Engineer

# Amélioration progressive

Une base HTML fonctionnelle, enrichie ensuite par couches

## Ce que c’est

L’amélioration progressive est une façon de construire les pages web par couches, en commençant par celle qui fonctionne toujours. La base est un document HTML qui porte tout le contenu et permet à lui seul d’accomplir chaque tâche : les liens mènent quelque part, les formulaires sont envoyés à un serveur, et la page peut être lue et utilisée sans rien charger d’autre. Les feuilles de style s’ajoutent pour organiser la page et la rendre agréable à regarder. Les scripts viennent en dernier pour accélérer ou fluidifier son utilisation. Chaque couche améliore celle du dessous, et la page ne dépend d’aucune couche au-dessus du HTML pour remplir sa fonction. Le GOV.UK Service Manual résume l’idée ainsi : faire d’abord fonctionner la page en HTML.

Le nom vient d’une conférence de Steven Champeon et Nick Finck, *Conception web inclusive pour l’avenir grâce à l’amélioration progressive*, que leur article situe au 11 mars 2003, à SXSW, à Austin au Texas. L’article écrit le terme avec des majuscules, Progressive Enhancement, et l’abrège en PE. Il l’oppose à la dégradation élégante, pratique plus ancienne consistant à construire pour le navigateur le plus performant en espérant que les autres puissent tout de même accéder au contenu. Telle que l’article la décrit, PE part d’un document utilisable par les appareils les moins capables, puis ajoute la présentation sous forme de logique distincte, pour ne pas alourdir ces appareils avec ce qui est destiné aux navigateurs modernes.

La différence tient au sens dans lequel on travaille. La dégradation élégante part de l’expérience complète et demande ce qu’il en reste quand certains éléments manquent. L’amélioration progressive part de ce qui est toujours disponible et demande ce que l’on peut ajouter. Dans un navigateur actuel avec une bonne connexion, les deux approches peuvent aboutir à la même page. Elles diffèrent

quand quelque chose échoue : une page construite depuis une base conserve cette base, tandis qu'une page construite depuis les couches supérieures conserve ce que ses auteurs ont pensé à prévoir.

C'est une méthode de travail, pas une bibliothèque ni une fonctionnalité. Rien dans HTML, CSS ou JavaScript ne porte le nom d'amélioration progressive, mais les normes lui laissent sa place. Un navigateur saute une déclaration CSS qu'il ne comprend pas et ignore un script dont il ne reconnaît pas le type. La règle `@supports` permet à une feuille de style de vérifier la présence d'une fonctionnalité avant de s'appuyer dessus.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que l'amélioration progressive ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des interfaces rapides, utilisables sur téléphone et cohérentes dans tous les navigateurs de vos clients.

**Découvrir nos prestations** Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Développement cartographique GIS full-stack — cartographie full-stack, de la base au navigateur

**Développement cartographique GIS full-stack** Contactez-nous →

## Pourquoi c'est important

Les couches échouent de façons différentes, que détaille le GOV.UK Service Manual. La couche CSS tolère assez bien les erreurs : un navigateur saute une déclaration qu'il ne reconnaît pas et poursuit, même si un sélecteur inconnu lui fait abandonner la règle entière. La couche JavaScript, elle, ne les tolère pas du tout. Quand un script utilise une syntaxe ou une interface que le navigateur ne possède pas, le reste du script ne s'exécute pas. Une page dont la tâche principale dépend du script n'est donc pas plus fiable que chacun des maillons entre le serveur et le lecteur.

Ces maillons sont nombreux. La même page énumère ce qui peut empêcher le chargement ou l'exécution du CSS ou du JavaScript : une erreur réseau temporaire, un bloqueur de publicités, un réseau de diffusion de contenu indisponible, une défaillance DNS, un bogue dans une mise à jour du navigateur, un pare-feu professionnel ou personnel. Aucun de ces problèmes n'a de rapport avec un navigateur ancien, et le lecteur ne peut pas savoir lequel s'est produit. Il voit un bouton qui ne fait rien. Le manuel ne chiffre pas la fréquence de chaque cause : son argument repose sur la liste, pas sur un pourcentage.

Certains lecteurs désactivent aussi volontairement des fonctions. Le manuel indique que des utilisateurs désactivent délibérément certaines fonctions de leur navigateur et que ce choix doit être respecté. Le HTML Standard précise que de nombreux utilisateurs désactivent les scripts. Pour cette raison, et parce que les fonctions déclaratives sont souvent plus faciles à maintenir, il encourage les auteurs à les utiliser à la place des scripts lorsque c'est possible. Son exemple est l'élément `details`, qui affiche ou masque une section de la page sans une ligne de JavaScript. La norme demande également aux auteurs de prévoir une dégradation élégante de leurs applications en l'absence de prise en charge des scripts.

Pour les services publics britanniques, le manuel en fait une règle : ils devraient fonctionner avec le seul HTML, et le JavaScript, lorsqu'il est utilisé, devrait seulement améliorer le HTML et le CSS pour que le service reste utilisable quand le script échoue. Cette règle vise les services d'un gouvernement, pas l'ensemble du web, mais son raisonnement s'applique à tout site dont les auteurs ne peuvent pas choisir les lecteurs.

L'équipe y gagne aussi. Quand une amélioration tombe en panne, le lecteur perd le perfectionnement, pas la possibilité d'accomplir sa tâche : un défaut de script devient donc une gêne plutôt qu'une interruption de service. Une nouvelle fonction peut être livrée aux navigateurs qui la prennent en charge sans prévoir de solution particulière pour les autres, puisqu'ils conservent la page qu'ils avaient déjà.

## À quoi cela ressemble en pratique

L'exemple donné par le HTML Standard pour un fonctionnement dégradé sans script est un tableau dont les en-têtes de colonnes permettent de le trier à nouveau : c'est un bon test de l'approche. La base est un tableau dont les en-têtes sont des liens ordinaires. Un clic demande au serveur le même tableau trié selon cette colonne, et le serveur le renvoie dans cet ordre. Cela fonctionne dans tout navigateur, avec ou sans script.

```
<table id="orders">
  <thead>
    <tr>
      <th><a href="?sort=date">Date</a></th>
      <th><a href="?sort=total">Total</a></th>
    </tr>
  </thead>
```

```
  <tbody>
    <tr>
      <td>2026-10-20</td>
      <td>120</td>
    </tr>
    <tr>
      <td>2026-10-24</td>
      <td>45</td>
    </tr>
  </tbody>
</table>
<script type="module" src="sort.mjs"></script>
```

Le script apporte l'amélioration. Il retrouve les mêmes liens, annule la navigation et réordonne les lignes dans la page, ce qui évite un aller-retour au serveur.

```
const table = document.querySelector("#orders");
const body = table.tBodies[0];
const collator = new Intl.Collator("en", { numeric: true });

function sortBy(column) {
  const text = (row) => row.cells[column].textContent;
  const rows = [...body.rows];
  rows.sort((a, b) => collator.compare(text(a), text(b)));
  body.append(...rows);
}

for (const link of table.tHead.querySelectorAll("a")) {
  link.addEventListener("click", (event) => {
    event.preventDefault();
    sortBy(link.closest("th").cellIndex);
  });
}
```

Si une défaillance survient avant qu'un écouteur soit attaché à un lien, celui-ci reste fonctionnel. Si `sort.mjs` n'arrive jamais, les liens font ce qu'ils faisaient auparavant. Si le script lève une exception à mi-parcours, les liens des en-têtes auxquels il n'a pas encore associé d'écouteur continuent de solliciter le serveur. Un navigateur qui ne prend pas en charge les scripts modules traite `type="module"` comme un type de script inconnu et, comme l'indique le HTML Standard, l'ignore. Un script module sans `async` s'exécute une fois l'analyse de la page terminée : le tableau existe donc quand le script le recherche. Cet exemple n'assure pas de repli si un gestionnaire de clic déjà attaché lève une exception après que `preventDefault()` a annulé la navigation.

La contrepartie se trouve sur le serveur : la base exige un point d'accès capable de trier, et les deux façons de trier doivent produire le même ordre. C'est le coût habituel de l'approche, payé une fois. L'avantage recherché est de pouvoir encore accomplir la tâche principale de la page grâce au repli sur le serveur ; un gestionnaire déjà attaché doit préserver ce repli si son propre traitement échoue.

Ce site explicite sa base. Sa déclaration d'accessibilité décrit des pages complètes en HTML, CSS et SVG, sans JavaScript exécutable. Le code de recherche est conservé mais désactivé : il n'y a donc pas de filtre dans la page dont il faudrait dépendre. La déclaration signale aussi une limite restante : les petites étiquettes affichées au survol ou à la prise de focus des icônes ne peuvent pas être fermées par une touche, ce qui exigerait du code côté navigateur que le site ne livre pas. C'est pourquoi il ne satisfait qu'en partie au critère 1.4.13 Contenu au survol ou au focus.

## Où les problèmes surviennent

**Un bloc `noscript` tient lieu de base.** L'élément `noscript` n'affiche son contenu que lorsque les scripts sont désactivés, et le HTML Standard le dit sans détour : « L'élément noscript est un instrument grossier. » Les scripts peuvent être activés et celui de la page échouer quand même ; le lecteur n'obtient alors ni la page enrichie par script ni le contenu de repli. Partez d'une page qui fonctionne sans script et laissez celui-ci la modifier, comme dans l'exemple de la norme où le script ajoute un résultat dynamique à un formulaire et supprime son bouton d'envoi.

**Une fonction non prise en charge arrête tout le script.** La couche JavaScript ne tolérant pas les erreurs, un seul élément de syntaxe récent dans un paquet peut faire tomber toutes les améliorations de ce paquet, y compris celles qui auraient fonctionné. Gardez la base indépendante du script, afin qu'une défaillance ne coûte que l'amélioration. Chargez le code moderne avec `type="module"`, que les anciens navigateurs ignorent. Si ces derniers ont réellement besoin d'une version améliorée, fournissez-leur un paquet classique distinct marqué `nomodule`, que les navigateurs prenant en charge les modules n'exécutent pas.

**Un nouveau sélecteur emporte des règles valides.** Un navigateur qui rencontre un sélecteur inconnu abandonne toute la règle, déclarations comprises. Placer une pseudo-classe récente à côté de sélecteurs compris par tous les navigateurs

peut donc supprimer des styles dans des navigateurs qui les auraient appliqués. Donnez une règle distincte à un sélecteur que certains navigateurs risquent de ne pas prendre en charge.

**Les styles de repli perturbent la mise en page améliorée.** Des largeurs écrites pour les navigateurs dépourvus d'un système de mise en page récent peuvent entrer en conflit avec ce système dans les navigateurs qui le possèdent. CSS Conditional Rules Level 3 montre la solution avec une mise en page flex et un bloc `@supports not (display: flex)` pour l'autre présentation. Le texte précise que les déclarations de largeur pourraient nuire à la mise en page flex : elles doivent donc rester réservées aux styles des navigateurs qui ne la prennent pas en charge.

**Les images sont choisies par script.** Le HTML Standard souligne que, si une page ne désigne aucune image dans son balisage et en laisse le choix à un script, aucune image n'est téléchargée pour les lecteurs qui ont désactivé les scripts. Si un script remplace une image déjà indiquée dans le balisage, le navigateur peut lancer deux téléchargements. Déclarez les possibilités en HTML avec `srcset` et `sizes`, ou avec `picture`, et laissez le navigateur choisir.

**Des interactions sans base fonctionnelle.** La liste de stratégies de Champeon et Finck demande une solution de repli pour chaque comportement : un vrai `href` derrière chaque gestionnaire de clic, et un contrôle sur le serveur derrière chaque contrôle dans le navigateur. Un élément interactif qui n'existe que sous forme de gestionnaire de clic, ou un formulaire validé seulement dans le navigateur, n'a aucune base. Commencez par de vrais liens et de vrais envois de formulaire, validez toujours sur le serveur et considérez la validation dans le navigateur comme un simple confort supplémentaire.

**N'y voir qu'une question d'anciens navigateurs.** Une équipe qui croit la base destinée aux anciens navigateurs la supprime dès qu'ils disparaissent. La liste du GOV.UK concerne les réseaux, les bloqueurs, les pare-feu et les mises à jour défectueuses, que rencontrent aussi les navigateurs actuels. Conservez la base comme parcours utilisé et testé : chargez de temps en temps la page avec les scripts bloqués et accomplissez sa tâche principale de bout en bout.

# Pour aller plus loin

- Utiliser l'amélioration progressive — les recommandations du GOV.UK Service Manual, mises à jour le 27 septembre 2024 : les trois couches, leurs modes de défaillance et les raisons pour lesquelles les services publics devraient fonctionner en HTML seul.
- HTML Standard : scripts — l'élément `script` avec `type="module"` et `nomodule`, et l'élément `noscript` accompagné de la recommandation de ne pas s'y fier.
- CSS Conditional Rules Module Level 3 — le projet de recommandation candidate du W3C du 15 août 2024 qui définit `@supports` avec `not`, `and` et `or`.
- HTML Standard : images — `srcset`, `sizes` et `picture`, et les raisons pour lesquelles choisir les images par script prive d'images les lecteurs qui ont désactivé les scripts.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

**Consulter les tarifs** Contactez-nous →

# Rendez la terminologie concrète

Demandez comment « Qu'est-ce que l'amélioration progressive ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

**Découvrir nos prestations**

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/progressive-enhancement/