Pour une page web qui dure 10 ans ?

Des pages web légères et moins gourmandes en ressources, du « low-tech » c'est plus écologique probablement, mais c'est aussi une des conditions pour rendre durables des contenus qui ont une fâcheuse tendance à se volatiliser… Jeff Huang est professeur d'informatique et dans la page que Framalang a traduite pour vous, il fait le pari que son contenu sera encore accessible dans dix ans au moins, tout en proposant 7 recommandations pour créer des pages web pérennes.

Il ne s'agit pas de solutions miracles, mais plutôt d'incitations à l'action et au débat, comme il l'écrit :

> *Cet article est destiné à provoquer et à susciter une action individuelle, et non à proposer une solution complète pour soigner le Web en déclin*

Donc il est clair que les conseils qu'il donne sont peut-être incomplets ou critiquables. Certain·e·s (comme un de nos traducteurs) vont par exemple sursauter de lire que les polices Google peuvent être utilisées car elles sont « probablement déjà dans le cache ». D'autres vont regretter que le recours à l'archivage ne soit pas assez mis en avant comme le fait cipherbliss dans cet article… bref, n'hésitez pas à ajouter des critiques constructives.

N'hésitez pas non plus à nous dire s'il existe vraiment des contenus web qui méritent selon vous d'être maintenus dix ans ou plus, et lesquels (créations et expressions personnelles, ressources des Communs, etc.), ou bien si vous acceptez avec fatalisme ou satisfaction que les pages web, comme tout le reste, finissent par disparaître…

Les commentaires, comme toujours sur ce blog, sont ouverts et

modérés.

Page originale : This Page is Designed to Last, A Manifesto for Preserving Content on the Web
Traduction Framalang : goofy, Côme, mo, wisi_eu, retrodev, tykayn

Un manifeste pour la pérennité des contenus sur le Web

Cette page est conçue pour durer

par Jeff Huang

Pour un professeur, la fin de l'année est l'occasion de faire le ménage et de se préparer pour le semestre à venir. Je me suis retrouvé à effacer de vieux marque-pages, eh oui, des marque-pages : cette fonctionnalité des navigateurs autrefois si appréciée qui paraît avoir perdu la bataille contre la « complétion automatique dans la barre d'adresse ». Mais ce geste nostalgique de nettoyage a fini par me déprimer.

Les uns après les autres, les marque-pages m'ont mené vers des liens morts. Disparus, de superbes écrits de kuroShin sur la culture technologique ainsi qu'une série de puzzles mathématiques et les discussions associées des universitaires que mon père m'avait présentés ; disparus aussi les tutoriels d'ingénierie inverse de Woodman qui datent de mes années d'étude, avec lesquels j'ai découvert le sentiment d'avoir le pouvoir sur les logiciels ; même mes plus récents marque-pages ont disparu, une série d'articles exposant sur Google+ la non-

conformité des chargeurs usb-c avec les normes…

Ce n'est pas seulement une question de liens fichus, c'est le problème de la difficulté croissante de maintenir en vie des contenus indépendants sur le Web, conduisant à une dépendance à des plateformes et à des formats de publication qui s'empilent de façon chronologique (blogs, fils, tweets…)

Bien sûr j'ai moi aussi contribué au problème. Un article que j'ai publié il y a sept ans comprend un résumé initial dans lequel un lien vers une démo a été remplacé par une page de spam avec une image de citrouille. C'est en partie à cause de la flemme de devoir renouveler et de maintenir une application web fonctionnelle année après année.
J'ai recommandé à mes étudiants de publier des sites web avec Heroku et des portfolios avec Wix. Mais toute plateforme au contenu irremplaçable meurt un jour. Geocities, LiveJournal, what.cd, maintenant Yahoo Groups. Un jour, Medium, Twitter et même les services d'hébergement comme GitHub Pages seront pillés puis jetés lorsqu'ils ne pourront plus se développer ou n'auront pas trouvé de modèle économique viable.

C'est un problème de nature plurielle.
Tout d'abord, maintenir du contenu demande du travail. Le contenu peut avoir besoin d'être mis à jour pour rester pertinent et devra éventuellement être ré-hébergé. Une grande partie du contenu – ce qui était autrefois la grande majorité du contenu – a été mise en place par des individus. Mais les particuliers (peut-être vous ?) finissent par se désintéresser, si bien qu'un jour, vous ne voudrez peut-être plus vous occuper de la migration d'un site web vers un nouvel hébergeur.

Deuxièmement, le nombre croissant de bibliothèques et de *frameworks* rend le Web plus sophistiqué, mais également plus complexe. Il y a d'abord eu jquery, puis bootstrap, npm, angular, grunt, webpack, et bien d'autres. Si vous êtes un développeur web qui se tient au courant des dernières

nouveautés, alors ce n'est pas un problème.

Mais dans le cas contraire, vous êtes peut-être programmeur de systèmes embarqués, directeur technique d'une start-up, un développeur Java d'entreprise ou un doctorant de chimie. Vous avez sans doute trouvé le moyen de mettre en place un serveur web et sa chaîne d'outils, mais continuerez-vous à le faire d'une année à l'autre, d'une décennie à l'autre ? Probablement pas ! Et lorsque, l'année suivante, vous rencontrerez un problème de dépendance à un paquet ou que vous découvrirez comment régénérer vos fichiers html, vous pourriez abandonner et zipper les fichiers pour vous en occuper « plus tard ».

Même les piles technologiques simples comme les générateurs de sites statiques (par exemple, Jekyll) nécessitent du travail et cesseront de fonctionner à un moment donné. Vous tombez dans l'enfer des dépendances aux paquets NPM, et vous oubliez la commande pour réaliser un déploiement. Et avoir un site web avec plusieurs pages html est complexe ; comment savoir comment chaque page est liée aux autres : « index.html.old », une copie de « about.html » « index.html (1) », « nav.html » ?

Troisièmement, et cela a déjà été rapporté par d'autres (et même réfuté), la disparition du Web « public » au profit du mobile et des applications web, des jardins clos (telles que les pages Facebook), du chargement en temps réel des WebSockets et de l'AMP diminue la proportion même de la toile dans la toile mondiale, qui ressemble désormais davantage à une toile continentale qu'à une « toile mondiale » (*World Wide Web*).

Alors, face à ces problèmes, que pouvons-nous faire ? Ce n'est pas un problème si simple qu'il puisse être résolu dans cet article. La Wayback Machine et archive.org permettent de conserver certains contenus plus longtemps. Et il arrive qu'un individu altruiste relocalise le contenu ailleurs.

Mais la solution se trouve sur plusieurs fronts. Comment créer

un contenu web qui puisse durer et être maintenu pendant au moins dix ans ? Étudiant l'interaction homme-machine, je pense naturellement aux parties prenantes que nous n'aidons pas : actuellement, la mise en ligne de contenu web est optimisée soit pour le développeur web professionnel (qui utilise les derniers frameworks et flux de travail), soit pour l'utilisateur non averti (qui utilise une plateforme).

Mais je pense que nous devrions considérer à la fois 1) le « responsable » occasionnel du contenu web, quelqu'un qui ne reste pas constamment à jour avec les dernières technologies web, ce qui signifie que le site web doit avoir de faibles besoins de maintenance ; 2) et les robots d'indexation qui préservent le contenu et les archiveurs personnels, « archiveur » implique que le site web doit être facile à sauvegarder et à interpréter.

Ma proposition consiste donc en sept lignes directrices non conventionnelles sur la manière dont nous traitons les sites web conçus pour être informatifs, pour les rendre faciles à entretenir et à préserver. Ma suggestion est que le responsable de la maintenance s'efforce de maintenir le site pendant au moins 10 ans, voire 20 ou 30 ans. Il ne s'agit pas nécessairement de points de vue controversés, mais d'aspirations qui ne sont pas courantes – un manifeste pour un site web durable.

1. Revenez à du HTML/CSS « Vanilla » (NdT : le plus simple, sans JavaScript) – Je pense que nous avons atteint le point où le html/css est plus puissant et plus agréable à utiliser que jamais. Au lieu de commencer avec un modèle obèse bourré de fichiers « .js », il est maintenant possible de simplement écrire en HTML, à partir de zéro. Les CSS « Flexbox » et « Grid », le canvas, les Selectors, le box-shadow, l'élément vidéo, le filtre, etc. éliminent une grande partie du besoin de bibliothèques JavaScript. Nous pouvons éviter le jQuery et le Bootstrap, car ils deviennent de toute façon moins pertinents. Plus il y a de bibliothèques intégrées au site

web, plus celui-ci devient fragile. Évitez les polyfills et les préfixes CSS, et n'utilisez que les attributs CSS qui fonctionnent sur tous les navigateurs. Et validez fréquemment votre HTML ; cela pourrait vous éviter un mal de tête à l'avenir lorsque vous rencontrez un bogue.

2. Ne réduisez pas ce HTML. Réduire (compresser) votre HTML et les CSS/JS associés vous semblera peut-être une précieuse économie de bande passante, et toutes les grandes entreprises le font. Pourquoi ne pas le faire ? Eh bien, vous n'économisez pas beaucoup parce que vos pages web doivent être compressées avant d'être envoyées sur le réseau, donc la réduction préventive de votre contenu compte probablement très peu dans l'économie de bande passante, voire pas du tout. Mais même si cela permettait d'économiser quelques octets (ce n'est après tout que du texte), vous devez maintenant avoir un processus de construction et l'ajouter à votre flux de travail, de sorte que la mise à jour d'un site web devient plus complexe. En cas de bogue ou d'incompatibilité future dans le HTML, le format minimisé est plus difficile à déboguer. De plus, il est peu convivial pour vos utilisateurs ; de nombreuses personnes commencent à utiliser le HTML en cliquant sur « Voir la source »[1], et la réduction de votre HTML empêche cet idéal d'apprentissage en regardant ce qui est fait. Réduire le HTML ne préserve pas sa qualité pédagogique, et ce qui est archivé n'est que le tas de code résultant.

3. Préférez maintenir une page plutôt que plusieurs. Plusieurs pages sont difficiles à maintenir. Vous pouvez oublier quelle page renvoient à quoi, et cela nécessite également la mise en place de modèles pour limiter les redondances. Combien de pages une personne peut-elle réellement maintenir ? Avoir un seul fichier, probablement juste un « index.html », est simple et facile à retenir. Profitez de ce défilement vertical infini. Vous n'aurez jamais besoin de fouiller dans vos fichiers ou de faire du grep pour voir où se trouve un certain contenu. Et comment gérer les multiples versions de ce

fichier ? Devriez-vous utiliser git ? Les placer dans un dossier « old/ » ? J'aime l'approche simple qui consiste à nommer les anciens fichiers avec la date à laquelle ils ont été retirés, comme index.20191213.html. L'utilisation du format ISO de la date permet de trier facilement, et il n'y a pas de confusion entre les formats de date américain et européen. Si j'ai plusieurs versions en une journée, j'utiliserais un style similaire à celui qui est habituel dans les fichiers journaux, index.20191213.1.html. Un effet secondaire agréable est que vous pouvez alors accéder à une version plus ancienne du fichier si vous vous souvenez de la date, sans vous connecter à la plateforme d'hébergement web.

4. Mettez fin à toutes les formes de liaison automatique (hotlinking). Ce mot d'avertissement semble avoir disparu du vocabulaire internet, mais c'est l'une des raisons pour lesquelles j'ai vu un site web parfaitement bon s'effondrer sans raison. Cessez d'inclure directement des images provenant d'autres sites web, cessez d'« emprunter » des feuilles de style en vous contentant de créer des liens vers celles-ci, et surtout cessez de créer des liens vers des fichiers JavaScript, même ceux qui sont hébergés par les développeurs d'origine. Les liaisons automatiques sont généralement considérées comme impolies car vos visiteurs utilisent la bande passante de quelqu'un d'autre, elles ralentissent l'expérience utilisateur, permettent un autre site web de pister vos utilisateurs et, pire encore, si l'endroit auquel vous vous connectez modifie la structure de ses dossiers ou se déconnecte tout simplement, la panne se répercute également sur votre site web. Google Analytics est inutile ; stockez vos propres journaux de serveur et configurez GoAccess ou découpez-les comme vous le souhaitez, ce qui vous donnera des statistiques plus détaillées. Ne donnez pas vos journaux à Google gratuitement.

5. N'utilisez que les 13 polices de caractères adaptées au Web – nous nous concentrons d'abord sur le contenu, comprenez :

les caractères décoratifs et inhabituels sont complètement inutiles. Maintenez un petit éventail et les 13 polices de caractères adaptées au Web. Peut-être avez-vous vraiment besoin d'une police de caractères néo-grotesque qui ne soit pas Arial/Helvetica, ou d'une police de caractères géométriques. Dans ce cas, utilisez le minimum nécessaire, comme Roboto (pour le néo-grotesque) et Open Sans (pour le géométrique) ; ce sont les deux polices les plus populaires de Google Fonts, il est donc probable qu'elles soient déjà en cache sur l'ordinateur de vos utilisateurs. Outre ces polices, votre objectif doit être de fournir le contenu à l'utilisateur de manière efficace et de faire en sorte que le choix de la police soit invisible, plutôt que de flatter votre ego en matière de conception. Même si vous utilisez les polices Google, elles n'ont pas besoin d'être liées automatiquement (hotlink). Téléchargez le sous-ensemble dont vous avez besoin et fournissez-les localement à partir de vos propres dossiers.

6. Compressez vos images de manière obsessionnelle. Ce sera plus rapide pour vos utilisateurs, moins gourmand en espace d'archivage et plus facile à maintenir lorsque vous n'avez pas à sauvegarder un énorme dossier. Vos images peuvent avoir la même haute qualité, mais être plus petites. Minimisez vos SVG, compressez sans perte vos PNG, générez des JPEG pour qu'ils correspondent exactement à la largeur de l'image. Cela vaut la peine de passer du temps à trouver la meilleure façon de compresser et de réduire la taille de vos images sans perte de qualité. Et une fois que WebP sera pris en charge par Safari, passez à ce format. Réduisez impitoyablement la taille totale de votre site web et gardez-la aussi petite que possible. Chaque Mo peut coûter cher à quelqu'un ; en effet mon opérateur de téléphonie mobile (Google Fi) facture un centime (de dollar) par Mo de données. Ainsi, un site web de 25 Mo, assez courant de nos jours, coûte lui-même 25 centimes, soit à peu près autant qu'un journal quand j'étais enfant.

7. Éliminez le risque de rupture d'URL. Il existe des services

de contrôle qui vous indiqueront quand votre URL est en panne, ce qui vous évitera de réaliser un jour que votre page d'accueil ne charge plus depuis un mois et que les moteurs de recherche l'ont désindexée. Car 10 ans, c'est plus long que ce que la plupart des disques durs ou des systèmes d'exploitation sont censés durer. Mais pour éliminer le risque de panne totale d'une URL, mettez en place un second service de contrôle. En effet, si le premier s'arrête pour une raison quelconque (passage à un modèle payant, fermeture, oubli de renouvellement, etc.), vous recevrez toujours une notification lorsque votre URL est hors service, puis vous réaliserez que l'autre service de surveillance est hors service parce que vous n'avez pas reçu la deuxième notification. N'oubliez pas que nous essayons de maintenir un service pendant plus de 10 ans (idéalement bien plus longtemps, même 30 ans), donc beaucoup de services vont s'arrêter pendant cette période, donc deux services de surveillance, c'est plus sûr…
Après avoir fait cela, placez un texte dans le pied de page, « La page a été conçue pour durer », avec un lien vers cette page expliquant ce que cela signifie. Ces quelques mots attestent que le responsable fera de son mieux pour suivre les idées de ce manifeste.

Avant que vous ne protestiez, il est évident que cela n'est pas adapté pour les applications web. Si vous créez une application, alors créez votre application web ou mobile avec le flux de travail dont vous avez besoin. Je ne connais pas une seule application web qui ait fonctionné de manière identique pendant 10 ans (sauf le tutoriel python de Philip Guo, en raison de sa stratégie minimaliste de maintenance), donc cela semble être une cause perdue de toute façon. Ce n'est pas non plus adapté pour les sites web maintenus par une organisation comme Wikipédia ou Twitter. Vous faites votre truc, et le salaire d'une équipe informatique est probablement suffisant pour maintenir quelque chose en vie pendant un certain temps.
En fait, il n'est même pas si important de suivre strictement

les 7 « règles », car ce sont plus des incitations que des règles impératives.

Mais admettons qu'une petite partie du Web commence à concevoir des sites web dont le contenu est censé durer. Que se passe-t-il alors ? Eh bien, les gens préféreront peut-être créer des liens vers ces sites car leur accès est garanti à l'avenir. Plus généralement, les gens peuvent être plus soucieux de rendre leurs pages plus permanentes. Et les utilisateurs et utilisatrices ainsi que les robots qui archivent économisent de la bande passante lorsqu'ils visitent et stockent ces pages.

Les effets sont à long terme, mais les réalisations sont progressives et peuvent être mises en œuvre par les propriétaires de sites web sans dépendre de quiconque ni attendre un effet de réseau. Vous pouvez le faire dès maintenant pour votre site web, et ce serait déjà un résultat positif. C'est comme utiliser un sac de courses recyclé au lieu d'un sac en plastique, c'est une petite action individuelle.

Cet article est destiné à provoquer et à susciter une action individuelle, et non à proposer une solution complète au déclin de la Toile. Il s'agit d'un petit pas simple pour un système sociotechnique complexe. J'aimerais donc voir cela se produire. J'ai l'intention de maintenir cette page pendant au moins 10 ans.

Merci à mes étudiants en doctorat Shaun Wallace, Nediyana Daskalova, Talie Massachi, Alexandra Papoutsaki, mes collègues James Tompkin, Stephen Bach, mon assistante d'enseignement Kathleen Chai et mon assistant de recherche Yusuf Karim pour leurs commentaires sur les versions précédentes.

Voir les discussions sur Hacker News et reddit/r/programming

Cette page est conçue pour durer.

Image réalisée avec https://framalab.org/gknd-creator/