GMIN250 - Travail d'Etude et de Recherche
Rapport de projet

Extraction et Annotation de Structures Visuelles

Remis en tant que partie du contrôle de connaissances du second semestre de

Master en Informatique
Mention
Architecture et Ingénierie du Logiciel et du Web

Remis par

BANGUET Warren
LEULLIETTE Baptiste
TURPAULT Arlémi
ZEGAOUI Taqiyéddine

Sous la responsabilité de
M M. MEYNARD

Département Informatique
Faculté des Sciences de Montpellier
Université Montpellier

Mai 2015

Table des matières

Table des figures

Remerciements

Nous remercions M. Michel MEYNARD de nous avoir encadré durant ce TER.
Nous remercions aussi M. Franck PETITDEMANGE pour sa base de travail qui nous a permis de réaliser notre projet.
Nous remercions Mme. Pascale HUMBERT, membre de la fondation VISIO à Paris pour son apport d'informations sur les déficients visuels.
Nous remercions M. Jacques FERBER, rapporteur de notre projet.
Nous remercions M. Eric BOURREAU, pour nous avoir fait découvrir les différentes techniques de gestion de projet.

Chapitre 1

Introduction

1.1 Présentation du projet

Dans le cadre du second semestre de la première année de Master informatique, il est demandé aux étudiants de réaliser un projet dans le cadre d'un TER, Travail d'Étude et de Recherche proposé par un professeur-tuteur. Ce projet a pour but de servir de support à l'apprentissage du travail en équipe, qui représente la majorité des projets professionnels dans le domaine de l'informatique, autant en entreprise qu'en laboratoire de recherche.

Ainsi, pour assurer la réussite du projet, le groupe chargé de sa réalisation, constitué de quatre étudiants, devra se montrer capable de mettre en place une organisation du groupe favorable à la réussite : en effet, le groupe devra décider d'une organisation capable de prendre les décisions relatives à la division du groupe en différentes équipes, à la répartition du travail entre ces différentes équipes, aux moyens d'établir une communication fiable et performante au sein du groupe et enfin d'assurer un contrôle des objectifs fixés. C'est en remplissant ces objectifs que les conditions de réussite du projet seront réunies, et cette méthode contiendra les ingrédients du travail en équipe.

Le projet sera encadré par le professeur responsable de la proposition du sujet ; sa mission comprendra plusieurs aspects tels que l'explicitation des zones obscures du sujet, le contrôle de l'avancement du projet, et, chaque fois que le besoin s'en fera sentir, la mise en œuvre d'une assistance au groupe. Le tuteur pourra non seulement aider le groupe à mieux cerner le sujet, mais aussi à saisir chaque aspect du travail en équipe. Ainsi, le groupe pourra compter sur l'aide de son professeur-tuteur en cas de problème au sein du groupe, que ce soit en cas de désistement ou de conflit.

1.2 Sujet

1.2.1 Contexte du sujet

Depuis les débuts de sa démocratisation dans les années 1990, Internet a massivement transformé le quotidien de millions de personnes à travers le monde, et depuis le début de cette décennie le rôle que joue Internet dans nos vies a augmenté de façon exponentielle. Néanmoins, de par les aspects les plus concrets et triviaux de nos moyens d'accès à Internet, un nombre certain de personnes se voient mises à l'écart de cette révolution numérique en raison de conditions diverses, notamment certains handicaps.

Ainsi en France, où ont été recensées plus d'un million de personnes affligées d'une déficience visuelle, nous pouvons parler d'une véritable fracture numérique. En effet, Internet étant composé dans une écrasante majorité de pages ne contenant que des textes et des images, ces personnes sont celles dont le handicap empêche de la façon la plus critique l'accès au World Wide Web. L'élaboration d'une méthode alternative d'accès aux ressources d'Internet destinée aux personnes aveugles ou malvoyantes est donc une préoccupation majeure à l'heure où Internet, de par sa croissance importante et son importance croissante, se présente comme un des éléments majeurs de la vie au XXIe siècle.

1.2.2 Situation actuelle

Afin de pouvoir accéder à Internet, les personnes déficientes visuellement ont aujourd'hui plusieurs possibilités à leur disposition suivant la sévérité de leur handicap et leurs besoins. Ainsi, les malvoyants profonds et les aveugles complets peuvent utiliser des navigateurs fonctionnant sur le modèle des logiciels appelés revus d'écran transformant une page Web en texte qui sera ensuite lu par un synthétiseur vocal ou transcrit tactilement par l'intermédiaire d'une plage braille. Pour les personnes souffrant de déficience visuelle dite moyenne, les outils utilisés sont la plupart du temps des logiciels s'apparentant à une loupe virtuelle permettant d'agrandir la partie de l'écran où se déplace la souris.

1.2.3 Problématique

Malgré une très grande amélioration des moyens d'accès à Internet pour les aveugles et les malvoyants profonds, les personnes atteintes de déficiences visuelles moyennes ou légères gagneraient à trouver à leur disposition des

outils permettant d'adapter la mise en forme des pages Web par rapport à leur handicap. En effet, en plus de l'effet de grossissement de l'affichage, il serait intéressant de pouvoir agir sur d'autres paramètres en rapport avec la lisibilité tels que les notions de contraste, de police de caractère et d'arrière-plan.

Chapitre 2

Analyse

2.1 État de l'art

2.1.1 Extraction et Annotation

De nombreuses publications traitent du sujet d'explorer le code source d'une page web afin d'en extraire des données, des structures. On distingue plusieures approches visant à analyser la structure d'un ensemble de données semi-structurées en général(page HTML, mais aussi Base de données par exemple), ces principes [2, 3] sont intéressants et auraient pu être utilisés dans notre approche.

Cependant, le contexte de notre travail, à savoir la modification dynamique de l'affichage d'une page web fait perdre toute utilité à ces méthodes qui se contentent d'une analyse statique du code HTML (donc sans prise en compte du style associé) là où nous souhaiterions pouvoir dynamiquement analyser l'arbre DOM juste avant l'affichage de la page, au moment où le style associé est effectivement appliqué aux balises HTML. De plus certaines approches ne sont pas complètement automatiques, ce qui est pour nous inenvisageable.

Etant donné nos contraintes établies ci-dessus, nous avons approfondi nos recherches dans l'espoir de trouver une méthode alternative à celle mise au point par les auteurs de [4](résumé en annexe A.1). En effet cette publication est celle sur laquelle F. PETITDEMANGE, ayant réalisé un travail sur l'extraction de structures visuelles [1], a imaginé suivre afin de réaliser l'annotation des structures visuelles qu'il extrait (il n'a cependant rien implémenté concernant l'annotation). Néanmoins très peu de papiers traitent spécifiquement et en détail de la procédure de détermination de la sémantique associée aux

structures d'une page web, encore moins quand il s'agit de prendre en compte le style associé au code HTML d'une page.

2.1.2 Adaptation

L'adaptation de sites Web, en fonction de filtres, applicable à n'importe quel site Internet, reste encore très peu développée. Des sites d'accès au handicap pour les malvoyants tels que Handicapzero.og permettent de modifier les couleurs, police d'écriture et/ou taille d'écriture sur le site en fonction des différents troubles liés à l'utilisateur.

2.2 Base de travail

Nous avons pris, comme base de travail, le script réalisé par F. PETIT-DEMANGE au cours de son stage recherche de M2. Ce dernier a réalisé la détection de blocs visuels par analyse du code source d'une page Web juste avant son affichage sur le navigateur de l'utilisateur. Son rapport [1] nous a également été partagé. Vous trouverez en annexe A.2.1 le document de travail nous ayant servi de support à l'explication du script lors d'une réunion avec notre tuteur.

2.2.1 Présentation

Le plug-in a été réalisé en JavaScript et fonctionne sur le navigateur Firefox grâce à l'extension Greasemonkey, un gestionnaire de scripts JavaScript. Il est composé, en plus du script principal, de trois fichiers contenant les différentes définitions de variables et fonctions :

Predicats : Contient toutes les fonctions d'exploration selon les différents axes d'un arbre complétant les primitives fournies par l'API DOM de JavaScript.

Paramètres : Contient toutes les variables statiques faisant office de paramètres tels que les seuils d'application de règle ou encore la borne de profondeur de parcours de l'arbre DOM.

API : Contient toutes les définitions de fonctions permettant le filtrage, la discrimination des blocs visuels. Si nous devons insérer des fonctions afin d'affiner l'extraction et/ou stocker des informations sur les blocs, ce sera dans ce fichier.

Le script se lance une fois tous les éléments de la page chargé, à l'appel de la fonction ready() de jQuery et a donc accès au code HTML de la page comportant les propriétés CSS des différentes balises, ainsi que le code généré dynamiquement auparavant. Grâce à l'API DOM et à des fonctions ad hoc, l'arbre DOM final peut donc être parcouru et toutes les propriétés utiles à la détermination de blocs visuels cohérents être récupérées et traitées.

Le but du script est de partitionner l'arbre DOM en un ensemble arborescent de structures correspondant à des blocs cohérents du point de vue des propriétés de style qu'ils partagent en interne, ou ne partagent pas entre deux blocs distincts.

2.2.2 Vocabulaire

Dans son rapport ainsi que son code source, F. PETITDEMANGE [1] utilise un vocabulaire particulier, quelque peu ambigu utilisant la notion d'atomicité pour décrire les nœuds représentant le plus bas niveau auquel son analyse du code source s'arrêtera. Nous avons donc prévu de redéfinir cette notion afin d'améliorer la compréhension du mécanisme de son extraction.

Nœud de contenu Représente tout nœud n'englobant qu'une valeur textuelle brute. N'est pas atomique au sens propre (car l'atome est ici le texte englobé), mais sera le plus bas niveau auquel nous nous intéresserons. Les nœuds de contenu représentent les balises telles que A, IMG...
= Nœud Atomique (dans la version de F. PETITDEMANGE)
= ContentNode (dans le code réécrit en anglais)

Nœud de contenu virtuel Représente un nœud dont le seul enfant est un nœud de contenu. Il n'a donc pas d'autre utilité que de décorer ou appliquer un style à son seul enfant. Ce sont actuellement ces nœuds qui sont le plus bas niveau de bloc reconnu par l'algorithme d'extraction et représentent donc les feuilles de l'arbre résultant de structures visuelles.
= Nœud Atomique Virtuel = NAV (dans la version de Franck PETITDEMANGE)
= VirtualContentNode (dans le code réécrit en anglais)

Bloc visuel Représente une branche entière de l'arbre DOM ayant une cohérence de style et dont on peut assurément dire qu'elle offre une fonctionnalité précise, quoiqu'inconnue au moment de sa détection.
= VisualStructure (dans le code réécrit en anglais)

Dans l'arbre fourni par le script en sortie, seuls apparaissent des Blocs Visuels et des Nœuds de contenu virtuels, ce qui simplifie grandement l'arbre décrivant la structure de la page Web.

2.2.3 Traitement réalisé

Le script réalise un parcours en profondeur de l'arbre DOM à partir de la balise BODY, cette dernière étant la racine de l'arbre de sortie. Pour chaque nœud rencontré lors du parcours, les tests suivants sont effectués :

- Filtrage du nœud (= n'apparait pas dans l'arbre de structures visuelles) si celui-ci n'est pas visible à l'affichage par le navigateur, ou s'il représente une balise inintéressante sur laquelle aucun traitement ne doit être effectué (balise SCRIPT par exemple).

- Application d'un ensemble de règles (expliquées en annexe A.2.1) permettant de déterminer si la branche issue du nœud représente une structure homogène du point de vue de ses paramètres de style et de sa structure. Si oui alors elle est ajoutée dans l'arbre des structures visuelles, sinon on divise le nœud et l'ensemble de règles est retesté sur ses enfants dans l'espoir de découvrir des structures visuelles.

- Division du nœud (= suppression du nœud dans l'arbre des structures visuelles avec adoption de ses enfants par son parent) si ce nœud englobe des structures incohérentes entre elles ou s'il s'agit d'une balise «fantôme» ne servant qu'à appliquer du style, le contenu réel se trouvant chez ses enfants.

- Si le nœud a été reconnu comme Bloc Visuel ou Nœud de Contenu Virtuel, alors il représente une structure homogène et est ajouté à l'arbre résultat.

Après exécution du script, on obtient donc un arbre réduit représentant le partitionnement en blocs visuels de la page analysée, masquant le contenu complet de ces blocs visuels, mais formant un mapping (tel que défini par Franck PETITDEMANGE dans [1]) avec l'arbre DOM. On obtient donc ainsi un arbre de références vers les nœuds du DOM qui englobent les structures visuelles extraites.

2.2.4 Bilan

Après divers tests et quelques ajustements dans le code pour adapter le script à notre projet, ce-dernier nous fournit bien le partitionnement nécessaire à l'identification de la sémantique des différentes parties composant une page Web. Néanmoins le manque de documentation du code et la complexité de certaines règles pourraient poser problème si nous devons ajouter des instructions directement dans le code source.

En conclusion, nous pouvons imaginer conserver cette partie dans le projet afin de pouvoir avancer sur les étapes d'annotation et d'adaptation et, ainsi, couvrir entièrement le sujet.

Chapitre 3

Conception

3.1 Architecture globale

3.1.1 Partitionnement

Considérant les résultats de l'étude détaillée, il a été décidé de partitionner le projet en deux modules complémentaires, possédant chacun une architecture et une fonction distincte.
Le module principal, chargé de parcourir l'arbre DOM de la page Web et d'en extraire les sous-arbres possédant une sémantique directement liée à l'aspect final de la page et à son contenu, le module d'annotation, chargé de parcourir l'arborescence de la page Web et d'annoter les blocs produits afin de disposer des informations importantes concernant la nature de ces blocs parcourus.

Le second module est celui de la gestion des préférences utilisateur permettant de recueillir les préférences visuelles de l'utilisateur en termes de polices de caractère, de couleurs, de contrastes et de taille des polices, puis de modifier la page Web afin d'adapter au mieux son contenu aux besoins de l'utilisateur.

Nous sommes partis de la base de travail fournie par le script de F. PETITDEMANGE afin d'y incorporer nos travaux. Le plug-in étant sous GreaseMonkey il était facile d'y ajouter un autre fichier qui servirait pour l'adaptation des sites.

3.1.2 Point de départ

Il a été choisi de prendre pour socle de base les travaux réalisés par F. PETITDEMANGE afin d'y ajouter de façon progressive les fonctionnalités

prévues par le sujet. Ce choix a été en effet motivé par plusieurs raisons d'ordre technique. Premièrement, de par son exhaustivité et sa pertinence, l'analyse théorique, notamment concernant le système d'annotation, permettait mettre en place un système fonctionnel et performant d'heuristiques d'identification et d'annotation des différents blocs sémantiques constituant une page web.

En outre, ignorer l'implémentation commencée par F. PETITDEMANGE aurait amené une longue analyse des différentes possibilités d'implémentation possibles et de leurs avantages respectifs. Cependant le choix de GreaseMonkey permet une évolutivité simple et pratique de l'application.

3.1.3 Articulation

Le script de F. PETITDEMANGE était le fichier principal du plug-in GreaseMonkey qui permettait le lancement et la reconnaissance des différents blocs visuels. Cependant, nous nous basons sur cette reconnaissance pour peaufiner la recherche et créer l'adaptation visuelle du site. C'est pourquoi le script d'adaptation est devenu le script principal, qui, au cours du fonctionnement, lancera le script de détection.

3.2 Annotation

3.2.1 Objectif

Le but de la démarche nous concernant ici est de pouvoir associer un sous-arbre de l'arbre DOM, identifié comme étant un bloc visuel, à sa fonction dans la page. En effet, la quasi-majorité des sites Web comportent des structures abstraites récurrentes que nous souhaiterions pouvoir identifier à partir du code source dont elles sont issues. Ces structures peuvent être (non exhaustif) :

- Des menus
- Des bannières
- Des références externes
- Des articles, blocs de texte...

Identifier de façon générique les branches de l'arbre DOM produisant à l'affichage les structures ci-dessus permettra donc d'effectuer sur elles des modifications ciblées, dictées par l'utilisateur, en vue de proposer à ce dernier un

accès facilité aux possibilités offertes dans une page, et donc une navigation plus adaptée à son handicap.

3.2.2 Problématique

Les utilisateurs du Web peuvent facilement reconnaitre les structures évoquées ci-dessus à l'écran, néanmoins l'exercice devient beaucoup plus difficile si on se limite à analyser la branche de l'arbre DOM dont est issue une page Web. D'autant plus qu'il existe autant de façon de structurer le code d'une page qu'il existe de développeurs Web, la généricité sera le plus gros enjeu de notre réalisation.

3.2.3 Démarche

La sémantique devant être associée à une structure découle de plusieurs paramètres, parmi eux on peut distinguer :

Ses caractéristiques intrinsèques comme le type et l'organisation des structures filles la composant. C'est pourquoi l'annotation doit se faire en partant des structures feuilles vers la racine afin de pouvoir déterminer la sémantique des blocs en fonction de celle de leurs enfants.

L'intention du développeur dictée par ses choix dans la mise en page et l'application de styles à la page Web qu'il conçoit, ces informations sont données par la partie Extraction de structures qui nous assure des structures cohérentes, du point de vue du style, en sortie.

Nous nous sommes inspirés du modèle établi par les auteurs de [4] (résumé en annexe A.1) afin de mettre au point notre démarche d'annotation. Notre décision a été grandement motivée par le fait que ces derniers indiquent avoir réalisé une reconnaissance de structures visuelles basée sur l'évaluation de la distance visuelle entre les différents composants de la page, le script écrit par F. PETITDEMANGE nous fournira l'extraction de ces structures selon ce même procédé.

Principe général

En premier lieu, nous définissons deux grandes catégories parmi les structures visuelles obtenues en sortie de l'extraction (voir 2.2) :

Les Objets Basiques regroupant les éléments feuilles de l'arbre des structures visuelles, c'est à dire les Nœuds de Contenu Virtuels, tels que reconnus par le script d'extraction. Ces objets possèdent des propriétés de base visant à déterminer leur fonction.

Les Objets Composites qui regroupent, par définition, l'ensemble des autres éléments de l'arbre des structures visuelles. Ils sont composés de plusieurs objets (basiques ou composites) qui sont liés entre eux par une homogénéité de style et/ou de mise en page puisqu'ils ont été reconnus comme structures visuelles lors de l'extraction.

En fonction de leurs caractéristiques intrinsèques, une valeur est attribuée aux propriétés caractérisant ces objets (voir ci-après). Concernant les objets basiques, c'est leur fonction prise en dehors du contexte qui nous importe, tandis que pour les objets composites le contexte est pris en compte afin de pouvoir, prenant en compte les fonctions offertes par les objets les composant, les associer aux grandes structures abstraites du Web.

Objets Basiques

Les propriétés que nous avons définies pour ces objets sont uniquement déterminées par une exploration de la branche de l'arbre DOM dont ils sont issus, leur position n'est pas prise en compte. Nous n'avons reproduit qu'une partie des propriétés établies dans l'article [4], ceci car l'extraction réalisée par Franck PETITDEMANGE est légèrement différente de celle des auteurs de [4] puisqu'elle filtre des nœuds que ces derniers traitent et utilisent dans leur démarche d'annotation. Nous posons donc les paramètres suivants pour tout objet basique :

Présentation : Définit la forme selon laquelle est présenté l'objet lors de son affichage dans le navigateur, peut prendre la valeur «media», «text», «img», stocke aussi la longueur du contenu des nœuds texte que comporte éventuellement l'objet.

hyperlink : Définit la capacité de l'objet a fournir un lien vers un autre contenu, interne ou externe au site Web. Peut prendre les valeurs «none», «unknown», «intern», «extern».

interaction : Définit la possibilité d'interaction avec l'utilisateur que fournit l'objet, vaut «display» lorsque l'objet fournit des informations a l'utilisateur, «button» lorsque l'utilisateur peut effectuer une action sur l'objet et «input» lorsque cet objet permet à l'utilisateur de saisir des informations.

semanteme : Définit la sémantique pouvant être éventuellement directement déduite de l'analyse d'une balise HTML dans le cas où celle-ci est suffisamment spécifique, par exemple les balises Hn (n entre 1 et 9) correspondent à un label introduisant le contenu suivant.

En fonction des valeurs de ces propriétés, nous pourrons déduire la fonction offerte par l'objet basique (voir 4.1.2)

Objets Composites

L'analyse des objets composites se fera par analyse de leur composition ainsi que de leur position sur le rendu navigateur final. La méthode d'identification de leur sémantique sera très spécifique à chacune des structures abstraites que l'on souhaitera pouvoir identifier, c'est pourquoi peu de propriétés communes sont utiles à stocker. C'est pourquoi nous ne conserverons de [4] que la propriété «clustering», basée sur les fonctions offertes par les objets composant l'objet composite analysé, et qui pourra prendre les valeurs suivantes :

complément : Les objets enfants directs de l'objet composite offrent des fonctionnalités différentes qui, en collaboration, permettent de lui fournir une fonctionnalité. Exemple : un objet composite de recherche est composé d'un objet permettant la saisie d'un texte et d'un objet bouton effectuant une action lors du clic. Le tout offrant la possibilité à l'utilisateur de faire une recherche.

parallel : Les objets enfants directs de l'objet composite offrent tous la même fonctionnalité et sont d'égale importance quant à la réalisation de la fonctionnalité offerte par l'objet composite. Exemple : Un menu est composé d'objets basiques de navigation interne permettant d'offrir à l'utilisateur l'accès à l'ensemble des pages du site Web.

Dans le but d'affiner la sémantique associée à ces objets composites, leur position sur l'affichage final sera prise en compte afin de différencier les grandes structures abstraites du Web. Par exemple un objet composite de navigation interne parallèle pourra être identifié comme menu principal s'il est situé dans le quart gauche du rendu, comme liste de références s'il est situé dans le pied de page, ou encore comme fil d'Ariane s'il est situé sous la bannière de la page et organisé horizontalement.

L'exemple précédent montre bien que notre démarche pourra reconnaitre de façon efficace les grandes structures abstraites que nous avons fixées comme objectif afin de pouvoir, par la suite, les adapter selon des paramètres spécifiques à chacune d'entre elles.

3.3 Adaptation

3.3.1 Contexte

La partie d'adaptation se place dans une optique d'aide au déficient afin de rendre la navigation plus simple et efficace. Ce dernier pourra choisir certains paramètres, couleurs ou tailles lui permettant de se rapprocher d'un confort de navigation qui lui est propre.

3.3.2 Objectifs

Il nous faut donc extraire les différents paramètres potentiels qui permettront à l'utilisateur de modifier l'apparence du site Web visité. Une IHM appropriée doit être réfléchie, évitant tous les éléments minuscules qui pourraient apparaitre et accroitre la difficulté d'utilisation et de compréhension. Pour finir, une documentation est prévue afin d'aider au maximum l'utilisateur dans sa démarche.

3.3.3 Contraintes techniques

De par la base de travail fournie et les objectifs fixés, il est obligatoire de faire l'intégralité du script d'Adaptation en JavaScript. La libraire jQuery nous permettra de réaliser facilement les modifications sur le site qui sera adapté.

3.3.4 Paramètres et actions disponibles

Nous avons analysé quatre catégories de paramètres qui influent sur la navigation :

- Taille d'éléments à caractères interactifs (boutons, liens...)
- Couleurs à filtrer/fixer, c'est à dire, éviter une plage de nuance d'une couleur trop agressive et/ou choisir une couleur pour le background du site.
- Police d'écriture, on parle de taille, couleur, type de police.

- Contraste entre les différents blocs, on retrouvera donc une relation avec la partie d'annotation d'éléments qui pourrait servir d'une liste de choix que l'utilisateur veut ou ne veut pas voir.

Une barre de recherche pourrait éventuellement apparaitre afin de permettre à l'utilisateur de rechercher une phrase ou un terme dans le site, et le mettre en surbrillance afin qu'il le remarque plus facilement.

Après avoir eu un contact avec Mme Pascale HUMBERT, responsable de la fondation Visio à Paris pour les malvoyants, il s'avère qu'il y a seulement cinq couleurs utiles pour la modification des textes. D'autres paramètres apparaissent :

- Le fil d'Ariane à faire apparaitre, en haut ou en bas
- L'affichage des images
- L'affichage des animations, applications et scripts automatiquement
- Linéarisation des tableaux, il s'agit de transformer les tableaux en listes à puces
- L'affichage des cadres frames/iFrames

Tous ces changements doivent être effectués en direct. L'utilisateur verra donc l'impact de ses changements sur le site et pourra réagir en conséquence.

3.3.5 Stockage des informations

Le fait d'avoir un formulaire interactif implique que l'utilisateur peut effectuer ses modifications une seule fois pour les voir s'appliquer à tout les sites. C'est pourquoi le stockage des différents paramètres saisis est un caractère important du plug-in.

Avec l'avancée des navigateurs Internet et l'apparition d'HTML5, l'option du «LocalStorage» semble être une excellente option. Un autre choix s'offrait à nous, mais restait cependant moins efficace, il s'agissait des cookies.

3.3.6 Interface

Une première version de l'interface a été réfléchie, qui permet à l'utilisateur de garder un accès clair, rapide et constant aux paramètres. En voici le schéma.

FIGURE 3.1 – Maquette avec barre donnant accès à l'interface

Nous avons imaginé une barre (voir figure 3.1) qui reste collée en haut de la fenêtre de navigation malgré les scrolls de l'utilisateur afin qu'il garde cet accès constant au plug-in. Cette barre peut évoluer vers d'autres perspectives d'interaction. Pour le moment, lors du clic sur cette barre, une transition apparait, grisant le fond et montre une page proposant les différents paramètres comme on peut le voir sur la figure 3.2.

Durant le second cycle de développement, une meilleure approche nous est apparue. Une approche de «Bulle» cliquable et déplaçable à travers toute la page est plus claire, visible, moins invasive que l'ajout d'une barre en haut du site et peut être placée n'importe où, sans gêner la lecture (figure 3.3).

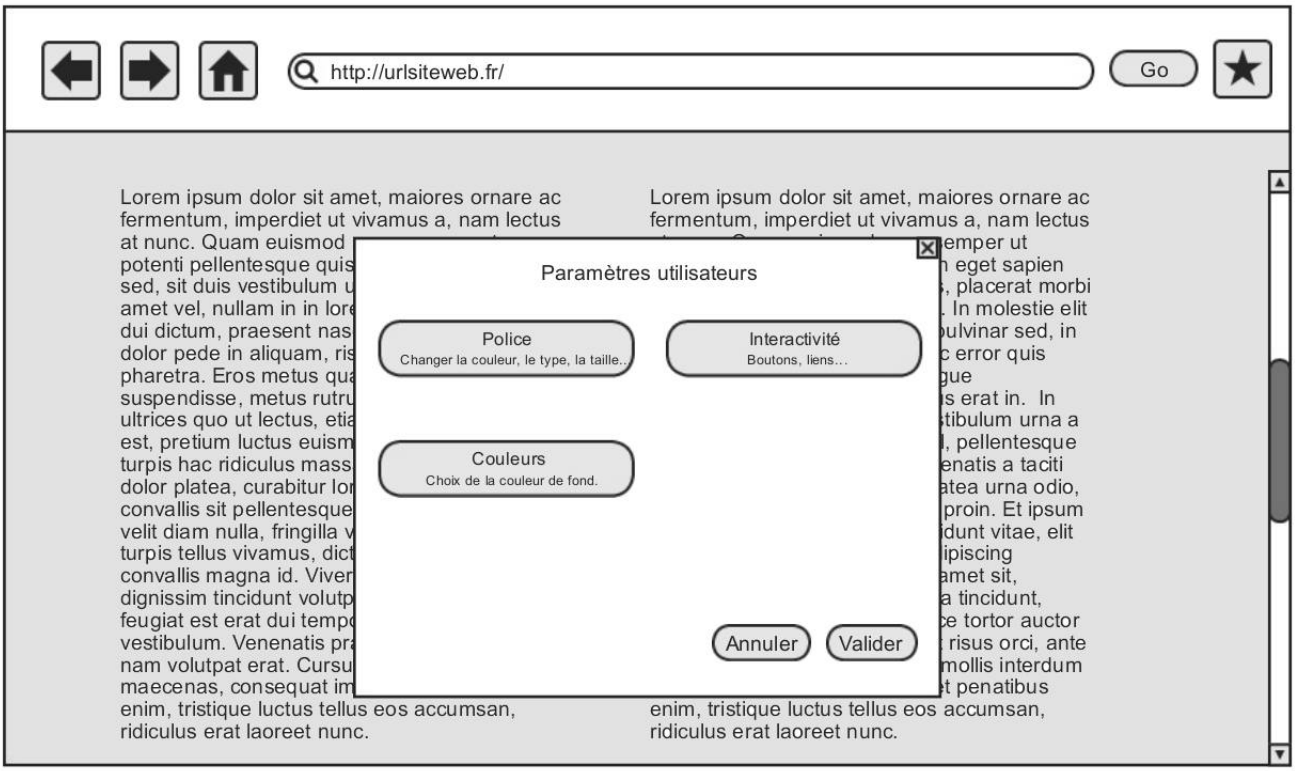

FIGURE 3.2 – Maquette de l'overlay de l'interface

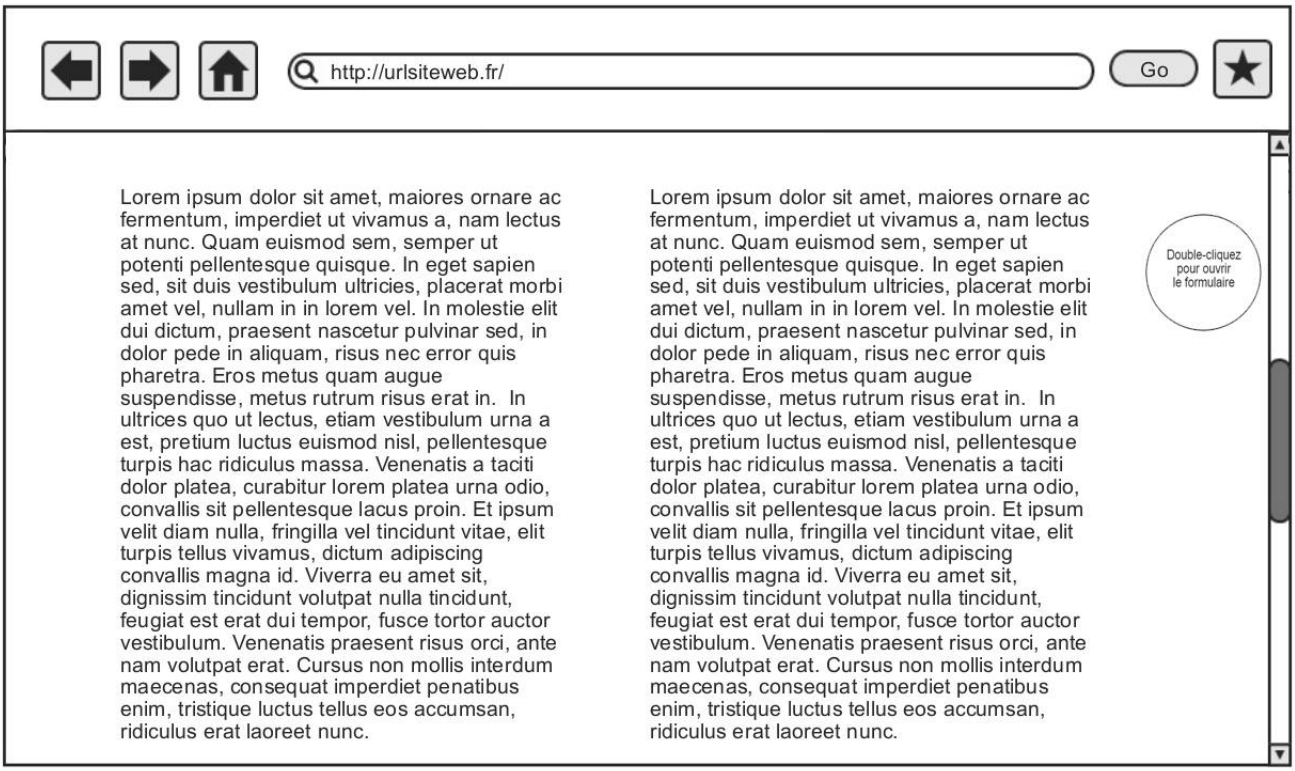

FIGURE 3.3 – Maquette de l'interface «Bulle»

Chapitre 4

Réalisations

4.1 Extraction et Annotation

4.1.1 Fusion des modules

Lors de la conception, nous avons considéré l'extraction et l'annotation de structures visuelles comme deux modules distincts, le second traitant l'arbre des structures visuelles généré par le premier. Cependant cette représentation est, du point de vue de l'implémentation, fastidieuse et gourmande en temps d'exécution car elle suppose de réexplorer la branche de l'arbre DOM dont est issue de la structure visuelle que l'on souhaite annoter.

La solution que nous avons adoptée est d'intégrer au travail réalisé par Franck PETITDEMANGE la détermination des propriétés des objets. En effet, il nous est possible, au fur et à mesure que les structures visuelles sont reconnues, de donner une valeur aux propriétés caractérisant les objets, et même de déterminer la sémantique des objets basiques. Cela représente un gain en temps d'exécution non négligeable.

De plus, nous avons réalisé une refonte du code de Franck PETITDEMANGE afin de le rendre plus accessible, en effet le manque de commentaires et un mélange de français et d'anglais dans les choix de nommage ont rendu la découverte et la compréhension du code plus difficile. Mis à part les ajouts relatifs à l'annotation, les modifications suivantes ont été effectuées dans le script original :

- Désambiguation du vocabulaire utilisé
- Uniformisation de la langue utilisée : modification des noms vers l'an-

glais

- Ajout de commentaires afin d'améliorer la réutilisation
- Explication des règles de détermination de structures visuelles (voir annexe A.2.1)

4.1.2 Annotations réalisées

Objets Basiques

Objets de navigation basiques : Objets basiques dont la propriété hyperlink est différente de «none», c'est à dire offrant une possibilité d'accéder à une nouvelle page.

Objets d'information : Objets dont la propriété d'interaction vaut «display», ayant une propriété «textlength» supérieure à un seuil fixé et dont la propriété «presentation_media» vaut «text» ou «img».

Objets d'interaction («special_control») : Objets dont la propriété d'interaction vaut «button» ou «input».

Objets Composites

Nous nous sommes aperçus, à l'usage, que les objets composites pouvant contenir d'autres objets composites, il était donc nécessaire de leur attribuer une sémantique similaire à celle des objets basiques afin de faciliter le processus. L'objectif final restant toujours d'identifier les grandes structures abstraites évoquées dans la conception, qui représentent l'ossature de la page Web. Ainsi il nous est possible de déduire de la même façon la sémantique d'un objet composite qu'il soit composé uniquement d'objets basiques ou d'un mélange d'objets basiques et composites.

Ceci est complété par l'analyse des propriétés propres à l'objet composite et sa position sur le rendu final, permettant d'affiner la sémantique que l'on donne à l'objet pour transformer cette sémantique générale en quelque chose qui représente réellement les structures que pourrait reconnaitre l'utilisateur.

4.1.3 Exemples

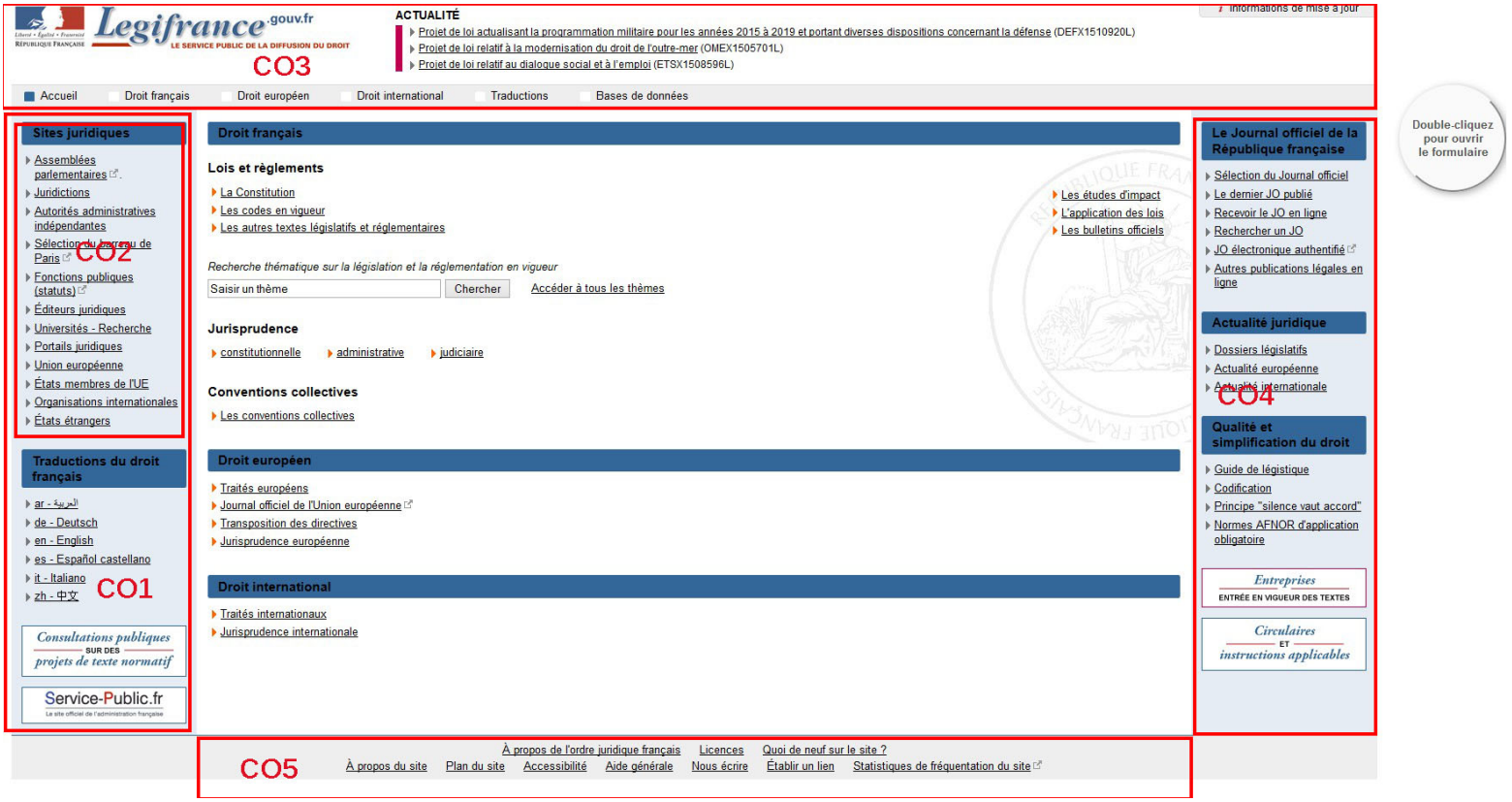

FIGURE 4.1 – Exemple d'annotations réalisées sur le site Legifrance

CO1 est un Objet Composite de navigation parallèle, en effet il est composé de deux Objets composites de navigation parallèles (dont CO2) ainsi que de deux Objets Basiques de navigation. On lui attribue la sémantique "Menu principal" en raison de sa position à gauche de la page, c'est ce qui le différencie de CO4 qui est lui aussi un Objet Composite de navigation parallèle, même un menu, mais dont le positionnement suggère qu'il est secondaire.

CO3 est un Objet Composite complémentaire composé d'un Objet Basique de navigation (logo), d'un Objet Composite de navigation parallèle horizontal (fil d'arianne) ainsi que d'un Objet composite de navigation parallèle (actualités). Sa position permet ici de reconnaitre la bannière du site.

CO5 est un Objet Composite de navigation parallèle, mais cette fois-ci sa position dans le pied de page suggère les références du site.

4.2 Adaptation

4.2.1 Organisation

Le développement du plug-in a été découpé en «cycles». Cette répartition nous a permis de mettre en place le cœur du script dans un premier temps et continuer à implémenter de nouvelles choses au fur et à mesure des cycles. Nous nous sommes concentrés sur l'implémentation des modifications de paramètres basiques, en attendant que la découverte des blocs sémantiques «utiles» soit finie.

4.2.2 1er Cycle — Le cœur

Mise en place de la maquette

Le fait d'avoir un script totalement indépendant du site sur lequel il s'applique nous force à mettre en place la structure dynamiquement, au chargement de la page. Tout ce fait via jQuery qui nous permet «d'append» la «Bulle», c'est-à-dire le code HTML de la «Bulle» en elle-même et l'overlay contenant les paramètres modifiables, ainsi que les propriétés CSS associées. Une fois la Bulle terminée, tous les handlers des différents boutons sont ajoutés dynamiquement et sont liés aux actions.
Lors de modifications basiques sur le site tel que la taille de police, nous nous sommes rendu compte que notre script était aussi modifié par effet de bord. De ce fait nous avons fait en sorte que tout les div qui sont générés dynamiquement par le script possèdent un préfixe «TER» qui permettra plus tard de filtrer notre script.

Le stockage des paramètres

Une fois la mise en place de la maquette terminée, il faut récupérer les éventuels paramètres disponibles afin de modifier le site et faire un pré chargement afin que l'utilisateur ne resaisisse pas les paramètres. C'est ici qu'intervient le LocalStorage.

```
Récuperation des parametres
IF parametres etablis THEN
    POUR chaque parametre etabli DO
        Application de la méta méthode pour chaque méthode
    END
END
```

De cette façon, quand le plug-in est lancé, avant toute interaction possible par l'utilisateur, il y a un chargement de l'intégralité des paramètres.

Quant à la sauvegarde des paramètres, il y a une mise à jour du Local-Storage avec les nouveaux paramètres, au moment où l'utilisateur clique sur le bouton Confirmer. Sous la forme «Cle -¿ Valeur», ces sauvegardes sont de simples entiers (pour la taille de la police), des chaines de caractères (pour la couleur du fond et/ou couleur de police) et des booléens pour l'affichage alternatif des images.

Parcours DOM avec fonction de premier ordre

Afin d'appliquer la modification des paramètres, nous avions deux choix :

- Réecrire un parcours DOM pour chaque modification
- Developpement d'une fonction de premier ordre qui prend la fonction à appliquer au nœud à modifier

Il était évident de mettre en place cette fonction de premier ordre, rien que pour la lisibilité et la compréhension du code.
Dans un premier temps, un simple parcours récursif de l'intégralité du DOM, tout en filtrant les noms (Filtre sur le nom «TER» pour éviter de modifier notre propre mise en page du script)

```
function DomTree(node, action)
{
    var obj = node;
    WorkToDo(obj, action);
    //on filtre tout le plugin avec les id "ter_"
    if (obj.hasChildNodes() && obj.id.indexOf("ter_") == -1)
    {
        var child = obj.firstChild;
        while (child)
        {
            if (child.nodeType === 1)
            {
                DomTree(child, action);
            }
            child = child.nextSibling;
        }
    }
}
```

C'est ici que s'applique la fonction de premier ordre «WorkToDo» :

```
function WorkToDo(Node, function) {
    if (nom(Node) != ter_) {
        switch(function) {
            function(Node); //Application de la fonction au noeud
        }
    }
}
```

De cette façon avec un seul parcours récursif et une seule méthode, nous pouvons mettre en place toutes les modifications possibles de base (Couleurs de fond/police et taille de police d'écriture)

4.2.3 2eme Cycle — les rajouts

Pendant le second cycle, nous avons implémenté les différents paramètres suivants :

- Augmentation/diminution de la taille de la police d'écriture
- Modification de la couleur de fond
- Modification de la couleur de la polie d'écriture
- Début de mise en place de la liste des structures «utiles»
- Option de remplacer des images par le texte alternatif

Modification des paramètres

Mise en place des différents paramètres dans la méthode WorkToDo :

- Gestion du paramètre «++» et «— —» qui permettent de, récursivement, modifier de +1px / —1px la taille de police d'écriture.

- Gestion du paramètre «color», c'est-à-dire un choix parmi l'énumération suivante : noir, blanc, jaune, vert, bleu

- Affichage du texte alternatif des images : Pour chaque balise img trouvée dans le DOM, l'image est cachée et remplacée par le texte contenu dans son attribut alt.

- Gestion du paramètre «Discover», qui va parser l'intégralité du DOM à la recherche de blocs sémantiques découverts par la partie Annotation. C'est ici que l'on retrouve le lien entre la partie Annotation et Adaptation :

```
function WorkToDo(Node, function) {
if (nom(Node) != ter_) {
    switch(function) {
        case: "Discover" :

            if EstUnBlocUtile(Node) {
                var CurrentNode = new NodeClass;
                CurrentNode.Name = NomBloc(Node); // Menu, Contenu...
                CurrentNode.Node = Obj;
                NodeArray.push(CurrentNode);
            }
        }
    }
}
```

La fonction EstUnBlocUtile permet donc de reconnaitre si un Node a été reconnu par la partie Annotation comme un bloc «Utile». C'est à dire, les structures qu'on autorisera l'utilisateur à afficher en focus (Menu, Contenu, Fil d'Ariane...).

Préparation à la récupération des blocs

Maintenant qu'il est possible, grâce au paramètre «Discover» de la fonction WorkToDo, de découvrir les blocs découverts par la partie annotation, nous obtenons une liste déroulante à l'intérieur du formulaire permettant de n'afficher que ces bloc et de supprimer tout ce qu'il y a autour (voir figure 4.3). En cliquant sur un certain nom de «Bloc», le plugin va remplacer l'intégralité de la page pour n'afficher que le bloc séléctionné. Cette fonction permet une concentration totale sur un bloc unique (voir figure 4.4).

FIGURE 4.2 – Affichage du site complet

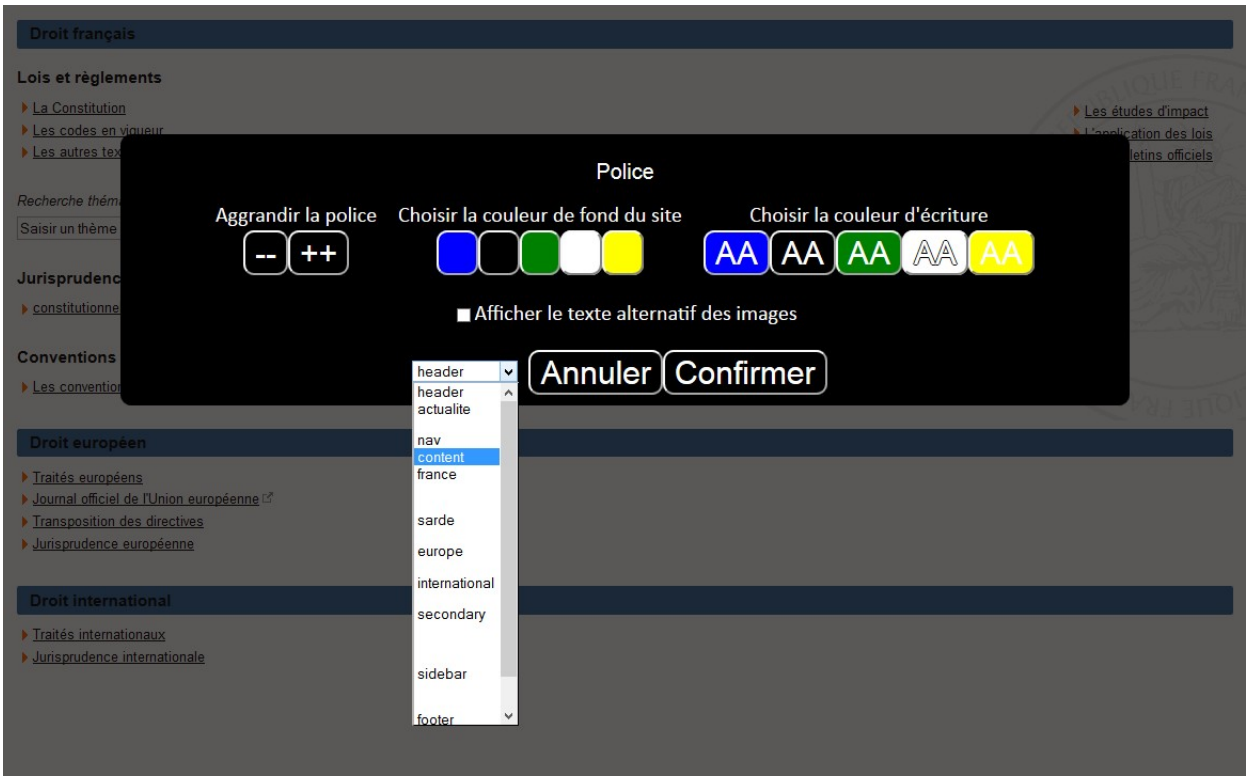

FIGURE 4.3 – Choix du bloc "Content" de la page

FIGURE 4.4 – Affichage du bloc choisi et suppression du reste

4.3 Discussion des résultats

4.3.1 Annotation

Les résultats de la partie annotation, sont en eux-mêmes insuffisants, mais remplissent néanmoins le rôle qui leur est attribué dans l'adaptation, c'est-à-dire permettre à l'utilisateur, de façon générique, de faire un focus sur les grandes structures composant un site Web. L'hétérogénéité du Web est le plus grand obstacle à un processus totalement générique d'annotation de structures visuelles. Tant qu'HTML4 sera toujours aussi utilisé pour concevoir les sites Web et que les standards du Web ne seront pas respectés, l'annotation demeurera problématique puisque la recherche de généricité nuira à l'efficacité et à la précision des procédures d'annotation.

4.3.2 Adaptation

Concernant les résultats escomptés, nous avons presque atteint les objectifs fixés. Des paramètres tels que la différence de contraste entre les différents blocs ou l'apparition du fil d'Ariane sont des paramètres qui peuvent être ajoutés au formulaire. Comme nous le pensions, la diversité des sites Web et leur façon d'être codés impliquent que le code fonctionne plus ou moins bien en fonction de la qualité de développement. Le script va se comporter correctement sur un site développé correctement, mais peut avoir un comportement bizarre sur des sites mal conçus. C'est pourquoi on ne peut être parés à tous les comportements possibles.

Chapitre 5

Perspectives d'amélioration

Il serait intéressant de rendre l'intégralité du plug-in multi-plateforme, retirant ainsi la limite d'utilisation qui est exclusivement réservée sous Firefox. Par la suite, d'autres paramètres que nous avons vus dans les objectifs ainsi que des améliorations de nos paramètres actuels peuvent être envisagés. Par exemple, au lieu de proposer 5 choix de couleurs, nous pouvons récupérer la couleur «globale» du site et proposer des jeux de couleurs qui sont dans une plage proche de cette couleur. Ainsi nous ne modifions pas le code couleur du site et respectons la charte graphique du développeur d'origine.

Offrir à l'utilisateur la possibilité déterminer des paramètres spécifiques à chaque type de structures composant un site Web pourrait être intéressant à l'avenir. Néanmoins les conflits que cette possibilité causerait devront être étudiés en détail et pourraient ici encore être un obstacle à la généricité.

Chapitre 6

Conclusion

Au cours de ce projet de recherche, nous avons travaillé à l'amélioration d'un plugin existant ayant pour but de faciliter l'accès au Web aux personnes atteintes d'un handicap visuel.

Ce plugin qui identifie des blocs visuels permet aujourd'hui de résoudre en partie ce problème d'accessibilité grâce à l'ajout d'une fonctionnalité principale permettant d'une part d'annoter ces blocs visuels afin d'en determiner la nature puis d'autre part de modifier ces blocs à la volée dans le DOM de la page Web en fonction de paramètres pré-sélectionnés par l'utilisateur.

Ce projet ce place comme une première étape vers l'accessibilité de l'informatique aux déficients visuels. En effet, il constitue désormais une base sur laquelle on peut imaginer ajouter plus de paramètres afin de mieux arranger les pages en fonction des envies de l'utilisateur mais aussi en respectant au plus proche la charte graphique d'origine du site concerné.

Annexe A

Annexes

A.1 Résumés d'articles

Résumé de publication
Function-Based Object Model Towards Website Adaptation
Warren Banguet

5 mars 2015

1 Introduction

Les auteurs se placent dans le contexte de l'internet contemporain, devenu multiple tant par la diversité et la masse d'informations disponible que par les moyens d'y accéder. En effet la navigation sur le web est aujourd'hui disponible via différents dispositifs et selon divers protocoles ou encore préférences utilisateur, ce qui rend indispensable de rendre adaptifs les sites web afin d'optimiser la navigation des utilisateurs.

Afin de représenter le contenu d'une page web dans l'optique de l'adapter, ils ont mis au point un modèle objet basé les les propriétés fonctionnelles des objets. Objets ici représentés par des parties cohérentes de la page web extraites au préalable. Des règles d'adaptation spécifiques à chaque Objet pourront ensuite être appliquées en fonction du résultat souhaité. Un exemple d'adaptation de page web pour une navigation via le WAP (Wireless Application Protocol) a été développé à titre d'exemple d'application.

2 Démarche

2.1 Génération du modèle objet

Dans leur démarche, les auteurs définissent deux niveaux dans leur modèle objet : l'un, basique, détermine les propriétés fonctionnelles des objets telles que son utilité de décoration de la page, les interactions qu'il offre à l'utilisateur, ses possibilités de navigation vers d'autres objets, etc... L'autre facette, spécifique cette fois, donne une catégorie de l'objet en prenant en compte les propriétés des objets le composant, cette catégorie reflète directement l'intention du développeur, c'est à dire l'usage qu'il est possible de faire de l'objet.

2.2 Adaptation pour le WAP

Après avoir catégorisé les structures contenues dans une page web, l'étape suivante de leur processus est de leur attribuer des règles dépendant de l'adaptation souhaitée. Ici dans le cas pratique de l'adaptation pour le WAP, les auteurs produisent un ensemble de règles générales à appliquer en fonction des propriétés du modèle objet basique.
Ces règles servent de support afin de réaliser l'adaptation basée sur le modèle objet spécifique, dans le cas pratique du WAP, cette adaptation se traduit par une réécriture en profondeur de la page web cible, le contenu étant fractionné en pages supplémentaires afin d'optimiser la navigation sur un écran bien plus petit et via un navigateur ne supportant pas la plupart des styles communs.

Résumé de publication
Extracting Content Structure for Web Pages based on Visual Representation

Baptiste LEULLIETTE

10 mars 2015

1 Introduction

L'objectif du papier est l'extraction de structures de contenus d'une page Internet. Ils envisagent l'utilisation du Tag Tree ou le DOM afin de récuperer cette structure de page. Le TagTree représente seulement les balises d'une page Web alors que le DOM représente des "Objets" Node, utilisés par le navigateur. Il s'agit d'une classe web, par abus de langage.

Le papier se base sur la hiérarchisation DOM du navigateur pour extraire les structures.

2 Démarche

Il y a une mise en place de deux définitions :

— Objet basique : représente une feuille de l'arbre DOM qui ne peux plus être décomposé.

— Structure de contenu visuel : un ensemble de blocs qui sont des "Objets Basiques".

Ces définitions permettent d'avoir un vocabulaire commun à tous. Aussi, ils définissent une page contenant 3 ensembles, un ensemble d'objets ou de "sous pages" nommé O, un ensemble des séparateurs visuels, horizontaux et veritcales, possèdant un poids et une relation de O vers O représentant les liens entre les "blocs".

Une fois cette structure extraite, les auteurs décident d'appliquer l'algorithme VIPS : il s'agit d'un algorithme qui vérifie s'il existe des propriétés dites "graphiques" associés au bloc de premier niveau dans chaque structure extraite afin d'en déduire si le bloc analysé est un bloc divisible ou non. Grace à cela, ils peuvent savoir si le bloc est divisible ou s'il faut le remplacer par ses propres fils et continuer récursivement.

Afin de peaufiner la divisibilité des blocs, les auteurs mettent en place diffèrentes heuristiques bien précises. Une fois qu'un bloc est totalement analysé, il est placé dans une "pool" de blocs.

Pour finir, à chaque entrée d'un bloc dans le pool, on analyse les séparateurs associés, en respectants des heuristiques précises. Ainsi on reconstruit, indirectement, l'ossature de la page web.

3 Résultats

D'après le papier, les résultats sont plutot excellents. En effet, sur 140 pages analysées, les auteurs considèrent 86 pages parfaitement extraites, 50 pages satisfaisantes et 4 pages échoués, à cause de mauvaise informations fournie par le navigateur (tests sous Internet Explorer). Soit 97 % de pages correctement extraites.

Résumé de publication
Recognition of Common Areas in a Web Page Using a Visualization Approach
Arlémi Turpault

13 mars 2015

1 Introduction

Cet article s'intéresse à une nouvelle façon de représenter une page WEB, de sorte à mettre en avant certains éléments clés d'une page tels que le haut de page, le bas de page, les colonnes etc.

Cette reconnaissance des blocs se fait par rapport à leur position vis-a-vis d'un écran virtuel et leur représentation sous forme d'un arbre hierarchique contenant toutes les informations nécesaires au positionnement de ces-dits blocs.

2 Démarche

2.1 Extraction des élements

Afin d'extraire les élements, les auteurs considérent un écran virtuel ayant une largeur fixe et une longueur infinie afin de représenter une page Web. Ils considérent pour la largeur une page plein écran standard (de l'époque) soit 1024px de large. Les heuristiques de cet article s'arrête à la reconnaissance des éléments suivants : Header, Footer, Left Menu, RIght Menu, Center. Par la suite, l'extraction se réalise en trois étapes :

- 1ère étape : Parsage de la page Web à l'aide d'un parseur HTML qui extrait deux types d'élements, les "tags" (reconnus par des chevrons <>) et le "data" (compris entre deux tags).
- 2ème étape : Chaque paire <TAG, DATA(s)> extraite et ajoutée à un arbre à l'aide d'un tree builder qui va construire la représentation de la page HTML appellé mTree.
- 3ème étape : On ajoute à chaque noeud de l'arbre la positions dans l'espace relatif de l'élément HTML qu'il représente. Ce nouvel arbre sera appellé MTree.

3 Résultats

Les résultats obtenus sont plutôt bons, après avoir passé 1600 sites dans leur crawler, 73% sont bien ou très bien reconnus.

	Header %	Footer %	Left Menu %	Right Menu %	Overall %
Not recognized	25	13	6	5	3
Bad	16	17	15	14	24
Good	10	15	3	2	50
Excellent	49	55	76	79	23

FIGURE 1 – Tableau de resultats

A.2 Documents de travail

A.2.1 État de l'existant

Document de travail
Etat de l'existant

Warren BANGUET

19 mars 2015

1 Généralités

F. Petidemange établit dans son papier un vocabulaire particulier afin de définir ses heuristiques. Ce sont ces dernières qui lui permettent de déterminer un arbre des structures visuelles de la page à partir de l'arbre DOM de cette dernière.

1.1 Vocabulaire

Noeud de contenu Représente tout noeud n'englobant qu'une valeure textuelle brute. N'est pas atomique au sens propre (car l'atome est ici le texte englobé) mais sera le plus bas niveau auquel nous nous intéresserons. Les noeuds de contenu représente les balises telles que <A>, <img>...
= Noeud Atomique

Noeud de contenu virtuel Représente un noeud dont le seul enfant est un noeud de contenu. Il n'a donc pas d'autre utilité que de décorer ou appliquer un style à son seul enfant. Ce sont actuellement ces noeuds qui sont le plus bas niveau reconnu par l'algorithme d'extraction.
= Noeud Atomique Virtuel = NAV

Noeud de description Représente un noeud donnant des méta-informations sur la page, telle la balise <style>

Noeud visible Représente un noeud qui sera effectivement affiché par le moteur graphique.

1.2 Heuristiques

Les 10 heuristiques présentes dans le mémoire de F.P. ont été implémentées dans sa version finale, cependant toutes ne sont pas appliquées dans les faits. Les tests d'application des heuristiques sont effectués "en série" (la première heuristiques applicable est directement appliquée), l'ordre dans lequel elles sont organisées est donc très important.

Règle 1 : Conditions : (Pas noeud de contenu) ET (Pas d'enfants visibles)
Effet : Filtrage

Règle 1bis : Conditions : (Noeud de Contenu Virtuel)
Effet : Annotation NAV Ajout dans l'Arbre des Structures Visuelles
Description : On obtient ici les noeuds englobant un et un seul noeud de contenu

Règle 2 : Conditions : : (un enfant visible) ET (enfant /= Noeud de contenu)
Effet : Division
Description : La division entraine un filtrage du noeud courant et une reprise du parcours sur les enfants du noeud courant.

Règle 3 : Conditions : (Tous les enfants = (Noeud de Contenu) || (Noeud de Contenu Virtuel))
Effet : Annotation Bloc Visuel Ajout dans l'Arbre des Structures Visuelles
Description : On obtient ici les noeuds conteneurs ne contenant QUE des noeuds de contenu (virtuels ou non)

Règle 4 : Conditions : (Il existe un enfant = (not inLine) ET (il existe un enfant dont taille > SEUIL))
Effet : Division
Description : Ici on filtre un noeud si l'un des enfants a une taille relative suffisamment importante pour être lui même un Bloc visuel à part entière.

Règle 7 : Conditions : (Le premier enfant visible du noeud est un titre)
Effet : Annotation Bloc Visuel Ajout dans l'Arbre des Structures Visuelles
Description : Partant du principe qu'un titre introduit un nouveau contenu, un bloc visuel est détecté.

Règle 5 : Conditions : (il existe un enfant = noeud <hr>)
Effet : Annotation 2 Blocs Visuels Ajout des 2 blocs dans l'Arbre des Structures Visuelles
Description : Un noeud HR introduit une séparation sémantique entre deux parties du bloc, ces 2 parties sont directement annotées come blocs visuels et ajoutées a l'arbre des blocs visuels

Règle 6 : Conditions : (SOMME(surface des enfants visibles) > surface noeud)
Effet : Annotation Bloc Visuel Ajout dans l'Arbre des Structures Visuelles
Description : Très peu utilisée dans les faits, problème de cohérence entre théorie et implémentation (papier dit filtrer mais implémentation annote bloc visuel) A VERIFIER

Règle 8 : Conditions : (parent(noeud) et noeud éloignés visuellement))
Effet : Annotation Bloc Visuel Ajout dans l'Arbre des Structures Visuelles
Description : La distance visuelle est déterminée par une fonction comparant les propriétés de style de deux balises. Cette différence permet de déduire l'existence d'un bloc visuel.

Règle 9 : Conditions : ((taille plus grand enfant) < SEUIL2)
Effet : Annotation Bloc Visuel Ajout dans l'Arbre des Structures Visuelles
Description : Si les enfants d'un noeud sont des noeuds conteneurs de taille inférieure à un certain seuil, alors ce noeud peut être annoté comme structure visuelle.

Règle 10 : Conditions : AUCUNE
Effet : Division
Description : Si aucune règle n'a pu s'appliquer au préalable, alors par défaut on divise.

On observe grace à ce listing qu'il existe 2 heuristiques possédant un seuil pouvant être ajusté au besoin.

2 Parcours de l'arbre DOM

Les heuristiques définies plus haut sont testées a chaque pas d'un parcours en profondeur afin de déterminer si chaque noeud est divisable. S'il l'est alors on divise : le noeud parent du noeud à diviser adopte les enfants de ce dernier et le noeud courant n'apparait pas dans l'arbre des structures visuelles en sortie. Dans ce cas on rappelle la fonction de division sur les enfants. L'autre alternative est une annotation comme structure visuelle du noeud.

Algorithme simplifié :

```
fonction constructionArbre(courant)
{
    poolExtrait = []
    divisionNoeud(courant,poolExtrait)
    pour tout noeud dans poolExtrait faire
      annoter(noeud)                // copie des annotations réalisées
        arbreBlocVisuel.ajouterEnfant(noeud)    // - sur l'arbre DOM
        constructionArbre(noeud)
}

fonction divisionNoeud(courant,pool)
{
    filtrage()
    heuristique 1BIS()      // Reconnaissance des noeuds de contenu
    si divisable(courant)   // Test des heuristiques
        pour tout noeud dans enfants(courant) faire
            si !(filtré(noeud))
                divisionNoeud(noeud, pool)
    else if visible(courant)
        pool += courant

}
```

3 Pistes

L'annotation pourra être effectuée en deux temps :

- Ajouts d'attributs sur les noeuds de contenu et conteneurs lors de leur découverte
- Typage des objets composites(blocs visuels) après la fin des appels récursifs sur leur branche

A.3 Comptes-rendus de réunions

Compte-rendu réunion TER
N°2 - CHOIX DES OUTILS ET REPARTITION

2 février 2015

1 Présents

- Arlémi TURPAULT
- Warren BANGUET
- Baptiste LEULLIETTE
- Taqiyédine ZEGAOUI

2 Objectifs

Determination des outils utilisés et répartition des taches et lectures. Dégager les objectifs du projet TER.

3 Compte-rendu

3.1 Outils choisis

Nous avions choisi auparavant ShareLatex pour l'écriture du rapport et GitLab pour le versioning du projet. Nous utilisons aussi maintenant Facebook dans un groupe privé afin de mettre en place une communication asynchrone.

3.2 Repartition des lectures

- Arlémi TURPAULT : Parametres utilisateurs
- Baptiste LEULLIETTE : Parametres utilisateurs
- Warren BANGUET : Annotation
- Taqiyédine ZEGAOUI : Annotation

3.3 Objectifs du rendu

Nous avons dégagé deux principaux objectifs :
- Annoter les differentes structures visuelles en fonction d'une sémantique établie
- Prendre en compte les préférences utilisateurs afin de modifier et améliorer l'affichage pour le déficient visuel

Compte-rendu réunion TER
N°3 - DIVISION EN SOUS-PROJETS ET REPARTITION

10 février 2015

1 Présents

- Warren BANGUET, secrétaire
- Arlemi TURPAULT, Chef de Projet
- Baptiste LEULLIETTE
- Taqiyéddine ZEGAOUI (Excusé car cours LOTS)

2 Objectifs

- Réaliser une division en sous projets (pré-WBS)
- Affectation des taches entre les collaborateurs
- Reflexions sur la stratégie de développement à adopter
- Aboutir a une ébauche de feuille de route

3 Compte-rendu

3.1 Division en sous projets

Première découpe évidente entre une partie **Annotation** et une partie **Paramètres Utilisateur**.
En effet ces deux parties sont indépendantes car l'arbre des blocs visuels est annoté avant que les paramètres soient appliqués à ses noeuds.

Division du sous projet Paramètres utilisateur en deux sous-sous-projets :

- l'un axé sur l'interface de saisie par l'utilisateur de paramètres puis leur synthèse (selection du meilleur des pires scénarios), en exprimant ces paramètres dans le langage naturel (contraste > 0.7, police > 16, etc...)
- l'autre axé sur la conversion de listes de paramètres ou règles à appliquer pour chaque bloc sémantique identifié en propriétés css et/ou code html pour ensuite insérer ces modifications dans le code de la page à afficher.

Une interface entre ces deux sous projets relatifs aux paramètres serait une structure de données où sera stockée la synthèse des paramètres utilisateurs et dans laquelle elle sera récupérée afin de la convertir et d'en appliquer les modifications (stockage probable dans un cookie).

3.2 pré-WBS et affectation des tâches

Nous avons extrait de ce premier découpage les étapes composant les sous-projets, le tout formant un WBS à étoffer.

3.2.1 Paramètres utilisateur

Cette partie sera réalisée par Baptiste et Arlémi

COMMUN :

- Cas d'utilisation (EP)
- Détermination des paramètres intéressants + faisabilité (EP)
- Etablissement d'une syntaxe commune pour les paramètres (ED)
- Détermination des modes de stockage et acces aux paramètres (ET + PROG)

Saisie + Synthèse :

- Synthese de l'ensemble de parametres util (ED)
- Gestion conflits/priorités, extraction du "meilleur des pires" scénarios (ED++ PROG)
- Creation de l'IHM de saisie (PROG)
- Ajouts de possibilités d'interaction sur la page (ED, ET PROG) (ex : La croix sur les blocs permettant de les masquer)

Conversion + Insertion

- Conversion en html/css d'un ensemble de parametres synthétisé (PROG)
- Insertion dans le code (ED + PROG)

3.2.2 Annotation

Cette partie sera réalisée par Warren et Taquiyedine

- Reprise du modèle proposé par F.P. (EP)
- Determination, Pertinence + faisabilité de l'ensemble E des types sémantiques (EP)
- Détermination de la valeur sémantique dans E d'un bloc donné (ED PROG) (Prog contraintes ou plus rigide ?)
- Syntaxe annotation du bloc (ED + PROG)

3.3 Stratégie de développement

Au vu du manque de visibilité concernant l'aspect technique de certaines parties du projet, il nous semble judicieux d'adopter une méthode de développement en spirale. Cela nous permettra de réaliser un noyau minimal contenant les focntionnalités les plus aisées à mettre en place, puis de l'étendre petit à petit par le biais de "rushs" afin d'implémenter chaque nouvelle fonctionnalité évoluée.

Compte-rendu réunion TER
N°4 - Avancée

13 février 2015

1 Présents

- Warren BANGUET
- Arlemi TURPAULT, Chef de Projet
- Baptiste LEULLIETTE
- Taqiyéddine ZEGAOUI (Excusé car cours LOTS)

2 Objectifs

- Constater l'avancée du projet
- Discuter des points bloquants

3 Compte-rendu

3.1 Code du plugin

Il y a actuellement deux codes sur le gitlab, un ne fonctionnant apparemment pas. Il faut comparer le contenu de ces deux codes pour voir ce qui a été modifié, comprendre pourquoi le plus récent ne fonctionne pas. Peut-être est-ce seulement dû à l'absence de coloration des blocs ? M. Meynard fera passer l'adresse email de Petitdemange pour que nous puissions entrer en contact avec lui et discuter de ça et d'autres choses.

3.2 Règles heuristiques

L'ordre des heuristiques n'est pas clair. Il faut que l'on valide les règles qui sont déjà définies, pourquoi telle règle est bonne ? Ou pourquoi elle ne l'est pas ? (Donner un exemple de site où ça ne fonctionne pas.) Le fait de proposer beaucoup de paramêtres risque de créer beaucoup de conflits (entre le contraste, la luminosité...), il sera donc utile de faire un développement en spirale. Au niveau de l'annotation il faut réfléchir au vocabulaire à utiliser. Il faudra ensuite discuter en réunion de sa validité.

3.3 Suivi de projet

Il faut penser à tous push sur le git.

Compte-rendu réunion TER
N°5 - Avancée

12 Mars 2015

1 Présents

- Warren BANGUET
- Arlemi TURPAULT, Chef de Projet
- Baptiste LEULLIETTE
- Taqiyéddine ZEGAOUI

2 Objectifs

- Constater l'avancée du projet
- Discuter des points bloquants

3 Compte-rendu

3.1 Paramètres utilisateurs

Baptiste Leulliette a contacté Pascale HUMBERT, membre de la fondation VISIO qui lui a donné des indications sur les paramètres les plus utiles pour les déficients visuels. Il ne faut cependant pas s'égarer, le but du projet n'est pas de proposer des modifications, c'est l'utilisateur qui doit pouvoir choisir ses paramètres. Le problème principal reste d'extraire les données sémantiques de la page web (partie Warren). On peut récupérer les fonctions objectives pour par exemple la gestion des contrastes. Pour les images l'alternatif visuel doit toujours être affiché, que les images le soient ou non. Pour la gestion de sauvegarde des paramètres, nous pensons utiliser la base de données SQLite qui existe de base dans GreaseMonkey

3.2 Analyse du code

Warren : correction du code de la dernière version qui est mieux et fait plus que la version précédente Petitdemange fait juste l'extraction de structure, il ne va du coup pas aussi "profond" que ce dont nous avons besoin, il extrait juste les blocs tandis que nous devons aller jusqu'au liens dans l'arbre des résultats -> Lui appelle ça un noeud atomique virtuel -> Il faut rediscuter du code, ça ne va pas ! Cela pose problème par exemple pour le CSS au cas où une balise enfant contient une propriété CSS qui écrase une propriété css plus haut. -> Il faut utiliser la balise .child() de jQuery qui permet de sélectionner tous les enfants d'une balise

3.3 Sémantique

On va se baser sur l'article qu'utilise Franck Petitdemange pour l'annotation, puis l'adapter car il est de base fait pour adapter les sites internets au WAP. Pour l'annotation on va rajouter différents attributs, propres à l'annotation par exemple "typeSemantique=""" Attention, atomique =/= composite, il faut se mettre d'accord sur le vocabulaire ! Pour la prochaine réunion de travail il faut emmener la fiche de vocabulaire et qu'on en discute pour être tous d'accord et qu'elle soit compréhensible par la communauté

3.4 Heuristiques

Les heuristiques ont changé entre les deux codes (Juin / Octobre), dans son code, pour chaque noeud exploré il y a une liste de conditionnelles. Dans son optique il cherche juste à repérer les blocs, peu importe ce qu'ils représentent tandis que pour nous il faut savoir à quoi ils correspondent Il faudrait déjà arriver à reconnaitre les menus dans le même type que HTML5 avec "nav".

Il y a deux types de règles, soit celles qui divisent soit une règle qui dit que c'est un bloc visuel s'applique. Il n'y a pas besoin de refaire sa partie, il y a juste a modifier un peu d'heuristiques... pour l'instant les annotations ne sont que liées aux règles qu'il a faite -> pas vraiment d'annotation. Est-ce qu'il faudrait annoter par rapport à un ensemble de règles ? Pour certaines choses c'est faisable oui.

L'idée est de faire deux parcours, un premier avec ses règles puis un deuxième parcours pour l'annotation qui créera un plus petit arbre. Le métamodele HTML5 n'est pas utile.

3.5 Prochaine rénion

Pour la prochaine réunion nous aurons une discussion sur le vocabulaire et les annotations telles que nous pourrons les déduire des 2 parcours (le sien puis le notre sur son arbre préalablement créé). TODO : -> Sortir liste des heuristiques

Annexe B

Bibliographie

References

[1] Franck PETITDEMANGE, *Inférence de la structure d'une page Web en vue d'améliorer son accessibilité*. LIRMM, Montpellier Stage M2 Recherche, juin 2014.

[2] S. NESTOROV, S. ABITEBOUL, R. MOTWANI *Inferring Structure in Semistructured Data*. Department of Computer Science, Stanford University, USA.

[3] D.W EMBLEY, Y.S JIANG, Y.-K NGY *Record-Boundary Discovery in Web Documents*. Department of Computer Science, Brigham Young University, USA.

[4] J. CHEN, B. ZHOU, J. SHI, H. ZHANG, Q. FENGWU *Function based Object Model towards website adaptation*. Microsoft Research China & Tsinghua Univ., WWW10, May 1-5, 2001, Hong Kong.

[5] Handicap Zero *Site adaptatif pour les déficients visuels* HandicapZero.org

[6] D. CAIL, S. YU, J-R WEN, W-Y MA. *Extracting Content Structure for Web Pages based on Visual Representation* Microsoft Research Asia, Peking University, Beijing, P.R.China.