

M1105 Conception de documents et d'interfaces numériques

**Jean-François Berdjugin
Mathieu Vigou**

M1105 Conception de documents et d'interfaces numériques

par Jean-François Berdjugin et Mathieu Vigou

Date de publication 9-11-2020

Table des matières

1. Introduction	1
1. Contexte	1
2. Mise en oeuvre	1
2. HTML	2
1. TP 1 : Bases HTML	2
1.1. Première page	2
1.2. Ajout de la navigation	3
1.2.1. Étape 1	4
1.2.2. Étape 2	4
1.2.3. Étape 3	4
1.3. Pour aller plus loin	5
1.3.1. Ajout d'une favicon	5
1.3.2. Figures	5
1.3.3. Détails et résumés	5
1.3.4. SVG	5
1.3.5. D'autres éléments	6
2. TP 2 : Formulaires	6
2.1. Minimum vital	6
2.1.1. Premier formulaire avec des valeurs par défaut	6
2.1.2. Formulaire avec fieldset et label	7
2.1.3. Formulaire avec espace réservé	7
2.1.4. Formulaire avec mot de passe et réinitialisation	8
2.1.5. Formulaire complet	8
2.2. Pour aller plus loin	9
2.2.1. Éléments et input	10
2.2.2. Champs requis et validation de patterns à la volée	10
2.2.3. Boutons	11
3. CSS	12
1. TP 3 : Bases CSS	12
1.1. Charger une CSS	12
1.2. Sélecteurs, règles	13
1.3. Polices	15
1.4. Pour aller plus loin	15
2. TP4 : Modèle de boîte et position	16
2.1. Modèle de boîte	17
2.1.1. Rappels de cours	17
2.1.2. Exercices	25
2.2. Position	27
2.2.1. Rappels de cours	27
2.2.2. Exercices	29
2.3. Pour aller plus loin	31
2.3.1. Exercice 8	32
2.3.2. Exercice 9	32
2.4. Limitations	32
3. TP 5 : Flottants et Flexbox	33
3.1. Flottants	33
3.1.1. Rappels de cours	33
3.1.2. Exercices	37
3.2. Flexbox	38
3.2.1. Rappels de cours	38
3.2.2. Exercices	51
3.3. Pour aller plus loin	52
4. TP 6 : Système de grille	54
4.1. Exercices	56
4.1.1. Exercice 1	56

4.1.2. Exercice 2	57
4.1.3. Exercice 3	57
4.1.4. Exercice 4	57
4.1.5. Exercice 5	58
4.2. Pour aller plus loin	59
5. TP 7 : CSS Grid	59
5.1. Rappels de cours	59
5.1.1. Les propriétés des conteneurs	60
5.1.2. Les propriétés des enfants	67
5.2. Exercices	71
5.2.1. Exercice 0	71
5.2.2. Exercice 1	71
5.2.3. Exercice 2	72
5.2.4. Exercice 3	72
5.2.5. Exercice 4	73
5.2.6. Exercice 5	73
5.3. Pour aller plus loin	74
6. TP 8 Pour aller plus loin : les médias queries et Materialize	75

Liste des illustrations

2.1. Ma première page	3
2.2. Lien cliquable	4
2.3. Liens externes	4
2.4. Légender une figure	5
2.5. Détails et résumé	5
2.6. Une bande audio, une vidéo sous-titrée	6
2.7. Architecture client serveur	6
2.8. Un premier formulaire avec des valeurs par défaut	7
2.9. Résultat de l'envoi du premier formulaire	7
2.10. Label et fieldset	7
2.11. Espace réservé	8
2.12. Password et réinitialisation	8
2.13. Réponse avec un champ caché	8
2.14. Un formulaire complet	9
2.15. Florilège	10
2.16. Champs requis et validation de patterns	11
2.17. Les boutons	11
3.1. Charger une CSS (page01.html)	12
3.2. CSS sélecteurs (page04.html)	14
3.3. CSS sélecteurs (page05.html)	15
3.4. Police et background (page06.html)	15
3.5. CSS menu imbriqués (page07.html)	16
3.6. Modèle de boîte	17
3.7. exemple 01	18
3.8. les dépassements	20
3.9. arrière plan	22
3.10. <i>outline</i>	23
3.11. <i>display</i>	25
3.12. Design en <i>inlineblock</i>	26
3.13. Un menu horizontal	26
3.14. Bandeau central	27
3.15. Le magasin	27
3.16. Positionnement relatif	28
3.17. Position absolue par rapport au body	28
3.18. Position absolue par rapport à un container	28
3.19. position fixe	29
3.20. Drop Down menu	30
3.21. Menu fixé	31
3.22. Position absolue et relative	31
3.23. Menu vertical avec animations	32
3.24. Zoom et niveaux de gris	32
3.25. Flottant à gauche	34
3.26. Flottant à droite	35
3.27. Deux flottants	36
3.28. Clear	36
3.29. before, after	37
3.30. after et clear	37
3.31. Des images qui flottent	38
3.32. Une image est des colonnes qui flottent	38
3.33. Flex container	39
3.34. Direction des axes	40
3.35. <i>display: flex</i>	40
3.36. <i>display: inline-flex</i>	41
3.37. <i>flex-direction: row</i>	41
3.38. <i>flex-direction: row-reverse</i>	41

3.39. flex-direction: column	42
3.40. flex-direction: column-reverse	42
3.41. flex-wrap	43
3.42. justify-content	44
3.43. align-items	46
3.44. align-content	48
3.45. order	49
3.46. flex-grow	49
3.47. flex-shrink	50
3.48. flex-basis	50
3.49. align-self	51
3.50. Une div centrée	52
3.51. Que du flex	52
3.52. Résolution horizontale de plus de 800 pixels	53
3.53. Résolution horizontale entre 800 et 600 pixels	53
3.54. Résolution horizontale de moins de 600 pixels	54
3.55. Exemple de grille avec Flexbox	56
3.56. Flex grille exercice 1	56
3.57. Flex grille exercice 2	57
3.58. Flex grille exercice 3	57
3.59. Flex grille exercice 4	58
3.60. Flex grille exercice 5	59
3.61. Flexbox CSS Grid	59
3.62. Première grille	60
3.63. repeat	61
3.64. auto	61
3.65. Les nommages	62
3.66. grid-template-areas	63
3.67. Les gouttières	63
3.68. Positionnement de la grille	65
3.69. grid-auto	66
3.70. grid-column 1/3	68
3.71. grid-column 2/-1	68
3.72. grid-column span	68
3.73. grid-area 4 paramètres	69
3.74. justify-content	70
3.75. align-self	71
3.76. CSS Grid exercice 1 et 2	72
3.77. CSS Grid exercice 3	73
3.78. CSS Grid exercice 1 et 2	73
3.79. CSS Grid exercice 5	74
3.80. @supports chrome	74
3.81. @supports mozilla	75

Chapitre 1. Introduction

1. Contexte

Le module M1105 a pour objectif, de savoir structurer et présenter des contenus numériques. Dans notre département, ce module est décomposé en deux parties, l'une "communicationnelle" M1105-1 et l'autre informatique M1105-2. Nous ne parlons que de cette dernière partie. Nous allons aborder :

l'HTML (HyperText Markup Language)

un langage de balise utilisé pour la structuration du contenu des pages web,

la CSS (Cascading Style Sheets)

un langage qui définit, la présentation des pages web.

Il est important de garder à l'esprit cette dichotomie entre le contenu (HTML) et sa représentation (CSS).

La CSS et l'HTML sont des familles de langages normalisés (standardisés) par le W3C (World Wide Web Consortium). Comme dans tout langage, il existe des contraintes à respecter dans l'écriture d'un document CSS ou HTML. Certaines contraintes sont de forme, comme par exemple, délimiter les chaînes de caractères par des " ou des '. D'autres contraintes sont le respect d'une grammaire, qui comme en français impose des règles de syntaxe. On dit qu'un document est valide (ou pas) par rapport à une grammaire. Le navigateur est un logiciel prévu pour être tolérant et capable de faire la présentation de documents à la syntaxe approximative. Le bon affichage d'un document ne sera pas la preuve de sa validité. La validité est cependant nécessaire pour garantir que le comportement reste le même quel que soit le navigateur. Nous vous demandons donc que l'ensemble de vos documents soient valides.

Les applications Web reposent côté client, principalement sur l'utilisation de l'HTML, de la CSS et du langage JavaScript. Nous n'aborderons le JavaScript qu'au troisième semestre. De même, dans ce module vous ne verrez que les bases d'HTML et de CSS, ce sera à vous par la suite de parfaire vos connaissances.

Ce cours ne se veut pas un support pour futur intégrateur web mais se veut comme un kit de découverte pour informaticien, l'accent sera mis sur la mise page et sur la communication avec le serveur.

La partie serveur ne sera pas non plus vue, elle ne vous sera enseignée qu'au troisième semestre.

Le thème **wordpress** suivant servira de support : <https://colorlib.com/shapely/>.

2. Mise en oeuvre

Le cours est décomposé en 5 CM (Cours Magistraux) d'une heure vingt et de 14 séances de TD (Travaux Dirigés) d'une heure vingt. Vous serez évalué par :

- des questionnaires en amphi,
- un examen sur machine.

Chaque séance est dépendante de la précédente et chaque séance est composée d'une partie obligatoire sur laquelle portera l'évaluation et d'une partie facultative.

Chapitre 2. HTML

Nous allons utiliser HTML5 la version la plus récente du langage, la spécification complète est disponible ici : <https://www.w3.org/TR/html52/> ce support de référence est complexe pour une première utilisation, celui-ci peut vous être plus utile et sera disponible le jour de l'examen : <https://developer.mozilla.org/fr/docs/Web>.

Chacune de vos pages devra être valide, vous pouvez utiliser le "validateur" W3C pour vous aider : <https://validator.w3.org/>.

1. TP 1 : Bases HTML

1.1. Première page

Vous travaillerez sur le fichier `index.html`. Pour visualiser un site web nous utiliserons un serveur web local quelques exemples de serveurs disponibles. À la racine de votre projet, lancer la ligne de commande suivante pour démarrer votre serveur web : **php -S localhost:8080**. Vous travaillerez avec des URLs de la forme `http://localhost:8080/`.

PHP

À la racine de votre projet, lancer la ligne de commande suivante pour démarrer votre serveur web : **php -S localhost:8080**. Vous travaillerez avec des URLs de la forme `http://localhost:8080/`.

codesandbox

<https://codesandbox.io/> : Une plate-forme en ligne qui vous permet des sites statiques.

Le serveur web de votre IDE

intégré dans IntelliJ, Live Server dans Visual Studio Code, ...

Le but est de reproduire la page suivante en utilisant les balises (éléments HTML) : `!doctype`, `html`, `head`, `meta`, `title`, `body`, `h1`, `h3`, `p`, `img`, `ul`, `ol`, `li`, `dl`, `dt`, `dd`.

Les balises (élément HTML) disposent d'attributs (i.e. propriétés) dont vous aurez à faire usage.

On souhaite également que le passage de la souris sur l'image du mac déclenche l'affichage d'une info bulle ("c'est beau un mac") et que le titre de l'onglet soit "TD1".

Les éléments multimédia se trouvent dans les sous répertoires de `assets`.

Figure 2.1. Ma première page

Shapely Demo

- HOME
- BLOG
- PORTFOLIO
- ABOUT THE TEST
 1. Page Image Alignment
 2. Page Markup And Formatting
 3. Clearing Floats
 4. Page with comments
 5. Page with comments disabled
- Level1
 - Level2
 - Level3
 - Level3a
 - Level3b
 - Level 2A
 - Level 2B
- SHOP
 - Page A
 - La page A
 - Page B
 - La page B



We Change Everything WordPress

This is the only WordPress theme you will ever want to use. [Download Now](#) [Read More](#)



SEO Friendly

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam pulvinar luctus sem, eget porta orci. Maecenas molestie dui id diam feugiat, eu tincidunt mauris aliquam. Duis commodo vitae ligula et interdum. Maecenas faucibus mattis imperdiet. In rhoncus ac ligula id ultricies.

[Read more](#)

1.2. Ajout de la navigation

Un site peut-être composé de plusieurs pages. Nous allons donc avec la balise `a` (une ancre) ajouter la navigation.

1.2.1. Étape 1

Un clic sur le texte BLOG de `index.html`, doit afficher le fichier `pages/blog/index.html`. La page ouverte doit contenir une image dont le clic affiche l'image en grand et un lien vers la page d'accueil (`index.html`).

Figure 2.2. Lien cliquable

[Shapely Demo](#)



1.2.2. Étape 2

Il est possible de faire des liens vers d'autres sites et même vers des endroits spécifiques d'autres sites. Dans le fichiers `./pages/about/index.html` créer des liens vers :

- https://codex.wordpress.org/Theme_Development
- https://codex.wordpress.org/Theme_Development#Code_Standards
- https://codex.wordpress.org/Theme_Development#Resources_and_References

Pouvez-vous expliciter à quoi correspond le # des URL ?

Figure 2.3. Liens externes

About

WordPress Theme Development Resources

1. See [Theme Development](#) for [code standards](#), examples of best practices, and [resources for Theme development](#).
2. See [Theme Unit Test](#) for a robust test suite for your Theme and get the latest version of the test data you see here.
3. See [Theme Review](#) for a guide to submitting your Theme to the [Themes Directory](#).

1.2.3. Étape 3

Vous l'avez découvert, il est possible de créer un lien vers un endroit précis de la page courante, ou d'une autre page afin de positionner correctement le navigateur. L'endroit précis est indiqué en utilisant l'attribut `id` et la référence se fait en utilisant `#`.

Dans `./pages/portfolio/index.html` faites en sorte qu'un clic sur le premier lien vous renvoie sur la première image.

1.3. Pour aller plus loin

Les exercices qui suivent ne font pas partie d'une évaluation mais contribuent à votre culture générale. Ils sont très peu guidés ; à vous de chercher.

1.3.1. Ajout d'une favicon

Ajouter la favicon présente dans `assets/icons` à la page d'accueil du site.

1.3.2. Figures

En utilisant `figure` et `figcaption` légender la première image du portfolio.

Figure 2.4. Légender une figure



Légende associée

1.3.3. Détails et résumés

En utilisant `détails` et `summary` réaliser dans `./pages/portfolio/index.html` le rendu suivant :

Figure 2.5. Détails et résumé



Un clic sur la flèche permet de masquer ou d'afficher les détails.

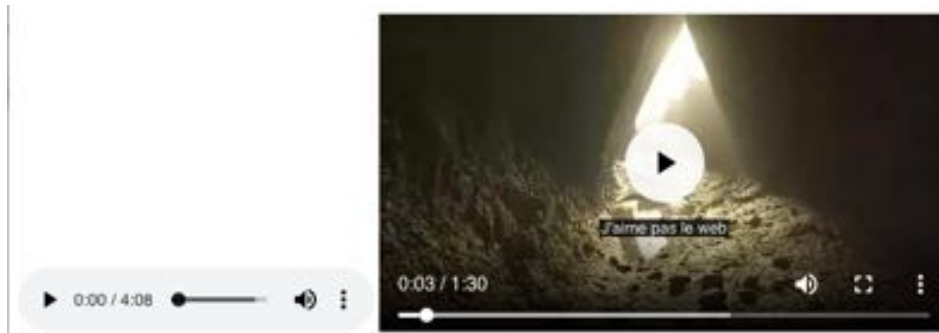
1.3.4. SVG

Le fichier `./demos/test.svg` contient une image au format SVG (Scalable Vector Graphics) afficher la dans votre navigateur et faites varier la taille de votre fenêtre. Faites de même avec `favicon.png`. Que constatez-vous ? Comment le SVG fonctionne-t-il ?

1.3.5. D'autres éléments

Tester dans `./assets/demos/multimedia.html` les balises suivantes : `video`, `source`, `audio`. Vous trouverez des éléments multimédia dans `./assets/videos`, `./assets/audios`. Il existe bien d'autres balise HTML5 comme `dialog`, `progress`, `meter` ou encore `article`, `section`, `nav`, `aside`, ... que nous n'aurons pas le temps de voir.

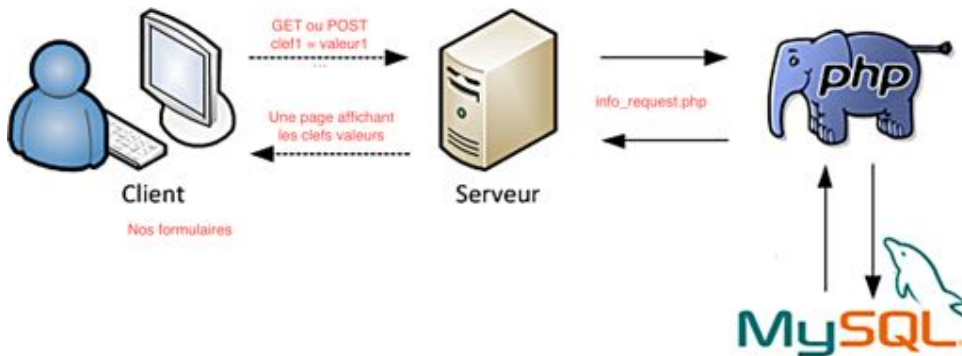
Figure 2.6. Une bande audio, une vidéo sous-titrée



2. TP 2 : Formulaires

Le but d'un formulaire est d'envoyer de l'information vers un serveur. Le protocole utilisé, pour dialoguer entre votre client (le navigateur) et votre serveur est le protocole HTTP. Ce protocole a comme principales méthodes d'envoi `POST` et `GET`. Dans les deux cas, l'information est transmise au serveur sous la forme de couples clef/valeur, la clef servant à distinguer les différentes informations transmises (par exemple : `note=18` où `note` est la clef et `18` la valeur associée).¹.

Figure 2.7. Architecture client serveur



Dans ce TP, tous vos scripts enverront de l'information au script PHP `info_request.php` (Si vous n'avez pas PHP d'installé vous pouvez utiliser celui-ci : http://berdjugin.com/temp/php/info_request.php). Celui-ci fabriquera une page HTML présentant l'information reçue. Dans un vrai site, le PHP pourrait solliciter une base de données pour générer cette page de retour.

2.1. Minimum vital

Les formulaires sont définis à minima par une balise `form`, un `input` de type `submit`, d'autres `input` dont le `name` sera la clef envoyée au serveur et dont la valeur sera la valeur saisie ou sélectionnée par l'utilisateur.

2.1.1. Premier formulaire avec des valeurs par défaut

Le fichier à modifier est `form1.html`.

¹Vous apprendrez plus tard, l'utilité des méthodes `PUT` et `DELETE`, mais pour le moment `POST` et `GET` nous suffiront.

Ici, vous aurez votre premier formulaire avec 2 input de type texte pré-remplis et avec un input de type submit.

Figure 2.8. Un premier formulaire avec des valeurs par défaut

Prénom :

Nom :

Le clic sur le "submit" conduira à l'affichage de la page suivante gérée par le script `info_request.php` :

Figure 2.9. Résultat de l'envoi du premier formulaire

Entête HTTP

```
Host: localhost
Connection: keep-alive
Content-Length: 19
Cache-Control: max-age=0
Origin: http://localhost:63342
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://localhost:63342/tp/code/tp2/tp2_correction/form1.html?_ijt=kqma4ptau4stnr2k0ccrhmbqa
Accept-Encoding: gzip, deflate, br
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: __ga=GA1.1.1437276879.1531144956; Idea-65419a4f=75794e47-7b74-4d82-9975-8185fbb243aa; ciNav=no
```

Méthode : POST

```
prenom: John
nom: doe
```

La partie entête sera différente mais la partie méthode devra être la même.

Faire de même avec la méthode GET et observer le retour.

2.1.2. Formulaire avec fieldset et label

Le fichier est `form2.html` (vous pouvez faire un copier coller du fichier précédent).

Dans ce formulaire, un clic sur le texte nom ou prénom doit donner le focus à l'input correspondant.

Figure 2.10. Label et fieldset

Un formulaire

Prénom :

Nom :

2.1.3. Formulaire avec espace réservé

Le fichier est `form3.html`. Vous utiliserez des placeholder.

Figure 2.11. Espace réservé

2.1.4. Formulaire avec mot de passe et réinitialisation

Le fichier est form4.html, il contient 2 input de type password, un input de type hidden, un input de type reset.

Figure 2.12. Password et réinitialisation

Bien évidemment, le mot de passe et sa confirmation doivent être envoyés au serveur, mais nous souhaitons également que la clef cache ait la valeur inconnue *Ce ne sont pas les mots de passe qui doivent-être cachés mais il faut envoyer au serveur une information qui est dans le code source mais qui n'est pas affichée.*

Figure 2.13. Réponse avec un champ caché

2.1.5. Formulaire complet

Le fichier est form5.html. Selon votre navigateur et votre système d'exploitation vous n'aurez probablement pas les mêmes affichage mais vous devrez être au plus près des contraintes suivantes :

Figure 2.14. Un formulaire complet

- Le genre ne peut-être que homme ou femme, un clic sur le label sélectionne le bouton radio, homme est sélectionné par défaut .
- Il est possible de sélectionner deux animaux de compagnie.
- La date de naissance doit être comprise entre le 01/01/2000 et le 31/12/2010
- Le groupe par défaut est le groupe 4
- Il est possible d'avoir plusieurs centres d'intérêts et, par défaut, trois sont affichés mais non sélectionnés.
- La zone de texte, contient 10 lignes et 50 colonnes.
- Le sport est un input de type list.
- 'Clic moi!' est un button dont l'attribut onclick est "alert('Un peu de JS')".

Pour dialoguer avec le serveur vous devez pour les centres d'intérêts utiliser un name suffixé par []. Pourquoi ?

2.2. Pour aller plus loin

Il existe de nombreux autres types de input. Tous ne seront pas supportés par votre navigateur. Le premier exercice qui suit en est un florilège. Le second exercice porte sur des vérifications de contraintes du côté client sans l'utilisation du JavaScript.

2.2.1. Éléments et input

Figure 2.15. Florilège

Un formulaire

Mois et année):

Date (jour et heure): ▼

Heure :

Semaine :

Choisir un fichier : Aucun fichier choisi

Nombre pair entre 0 et 100:

Entre 0 et 10 :

Search Google:

Couleur :

Le fichier à utiliser ici est `form6.html`, vous devez avoir :

- un input avec mois et année,
- un input avec jour et heure,
- un input avec heure,
- un input avec choix d'un fichier,
- un input qui contient un nombre compris entre 0 et 100 avec un pas de 2 et une valeur initiale de 2,
- un input qui contient une plage entre 0 et 10,
- un input de type search,
- un input de type color, ...

2.2.2. Champs requis et validation de patterns à la volée

Dans cet exercice `form7.html`, vous allez utiliser l'attribut `required` pour forcer une saisie et l'attribut `pattern` pour vérifier une expression régulière. L'expression régulière aura la forme : `[12][\.-]?[0-9]{2}[\.-]? (0[1-9]|[1][0-2])[\.-]?([0-9]{2}|2A|2B)[\.-]?[0-9]{3}[\.-]?[0-9]{3}[\.-]?[0-9]{2}` qui correspond à un numéro de sécurité sociale. Les deux input devront être remplis.

Figure 2.16. Champs requis et validation de patterns

Un formulaire

No de sécurité sociale Ville

 Veuillez respecter le format requis.

2.2.3. Boutons

Les input sont limités dans leurs affichages, les button sont plus souples. Par exemple, le code de `form8.html` permet d'avoir un bouton avec du SVG. Le clic sur l'étoile doit vous envoyer vers <https://www.google.com/bookmarks/?hl=fr> et le clic sur l'autre bouton vers votre script.

Figure 2.17. Les boutons

Un formulaire

First name:

Last name:



Vers mes favoris

Chapitre 3. CSS

L'HTML a pour responsabilité la description du contenu, la CSS¹ *Cascading Style Sheets* a pour responsabilité la présentation de l'HTML sur plusieurs types de médias. Dans ce cours, nous n'utiliserons que le média `screen`. À titre indicatif, le document que vous utilisez, repose sur le média "paper". Ce cours n'abordera pas non plus la partie *responsive design*, qui permet d'offrir différentes présentations suivant la résolution des écrans (fixes, tablettes, portables).

1. TP 3 : Bases CSS

Ici vous allez apprendre comment charger une ou des CSS, utiliser des sélecteurs et des règles.

1.1. Charger une CSS

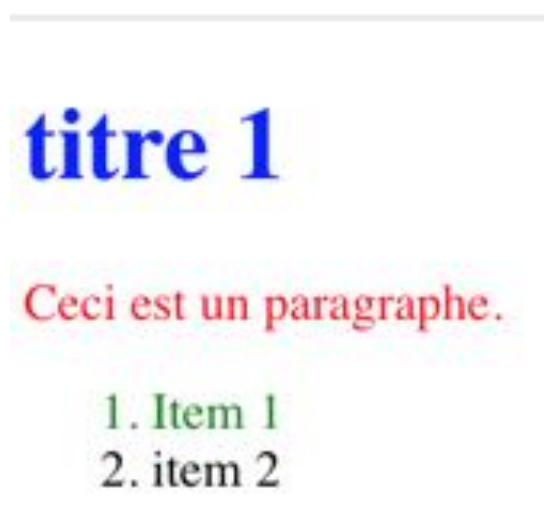
Il existe principalement trois méthodes pour charger une CSS :

- charger une feuille de style externe dans l'entête (au niveau de l'élément `head`) en utilisant l'élément HTML `link`. Exemple : `<link rel="stylesheet" type="text/css" href="../assets/css/page01.css" media="screen">`
- charger une feuille de style interne dans la page en utilisant l'élément HTML `style`. Exemple : `<style type="text/css"> h1 { color: blue; } </style>` (la couleur du texte des `h1` est bleu)
- dans un élément en utilisant l'attribut HTML `style`. Exemple : `style="text-transform: capitalize; color: green"` (la couleur du texte de l'élément portant l'attribut est verte et sa première lettre est en majuscule).

Selon vous quelle sera la méthode à favoriser ?

En utilisant les fichiers `page01.html` et `page01.css` (dans le répertoire `assets`), reproduire la page suivante :

Figure 3.1. Charger une CSS (page01.html)



Les CSS peuvent se trouver sur des sites distants. En utilisant la `page02.html`, charger les feuilles suivantes : <https://maxcdn.bootstrapcdn.com/4.4.3/css/bootstrap.min.css> et <https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.0/animate.min.css>.

Si vous avez réussi, vous devez observer une image qui se transforme et les boutons d'un lecteur audio.

¹La CSS en est à sa version 3 et une version 4 devrait bientôt être normalisée.

Enfin, il est possible, dans un fichier CSS de charger d'autres fichiers CSS. En utilisant `page03.html` et le fichier `page03.css`, réaliser la même présentation que précédemment. Vous aurez besoin dans votre CSS de `@import "url de la CSS à importer" screen;`.

1.2. Sélecteurs, règles

Nous avons vu comment charger une CSS ou des CSS. Une fois celles-ci chargées, nous allons pouvoir cibler (désigner, sélectionner) les éléments que nous souhaitons modifier (styler). Les éléments sont ciblés en utilisant des sélecteurs et sont présentés conformément à des règles de la forme : `sélecteur {règle1, règle2;}`.

Voici un rappel rapide (non exhaustif) sur les sélecteurs à connaître² :

Sélecteurs simples :

de type

Il est possible de sélectionner des éléments par leur nom :

```
/* l'ensemble des input */
input {}
```

de classe

Il est possible de sélectionner un ensemble d'éléments qui appartiennent à une même classe :

```
<style>
  .menu {}
</style>
<div class="menu important"> <!-- selectionnee -->
  <div class="menu"> <!-- selectionnee -->
  </div>
</div>
<div class="menu null"> <!-- selectionnee -->
</div>
<div class="null">
</div>
```

d'identifiant

Un identifiant est unique dans une page et permet de cibler un élément :

```
#unId {} /* l'élément d'identifiant unId */
```

d'attribut

Il est possible de sélectionner en utilisant un attribut :

```
input[type="submit"]{} /* Les input de type submit */
...
```

universel

Il est possible de sélectionner tous les éléments d'une page :

```
* { }
```

Combinateurs

```
/* les descendants */
```

²Je vous invite à consulter votre cours.

Pseudo classe

```
a:visited {} /* Les ancres déjà visitées */
span:hover {} /* Les span qui reçoivent le focus */
a:active {} /* Une ancre lors de sa sélection */
span:first-child {} /* Les spans qui sont les premiers fils */
... La liste est longue
```

Les pseudo éléments sont des entités HTML non décrites dans le document, c'est le seul moyen pour créer de l'HTML à partir de la CSS :

Avant une mise en pratique sur votre page je vous propose le jeu en ligne suivant : <https://flukeout.github.io/>.

Figure 3.2. CSS sélecteurs (page04.html)

Quae *provinciae bellae* quondam gratior extiterit minime praedictum a Servilio pro consueta missae sub iugum factae sunt vestigiales, et hae quidem regiones vixit in promissum terrarum laquei postuae ob eorum non moxte Amaro disparantur.

Ipsam vero urbem Byzantium fuisse refoetissimam aequae ornatiissimam signis quae ignorat? Quae illi, exhausti sumptibus bellisqae maximis, cum omnis Mithridaticus impetus totamqae Pontum amantem afforescentem in Asiam atqae crumpentem, ore repulsum et cervicibus interclusum suis sustinent, tum, inquam, Byzantii et postea signa illa et reliqae urbis remanens sanctionis custodia tementur.

Tempore quo primis auspiciis in mundum fulgorem surgetur victura dum erunt homines Roma, ut augetur sublimibus incrementis, fodere pacis aeternae Virtus convenit atqae Fortuna plerumqae dissidentes, quarum si altera deficiasset, ad perfectum non venerat summatem.

Den haec in *oriente praeter*, Arelate hiemans quies Constantius post theatralis ludos atqae circensium antiochie editos apparuit diem sextum idus Octobres, qui imperii eius annum incensum terminavit, insensitiora pondera gravata libens, siquid dabam deferrebat aut falsum, pro lapido accipiens et conepo, inter alia exarumficatum Genotium Magentianae comitis patris exulati macrone multavit.

Iste **duae provinciae bello** quendam praetium in se mistae praedonum a Servilio pro comitate missae sub iugum factae sunt vestigales, et hae quidem regiones vixit in prominenti terrarum linguas postitae ubi orbe eo mox ante dispartimur.

Ipsam vero urbem Byzantium fuisse refertissimam atque certatissimam signis quae ignorat? Quae illi, exhausti sumptibus bellisque maximis, cum omnis Mithridaticus impetus totaque Pontum armenta afferrescentem in Asiam atque erumpentem, ore repulsum et cervicibus interclusum suis sustinerent, tum, inquam, Byzantii et postea signa illa et reliqua urbis ornamenta sanctissime custodia tenebant.

Tempore quo primis auspiciis in mundum fulgorem surget victura dum erant homines Roma, in augeret sublimitibus incrementis, foedere pacis aeternae Virtus convenit atque Fortuna plerumque dissidentes, quarum si altera defaisset, ad perfectam non venerat summatem.

in oriente a prae

Figure 3.3. CSS sélecteurs (page05.html)



1.3. Polices

Le petit exercice page06.html et page06.css consiste à reproduire l'image suivante sans toucher page06.html.

Figure 3.4. Police et background (page06.html)



Quelques indications :

1. la fonte robot de google peut-être trouvée ici : <https://fonts.googleapis.com:443/css?family=Roboto>.
2. l'ensemble des marges internes et externes sont nulles (margin, padding),
3. la fonte de l'élément body est roboto ou sans-serif,
4. l'élément body possède une image de fond,
5. la fonte des h2 est "Comic Sans MS"
6. la couleur des h2 est en RGB 255,140,0
7. la taille de la fonte des h1 est de 35 pixels,
8. la couleur est FF7F50,
9. la couleur des ancres sélectionnées est aliceblue; à la sélection, elles sont également encadrées par des tirets rose mais toujours pas soulignées,
10. la documentation des icônes bootstrap est ici <https://getbootstrap.com/docs/3.3/components/>.

1.4. Pour aller plus loin

Dans cet exercice page07.html, vous avez des listes imbriquées (3 niveaux) :

1. Initialement les ancres ne sont pas soulignées,
2. le passage sur un li de niveau 1 doit mettre la couleur de son ancre en rouge et souligner les ancres des listes de niveau 2,
3. le passage sur un li de niveau 2 doit mettre la couleur de son ancre en bleu et souligner les ancrs des listes de niveau 2 et 3,
4. le passage sur un li de niveau 3 doit mettre la couleur de son ancre en rose et souligner les ancrs de la liste de niveau 2 qui la contienne.

Figure 3.5. CSS menu imbriqués (page07.html)

- HOME
- BLOG
- PORTFOLIO
- ABOUT THE TEST
 1. Page Image Alignment
 2. Page Markup And Formatting
 3. Clearing Floats
 4. Page with comments
 5. Page with comments disabled
- Level1
 - Level2
 - Level3
 - Level3a
 - Level3b
 - Level 2A
 - Level 2B
- SHOP
 1. Page A
 2. Page B

2. TP4 : Modèle de boîte et position

Si vous ne l'avez pas encore fait, il est important de s'appropriier les outils de "débogage" de votre navigateur. Vous pouvez également, vous aider du site suivant qui sera disponible le jour de l'examen : <https://developer.mozilla.org/fr/docs/Web/CSS/Reference>.

2.1. Modèle de boîte

2.1.1. Rappels de cours

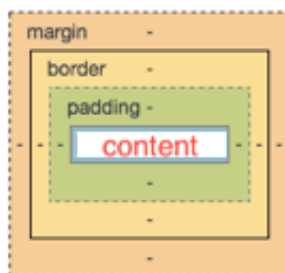
Cette première partie n'est qu'un simple rappel qui ne peut se substituer au cours.

2.1.1.1. Propriétés

Chaque boîte possède les zones :

- *content* : cette zone définit le contenu. Elle dispose des propriétés :
 - *width* : largeur
 - *height* : hauteur
 - *max-width*, *min-width* : largeur maximum et minimum,
 - *max-height*, *min-height* : hauteur maximum et minimum.
- *padding* : la boîte de remplissage, elle correspond à la zone entre le contenu et la bordure. Elle est définie par les propriétés suivantes :
 - *padding-top*, *padding-right*, *padding-bottom*, *padding-left*,
 - *padding*.
- *border* : la bordure possède, par défaut, une taille de 0. Elle est définie par les propriétés suivantes :
 - *border*,
 - *border-size*, *border-style*, *border-color*
- *margin* : la marge représente la distance entre deux boîtes. Les propriétés sont :
 - *margin*,
 - *margin-top*, *margin-right*, *margin-bottom*, *margin-left*.

Figure 3.6. Modèle de boîte



```
/* pour tous les paragraphes */
p {
  /* largeur 200px */
  width: 200px;
  /* hauteur auto */
  height: auto;
  /* marge interne de 20px */
  padding: 20px;

  /* bordure solide de 2px noire */
  border: 2px solid black;
}
```

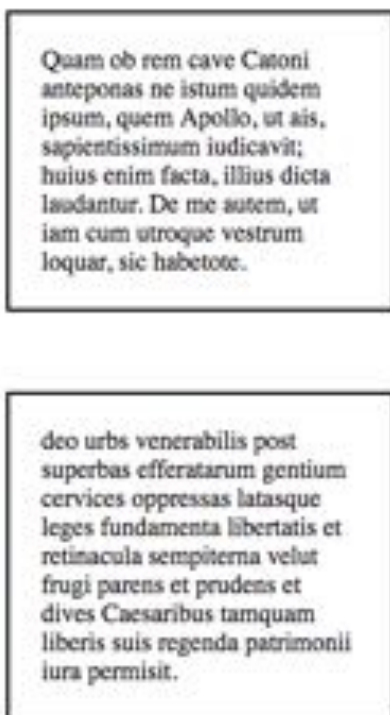
```

/* marge supérieure de 50px */;
margin-top: 50px;
}

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css"
      href="ressources/css/exemple01.css" media="screen" />
    <title> modèle de boîte</title>
  </head>
  <body>
    <p> Quam ob rem cave Catoni anteponas ne istum quidem ipsum,
    quem Apollo, ut ais, sapientissimum iudicavit; huius enim facta,
    illius dicta laudantur. De me autem, ut iam cum utroque vestrum
    loquar, sic habetote.</p>
    <p> deo urbs venerabilis post superbas efferatarum gentium
    cervices oppressas latasque leges fundamenta libertatis et
    retinacula sempiterna velut frugi parens et prudens et dives
    Caesaribus tamquam liberis suis regenda patrimonii iura
    permisit.</p>
  </body>
</html>

```

Figure 3.7. exemple 01



La taille horizontale d'une boîte est donc $\text{padding-left} + \text{border-left} + \text{width} + \text{border-right} + \text{padding-right}$. Lorsque l'on travaille en pourcentages, il peut-être intéressant d'inclure la bordure et le remplissage dans la taille, cela peut-être réalisé en utilisant la propriété `box-sizing`.

2.1.1.2. Interactions

2.1.1.2.1. Dépassements

Lorsque la taille est fixée, le contenu peut dépasser, la propriété *overflow* permet de gérer le dépassement, les valeurs possibles sont :

auto

un ascenseur apparaît

hidden

le dépassement n'est pas affiché

visible

le contenu dépasse de la boîte (comportement par défaut).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css"
      href="ressources/css/exemple02.css" media="screen" />
    <title>dépassement</title>
  </head>
  <body>
    <p> Quam ob rem cave Catoni anteponas ne istum quidem ipsum,
    quem Apollo, ut ais, sapientissimum iudicavit;
    huius enim facta, illius dicta laudantur. De me autem,
    ut iam cum utroque vestrum loquar, sic habetote.</p>
    <p> Quam ob rem cave Catoni anteponas ne istum quidem ipsum,
    quem Apollo, ut ais, sapientissimum iudicavit;
    huius enim facta, illius dicta laudantur. De me autem,
    ut iam cum utroque vestrum loquar, sic habetote.</p>
    <p> Quam ob rem cave Catoni anteponas ne istum quidem ipsum,
    quem Apollo, ut ais, sapientissimum iudicavit;
    huius enim facta, illius dicta laudantur.
    De me autem, ut iam cum utroque vestrum loquar, sic habetote.</p>
  </body>
</html>
```

```
/* pour tous les paragraphes */
p {
  /* largeur 200px */
  width: 200px;
  /* hauteur 100px */
  height: 100px;
  /* marge interne de 20px */
  padding: 20px;
  /* bordure solide de 2px noire */
  border: 2px solid black;
  /* marge supérieure de 50px */
  margin-top: 50px;
}

/* le premier des paragraphes */
p:nth-child(1){
  overflow: hidden;
}

/* le deuxième des paragraphes */
p:nth-child(2){
  overflow: auto;
}

/* le troisième des paragraphes */
p:nth-child(3){
  overflow: visible;
}
```

Quam ob rem cave Catoni
anteponas ne istum quidem
ipsum, quem Apollo, ut ais,
sapientissimum iudicavit;
huius enim facta, illius dicta
laudantur. De me autem, ut
~~iam cum utroque vestrum~~

anteponas ne istum quidem
ipsum, quem Apollo, ut ais,
sapientissimum iudicavit;
huius enim facta, illius dicta
laudantur. De me autem, ut
iam cum utroque vestrum
loquar, sic habetote.

Quam ob rem cave Catoni
anteponas ne istum quidem
ipsum, quem Apollo, ut ais,
sapientissimum iudicavit;
huius enim facta, illius dicta
laudantur. De me autem, ut
~~iam cum utroque vestrum~~
loquar, sic habetote.

2.1.1.2.2. Arrière plan

L'arrière plan est dessiné sous la bordure, le remplissage et le contenu, la propriété *background-clip* permet de définir la zone d'attachement :

- *border-box*
- *padding-box*
- *content-box*

```
/* pour tous les paragraphes */
p {
  /* largeur 200px */
  width: 200px;
  /* hauteur auto */
  height: auto;
  /* marge interne de 20px */
  padding: 20px;
  /* bord noir solid de 20px avec une transparence */
  border : 20px solid rgba(0, 0, 0, 0.5);

  /* marge supérieure de 50px */
  margin-top: 50px;

  background-image: url(../images/furet.jpg);
  background-repeat: no-repeat;
  background-color: blue;
}

/* le premier des paragraphes */
p:nth-child(1){
  background-clip : border-box;
}

/* le deuxième des paragraphes */
p:nth-child(2){
  background-clip : padding-box;
}

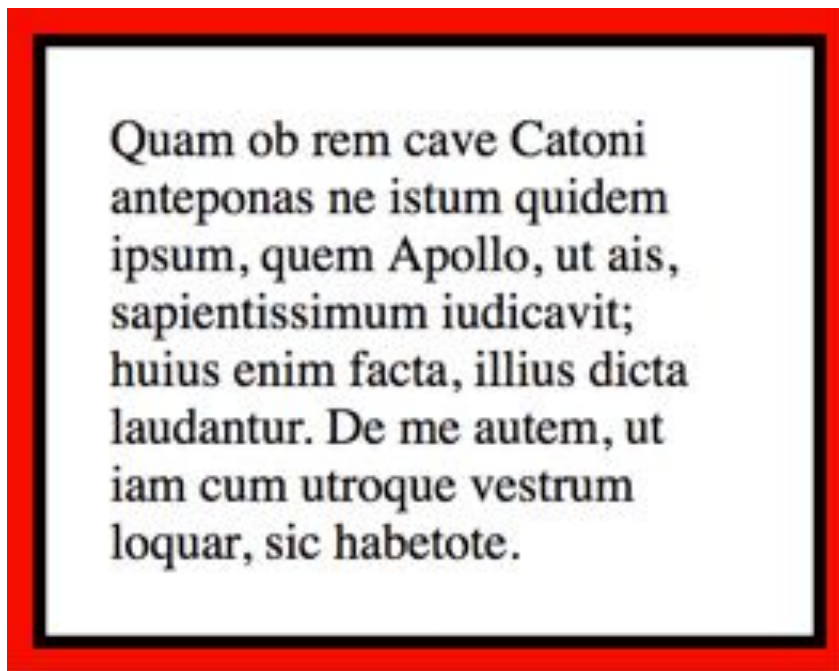
/* le troisième des paragraphes */
p:nth-child(3){
  background-clip : content-box;
}
```

Figure 3.9. arrière plan

2.1.1.2.3. Contour

Le contour est dessiné autour de la boîte dans la zone dédiée à la marge (après la bordure), la propriété est *outline*.

```
/* pour tous les paragraphes */
p {
  /* largeur 200px */
  width: 200px;
  /* hauteur auto */
  height: auto;
  /*marge interne de 20px */
  padding: 20px;
  /* bord nord solide de 4px */
  border: 4px solid black;
  /* outline rouge solide de 8px */
  outline: solid red 8px;
}
```

Figure 3.10. *outline***2.1.1.3. Types de boîtes**

Il existe trois type de boîtes définis par la propriété *display* :

inline

elles flottent dans le texte, les propriétés du modèle de boîte s'appliquent mais n'affectent pas les boîtes environnantes.

block

ce sont des blocs qui s'empilent, le modèle de boîte est applicable.

inline-block

elles ont le même comportement que les *inline*, elles flottent dans le texte mais le modèle de boîte affectent les boîtes environnantes.

Une chose à savoir pour les *inline-block* et les *inline* est la notion de *white-space*. Tout caractère blanc séparant deux éléments créera un espace, de taille variable suivant les navigateur. Ainsi

```
<span></span><span></span> <!-- pas d'espace entre -->
```

est différent de

```
<span>
</span>
<span>
</span>
<!-- un espace entre dépendant du navigateur -->
```

L'exemple suivant illustre les différents types de blocs.

```
/* pour tous les paragraphes */
p {
  width: 100px;
}

/* le premier des paragraphes */
p:nth-child(1){
  color : red;
```

```
    display: inline-block;
}

/* le deuxième des paragraph */
p:nth-child(2){
    color: blue;
    display: inline-block;
}

/* le troisieme des paragraph */
p:nth-child(3){
    color: aqua;
    display: inline;
}

/* le quatrième des paragraph */
p:nth-child(4){
    color: fuchsia;
    display: inline;
}

/* le cinquieme des paragraph */
p:nth-child(5){
    color: green;
    display: block;
}

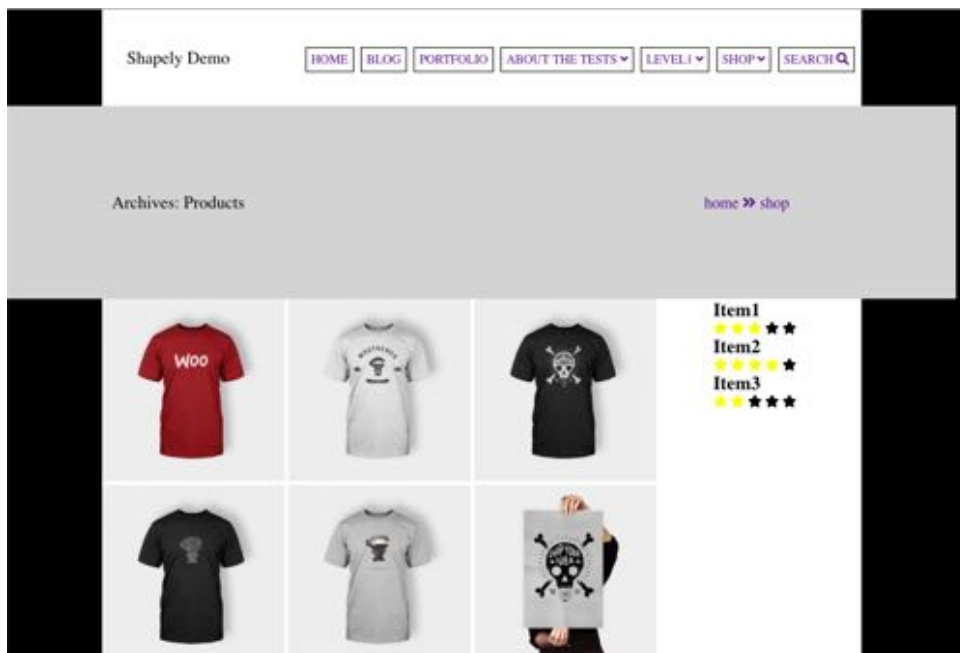
/* le sixième des paragraph */
p:nth-child(6){
    color: orange;
    display: block;
}
```

Figure 3.11. display

2.1.2. Exercices

Dans les exercices qui suivent le fichier CSS n'est pas lié, c'est à vous de le faire en utilisant la balise `link`.

Le but est de reproduire en plusieurs étapes le design suivant :

Figure 3.12. Design en inlineblock**2.1.2.1. Exercice 1**

Nous allons réaliser, en utilisant `exercice01.html` et `exercice01.css`, le header d'une page. Les sélecteurs sont donnés.

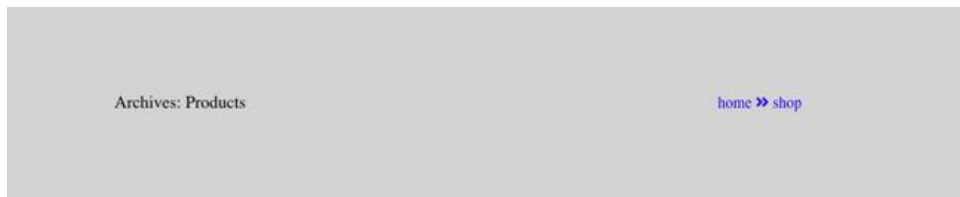
Figure 3.13. Un menu horizontal

Voici quelques contraintes et indications :

- les marges internes et externes sont par défaut nulles,
- la couleur de fond du body est black,
- Le header est centré et utilise 80% de la largeur du body, le `span#title` occupe 20% de son conteneur, et la `nav nav.menu_horizontal` 79%,
- Il faut que le lien s'active dès que le `li` reçoit le focus. Le lien doit donc avoir la hauteur du `li`,
- Les propriétés : `height`, `line-height`, `vertical-align`, `text-align` peuvent vous aider,
- Les tailles des polices sont : `small` et `large`,
- Le header à une hauteur de 100px, les `li` 20px, les marges externes gauches et droites sont calculée automatiquement avec un header qui occupe 80% de la page.

2.1.2.2. Exercice 2

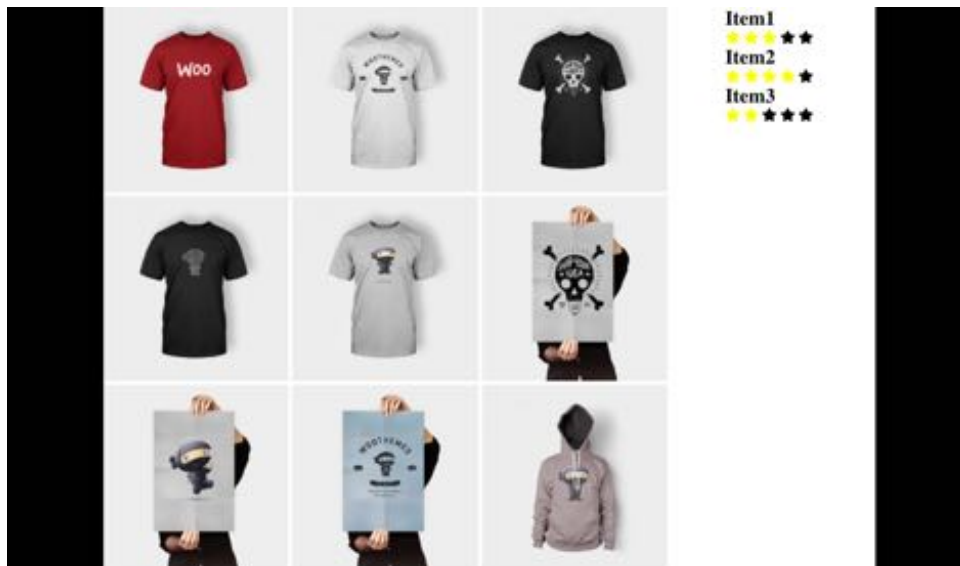
Nous allons maintenant réaliser, en utilisant `exercice02.html` et `exercice02.css`, la partie centrale de notre page.

Figure 3.14. Bandeau central

Peu de nouveauté par rapport à l'exercice précédent. Mais, cette fois, les sélecteurs ne sont plus fournis. Il faut également que l'ancre sur "shop" ne soit pas clicable.

2.1.2.3. Exercice 3

Nous allons maintenant afficher les articles. Les fichiers à utiliser sont `exercice03.html` et `exercice03.css`. Le fichier HTML n'est pas à modifier.

Figure 3.15. Le magasin

2.1.2.4. Exercice 4

Vous allez maintenant intégrer vos trois feuilles de style dans le fichier `exercice4.html`.

2.2. Position

La propriété `position` sert à définir la position des éléments dans une page HTML, nous allons la mettre en oeuvre.

2.2.1. Rappels de cours

2.2.1.1. Positionnement relatif

Le positionnement relatif (`position: relatif`) permet à un élément, tout en restant dans le flux, d'être déplacé visuellement en utilisant les propriétés *top*, *right*, *bottom*, *left*.

```
<p> Un déplacement <span> relatif </span> du span </p>
```

```
p {
  width: 250px;
  border: 1px solid black;
}
span {
```

```

/* position relative */
position: relative;
/* décalage de 10px vers le haut */
top: -10px;
}

```

Figure 3.16. Positionnement relatif



2.2.1.2. Positionnement absolu

Le positionnement absolu « retire » totalement le contenu concerné du flux. Sa position est déterminée par référence aux limites du premier conteneur positionné (en absolu ou relatif). Si il n'y a pas de conteneur, le body est utilisé. Le positionnement est réalisé en utilisant `top`, `right`, `bottom` et `left`.

Dans ce premier exemple la position de la div est calculée par rapport au body car il n'y a pas de conteneur positionné.

```

<div class="container">
  <div> blabla </div>
  <div class='absolute'>absolue</div>
</div>

```

```

div.container {
  width: 80%;
  margin-left: auto;
  margin-right: auto;
  border: 1px solid black;
}
div.absolute {
  position: absolute;
  top:0;
  left: 0;
  color: grey;
}

```

Figure 3.17. Position absolue par rapport au body



Dans l'exemple qui suit le conteneur est positionné (en relatif).

```

div.container {
  width: 80%;
  margin-left: auto;
  margin-right: auto;
  border: 1px solid black;

  position: relative;
}
div.absolute {
  position: absolute;
  top:0;
  right: 0;
  color: grey;
}

```

Figure 3.18. Position absolue par rapport à un container



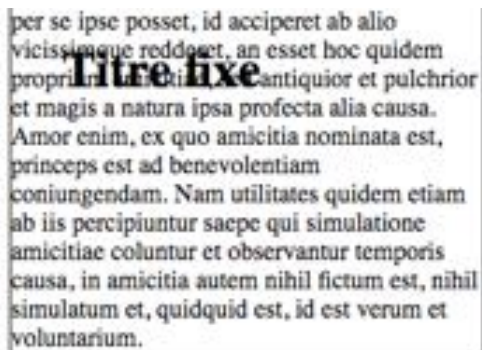
2.2.1.3. Positionnement fixé

Le positionnement *fixed* fixe l'élément par rapport à l'écran, le positionnement est donc indépendant du scrolling.

```
<div class="header">
  <h1> Titre fixe </h1>
</div>
<div class="container">
  Saepissime igitur ...
</div>
```

```
div.container {
  width: 80%;
  margin-top: 100px;
  margin-left: auto;
  margin-right: auto;
  border: 1px solid black;
}
div.header {
  position: fixed;
  top: 0;
  left: 20%;
  color: black;
}
```

Figure 3.19. position fixe



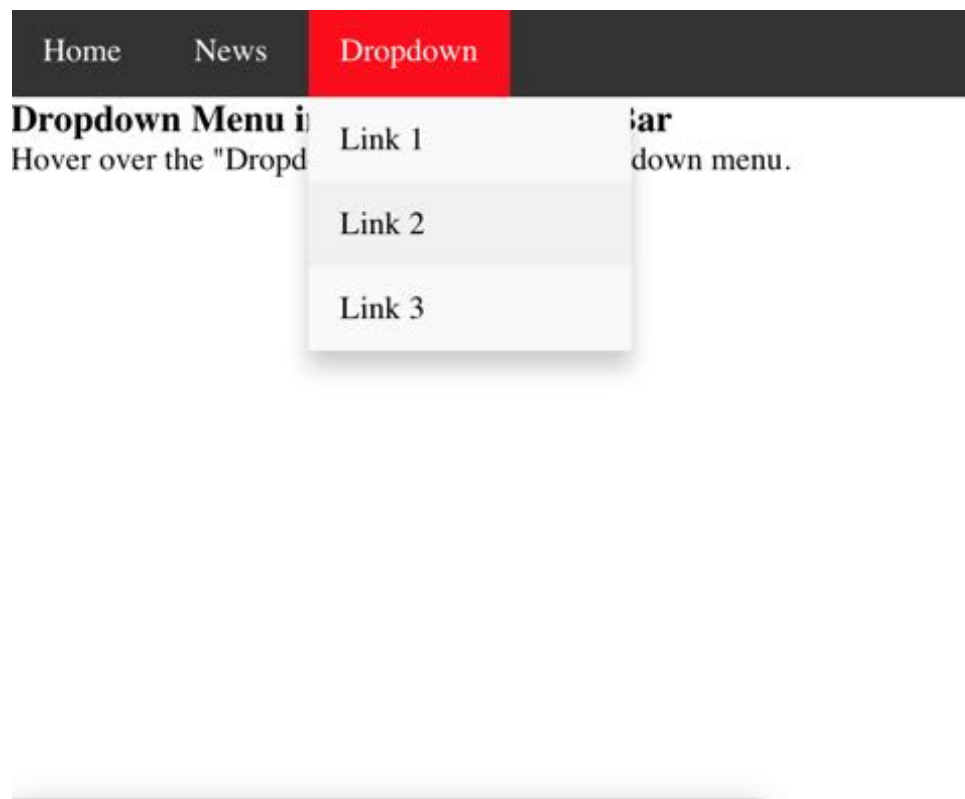
2.2.2. Exercices

Dans les exercices qui suivent, vous allez utiliser les positions : *absolute*, *fixed* et *relative*

2.2.2.1. Exercice 5

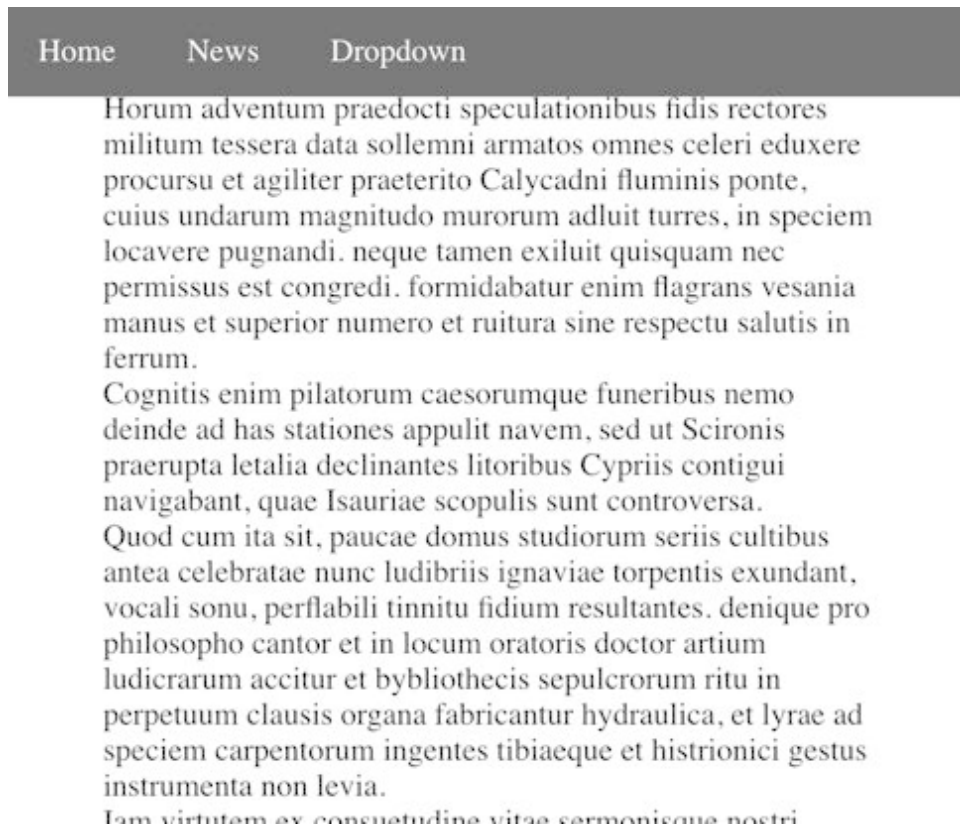
Dans cet exercice vous utiliserez les fichiers `exercice5.html` et `exercice5.css`. La partie HTML n'est pas à modifier ; seule la partie `exercice5.css` est à modifier pour obtenir un menu de type "drop down". Cet exercice contient une nouveauté que nous n'étudierons pas : les *z-index*. Ils correspondent en CSS à la notion de claques et permettent des superpositions. Ici vous devez utiliser les propriétés `display` et `position`.

Figure 3.20. Drop Down menu

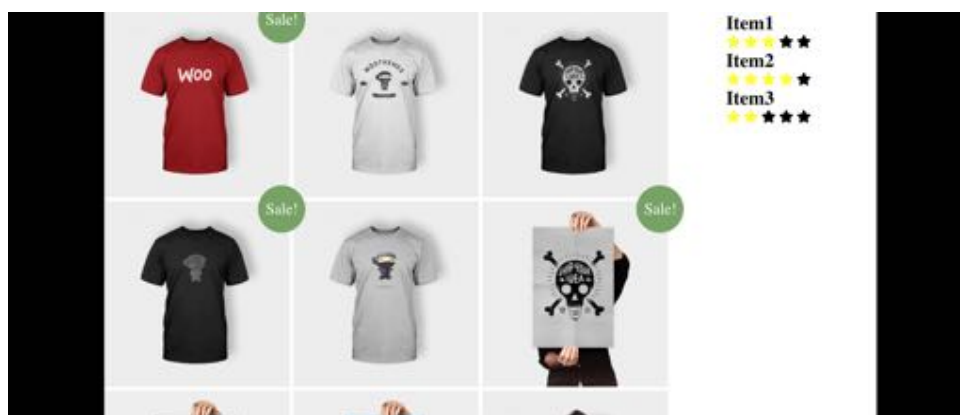


2.2.2.2. Exercice 6

Dans cet exercice, vous allez modifier le fichier `exercice6.css` pour que le menu soit fixé lors du défilement de la page.

Figure 3.21. Menu fixé**2.2.2.3. Exercice 7**

Dans cet exercice des `` ont été ajoutés et doivent-être affichés comme suit :

Figure 3.22. Position absolue et relative

À vous de modifier le fichier `exercice7.html` pour obtenir le même résultat. Pour information, vous pouvez utiliser : une couleur de fond (`#77a464`), un `z-index`, un `border-radius` et savoir que les valeurs peuvent-être négatives.

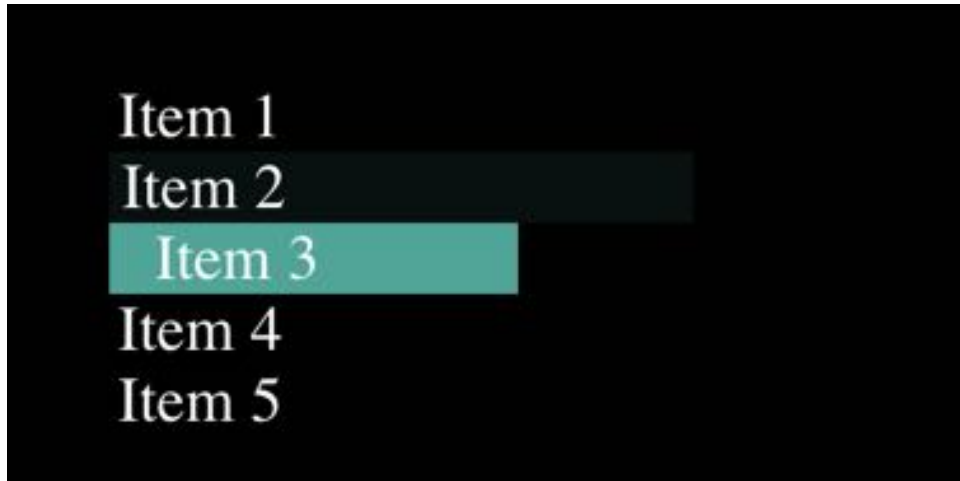
2.3. Pour aller plus loin

Cette partie comme son nom l'indique ne sera pas évaluée. Nous allons nous intéresser aux animations.

2.3.1. Exercice 8

Vous allez ici réaliser des animations sur un menu vertical avec des animations. Le fichier à modifier est `exercice8.css`. Lors de la réception du focus sur une ancre, le texte est décalé de 20 pixels vers la droite, la couleur de fond est #54B6A8 et la taille est de 130 pixels.

Figure 3.23. Menu vertical avec animations



2.3.2. Exercice 9

Vous allez ici réaliser des animations sur les images. Le fichier à modifier est `exercice9.css`. Au chargement de la page, les images sont en noir et blanc, au passage de la souris sur les 1: les images retrouvent leur couleur et un zoom de 1.2 est appliqué avec un dépassement caché.

Figure 3.24. Zoom et niveaux de gris



Les propriétés `filter` et `scale` peuvent vous aider.

2.4. Limitations

Les *white-space*, si ils sont problématiques, peuvent être supprimés avec des marges négatives mais ce n'est pas élégant et peut poser des problèmes suivant les navigateurs.

Également, comment envoyer simplement un élément à droite, comme par exemple notre menu horizontal ?

Ou encore, comment avoir une image entourée par du texte ?

Le prochain TP répond à ces problèmes, avec l'utilisation des flottants puis des flexbox.

3. TP 5 : Flottants et Flexbox

3.1. Flottants

3.1.1. Rappels de cours

Les flottants sont plus exactement le positionnement de boîtes flottantes, ces boîtes restent positionnée dans le flux global de texte mais le texte coule autour.

```
<div>
  <p>Premier paragraphe flottant</p>
  <p>Nec piget dicere avide magis hanc insulam ...
</p>
</div>
```

```
div{
  width : 200px;
  border: 1px solid black;
}
p {
  text-align: justify;
}
/* Le premier paragraphe */
p:first-child{
  /* Le premier paragraphe flotte à gauche */
  float: left;
  width: 100px;
  background-color: grey;
  margin: 5px;
}
```

Figure 3.25. Flottant à gauche



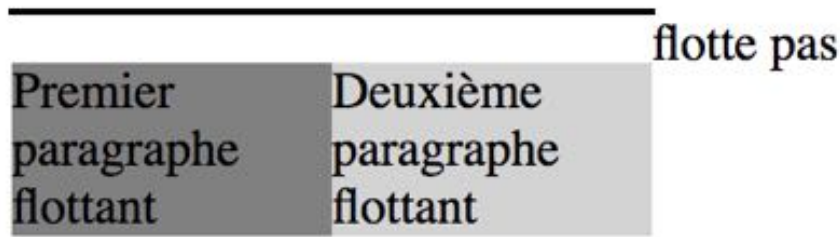
```
/* Le premier paragraphe flotte à droite */  
float: right;
```


Figure 3.26. Flottant à droite



```
<div>
  <p>Premier paragraphe flottant</p>
  <p> Deuxième paragraphe flottant </p>
</div>
<span> flotte pas </span>
```

```
div{
  width : 200px;
  border: 1px solid black;
}
p {
  text-align: justify;
  width: 100px;
}
/* Le premier paragraphe */
p:first-child{
  float: left;
  background-color: grey;
}
/* Deuxième paragraphe */
p:last-child{
  float:left;
  background-color: lightgrey;
}
```

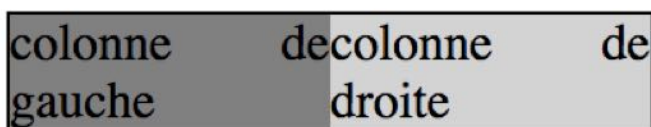
Figure 3.27. Deux flottants

La *div* de l'exemple précédent ne contient que des flottants et sa taille étant définie par les non flottant, elle a une hauteur de zéro. Le prix à payer avec les flottants est la gestion de la taille du conteneur.

La propriété *clear* s'applique au flottant et aux non flottants, elle permet d'interdire la présence des flottants. Lorsqu'elle est appliquée aux blocs non-flottants, elle déplace le bord de la bordure de l'élément sous le bord de la marge de tous les éléments flottants concernés. Ce mouvement (lorsqu'il se produit) empêche la fusion des marges (*margin collapsing*). Lorsqu'elle est appliquée aux éléments flottants, elle déplace le bord de la marge de l'élément sous le bord de la marge de tous les éléments flottants concernés. Cela impacte la position des éléments flottants suivants car ceux-ci ne peuvent pas être situés plus haut que les éléments flottants qui les précèdent. Les valeurs possibles sont : *left*, *right*, *both*, *inline-start*, *inline-end*.

```
<div class="container">
  <div class="column">colonne de gauche</div>
  <div class="column"> colonne de droite </div>
  <div class="clear"> </div>
</div>
```

```
div.container{
  width : 200px;
  border: 1px solid black;
}
/* toutes les div de class column */
div.column {
  text-align: justify;
  width: 50%;
  float: left;
}
/*Premierr div */
div.container div:first-child{
  background-color: grey;
}
/* Deuxième div */
div.container div:nth-child(2){
  background-color: lightgrey;
}
/* on interdit les flottants à droite et à gauche */
div.clear {
  clear:both; /* clear left fonctionne aussi */
}
```

Figure 3.28. Clear

La solution précédente, nous a obligé à ajouter de l'HTML (<div class="clear"> </div>), une autre solution existe les pseudo éléments *::before* et *::after*, ils permettent depuis la CSS de créer du texte dans un élément. Les pseudo éléments disposent de la propriété *content* qui spécifie le texte.

```
<div>
  un texte
</div>

div::before {
  content: "avant";
}
div::after {
  content: "après";
}
```

Figure 3.29. before, after

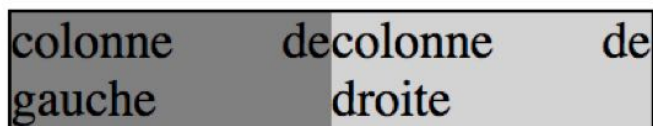
avant un texte après

Nous pouvons reprendre l'exemple avec 2 colonnes flottantes.

```
<div class="container">
  <div class="column">colonne de gauche</div>
  <div class="column"> colonne de droite </div>
</div>
```

```
div.container {
  width: 200px;
  border: 1px solid black;
}
/* toutes les div de class column */
div.column {
  text-align: justify;
  width: 50%;
  float: left;
}
/*Premierr div */
div.container div:first-child {
  background-color: grey;
}
/* Deuxième div */
div.container div:last-child {
  background-color: lightgrey;
}
/* on interdit les flottants à droite et à gauche
en ajoutant un élément */
div.container::after {
  content: "";
  display: block;
  clear: both;
}
```

Figure 3.30. after et clear



3.1.2. Exercices

Les flottants sont peu à peu remplacés par les flexbox et les systèmes de grilles. Ils conservent cependant leur usage de faire flotter un contenu autour d'un élément.

3.1.2.1. Exercice 1

Vous ne devez modifier que `exercice1.css` pour obtenir le rendu suivant :

Figure 3.31. Des images qui flottent



3.1.2.2. Exercice 2

Reproduire en modifiant `exercice2.css` le design suivant :

Figure 3.32. Une image est des colonnes qui flottent



3.2. Flexbox

Le positionnement en flottants impose de gérer les tailles. Une solution plus simple serait de décrire au niveau du conteneur la disposition, les flexbox répondent à cette attente.

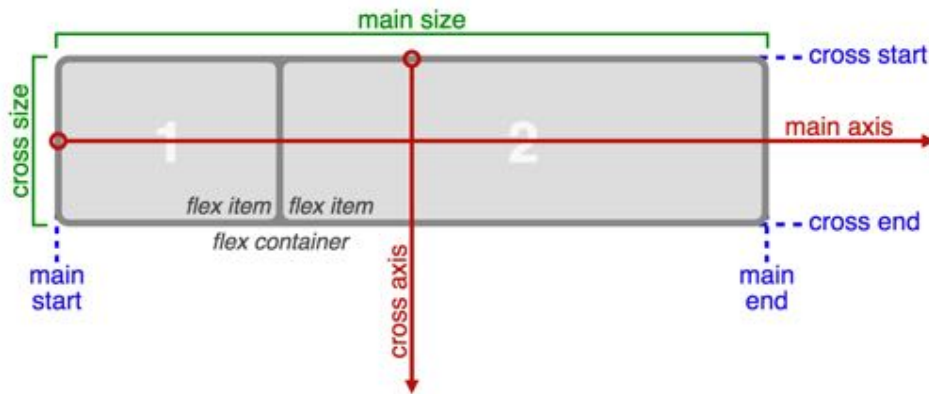
3.2.1. Rappels de cours

La mise en page avec les flexbox est principalement réalisée au niveau des conteneurs. L'idée est de définir le positionnement au niveau du conteneur, ce dernier aura la possibilité de changer la largeur et la hauteur des éléments contenus (items). Ainsi les item pourront être réduits ou augmentés pour s'adapter au mieux à l'espace et au choix de disposition du conteneur. Vous trouverez ici la documentation officielle : <https://www.w3.org/TR/css-flexbox-1/>, d'autres supports plus lisibles mais moins officiels sont disponibles ici : <https://la-cascade.io/flexbox-guide-complet/>, <https://www.alsacreations.com/tuto/lire/1493-css3-flexbox-layout-module.html>. Une grande partie de ce qui suit vient du guide complet.

Le positionnement flexbox repose sur l'utilisation algorithmes, ce positionnement est donc moins précis et plus consommateur de ressources que celui flottant.

3.2.1.1. Le modèle

Le positionnement habituel est basé sur les directions de flux block et inline, le positionnement flex, lui, est basé sur les directions "flex-flow". L'illustration ci-dessous, tirée des spécifications, explique l'idée qui est à la base du positionnement flex :

Figure 3.33. Flex container

Les items seront disposés soit sur l'axe principal (main axis) depuis main-start ou main-end, ou sur l'axe perpendiculaire (cross axis) en partant de cross-start ou de cross-end. Le flex-flow suit donc l'axe principal ou l'axe perpendiculaire.

main axis

L'axe principal d'un container flex est l'axe primaire sur lequel les items sont disposés. Attention, il n'est pas forcément horizontal, tout dépendra de la propriété justify-content (voir ci-dessous).

main-start | main-end

À l'intérieur du container, les items flex sont placés entre un point de départ (main-start) et un point d'arrivée (main-end).

main size

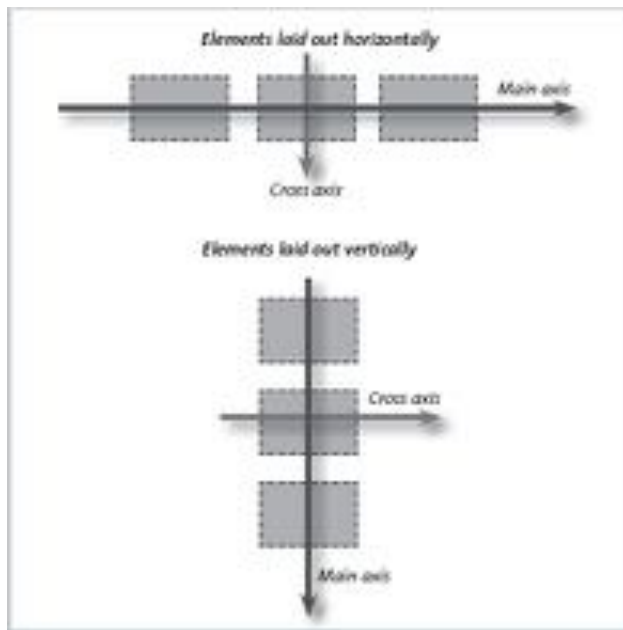
La taille d'un item flex qui se trouve dans l'axe principal, qu'il s'agisse de la hauteur ou de la largeur, est la taille principale (main size). La propriété main size est soit "width", soit "height", selon l'orientation. cross axis - L'axe perpendiculaire à l'axe principal est appelé cross axis. Sa direction dépend de la direction de l'axe principal.

cross-start | cross-end

Les lignes sont remplies avec les items et sont placées dans le container en partant du côté cross-start et en allant vers cross-end.

cross-size

La largeur ou la hauteur d'un item, selon la dimension dans laquelle on se trouve (même principe que main size).

Figure 3.34. Direction des axes

3.2.1.2. Propriétés

3.2.1.2.1. Propriétés du conteneur

3.2.1.2.1.1. display

C'est ainsi qu'on définit un container flex, il est block par défaut ou inline selon la valeur donnée. Cela crée un contexte flex pour tous les descendants directs.

```
display: flex | inline-flex;
```

flex

Cette valeur génère un container flex, de niveau block, à l'intérieur de l'élément

inline-flex

Cette valeur génère un container flex, de niveau inline, à l'intérieur de l'élément.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<span> Un truc après </span>
```

```
div.item {
  border: 1px solid black;
}
```


```
div.container {
  display: flex;
}
```

Figure 3.35. display: flex

Item 1Item 2Item 3
Un truc après

```
div.container {
  display: inline-flex;
}
```

Figure 3.36. display: inline-flex



3.2.1.2.1.2. flex-direction

La propriété flex-direction établit l'axe principal.

```
flex-direction: row | row-reverse | column | column-reverse
```

row (valeur par défaut)

de gauche à droite si la lecture se fait dans ce sens, de droite à gauche dans le cas inverse,

row-reverse

inverse le sens,

column

comme row mais du haut vers le bas,

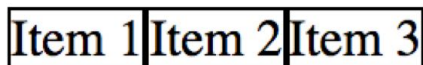
column-reverse

comme row-reverse mais du bas vers le haut.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

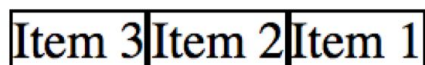
```
div.container {
  display: flex;
  flex-direction: row;
}
```

Figure 3.37. flex-direction: row



```
div.container {
  display: flex;
  flex-direction: row-reverse;
}
```

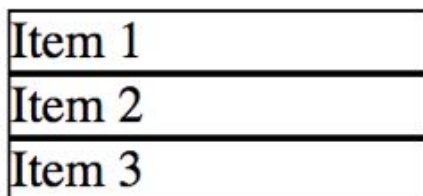
Figure 3.38. flex-direction: row-reverse



```
div.container {
```

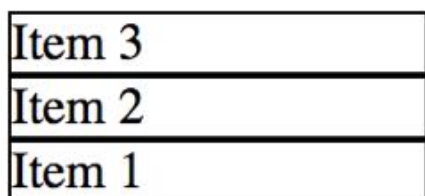
```
display: flex;
flex-direction: column;
}
```

Figure 3.39. flex-direction: column



```
div.container {
display: flex;
flex-direction: column-reverse;
}
```

Figure 3.40. flex-direction: column-reverse



3.2.1.2.1.3. flex-wrap

Cette propriété définit si le container comprend une seule ligne ou plusieurs et la direction sur l'axe perpendiculaire (cross-axis), qui détermine la direction dans laquelle les nouvelles lignes seront empilées.

flex-wrap: nowrap | wrap | wrap-reverse

nowrap (valeur par défaut)

sur une seule ligne, de gauche à droite dans un système ltr (left to right), sinon l'inverse. La ligne peut déborder de son contenant.

wrap

multiligne, de gauche à droite dans un système ltr, sinon l'inverse. Pas de débordement, on passe à la ligne.

wrap-reverse

multiligne, de droite à gauche dans un système ltr, sinon l'inverse.

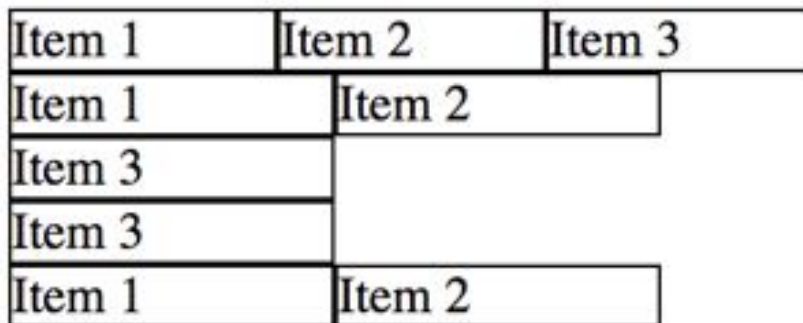
```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
```



```
<div class="item">Item 2</div>
<div class="item">Item 3</div>
</div>
```

```
div.container {
  display: flex;
}
div.container:nth-child(1){
  flex-wrap: nowrap;
}
div.container:nth-child(2){
  flex-wrap: wrap;
}
div.container:nth-child(3){
  flex-wrap: wrap-reverse;
}
```

Figure 3.41. flex-wrap



3.2.1.2.1.4. flex-flow

Cette propriété est un raccourci des propriétés "flex-direction" et "flex-wrap" qui ensemble définissent les axes "main" et "cross" du container flex. La valeur par défaut est row nowrap.

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

3.2.1.2.1.5. justify-content

La propriété justify-content définit l'alignement le long de l'axe principal. Elle permet de distribuer l'espace excédentaire lorsque tous les items flex sur une ligne sont inflexibles ou lorsqu'ils ont atteint leur taille maximale. Elle contrôle aussi l'alignement des items lorsqu'ils débordent.

```
justify-content: flex-start | flex-end | center | space-between | space-around
```

flex-start (par défaut)

les items sont regroupés en début de ligne

flex-end

les items sont regroupés en fin de ligne

center

les items sont centrés le long de la ligne

space-between

les items sont répartis sur la ligne; le premier est collé du côté start, le dernier du côté end.

space-around

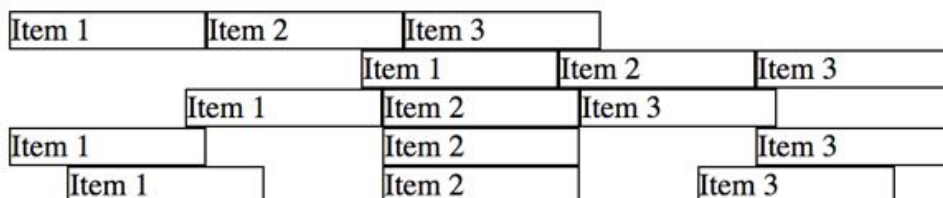
les items sont répartis sur la ligne avec un espacement égal autour de chacun.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```
div.container {
  display: flex;
}

div.container:nth-child(1){
  justify-content: flex-start;
}
div.container:nth-child(2){
  justify-content: flex-end;
}
div.container:nth-child(3){
  justify-content: center;
}
div.container:nth-child(4){
  justify-content: space-between;
}
div.container:nth-child(5){
  justify-content: space-around;
}
```

Figure 3.42. justify-content



3.2.1.2.1.6. align-items

La propriété `align-items` définit la façon dont les items d'une ligne sont disposés le long de l'axe "cross". On peut le voir comme la version de `justify-content` pour "cross axis".

```
align-items: flex-start | flex-end | center | baseline | stretch
```

flex-start

l'item est placé au début de la ligne cross-start. flex-end

flex-end

la marge "cross-end" de l'item est placée sur la ligne cross-end

center

les items sont centrés sur l'axe cross

baseline

les items sont alignés sur leur ligne de base

stretch (par défaut)

les items sont étirés jusqu'à remplir le container (tout en respectant min-width/max-width)

```
div.container {  
  display: flex;  
  height: 100px;  
  line-height: 50px;  
  border: 1px solid black;  
}  
  
div.container:nth-child(1){  
  align-items: flex-start;  
}  
div.container:nth-child(2){  
  align-items: flex-end;  
}  
div.container:nth-child(3){  
  align-items: center;  
}  
div.container:nth-child(4){  
  align-items: baseline;  
}  
div.container:nth-child(5){  
  align-items: stretch;  
}
```

Figure 3.43. align-items

Item 1	Item 2	Item 3	
Item 1	Item 2	Item 3	
Item 1	Item 2	Item 3	
Item 1	Item 2	Item 3	
Item 1	Item 2	Item 3	

3.2.1.2.1.7. align-content

La propriété align-content aligne les lignes d'un container flex à l'intérieur de l'espace où il reste de l'espace sur l'axe cross, un peu comme justify-content aligne les items sur l'axe principal. Note : cette propriété n'a pas d'effet quand la flexbox n'a qu'une seule ligne.

```
align-content: flex-start | flex-end | center | space-between | space-around  
| stretch
```

flex-start

lignes regroupées au début du container

flex-end

lignes regroupées à la fin du container

center

lignes regroupées au centre du container

space-between

les lignes sont réparties, la première est collée du côté start, la dernière du côté end

space-around

les lignes sont réparties avec un espacement égal autour de chacune

stretch (par défaut)

les lignes s'étirent pour remplir tout l'espace.

```
div.container {  
  display: flex;  
  height: 100px;  
  flex-wrap: wrap;  
  border: 1px solid black;  
}  
  
div.container:nth-child(1){  
  align-content: flex-start;  
}  
div.container:nth-child(2){  
  align-content: flex-end;  
}  
div.container:nth-child(3){  
  align-content: center;  
}  
div.container:nth-child(4){  
  align-content: space-between;  
}  
div.container:nth-child(5){  
  align-content: space-around;  
}  
div.container:nth-child(6){  
  align-content: stretch;  
}
```

Item 1	Item 2
Item 3	
Item 1	Item 2
Item 3	
Item 1	Item 2
Item 3	
Item 1	Item 2
Item 3	
Item 1	Item 2
Item 3	
Item 1	Item 2
Item 3	

CSS

3.2.1.2.2. Propriétés des items

3.2.1.2.2.1. order

Par défaut, les items flex sont disposés par ordre d'arrivée. Cependant, la propriété `order` permet de contrôler l'ordre dans lequel ils apparaissent dans le container.

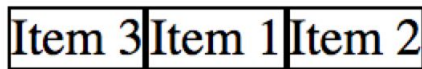
`order: <nombre entier>`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```
div.container {
  display: flex;
}

div.item:nth-child(1){
  order : 2
}
div.item:nth-child(2){
  order : 3
}
div.item:nth-child(3){
  order: 1
}
```

Figure 3.45. `order`



3.2.1.2.2.2. flex-grow

La propriété `flex-grow` définit la possibilité pour un item de grandir, si nécessaire. Elle accepte une valeur sans unité qui sert de proportion. Elle dicte l'espace que peut prendre l'item à l'intérieur de l'espace disponible dans le flex container. Si tous les items ont `flex-grow` défini à 1, chaque enfant aura le même espace dans le container. Si vous donnez à l'un des enfants une valeur de 2, cet enfant prendra deux fois plus de place que les autres.

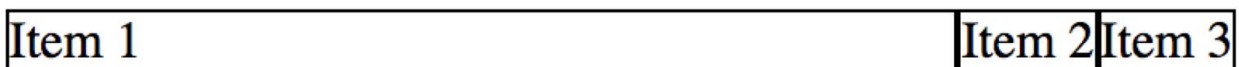
`flex-grow: <nombre entier> (par défaut = 0)`

Les chiffres négatifs ne sont pas valides.

```
div.container {
  display: flex;
}

div.item:nth-child(1){
  flex-grow : 2
}
```

Figure 3.46. `flex-grow`



3.2.1.2.2.3. flex-shrink

La propriété `flex-shrink` définit la possibilité pour un item flex de rétrécir si nécessaire.

`flex-shrink: <nombre entier> (par défaut = 1)`

Les chiffres négatifs ne sont pas valides.

```
div.item {
  border: 1px solid black;
  flex-basis: 40%
}

div.container {
  display: flex;
}

div.item:nth-child(1){
  flex-shrink: 8;
}
div.item:nth-child(2){
  flex-shrink: 1;
}
div.item:nth-child(3){
  flex-shrink: 1;
}
```

Figure 3.47. flex-shrink

Item 1	Item 2	Item 3
--------	--------	--------

3.2.1.2.2.4. flex-basis

`flex-basis` La propriété `flex-basis` définit la taille par défaut d'un élément avant que l'espace restant soit réparti.

`flex-basis: <longueur> | auto (par défaut = auto)`

```
div.container {
  display: flex;
}

div.item:nth-child(1){
  flex-basis: 40%
}
div.item:nth-child(2){
  flex-basis: 10%
}
div.item:nth-child(3){
  flex-basis: 50%
}
```

Figure 3.48. flex-basis

Item 1	Item 2	Item 3
--------	--------	--------

3.2.1.2.2.5. flex

Cette propriété est le raccourci de `flex-grow`, `flex-shrink` et `flex-basis`. Les deuxième et troisième paramètres sont optionnels. La valeur par défaut est `0 1 auto`.

`flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]`

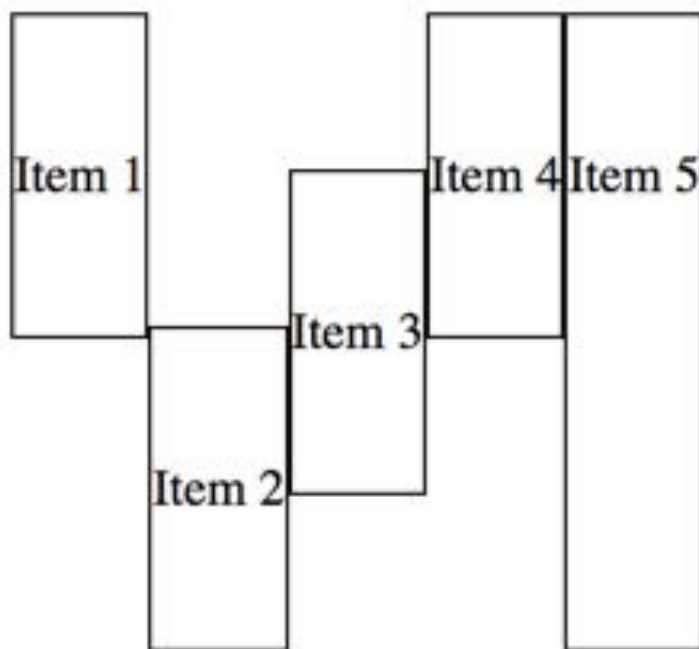
3.2.1.2.2.6. align-self

La propriété `align-self` permet à des items flex de passer outre aux alignements par défaut ou à ceux spécifiés par `align-items`. Les valeurs sont les mêmes que pour ce dernier.

`align-self: auto | flex-start | flex-end | center | baseline | stretch`


```
div.item {  
  border: 1px solid black;  
  line-height: 100px;  
}  
  
div.container {  
  display: flex;  
  height: 200px;  
}  
  
div.item:nth-child(1){  
  align-self: flex-start  
}  
div.item:nth-child(2){  
  align-self: flex-end  
}  
div.item:nth-child(3){  
  align-self: center  
}  
div.item:nth-child(4){  
  align-self: baseline  
}  
div.item:nth-child(5){  
  align-self: stretch  
}
```

Figure 3.49. align-self



3.2.2. Exercices

3.2.2.1. Exercice 0

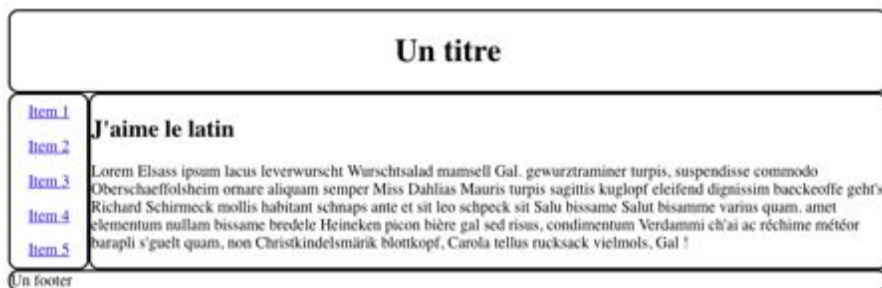
Le site <http://flexboxfroggy.com/#fr>, vous propose de tester les flexbox. Essayez de réaliser les exercices.

3.2.2.2. Exercice 1

Dans cet exercice nous allons réaliser une `div` centrée horizontalement et verticalement. Vous utiliserez les fichiers `exercice1.html` et `exercice1.css`. Seul `exercice1.css` est à modifier.

Figure 3.50. Une div centrée**3.2.2.3. Exercice 2**

Dans cet exercice nous allons réaliser un gabarit avec un header, un menu vertical à gauche et un footer. Les fichiers à utiliser sont `exercice2.html` et `exercice2.css`. Seul `exercice2.css` est à modifier. Vous ne devrez exception faite des bordures n'utiliser que des propriétés flex.

Figure 3.51. Que du flex**3.2.2.4. Exercice 3**

Nous avons réalisé un design en `inline-block` nous allons le reproduire à moindre coût en flexbox. L'HTML a été augmenté de nouvelles div, nous n'avez pas à le modifier, seul `exercice3.css` est à modifier.

3.3. Pour aller plus loin

Il est possible d'adapter la présentation à la résolution du périphérique, les media queries répondent à ce problème en offrant des règles qui ne s'activent que pour certaines résolutions.

L'exercice et les images qui suivent sont tirées de <https://www.w3schools.com>. Vous devez produire les 3 affichages suivants avec la même feuille de style `exercice4.css`, l'HTML n'est pas à modifier :

Figure 3.52. Résolution horizontale de plus de 800 pixels

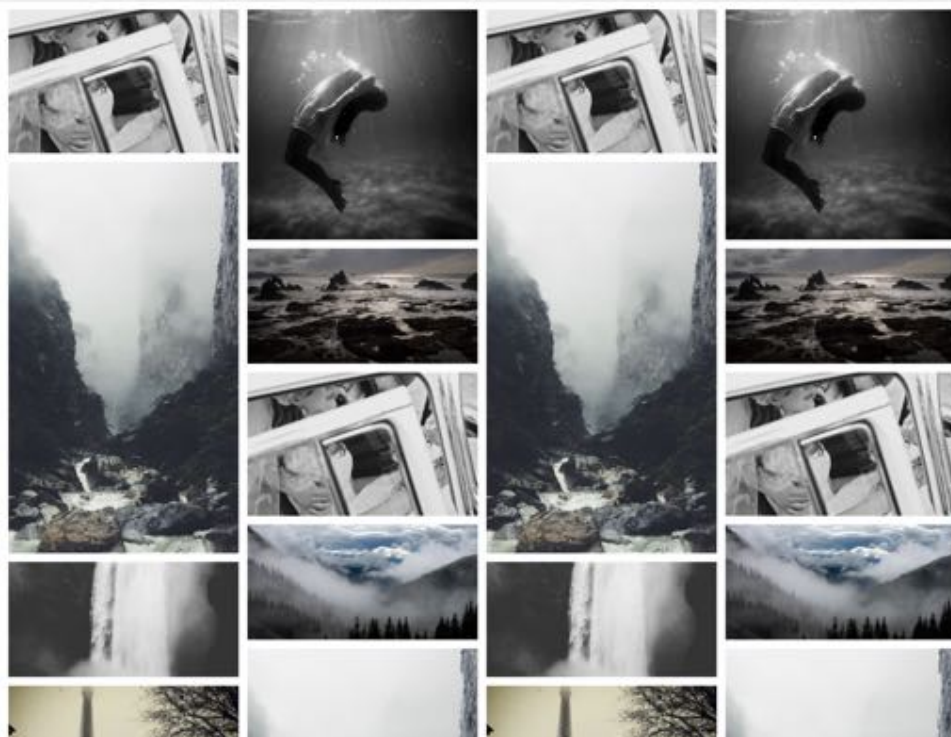


Figure 3.53. Résolution horizontale entre 800 et 600 pixels



Figure 3.54. Résolution horizontale de moins de 600 pixels



4. TP 6 : Système de grille

Définir des propriétés sur des conteneurs et items peut-être compliqué, une idée simple est de découper la page en une grille dont on consomme des lignes et des colonnes.

Les systèmes de grilles sont nombreux, pratiques mais ne répondent pas à tous les problèmes. Certains intégrateurs attachés à l'approche sémantique ou toutes les balises doivent avoir un sens y seront opposés, d'autres plus pragmatiques ne seront pas gênés par du tout en ou les balises traduisent directement un résultat visuel.

Parmi les critères de choix des systèmes de grilles nous pouvons trouver :

- fixe (pixels, nombre de colonnes figé), ou fluide (pourcentages, colonnes variables) ?
- complexité de la mise en oeuvre ?
- évolutive (possibilité de modifier le nombre d'éléments), ou figée ?
- compatible anciens navigateurs ou non ?

- offsets, pull et push (capacité à réaliser des “trous” dans la grille),
- gouttière ou pas gouttière ?

Voici un exemple de système de grille en douze colonnes réalisé avec des flexbox.

```
* {
  box-sizing: border-box;
}

.row {
  display: flex;
  flex-wrap: wrap;
}

.row.nested {
  margin-left: -10px;
  margin-right: -10px;
  margin-top: -10px;
  margin-bottom: -10px;
}

[class*="col-"] {
  flex-shrink: 0;
  padding: 10px;
  box-sizing: border-box;
}

[class*="col-"].no-padding {
  padding: 0;
}

.col-1 {width: 8.3333%;}
.col-2 {width: 16.6667%;}
.col-3 {width: 25%;}
.col-4 {width: 33.3333%;}
.col-5 {width: 41.6667%;}
.col-6 {width: 50%;}
.col-7 {width: 58.3333%;}
.col-8 {width: 66.6667%;}
.col-9 {width: 75%;}
.col-10 {width: 83.3333%;}
.col-11 {width: 91.6667%;}
.col-12 {width: 100%;}
```

Important

Il vous faut comprendre cette CSS avant d'aller plus loin.

- Avantages
 - Il s'agit d'un positionnement dans le flux
 - Les hauteurs des colonnes sont identiques par défaut
 - S'il n'y a pas besoin de gouttière, ou si la taille de gouttière importe peu, alors le positionnement sera très intuitif (justify-content: space-between;)
 - Pull et push très intuitifs (un simple margin-left: auto; ou margin-right: auto; suffit)
- Inconvénients
 - Si la grille nécessite des gouttières, la présence d'un conteneur global supplémentaire sera indispensable

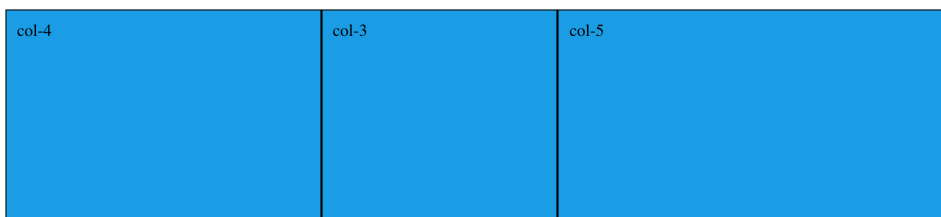
- La gestion des gouttières nécessite quelques adaptations non intuitives (marges négatives et/ou padding, overflow: hidden sur le conteneur, etc.)
- Préfixes constructeurs encore nécessaires

Voici un exemple de trois boîtes identiques :

```
<style>
  div > div {height: 200px;
    background-color: #009CE8;
    border: 1px solid black;}
</style>

<div class="row">
  <div class="col-4">
    col-4
  </div>
  <div class="col-3">
    col-3
  </div>
  <div class="col-5">
    col-5
  </div>
</div>
```

Figure 3.55. Exemple de grille avec Flexbox



4.1. Exercices

4.1.1. Exercice 1

Seul le fichier `exercice1.html` est à modifier. Vous devez reproduire l'écran suivant :

Figure 3.56. Flex grille exercice 1



4.1.2. Exercice 2

Seul le fichier `exercice2.html` est à modifier. Vous devez reproduire l'écran suivant :

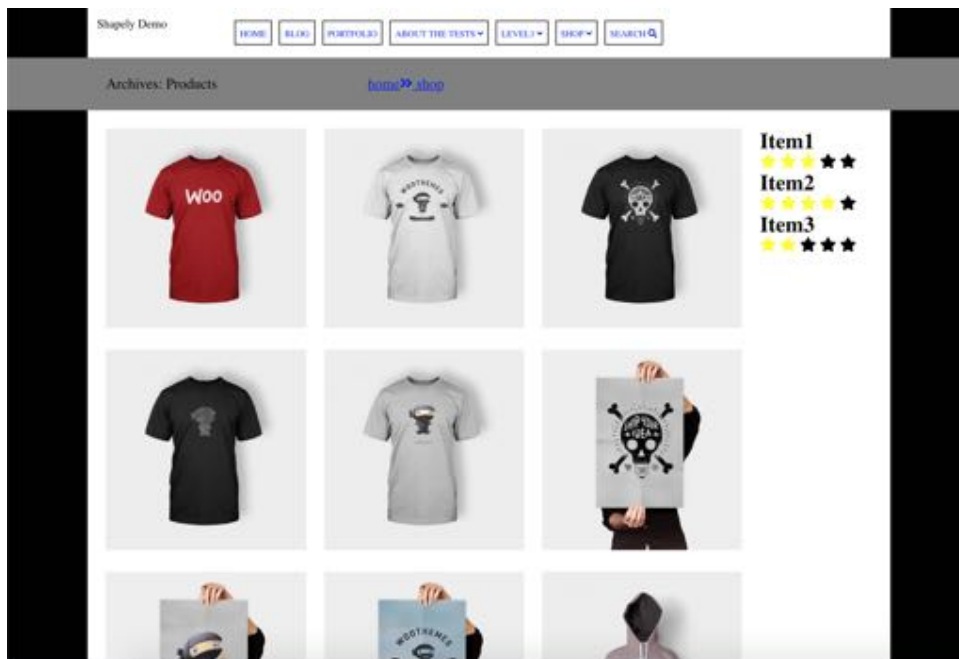
Figure 3.57. Flex grille exercice 2



4.1.3. Exercice 3

Seul le fichier `exercice3.html` est à modifier. Vous remarquerez que le micro framework a été augmenté pour gérer des offset (`.left-?` et `.right-?`). Vous devez reproduire l'écran suivant :

Figure 3.58. Flex grille exercice 3



4.1.4. Exercice 4

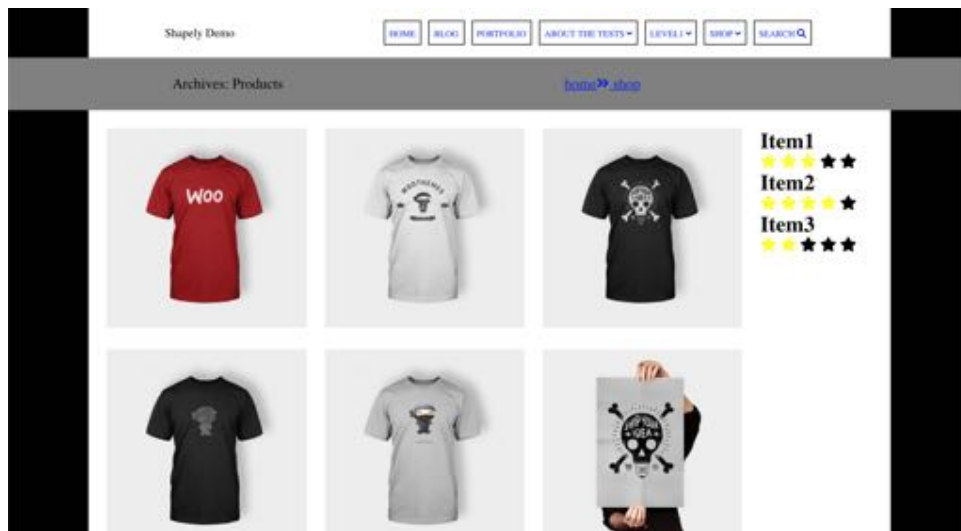
Notre micro framework est limité, vous allez le modifier (`gabarit_12_colonnes_full.css`) pour obtenir avec `exercice4.html` l'affichage suivant.

Figure 3.59. Flex grille exercice 4

A ce stade, vous avez construit un micro framework très proche de : <https://lab.anybodesign.com/pridx/>.

4.1.5. Exercice 5

Seul le fichier `exercice5.html` est à modifier. Pour obtenir l'écran suivant :

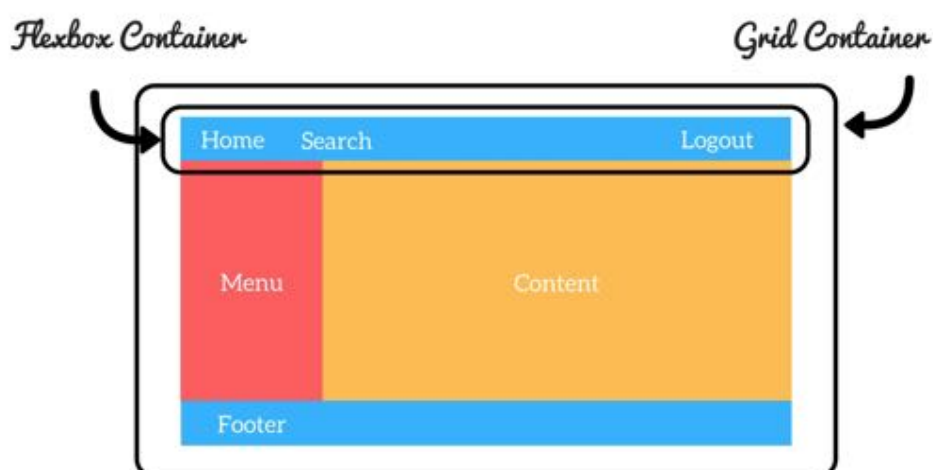
Figure 3.60. Flex grille exercice 5

4.2. Pour aller plus loin

Nous avons un gabarit 12 colonnes, nous souhaitons un gabarit n colonnes, la solution est l'utilisation d'un langage dynamique de compilation de feuilles de style comme Sass (Syntactically Awesome Stylesheets). Le compilateur sass est installé sur vos machines, vous disposez du fichier `gabarit_x_colonnes_full_sass.scss` à faire évoluer, vous aurez probablement besoin de `for` et de `&`. Vous trouverez l'information nécessaire ici : <https://sass-lang.com/>.

5. TP 7 : CSS Grid

Le conteneur flexbox a la possibilité de redimensionner et de réordonner ses composants, il reste cependant limité à une dimension, il faut inclure d'autres conteneur pour créer la deuxième dimension, les CSS Grid répondent à ce problème.

Figure 3.61. Flexbox CSS Grid

5.1. Rappels de cours

Les CSS Grid, ne sont supportées que par les navigateurs récents et le module n'en est encore qu'à l'état de draft (<https://drafts.csswg.org/css-grid/>).

Pour faire simple `display:grid` permet d'activer le système de grille, les dimensions sont choisies avec `grid-template-columns` et `grid-template-rows`. Les éléments enfants sont placés dans la grille avec `grid-column` et `grid-row`. Le système permet comme pour Flexbox de modifier l'ordre du placement. Le système peut aussi se redimensionner avec des items non initialement prévus.

5.1.1. Les propriétés des conteneurs

Avant de commencer, il faut savoir que `float`, `vertical-align`, `clear`, `column` n'ont aucun effet sur un conteneur grille. Il vous faudra utiliser les propriétés du système de grille.

5.1.1.1. display

La propriété `display` peut prendre 2 valeurs `grid` ou `inline-grid`, `grid` pour être en bloc et `inline-grid` pour être inline.

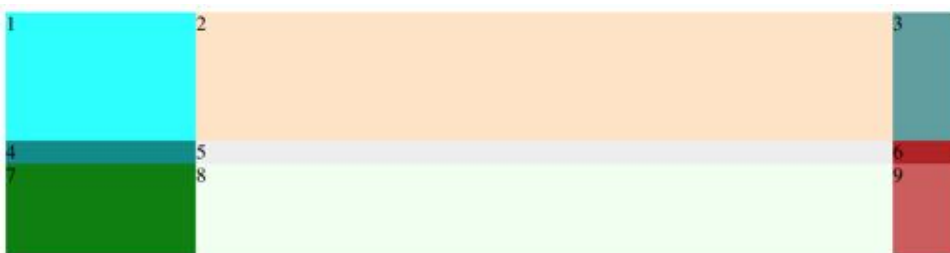
5.1.1.2. grid-template-columns, grid-template-rows

Ces 2 propriétés permettent de définir les colonnes et les lignes, voir de les nommer. Une nouvelle unité est introduite, la fraction `fr`. Dans l'exemple qui suit, nous avons une grille avec 3 colonnes et 3 ligne. Les pourcentages et les fractions sont calculées après les valeurs fixes.

```
<style>
  .container{
    display: grid;
    grid-template-columns: 20% auto 50px;
    grid-template-rows: 100px 1fr 4fr;
  }
</style>

<!-- les c? ne sont utilisés que pour les couleurs -->
<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
  <div class="c5"> 5 </div>
  <div class="c6"> 6 </div>
  <div class="c7"> 7 </div>
  <div class="c8"> 8 </div>
  <div class="c9"> 9 </div>
</div >
```

Figure 3.62. Première grille



Il est également possible de réaliser des boucles.

```
<style>
  .container{
    display: grid;
    grid-template-columns: repeat(4, 1fr) 4fr;
    /* 1fr 1fr 1fr 1fr 4fr */
    grid-template-rows: 1fr;
  }
</style>
```

```
<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
  <div class="c5"> 5 </div>
  <div class="c6"> 6 </div>
  <div class="c7"> 7 </div>
  <div class="c8"> 8 </div>
  <div class="c9"> 9 </div>
</div >
```

Figure 3.63. repeat

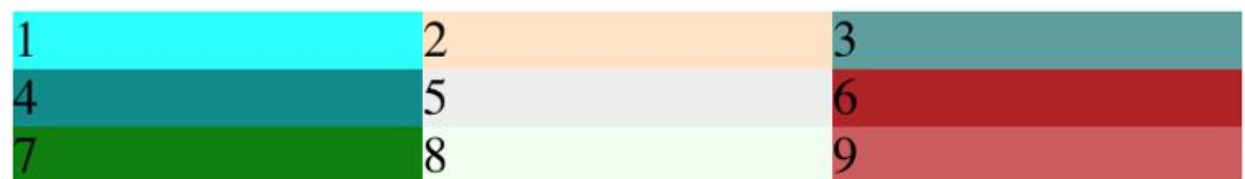


La valeur auto, permet un placement automatique.

```
<style>
  .container{
    display: grid;
    grid-template-columns: auto auto auto;
  }
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
  <div class="c5"> 5 </div>
  <div class="c6"> 6 </div>
  <div class="c7"> 7 </div>
  <div class="c8"> 8 </div>
  <div class="c9"> 9 </div>
</div >
```

Figure 3.64. auto



L'exemple suivant introduit la possibilité de nommage. Le nommage pourra être utilisé pour placer les items. Une colonne ou une ligne a sont nom placés entre crochets et peut avoir plusieurs noms séparés par des espaces.

```
<style>
  .container{
    display: grid;
    grid-template-columns: [main-start] 1fr [content-start] 1fr
                          [content-end] 1fr [main-end];
    grid-template-rows: [main-start] 100px [content-start] 100px
                       [content-end] 100px [main-end];
  }
  .c1 {
    grid-column-start: main-start;
    grid-row-start: main-start;
    grid-row-end: main-end;
    background-color: aqua;
  }
</style>
```

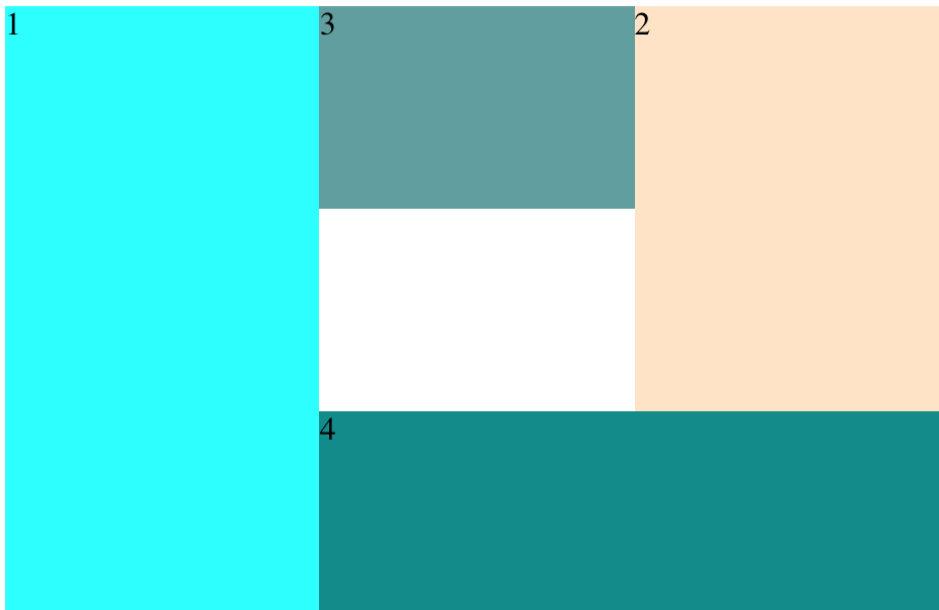
```

}
.c2 {
  grid-column-start: content-end;
  grid-row-start: main-start;
  grid-row-end: content-end;
  background-color: bisque;
}
.c3 {
  grid-column-start: content-start;
  grid-row-start: main-start;
  background-color: cadetblue;
}
.c4 {
  grid-column-start: content-start;
  grid-column-end: main-end;
  grid-row-start: content-end;
  background-color: darkcyan;
}
}
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
</div >

```

Figure 3.65. Les nommages



Si une grille est elle-même un item de grille alors la valeur `subgrid` peut-être utilisée pour hériter des propriétés de la grille mère.

```

grid-template-columns: <track-size> ... | <line-name> <track-size> ... | subgrid;
grid-template-rows: <track-size> ... | <line-name> <track-size> ... | subgrid;

```

5.1.1.3. grid-template-areas

`grid-template-areas` permet de définir un template de grille utilisé par les la propriété `grid-area`. Si l'on répète le nom d'une zone de grille, cela étend la surface couverte par ces cellules. Un point (.) signifie que la cellule est vide. La syntaxe elle-même fournit une visualisation de la structure de la grille.

```

<style>
  .container{
    display: grid;

```

```

    grid-template-columns: repeat(4, 25%);
    grid-template-rows: auto;
    grid-template-areas: "header header header header"
                        "sidebar . content content"
                        "footer footer footer footer";
  }
  .c1 {
    grid-area: header;
    background-color: aqua;
  }
  .c2 {
    grid-area: sidebar;
    background-color: bisque;
  }
  .c3 {
    grid-area: content;
    background-color: cadetblue;
  }
  .c4 {
    grid-area: footer;
    background-color: darkcyan;
  }
}
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
</div>

```

Figure 3.66. grid-template-areas



5.1.1.4. grid-column-gap, grid-row-gap et grid-gap

Le moyen de créer des gouttières entre les colonnes (`grid-column-gap`) ou les lignes (`grid-row-gap`). `grid-gap` est un raccourci `grid-gap`: `<grid-column-gap> <grid-row-gap>`.

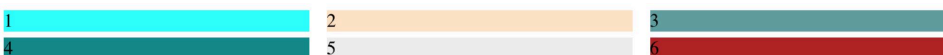
```

<style>
  .container {
    display: grid;
    grid-template-columns: repeat(3, 25%);
    grid-template-rows: auto;
    grid-column-gap: 15px;
    grid-row-gap: 5px;
  }
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
  <div class="c4"> 4 </div>
  <div class="c5"> 5 </div>
  <div class="c6"> 6 </div>
</div>

```

Figure 3.67. Les gouttières



Si `grid-row-gap` n'est pas positionné alors il prend la valeur de `grid-col-gap`.

5.1.1.5. justify-items, align-items

Ces propriétés sont équivalentes à celles des flexbox, `justify-items` travail sur l'axe horizontal (l'axe des colonnes) et `align-items` sur l'axe vertical (l'axe des lignes).

`justify-items :`

`start`

aligne le contenu à partir de la gauche

`end`

aligne le contenu en partant de la droite

`center`

aligne le contenu au centre

`stretch`

remplit toute la largeur

`align-items :`

`start`

aligne le contenu à partir du haut

`end`

aligne le contenu en partant du bas

`center`

aligne le contenu au centre

`stretch`

remplit toute la hauteur

5.1.1.6. justify-content, align-content

Si vous utilisez des unités non flexibles, votre grille peut-être plus petite que son conteneur, les propriétés `justify-content` et `align-content` permettent de modifier le placement et le dimensionnement de la grille. L'exemple suivant illustre le problème des unités non flexibles.

```
<style>

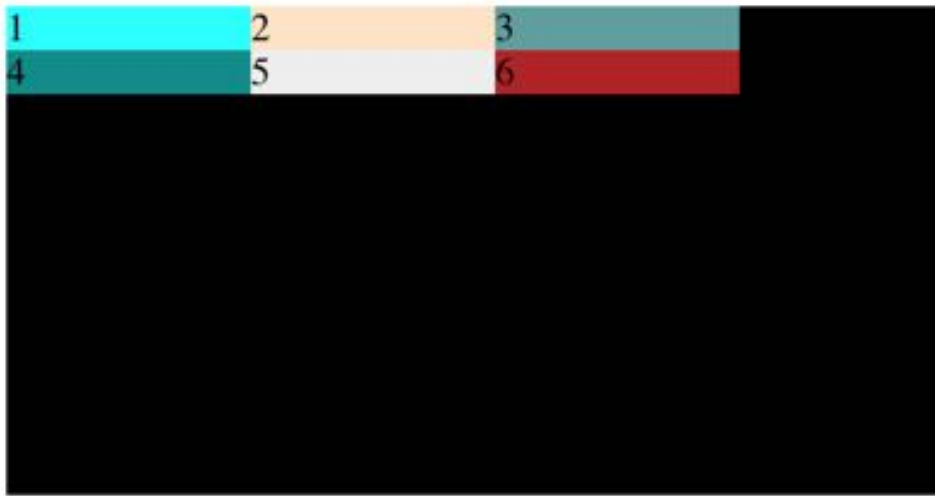
  #sup {
    background-color: black;
    height: 200px;
  }
  div.container {
    display: grid;
    grid-template-columns: repeat(3, 100px);
    grid-template-rows: auto;
  }

</style>

<div id="sup">
  <div class="container">
    <div class="c1"> 1 </div>
    <div class="c2"> 2 </div>
    <div class="c3"> 3 </div>
    <div class="c4"> 4 </div>
    <div class="c5"> 5 </div>
```

```
<div class="c6"> 6 </div>
</div >
</div>
```

Figure 3.68. Positionnement de la grille



`justify-content :`

`start`

aligne la grille en partant de la gauche

`end`

aligne la grille à partir de la droite

`center`

aligne la grille au centre

`stretch`

redimensionne les items pour permettre à la grille de remplir toute la largeur

`space-around`

place un espace égal entre chaque item

`space-between`

place un espace égal entre chaque item et aucun espace aux extrémités

`space-evenly`

place un espace égal entre chaque item de grille, y compris aux extrémités

`align-content :`

`start`

aligne la grille en partant du sommet

`end`

aligne la grille à partir de la base

`center`

aligne la grille au centre

stretch

redimensionne les items pour permettre à la grille de remplir toute la hauteur

space-around

place un espace égal entre chaque item

space-between

place un espace égal entre chaque item et aucun espace aux extrémités

space-evenly

place un espace égal entre chaque item de grille, y compris aux extrémités

5.1.1.7. grid-auto-columns, grid-auto-rows

Spécifie la dimension de toute piste auto-générée (également appelée piste implicite). Les pistes implicites sont créées lorsque vous positionnez explicitement des rangées ou des colonnes (via `grid-template-rows` ou `grid-template-columns`) qui se trouvent en dehors de la grille définie.

```
<style>
  .container {
    display: grid;
    grid-template-columns: 1fr 100px;
    grid-template-rows: 1fr 2fr;

    grid-auto-rows: 125px;
    grid-auto-columns: 1fr;
  }
  .c1 {
    background-color: aqua;
    grid-column: 3;
  }
  .c2{
    background-color: bisque;
    grid-row: 3;
  }
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
</div >
```

Figure 3.69. grid-auto



5.1.1.8. grid-auto-flow

Un algorithme de placement automatique intervient pour placer automatiquement les items non explicitement placés. La propriété `grid-auto-flow` paramètre l'algorithme. Cette propriété contrôle la façon dont l'algorithme de placement automatique fonctionne :

row

indique à l'algorithme de remplir chaque rangée tout à tour, en ajoutant de nouvelles rangées si nécessaire

column

indique à l'algorithme de remplir chaque colonne tout à tour, en ajoutant de nouvelles colonnes si nécessaire

dense

indique à l'algorithme d'essayer de remplir les trous le plus tôt possible dans la grille au cas où de plus petits items devaient apparaître plus tard

5.1.1.9. grid

```
grid: none |
      subgrid |
      <grid-template-rows> / <grid-template-columns> |
      <grid-auto-flow> [<grid-auto-rows> [/ <grid-auto-columns>]];
```

5.1.2. Les propriétés des enfants

Tout comme en flexbox, il est possible de donner des propriétés spécifiques aux enfants.

5.1.2.1. grid-column-start, grid-column-end, grid-row-start, grid-row-end

Ces propriétés permettent de placer explicitement les composants:

```
grid-column-start: <number> | <name> | span <number> | span <name> | auto
grid-column-end:   <number> | <name> | span <number> | span <name> | auto
grid-row-start:    <number> | <name> | span <number> | span <name> | auto
grid-row-end:      <number> | <name> | span <number> | span <name> | auto
```

5.1.2.2. grid-column, grid-row

La propriété grid-column est une propriété raccourcie pour grid-column-start et La propriété grid-row est une propriété raccourcie pour grid-row-start et grid-row-end. Elles permettent de définir la taille et l'emplacement d'un élément sur la grille en indiquant l'emplacement du début, de la fin et/ou sa taille.

```
<grid-line> [ / <grid-line> ]?
```

avec

```
<grid-line> = auto | <custom-ident> | [ <integer> && <custom-ident>? ] | [ span && [ <integer> || <custom-
```

```
<grid-line> [ / <grid-line> ]?
```

avec

```
<grid-line> = auto | <custom-ident> |
              [ <integer> && <custom-ident>? ] |
              [ span && [ <integer> || <custom-ident> ] ]
```

```
<style>
  .container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: auto;
  }
  .c1 {
    background-color: aqua;
    grid-column: 1/3;
  }
  .c2{
    background-color: bisque;
  }
  .c3{
    background-color: cadetblue;
  }
</style>
<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
</div >
```

Figure 3.70. grid-column 1/3

```
<style>
  .container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: auto;
  }
  .c1 {
    background-color: aqua;
    grid-column: 2/-1;
  }
  .c2{
    background-color: bisque;
  }
  .c3{
    background-color: cadetblue;
  }
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
</div >
```

Figure 3.71. grid-column 2/-1

```
<style>
  .container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: auto;
  }
  .c1 {
    background-color: aqua;
    grid-column: 1 / span 2;
  }
  .c2{
    background-color: bisque;
  }
  .c3{
    background-color: cadetblue;
  }
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
</div >
```

Figure 3.72. gid-column span

5.1.2.3. grid-area

La propriété `grid-area` est une propriété raccourcie pour `grid-row-start`, `grid-column-start`, `grid-row-end` et `grid-column-end` qui permet de définir la taille d'un objet de la grille et son emplacement.

```
<grid-line> [ / <grid-line> ]{0,3}
avec
<grid-line> = auto | <custom-ident> |
              [ <integer> && <custom-ident>? ] |
              [ span && [ <integer> || <custom-ident> ] ]
```

Si quatre valeurs `<grid-line>` sont fournies, la première sera appliquée à `grid-row-start`, la deuxième à `grid-column-start`, la troisième à `grid-row-end` et la quatrième à `grid-column-end`.

```
<style>
.container {
  display: grid;
  grid-template: repeat(4, 1fr) / auto;
}
.c1 {
  background-color: aqua;
  grid-area: 2 / 2 / auto / span 3;
}
.c2{
  background-color: bisque;
}
.c3{
  background-color: cadetblue;
}
</style>

<div class="container">
  <div class="c1"> 1 </div>
  <div class="c2"> 2 </div>
  <div class="c3"> 3 </div>
</div >
```

Figure 3.73. grid-area 4 paramètres



5.1.2.4. justify-self

La propriété CSS `justify-self` définit la façon dont une boîte est alignée sur l'axe en ligne du conteneur.

```
auto | normal | stretch |
<baseline-position> |
<overflow-position>? [ <self-position> | left | right ]
avec
<baseline-position> = [ first | last ]? baseline
<overflow-position> = unsafe | safe
<self-position> = center | start | end | self-start |
                  self-end | flex-start | flex-end
```

```
<style>
.container {
  display: grid;
  grid-template: repeat(4, 1fr) / auto;
}
.c1 {
  background-color: aqua;
  justify-self: start;
}
```

```

    }
    .c2{
        background-color: bisque;
        justify-self: end;
    }

    .c3{
        background-color: cadetblue;
    }
</style>

<div class="container">
    <div class="c1"> 1 </div>
    <div class="c2"> 2 </div>
    <div class="c3"> 3 </div>
</div >

```

Figure 3.74. justify-content



5.1.2.5. align-self

La propriété CSS `align-self` permet d'aligner les objets flexibles d'une ligne flexible en surchargeant la valeur donnée par `align-items`. Si l'un des objet a une marge automatique (`auto`) pour l'axe perpendiculaire à l'axe principal, `align-self` sera ignoré. Lorsque le conteneur est une grille, `align-self` permet d'aligner l'élément au sein de la zone de grille.

```

auto | normal | stretch | <baseline-position> |
<overflow-position>? <self-position>
avec
<baseline-position> = [ first | last ]? baseline
<overflow-position> = unsafe | safe
<self-position> = center | start | end |
                  self-start | self-end | flex-start | flex-end

```

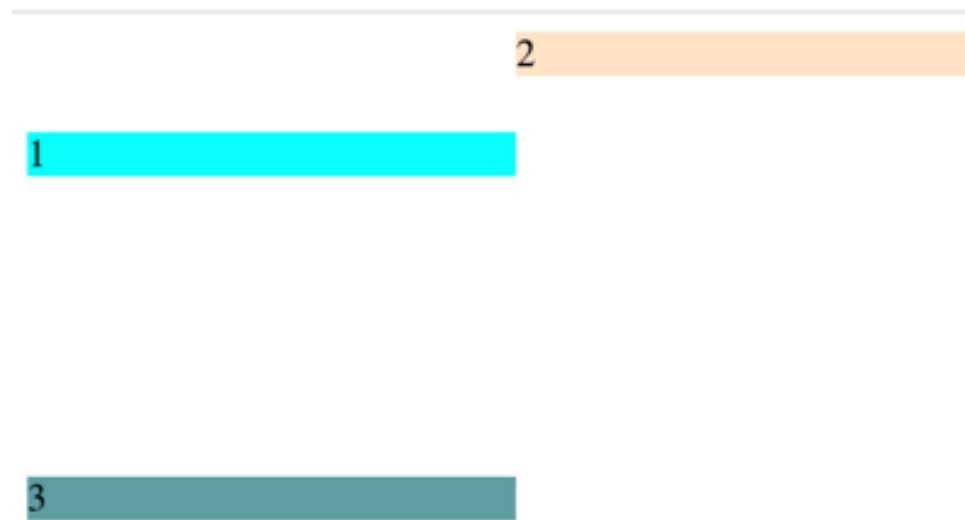
```

<style>
    .container {
        display: grid;
        grid-template: repeat(4, 1fr) / 200px 300px;
        height: 400px;
    }
    .c1 {
        background-color: aqua;
        align-self: center;
    }
    .c2{
        background-color: bisque;
        align-self: baseline;
    }

    .c3{
        background-color: cadetblue;
        align-self: end;
    }
</style>

<div class="container">
    <div class="c1"> 1 </div>
    <div class="c2"> 2 </div>
    <div class="c3"> 3 </div>
</div >

```

Figure 3.75. align-self

5.2. Exercices

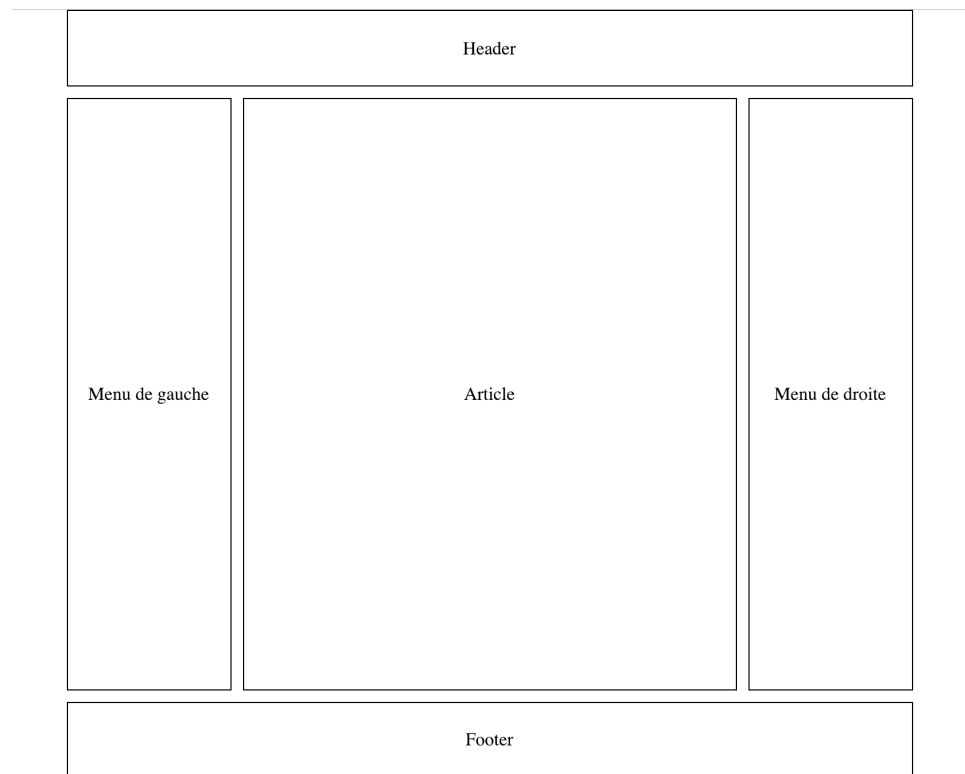
Le petit site suivant contient un bon résumé des propriétés du système de grilles : <http://grid.malven.co/>. Dans l'ensemble des exercices seuls les fichiers CSS sont à modifier.

5.2.1. Exercice 0

Comme de coutume, commençons par un jeu en ligne : <https://cssgridgarden.com/#fr>.

5.2.2. Exercice 1

Dans cet exercice les colonnes sont réparties en 1 fragment, 3 fragments et 1 fragments. Les lignes sont réparties en 10% à gauche, 10 à droite et le reste pour le centre. L'exercice est à réaliser sans utiliser `grid-template-areas` ou de nommage des colonnes et lignes. Vous utiliserez `exercice1.css` et `exercice1.html`.

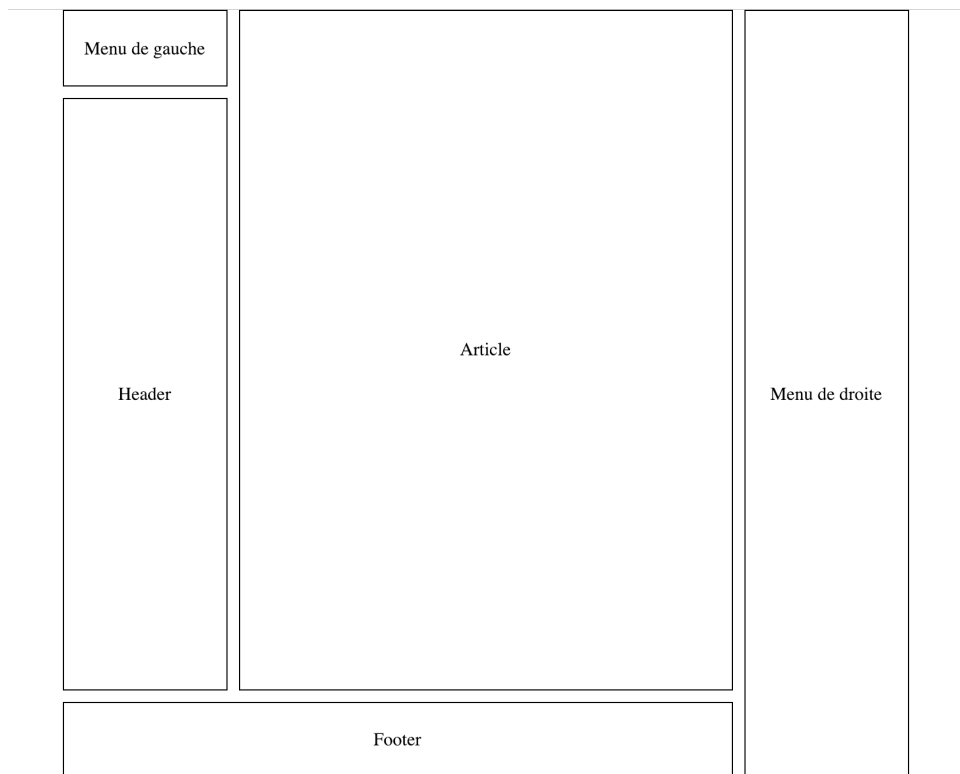
Figure 3.76. CSS Grid exercice 1 et 2

5.2.3. Exercice 2

Le design à obtenir est identique à celui de l'exercice 1, mais cette fois-ci vous utiliserez `grid-template-areas` (déjà fournie). Vous utiliserez `exercice2.css` et `exercice2.html`.

5.2.4. Exercice 3

Pour finir vous utiliserez les nommages des lignes et des colonnes. Les fichiers à utiliser sont `exercice3.css` et `exercice3.html`.

Figure 3.77. CSS Grid exercice 3

5.2.5. Exercice 4

Reproduire le formulaire suivant avec la solution de votre choix. Vous utiliserez `exercice4.css` et `exercice4.html`.

Figure 3.78. CSS Grid exercice 1 et 2

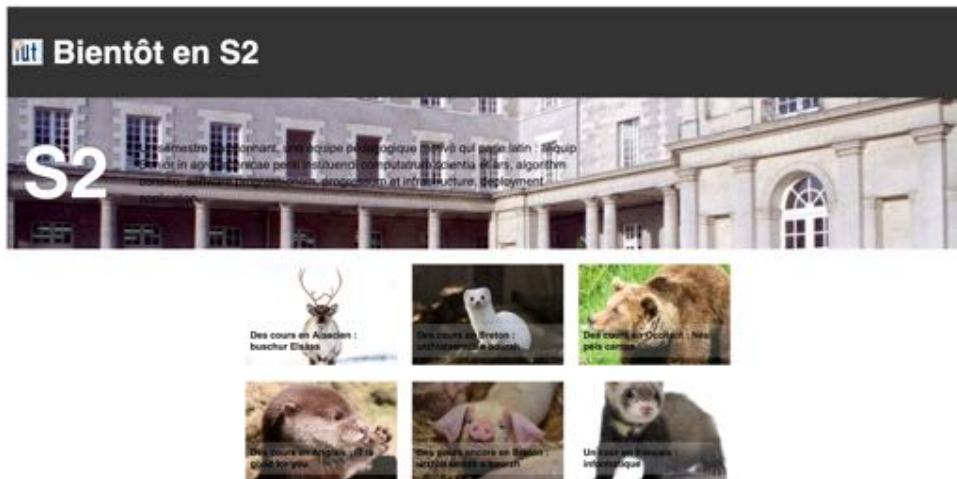
The form layout is displayed on a blue background. It contains the following elements:

- Two input fields: 'Entrer votre nom' and 'Entrer votre prenom'.
- One input field: 'Entrer votre email'.
- Two input fields: 'Entrer un mot de passe' and 'Confirmer le mot de pas'.
- One button: 'Créer'.

5.2.6. Exercice 5

Ce petit exercice de synthèse contient du positionnement relatif, absolu, flex et gris. Vous utiliserez `exercice5.css` et `exercice5.html`.

Figure 3.79. CSS Grid exercice 5



5.3. Pour aller plus loin

La CSS 3 a introduit `@support` qui permet de tester si des fonctionnalités sont prises en charge par le navigateur, vous trouverez ici quelques exemples : <https://developer.mozilla.org/fr/docs/Web/CSS/@supports>. Vous utiliserez `exercice6.css` et `exercice6.html`.

```
/* la regle n'est prise en compte que si le navigateur support display: grid */
@supports (display: grid) {
  div {
    display: grid;
  }
}
```

Reproduire le design suivant en utilisant `@supports`, sachant que `border-radius: 25em` et `shape-outside: circle()` sont supportés sur presque tous les navigateurs et que seul mozilla support `background: -moz-element(#css-source)`.

Figure 3.80. @supports chrome



Figure 3.81. @supports mozilla

6. TP 8 Pour aller plus loin : les médias queries et Materialize

//TODO