



Livre blanc

Formulaires web

Accessibilité, design et ergonomie

Nicolas Catherin | Vincent Valentin | 2011

 Clever Age

Table des matières

Avant-propos	4
I. Présentation et personnalisation des éléments.....	6
1. Les éléments natifs pré-HTML5	7
<i>Input</i>	7
<i>Label.....</i>	7
<i>Champs de saisie.....</i>	8
<i>Zones de saisie (textarea).....</i>	9
<i>Cases à cocher</i>	10
<i>Boutons radio</i>	10
<i>Bouton parcourir, un cas à part.....</i>	10
<i>Listes déroulantes.....</i>	11
<i>Boutons d'action et lien d'action.....</i>	12
2. Les limites de la personnalisation	13
3. HTML5 : apports et limites.....	15
<i>Le placeholder</i>	15
<i>L'autofocus.....</i>	16
<i>Les champs obligatoires : required.....</i>	17
<i>Les datalist.....</i>	17
<i>Input search.....</i>	17
<i>Input number.....</i>	18
<i>Input number.....</i>	18
<i>Date et heure.....</i>	19
II. Ergonomie et bonnes pratiques	21
1. L'objectif est la complétion	22
<i>Nommer et introduire le formulaire.....</i>	22
<i>Limiter sa longueur.....</i>	22
<i>Ne demander que les champs indispensables.....</i>	22
<i>Lorsqu'un champ n'est pas éditable, ne l'affichez pas !.....</i>	23
<i>Afficher progressivement les champs dépendants</i>	23
<i>Tunnel à étapes successives ou système à page unique</i>	23
<i>Regrouper les contenus similaires.....</i>	25
<i>Permettre une navigation au clavier</i>	25
<i>Savoir se passer de Captcha (ou tests de Turing)</i>	25
2. Aligner le couple label / champ	28
<i>Label placé en fer à gauche</i>	28
<i>Label placé en fer à droite.....</i>	29
<i>Label placé au dessus.....</i>	29
3. Longueur de champ et « affordance »	31
<i>Le cas particulier des sites multilingues</i>	32
4. Hiérarchisation des actions.....	33
<i>Les actions primaires</i>	33
<i>Les actions secondaires</i>	33
<i>Les actions tierces.....</i>	34

<i>Le positionnement des boutons de validation</i>	35
5. Indiquer le caractère obligatoire (ou non) des champs	36
<i>Regrouper autant que possible les champs obligatoires</i>	36
6. Gérer le feedback utilisateur	38
<i>La validation à la volée</i>	38
<i>Les messages d'erreurs</i>	40
<i>L'indication du processus en cours</i>	40
Concluons !	41
Webographie	44

Crédits

Ils ont contribué de près ou de plus loin à la réalisation de ce livre blanc, merci à eux : Sébastien Delorme, Xavier Lacot, Yan Paquis, Jérémie Patonnier, Sophie Taboni, Clever Age, et ceux que nous oublions peut-être !

Ce livre blanc est distribué sous licence Creative Commons, il peut être librement utilisé à condition de l'attribuer aux auteurs en citant leurs noms (droit de paternité) et en l'utilisant selon une licence identique (licence CC-by-sa).



Avant-propos

Avant toute chose, sachez que nous sommes conscients qu'écrire un guide de recommandations sur un support aussi mouvant qu'Internet est destiné à devenir partiellement caduque dès que les spécifications ou les usages évoluent. Cette publication est donc logiquement ouverte aux remarques et à l'enrichissement par la communauté afin de nourrir sa propre évolution.

Nous ne détenons pas la vérité ! L'accessibilité comme l'ergonomie sont des disciplines où le rapport entre l'homme et l'interface est sans cesse différent. Les experts tirent de grands schèmes de comportements qu'il fait bon connaître pour savoir s'ils sont applicables et légitimes à nos situations propres, mais n'oublions pas que les tests utilisateurs et leur propre remise en question au cours de la vie d'un site Internet sont la véritable clef de voûte de la longévité du projet.

Dans cet ensemble, les formulaires web ne sont qu'une infime partie des éléments interactifs offerts aux internautes, et sont aussi parmi les plus rébarbatifs. Mais inutile d'essayer de s'y soustraire car ils représentent un enjeu majeur, notamment pour les processus de transformation.

Afin de rendre digeste ce vaste sujet, nous avons privilégié la synthèse. Ce livre blanc ne couvre donc pas 100 % des problématiques liées aux formulaires, mais tend à vous proposer un mémo argumenté et illustré de ce que nous jugeons être des pratiques allant dans le sens d'une démarche de « qualité web ».

I. Présentation et personnalisation des éléments

Les éléments natifs d'un formulaire sont propres au navigateur Internet et au système d'exploitation utilisé. Leur aspect graphique n'est pas prévu pour être en harmonie avec l'esthétique de vos sites Web et c'est pourquoi le designer peut souhaiter, à raison, ajuster ces éléments afin qu'ils s'intègrent davantage à la direction artistique du projet.

Deux degrés d'adaptation s'offrent à lui :

- il modifie entièrement l'aspect graphique de ces éléments par le biais de *JavaScript* ;
- il apporte une amélioration progressive en utilisant les propriétés *CSS*.

Cette seconde solution est graphiquement frustrante parce qu'elle ne nous permet pas d'aller jusqu'au bout de la personnalisation. Il s'agit néanmoins de la seule manière, à date d'écriture de ces lignes, de satisfaire entièrement les exigences d'accessibilité et d'utilisabilité que requiert la démarche qualité.

Noter également qu'HTML5 n'est pas fiable à 100 % (calendrier, carrousel), mais les éditeurs de navigateurs y travaillent !

Les conséquences de la personnalisation graphique et fonctionnelle étant abordées plus tard, commençons par une présentation des éléments natifs pré-HTML5 et de leur bon usage. Ergonomie et bonnes pratiques sont donc de la partie.

1. Les éléments natifs pré-HTML5

L'évolution du langage a pris en considération la démocratisation de certains éléments traités en *JavaScript* pour les intégrer dans ses propres spécifications. Concentrons nous donc, dans un premier temps, sur les briques élémentaires des formulaires et définissons-les.

INPUT

L'*input* est un élément clé. Il permet de recueillir une information venant de l'utilisateur, qu'il s'agisse d'un espace de saisie (ex : champ de saisie), de sélection (ex : bouton *radio*) ou de soumission (ex : bouton).

Cette brique est un parfait couteau suisse et nécessite un « *type* » pour se spécialiser :

- checkbox ;
- file ;
- image ;
- password ;
- radio ;
- reset ;
- submit ;
- text ;
- hidden ;
- button ;
- ...

Cette variété de spécialisations est encore plus riche avec HTML5.

LABEL

Il s'agit du texte qui, lié à l'*input*, précise l'information demandée. On le préfère court et explicite. *Label* et *input* forment un couple qui ne doit ni être séparé, ni associé à un élément supplémentaire. On prendra naturellement soin de les rapprocher visuellement puisqu'ils vont ensemble, en toutes circonstances.

Basic Information

Fill-in some quick information about yourself below. Remember that your invite code to your email address. So, please use a real one.

First Name

Last Name

Email Address

Twitter Username

so we can send you an invite code *optional, but recommended*

Your Website or URL to Something You've Made

Légende : Couples label/input où l'on applique un style de curseur « pointer » ([Forst](#)).

Allons un peu plus loin : le label est cliquable par défaut et active l'*input* auquel il est rattaché. On peut donc opter pour un style de curseur « pointer » que l'on retrouve nativement au survol des liens. Il s'agit d'un détail et nous n'avons pas eu l'occasion d'en mesurer l'effet sur l'utilisateur, mais l'idée nous est séduisante, bien davantage que cette pratique consistant à souligner ses intitulés, voire à les habiller du bleu hypertexte.

Sachez également que le *label* va permettre de relier techniquement le champ du formulaire à son intitulé. Cela prend toute son importance pour les personnes aveugles ou présentant des difficultés de précisions à la souris, surtout pour les éléments de petites tailles (boutons *radio*, cases à cocher...).

CHAMPS DE SAISIE

Le champ de saisie (que l'on distingue de la zone de saisie – ou *textarea*) suggère un nombre de caractères restreints que l'on peut lui imposer en fonction d'une norme à respecter (exemple de la norme AFNOR¹ qui préconise 38 caractères par ligne – espaces inclus – pour la saisie des adresses postales). On peut également ne rien lui imposer, la saisie est alors infinie mais toujours sur une ligne.

¹ Lire les 6 règles d'or de la norme [AFNOR](#)

Il est délicat de contraindre l'utilisateur à une saisie normée et stricte. On suggère ainsi d'être le plus permissibile possible car une même information peut être saisie de diverses façons (un numéro de téléphone existe au format international ou national). En considérant la contrainte de la consistance des bases de données, on pourra indiquer à l'utilisateur quel est le format attendu par une aide à proximité du champ. Le contrôle final se fera, à minima, côté serveur.

Il est important de prévoir un champ suffisamment long pour que le texte saisi soit entièrement visible en utilisant systématiquement l'attribut `size` que l'on pourra compléter d'une sur-couche CSS dans un souci d'esthétique (alignement sur une grille typographique par exemple). L'information de longueur est ainsi préservée même si les feuilles de style sont désactivées. Elle est donc ici d'ordre ergonomique et non graphique.

Le diagramme présente trois exemples de champs de saisie rectangulaires de différentes longueurs, alignés à gauche. Le premier est étiqueté 'Petit champ', le second 'Champ moyen' et le troisième 'Grand champ'.

Légende : La longueur des champs de saisie oriente l'utilisateur sur le nombre de caractères attendus.

ZONES DE SAISIE (TEXTAREA)

Il s'agit d'une zone d'expression libre permettant à l'internaute un plus grand confort pour rédiger des contenus sur plusieurs lignes.

Un des avantages de ce champ de formulaire est de pouvoir être redimensionné par l'internaute s'il n'est pas à l'aise avec la barre de défilement verticale sur les navigateurs modernes (Safari, Chrome, Firefox 4+).

Un bémol cependant, cette action n'est pas possible au clavier. Une fois n'est pas coutume, certains scripts bien pensés permettent de palier à cette absence.

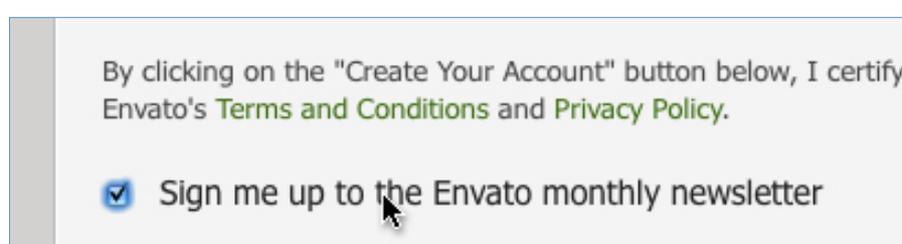
Légende : Le coin inférieur droit d'un champ de saisie permet, sous Safari, Chrome et Firefox 4+, de redimensionner la taille de l'input (Forrst).

L'image montre un formulaire web intitulé 'Your Feedback for Trevor Morris'. À l'intérieur, il y a un grand champ de saisie (textarea) contenant le texte 'Here is what I like about the post above, and my thoughtful improvements'. En bas à droite du champ, un petit curseur de redimensionnement est visible. En dessous du champ, il y a un bouton vert avec le texte 'Request My Forrst Invite'. En haut à droite du formulaire, il y a un lien '(Markdown is supported)'. Le formulaire est sur un fond gris clair.

CASES À COCHER

Elles offrent la possibilité de choisir plusieurs options simultanément. L'utilisateur peut cocher l'ensemble, quelque-unes ou aucune s'il ne s'agit pas d'une information obligatoire. À noter que la case à cocher peut être utilisée seule, comme dans le cas d'acceptation des conditions générales de vente pour les sites de commerce électronique.

Une remarque : dans certains navigateurs et dans le cas où elle est associée à un *label* contenant un lien hypertexte, le clic sur ce dernier coche la case. Il faudra donc distinguer le lien du *label*.

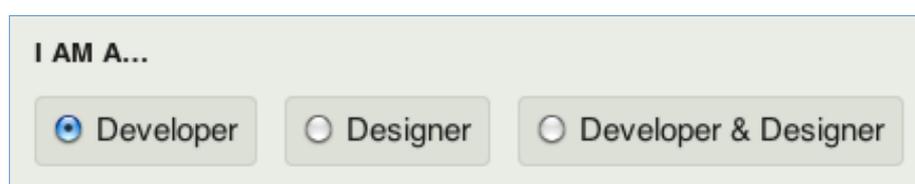


Légende : Case à cocher unique, au clic sur le label, l'input est complété (Forrst).

BOUTONS RADIO

Ils contraignent l'utilisateur à ne choisir qu'une option parmi plusieurs tout en affichant l'ensemble des choix proposés. Les boutons *radio* vont toujours, au minimum, par deux.

Lorsque l'un est activé, il n'est plus possible de le décocher autrement qu'en sélectionnant un autre bouton *radio*. Il devient nécessaire d'utiliser, au besoin, un bouton permettant l'annulation.



Légende : Bouton radio, chacun des boutons est visuellement proche de son label qui se positionne à sa suite (Forrst).

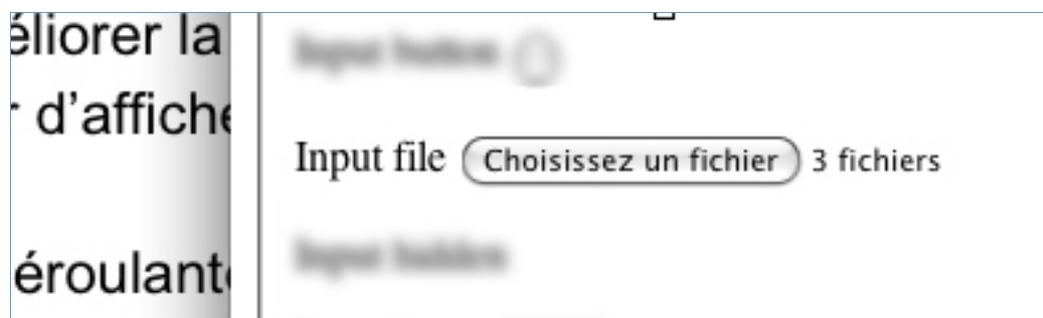
BOUTON PARCOURIR, UN CAS À PART

Il permet d'envoyer un document depuis l'ordinateur de l'utilisateur vers le site Web.

Ce bouton est une plaie pour tous les designers car il s'agit du seul à ne pas pouvoir être personnalisé du tout (sans solution *JavaScript* ou *Flash*).

Notons déjà que HTML5 prévoit à terme des mécanismes qui pourraient contourner ce problème épineux.

De leur côté, les navigateurs modernes² permettent la sélection multiple de documents, pratique !

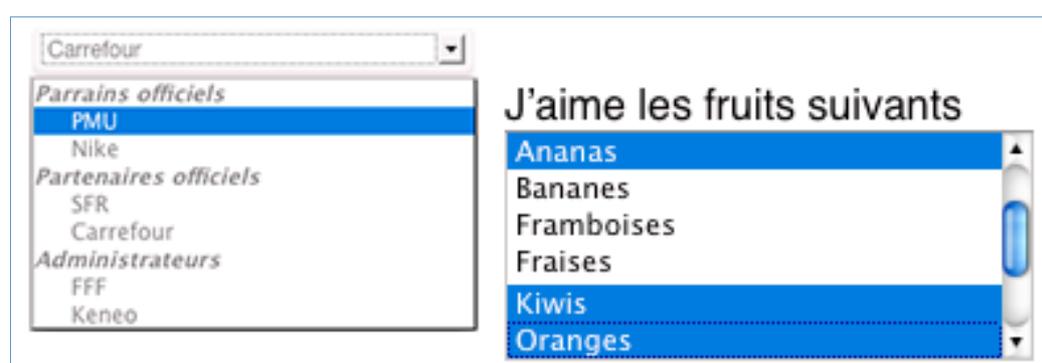


Légende : Le bouton « parcourir » est stylisé selon la mode des boutons natifs au système d'exploitation utilisé (mac OS).

LISTES DÉROULANTES

Elles autorisent le choix parmi un grand nombre d'options pour une occupation minimum de l'espace visuel. On peut trier et appliquer des séparateurs aux éléments de la liste dans le souci d'améliorer la lisibilité des informations, tout comme on peut choisir d'afficher n'importe quel élément de cette liste par défaut.

Les listes déroulantes acceptent les différents comportements de sélection unique ou multiple à la manière des cases à cocher. Il est également possible de leur imposer une taille size affichant un nombre déterminé d'options. Présenter ces éléments dans un ordre logique (alphabétique, numérique, degré d'importance) est par ailleurs tout à fait recommandé !



Légende : Les listes déroulantes permettent la classification et la sélection multiple.

² Chrome, Firefox, Opera et Safari.

BOUTONS D’ACTION ET LIEN D’ACTION

Avant toute chose, nous considérons que les actions primaires sont traduites par des boutons d’action quand les actions secondaires sont des liens d’action. Cette distinction est primordiale dans les situations où les styles CSS sont désactivés. Ainsi, une action primaire garde sa prééminence à l’aide de la balise *button* quand une action secondaire devient un lien hypertexte. On préserve ainsi l’affordance³ et hiérarchie visuelle.

Rappelons aussi que les liens ne peuvent pas par défaut interagir avec les formulaires, ce sont des composants de l’interface qui ne sont pas liés entre eux techniquement.

The image shows a login form with a light gray background. It contains two input fields: 'Email *' and 'Mot de passe *'. Below the input fields, there are two buttons: a prominent orange button labeled 'Se connecter' and a smaller, less prominent link labeled 'Retour à l'étape 1'.

Légende : Bouton d’action et lien d’action se distingue graphiquement de manière très significative. L’action primaire attire davantage l’œil de l’utilisateur (Clever Institut).

Le **bouton d’action** permet de valider une étape, de soumettre le formulaire ou encore d’appliquer une remise à zéro sur l’ensemble des champs. D’un point de vue ergonomique, on préconise de ne pas utiliser de *reset*⁴. Si l’utilisateur souhaite modifier un ou plusieurs champs, il peut le faire sans effacer l’ensemble de sa saisie et conserver les précieuses informations renseignées.

Les **liens d’action** sont utilisés dans tous les autres cas ! Avec la sur-couche CSS ou directement en images, ils s’intègrent tout-à-fait à leur environnement graphique. On les utilisera pour indiquer le retour vers des étapes précédentes, pour modifier, afficher, masquer, sauvegarder...

Nous y sommes ! Nous venons de lister les éléments natifs, tous ou presque, et nous en avons profité pour apporter une première couche de qualifications. Abordons désormais leur personnalisation.

³ *Il s’agit de la capacité d’un objet à suggérer ce à quoi il sert avant même de l’utiliser et sans aide tierce. Elle est influencée par nos réflexes innés et notre expérience.*

⁴ *Ergonomie Web, pour des sites web efficaces, Amélie Boucher, éd. Eyrolles novembre 2007.*

2. Les limites de la personnalisation

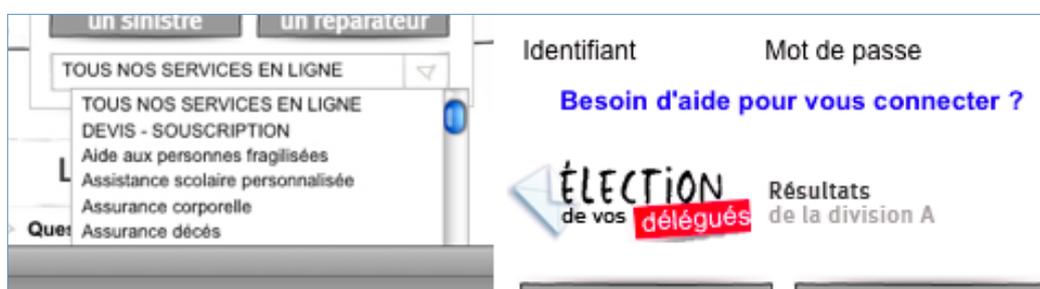
Nous vous l'avions annoncé en introduction : le recours systématique à la personnalisation utilisant *JavaScript* n'est pas conseillé.

En effet, la rendre accessible implique un travail de réflexion plus conséquent qu'en passant par l'utilisation des CSS. Nous nuancions cependant notre propos pour les éléments de formulaire de l'ère post-HTML5 car les éditeurs de navigateurs ne les rendent pas encore pleinement compatibles. Dans certains cas, donc, les scripts JS pourront être préférés.

À ceux qui maugréent que leur cible n'est pas sujette au handicap quel qu'il soit, rappelons que rendre un site accessible, c'est aussi permettre son utilisation lorsque l'on ne bénéficie pas d'un confort optimal (absence de souris ou *trackpad*, navigation sur mobile, *plugins* bloqués, écran en plein soleil...).

Et pour vous convaincre, voici des exemples de conséquences d'une mauvaise personnalisation via *JavaScript* :

- la liste déroulante ne peut plus être atteinte via la touche « tabulation ». Toutes les surfaces n'étant pas encore tactile, sans souris il sera impossible d'atteindre les champs ;
- le texte contenu dans une liste déroulante sort de son cadre si l'on effectue un zoom texte (à ne pas confondre avec un zoom page), et peut ne plus être lisible ;
- lorsqu'une liste déroulante comprenant de nombreuses options se situe en bas de la fenêtre navigateur, elle ne sort plus de la fenêtre de ce dernier. Le contenu est masqué et inaccessible ;
- lorsque les couleurs sont désactivées, un *input* n'est plus identifiable comme tel, voire n'est plus visible du tout ;
- si *JavaScript* est désactivé, l'affichage peut vraiment devenir chaotique.



Légende : Liste déroulante dont le contenu sort de l'écran sans passer par dessus la fenêtre du navigateur, et input dont le contour n'est plus visible ([Maif](#)).

Il existe pourtant des plugins basés sur l'ensemble des *frameworks*⁵ existants (ou presque) qui tendent à permettre l'accessibilité des éléments de formulaires personnalisés (voir le travail du [Filament Group](#)). Cependant, si vous n'avez pas de solides connaissances en *JavaScript* et en accessibilité, nous vous déconseillons de vous en remettre aveuglément à un *plugin* « clef-en-main » et dans la mesure du possible, testez et re-testez en situation réelle.

Vous l'aurez compris, nous privilégions la personnalisation via CSS et ceci même si le design est d'autant plus contraint. La possibilité d'appliquer certains styles à certains éléments (champ de saisie à ligne unique, zone de saisie, boutons et liens) signifie surtout que vous n'avez pas la main sur l'ensemble des paramètres graphiques, sans compter les différences de résultats que vous obtiendrez d'un navigateur à l'autre, d'un système d'exploitation à l'autre (même si cette référence date un peu, les tests réalisés par [456 Berea Street](#) sont toujours valables).

Il est toujours envisageable d'utiliser les styles CSS, en s'aidant d'un peu de *JavaScript*, pour tendre vers un rendu commun quelque soit le système d'exploitation et le navigateur utilisé. Nous n'avons pas eu l'occasion de mettre en pratique ce type de solutions et donc d'en mesurer l'impact sur l'accessibilité, si vous souhaitez en apprendre davantage il vous est possible d'explorer les réalisations de [Formalize](#).

Au delà de l'aspect technique, notre souhait est également de vous sensibiliser à la problématique des **modèles mentaux** et de leur représentation. Les utilisateurs ont tous une idée, une « image » de ce à quoi doit ressembler un bouton, un lien, un champ de saisie ou une liste déroulante. Leurs cultures, leurs habitudes et leurs usages façonnent ces représentations. En créant de nouvelles formes par le biais de la personnalisation, nous risquons de leur demander au mieux de s'adapter promptement, au pire de se ré-approprier l'outil. Rappelons que nous sommes dans une situation délicate. Les formulaires ont une nécessité de transformation forte et ils sont, par principe, rébarbatifs. Imposer à l'utilisateur de fournir un effort n'ayant aucun lien avec l'objectif, nous apparaît comme indélicat et susceptible de le décourager.

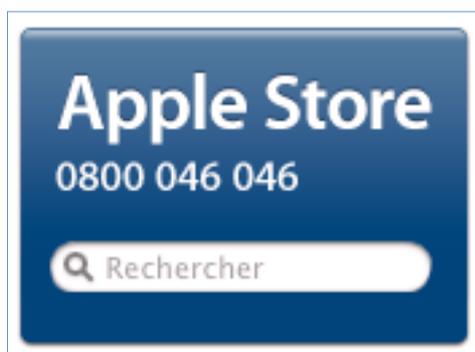
⁵ Un *framework* est un ensemble de briques structurelles permettant de composer les fondations d'une architecture technique ([Définition du framework sur l'encyclopédie Wikipédia](#)).

3. HTML5 : apports et limites

Nous l'avons évoqué : HTML5 enrichit les éléments natifs avec un lot de fonctionnalités qui nécessitaient auparavant l'emploi de *JavaScript*. Si l'intention est plutôt bonne, on se retrouve malheureusement confronté au problème de la reconnaissance de cette version du langage par les navigateurs. Prenez donc soin de toujours proposer une alternative compatible.

LE PLACEHOLDER

Son rôle est d'apporter de l'aide à la saisie en indiquant l'information ou le format de l'information attendue. Il se positionne à l'intérieur même de l'*input*, se présente le plus souvent dans une couleur plus claire et disparaît lorsque l'utilisateur positionne son curseur dans le champs. Si l'on en vient à quitter ce dernier sans avoir renseigné le moindre texte, le texte réapparaît.



Légende : Placeholder d'un champ de recherche ([Apple](#)).

Il présente l'intérêt de fournir une aide contextuelle lorsque le design est contraint par un espace restreint. Attention toutefois à ne pas le faire ressembler à un texte déjà saisi sous peine de voir l'internaute passer complètement à côté.

Le contraste est donc la clé. Trop fort, on pourrait l'assimiler à un texte saisi. L'utilisateur risquerait de concentrer son attention sur le texte et donc briser son rythme de complétion ou passer complètement à côté en le considérant comme déjà rempli. Trop faible il devient inutile et ne répond pas aux normes d'accessibilité. À l'heure actuelle, la personnalisation des *placeholder* via *CSS* est encore très limitée.

Noter également qu'il disparaît à l'activation du champ, si l'utilisateur est distrait (ou s'il navigue par tabulation), il peut ne jamais avoir connaissance de cette aide. C'est d'autant plus délicat si vous avez la mauvaise idée de l'utiliser à la place des intitulés. Certains tests suggèrent que les lecteurs d'écran semblent en mesure de palier à l'absence de *label* en utilisant le *placeholder* pour générer l'*AccessibleName*. En présence du *label*, le *placeholder*, et donc son caractère informatif, seront ignorés. Étrangement, le comportement des lecteurs d'écran n'est pas identique à ce sujet : en l'absence de *label*, NVDA⁶ donne la priorité au *title* quand Jaws⁷ la donne au *placeholder*.

OPQUAST préconise de placer les aides indiquant le format d'information attendu⁸ (pour une date de naissance par exemple) dans le *label*. Il s'agit sans doute de la meilleure solution en terme d'immédiateté de l'information.

Elle peut néanmoins poser des problèmes d'ordre ergonomique et graphique dans le cas où les intitulés ont des longueurs très différentes (cas des formulaires avec alignement gauche ou droite) en créant des décrochages très forts gênant la lecture et donc la complétion.

Le *label* peut englober intitulé et champ dans leur ensemble (tout en conservant l'attribut *for=""*) permettant de placer ces informations soit avant, soit après le champ. L'éventuel *placeholder* sera alors intégré au *label*, et pourra être porteur d'une information complémentaire.

L'AUTOFOCUS

Il permet de positionner le curseur dans le premier champ du formulaire lors du chargement de la page. L'intention semble plutôt bonne, l'utilisateur est assez discrètement positionné dans les starting-block pour attaquer la complétion. Néanmoins, ce procédé peut priver l'utilisateur du contexte. Les lecteurs d'écran non mis à jour placeront l'internaute au premier *label* sans information complémentaire.

Autre bémol s'adressant à une part plus restreinte d'utilisateurs, le défilement de page à l'aide de la barre d'espace ne fonctionne plus (l'utilisateur saisit des espaces dans le champ).

Il existe un cas très spécifique où l'autofocus est intéressant : lorsque le formulaire est le seul élément fonctionnel de la page et conditionne son usage (comme une page de recherche : Google).

⁶ Accéder au site de la [NonVisual Desktop Access](#).

⁷ Accéder au site de [JAWS](#).

⁸ Bonne pratique n° 119 : l'étiquette de chaque champ de formulaire indique, le cas échéant, quel format de saisie doit être respecté ([référentiel OPQUAST](#)).

LES CHAMPS OBLIGATOIRES : REQUIRED

La vérification d'une saisie d'information dans les champs déclarés obligatoires peut s'effectuer désormais en HTML5. Sa bonne prise en compte en accessibilité passera, pour l'instant, par l'utilisation d'*ARIA-required=""*. Nous notons que l'élément *required* est de plus en plus supporté (Jaws 11 et NVDA sous Firefox 6 par exemple) et le champ obligatoire sera annoncé. Le bémol restant concerne les messages d'erreurs non lus par Jaws, lus par NVDA mais sans qu'il ne précise à quel champ il se rapporte. Ces évolutions sont en marche, montrons-nous patients.

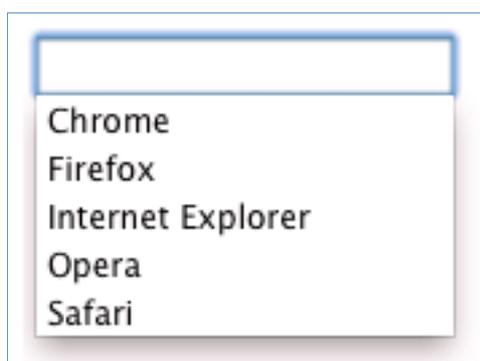
Si le champ n'est pas renseigné une erreur sera remontée à la validation du formulaire.

Cette vérification côté client pouvant être facilement contournée par les utilisateurs il est évidemment obligatoire de la doubler par un contrôle serveur.

LES DATALIST

Il s'agit d'un *input* mêlant champ de saisie et liste déroulante. Si l'utilisateur ne trouve pas ce qu'il souhaite dans la liste, il peut librement saisir de l'information. Un exemple d'utilisation de cet élément : le cas où vous voulez vous passer d'une option « si autre, précisez » qui ferait apparaître un champ de saisie à la suite. Cet élément est complètement nouveau et demandera à l'utilisateur d'acquérir l'expérience nécessaire à son bon usage. Une aide contextuelle peut alors être tout à fait appropriée.

C'est aussi cet élément qui remplacera avantageusement les champs auto-complétés, la liste d'éléments pré-remplis étant alors mise à jour en fonction de la saisie effectuée.



Légende : L'input DataList mêle champ de saisie et liste déroulante.

INPUT SEARCH

Son comportement ne change pas d'un champ de saisie, en revanche son apparence est déterminée par le navigateur afin qu'il ressemble au champ de recherche du système d'exploitation. Cette homogénéité de rendu réduit la charge cognitive imposée à l'utilisateur.

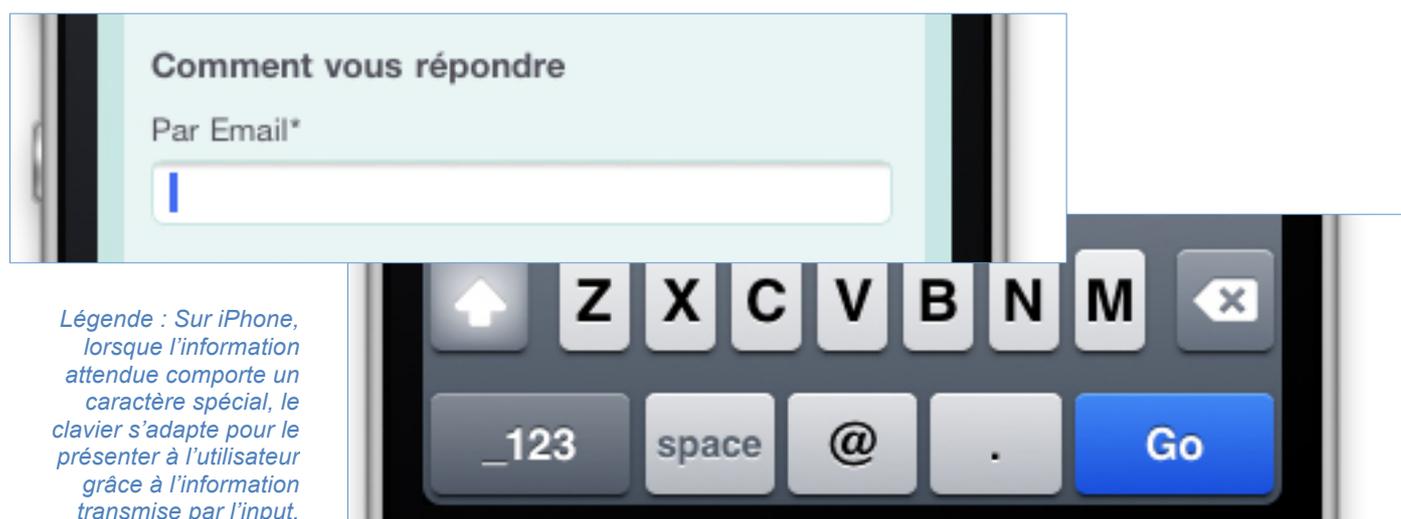
Input type "text" :

Input type "Search" :

Légende : Sous Opera, par exemple, l'input type « Search » voit ses extrémités arrondies, à la manière du champ de recherche du navigateur.

INPUT EMAIL / URL

Visuellement, ces *input* restent des champs de saisie traditionnels. Le navigateur bénéficie d'informations complémentaires sur la nature des caractères attendus.



Légende : Sur iPhone, lorsque l'information attendue comporte un caractère spécial, le clavier s'adapte pour le présenter à l'utilisateur grâce à l'information transmise par l'input.

Dans le cas d'une interface tactile, le système de cette dernière pourra décider s'il affiche le clavier numérique ou alphabétique à l'activation du champ. L'expérience utilisateur n'en est que meilleure.

INPUT NUMBER

Ce type ajoute des boutons fléchés à un champ de saisie attendant une valeur numérique. L'utilisateur pourra renseigner la valeur de son choix et l'augmenter ou la diminuer à l'aide de ces flèches. Il est nécessaire de tester la proximité de ces deux boutons afin de déterminer si les utilisateurs parviennent à cliquer facilement.

Avec ce type de champ, on précise simplement qu'une valeur numérique est attendue. Bien souvent, le navigateur adaptera l'interface du champ pour offrir des boutons facilitant l'incrémentation ou la décrémentation de cette valeur.

Input number 

Légende : Les boutons d'incrément / décrémentation situés à droite de l'input sont susceptibles d'être trop petit pour être réellement utilisables.

DATE ET HEURE

Les calendriers sont devenus monnaie courante parmi les sites de réservation en ligne ou de création d'évènement. Il serait donc tout à fait pertinent de les voir devenir natifs afin qu'ils bénéficient de l'homogénéité qui leur faisait défaut. Leur paramétrage est assez poussé, on peut définir la date (jour, mois, année), le fuseau horaire, l'horaire (heure, minute, seconde) ou une combinaison de tout ceci. Les utilisateurs bénéficient déjà d'une bonne connaissance du fonctionnement de cet outil, son passage en natif ne devrait pas souffrir d'un besoin d'apprentissage.

Malheureusement, les calendriers ne sont pas accessibles et ne semblent pas prêts de l'être dans un avenir proche ! La solution consiste donc à ne pas se satisfaire du *datepicker* HTML5 mais d'utiliser *JavaScript* et de présenter un champ de saisie libre (pour lequel nous aurons pris le soin d'informer l'utilisateur du format attendu si nécessaire), et d'adjoindre un bouton permettant de déclencher l'affichage du calendrier.

Input datetime

2011-06-2  00:01  UTC

Juin							2011
Lun	Mar	Mer	Jeu	Ven	Sam	Dim	
30	31	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	1	2	3	
4	5	6	7	8	9	10	

Aujourd'hui

Légende : Les datepickers HTML5 sont riches de possibilités mais n'apportent pas une solution satisfaisante d'un point de vue accessibilité. Une solution JavaScript bien pensée restera plus intéressante.

C'en est tout pour HTML5 ! Nous ne rentrerons pas davantage dans les détails. Les éléments présentés ci-dessus nous ont semblés les plus intéressants, soit parce qu'ils apportent véritablement un gain pour l'expérience utilisateur ou l'accessibilité, soit parce qu'ils interrogent l'ergonomie et les usages.

Lorsque vous décidez d'utiliser HTML5 pour la réalisation de votre site, gardez plus que jamais à l'esprit d'assurer la conformité avec les lecteurs d'écran. Pour cela, la spécification WAI-ARIA propose de définir les rôles, les états et les propriétés des *widgets* personnalisés dans le but qu'ils soient reconnaissables par les technologies d'assistance. Cela vous permettra d'attendre patiemment le support de cette version du langage.

II. Ergonomie et bonnes pratiques

Nous venons donc de définir les éléments constitutifs des formulaires. Concentrons-nous désormais sur notre leitmotiv : la complétion.

Avant tout, notez que les utilisateurs doivent identifier ces champs comme des espaces de saisie ou d'action. Il est ainsi recommandé de les mettre en exergue du fond de page. Nous privilégions un traitement blanc sur un fond de page coloré pour ces champs plutôt que « ton sur ton » ou colorés sur blanc, puisque c'est ainsi que les représentent les interfaces des systèmes d'exploitation et les navigateurs. Si la contrainte est trop forte, n'hésitez pas à utiliser les caractéristiques modifiables (couleur et épaisseur de la bordure, ombre portée).

Il n'est, en outre, pas utile de charger les formulaires de fioritures graphiques intrusives. Identification, simplicité et brièveté doivent résumer la tâche demandée à vos utilisateurs. Mais ceci ne signifie pas pour autant qu'ils doivent être austères : l'élégance est indissociable du plaisir et du confort d'usage.

1. L'objectif est la complétion

NOMMER ET INTRODUIRE LE FORMULAIRE

L'utilisateur doit savoir ce qu'on attend de lui, il ne doit pas le découvrir. Accompagner le formulaire d'un titre clair et d'une introduction rassurante permet de situer le contexte, de lui communiquer ce qu'on attend et ce qu'il obtiendra. C'est également l'occasion rêvée de lui faire part d'informations importantes comme l'identification du caractère obligatoire des champs.

LIMITER SA LONGUEUR

La longueur d'un formulaire est une caractéristique rédhibitoire. La saisie est un exercice cognitif dont l'internaute se passerait volontiers. Si vous constatez que votre formulaire ne transforme pas suffisamment, demandez-vous si votre scénario est approprié :

- une inscription est-elle indispensable pour poster un commentaire sur un message ;
- faut-il créer un compte pour acheter un produit sur votre site ;
- votre activité nécessite t-elle de récolter des informations qualifiant fortement vos internautes au risque de constater une diminution du nombre de soumissions (ex. cas des formulaires longs demandant des données relatives à l'intimité) ;
- pouvez-vous privilégier un formulaire réduit à son strict minimum, favorisant la transformation, quitte à demander un complément d'information plus tard (un formulaire d'inscription peut être réduit au choix de l'identifiant et du mot de passe, quitte à inciter l'utilisateur à compléter son profil plus tard) ?

Il n'y a pas de solution évidente, le choix est à considérer au cas par cas, suivant l'ensemble des contraintes et objectifs des différentes parties prenantes au projet (marketing, design, expérience utilisateur).

NE DEMANDER QUE LES CHAMPS INDISPENSABLES

La longueur du formulaire est intimement liée à la quantité d'information requise. Dans l'idéal, nous vous conseillons de demander uniquement celles allant dans le sens de la requête de l'internaute. Date de naissance, numéro de sécurité social ou numéro de fax pour un utilisateur venu acheter des chaussettes ne semblent pas forcément indispensables ! Ménagez vos visiteurs, réduisez vos formulaires au strict minimum.

LORSQU’UN CHAMP N’EST PAS ÉDITABLE, NE L’AFFICHEZ PAS !

Un élément non éditable est très souvent inutile. Il alourdit graphiquement la page et pourrait frustrer l'utilisateur souhaitant changer la valeur qu'il contient. Ne pas pouvoir modifier une information c'est susciter l'impression de dysfonctionnement, créant une rupture forte du processus de complétion et décourageant l'internaute.

AFFICHER PROGRESSIVEMENT LES CHAMPS DÉPENDANTS

Dans le cas de formulaires complexes ou pour un gain d'espace, il est possible de ne faire apparaître certains champs qu'à l'instant où d'autres sont remplis. On pense, par exemple, à l'apparition d'un champ « Nom de jeune fille » lorsque l'utilisateur spécifie une civilité « Madame ».

Comme toutes bonnes choses, il est recommandé de ne pas en abuser, à défaut de quoi l'internaute pourrait s'imaginer que chaque champ peut en cacher un autre.

TUNNEL À ÉTAPES SUCCESSIVES OU SYSTÈME À PAGE UNIQUE

Les tunnels de commande sont un bon exemple de formulaires complexes où la complétion est un enjeu clé. Ils se présentent sous deux formes : par *étapes successives* ou en une *page unique*.

Le format par étape est le premier à avoir été massivement utilisé par les sites d'e-commerce. Il a l'avantage de proposer une liste de champs restreinte, facilitant la gestion des erreurs. Le revers de la médaille vient à la fois de la lenteur de complétion que requiert la validation des étapes et de la difficulté d'appréhender la longueur totale de la tâche.

Certaines des analyses de *tracking* ont montré une diminution du nombre d'utilisateurs proportionnellement au nombre d'étapes. La pertinence du système repose donc sur l'équilibre entre la longueur de chacune des étapes et leur quantité. Les bonnes pratiques OPQUAST suggèrent d'être transparent sur ce nombre et de les rendre visibles⁹.

⁹ Bonne pratique n°37 : les processus complexes sont accompagnés de la liste de leurs étapes.

Bonne pratique n°38 : l'étape en cours d'un processus complexe indiquée. ([référentiel OPQUAST](#))

Le format en une seule page est plus récent. Il simule une étape unique, simple et rapide. L'appréhension du temps nécessaire à l'accomplissement de la tâche est relativement aisée. Lorsque son ergonomie est bien pensée, la complétion est rapide : l'enjeu est de savoir guider efficacement l'utilisateur en proposant un parcours intuitif que vous aurez imaginé pour lui, en adéquation avec une série de tests utilisateurs préalablement menés.

Dans ce type d'exercice, le designer est partagé entre la possibilité de composer une liste en pleine largeur de page, ou d'utiliser l'espace pour positionner côte-à-côte des étapes successives. Cette seconde disposition peut rompre avec le sens usuel de lecture et nécessite donc un effort important sur la signalétique. La navigation par la touche « tabulation » devra également suivre ce sens sous peine de perdre totalement l'utilisateur.

L'autre inconvénient vient de la gestion des dépendances entre champs nécessitant un rechargement de la page total ou partiel. Par exemple : la saisie du code postal détermine les modes de livraisons disponibles alors que ces étapes sont éloignées, non visibles simultanément à l'écran. Si le temps de chargement des modes de livraison est long, l'utilisateur pourra s'interroger sur la cause du ralentissement.

Ce second format se prête au jeu des utilisateurs plus coutumiers de l'usage des formulaires. On peut imaginer que l'essor du commerce électronique y est pour beaucoup. Nous constatons cependant que les cadors du genre (Fnac, Amazon, Cdiscount) procèdent toujours par étapes.

1 Coupon de réduction

Vous possédez un coupon? Entrez son code et validez si vous souhaitez l'utiliser.
Saisissez votre code

2 Mode de paiement

Paiement par carte bancaire

Paiement par PayPal

Paiement par carte privilège

3 Type de livraison

Colissimo (5,00 €)
> Vous serez livrés en 5...

Kiala (?) (Livraison gratuite)
> Vous serez livrés en 5... maximum dans un relais vous.

Chronopost (7,00 €)
> Vous serez livrés en e...

4 Adresse de livraison

5 Paiement sécurisé

Contenu de votre panier Spartoo

	1x OUS Taille: 27	48,00 €
		Sous total : 48,00 €
		Livraison : 5,00 €
		Total Spartoo : 53,00 €

1 Article Disponibilité Frais de livraison Quantité Prix total

Canon PowerShot A495 Argent	En Stock	6 €	1	55,00 €
-----------------------------	----------	-----	---	---------

Le Fnac vous recommande : Garantie 100% remboursé 2 ans 19 €

Sous-total : 55,00 €

Frais de livraison estimés : 6,00 €
Vous pourrez modifier ce choix par la suite

Code avantage ou bon de réduction (uniquement sur Fnac.com) : Saisissez le ici

REGROUPER LES CONTENUS SIMILAIRES

Ce pourrait être une évidence, ce devrait l'être, mais ce n'est malheureusement pas toujours le cas. Parfois, lorsqu'un formulaire est long, la solution consiste à l'aérer et à le segmenter.

En regroupant par thème certains champs et en leur accolant un titre de section, on rend l'ensemble plus digeste. Pourquoi ? Parce qu'il est alors plus simple pour l'utilisateur de parcourir les informations demandées.

Pour cela, il est utile d'avoir recours aux balises *fieldset* et *legend*. La première regroupe, d'un point de vue sémantique, tous les *input* qu'elle contient. La seconde ajoute un texte décrivant ce contenu. Cette légende doit être concise ! Son rôle est informatif, non descriptif. Que l'utilisateur soit ou non dépendant d'une technologie d'assistance, il souhaite aller à l'essentiel.

Pour chacun des champs, un lecteur d'écran annoncera la légende correspondante, le rythme de complétion peut donc se voir fortement ralenti et entraîner un découragement de l'utilisateur. Cela peut s'avérer très utile dans un contexte où plusieurs champs comportent le même intitulé sur une même page (ex. « ville » et « ville »), le premier peut être associé au *fieldset* « Départ » et l'autre au *fieldset* « Arrivée ».

Les *fieldset* comme les légendes acceptent certaines caractéristiques CSS (couleur de fond, contours, coins arrondis, ...) qu'il vous faudra tester car les comportements diffèrent en fonction des navigateurs.

PERMETTRE UNE NAVIGATION AU CLAVIER

La touche « tabulation » permet un déplacement d'élément en élément. Combinée aux touches « entrée » et/ou « espace » elle compose un système de navigation à part entière. Les technologies d'assistance peuvent en tirer profit, mais cela peut s'avérer utile en cas d'absence de moyens de navigation comme la souris ou le *trackpad*. Cette navigation est indispensable et ce, peu importe la manière dont sont agencés les champs de formulaire.

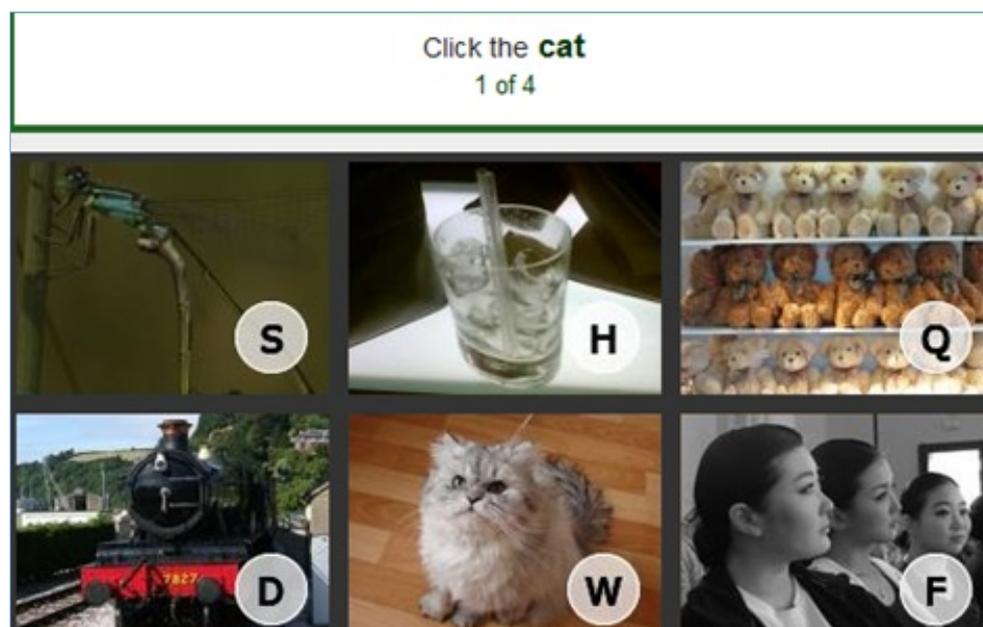
SAVOIR SE PASSER DE CAPTCHA (OU TESTS DE TURING)

Ce principe de filtre anti-spam, dont l'objectif est de garantir que les informations saisies sont bien l'œuvre d'un humain, pose de sérieux problèmes de compréhension aux internautes et pis, se montre de moins en moins efficace contre les robots.

Si, depuis leurs premières apparitions, ces systèmes de test ont évolué pour faciliter leur résolution par les utilisateurs (aide à la distinction des caractères 0 et O, chargement d'une autre série de caractères si celle affichée est trop complexe, traduction sonore pour les personnes malvoyantes), les robots ont suivi un parcours similaire pour finalement amener les *Captcha* à se complexifier davantage (réponse à une question, reconnaissance d'un élément commun). Aujourd'hui, ce sont entre 88 % et 100 % des tests de Turing qui peuvent être vaincus par les robots¹⁰.



Légende : Ci-dessus, exemples de tests de Turing utilisant une image dont les caractères sont très difficilement lisibles. Ci-contre, le CAPTCHA présente une énigme à laquelle il faut répondre en choisissant la bonne vignette.



¹⁰ Documentation du W3C au sujet des tests de turing (CAPTCHA) : <http://www.w3.org/TR/turingtest/>

En terme d'ergonomie, il nous semble évident que ces systèmes ne favorisent pas une complétion simple et rapide d'un formulaire. De plus, ils se montrent très souvent peu ou pas du tout accessibles et posent de fait une question fondamentale : est-ce à l'utilisateur de devoir prouver qu'il est humain ou est-ce à l'outil de garantir que les informations saisies sont bien l'œuvre d'un internaute ?

De notre point de vue, la réponse est limpide : l'utilisateur ne doit pas se préoccuper des problématiques qui ne le concerne pas lui-même.

Rassurez-vous, il existe de vraies solutions anti-spam qui ne se déploient pas du côté de l'interface et qui sont plus efficaces que les *Captcha*. Prenez par exemple [Akismet](#) qui gère le spam par contenu ou [Bouncer](#) qui identifie les connexions.

2. Aligner le couple label / champ

Le *label* se positionne avant le champ auquel il se rapporte pour préserver un sens logique de lecture. On constate trois manières d'aligner le couple :

- au dessus ;
- à droite ;
- à gauche.

Chacun de ces alignements possède des spécificités, avantages ou inconvénients. Le but du jeu est donc de déterminer celui qui sera le plus en adéquation avec votre contexte.

LABEL PLACÉ EN FER À GAUCHE

Le *label* et son champ sont sur une même ligne, favorisant une occupation verticale moindre. Cette disposition suit le sens de lecture, elle permet donc une attention particulière.

Il est nécessaire de travailler la longueur des intitulés de sorte que la différence entre le plus long et le plus court soit inférieure à dix caractères¹¹.

Ce type d'alignement est préconisé dans le cas de formulaires longs et complexes requérant une attention importante de l'utilisateur. Dans le cas de formulaires courts, on constate un temps de complétion plus long que les autres présentations. Les personnes malvoyantes grossissant fortement l'affichage pourraient également se retrouver avec des intitulés sans champ visible à l'écran. Lorsque l'espace entre le champ et son étiquette est trop important, il existe un risque que ce dernier passe à la ligne au-dessous ou au-dessus.

Légende : La différence de caractères entre le label le plus court (Nom) et le label le plus long (Adresse suite) dépasse le nombre recommandé de seulement deux caractères et présente déjà un blanc important entre les labels Nom et Ville et leurs champs.

Prénom *	<input type="text"/>
Nom *	<input type="text"/>
Adresse *	<input type="text"/>
Adresse (suite)	<input type="text"/>
Code postal *	<input type="text"/>
Ville *	<input type="text"/>

¹¹ Plus précisément de six à huit caractères selon Amélie Boucher, *Ergonomie web pour des sites web efficaces*, éd. Eyrolles 2007.

LABEL PLACÉ EN FER À DROITE

Le *label* et son champ sont sur une même ligne favorisant une occupation verticale moindre. Leur proximité favorise une association rapide et donc une complétion efficace pour peu que les *labels* présentent une longueur similaire ou presque (qu'ils soient familiers pour l'utilisateur). Dans le cas contraire, l'absence d'alignement gauche n'aide pas la lecture. Difficulté supplémentaire lorsqu'il s'agit d'un site multilingue où il devient compliqué de gérer la longueur des mots.

On l'utilisera plutôt dans le cas d'un formulaire court où le design est contraint verticalement et on prendra le soin de raccourcir, dans la mesure du possible, la longueur des intitulés.

Le diagramme illustre un formulaire web avec les éléments suivants alignés à droite :

- Prénom * [champ]
- Nom * [champ]
- Adresse * [champ]
- Adresse (suite) [champ]
- Code postal * [champ]
- Ville * [champ]

Légende : Avec les mêmes labels et champs l'alignement à droite se comporte mieux en terme de lisibilité. Les informations demandées étant très communes, l'effort de lecture provoqué par le non alignement à gauche est moindre.

LABEL PLACÉ AU DESSUS

Le *label* et le champ sont disposés l'un au dessus de l'autre impliquant une occupation verticale importante. Leur proximité favorise une lecture rapide et donc une complétion efficace.

La gestion de l'espace est primordiale pour associer les couples, on empruntera à la typographie sa gestion des espaces :

- entre le *label* n°1 et son champ, on appliquera un espace simple ;
- entre le champ n°1 et le *label* n°2, on appliquera un espace double.

Lorsque les données sont familières, des tests utilisateurs ont permis de mettre en évidence que ce mode de disposition est celui pour lequel le temps de complétion est le plus rapide.

New Account

Creating an account at MyFonts allows you to customize how you purchase fonts, and re-download your orders at any time in the f

Your name

Email address

Password

Password again

Country

We will e-mail you to confirm that your account has been created.

Create Account

Légende : Formulaire d'inscription du site [MyFonts](#), peu de champs, une bonne gestion des espaces permettant de raccrocher label et champ. Seul regret, la taille des polices de caractères est particulièrement petite (cela est d'autant plus vrai pour le choix du pays).

3. Longueur de champ et « affordance »

L'affordance est la capacité d'un objet à suggérer ce à quoi il est utile et d'induire sa manipulation sans avoir recours à une explication. Dans le cas des formulaires, l'action attendue est leur complétion, mais cela ne caractérise pas les informations attendues (alphabétiques, numériques ou symboliques), ni leur quantité. Pour cela, nous pouvons utiliser des indices issus de la culture de la population ciblée. En France, nous constatons :

- qu'un code postal possède cinq chiffres ;
- un numéro de téléphone dix chiffres.

Attention, toutefois, de vérifier que ces informations sont vraies pour votre population cible. Si vous demandez un numéro de téléphone portable, n'imposez pas systématiquement un 06 alors que de plus en plus les numéros commencent par 07.

Nous constatons facilement que ces champs n'attendent pas la même longueur d'information, et nous pouvons déduire que celui du téléphone devrait être deux fois plus grand que celui du code postal. De même, si un nombre escompté est strictement inférieur à 100, ne permettez pas la saisie de trois chiffres et n'autorisez pas l'internaute à penser qu'il pourrait en ajouter davantage. Ou encore, lors de la saisie d'un mot de passe (que vous n'oublierez pas de doubler pour la vérification s'il s'agit d'une création ou d'une modification – bonne pratique OPQUAST n°32¹²), faites en sorte que les caractères ne soient pas lisibles.

En tant que designer, nous nous retrouvons face à une situation délicate : les champs ne présentent pas une longueur homogène et créent des vides irréguliers, disgracieux. Malheureusement, il n'existe pas de solution absolue. Afin de modérer cet effet, le regroupement des champs par thème peut atténuer cette impression. Il vous reste plus qu'à jouer de votre ingéniosité graphique pour trouver la solution la mieux adaptée à votre contexte.

¹² Bonne pratique n°32 : La création d'un mot de passe par l'utilisateur est validée par une double saisie ([référentiel Opquast](#)).

LE CAS PARTICULIER DES SITES MULTILINGUES

S'agissant d'une information culturelle qui peut différer d'un pays à l'autre, vous devez être en mesure d'afficher les champs à la bonne longueur, en utilisant le paramètre de langue sélectionné, d'afficher les champs à la bonne longueur. Vous pouvez également en profiter pour pré-remplir le champ avec l'indicatif du pays dont la suppression sera permise si votre internaute est expatrié et non résident.

Les bonnes pratiques OPQUAST préconisent l'uniformisation par l'adoption de la norme internationale¹³. Néanmoins, cette pratique est contraire au principe selon lequel il faut laisser à l'utilisateur le choix de saisir les informations comme il l'entend.

¹³ Bonne pratique n° 51 : Chaque numéro de téléphone est indiqué au format international ([référentiel Opquast](#)).

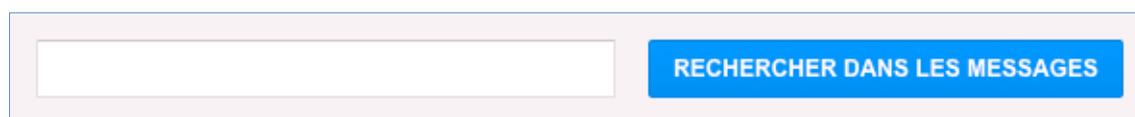
4. Hiérarchisation des actions

Une bonne hiérarchisation favorise la lisibilité et la compréhension de la priorité des actions attendues. Nous distinguons trois niveaux d'action.

LES ACTIONS PRIMAIRES

Les boutons d'action primaire permettent la progression : « Continuer », « Sauvegarder », « Envoyer »... Ils doivent se présenter à leur avantage, et être particulièrement visibles. Les tests de rapidité nous indiquent qu'il est préférable de les aligner avec les champs du formulaire.

Surtout, n'oubliez pas qu'ils sont indispensables ! Un formulaire sans bouton d'action primaire n'a aucune chance d'être transmis, il est nécessaire dès l'ajout d'un champ (ex : une recherche nécessite un bouton de validation). Un lien ne permet pas cette validation, d'où la nécessité d'utiliser la balise *button* ou un *input* de soumission.



Légende : La recherche est un formulaire ! Elle nécessite donc un champ et un bouton d'action, à minima, et avec un label c'est encore mieux !

LES ACTIONS SECONDAIRES

On entend par bouton d'action secondaire les liens du type « Annuler », « Effacer » ou « Retour ». Ils vont dans le sens inverse de la marche attendue et souhaitée du processus de complétion. Ils sont néanmoins importants car ils offrent la possibilité d'un retour en arrière pour l'utilisateur.

Remarquez que les ergonomes utilisent un abus de langage en affectant à ces liens le nom de « bouton ». En effet, il n'est pas souhaitable que ces actions prennent l'apparence d'un bouton lorsque les styles CSS sont désactivés. Ils seraient alors identiques aux actions primaires. En intégration, privilégiez l'utilisation de lien hypertexte que vous habillerez à la convenance de la direction artistique, d'autant que ces actions n'entraînent pas une soumission du formulaire.

Dans un formulaire, l'internaute doit pouvoir identifier ce que l'on attend de lui. Bien hiérarchiser ces actions c'est lui indiquer la bonne marche à suivre pour arriver au terme de la complétion, une manière de baliser sa route.

Dans la mesure du possible, on limitera l'usage des actions secondaires au seul retour en arrière. L'annulation s'obtient simplement en quittant la page en cours, et vouloir effacer ce que l'on vient de saisir est tout simplement contre-productif car il est peu probable que l'utilisateur se trompe sur le remplissage de l'ensemble des champs (ou alors le problème ne vient peut-être pas de lui).

Il est commun de retrouver le(s) action(s) secondaire(s) à droite de l'action primaire. Le sens de lecture participe également à leur hiérarchisation.

LES ACTIONS TIERCES

Il s'agit des actions ne permettant ni un retour en arrière, ni une progression dans le processus de complétion. On les retrouve au cœur même des formulaires par des liens tels que « modifier », « annuler » où ils suggèrent l'action avec discrétion et efficacité car ils ne peuvent perturber la hiérarchie générale du formulaire.

Il est alors nécessaire de les présenter différemment, en fonction du contexte.

Marque & modèle	Citroen C2 1.4 HDI PACK AMBIANCE	Modifier
	<ul style="list-style-type: none"> • P.Fiscale 4 • Berline 3 portes • Gazoil 	
Etat du véhicule	<input type="radio"/> Neuf <input checked="" type="radio"/> Occasion	<input type="text" value="2008"/> Aide
<input type="button" value="Revenir à l'étape 1"/>		<input type="button" value="Accéder à l'étape 2 : Conducteur"/>

Légende : ici, on distingue trois niveau hiérarchiques d'actions : primaire (accéder à l'étape 2), secondaire (revenir à l'étape 1) et tierce (Modifier / aide).

Les liens « **Modifier** » et « **Aide** » sont ici présentés comme des actions tierces. Leur design et donc leur affordance sont différents des actions primaires et secondaires. Il convient de les rendre visible au sein du texte sans leur donner une importance démesurée.

LE POSITIONNEMENT DES BOUTONS DE VALIDATION

Un bouton de validation conclut la complétion d'une page de formulaire, il est donc normalement attendu après le dernier champ. Évitez de positionner des options au-dessous car vos utilisateurs risqueraient de passer à côté et cela est d'autant plus vrai pour les personnes aveugles.

Légende : mappy.fr (dans son ancienne version) et leboncoin.fr où des actions sont proposées au-dessous du bouton d'action

5. Indiquer le caractère obligatoire (ou non) des champs

Un champ peut ne pas être obligatoire tout en se révélant hautement important ! Prenons le cas d'une adresse postale. La norme française veut que l'on permette d'indiquer une adresse et un complément d'adresse pour les résidents d'immeubles (résidences des « Lilas », bâtiment 3, escalier B, 4ème étage, porte face) quand d'autres personnes jouissent d'une adresse réduite à son plus simple appareil (un simple lieux-dit par exemple). Cette disparité ne doit pas avoir d'incidence sur la possibilité de remplir notre formulaire. C'est pourquoi ce champ « complément d'adresse » ne peut revêtir un caractère obligatoire mais ne peut pas, pour autant, être supprimé.

En conséquence, voici les éléments auxquels vous devez prêter attention :

- Préciser, par une légende, comment seront identifiés vos champs obligatoires et ce, avant le formulaire (suivant l'ordre logique de lecture). Veuillez rendre cette indication visible (ne pas la noyer dans un paragraphe d'introduction par exemple) ;
- ajouter un astérisque « * » (sans doute l'un des symboles plus coutumiers des internautes) ou un autre caractère différenciant à la fin du *label* ;
- lui appliquer une autre couleur ou une autre graisse de caractère que celles des intitulés (attention, ces caractéristiques ne peuvent être utilisées comme seuls éléments différenciant, l'information serait perdue pour les personnes aveugles ou utilisant des contrastes particuliers).

REGROUPER AUTANT QUE POSSIBLE LES CHAMPS OBLIGATOIRES.

Quand tous les champs sont obligatoires, il semble qu'il ne soit pas nécessaire d'ajouter un astérisque. La légende « *merci de compléter l'ensemble des champs* », en tête de formulaire, doit suffire et évite de surcharger la page en éléments graphiques. Cependant, une attention particulière doit être portée à la visibilité de cette légende car l'utilisateur distrait peut facilement passer à côté.

Il est également possible d'indiquer le caractère « *optionnel* » de ceux qui le sont dans la mesure où ils sont **peu nombreux**. Cependant, ce choix est discutable car la distinction obligatoire / non obligatoire peut apparaître tardivement si l'utilisateur n'a pas prêté attention à notre phrase d'introduction. Notons également que les technologies d'assistances requièrent l'information. Si vous faites le choix de ne pas mentionner les champs obligatoires, graphiquement parlant, utilisez *Aria-required="true"* qui permettra aux lecteurs d'écran d'indiquer qu'une information est néanmoins attendue.

Le caractère facultatif est, ici, délicat à cerner. D'une part, la phrase d'introduction, peu valorisée, peut ne pas être lue. D'autre part, la mention « facultatif » se retrouve coincée entre les *labels* « Téléphone » et « Ville » sans qu'il soit évident de savoir auquel il se rattache.

Une question, un commentaire, nous sommes à votre écoute

Merci de renseigner l'ensemble des champs, hormis ceux indiqués comme facultatifs.

Nom

Prénom

Téléphone (facultatif)

Ville

E-mail

Une question, un commentaire, nous sommes à votre écoute

* Les champs précédés d'un astérisque sont obligatoires

* Nom

* Prénom

Téléphone

Ville

* E-mail

Légende : L'indication (facultatif) est ici mal positionnée et se montre confuse quant au label concerné.

Légende : Les astérisques sont placés avant les labels. On peut supposer que le caractère obligatoire est la première chose perçue, mais il est possible que l'utilisateur identifie l'astérisque comme une puce et ne considère pas l'information.

Il arrive aussi que les formulaires soient sans indication. Ce choix est valable lorsque tous les champs sont au même niveau d'obligation, particulièrement pour les formulaires très courts : les formulaires de connexion (deux champs : identifiant / mot de passe) en sont un bon exemple.

En cas de doute, la solution la moins risquée semble être d'indiquer le caractère obligatoire par un astérisque. Les champs optionnels n'ont pas le besoin d'être indiqués comme tels.

6. Gérer le feedback utilisateur

Seul face à leur écran, les utilisateurs ont besoin d'être accompagnés en cas de doute ou d'erreur. N'étant pas en mesure d'assurer une présence physique derrière chacun d'entre eux, il nous reste la possibilité de les informer, de les aider, de les rassurer pour chacune de leurs actions.

LA VALIDATION À LA VOLÉE

Comme tout élément graphique susceptible de complexifier l'apparence d'un formulaire, elle doit être utilisée pour les champs dont le risque d'erreur est plus important. Elle nous semble judicieuse dans des cas comme la vérification de disponibilité d'un pseudonyme, le niveau de sécurité d'un mot de passe ou encore la conformité du second champ de contrôle (email, mot de passe).

Elle ne nous semble pas du tout pertinente pour les nom ou prénom. Dans ces cas là, c'est la validation lors de l'envoi global du formulaire qui retournera l'erreur.

La validation à la volée du champ ne doit avoir lieu que si ce dernier est complété, et uniquement lorsque l'on passe au champ suivant.

Elle doit également présenter une certaine tolérance d'écriture. Un numéro de téléphone, par exemple, peut être renseigné de diverses façons :

- 0123456789 ;
- 01 23 45 67 89 ;
- 01.23.45.67.89 ;
- +33(1) 23 45 67 89 ;
- 012 345 789...

Cette tolérance n'est parfois pas possible, pour des raisons liées à la base de données notamment. Dans ce cas là, une aide contextuelle doit être apportée au préalable, afin que l'utilisateur comprenne ce que l'on attend de lui. Dans l'idéal, mieux vaut lui laisser toute latitude à ce sujet et traiter son information par la suite, à votre convenance.

Néanmoins, si cette aide doit être **visible avant** que l'utilisateur ne débute sa saisie, il sera d'usage de la retrouver plutôt à la suite, en dessous, à l'intérieur du champ via un *placeholder* ou au niveau du *label*.

Cette dernière solution est intéressante car elle fournit l'information dans le sens de lecture, avant complétion du champ. En revanche, elle peut augmenter la taille du *label* et détériorer la lisibilité des alignements *label / champ* (cf. *Aligner le couple label / champ*, chap. II.2 page 28).

Le *placeholder* pourrait être une bonne alternative mais il nous confronte à divers problèmes d'accessibilité et d'ergonomie (contraste, impression de champ déjà rempli... cf. *HTML5 apports et limites*, chap. I.3 page 15).

Nous ne retenons le positionnement en dessous du champ que dans le cas d'une contrainte d'espace très forte car l'information n'apparaît pas dans le sens de lecture. Quant à l'aide située après le champ, elle peut obliger l'utilisateur à effacer ce qu'il vient de saisir et nécessite d'attirer l'attention.

Step 2: Pick a username and a password for your account

USERNAME: Your username may contain letters, numbers, underscores and spaces. Will be used in your Wishlitr url.

PASSWORD: Your password must be at least 3 characters long. Spaces are not allowed.

Password *
Like a secret agent.

At least 6 characters long.

Once more to verify, please.

Date of birth:

Credit Card:

Légende : Dans ce 3^{ème} écran, les lecteurs d'écran risquent d'ignorer purement et simplement les caractères #. En l'absence d'une aide autre que le placeholder, l'utilisateur n'aura pas accès au format attendu.

Il n'y a pas de solution idéale, nous préconisons de privilégier l'aide au niveau du *label* à l'instar des bonnes pratiques OPQUAST¹⁴. Lorsque l'on attend un nombre limité de caractères, on peut indiquer en temps réel la quantité de signes restants. Si l'on impose un minimum, la démarche est similaire : il suffit d'indiquer la quantité à atteindre et la quantité actuelle. Dans le cas où cette valeur doit être comprise entre X et Y, une solution mixant les deux cas précédents peut facilement être trouvée.

Quoi de neuf ?

Nous recherchons un Chef de Projets Web en CDI dans l'univers du luxe. Rejoignez-nous ! <http://xa.vc/75684> #emploi #job #paris

Le lien sera raccourci **13**

Légende : La limite de caractère étant une caractéristique fondamentale du format de communication de Tweeter, l'indication du nombre de signe restant est primordiale.

¹⁴ Bonne pratique n°119 : L'étiquette de chaque champ de formulaire indique, le cas échéant, quel format de saisie doit être respecté.

LES MESSAGES D'ERREURS

En dehors de la validation à la volée, toutes les erreurs doivent être traitées en une seule fois, ce qui permet à l'utilisateur de procéder à une correction globale. Pour ce faire, l'apparence des messages d'erreur doit attirer l'attention (modification du corps de police, des couleurs, de la bordure, utilisation d'une icône). On préconise de placer un message indiquant d'une manière générale la présence d'erreurs au-dessus du formulaire, accompagné de messages contextuels au niveau des champs erronés.

Le champ et son message doivent être visuellement liés (proximité, couleur). Confronter l'utilisateur à une erreur n'est pas satisfaisant en soi, il est nécessaire de l'accompagner dans sa résolution. Un rédactionnel concis est à privilégier : « Vous avez effectué telle chose, vous devriez faire telle autre ».

Il est important de ne pas effacer les informations erronées, au risque que l'internaute ne sache plus quelle erreur il a commis. Et naturellement, n'effacez pas les champs valides !

À noter que la validation des champs via les mécanismes de HTML5 ne permettent pas encore ni de modifier ni d'agir sur le design des messages d'erreurs.

L'INDICATION DU PROCESSUS EN COURS

Il est nécessaire d'indiquer à l'utilisateur qu'une action est en cours lorsque celle-ci requiert un temps d'exécution long (soumission du formulaire, calcul de données, *upload* de document). Pour cela, utilisez un procédé graphique simulant l'exécution en cours (un pourcentage, une barre de progression, trois points qui s'animent, une roue...).

Il s'agit du seul cas où l'on tolère de désactiver le bouton de soumission du formulaire pour éviter que les envois soient doublés. Cette solution a le mérite d'être visible pour l'internaute pour peu que le message accompagnant l'animation de chargement soit clair et précis.

Afin de vous prémunir de tout envoi doublé, n'omettez pas de mettre en place des solutions côté serveur (via les « jetons » ou « *token* » qui invalident le formulaire en cas de renvoi) et côté client via *JavaScript* (qui ajoute un « *disabled* » sur le bouton de soumission après la validation, en tenant compte que cette solution ne fonctionne pas lorsque *JavaScript* est désactivé).

Une fois votre formulaire complété et envoyé, communiquez clairement sur la réussite de l'envoi. L'utilisateur a besoin de savoir si l'effort fourni est couronné de succès. Si l'envoi d'un email de confirmation est nécessaire, ne vous en contentez pas. Un message de confirmation à la suite de la bonne complétion du formulaire annonçant en plus l'envoi de cet email est plus approprié.

Concluons !

C'est un fait : vos formulaires sont désormais, sinon plus beaux, au moins plus accessibles et font preuve d'une ergonomie certaine. Si tout ceci est encourageant, n'oubliez pas que rien n'est jamais acquis !

Nouveaux utilisateurs, nouveaux supports et nouveaux usages remettent en question votre travail, et c'est tant mieux. Cet enthousiasme tient du fait que l'idée de découvrir chaque jour de nouvelles idées, de nouvelles expériences où les différences se jouent au détail près, est un enrichissement permanent.

Il existe ainsi des formulaires qui « racontent » des histoires ! Ces formulaires narratifs ont la particularité d'inclure *label* et *champ de saisie* au cœur d'un texte intelligible. Ils sont plutôt rares et les retours d'expériences sur leur succès le sont tout autant. S'ils correspondent à votre ligne éditoriale : foncez, testez !

BEFORE	AFTER
<p>Contact This Dealer</p> <p>> Infiniti of Coconut Creek 800-577-7300</p> <p>> See All Dealer Inventory</p> <p>First Name: <input type="text"/></p> <p>Last Name: <input type="text"/></p> <p>Street Address: <input type="text"/></p> <p>ZIP Code: <input type="text"/></p> <p>Email: <input type="text"/></p> <p>Phone: <input type="text"/> <input type="text"/> <input type="text"/></p> <p>Comments: <input type="text"/></p> <p><input checked="" type="checkbox"/> Yes, I'm interested in receiving news and special offers from Kelley Blue Book.</p> <p>SEND</p> <p>> Privacy Policy</p> <p>> Get your CARFAX report for this GMC</p> <p>> Calculate your monthly payment</p> <p>> Get your credit score now</p> <p>> Get a free insurance quote for this GMC</p>	<p><input checked="" type="checkbox"/> I'm Interested</p> <p>To:  Lehmer's Buick Pontiac GMC 1-866-607-2809 ... More Info</p> <p>From: <input type="text" value="my email address"/></p> <p>Your message:</p> <p>Hello, my name is <input type="text" value="first name"/> <input type="text" value="last name"/> and I'm writing you today to learn more about the 2009 CHEVROLET SILVERADO 1500 LT listed for \$20,995. I live at <input type="text" value="my street address (optional)"/> in the <input type="text" value="ZIP"/> area and I would like to hear back from you soon and learn more about this vehicle. Please call me back on <input type="text" value="my phone number"/> at your earliest convenience.</p> <p>personalize this message</p> <p>Thank you.</p> <p><input checked="" type="checkbox"/> Yes, I'm interested in receiving news and special offers from Kelley Blue Book.</p> <p>SEND THIS MESSAGE</p> <p>> Privacy Policy</p> <p>> View the free CARFAX Report</p> <p>> Calculate your monthly payment</p> <p>> Get your credit score now</p> <p>> Get a free Progressive insurance quote</p>

Légende : Transformation narrative d'un formulaire de contact. Voir l'expérience menée par [Luke Wroblewski](#).

The image shows three sequential screenshots of a web form titled "Sign In to Bagcheck".

- Top screenshot:** The form has the title "Sign In to Bagcheck" and a label "Enter Your Name (or email)". Below it is an empty text input field. A link "Join Now" is visible below the input field.
- Middle screenshot:** The same form is shown, but the input field now contains the text "Luk". A dropdown menu of suggestions is displayed below the input field, listing names with small profile pictures: "Luke Wroblewski", "Luke Dorny", "Luke Brooker", "Luke Donovan", "Lukas Van Driel", "Jenn Lukas", and "Luke Holder".
- Bottom screenshot:** The form is shown with the input field containing "Luke Wroblewski". Below the input field, it says "You can sign in with: Twitter, Bagcheck." There are two buttons: "TWITTER" with a Twitter icon and "BAGCHECK" with a Bagcheck icon.

Légende : Une expérimentation a été menée par Luke Wroblewski (oui, encore lui !) sur Bagcheck pour simplifier la connexion / inscription sur un site.

À la saisie de votre nom, un moteur de recherche suggestif vous permet d'une façon ludique de retrouver votre identifiant.

À la suite de cette identification, il vous permet de choisir votre méthode de connexion en fonction des informations qu'il possède. Ce processus offre la possibilité de supprimer la notion d'inscription à un service.

On peut, cependant, reprocher à ce système de « divulguer » aux yeux de tous la création d'un compte pour ce service. Une écoute attentive des réactions de vos utilisateurs face au respect de la vie privée est alors fortement recommandée.

[Lire l'expérimentation.](#)

Il existe également des processus de connexion qui vous indiquent facilement si vous avez déjà un compte sur le site Internet, sans utiliser de message d'erreur et sans avoir à chercher quelle est l'adresse email utilisée. En adaptant le système des champs de recherche suggestifs, il est aisé de vous indiquer si votre identification passe par un service tiers (facebook, twitter) ou si vous avez utilisé le processus classique. Pas convaincu ? Pensez aux tests A/B ¹⁵!

¹⁵ Les tests A/B consistent à segmenter votre population de visiteurs en deux lots qui observeront chacun une version A ou une version B de votre site, d'une bannière, d'un module. Vous pourrez alors mesurer la transformation qu'offre chacune des solutions et opter pour celle qui vous convainc.

C'est un détail, mais lorsque vous saisissez un texte un peu long dans un champ de saisie et que celui-ci est trop court, votre saisie n'est pas visible dans son ensemble, ce qui peut être une source d'erreur. Envisagez d'adapter le corps de police à la longueur de votre texte pour que l'ensemble reste apparent. Une seule limite, le seuil de lisibilité des caractères !

The image displays four examples of login forms, each with a blue background. Each form consists of three input fields: an email field, a password field labeled 'Mot de passe', and a yellow 'Se connecter' button. The email fields contain text of varying lengths to illustrate how the font size in the input field adapts to the text length. In the first example, the text 'exemple@me.com' is short, and the font is small. In the second, 'ncatherin@email.com' is longer, and the font is larger. In the third, 'ncatherin@clever-age.com' is even longer, and the font is larger still. In the fourth, 'nicolas.catherin@clever-age.com' is the longest, and the font is the largest, ensuring the text is clearly visible within the field's boundaries.

Légende : Exemple de l'adaptabilité de la longueur de l'information saisie dans un champ sur [Mobile me](#).

Ces exemples n'ont qu'un seul objectif : celui de vous ouvrir à l'expérimentation réfléchie. Les bases sont connues, attachez-vous aux détails !

Webographie

- Référentiel OPQUAST spécifique aux formulaires
 - + <https://checklists.opquast.com/opquastv2?q=formulaires>
- CSS 3 Aujourd'hui, conférence donnée à Paris Web 2010
 - + <http://jeremie.patonnier.net/post/2011/02/16/Paris-Web-2010-CSS3-Aujourd-hui>
- Liste des input
 - + <http://dev.w3.org/html5/markup/input.html>
- Recommandations ARIA
 - + <http://www.w3.org/TR/wai-aria/>
- Travaux de 456 Berea Street
 - + http://www.456bereastreet.com/archive/200409/styling_form_controls/
- Travaux du Filament Group
 - + http://www.filamentgroup.com/lab/accessible_custom_designed_checkbox_radio_button_inputs_styled_css_jquery/
- Recherches, expérimentations, réflexion de Monsieur formulaires (entre autres !) : Luke Wroblewski
 - + <http://www.lukew.com/>
- Accessibilité de HTML5
 - + <http://html5accessibility.com/tests/placeholder-labelling.html>
- Pour se passer de CAPTCHA
 - + <http://akismet.com/>
 - + <http://h6e.net/bouncer/>

Bibliographie

- [Ergonomie Web, pour des sites web efficaces](#)
Amélie Boucher, Eyrolles, novembre 2011
- [HTML5 for Web Designers](#)
Jeremy Keith, Eyrolles Collection A Book Apart, septembre 2010
- [Don't Make Me Think, a common sens approach to web usability](#)
Steve Krug, New Riders, 2005 (2nd Édition)
- [Web Form Design, Filling the blanks](#)
Luke Wroblewski, Rosenfeld Media, mai 2008