CHELLE Raymond 08/10/2025

Rapport de TP : CSS Avancé 2

Sommaire :

I. Introduction

Objectif :
L'objectif de ce travail pratique est de me familiariser davantage avec la **mise en page avancée en CSS** à travers l'utilisation de la **Flexbox** et des propriétés de **visibilité**. L'ensemble de ces exercices m'a permis de comprendre comment manipuler efficacement la structure d'une page HTML, gérer les alignements, répartir les espaces, et contrôler l'affichage d'éléments de manière dynamique et esthétique.

Mise en situation :
En tant qu'étudiant en **BTS SIO**, je dois être capable de concevoir des interfaces web structurées, claires et adaptables. Ces exercices m'ont permis de consolider mes connaissances sur les **bases du positionnement CSS**, la gestion des **sous-menus**, la **hiérarchie des sélecteurs**, ainsi que l'utilisation de propriétés essentielles telles que display: flex, justify-content, align-items, visibility ou encore background-color. À travers la réalisation d'exercices progressifs (du 1 au 12), j'ai appris à transformer une page statique en une mise en page cohérente, moderne et ergonomique.

Résolution :
J'ai travaillé directement sur les fichiers HTML et CSS fournis, en appliquant étape par étape les propriétés demandées. J'ai notamment

utilisé **Flexbox** pour organiser le contenu (<header>, <main>, <nav>, <aside>, <blockquote>), ajusté les largeurs et les espacements pour obtenir des alignements précis, et appris à manipuler la **visibilité** d'éléments (display:none et visibility:hidden) afin d'améliorer l'interactivité du menu. Ces manipulations m'ont permis d'obtenir un affichage fluide, un menu fonctionnel avec des sous-menus propres, et une mise en page équilibrée sur l'ensemble de la page.

II. Le travail à réaliser

Exercice 1 :
Pour cet exercice, on sait que a = 1 car toutes ces règles sont définies dans un fichier externe. Voici le classement des priorités (a, b , c , d) :

1) #id (1, 1, 0, 0)

2) nav.titi .tata div div div div div (1, 0, 2, 7)

3) ul li div.skill (1, 0, 1, 3)

4) .titi span (1, 0, 1, 1)

5) div span (1, 0, 0, 2)

6) div > a (1, 0, 0, 2)

7) div + a (1, 0, 0, 2)

Exercice 2 :
Cas 1 :
Le fichier styles.css est chargé **en externe** (donc a=1) : color: blue

La règle dans <style> est **interne** (a=1) : color: pink

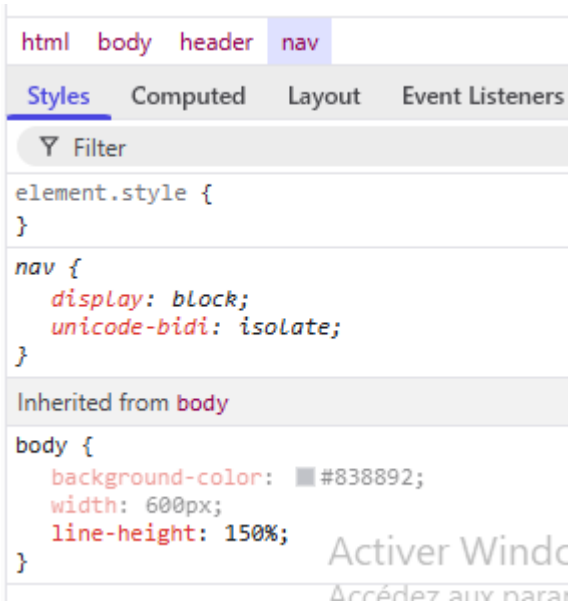
La règle **inline** dans le <p> a la priorité **a=2** (plus haute que externe et interne)
Donc la couleur affichée est rouge car la règle inline est prioritaire.

Cas 2 :
La couleur affichée est bleu, car la règle externe est chargée après la règle interne et a donc priorité.

Exercice 3 :
1)

```
<header>
    <nav>
        <div><a href="./index.htm">Accueil</a></div>
        <div><a href="./facts.html">Facts</a></div>
        <div><a href="./news.html">Actualités</a></div>
        <div><a href="./contact.htm">Contact</a></div>
    </nav>
</header>
```

2)



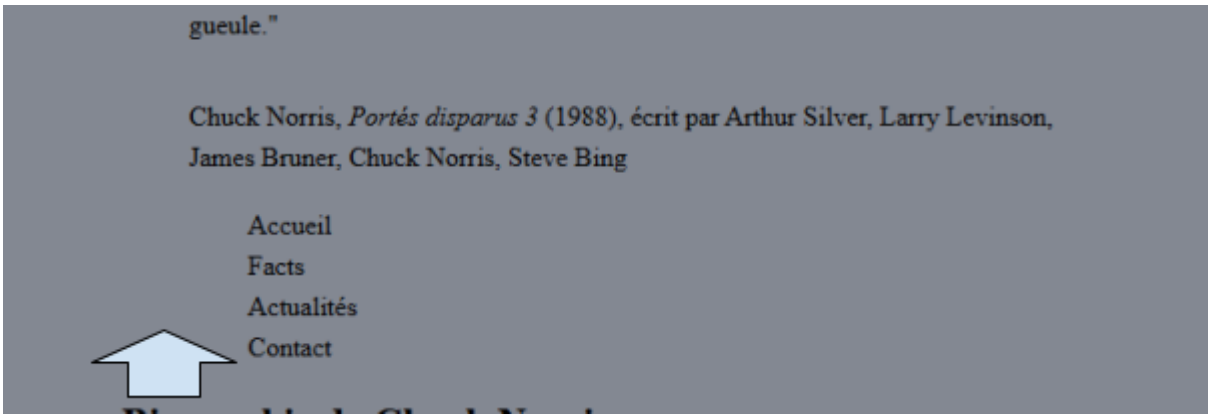
Le <nav> est par défaut un élément de type block, ce qui signifie qu'il occupe toute la largeur disponible de son conteneur parent.
3)

```
nav {
  width : 75%;
}
```

On constate que <nav> devient **plus étroit** (75% de la largeur de <header>).
4)

```css
nav {
  width : 75%;
  margin: 0 auto;
}
```

Résultat :

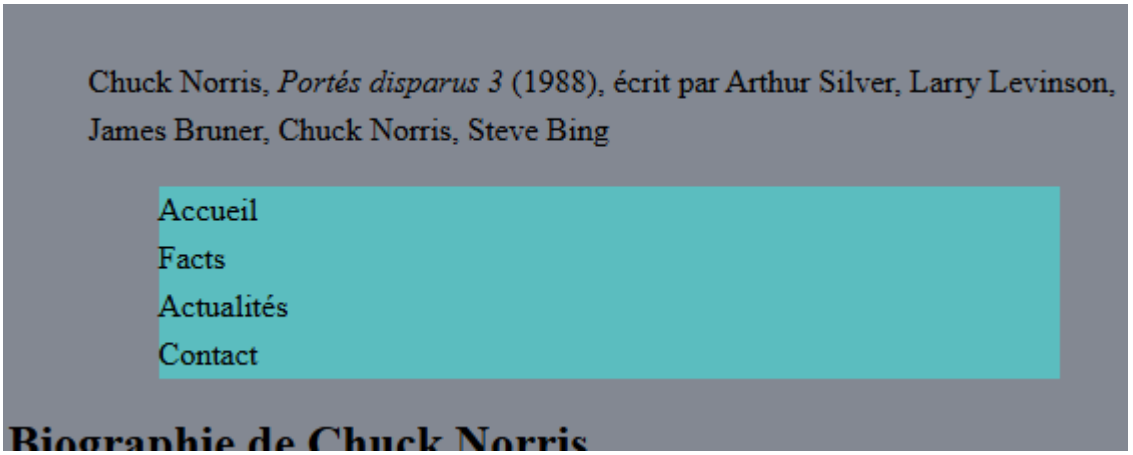


Grâce à margin: 0 auto, le navigateur répartit automatiquement les marges gauche et droite pour que le bloc soit bien centré dans son parent <header>. Cela rend l'ensemble plus esthétique car le menu ne colle plus à gauche mais est placé au milieu.

5)

```css
nav > div {
  background-color: #5BBDBF;
}
```

Résultat :



J'ai donc appliqué une couleur de fond bleue (#5BBDBF) aux qui sont les enfants directs du <nav>.

6)

```css
nav a {
  background-color: #c0d5c2;
}
```

Résultat :

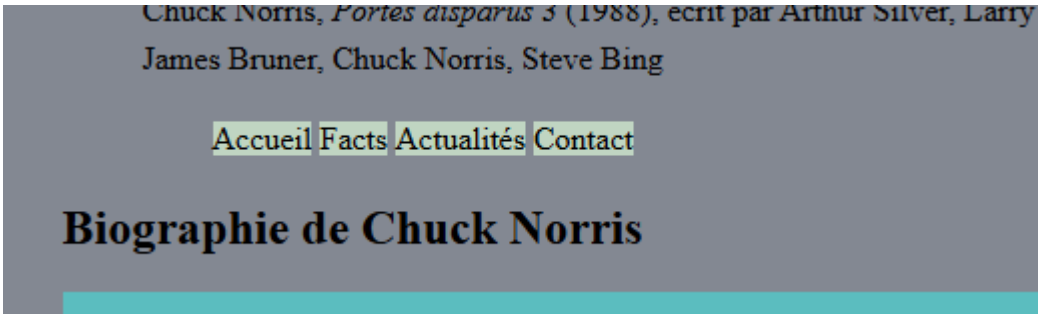
Enfin, pour l'étape 6, j'ai modifié la couleur de fond des liens eux-mêmes, les balises <a>, en leur donnant un vert clair (#c0d5c2).

Exercice 4 :

1)

```css
nav > div {
  background-color: #5BBDBF;
  display: inline;
}
```

Résultat :



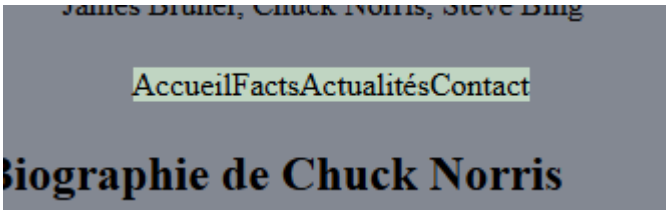
On remarque que les éléments s'affichent côte à côte plutôt qu'en colonne.

2)

```html
<header>
    <nav>
        <div><a href="./index.htm">Accueil</a></div><!--
        --><div><a href="./facts.html">Facts</a></div><!--
        --><div><a href="./news.html">Actualités</a></div><!--
        --><div><a href="./contact.htm">Contact</a></div>
    </nav>
</header>
```

Les <!-- --> sont des **commentaires HTML** qui empêchent les espaces d'apparaître à l'écran.

Résultat :

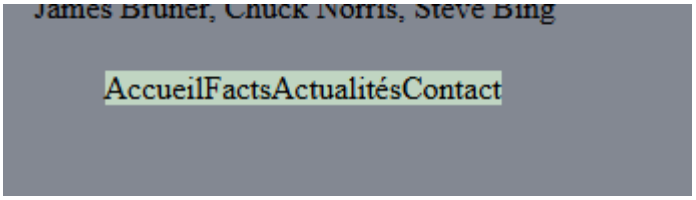


On remarque que les espaces ont bien disparus.

3)

```css
nav {
  width : 75%;
  margin: 0 auto;
  height: 50px;
}
```

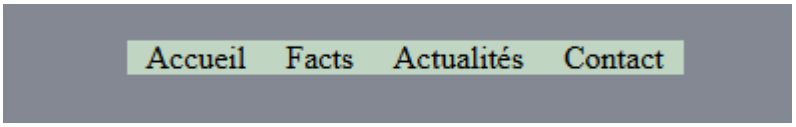
Résultat :



4)

```css
nav a {
  background-color: #c0d5c2;
  padding: 0 10px;
}
```

Résultat :



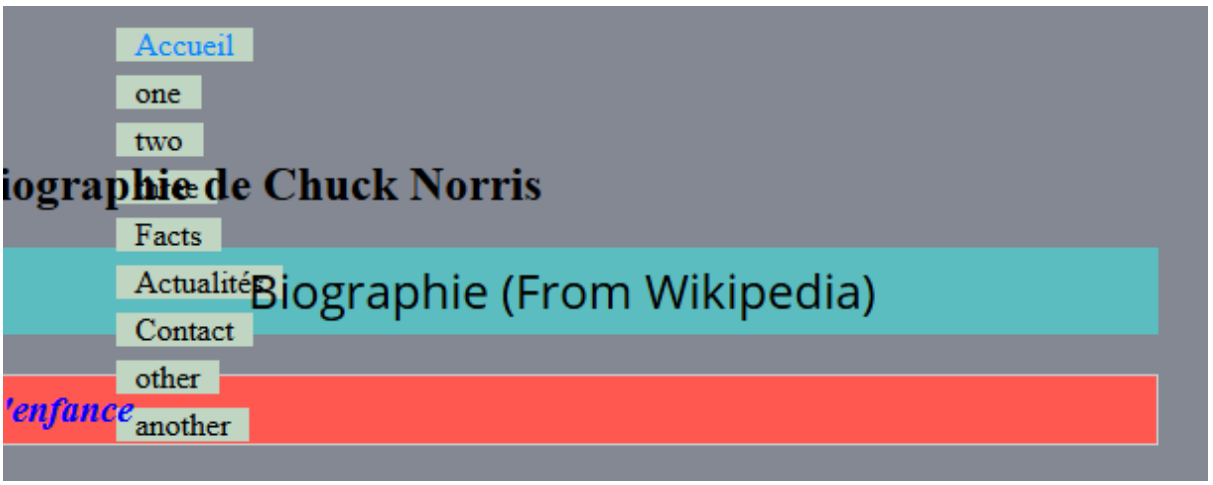
Exercice 5 :

Je place chaque `<div class="submenu">` juste **après** le lien `<a>` correspondant, **dans le même parent `<div>`**. Concrètement, pour "Accueil" et "Contact", la structure devient

```
<header>
    <nav>
        <div>
            <a href="./index.html">Accueil</a>
            <div class="submenu">
                <div><a href="./one.html">one</a></div>
                <div><a href="./two.html">two</a></div>
                <div><a href="./three.html">three</a></div>
            </div>

        <div><a href="./facts.html">Facts</a></div>
        <div><a href="./news.html">Actualités</a></div>
        <div>
            <a href="./contact.hun!">Contact</a>
            <div class="submenu">
                <div><a href="./other.html">other</a></div>
                <div><a href="./another.html">another</a></div>
            </div>
        </div>
    </nav>
</header>
```

Résultat :



Les sous-menus se sont bien ajoutés, cependant la mise en page n'est pas correcte.

2) Positionnement des sous-menus

i) Valeur de position adaptée

Je veux que l'affichage du reste de la page ne "**réserve pas d'espace**" pour les sous-menus (comme s'ils n'existaient pas dans le flux normal). La bonne valeur est donc position: absolute pour les sous-menus.

ii) Règle CSS pour .submenu avec décalage top: 50px; left: 0;

Ce que j'ajoute dans le CSS :

iii)

Les sous-menus se placent par rapport à la balise (ou) car ce sont les premiers ancêtres positionnés. Donc ils apparaissent tout à gauche de la page.

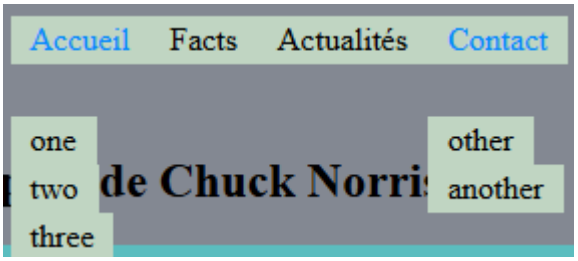
iv) Faire en sorte qu'ils se positionnent par rapport au parent

Ce que je fais :

Je rends chaque conteneur de lien de niveau menu (les nav > div) **positionné sans déplacer la mise en page**. La valeur idéale est **position: relative**. Cela ne bouge rien visuellement pour ces , mais ça crée un **contexte de positionnement** pour leurs enfants en absolute (nos sous-menus).

```css
nav{
  width : 75%;
  margin : 0 auto;
  height : 50px
}
nav > div {
  position: relative;
  background-color: #5BBDBF;
  display : inline-block;
}
nav a {
  display: inline-block;
  background-color : #c0d5c2;
  padding : 0 10px;
}
.submenu {
  position: absolute;
  top: 50px;   /* 50px depuis le haut */
  left: 0;     /* 0px depuis la gauche */
  display: block;
}
```

Résultat :



Exercice 6 :

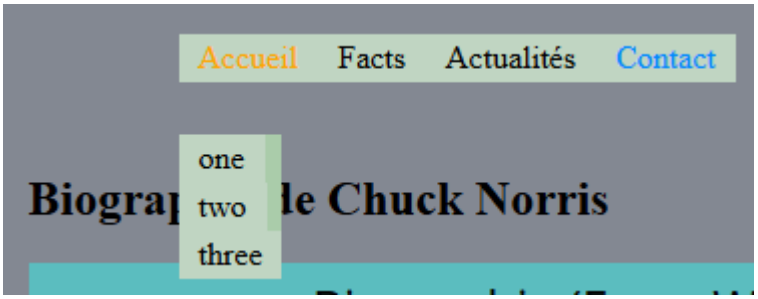
J'ai mis **background-color: #aca** sur .submenu et je les ai **masqués par défaut** avec display: none.

Pour les **afficher au survol**, j'utilise **nav > div:hover > .submenu** : quand je passe la souris sur le du menu (ex. "Accueil" ou "Contact"), **seul son sous-menu** repasse en display: block.

Je garde position: absolute; top: 100%; left: 0; pour que le sous-menu apparaisse **juste sous** le lien sans pousser le reste de la page.

```css
.submenu {
  position: absolute;
  top: 50px;    /* 50px depuis le haut */
  left: 0;      /* 0px depuis la gauche */
  background-color: #aca;  /* Ex.6 – fond demandé */
  display: none;
}
nav > div:hover > .submenu {
  display: block;
}
```

Résultat :



Exercice 7 :

1) nav en flex

Je transforme la barre en **flex container** : les items (les du menu) deviennent des **flex items** alignés sur l'axe horizontal.

```css
nav{
  display :flex;
  width : 75%;
  margin : 0 auto;
  height : 50px
}
```

2) `nav > div` en `block`, puis en `inline`

Lorsqu'on change le display:inline en display:block, ça ne change rien du tout.

3) flex-direction: column

```css
nav{
  display :flex;
  flex-direction: column;
  width : 75%;
  margin : 0 auto;
  height : 50px
}
```

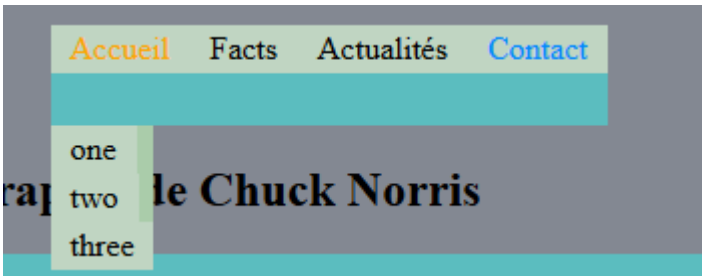
Je change l'axe principal : les items du menu **s'empilent verticalement** (un par ligne). Les sous-menus en `absolute` restent accrochés à leur parent, mais le menu principal devient une **colonne**.

Résultat :

4) Remettre la direction par défaut (`row`)

```css
nav{
  display :flex;
  flex-direction: row;
  width : 75%;
  margin : 0 auto;
  height : 50px
}
```

Résultat :



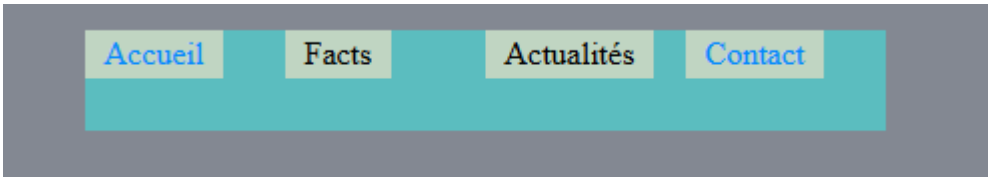
Je reviens à une **barre horizontale** classique.

5) Largeur fixe 100 px pour tous les `<div>` descendants de `nav`

Maintenant on va donner une largeur au div en tant que descendant pour pouvoir avoir la même largeur de menu et de sous menu :

```css
nav div {
  width: 100px;
}
```

Résultat :



Exercice 8 :

1) Hauteur 200 px à `nav` + fond, et sous-menus décalés de 200 px

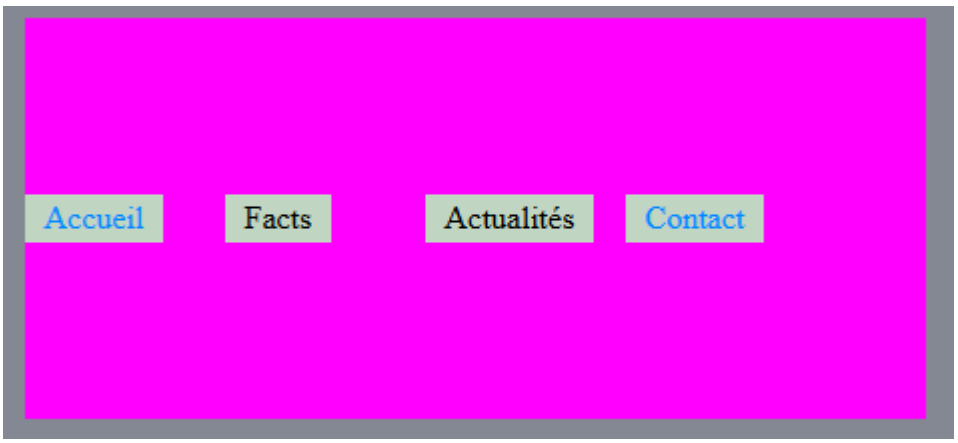
```css
nav{
  display :flex;
  width : 75%;
  margin : 0 auto;
  height : 200px;
  background : #FF00FF
}
```

Résultat :

2) `align-items: center` sur `nav`

```css
nav{
  display :flex;
  align-items: center;
  width : 75%;
  margin : 0 auto;
  height : 200px;
  background : #FF00FF
}
```

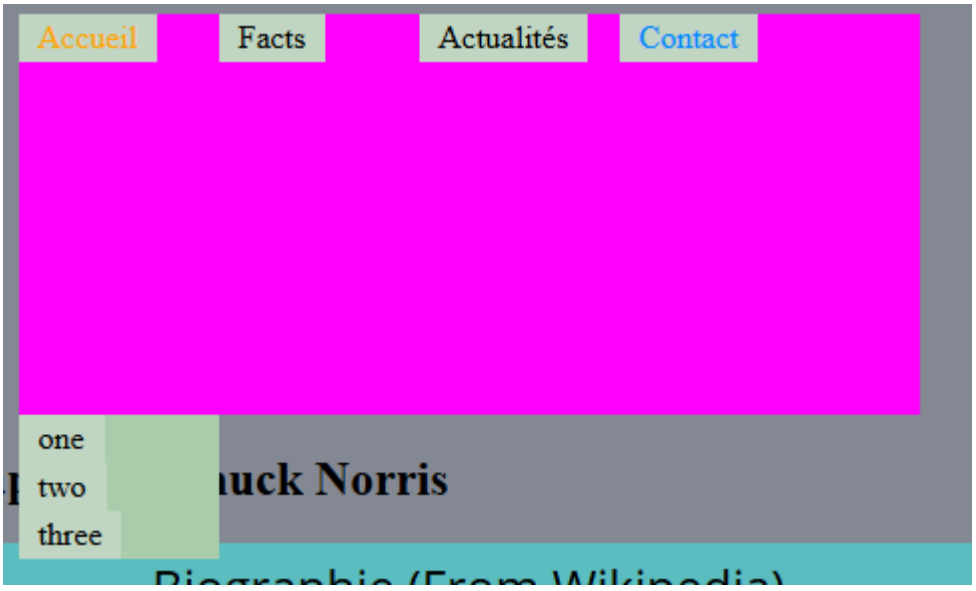
Résultat :



3) `align-items: stretch`

```css
nav{
  display :flex;
  align-items: stretch;
  width : 75%;
  margin : 0 auto;
  height : 200px;
  background : #FF00FF
}
```

Résultat :



Les enfants prennent **toute la hauteur** (200 px) → les sous-menus redeviennent **accessibles** (pas de trou).

Mais le **texte <a> n'est pas centré verticalement** dans chaque (il reste en haut si on ne fait rien).

4)

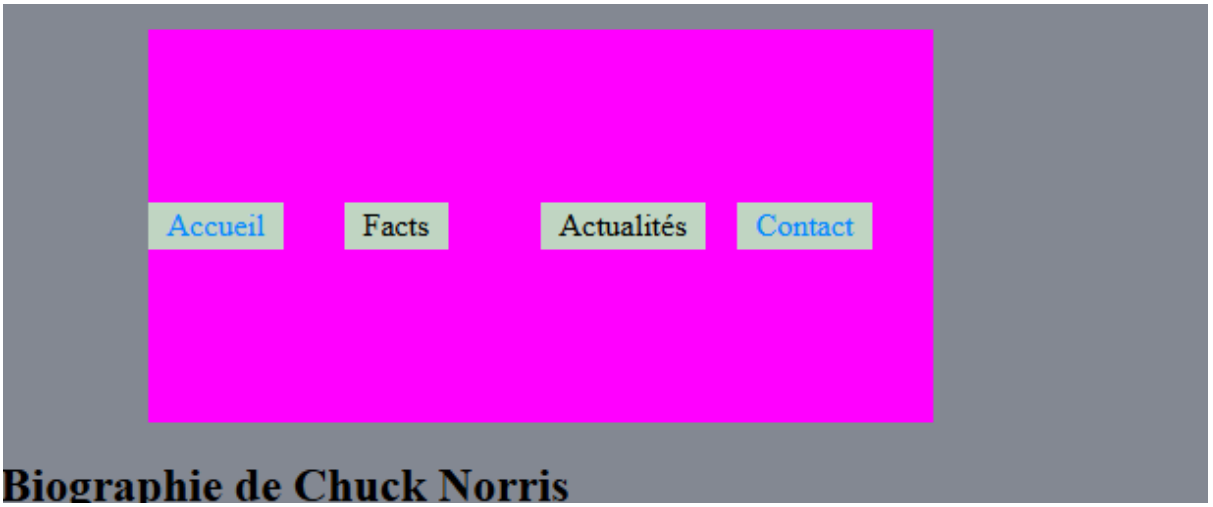
i.Maintenant on veut essayer d'utiliser align-item:center et align items:stretch dans le code, pour ça on va laisser stretch dans et on va ajouter align-items:center à nav > div :

```css
nav{
  display :flex;
  align-items: stretch;
  width : 75%;
  margin : 0 auto;
  height : 200px;
}
```

```css
nav > div {
 position: relative;
 display : block;
 align-items: center;
 background-color : #ff00ff;
}
```

ii. Pour centrer les il faut remplacer le display:block par un display:flex à :

```css
nav > div {
  position: relative;
  display : flex;
  align-items: center;
  background-color : #ff00ff;
}
```



5) Nous remettons le code d'origine (comme au début de l'exercice)

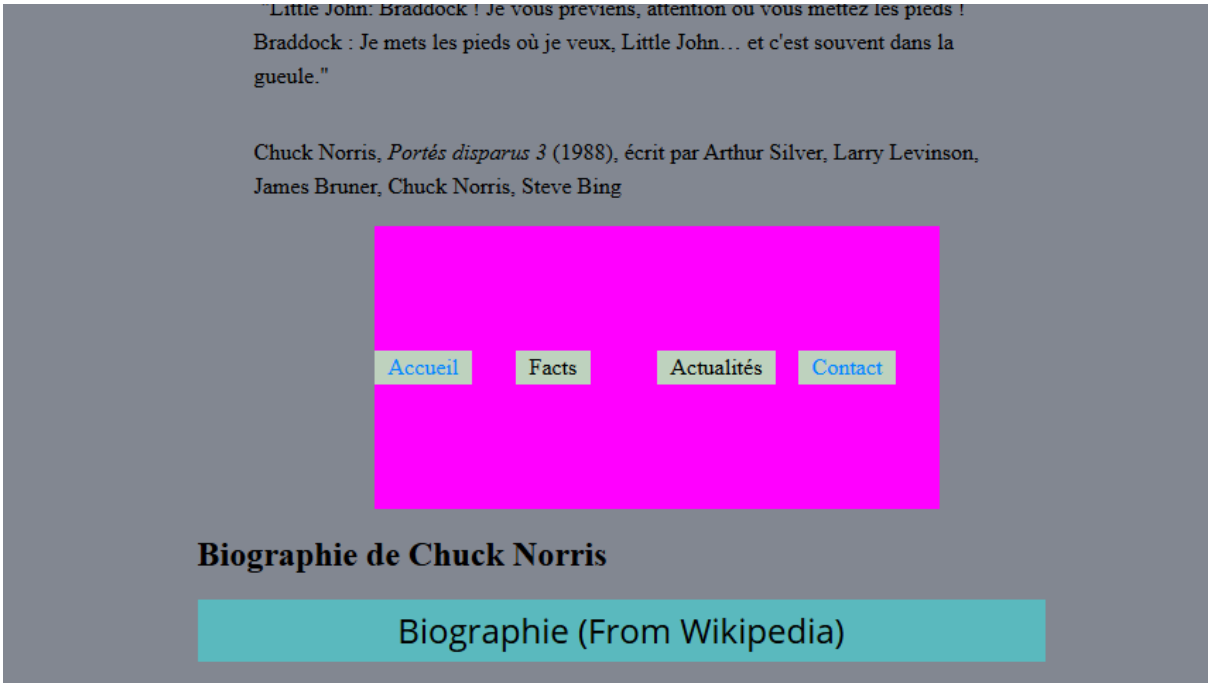
```css
nav {
  display: flex;
  align-items: stretch;
  height: 200px;
}
```

Exercice 9 :

Nous souhaitons aligner les blocs de titres à droite dans la balise en utilisant la propriété CSS justify-content. Pour cela, nous devons utiliser une valeur de type flex qui aligne les éléments à la fin de l'axe principal, pour cela on va utiliser flex-end :

```css
nav{
  display :flex;
  align-items: stretch;
  width : 75%;
  margin : 0 auto;
  height : 200px;
  justify-content:flex-end;
}
```

Résultat :



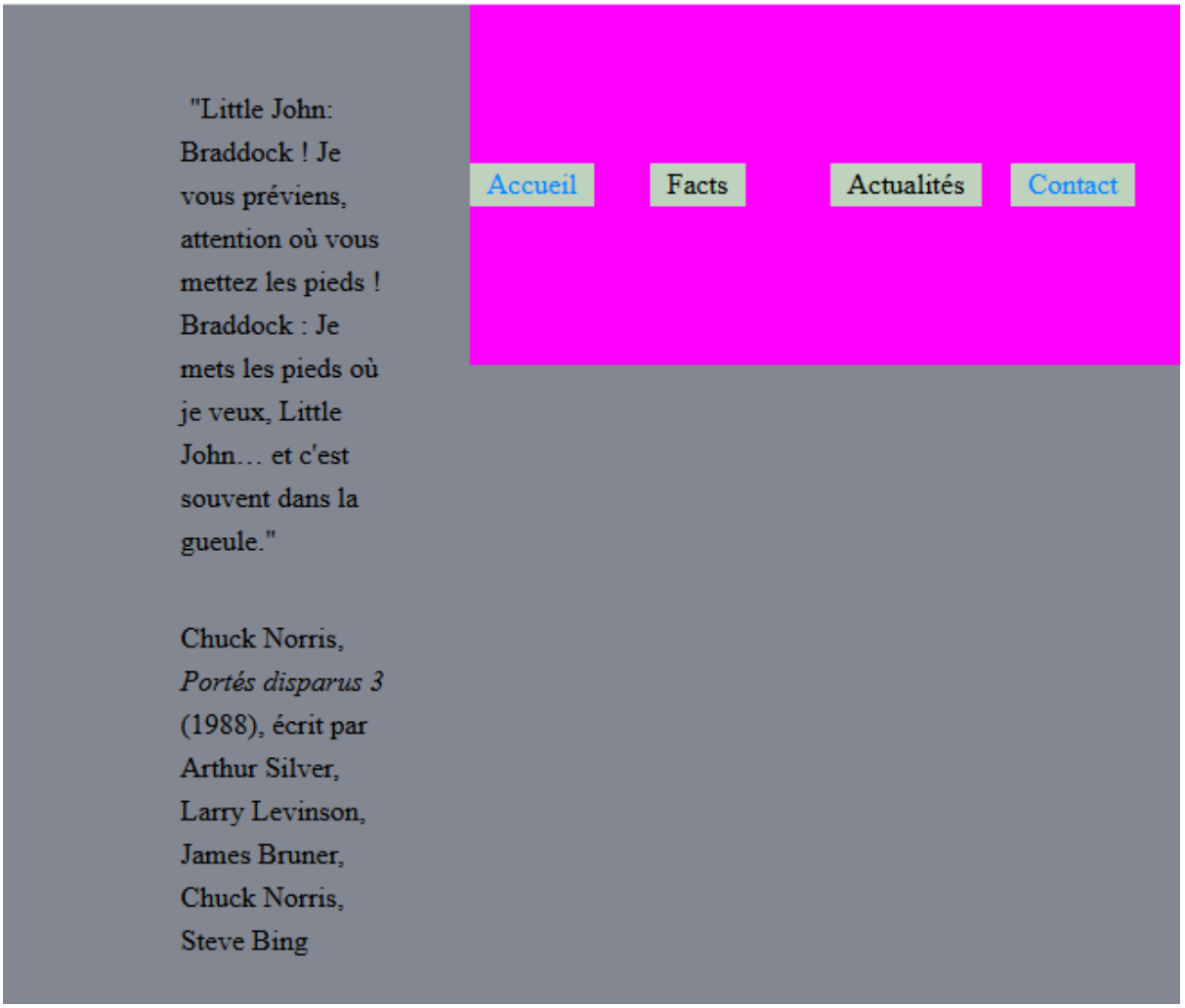
Exercice 10 :

1)

On me demande ici de placer la **citation** et le **menu de navigation** sur **une seule ligne**.
Pour cela, j'utilise : **Flexbox**.

Je sais que les deux éléments <blockquote> et <nav> sont **des enfants directs de <header>**, donc c'est sur ce dernier que je vais appliquer le display: flex;

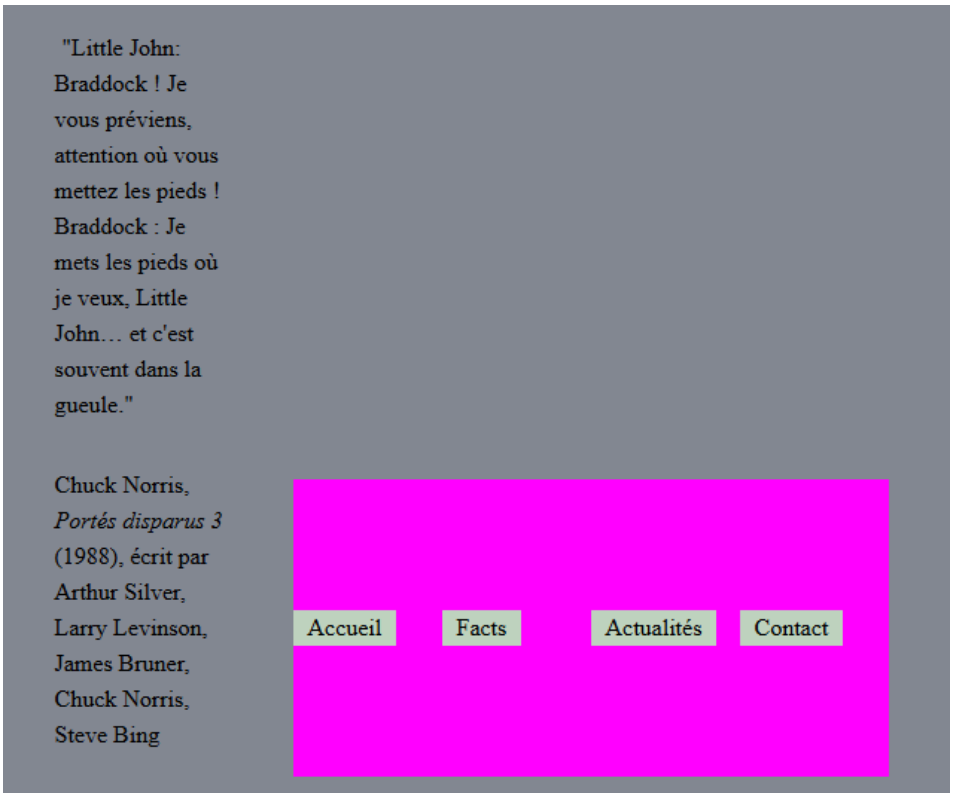
```
header {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
}
```



2) On me demande ensuite de **descendre le menu** pour qu'il soit collé au bas du <header>.

```css
header nav {
    align-self: flex-end;
}
```

Résultat :

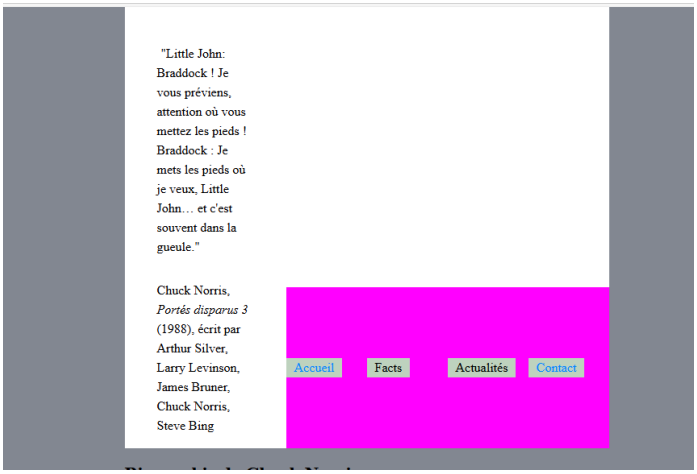


Ainsi, le menu de navigation reste à droite mais descend en bas du bloc <header>, tandis que la citation reste en haut.

3)
Pour rendre le haut de la page plus visible, je mets un **fond blanc** à la balise <header>.

```css
header {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    background-color: white;
}
```

Résultat :



4)

Enfin, je dois **enlever la couleur de fond** des liens présents dans le menu de navigation.

Je modifie donc la règle CSS du sélecteur nav a pour que la propriété background-color soit **transparente**.

```css
nav a {
  background-color: transparent;
  text-decoration: none;
  color: inherit;
  display: inline-block;
  padding : 0 10px;
}
```

Exercice 11 :

1. Dans le CSS, ajoutez width: 900px; à la règle du body.

```css
body {
  margin: 0 auto;
  width: 900px;
}
```

2.En regardant mon HTML, la table “Compétences des acteurs en arts martiaux” est **déjà dans <aside>.** Donc je **ne change rien** ici, la condition est déjà remplie.

3. Je transforme le <main> en conteneur flex pour obtenir deux colonnes :

```css
main {display: flex;
  align-items: flex-start;
}
```

Résultat :

4. Je répartis les largeurs pour faire 100% au total, avec un espace entre les deux via margin-left sur l'aside :

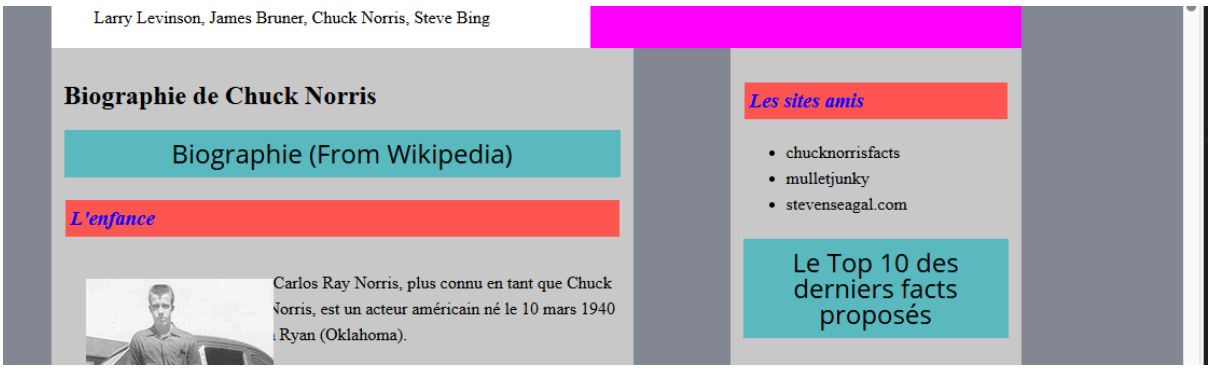
```css
main > article {
  width: 60%;
}

main > aside {
  width: 30%;
  margin-left: 10%;
}
```

5. J'uniformise le fond des deux colonnes :

```css
main > article,
main > aside {
  background-color: #CCC;
  padding: 12px;    /*plus esthétique*/
  box-sizing: border-box;
}
```

Résultat :



Exercice 12 :

1. Je place dans le HTML une petite balise `<span>` de classe `puce` juste avant le texte de chaque lien du menu :

```
<div><a href="index.html"><span class="puce">*</span>Accueil</a>
  <div class="submenu">
    <div><a href="one.html">one</a></div>
    <div><a href="two.html">two</a></div>
    <div><a href="three.html">three</a></div>
  </div>
</div><!--
--><div><a href="facts.html"><span class="puce">*</span>Facts</a></div><!--
--><div><a href="news.html"><span class="puce">*</span>Actualités</a></div><!
--><div><a href="contact.html"><span class="puce">*</span>Contact</a>
  <div class="submenu">
    <div><a href="other.html">other</a></div>
    <div><a href="another.html">another</a></div>
  </div>
</div>
```

Résultat : 

Chaque lien contient désormais une petite étoile, mais elle sera invisible tant que je ne survole pas l'élément.

2. Je vais maintenant rendre la puce invisible au repos, puis visible uniquement au survol de son lien parent.

```
.puce {
  visibility: hidden;
  margin-right: 5px;
  color: red;
}
```

3. Quand je survole un lien du menu, la puce juste avant doit devenir visible. Pour cela, j'utilise le sélecteur `a:hover .puce` :

```
a:hover .puce {
  visibility: visible;
}
```

Résultat :

Ainsi, quand ma souris passe sur “Accueil” ou “Contact”, la petite étoile rouge s’affiche à gauche du texte.

III. Conclusion

Ce travail m’a permis de mieux comprendre la **logique et la puissance de la mise en page avec Flexbox**, ainsi que les différences entre display:none et visibility:hidden.
Grâce à la réalisation des exercices, j’ai appris à **structurer un site web complet**, à **gérer la disposition des blocs** (articles, asides, menus, citations) et à **rendre un site plus intuitif visuellement**.