TP Template / HTML / CSS/ manipulation du DOM :

Client : Agence marketing produit tech (ex : casque audio, drone)

Projet : Landing page interactive pour le lancement d’un produit

Contexte

Le client souhaite lancer une **landing page interactive** pour promouvoir un nouveau produit tech.

L'objectif est de :

- **donner envie aux utilisateurs** grâce à une présentation dynamique,
- **faciliter leur compréhension** du produit,
- et **augmenter les conversions** en les incitant à nous contacter via un formulaire.

Le site est **full front-end** : pas de back-end prévu dans ce projet.

Le site sera ensuite intégré dans un CMS ou une plateforme marketing.

Le client souhaite que chaque partie de la landing page soit **vivante, animée et moderne**.

Livrables attendus

- Une page **HTML complète** respectant les standards d’accessibilité.
- Une **organisation modulaire** du code JS : un fichier par fonctionnalité.
- Un design responsive (au moins desktop + tablette) — pas besoin de mobile parfait pour cette phase.

- Aucune dépendance JS externe (pas de jQuery, React, etc.)
- Tous les comportements doivent être gérés en Vanilla JS.

Structure générale de la page

Sections attendues :

1. **Espace avis clients dynamiques**
2. **Simulateur de prix interactif**
3. **FAQ accordéon**
4. **Formulaire de contact**

Ressources :

- **JavaScript.info – Templates et DOM :** https://javascript.info/templates
- **W3Schools – HTML Templates** :https://www.w3schools.com/html/html_templates.asp
- Chart.js Official : https://www.chartjs.org/docs/latest/getting-started/
- **Chart.js Tutorial (W3Schools) :** https://www.w3schools.com/js/js_graphics_chartjs.asp
- **JavaScript.info – Chart.js :** https://javascript.info/chartjs

---

Répartition par feature

Chaque étudiant devra prendre en charge **une feature complète** :

Feature 1 : Espace Avis Clients dynamique - moyen

Objectifs

- Permettre aux visiteurs de voir les derniers avis clients.

- Ajouter un avis via un mini-formulaire.
- Les avis doivent être affichés immédiatement sans recharger la page.

Détails attendus

- Section contenant un `<template>` d'avis.
- Les avis existants sont stockés dans un tableau JS et injectés dynamiquement au chargement.
- Lorsqu'un avis est ajouté, il est inséré en haut de la liste.
- Possibilité de supprimer un avis (bouton supprimer).

Bonus possibles

- Animation de l'ajout/suppression d'un avis.
- Filtre des avis par note (ex : voir que les 5 étoiles).
- Les avis sont récupérés sous la forme d'un Lorem Ipsum depuis Faker.js

---

Feature 2 : Simulateur de prix interactif - moyen/ difficile

Objectif :

Proposer un configurateur qui permet de choisir plusieurs options et de voir le prix s'ajuster en temps réel.

Fonctionnalités attendues :

- Choix de la couleur (via un `select` ou des boutons).
- Choix de la capacité ou de la version du produit (ex : 64Go, 128Go, Premium...).
- Choix d'options supplémentaires (ex : garantie étendue, accessoires).
- Affichage en temps réel du **prix total** en fonction des options choisies.

Contraintes techniques :

- Toutes les interactions doivent déclencher une mise à jour dynamique du prix dans le DOM.
- Les prix de base et les surcoûts doivent être définis en JS, pas codés en dur dans le HTML.

Bonus optionnels :

- Ajout d'une offre temporaire "Économisez 20%" avec un bouton et/ou un timer.
- Mode "challenge" avec budget limité
  - La personne peut choisir un budget max (ex : 500€).
  - Le simulateur **indique en temps réel si le choix dépasse le budget** et bloque certaines options ou affiche un message fun ("Oops, ton budget explose ! 💥").
- Graphique dynamique du prix
  - Afficher l'évolution du prix selon les options choisies dans un **mini-graph** (barre ou ligne) avec **Chart.js** ou **SVG simple**.
  - Chaque modification des options fait **mettre à jour le graphique en temps réel**.

Fichier :

`js/price.js`

Feature 3 : FAQ accordéon - moyen

Objectif :

Afficher une section de questions/réponses interactives pour répondre aux doutes des visiteurs.

Fonctionnalités attendues :

- Une liste d'au moins 5 questions visibles.
- Chaque question est cliquable pour afficher/masquer sa réponse.
- Il doit être possible d'ouvrir/fermer plusieurs réponses en même temps.

Contraintes techniques :

- Les réponses doivent être masquées par défaut.
- Le tout doit être géré en JS.
- Les questions/réponses doivent être générées dynamiquement à partir d'un tableau JS.

Bonus optionnels :

- **Limitation à une seule réponse ouverte à la fois**
  - Quand une question s'ouvre, toutes les autres se ferment automatiquement.
  - Ça demande de gérer un **toggle global** sur toutes les réponses.
- **Animation d'ouverture/fermeture douce**
  - Par exemple, utiliser une **transition de hauteur ou slide** pour que la réponse "glisse" vers le bas ou le haut.
- **Icône qui pivote +/– ou flèche**
  - Ajouter une petite flèche ou +/– sur chaque question qui **pivote quand la réponse s'ouvre**.
  - Permet de visualiser facilement l'état.
- **Recherche dynamique**
  - Ajouter un champ texte au-dessus de la FAQ pour **filtrer les questions en temps réel**.
  - Par exemple, taper "livraison" ne montre que les questions contenant ce mot.
- **Poser une question**

Fichier :

`js/faq.js`

Formulaire de contact avec modale - Facile

Objectif :

Proposer un formulaire simple et propre pour permettre aux utilisateurs de nous contacter.

Fonctionnalités attendues :

- Champs obligatoires : **Nom, Email, Message**.
- Validation du formulaire :
  - Champ vide → message d'erreur affiché dans le DOM.
  - Email non valide → message d'erreur.
- À la soumission réussie : afficher une **modale** de confirmation (et non un `alert()`).
- La modale doit être créée dynamiquement en JS.

Contraintes techniques :

- La validation doit se faire sans rechargement de page.
- La modale doit pouvoir être fermée par l'utilisateur.

Bonus optionnels :

- Validation en temps réel : afficher les erreurs dès la saisie.
- Animation d'apparition de la modale.
- Animation sur le bouton "Envoyer" (chargement, confirmation).

Fichier :

`js/contact.js`

Contraintes générales

- Respecter l'accessibilité :
  - Labels sur tous les champs de formulaire.
  - Boutons cliquables avec rôle clair.
  - Contrastes suffisants.

- Code commenté, lisible et structuré.
- Responsiveness minimum : desktop & tablette.
- Organisation en modules distincts : pas tout dans `main.js`.

Pour aller plus loin

- Galerie d’images : carrousel + zoom
- Bouton qui ouvre une interface de chat dans lequel l’utilisateur peut envoyer sa demande / question
- Systèmes d’onglets pour intégrer les différentes fonctionnalités : description, avis, produits similaires, etc...