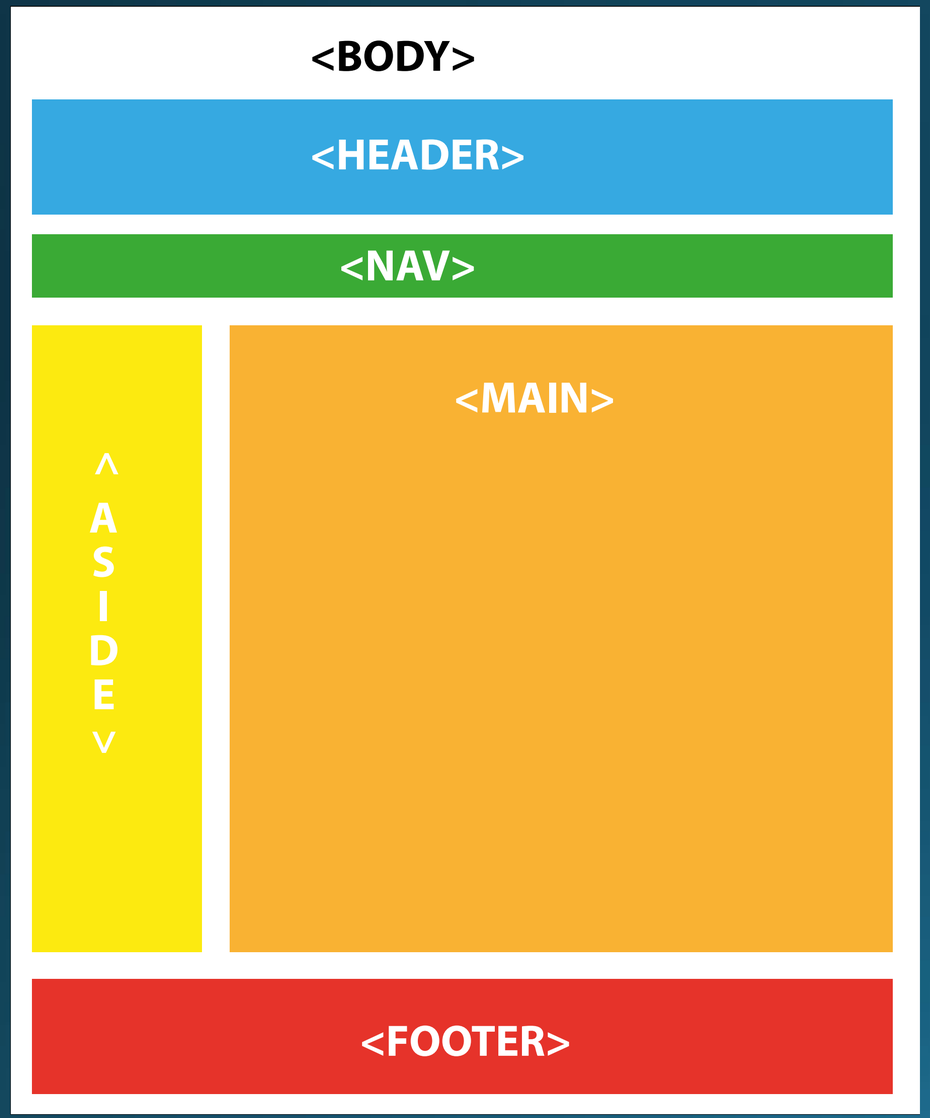
Révisions des Fondamenaux du HTML-CSS

SOMMAIRE

Structure HTML et Balise Sémantique

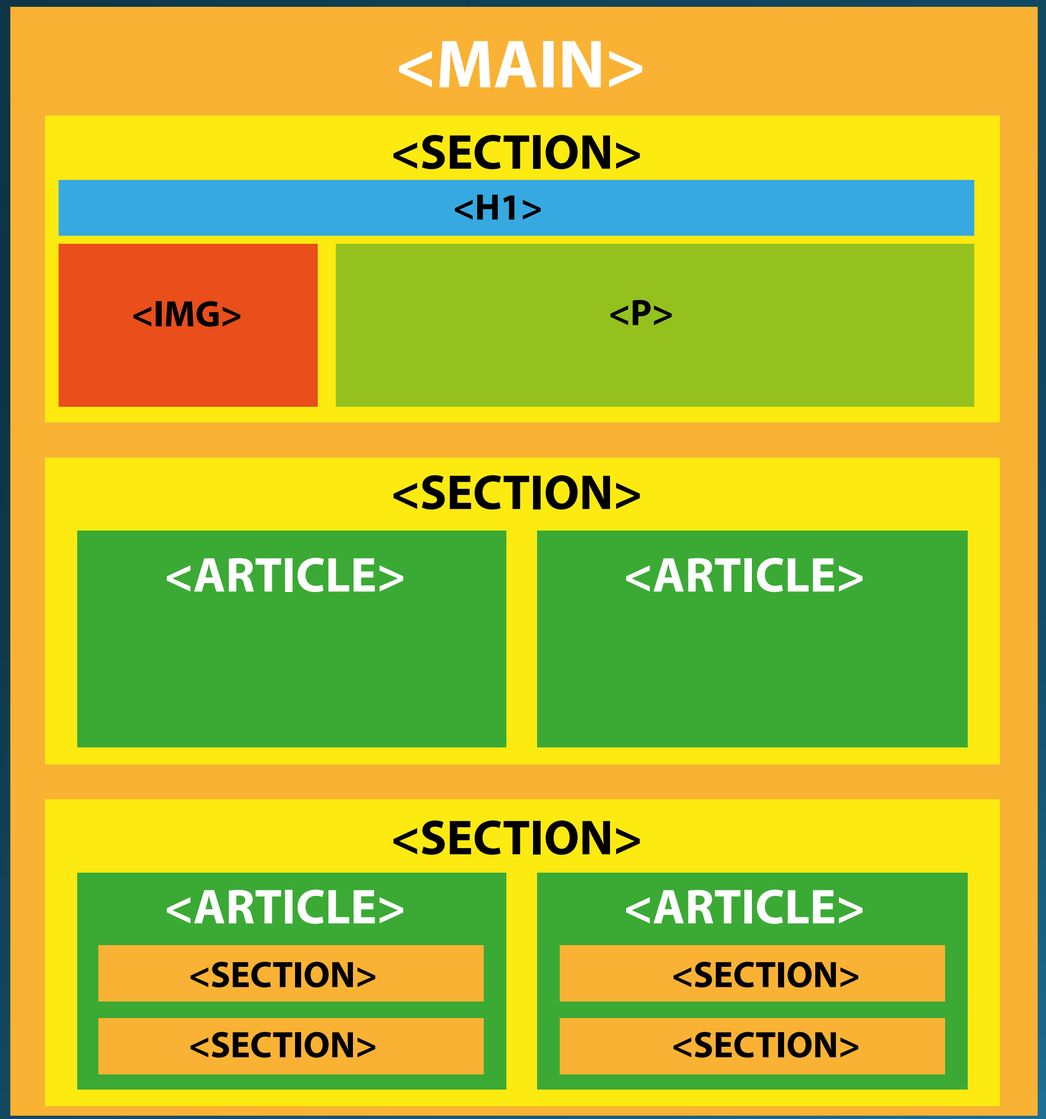


Structure HTML Classique d'un site web :

- HEADER : pour l'entête, le logo, la bannière, ...
- NAV : pour le menu. On peut en avoir plusieurs, à plusieurs endroits de la page.
- MAIN : pour le contenu principal
- ASIDE : pour le contenu indirectement lié au contenu principal (avis de consommateur, catégorie d'article, ...), ou le contenu auxiliaire (widget météo, publicité, ...)
- FOOTER : pour le pied de page, les informations liées au site (mention légal, réseaux sociaux, ...), le contact, ... Toutes les infos qui n'appartiennent pas au contenu principal

ATTENTION : Cela ne veut pas dire que TOUS les sites doivent respecter cette structure. Il faut surtout comprendre le Pourquoi.

Structure HTML et Balise Sémantique



<u>Pourquoi structurer en utilisant des balises sémantiques :</u>

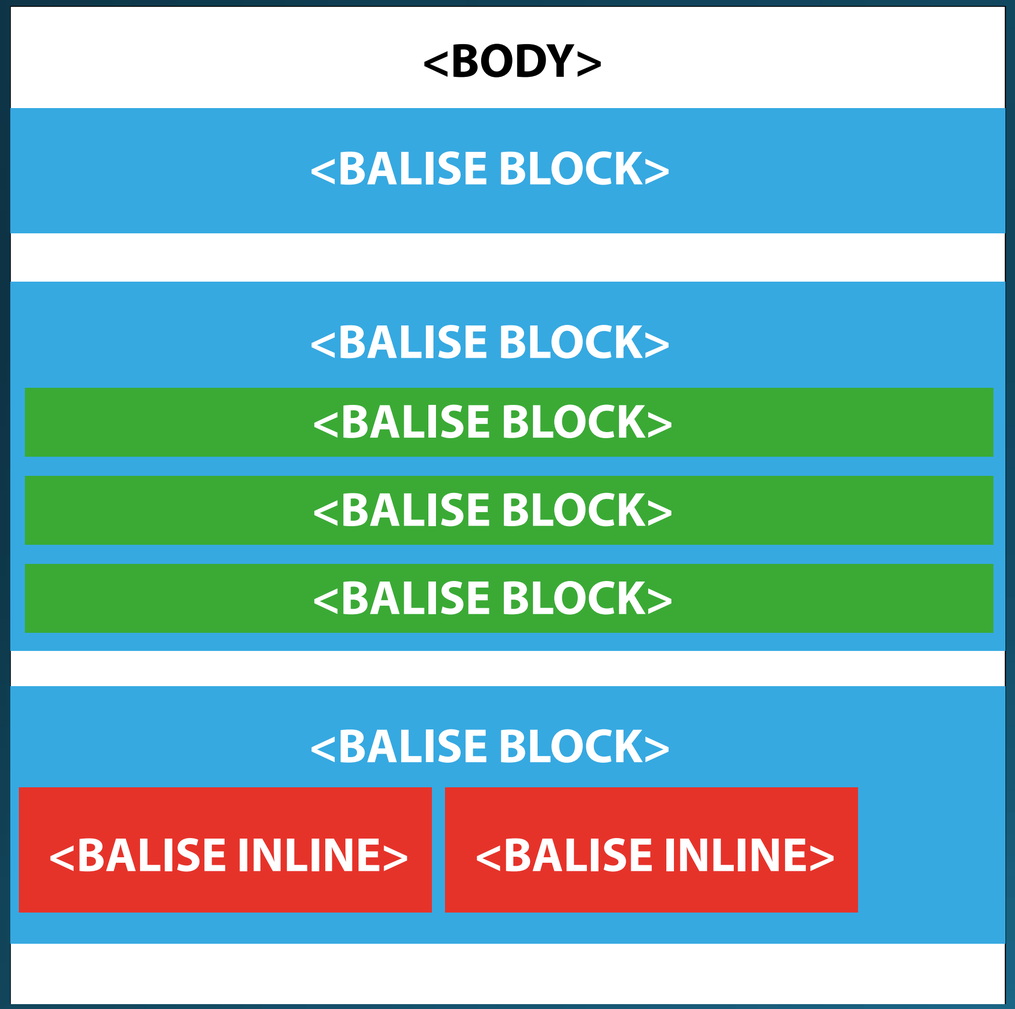
- Plus facilement lisible et compréhensible pour une équipe de Dev
- Limite le nombre de classe pour cibler nos éléments (cibler un ID est moins performant que cibler une Class, et cibler une Class est moins performant que cibler une Balise)
- Favorise le SEO auprès des bots de Google

Il existe de nombreuse balises sémantiques :

https://developer.mozilla.org/fr/docs/Glossary/Semantics

https://www.w3schools.com/html/html5_semantic_elements.asp

Les Balises Block et les Balises Inline



Balise Block :

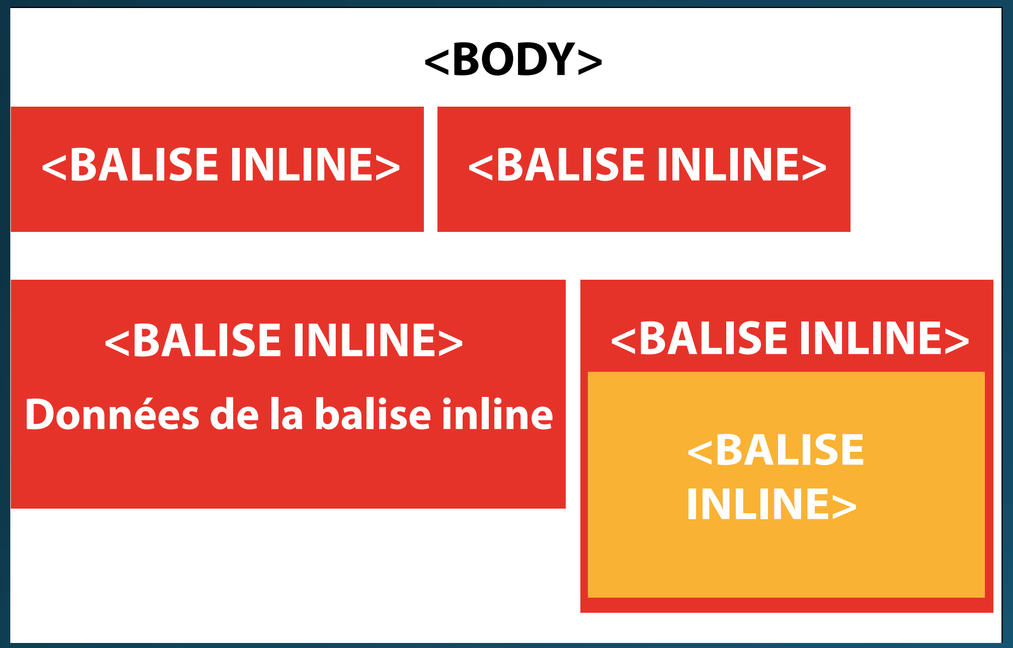
- Prennent toute la largeur disponible
- Se place automatiquement sur une nouvelle ligne
- Peuvent contenir des balises inline et d'autres balises block

Quelques exemples : <article>, <aside>, , <fieldset>, <footer>, <form>, <balise titre h1, h2, ...>, <header>, <li>, <main>, <nav>, <ol et ul>, <p>, <section>, <table>, etc.

A lire :

https://developer.mozilla.org/fr/docs/Web/HTML/Block-level_elements

Les Balises Block et les Balises Inline



Balise Inline :

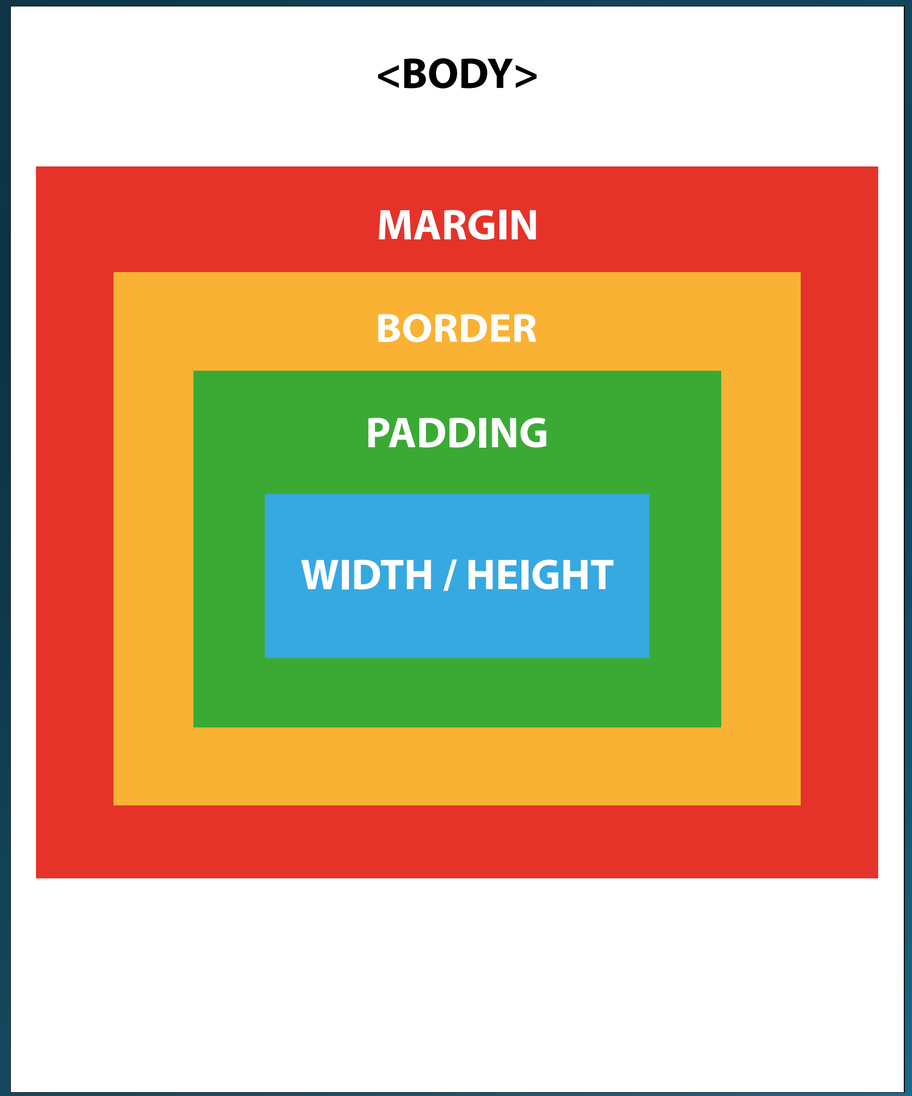
- Prennent la largeur de leur contenu
- Se place à la suite sur la même ligne
- Peuvent contenir uniquement des données ou des balises inline

*Quelques exemples : <a>, <audio>,
, <button>, , <iframe>, <img>, <input>, <li>, <script>, <select>, , , etc.*

A lire :
https://developer.mozilla.org/fr/docs/Web/HTML/Inline_elements

NOTE : Il est possible de modifier le comportement Block ou Inline d'un élément HTML grâce à la propriété CSS **DISPLAY**

Le Modèle de Boîte



L'Espace occupé par un élément HTML est défini par le cumul :

- Du **Width/Height** (largeur/hauteur) de l'élement, correspondant à la taille de son contenu (balise inline, uniquement sur le Height pour les balises block), toute la largeur disponible (balise block), ou les propriétés width et height si elles sont définies
- Du **Padding**, correspondant à la marge intérieure (entre le contenu et la bordure)
- Du **Border**, correspondant à la bordure de l'élément et qui sépare l'intérieur de l'extérieur.
- Du **Margin**, correspondant à la marge extérieure (entre l'élément et ses voisins)

A lire :

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model

Lier un fichier via son chemin relatif

Concerne les balises ayant un attribut «src»

Exemple : <link>, <img>, <audio>, <video>, ...

Valeur de l'attribut «src»

- Le fichier se trouve dans le même dossier que notre page html -> écrire le nom_du_fichier

  Exemple : <link src="style.css">

- Le fichier se trouve dans un dossier, et ce dossier se trouve dans le même dossier que notre page html -> écrire le nom_du_dossier/nom_du_fichier

Exemple : <img src="dossier_mes_images/mon_image.jpg">

- Le fichier se trouve à l'extérieur de mon dossier où se trouve notre page html -> je dois sortir du dossier en écrivant ../ puis en écrivant le chemin menant à mon fichier

Exemple : <audio src="../dossier_musique/ma_chanson.mp3">

IMPORTANT : dans le chemin que j'écris, je ne commence JAMAIS par un **/** seul. Soit je commence par **./** pour rester dans le même dossier, soit je commence par **../** pour sortir de mon dossier.

Des Images bien redimensionnées

Une image est une balise inline possédant ses propres dimensions

Taille : 500x500px

En modifiant soit sa largeur, OU soit sa hauteur, elle conserve ses proportions

Width:250px

Taille : 250x250px

Mais en modifiant sa largeur ET sa hauteur, elle peut perdre ses proportions

Height:500px

Width:250px

Taille : 250x500px

LES SELECTEURS CSS

*** :** Sélecteur universel
balise : Sélecteur d'une balise
.class : Sélecteur d'un élément possédant la class
#id : Sélecteur d'un élément possédant l'id

A[attributC] : Sélecteur des éléments A possédant l'attribut C
A[attributC="valeurD"] : Sélecteur des éléments A possédant l'attribut C dont la valeur est D
A[attributC^="valeurD"] : Sélecteur des éléments A possédant l'attribut C dont la valeur commence par D
A[attributC$="valeurD"] : Sélecteur des éléments A possédant l'attribut C dont la valeur se termine par D
A[attributC*="valeurD"] : Sélecteur des éléments A possédant l'attribut C dont la valeur contient D

A B : Sélecteur des éléments B descendants des éléments A
A > B : Sélecteur des éléments B DIRECTEMENT enfants des éléments A
A + B : Sélecteur des éléments B DIRECTEMENT voisins des éléments A
A ~ B : Sélecteur des éléments B voisins (pas forcément direct) des éléments A

LES PSEUDO-CLASSES CSS

:first-child : cible un élément qui est le premier enfant de son parent
:last-child : cible un élément qui est le dernier enfant de son parent
:nth-child(x) : cible un élément dont la position au sein de son parent correspond à X (even, odd, 2, 3, n+1, 2n, ...)

:active : cible un élément lorsque celui-ci est activé (clic de la souris, touche tab)
:checked : cible une case à cocher ou un bouton radio coché, ainsi qu'une option sélectionnée
:disabled : cible un élément désactivé (qui ne peut être :active, ou recevoir le :focus)
:enabled : cible un élément activé (qui peut être :active, ou recevoir le :focus)
:focus : cible un élément recevant le focus (sélectionné avec le clavier ou activé avec la souris)
:hover : cible un élément recevant le survol du pointeur
:link : cible des liens n'ayant pas été visités
:target : cible l'unique élément dont l'ID correspond au fragment d'identification de l'URL
:visited : cible des liens ayant été visités

NOTE : Il est possible de combiner les Sélecteurs et les Pseudo-classes

OUTIL 1 : Des unités proportionnelles pour la taille

- Les **Pourcentages** : définit une taille proportionnellement au parent de l'élément.

Exemple : <article> possédant une image avec un width à 60%, l'image occupera 60% de la taille de <article>

- Le **vw** (Viewport-Width) et le **vh** (Viewport-Height) : définit une taille proportionnellement à la largeur ou à la hauteur de l'écran.

Exemple : une image à 60vh occupera 60% de la hauteur de l'écran

- Le **em** : unité de taille relative à la taille de la police définit par le parent. Par défaut elle vaut 16px.

Exemple : un margin (d'un élément A) de 2em fera 2 fois la taille du texte du parent (de l'élément A). Il vaudra donc 40px si le Font-Size de la DIV contenant l'élément A est définit à 20px.

- Le **rem** : comme le em, mais par rapport à la balise racine (<html>).

A lire :

https://www.codeur.com/tuto/css/unite-de-mesure-taille-px-em-rem/

OUTIL 2 : Les Media Queries

Permet de créer des règles CSS qui ne vont s'appliquer que sous certaines conditions d'affichages (Suis-je sur un écran ? Suis-je en train d'être imprimé ? Est-ce que je fais au moins 1024px de large ? ...)

Syntaxe : **@media** suivi de conditions composées de **type_de_media**, de **(Caractéristique du média)**, et d'***Opérateur Logique***

- **Type de Media :** Print, Screen, All
- **Caractéristique du Média :** max-width, min-width, device-height, device-width, orientation, ...
- **Opérateur Logique :** AND, NOT, ONLY, la Virgule ; permet de combiner les règles

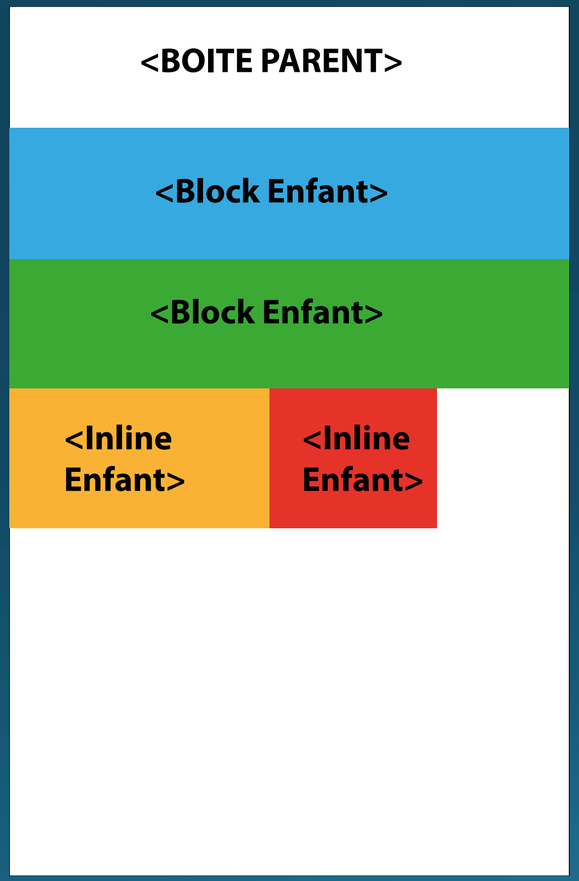
Exemple : @media print {..} / @media (max-width: 720px) {..} / @media only screen and (min-width: 1024px) {..}

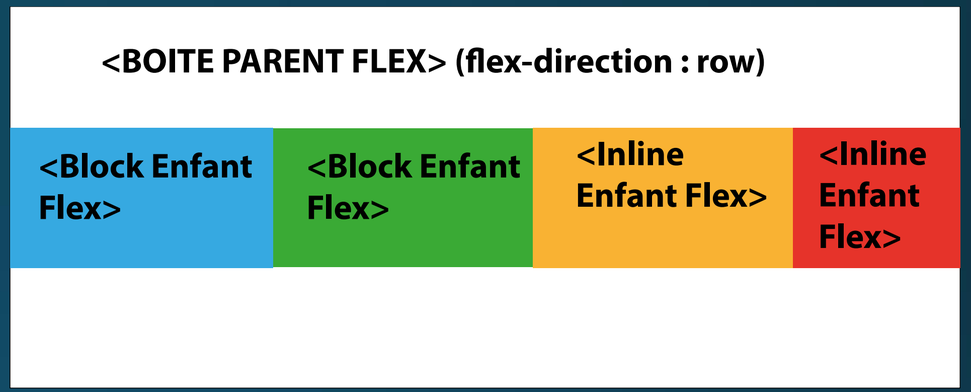
A lire :
https://developer.mozilla.org/fr/docs/Web/CSS/Media_Queries/Using_media_queries

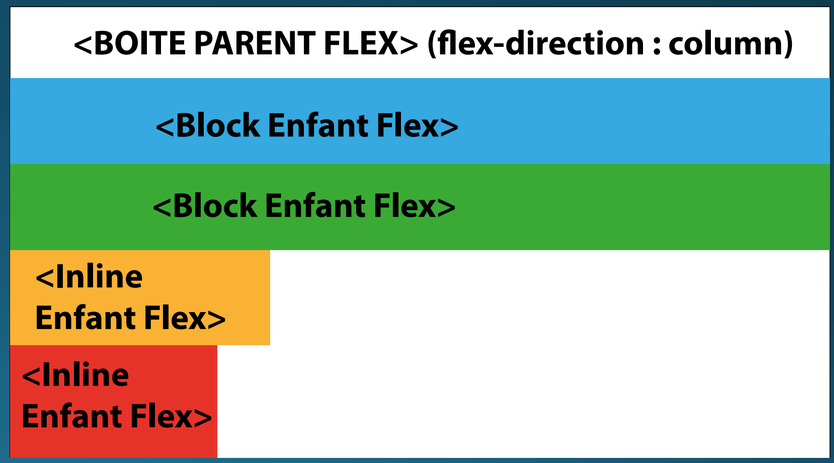
OUTIL 3 : Les Flexbox

Permet de réarranger et positionner des éléments au sein d'un parent flexible. Par défaut le réarrangement est en ligne.

A lire : https://developer.mozilla.org/fr/docs/Learn/CSS/CSS_layout/Flexbox







Les Outils pour un Site Web Responsive

PROPRIETES DU PARENT

Display (flex) : rend le parent flexible et permet le réarrangement selon le modèle des flexbox

Flex-direction (row, column, row-reverse, column-reverse) : définit l'axe principal du réarrangement

Justify-content : définit le positionnement sur l'axe principal

Align-items : définit le positionnement sur l'axe secondaire

Flex-wrap (wrap, nowrap, wrap-reverse) : autorise le passage à a ligne en cas de dépassement

PROPRIETES DES ENFANTS

Align-self : définit le positionnement de cet élément enfant sur l'axe secondaire

Flex-grow : permet l'agrandissement de l'élément enfant par redistribution de l'espace restant au sein du parent

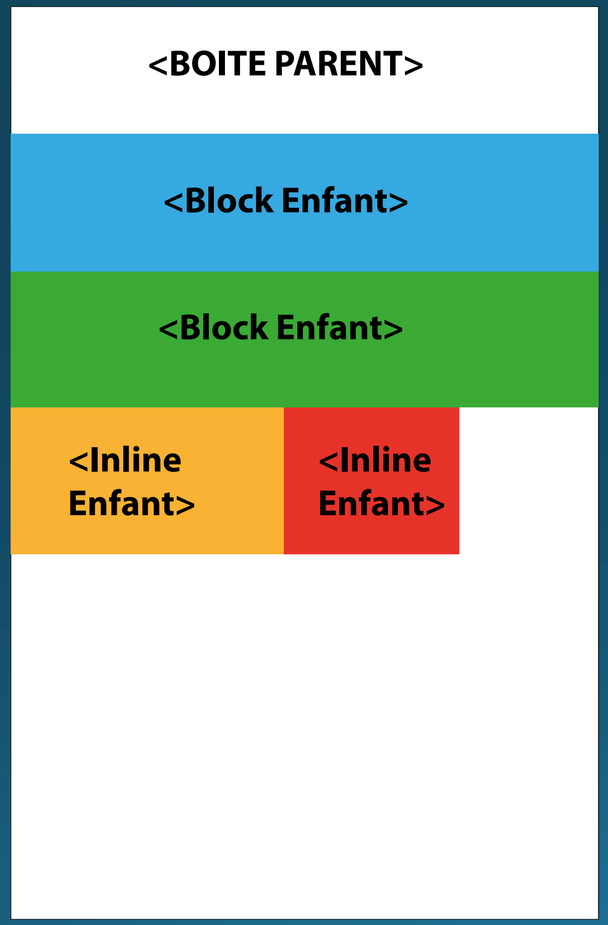
Flex-shrink : permet le rétrécissement de l'élément enfants proportionnellement aux autres enfant shrink

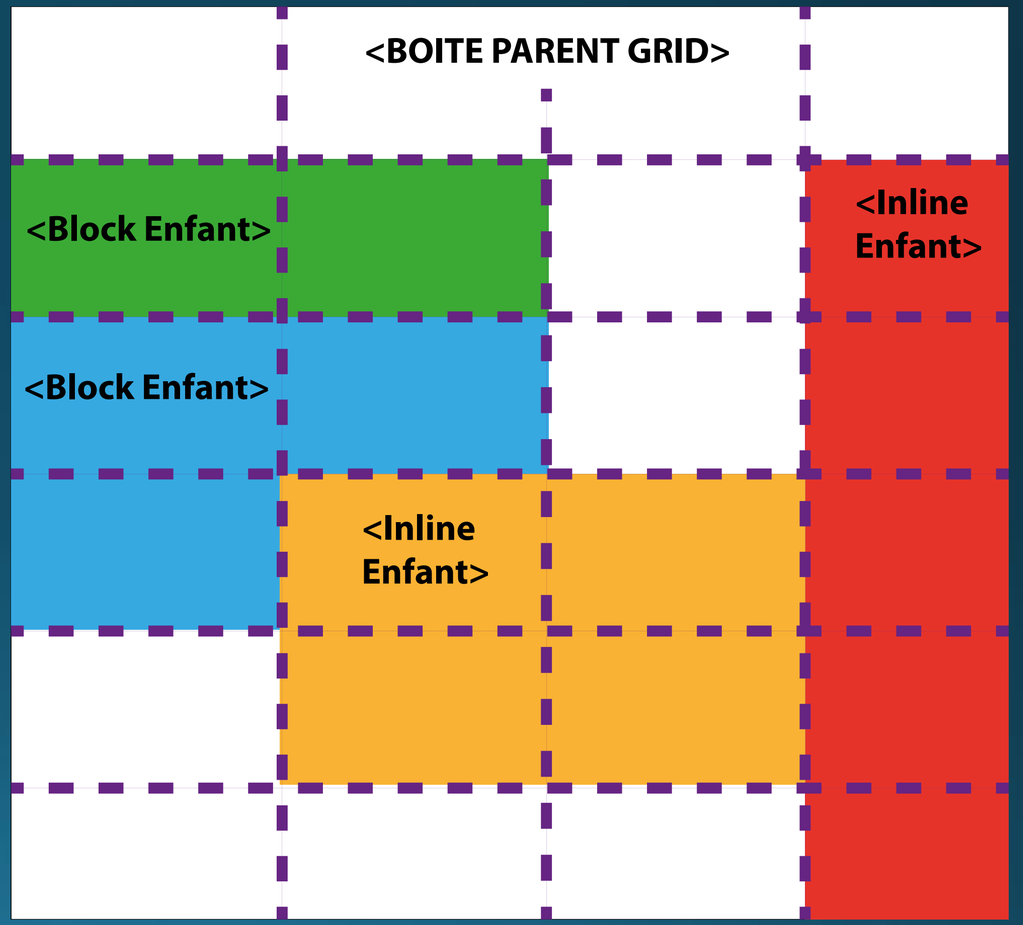
Flex-basis : définit la taille de base en flexbox

Order : définit l'ordre de réarrangement des éléments enfants

OUTIL 4 : Les Grid CSS

Permet d'arranger des éléments au sein d'une grille de mise en page définit par le parent.





A lire : https://developer.mozilla.org/fr/docs/Learn/CSS/CSS_layout/Grids

Les Outils pour un Site Web Responsive

PROPRIETES DU PARENT

Display (grid) : donne au parent les propriétés d'une grille

Grid-template-columns/rows : définit le nombre de colonne/ligne de la grille, ainsi que leur taille

Gap : définit la taille de goutière entre les colonnes et/ou les lignes

Justify-items : définit le positionnement des enfants sur l'axe en ligne (axe horizontal)

Align-items : définit le positionnement des enfants sur l'axe de bloc (axe vertical)

PROPRIETES DES ENFANTS

Grid-column : définit la colonne de placement de l'élément et son étendu horizontale dans la grille

Grid-row : définit la ligne de placement de l'élément et son étendu verticale dans la grille

Justify-self : définit le positionnement de cet élément enfant sur l'axe en ligne (axe horizontal)

Align-self : définit le positionnement de cet élément enfant sur l'axe de bloc (axe vertical)

Flexbox ou Grid ?

Flexbox et Grid sont complémentaires, et les résultats obtenus vis à vis du responsive design seront plus efficaces en utilisant une combinaison des deux techniques.

- **Grid :** Grid a une structure plus rigide que flexbox, mais permet un arrangement des éléments de manière simple et très libre, en plus de savoir facilement gérer le positionnement sur les 2 axes. Il est approprié pour structurer de grands ensembles, comme la mise en page global de la page en l'appliquant à l'élément <body>.

- **Flexbox :** Flexbox est très souple dans le réarrangement de ses éléments et dans l'espace que ces derniers occupes. Il est très performant pour gérer le positionenement sur un des deux axes (de manière plus rapide que Grid). Il est approprié pour organiser de petits ensembles, comme un menu de navigation ou une cards de portfolio.

Les Outils pour un Site Web Responsive