Séance 9 : CSS Avancé et Frameworks

Repousser les Limites du Design CSS

Fossé Stéphane

30 Mai 2025

IUT de Bordeaux

Aperçu de la Séance 9

Objectifs de la Séance

- Découvrir des fonctionnalités CSS plus puissantes et créatives.
- Approfondir les transformations et animations CSS.
- Explorer des sélecteurs avancés.
- Formaliser l'utilisation des Frameworks CSS.

1. Introduction : Repousser les Limites du Design CSS

- Solide base en HTML et CSS, incluant variables CSS et Tailwind CSS.
- Cette séance : niveau supérieur avec des fonctionnalités CSS plus puissantes et créatives.
- Approfondir transformations et animations, explorer sélecteurs avancés, formaliser Frameworks CSS.

2. Maîtriser le CSS Dynamique et Modulaire

Transformations CSS (2D & 3D)

- `transform` : modifier forme et position sans affecter le flux.
- 2D : `translate(x, y)`, `rotate(angle)`, `scale(x, y)`, `skew(x-angle, y-angle)`.
- 3D : `perspective` sur le parent, `rotateX`, `rotateY`, `rotateZ`, `translateZ`.
- `transform-origin` : point de pivot.

Exemples de Transformations

```css
.box-2d:hover {
    transform: translateX(20px) rotate(15deg) scale(1.1);
}
.parent-3d {
    perspective: 1000px; /* Important pour la 3D */
}
.box-3d:hover {
    transform: rotateY(180deg); /* Faire pivoter sur l'axe Y */
}
```

2.2 Mise en Page Créative : shape-outside

- `shape-outside` : permet au texte de s'enrouler autour de formes complexes (cercle, ellipse, polygone).
- Brise la monotonie des mises en page rectangulaires.

```
.shape {
    float: left;
    width: 150px;
    height: 150px;
    background-color: #818CF8;
    clip-path: circle(50%);
    shape-outside: circle(50%);
    margin-right: 20px;
}
```

2.3 Pseudo-classes et Pseudo-éléments Avancés

Pseudo-classes Avancées

- `nth-child(n)`, `nth-of-type(n)` : cibler par position.
- `first-child`, `last-child`, `only-child`.
- `not(selector)` : exclure des éléments.
- `focus-within` : cibler parent si descendant a le focus.
- Formulaires : `valid`, `invalid`, `checked`, `disabled`.

Pseudo-éléments `::before` et `::after`

- Insérer du contenu cosmétique avant/après un élément.
- Nécessitent `content`, souvent `display: block` et `position: absolute`.

Exemples de Pseudo-classes et Pseudo-éléments

```css
ul li:nth-child(odd) {
    background-color: #E0F2FE;
}
.button-icon::before {
    content: '⬇️';
    position: absolute;
    left: 0.5rem;
    top: 50%;
    transform: translateY(-50%);
}
```

2.4 Graphiques Vectoriels (SVG)

- Standard pour icônes et logos : infiniment redimensionnables sans perte de qualité.
- Intégration directe dans HTML, stylisables avec CSS.

```
<svg class="icon-heart" viewBox="0 0 24 24">...</svg>
```

```
.icon-heart {
    width: 50px; height: 50px;
    stroke: #4F46E5; fill: none;
    transition: all 0.3s ease;
}
.icon-heart:hover {
    fill: #FCA5A5; stroke: #DC2626;
}
```

2.5 Filtres, Modes de Fusion et Dégradés

- Filtres CSS (`filter`) : flou, saturation, etc.
  - `filter: blur(5px);`, `filter: grayscale(100%);`
- Effet “Verre Dépoli” (`backdrop-filter`) : applique un filtre à la zone derrière un élément.
  - `backdrop-filter: blur(10px);`
- Dégradés (`background-image`) : `linear-gradient`, `radial-gradient`.
  - `background-image: linear-gradient(to right, #818CF8, #4F46E5);`

2.6 Animations CSS (@keyframes)

- Créer des séquences d'animations complexes avec plusieurs étapes.
- `keyframes` : définit les styles à des points spécifiques (`0%` à `100%`, `from`/`to`).

```
@keyframes slideIn {
    0% { transform: translateX(-100%); opacity: 0; }
    100% { transform: translateX(0); opacity: 1; }
}
```

- Propriété `animation` : `animation-name`, `animation-duration`, `animation-timing-function`, `animation-delay`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode`, `animation-play-state`.
- Raccourci : `animation: slideIn 1s ease-out forwards;`

Exemple d'Animation

```
@keyframes bounce {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-30px); }
}
.bouncing-ball {
    width: 50px; height: 50px;
    background-color: #6EE7B7;
    border-radius: 50%;
    animation: bounce 1s infinite alternate;
}
```

2.7 Animer au Rythme du Défilement

- Lier la progression d'une animation au défilement d'un conteneur (CSS Scroll-driven Animations).
- Exemple : barre de progression de lecture.

```
@keyframes grow-progress {
  from { transform: scaleX(0); }
  to { transform: scaleX(1); }
}
.progress-bar {
    position: fixed; top: 0; left: 0;
    width: 100%; height: 8px;
    background-color: #4F46E5;
    transform-origin: left;
    animation: grow-progress linear;
    animation-timeline: scroll(root); /* La magie est ici */
}
```

2.8 Les Frameworks CSS

Qu'est-ce qu'un Framework CSS ?

- Collection de fichiers CSS pré-écrits (et JS) pour faciliter et accélérer le développement.
- Bibliothèque de briques de construction préfabriquées.

Pourquoi Utiliser un Framework CSS ?

- Rapidité de développement.
- Cohérence du design.
- Responsivité intégrée.
- Bonnes pratiques.

Deux Approches Communes

Frameworks Basés sur les Composants (ex: Bootstrap)

- Fournissent des “composants” prêts à l’emploi (boutons, navbars, cartes).
- Utilisation via classes spécifiques.

```
<button class="btn btn-primary">Bouton Principal</button>
```

Frameworks “Utility-First” (ex: Tailwind CSS)

- Fournissent des milliers de “classes utilitaires” atomiques.
- Construction du design en composant ces classes directement dans HTML.

```
<button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded">
    Mon Bouton Stylisé
</button>
```

Notre Choix : Tailwind CSS

Avantages de Tailwind CSS

- Contrôle total sur chaque aspect du style.
- Développement rapide.
- Moins de CSS personnalisé.
- Facile à apprendre (noms de classes intuitifs).
- Continuons à combiner Tailwind avec les techniques CSS avancées.

3. Exercices d'Application

- Exercice 1 : Gestion de Thème avec Variables CSS (mode sombre).
- Exercice 2 : Effets Visuels avec Transformations 2D et 3D.
- Exercice 3 : Liste Stylisée avec Pseudo-classes et Pseudo-éléments.
- Exercice 4 : Animation CSS (Loader).
- Exercice 5 : Création de Composant avec Tailwind CSS (aucun CSS personnalisé).

4. TP : Card de Produit Interactive

Objectif

- Concevoir une “carte de produit” responsive et interactive.
- Utiliser variables CSS, transformations/animations, pseudo-éléments, classes Tailwind CSS.

Consignes

- Nouveau dossier `tp-seance9-card-interactive`.
- `index.html` : structure de card (image, titre, description, prix, bouton “Ajouter au panier” avec SVG).
- `styles.css` :
  - Variables CSS pour palette de couleurs.
  - Tailwind CSS pour layout général.
  - Transformations : `scale`, `box-shadow` au survol de la carte.
  - Animations : bouton “Ajouter au panier” avec `fadeInUp`.

Consignes

- Pseudo-éléments : `::before`/`::after` sur bouton, “badge” “Nouveau !” sur image.
- SVG : styliser icône SVG.
- Filtres et Dégradés : `grayscale()` sur image au survol, dégradé pour fond de bouton.
- Transitions pour effets fluides.

Fin de la Séance 9

Récapitulatif

- Variables CSS pour maintenabilité.
- Transformations et animations pour expériences utilisateur engageantes.
- Pseudo-classes et pseudo-éléments pour ciblage précis.
- Frameworks CSS (Tailwind) pour développement rapide et standardisé.

Prochaines Étapes

- Pratiquer, expérimenter, imaginer de nouvelles applications.