GUIDE PRATIQUE

6 mois pour devenir développeur web junior : roadmap complète

De la toute première ligne de code à ton premier contrat. Une méthode structurée, des outils concrets, un calendrier réaliste.

lapolaris.fr | 2026 | Édition gratuite

Reconversion & Débutants

SOMMAIRE

Ce guide est fait pour toi si tu pars de zéro

Pas de prérequis. Pas de diplôme technique exigé. Juste une méthode claire et un calendrier qu'on peut tenir.

Tu n'as jamais écrit une ligne de code de ta vie. Ou presque. Tu regardes des tutoriels depuis des semaines sans savoir vraiment par où commencer. Tu entends parler de reconversion dans le dev web mais tu ne sais pas si c'est réaliste pour toi, dans ta situation.

Ce guide répond à une seule question : **comment passer de zéro à développeur web junior en 6 mois, concrètement ?**

Pas de promesse sur le salaire. Pas de "tu peux le faire en 3 mois si tu es motivé". Une roadmap réaliste, structurée mois par mois, avec les bons outils et les bonnes ressources.

Comment utiliser ce guide

Ce n'est pas un cours. C'est une carte. Il te montre le chemin, les étapes, les bifurcations. Le travail, tu le fais toi.

Chaque chapitre correspond à une phase de ta progression. Lis-les dans l'ordre la première fois, puis reviens sur les parties dont tu as besoin. À chaque étape, tu trouveras les ressources gratuites pour progresser et les repères pour savoir si tu es prêt à passer à la suite.

> À RETENIR
>
> 6 mois, c'est une durée réaliste si tu consacres entre 2 et 4 heures par jour à l'apprentissage. Moins que ça, prévois 9 à 12 mois. Plus que ça, tu peux aller plus vite.

Ce que "junior" veut vraiment dire

Le titre de développeur junior attire de nombreuses entreprises qui cherchent des profils à former. Ce qu'on attend vraiment d'un junior sur le marché français :

- Comprendre et écrire du code lisible dans au moins une technologie (HTML/CSS, JavaScript, PHP, Python)
- Utiliser Git au quotidien sans supervision constante
- Lire de la documentation en anglais et déboguer de manière autonome
- Communiquer clairement sur ce qu'on fait, ce qui bloque, et pourquoi
- Avoir un ou deux projets concrets à montrer (portfolio, GitHub)

> CE GUIDE NE TE PROMET PAS UN JOB
>
> Il te donne les bases, la structure et les outils pour te mettre en position de décrocher un entretien. Le reste dépend de ton travail, de ta constance et du marché au moment où tu postules.

Avant de coder, comprendre dans quoi tu t'engages

Le développement web recouvre des réalités très différentes. Choisir sa direction dès le départ évite de perdre 3 mois sur la mauvaise piste.

Beaucoup de personnes en reconversion commencent par React parce qu'elles ont vu que c'était populaire, se perdent dans la complexité, et abandonnent avant d'avoir les bases. Avant de toucher à du code, il faut comprendre les trois grands profils du métier.

Les trois profils du développeur web

FRONTEND

Développeur Frontend

Ce que l'utilisateur voit et avec quoi il interagit. Boutons, formulaires, responsive. Technologies : HTML, CSS, JavaScript, React ou Vue.

BACKEND

Développeur Backend

Ce qui tourne côté serveur : bases de données, APIs, logique métier, authentification. Technologies : PHP/Symfony, Python, Node.js.

FULLSTACK

Développeur Fullstack

Il fait les deux. Très demandé dans les PME et startups. En pratique, souvent plus fort d'un côté. Le fullstack junior est rare.

PLUS TARD

Autres rôles proches

Développeur mobile (React Native, Flutter), DevOps, intégrateur. Ces spécialités viennent après une première expérience.

> CONSEIL CONCRET
>
> En reconversion, commence par le frontend ou le backend, pas les deux. Le fullstack, c'est une évolution naturelle après 1 ou 2 ans d'expérience.

Les salaires juniors en France en 2026

28k	35k	42k
Salaire junior bas de gamme (province, agence) [1]	Salaire junior médian (France, tous secteurs) [1]	Salaire junior haut (Paris, startups, backend) [2]

Ces chiffres sont indicatifs. Un développeur en Île-de-France gagne en moyenne 17% de plus qu'en province[3]. En dessous de 33k pour un poste full remote à Paris, c'est sous-évalué.

La réalité du marché junior

Le recrutement junior dans la tech a subi une baisse de 19% en 2024 selon l'APEC[4]. Il y a plus de candidats que d'offres, notamment parce que les bootcamps intensifs ont formé beaucoup de profils en peu de temps. C'est une raison de le faire bien, pas de ne pas se lancer.

Ce qui fait la différence entre deux profils juniors :

- La qualité du portfolio : des projets réels, déployés, avec du code propre sur GitHub
- La capacité à expliquer ses choix techniques, même simples
- L'attitude en entretien : curiosité, humilité, envie d'apprendre
- La spécialisation : un junior PHP/Symfony est plus employable qu'un junior "un peu de tout"[5]
- Le réseau : LinkedIn actif, meetups, contributions open source même mineures

Télétravail : qu'est-ce qu'on peut espérer en junior ?

Un junior full remote dès le premier job, c'est rare mais possible. Selon le Stack Overflow Developer Survey 2024, 42% des développeurs travaillent en hybride, et le présentiel est en hausse pour la troisième année consécutive[6]. La plupart des entreprises préfèrent leurs juniors en présentiel les 6 premiers mois. Ce n'est pas un frein, c'est souvent une opportunité d'apprendre plus vite.

> **À RETENIR POUR LA SUITE**
>
> Tu n'as pas besoin de choisir ta spécialité maintenant. Les deux premiers mois de la roadmap sont communs à tous les profils. La bifurcation frontend / backend arrive au mois 3.

RÉFÉRENCES DE CE CHAPITRE

1 Welcome to the Jungle, "Salaire d'un Développeur Web en 2024", welcometothejungle.com

2 Le Journal des Entreprises, baromètre Seyos 2025, lejournaldesentreprises.com

3 Softy Pro, "Marché du travail 2025", recrutement.softy.pro

4 APEC, Prévisions 2025, baisse de 19% des recrutements juniors informatique en 2024, emploi.developpez.com

5 Externatic, "Pourquoi les développeurs juniors ont-ils du mal à trouver un emploi ?", externatic.fr

6 Stack Overflow Developer Survey 2024, section Work, survey.stackoverflow.co/2024/work

Les fondations : HTML, CSS, Git

Avant tout framework, avant tout langage complexe, il y a un socle que tout développeur web doit maîtriser. Ce mois, tu le construis.

HTML et CSS ne sont pas des langages de programmation. Ce sont des langages de structure et de mise en forme. Mais les maîtriser vraiment — modèle de boîte, flux, positionnement, responsive — prend un mois complet si tu travailles sérieusement.

Ce que tu dois apprendre ce mois

HTML : LA STRUCTURE

- Balises essentielles : `h1` – `h6`, `p`, `a`, `img`, `ul`, `li`, `div`, `span`
- Sémantique HTML5 : `header`, `nav`, `main`, `section`, `article`, `footer`
- Formulaires : `input`, `textarea`, `button`, `label`, `form`

CSS : LA MISE EN FORME

- Modèle de boîte : `margin`, `padding`, `border`, `box-sizing`
- Flexbox (indispensable) et Grid (mises en page complexes)
- Media queries pour le responsive, variables CSS, pseudo-classes `:hover` et `:focus`

GIT ET CLI : LES OUTILS DU MÉTIER

- Terminal : `cd`, `ls`, `mkdir`, `rm`
- Git : `git init`, `git add`, `git commit`, `git push`, `git pull`
- GitHub : créer un dépôt, pousser son code, lire le code des autres

> PROJET DU MOIS
>
> Crée une page personnelle de A à Z : photo, bio, liste de compétences, formulaire de contact. Pas de template, pas de copier-coller. Pousse le résultat sur GitHub.

Ressources gratuites recommandées

DOCUMENTATION

MDN Web Docs

La référence absolue pour HTML et CSS. En français, exhaustive, gratuite.

EXERCICES INTERACTIFS

freeCodeCamp

Parcours Responsive Web Design entièrement gratuit, des centaines d'exercices progressifs.

CSS INTERACTIF

Flexbox Froggy

Apprendre Flexbox en jouant. 24 niveaux, disponible en français.

GIT

Learn Git Branching

Visualiser et comprendre Git par la pratique. Le meilleur outil pour apprendre le versioning.

Comment savoir si tu es prêt à passer au mois 2

- [ ] Tu peux créer une page HTML structurée de zéro, sans regarder de documentation
- [ ] Tu comprends la différence entre `margin` et `padding` sans hésiter
- [ ] Tu peux centrer un élément avec Flexbox en moins de 30 secondes
- [ ] Ta page est lisible sur mobile (responsive)
- [ ] Ton projet du mois est poussé sur GitHub avec des commits clairs
- [ ] Tu utilises le terminal pour naviguer dans tes dossiers sans passer par l'explorateur

La logique de programmation avec JavaScript

JavaScript, c'est le premier vrai langage de programmation que tu vas apprendre. Il va te faire travailler. Et c'est exactement ce qu'il faut.

HTML et CSS t'ont appris à construire et à mettre en forme. JavaScript t'apprend à raisonner. C'est ici que beaucoup de reconvertis décrochent, parce que le gap entre "comprendre un exemple" et "écrire son propre code" est brutal.

> POURQUOI JAVASCRIPT AVANT UN FRAMEWORK
>
> Beaucoup de tutoriels commencent directement avec React. C'est une erreur. Si tu ne comprends pas les fonctions, les tableaux, les objets et le DOM en JavaScript pur, React sera une boîte noire. Apprends d'abord la mécanique, ensuite l'outil.

Ce que tu dois apprendre ce mois

LES BASES DU LANGAGE

- Variables : `let`, `const`, types (string, number, boolean, null, undefined)
- Conditions : `if`, `else`, opérateurs `===`, `!==`, `>`
- Boucles : `for`, `while`, `forEach`
- Fonctions : déclaration, paramètres, valeur de retour, fonctions fléchées (`=>`)
- Tableaux : `push`, `pop`, `map`, `filter`, `find`
- Objets : création, accès aux propriétés, méthodes

INTERACTION AVEC LA PAGE (DOM)

- Sélectionner : `document.querySelector`, `getElementById`
- Modifier : `innerHTML`, `textContent`, `value`
- Écouter des événements : `addEventListener` (clics, formulaires, clavier)

NOTIONS MODERNES ESSENTIELLES

- `Promise` et `async/await` pour comprendre l'asynchrone
- `fetch` API pour récupérer des données depuis une API externe
- JSON : lire et manipuler des données structurées

PROJET DU MOIS

Construis une petite application : liste de tâches, convertisseur de devises via une API gratuite, ou un quiz interactif. L'objectif : l'utilisateur interagit, le résultat change dynamiquement sans recharger la page.

Ressources gratuites recommandées

COURS COMPLET

javascript.info

Le meilleur cours JavaScript gratuit. Structuré, progressif, exercices à chaque chapitre.

EXERCICES

freeCodeCamp JS

Parcours JS Algorithms and Data Structures. Des centaines d'exercices, du plus simple au complexe.

PRATIQUE QUOTIDIENNE

Codewars

Des katas classés par difficulté. 15 minutes par jour suffisent pour progresser en logique.

DOCUMENTATION

MDN JavaScript

La référence officielle pour chaque méthode et objet natif. À garder en favori permanent.

Checklist de validation

- [ ] Tu peux écrire une fonction sans regarder un exemple
- [ ] Tu sais utiliser `map` et `filter` sur un tableau
- [ ] Tu as réussi à faire un appel `fetch` et afficher les données sur une page
- [ ] Tu sais déchiffrer un message d'erreur dans la console du navigateur

Formation JavaScript chez LaPolaris

Les fondamentaux du langage jusqu'à la manipulation du DOM, avec projets pratiques.

Voir la formation

CHAPITRE 04 / MOIS 3

Choisir sa spécialité : frontend ou backend

C'est le moment de la bifurcation. Ce choix va orienter les 3 prochains mois de ton apprentissage. Il mérite réflexion, pas précipitation.

La voie Frontend

Tu construis ce que les utilisateurs voient et avec quoi ils interagissent. Le framework dominant est React, suivi par Vue.

CHOISIR SI...

Tu es à l'aise avec le visuel

Tu aimes ce qui se voit, tu es sensible au design, le résultat de ton travail est immédiatement visible. Tu préfères itérer sur une interface plutôt qu'optimiser une requête SQL.

ATTENTION

Marché plus concurrentiel

Les profils frontend junior sont plus nombreux. Un bon portfolio visuel et la maîtrise de React font toute la différence. La barre d'entrée est élevée.

La voie Backend

Tu construis la logique invisible : APIs, bases de données, authentification, performances serveur. Technologies : PHP/Symfony, Python, Node.js.

CHOISIR SI...

Tu aimes la logique pure

Tu préfères résoudre des problèmes de structure, d'organisation de données, de performance. Tu n'as pas besoin de voir un résultat visuel immédiat.

AVANTAGE

Moins de concurrence junior

Les profils backend juniors sont moins nombreux et souvent mieux rémunérés. PHP/Symfony reste très demandé dans les agences et PME françaises.

Ce que tu travailles ce mois selon ta voie

FRONTEND / REACT

react.dev/learn

JSX, composants, props, premier `useState`. La doc officielle de React, interactive et progressive.

BACKEND / PHP

php.net/manual/fr

PHP moderne, POO fondamentale, intro Symfony MVC, premiers contrôleurs, bases SQL.

BACKEND / PYTHON

docs.python.org/fr/3

Syntaxe, listes, dicts, fonctions, classes de base. Intro FastAPI ou Django selon l'objectif.

SQL (TOUS PROFILS)

SQLZoo

Apprendre SQL par la pratique dans le navigateur. Gratuit, progressif, efficace.

CHAPITRE 05 / MOIS 4

Monter en compétence sur sa stack

Tu as choisi ta voie. Ce mois, tu creuses. L'objectif : passer de "je comprends les exemples" à "je construis moi-même".

> **LE MUR DU MOIS 4**
>
> Presque tous les autodidactes rencontrent un mur autour du quatrième mois. La progression semble ralentir, les erreurs s'accumulent, la motivation baisse. Ce n'est pas un signe que tu n'es pas fait pour ça. Continue.

Voie Frontend

- React intermédiaire : `useEffect`, `useContext`, Context API
- Routing avec React Router : navigation SPA entre plusieurs pages
- Appels API depuis React : `fetch` ou Axios, gestion des états de chargement et d'erreur
- Introduction à TypeScript : typage de base, props typées
- Déploiement sur Vercel ou Netlify

Voie Backend PHP/Symfony

- Doctrine ORM : entités, migrations, relations (OneToMany, ManyToMany)
- Formulaires Symfony : création, validation, gestion des erreurs
- Authentification avec Security Bundle, gestion des rôles
- API REST basique : retourner du JSON depuis un contrôleur, tests PHPUnit

Voie Backend Python

- Python orienté objet : classes, héritage, encapsulation
- FastAPI : routes, paramètres, modèles Pydantic, réponses JSON
- SQLAlchemy ou SQLModel pour la persistance, authentification JWT basique
- Déploiement sur Railway ou Render

PROJET DU MOIS

Frontend : app de gestion de liste avec React, persistance API ou localStorage. Backend : API REST avec 3 ressources, authentification simple, base de données fonctionnelle.

À ce stade, les tutoriels gratuits montrent souvent des cas simplifiés. La réalité d'un projet en entreprise est plus complexe : gestion des erreurs, organisation du code, bonnes pratiques, code review. C'est là qu'une formation structurée change la donne.

Coder avec l'IA : l'outil que tout junior doit maîtriser

En 2026, l'IA n'est plus un gadget. C'est un outil du quotidien pour les développeurs, y compris les juniors. Savoir l'utiliser correctement est devenu une compétence à part entière.

POURQUOI UN CHAPITRE DÉDIÉ

L'IA ne remplace pas l'apprentissage. Mais elle accélère la compréhension, aide au débogage, et fait gagner un temps considérable sur les tâches répétitives. Ignorer ces outils en 2026, c'est comme ignorer Google en 2010.

Les outils IA essentiels pour un développeur

ASSISTANTS DE CODE

GitHub Copilot

Autocomplétion intelligente directement dans VS Code. Suggère du code en temps réel pendant que tu tapes. Plan gratuit disponible pour les étudiants.

ASSISTANTS CONVERSATIONNELS

Claude & ChatGPT

Poser des questions techniques, expliquer du code, générer des exemples, déboguer une erreur. Comme un mentor disponible 24h/24.

IDE AUGMENTÉ

Cursor / Claude Code

Éditeurs de code avec IA intégrée. Tu décris ce que tu veux en français, l'IA génère ou modifie le code dans ton projet.

DOCUMENTATION

IA + docs officielles

Utilise l'IA pour résumer une doc complexe ou reformuler un concept que tu ne comprends pas. Toujours vérifier avec la source officielle.

Comment bien utiliser l'IA quand on apprend

- **Ne copie pas aveuglément.** Si l'IA te génère du code, lis-le ligne par ligne. Tu dois pouvoir l'expliquer
- **Utilise l'IA pour comprendre, pas pour éviter de réfléchir.** Demande "explique-moi ce code" avant "écris-moi ce code"
- **Apprends à écrire de bons prompts.** Plus ta question est précise, meilleure est la réponse. Donne le contexte, le langage, le résultat attendu
- **Vérifie toujours.** L'IA peut halluciner : inventer des fonctions qui n'existent pas, ou proposer du code obsolète
- **Code d'abord, IA ensuite.** Essaie de résoudre le problème seul pendant 15 minutes avant de demander de l'aide à l'IA

Cas d'usage concrets au quotidien

DÉBOGAGE

Coller une erreur dans Claude ou ChatGPT

Copie le message d'erreur + le code concerné. L'IA t'explique le problème et propose une correction en quelques secondes.

APPRENTISSAGE

Demander des explications pas à pas

"Explique-moi useEffect en React comme si j'avais 10 ans." L'IA adapte son niveau à ta demande.

PRODUCTIVITÉ

Générer du code répétitif

Formulaires, validations, structures de fichiers : l'IA te fait gagner du temps sur le boilerplate.

CODE REVIEW

Faire relire son code par l'IA

"Quels problèmes vois-tu dans ce code ?" L'IA repère les mauvaises pratiques et propose des améliorations.

ATTENTION EN ENTRETIEN

De plus en plus de recruteurs demandent : "Quels outils IA utilises-tu et comment ?" Savoir répondre honnêtement — "j'utilise Copilot pour l'autocomplétion et Claude pour le débogage, mais je comprends ce que le code fait" — c'est un signal positif. Dire "je n'utilise pas l'IA" en 2026, c'est un signal négatif.

Ce que l'IA ne remplace pas

- La compréhension des fondamentaux : si tu ne comprends pas les boucles, l'IA ne te sauvera pas en entretien technique
- La capacité à architecturer un projet : l'IA aide sur les détails, pas sur la vision d'ensemble
- Le travail en équipe : communiquer avec des humains, faire des code reviews, discuter des choix techniques
- Le sens critique : savoir quand le code généré est mauvais, c'est une compétence humaine

RÈGLE D'OR

L'IA est un accélérateur, pas un raccourci. Un junior qui utilise bien l'IA progresse deux fois plus vite. Un junior qui copie-colle sans comprendre stagne deux fois plus vite.

CHAPITRE 07 / MOIS 5

Construire son portfolio

Le portfolio, c'est ta carte de visite technique. En l'absence d'expérience professionnelle, c'est lui qui parle à ta place lors d'un entretien.

Un bon portfolio n'a pas besoin d'être impressionnant. Il doit être honnête, propre, et montrer que tu sais construire quelque chose de fonctionnel de bout en bout.

Quels projets inclure

RECOMMANDÉ

2 à 3 projets personnels

Des applications que tu as construites toi-même. Elles doivent fonctionner, être déployées, et avoir un code lisible.

RECOMMANDÉ

1 projet plus ambitieux

Authentification, CRUD complet, appels API, interface responsive. C'est celui-là dont tu parleras en entretien.

OPTIONNEL

Contributions open source

Même une petite correction de bug sur GitHub montre que tu sais travailler dans un contexte de code existant.

À ÉVITER

Clones de tutoriels non modifiés

Un clone YouTube sans modification personnelle, ça se voit. Si tu l'inclus, modifie-le significativement.

GitHub : présenter son code correctement

- Chaque projet doit avoir un `README.md` : description, technos, captures d'écran, instructions d'installation
- Commits lisibles : "feat: add user authentication" vaut mieux que "update"
- Un `.gitignore` propre, pas de `.env` ni de `node_modules` dans le dépôt

Déployer ses projets

FRONTEND

Vercel

Déploiement en un clic depuis GitHub. Gratuit pour les projets personnels. HTTPS automatique.

FRONTEND ALTERNATIF

Netlify

Idéal pour les sites statiques. Formulaires intégrés, fonctions serverless gratuites.

BACKEND / FULLSTACK

Railway

PHP, Python, Node.js avec base de données. Tier gratuit disponible pour les démos.

STATIQUE GRATUIT

GitHub Pages

Héberger gratuitement un site statique directement depuis ton dépôt GitHub.

Checklist portfolio

- [ ] Au moins 2 projets déployés, accessibles via une URL publique
- [ ] Chaque projet a un README clair avec captures d'écran
- [ ] Tu peux expliquer chaque ligne de code de ton projet principal
- [ ] Ton profil GitHub est propre : photo, bio, dépôts épinglés

Construire un site portfolio personnel

En plus de GitHub, un site portfolio dédié est un vrai plus. Il montre ta capacité à déployer un projet de A à Z et donne un support visuel pour les recruteurs non techniques.

- **Page d'accueil :** ton nom, ton titre (ex. "Développeur React junior"), une phrase d'accroche
- **Section projets :** capture d'écran, description courte, technos utilisées, liens vers le code et la démo
- **Section à propos :** ton parcours en 3 à 5 lignes, ce qui te motive, ce que tu cherches
- **Contact :** formulaire simple ou liens LinkedIn / email. Pas besoin de plus

CONSEIL TECHNIQUE

Ton site portfolio est lui-même un projet. Construis-le avec ta stack : React si tu es frontend, une page statique propre si tu es backend. Déploie-le sur Vercel ou Netlify avec un nom de domaine personnalisé (environ 10€/an).

ERREUR FRÉQUENTE

Ne passe pas 3 semaines à peaufiner le design de ton portfolio au détriment du code de tes projets. Le contenu (tes projets fonctionnels) compte 10 fois plus que l'apparence du site portfolio lui-même.

CHAPITRE 08 / MOIS 6

Décrocher son premier job

Tu as les bases, tu as des projets, tu as un profil GitHub. Ce mois, tu apprends à te vendre. C'est un métier en soi.

Le CV du développeur junior

- Une page maximum. Deux si tu as un parcours professionnel long et pertinent
- Section "Compétences techniques" : langages maîtrisés, frameworks connus, outils utilisés
- Section "Projets" aussi importante que l'expérience : mets les liens GitHub et les démos
- Pas de barres de progression ("JavaScript : 7/10"). Ça ne veut rien dire et ça agace les recruteurs techniques

LinkedIn : ton canal de prospection principal

- Titre clair : "Développeur web junior | React / JavaScript" ou "Développeur backend | PHP Symfony | En recherche"
- Résumé : 3 à 5 lignes sur ton parcours, ta reconversion, ce que tu cherches. Humain, pas robotique
- Active "Open to Work" en mode visible par les recruteurs
- Publie régulièrement : projet, problème résolu, apprentissage. La visibilité organique fonctionne

Où postuler

OFFRES D'EMPLOI

Welcome to the Jungle

La plateforme de référence pour les offres tech en France. Fiches entreprises détaillées.

OFFRES D'EMPLOI

LinkedIn Jobs

Active les alertes "développeur junior" + ta stack. Les offres arrivent aussi directement de recruteurs.

STARTUPS FRANÇAISES

Maddyness

Offres dans l'écosystème startup français. Moins de candidats, plus de chances pour un junior motivé.

FREELANCE

Malt

Premières missions freelance. Accessible dès le niveau junior si ton portfolio est solide.

PITCHER SA RECONVERSION EN ENTRETIEN

Structure efficace : ce que tu faisais avant (2 phrases), le déclic (1 phrase honnête), ce que tu as fait concrètement depuis (formations, projets), ce que tu cherches et pourquoi cette entreprise.

Tu es prêt quand ces signaux sont là

Pas de checklist parfaite, pas de certificat qui te dit que tu es junior. Mais il y a des signaux concrets.

6 mois de travail sérieux, c'est transformateur.

Ce n'est pas une promesse de job garanti. C'est une promesse de compétences réelles, d'un portfolio honnête, et d'une posture professionnelle qui te met en position de décrocher un premier entretien.

Tu débogues seul

Face à une erreur, tu lis le message, tu cherches, tu testes. Tu avances.

Tu as des projets déployés

Au moins 2 apps accessibles en ligne, code propre sur GitHub, README clair.

Tu expliques ce que tu fais

Tu peux décrire tes choix techniques à quelqu'un qui ne code pas.

Tu lis la documentation

Tu n'as plus peur d'une doc officielle en anglais. Tu sais chercher.

Tu utilises Git naturellement

Commits, branches, merges : tu les utilises sans y penser.

Tu as envie de continuer

Tu codes par plaisir autant que par obligation. C'est le signal le plus fort.

Les pièges à éviter jusqu'au bout

- Le tutorial hell : enchaîner les cours sans jamais construire seul. À partir du mois 3, code plus que tu ne regardes
- Vouloir tout savoir avant de postuler. Postule quand tu as les signaux ci-dessus, pas à "100%"
- Ignorer le réseau. La majorité des offres junior se décrochent via une recommandation directe
- Négliger le code propre. Un code fonctionnel mais illisible est un signal négatif en entretien

UN DERNIER MOT

Ce guide t'a donné la structure. Le reste, c'est toi. La reconversion dans le dev web n'est pas facile, mais elle est réelle. Ce n'est pas une question de talent, c'est une question de méthode et de constance.