

Javascript et CSS

Présentation

- Nous pouvons spécifier le style de l'élément en trois endroits :

- En attribut de l'élément :

`<p style="background:navy;color:white;"></p>` C'est le style dit *inline*.

- Dans une feuille de style interne à la page, dans l'élément style enfant de l'élément head.
 - Dans une feuille de style externe, spécifiée par un élément link enfant de head :

```
<link rel="stylesheet" type="text/css" href="style.css"/>
(HTML 5: l'attribut type n'est pas nécessaire.)
```

Règles css

- Dans les feuilles de style internes ou externes, nous pouvons donner à un ensemble d'éléments un même style, appelé *sélecteur*. L'association d'un sélecteur et de une ou plusieurs directives constitue une règle.
- Exemple:

```
<style>
p, li {
color: navy;
} </style>
```
- Priorité d'application des règles:
d'abord celle en attribut, puis en feuille de style interne,
puis en feuille de style externe. S'il y a plusieurs feuilles externes, la dernière l'emporte

Règles css

Types de sélecteurs:

Il existe trois types de sélecteurs de base :

- Un nom de balise : les directives s'appliquent à tous les éléments de ce type. Le signe étoile (*) signifie : tout élément.
- .nomDeClasse : les directives s'appliquent à tous les éléments portant un attribut class="nomDeClasse".
- #idElement : les directives s'appliquent seulement à l'élément (unique) dont l'id est idElement.
- Les sélecteurs a:hover ou a:visited indiquent les liens quand la souris est au dessus du lien ou quand ils ont déjà été visités.
- Le sélecteur id l'emporte sur le sélecteur class, qui l'emporte sur le sélecteur de balise.

L'attribut *style* de *JavaScript*

- Tout élément DOM possède un attribut `style`, et chaque directive CSS en devient une propriété, son nom étant « camélisé ». Par exemple, `font-size` devient `fontSize`.
- l'attribut `style` d'un élément `ele` fait référence à l'attribut `style` de l'élément correspondant à `ele` dans le document HTML. Modifier une de ses propriétés ne pose aucun problème. Par contre, nous ne pouvons lire que celles définies dans l'attribut `style` de l'élément en HTML ou déjà modifiées en JavaScript. Pour les autres, nous obtenons une chaîne vide.

Les boîtes

- `Display:bloc; display:inline`
- `element.style.display = "none"; // masque l'élément`
- `element.style.display = ""; // rétablit son affichage par défaut`

Taille des boîtes : *width et height*

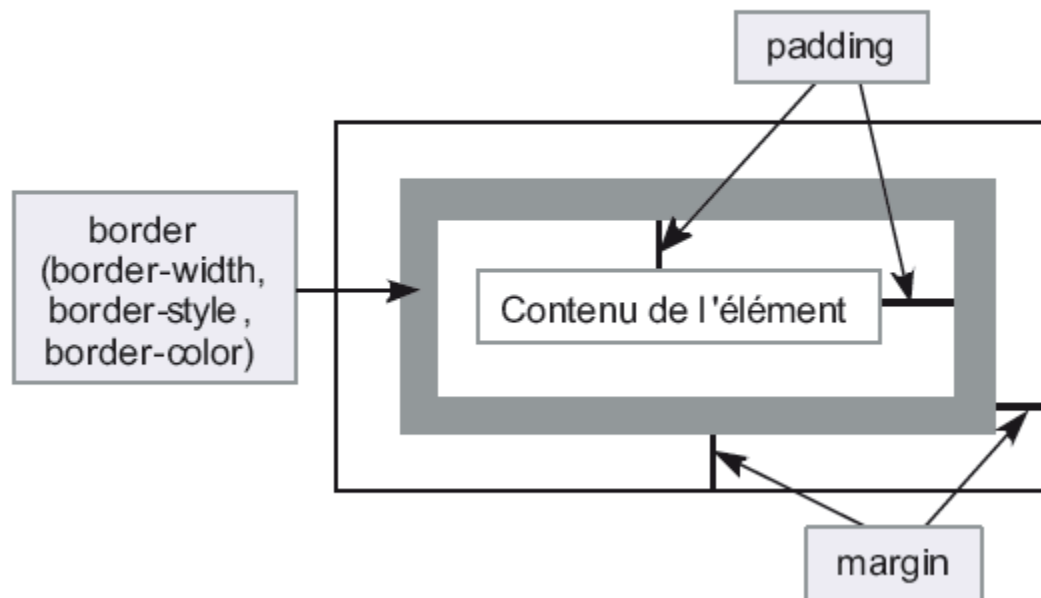
Les directives `width` et `height`, qui contrôlent la taille des boîtes, sont interprétées différemment selon le navigateur :

Internet Explorer $\leq$ IE 8, considère que ces tailles incluent la marge interne et la bordure, au contraire des autres navigateurs, qui suivent la recommandation W3C, pour qui elles indiquent uniquement la taille du contenu.

Les boîtes

Il est possible de changer la signification donnée par le W3C en celle donnée par Microsoft, grâce à la directive `box-sizing`, ajoutée dans CSS 3 .

- `box-sizing: border-box; /* la taille inclut border et padding (remplissage) , valeurs possibles :content-box | border-box | inherit */`



Exemple 1

Style:

```
<style type="text/css">
    td {
        background: #F6F6F6;
        padding: 4px;
    }
    div {
        background: #CCDDEE;
        margin: 1ex;
        padding: 1ex;
    }
</style>
```

Contenu:

```
<table>
    <tr>
        <td>
            <div id="id1" onmouseover="afficher(1)">HTML</div>
            <div id="id2" onmouseover="afficher(2)">AJAX</div>
            <div id="id3" onmouseover="afficher(3)">JAVASCRIPT</div>
        </td>
        <td id="detail">Survolez un lien pour avoir
            Une définition</td>
    </tr>
</table>
```

Exemple 1

Fonction Javascript

```
function afficher(index) {  
    // Afficher les détails d'un élément donné  
    var id = "id" + index;  
    obj=document.getElementById(id);  
    var donnees = requeteAjax(url+ "?mot="+obj.innerHTML);  
  
    document.getElementById("detail").innerHTML = donnees;  
    // Mettre les zones au style par défaut  
    for (var i=1 ; i<=3 ; i++) {  
document.getElementById("id" + i ).style.background =  
"#CCDDEE";  
    }  
    // Mettre en exergue celui qu'on détaille  
    obj.style.background = "#99AABB";  
}
```

Example 2

```
<style>
  .infoBox {
    border: solid 1px #BBCCDD;
    width: 15em;
  }
  .infoBoxEntete {
    background: #DAEAF0;
    cursor: pointer;
    text-align: center;
    font-weight: bold;
  }
  .infoBoxMessage {
    background: #F0F7FA;
  }
</style>
```

Exemple 2

- Script

```
<script type="text/javascript">
    function afficher(idDetail) {
        var style = document.getElementById(idDetail).style;
        style.display = (style.display == "none") ? "" : "none";
    }
</script>
```

- Contenu

```
<h1>Afficher/masquer</h1>
<div class="infoBox">
    <div class="infoBoxEntete" onClick="afficher('details')">
        La météo
    </div>
    <div id="details" class="infoBoxMessage" style="display:none">
        Ici les données Météo
        <br/>Température, ciel, vent
    </div>
</div>
```

Les calques et le positionnement des boîtes

- La directive `visibility` est voisine de `display`. Si elle vaut `visible`, l'élément apparaît normalement ; si elle vaut `hidden`, l'élément disparaît, mais, à la différence de `display`, la place qu'il occupait demeure, et l'utilisateur voit une zone vide à la place.
- Les boîtes
 - La position d'une boîte est définie par les coordonnées de son coin supérieur gauche par rapport à celui de son bloc conteneur.
 - Pour les boîtes dans le flot, cette position est calculée par le navigateur en fonction des marges et des tailles des boîtes englobantes ou précédentes. Pour les autres, ces coordonnées sont spécifiées par les directives `left` et `top`, d'une part, et la directive `position`, d'autre part.
- Une boîte peut être sortie du flot normal en donnant à la directive `position` une valeur autre que la valeur par défaut `static`. La boîte définit alors un nouveau flot pour ses éléments descendants, pour lesquels elle est considérée comme le bloc conteneur (*containing block*). Un bloc conteneur est la racine d'un flot. Pour le flot normal, le bloc conteneur est l'élément `html`.

Les calques et le positionnement des boîtes

directive position :

- absolute : les coordonnées sont relatives au premier bloc conteneur ancêtre positionné (s'il n'existe aucune boîte ancêtre positionnée alors les coordonnées left, right, bottom et top sont relatives au document : balise body) La boîte n'occupe plus de place dans le flux normal de ce bloc.
- fixed : les coordonnées sont relatives au coin supérieur gauche de la fenêtre. L'utilisateur voit toujours la boîte au même endroit de la fenêtre, même s'il fait défiler la page. Notons que cela ne fonctionne pas dans Internet Explorer 6.
- relative : les coordonnées sont relatives à la position qu'aurait la boîte si elle était dans le flux normal. La place qu'elle y occuperait est remplacée par une zone vide, un peu comme pour visibility:hidden.

Le positionnement flottant

- **Le positionnement flottant**

On positionne un élément en flottant avec une déclaration `float: left` ou `float: right`. Il est alors retiré du flux normal pour prendre place à gauche (respectivement à droite) du bloc qui le contient : c'est devenu un « flottant ». L'élément qui le suit s'écoulera alors dans l'espace ainsi laissé libre.

Exemple

- Style

```
<style>
div {
width: 300px;
height: 200px;
background: yellow;
}
#premier {
width: 100px;
background: orange;
float:left;
}
2#second {
background: green;}
</style>
```

- Html

```
<div>
<p id="premier">premier paragraphe de texte</p>
<p id="second">deuxième paragraphe de texte deuxième paragraphe de texte
deuxième paragraphe de texte deuxième paragraphe de texte deuxième
paragraphe de texte </p>
</div>
```

Une liste de suggestion

`<style>`

```
#suggestion {  
border: solid 1px black;  
padding: 2px;  
background: white;  
position: absolute;  
display: none;  
}
```

`</style>`

Le débordement de contenu (Overflow)

Les propriétés overflow, overflow-x, overflow-y permettent gérer le contenu quand il est plus important que le conteneur.

Valeurs possibles:

- visible : le contenu est toujours visible.
- hidden : le contenu est toujours masqué.
- scroll : le contenu est visible par l'intermédiaire d'une barre de défilement qui est toujours visible.
- auto : barre de défilement affichée uniquement quand c'est nécessaire.

unités

Les longueurs CSS s'expriment en unités absolues, comme le pixel (px), ou relatives.

Celles-ci comprennent la hauteur du caractère *x dans la police de l'élément (ex), la taille de la police (em)* ou le pourcentage (par exemple, 50%), celui-ci étant réservé à certaines directives, comme la largeur (width). L'avantage des unités relatives est de s'adapter aux préférences de l'utilisateur.

- **Les unités de longueur absolues**

- Les unités absolues ont un certain rapport entre elles comme l'indiquent les valeurs suivantes :
- cm (centimeter = 1cm = 10 mm)
- in (inche = 1in = 2.54 cm)
- mm (millimeter)
- pc (pica = 1pc = 12pt)
- pt (point = 1pt = 1/72 in)

- Relatives

- **L'unité em**

- L'unité em se rapporte à la taille de la police Avec elle on peut affecter une mesure relative à la taille de police de l'élément parent. **Elle permet d'avoir des feuilles de style plus facilement adaptables d'un média à un autre.** Les nombres décimaux sont autorisés, mais il faut tout simplement remplacer la virgule par un point. Cette valeur em est utilisable pour d'autres propriétés acceptant la mention de longueur.

- **L'unité px**

- L'écran d'un ordinateur ou moniteur est formé par plusieurs petits "carrés". Ces carrées définissent la résolution ou densité de l'écran en pixels d'affichage selon l'unité de sortie, c'est-à-dire l'écran de l'ordinateur. L'unité px qui veut dire pixel correspond à un de ces petits carrés.
 - C'est en fait le plus petit élément de la résolution d'écran. Cette unité peut être utilisée pour toutes les propriétés acceptant les mentions de longueur.

- L'unité rem (root em): introduite en css3, cette unité est relative à l'élément racine.