# Support de cours CSS3 - 3

Jérôme AMBROISE

Next Formation
Semaine 27/02/2017 - 01/03/2017

# 1.

# Définir des colonnes de textes

# Définir des colonnes

## Présentation

La mise en page multi-colonnes CSS3 permet de définir facilement de multiples colonnes de texte.

Propriété : column-count

Valeur : nombre de colonnes

**Exemple :**

```
#column-list-section ul
{
   -webkit-column-count: 3; /* Chrome, Safari, Opera */
   -moz-column-count: 3; /* Firefox */
   column-count: 3;
}
```

# Définir des colonnes

## Présentation

Il est possible de définir des espaces entre chaque colonne.

Propriété : column-gap

Valeur : unité de mesure

**Exemple :**

```
#column-list-section ul
{
   -webkit-column-count: 3; /* Chrome, Safari, Opera */
   -moz-column-count: 3; /* Firefox */
   column-count: 3;
   -webkit-column-gap: 3rem;
   -moz-column-gap: 3rem;
   column-gap: 3rem;
}
```

# Définir des colonnes
## Présentation

Nous pouvons définir les règles des bordures entre chaque colonne comme nous définirons une bordure.

Propriété : column-rule

Valeur : (unité de mesure) (solid/double/inset/dashed/…) (couleur)

**Exemple :**

```
#column-p-section p
{
    …
     -webkit-column-rule: 1px dashed SlateGrey; /* Chrome, Safari, Opera */
    -moz-column-rule: 1px dashed SlateGrey; /* Firefox */
    column-rule: 1px dashed SlateGrey;
}
```

# Définir des colonnes
## Présentation

Pour créer une exception dans le système de colonnes : un élément doit prendre toute la largeur disponible, nous utiliserons “column-span” :

Propriété : column-rule

Valeur : all

**Exemple :**

```
#column-p-section h2 {
   -webkit-column-span: all; /* Chrome, Safari, Opera */
   column-span: all;
}
```

Remarque : “column-span” n’est pas supporté par Firefox

# 2.

# Tricks

# Tricks : Box sizing

La propriété “box-sizing” permet d’ éliminer un inconvénient du modèle de boîte en prenant en compte les marges intérieures et l’épaisseur de la bordure comme étant la largeur de l’élément.

# Tricks : Box sizing

Propriété : box-sizing
Valeur : border-box

- ➔ Content-box : c'est la valeur par défaut, elle spécifie que les largeur et hauteur définies s'appliquent aux largeur et hauteur du contenu seul, et que le padding, la bordure et les marges sont en dehors de ces dimensions.
- ➔ Border-box : les largeur et hauteur spécifiées sont celles de l'élément dans son ensemble, en conséquence les dimensions du padding ou des bordures viennent diminuer les dimensions du contenu.
- ➔ Inherit : comme pour toute valeur héritée, la valeur de box-sizing peut dépendre de celle de son élément parent.

# Tricks : Box sizing

Une des recommandations pour utiliser “box-sizing” et de l’appliquer à tous les éléments de la page HTML.

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

# Tricks : calc(), attr ()

➔ La fonction calc() permet d'effectuer des calculs :

```
Section > article {
    width : (100% - 15rem);
}
```

➔ La fonction attr () permet de retourner un attribute HTML de l'élément

```
a:after { content: " de source (" attr (href,  'inconnue') ")"; }
```

# Exploiter le modèle de boite

Pour les listes à puces de menu, les boutons, les liens : inline-block, dimensions, fond en couleur, pudding, border.

Idée :
Border-bottom sur les puces de menu (changement de couleur de bordure au survol)
Image : transition de taille au survol (100%)

# Enregistrer avec l'inspecteur

# 3.

# Flexbox

Géolocalisation

Drag & Drop

Stockage local

Cache d’application

WebWorker

# Flexbox

## Présentation

Les boîtes flexibles, ou flexbox, est un nouveau mode de mise en page dans CSS3.

L'utilisation de flexbox assure que les éléments se comportent de manière prévisible lorsque la mise en page doit s'adapter à différentes tailles d'écran et à différents dispositifs d'affichage.

C’est un modèle récent et à préférer sur tous les nouveaux projets autant que possible.

# Flexbox

## Mise en place simple

Le modèle “flexbox” se met en place sur le conteneur des items que l’on souhaite disposer.

Par défaut, les éléments se disposent sur une seule ligne.

# Flexbox

## Mise en place simple

### Exemple : Liste à puce basique (avec et sans flexbox)

```
#flexbox-basic ul, #no-flexbox ul
{
        width:50%;
        background-color:PapayaWhip ;
}
#flexbox-basic li, #no-flexbox li
{
        width:10rem;
        height:5rem;
        background-color:PaleGoldenRod ;
}

#flexbox-basic ul
{
        display: -webkit-flex;
   display: flex;
}
```

# Flexbox

## Changement de sens

Pour changer l'alignement par défaut d'un conteneur flexbox, il faut modifier sa direction.
Propriété : direction
Valeur : rtl (right to left)/ ltr (left to right)

**Exemple :**

```
#flexbox-basic ul
{
        display: -webkit-flex;
        display: flex;
        direction: rtl;
}
```

# Flexbox

## Alignement axe principal

Par défaut, les éléments d'un conteneur flexbox s'aligne dans l'ordre sur une ligne, nous pouvons changer ce comportement.
Propriété : flex-direction
Valeur : row (par défaut), row-reverse, column, column-reverse
**Exemple :**

```
#flexbox-column-reverse ul
{
        display: -webkit-flex;
        display: flex;
        flex-direction: column-reverse;
}
```

Remarque : "reverse" inverse l'ordre d'apparition des éléments

# Flexbox

## Alignement axe principal

Lorsque les éléments d'un conteneur flexbox n'occupe pas tout l'espace du conteneur, différentes propriétés permettent de les aligner.

Propriété : justify-content

Valeurs : flex-start, flex-end, center, space-between, space-around

**Exemple :**

```
#flexbox-justify-spacearound ul
{
        display: -webkit-flex;
        display: flex;
        -webkit-justify-content: space-around;
        justify-content: space-around;
}
```

# Flexbox

## Justification des éléments - Distinctions

Comment faire la distinction entre “space-around” et “space-between” ?

- ➔ “space-between” : Le premier et le dernier élément sont placés aux extrémités du conteneur. L’espace alors inoccupé entre (between) les éléments est alors réparti de façon égal.

- ➔ “space-around” : L’espace inoccupé par les éléments est réparti de façon égal autour d’eux.

# Flexbox

## Alignement axe secondaire

Si les éléments sont organisés en ligne (row, row-reverse) et qu'il n'occupent pas toute la hauteur, il y a différents façons de les organiser.

Propriété : align-items
Valeurs : stretch (tout l'espace), flex-start, flex-end, center, baseline

Remarque : il en va de même si les éléments sont organisés en colonnes et qu'ils n'occupent pas tout la largeur.

# Flexbox

## Alignement axe secondaire

**Exemple :**

```
.flexbox2-same ul
{
   width:50%;
   height: 30rem;
   background-color:PapayaWhip ;
   padding-left: 0;
   list-style-type: none;
}
.flexbox2-same li
{
   width:10rem;
   background-color:PaleGoldenRod ;
   margin: 1rem;

}
```

```
#flexbox-align-stretch ul
{
        display: -webkit-flex;
        display: flex;
        -webkit-align-items: stretch;
        align-items: stretch;
}
```

# Flexbox

## Comportement forcé sur une ligne

Par défaut, les éléments d'un conteneur "flexbox" s'organisent sur une seule ligne malgré les dimensions définies.

**Exemple :**

- ➔ Une liste à puce fait 700px de large
- ➔ La liste à puce possède 4 puces
- ➔ Chaque puce s'est vue attribuée 200px de large

Lorsqu'on attribut "dispay:flex;" à la liste à puce, chaque puce tient sur une seule ligne…

# Flexbox

## Retour à la ligne

Pour gérer plus finement le retour à la ligne, il existe une propriété.
Propriété : flex-wrap
Valeur : nowrap(par défaut), wrap, wrap-reverse
**Exemple :**

```
#flexbox-example-wrap ul
{
	display: -webkit-flex;
	display: flex;
	 -webkit-flex-wrap: wrap;
	flex-wrap: wrap;
}
```

# Flexbox

## Ordre des éléments flex

Par défaut chaque élément flex est affiché dans l'ordre où il a été écrit en HTML (ou en ordre "inverse).
Il est possible de changer l'ordre des éléments grâce à  la propriété "order".

➔ Par défaut tous les éléments ont un "order" à 0, ce qui signifie qu'ils occupent leur place sans poids particulier.

➔ Il est possible d'ajouter ou de retirer des poids sur les éléments pour changer leur ordre d'apparition.

# Flexbox

## Ordre des éléments flex

Propriété : order
Valeur : (nombre entier positif ou négatif)
**Exemple :**

```
#flexbox-order ul > li:nth-child(1)
{
	-webkit-order: 1;
	order: 1; /* poids +1 */
}
#flexbox-order ul > li:nth-child(4)
{
	-webkit-order: -1;
	order: -1; /* poids -1 */
}
```

# Flexbox

## Flexibilité

Il arrive de devoir définir son modèle de boîte de manière complexe (largeur, hauteur, marges, … ).
Le modèle “flexbox” permet de définir des dimensions pour vous.

La philosophie est de diviser l’espace restant dans le contenant (ayant une largeur définie) entre les différents éléments.

Chaque élément possède “des parts” de l’espace restant, plus il a de parts, plus il est large.

# Flexbox

## Flexibilité

Propriété : flex (sur l'élément et non sur le contenant)

Valeur : (nombre entier positif)

**Exemple :**

```
#flexbox-flex ul {
        width:80rem;
        background-color:PapayaWhip ;
        padding-left: 0;
        list-style-type: none;
        display: -webkit-flex;
        display: flex;
}
#flexbox-flex li {
        background-color:PaleGoldenRod ;
        margin:0.5rem;
        height:5rem;
}
```

```
/* 1 part sur 8 */
#flexbox-flex li:nth-child(1){flex: 1;}

/* 2 parts sur 8 */
#flexbox-flex li:nth-child(2){flex: 2;}

/* 3 parts sur 8 */
#flexbox-flex li:nth-child(3){flex: 3;}

/* 4 parts sur 8 */
#flexbox-flex li:nth-child(4){flex: 4;}
```

# Flexbox

## Flexibilité

La propriété “flex” est enfait une super-propriété :

[ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

- ➔ Lorsque nous utilisons “flex:3;” cela nous permet d’agrandir notre élément pour posséder 3 “parts” sur l’espace restant
- ➔ “flex-shrink” : permet de réduire la taille de l’élément (pour effectuer des rectifications en cas de décallage)
- ➔ “flex-basis” : attribue une taille à l’élément **AVANT** de redistribuer l’espace restant (permet d’imposer des valeurs pour des éléments par rapport aux autres)

# Flexbox

## Flexibilité

Cas particulier :

La propriété flex-basis est une sous-propriété de “flex”. Elle spécifie la dimension initiale de l'élément flex, avant qu'un éventuel espace disponible soit distribué selon les facteurs flex.

- ➔ Lorsqu'elle est omise dans le raccourci flex, elle a pour valeur par défaut 0
- ➔ Une valeur de flex-basis à “auto” dimensionne l'élément selon sa propriété de dimension
- ➔ Une valeur à 100% force l’élément à posséder la totalité de la largeur de l’écran

# 4.

# Media queries

Géolocalisation

Drag & Drop

Stockage local

Cache d'application

WebWorker

# Media queries

## Présentation

Les media queries permettent de présenter une mise en forme différentes selon la largeur d'écran du périphérique qui affiche la page.

- ➔ L'intégration des media queries se fait via la directive "@media"
- ➔ La philosophie des media queries est "mobile first" : la définition des médias va de la plus petite largeur à la plus grande

# Media queries

## Présentation

**Exemple :**

```
/* Définition pour les téléphones par défaut */

/* Petits périphériques (tablettes, 768px et plus) */
@media (min-width: 768px) { ... }

/* Périphériques moyens (ordinateurs de bureau, 992px et plus) */
@media (min-width: 992px) { ... }

/* Grands périphériques (grands ordinateurs de bureau, 1200px et plus) */
@media (min-width: 1200px) { ... }
```

# 5.

# Créons notre propre système de grille façon Bootstrap

Jérôme AMBROISE
Next Formation
CSS3
Semaine 20/02/2017 - 24/02/2017

# 6.

# Créons notre propre layout flexbox

Jérôme AMBROISE
Next Formation
CSS3
Semaine 20/02/2017 - 24/02/2017

Merci de votre attention !

=)