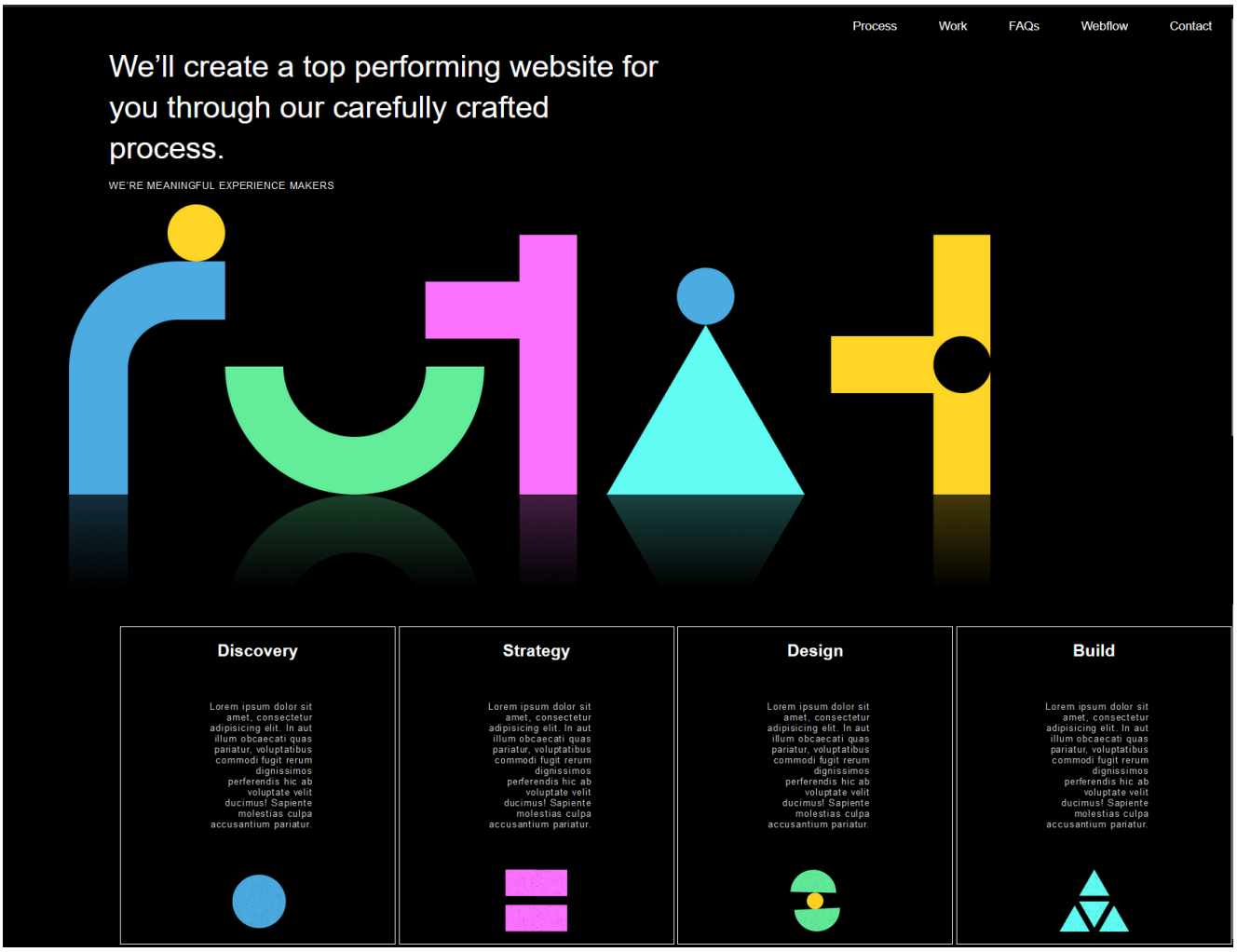
Info 8 - Html Css – partie 2

On se propose dans ce tp de continuer l'apprentissage du langage Css, en se fixant l'objectif de créer une page d'accueil de site, qui comprend un menu en haut de page, un petit texte de présentation accompagné d'une image en partie principale et un bas de page, composé de 4 articles avec images :



Les propriétés Css suivantes seront vus dans cette activité. **Il s'agira ensuite de pouvoir les maitriser** :

- Gestion des textes :
  - Police d'écriture : *font-family*
  - Taille police : *font-size*
  - Couleur police : *color*
  - Épaisseur police : *font-weight*
  - Taille police : *font-size*
  - Distance entre lettres : *letter-spacing*
  - Taille police : *font-size*
  - Soulignement *: text-decoration*

- Média Queries :
  - Css conditionnel : *@media screen*

- Animation :
  - Animation au chargement : *@keyframes*

- Gestion des blocs :
  - Gestion inline ou block : *display*
  - Position : *position*
  - Marge intérieure : *padding*
  - Marge extérieure : *margin*
  - Centrage des « inline » : *text-align*
  - Centrage des « block » : *margin auto*
  - Bordures : *border*
  - Couleur d'arrière-plan : *background-…*
  - Largeur max : *widht-max*
  - Hauteur : *height*

- Apparence dynamique :
  - Modification au passage souris : *:hover*

1- DEMARRAGE :

⇨ Dans *Visual Studio Code*, ouvrir dans un répertoire de travail, 2 nouveaux fichiers textes, l'un sera nommé *pageAccueilSite.html*, l'autre *pageAccueilSite.css*.

⇨ Dans le fichier *pageAccueilSite.html*, écrire le « *! Doctype* » en écrivant simplement ! suivi d'une validation sur la touche *Entrée*.

⇨ Rajouter dans la partie <*head*> la balise qui lie ce fichier au fichier *pageAccueilSite.css* : taper simplement « *link* » jusqu'à ce que l'autocomplétion de *Visual Studio Code* reconnaisse cette balise et valider avec la touche *Entrée*. Compléter l'attribut « *href* » .

⇨ Saisir un titre dans la balise <*title*>.

⇨ Dans *Visual Studio Code*, avec un clic droit sur l'onglet du fichier *.html,* copier le chemin de ce fichier. Ouvrir le navigateur *Firefox* et coller ce lien dans la barre d'URL afin de pouvoir lire ce fichier.

2- CSS DU BLOC <BODY> :

Point Cours : Propriétés Css pour définir les couleurs et les polices d'écriture :

- Couleur d'arrière-plan : `background-color:`
- Couleur du texte : `color:`
- Police d'écriture : `font-family:`
- Taille de la police : `font-size:`

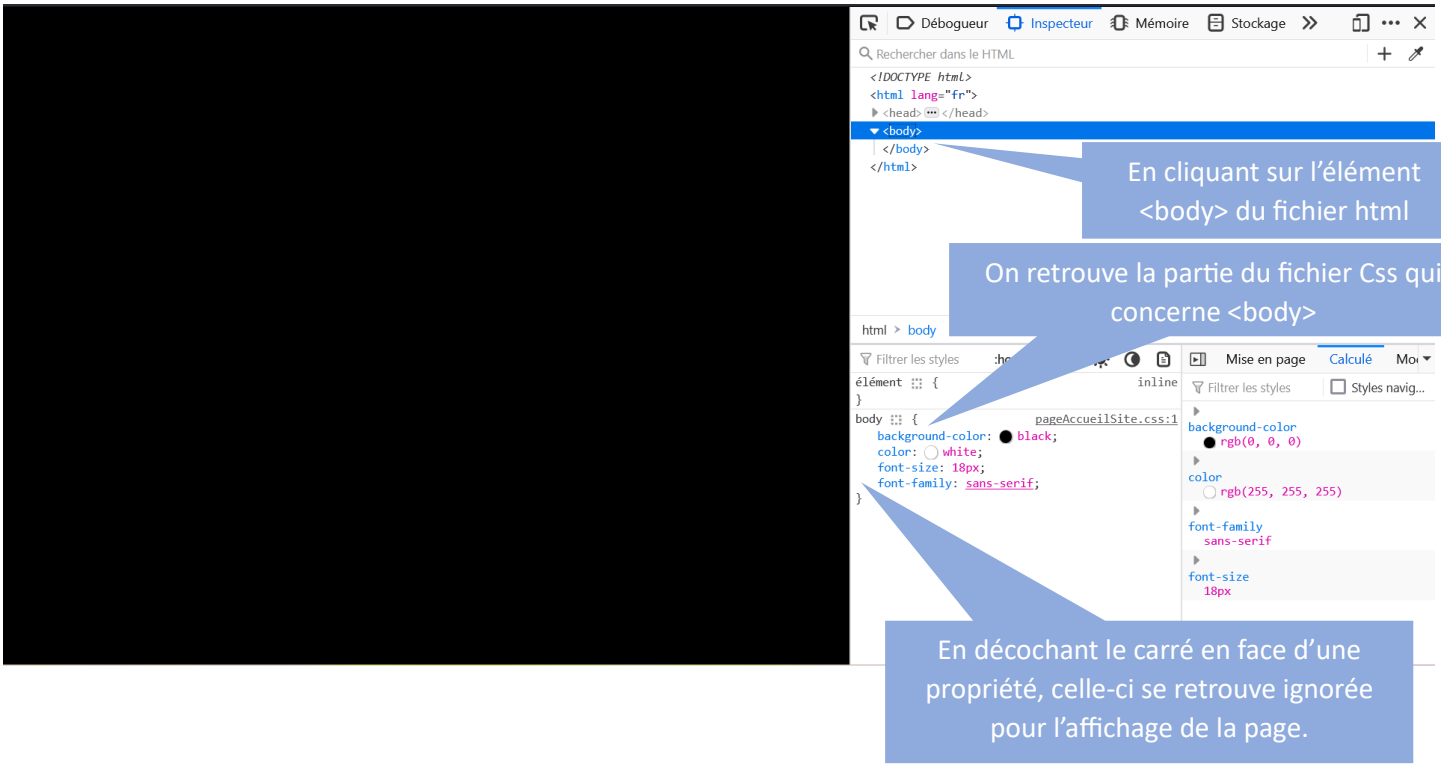
`font-family:`	`font-size:`
Les polices les plus courantes (sans-serif, arial, Verdana, …) fonctionnent nativement sur la plupart des navigateurs. Elles sont proposées en autocomplétion dans *Visual Studio Code*. Pour créer un site attrayant, il est souvent nécessaire d'utiliser des polices importées, gratuites ou payantes. La manipulation d'import est très facile (voir sur google). 	La taille de la police est définie est **absolue** (pixels le plus souvent) ou en **relatif** (**rem** ou em). L'unité « ***rem*** » signifie « *root em* », root signifiant « *racine* ». Dans cet exercice, on définira une taille **absolue** `font-size: 18px;` pour l'élément <body> , qui est l'élément racine de la page html. Pour les autres éléments de la page, on définira des tailles relatives en « ***rem*** » . Par exemple, si on définit dans un paragraphe de <body> une taille `font-size: 1.2rem;` , la taille des caractères y sera de (18×1.2) px . Si on avait défini une taille `font-size: 0.5rem;` , la taille aurait été de (18×0.5) px.

Dans la page à construire, la couleur d'arrière-plan sera en noir, la couleur du texte en blanc, la taille de la police 18px et la police d'écriture sera le sans-serif.

⇨ Compléter le fichier Css en donnant les valeurs de ces 4 propriétés pour l'élément body. Sauvegarder le fichier.

⇨ Visualiser la page html sur Firefox ….. vous obtenez une page noire.

⇨ Avec un clic droit sur la page, choisir « *Inspecter* » …. Vous obtenez l'affichage suivant :



⇨ Tester le décochage des propriétés …. La modification est prise en compte directement. Par contre si on recharge la page avec l'icone ou la touche F5 du clavier, on retrouve la page définie dans les fichiers *html* et *css*.

3- CREATION DU MENU DE NAVIGATION – PARTIE HTML :

Pour obtenir un menu, on crée généralement un bloc <*nav*> dans lequel insère une liste, ici de 5 items qui seront des liens vers les différentes pages du site.

⇨ Créer un bloc <*nav*> en écrivant simplement *nav* jusqu'à la reconnaissance de l'autocomplétion et en validant ensuite avec la touche entrée.

⇨ Dans ce bloc <*nav*>, créer un bloc <*ul*> en utilisant toujours l'autocomplétion.

⇨ Dans ce bloc <*ul*>, créer 5 blocs de liens <*a*>. Pas la peine de renseigner l'attribut href.

⇨ Dans chacun de ces 5 blocs <*a*>, créer 1 bloc <*li*> qui contiendra un des titres, des items du menu :

Process Work FAQs Webflow Contact

Bien sûr, ne pas hésiter à utiliser le copié-collé.

On obtient à l'affichage …..

- Process
- Work
- FAQs
- Webflow
- Contact

L'affichage n'a rien à voir avec le rendu souhaité. C'est normal, car les listes non ordonnées <*ul*> s'affichent toujours en colonne avec des puces et les liens *html* s'affichent toujours par défaut en bleu et soulignés. C'est la partie *Css* qui permettra d'obtenir le résultat suivant :

Process Work FAQs Webflow Contact

4- CREATION DU MENU DE NAVIGATION – PARTIE CSS :

a. MODIFICATION DE L'APPARENCE DES LIENS :

⇨ Rajouter dans le Css :

Cela signifie que pour tous les éléments <a> de la page, la propriété `text-decoration:` qui gère le soulignement, aura la valeur `none` . D'autre part, pour ces éléments <a>, la propriété `color:` qui gère la couleur du texte, aura la valeur `unset`, ce qui permet **de rétablir la valeur de couleur héritée de l'élément parent, qui est ici la couleur blanche** :

```
<body>
    <nav>
        <ul>
            <a href=""><li>Process</li></a>
            <a href=""><li>Work</li></a>
            <a href=""><li>FAQs</li></a>
            <a href=""><li>Webflow</li></a>
            <a href=""><li>Contact</li></a>
        </ul>
    </nav>
</body>
```

La couleur blanche définie dans l'élément <*body*> **est héritée** :

- <*nav*> hérite de `color: white;` car <*nav*> est un fils de <body>
- <*ul*> hérite de `color: white;` car <*ul*> est un fils de <*nav*>
- <*a*> hérite de `color: white;` car <*a*> est un fils de <*ul*>

Point Cours : Css fonctionne sur le principe de l'HERITAGE :

- Certaines propriétés sont héritées par les éléments enfants (*color, font-size, font-family,* …).
- D'autres ne le sont pas (*width, margin, padding*, …)

b. MODIFICATION DE L'AFFICHAGE DU MENU :

⇨ Rajouter dans le Css les parties suivantes. On les complètera ensuite directement dans le navigateur Firefox.

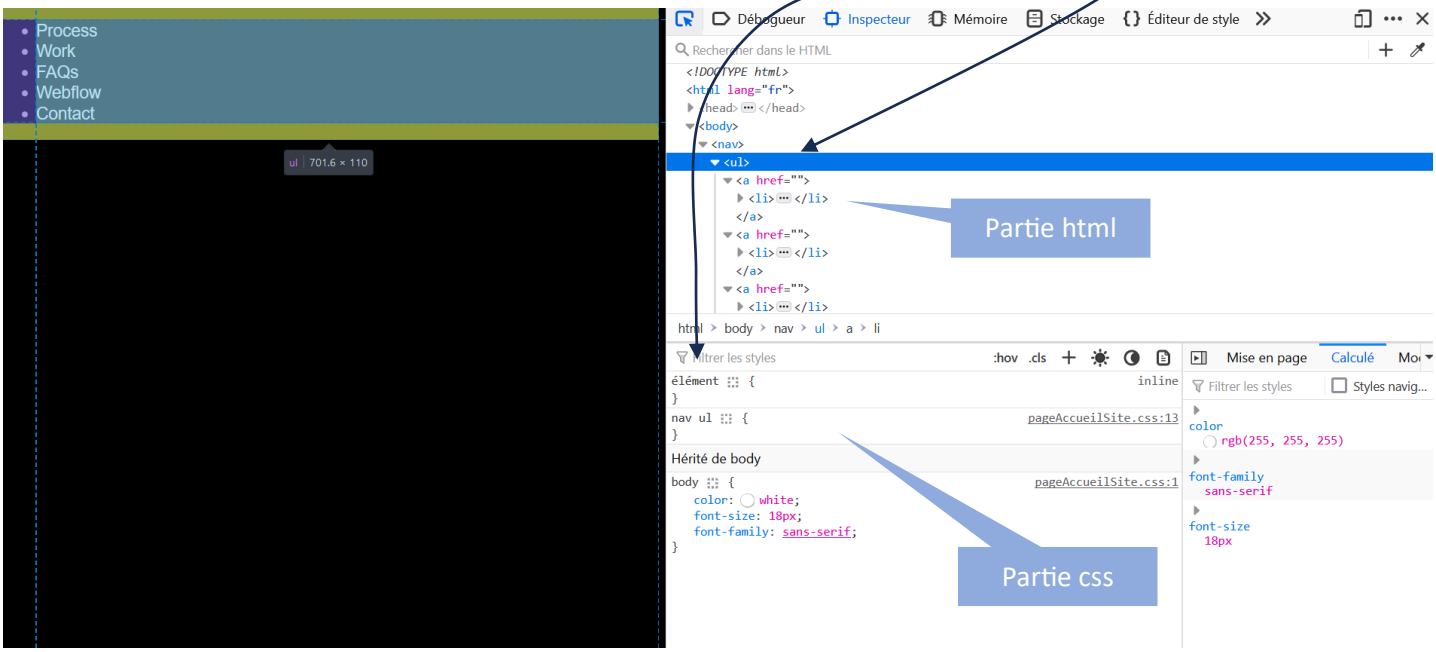
```
1   body{
2       background-color: black;
3       color: white;
4       font-size: 18px;
5       font-family: sans-serif;
6   }
7
8   /*  Gestion de <nav>    */
9   a {
10      text-decoration: none;
11      color:unset;
12  }
13  nav ul{
14
15  }
16  nav ul li{
17
18  }
```

Ligne de commentaire. Elle commence par /* et se termine par */

Les propriétés que l'on écrira entre les 2 accolades s'appliqueront aux blocs <ul> qui se trouvent DANS un bloc <nav>

… pareil pour les blocs <li> qui se trouvent DANS un bloc <ul> qui se trouvent eux-mêmes DANS un bloc <nav>

⇨ Dans Firefox, utiliser l'outil de sélection  pour sélectionner le bloc <*ul*> dans la zone graphique, et cliquer ensuite dans ce bloc. Le bloc <*ul*> se retrouve en surbrillance dans la partie html à droite. Les propriétés Css qui concernent <*ul*> sont affichées dans la partie Css.



Dans la partie Css affichée par le navigateur, on va à présent rajouter des propriétés, directement dans le navigateur, afin de pouvoir observer leurs effets immédiatement.

⇨ On commence par afficher une bordure blanche de 1px, en trait continue (*solid*) au bloc <*ul*> qui se trouve dans <*nav*> :

```
nav ul {                                    pageAccueilSite.css:13
    border: 1px solid white;
}
```

⇨ On continue, en affichant une bordure rouge autour des éléments <*li*>, qui se trouvent dans <*ul*>, qui se trouve dans <*nav*> :

```
nav ul li {                                          pageAccueilSite.css:17
    border: 1px solid red;
}
```

⇨ On continue, en rendant *INLINE* ces blocs <*li*>. A leur création, ces blocs <*li*> occupent toute la largeur de la page car ce sont des balises dites BLOCK. En modifiant la valeur de leur propriété *display* : `display: inline;` , le bloc <*li*> devient un élément INLINE. Il n'occupe plus la largeur de la page. Ainsi, les éléments <*li*> de la liste se potironneront les uns à la suite des autres, sans retour à la ligne :

```
nav ul li {                                          pageAccueilSite.css:17
    border: 1px solid red;
    display: inline;
}
```

⇨ On continue, en augmentant la taille des caractères du texte dans <*li*>. On utilise l'unité relative **rem** , avec ici 1rem = 18px. On utilise dans <*li*> une police 1.2 fois plus grande que celle définie dans <*body*> :

```
nav ul li {                                          pageAccueilSite.css:17
    border: 1px solid red;
    display: inline;
    font-size: 1.2rem;
}
```

⇨ On continue en espaçant les items du menu de navigation. On rajoute 10px de marge intérieure et 20px de marge extérieure :

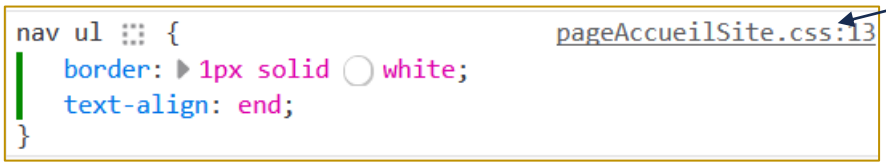
```
nav ul li {                                          pageAccueilSite.css:17
    border: 1px solid red;
    display: inline;
    font-size: 1.2rem;
    padding: 10px;
    margin: 20px;
}
```

⇨ Tester le décochage des propriétés …. la modification étant prise en compte directement, ces « décochages » permettent de bien se rendre compte de l'importance de chacune des propriétés. On peut ainsi mieux les mémoriser.

⇨ On continue en positionnant tous ces items du menu, sur la partie droite de la page. Comme ces éléments <*li*> sont du type INLINE, on peut utiliser la propriété `text-align:` sur l'élément « père » qui contient les <*li*>. Il s'agit ici de l'élément <*ul*>. `text-align:` peut essentiellement prendre comme valeur « *start* » ou « *center* » ou « *end* ». On choisit ici « *end* ». Toujours directement dans le navigateur, pour le bloc <*ul*> :

```
nav ul {                                             pageAccueilSite.css:13
    border: 1px solid white;
    text-align: end;
}
```

⇨ SAUVEGARDER à présent ces changements dans le fichier Css. Pour cela, il faut cliquer sur le nom du fichier « *pageAccueilSite.css* », juste à droite … le fichier s'ouvre dans le navigateur … et conclure en cliquant sur « *Enregistrer* ».



⇨ Vérifier bien que les changements ont été pris en compte dans *Visual Studio Code* (faire attention aux messages d'alertes). Ne recharger la page du navigateur que si vous êtes sûr que les modifications apportées ont été enregistrées.

Le fichier « *pageAccueilSite.css* » contient pour l'instant les lignes suivantes :

```css
body{
    background-color: black;
    color: white;
    font-size: 18px;
    font-family: sans-serif;
}

/*  Gestion de <nav>   */
a {
    text-decoration: none;
    color:unset;
}
nav ul{

    border: 1px solid white;
    text-align: end;
}
nav ul li{

    border: 1px solid red;
    display: inline;
    font-size: 1.2rem;
    padding: 10px;
    margin: 20px;
}
```

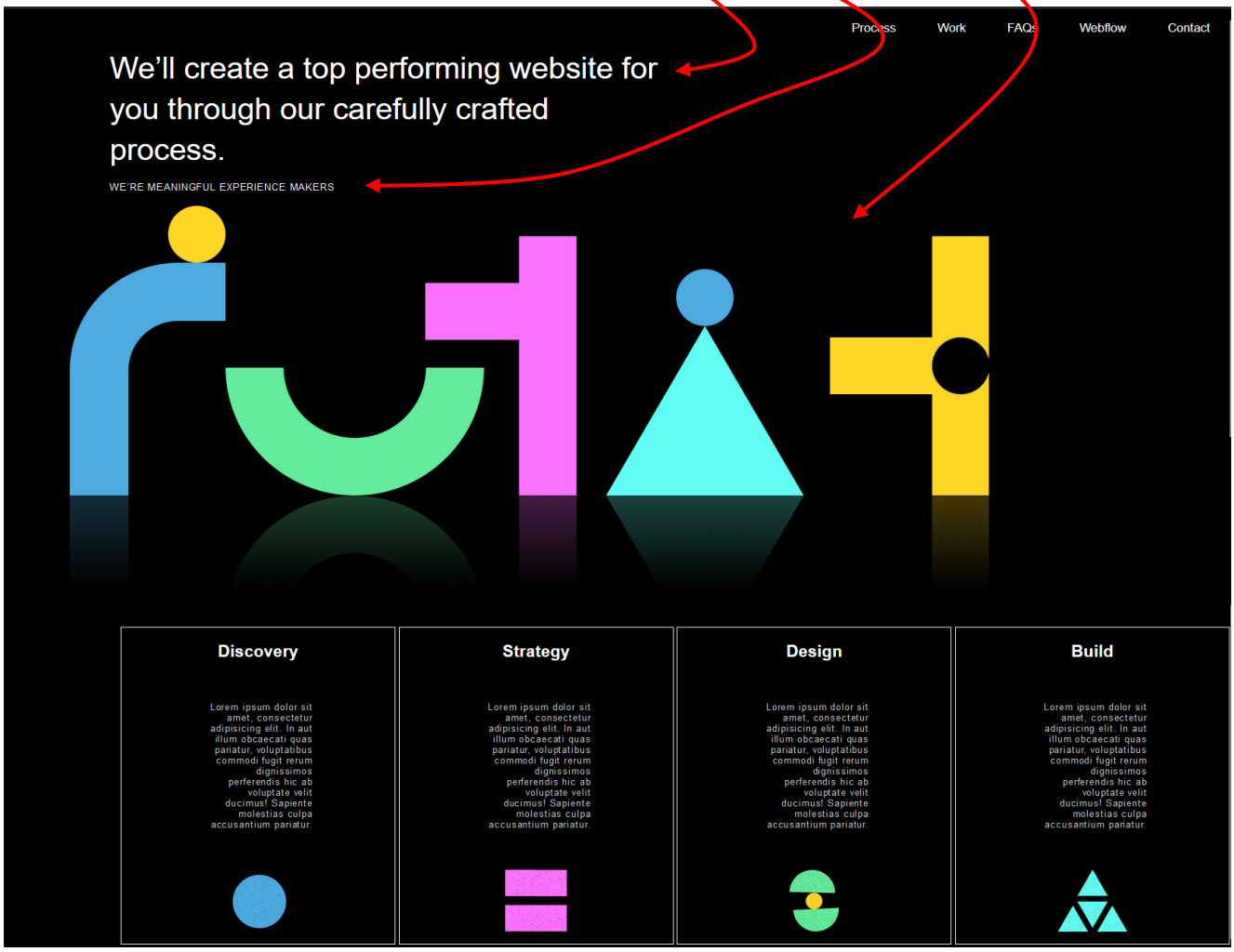
La partie « menu de navigation » est presque terminée. On a pour l'instant manipulé les nouvelles propriétés css suivantes :

Point Cours : Nouvelles propriétés Css à connaitre :

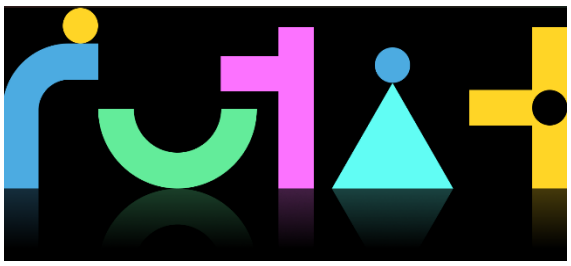
- Rendre un élément INLINE : `display: inline;`
- Rendre un élément BLOCK : `display: block;`
- Enlever le soulignement d'un lien <a> : `text-decoration: none;`
- Enlever la couleur bleue d'un lien <a> : `color:unset;`
- Centrer un bloc INLINE → dans son conteneur père : `text-align:center;`

5- CREATION DE LA PARTIE PRINCIPALE DE LA PAGE – PARTIE HTML :

Bien que la mise en forme du menu de navigation ne soit pas totalement terminée, on continue la création de la page en s'intéressant à la partie principale qui comprend un titre, un paragraphe et une image :



⇨ Télécharger l'image centrale dans votre répertoire (lien en cliquant sur l'image ci-contre avec la touche CTRL appuyée, pour l'ouvrir dans le navigateur, puis « *click droit et enregistrer sous* »).



Dans *Visual Studio Code*, dans le fichier *pageAccueilSite.html* :

⇨ Créer un bloc <*section*>. Le tag *HTML* <*section*> a été récemment introduit dans HTML5. Il est utilisé pour regrouper des contenus liés logiquement et créer des *sections* de page.

Dans ce bloc <section>, créer les éléments fils suivants :

⇨ Créer un bloc <*h1*> qui contiendra le texte : « `We'll create a top performing website for you through our carefully crafted process.` »

⇨ Créer un bloc <*p*> qui contiendra le texte : « `WE'RE MEANINGFUL EXPERIENCE MAKERS` »

⇨ Insérer l'image centrale téléchargée juste avant. Ajouter des attributs *alt* et *title*.

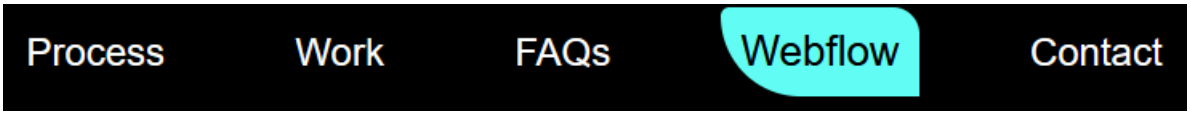
6- CREATION D'UNE APPARENCE DYNAMIQUE SUR LE MENU DE NAVIGATION – PARTIE CSS :

Avant de poursuivre sur le bloc <*section*> que l'on vient de créer, on revient sur la partie Css du menu de navigation. L'objectif dans ce paragraphe est de modifier l'affichage des items, lors de leur survol par la souris :

Ou :

Ou :

Ou :



Ou :

Pour créer une apparence dynamique qui modifie le code css de l'élément qui est survolée par la souris, on utilise ce que l'on appelle la pseudo-classe `:hover`. On rajoute ainsi dans le fichier css :

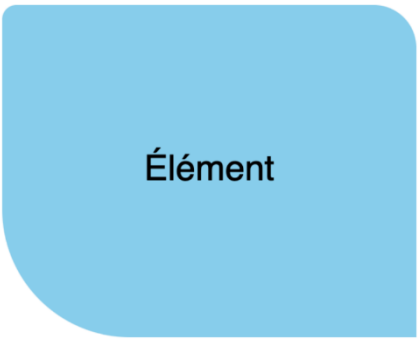
Pour tous les <a> ,contenus dans un <ul>, lui-même contenu dans <nav>, en cas de survol souris, les propriétés entre accolades s'appliquent.

Pour le 1er fils <a> ,contenu dans un <ul>, lui-même contenu dans <nav>, en cas de survol souris, la propriété entre accolades s'applique.

```css
nav ul a:hover{
    color: black;
    border-radius: 10px 30px 0px 90px;
    font-size: 1.3rem;
}
nav ul a:nth-child(1):hover{
    background-color: #4cabe1;
}
```

La propriété `border-radius:` permet d'arrondir les coins de l'élément. On repère soit 1 seul rayon en px, soit 4 rayons en px. Dans ce dernier cas, la 1ère valeur est le rayon du coin HAUT-GAUCHE. On tourne ensuite dans le sens des aiguilles d'une montre :

```css
border-radius: 10px 30px 0px 90px;
```



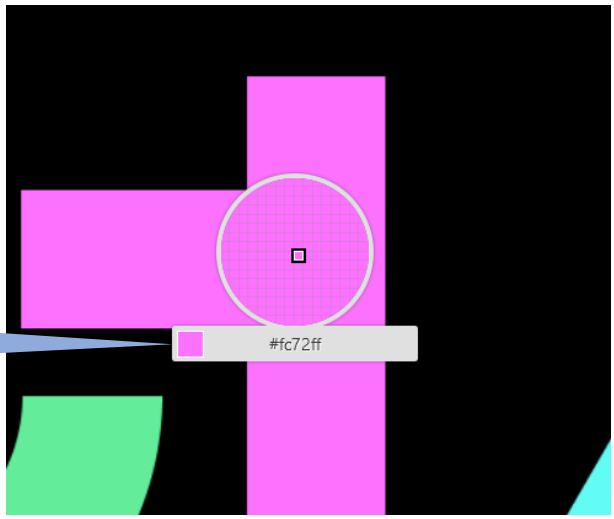
Note : sur les écrans tactiles, `:hover` est problématique, voire impossible. La pseudo-classe `:hover` n'y est jamais valide, ou seulement pendant un très court instant après avoir touché l'élément. Puisque les appareils à écrans tactiles sont très courants, il est important que les développeurs web ne placent pas de contenu accessible seulement lors du survol, puisque ce contenu sera caché pour les utilisateurs de tels appareils.

⇨ Dupliquer dans le Css, à la suite, le bloc ci-contre 4 fois, afin de définir une couleur différente pour chacun des items du menu. L'élément <*a*> du deuxième item sera l'élément fils n°2, etc ….

```
nav ul a:nth-child(1):hover{
    background-color: #4cabe1;
}
```

Les codes hexa des couleurs sont à relever avec l'outil pipette , **sur l'image centrale** déjà affichée sur la page. Cela permet de faire un rappel des couleurs sur cette page : TRES IMPORTANT pour le rendu ….

Code hexa de la couleur du pixel ciblé avec l'outil



⇨ Tester l'ensemble. Si ok, mettre les propriétés de bordures rouge et blanches du css en commentaire (/* et */). La partie gestion du menu de navigation est terminée.

Point Cours : La pseudo-class :hover permet de définir les propriétés Css qui seront appliquées à un élément, si celui-ci est survolé par la souris.

7- CREATION DE LA PARTIE PRINCIPALE DE LA PAGE – PARTIE CSS :

⇨ On commence par ajouter un commentaire dans le fichier css afin de bien séparer ce qui va être écrit et qui concernera le bloc <*section*>, de ce qui a déjà été écrit et qui concernait le bloc <*nav*>.
D'autre part, sachant que l'on va définir des propriétés css pour les éléments <*h1*>, <*p*> et <*img*> qui sont des fils de <*section*>, on prépare les sélecteurs avec des ensembles entre accolades vides.
Pour mieux contrôler la dimension et la position des éléments <h1> et <p>, on demande l'affichage d'une bordure.
On ajoute ainsi dans le fichier Css, le contenu ci-contre :

```
/*  Gestion de <section>   */
section{

}
section h1{
    border:1px solid white;

}
section p{
    border:1px solid white;

}
section img{

}
```

On s'occupe tout d'abord de l'image, car on voit bien qu'elle déborde manifestement de l'écran. Sa taille d'enregistrement est de 1440px par 642px.

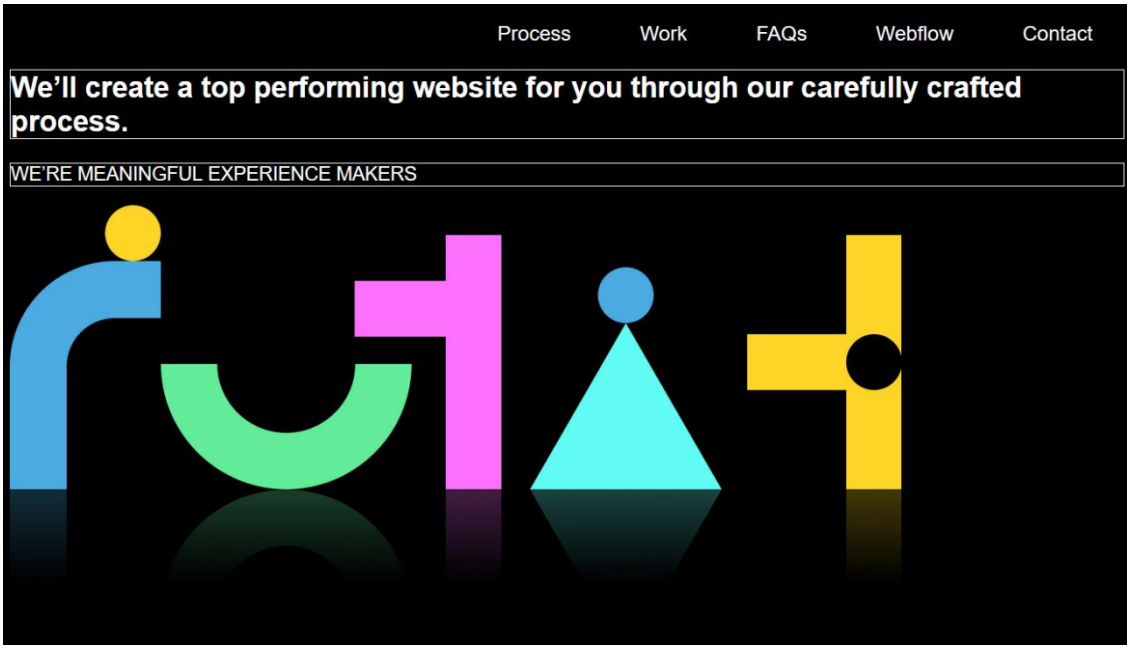
⇨ Dans le navigateur, sélectionner l'image dans la partie *html* afin de faire apparaitre dans la partie Css :

```
élément { }                                   inline
section img { }                               pageAccueilSite.css:50
Hérité de body
body { }                                      pageAccueilSite.css:1
    color: white;
    font-size: 18px;
    font-family: sans-serif;
```

⇨ Insérer la propriété *max-width* qui limitera la largeur de l'image à 80% de la largeur de l'élément parent qui est ici le bloc <*section*> et dont la largeur est égale à celle de l'écran.

```
section img {                                 pageAccueilSite.css:50
    max-width: 80%;
}
```

On obtient :



On va à présent CENTRER cet élément dans son bloc parent <*section*> , qui lui occupe la largeur de la page. Les éléments images <*img*> sont de type INLINE. Pour centrer ce type d'élément, le moyen le plus simple est d'appliquer la propriété `text-align: center;` à son bloc parent.

⇨ Dans le navigateur, sélectionner l'élément <*section*> dans la partie html afin de faire apparaitre dans la partie Css

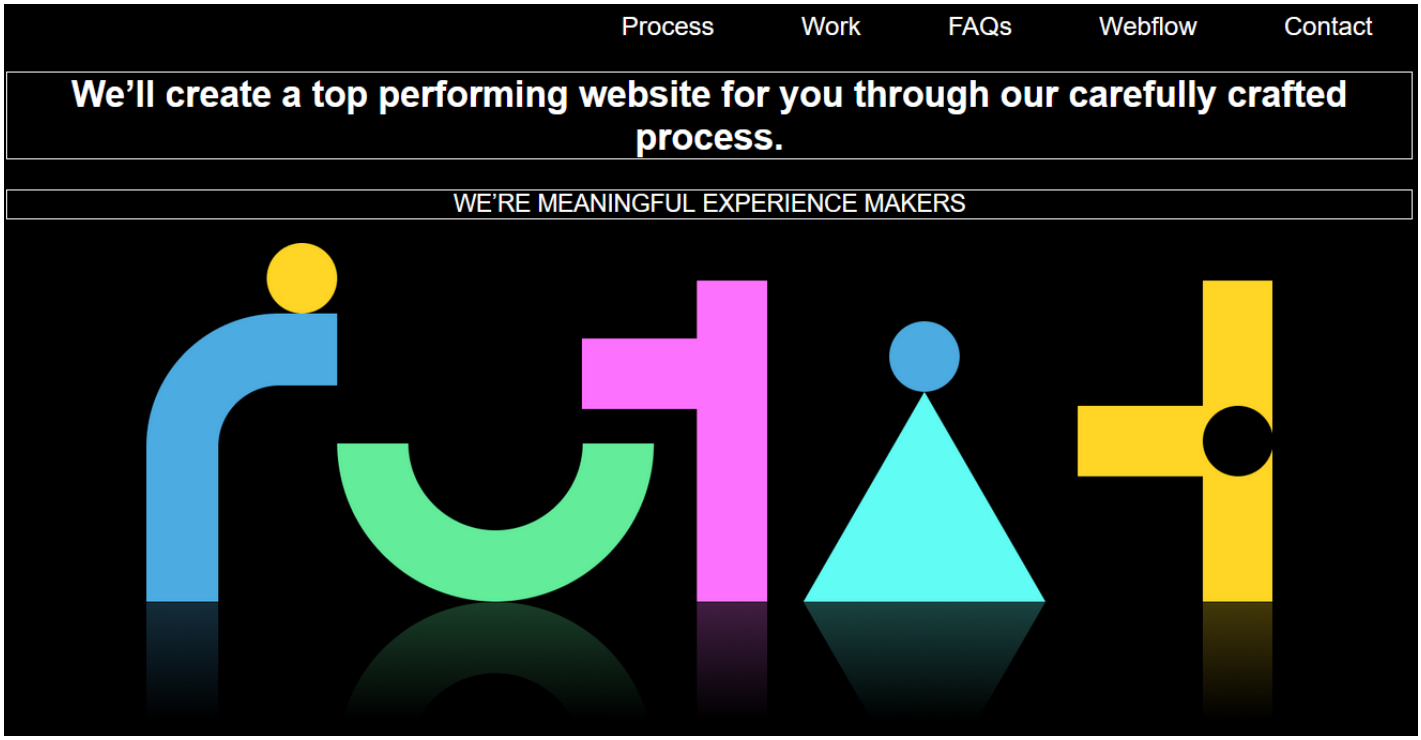
```
pageAccueilSite.css:48
section {
}
```

⇨ Insérer la propriété *text-align* et lui donner la valeur *center* :

Cliquer sur le nom du fichier juste à coté et sur enregistrer.

```
pageAccueilSite.css:58
section {
    text-align: center;
}
```

On obtient ainsi :

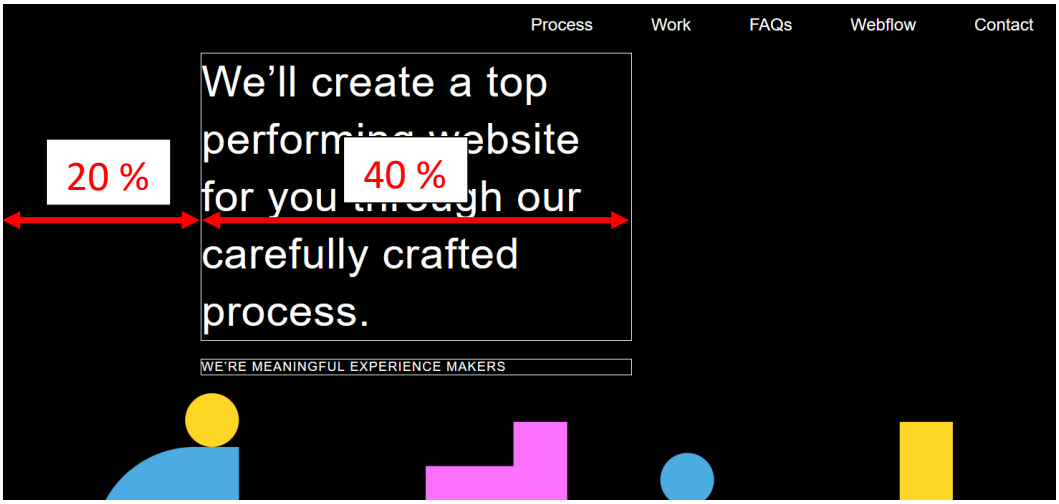


On constate que les blocs <*h1*> et <*p*> ont aussi hérité de la propriété `text-align: center;` définie dans <*section*> . Ainsi les textes dans ces blocs <*h1*> et <*p*> se sont centrés aussi. En sélectionnant le bloc <*h1*> dans le navigateur, on voit d'ailleurs apparaitre dans la partie Css :

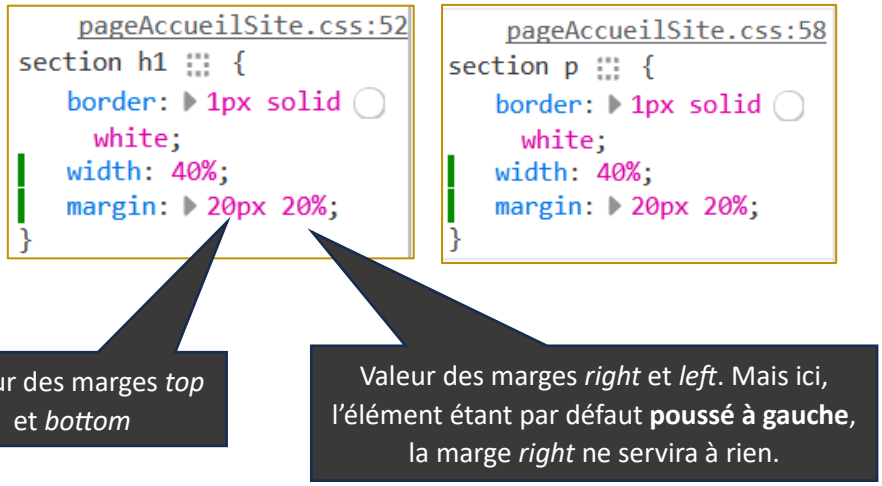
Hérité de section

On s'intéresse à présent aux éléments <*h1*> et <*p*>. Le rendu final que l'on souhaite avoir est le suivant :

Il s'agit tout d'abord de limiter la largeur de <*h1*> et <*p*> à 40% de celle du bloc parent ET de définir une marge gauche égale à 20% de celle du bloc parent ET des marges haut et bas de 20px.



⇨ On rajoute ainsi les propriétés css suivantes directement dans le navigateur en utilisant les outils du développeur :



On continue la mise en forme des éléments <*h1*> et <*p*>, en modifiant la taille de la police utilisée, l'espacement entre lettres, la finesse des caractères et l'écartement interlignes. Ces « *petits changements* » paraissent mineurs, mais ils vont permettre de rendre la page bien plus attrayante. Par simplicité, on ne modifiera pas la propriété `font-family: sans-serif;`. Mais le type de police utilisée **joue énormément** sur le rendu d'une page. De nombreuses « *fonts* » ont été créées à travers le monde, certaines sont gratuites, d'autres ne le sont pas ….

⇨ On rajoute ainsi les propriétés css suivantes directement dans le navigateur, afin de pouvoir observer directement leur effet sur l'affichage de la page :

```css
section h1{
    border:1px solid white;
    width:40%;
    margin:20px 20%;
    font-weight: 100;
    font-size: 3rem;
    line-height: 4rem;
    letter-spacing: 1px;
    text-align: left;
}
```

```css
section p{
    border:1px solid white;
    width:40%;
    margin:20px 20%;
    font-size: 0.8em;
    font-weight: 400;
    letter-spacing: 1px;
    text-align: left;
}
```

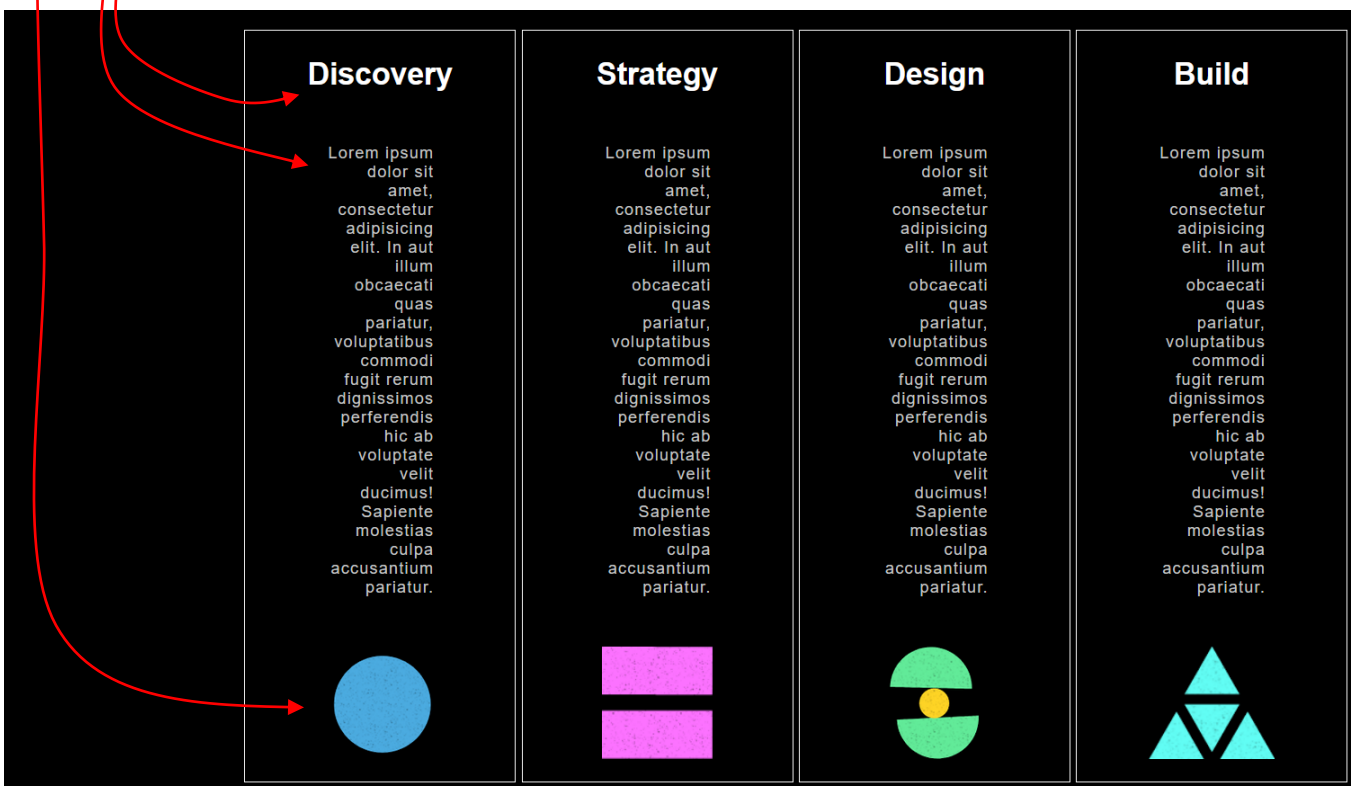
⇨ Sauvegarder le css en cliquant sur le nom du fichier et sur enregistrer.

⇨ Sur *Visual Studio Code*, supprimer les propriétés `border:1px solid white;` pour les éléments <*h1*> et <*p*>.

8- CREATION DU FOOTER DE LA PAGE – PARTIE HTML :

Le résultat que l'on souhaite obtenir pour le bas de page, est donné ci-dessous. Ce bloc <*footer*> contient 4 blocs que l'on nommera avec le nom générique <*div*>. Chacun d'eux contiendra lui-même, 3 blocs :

- Un bloc <h2> pour le titre « Discovery » ou « Stategy » ou « Design » ou « Build »,
- Un bloc <p> qui contiendra le texte « *Lorem ipsum* …. »,
- Une image  ou  ou  ou  que vous téléchargerez dans votre répertoire (lien en cliquant sur chacune des images ci-dessus avec la touche CTRL appuyée, pour l'ouvrir dans le navigateur, puis « *click droit et enregistrer sous* »).



On détaille ici ce qu'il s'agit de faire :

⇨ Créer un bloc <*footer*>.

⇨ Dans ce bloc < *footer* >, créer un premier éléments fils <*div*>

Dans ce bloc <*div*> :

⇨ Créer un bloc <*h2*> qui contiendra le texte : « `Discovery` »

⇨ Créer un bloc <*p*> qui contiendra le texte : « `Lorem ipsum …..` »

⇨ Insérer la première image téléchargée juste avant. Ajouter à la balise <*img*>, l'attribut `class="petites"` et des attributs *alt* et *title*

⇨ Par un copié-collé, dupliquer le bloc <*div*> afin d'en avoir 4 en tout. Modifier les titres et les liens images.

9- CREATION DU FOOTER DE LA PAGE – PARTIE CSS :

⇨ Afficher la page sur le navigateur …. on est loin du compte …..

⇨ On commence par créer ses blocs dans le Css, et comme précédemment, on les complètera directement par l'intermédiaire des outils du développeur du navigateur. On obtient ainsi un fichier Css qui ressemble à celui-ci-contre :

```
/******  Gestion de <section>   *******/
section{ ···
}
section h1{ ···
}
section p{ ···
}
section img{ ···
}

/******  Gestion de <footer>   *******/
*{
    border:1px solid white;
}
footer{

}
footer div {

}
footer div h2{

}
footer div p{

}
.petites{

}
```

On peut fermer les volets pour les parties qui ont déjà été traitées.

On prévoit une bordure sur tous les « *descendants* » de <*body*> , afin de faciliter la mise au point …

On commence par réduire la largeur des 4 <*div*> à 20% de la largeur du bloc parent. On alignera ensuite ces 4 sur le coté droit du bloc parent …… cela donne dans le détail :

⇨ Rajouter la propriété Css suivante directement dans le navigateur, afin de pouvoir observer directement l'effet sur l'affichage de la page :

```
footer div ⬚ {    pageAccueilSite.css:81
    width: 20%;
}
```

….. la largeur se réduit, mais comme les éléments <*div*> sont de type BLOCK, les autres éléments qui suivent ne vont pas se mettre à coté, mais en-dessous. Pour arriver à notre objectif, il y a plusieurs solutions. On reprend ici la solution « *historique* » qui consiste à donner à ces blocs <*div*> le statut d'élément INLINE-BLOCK.

⇨ Rajouter ainsi la propriété Css :

```
footer div ⬚ {    pageAccueilSite.css:81
    width: 20%;
    display: inline-block;
}
```

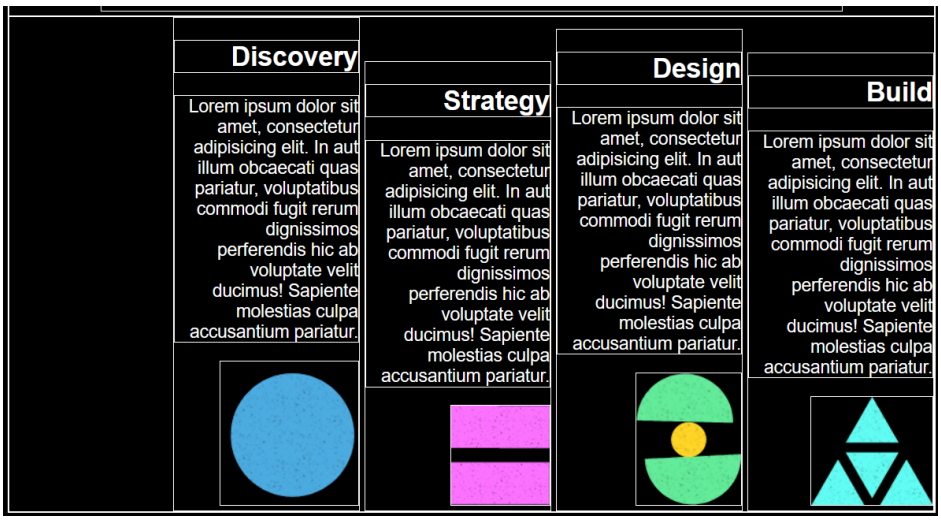
Pour aligner les éléments INLINE ou INLINE-BLOCK sur le coté droit de l'élément parent, on applique la propriété `text-align: end;` sur le bloc PARENT qui est ici <*footer*> :

```
footer ⬚ {        pageAccueilSite.css:78
    text-align: end;
}
```

On constate que ces 4 <*div*> qui sont à présent INLINE-BLOCK, ne sont pas alignés sur le haut de <*footer*>. On peut alors utiliser la propriété

`vertical-align: top;`

sur les blocs .



⇨ On rajoute ainsi cette propriété Css :

```
footer div ⬚ {    pageAccueilSite.css:80
    display: inline-block;
    width: 20%;
    vertical-align: top;
}
```

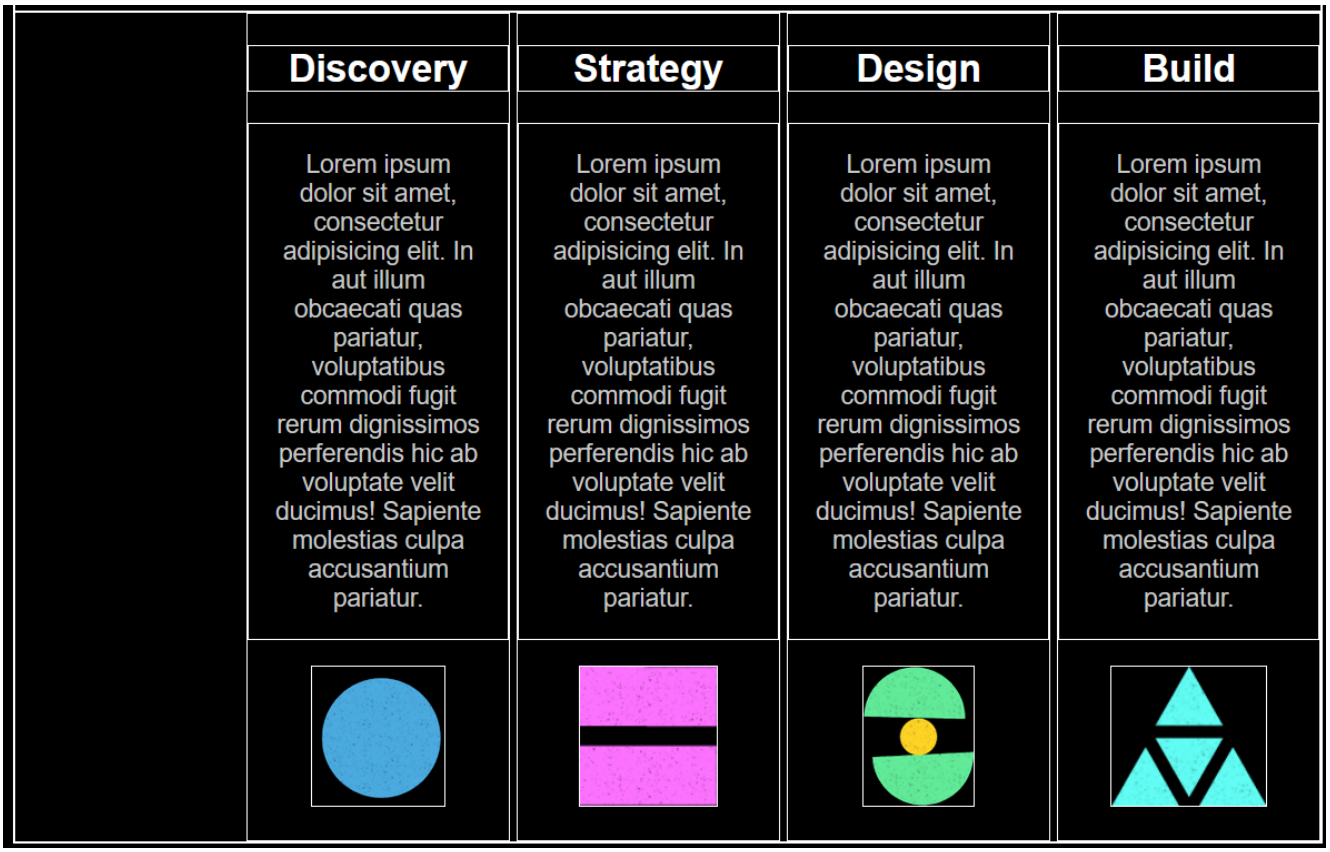
Pour un meilleur rendu, on peut rajouter :

Toujours tester l'effet en cochant - décochant ….. meilleur moyen pour mémoriser, car c'est du visuel …

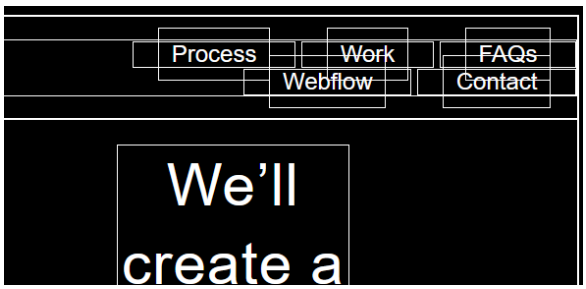
```
footer div ⬚ {    pageAccueilSite.css:80
    display: inline-block;
    width: 20%;
    vertical-align: top;
    ☑ padding-bottom: 20px;
```

Les 4 images du pied de page ont une hauteur d'environ 150 px. On impose par contre au navigateur de les afficher avec une hauteur de 100px.

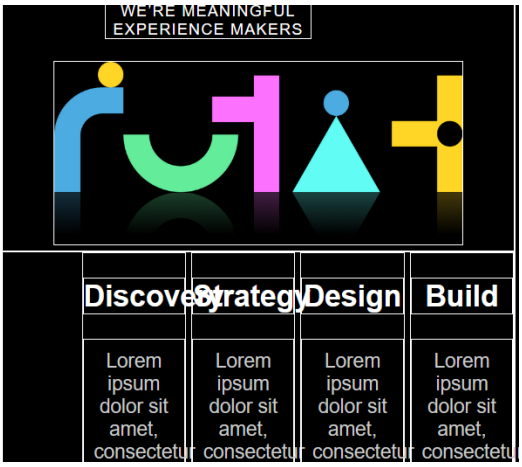
⇨ En utilisant : `color: #c9caca; padding:10%; height:100px; text-align: center;` compléter votre Css pour avoir finalement le bas de page suivant :



⇨ Tester l'affichage de la page pour des largeurs de fenêtre différentes. …..



On constate que cela ne fonctionne plus très bien lorsque la fenêtre devient étroite (écrans de smartphones).



Pour améliorer l'affichage, une modification simple est possible, en rajoutant la propriété `min-width: 250px;` aux blocs <*div*> de <*footer*>. Ainsi la propriété `width:20%;` ne s'appliquera plus lorsque la largeur absolue passe sous le seuil de 250px. Il sera ainsi toujours possible de lire le texte du <*footer*> sur un smartphone. Dans ce cas, les autres <*div*> se positionnent en-dessous. On peut faire mieux, mais par simplicité, on laisse les choses ainsi pour l'instant …

INFO : En cliquant sur l'icône nommé « *vue adaptative* » vous pouvez modifier la largeur de la fenêtre et voir sa largeur en px, en haut de la fenêtre : 

10- RENDRE LE CSS « RESPONSIVE » :

Un site ***responsive*** est un site qui est conçu et développé de façon à pouvoir s'adapter à toutes les résolutions d'écran. C'est donc un seul et même site qui peut être consulté sur ordinateur, sur smartphone ou sur tablette.

On vient juste de voir que, malgré le fait d'avoir défini des largeurs de blocs en % (valeur relative), l'affichage posait problème à un moment donné.

Pour avoir un rendu responsive, il est nécessaire de changer le Css lorsque l'écran devient trop étroit. On utilise alors des requêtes media que l'on appelle : « ***media-queries*** ». Dans le Css, on rajoute des lignes à l'intérieur d'ensembles entre accolades du type :

```
@media screen and (max-width: 750px) {

}
```

Les propriétés qui seront écrites entre les accolades ne s'exécuteront que si la largeur de l'écran est AU MAXIMUM égale à 750px. Ces propriétés deviennent prioritaires.

Pour remédier aux défauts mis en évidence juste avant, on insère dans le Css des blocs `@media screen, qui,` si la largeur de l'écran est inférieure à 750px, demandent au navigateur de :

- Sur <footer>, remplacer le `text-align: end` par `text-align: center`
- Sur <section> :
  - pour <h1> et <p> : remplacer les propriétés `width:40%;margin:20px 20%;` par : `width:90%;margin:20px auto;`
  - Pour <img> : remplacer `max-width: 80%;` par : `max-width: 95%`
- Sur <li> de <nav> : remplacer les propriétés `font-size: 1.2rem; padding: 10px; margin: 20px;` par `font-size: 1rem; padding: 5px; margin: 5px`

Concrètement, cela se traduit par la création des 3 blocs `@media screen` suivant :

⇨ Rajouter dans le fichier Css, dans la partie <*footer*> :

```
@media screen and (max-width: 750px) {
    footer {
        text-align: center;
    }
}
```

⇨ Rajouter dans le fichier Css, dans la partie <*section*> :

```css
@media screen and (max-width: 750px) {
    section p , section h1{
        width:90%;
        margin:20px auto;
        text-align: left;
    }
    section img{
        max-width: 95%;
    }
}
```

⇨ Rajouter dans le fichier Css, dans la partie <*nav*> :

```css
@media screen and (max-width: 750px) {
    nav ul li{
        font-size: 1rem;
        padding: 5px;
        margin: 5px;
    }
}
```

⇨ Tester l'ensemble. Si le résultat ne vous satisfait pas, vous pouvez rajouter autant de blocs `@media screen` que nécessaire. Par exemple :

```css
@media screen and (min-width: 750px) and (max-width: 1000px) {


}
```

11- RAJOUT D'ANIMATIONS :

Les sites professionnels comportent de plus en plus d'animations qui rendent la navigation attrayante et poussent le visiteur de la page à revenir. On ouvre ici encore un nouveau chantier. Pour maitriser totalement les animations en Css, il est nécessaire d'investir du temps. Dans ce Tp, par simplicité ; on se contente de donner 3 animations simples et on les commente avec un minimum d'explications.

a. ANIMATION QUI SE DECLENCHE **AU CHARGEMENT DE LA PAGE :**

⇨ Saisir le code ci-contre dans la zone <*section*> de votre fichier Css. On donne du sens aux différentes lignes :

```
/******  Gestion de <section>   *******/

section{ ···
}
section h1 { ···
}
section p { ···
}
section img{ ···
}

@media screen and (max-width: 750px) { ···
}

section p {
    animation: animationAuChargement 1000ms 100ms;
}

@keyframes animationAuChargement{
0% {
    transform: translateX(200px);
    letter-spacing: 5px;
}
50% {
    transform: translateX(100px);
    letter-spacing: -5px;
}
100% {
    transform: translateX(0);
    letter-spacing: 1px;
    }
}
```

On ferme les volets des parties déjà traitées

On applique au bloc <*p*> de <*section*> …

… les propriétés définies dans une animation ici nommée « *animationAuChargement* » …. ceci durant 1 seconde, 0.1 seconde après le début du chargement de la page.

L'animation nommée « *animationAuChargement* » est définie entre les accolades de ce bloc @keyframes. 0%{ … } donne les propriétés à appliquer au début et 100%{ … } celles à appliquer à la fin.

⇨ Tester cette animation qui ne se déclenche qu'au chargement de la page.

b. ANIMATION QUI SE DECLENCHE **AU SURVOL DE LA SOURIS :**

⇨ Saisir le code ci-contre dans la zone <*footer*> de votre fichier Css. On donne du sens aux différentes lignes :

```
/******  Gestion de <footer>   *******/
*{ ···
}
footer{ ···
}
footer div { ···
}
footer div h2{ ···
}
footer div p{ ···
}
.petites{ ···
}
```

On ferme les volets des parties déjà traitées

Lors du survol souris du bloc <*h2*> de <*footer*> …

… les propriétés définies dans une animation ici nommée « *effetHover1* » sont appliquées …. ceci durant 1 seconde, sans délai après le début du survol.

```
footer h2:hover{
    animation: effetHover1 1000ms 0ms;
}
.petites:hover{
    animation: effetHover2 1000ms 0ms;
}
```

Même chose sur le bloc repéré avec la class *''petites''*, avec les propriétés définies dans une animation nommée « *effetHover2* »

L'animation nommée « *effetHover1* » est définie entre les accolades de ce bloc @keyframes. 0%{ … } donne les propriétés à appliquer au début et 100%{ … } celles à appliquer à la fin.

```
@keyframes effetHover1{
    0% {
        letter-spacing: 5px;
    }
    50% {
        letter-spacing: -10px;
    }
    100% {
        letter-spacing: 1px;
    }
}
@keyframes effetHover2{
    0% {
        transform:scale(1.5);
    }
    50% {
        transform:rotate(45deg);
    }
    100% {
        transform:scale(0.5);
    }
}
```

Même chose pour l'animation nommée « *effetHover2* »

⇨ Tester ces animations qui ne se déclenchent qu'au survol souris de <*h2*> ou <*img*> dans <*footer*>.

12- FINALISATION :

⇨ Enlever les bordures blanches mises sur tous les éléments de la page. Par contre, en rajouter uniquement pour les blocs <*div*> de <*footer*>.

⇨ Tester l'ensemble en réduisant la largeur de la page avec l'outil « *Vue adaptative* » 

⇨ Finaliser l'ensemble de la page en l'améliorant si vous le souhaitez, mais sans y passer trop de temps …

⇨ Mettre votre travail ligne sur *nsibranly.fr* :

Pour l'instant, votre page en ligne (fichier *index.html*) donne dans le navigateur :

- Ma blague à 2 balles

⇨ Rajouter dans ce fichier *index.html* le lien vers le fichier *pageAccueilSite.html* :

- Ma blague à 2 balles
- Page accueil site

⇨ Transférer les fichiers *.html* , *.css* et *.png* sur nsibranly.fr avec le code ***web***.

⇨ Sur votre espace du serveur de *nsibranly.fr*, il commence à y avoir pas mal de fichiers. La **bonne pratique** est de placer les fichiers *.css* dans un dossier nommé « *css* » et les fichiers images dans un dossier nommé « *images* ». Cette opération est possible sur *nsibranly.fr* , grâce au menu accessible en cliquant sur le nom du fichier :

Par contre si le fichier *pageAccueilSite.css* est déplacé dans le dossier */css*, il sera nécessaire de le spécifier dans le fichier *pageAccueilSite.html* **en modifiant** la balise <*link*> :

```
<link rel="stylesheet" href="pageAccueilSite.css">
```
est remplacé par
```
<link rel="stylesheet" href="css/pageAccueilSite.css">
```

Il faudra modifier de même le chemin pour chacune des images si elles sont déplacées dans le dossier « *images* ».

La bonne pratique est d'avoir en ligne un dossier qui soit **un clone** de celui qui est sur votre poste de travail, en local.

13- LA SUITE … :

Votre page d'accueil sur *nsibranly.fr* est pour l'instant la suivante :

- Ma blague à 2 balles
- Page accueil site

La suite de ce Tp consiste à créer une page d'accueil personnalisée. Elle sera constituée d'un menu <*nav*>, d'un bloc <*section*> et d'un bloc <*footer*>. Le menu de navigation renverra le lecteur, entre autres, aux pages déjà réalisées : « *Ma blague à 2 balles* » et « *Page accueil site* ». Pour le reste, choisissez une thématique et créez VOTRE page.

ANNEXE

Lorsque l'on intègre une image, on peut être amené à réduire ses dimensions. Il est par exemple inutile d'utiliser une image de 2500px de large si son affichage sur le site ne dépasse pas les 300px.

Pour réduire les dimensions d'une image, ouvrir cette image avec Gimp qui est un peu l'équivalent gratuit de Photoshop.

- Dans l'onglet « *Image* », choisir « *Echelle et taille de l'image* », modifier les dimensions et cliquer sur « *Mise à l'échelle* ».

- Dans l'onglet « *Fichier* », choisir « *Ecraser* …. » ou « *Exporter sous* … ».