforme de développement web basé sur le moteur V8 de Google et ainsi exploitant le langage JavaScript. Mais qu'est-ce que cela implique au final ? Quel est l'intérêt ? Voici une explication sur le fonctionnement et les raisons pour lequel cette technologie a de l'intérêt. Dans un premier temps, parlons du fonctionnement asynchrone de Node.js puis un point sur l'un des frameworks les plus utilisés de Node.js qui lui permet de faire du temps réel via les websockets.

Moteur V8 de Google :



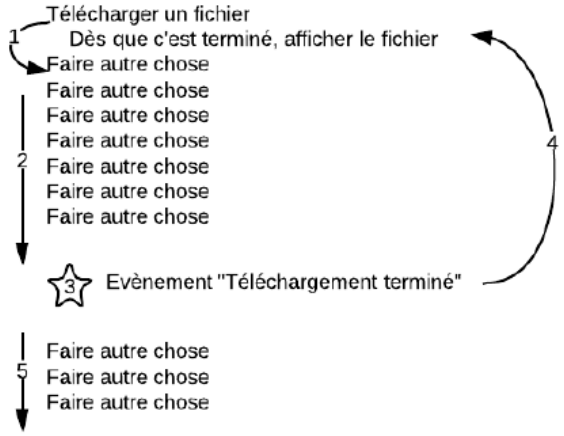
Non bloquant

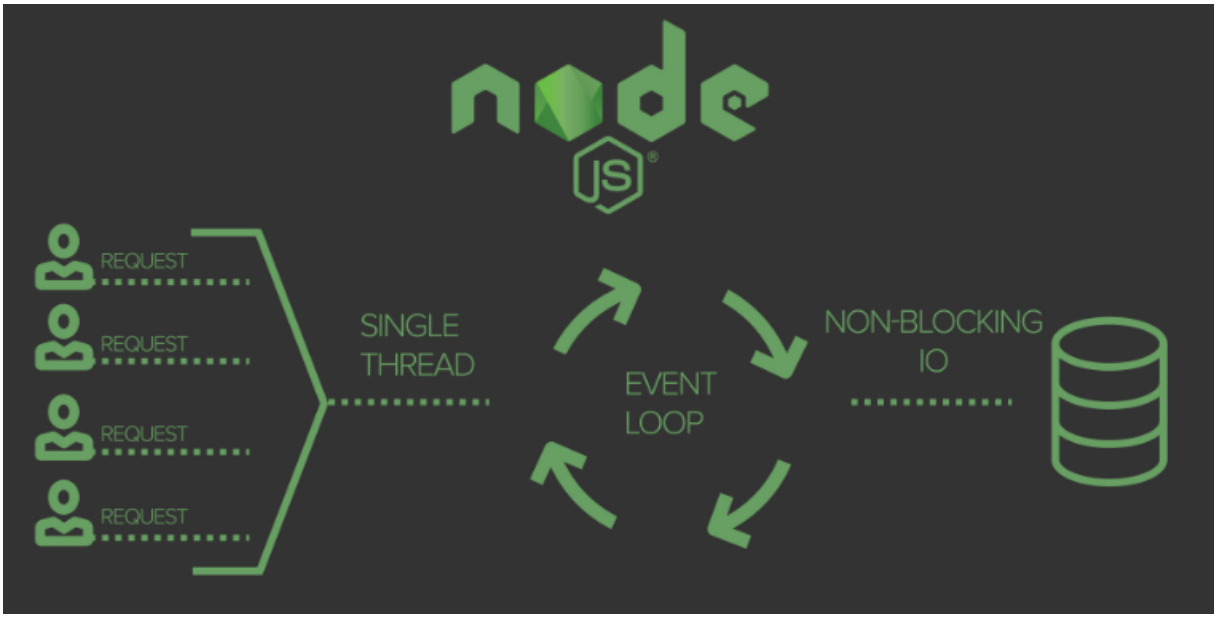
Node.js fonctionne avec un système d'Entrée / Sortie Non bloquant. Ce mode de fonctionnement dit asynchrone est très employé en JavaScript. Coté client, on l'utilise fréquemment pour communiquer avec le serveur, via des requêtes AJAX (Asynchronous JavaScript and XML). C'est l'une des fonctionnalités de Node.js qui le rend si performant. Pour bien comprendre l'intérêt de ce fonctionnement, il faut tout d'abord comprendre la différence entre le mode bloquant (synchrone) et le mode non bloquant.

Un serveur en mode bloquant, va lorsqu'il sera sollicité par une requête, affecter un thread au traitement à réaliser. Celui-ci va interroger une ressource (une base de données, une API, le disque dur, etc..) puis va attendre sa réponse. Pendant toute la durée d'attente du thread, celui-ci ne fera rien et tout l'espace mémoire qui lui est affecté sera verrouillé. Le serveur aura toujours la possibilité

de créer de nouveaux threads pour gérer de nouvelles requêtes, mais il lui faudra alors payer le coût en espace mémoire.

Alors que sur un serveur en mode non bloquant, lorsque le serveur sera sollicité, il affectera à son tour un thread à cette requête qui va interroger la ressource, mais le thread n'attendra pas la réponse. On va laisser le système gérer l'attente du retour qui nous alertera au moment de sa réception. Ainsi, on pourra alors s'occuper de cette réponse. Pendant ce temps, le thread va s'occuper d'autres requêtes. Ce système permet d'optimiser les ressources.





Un exemple du fonctionnement en mode non bloquant :
1. Le thread lance le téléchargement d'un fichier sur un disque dur

2. Pendant ce temps, le thread s'occupe d'autres requêtes

3. Le serveur reçoit un événement lui indiquant que le téléchargement est terminé

4. Le thread s'occupe du traitement qui suit le téléchargement

Puis il continue à traiter le reste des actions

Websocket

Introduction

Websocket est un protocole assez récent qui n'a été normalisé que récemment par l'IETF. Ainsi ce protocole permet la communication entre les navigateurs et les serveurs web via un canal de communication bidirectionnel et full-duplex sur un socket TCP.

Ce protocole étant assez récent, il n'est pas interprété par tous les navigateurs. Ainsi Microsoft n'exploite ce protocole que depuis la version 10 d'internet explorer. C'est dans ce contexte que Socket.io sortie sa première version stable en avril 2012. Afin d'être utilisable par la plupart des navigateurs, les développeurs du module Socket.io auront la bonne idée de mettre en place des alternatives pour les anciens navigateurs. Pour se faire, Socket.io va être capable de simuler les websockets en s'adaptant au navigateur et aux solutions installées sur le poste client. Pour cela, il va s'appuyer sur la liste des technologies suivantes :

- WebSocket
- Adobe Flash Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Grâce à ces différentes techniques de communication, socket.io supporte un très grand nombre de navigateurs, même anciens :

- Internet Explorer 5.5+
- Safari 3+
- Google Chrome 4+
- Firefox 3+
- Opera 10.61+
- Safari sur iPhone et iPad
- Le navigateur Android

Temps réel

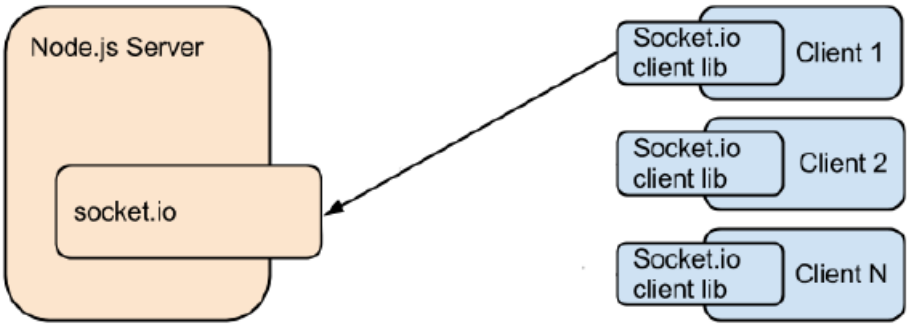
Node.js via le protocole websocket, va pouvoir exploiter un nouveau concept, le temps réel. Mais en quoi consiste ce concept ?

Pour faire simple, le serveur et les postes clients sont maintenant connectés en permanence. Ainsi le serveur est maintenant capable d'envoyer des informations aux clients quand il le désire, il n'est plus nécessaire que ce client lui transmet une requête pour que le serveur puisse lui transmettre des informations. Il s'agit du principe de « push ».

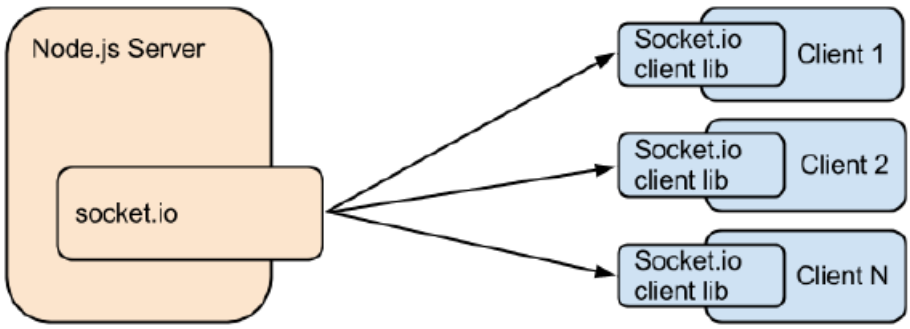
Ainsi comme dans l'exemple ci-dessous, une action est réalisée par un client et par la suite des informations peuvent être retransmises à tous les clients connectés.

Communication Client/Serveur via Socket.io

Un client va transmettre des informations au serveur Node.js via une websocket.



Node.js va interpréter ses données et transmettre une information à tous les utilisateurs actuellement connectés.



Cas d'utilisation

Ce concept de temps réel va être très intéressant pour plusieurs types d'utilisation. Tout d'abord dans tout ce qui va requérir une communication temps réel entre les différents utilisateurs, par exemple, les messageries instantanées.

Un des autres axes où le temps réel est très utilisé, est dans les jeux multijoueurs, où chaque client va devoir transmettre en temps réel les actions aux autres joueurs.

En fait, le temps réel va être exploité dès qu'il s'agit de développer des fonctionnalités orientées multi-utilisateur.

On voit apparaître de nombreux outils collaboratifs fonctionnant avec cette fonctionnalité.

Il y a encore beaucoup de choses à faire, ce concept n'a que 2 ans, Node.js ouvre de nouvelles possibilités pour les sites et applications web.

F. JavaScript Fullstack

Introduction

La plateforme Node.js permettant de développer en JavaScript coté serveur, permet une chose unique dans le développement d'application web. Il donne enfin la possibilité de développer avec un seul langage (hormis le HTML et CSS), qu'il s'agisse de la partie client, du code coté serveur, comme des requêtes à la base de données. Cette possibilité offerte par Node.js, permet aux développeurs qui vont travailler sur ce type de projet de n'avoir besoin que de l'expertise d'un seul langage. De plus la communication entre le client et le serveur va s'en trouver faciliter car utilisant la même syntaxe de communication. Ainsi, le défaut majeur des projets web qui se veut être la lenteur de ces projets par rapport aux applications classiques se veut grandement réduit.

JavaScript côté client

Il existe maintenant de nombreux frameworks qui exploitent les possibilités du langage et permettent de créer un code plus structuré. Ils sont totalement dissociés de Node.js mais peuvent être combinés avec. Voici différentes solutions existantes en JavaScript côté client pour le développement d'application web.

Angular.js

Commençons avec Angular.js que nous avons brièvement vu auparavant. Angular.js est le Framework JavaScript de Google, la première version de ce framework est sortie en 2009. Comme tous les frameworks, il comble les lacunes de JavaScript et implémente de nouvelles fonctionnalités au langage. Mais plus particulièrement, il apporte une manière un peu différente de développer une application web.

Angular.js est conçu pour développer des applications SPA (Single Page Application), il s'agit d'applications où il n'y a pas de rafraîchissement de page, tout est intégré dans une et unique page web. À la différence des autres frameworks, le développeur devra éviter la manipulation du DOM (Manipulation des éléments d'une page web : image, bloc, texte, etc.). Angular.js va permettre d'ajouter des attributs aux balises de notre page HTML pour pouvoir y intégrer des interactions entre les éléments sans avoir besoin de les développer en JavaScript (Ainsi vous pourrez directement sur la structure HTML de votre page, indiquez à un bouton de faire afficher à la page un élément lorsqu'on cliquera dessus).

Le code JavaScript jouera le rôle du contrôleur et fera le lien entre la vue HTML et les requêtes envoyées au serveur. Ce système réduit grandement la taille du code d'un projet JavaScript et simplifie la lecture du code. Ce code sera structuré en module, pour séparer les différents traitements effectués sur la page.

L'autre force d'Angular.js qui en fait le Framework le plus utilisé actuellement, est sa testabilité. Angular.js grâce à son fonctionnement modulaire permet de mettre en place très efficacement des tests unitaires, ce qui est souvent très compliqué en JavaScript.

Backbone.js

Le framework Backbone crée en 2010 est l'un des frameworks les plus légers de sa catégorie. C'est sa légèreté et sa faible dépendance à d'autres bibliothèques qui ont fait sa popularité. Il n'est ainsi dépendant que de « Underscore.js » qui est une librairie développée par la même équipe. De nombreuses applications populaires ont été développées sur ce Framework : Twitter, Foursquare, LinkedIn. Ce Framework se veut spécialiser dans le développement d'application à page unique.

L'un des principaux défauts de Backbone est qu'il est souvent considéré comme étant assez limité dans ces fonctionnalités. Mais il est difficile de pallier légèreté et richesse d'un framework.

Ember.js

Ember.js a été développé pour sa part en 2011, il est basé sur un autre projet de Framework nommé « Sproutcore » développé par Apple. Initialement Ember.js se nommait « Sproutcore 2 ».
Il est le plus imposant des Frameworks cités dans cette liste. Très riche, il est même considéré comme trop complexe et donc peu adapté à de petits projets. Ember.js est un Framework MVC complet et aidera à bien structurer une application.

Knockout

Knockout a été développé en 2010, il a été conçu et est maintenu par un développeur de chez Microsoft. Knockout est un Framework MVVM, qui repose sur le databinding, afin d'avoir toujours les informations à jour sur la vue. Il s'agit de lier des éléments d'une page web à des objets côté JavaScript, les informations de la page s'actualiseront automatiquement lors de la modification des objets.

Ces différents Framework vont permettre aux développeurs de structurer leur code coté client et de transmettre au serveur des données au format « json », ce format est concurrent au XML, mais plus adapté dans le cas d'une application full JavaScript, vu que le json n'est qu'un simple objet « JavaScript ».

JavaScript côté serveur

Pour la partie serveur, sur une application full JavaScript, il est possible de travailler avec du Node.js. Le fait d'être sur du code JavaScript coté serveur, va simplifier la réception des données. Car les échanges se feront au format « json » qui n'est qu'un simple objet JavaScript, cela évite donc de manipuler les données pour les adapter au langage du serveur. L'un des autres points importants coté serveur, c'est la possibilité d'utiliser certains des frameworks client du côté de Node.js afin de structurer le code. Une explication plus détaillée du JavaScript côté serveur est disponible dans la partie **« Frameworktisation de JavaScript serveur (Node.js) ».**

Base de données

Enfin le dernier point important dans le développement d'une application full JavaScript est tout simplement la communication avec la base de données. Fort heureusement, avec Node.js et les bases de données NoSQL (Not only SQL), il est possible de faire des requêtes en JavaScript. Mais il est tout de même possible de travailler sur des bases SQL.

MongoDB

MongoDB est une BDD (base de données) de type NoSQL, elle a été développée en 2007 par 10gen. Il s'agit de la base de données la plus utilisée avec Node.js, elle permet grâce à de nombreux modules dont « mongoose » d'effectuer des requêtes à la base de données en JavaScript. C'est une base de données dite orienter document, car elle permet de stocker des objets type « json » ce qui est impossible en SQL. Pour un projet en full JavaScript c'est un avantage incontestable, plus besoin de lourdes requêtes pour stocker toutes les informations d'un objet JavaScript, on peut ainsi le stocker immédiatement sans traitement préalable.

Mongoose l'un des modules Node.js les plus populaires pour MongoDB, permet de tester les modifications faites sur la base de données à la volée. Ceci afin de vérifier si les informations sont transmises au bon format. Ainsi il y a la possibilité de tester si le type de la valeur correspond à au type attendu, ainsi que sa taille ou son format s'il s'agit d'une chaîne de caractères (dans le cas où un numéro de téléphone (donc 10 chiffres) est attendu, ou si un mot de passe avec un minimum de complexité est attendu par exemple).

Autres bases NoSQL

Node.js fonctionne avec bien d'autres bases NoSql, comme Redis, Apache couchDB ou DynamoDB, de nombreux modules existent sur NPM pour travailler sur ces différentes bases. Chaque base de données à ses particularités, mais elles ont toutes l'avantage de pouvoir travailler directement avec des objets « json » et donc du JavaScript, ce qui est un réel gain de temps.

Bases SQL

Node.js permet tout de même de travailler sur des bases de données SQL, il existe de nombreux modules permettant de se connecter sur les différentes bases de données existantes. La plupart des modules proposent une syntaxe à la mongoose, c'est-à-dire via des fonctions JavaScript plutôt que des requêtes complètes en SQL. Le seul problème viendra lors de l'insertion des données JavaScript dans la base, ils ne pourront pas être stockés dans un format « json », il sera donc obligatoire d'extraire les données pour les intégrer dans les différents champs de la table.

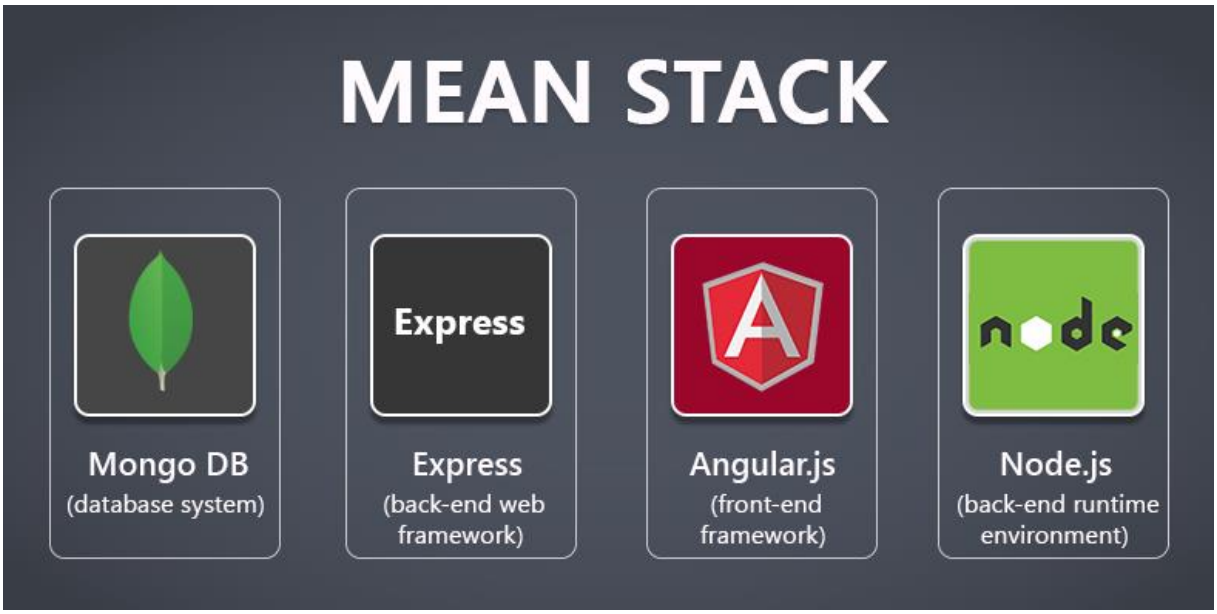
Remarques

Il est également important de noter que Node.js complète réellement les frameworks JavaScript. Par exemple, il est possible d'utiliser avec Node.js, les frameworks MVC JavaScript client Angular et React simultanément ou individuellement. Cependant, il faut savoir que Node.js supporte des milliers d'autres paquets JavaScript qui peuvent être déroutants au début. Par exemple, on peut trouver des paquets Node.js comme Sails et Express qui sont aussi des frameworks MVC JavaScript que nous verrons un peu plus loin. Ces derniers paquets sont des frameworks côtés serveur MVC JavaScript, à ne pas confondre avec les frameworks côtés client MVC JavaScript comme Angular ou Ember. Il est donc possible de finir par utiliser de nombreux frameworks dans la même application (par exemple, React pour les composants JavaScript, Angular pour JavaScript MVC sur le client, Node.js pour l'exécution du serveur JavaScript et Express pour JavaScript MVC sur le serveur).

Solution envisageable : MEAN

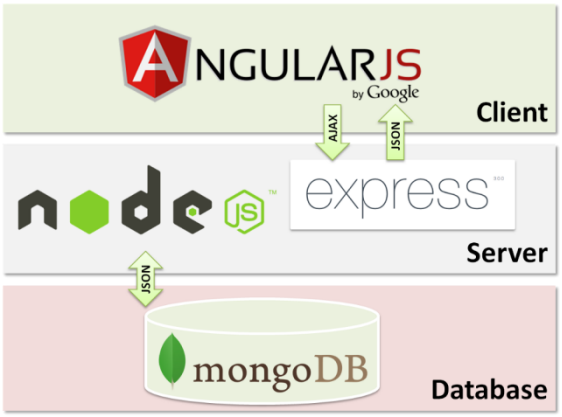
Pour créer un projet JavaScript full-stack, plusieurs solutions existent déjà sur le marché. L'une d'entre elle est la solution MEAN stack.

Le terme MEAN stack fait référence à une collection de technologies basées sur JavaScript utilisées pour développer des applications Web. MEAN est un acronyme de MongoDB, ExpressJS, AngularJS et Node.js. Du client au serveur à la base de données, MEAN est une pile complète de JavaScript.



La solution MEAN offre une approche moderne du développement web. Il utilise également la puissance des SPA modernes (Single Page Applications), ce qui ne nécessite pas de rafraîchir complètement une page web pour chaque requête de serveur comme le font la plupart des applications web traditionnelles. Aujourd'hui, il existe une tendance à développer plusieurs frameworks qui correspondent à un langage de programmation unique et forment une solution full-stack. Mais, à partir de maintenant, l'utilisation de MEAN est une approche très efficace pour le développement web.

Il y a seulement quelques années, MongoDB, Express.js, AngularJS et Node.js ont fait leur apparition. Maintenant, ils ont grandi et se sont alignés ensemble. Il y a une très bonne soutenabilité de ces frameworks, beaucoup de développeurs provenant de LAMP ont basculé vers la solution MEAN afin d'être sur du mono langage.



Mais pour être honnête la solution MEAN a quand même quelques inconvénients :

- **Il faut des experts :** La pile de technologie MEAN a beaucoup de "limites" et de bugs, qu'il est nécessaire de connaître, afin de ne pas construire son projet dans la mauvaise direction. Angular par exemple a beaucoup de limitations (même si certains de ces problèmes sont résolus dans Angular 2). C'est également le cas de MongoDB et des autres technologies de la pile. Cela amène aussi à la prochaine grande question :

- **Difficile de trouver des experts :** Il est plutôt difficile de trouver des experts sur ces nouvelles technologies. Donc, il faut parfois se tourner vers des programmeurs d'entrée de gamme sur ces technologies et essayez de les faire monter en compétence rapidement. Mais généralement ce n'est pas forcément une solution viable sauf peut-être si les développeurs en question sont passionnés par ces technologies.

- **Pas essayé ni testé :** Il y a peu de réponses en ligne aux différents problèmes qu'un développeur pourrait rencontrer en ce qui concerne MEAN en ligne. Certaines technologies sont si nouvelles, qu'il n'y a qu'une petite communauté en ligne qui discute des problèmes qui pourraient surgir pendant le développement.

Qui utilise MEAN stack ?

Il existe de nombreuses organisations bien connues qui utilisent Node.js en production, notamment PayPal, LinkedIn, Netflix et le New York Times. Voici quelques exemples de types d'applications qui peuvent bénéficier de l'utilisation de Node : les API REST, les applications de chat et les applications de suivi en temps réel (tableaux de bord de courtage, statistiques en temps réel sur les utilisateurs, etc.)

Cette nouvelle approche nécessite l'utilisation d'applications back-end très rapides, évolutives, faciles à déployer et à entretenir. Toutes les grandes entreprises ont commencé à migrer leurs applications

vers Node.js et d'autres construisent des applications à partir de zéro en utilisant la technologie MEAN stack. PayPal, Dow Jones et Uber ont déjà placé des solutions d'affaires Node.js en production. Yahoo, HP et beaucoup d'autres ont planifié leur prochaine génération de produits sur cette plateforme de pile MEAN.

Conclusion

Il est donc possible de travailler de bout en bout sur un projet web avec le langage JavaScript. Coté client, depuis 2010, de nombreux frameworks ont vu le jour permettant de développer des applications web structurées, permettant de pallier les défauts de JavaScript.

Coté serveur, Node.js et ses différents modules permettent de travailler efficacement avec la partie cliente en communicant en « json », le mode de communication natif de JavaScript.

Enfin, de nombreux modules permettent de travailler avec n'importe quel type de base SQL ou NoSQL. Les bases NoSQL dont principalement MongoDB, gagnent en popularité grâce aux possibilités qu'elle apporte sur Node.js.

G. Conclusion

Il est fortement conseillé de prendre son temps pour apprendre sur Node.js et le développement JavaScript côté serveur, c'est l'un des éléments les plus intimidants de la technologie JavaScript en raison de ses nombreuses options de portée. Cependant, cela en vaut la peine, car pratiquement tout le développement JavaScript moderne nécessite de naviguer dans l'écosystème Node.js (par exemple npm qui fait partie de Node.js pour construire des projets côté client JavaScript ou des projets Node.js pour fonctionner sur le serveur).

III) Frameworktisation de JavaScript serveur (Node.js)

A. Introduction

Fini le temps où JavaScript était la langue des applications de navigateur côté client seulement. Aujourd'hui, le JavaScript full-stack est une approche courante pour la construction de systèmes à l'échelle de l'entreprise avec de riches capacités que ce soit côté client et côté serveur. Cela est devenu possible après que Node.js est apparu en 2009. Contrairement au framework front-end Angular JS, Node.js étend les capacités de JavaScript pour permettre son utilisation pour le travail côté serveur.

Selon l'enquête StackOverflow de 2017, non seulement JavaScript est la langue la plus utilisée (62,5 % sur 36 625 réponses), mais Node.js est la technologie la plus populaire (47,1 % de 20 229 réponses). Une telle popularité transite dans la prolifération d'un ensemble d'outils polyvalent qui peut être utilisé avec Node.js.

B. Concept

Il y a un débat en cours autour du terme framework. Les frameworks, par définition, sont des ensembles de conventions pour interpréter le code. Mais en termes simples, les frameworks déforment l'environnement de développement original (par exemple Node.js) pour apporter plus de confort aux ingénieurs logiciels.

La commodité, dans ce cas, a une signification très variable qui dépend de la longueur d'un projet, de sa complexité, de son évolutivité et pour finir, des préférences d'ingénieries. C'est pourquoi, toutes les technologies populaires, JavaScript, Node.js ou autres sont généralement entourées d'une communauté animée qui suggère continuellement de nouveaux frameworks pour différents cas d'utilisation. Ainsi, les frameworks Node.js ont des goûts différents et généralement du point de vue technique, ils sont divisés en 4 groupes principaux :

C. Types de framerwork

Frameworks MVC : Tout est dans le nom. Ces frameworks offrent une grande liberté de développement, une grande flexibilité sans ajouts et sont basés sur le modèle de conception Model-View-Controller.

Les frameworks MVC full-stack : Ces instruments sont livrés avec des bibliothèques, des intégrations, des moteurs de gabarit, des scaffolding et d'autres add-ons qui permettent aux ingénieurs d'automatiser leur travail, mais généralement au prix de la flexibilité.

Les frameworks d'API REST : Ces outils rationalisent le développement du serveur API REST et peuvent être utilisés en combinaison avec tout autre groupe de frameworks.

Autres : Et il existe de nombreux logiciels intermédiaires, bibliothèques et autres solutions spécifiques qui peuvent être considérés comme des frameworks dans une certaine mesure.

D. Les frameworks Node.js populaire

Cela dit, voici les frameworks côtés serveur Node.js les plus populaires en fonction du nombre d'étoiles Git qu'ils possèdent. Les frameworks développent le Node.js d'origine ou fournissent un développement léger et flexible avec un nombre limité de fonctionnalités.

Express Js : le plus populaire et flexible des middlewares

Site officiel : https://expressjs.com/

Introduction

Express est le framework Node.js le plus populaire. Si on n'est pas familier avec le concept MEAN, E signifie Express. Et MEAN est une pile complète traditionnelle de technologies suffisantes pour couvrir les principales tâches d'ingénierie logicielle :

- MongoDB : base de données
- Express : back-end middleware
- Angular : front-end
- Node.js : environnement d'exécution.

Middleware signifie simplement que le framework fournit des instruments pour construire le pont entre le côté serveur avec une base de données et un côté client. Express est suggéré comme un cadre léger, flexible et non-dopé. Il n'est pas utilisé parce qu'un ingénieur logiciel n'est pas limité à l'utilisation de pratiques de développement câblées, il offre donc beaucoup de liberté. Par exemple, Express ne dicte aucun des modèles de conception (MVC, MVP, MVVM, etc.). Cela permet aux ingénieurs de créer des applications à leur manière. Et pourtant Express est couramment appliqué dans le modèle MVC.

La liberté s'étend également à l'intégration de la base de données. Alors que MEAN contient MongoDB, Express n'exige pas d'intégration spécifique à la base de données. Les développeurs peuvent choisir la technologie de stockage de données qu'ils préfèrent et installer le package respectif en tant que pilote de base de données.

Express – Les principales caractéristiques

Liberté **:** Les développeurs ne sont limités que par leurs compétences et leurs préférences. Cela garantit également une courbe d'apprentissage duce pour les nouveaux arrivants.

Haute performance : Node.js est célèbre pour ses performances. Express ne fournit qu'une fine couche sans compromettre les performances.

Grande communauté : Express est une technologie qu'on devrait considérer par défaut car elle reste la pratique courante chez la plupart des ingénieurs.

Express.js - Cas d'utilisation

Si vous gérez une petite équipe d'ingénierie et que votre produit n'est pas susceptible de prendre rapidement de la monstruosité à l'échelle de l'entreprise, optez pour Express. C'est le meilleur choix pour des projets inédits qui ne subissent pas le fardeau de l'héritage. Mais au fur et à mesure que votre projet et l'équipe grandissent, l'absence de normes peut mener par inadvertance à la gestion de code. Il faut se méfier.

Meteor : le tout-en-un automatisé

Site officiel : https://www.meteor.com/

Introduction

Meteor est un autre framework JavaScript populaire, qui se trouve sur le pôle opposé du globe Node.js. Contrairement à Express.js, Meteor prend la philosophie de la standardisation et de l'out-of-the-box à l'extrême. Le package permet à un ingénieur disposant d'outils d'intégration suffisants pour créer des applications serveur, mobile, Web et logicielles. Meteor s'intègre à Angular ou React. JS pour connecter le back-end Node.js aux applications frontales. Il utilise MongoDB comme stockage et peut être intégré à Cordova pour créer des applications hybrides utilisant HTML, CSS et JS via WebView. Bien que l'on puisse échanger la plupart des pièces de la pile, cet emballage rationalise une grande partie du travail au sol.

En plus de cela, Meteor est le framework MVC opiniâtre. À la différence des frameworks non-piqués, il ne laisse pas beaucoup de liberté aux ingénieurs mais assure un développement rapide grâce à l'automatisation.

Meteor : principales caractéristiques

Magie de la synchronisation des données : Meteor active automatiquement la synchronisation bidirectionnelle des données entre une application client et un serveur. Ceci est réalisé en créant une copie de mini-base de données sur le côté client qui contient un petit morceau de données auquel un utilisateur s'est abonné. Une fois qu'un client demande de nouvelles données, le système le connecte automatiquement au serveur pour récupérer les données demandées dans la base de données. Lorsque des changements sont appliqués à une extrémité, ils sont automatiquement et instantanément répercutés à l'autre extrémité. Les ingénieurs n'ont pas besoin de configurer manuellement la synchronisation.

Intégration prête à l'emploi : C'est peut-être la qualité principale de Meteor. Il s'agit d'un tout-en-un et réduit une grande partie de l'effort d'intégration à moins que les ingénieurs optent pour des technologies au-delà du bundle standard.

Construire l'automatisation : Les ingénieurs derrière Meteor se vantent d'avoir un système d'emballage qui automatise l'emballage de la même source de code dans plusieurs plates-formes finales, que ce soit mobile, web ou même logicielles. Par exemple, avec l'aide de Cordova, un outil de développement mobile hybride, vous pouvez modifier et empaqueter le code HTML / JavaScript / CSS pour iOS et Android sans trop d'effort.

Meteor – Cas d'utilisation

Étant hautement automatisé, Meteor encourage le prototypage rapide et la création de MVP (produits minimums viables). Vous développez rapidement votre application sur plusieurs plates-formes, découvrez son aspect, la mettez à jour de manière itérative, puis accédez en direct aux commentaires initiaux. Ce n'est pas que vous ne pouvez pas l'utiliser pour des applications à l'échelle de l'entreprise, c'est plutôt que votre équipe d'ingénierie aille probablement souffrir de problèmes d'évolution à long terme. La création d'applications hybrides, par exemple, met généralement en péril l'expérience utilisateur, car les applications semblent décalées et généralement lentes. La synchronisation automatique des données peut également devenir un goulot d'étranglement, une fois que votre base d'utilisateurs atteint des charges à l'échelle de l'entreprise.

Site officiel : https://sailsjs.com/

Introduction

Sails.js est un peu plus standardisé que le framework Express, mais c'est essentiellement une version étendue de ce dernier qui ajoute des fonctionnalités de plus haut niveau qui augmentent la vitesse de développement par rapport à Express. Si vos ingénieurs sont familiers avec le développement de Laravel ou de Ruby-on-Rails, Sails n'élèvera pas une barrière d'adoption élevée pour eux. Il applique une conception traditionnelle Model-View-Controller, ce qui rend le développement organisé et intuitif. Le modèle couvre la gestion des données, la vue est responsable de la représentation en sortie et le contrôleur gère la logique métier côté serveur et agit comme une couche entre la vue et les données.

Sails.js prend le juste milieu entre l'automatisation et la liberté d'ingénierie, ce qui en fait peut-être l'une des options les plus équilibrées dans le développement de Node.js. Bien que vos ingénieurs ne soient pas nécessairement des experts Ruby, MVC est assez commun pour prétendre que Sails a une courbe d'apprentissage douce.

Sails.js : principales caractéristiques

La connexion en temps réel entre le serveur et le client : Sails.js est livré avec Socket.io, qui est un moteur populaire pour la communication en temps réel, basée sur les événements entre le serveur et le client. Il simplifie considérablement la connexion au serveur pour les fonctionnalités en temps réel telles que les outils de messagerie ou de collaboration.

ORM pour plusieurs bases de données : Sails dispose également d'un outil de mapping relationnel objet (ORM) appelé Waterline. Il s'agit d'une couche au-dessus de la base de données qui simplifie la communication DB, ce qui évite aux développeurs de configurer des requêtes de base de données variables pour les bases de données SQL et NoSQL. Fondamentalement, l'ORM permet l'utilisation de n'importe quelle base de données en faisant de petits changements dans l'adaptateur Waterline.

Développement rapide de l'API REST : Sails génère automatiquement des API REST permettant aux développeurs de configurer davantage ce qu'ils doivent générer. Cela rend la configuration de l'API beaucoup plus rapide.

Sails.js – Cas d'utilisation

Certains ingénieurs affirment que Sails est le meilleur framework que vous pouvez trouver pour le développement de Node.js. Alors que Sails est bon pour de nombreuses tâches, il va vraiment briller avec les applications qui auront la messagerie instantanée ou des mises à jour en temps réel basées sur les événements. Ce sont des messagers ou des applications qui impliquent des capacités de messagerie, des tableaux de bord variés et des analyses en continu avec des tableaux de mise à jour en permanence, des outils collaboratifs pour la gestion de projet et l'édition de texte. Les cas d'utilisation peuvent même s'étendre aux jeux multijoueurs.

Koa.js : plus minimaliste et pas de fonctions de rappel

Site officiel : https://koajs.com/

Introduction

Koa est une autre incarnation du minimalisme par les développeurs d'Express. Le but de Koa est de réduire le bundle middleware qui était dans Express et de donner encore plus de liberté aux ingénieurs sans aucune automatisation. Fondamentalement, cela permet de tout contrôler.

Peut-être, le mot-clé que l'on peut utiliser pour décrire Koa est « léger » car il n'a que 550 lignes de code. Cependant, Koa a toujours une communauté plus petite que le framework Express et manque de compatibilité avec les autres systèmes Node.js. Il n'est pas aussi simple qu'Express et peut avoir une courbe d'apprentissage assez raide.

Koa : principales caractéristiques

Combattre les fonctions rappel : L'un des problèmes que les développeurs JavaScript trébuchent est « l'enfer des fonctions de rappel ». Et Koa vise à le combattre. Les rappels sont des fonctions déclenchées uniquement lorsque des événements spécifiques se produisent. Le reste du temps, ils ne passent pas en attente de l'événement déclencheur. Ceux-ci sont appelés des rappels asynchrones et ils sont activement utilisés en JavaScript. L'appel de rappel se produit lorsque les séquences de rappels sont trop longues et complexes et que l'exécution du code doit se poursuivre de haut en bas.

Source de l'image : https://collinmakersquare.files.wordpress.com/2016/02/b4uajfmcqae67qb.png

Koa utilise le générateur ES6 qui s'attaque à ce problème. Il permet au système d'arrêter l'exécution s'il y a un problème de blocage à un certain niveau et une fois le problème résolu, de reprendre l'exécution.

Minimalisme : Le minimalisme définit la modularité et la légèreté du framework, ce qui signifie que les ingénieurs, une fois qu'ils abordent les spécificités, ne consacreront pas beaucoup de temps à la gestion du code et n'utiliseront que les fonctionnalités dont ils ont besoin.

Koa.js – Cas d'utilisation

Le framework convient parfaitement aux projets construits à partir de zéro en utilisant l'architecture des microservices. Le générateur ES6 garantit que les différents services au sein d'une telle architecture fonctionnent correctement sans aucune gestion des fonctions de rappel. Un autre cas d'utilisation courant consiste à créer des API pour un système distribué.

Keystone.js : Node.js CMS & Express en back-end

Site official : http://keystonejs.com/

Introduction

La raison pour laquelle Keystone.js figure dans cette liste est simple. Le projet compte 11 116 étoiles sur Git et figure parmi les meilleurs frameworks Node.js. Mais celui-ci est aussi spécial. Il remet en question la dominance de CMS de WordPress, suggérant un CMS et un environnement de développement d'applications gérés par Node.js.

Techniquement, Keystone est un framework complet. Il contient le framework Express configuré MVC dans son noyau et utilise MongoDB comme stockage. Il est facile à installer et suggère également une courbe d'apprentissage légère pour les ingénieurs back-end. Cela ne peut cependant pas être dit sur les développeurs frontaux. S'ils ne sont pas familiers avec la conception de MVC, l'apprentissage de Keystone peut prendre un certain temps avant de créer un CMS.

Mais le principal point de différenciation est l'ensemble des fonctionnalités intégrées qui automatisent et rationalisent l'ingénierie CMS.

Keystone : principales caractéristiques

Interface d'administration générée automatiquement : C'est auto-explicatif. Keystone va générer l'interface utilisateur d'administration, ce qui simplifie les choses au cours du développement et réduit l'effort requis pour créer une interface d'administration pour les utilisateurs non techniques après la production.

Traitement automatique des formulaires : Les ingénieurs doivent définir des modèles de données et le framework activera automatiquement les validations de formulaires, les téléchargements d'images et les mises à jour de bases de données.

Authentification prête à l'emploi et gestion de session : Il n'est pas nécessaire d'utiliser beaucoup de temps à créer et mettre en place un système d'authentification comprenant la connexion, la déconnexion et les fonctionnalités de chiffrement des champs de mot de passe. Tout cela est inclus de bas avec le paquet.

Intégration de messagerie : Enfin, le framework comprend des capacités de courrier électronique qui simplifient l'ingénierie de la gestion des courriels. Pour les adeptes de Mailchimp, il faut savoir qu'il fonctionne avec le service Mandrill associé.

Keystone.js – Cas d'utilisation

La réponse est évidente. Si vous n'aimez pas WordPress pour son flux laggy et la gestion désordonnée, essayez Keystone. Bien que WordPress ait publié Calypso, un environnement basé sur JavaScript pour les utilisateurs de WordPress avec plusieurs sites de contenu à gérer, il est encore moins configurable que ce que Keystone fournit. L'inconvénient ici est la communauté et les modules préconstruits. WordPress est important non pas en raison de la technologie sous-jacente, mais en raison de son écosystème, quelque chose que vous n'aurez pas avec Keystone.

LoopBack.js : Générateur automatique d'API REST

Site officiel : https://loopback.io/

Introduction

LoopBack est le représentant le plus clair des frameworks d'API REST. Il est construit sur la base du framework Express et permet aux ingénieurs de créer des API qui connectent le côté serveur avec les applications client de manière semi-automatique.

Malgré le fait que LoopBack est livré avec des modules Node.js, il est toujours entièrement configurable si nécessaire. Les ingénieurs derrière le framework suivent le mantra de la convention sur la configuration. Cela signifie essentiellement que les ingénieurs ne sont pas tenus de prendre de nombreuses décisions lorsqu'ils travaillent avec l'outil et que la flexibilité n'est pas sacrifiée.

Les ingénieurs sont friands de bonnes capacités de gestion de code et de la fonction Explorer qui permet de naviguer dans vos API. L'outil est simple à prendre en main car il suit la philosophie agnostique back-end et nécessite peu ou pas de code de composition.

1) LoopBack.js : principales caractéristiques

Génération de l'API REST : La génération d'API semi-automatique permet aux utilisateurs de LoopBack d'éviter la configuration de modèles, de sécurité, de paramètres et de points de terminaison REST. Cela réduit considérablement le temps consacré à la création d'API REST par rapport au processus Express.

API Explorer : La fonctionnalité permet à l'utilisateur de parcourir les API, de les gérer et d'éviter de créer une documentation API.

LoopBack.js – Cas d'utilisation

Comme vous l'avez deviné, LoopBack serait le meilleur outil pour travailler lorsque votre ingénierie repose largement sur l'utilisation des API REST et que vous souhaitez rationaliser leur développement jusqu'à un flux de travail hautement automatisé.

Conclusion

L'arrivée de la plateforme d'exécution Node.js a donné vie à une frameworktisation du JavaScript côté serveur. En effet de nombreux framerwork ont vu le jour et chacun à sa particularité. Ils naissent souvent dans le cadre d'un projet ou d'un cas d'utilisation précis. En effet le framework standard Node.js offre la possibilité d'étendre sa solution et de pouvoir créer plusieurs framerwork.

Mais trop de choix peuvent nous faire tourner la tête c'est pourquoi plusieurs étapes à suivre sont recommandées afin de choisir le framework adapté à son besoin.

Premièrement, il est nécessaire d'évaluer son équipe d'ingénierie afin d'identifier les technologies avec lesquelles l'équipe est en phase. Si l'on souhaite se lancer dans le développement de Node.js, il est préférable de s'en tenir à des frameworks simples et couramment utilisés qui ne nécessitent pas de longue intégration. Les solutions complètes, standardisées et complètes exigeront certainement plus de temps.

Ensuite, il faut définir les spécificités du projet. Analyser comment les fonctionnalités du framework correspondent au projet. Si l'on dispose d'une équipe d'ingénieurs expérimentés et que l'on souhaite construire un prototype ou un MVC rapide, il est recommandé d'opter pour des solutions plus

standardisées et full-stack. Pour un développement à long terme, il est préférable de s’en tenir à des instruments légers et flexibles.

Avant de se lancer avec un framework il faut vérifier ses limites de recherche et le soutien de sa communauté. Tout outil d'ingénierie logicielle ne prospère que s'il est soutenu par une communauté puissante. Bien qu'il y ait de bons cadres, le manque de soutien par les pairs peut devenir un bloqueur de routine, car les ingénieurs devront trouver eux-mêmes des problèmes.

Enfin, il est également important de considérer la maturité d'un framework. Certains outils sont encore en développement. Ils peuvent souffrir de bugs ou certaines fonctionnalités peuvent être encore en développement. Il faut les évaluer pour s’assurer d'avoir toute la liste des fonctionnalités dont on a besoin.

JavaScript : Est-ce une Solution ?

I) Pour quel cas utilise-t-on JavaScript côté serveur (Node.js) ?

A. Introduction

Le choix de la technologie du back-end est l'une des décisions les plus importantes que chaque PDG et CTO sont amenées à faire. Il détermine à quelle vitesse un produit peut être expédié sur le marché, quel est le coût total et à quel point le maintien de la douleur sera important.

JavaScript a été l'un des langages de programmation côté client les plus populaires et un outil de développement Web front-end couramment utilisé. Cependant, il a également gagné du terrain dans différents domaines d'application et sur des plates-formes distinctes, telles que React Native, Appcelerator Titanium, Apache Cordova / PhoneGap, NativeScript et Node.js, ce qui est totalement différent des autres Frameworks JavaScript couramment utilisés.

Node.js est un environnement d'exécution d'application qui vous permet d'écrire des applications côté serveur en Javascript. Grâce à son modèle d'Entrées / Sorties unique, il excelle dans le type de situations évolutives et en temps réel. C'est aussi léger, efficace et il permet d'utiliser Javascript à la fois sur le frontend et le back-end ouvrant ainsi de nouvelles possibilités. Il n'est pas surprenant que tant de grandes entreprises l'ont exploité dans la production, à savoir Walmart, Netflix, Medium, LinkedIn ou Groupon.

B. Internet of Things

Depuis 2012, lorsque la popularité de l'IoT (l'internet des objets connectés) a augmenté de façon spectaculaire, Node.js est devenu l'un des préférés des solutions pour les entreprises et les organisations cherchant à développer leurs systèmes IoT privés et publics. L'avantage le plus évident de Node.js en tant que back-end pour de tels réseaux est sa capacité à traiter plusieurs demandes simultanées et des événements émis par des milliers ou même des millions de périphériques sur le réseau. L'avalanche de requêtes et de données provenant des périphériques IoT ne bloque pas les serveurs Node.js grâce à leur architecture événementielle et au traitement asynchrone adapté aux lourdes opérations d'Entrées / Sorties sur le réseau IoT. Cela rend Node.js rapide en tant que couche d'application entre ces périphériques et les bases de données utilisées pour stocker les données qui en proviennent.

En outre, les développeurs IoT travaillant dans des scénarios à forte intensité de données peuvent tirer parti de la faible ressource exigences de Node.js. Les besoins en mémoire insuffisante permettent l'intégration facile de Node.js en tant que logiciel dans des contrôleurs à carte unique tels que Arduino, largement utilisé pour la construction d'appareils numériques qui constituent des systèmes IoT. Enfin, la communauté Node a été une des premières à adopter la technologie IoT, créer plus de 80 paquets pour les contrôleurs Arduino et plusieurs paquets pour le Pebble et Fitbit dispositifs portables largement utilisés dans les systèmes IoT.

C. Real-Time Chats

Node.js fournit toutes les fonctionnalités de base pour créer des tchats en temps réel de toute complexité. En particulier, Node dispose d'une Event API puissante qui facilite la création de certains types d'objets de ("émetteurs") qui émettent périodiquement des événements nommés "écoutés" par les gestionnaires d'événements. Grâce à cette fonctionnalité, Node.js facilite l'implémentation d'événements côté serveur et de notifications push largement utilisées dans la messagerie instantanée et d'autres applications en temps réel.

L'architecture basée sur les événements de Node fonctionne également bien avec le protocole WebSockets qui facilite un échange bidirectionnel rapide de messages entre le client et le serveur via une connexion ouverte. En installant les bibliothèques WebSockets sur le serveur et le côté client, vous pouvez implémenter en temps réel une messagerie qui a des frais généraux, une latence plus faible et un transfert de données plus rapide que la plupart des autres solutions, plus conventionnelles.

Node dispose d'un excellent support pour les WebSockets via des bibliothèques telles que socket.io, ws, ou websocket-node, grâce auquel vous pouvez facilement déployer des tchats et des applications efficaces en temps réel. Avec socket.io, par exemple, tout ce que vous avez à faire pour créer un tchat live basique est d'installer la librairie socket.io sur le serveur et le client, créer des émetteurs d'événements et des diffuseurs qui feront passer les messages par la connexion WebSockets ouverte. Cette fonctionnalité de base peut être obtenue avec seulement quelques lignes de code.

D. Complex Single-Page Applications

Node.js est un excellent choix pour les single-page applications (SPA) grâce à sa gestion efficace
D'appels asynchrones et les lourdes charges de travail d'Entrées / Sorties caractéristiques de ces applications. La boucle d'événement de Node.js permet de "retarder" plusieurs demandes

simultanées du client, ce qui assure des transitions en douceur entre les vues et les mises à jour de données en continu. En outre, Node.js fonctionne bien avec les SPA pilotés par les données, où le serveur agit comme un back-end qui fournit des données au client alors que le client fait tout le rendu HTML.

Aussi, Node.js est bon pour les SPA car il utilise le même langage (JavaScript) comme beaucoup de frameworks JavaScript populaires (Ember, Meteor, React, Angular) utilisés dans la construction de SPA. Node.js et les navigateurs utilisent JavaScript, il y a moins de changement de contexte entre eux. Ainsi les développeurs peuvent utiliser les mêmes structures de données de langage et des approches modulaires à la fois sur le serveur et le côté client. Cela se traduit par un développement plus rapide et une meilleure maintenance de vos SPA. Les avantages de Node.js ont été exploités par des SPA aussi célèbres que Netflix, Linkedin et Medium, pour en nommer quelques-uns.

E. Real-Time Collaboration Tools

Comme dans le cas des discussions en temps réel, l'architecture asynchrone et événementielle de Node est parfaite pour les applications de collaboration. Sur ces applications, plusieurs événements et demandes d'Entrées / Sorties se produisent simultanément. Par exemple, plusieurs utilisateurs peuvent modifier le même paragraphe, commenter, publier des messages et joindre des fichiers. Les modifications d'un élément de contenu peuvent être appliquées uniquement après une cascade d'événements où chaque étape dépend de la précédente.

Les API WebSockets et Event de Node garantissent que les lourdes opérations d'Entrées / Sorties effectuées par de nombreux utilisateurs ne bloquent pas le serveur. Ainsi, tous les événements et données côté serveur sont renvoyés au client à temps. En émettant des notifications push sur le client, Node.js mettra également à jour instantanément la collaboration environnement afin que tous les utilisateurs aient une représentation unique et cohérente de l'application. C'est précisément la raison pour laquelle l'équipe de l'application de gestion de projet Trello (http://trello.com) utilise le Node.js empiler. L'équipe d'ingénierie de Trello trouve que Node.js serait génial pour se propager instantanément beaucoup de mises à jour et maintenir beaucoup de connexions ouvertes, grâce à son événementiel et son architecture non bloquante. Parmi les autres applications de collaboration en temps réel construites sur Node.js, nous devrions également mentionner Yammer, un service de réseau social freemium facilitant la communication privée dans les entreprises.

F. Streaming apps

Contrairement aux applications de serveurs distants, dans le streaming d'applications le programme est exécuté sur la machine locale de l'utilisateur final. Le streaming d'applications permet de télécharger des parties de l'application à la demande sans surcharger le serveur et l'ordinateur local. Initialement, seulement certaines parties de l'application nécessaire pour le bootstrap sont téléchargées, tandis que le reste peut être téléchargé en arrière-plan si besoin. Lorsque l'application est complètement téléchargée, elle peut fonctionner sans aucune connexion internet. Dans le cas où vous souhaitez enregistrer des données dans votre compte, l'application peut lancer des demandes auprès du serveur. De même, les événements du serveur peuvent mettre à jour votre application locale sans trop de frais généraux de trafic réseau.

Node.js est excellent pour le développement de telles applications de streaming grâce à son API native de Stream. En particulier, Node.js dispose d'une interface de flux lisibles et inscriptibles pouvant être traité et surveillé très efficacement. Les instances de flux sont essentiellement des tuyaux Unix qui permettent de transmettre des parties du code exécutable de l'application à la machine locale tout en maintenant une connexion ouverte pour les nouveaux composants à télécharger à la demande. Les flux permettent aux utilisateurs de se canaliser les uns aux autres et diffuser des données directement à sa destination finale. En prime, les flux ne nécessitent pas de mise en cache et de données temporaires. Il suffit d'une connexion ouverte pour diffuser des données d'application d'un endroit à un autre.

G. Micro services Architecture

Node.js est une excellente solution pour développer des micro services et créer des API faciles à utiliser pour les connecter. En particulier, le référentiel Node.js propose les frameworks Express et Koa, qui permet facilement de monter plusieurs instances de serveur pour chaque micro-service et de concevoir des adresses de routage pour celles-ci. Node.js avec Express permet de créer des modules très flexibles responsables de certaines parties d'une application.

De plus, Node.js peut être facilement intégré à Docker et vous permettra ainsi d'encapsuler des micro-services dans des conteneurs hermétiques pour éviter tout conflit entre le développement de l'application et les environnements utilisés dans chacun d'eux. L'utilisation de Node.js pour les micro-services bénéficie également de légères exigences. Node.js avec l'architecture micro-services réduit considérablement le temps de déploiement des applications et améliore l'efficacité, la soutenabilité et l'évolution de vos applications. L'architecture des micro-services aide également à gérer

efficacement la division du travail dans vos équipes d'ingénierie, leur permettant de travailler sur des tâches spécifiques sans affecter d'autres parties de votre application.

Ces avantages ont été exploités avec succès par PayPal, le premier système de paiement en ligne au monde, qui utilise Node.js pour alimenter son architecture de micro-services depuis 2013. PayPal modularise sa pile d'applications et divise le processus de développement en plusieurs micro-services. Ainsi, il organise ses équipes pour y travailler plus efficacement. PayPal a pu faire évoluer Node.js pour que plusieurs équipes puissent travailler sur le même projet. Les résultats de cette transition étaient stupéfiants.

L'application Node.js Paypal a pu être construite deux fois plus rapidement et avec moins de ressources humaines. L'entreprise a réussi à réduire sa base de code et améliorer les performances, où une seule application de base Node pourrait gérer deux fois plus rps (demandes par seconde) que 5 applications Java utilisées par PayPal auparavant.

I) Pour quel cas ne pas utiliser Node.js ?

Même si Node.js est dans de nombreux cas la solution idéale pour construire des applications, il existe encore quelques cas d'utilisation lorsque vous ne devriez pas envisager de l'utiliser.

La première chose qui vient à l'esprit généralement est les applications informatiques lourdes. Node.js est basé sur un modèle d'évènements non bloquant d'Entrées / Sorties et n'utilise qu'un seul cœur de CPU. Les opérations lourdes CPU vont juste bloquer les demandes entrantes, rendant le plus grand avantage de Node.js inutile. Par conséquent, si vous considérez la construction de certains logiciels CPU-lourds, essayez une technologie différente, plus appropriée qui vous donnera un meilleur résultat.

Il y a plus de choses à considérer avant de commencer avec Node.js. De nombreux paquets pour les applications Node.js sont disponibles sur la plateforme npm. La communauté est dynamique, la technologie arrive à maturité et le npm est le plus grand référentiel disponible pour le moment. Cependant, les paquets varient dans leur qualité. Parfois, on peut toujours rencontrer des problèmes avec des paquets pris en charge uniquement par des utilisateurs individuels et non maintenus correctement. Par exemple, lors de la connexion d'une application Node à un système de base de données obscur ou ancien. Enfin, l'utilisation de Node.js est inutile pour les applications HTML ou CRUD (Create, Read, Update, Delete) simples dans lesquelles vous n'avez pas besoin API séparée et où toutes les données proviennent directement du serveur. Votre application pourrait être

légèrement plus évolutive, mais ne vous attendez pas à un trafic plus important sur votre application uniquement parce que vous avez utilisé Node.js. Dans certains cas, vous devriez vous en tenir aux cadres éprouvés tels que Ruby on Rails.

Ruby on Rails et Node peuvent atteindre les mêmes résultats. Cependant, il existe différentes situations où chacun est plus performant que l'autre. Avec Rails, vous pouvez prototyper et bouger rapidement, casser des choses et obtenir un CRUD application en une heure.

II) JavaScript (Node.js) correspondra-t-il à votre projet de demain ?

A. Introduction

Pourquoi tant de grands groupes ont choisi Node.js comme technologie en back-end ? Voici les différents comparatifs entre Node Js et d'autres technologies back-end avec les principaux avantages et inconvénients de ces environnements, mais aussi quelques inconvénients qu'il faut prendre en compte avant de faire un choix. Une mauvaise décision peut coûter énormément d'argent, alors il faut choisir judicieusement.

B. Node Js VS Ruby on Rails

Node.js est plus approprié pour les applications dynamiques avec plusieurs demandes de serveur et des mélanges de données fréquents entre le client et le serveur. Quels types d'applications sont-ils ? Ceux-ci sont les applications de messagerie instantanée comme les chats et les applications collaboratives (dessin, vidéoconférence) collectivement dénommé RTA (Real-Time Applications). L'architecture Node.js basée sur les événements est parfaite pour la gestion opérations d'Entrées / Sorties lourdes, les demandes de serveur et flux de données dans ces applications. Pour la même raison, Node.js est un choix préféré pour les applications de page unique (SPA) qui impliquent un traitement côté client lourd, rendu et où la fonction principale du back-end est de fournir une API REST. De même, à chaque fois, la performance et l'évolution des applications web sont une préoccupation majeure pour le choix d'une technologie. Léger et rapide Node.js surpasse le framework Rails qui est une alternative au Node.js. C'est pourquoi des entreprises telles que Linkedin ont commencé à utiliser Node.js pour des raisons de performance et d'évolution.

En revanche, Ruby on Rails fonctionne mieux que Node.js dans les applications gourmandes en ressources. Node.js est un environnement monothread qui n'a pas été conçu pour les opérations intensives de CPU qui gère des graphiques, images et données. Faire des calculs sur de vastes tableaux de données peut simplement bloquer toutes les demandes en entrées rendant l'avantage

principal du Node.js inutile. Si vous souhaitez construire un processeur lourd d'applications, Rails est certainement une meilleure option que Node. Rails est également mieux quand le temps de développement est critique. Avec tous les modules et générateurs disponibles hors de la boîte, Rails est très puissant dans le développement rapide d'application (RAD). Juste en quelques commandes, vous pouvez avoir pleinement un prototype fonctionnel qui peut être modifié avec des fonctionnalités supplémentaires plus tard. Node.js peut aussi fournir des scripts de générateur pour accélérer le développement, mais le prototypage rapide est la spécialité de Rails.

Ainsi, lorsque vous choisissez entre Node et Rails, vous devriez prendre en compte le temps de développement, les performances, l'évolution de votre application, ainsi que le type de cas d'utilisation auxquels elle s'applique. S'il y a une exigence sur le temps de développement et si des traitements lourds du CPU sont nécessaires alors il est préférable de s'orienter vers Rails. En revanche, si vous avez des exigences en termes de RTA, SPA et à d'autres solutions lourdes d'Entrées / Sorties, orientez-vous vers Node.js. Je précise que ceci est mon avis en me basant sur ce que j'ai pu constater à travers ces technologies et les différents avis que j'ai pu lire dessus.

C. Node Js VS PHP

Introduction

Même si PHP et Node.js peuvent gérer des applications de toute complexité, ils sont construits autour de différents concepts et architectures. Si vous êtes un propriétaire d'applications choisissant entre ces deux environnements, vous devez être conscient de leurs principaux avantages et limites.

Node.js et PHP sont deux solutions de développement web très populaires. PHP, un langage de script créé par Rasmus Lerdorf en 1994, était l'un des meilleurs langages de l'ère du Web 1.0. Les manifestations éloquentes du succès de PHP sont les CMS (Content Management Systems), tels que WordPress, Joomla ou Drupal, qui alimentent des millions de blogs et de portails Web. Node.js est un représentant d'une génération de développement web plus jeune. Contrairement à PHP, Node.js n'est pas une langue, mais un environnement d'exécution qui utilise JavaScript pour le développement d'applications côté serveur. Lancé en 2009, Node.js a démontré la puissance de JavaScript dans la construction d'applications basées sur les événements, pilotées par les données, avec beaucoup d'E / S pour l'ère du Web 2.0.

Points de comparaisons

Mélange de code et de contenu → PHP

Avec PHP, vous ouvrez les balises PHP magiques et commencez à écrire du code en quelques secondes. Pas besoin de modèles ! Pas besoin de fichiers supplémentaires ou d'architectures élaborées, seulement une puissance logistique programmable à portée de main.

Séparation des préoccupations → Node.js

Les programmeurs ajoutent une structure et séparent la couche cosmétique de la couche logique. Le code devient ainsi plus propre, plus facile à comprendre et plus facile à maintenir. Les frameworks fonctionnant sur Node.js sont construits par des programmeurs qui savent que la vie est meilleure quand le modèle, la vue et le contrôleur sont séparés.

Nouveauté signifie plus de fonctionnalités modernes → Node.js

Les plug-ins Node.js ne sont pas seulement plus récents que ceux de PHP, ils ont été construits en parfaite connaissance des dernières approches architecturales.

Simplicité (en quelque sorte) → PHP

Il n'y a pas grand-chose avec PHP : quelques variables et fonctions de base pour jongler avec les chaînes et les nombres. C'est une couche mince et simple.

Des dizaines d'options de langue → Node.js

Contrairement à PHP, Node.js peut compiler beaucoup de langages majeurs de manière croisée pour fonctionner en JavaScript. Il y existe des langages bien connus comme Java, C# ou Lisp et des dizaines d'autres comme Scala, OCaml et Haskell. Il y a même des cadeaux pour les amants nostalgiques du BASIC ou du Pascal.

Les appels de service sont plus minces que les appels PHP HTML → Node.js

Une fois que le code JavaScript est dans le cache du navigateur, la seule chose qui se déplace le long des fils est la nouvelle donnée. Il n'y a pas de voyages répétés pour télécharger la page entière. Seules les données ont changé. Si vous êtes prêt à consacrer du temps à la création d'une application web côté navigateur, il y a un gros avantage. Node.js est optimisé pour fournir les données et uniquement les données via les services Web.

Le nouveau code l'aide à rattraper son retard → PHP

Si vous êtes un programmeur qui veut faire plus qu'interagir avec une base de données et formater les résultats, vous pouvez maintenant faire plus avec PHP sans vous taire. Le HHVM de Facebook

ajoute le support de Hack, un langage complet rempli de fonctionnalités modernes comme les annotations de type, les génériques et les expressions lambda. Utiliser ceci limite votre code à l'exécution uniquement sur le HHVM, mais ce n'est pas la pire chose au monde et c'est plutôt rapide.

SQL → PHP

PHP a été construit pour coexister avec MySQL et ses nombreuses variantes. Il existe également d'autres bases de données SQL d'Oracle et de Microsoft. Votre code peut changer avec quelques modifications à vos requêtes. Le vaste monde SQL ne s'arrête pas à ses frontières. Certains des codes les plus stables et les plus développés vont s'interfacer avec une base de données SQL, ce qui signifie que toute cette puissance peut également être facilement intégrée dans un projet PHP.

JSON → Node.js

Si vous devez avoir accès à SQL, Node.js a des bibliothèques pour cela. Mais Node.js parle aussi JSON, la lingua franca pour interagir avec plusieurs des dernières bases de données NoSQL. Cela ne veut pas dire que vous ne pouvez pas obtenir de bibliothèques JSON pour votre pile PHP, mais il y a quelque chose de fluide dans la simplicité de travailler avec JSON lorsque vous utilisez JavaScript car rappelons-le JSON signifie littéralement **JavaScript Object Notation**. C'est une syntaxe du navigateur au serveur Web à la base de données. Cela seul vous sauvera des heures de frustration.

Temps de développement → PHP

Pour la plupart des développeurs, l'écriture de PHP pour les applications web est plus rapide : pas de compilateur, pas de déploiement, pas de fichier JAR ou de préprocesseur, simplement votre éditeur favori et quelques fichiers PHP dans un répertoire. Quand il s'agit de créer rapidement un projet ensemble, PHP est un bon outil à utiliser.

Vitesse brute → Node.js

L'écriture de code JavaScript est un peu plus difficile lorsque vous comptez les accolades et les parenthèses, mais lorsque c'est fait, votre code Node.js peut voler. Le mécanisme de rappel est génial car il vous évite de jongler avec les threads. Le noyau est bien construit et conçu pour faire tout cela pour vous. N'est-ce pas ce que tout le monde veut ?

Conclusion

Les deux sont des technologies de back-end, mais Node.js peut offrir un avantage si vous cherchez à avoir une pile de technologie totalement JavaScript à la fois client et serveur. Si vous essayez de

choisir entre des technologies dorsales ou si vous construisez une pile de solutions complète, il est utile d'entrer dans un peu plus de détails.

Vous devriez considérer PHP si votre projet implique :

- Piles logicielles comme la pile LAMP (Linux, Apache, MySQL, PHP)
- CMS comme WordPress, Drupal, ou Joomla etc.
- Serveurs tels que MySQL, SQL, MariaDB, Oracle, Sybase et Postgresql, etc.

Vous devriez considérer Node.js si votre projet implique :

- Piles logicielles telles que la pile MEAN (MongoDB, Express.js, AngularJS, Node.js)
- Single Page Applications dynamiques (SPA)
- Les technologies frontales comme jQuery, AngularJS, Backbone.js, Ember.js, ReactJS, etc.
- Les technologies côté serveur comme Node.js, MongoDB, Express.js, etc.

Il faut garder à l'esprit qu'aucune de ces listes n'est exhaustive. Ceux-ci sont seulement conçus comme un point de départ pour vous aider à avoir une idée de ce à quoi vous pouvez vous attendre et des mots-clés que vous pouvez utiliser pour évaluer la meilleure langue pour vos besoins.

D. Node Js VS Java EE

Introduction

Java EE est le leader des applications d'entreprise. C'est pourquoi je souhaite faire un comparatif en Java EE et Node.js afin de savoir dans quel domaine l'un ou l'autre domine. À l'issue de cette analyse, nous aurons un avis objectif qui permettra de déterminer dans quel cas faut-il opter pour Java EE ou pour Node.js selon votre besoin. Depuis 2009, NodeJS a envahi le petit monde du développement web, et petit à petit fait son trou parmi les plateformes de référence pour réaliser une application, site ou API de services. Le vieux Java EE résiste pourtant, et quiconque souhaite aujourd'hui démarrer un nouveau projet peut se confronter à cette problématique : NodeJS ou Java EE ?

Points de comparaisons

Une base solide comme le roc → Java

Oui, Java a des pépins et des bugs, mais relativement parlant, c'est le Rocher de Gibraltar. En fait, il peut s'écouler des décennies avant que l'équipe JavaScript n'écrive autant de tests de régression que Sun / Oracle développé pour tester la machine virtuelle Java. Lorsque vous démarrez une JVM, vous bénéficiez de 20 années d'expérience auprès d'un conservateur solide déterminé à dominer le serveur d'entreprise. Lorsque vous démarrez JavaScript, vous obtenez le travail d'une coalition

souvent acharnée qui veut parfois collaborer et qui veut parfois utiliser la norme JavaScript pour lancer des attaques passives agressives.

Ubiquité → Node.js

Grâce à Node.js, JavaScript trouve un accueil sur le côté serveur et le côté client. Le code que vous écrivez pour un côté sera plus que probablement exécuté de la même manière sur les deux côtés. Rien n'est garanti dans la vie, mais c'est aussi proche que cela se fait dans le secteur informatique. Il est beaucoup plus facile de coller avec JavaScript pour les deux côtés client / serveur que d'écrire quelque chose une fois en Java et encore en JavaScript, ce que vous feriez probablement si vous décidiez de déplacer la logique métier que vous avez choisi Java pour le côté serveur. Quoi qu'il en soit, Node.js et JavaScript facilitent grandement la migration du code.

IDE plus performants → Java

Les développeurs Java ont Eclipse, NetBeans ou IntelliJ, trois outils de premier ordre qui sont bien intégrés avec les débogueurs, décompileurs et serveurs. Chacun a des années de développement, des utilisateurs dévoués et des écosystèmes solides remplis de plug-ins.

Pendant ce temps, la plupart des développeurs Node.js tapent des mots dans la ligne de commande et codent dans leur éditeur de texte favori. Certains utilisent Eclipse ou Visual Studio, tous deux prenant en charge Node.js. Bien sûr, le regain d'intérêt pour Node.js signifie que de nouveaux outils arrivent, dont certains, comme le Node-RED d'IBM, offrent des approches intrigantes, mais ils sont encore loin d'être aussi complets qu'Eclipse. WebStorm, par exemple, est un outil commercial solide de JetBrains, reliant de nombreux outils de construction en ligne de commande.

Bien sûr, si vous cherchez un IDE qui édite et jongle les outils, les nouveaux outils qui supportent Node.js sont assez bons. Mais si vous demandez à votre IDE de vous laisser éditer pendant que vous travaillez sur le code source en cours d'exécution comme un chirurgien cardiaque ouvre un coffre, les outils Java sont beaucoup plus puissants. Tout est là et tout est local.

Processus de construction simple → Node.js

Des outils de construction complexes comme Ant et Maven ont révolutionné la programmation Java. Mais il n'y a qu'un problème. Vous écrivez la spécification en XML, un format de données qui n'a pas été conçu pour prendre en charge la logique de programmation. Bien sûr, il est relativement facile d'exprimer une branche avec des balises imbriquées, mais il y a toujours quelque chose d'ennuyeux à passer de Java à XML simplement pour construire quelque chose. Avec JavaScript, il n'y a pas de changement de vitesse.

Débogage à distance → Java

Java possède des outils incroyables pour surveiller les grappes de machines. La JVM est dotée de crochets profonds et d'outils de profilage élaborés pour aider à identifier les goulots d'étranglement et les échecs. La pile d'entreprise Java exécute certains des serveurs les plus sophistiqués de la planète et les entreprises qui utilisent ces serveurs ont exigé le meilleur de la télémétrie. Tous ces outils de surveillance et de débogage sont assez matures et prêts à être déployés.

Requêtes de base de données → Node.js

Les requêtes pour certaines des bases de données les plus récentes, comme CouchDB, sont écrites en JavaScript. Mélanger Node.js et CouchDB ne nécessite aucun changement de vitesse et encore moins besoin de se souvenir des différences de syntaxe.

Pendant ce temps, de nombreux développeurs Java, utilisent SQL. Même lorsqu'ils utilisent Java DB (anciennement Derby), une base de données écrite en Java pour les développeurs Java, ils écrivent leurs requêtes en SQL. On pourrait penser qu'ils appellent simplement des méthodes Java, mais vous auriez tort. Vous devez écrire votre code de base de données dans SQL, puis laissez Derby analyser le SQL. C'est un langage agréable, mais c'est complètement différent et beaucoup d'équipes de développement ont besoin de personnes différentes pour écrire SQL et Java.

Bibliothèques → Java

Il y a une énorme collection de bibliothèques disponibles en Java et elles offrent certains des travaux les plus sérieux. Les outils d'indexation de texte comme Lucene et les outils de vision par ordinateur comme OpenCV sont deux exemples de projets Open Source qui sont prêts à être la base d'un projet sérieux. Il y a beaucoup de bibliothèques écrites en JavaScript et certaines sont incroyables, mais la profondeur et la qualité de la base de code Java est supérieure.

JSON → Node.js

Lorsque les bases de données crachent des réponses, Java va élaborer des longueurs pour transformer les résultats en objets Java. Les développeurs argumenteront pendant des heures sur les mappages POJO, Hibernate et d'autres outils. Les configurer peut prendre des heures voir des jours. Finalement, le code Java récupère les objets Java après toute la conversion.

De nombreux services Web et bases de données renvoient des données dans JSON, une partie naturelle de JavaScript. JSON est maintenant si commun et utile que de nombreux développeurs Java utilise le format, de sorte qu'un certain nombre de bons analyseurs JSON sont également disponibles

en tant que bibliothèques Java. Mais JSON fait partie de la fondation de JavaScript. Vous n'avez pas besoin de bibliothèques. Tout est là et prêt à être utilisé.

Ingénierie solide → Java

C'est un peu difficile à quantifier, mais beaucoup de paquets complexes pour un travail scientifique sérieux sont écrits en Java parce que Java a de solides bases mathématiques. Sun a passé longtemps à transpirer les détails des classes d'utilité et ça se voit. Il existe des BigIntegers, des routines d'E/S élaborées et un code Date complexe avec des implémentations des calendriers grégorien et julien.

JavaScript est bien pour les tâches simples, mais il y a beaucoup de confusion dans les tripes. Un moyen facile de voir cela est dans les trois résultats différents de JavaScript pour les fonctions qui n'ont pas de réponses : undefined, NaN et null. Lequel a raison ? Eh bien, chacun a son rôle, l'un d'entre eux étant de pousser les programmeurs à essayer de les garder droits.

Vitesse → Node.js

Les gens aiment faire l'éloge de la vitesse de Node.js. Les données arrivent et les réponses sortent comme des éclairs. Node.js ne dérange pas avec la configuration de threads séparés avec tous les maux de tête de verrouillage. Il n'y a pas de frais généraux pour ralentir quoi que ce soit. Vous écrivez du code simple et Node.js fait le bon choix le plus rapidement possible.

Cette louange vient avec une mise en garde. Votre code Node.js doit être simple et mieux fonctionner correctement. En cas d'interblocage, le serveur entier pourrait se bloquer. Les développeurs du système d'exploitation ont retiré leurs cheveux en créant des filets de sécurité qui peuvent supporter les erreurs de programmation, mais Node.js jette ces filets.

Threads → Java

Le code rapide est génial, mais il est généralement plus important qu'il soit correct. Voici où les fonctionnalités supplémentaires de Java ont un sens.

Les serveurs Web de Java sont multithread. Créer plusieurs threads peut prendre du temps et de la mémoire, mais c'est payant. Si un thread se bloque, les autres continuent. Si un thread nécessite un calcul plus long, les autres threads ne sont pas affamés (habituellement).

Si une requête Node.js s'exécute trop lentement, tout ralentit. Il n'y a qu'un seul thread dans Node.js et il arrivera à votre événement quand c'est bon et prêt. Il peut sembler superflu, mais en dessous, il

utilise la même architecture qu'un bureau de poste à guichet unique la semaine précédant Noël.

Il y a eu des décennies de travail consacré à la construction de systèmes d'exploitation intelligents qui peuvent jongler avec de nombreux processus différents en même temps. Pourquoi remonter dans le temps jusqu'aux années soixante quand les ordinateurs ne pouvaient gérer qu'un seul thread ?

Momentum → Node.js

Oui, toutes les leçons de nos grands-parents sur l'épargne sont vraies. Ne gaspille pas ! Il peut être douloureux de voir le dévouement insensé de Silicon Valley au « nouveau » et au « perturbateur », mais parfois, il est plus logique de nettoyer le croupion. Oui, Java peut suivre, mais il y a du vieux code partout. Bien sûr, Java a de nouvelles routines IO, mais il a aussi de vieilles routines IO. Beaucoup de classes d'applet et d'outil peuvent gêner.

La compilation croisée entre Java et Node.js → Node.js & Java

Le débat sur l'utilisation de Java ou Node.js sur vos serveurs peut et continuera pendant des années. Contrairement à la plupart des débats, cependant, nous pouvons l'avoir dans les deux sens. Java peut être compilée en JavaScript. Google le fait fréquemment avec Google Web Toolkit et certains de ses sites Web les plus populaires utilisent du code Java, Java qui a été traduit en JavaScript.

Il y a aussi un chemin dans l'autre direction. Les moteurs JavaScript comme Rhino exécutent JavaScript dans votre application Java, où vous pouvez créer un lien vers celui-ci. Si vous êtes vraiment ambitieux, vous pouvez créer un lien dans le moteur V8 de Google.

Tableau comparatif

Technologies / Fonctionnalités	Node.js		Java EE	
	Les +	**Les -**	**Les +**	**Les -**
Environnement et framework	Simplicité d'intégration des librairies et d'exploitation	Fonctionnalités d'exploitation réduites	De nombreuses librairies et fonctionnalités d'exploitation	Gestion des dépendances et paramétrage des serveurs d'applications
Prise en main	Architecture logicielle ultra-modulaire	Nécessité de se documenter sur chaque module importé	Frameworks proposant des couches d'abstraction élevées	Difficultés liées à l'utilisation de ces frameworks
Moteur d'exécution et performance	Bonnes performances, résiste à la montée en charge	Quelques zones d'ombres sur la gestion de la compilation partielle par V8	Bonnes performances	Baisse de régime voire crash du serveur lors de montées en charge
Maintenance et évolution	Plateforme très dynamique, beaucoup de nouvelles fonctionnalités à venir	La lisibilité très moyenne de la syntaxe n'est pas aidée par l'absence d'IDE digne de ce nom	Les IDE	Peu de nouveautés à attendre
Sécurité	La possibilité de customiser entièrement sa stratégie sécuritaire par une gestion plus bas niveau	La nécessité de connaître le sujet, ou bien, si l'on choisit d'intégrer des modules préexistants, la difficulté de les paramétrer, comme pour Java EE	Tout est déjà implémenté et prêt à être intégré	Il est parfois difficile de se retrouver dans la jungle des paramétrages

Conclusion

Les deux plateformes sont assez différentes et possèdent chacune une identité propre assez marquée. Pour autant, **il n'est pas évident de trancher entre l'une et l'autre des solutions.**

Pour résumer, Java EE est une plateforme très complète, fiable et éprouvée, qui offre des facilités de développement, mais peu de souplesse. Ses performances sont bonnes, à nuancer en cas de montée de charge. On notera également la complexité des paramétrages et l'âge de la technologie, qui laisse espérer peu de nouveautés à venir.

De son côté, NodeJS est une plateforme innovante, dynamique, souple et performante, qui permet une architecture logicielle modulaire. Néanmoins, le développement événementiel peut s'avérer complexe d'approche et certains points techniques comme la sécurité peuvent parfois devoir être

gérés à bas niveau. Il est aussi parfois difficile de choisir un module pour tel ou tel fonctionnalité, ainsi que d'accéder à une documentation digne de ce nom.
Pour conclure, je dirais que **Java EE est plus adapté pour des gros projets, très cadrés, avec des perspectives d'évolutions moyennes à élevées et un cycle de maintenance long.** Il convient parfaitement à des développeurs de niveau technique moyen.

NodeJS sera parfait pour des projets de taille plus réduite, avec une gestion de projet plus souple, des perspectives d'évolution faibles ou moyennes et un cycle de maintenance moyen à long. Il nécessite d'avoir des développeurs (ou a minima un architecte) de niveau technique un peu plus élevé.

Conclusion

Le JavaScript côté serveur sera-t-il aussi pérenne que le Java et le PHP ? Pour répondre à cette question, faisons un point sur ce que nous venons de voir dans les différentes parties. JavaScript côté serveur (Node.js) dispose actuellement comme on a pu le constater de la communauté la plus importante de tous les langages, ceci grâce à un engouement extrêmement important pour JavaScript depuis maintenant quelques années. Il ne cesse de grandir depuis sa création en 2009, cela fait pratiquement maintenant 10 ans. On peut dire que cette technologie n'est plus dans la catégorie où les développeurs s'y intéressent par curiosité ou pour découvrir une nouveauté. De nombreux articles, livres et vidéos permettent à tout un chacun de s'essayer à Node.js. Les différents acteurs du web ont déjà adopté cette technologie, il n'existe pas une UI JavaScript qui ne fournit pas des exemples de script pour Node.js. De nombreux IDE permettent de développer sur cette plateforme et de nombreux hébergeurs proposent des solutions pour accueillir son application.

Nous avons pu constater les possibilités techniques de Node.js, son fonctionnement en asynchrone qui lui permet d'être très performant et sa rapidité. Mais également le temps réel qui permet aux développeurs de concevoir de nouveaux concepts.

Node.js n'est pas qu'une idée, beaucoup d'entreprises ont déjà fait le pas et de grands groupes comme Groupon ou PayPal ont appliqué cette technologie sur leurs produits, avec des retours extrêmement positifs. Cette technologie ne s'est pas arrêtée aux grandes entreprises, car de nombreuses startups se lancent désormais sur Node.js. Ce sont ces startups qui ont déjà réalisé et popularisé des applications comme Trello ou Voxer, des produits novateurs dans leur domaine.

Les technologies JavaScript ont connu une évolution importante ces dernières années. La convergence des nouvelles technologies du web, les bases de données NoSQL et l'environnement d'exécution Node.js ont apportés une avancée conséquente et ont permis au JavaScript d'étendre son emprise sur la partie serveur.

Mais si cette convergence est bénéfique au JavaScript et permet de programmer avec un seul langage (JavaScript Full-stack), il semblerait que cela impacterait les autres langages serveurs comme le PHP qui de ce fait essaie de repousser ses limites avec la mise en place par Facebook de l'ensemble de logiciels HHVM. Au niveau du JavaScript serveur, cela a permis de créer de nouvelles fonctionnalités comme le temps réel avec son module socket.io. De plus avec l'informatisation du monde et la croissance rapide des machines notamment la puissance des processeurs actuels,

Node.js qui est mono-thread peut exploiter au maximum un cœur de processeur pour être davantage plus puissant.

Toutes ces points importants montrent la puissance du JavaScript côté serveur. Sa technologie Node.js s'implante sur beaucoup de projets web mais les technologies comme Java et PHP reste plus avantageux sur certains cas d'utilisation.

Pour conclure, je dirais que Node.js est la technologie la plus approprié pour certains cas d'utilisations et Java ou PHP sont appropriés à d'autres cas d'utilisation. Cela permet à chacun d'avoir sa part du gâteau mais l'avenir est rempli de surprises. Qui aurait cru un jour que JavaScript serait un concurrent direct de Java et PHP ? C'est pourquoi il ne faut pas tirer de conclusions trop hâtives car il serait fort probable que le JavaScript puisse concurrencer ces langages sur des cadres de projet qui implique des calculs mathématiques à grande échelle où Java domine pour le moment.

Ainsi, le JavaScript côté serveur confirme son évolution et sa pérennité durant pratiquement dix ans et bien plus d'années à venir, cette technologie est déjà très populaire aux États-Unis, souvent précurseur dans tous les domaines technologiques. J'ai pu constater moi-même lors de mes recherches d'alternances durant ces trois dernières années que cette technologie commence à être très demandée, même si encore quelques entreprises n'en connaissent même pas l'existence.

C'est aussi la motivation de ce présent mémoire, il a pour objectif de faire découvrir cette technologie et de vous convaincre de l'adopter, vous guider et vous montrer les cas d'utilisations pour lesquelles Node.js serait un très bon choix tout en sachant que Node.js est quasiment déjà un standard du web !

Glossaire

Java : langage de programmation orienté objet informatique

Architecture : En informatique, architecture désigne la structure générale inhérente à un système informatique, l'organisation des différents éléments du système (logiciels et/ou matériels et/ou humains et/ou informations) et des relations entre les éléments

Framework : Un framework (appelé aussi cadre applicatif, cadre d'applications, cadriciel, socle d'applications ou encore infrastructure de développement) désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).

JavaScript : C'est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js

AngularJS : C'est un framework JavaScript libre et open source développé par Google.

Applications type client lourd : C'est un logiciel qui propose des fonctionnalités complexes avec un traitement autonome. La notion de client s'entend dans une architecture client-serveur. Et contrairement au client léger, le client lourd ne dépend du serveur que pour l'échange des données dont il prend généralement en charge l'intégralité du traitement.

Application web : Appelée web application en anglais, c'est une application manipulable directement en ligne grâce à un navigateur web et qui ne nécessite donc pas d'être installée.

Langage serveur : Un langage serveur, ou plus précisément un langage de script côté serveur (de l'anglais : server-side scripting) est un langage de programmation mis en œuvre sur un serveur HTTP pour produire une page Web dynamique.

Serveur d'hébergement : Un hébergeur web (ou hébergeur internet) est une entité ayant pour vocation de mettre à disposition des internautes des sites web conçus et gérés par des tiers.

HTML : C'est le langage de balisage conçu pour représenter les pages web.

Front-end : En informatique, un front-end ou frontal peut désigner une interface de communication entre plusieurs applications hétérogènes ou un point d'entrée uniformisé pour des services différents. Plus généralement, il s'agit de la mise en place d'un serveur permettant la dissimulation d'un autre. Dans ce cas, le serveur frontal intercepte les requêtes utilisateur et les ré-envoie vers le serveur backend. Le serveur frontal agit donc comme un proxy. La mise en place d'un tel système crée un temps de latence relatif à la distance entre les deux serveurs.

Back-end : En informatique, un back-end (parfois aussi appelé un arrière-plan) est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end (aussi appelé un frontal) qui lui est la partie visible de l'iceberg.

Full Stack : Le full stack est la combinaison du back-end et du front-end

Node.js : Node.js est une plateforme logicielle libre et événementielle en JavaScript orientée vers les applications réseau qui doivent pouvoir monter en charge.

NPM : npm (node package modules) est le gestionnaire de paquets officiel de Node.js.

Socket.IO : C'est une bibliothèque JavaScript pour les applications Web en temps réel

PHP : Hypertext Preprocessor, plus connu sous son sigle PHP (acronyme récursif), est un langage de programmation libre

Paradigme : Un paradigme de programmation est une façon d'approcher la programmation informatique et de traiter les solutions aux problèmes et leur formulation dans un langage de programmation approprié. Il s'oppose à la méthodologie, qui est une manière d'organiser la solution des problèmes spécifiques du génie logiciel.

Ruby : C'est un langage de programmation libre. Il est interprété, orienté objet et multi-paradigme.

HHVM : HipHop Virtual Machine est une machine virtuelle open-source basée sur la compilation juste-à-temps (JIT) qui sert de moteur d'exécution pour les langages de programmation PHP et Hack.

Bibliographie

Site web

https://www.developpez.net

https://stackoverflow.com/

https://openclassrooms.com/

https://www.veracode.com/

https://www.levenez.com/lang/

https://www.altexsoft.com/

https://www.journaldunet.com

http://www.modulecounts.com/

https://www.tiobe.com/

https://softwareengineeringdaily.com/2016/03/03/socket-io-and-realtime-applications-with-guillermo-rauch/

Livres

Titre : Programmation avec Node.js, Express.js et MongoDB : JavaScript côté serveur

Auteur : Eric Sarrion

Date de publication : 2014

Edition : EYROLLES