

HTML



CSS



Sass

Intégration web

ESGI 3IW 2023-2024
Edouard Sombié

Introduction

L'objet de ce cours est de passer en revue les bonnes pratiques de l'intégration web ainsi que les outils mis à notre disposition.

Le cours sera articulé selon les grandes parties suivantes :

- Le balisage / HTML 5
- La mise en forme / CSS3
- La compilation de feuilles de style / Sass
- Créer un framework css 🤘

Pour prolonger le cours, vous pouvez aller jeter un œil aux sites de références :

<https://developer.mozilla.org/fr/docs/Web/HTML>

<https://developer.mozilla.org/fr/docs/Web/CSS>

<https://sass-lang.com/>

HTML5 - rappels

Le langage html

- signifie Hyper Text Markup Language
- est le langage de balisage standard pour la création de pages Web
- décrit la structure d'une page Web
- se compose d'une série d'éléments (balises)
- Les éléments HTML indiquent au navigateur comment afficher le contenu
- Les éléments HTML étiquettent des éléments de contenu tels que "ceci est un titre", "ceci est un paragraphe", "ceci est un lien", etc.
- date de 1991

HTML5 - rappels

- **Les balises**

`<tagname>` Le contenu va ici ... `</tagname>`

`<h1>` Mon premier cap `</h1>`

Certaines balises sont vides par définition...

`
`

- **Les attributs**

`<tagname attr="value">... </tagname >`

<https://developer.mozilla.org/fr/docs/Web/HTML>

HTML5 - rappels

- La structure de base

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

Elles servent à quoi ces meta ?

HTML5 - rappels

- **Les différents types d'images**
 - **JPEG** 16 millions de couleurs (compressé)
 - **PNG** 16 millions plus transparence
 - **GIF** 256 couleurs, peut être animé
 - **SVG** vectoriel
 - **APNG** comme le png mais animé
 - **WEBP** comme le png ou jpg mais compression plus efficace
- | | |
|--|---|
| | Pas encore dans
les standards du W3C |
|--|---|

<https://caniuse.com/>

HTML5 - rappels

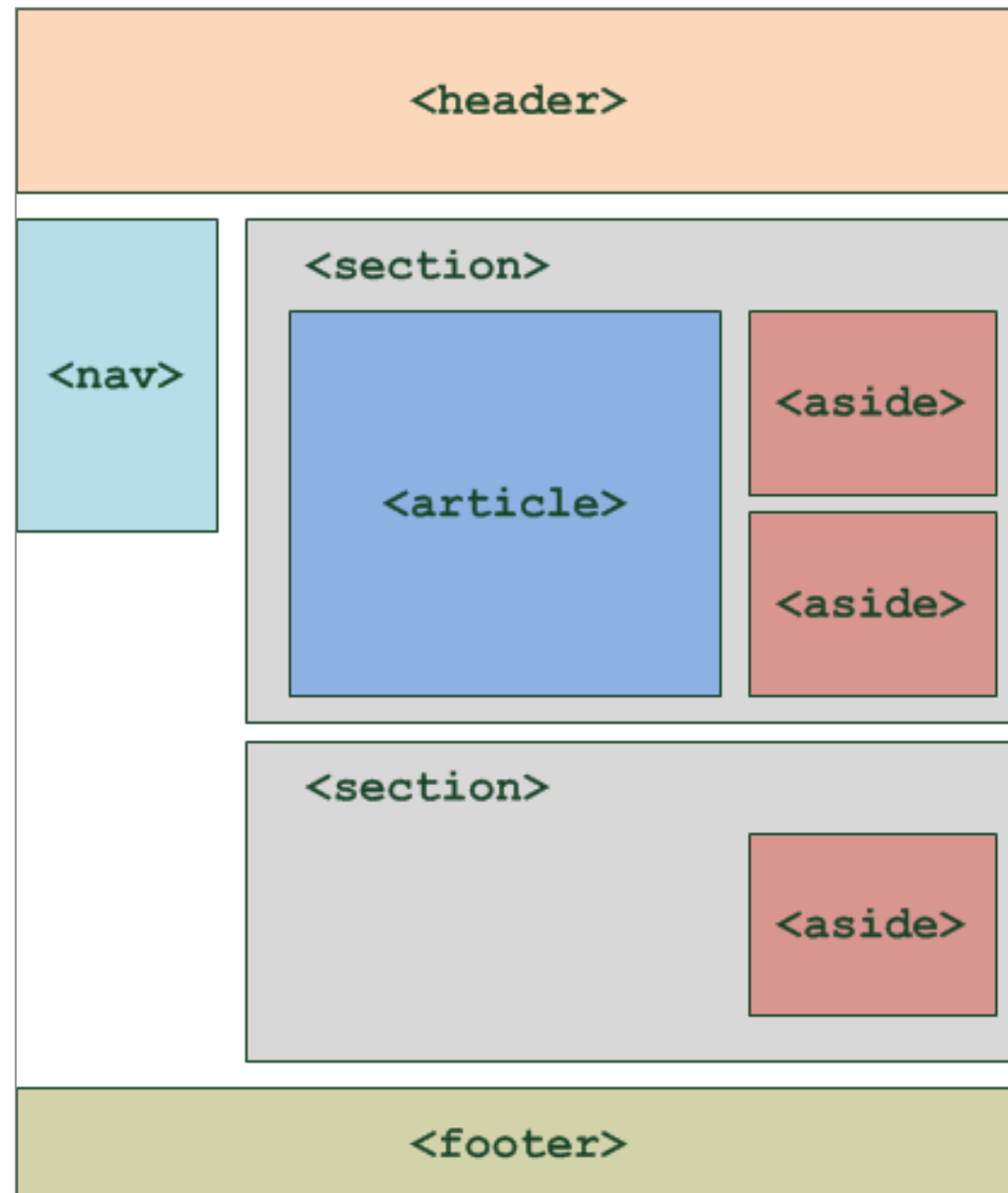
- **Les principales balises structurantes du html5**

<code><header></code>	Entête
<code><main></code>	Contenu principal
<code><footer></code>	Pied de page
<code><nav></code>	Principaux liens de navigation
<code><section></code>	Section dans un document

- **Les autres balises html5 importantes**

<code><article></code>	Un article dans un document
<code><aside></code>	Le contenu annexe de la page
<code><figure></code>	Contenu autonome (image et légende)
<code><figcaption></code>	Légende pour un élément <code><figure></code>
<code><time></code>	Date ou heure

HTML5 - rappels



Quelle balise manque à cette page ?

HTML5 - rappels

Markup et SEO

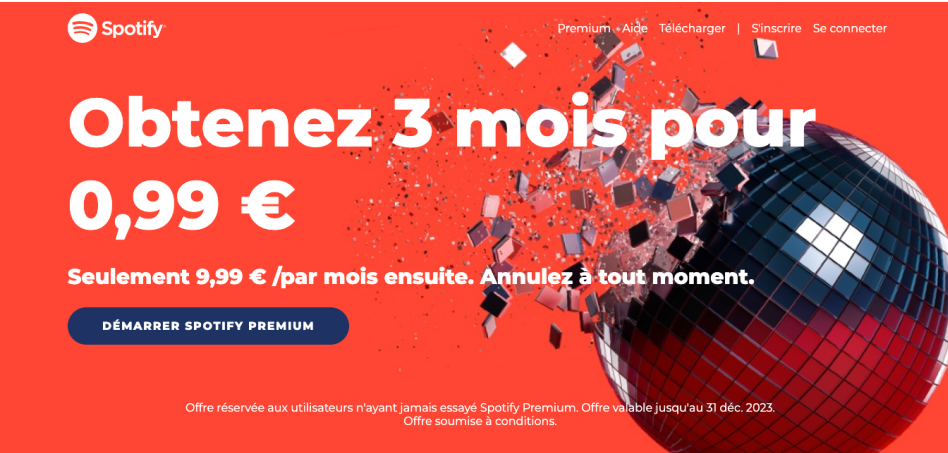
La structuration du contenu selon des balises « signifiantes » est considérée comme une bonne pratique. Notamment pour le référencement par les moteurs de recherche. En plus de ces considérations de SEO, une telle structuration est encouragée pour rendre le contenu facilement accessible aux clients « machines » présents et à venir.

Les balises structurantes du html5 doivent être utilisées au détriment des balises <div>.


A ces balises, il faut ajouter les balises classiques , et <h1> qui jouent un rôle important dans l'indexation d'une page.

Exercice 1

Structure html à partir d'un template graphique.




Pourquoi passer à Spotify premium ?




Téléchargez votre musique.

Profitez-en même sans connexion internet.




Écoutez sans pubs.

Profitez de vos titres sans interruption.



Écoutez les titres de votre choix.

Même sur votre mobile.



Zapping à l'infini.

Cliquez simplement sur suivant.

Écoutez gratuitement ou abonnez-vous à Spotify Premium.

Spotify Free

0,00 € / mois

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zappez les titres sans limite
- ✓ Écouter hors connexion
- ✓ Écoutez les titres de votre choix
- ✓ Son de qualité supérieure

DÉMARRER

Spotify Premium

3 mois pour 0,99 €

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zappez les titres sans limite
- ✓ Écouter hors connexion
- ✓ Écoutez les titres de votre choix
- ✓ Son de qualité supérieure

DÉMARRER SPOTIFY PREMIUM

Seulement 9,99 € /par mois ensuite. Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2023. Offre soumise à conditions.

[Légal](#) [Cookies](#) [À propos des pubs](#)

© 2023 Spotify AB

- [Premium](#)
- [Aide](#)
- [Télécharger](#)
- [S'inscrire](#)
- [Se connecter](#)

Obtenez 3 mois pour 0,99 €

Seulement 9,99 € /par mois ensuite. Annulez à tout moment.

[Démarrer Spotify Premium](#)

Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2023. Offre soumise à conditions.

Pourquoi passer à Spotify premium ?



Téléchargez votre musique.

Profitez-en même sans connexion internet.



Écoutez sans pubs.

Profitez de vos titres sans interruption.



Écoutez les titres de votre choix.

Même sur votre mobile.



Zapping à l'infini.

Cliquez simplement sur suivant.

Écoutez gratuitement ou abonnez-vous à Spotify Premium.

Spotify Free

0,00 € / mois

- Lecture aléatoire
- Sans interruptions
- Zappez les titres sans limite
- Écouter hors connexion
- Écoutez les titres de votre choix
- Son de qualité supérieure

[Démarrer](#)

Spotify Premium

3 mois pour 0,99 €

- Lecture aléatoire
- Sans interruptions
- Zappez les titres sans limite
- Écouter hors connexion
- Écoutez les titres de votre choix
- Son de qualité supérieure

[Démarrer spotify premium](#)

Seulement 9,99 € /par mois ensuite. Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2023. Offre soumise à conditions.

- [Légal](#)
- [Cookies](#)
- [À propos des pubs](#)

© 2023 Spotify AB

CSS3 - rappels

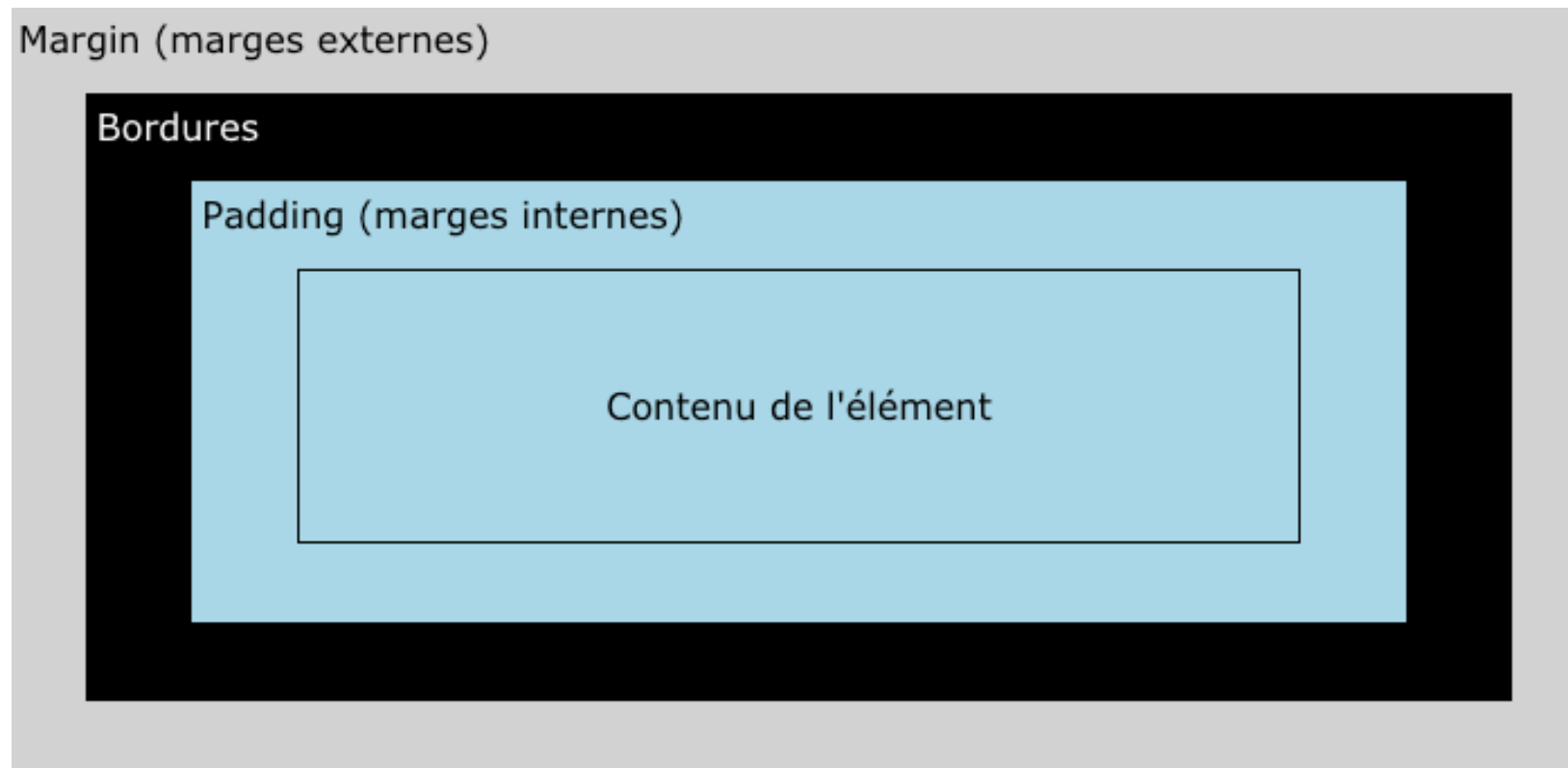
Dimensionnement

On distingue en html 2 grands types de balises, celles qui se présentent sous forme de rectangle (**block**) et les autres qui se présentent en ligne (**inline**)

Balises de type *block* : h1, main, p, table...

Balises de type *inline* : a, em, span...

On ne peut dimensionner que les balises de type *block*. De même, cela n'a pas de sens d'ajouter une marge interne ou externe à un élément *inline*.



box-sizing

content-box (valeur par défaut)

border-box (le plus utile) 🔥

margin : 0 auto

Centre horizontalement

CSS3 - rappels

Positionnement

Le positionnement des éléments est un des principaux enjeux de l'intégration web. Pour atteindre nos objectifs, nous avons 3 approches différentes qui peuvent être complémentaires.

- Le positionnement flottant : **float**
- Le type d'affichage : **display**
- Le type de position : **position**

CSS3 - rappels / Positionnement

float

none (défaut), right, left

- Si différent de none, élément retiré du flux normal, se place à droite ou à gauche de son conteneur. Demeure tout de même dans le flux.
- Implique l'utilisation d'une disposition en bloc : change la valeur du display.
- Sa hauteur n'est pas prise en compte par son parent 😓
- Nécessite un **clearfix** 🙏 qui repose sur la propriété **clear**

Note :

Ce type de positionnement est de moins en moins fréquent et peut (à juste titre) être considéré comme désuet. Toutefois, il est important d'en comprendre le fonctionnement et de savoir comment prendre en compte des éléments flottants. Il arrive, dans de rares cas, d'avoir à faire flotter un bloc 😅

CSS3 - rappels / Positionnement

display

block, inline...

- none : élément pas affiché
- inline-block : en ligne mais forme un rectangle (ne va pas à la ligne)
- **flex** : se comporte comme un conteneur flexible
- list-item, table, **grid**...

On reparlera des flexboxes un peu plus tard... 🐸

CSS3 - rappels / Positionnement

position

- **static** (défaut) : élément positionné dans le flux avec sa position par défaut
- **relative** : positionné dans le flux, mais peut être décalé avec les propriétés top, right, bottom, left
- **absolute** : sorti du flux, positionné avec top, right, bottom, left, par rapport à son plus proche ancêtre positionné.
- **fixed** : sorti du flux, positionné avec top, right, bottom, left, par rapport à la fenêtre (ne bouge pas lors du scroll).
- **sticky** : positionné dans le flux mais sorti du flux et positionné de façon fixe par rapport au parent ("collé" en haut).

CSS3 - rappels

les unités

- Les unités absolues

px, pt, mm, in...

Dans la plupart des cas, elles sont **à éviter** !

Elles nous serviront uniquement à évaluer la taille du viewport ou à la création de styles pour l'impression.

- Les unités relatives

%, em, **rem**, vw, vh, vmin, vmax...

Les unités relatives sont à privilégier car elles permettent une intégration simple et facilement modifiable. De plus, elles permettent d'avoir une interface en adéquation avec les réglages du navigateur des visiteurs.

```
font-size: 1rem;
```


CSS3 - rappels

les sélecteurs et les combinateurs

On est bien d'accord sur le fait qu'on connaît ces sélecteurs ?

.selector1

#selector1

[attr]

selector1 selector2

selector1 > selector2

selector1 + selector2

selector1 ~ selector2

Si ce n'est pas le cas, c'est le moment de se mettre à jour...

CSS3 - rappels

les pseudo classes

Permettent de sélectionner un élément dans un état particulier.

Elles commencent par :

:hover, :checked, :active
:first-child, :last-child
:nth-child(n), :not(selector)...

les pseudo éléments

Permettent d'ajouter ou de sélectionner un élément à l'intérieur d'un autre.

Ils commencent par ::

::after, ::before
::first-letter, ::first-line
::selection

```
*::after{  
    content:"";  
}
```

Note : sans valeur de **content**,
le pseudo élément ::before ou ::after n'est pas créé !

CSS3 - rappels

les polices de caractères

Il existent deux formats officiels pour les polices de caractères sur le web.

On n'utilisera que celles-ci !

.woff et **.woff2**

Beaucoup de projets reposent sur des polices Google fonts.

Dans ce cas, on les utilisera directement à partir du CDN Google avec l'instruction

@import

Pour les projets nécessitant une police commerciale, il faudra la convertir du format d'origine (.ttf, .otf ...) vers un format web.

De nombreux outils de conversion en ligne et gratuits existent, retenons celui-ci :

<https://transfonter.org/>

CSS3 - rappels

Quelques règles css

En principe, vous connaissez déjà les principales règles css, mais connaissez-vous celles-ci ?

scroll-behavior, scroll-margin-top

transition, animation

transform

object-fit, object-position

aspect-ratio

align-items, justify-content

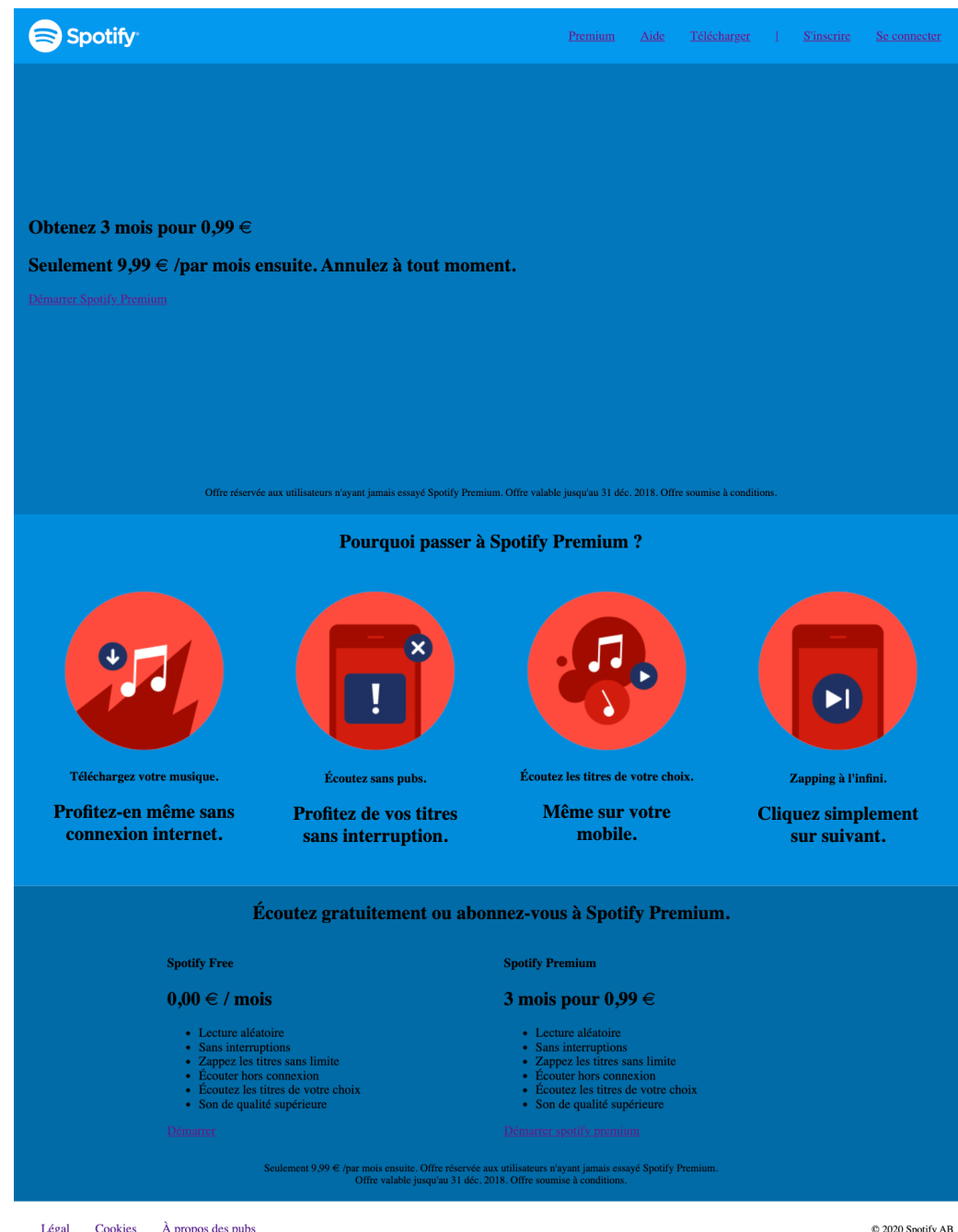
Des mot-clés à connaître et à utiliser

inherit, initial, unset

currentColor

Exercice 2

Positionner tous les éléments du TP1 avec des flottants !! 🥵



Ça va être un peu galère, mais on va voir des pratiques très importantes 🧐 :

- Utiliser un **container**
- Contrôler le padding avec le **box-sizing**
- Contrôler les flottants avec un **clearfix**
- Contrôler le **margin-collapsing**
- Contrôler le **positionnement vertical**
- Utiliser **calc()** et **nth-child()**

CSS3

le positionnement flex

On oublie les flottants et on revient en 2023 !

C'est clairement la meilleure manière d'agencer des éléments dans une mise en page. Elle règle un certain nombre de problèmes liés au margin-collapsing et à l'alignement vertical et offre des actions jusque là impossibles sans javascript.

Pour faire simple, le positionnement flex va nous permettre de :

- Dimensionner les blocs enfants indépendamment de leur contenu.
- Les présenter horizontalement (row) ou verticalement (column).
- Simplifier l'alignement des éléments (axe vertical).

display : flex;

La seule alternative crédible au positionnement *flex* est le *grid system* : plus puissant mais aussi plus complexe à mettre en œuvre...

CSS3 - le positionnement flex

les propriétés de base d'une flexbox

- flex-direction
- flex-wrap
- justify-content
- align-items

- align-content
- flex-flow (flex-direction, flex-wrap)

les propriétés de base d'un item flex

- order
- align-self
- flex (flex-grow flex-shrink flex-basis) - valeur par défaut : 0 1 auto

CSS3 - le positionnement flex

Mettons cela en pratique avec un petit exercice en ligne :

<https://flexboxfroggy.com/#fr>

FLEXBOX FROGGY

Niveau 1 de 24

Bienvenue à Flexbox Froggy, un jeu où vous aidez Froggy la grenouille et ses amis en écrivant du code CSS! Guidez cette grenouille au nénuphar à la droite de l'étang en utilisant la propriété `justify-content`, qui aligne les éléments horizontalement et accepte les valeurs suivantes :

- `flex-start` : Les éléments s'alignent au côté gauche du conteneur.
- `flex-end` : Les éléments s'alignent au côté droit du conteneur.
- `center` : Les éléments s'alignent au centre du conteneur.
- `space-between` : Les éléments s'affichent avec un espace égal entre eux.
- `space-around` : Les éléments s'affichent avec un espacement égal à l'entour d'eux.



Par exemple, `justify-content: flex-end;` bougera la grenouille vers la droite.

```
1 #pond {
2   display: flex;
3
4 }
5
6
7
8
9
10
```

Suivant

Flexbox Froggy est créé par [Codepip](#) • [GitHub](#) • [Twitter](#) • [Paramètres](#)

Vous voulez apprendre CSS Grid ? Jouez à [Grid Garden](#).



Exercice 3

Positionner tous les éléments du TP1 avec des flexboxes !! 😎



Beaucoup plus simple et robuste que dans l'exercice précédent...

- flex-direction
- flex-wrap
- flex-direction
- justify-content
- align-items

- flex-grow
- flex-shrink
- flex-basis
- order