Un formulaire de contact qui s’adapte

La découverte conditionnelle des champs — principes et mise en œuvre

En bref

- Un formulaire de contact utile se comporte comme une conversation : il oriente, ne montre que ce qui est pertinent, valide en douceur et ne bloque jamais.
- Ce document explique la découverte conditionnelle (progressive disclosure) et son application concrète à un formulaire de contact professionnel.

1. Le principe : la découverte conditionnelle

Introduite par Jakob Nielsen en 1995, la progressive disclosure consiste à n’afficher d’emblée que l’essentiel et à différer les options secondaires jusqu’à ce qu’elles deviennent pertinentes. L’objectif : réduire la charge cognitive — le visiteur ne traite qu’une chose à la fois. Appliqué à un formulaire, le principe devient : faire apparaître un champ seulement après le choix qui le rend nécessaire.

- On évite le « mur de champs » qui décourage et multiplie les erreurs.
- On garde une interface simple pour le cas général, sans priver les cas particuliers des champs dont ils ont besoin.
- On limite à deux le nombre de niveaux de révélation : au-delà, l’utilisateur se perd (recommandation NN/g).

2. La carte des champs conditionnels

Quatre mécanismes de révélation se combinent, du plus général au plus fin :

Déclencheur	Ce qui apparaît	Intention
Pays (orientation)	le bon interlocuteur / le formulaire local	ne pas faire remplir un formulaire qui n’est pas le sien
Objet (modèle / nature)	précision, n° de série, description du problème	ne demander les détails que s’ils ont un sens
Qualité de l’identification	formulaire court ↔ complet	court pour un pro identifié, complet sinon — sans rejet
Préférence de réponse	bloc « échange » (créneau, moyen)	n’afficher les modalités que si l’utilisateur les demande

Règle d’hygiène : tout champ conditionnel qui redevient masqué est vidé avant l’envoi — aucune donnée parasite ne part. Et la logique est pilotée par des données (un indicateur « précision requise », « n° de série nécessaire »…), pas codée en dur : on fait évoluer le comportement en éditant une table.

Exemple de parcours

Un client professionnel français ouvre le formulaire :

- 1. Le pays est déjà « France » : le formulaire local s’affiche en version courte.
- 2. Il saisit une adresse d’entreprise et un identifiant légal reconnu : deux drapeaux verts, rien de plus à remplir.
- 3. Il choisit un modèle puis la nature « demande de devis » : le champ « précision » apparaît, le numéro de série reste masqué (inutile ici).
- 4. Il rédige son message ; le compteur passe au vert au-delà du minimum, et le bouton « Envoyer » s’active.

À aucun moment il n’a vu un champ inutile. Un particulier, ou un identifiant douteux, aurait au contraire vu apparaître les champs « société » et « coordonnées » — toujours sans blocage.

3. Valider sans bloquer

Chaque champ sensible porte un retour visuel immédiat, doublé d’un texte (jamais la couleur seule) :

État	Signification	Effet
✓ vert	format et clé de contrôle reconnus	aucun ; le parcours court continue

État	Signification	Effet
⚠ orange	plausible mais douteux (faute de frappe, cas à vérifier)	escalade douce vers le formulaire complet
× rouge	format non conforme	escalade ; l'utilisateur corrige ou complète

Le point capital : un doute n'est jamais un rejet. Il déclenche une escalade (on demande un peu plus de contexte), avec la possibilité de revenir au formulaire court dès que l'identifiant redevient valide.
Exemple d'heuristique légère côté navigateur, non bloquante :

```
// un identifiant valide mais "rare" est signale, pas rejete
if (identifiantValide(v) && estSuspect(v)) {
etat = "a_verifier"; // orange, escalade
}
```

4. L’activation de l’envoi

- Le bouton Envoyer ne s’active que lorsque les champs critiques sont suffisants (objet, message assez long, consentement, identification ou coordonnées selon le mode).
- S’il reste grisé, un message indique précisément ce qui manque plutôt que de laisser deviner — c’est la différence entre un formulaire frustrant et un formulaire coopératif.
- Un lien e-mail de secours pré-rempli reste toujours visible : si quoi que ce soit empêche l’envoi, le contact n’est jamais perdu.

Le fil conducteur : ne jamais bloquer

- Orientation, validation, escalade : chaque mécanisme guide sans fermer la porte.
- Un identifiant douteux, une adresse grand public, un champ oublié → on demande mieux, on ne refuse pas.

À éviter

- Révéler puis re-masquer des champs de façon instable (effet « clignotement ») : déclencher sur un choix explicite, pas à chaque frappe.
- Empêcher l’envoi sans dire pourquoi : toujours afficher ce qui manque.
- S’appuyer sur la seule couleur pour signaler une erreur.
- Multiplier les niveaux de révélation : deux suffisent presque toujours.

5. Accessibilité et anti-spam

- Champs obligatoires marqués visuellement et programmatiquement (`required`, attributs `aria`) ; les champs simplement recommandés reçoivent un marqueur distinct, avec explication.
- Le retour de validation ne repose jamais sur la seule couleur : un texte et un `aria-label` l’accompagnent (utilisateurs de lecteurs d’écran, daltoniens).
- Un honeypot (champ invisible que seuls les robots remplissent) filtre le spam sans imposer de captcha, tant que le trafic reste modéré.

Microcopy : des étiquettes qui parlent

- Une étiquette dit ce qu’on attend (« N° de TVA ou SIREN ») ; une aide brève dit le format ou la raison.
- Distinguer clairement obligatoire, recommandé et facultatif : un champ recommandé n’est pas obligatoire, mais on explique l’intérêt de le remplir.
- Formuler la validation comme une aide (« vérifiez ce numéro ») plutôt qu’un reproche (« numéro invalide »).

6. Une architecture simple

Le formulaire est volontairement découplé du traitement :

- Un formulaire statique (HTML + un peu de JavaScript) porte toute la logique de révélation et de validation. Rapide à servir, facile à mettre en cache.
- Un point d’entrée côté serveur reçoit l’envoi, re-vérifie l’essentiel, enregistre la demande et notifie par e-mail. Les contrôles visuels du client sont une aide à la saisie ; les décisions importantes sont revalidées côté serveur.
- Les référentiels (pays, modèles, natures…) sont des données externes chargées par le formulaire : on met à jour le comportement sans toucher au code.

Ce découplage a un autre mérite : le formulaire reste léger et témoigne d’une intention claire — la logique est là où l’utilisateur interagit, le traitement là où les données doivent être sécurisées.

7. Bilan & références

La découverte conditionnelle transforme un questionnaire en dialogue : on oriente, on ne dévoile que le nécessaire, on valide en douceur, on ne bloque jamais. C’est autant une affaire d’ergonomie que de respect du visiteur.

- Nielsen Norman Group — Progressive Disclosure : nngroup.com/articles/progressive-disclosure/

- Nielsen Norman Group — Reduce Cognitive Load in Forms : nngroup.com/articles/4-principles-reduce-cognitive-load/
- GOV.UK Design System — patrons de formulaires (« one thing per page ») : design-system.service.gov.uk/patterns/
- MDN Web Docs — ARIA & accessibilité des formulaires : developer.mozilla.org