CSS

OBJECTIFS DU COURS

Pour les CSS :

- Présenter les concepts généraux
- Expliquer quelques points délicats
- Donner un aperçu des possibilités
- Ne pas faire de catalogue : vous compléterez selon vos besoins

Pour Bootstrap :

- Présenter le principe de fonctionnement
- Donner de quoi débuter si vous souhaiter l'employer

INTRODUCTION

INTÉGRATION WEB

Une parenthèse, métier d'intégrateur web :

- Un graphiste prépare une maquette, en concertation avec le client
- Un informaticien (intégrateur) se charge de la transposer en HTML et CSS

Votre formation : éventuellement développeur web

- Pas intégrateur « à plein temps »
- Mais possible nécessité de créer des mises en pages propres rapidement (produit final ou prototype)

CSS

- **C**ascading **S**tyle **S**heets : feuilles de style en cascade
- En cascade : si une règle apparaît plusieurs fois chaque nouvelle occurrence complète ou modifie la précédente.
- Langage de description de styles pour les éléments HTML : mise en forme
- Définition de feuilles de styles, qui peuvent s'appliquer à un nombre quelconque de documents HTML.

- Séparation structuration / mise en forme :

Oui, mais la mise en forme s'appuie sur la structure, il faut un code HTML propre et conçu pour permettre la mise en page souhaitée.

ECRITURE DE CSS
QUELQUES BASES

CSS

- Version actuelle (septembre 2019) : CSS 3
- Permet de définir des couleurs, des polices de caractères, des animations (transitions)…
- Peut se définir :
  1. Dans des fichiers .css, importés dans les fichiers HTML
     importation (ex.) : `<link rel="stylesheet" href="css/jolisStyles.css">`
  2. Dans une balise <style>, située dans la section <head> du document
  3. Directement dans une balise, avec l'attribut style

SYNTAXE GÉNÉRALE

- La syntaxe de base d'une règle CSS est :

```
monSelecteur
{
     propriete1 : valeur;
     propriete2 : valeur;
}
```

SELECTEURS SIMPLES

- Un sélecteur permet de désigner un ou plusieurs éléments du document HTML, pour appliquer les styles.

- Pour désigner une balise HTML : juste son nom

```
body
{
    propriete1 : valeur;
    propriete2 : valeur;
}
```


SELECTEURS SIMPLES

- Pour désigner un élément par son attribut id, son identifiant : # devant le nom

```
#monId /*Sera appliqué à l'élément qui a pour attribut : id="monId"  */
{
    propriete1 : valeur;
    propriete2 : valeur;
}
```

Pour désigner un élément par une valeur de son attribut class : point devant le nom

```
.maClasse
{
  background-color : yellow; /* une propriété */
}
```

Remarque : les commentaires CSS s'écrivent entre /* */ uniquement. Pas de //.

SYNTAXE DES PROPRIÉTÉS

```
.maClasse
{
      background-color : yellow; /* une propriété */
}
```

- Propriété : `nomAttribut: valeur;`
- Une règle CSS : énumération de propriétés

SYNTAXE DES PROPRIÉTÉS

Exemples :

```
.encadrementV1
{
      border-width: 2px;          /* unité obligatoire */
      border-style: dashed;
      border-color: red;
}
```

Peut se raccourcir (« shorthand property ») :

```
.encadrementV2
{
      border: 2px dashed red;     /* propriété raccourcie*/
}
```

SYNTAXE DES PROPRIÉTÉS

Exemples :

```
.fondRougeJG
{
    background-color: #FF0000;
}
```

D'autres notations existent pour les couleurs, par exemple `rgb(255,0,0)`

https://www.w3schools.com/css/css3_colors.asp

ECRITURE DE CSS
QUELQUES PROPRIÉTÉS

COULEURS, TEXTES, ARRIÈRE PLAN

Nom de la propriété	Rôle	Valeurs et syntaxes(non exhaustif)

color	Couleur des caractères	Rouge, vert, bleu (+alpha) :
		color : #F10023;
		color : rgb(10,12,255);
		color : rgba(10,12,255, 0.5);
		Nom prédéfini :color : blue;

text-align	Alignement horizontal	center, right, left, justify
text-decoration		none underline : souligné line-trough : barré overline : ligne au-dessus
text-transform	Contrôle la casse	none : pas de modification capitalize : première lettre en majuscule uppercase : tout en majuscules lowercase : tout en minuscules
vertical-align	Alignement vertical	top, middle, bottom

COULEURS, TEXTES, ARRIÈRE PLAN

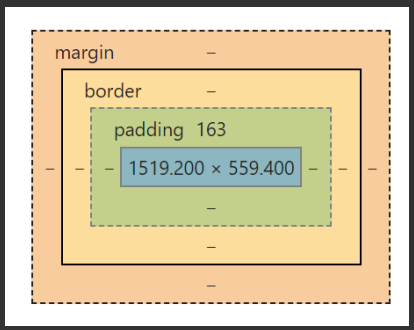
font-family	Choix de la police de caractère, avec des alternatives.	font-family :Verdana, sans-serif ; Nom de police précis ou nom générique : si plusieurs valeurs, le navigateur applique la première qu'il peut produire
font-size	Taille de caractères	10px : sans espace avant px (idem pour les autres unités : cm, em, pc…) 120% : par rapport à ce que serait la taille non modifiée
font-weight	Contrôle de « l'épaisseur » du trait des caractères (« graisse »)	bold, normal, semi-bold valeur numérique : 100 à 900, 400 signifiant normal et 700 gras (bold)
font-style	Italique ou non	italic, normal
font-variant	Par exemple pour un effet de « petites majuscules »	normal, small-caps

COULEURS, TEXTES, ARRIÈRE PLAN

background-color	Couleur de fond	Même principe que « color » (couleur de caractères, précédemment dans la liste)
background-image	Image de fond	background-image : url('repertoireImage/uneImage.jpg') Le chemin est relatif à celui du fichier qui contient la définition css.
background-repeat	Mode de répétition de l'image (si elle est plus petite que son conteneur)	top, middle, bottom
background-position	Alignement vertical	top, middle, bottom

DIMENSIONNEMENT ET MARGE DES BLOCS

Dimensionnement et marges des éléments de type bloc (, <header>, …)



Un bloc dispose :

- D'une marge intérieure, l'espace entre le texte et le bord : padding
- D'une marge extérieure, l'espace entre lui et ce qui l'entoure : margin
- D'une bordure : border
- De dimensions : width, height. Les dimensions, par défaut, n'incluent pas le padding.

Informations visualisables dans les outils de dev des navigateurs

DIMENSIONNEMENT ET MARGE DES BLOCS

- Contrôle sur les 4 côtés : top, right, bottom, left. Propriétés :
  - `margin-top, margin-right, margin-bottom, margin-left`
  - `padding-top, padding-right, padding-bottom, padding-left`

- Valeurs possibles :
  - `auto` : permet par exemple de créer des centrages, en fixant margin-left et margin-right à auto
  - mesure : en pixels, centimètres, etc. L'unité est obligatoire, par exemple `15px`.
  - pourcentage, par exemple `50%` : donné par rapport au bloc parent
  - `inherit` : reprend les mêmes valeurs que son parent

vw

DIMENSIONNEMENT ET MARGE DES BLOCS

Un mot sur les unités :

- Des unités « classiques » : cm, px…
- Des unités particulières :
  - em : 1 em correspond à la taille de la police de caractère de l'élément HTML courant, donc variable selon cette taille
  - rem : comme em, mais en prenant pour référence l'élément racine
  - ex : taille des petites lettres de l'élément HTML courant
  - vw et wh : valent 1 centième de la largeur et de la largeur (resp) de la fenêtre
  - …

https://www.w3.org/Style/Examples/007/units.fr.html#units

ECRITURE DE CSS
QUELQUES CONCEPTS

REDÉFINITION

- Si plusieurs règles avec le même nom, par exemple plusieurs règles pour la balise <body> :
  - Les règles se complètent, se cumulent
  - Si une propriété est définie plusieurs fois, seule la dernière valeur est prise en compte

- *Mais pourquoi aurait-on plusieurs règles qui portent le même nom ?...*
- Peut se produire également si un même élément est ciblé par plusieurs règles de nom différent (balise, id, classes...)
- Styles variant selon les supports : media queries

HÉRITAGE EN CSS

- Héritage : sans rapport avec la programmation oriéntée objet
- Transmission d'une propriété à des éléments descendants dans le DOM.
- Exemple : une police de caractère appliquée à un élément s'applique à ses descendants.

SELECTEURS, COMPLÉMENTS

- Cibler plusieurs sélecteurs avec le même style : séparation par une virgule

```
h1, h2, h6 {
    color : red;
}
```

- Sélecteurs plus avancés :
  - `#contenu p { … }` : les balises p qui descendent de l'id contenu
  - `.formContact .nom { … }` : idem, avec des classes (descendants)
  - `#contenu > p { … }` : les balises p qui sont fils de l'id contenu
  - `input[type="text"] { .. }` : une balise avec un attribut spécifique
  - `h3 + p{ … }` : frères adjacents
  - `h2 ~ .info{ … }` : frères quelconques

SELECTEURS, COMPLÉMENTS

- Pseudo classes : des valeurs prédéfinies préfixées par deux points
- Cas des liens :
  - `a :link    {/* style pour un lien “normal”, non visité*/ }`
  - `a :visited {/* lien déjà visité*/ }`
  - `a :hover   {/* lien sur lequel "le pointeur" se trouve*/ }`
  - `a :active  {/* lien activé (par exemple on est train de cliquer dessus)*/ }`
- Autres exemples (non exhaustif):
  `tr:nth-child(odd) { … };`
  `tr td:nth-child(3n+4) { … }` : cibler des lignes par « modulo de numéro »
- https://www.w3schools.com/css/css_pseudo_classes.asp

SELECTEURS, COMPLÉMENTS

- Pseudo éléments (non exhaustif) :
- ::after et ::before : pour ajouter du contenu après ou avant l’élément ciblé. S’utilise avec la propriété content.

Exemple :

```
details ::before
{
      content : ">>>";
}
```

https://www.w3schools.com/css/css_pseudo_elements.asp

ECRITURE DE CSS
POSITIONNEMENT

POSITIONNEMENT ?

- Objectif : placer un bloc à un endroit précis de la page
- Définition d'une disposition : « layout »
- Nécessite de connaître quelques éléments techniques : beaucoup moins direct et simple que de changer une couleur.

NOTION DE FLUX

- Flux : ici, le flux de « traitement », ou de « positionnement » du navigateur.
- Deux règles de départ, si on s'en tient au comportement par défaut :

  - Les éléments de type bloc s'affichent l'un en dessous de l'autre

  - Les éléments de type inline s'affichent l'un à côté de l'autre, avec des retours à la ligne lorsque c'est nécessaire

POSITION

- Propriété position : définit la façon dont les « coordonnées » d'un bloc sont données.
- Propriétés top, right, bottom, left : donnent la localisation, en pixels par exemple

Par exemple

```
#monBloc
{
        position: relative;
        top: 10px;
        right: 30px;
}
```

POSITION

- Valeurs possibles pour position :
  - static : valeur par défaut
  - relative :
    - le bloc est positionné relativement à la position qu'il aurait dû avoir.
    - Pour les éléments suivants dans le flux, est considéré à sa position d'origine.
  - absolute :
    - le bloc est positionné par rapport au premier de ses ascendants non static.
    - Le bloc est sorti du flux (ignoré pour le positionnement des suivants)

POSITION

- Valeurs possibles pour position :
  - fixed :
    - le bloc est positionné par rapport à la fenêtre de navigation.
    - Le bloc est sorti du flux (ignoré pour le positionnement des suivants)
  - sticky :
    - Lorsqu'il est positionné dans une zone qui peut « scroller », le bloc se « heurte » aux bords
    - Il reste dans le flux, à sa position initiale

Pour tester rapidement les valeur de position :
https://developer.mozilla.org/fr/docs/Web/CSS/position

PROPRIÉTÉ DISPLAY

Display : indique comment un élément est affiché

Valeurs possibles (non exhaustif) :

- inline
- block
- none : ne s'affiche pas.
- inline-block : se comporte comme un élément inline mais peut être dimensionné comme un bloc (marges, largeur, hauteur, etc.)
- flex : un mode de positionnement flexible, détaillé ci-dessous.
- grid : un mode de positionnement en grille, détaillé plus loin.

PROPRIÉTÉ DISPLAY : FLEX

Modèle d'affichage (layout) : « flexbox », boîtes flexibles

Principe général :

- Un conteneur, et des items
- Le conteneur organise la disposition des items, en fonction de propriétés CSS

Exemple

```
.monConteneur
{
    position: flex;
    align-items: center;
}
```

```
.unItem
{
    /* propriétés propres à l'item */
    order : 5;
    flex-grow: 2;
}
```

Un lien, « pour tout comprendre » : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

PROPRIÉTÉ DISPLAY : GRID

Display : grid

Modèle d'affichage (layout) : une grille

Principe général :

- Un conteneur, et des items : comme pour le layout flex
- Le conteneur définit la grille
- Les items indiquent leur position dans la grille

Pour vous documenter :
https://css-tricks.com/snippets/css/complete-guide-grid/#grid-introduction
https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Grid_Layout

ECRITURE DE CSS
MEDIA QUERIES

MEDIA QUERIES

Media queries : indiquer des règles css différentes selon le support d'affichage (le média).

Par exemple :

- Un style pour des écrans larges (ordinateurs)
- Un style pour les écrans de smartphone
- Un style pour l'impression
- …

Utile pour un design adaptatif (« responsive design »)

https://www.w3schools.com/css/css_rwd_mediaqueries.asp

MEDIA QUERIES

Syntaxe :

On indique à quels medias s'adressent certaines propriétés :

```
@media indicationDeLaCible
{
        .uneClasse
        {
                propriété: valeur;
                propriété: valeur;
        }
}
```

MEDIA QUERIES

Syntaxe générale :

```
@media not|only mediatype and (mediafeature and|or|not mediafeature …)
{
  …
}
```

Type de média, optionnel : all, print, screen, speech (media queries 4, liste exhaustive)

Caractéristique de média : elles sont nombreuses, par exemple width, height, orientation…

https://developer.mozilla.org/fr/docs/Web/CSS/Requ%C3%AAtes_m%C3%A9dia/Utiliser_les_Media_queries#types

MEDIA QUERIES

Utilisation classique :

- On définit des styles communs à tous les media, sans media query
- On définit ensuite des blocs media, qui complètent ou modifient les règles en fonction d'un media particulier

Remarque : on utilise souvent display:none; avec les media queries, pour faire disparaître des éléments, par exemple pour un écran plus petit ou pour une impression.

ECRITURE DE CSS
TRANSITIONS, ANIMATIONS, TRANSFORMATIONS

TRANSITIONS

- Animer le changement de valeur d'une propriété (couleur, hauteur, position…)
- Pour en tirer partie il faut une modification « interactive » :
  - CSS : hover
  - JS : modification d'un style CSS

TRANSITIONS

- Syntaxe (avec préfixes), on donne une propriété transition dans la règle :

```
transition: all 4s ease 1s;
```

- 4 paramètres :
  - Propriété (ou all)
  - durée de transition
  - fonction interpolation
  - délai avant démarrage

  Et c'est fait !

https://www.w3schools.com/css/css3_transitions.asp

ANIMATIONS

- Principe : définition d'une série de clés (keyframes)
- Syntaxe :

```
@keyframes uneAnim
{
     0%   {top:0px;}
     25%  {top:200px;}
     50%  {top:100px;}
     75%  {top:200px;}
     100% {top:0px;}
}

#monElementAvecAnim
{
     animation: uneAnim 5s; /* nom de l'animation, durée */
}
```

https://www.w3schools.com/css/css3_animations.asp

TRANSFORMATIONS

- Transformations géométriques, notamment :
  - Rotation : rotate()
  - Translation : translate()
  - Echelle : scale()
- 2D ou 3D
- Exemple :

```
.imageTournee{
   transform: rotate(30deg);
}
```

Ne produit pas directement d'animation, mais peut être utilisé en animation.

ECRITURE DE CSS
CONCLUSION

CONCLUSION

Intégration : CSS + HTML, il faut une structure HTML bien conçue pour pouvoir appliquer les CSS.

Pour vous :

- peut-être plutôt développement d'applications web que création de sites web
- tendance à faire les CSS 'au plus vite' :
  - Utilisation de templates
  - Utilisation de « frameworks » type bootstrap
- Attention aux copier-coller de sources hétérogènes : tout ne peut pas s'assembler, en particulier pour le positionnement ou les media queries.

BOOTSTRAP
UN APERÇU, POUR DÉBUTER

PRESENTATION

- BootStrap : un « framework », c'est-à-dire un ensemble d'outils accompagnés de règles d'utilisation.
- A l'origine : développé pour Twitter
- Contenu : des feuilles de styles CSS , et des composants JavaScript.

- Des liens pour vous documenter :

  https://getbootstrap.com/docs/4.3/getting-started/introduction/

  https://www.w3schools.com/bootstrap4/default.asp

UTILISER LES STYLES

Pour utiliser les styles CSS :

- On importe les CSS, après téléchargement par exemple :

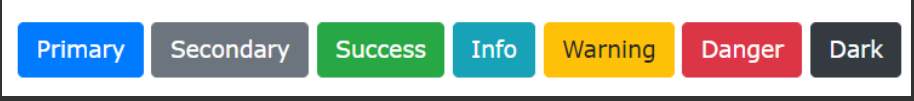
```
<link rel="stylesheet" href="css/bootstrap.min.css">
```

- Puis il suffit d'utiliser les classes qui nous intéressent
- Voir la documentation :

COULEURS

Un jeu de couleurs « sémantiques » est utilisé, avec une notation en suffixe pour chaque classe. Par exemple:

- Texte : text-primary, text-danger, text-success, etc.
- Arrière-plan (background) : bg-primary, bg-danger, bg-success, etc.
- Boutons : btn-primary, btn-danger, btn-success, etc.
- …



Tiré de : https://www.w3schools.com/bootstrap4/bootstrap_buttons.asp

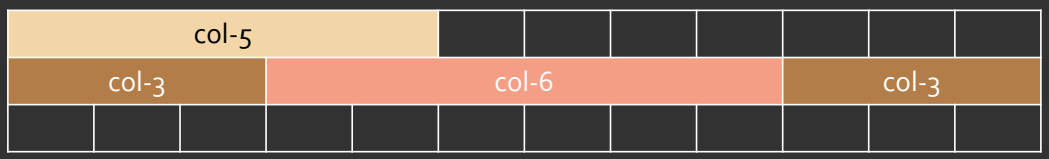
COULEURS

Convention de nommage des classes :

- Pour appliquer un style on utilise une classe de base dont le nom est en rapport avec la nature de l'élément HTML :
  - `<table class="table">`
  - `<button class="btn">`
- Pour préciser le style, on ajoute d'autres propriétés :
  - `<table class="table table-hover table-striped">`
  - `<button class="btn btn-danger">`
- Ne pas omettre la classe de base : Danger au lieu de Danger

MISE EN PAGE (LAYOUT)

- Pour appliquer les classes liées au positionnement, il faut appliquer une classe conteneur : c'est le contenu du conteneur qui est mis en page.
- Classes disponibles : `container` ou `container-fluid` (pleine largeur).
- Positionnement : une grille, disposée sur un « quadrillage » de 12 colonnes.
  - Des lignes : classe `row`
  - Qui contiennent des colonnes : classe `col`.
- On peut préciser la largeur des colonnes : col-5 indique une colonne qui s'étend sur 5 des 12 cases du « quadrillage »



MISE EN PAGE (LAYOUT)

- Documentation officielle pour les grilles : https://getbootstrap.com/docs/4.3/layout/grid/

QUELQUES ÉLÉMENTS UTILES

- Certains de ces éléments utilisent JavaScript, il faut quelques inclusions supplémentaires dans les entêtes : consultez les exemples.

- Formulaires :

https://www.w3schools.com/bootstrap4/bootstrap_forms.asp

https://getbootstrap.com/docs/4.3/components/forms/

- Barre de navigation :
https://www.w3schools.com/bootstrap4/bootstrap_navbar.asp

https://getbootstrap.com/docs/4.3/components/navbar/

QUELQUES ÉLÉMENTS UTILES

- Fenêtre modale (nécessite de connaître un peu javascript) :

https://www.w3schools.com/bootstrap4/bootstrap_modal.asp

https://getbootstrap.com/docs/4.3/components/modal/

- Carousel :

https://www.w3schools.com/bootstrap4/bootstrap_carousel.asp

https://getbootstrap.com/docs/4.3/components/carousel/

EN GUISE DE CONCLUSION

Bootstrap :

- Accélère la réalisation de la mise en page
- Facilite la création d'un design responsive
- Donne une apparence « propre »

- N'est pas adapté à des sites avec un design original
- N'est pas le seul framework CSS existant (loin de là)