Bonne pratique n°1

Éliminer les fonctionnalités non essentielles

PRIORITAIRE	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

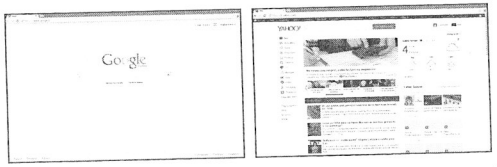
Plusieurs études (Cast Software et Standish Group, notamment) démontrent que 70 % des fonctionnalités demandées par les utilisateurs ne sont pas essentielles et que 45 % ne sont jamais utilisées.

En réduisant la couverture et la profondeur fonctionnelle de l'application, on abaisse son coût de développement initial, sa dette technique et les impacts environnementaux associés. On diminue donc ainsi mécaniquement l'infrastructure nécessaire à son exécution. Par ailleurs, à niveau ergonomique constant, plus l'application est pauvre fonctionnellement, plus elle sera simple à utiliser. Il faut donc réduire le plus possible la couverture fonctionnelle de l'application, en la centrant sur le besoin essentiel de l'utilisateur.

Si l'application est déjà développée, il faut mesurer le taux d'utilisation des fonctionnalités et, si l'architecture applicative le permet, désactiver, désinstaller ou supprimer les fonctionnalités non utilisées.

Exemple

Les succès récents du Web - Google, Twitter, WhatsApp, Pinterest, Instagram, etc. - fournissent un seul service et misent sur une grande sobriété fonctionnelle.



Plus sobre fonctionnellement, l'interface de Google est aussi deux fois plus légère à télécharger que celle de Yahoo!.

Bonne pratique n° 2

Quantifier précisément le besoin

PRIORITAIRE	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Les « mensurations » de chaque fonctionnalité doivent être définies précisément et dans leur ensemble. Il peut s'agir d'un taux de compression pour les images de l'interface graphique, du temps de réponse maximum pour une requête HTTP, du nombre d'items affichés dans une liste, etc.

Plus les « mensurations » et exigences associées à chaque fonctionnalité collent au métier, plus on évite la surqualité. La logique doit donc être inversée par rapport aux habitudes actuelles. Si une information n'est pas précisée, c'est le niveau de qualité ou la quantité minimale qui est proposé. Par exemple, en l'absence de précision, le nombre d'items d'une liste est limité à 5 éléments ou au nombre maximal affichable sur le plus petit écran cible de l'application.

Exemple

Gain potentiel : en jouant sur le nombre d'items affichés sur la page de résultats de son moteur de recherche Bing, Microsoft Research a démontré qu'il était possible de réduire jusqu'à 80 % l'infrastructure physique (nombre de serveurs) sous-jacente.

Bonne pratique n° 3

Fluidifier le processus

CONSEILLÉ	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE MOYEN

RESSOURCES ÉCONOMISÉES

Le temps passé par l'utilisateur sur un site web ou un service en ligne est le facteur le plus déterminant pour réduire ou augmenter l'empreinte environnementale de ce site. Autrement dit, moins l'utilisateur passe de temps sur le site ou service en ligne, plus on réduit les impacts environnementaux associés.

Il faut donc veiller à réduire au minimum le nombre d'écrans, d'étapes et d'interactions inutiles, sans pour autant créer des écrans trop riches et complexes, qui seraient alors contre-productifs.

Exemple

Si le processus en ligne commence par un e-mail et qu'il implique la fourniture de pièces justificatives par l'internaute, on demandera ces pièces dans l'e-mail avant que l'internaute ne se connecte. Il peut ainsi les préparer avant de démarrer la session web, et compléter ainsi le processus en ligne en une seule session au lieu de deux.

Gain potentiel : jusqu'à deux fois moins de gaz à effet de serre émis, selon le type et la durée du processus.

Bonne pratique n° 4

Préférer la saisie assistée à l'autocomplétion

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Le complètement automatique (autocomplétion) guide les utilisateurs en complétant automatiquement la fin du texte saisi dans un champ. Cette fonctionnalité est parfois très pratique pour éviter des erreurs ou suggérer un axe de recherche, mais elle nécessite des allers-retours incessants entre le navigateur et le serveur (malgré la possibilité de « caper » les échanges). Le navigateur envoie en effet chaque nouveau caractère ou mot saisi au serveur, qui lui renvoie un texte pour compléter la saisie de l'utilisateur. Le volume de données échangées est très faible, mais il sollicite beaucoup les serveurs et le réseau en termes de requêtes.

Dans la mesure du possible, cette fonctionnalité est à éviter et à remplacer si possible par la saisie assistée. Cela consiste à guider l'utilisateur par un ensemble d'informations et d'indices (présentation du format attendu en grisé dans le champ de saisie, réaction de l'interface avec un message d'erreur, une aide lorsque la saisie est incorrecte...). Les interactions liées à la saisie assistée sont gérées localement, ce qui réduit les échanges avec le serveur.

Exemple

Gain potentiel : à chaque fois que l'on utilise la saisie assistée pour une fonctionnalité, plutôt que l'autocomplétion, on réduit le nombre de requêtes associées par un facteur 10.

Bonne pratique n° 5

Respecter le principe de navigation rapide dans l'historique

CONSEILLÉ	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Les navigateurs possèdent une fonction de navigation rapide dans l'historique (boutons Page précédente et Page suivante). Grâce à cette fonction, il n'est pas nécessaire de redemander la page au serveur, puis de la recharger. On évite ainsi de consommer inutilement de la bande passante et de générer des requêtes HTTP supplémentaires.

Par conséquent, ne pas prévoir d'éléments qui rendent la page inutilisable après l'avoir quittée.

Exemple

Éviter :

- les boutons de formulaire qui se désactivent lors de la soumission ;
- les menus qui ne fonctionnent plus après un clic ;
- les effets JavaScript qui font disparaître le contenu de la page lorsqu'on la quitte.

Bonne pratique n°6

Favoriser un design simple, épuré et adapté au Web

PRIORITAIRE	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT
RESSOURCES ÉCONOMISÉES		

Exploiter les fonctionnalités d'HTML5 et de CSS3 pour concevoir le design d'un site. Il ne faut surtout pas imaginer un site sans prendre en compte les contraintes techniques, puis essayer de le réaliser.

Exemple

Proposer des dégradés de couleurs, un style de coins arrondis, etc., qui puissent être réalisés en CSS3 au lieu de nécessiter des images.

```
/* Exemple de bords arrondis */
.box_round {
  -webkit-border-radius: 12px; /* Saf3-4, iOS
1-3.2, Android ≤1.6 */
          border-radius: 12px; /* Opera 10.5, IE9,
Saf5, Chrome, FF4+, iOS 4, Android 2.1+ */
  /* useful if you don't want a bg color from
leaking outside the border: */
  -moz-background-clip: padding; -webkit-background-
clip: padding-box; background-clip: padding-box;
}

/* Exemple d'ombre sous la boite */
.box_shadow {
  -webkit-box-shadow: 0px 0px 4px 0px #ffff; /*
Saf3-4, iOS 4.0.2 - 4.2, Android 2.3+ */
          box-shadow: 0px 0px 4px 0px #ffff; /*
Opera 10.5, IE9, FF4+, Chrome 6+, iOS 5 */
}
```

Imaginer des bandeaux, headers, etc., de sorte que le texte, les fonds de couleur, etc. soient réalisables en HTML/CSS. Pour plus d'exemples, voir :

http://css3please.com

Bonne pratique n°7

Préférer l'approche « mobile first » ou, à défaut, RESS plutôt que RWD

CONSEILLÉ	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FOR
RESSOURCES ÉCONOMISÉES		

Lorsque le contexte le permet, privilégier l'approche « mobile first » qui consiste à concevoir un site/service en ligne pour les terminaux mobiles.

Et n'élargir sa couverture fonctionnelle pour de plus grands écrans que si l'apport fonctionnel/ergonomique est justifié. Dans ce cas, opter alors pour l'architecture *Responsive Design + Server Side Components* (RESS). Cette architecture reprend les principes RWD (adaptation automatique de l'interface au contexte d'utilisation) mais sélectionne côté serveur les ressources qui seront envoyées au terminal. On s'assure ainsi de ne pas consommer inutilement de la bande passante, ni de trop solliciter le processeur et la mémoire du terminal pour des traitements inutiles. Il s'agit de pousser à l'extrême la bonne pratique qui consiste à fournir du code spécifique à un navigateur en particulier.

Parmi les solutions clés en main, RESS.io automatise la mise en œuvre de bonnes pratiques responsive orientées efficience, comme servir des images redimensionnées pour les petits écrans, compresser automatiquement (en gzip) les pages et les ressources CSS et JavaScript en sortie de serveur HTTP, fusionner les feuilles de styles, etc.

Exemple

La mise en place d'une architecture RESS plutôt qu'une approche responsive design standard (RWD) peut aller jusqu'à diviser par 4 la bande passante consommée, pour de meilleurs temps de réponse. Plus la taille de l'écran est petite, plus l'approche RESS sera intéressante.

Bonne pratique n° 8

Proposer un traitement asynchrone lorsque c'est possible

CONSEILLÉ	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE MOYEN

RESSOURCES ÉCONOMISÉES

Lorsque l'interaction avec l'utilisateur induit un traitement lourd et long côté serveur, proposer un traitement asynchrone lorsque c'est possible. L'idée est d'encourager l'utilisateur à déclencher le traitement, puis à se reconnecter quand celui-ci est terminé ; par exemple, via la réception d'un e-mail contenant un lien.

Cette approche permet de réaliser des traitements par lots (batchs), souvent plus efficients en ressources que des traitements synchrones à la volée. On libère ainsi les serveurs de présentation, qui peuvent prendre en charge d'autres internautes pendant que le traitement s'effectue en mode asynchrone côté serveur.

Exemple

Dans le cas d'un service en ligne de conversion de documents bureautiques, inciter l'utilisateur à déposer ses fichiers en une seule fois, puis l'avertir par e-mail lorsque le traitement est terminé. Pour optimiser le processus, l'ensemble des fichiers peut être regroupé et compressé dans une archive.

Bonne pratique n° 9

Limiter le nombre de requêtes HTTP

PRIORITAIRE	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Le temps de chargement d'une page côté navigateur est directement corrélé au nombre de fichiers que le navigateur doit télécharger, et au poids unitaire de chaque fichier.

Pour chaque fichier, le navigateur émet un GET HTTP vers le serveur. Il attend sa réponse, puis télécharge la ressource dès qu'elle est disponible. Selon le type de serveur web que vous utilisez, plus le nombre de requêtes par page est important, moins vous pourrez servir de pages par serveur. Diminuer le nombre de requêtes par page est crucial pour réduire le nombre de serveurs HTTP nécessaires au fonctionnement du site, et donc les impacts environnementaux associés.

De nombreuses approches permettent de réduire le nombre de requêtes par page :

- combiner les fichiers statiques : librairies CSS et JavaScript, notamment (voir la bonne pratique n° 81) ;
- utiliser un sprite CSS (voir la bonne pratique n° 23) pour regrouper les images de l'interface ;
- préférer les glyphes aux images (voir la bonne pratique n° 18) et plus généralement les images vectorielles aux matricielles ;
- mettre en cache navigateur tout ce qui peut l'être.

Exemple

Gain potentiel : réduction de la charge serveur, donc du nombre d'équipements nécessaires - de leur empreinte environnementale et économique -, des serveurs HTTP jusqu'aux serveurs d'applications et aux SGBD/R.

Bonne pratique n° 10

Stocker localement les données statiques

PRIORITAIRE	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

L'une des grandes nouveautés de HTML 5 est la possibilité de stocker localement des données structurées sous différentes formes : paires valeurs-clés (Web Storage), base de données relationnelle SQL (IndexedDB), et la mise en cache applicatif (Service Workers).

L'intérêt du stockage local est double. D'une part, on évite les allers-retours inutiles avec le serveur, ce qui économise des ressources et du temps de réponse. D'autre part, comme les données sont locales, il est plus facile et plus rapide de les manipuler au sein de l'interface.

À l'heure où nous écrivons ces lignes (juillet 2015), le support par les dernières versions des navigateurs de Web Storage est total et celui de IndexedDB est bon. On peut consulter le site *www.caniuse.com*, très utile pour tester l'avancée du support de chacune de ces approches par les navigateurs.

Exemple

Gain potentiel : réduction de la charge serveur, donc du nombre d'équipements nécessaires (de leur empreinte environnementale et économique), des serveurs HTTP jusqu'aux mainframes.

Bonne pratique n° 11

Choisir les technologies les plus adaptées

PRIORITAIRE	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FO
		

RESSOURCES ÉCONOMISÉES 

Le choix des technologies étant primordial pour optimiser les ressources, sélectionner l'outil le plus économe en fonction de ses besoins et de ses contraintes métier.

Voici les cinq grandes familles de solutions disponibles, classées de la plus à la moins performante en termes de green IT :

- site statique (réalisé avec un logiciel spécialisé tel que Dreamweaver, ou avec un éditeur de code) ;
- site généré (par exemple avec Jekyll, outil basé sur Ruby qui apporte des systèmes d'inclusion de templates, des mécanismes de génération d'URL, etc.) ;
- site dynamique développé sur mesure (avec PHP, J2EE, .NET, etc.) ;
- site dynamique développé sur mesure avec un framework (de type Symfony) ;
- site dynamique développé avec un CMS (comme Drupal, Joomla!, Jahia, etc.).

En effet, plus la solution retenue est « packagée », plus elle empile des couches d'abstraction qui dégradent la performance.

Exemple

Un système de tchat sera bien plus performant et économique s'il est développé en JavaScript via Node.js qu'avec une solution PHP.

Bonne pratique n° 12

Utiliser un framework ou développer sur mesure

CONSEILLÉ	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Utiliser un framework ou un développement sur mesure, afin de bénéficier d'une plus grande liberté dans l'optimisation de certains processus. Les CMS sont en effet plus contraignants et imposent des fonctionnements parfois gourmands en ressources pour atteindre leur principal objectif, la souplesse.

Ainsi, pour la gestion de ses modules, le CMS Drupal utilise un système de « hook », qui repose sur une convention de nommage des fonctions contenues dans ces modules. Mais tester l'existence de fonctions est un processus qui consomme des ressources. Tandis que les développements sur mesure n'ont pas à « découvrir » l'existence de fonctions puisqu'elles sont déjà connues.

Exemple

Éviter ce type d'implémentation lorsque c'est possible :

```
<?php
foreach ($list as $module) {
  if (function_exists($module . '_' . $hook)) {
    // Do some stuff here...
  }
}
?>
```


Bonne pratique n° 13

Limiter le recours aux plug-ins

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE MOYEN

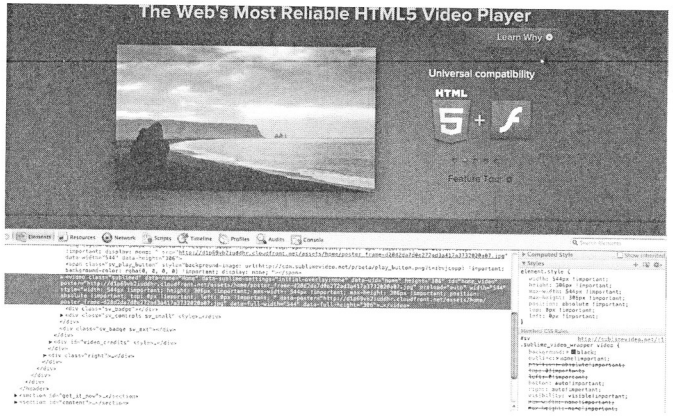
RESSOURCES ÉCONOMISÉES

Éviter d'utiliser des plug-ins (Flash Player, machines virtuelles Java et Silverlight, etc.), car certains consomment beaucoup de ressources (cycles CPU et mémoire vive). C'est notamment le cas du Flash Player d'Adobe, à tel point qu'Apple a décidé de ne pas installer cette technologie sur ses terminaux mobiles afin de garantir une meilleure autonomie.

Préférer les technologies standards comme HTML5, ECMAScript, etc.

Exemple

Pour réaliser un back office basé sur du *drag and drop*, préférer l'utilisation de jQuery à celle de Flash.



Les navigateurs compatibles HTML5 peuvent lire nativement des vidéos afin d'éviter l'utilisation d'un plug-in (source : Jilion).

Bonne pratique n° 14

Limiter l'utilisation de Flash

CONSEILLÉ	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Pour animer des éléments d'une page, préférer HTML + JavaScript à Flash, qui requiert l'installation du plug-in Flash Player. Même si la performance pure du couple HTML + JavaScript - en termes de FPS *(Frame Per Second)*, par exemple - est moins bonne, elle demeure en effet suffisante dans la majorité des cas.

En outre, les bibliothèques JavaScript (comme jQuery) étant déjà chargées pour d'autres parties du site, on évite le téléchargement d'une bibliothèque spécifique, et donc l'ajout ou le téléchargement de code supplémentaire.

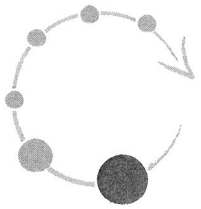
Exemple

Pour construire un slider avec des effets variés, jQuery fournit un ensemble de méthodes dédiées permettant de jouer sur :

- les couleurs ;
- les classes CSS ;
- 16 effets de transition ;
- des dizaines de déplacements ;
- ...

Pour aller plus loin :

http://jqueryui.com/demos



Templating

L'intégration, qui consiste à traduire la maquette d'un site web en une interface graphique utilisateur fonctionnelle basée sur HTML5, CSS3 et JavaScript, est une étape importante en matière d'éco-conception web. Nombre de choix qui seront faits à ce niveau vont avoir un impact sur la bande passante consommée, sur le poids total des images, sur la consommation CPU du navigateur, etc.

Les bonnes pratiques que nous présentons dans ce chapitre ne sont pas compliquées à mettre en œuvre, mais elles sont trop souvent reléguées au rang d'optimisation à réaliser en fin de projet. Or, comme nous le savons tous, ces optimisations sont rarement appliquées car l'urgence en fin de projet est souvent de livrer... pas trop en retard. Il faut donc absolument intégrer les bonnes pratiques que nous vous présentons dès le début, dans un esprit d'intégration continue.

Lors de cette étape, les bonnes pratiques consistent à minimiser les allers-retours avec le serveur, à réduire le poids des ressources graphiques, et à utiliser correctement tout le potentiel des feuilles de styles.

HTML 59
POLICES 61
IMAGES 63
CSS 67

Bonne pratique n° 15

Valider les pages auprès du W3C

CONSEILLÉ	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FAIBLE
		

RESSOURCES ÉCONOMISÉES 

Vérifier que le code HTML des pages est bien formé. Dans le cas contraire, le navigateur corrigera dynamiquement un certain nombre d'éléments pour afficher au mieux les pages posant problème. Ces corrections dynamiques consomment inutilement des ressources à chaque chargement des pages concernées.

Exemple

Utiliser le validateur du W3C *(World Wide Web Consortium)* pour vérifier que les pages sont bien valides et que le code HTML est correctement formé :

http://validator.w3.org

Bonne pratique n° 16

Externaliser les CSS et JavaScript

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Veiller à ce que les codes CSS et JavaScript ne soient pas embarqués dans le code HTML de la page, à l'exception d'éventuelles variables de configuration pour les objets JavaScript.

En effet, si vous incluez du code CSS ou JavaScript dans le corps du fichier HTML, alors que ce dernier est utilisé par plusieurs pages (voire tout le site), ce code doit être transféré pour chaque page demandée par l' internaute, ce qui augmente le volume de données transmises. En revanche, si les codes CSS et JavaScript sont inclus dans leurs propres fichiers, le navigateur peut les stocker dans son système de cache local afin de ne pas les redemander.

Exemple

Dans le code HTML, ne pas écrire :

```
<style type="text/css" media="screen">
  p { color: #333, margin: 2px 0 }
  /* Toutes les déclarations CSS du site */
</style>
```

mais plutôt :

```
<link href="css/styles.css" rel="stylesheet">
```


Bonne pratique n° 17

Favoriser les polices standards

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Préférer les polices standards, car elles sont déjà présentes sur l'ordinateur de l'internaute, qui n'a donc pas besoin de les télécharger. On économise ainsi de la bande passante, tout en accélérant l'affichage du site.

Exemple

Dans la mesure du possible, privilégier des polices de caractères comme :

- Courier New ;
- Georgia ;
- Arial ;
- Comic ;
- Impact ;
- Tahoma ;
- Trebuchet MS ;
- Times New Roman ;
- Verdana ;
- Segoe UI.

Pour aller plus loin :

http://en.wikipedia.org/wiki/List_of_typefaces_included_with_Mac_OS_X

www.awayback.com/revised-font-stack

www.codestyle.org/css/font-family/sampler-CombinedResultsFull.shtml

Bonne pratique n° 18

Préférer les glyphes aux images

CONSEILLÉ	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Les glyphes sont inclus dans les polices de caractères du système d'exploitation. Les préférer par conséquent aux images, notamment pour réaliser des effets sur les listes HTML, car ils sont plus économes en bande passante.

Exemple

Si vous remplacez une image par un glyphe présent dans une police système, la bande passante économisée est égale au poids de l'image non utilisée.

Pour aller plus loin :

http://coding.smashingmagazine.com/2011/03/19/styling-elements-with-glyphs-sprites-and-pseudo-elements

Bonne pratique n° 19

Supprimer les balises images dont l'attribut SRC est vide

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Si une balise image est présente et que son attribut SRC est vide, le navigateur va appeler la page d'index du niveau d'arborescence où il se situe, générant des requêtes HTTP supplémentaires et inutiles.

Exemple

La balise image suivante demandera au serveur le fichier `index` du répertoire `foo` :

```
<img src='' alt=''>
```

sur une page située à l'URL :

http://domain.tld/foo/bar.html

Bonne pratique n° 20

Redimensionner les images en dehors du navigateur

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
RESSOURCES ÉCONOMISÉES		

Ne pas redimensionner les images en utilisant les attributs `HEIGHT` et `WIDTH` du code HTML. Cette approche impose en effet de transférer ces images dans leur taille originale, gaspillant ainsi de la bande passante et des cycles CPU.

Exemple

Une image de 350 × 300 pixels encodée en PNG 24 pèse 41 Ko. Redimensionnée dans le code HTML, la même image affichée en vignette à 70 × 60 pixels pèse toujours 41 Ko, alors qu'elle ne devrait pas dépasser 3 Ko ! Soit 38 Ko téléchargés à chaque fois pour rien...

La meilleure solution consiste à utiliser des images redimensionnées en dehors du code HTML à l'aide d'un logiciel de type Photoshop.

Dans la mesure où le contenu proposé par les utilisateurs du site web n'a pas de valeur ajoutée particulière, il est préférable de leur interdire la possibilité d'insérer des images à partir d'un éditeur WYSIWYG comme CKEditor.

Bonne pratique n° 21

Éviter d'utiliser des images bitmap pour l'interface

PRIORITAIRE	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT
RESSOURCES ÉCONOMISÉES		

Choisir le bon format d'image est crucial pour éviter de transporter des octets inutilement et économiser ainsi de la bande passante. Par ailleurs, avec la multiplication des terminaux, des tailles d'écran et l'augmentation de leur résolution, une approche vectorielle doit être privilégiée par rapport à des images matricielles (bitmap).

Grâce à cette bonne pratique, l'interface est indépendante de la résolution de l'écran. On limite donc aussi la dette technique.

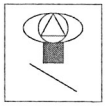
La première règle consiste à remplacer les images bitmap (GIF, PNG, JPEG, WebP, etc.) par des styles (CSS), des pictos, des glyphes ou des icônes fournis par une webfont ou une police standard. L'internaute n'a ainsi aucune ressource supplémentaire à télécharger.

S'il n'est pas possible d'utiliser des CSS ou une police standard (déjà installée sur le terminal de l'internaute), vous pouvez aussi :

- employer une webfont ;
- recourir à une image vectorielle au format standard SVG.

Exemple

Cette image de 198 × 198 pixels pèse :



- 118 Ko dans un format bitmap non compressé ;
- 6,5 Ko en JPEG (compression à 90 %) ;
- 3,8 Ko en PNG ;
- 0,7 Ko en SVG minifié.

Le format vectoriel est, dans ce cas précis, 5 à 10 fois moins lourd qu'un format bitmap tout en pouvant être retaillé à l'infini.

Bonne pratique n° 22

Optimiser les images vectorielles

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Les navigateurs modernes sont tous compatibles avec le format d'image vectorielle SVG *(Scalable Vector Graphics)*, basé sur un ensemble de vecteurs décrits en XML. Les images SVG ont deux avantages indéniables : d'une part, elles peuvent être réduites et agrandies à l'infini sans dégradation de qualité ; d'autre part, elles sont, la plupart du temps, moins lourdes que des images bitmap.

Cependant, la plupart des images SVG contiennent de nombreuses métadonnées qui ont été nécessaires à leur création. C'est par exemple le cas des informations de couche (layer), des commentaires, etc., qui sont indispensables pour éditer l'image, mais inutiles pour l'afficher. D'où l'idée de les supprimer pour réduire le poids des fichiers.

De nombreux outils de minification et d'optimisation, tels que Compressor.io, SVG Cleaner, ou SVGO sont disponibles.

Le taux de compression via gzip varie selon la complexité de l'image. Mais il est toujours élevé, car il s'agit de compresser du texte : en général, on atteint des ratios de l'ordre de 75 % à 80 %.

Exemple

Gain potentiel : jusqu'à 75 % de Ko en moins.

Nous avons testé SVGO sur un fichier SVG de 1 Ko. Il a réduit sa taille de 36 %, le faisant passer de 1101 à 700 octets. En compressant le fichier via gzip avant son transfert, le poids passe à 498 octets, soit moins de la moitié (45 %) de la taille initiale, sans aucune perte de qualité à l'affichage.

Bonne pratique n° 23

Générer des spritesheets CSS

CONSEILLÉ	MISE EN ŒUVRE DIFFICILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Regrouper les images de petite taille (celles de l'interface du site, par exemple) dans une seule image de plus grande taille appelée spritesheet.

Ce procédé réduit significativement le nombre de requêtes HTTP. De nombreux services en ligne gratuits (CSS Sprite Generator du Project Fondue, CSSsprites.com, SpriteMe.org...) permettent de générer ces spritesheets.

Exemple

Voici quelques adresses de générateurs de sprites :

http://csssprites.com

http://csspritegenerator.net

http://spritegen.website-performance.org

http://spriteme.org

http://csssprites.org

http://css-sprit.es

http://wearekiss.com/spritepad

http://draeton.github.com/stitches

Bonne pratique n° 24

Découper les CSS

PRIORITAIRE	MISE EN ŒUVRE STANDARD	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Employer un ensemble de CSS plutôt qu'une seule, et appeler uniquement les CSS utiles en fonction du contexte. Cettte méthode permet de limiter le poids de la page lors du premier téléchargement, donc d'économiser de la bande passante et de réduire la charge CPU.

Exemple

Découper les CSS en fonction de la logique fonctionnelle :

- layout ;
- content ;
- module x ;
- module y ;
- etc.

Dans le cas d'un site fonctionnellement riche, cela permettra d'exclure toutes les CSS des modules non utilisés. Le nombre de CSS doit rester raisonnable, plus pour des questions de maintenabilité que de performance, dans la mesure où les CSS générales (« layout » et « content » dans notre exemple) seront concaténées en un seul fichier. Les CSS complémentaires (ici, « module x » et « module y ») seront téléchargées en fonction du contexte (page, fonctionnalités...).

Bonne pratique n° 25

Limiter le nombre CSS et les compresser

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Limiter le nombre de CSS pour ne pas multiplier les requêtes HTTP. Si plusieurs feuilles de style sont utilisées sur toutes les pages du site, concaténez-les dans un seul fichier.

Certains CMS et frameworks fournissent des solutions automatiques pour effectuer ce type d'optimisation. Il est également possible de paramétrer le serveur HTTP pour qu'il réduise la taille des feuilles de style en les compressant (voir la bonne pratique n° 83).

Exemple

Avec le serveur web Apache, il suffit d'ajouter dans le fichier de configuration `.htaccess` la ligne suivante :

```
# compress css :
AddOutputFilterByType DEFLATE text/css
```

Cette instruction active le mode Deflate qui compresse toutes les feuilles de style entre le serveur et le client HTTP.

En savoir plus sur Deflate :

http://httpd.apache.org/docs/2.4/mod/mod_deflate.html

Bonne pratique n° 26

Préférer les CSS aux images

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT

RESSOURCES ÉCONOMISÉES

Utiliser les propriétés CSS3 à la place d'images. En effet, le poids d'une feuille de style est bien plus faible, surtout si elle est compressée.

En outre, l'appel d'une feuille de style ne génère qu'une seule requête HTTP, contre un grand nombre si l'on emploie beaucoup d'images (une requête HTTP pour chaque image).

Exemple

Les coins arrondis des cases doivent être gérés en CSS3 plutôt qu'avec des images.

Préférer l'écriture :

```
#cadre {
  border-radius: 10px;
}
<dvi id="cadre">
  <p>
  Lorem ipsum dolor sit amet, consectetur
adipiscing elit.
  </p>
</div>
```


Bonne pratique n° 27

Écrire des sélecteurs CSS efficaces

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FAIBLE

RESSOURCES ÉCONOMISÉES

Privilégier les sélecteurs basés sur des ID ou des classes. Ils seront ainsi filtrés plus rapidement, économisant des cycles CPU à la machine interprétant les règles.

Exemple

Ne pas écrire :

```
treeitem[mailfolder="true"] > treerow > treecell {…}
```

mais plutôt :

```
.treecell-mailfolder {…}
```

Ne pas écrire :

```
treehead > treerow > treecell {…}
```

mais plutôt :

```
.treecell-header {…}
```

Pour aller plus loin :

https://developer.mozilla.org/en/Writing_Efficient_CSS

Bonne pratique n° 28

Grouper les déclarations CSS similaires

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
RESSOURCES ÉCONOMISÉES		

Lorsque plusieurs éléments du DOM *(Document Object Model)* ont des propriétés CSS communes, les déclarer ensemble dans la même feuille de style. Cette méthode permet de réduire le poids de la CSS.

Exemple

Ne pas écrire :

```
h1 {
  background-color: gray;
  color: navy;
}

h2 {
  background-color: gray;
  color: navy;
}

h3 {
  background-color: gray;
  color: navy;
}
```

mais plutôt :

```
h1, h2, h3 {
  background-color: gray;
  color: navy;
}
```


Bonne pratique n° 29

Utiliser les notations CSS abrégées

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
RESSOURCES ÉCONOMISÉES		

Utiliser les notations CSS abrégées pour réduire le poids de la feuille de style.

Exemple

Ne pas écrire :

```
margin-top:1em;
margin-right:0;
margin-bottom:2em;
margin-left:0.5em;
```

mais plutôt :

```
margin:1em 0 2em 0.5em;
```

Pour aller plus loin :

www.w3.org/TR/CSS2

www.456bereastreet.com/archive/200502/efficient_css_with_shorthand_properties

Bonne pratique n° 30

Toujours fournir une CSS print

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Outre le service apporté à l'internaute, cette feuille de style réduit le nombre de pages imprimées, et donc indirectement l'empreinte écologique du site web. La plus dépouillée possible, elle doit proposer une police de caractères économe en encre (Century Gothic, par exemple). Pensez aussi à masquer le header, le footer, le menu, le sidebar, supprimer toutes les images sauf celles du contenu, etc.

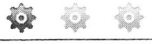
Exemple

Cette CSS print « nettoie » la page affichée à l'écran afin de proposer une impression épurée :

```
body {
  background-color :#fff;
  font-family :Serif;
  font-size :15pt;
}
#page {
  margin :0;
  border :none;
}
#banner, #menuright, #footer {
  display :none;
}
h1#top {
  margin :0;
  padding :0;
  text-indent :0;
  line-height :25pt;
  font-size :25pt;
}
(...)
```


Bonne pratique n° 31

Utiliser les commentaires conditionnels

PRIORITAIRE	MISE EN ŒUVRE FACILE	IMPACT ÉCOLOGIQUE FORT
		

RESSOURCES ÉCONOMISÉES 

Si des ressources spécifiques doivent être transmises à certains navigateurs, utiliser les commentaires conditionnels afin que les autres navigateurs ne téléchargent pas inutilement ces ressources.

Exemple

Dans le cas de la prise en charge du navigateur Internet Explorer 7, on peut par exemple ajouter le commentaire conditionnel suivant :

```
<!--[if lte IE 7]>
<link type="text/css" rel="stylesheet" media="all"
href="/fix-ie.css" />
<![endif]-->
```