1 Multimédia

1.1 Images

- Les images sont insérées dans la page HTML avec `<img>` :

  ```
  <img src="URL image" width="..." height="..." alt="...">
  ```

- Insérer une nouvelle image : créer le DOM de l'élément `<img>` et l'insérer au bon endroit dans le document HTML
- Changer l'image affichée : changer la valeur de l'attribut `src` fait charger l'image correspondante et l'afficher à la place dans le navigateur
- Pré-charger une image dans le cache du navigateur : créer un objet `Image` et définir sa propriété `src`

  ```
  (new Image()).src = "URL image";
  ```

1.2 Images vectorielles : SVG

– SVG (Scalable Vector Graphic) est un format XML d'image vectorielle :

```
<?xml version="1.0" ?>
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="100" y="200" width="800" height="600"
        stroke="black" stroke-width="25"/>
  ...
</svg>
```

– Inclure un fichier SVG externe :

  – `<object data="`*URL svg*`" type="image/svg+xml" width="..." height="..."></object>`

  – `<img src="`*URL svg*`" width="..." height="..." alt="...">`

- Inclure une figure SVG directement dans le document HTML :
  - `<svg width="..." height="..."> ... </svg>`
- On peut créer/manipuler du SVG embarqué grâce aux méthodes du DOM
- Permet de modifier facilement des parties de l'image
- Le rendu d'une image SVG est plus coûteux qu'une image bitmap

1.3 Dessin avec `canvas`

– Avoir un élément `canvas` dans la page :

```
<canvas width="..." height="..."
        role="img" aria-label="description" ></canvas>
<canvas width="..." height="...">contenu alternatif</canvas>
```

– Récupérer un contexte de rendu 2D :

```
//<canvas id="moncanvas" ...>
const canvas = document.getElementById("moncanvas");
const ctx = canvas.getContext('2d');
```

– Dessiner en utilisant les propriétés et méthodes du contexte de rendu

– Ou utiliser des bibliothèques (`charts.js` ...) qui dessinent dans un canevas

1.4 Son et vidéo

- Inclure du son ou de la vidéo dans une page web (HTML 5) : éléments `<audio>` et `<video>`, indiquant les sources
  - dans leur attribut `src` (éléments alors vides)
  - dans des sous-éléments `<source>`
- Les éléments `audio` et `video` ont des propriétés et des méthodes permettant de lancer/arrêter la lecture, changer le volume ...
- Les modifications dans la lecture, les paramètres ... donnent lieu aussi à des événements auxquels on peut réagir

2 Mise en page : feuilles de style CSS

2.1 Feuilles de style

- HTML décrit le contenu sémantique d'une page
- CSS décrit le style du document HTML, c.-à-d. comment les éléments doivent être affichés

→ Choisir un élément d'après sa fonction, et non pas d'après son apparence qui peut être changée avec CSS

- Syntaxe générale :
  - *sélecteur { déclarations de style }*
  - déclaration de style : *propriété: valeur;*
- Ajouter du style à une page : style externe, style interne, style en-ligne

2.1.1 Style externe

- Style écrit dans un fichier css externe
- Charger le fichier dans l'élément `<head>`

```
<link rel="stylesheet" type="text/css" href="fichier.css">
```

2.1.2 Style interne

- Style écrit directement dans la page dans un élément `<style>` dans l'élément `<head>`

2.1.3 Style en-ligne

- Propriétés directement définies sur l'élément qui sont prioritaires par rapport aux règles des feuilles de styles
- HTML : style mis directement dans l'attribut `style` de l'élément

2.2 Sélecteur

- Un sélecteur permet de sélectionner des éléments
- Éléments simples :
  - `elem` : de type `<elem>`
  - `*` : de n'importe quel type
  - `#identifiant` : d'attribut `id="identifiant"`
  - `.classe` : dont la classe contient `classe`
  - `elem.classe` : de type `<elem>` dont la classe contient `classe`

- Éléments qui ont un attribut `attribut`
  - `[attribut]` : présent
  - `[attribute = "valeur"]` : valant `valeur`
  - `[attribute ~= "valeur"]` : contenant `valeur`
  - `[attribute ^= "valeur"]` : commençant par `valeur`
  - `[attribute $= "valeur"]` : terminant par `valeur`
  - `[attribute *= "valeur"]` : contenant `valeur`
  - `[attribute |= "valeur"]` : commençant par `valeur` où valeur est un mot entier ou terminant par un tiret `-`

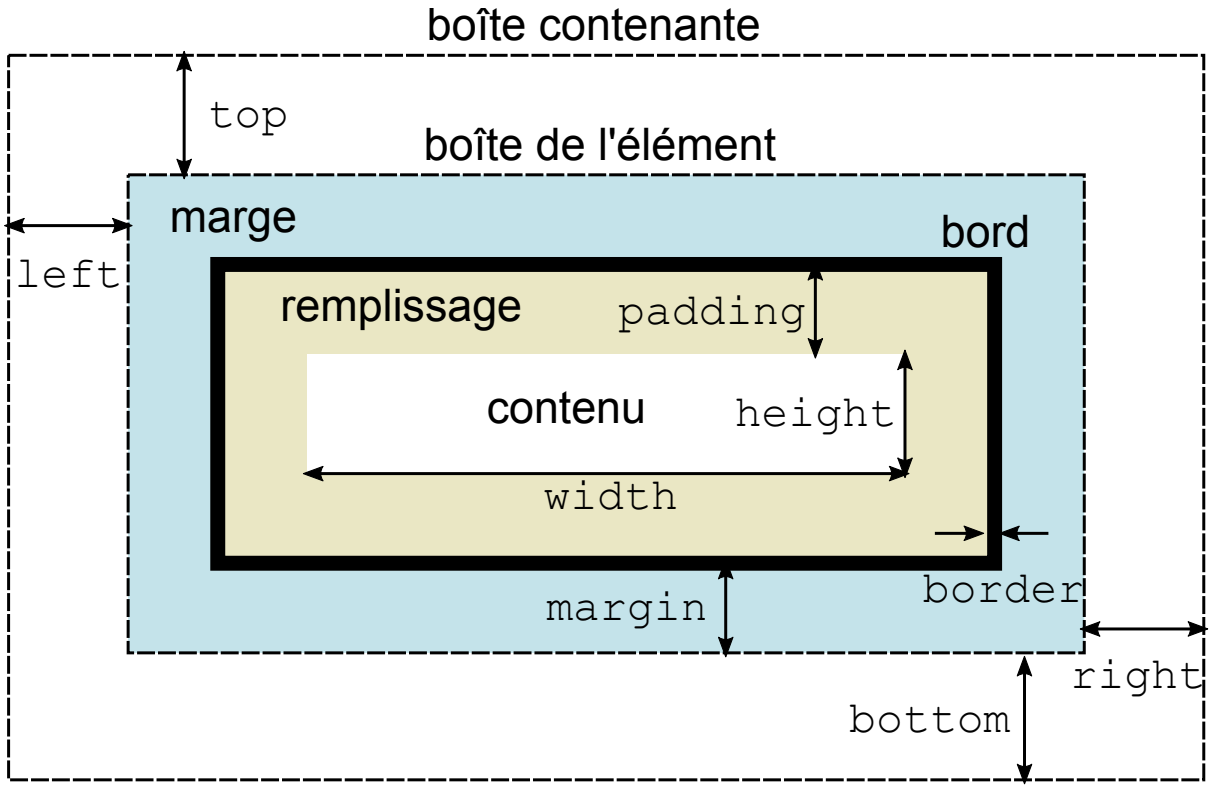
- Sélection d'éléments en fonction de leur position : éléments `<element>`
  - `ancetre element` : descendants de (c.-à-d. dans) `<ancetre>`
  - `parent > element` : enfants de `<parent>`
  - `colatprec + element` : collatéraux adjacents de `<colatprec>`
  - `colat ~ element` : collatéraux de `<colat>`
- Groupe de sélecteurs : liste de sélecteurs séparés par une virgule

2.3 Style d'affichage

- CSS a de nombreuses propriétés pour styler l'affichage :
  - texte : fonte, taille, alignement, ...
  - couleur : tracé, fond, bord, ...
  - ...

2.4 Affichage, positionnement et dimensionnement

– Le contenu des éléments sont mis dans des boîtes dimensionnées et positionnées suivant certaines propriétés CSS



2.4.1 Affichage : type des boîtes

- Affichage d'un élément : `display`
  - `inline` : en-ligne (dans le flot courant)
  - `block` : en bloc (crée un nouveau flot)
  - `flex` : conteneur flexible (bloc)
  - `grid`, `inline-grid` : conteneur grille (bloc), item d'une grille
  - valeurs liées à l'affichage en tableau
  - `list-item` : comme un élément `<li>`
  - `contents` : l'élément disparaît, ce sont ses enfants qui sont affichés
  - `none` : l'élément est enlevé de l'affichage
- Visibilité d'un élément : `visibility`
  - `visible` : l'élément est visible (défaut)
  - `hidden` : l'élément est caché (mais occupe toujours sa place)

2.4.2 Positionnement

- Pour positionner un élément :
  - indiquer la méthode de positionnement utilisée : `position`
  - indiquer ensuite le positionnement relatif : `top`, `bottom`, `left` et `right`
- Positionnement dans le flot normal : `position`
  - `static` : positionnement normal (`top`, `bottom`, `left` et `right` n'ont pas d'effet)
  - `relative` : l'élément occupe sa place normale, mais l'affichage est décalé de `top`, `bottom`, `left` et `right`

- Positionnement hors du flot normal : `position`
  - l'affichage de l'élément est enlevé du flot normal et ajouté décalé de `top`, `bottom`, `left` et `right` par rapport
  - `fixed` : à la fenêtre du navigateur
    → reste fixe quand on fait défiler la page
  - `absolute` : au premier ancêtre positionné (c.-à-d. dont la propriété `position` a été définie), sinon par rapport à `<body>`
    → bouge avec la page
- Avec les positionnements les affichages peuvent se chevaucher
- Les éléments sont affichés (éventuellement l'un sur l'autre) par défaut dans l'ordre du document
- Changer l'ordre d'affichage avec `z-index` : un élément de `z-index` plus élevé qu'un autre est affiché par dessus

2.4.3 Dimensionnement

– Dimension du contenu : `width` et `height`

2.5 Animation

– Quand le style d'un élément change :

  – par défaut l'élément est ré-affiché instantanément

  – on peut indiquer une transition entre les changements de style

– `transition:` *`propriété durée vitesse délai`*`;`

  – *`propriété`* : nom de la propriété CSS pour laquelle on établit la transition

  – *`durée`* : durée (en secondes ou milli-secondes) de la transition

  – *`vitesse`* : courbe de vitesse de la transition (`ease`, `linear`, ...)

  – *`délai`* : délai avant début de la transition

2.6 Pseudo-classes

- Pseudo-classes : s'appliquent à un élément en fonction de son état
- État qui peut changer à la suite d'action de l'utilisateur
- `sélecteur:pseudo-classe { déclarations de style }`

2.6.1 État des contrôles des formulaires

- `:focus` : contrôle qui a le focus
- `:checked` : contrôles sélectionnés (`<input>` boutons radio et case à cocher, `<option>`)
- `:in-range`, `:out-of-range` : contrôles dont la valeur est dans, hors de l'intervalle des valeurs permises
- `:valid`, `:invalid` : contrôles dont la valeur est valide, invalide
- `:optional`, `:required` : contrôles dont la valeur n'est pas, est requise (attribut `required`)

– `:enabled`, `:disabled` : contrôles activés, désactivés

– `:read-only`, `:read-write` : contrôles avec attribut `readonly`, sans cet attribut

2.6.2 Navigation

– État du lien d'une ancre `<a>` :

  `a:link`, `a:link`, `a:link` : lien non visité, visité, actif

– `:hover` : éléments tels que la souris est sur leur contenu

– `:target` : élément sur lequel on vient de naviguer en cliquant sur une ancre de lien interne `<a href="#id">`

2.6.3 Position dans le DOM

– Des pseudo-classes s'appliquent aux élément en fonction de leur position dans le DOM

2.7 Applications sur une Seule Page avec CSS

– Application sur une Seule Page (SPA) : page qui affiche des contenus différents dans le navigateur sans quitter la page HTML

2.7.1 Principe général

– Les différents contenus sont tous présents dans la page dans des éléments identifiés par un `id`

– Ces contenus sont cachés par défaut

– Et affichés quand on navigue dessus

– Fournir des liens pour naviguer sur ces contenus `<a href="#id">`

– Si besoin, s'aider de javascript pour naviguer sur un contenu : `window.location="#id";`

2.7.2 Dialogue modaux

– Dialogue modal : dialogue qui bloque les interactions avec l'application en dehors de lui-même

```
<div class="dialogue">
  <div class="contenu-dialogue">
    <!-- contenu du dialogue -->
  </div>
</div>
```

– `<div class="dialogue">` :
  – prend toute la place de la fenêtre, affiché au-dessus du contenu la page, fond opaque ou semi-transparent
  – caché par défaut, affiché quand on naviguera dessus
– `<div class="contenu-dialogue">` :
  – contient le contenu du dialogue
  – à positionner par rapport à son parent `<div class="dialogue">`

2.8 Manipuler le style avec javascript

2.8.1 Style en-ligne d'un élément

- Propriété `style` : a pour propriétés les propriétés CSS
  - en bosse de dromadaire (`font-size` → `fontSize`)
  - ayant pour valeur une chaîne contenant la valeur à la syntaxe CSS (`e.style.fontSize = "24pt";`)
- `style` a aussi pour propriétés et méthodes (utilisant la syntaxe CSS)
  - `cssText` : toutes les propriétés
  - `getPropertyValue("nom CSS")` : valeur de la propriété
  - `setProperty("nom CSS","valeur CSS","")` : change la valeur de la propriété
  - `removeProperty("nom CSS")` : enlève la propriété

2.8.2 Classes de style

- Au lieu de changer le style en-ligne d'un élément
- Changer ses classes CSS pour changer le style qui lui est appliqué :
  - propriété `className` : chaîne, valeur de l'attribut `class`
  - propriété `classList` : permet de manipuler les noms de classe individuellement grâce à ses méthodes `add(`*nom*`)`, `remove(`*nom*`)`, `toggle(`*nom*`)`, `contains(`*nom*`)`

2.8.3 Style externe

- Charger dynamiquement une feuille de style externe : avec les méthodes du DOM créer l'élément `<link>` correspondant et l'insérer dans `<head>`
- Activer/désactiver une feuille de style : propriété `disabled` des éléments `<style>` et `<link>`

3 Amélioration progressive

- Une page Web doit pouvoir s'adapter
  - au fait que les feuilles de style et javascript ne sont pas forcément activés
  - aux navigateurs particuliers (en mode texte, en mode audio pour les personnes handicapées par exemple)
- Amélioration progressive :
  - la page doit pouvoir fonctionner correctement dans tous les cas
  - les fonctionnalités supplémentaires apportant des améliorations ne doivent s'ajouter que si elles sont supportées

- Mettre en application l'amélioration progressive :

  1. avoir un document HTML complet et bien structuré contenant tout le contenu sémantique et toutes les fonctionnalités nécessaires et utilisables
  2. ajouter de la mise en page avec des feuilles de styles
  3. si javascript est disponible, ajouter des améliorations grâce à javascript :

     - mettre un script qui charge le script de l'application
     - si l'apparence de certains éléments doit être changée, changer leur classe avec javascript
       (et éventuellement charger la feuille style correspondant à ces classes pour qu'elle ne soit pas chargée par défaut)

- Pour permettre un meilleur développement et une meilleure maintenance
  - bien séparer le contenu HTML, la mise en page CSS, les fonctionnalités javascript :
    - HTML : pas de style ou de javascript inline
    - javascript : limiter la création de contenu HTML, limiter la manipulation de style inline
  - avoir un objet définissant des constantes pour les noms de classes, de certains id utilisés dans le HTML et les utiliser dans le code javascript

```
const sc = {
// CSS classes
  hidingClass:'hide',    // Hide elements
  dynClass:   'dynamic', // Indicate dynamic site support
  HLStyleSheet:'highlevel.css', // Style sheet for dynamic site
};
```