CSS

Support réalisé par Pierrick Hauguel

Pseudo-element

Un pseudo-élément CSS est utilisé pour styliser une partie spécifique d'un élément. Il permet d'ajouter des effets sur une partie d'un élément sans modifier le HTML.

```css
/* Ajoute du contenu avant chaque paragraphe */
p::before {
  content: "Note: ";
  color: green;
}

/* Ajoute un style à la première lettre d'un paragraphe */
p::first-letter {
  font-size: 200%;
  color: blue;
}
```

CSS Pseudo-class

Les pseudo-classes CSS permettent de définir des styles pour des états spéciaux d'un élément, comme lorsqu'il est survolé par la souris ou lorsqu'il est sélectionné.

Styliser les liens lorsqu'ils sont survolés :

```
a:hover {
  color: red;
}
```

Styliser les éléments de formulaire en focus :

```
input:focus {
  border-color: blue;
}
```

Styliser les éléments sélectionnés :

```css
p::selection {
  background-color: yellow;
  color: black;
}
```

CSS Attribute Selectors

Les sélecteurs d'attributs permettent de cibler des éléments en fonction de la présence ou de la valeur de leurs attributs.

Cibler les éléments avec un attribut spécifique :

```css
/* Sélectionner tous les liens avec l'attribut target */
a[target] {
  color: blue;
}
```

Cibler les éléments avec une valeur d'attribut spécifique :

```
/* Sélectionner les liens qui s'ouvrent dans une nouvelle fenêtre */
a[target="_blank"] {
  color: red;
}
```

Cibler les éléments avec une partie de valeur d'attribut :

```
/* Sélectionner les liens dont l'URL contient 'example' */
a[href*="example"] {
  color: green;
}
```

CSS Forms

Le style des formulaires permet d'améliorer l'ergonomie et l'apparence des éléments de formulaire comme les champs de texte, les boutons et les sélecteurs.

Style de base pour les champs de formulaire :

```css
input[type=text], select {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
```

Style pour les boutons de soumission :

```css
input[type=submit] {
  width: 100%;
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

input[type=submit]:hover {
  background-color: #45a049;
}
```

Math Functions

Les fonctions mathématiques en CSS permettent de faire des calculs dynamiques directement dans les propriétés de style. Les fonctions les plus courantes sont calc(), min(), max() et clamp().

Utilisation de calc()

```css
/* Calculer la largeur en soustrayant 50px de la largeur totale */
div {
  width: calc(100% - 50px);
}

/* Ajouter deux valeurs ensemble */
p {
  margin: calc(1em + 10px);
}
```

Utilisation de min(), max() et clamp()

```css
/* Choisir la valeur minimum entre 50% et 300px */
div {
  width: min(50%, 300px);
}

/* Choisir la valeur maximum entre 50% et 300px */
div {
  width: max(50%, 300px);
}

/* Limiter une valeur entre une plage donnée */
div {
  font-size: clamp(1rem, 2vw, 2rem);
}
```

CSS Opacity

La propriété opacity en CSS est utilisée pour spécifier l'opacité d'un élément. Une valeur de 1 signifie complètement opaque, tandis qu'une valeur de 0 signifie complètement transparent.

```css
/* Élément complètement opaque */
div {
  opacity: 1;
}

/* Élément semi-transparent */
div {
  opacity: 0.5;
}
```

Navigation Bar

Une barre de navigation est un élément de l'interface utilisateur qui permet aux utilisateurs de naviguer entre les différentes sections d'un site web.

```css
/* Style de la barre de navigation */
nav {
  background-color: #333;
  overflow: hidden;
}

nav a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

nav a:hover {
  background-color: #ddd;
  color: black;
}
```

Image Sprites

Un sprite d'image est une collection d'images combinées en une seule grande image. Cela permet de réduire le nombre de requêtes HTTP et d'améliorer les performances du site.

```css
/* Sprite d'image */
.sprite {
  width: 50px;
  height: 50px;
  background-image: url('sprite.png');
  background-position: 0 0; /* Position de la première image */
}

.sprite:hover {
  background-position: 0 -50px; /* Position de la deuxième image */
}
```

CSS Units

Les unités CSS définissent la mesure utilisée pour les propriétés de style. Les unités courantes incluent px, em, rem, %, vh, et vw.

```css
/* Unités CSS */
div {
  width: 50%;
  height: 100vh;
  padding: 1em;
  font-size: 2rem;
}
```

CSS !important

La règle !important en CSS est utilisée pour donner plus de poids à une déclaration de style, surpassant toute autre règle.

```css
/* Utilisation de !important */
p {
  color: blue !important;
}

p {
  color: red;
}
```

CSS z-index

La propriété z-index en CSS spécifie la profondeur d'un élément par rapport aux autres éléments. Un élément avec un z-index plus élevé sera placé au-dessus des éléments avec un z-index inférieur.

Utilisation basique du z-index :

```css
/* Élément avec un z-index plus bas */
.box1 {
  position: absolute;
  left: 50px;
  top: 50px;
  width: 100px;
  height: 100px;
  background-color: red;
  z-index: 1;
}

/* Élément avec un z-index plus élevé */
.box2 {
  position: absolute;
  left: 100px;
  top: 100px;
  width: 100px;
  height: 100px;
  background-color: blue;
  z-index: 2;
}
```

Contextes d'empilement :

```css
/* Création de contextes d'empilement */
.container {
  position: relative;
  z-index: 1;
}

.child {
  position: absolute;
  z-index: 2;
}
```

CSS Transform

La propriété transform en CSS permet d'appliquer des transformations 2D ou 3D à un élément. Les transformations courantes incluent la rotation, la mise à l'échelle, la translation et l'inclinaison.

Translation :

```css
/* Déplacer un élément de 50px à droite et de 20px vers le bas */
.move {
  transform: translate(50px, 20px);
}
```

Rotation :

```css
/* Faire pivoter un élément de 45 degrés */
.rotate {
  transform: rotate(45deg);
}
```

Mise à l'échelle :

```css
/* Agrandir un élément à 150% de sa taille d'origine */
.scale {
  transform: scale(1.5);
}
```

Combinaison de transformations :

```css
/* Appliquer plusieurs transformations */
.combined {
  transform: translate(50px, 20px) rotate(45deg) scale(1.5);
}
```

CSS Animations

Les animations CSS permettent de créer des effets visuels dynamiques en modifiant les styles d'un élément sur une période définie.

Définir une animation de base :

```css
/* Définir une animation nommée "example" */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* Appliquer l'animation à un élément */
.animate {
  animation-name: example;
  animation-duration: 4s;
}
```

Animation avec plusieurs étapes :

```css
/* Définir une animation avec plusieurs étapes */
@keyframes example {
  0%   {background-color: red; left: 0px; top: 0px;}
  50%  {background-color: yellow; left: 200px; top: 0px;}
  100% {background-color: blue; left: 200px; top: 200px;}
}

/* Appliquer l'animation */
.animate {
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}
```

Animation en boucle et autres propriétés :

```css
/* Appliquer une animation infinie avec un délai et une fonction de timing */
.bounce {
  animation: bounce 2s infinite;
  animation-delay: 1s;
  animation-timing-function: ease-in-out;
}

@keyframes bounce {
  0%, 100% {transform: translateY(0);}
  50% {transform: translateY(-100px);}
}
```

Introduction aux Media Queries

Les Media Queries en CSS permettent d'appliquer des styles spécifiques en fonction des caractéristiques de l'appareil, telles que la largeur de la fenêtre, la hauteur, la résolution, et bien plus. Elles sont essentielles pour créer des sites web réactifs (responsive design).

Syntaxe de Base des Media Queries

La syntaxe des Media Queries commence par @media suivi par une ou plusieurs conditions entre parenthèses. Les styles à appliquer sont placés à l'intérieur des accolades.

```
@media (condition) {
  /* Styles à appliquer */
}
```

Media Query par Largeur de la Fenêtre

Les Media Queries les plus courantes sont basées sur la largeur de la fenêtre. Cela permet de définir des styles différents pour les écrans de différentes tailles.

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Media Query pour les Appareils Mobiles

Les Media Queries peuvent cibler spécifiquement les appareils mobiles pour améliorer l'expérience utilisateur sur les smartphones et les tablettes.

```
@media (max-width: 768px) {
  .menu {
    display: none;
  }
}
```

Media Query pour les Appareils de Bureau

Les Media Queries peuvent également cibler les appareils de bureau pour appliquer des styles spécifiques aux écrans plus grands.

```
@media (min-width: 1024px) {
  .sidebar {
    width: 300px;
  }
}
```

Media Queries pour la Hauteur de la Fenêtre

Il est possible de cibler la hauteur de la fenêtre avec les Media Queries pour adapter le contenu verticalement.

```
@media (max-height: 800px) {
  .footer {
    display: none;
  }
}
```

Media Queries pour l'Orientation de l'Écran

Les Media Queries peuvent détecter l'orientation de l'écran (portrait ou paysage) et appliquer des styles en conséquence.

```
@media (orientation: landscape) {
  .gallery {
    flex-direction: row;
  }
}

@media (orientation: portrait) {
  .gallery {
    flex-direction: column;
  }
}
```

Media Queries pour la Résolution de l'Écran

Les Media Queries peuvent cibler la résolution de l'écran pour optimiser les images et les contenus pour des écrans haute résolution comme les écrans Retina.

```
@media (min-resolution: 2dppx) {
  .logo {
    background-image: url('logo@2x.png');
  }
}
```

Media Queries Combinées

Il est possible de combiner plusieurs conditions de Media Queries pour un contrôle plus précis des styles appliqués.

```
@media (min-width: 600px) and (max-width: 1200px) {
  .container {
    width: 80%;
  }
}
```