La notion de front-end

Le développement web se découpe traditionnellement en deux métiers distincts. D'une part, le métier de développeur front-end, d'autre part le métier de développeur back-end. Bien que l'on se retrouve très souvent avec des personnes dans l'obligation de remplir ces deux tâches, il s'agit véritablement de fonction distinctes qui font appelle à des modes de développements différents, tant du point de vue du langage que des résultats visibles.

Par front-end et back-end, on distinge en fait deux morceaux de la chaîne de communication qui permet à un site de fournir une information à l'utilisateur. Cette distinction se pose entre ce qui est calculé du côté serveur, c'est-à-dire la machine qui héberge le site web, et du côté client, c'est à dire du côté de l'ordinateur de l'utilisateur.

Dans le cas d'une ouverture de page web telle que le repository github de ce cours, on distinguera deux types d'interactions avec la page : d'une part, le calcul par le serveur de ce qui est demandé et la communication des données sous la forme d'une page html, d'autre part le calcul par l'ordinateur de l'utilisatrice de ce que veut dire l'ensemble des tags html, des styles CSS et des fonctions javascript qui produisent le résultat visuel pour la lectrice.

Sur l'image suivante, où les étapes sont montrées dans leur ordre écrit en rouge, on distingue la gestion par le serveur de la selection des fichiers à envoyer du rendu final executé par le navigateur en étape 6.

(Changement de slide)

Client ou front-end

Le front-end connaît trois langages principaux : le html, le css et le javascript. Ces trois langages ont connu des versions différentes. Les version actuelles sont : HTML5, CSS3 et Javascript ES7 (où ES veut dire Ecmascript).

(Changement de slide)

La reponsabilité de ces langages peut-être distinguées comme suit : le html est responsable de la structuration de l'information, le css pour la mise en page (style, découpage graphique de la page, animations de transition), le javascript pour l'interaction (automatiquement convertir du texte en markdown, récupérer des informations sur d'autres sites, réaliser des objets complexes, d'une carte à un carroussel/slideshow)

(*Demander si il y a des questions*)

(*Changement de slide*)

La notion de framework

Un framework, ou cadriciel bien que le terme français ne soit JAMAIS utilisé, est un ensemble de composant permettant de développer rapidement l'architecture d'un logiciel (il faut comprendre ici logiciel au sens large : design web, application web, application native, etc.).

Un framework fournit ainsi les éléments essentiels et estimés basiques pour le développement

applicatif. À la différence d'une librairie (ou bibliothèque) qui sert un objectif précis, le framework n'est qu'un ensemble d'outil et de matériaux auxquels il convient de donner sens dans le cadre de son propre développement.

Pour prendre un parallèle, on peut penser les legos comme un framework. Ils fournissent un ensemble de pièces diverses qui, assemblées par des personnes différentes, créeront des résultats très variés.

(*Changement de slide*)

Cependant, les formes de bases contraignent au final le résultat : on ne peut pas produire un BB8 totalement rond, car les pièces ne le permettent pas. C'est un des défauts des frameworks : ils conditionnent souvent les possibles.

Même si le résultat est conditionné, il reste très avantageux d'utiliser un framework : cela rend beaucoup plus rapide et souvent sécurisé le développement des bases d'une applications. Dans le cadre d'une application web, il gèrera par exemple la connexion à une base de données qu'il faudra juste paramètrer.

(*Changement de slide*)

Les frameworks front-end

Article Source : [AltiCreations, Introduction aux frameworks front-end (Bootstrap, Foundation)]

On compte de nombreux types de framework mais celui qui nous intéressera aujourd'hui est celui du front-end. Un framework front-end se distingue des autres frameworks en se concentrant à fournir l'ensembles des outils nécessaires à la création graphique de la structure d'un site web. Il permet *a minima* de prototyper un design rapidement.

Un framework front-end, aussi appelé framework CSS par extension, comprend généralement deux types de fichiers : des fichiers CSS et des fichiers javascripts. Il est, comme l'ensemble des frameworks, accompagné d'une documentation qui propose des snippets, ou morceau de code, offrant des démonstrations de ce que peut faire le framework.

On utilise donc un framework front-end simplement en insérant les fichiers dans son code HTML et en suivant les règles qu'ils établissent

(Changement de slide)

Je suis parti des besoins identifiés par Alexis Blondin d'AltiCréation qui, à mon sens, offre un très bon découpage des ressources nécessaires au développement d'une interface web. pour proposer un système un peu plus rationnel.

On distinguera donc 4 types de besoins fournis par les frameworks frontend :

- la **Typographie et effets de styles**, qui s'étend de la taille de la police aux effets d'affichage liés aux états des éléments (actif/inactif, visible/invisible, survolé/selectionné..)
  (*Changement de slide*)
- le **Layout** (agencement en français) qui représente le découpage en cadre de la page : l'en-tête, le menu, le pied de page. En général, le découpage en lignes et colonnes de la page. Cela comprend l'adaptativité de ce layout aux différentes tailles d'écran ou aux différents types de support. Par exemple, certains frameworks fournissent des règles CSS qui permettent d'épurer la page si elle est imprimée.
  (*Changement de slide*)

- les **Composants**, c'est à dire des éléments récurrents qui se retrouvent dans l'ensemble des groupements agencés et qui spécialisent le comportement d'éléments html: les boutons de formulaires, les listes, les liens, les images..
  (*Changement de slide*)
- les **Modules interactifs ou Widgets** : les défilés d'images, les menus déroulants, etc. qui bien souvent nécessitent en plus du javascript fourni par le framework.
  (*Changement de slide*)

SASS et LESS

Permettons-nous une petite échappée. Vous trouverez très souvent dans les fichiers sources des frameworks des fichiers en .less ou en .scss.
LESS et SCSS , aussi appelé SASS, sont des langages très proches du CSS mais qui apportent des fonctionnalitéss malheureusement inexistantes en CSS normal : ils simplifient l'import de code, ce qui permet de diviser les projets en de nombreux fichiers. Il permet aussi la déclaration de variable qui est ensuite partagé dans l'ensemble des fichiers important celles-ci.

Traditionnellement, on compile le SASS ou le LESS via une application en fichier CSS afin de servir un fichier CSS final aux utilisateurs. Certains sites utilisent des compilateurs qui font ce travail au moment où le visiteur affiche la page. Je me permets de déconseiller ce genre de pratique qui coûte inutilement cher en terme de ressources du côté utilisateur.

(Changement de slide)

Par ailleurs, l'un des grands avantages du sass est sans conteste sa gestion de la hiérarchie qui permet d'écrire du code bien plus proprement en évitant les écueils des feuilles de style en cascade.

(Changement de slide)

Avantages des frameworks front-end

L'utilisation d'un framework apporte évidemment des avantages et je recommande de s'en approprier un pour rapidement développer des interfaces dans le cadre professionnel. Cela permet aussi de faire des démos sans pour autant passer des heures à coder des modules et widgets très courrants.

Je reprends donc les 5 raisons d'AltiCreations. Utiliser un framework frontend représente :
1. Un Gain de temps : on ne développe pas l'ensemble des fonctionnalités de base
2. une assurance de standardisation et performances : les frameworks sont développés par des professionnels qui s'assurent de leur rapidité et de leur compatibilité avec l'ensemble des navigateurs existants
3. une assurance d'évolutivité et de mises à jour : le web et le graphisme avance à une vitesse incroyable. Il y a 15 ans, faire un design sans table était impossible ou presque. Dans les 5 à 10 dernières années, la part de navigation mobile a explosé, tout autant que la disparité des navigateurs. Les frameworks évoulent et suivent ces tendances pour assurer une solidité à toute épreuve
4. ils sont gratuits et libres, très souvent (je ne connais pas de framework front end payant). Je n'ai pas besoin d'expliquer celle-ci plus que cela.
5. Ils sont fiables. Fiables car ils sont portés soit par de grandes entreprises soit par des communautés qui sont aujourd'hui immenses et qui assurent de fait un suivi des bugs sans pareil.

(Changement de slide)

Défauts des frameworks front-end

Cependant, ne rêvont pas, les frameworks ont aussi leurs défauts

i. Ils produisent des sites qui se ressemblent, parce que beaucoup de personne ne repersonnalise pas le code
ii. On se retrouve avec beaucoup de code unitilisé qui alourdit les pages : beaucoup de personnes n'utilisent pas les alertes par exemples ou les slideshow, pourtant ils sont déclarés dans le code.
iii. Si on ne fait pas que du prototypage mais que l'on cherche à coller à un graphisme donné, il y a la nécessité de modifier ou ajouter du code
iv. Mais modifier ou adapter le code originel du framework peut être long voire laborieux. Vraiment.
v. Enfin, il existe un risque de conflits avec d'autres bibliothèques ou plugins qui définiraient eux-mêmes leurs classes. Un exemple assez commun que j'ai rencontré est un bug sur des dessins vectoriels en XML SVG qui posaient problèmes avec bootstrap. Mais un autre exemple est possible : imaginons que j'utilise le framework X mais que framework Y est utilisé par la librairie que j'utilise, que va-t-il résulter de ce conflit ?

Quelques exemples

Les frameworks les plus connus sont sans doute Bootstrap, Foundation et Semantic UI. Dans le cadre de ce cours, nous utiliserons bootstrap mais rien ne vous empêche de vous essayez aux deux autres ou à d'autres un peu moins connus.

(Demander si questions)
(Changement de slide vers TD.MD)