📚 Synthèse Complète : Développement Web (HTML/CSS)

1. Structure et Sémantique HTML

Le Squelette Fondamental

Tout document HTML5 valide doit respecter une structure précise pour être interprété correctement par le navigateur.

- **<!DOCTYPE html>** : Ce n'est pas une balise, mais une instruction qui doit être sur la **toute première ligne**. Elle indique au navigateur qu'il s'agit d'un document HTML5 moderne.
- **<html>** : La balise racine. Elle doit contenir l'attribut lang (ex: <html lang="fr">) pour l'accessibilité et le référencement.
- **<head>** : La "tête pensante". Elle contient les métadonnées invisibles (encodage charset="utf-8", titre, liens CSS).
- **<body>** : Le "corps visible". Tout ce qui s'affiche à l'écran doit se trouver ici.

La Sémantique Textuelle et Structurelle

Le HTML sert à donner du **sens** (sémantique), pas à faire de la mise en page.

- **Hiérarchie :** Il ne doit y avoir qu'un seul <h1> par page (le titre principal).
- **Emphase :**
  - : Indique une **forte importance** (gras par défaut).
  - : Indique une *emphase* qui change le sens de la phrase (italique par défaut).
  - *Piège QCM :* Ne confonds pas avec <b> ou <i> qui sont purement visuels et à éviter.
- **Citations :**
  - <q> (Quote) : Pour une citation courte **en ligne** (dans le flux du texte).
  - <blockquote> : Pour une citation longue formant un **bloc** distinct.
- **Structure de page :**
  - <main> : Contient le contenu **principal et unique** de la page. Il ne doit être utilisé qu'**une seule fois** par page visible.
  - <header>, <nav>, <footer>, <article>, <section> organisent le contenu logiquement.

Les Médias (Images et Figures)

- **Balise <img>** : C'est une balise **autofermante**. Elle nécessite deux attributs obligatoires :
    1. src : Le chemin du fichier.
    2. alt : La description pour l'accessibilité.
        - *Règle d'or :* Si une image est purement **décorative**, l'attribut alt doit être présent mais **laissé vide** (alt="") pour être ignoré par les lecteurs d'écran.
- **Légendes** : Pour associer sémantiquement une image et sa légende, on utilise <figure> (conteneur) et <figcaption> (légende).

Les Tableaux

- **Usage strict :** Les tableaux (<table>) sont réservés exclusivement aux **données tabulaires** (statistiques, calendriers). Ils ne doivent **jamais** être utilisés pour la mise en page.
- **Structure :** <table> (conteneur) > <tr> (ligne) > <th> (cellule d'en-tête) ou <td> (cellule de donnée).

---

2. Les Formulaires

Balises et Attributs Clés

- **<form>** : Le conteneur. Il définit l'attribut action (URL cible) et method (envoi).
- **<label>** : Indispensable pour l'accessibilité. Il doit être lié à son champ via l'attribut for qui correspond à l'id du champ (<label for="nom"> -> <input id="nom">).
- **L'attribut name** : Crucial. Si un champ n'a pas d'attribut name, sa donnée n'est **jamais envoyée** au serveur.
- **Attributs UX :**
    - placeholder : Texte indicatif temporaire.
    - required : Rend le champ obligatoire pour la soumission.
    - autofocus : Place le curseur dans le champ au chargement.
    - pattern : Valide le contenu avec une expression régulière (Regex).

Boutons et Types d'Input

- **<input type="range">** : Crée un curseur. Attributs : min, max, step.
- **Boutons de soumission :**

- <input type="submit"> : Bouton classique, libellé via l'attribut value.
- <button> : Plus flexible, peut contenir du HTML (icônes, etc.).
- *Piège QCM :* S'il n'a pas d'attribut type spécifié, un <button> dans un formulaire se comporte par défaut comme un **submit**. Pour un bouton neutre (JS), il faut type="button".

Méthodes d'Envoi : GET vs POST

Méthode	**Utilisation**	**Visibilité des données**	**Emplacement (Debug)**
GET	Recherche, filtres	Visibles dans l'URL	Onglet Réseau -> **En-têtes** (URL)
POST	Connexion, données sensibles	Cachées à l'utilisateur	Onglet Réseau -> **Charge utile** (Payload)

---

3. CSS : Fondamentaux et Cascade

Séparation et Priorité

Le CSS gère la présentation. Pour la mise en page, on n'utilise jamais de balises HTML (
, vides), mais des propriétés CSS (marges, padding).

L'ordre de priorité (Cascade) du plus fort au plus faible :

1. **Style inline** (dans la balise HTML) : Priorité maximale.
2. **Style interne** (dans <head><style>) : Priorité moyenne.
3. **Feuille de style externe** (fichier .css) : Priorité la plus faible (mais c'est la bonne pratique !).

Sélecteurs

- **Classe (.)** : Peut être utilisée sur plusieurs éléments.
- **ID (#)** : Doit être **unique** dans la page. Il cible un seul élément spécifique.

---

4. Le Modèle de Boîte (Box Model) et Positionnement

Composition d'une Boîte

De l'intérieur vers l'extérieur :

1. **Content** : Le contenu réel (texte, image).
2. **Padding** : Marge *intérieure* (espace entre contenu et bordure).
3. **Border** : La bordure.
4. **Margin** : Marge *extérieure* (espace entre les boîtes).

Important : box-sizing: border-box est recommandé. Avec cette propriété, le padding et la bordure sont inclus dans la largeur (width) déclarée, ce qui simplifie les calculs.

Flux et Affichage (display)

- **block** : Prend toute la largeur, crée un retour à la ligne (ex: , <p>, <h1>).
- **inline** : Ne prend que la largeur du contenu, pas de retour à la ligne, width/height ignorés (ex: , <a>,).
- **inline-block** : Hybride. Se place en ligne mais accepte les dimensions (width, height) et les marges verticales.

Pièges Classiques du QCM

- **Visibilité :**
    - display: none : Retire l'élément du flux. Il n'existe plus visuellement et **ne prend plus de place**.
    - visibility: hidden : Rend l'élément transparent mais il **conserve son espace** (il y a un "trou" vide).
- **Fusion des marges (Collapsing Margins) :**
    - Verticalement, si deux marges se touchent (ex: margin-bottom: 25px et margin-top: 40px), elles ne s'additionnent pas. La plus grande l'emporte. L'écart sera de **40px** (pas 65px).
- **Positionnement :**
    - position: fixed : L'élément sort du flux et se positionne par rapport à la **fenêtre du navigateur (viewport)**. Il reste fixe au scroll.

---

5. Flexbox (Mise en page moderne)

Pour activer Flexbox : display: flex sur le parent (conteneur).

Propriétés du Conteneur (Parent)

- **flex-direction** : Définit l'axe principal (row horizontal, column vertical).

- **flex-wrap** : Gère le passage à la ligne (nowrap, wrap).
- **justify-content** (Axe Principal) : Répartit l'espace libre.
    - space-between : Espace uniquement *entre* les éléments (collés aux bords).
    - space-around : Espace *autour* (les bords ont une demi-espace).
    - space-evenly : Espace **parfaitement égal** partout (avant le 1er, entre, après le dernier).
- **align-items** (Axe Secondaire) : Aligne les éléments (ex: center pour centrer verticalement).

Propriétés des Enfants (Items)

- **flex-basis** : Taille idéale de base avant distribution de l'espace.
- **flex-grow** : Capacité à s'agrandir pour remplir le vide.
- **flex-shrink** : Capacité à rétrécir si manque de place.
- **order** : Modifie l'ordre **visuel** des éléments sans changer le HTML.
- **align-self** : Permet à un seul élément de déroger à la règle align-items du parent.

---

6. Responsive Design & Media Queries

Le Viewport

La balise <meta name="viewport" ...> est **obligatoire**. Sans elle, un mobile simulera un écran large (ex: 980px) et dézoomera, rendant le site illisible.

Les Media Queries

- **Syntaxe :** @media (condition) { ... }.
- **Stratégies :**
    - **Mobile First** : On code pour le mobile d'abord. On utilise min-width pour adapter vers les écrans plus larges (on "monte").
    - **Desktop First** : On code pour l'ordi d'abord. On utilise max-width pour adapter vers les petits écrans (on "descend").
    - **Orientation** : (orientation: portrait) détecte si la hauteur est supérieure à la largeur.

Images Fluides

Pour qu'une image soit responsive (ne déborde pas et garde ses proportions) :

CSS

```
img {
  max-width: 100%;  /* Ne dépasse jamais la largeur du parent */
  height: auto;     /* Conserve le ratio hauteur/largeur */
}
```