

CODE & CASH :

TRANSFORME DE PETITS

PROJETS WEB EN GROS

RÉSULTATS

Prepared By
Ech-Chkoubi Mohamed

0755683285

mohamed.echchkoubi@gmail.com

PETITS PROJETS WEB À GRANDS EFFETS

Table des Matières

Introduction

- ☒ Pourquoi ce guide ? Objectifs
 - ☒ d'apprentissage Technologies
 - ☒ utilisées Philosophie : petits projets, grands effets
-

1. Créer un site vitrine simple pour un artisan

- ☒ local Structure de base (HTML/CSS) Sections
 - ☒ typiques (Accueil, Services, Contact)
 - ☒ Optimisation responsive Monétisation : site clé-en-main pour indépendants
-

2. Construire une landing page responsive pour un produit fictif

- ☒ Importance d'une landing
 - ☒ page Hiérarchie visuelle et
 - ☒ call-to-action HTML/CSS +
 - ☒ flexbox/grid Idée de produit à mettre en avant Astuces UX/UI
-

3. Faire un convertisseur PDF → Excel via interface web

- ☒ Besoin métier ciblé Interface web en
 - ☒ HTML/CSS/JS Appel à une API externe
 - ☒ (PDF parsing) Affichage et
 - ☒ téléchargement du fichier Excel Cas d'usage : comptables, indépendants
-

4. Automatiser un fichier Excel avec VBA (ex : suivi de budget)

- ☒ Utilisation de macrospour simplifier la
 - ☒ gestion Crédit : création d'un dashboard simple
 - ☒ dans Excel Automatiser les calculs de
 - ☒ dépenses Déclencheurs et boutons VBA
 - ☒ Valeur ajoutée pour les clients non-tech
-

5. RÉALISER UN MINI E-COMMERCE SANS CMS (JUSTE HTML/CSS/Javascript)

- ☒ Produits statiques affichés dynamiquement
 - ☒ Système de panier local (JS) Validation de
 - ☒ commande simulée Interface responsive À
 - ☒ proposer à de petits commerçants ou artisans
-

6. GÉNÉRER UN CV WEB DYNAMIQUE

- ☒ Structure propre d'un CV HTML Intégration de
 - ☒ données dynamiques (JSON ou JS) Mise en forme attractive (animations, polices pro)
 - ☒ Hébergement facile pour partage (GitHub Pages)
 - ☒ À proposer comme service de personal branding
-

7. CRÉER UN TABLEAU DE BORD INTERACTIF EN JAVASCRIPT

- ☒ Présentation des KPIs Filtres de période (jour, semaine, mois, année) Graphiques avec
 - ☒ Chart.js Chargement et mise à jour dynamique
 - ☒ des données Idées d'utilisation : gestion, RH, marketing
-

8. CRÉER UN FORMULAIRE INTELLIGENT AVEC VALIDATIONS

- ☒ Structure HTML propre Validation en temps réel (JavaScript) Feedback visuel
 - ☒ (couleurs, messages, icônes) UX +
 - ☒ Accessibilité (ARIA, messages clairs)
 - ☒ Revente de formulaires personnalisés
-

9. INTÉGRER UNE API MÉTÉO, CRYPTO OU AUTRE DANS UNE APP SIMPLE

- ☒ Obtenir une clé API (ex : OpenWeatherMap)
 - ☒ Requête HTTP avec fetch Parsing JSON et
 - ☒ affichage dynamique Design responsive et
 - ☒ user-friendly Évolutions possibles : intégration de plusieurs APIs
-

10. OFFRIR UN SCRIPT "PRÊT-À-VENDRE" POUR DES PETITS BESOINS CLIENTS

- ☒ Identifier des besoins simples & fréquents Exemple
- ☒ concret : script d'affichage de témoignages
- ☒ Fichier JSON + affichage dynamique (JS)

- ☒ Intégration facile sur n'importe quel site
 - ☒ Packaging, documentation & support client
-

CONCLUSION : BONUS

- ☒ Résumé des compétences acquises Comment aller plus loin : frameworks, backend, bases de données Conseils pour freelancing et vente de produits numériques L'importance de pratiquer et publier ses projets Votre roadmap vers l'indépendance technique
-

INTRODUCTION

Bienvenue dans Petits Projets Web à Grands Effets

!

Ce guide est conçu pour vous – étudiant, développeur débutant ou freelance ambitieux – qui souhaitez renforcer vos compétences, bâtir un portfolio concret et prendre confiance dans votre capacité à créer des projets utiles, concrets et monétisables.

Dans le monde du développement web, il ne suffit pas de connaître la théorie. Ce qui fait la différence, c'est la pratique. Chaque projet présenté dans ce livre est une opportunité de :

- Mettre en application vos connaissances,
- Apprendre à résoudre des problèmes réels,
- Construire quelque chose de tangible que vous pouvez montrer ou même vendre.

Ces projets sont volontairement simples mais puissants : ils ne nécessitent pas des compétences avancées, mais ils ont un fort potentiel pour faire progresser votre apprentissage et pour enrichir votre portfolio.

Vous trouverez des projets répartis autour de trois axes :

1. Sites vitrines – pour des clients locaux ou pour vous-même ;
2. Applications interactives – comme un convertisseur PDF/Excel ou une mini-boutique en ligne ;
3. Outils pratiques – utiles dans la vie de tous les jours ou pour le travail.

Chaque fiche-projet comprend :

- Une explication du "Pourquoi" – l'intérêt technique et/ou professionnel du projet ;
- Un guide pas à pas – avec le HTML, CSS, JavaScript et parfois un peu de backend ;
- Des conseils techniques – pour aller plus loin ou éviter les pièges courants ;
- Des pistes de monétisation – pour transformer un projet en source de revenu potentiel.

L'accent est mis sur :

- La simplicité : vous pouvez commencer même si vous n'êtes pas un expert ;

- La rapidité : chaque projet peut être réalisé en quelques heures à quelques jours;
 - La valeur réelle : ces projets répondent à des besoins concrets.
-

Objectif de cet ebook :

Vous aider à passer de la théorie à la pratique, à transformer vos lignes de code en projets concrets, et à développer peu à peu une autonomie technique ET entrepreneuriale.

Prêt à relever le défi ?

Alors préparez votre éditeur de code préféré, et commençons cette aventure.
Parce que parfois, ce sont les petits projets qui font les plus grands effets.

1. CRÉER UN SITE VITRINE SIMPLE POUR UN ARTISAN LOCAL

Pourquoi ce projet ?

Un site vitrine est la carte de visite numérique d'une entreprise. Pour un artisan local (plombier, boulanger, coiffeur, etc.), c'est souvent le premier point de contact avec de nouveaux clients.

Ce projet est excellent pour débuter car il met l'accent sur les fondamentaux du web :

- HTML pour la structure,
 - CSS pour le style,
 - Et un peu de JavaScript pour l'interactivité.
-

Comment faire ?

1.Comprendre le besoin Commencez par imaginer ou discuter avec un artisan (réel ou fictif) pour cerner ses attentes. Posez-vous ces questions :

- Quels services propose-t-il ?
 - Quelles sont ses coordonnées ?
 - A-t-il des photos de son travail ?
 - A-t-il une identité visuelle ou des couleurs spécifiques ?
-

2. Structure HTML (index.html) Voici

les sections classiques à intégrer :

- En-tête (`<header>`) : nom de l'artisan + menu de navigation.
- Section d'accueil (`#hero`) : une image, un slogan ou une accroche.
- À propos (`#about`) : présentation de l'artisan, son parcours.
- Services (`#services`) : liste claire des prestations proposées.

- Galerie (#gallery) : images de réalisations.
- Contact (#contact) : formulaire + coordonnées.
- Pied de page (<footer>) : mentions légales et réseaux sociaux.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nom de l'Artisan - Votre Expert Local</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <nav>
            <!-- Menu de navigation -->
        </nav>
    </header>
    <main>
        <section id="hero"><!-- Section d'accueil --></section>
        <section id="about"><!-- Section À propos --></section>
        <section id="services"><!-- Section Services --></section>
        <section id="gallery"><!-- Section Galerie --></section>
        <section id="contact"><!-- Section Contact --></section>
    </main>
    <footer>
        <!-- Pied de page -->
    </footer>
    <script src="script.js"></script>
</body>
</html>
```

3. Stylisation CSS (style.css)

- Utilisez Flexbox ou CSS Grid pour la mise en page.
- Utilisez une palette de couleurs simple et cohérente.
- Assurez la responsivité mobile avec des media queries.
- Ajoutez des transitions ou effets subtils au survol.

```
/* Styles généraux */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Media queries pour le responsive */
@media (max-width: 768px) {
    /* Styles pour les écrans plus petits */
}
```

4. Interactivité JavaScript (script.js – optionnel)

Ajoutez des petits effets pour dynamiser le site :

- Menu hamburger pour mobile.

- Effets de scroll (apparition au défilement).
- Validation du formulaire de contact (champs requis, format email, etc.).

```
// Exemple de menu hamburger
const navToggle = document.querySelector('.nav-toggle');
const navMenu = document.querySelector('.nav-menu');

if (navToggle) {
    navToggle.addEventListener('click', () => {
        navMenu.classList.toggle('active');
    });
}
```

CONSEILS ET ASTUCES

- Simplicité : allez à l'essentiel. Les visiteurs veulent une info rapide.
- Photos de qualité : un artisan se vend avec son travail visuel.
- CTA clair : bouton « Appeler », « Réserver », « Demander un devis ».
- SEO local : utilisez des balises sémantiques, des titres h1/h2 pertinents, et remplissez les alt des images.
- Hébergement facile : déployez gratuitement via GitHub Pages, Netlify ou Vercel.
-

MONÉTISATION

Voici comment rentabiliser ce type de projet :

- Proposer vos services à d'autres artisans locaux (ex. : création de leur site vitrine).
- Offrir un forfait de maintenance mensuel pour mise à jour de contenu, sauvegardes, sécurité.

2. CONSTRUIRE UNE LANDING PAGE RESPONSIVE POUR UN PRODUIT FICTIF

Pourquoi ce projet ?

Une landing page (page d'atterrissement) est une page web conçue dans un but unique : convertir le visiteur en prospect ou en client. Elle est essentielle pour les campagnes marketing, le lancement de produits ou la collecte d'inscriptions. Ce projet vous permettra de maîtriser le design orienté conversion, le responsive design avancé et l'intégration d'éléments persuasifs.

Comment faire ?

1. Définir le produit et l'objectif

Imaginez un produit fictif (ex : une application de méditation, un gadget technologique, un service de coaching). Déterminez l'objectif principal de la page : collecter des emails, vendre un produit, inciter à télécharger une app, etc.

2. Créer la structure HTML (index.html)

Une landing page efficace est généralement composée de sections claires et concises :

- Section Héro : Un titre accrocheur, un sous-titre qui exprime la proposition de valeur, une image ou vidéo du produit, et un bouton d'appel à l'action principal (ex : "S'inscrire", "Acheter maintenant").
- Section Bénéfices / Fonctionnalités : Présentez les avantages du produit, pas juste ses caractéristiques. Utilisez des icônes ou des illustrations simples.
- Section Témoignages : Citez quelques "clients satisfaits" fictifs pour renforcer la crédibilité.
- Section Comment ça marche ? (Optionnelle) : Expliquez les étapes simples d'utilisation du produit.
- Section FAQ (optionnelle) : Répondez aux questions fréquentes que l'utilisateur pourrait se poser.
- Section Appel à l'action final : Répétez le bouton principal, souvent accompagné d'un formulaire simple.

- Pied de page : Liens vers la politique de confidentialité et les conditions d'utilisation.

Exemple de base :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Landing Page - Votre Produit Incroyable</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <main>
        <section id="hero">
            <h1>Titre Accrocheur de Votre Produit</h1>
            <p>Sous-titre expliquant la valeur unique.</p>
            
            <button class="cta-button">Appel à l'action</button>
        </section>
        <section id="benefits"></section>
        <section id="testimonials"></section>
        <section id="final-cta"></section>
    </main>
    <footer></footer>
</body>
</html>
```

3. Styliser avec CSS (style.css)

Le responsive design est essentiel. Voici quelques bonnes pratiques :

- Mobile-first : Commencez par les petits écrans, puis utilisez des media queries pour les plus grands.
- Typographie : Choisissez des polices lisibles et utilisez des tailles en rem ou em pour qu'elles soient adaptatives.
- Espacement : Utilisez marges et paddings généreux pour améliorer la lisibilité.
- Couleurs & images : Des couleurs vives pour les boutons (CTA) attirent l'attention. Les images doivent être optimisées pour le web.
- Animations légères : Les transitions ou effets au survol peuvent rendre la page plus dynamique, sans la surcharger.

Exemplesimple :

```
/* Mobile-first */
body {
    font-size: 16px;
}

.cta-button {
    padding: 15px 30px;
    background-color: #FF6B6B;
    color: white;
    border-radius: 5px;
    text-decoration: none;
}

/* Desktop */
@media (min-width: 768px) {
    body {
        font-size: 18px;
    }
}
```

4. Ajouter un peu d'interactivité avec JavaScript (optionnel mais recommandé)

- Validation de formulaire : Empêchez les envois incomplets (email invalide, champs requis non remplis...).
- Effets de défilement : Par exemple, apparition des éléments au scroll ou effet parallaxe.
- Compteur : Pour créer un sentiment d'urgence (ex : "Offre valable pendant encore 12h").
- Intégration d'un service d'emailing : MailChimp, Sendinblue, etc., pour capturer les adresses email.

CONSEILS ET ASTUCES

- Un seul objectif par page : Évitez de distraire le visiteur. Tout doit le guider vers l'action souhaitée.
- Clarté et concision : Soyez directs. Les utilisateurs scannent les pages plutôt qu'ils ne lisent.
- Preuve sociale : Témoignages, logos de clients, chiffres de réussite ("+1 000 utilisateurs") rassurent.

- Tests sur tous les appareils : Assurez-vous que votre page est parfaite sur mobile, tablette et ordinateur.
 - Vitesse de chargement : Un site lent = visiteurs qui partent. Compressez vos images et minifiez votre code.
-

MONÉTISATION

- Vendre vos services : Créez des landing pages pour des PME, entrepreneurs, associations... la demande est forte.
- Spécialisez-vous : Devenez expert des landing pages pour un secteur donné (immobilier, e-commerce, SaaS...).
- Optimisation continue : Proposez des services de test A/B pour améliorer les taux de conversion – une vraie valeur ajoutée pour vos clients.

3. FAIRE UN CONVERTISSEUR PDF → EXCEL VIA INTERFACE WEB

POURQUOI CE PROJET ?

La conversion de documents est un besoin fréquent, notamment en entreprise. Les fichiers PDF sont omniprésents, mais extraire des données structurées (comme des tableaux) pour les analyser dans Excel est souvent compliqué. Ce projet consiste à créer un outil web simple qui automatise cette tâche.

Vous apprendrez à :

- Manipuler les fichiers côté client avec JavaScript,
- Créer une interface web intuitive,
- Développer un backend pour effectuer la conversion avec des outils puissants comme Python ou Node.js.

COMMENT FAIRE ?

Ce projet se divise en deux grandes

parties :

1. L'interface utilisateur (frontend) : HTML, CSS, JavaScript

2. Le moteur de conversion (backend) : Python (ex. avec Flask), Node.js ou PHP

1. INTERFACE UTILISATEUR (FRONTEND)

a) HTML – index.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Convertisseur PDF vers Excel</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Convertisseur PDF vers Excel</h1>
        <p>Téléchargez votre fichier PDF pour le convertir en Excel.</p>
        <input type="file" id="pdfUpload" accept=".pdf">
        <button id="convertBtn">Convertir</button>
        <div id="status"></div>
        <div id="downloadLink"></div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

b) CSS – style.css

```
body {
    font-family: sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: #f4f4f4;
    margin: 0;
}

.container {
    background-color: white;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    text-align: center;
}

input[type="file"] {
    margin-bottom: 20px;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

#status {
    margin-top: 20px;
    font-weight: bold;
}
```

c) JavaScript – script.js

```

document.getElementById('convertBtn').addEventListener('click', async () => {
  const pdfFile = document.getElementById('pdfUpload').files[0];
  const statusDiv = document.getElementById('status');
  const downloadLinkDiv = document.getElementById('downloadLink');

  if (!pdfFile) {
    statusDiv.textContent = 'Veuillez sélectionner un fichier PDF.';
    return;
  }

  statusDiv.textContent = 'Conversion en cours...';
  downloadLinkDiv.innerHTML = '';

  const formData = new FormData();
  formData.append('pdf', pdfFile);

  try {
    const response = await fetch('/upload-pdf', {
      method: 'POST',
      body: formData
    });

    if (response.ok) {
      const result = await response.json();
      statusDiv.textContent = 'Conversion réussie !';
      downloadLinkDiv.innerHTML = `Télécharger le fichier Excel`;
    } else {
      const errorText = await response.text();
      statusDiv.textContent = `Erreur de conversion : ${errorText}`;
    }
  } catch (error) {
    statusDiv.textContent = `Une erreur est survenue : ${error.message}`;
    console.error('Erreur:', error);
  }
});

```

2. Moteur de conversion (Backend)

Exemple de backend Python avec Flask

Ce backend reçoit le fichier PDF, extrait les tableaux avec `pdfplumber`, et génère un fichier Excel avec `openpyxl`.

```
from flask import Flask, request, jsonify, send_file
import pdfplumber
from openpyxl import Workbook
import os

app = Flask(__name__)
UPLOAD_FOLDER = './uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/upload-pdf', methods=['POST'])
def upload_pdf():
    if 'pdf' not in request.files:
        return 'Aucun fichier PDF fourni', 400
    file = request.files['pdf']
    if file.filename == '':
        return 'Nom de fichier invalide', 400

    pdf_path = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(pdf_path)

    excel_filename = file.filename.replace('.pdf', '.xlsx')
    excel_path = os.path.join(UPLOAD_FOLDER, excel_filename)

    try:
        workbook = Workbook()
        # Supprimer la feuille par défaut
        default_sheet = workbook.active
        workbook.remove(default_sheet)

        with pdfplumber.open(pdf_path) as pdf:
            for i, page in enumerate(pdf.pages):
                sheet = workbook.create_sheet(f'Page_{i+1}')
                tables = page.extract_tables()
                for table in tables:
                    for row in table:
                        sheet.append(row)

        workbook.save(excel_path)
        return jsonify({'excelUrl': f'/download-excel/{excel_filename}'})
    except Exception as e:
        return str(e), 500
    finally:
        os.remove(pdf_path)

@app.route('/download-excel/<filename>', methods=['GET'])
def download_excel(filename):
    return send_file(os.path.join(UPLOAD_FOLDER, filename), as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

CONSEILS ET ASTUCES

- Gestion des erreurs : Indiquez clairement quand un fichier est manquant, invalide ou non convertible.
- Sécurité : Supprimez tous les fichiers temporaires. Limitez la taille maximale des fichiers.
- UX fluide : Affichez des messages en temps réel. Une barre de chargement est un vrai plus.
- Scalabilité : Si le service est en ligne, prévoyez un système de file d'attente ou de tâches asynchrones.
- Fonctionnalités avancées :
 - Choisir les pages à convertir
 - Fusionner les tableaux
 - Supporter plusieurs langues

MONÉTISATION

Voici plusieurs manières de monétiser ce projet :

- Freemium : Offrez un nombre limité de conversions gratuites, puis facturez l'accès illimité.
- API commerciale : Fournissez une API REST à d'autres applications (modèle B2B).
- Licence marque blanche : Vendez la plateforme personnalisée à d'autres entreprises.
- Publicité : Affichez des pubs sur la version gratuite (attention à l'expérience utilisateur).

4. AUTOMATISER UN FICHIER EXCEL AVEC VBA (EX : SUIVI DE BUDGET)

Pourquoi ce projet ?

Même si cet ebook est centré sur le web, l'automatisation dans Excel est une compétence très recherchée, notamment dans les PME et les services administratifs. Beaucoup de processus quotidiens sont encore manuels, fastidieux, et facilement automatisables avec VBA (Visual Basic for Applications).

Ce projet vous montre comment transformer un fichier Excel simple en un véritable outil de suivi de budget semi-automatisé, capable d'enregistrer des transactions et de générer des synthèses automatiquement.

COMMENT FAIRE ?

1. Préparer le fichier Excel

- Créez un nouveau classeur Excel.
- Renommez la première feuille en Transactions.
- Créez les colonnes suivantes :
Date | Description | Catégorie | Montant | Type
(Revenu/Dépense)
- Créez une deuxième feuille nommée Synthèse, qui affichera les totaux par catégorie.

2. Activer l'onglet "Développeur"

Si ce n'est pas déjà fait :

- Allez dans Fichier > Options > Personnaliser le ruban.
- Cochez la case Développeur pour l'afficher dans le ruban.

3. Ouvrir l'éditeur VBA

- Raccourci clavier : Alt + F11
- Puis : Insertion > Module → un nouveau module s'ouvre.

4. ÉCRIRE LES MACROS

► Macro 1 : Ajouter une transaction

Elle demande les infos à l'utilisateur, puis les ajoute à la feuilleTransactions.

```
Sub AjouterTransaction()
    Dim wsTransactions As Worksheet
    Set wsTransactions = ThisWorkbook.Sheets("Transactions")

    Dim lastRow As Long
    lastRow = wsTransactions.Cells(wsTransactions.Rows.Count, "A").End(xlUp).Row + 1

    Dim transactionDate As Date
    Dim description As String
    Dim category As String
    Dim amount As Double
    Dim transactionType As String

    On Error GoTo ErrHandler

    transactionDate = InputBox("Entrez la date (JJ/MM/AAAA) :", "Date de la transaction")
    description = InputBox("Entrez la description :", "Description")
    category = InputBox("Entrez la catégorie :", "Catégorie")
    amount = InputBox("Entrez le montant :", "Montant")
    transactionType = InputBox("Entrez le type (Revenu ou Dépense) :", "Type")

    If LCase(transactionType) <> "revenu" And LCase(transactionType) <> "dépense" Then
        MsgBox "Le type doit être 'Revenu' ou 'Dépense'.", vbCritical
        Exit Sub
    End If

    wsTransactions.Cells(lastRow, 1).Value = transactionDate
    wsTransactions.Cells(lastRow, 2).Value = description
    wsTransactions.Cells(lastRow, 3).Value = category
    wsTransactions.Cells(lastRow, 4).Value = amount
    wsTransactions.Cells(lastRow, 5).Value = transactionType

    MsgBox "Transaction ajoutée avec succès !", vbInformation
    Exit Sub

ErrorHandler:
    MsgBox "Erreur lors de la saisie. Veuillez vérifier les formats.", vbCritical
End Sub
```

► Macro 2 : Générer la synthèse par catégorie

Elle calcule les totaux revenus/dépenses par catégorie, avec solde.

```
Sub GenererSynthese()
    Dim wsTransactions As Worksheet
    Dim wsSynthese As Worksheet
    Set wsTransactions = ThisWorkbook.Sheets("Transactions")
    Set wsSynthese = ThisWorkbook.Sheets("Synthèse")

    wsSynthese.Cells.ClearContents

    wsSynthese.Cells(1, 1).Value = "Catégorie"
    wsSynthese.Cells(1, 2).Value = "Total Revenus"
    wsSynthese.Cells(1, 3).Value = "Total Dépenses"
    wsSynthese.Cells(1, 4).Value = "Solde"

    Dim lastRowTransactions As Long
    lastRowTransactions = wsTransactions.Cells(wsTransactions.Rows.Count, "A").End(xlUp).Row

    Dim dataRange As Range
    Set dataRange = wsTransactions.Range("A2:E" & lastRowTransactions)

    Dim dictRevenus As Object
    Dim dictDepenses As Object
    Set dictRevenus = CreateObject("Scripting.Dictionary")
    Set dictDepenses = CreateObject("Scripting.Dictionary")
```

```
Dim cell As Range
For Each cell In dataRange.Columns(3).Cells
    Dim category As String
    category = cell.Value
    Dim amount As Double
    amount = cell.Offset(0, 1).Value
    Dim transactionType As String
    transactionType = cell.Offset(0, 2).Value

    If Not IsEmpty(category) Then
        If LCase(transactionType) = "revenu" Then
            If dictRevenus.Exists(category) Then
                dictRevenus(category) = dictRevenus(category) + amount
            Else
                dictRevenus.Add category, amount
            End If
        ElseIf LCase(transactionType) = "dépense" Then
            If dictDepenses.Exists(category) Then
                dictDepenses(category) = dictDepenses(category) + amount
            Else
                dictDepenses.Add category, amount
            End If
        End If
    End If
Next cell
```

```

Dim key As Variant
For Each key In dictRevenus.Keys
    wsSynthese.Cells(currentRow, 1).Value = key
    wsSynthese.Cells(currentRow, 2).Value = dictRevenus(key)
    If dictDepenses.Exists(key) Then
        wsSynthese.Cells(currentRow, 3).Value = dictDepenses(key)
    Else
        wsSynthese.Cells(currentRow, 3).Value = 0
    End If
    wsSynthese.Cells(currentRow, 4).Value = wsSynthese.Cells(currentRow, 2).Value - wsSynthe
    currentRow = currentRow + 1
Next key

For Each key In dictDepenses.Keys
    If Not dictRevenus.Exists(key) Then
        wsSynthese.Cells(currentRow, 1).Value = key
        wsSynthese.Cells(currentRow, 2).Value = 0
        wsSynthese.Cells(currentRow, 3).Value = dictDepenses(key)
        wsSynthese.Cells(currentRow, 4).Value = -dictDepenses(key)
        currentRow = currentRow + 1
    End If
Next key

wsSynthese.Columns("A:D").AutoFit
MsgBox "Synthèse générée avec succès !", vformation
End Sub

```

5. Ajouter des boutons pour exécuter les macros

- Onglet Développeur > Insérer > Bouton (contrôle de formulaire)
- Sur la feuille Transactions, ajoutez un bouton pour la macro `AjouterTransaction`
- Sur la feuille Synthèse, ajoutez un bouton pour `GenererSynthese`

6. Enregistrer au bon format Enregistrez le fichier en .xlsm (Classeur Excel prenant en charge les macros).

CONSEILS ET ASTUCES

- Utilisez des listes déroulantes dans les colonnes Excel (via validation des données) pour éviter les fautes de frappe.
 - Pour des interfaces plus professionnelles, apprenez à créer des UserForms VBA.
 - Testez toujours votre macro sur une copie du fichier.
 - Commenter son code est indispensable pour s'y retrouver plus tard.
 - Si vous travaillez avec des clients, prévoyez une fonction de sauvegarde automatique.
-

MONÉTISATION

- 💡 Automatisation pour PME : beaucoup utilisent encore Excel manuellement. Vous pouvez les aider à gagner un temps fou.
- 💡 Formations : proposez des sessions de formation en ligne ou en entreprise sur Excel/VBA.
- 💡 Modèles à vendre : créez des fichiers Excel intelligents (gestion de stock, suivi de budget, gestion de planning, etc.) et vendez-les sur Gumroad, Etsy ou via un site perso.
- 💡 Conseil en : identifiez les pertes de temps dans les tâches d'un service, proposez une solution automatisée avec Excel + VBA → valeur directe, facturable.

5. RÉALISER UN MINI E-COMMERCE SANS CMS (JUSTE HTML/CSS/Javascript)

Pourquoi ce projet ?

Créer un site e-commerce de A à Z sans utiliser de CMS comme Shopify ou WooCommerce est un excellent exercice pour comprendre les mécanismes essentiels du commerce en ligne. Ce projet vous permet :

- Gérer des données (produits),
- Déconstruire un panier interactif côté client,
- Simuler un processus de commande.

C'est une compétence qui montre votre capacité à gérer la logique d'une boutique en ligne sans dépendance à une plateforme tierce.

Comment faire ?

Ce mini e-commerce fonctionnera exclusivement côté client. Il ne traitera pas les paiements ni le stock, mais simulera une vraie expérience utilisateur.

1. Structure HTML (index.html)

Votre page principale affichera :

- Une liste de produits (image, nom, prix, description, bouton d'ajout au panier),
- Un lien vers le panier,
- Une section ou page dédiée pour voir les articles ajoutés et le total,
- Un bouton "Passer la commande" (simulation uniquement).

```

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mini E-commerce</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>Ma Boutique Simple</h1>
        <nav>
            <a href="#products">Produits</a>
            <a href="#cart">Panier (<span id="cart-count">0</span>)</a>
        </nav>
    </header>

    <main>
        <section id="products">
            <h2>Nos Produits</h2>
            <div id="product-list" class="grid-container">
                <!-- Produits ajoutés dynamiquement avec JS -->
            </div>
        </section>

        <section id="cart" style="display:none;">
            <h2>Votre Panier</h2>
            <div id="cart-items"></div>
            <p>Total : <span id="cart-total">0.00</span> €</p>
            <button id="checkout-btn">Passer la commande</button>
        </section>
    </main>

    <script src="script.js"></script>
</body>
</html>

```

2. Stylisation CSS (style.css) Voici une base de mise en page responsive avec des styles clairs pour vos produits et le panier.

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: white;
    padding: 1em;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

nav a {
    color: white;
    margin-left: 15px;
    text-decoration: none;
}

main {
    padding: 20px;
}

.grid-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
}

.product-card {
    background-color: white;
    border: 1px solid #ddd;
    padding: 15px;
    text-align: center;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.product-card img {
    max-width: 100%;
    height: auto;
}

.add-to-cart-btn {
    background-color: #28a745;
    color: white;
    padding: 10px 15px;
    border: none;
    cursor: pointer;
    margin-top: 10px;
}

.cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 0;
    border-bottom: 1px solid #eee;
}

#checkout-btn {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    margin-top: 20px;
}

```

3. Logique JavaScript (script.js) Voici

un système de panier dynamique :

- Les produits sont affichés dynamiquement,
- Le panier est géré avec un tableau JavaScript,
- Les articles s'ajoutent, les totaux sont recalculés,
- Le panier est stocké dans localStorage pour persister,
- Un message de confirmation s'affiche à la commande.

```

const products = [
  { id: 1, name: 'T-shirt Cool', price: 25.00, image: 'https://via.placeholder.com/150/FF5733/FF'
  { id: 2, name: 'Casquette Stylée', price: 15.00, image: 'https://via.placeholder.com/150/33FF99'
  { id: 3, name: 'Mug Magique', price: 12.50, image: 'https://via.placeholder.com/150/3357FF/FFF'
];

let cart = JSON.parse(localStorage.getItem('cart')) || [];

const productListDiv = document.getElementById('product-list');
const cartCountSpan = document.getElementById('cart-count');
const cartItemsDiv = document.getElementById('cart-items');
const cartTotalSpan = document.getElementById('cart-total');
const checkoutBtn = document.getElementById('checkout-btn');
const cartSection = document.getElementById('cart');
const productsSection = document.getElementById('products');

function renderProducts() {
  productListDiv.innerHTML = '';
  products.forEach(product => {
    const productCard = document.createElement('div');
    productCard.className = 'product-card';
    productCard.innerHTML = `
      
      <h3>${product.name}</h3>
      <p>${product.price.toFixed(2)}</p>
      <button class="add-to-cart-btn" data-id="${product.id}">Ajouter au panier</button>
    `;
    productListDiv.appendChild(productCard);
  });
}

document.querySelectorAll('.add-to-cart-btn').forEach(button => {
  button.addEventListener('click', addToCart);
});

function addToCart(event) {
  const productId = parseInt(event.target.dataset.id);
  const existingItem = cart.find(item => item.id === productId);

  if (existingItem) {
    existingItem.quantity++;
  } else {
    const product = products.find(p => p.id === productId);
    cart.push({ ...product, quantity: 1 });
  }

  saveCart();
  updateCartDisplay();
}

function saveCart() {
  localStorage.setItem('cart', JSON.stringify(cart));
}

```

```

function updateCartDisplay() {
    cartCountSpan.textContent = cart.reduce((sum, item) => sum + item.quantity, 0);
    cartItemsDiv.innerHTML = '';
    let total = 0;

    if (cart.length === 0) {
        cartItemsDiv.innerHTML = '<p>Votre panier est vide.</p>';
    } else {
        cart.forEach(item => {
            const cartItemDiv = document.createElement('div');
            cartItemDiv.className = 'cart-item';
            cartItemDiv.innerHTML = `
                <span>${item.name} x ${item.quantity}</span>
                <span>${(item.price * item.quantity).toFixed(2)} €</span>
            `;
            cartItemsDiv.appendChild(cartItemDiv);
            total += item.price * item.quantity;
        });
    }

    cartTotalSpan.textContent = total.toFixed(2);
}

function checkout() {
    if (cart.length === 0) {
        alert('Votre panier est vide !');
        return;
    }
    alert('Commande passée avec succès ! Merci pour votre achat.');
    cart = [];
    saveCart();
    updateCartDisplay();
    showSection('products');
}

function showSection(sectionId) {
    productsSection.style.display = 'none';
    cartSection.style.display = 'none';

    if (sectionId === 'products') {
        productsSection.style.display = 'block';
    } else if (sectionId === 'cart') {
        cartSection.style.display = 'block';
    }
}

document.querySelector('nav a[href="#cart"]').addEventListener('click', (e) => {
    e.preventDefault();
    showSection('cart');
});

document.querySelector('nav a[href="#products"]').addEventListener('click', (e) => {
    e.preventDefault();
    showSection('products');
});

checkoutBtn.addEventListener('click', checkout);

renderProducts();
updateCartDisplay();

```

Conseils et Astuces

- Séparation des données : Stocker les produits dans un `.json` est une fichier bonne pratique pour simuler une base de données.
 - `localStorage` : Permet de conserver le panier même après un rafraîchissement ou une fermeture du navigateur.
 - Expérience utilisateur : Affichez des messages comme « Produit ajouté au panier ».
 - Responsivité : Assurez une bonne expérience sur mobile.
 - CSS Framework (optionnel) : Utilisez Bootstrap ou Tailwind pour améliorer rapidement le rendu visuel.
-

Monétisation

- Proposez ce service à des petits vendeurs/artisans qui n'ont pas besoin d'un CMS compliqué.
 - Ajoutez des paiements réels (Stripe, PayPal) une fois les bases maîtrisées.
 - Pack maintenance : mise à jour des produits, gestion des commandes fictives, sécurité.
- Spécialisez-vous dans une niche (ex : boutique pour créateurs, artistes, etc.) pour vous différencier.

6. GÉNÉRER UN CV WEB DYNAMIQUE

Pourquoi ce projet ?

Un CVweb dynamique est plus qu'une version numérique de votre CV papier : c'est une vitrine interactive de vos compétences en développement web. Il permet d'intégrer :

- Animations,
- Portfolio,
- Liens interactifs,
- Graphiques de compétences.

C'est un excellent outil pour se démarquer lors d'une recherche d'emploi, tout en montrant concrètement ce que vous savez faire.

Comment faire ?

Ce projet repose sur la présentation de données (votre profil, expériences, compétences, etc.) via une interface web élégante et dynamique.

1. Structure HTML (index.html) Organisez votre CV en sections :

- En-tête : nom, titre, photo, contacts
- Résumé : petite présentation
- Expériences
- Formations
- Compétences
- Projets
- (Optionnel : langues, centres d'intérêts)

```

<main>
  <section id="summary">
    <h2>À Propos de Moi</h2>
    <p></p>
  </section>

  <section id="experience">
    <h2>Expériences Professionnelles</h2>
  </section>

  <section id="education">
    <h2>Formation</h2>
  </section>

  <section id="skills">
    <h2>Compétences</h2>
  </section>

  <section id="projects">
    <h2>Projets</h2>
  </section>
</main>

<script src="script.js"></script>
</body>
</html>
```

```

<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mon CV Dynamique - [Votre Nom]</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <div class="profile-pic"></div>
    <h1>Votre Nom Prénom</h1>
    <p class="title">Votre Titre / Spécialité</p>
    <div class="contact-info"></div>
  </header>
  <div>
```

2. Stylisation CSS (style.css) Rendez votre CV professionnel, lisible et responsive.

```

body {
  font-family: 'Roboto', sans-serif;
  line-height: 1.6;
  margin: 0;
  padding: 0;
  background-color: #f4f7f6;
  color: #333;
}

header {
  background-color: #2c3e50;
  color: white;
  padding: 40px 20px;
  text-align: center;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.profile-pic {
  width: 150px;
  height: 150px;
  border-radius: 50%;
  background-color: #ecf0f1;
  margin: 0 auto 20px;
  background-image: url('your-photo.jpg');
  background-size: cover;
  background-position: center;
}
```

```
main {
    max-width: 900px;
    margin: 20px auto;
    padding: 20px;
    background-color: white;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

section {
    margin-bottom: 30px;
}

h2 {
    color: #2c3e50;
    border-bottom: 2px solid #3498db;
    padding-bottom: 5px;
    margin-bottom: 20px;
}

.experience-item, .education-item {
    margin-bottom: 15px;
}

.skill-category {
    margin-bottom: 15px;
}

.skill-bar-container {
    background-color: #eee;
    border-radius: 5px;
    height: 10px;
    margin-top: 5px;
}
```



3. JavaScript (script.js) Stockez vos données dans un objet JavaScript, puis utilisez le DOM pour générer dynamiquement les sections.

```
const cvData = {
    name: "Votre Nom Prénom",
    title: "Développeur Web Fullstack",
    contact: {
        email: "votre.email@example.com",
        phone: "+33 6 12 34 56 78",
        linkedin: "https://linkedin.com/in/votreprofil",
        github: "https://github.com/votregithub"
    },
    summary: "Passionné par le développement web, je crée des applications robustes et intuitives.",
    experience: [
        {
            title: "Développeur Front-end",
            company: "Tech Solutions Inc.",
            dates: "Jan 2023 - Présent",
            description: [
                "Interfaces avec React/Redux",
                "Intégration d'APIs REST",
                "Optimisation UX/UI"
            ]
        },
        {
            title: "Stagiaire Développeur Web",
            company: "Startup Innovante",
            dates: "Juin 2022 - Déc 2022",
            description: [
                "Site e-commerce en HTML/CSS/JS",
                "Responsive design"
            ]
        }
    ],
}
```



```

        education: [
          {
            degree: "Licence Pro Métiers du Web",
            institution: "Université de la Tech",
            dates: "2020 - 2023"
          },
          {
            degree: "DUT Informatique",
            institution: "IUT du Code",
            dates: "2018 - 2020"
          }
        ],
        skills: {
          "Langages": [
            { name: "JavaScript", level: 90 },
            { name: "HTML5", level: 95 },
            { name: "CSS3", level: 90 },
            { name: "Python", level: 70 }
          ],
          "Frameworks": [
            { name: "React", level: 85 },
            { name: "Node.js", level: 75 }
          ],
          "Outils": [
            { name: "Git", level: 85 },
            { name: "VS Code", level: 90 }
          ]
        },
      ],
      projects: [
        {
          name: "Convertisseur PDF-Excel",
          description: "Outil web pour extraire des tableaux PDF vers Excel.",
          link: "#",
          technologies: "HTML, CSS, JavaScript, Flask"
        }
      ]
    };

    function renderCV() {
      document.querySelector('header h1').textContent = cvData.name;
      document.querySelector('header p.title').textContent = cvData.title;

      const contactDiv = document.querySelector('.contact-info');
      contactDiv.innerHTML = `
        <p>Email : <a href="mailto:${cvData.contact.email}">${cvData.contact.email}</a></p>
        <p>Téléphone : ${cvData.contact.phone}</p>
        <p>LinkedIn : <a href="${cvData.contact.linkedin}">${cvData.contact.linkedin}</a></p>
        <p>GitHub : <a href="${cvData.contact.github}">${cvData.contact.github}</a></p>
      `;

      const expSection = document.getElementById('experience');
      cvData.experience.forEach(exp => {
        const div = document.createElement('div');
        div.className = 'experience-item';
        div.innerHTML = `
          <h3>${exp.title} chez ${exp.company}</h3>
          <p>${exp.dates}</p>
          <ul>${exp.description.map(d => `<li>${d}</li>`).join('')}</ul>
        `;
        expSection.appendChild(div);
      });

      const eduSection = document.getElementById('education');
      cvData.education.forEach(edu => {
        const div = document.createElement('div');
        div.className = 'education-item';
        div.innerHTML = `<h3>${edu.degree}</h3><p>${edu.institution} (${edu.dates})</p>`;
        eduSection.appendChild(div);
      });
    }
  
```

```

const skillsSection = document.getElementById('skills');
for (const cat in cvData.skills) {
    const catDiv = document.createElement('div');
    catDiv.className = 'skill-category';
    catDiv.innerHTML = `

#### ${cat}

`;
    cvData.skills[cat].forEach(skill => {
        catDiv.innerHTML += `
            <p>${skill.name}</p>
            <div class="skill-bar-container">
                <div class="skill-bar" style="width:${skill.level}%"></div>
            </div>
        `;
    });
    skillsSection.appendChild(catDiv);
}

const projSection = document.getElementById('projects');
cvData.projects.forEach(proj => {
    const div = document.createElement('div');
    div.className = 'project-item';
    div.innerHTML = `
        <h3><a href="${proj.link}" target="_blank">${proj.name}</a></h3>
        <p>${proj.description}</p>
        <p><strong>Technologies :</strong> ${proj.technologies}</p>
    `;
    projSection.appendChild(div);
});
}

document.addEventListener('DOMContentLoaded', renderCV);

```

CONSEILS & ASTUCES

- Hébergement : utilisez GitHub Pages, Netlify ou Vercel pour mettre votre CV en ligne gratuitement.
 - Mode impression : ajoutez une version imprimable avec @media print dans votre CSS.
 - Accessibilité : balises sémantiques, contrastes élevés, navigation clavier.
 - Évolution : vous pouvez facilement créer une version multilingue, ajouter un filtre pour les projets, ou intégrer des animations au scroll.
-

MONÉTISATION

- Vous faire embaucher : votre CV web peut faire la différence lors d'une candidature.

- Création de CV pour d'autres : proposez vos services à des freelances, étudiants ou professionnels.
- Templates payants : développez et vendez des modèles de CV dynamiques.
- Consulting Personal Branding : combinez design, stratégie LinkedIn, et CV web pour proposer des offres complètes.

:

7. CRÉER UN TABLEAU DE BORD INTERACTIF EN JAVASCRIPT

Pourquoi ce projet ?

Les tableaux de bord (ou Dashboard) permettent de transformer des données brutes en informations visuelles claires et utiles. Que ce soit en entreprise (finance, marketing, logistique...) ou dans des projets personnels, la capacité à construire des visualisations dynamiques et lisibles est une compétence très recherchée.

Ce projet vous apprendra à créer un tableau de bord interactif, en utilisant HTML,

CSS,

JavaScript, et une bibliothèque de graphiques (Chart.js). Vous apprendrez à manipuler dynamiquement le DOM, à gérer des données simulées, et à créer des interfaces visuelles

modernes et adaptables.

Comment faire ?

1. Créer la structure HTML (index.html)

On va concevoir une interface

contenant :

- Des boutons filtres (Jour, Semaine, Mois, Année),
- Des indicateurs clés (KPIs) sous forme de cartes,
- Deux graphiques dynamiques affichés avec Chart.js.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Tableau de Bord Interactif</title>
    <link rel="stylesheet" href="style.css" />
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <header>
        <h1>Mon Tableau de Bord</h1>
    </header>

    <main>
        <section class="filters">
            <button data-timeframe="day" class="active">Jour</button>
            <button data-timeframe="week">Semaine</button>
            <button data-timeframe="month">Mois</button>
            <button data-timeframe="year">Année</button>
        </section>
    </main>
</body>
```

```

<section class="kpis">
    <div class="kpi-card"><h3>Ventes Totales</h3><p id="total-sales">0</p></div>
    <div class="kpi-card"><h3>Utilisateurs Actifs</h3><p id="active-users">0</p></div>
    <div class="kpi-card"><h3>Revenu Moyen</h3><p id="average-revenue">0 €</p></div>
</section>

<section class="charts">
    <div class="chart-container"><canvas id="salesChart"></canvas></div>
    <div class="chart-container"><canvas id="usersChart"></canvas></div>
</section>
</main>

<script src="script.js"></script>
</body>
</html>

```

2. Styliser avec CSS (style.css)

Objectifs :

- Interface moderne et épurée
- Mise en page responsive
- Mise en valeur des indicateurs et graphiques

```

body {
    font-family: 'Segoe UI', sans-serif;
    margin: 0;
    background-color: #f0f2f5;
    color: #333;
}

header {
    background-color: #2c3e50;
    color: white;
    padding: 20px;
    text-align: center;
}

main {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}

.filters {
    display: flex;
    justify-content: center;
    gap: 10px;
    margin-bottom: 20px;
}

```

```

.filters button {
    padding: 10px 20px;
    border: none;
    background-color: #e0e0e0;
    border-radius: 5px;
    cursor: pointer;
}

.filters button.active,
.filters button:hover {
    background-color: #3498db;
    color: white;
}

.kpis {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

```

```

.kpi-card {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.05);
    text-align: center;
}

.kpi-card h3 {
    margin: 0;
    color: #666;
}

.kpi-card p {
    font-size: 2.5em;
    font-weight: bold;
    margin: 10px 0 0;
    color: #3498db;
}

.charts {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(400px, 1fr));
    gap: 20px;
}

.chart-container {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.05);
}

```

3. Ajouter la logique JavaScript (script.js)

On va:

- Définir des données fictives
- Mettre à jour les KPIs et les graphiques
- Gérer les événements des filtres

```

let salesChart, usersChart;                                ⌂ Copier ⌂ Ma

function updateKPIs(kpis) {
    document.getElementById('total-sales').textContent = kpis.totalSales.toLocaleString();
    document.getElementById('active-users').textContent = kpis.activeUsers.toLocaleString();
    document.getElementById('average-revenue').textContent = kpis.averageRevenue.toFixed(2)
}

function renderCharts(timeframeData) {
    const salesCtx = document.getElementById('salesChart').getContext('2d');
    const usersCtx = document.getElementById('usersChart').getContext('2d');

    if (salesChart) salesChart.destroy();
    if (usersChart) usersChart.destroy();

    salesChart = new Chart(salesCtx, {
        type: 'bar',
        data: {
            labels: timeframeData.labels,
            datasets: [{
                label: 'Ventes',
                data: timeframeData.sales,
                backgroundColor: 'rgba(54, 162, 235, 0.6)',
                borderColor: 'rgba(54, 162, 235, 1)',
                borderWidth: 1
            }]
        },
        options: { responsive: true, scales: { y: { beginAtZero: true } } }
    });
}

```

```
const data = {
  day: {
    labels: ['0h', '3h', '6h', '9h', '12h', '15h', '18h', '21h'],
    sales: [10, 15, 20, 30, 45, 40, 35, 25],
    users: [5, 8, 12, 18, 25, 22, 19, 15],
    kpis: { totalSales: 2300, activeUsers: 150, averageRevenue: 25.5 }
  },
  week: {
    labels: ['Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam', 'Dim'],
    sales: [120, 150, 130, 180, 200, 250, 100],
    users: [80, 90, 85, 110, 120, 150, 70],
    kpis: { totalSales: 12300, activeUsers: 850, averageRevenue: 30.2 }
  },
  month: {
    labels: ['Sem 1', 'Sem 2', 'Sem 3', 'Sem 4'],
    sales: [500, 600, 550, 700],
    users: [300, 350, 320, 400],
    kpis: { totalSales: 23500, activeUsers: 1800, averageRevenue: 28.9 }
  },
  year: {
    labels: ['Jan', 'Fév', 'Mar', 'Avr', 'Mai', 'Juin', 'Juill', 'Août', 'Sep', 'Oct', 'Nov', 'Déc'],
    sales: [2000, 2200, 2500, 2300, 2800, 3000, 2700, 2900, 3100, 3300, 3500, 3800],
    users: [1200, 1300, 1500, 1400, 1600, 1800, 1700, 1900, 2000, 2100, 2200, 2400],
    kpis: { totalSales: 350000, activeUsers: 25000, averageRevenue: 32.1 }
  }
};
```

```
usersChart = new Chart(usersCtx, {
  type: 'line',
  data: {
    labels: timeframeData.labels,
    datasets: [{
      label: 'Utilisateurs',
      data: timeframeData.users,
      borderColor: 'rgba(75, 192, 192, 1)',
      borderWidth: 2,
      fill: false
    }]
  },
  options: { responsive: true, scales: { y: { beginAtZero: true } } }
});
```

```

function updateDashboard(timeframe) {
    const timeframeData = data[timeframe];
    updateKPIs(timeframeData.kpis);
    renderCharts(timeframeData);

    document.querySelectorAll('.filters button').forEach(btn => btn.classList.remove('active'));
    document.querySelector(`.filters button[data-timeframe="${timeframe}"]`).classList.add('active')
}

document.querySelectorAll('.filters button').forEach(button => {
    button.addEventListener('click', () => {
        const timeframe = button.dataset.timeframe;
        updateDashboard(timeframe);
    });
});

document.addEventListener('DOMContentLoaded', () => {
    updateDashboard('day');
});

```

Conseils et Astuces

- 🌟 Chart.js est simple et puissant. Pour plus de personnalisation, [regardez aussi ApexCharts ou D3.js.](#)
- 🌟 Pour lier à de vraies données, utilisez `fetch()` ou une API back-end (Node.js, PHP, Firebase...).
- 🌟 Évitez de surcharger les graphiques : gardez-les simples et visibles.
- 🌟 Pensez à ajouter un mode sombre pour plus de professionnalisme.

Monétisation

-  Création de dashboards sur mesure pour des clients (TPE, start-ups, PME, consultants, RH...).
-  Connexion à des APIs populaires : Google Analytics, Stripe, Notion, etc.
-  Vente de templates : proposez des modèles de dashboards adaptables sur des marketplaces.
-  Conseil en visualisation de données : montrez à vos clients comment mesurer ce qui compte vraiment (KPI pertinents).

8. CRÉER UN FORMULAIRE INTELLIGENT AVEC VALIDATIONS

Pourquoi ce projet ?

Sur le web, les formulaires sont omniprésents : inscription, commande, contact, newsletter, connexion... Pourtant, la majorité sont mal optimisés. Un bon formulaire, bien conçu, offre une expérience utilisateur agréable, réduit les erreurs et augmente les conversions.

Ce projet vous apprendra à créer un formulaire d'inscription moderne, avec des validations en temps réel en JavaScript. Vous y apprendrez à gérer les erreurs, à afficher des retours utilisateurs, et à rendre vos formulaires plus accessibles, plus fiables et plus intuitifs.

COMMENT FAIRE ?

1. Construire la structure HTML On crée un formulaire d'inscription avec quatre champs :

- Nom d'utilisateur
- Adresse email
- Mot de passe
- Confirmation du mot de passe

Chaque champ est accompagné d'un message d'erreur personnalisé.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulaire d'Inscription Intelligent</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="form-container">
        <h1>Inscription</h1>
        <form id="registrationForm">
            <div class="form-group">
                <label for="username">Nom d'utilisateur :</label>
                <input type="text" id="username" name="username" placeholder="Choisissez un nom" />
                <span class="error-message" id="usernameError"></span>
            </div>

            <div class="form-group">
                <label for="email">Email :</label>
                <input type="email" id="email" name="email" placeholder="Votre adresse email" />
                <span class="error-message" id="emailError"></span>
            </div>

            <div class="form-group">
                <label for="password">Mot de passe :</label>
                <input type="password" id="password" name="password" placeholder="Minimum 8 caractères" />
                <span class="error-message" id="passwordError"></span>
            </div>

            <div class="form-group">
                <label for="confirmPassword">Confirmer le mot de passe :</label>
                <input type="password" id="confirmPassword" name="confirmPassword" placeholder="Confirmez votre mot de passe" />
                <span class="error-message" id="confirmPasswordError"></span>
            </div>

            <button type="submit">S'inscrire</button>
        </form>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

2. Styliser avec CSS

Un design propre, lisible, avec mise en évidence des champs valides et invalides.

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
}

.form-container {
    background-color: white;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    width: 100%;
    max-width: 400px;
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
}

.form-group {
    margin-bottom: 20px;
}

label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    color: #555;
}

input[type="text"],
input[type="email"],
input[type="password"] {
    width: calc(100% - 20px);
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 1em;
}

input:focus {
    outline: none;
    border-color: #007bff;
    box-shadow: 0 0 0 0.2rem rgba(0, 123, 255, 0.25);
}

.error-message {
    color: #dc3545;
    font-size: 0.85em;
    margin-top: 5px;
    display: block;
}

input.valid {
    border-color: #28a745;
}

button[type="submit"] {
    width: 100%;
    padding: 12px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    font-size: 1.1em;
    cursor: pointer;
}

button[type="submit"]:hover {
    background-color: #0056b3;
}
```

3. Ajouter l'intelligence avec JavaScript Validation dynamique et visuelle des champs du formulaire.

```
const form = document.getElementById('registrationForm');
const usernameInput = document.getElementById('username');
const emailInput = document.getElementById('email');
const passwordInput = document.getElementById('password');
const confirmPasswordInput = document.getElementById('confirmPassword');

const usernameError = document.getElementById('usernameError');
const emailError = document.getElementById('emailError');
const passwordError = document.getElementById('passwordError');
const confirmPasswordError = document.getElementById('confirmPasswordError');

function displayError(input, errorSpan, message) {
    input.classList.add('invalid');
    input.classList.remove('valid');
    errorSpan.textContent = message;
}

function displaySuccess(input, errorSpan) {
    input.classList.remove('invalid');
    input.classList.add('valid');
    errorSpan.textContent = '';
}

function validateUsername() {
    const value = usernameInput.value.trim();
    if (value.length < 3) {
        displayError(usernameInput, usernameError, "Minimum 3 caractères.");
        return false;
    }
    displaySuccess(usernameInput, usernameError);
    return true;
}

function validateEmail() {
    const value = emailInput.value.trim();
    const regex = /^[^@\s]+@[^\s]+\.[^\s]+$/;
    if (!regex.test(value)) {
        displayError(emailInput, emailError, "Email invalide.");
        return false;
    }
    displaySuccess(emailInput, emailError);
    return true;
}

function validatePassword() {
    const value = passwordInput.value;
    const regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[\W_]).{8,}$/;
    if (!regex.test(value)) {
        displayError(passwordInput, passwordError, "8+ caractères, majuscule, minuscule, chiffre");
        return false;
    }
}

function validateConfirmPassword() {
    if (confirmPasswordInput.value !== passwordInput.value) {
        displayError(confirmPasswordInput, confirmPasswordError, "Les mots de passe ne correspondent pas");
        return false;
    }
    displaySuccess(confirmPasswordInput, confirmPasswordError);
    return true;
}

usernameInput.addEventListener('input', validateUsername);
emailInput.addEventListener('input', validateEmail);
passwordInput.addEventListener('input', validatePassword);
confirmPasswordInput.addEventListener('input', validateConfirmPassword);

form.addEventListener('submit', function (e) {
    e.preventDefault();
    const valid =
        validateUsername() &&
        validateEmail() &&
        validatePassword() &&
        validateConfirmPassword();

    if (valid) {
        alert("Formulaire soumis avec succès !");
        form.reset();
        [usernameInput, emailInput, passwordInput, confirmPasswordInput].forEach(input => {
            input.classList.remove('valid', 'invalid');
        });
    } else {
        alert("Veuillez corriger les erreurs.");
    }
});
```

Conseils et Astuces

- Validez toujours aussi côté serveur, même si la validation JS est déjà faite (sécurité).
- Rendez l'expérience fluide : validez en temps réel, donnez des conseils utiles.
- Feedback visuel clair : couleurs, icônes, animations douces si possible.
- Accessibilité : utilisez aria-describedby pour relier les messages d'erreur aux champs.
- Extensions possibles : formulaire multi-étapes, ajout de règles conditionnelles, captcha.

Monétisation

- Création de formulaires personnalisés pour les entreprises (devis, candidatures, événements).
- Optimisation UX/UI de formulaires existants (souvent négligés, mais critiques pour les conversions).
- Intégration avec des outils tiers : Mailchimp, Airtable, CRM, Google Sheets...
- Modules et plugins de formulaire pour CMS (WordPress, Shopify, Webflow).
- Formations / tutoriels pour débutants ou clients non-tech souhaitant gérer leurs formulaires seuls.

9. INTÉGRER UNE API MÉTÉO, CRYPTO OU AUTRE DANS UNE APP SIMPLE

Pourquoi ce projet ?

L'intégration d'API (Application Programming Interface) est une compétence clé pour tout développeur web moderne. Grâce aux APIs, vous pouvez connecter votre application à des services tiers pour obtenir des données en temps réel : météo, taux de change, cryptomonnaies, géolocalisation, actualités, etc.

Ce projet vous permettra d'apprendre à :

- Envoyer une requête HTTP avec JavaScript (`fetch`),
 - Traiter une réponse JSON,
 - Afficher dynamiquement des données dans votre page,
 - Et gérer les erreurs proprement.
-

Objectif du projet

Créer une application météo simple où l'utilisateur saisit le nom d'une ville et obtient les informations météo en direct (température, description, icône, etc.).

Étapes de réalisation

1. Obtenir une clé API gratuite Allez sur <https://openweathermap.org/> ou <https://weatherapi.com>, créez un compte gratuit et récupérez une clé API personnelle. Cette clé est nécessaire pour interroger l'API.

2. Structure HTML de l'application On prépare une interface avec un champ de recherche, un bouton et une section pour afficher les résultats météo.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Application Météo</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="weather-app">
        <h1>Météo en Direct</h1>
        <div class="search-box">
            <input type="text" id="city-input" placeholder="Entrez une ville...">
            <button id="search-btn">Rechercher</button>
        </div>
        <div class="weather-info" style="display:none;">
            <h2 id="city-name"></h2>
            <p id="temperature"></p>
            <p id="description"></p>
            <img id="weather-icon" src="" alt="Icône météo">
        </div>
        <p id="error-message" class="error-message"></p>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

3. Mise en forme CSS Un design simple et moderne, avec un fond coloré et une boîte centrée.

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(to right, #6a11cb, #2575fc);
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
}

.weather-app {
    background-color: rgba(0, 0, 0, 0.3);
    padding: 30px;
    border-radius: 15px;
    text-align: center;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
    max-width: 400px;
    width: 100%;
}

.search-box {
    display: flex;
    margin-bottom: 20px;
}

#city-input {
    padding: 10px;
    border: none;
    border-radius: 5px 0 0 5px;
    width: 70%;
```



```
.search-box {
    display: flex;
    margin-bottom: 20px;
}

#city-input {
    padding: 10px;
    border: none;
    border-radius: 5px 0 0 5px;
    width: 70%;
```



```
#search-btn {
    padding: 10px 15px;
    border: none;
    background-color: #28a745;
    color: white;
    border-radius: 0 5px 5px 0;
    cursor: pointer;
}

#search-btn:hover {
    background-color: #21883b;
```



```
.weather-info {
    margin-top: 20px;
}
```



```
#temperature {
    font-size: 3em;
    margin: 0;
}

#description {
    font-size: 1.5em;
}

.error-message {
    color: #ffddcc;
    margin-top: 20px;
    font-weight: bold;
}
```

4. Logique JavaScript : appel de l'API Voici le script JS principal, avec gestion d'erreur, conversion JSON, et affichage dynamique.

```
const API_KEY = 'VOTRE_CLE_API'; // Remplacez par votre clé OpenWeatherMap
const searchBtn = document.getElementById('search-btn');
const cityInput = document.getElementById('city-input');
const cityNameElem = document.getElementById('city-name');
const temperatureElem = document.getElementById('temperature');
const descriptionElem = document.getElementById('description');
const weatherIconElem = document.getElementById('weather-icon');
const weatherInfoDiv = document.querySelector('.weather-info');
const errorMessageElem = document.getElementById('error-message');

searchBtn.addEventListener('click', fetchWeather);
cityInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter') {
        fetchWeather();
    }
});

async function fetchWeather() {
    const city = cityInput.value.trim();
    if (city === '') {
        displayError("Veuillez entrer une ville.");
        return;
    }
}

weatherInfoDiv.style.display = 'none';
errorMessageElem.textContent = '';

const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}&units=metric`;

try {
    const response = await fetch(url);
    if (!response.ok) {
        if (response.status === 404) {
            displayError("Ville non trouvée.");
        } else {
            throw new Error(`Erreur: ${response.status}`);
        }
        return;
    }
    const data = await response.json();
    displayWeather(data);
} catch (error) {
    displayError("Erreur réseau : " + error.message);
}
}
```

```

function displayWeather(data) {
    cityNameElem.textContent = data.name;
    temperatureElem.textContent = `${Math.round(data.main.temp)}°C`;
    descriptionElem.textContent = data.weather[0].description;
    weatherIconElem.src = `https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`;
    weatherInfoDiv.style.display = 'block';
}

function displayError(message) {
    errorMessageElem.textContent = message;
    weatherInfoDiv.style.display = 'none';
}

// Charger Paris par défaut
document.addEventListener('DOMContentLoaded', () => {
    cityInput.value = 'Paris';
    fetchWeather();
});

```

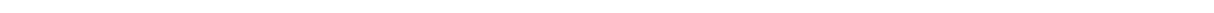
Conseils et Astuces

- ☀ Sécurité : ne jamais exposer votre clé API dans un site public en production.
Utilisez un backend pour cacher la clé.
- ☀ Comprendre JSON : explorez la structure des réponses dans l'onglet Réseau de votre navigateur.
- ☀ Gestion des erreurs : affichez un message clair si la ville est inconnue ou s'il y a un problème de connexion.
- ☀ Multi-API : une fois cette base maîtrisée, testez d'autres APIs : crypto, livres, YouTube, recettes, etc.
- ☀ Responsive : votre application doit être utilisable sur mobile dès le début.

Monétisation

- ☀ Apps utilitaires : développez de petites applications pratiques autour d'APIs publiques (convertisseur, météo, crypto).
- ☀ Intégration client : ajoutez des widgets API dans les sites de vos clients (ex : météo, finance, actualités).

- 📊 Dashboard data : affichez des KPIs en direct dans un back-office client.
- 📊 Plugins : créez des plugins WordPress ou Shopify utilisant des APIs populaires.
- 📊 Formation : vendez des tutos, formations ou ebooks expliquant comment intégrer une API dans différents contextes.



10. OFFRIR UN SCRIPT “PRÊT-À-VENDRE” POUR DES PETITS BESOINS CLIENTS

Pourquoi ce projet ?

De nombreux petits entrepreneurs, associations ou indépendants ont des besoins numériques simples mais récurrents. Ils ne souhaitent pas forcément investir dans un développement sur mesure long et coûteux. C'est là que les scripts prêts-à-l'emploi deviennent très utiles : ce sont de petits outils autonomes, faciles à intégrer et à réutiliser, que vous pouvez adapter et vendre à plusieurs clients.

Ce projet vous apprend à :

- Identifier un besoin courant dans le monde réel,
 - Créer une solution simple, autonome et réutilisable,
 - La packager proprement pour la revente.
-

Exemple concret : un système de témoignages clients

Problème : beaucoup de petites entreprises souhaitent afficher des témoignages clients sur leur site web, mais n'ont pas de CMS ou de back-office.

Solution : un simple script JavaScript autonome, basé sur un fichier JSON

modifiable, qui affiche dynamiquement des témoignages dans une section HTML.

ÉTAPES DU PROJET

1. Définir la structure des données Vous fournissez un fichier JSON contenant les témoignages. Voici un exemple :

```
[  
  {  
    "author": "Alice Dupont",  
    "company": "Boulangerie du Coin",  
    "text": "Un service exceptionnel et des résultats qui dépassent nos attentes !",  
    "rating": 5  
  },  
  {  
    "author": "Bob Martin",  
    "company": "Plomberie Express",  
    "text": "Le convertisseur PDF-Excel nous a fait gagner des heures !",  
    "rating": 4  
  }  
]
```

Le client pourra simplement éditer ce fichier avec un éditeur de texte pour ajouter ou modifier ses témoignages.

2. Intégration HTML pour le client Sur son site, le client n'a qu'à intégrer une section comme ceci :

```
<section id="testimonials-section">  
  <h2>Ce que nos clients disent de nous</h2>  
  <div id="testimonials-container"></div>  
</section>  
  
<script src="testimonial-script.js"></script>
```

3. Script JavaScript intelligent Le script lit le fichier JSON, crée dynamiquement les cartes témoignages avec le nom, l'avis, l'entreprise et la notation par étoiles, et les injecte dans la page.

4. Stylisation CSS simple Fournissez un petit fichier CSS qui stylise les cartes, les étoiles, et la disposition responsive.
Le client peut le personnaliser facilement selon la charte de son site.

CE QUE CONTIENT LE "PACK PRÊT-À-VENDRE"

- Le fichier JavaScript (testimonial-script.js)
 - Un exemple de fichier JSON (testimonials.json)
 - Un fichier CSS (testimonial-styles.css)
 - Une documentation PDF ou Markdown expliquant :
 - Comment intégrer le script
 - Comment modifier les témoignages
 - Comment personnaliser les couleurs ou la mise en page
-

CONSEILS & ASTUCES

- Gardez-le simple : ne forcez pas le client à comprendre le JavaScript. Tout doit être plug & play.
 - Évitez les dépendances : pas de framework requis, juste du HTML/CSS/JS pur.
 - Soyez générique : évitez de coder en dur des couleurs ou des noms. Utilisez des classes CSS faciles à surcharger.
 - Ajoutez des options : nombre de témoignages à afficher, rotation automatique, tri aléatoire...
 - Fournissez un exemple complet : créez une page de démo pour prouver que le script fonctionne dès la première utilisation.
-

IDÉES DE SCRIPTS À REPRODUIRE

- Affichage de l'heure ou de la météo locale
 - Galeries d'images responsives
 - Modules de contact AJAX sans back-end
 - Convertisseurs d'unités
 - Outils de comparaison de produits
 - Tableaux de prix dynamiques
-

MONÉTISATION

- Vente à l'unité : via votre propre site, Gumroad, Payhip, CodeCanyon...
 - Pack de scripts : regroupez plusieurs scripts utiles dans un "bundle" à prix réduit.
 - Service clé en main : facturez l'intégration du script + personnalisation (logos, couleurs...).
 - Offre sur abonnement : proposez un accès illimité à une bibliothèque de scripts avec mises à jour.
 - Produits dérivés : adaptez vos scripts sous forme de widgets WordPress, Shopify ou Webflow.
-

POURQUOI CE PROJET EST MALIN

- Il crée de la valeur immédiatement pour le client.
- Il est facile à dupliquer sans coût supplémentaire.
- Il montre que vous savez résoudre un problème réel avec une solution technique propre, maintenable et professionnelle.

CONCLUSION : BONUS

Félicitations !

Vous êtes arrivé au bout de ce guide. Si vous avez suivi, expérimenté ou même simplement réfléchi à chacun des projets proposés, vous avez déjà posé les bases solides de votre progression dans le monde du développement web.

"Petits Projets Web à Grands Effets" n'est pas seulement un catalogue d'idées. C'est une approche pragmatique de l'apprentissage : apprendre en faisant, comprendre en construisant, progresser en créant de la valeur.

Ce que vous avez touché du doigt

Chaque projet vous a permis d'explorer concrètement :

- La structure : HTML, balisage sémantique, accessibilité,
 - Le style : CSS, responsive design, expérience visuelle,
 - L'interactivité : JavaScript, événements, DOM dynamique,
 - La logique métier : validations, automatisations simples,
 - La connexion aux données : APIs, JSON, stockage local ou distant.
-

Votre tremplin vers l'indépendance professionnelle

L'objectif de ces projets n'était pas juste d'apprendre à coder, mais de vous montrer comment ces compétences peuvent générer de la valeur dans le monde réel. Freelance, side projects, emploi, consulting... les opportunités sont nombreuses si vous savez :

1. Observer les besoins : Soyez à l'écoute. Les problèmes simples mal résolus sont des opportunités d'or.
 2. Créer de la valeur concrète : Chaque ligne de code doit améliorer une situation. Gagnez du temps, simplifiez des tâches, améliorez une expérience.
 3. Montrer votre travail : Un projet visible est un projet qui peut convaincre. Partagez vos réalisations, documentez-les.
 4. Continuer d'apprendre : Le développement web change vite. Soyez toujours en mouvement, testez de nouveaux outils, sortez des sentiers battus.
-

ET MAINTENANT, QUE FAIRE ?

Voici quelques pistes pour aller plus loin

- ●  Solidifiez vos fondamentaux : maîtrisez parfaitement JavaScript pur, les principes CSS avancés, et les bases du responsive design.
-  Apprenez un framework : React, Vue, ou Angular côté frontend. Express, Django, ou Laravel côté backend.
-  Explorez les bases de données : MySQL, PostgreSQL, MongoDB... Comprendre les modèles relationnels et NoSQL est crucial.
-  Déployez vos projets : Netlify, Vercel, Heroku, ou même Docker sur un VPS. Mettez vos applications en ligne.
-  Utilisez Git/GitHub : versionnez tout. C'est indispensable pour collaborer, documenter, et montrer votre code proprement.
-  Tissez un réseau : forums, meetups, groupes Discord, LinkedIn. Parlez de vos projets, échangez avec d'autres développeurs, trouvez des mentors ou des collaborateurs.

LE MOT DE LA FIN

Ce livre est le point de départ, pas la destination. Continuez à construire des choses. Même simples. Même imparfaites. Parce que chaque projet vous rend plus compétent, plus confiant, et plus visible. Le web est un terrain de jeu infini pour qui veut apprendre et entreprendre.

Vous avez désormais les premières clés. Ouvrez les bonnes portes.

Bonne chance, et surtout : codez avec plaisir.

