

Les feuilles de style CSS

Nathalie Friburger

Département R&T
IUT de Blois

31 août 2022

Les feuilles de style CSS

Nathalie Friburger

Département R&T
IUT de Blois

31 août 2022

Plan

Plan

Plan

Plan

Historique

- 1991 : création de HTML
- de 1991 à 1996 : mise en forme faite uniquement en HTML
ex : ``
- 1996 : création de CSS (Cascading Style Sheet = feuille de style en cascade)
- 1998 : CSS2, ajout de positionnement
- depuis 2000 : développement de CSS3, en partie mis en oeuvre par les navigateurs

Historique

- 1991 : création de HTML
- de 1991 à 1996 : mise en forme faite uniquement en HTML
ex : ``
- 1996 : création de CSS (Cascading Style Sheet = feuille de style en cascade)
- 1998 : CSS2, ajout de positionnement
- depuis 2000 : développement de CSS3, en partie mis en oeuvre par les navigateurs

Historique

- 1991 : création de HTML
- de 1991 à 1996 : mise en forme faite uniquement en HTML
ex : ``
- 1996 : création de CSS (Cascading Style Sheet = feuille de style en cascade)
- 1998 : CSS2, ajout de positionnement
- depuis 2000 : développement de CSS3, en partie mis en oeuvre par les navigateurs

Historique

- 1991 : création de HTML
- de 1991 à 1996 : mise en forme faite uniquement en HTML
ex : ``
- 1996 : création de CSS (Cascading Style Sheet = feuille de style en cascade)
- 1998 : CSS2, ajout de positionnement
- depuis 2000 : développement de CSS3, en partie mis en oeuvre par les navigateurs

Historique

- 1991 : création de HTML
- de 1991 à 1996 : mise en forme faite uniquement en HTML
ex : ``
- 1996 : création de CSS (Cascading Style Sheet = feuille de style en cascade)
- 1998 : CSS2, ajout de positionnement
- depuis 2000 : développement de CSS3, en partie mis en oeuvre par les navigateurs

Navigateurs Web

Des affichages différents ...

Page google.fr



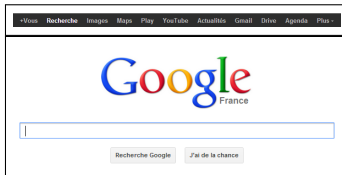
Mozilla firefox



Internet Explorer 10








Google Chrome



Navigateurs Web

Fonctionnalités CSS gérées par les différents navigateurs

source : <http://www.normansblog.de/demos/browser-support-checklist-css3/>

Browser Rendering Engine	 Firefox Gecko	 Safari Webkit	 Chrome Webkit	 Internet Explorer Trident	 Opera Presto						
Version	20	4	5	5.1+	26	6 – 8	9	10	11.1	11.6	12.1
Animations	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓
Background Gradients	✓	✓	✓	✓	✓	✗	✗	✓	▲	✓	✓
Background Size	✓	▲	✓	✓	✓	✗	✓	✓	✓	✓	✓
Border Image	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
Border Radius	▲	▲	▲	▲	✓	✗	✓	✓	✓	✓	✓
Box Shadow	✓	▲	✓	✓	✓	✗	✓	✓	✓	✓	✓
Columns	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Font Face	✓	✓	✓	✓	✓	▲	✓	✓	✓	✓	✓
HSLa	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Hyphens	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
Multiple Backgrounds	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Opacity	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
RGBA	✓	▲	▲	✓	✓	✗	✓	✓	✓	✓	✓
Text Overflow	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Text Shadow	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Transforms	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Transforms 3D	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗
Transitions	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Total CSS3 Support *	97.2%	83.4%	88.9%	97.2%	94.4%	8.3%	55.6%	94.4%	80.6%	83.3%	88.9%

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site !
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- **HTML ne permet de gérer :**
 - les marges, indentations, distance entre lignes, polices

Différences d'affichage entre navigateurs

HTML permet de mettre en forme les pages d'un site web

MAIS ...

- Pour changer l'aspect du site, il faut :
 - Modifier toutes les pages
 - Assurer le respect de la charte graphique du site!
- Les pages sont lentes à charger
- HTML ne permet de gérer :
 - les marges, indentations, distance entre lignes, polices

Pourquoi ?

Création d'une page web HTML : Limitations

Page HTML sans CSS

Affichage

```
1 <html>
2   <head> </head>
3   <body>
4     <h1>Exemple de code HTML</h1>
5     <h2>Premier sous-titre</h2>
6     <p>Voici un paragraphe<br>
7       et bla bla bla bla
8       bla<br> bla bla bla</p>
9     <h2>Deuxième sous-titre</h2>
10    <p>Voici un deuxième
11      paragraphe<br>
12      et rebla bla bla</p>
13  </body>
14 </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Création d'une page web HTML : Limitations

Page HTML sans CSS

Affichage

```
1 <html>
2   <head> </head>
3   <body>
4     <h1>Exemple de code HTML</h1>
5     <h2>Premier sous-titre</h2>
6     <p>Voici un paragraphe<br>
7       et bla bla bla bla
8       bla<br> bla bla bla</p>
9     <h2>Deuxième sous-titre</h2>
10    <p>Voici un deuxième
11      paragraphe<br>
12      et rebla bla bla</p>
13  </body>
14 </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Création d'une page web HTML : Limitations

Page HTML sans CSS

Affichage

```
1 <html>
2   <head> </head>
3   <body>
4     <h1>Exemple de code HTML</h1>
5     <h2>Premier sous-titre</h2>
6     <p>Voici un paragraphe<br>
7       et bla bla bla bla
8       bla<br> bla bla bla</p>
9     <h2>Deuxième sous-titre</h2>
10    <p>Voici un deuxième
11      paragraphe<br>
12      et rebla bla bla</p>
13  </body>
14 </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code HTML (idem que le précédent)

```
1 <html>
2   <head>
3     <link rel="stylesheet "
4           type="text/css"
5           href="mon_css.css">
6   </head>
7   <body>
8     ...
9   </body>
10  </html>
```

Affichage

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

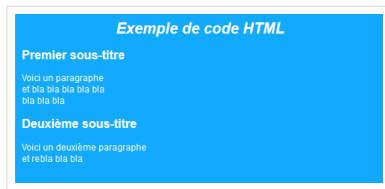
Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code HTML (idem que le précédent)

```
1 <html>
2   <head>
3     <link rel="stylesheet "
4           type="text/css"
5           href="mon_css.css">
6   </head>
7   <body>
8     ...
9   </body>
10  </html>
```

Affichage



Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code HTML (idem que le précédent)

Affichage

```
1 <html>
2   <head>
3     <link rel="stylesheet "
4           type="text/css "
5           href="mon_css.css">
6   </head>
7   <body>
8     ...
9   </body>
10  </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code CSS

```
1 body{
2   color:white;
3   background-color:#11aaff;
4   font-family:sans-serif;
5 }
6 p{
7   font-size:12px;
8 }
9 h1{
10  font-size:20px;
11  font-style:italic;
12  font-weight:bold;
13  text-align:center;
14 }
15 h2{
16  font-size:16px;
17  font-weight:bold;
18 }
```

Affichage

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code CSS

```
1 body{
2   color:white;
3   background-color:#11aaff;
4   font-family:sans-serif;
5 }
6 p{
7   font-size:12px;
8 }
9 h1{
10  font-size:20px;
11  font-style:italic;
12  font-weight:bold;
13  text-align:center;
14 }
15 h2{
16  font-size:16px;
17  font-weight:bold;
18 }
```

Affichage

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Solution : Page HTML avec feuille de style CSS

Code CSS

```
1 body{
2   color:white;
3   background-color:#11aaff;
4   font-family:sans-serif;
5 }
6 p{
7   font-size:12px;
8 }
9 h1{
10  font-size:20px;
11  font-style:italic;
12  font-weight:bold;
13  text-align:center;
14 }
15 h2{
16  font-size:16px;
17  font-weight:bold;
18 }
```

Affichage

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : code HTML avec feuille de style CSS et boîtes

Code HTML : ajout des blocs Affichage

`<div>`

```
1 <html>
2   <head>
3     <link rel="stylesheet"
4           type="text/css"
5           href="mon_css2.css">
6   </head>
7   <body>
8     <header>
9       <h1>Exemple de code HTML<
10        /h1>
11     </header>
12     <section>
13       <h2>Premier sous-titre</
14        h2>
15     </section>
16 </body>
</html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : code HTML avec feuille de style CSS et boîtes

Code HTML : ajout des blocs

Affichage

`<div>`

```
1 <html>
2   <head>
3     <link rel="stylesheet"
4           type="text/css"
5           href="mon_css2.css">
6   </head>
7   <body>
8     <header>
9       <h1>Exemple de code HTML<
10        /h1>
11     </header>
12     <section>
13       <h2>Premier sous-titre</
14        h2>
15     </section>
16   </body>
17 </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : code HTML avec feuille de style CSS et boîtes

Code HTML : ajout des blocs **Affichage**

`<div>`

```
1 <html>
2   <head>
3     <link rel="stylesheet"
4           type="text/css"
5           href="mon_css2.css">
6   </head>
7   <body>
8     <header>
9       <h1>Exemple de code HTML<
10        /h1>
11     </header>
12     <section>
13       <h2>Premier sous-titre</
14        h2>
15     </section>
16   </body>
17 </html>
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : Page HTML avec feuille de style CSS et boîtes

code CSS : style des boîtes <div>

```
1  ...
2  header{
3      padding:5px;
4      margin:1px;
5      width:300px;
6      background-color:#2288FF;
7      height:60px;
8      float:left;
9      text-align:center;
10 }
11 section{
12     padding:5px;
13     margin:1px;
14     width:300px;
15     background-color:#66ccFF;
16     float:left;
17     clear:both;
18 }
```

Affichage

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : Page HTML avec feuille de style CSS et boîtes

code CSS : style des boîtes <div>

Affichage

```
1  ...
2  header{
3      padding:5px;
4      margin:1px;
5      width:300px;
6      background-color:#2288FF;
7      height:60px;
8      float:left;
9      text-align:center;
10 }
11 section{
12     padding:5px;
13     margin:1px;
14     width:300px;
15     background-color:#66ccFF;
16     float:left;
17     clear:both;
18 }
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Amélioration : Page HTML avec feuille de style CSS et boîtes

code CSS : style des boîtes <div>

Affichage

```
1  ...
2  header{
3      padding:5px;
4      margin:1px;
5      width:300px;
6      background-color:#2288FF;
7      height:60px;
8      float:left;
9      text-align:center;
10 }
11 section{
12     padding:5px;
13     margin:1px;
14     width:300px;
15     background-color:#66ccFF;
16     float:left;
17     clear:both;
18 }
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Un autre exemple de style css pour la même page

Affichage

```
1 body{
2   font-family:cursive;
3 }
4 p{
5   font-size:12px;
6 }
7 h1{
8   font-size:24px;
9   font-style:italic;
10  font-weight:bold;
11  border:solid 2px grey;
12  padding: 5px;
13 }
14 h2{
15   font-size:16px;
16   font-weight:bold;
17   margin:20px;
18 }
19 header{
20   padding:5px;
21   margin:10px;
22   width:300px;
23   height:40px;
24   float:left;
25   text-align:center;
26 }
27 section{
28   padding:5px;
29   margin:10px;
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Un autre exemple de style css pour la même page

Affichage

```
1 body{
2   font-family:cursive;
3 }
4 p{
5   font-size:12px;
6 }
7 h1{
8   font-size:24px;
9   font-style:italic;
10  font-weight:bold;
11  border:solid 2px grey;
12  padding: 5px;
13 }
14 h2{
15   font-size:16px;
16   font-weight:bold;
17   margin:20px;
18 }
19 header{
20   padding:5px;
21   margin:10px;
22   width:300px;
23   height:40px;
24   float:left;
25   text-align:center;
26 }
27 section{
28   padding:5px;
29   margin:10px;
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

Pourquoi ?

Un autre exemple de style css pour la même page

Affichage

```
1 body{
2   font-family:cursive;
3 }
4 p{
5   font-size:12px;
6 }
7 h1{
8   font-size:24px;
9   font-style:italic;
10  font-weight:bold;
11  border:solid 2px grey;
12  padding: 5px;
13 }
14 h2{
15   font-size:16px;
16   font-weight:bold;
17   margin:20px;
18 }
19 header{
20   padding:5px;
21   margin:10px;
22   width:300px;
23   height:40px;
24   float:left;
25   text-align:center;
26 }
27 section{
28   padding:5px;
29   margin:10px;
```

Exemple de code HTML

Premier sous-titre

Voici un paragraphe
et bla bla bla bla bla
bla bla bla

Deuxième sous-titre

Voici un deuxième paragraphe
et rebla bla bla

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Principes d'IHM

- Séparation du contenu (HTML) et de la mise en forme (CSS).
 - Modifier facilement l'aspect d'un site sans en modifier le contenu
 - Cohésion de la présentation tout au long du site
 - Réduire le temps de chargement des pages
 - Correction de certains écueils d'HTML
 - Contrôle des polices, de la distance entre les lignes, des marges et des indentations
- Moins de code, mieux structuré → sites plus faciles à maintenir et à créer.

Créer un style CSS

Fichiers textes

- Éditeurs de css
Ex : *top Style 3* gratuit!!!

Créer un style sur un élément

Définir les propriétés des éléments HTML

```
selecteur {  
    propriété1 : valeur1 ;  
    propriété2 : valeur2 ;  
    propriété3 : valeur3 ;  
}
```

Sélecteur : le nom d'un élément HTML sur lequel on veut mettre un style

Propriété : ex : la couleur, la fonte, les bordures etc.

Ex :

```
1 p {  
2   color : red;  
3   font-size : 12px;  
4 }
```


Où spécifier les styles ?

3 méthodes

Dans un fichier .css à part, lié dans l'entête <head>

```
1 | <link rel="stylesheet" href="monstyle.css">
```

Dans un fichier .html, dans l'entête <head>

```
1 | <style type="text/css">  
2 |   h1{color:green;}  
3 | </style>
```

Dans n'importe quelle balise du code html

```
1 | <element style="...">  
2 |  
3 | <li style="color:red;">Voici une liste rouge</li>
```

Où spécifier les styles ?

3 méthodes

Dans un fichier .css à part, lié dans l'entête <head>

```
1 | <link rel="stylesheet" href="monstyle.css">
```

Dans un fichier .html, dans l'entête <head>

```
1 | <style type="text/css">  
2 |   h1{color:green;}  
3 | </style>
```

Dans n'importe quelle balise du code html

```
1 | <element style="...">  
2 |  
3 | <li style="color:red;">Voici une liste rouge</li>
```

Où spécifier les styles ?

3 méthodes

Dans un fichier .css à part, lié dans l'entête <head>

```
1 | <link rel="stylesheet" href="monstyle.css">
```

Dans un fichier .html, dans l'entête <head>

```
1 | <style type="text/css">  
2 |   h1{color:green;}  
3 | </style>
```

Dans n'importe quelle balise du code html

```
1 | <element style="...">  
2 |  
3 | <li style="color:red;">Voici une liste rouge</li>
```

Les commentaires dans les CSS

- Entre `/* */`
- `<!--` et `-->` acceptés (commentaire HTML), permet de cacher le CSS pour un navigateur qui ne les comprend pas!

Prise en charge des erreurs : **aucune!**

- Propriétés inconnues
- Valeurs invalides (unité!)

Les commentaires dans les CSS

- Entre `/* */`
- `<!--` et `-->` acceptés (commentaire HTML), permet de cacher le CSS pour un navigateur qui ne les comprend pas!

Prise en charge des erreurs : **aucune!**

- Propriétés inconnues
- Valeurs invalides (unité!)

Les commentaires dans les CSS

- Entre `/* */`
- `<!--` et `-->` acceptés (commentaire HTML), permet de cacher le CSS pour un navigateur qui ne les comprend pas !

Prise en charge des erreurs : **aucune !**

- Propriétés inconnues
- Valeurs invalides (unité !)

Les commentaires dans les CSS

- Entre `/* */`
- `<!--` et `-->` acceptés (commentaire HTML), permet de cacher le CSS pour un navigateur qui ne les comprend pas !

Prise en charge des erreurs : **aucune** !

- **Propriétés inconnues**
- Valeurs invalides (unité !)

Les commentaires dans les CSS

- Entre `/* */`
- `<!--` et `-->` acceptés (commentaire HTML), permet de cacher le CSS pour un navigateur qui ne les comprend pas !

Prise en charge des erreurs : **aucune** !

- Propriétés inconnues
- Valeurs invalides (unité !)

Compatibilité html5/css3

Des sites listent les compatibilités de html5/css3 sur les différents navigateurs

# CSS3 Multiple backgrounds - Candidate Recommendation									
Method of using multiple images as a background						Usage stats:		Global	
						Support:		86.62%	
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
								2.2	
						3.2		2.3	
						4.0-4.1		3.0	
						4.2-4.3		4.0	
	8.0	19.0	25.0						
	9.0	20.0	26.0	5.1		5.0-5.1		4.1	7.0
Current	10.0	21.0	27.0	6.0	12.1	6.0	5.0-7.0	4.2	10.0
Near future	11.0	22.0	28.0						
Farther future		23.0							
<div><div></div> = Supported <div></div> = Not supported <div></div> = Partially supported <div></div> = Support unknown</div>									
									itHub

Source <http://caniuse.com/>

Plan

Plan

Les couleurs

Codes de Couleurs

Nom	RGB HEX	RGB INTEGER
aqua	#00FFFF	rgb(0,255,255)
Black	#000000	rgb(0,0,0)
blue	#0000FF	rgb(0,0,255)
fuchsia	#FF00FF	rgb(255,0,255)
gray	#808080	rgb(128,128,128)
green	#008000	rgb(0,128,0)
lime	#00FF00	rgb(0,255,0)
maroon	#800000	rgb(128,0,0)
navy	#000080	rgb(0,0,128)
olive	#808000	rgb(128,128,0)
purple	#800080	rgb(128,0,128)
red	#FF0000	rgb(255,0,0)
silver	#C0C0C0	rgb(192,192,192)
teal	#008080	rgb(0,128,128)
white	#FFFFFF	rgb(255,255,255)
yellow	#FFFF00	rgb(255,255,0)




white	
silver	
gray	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

Toute combinaison de chiffres entre #000000 et #FFFFFF est une couleur

Les couleurs

Codes de Couleurs

Nom	RGB HEX	RGB INTEGER
aqua	#00FFFF	rgb(0,255,255)
Black	#000000	rgb(0,0,0)
blue	#0000FF	rgb(0,0,255)
fuchsia	#FF00FF	rgb(255,0,255)
gray	#808080	rgb(128,128,128)
green	#008000	rgb(0,128,0)
lime	#00FF00	rgb(0,255,0)
maroon	#800000	rgb(128,0,0)
navy	#000080	rgb(0,0,128)
olive	#808000	rgb(128,128,0)
purple	#800080	rgb(128,0,128)
red	#FF0000	rgb(255,0,0)
silver	#C0C0C0	rgb(192,192,192)
teal	#008080	rgb(0,128,128)
white	#FFFFFF	rgb(255,255,255)
yellow	#FFFF00	rgb(255,255,0)






white	
silver	
gray	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

Toute combinaison de chiffres entre #000000 et #FFFFFF est une couleur

Les couleurs

Codes de Couleurs

Nom	RGB HEX	RGB INTEGER
aqua	#00FFFF	rgb(0,255,255)
Black	#000000	rgb(0,0,0)
blue	#0000FF	rgb(0,0,255)
fuchsia	#FF00FF	rgb(255,0,255)
gray	#808080	rgb(128,128,128)
green	#008000	rgb(0,128,0)
lime	#00FF00	rgb(0,255,0)
maroon	#800000	rgb(128,0,0)
navy	#000080	rgb(0,0,128)
olive	#808000	rgb(128,128,0)
purple	#800080	rgb(128,0,128)
red	#FF0000	rgb(255,0,0)
silver	#C0C0C0	rgb(192,192,192)
teal	#008080	rgb(0,128,128)
white	#FFFFFF	rgb(255,255,255)
yellow	#FFFF00	rgb(255,255,0)

white	
silver	
gray	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

Toute combinaison de chiffres entre #000000 et #FFFFFF est une couleur

La propriété color

color

Décrit la couleur du texte d'un élément.

S'applique à tous les éléments (p, body, span, div etc.)

Ex :

```
1 | p {  
2 |   color: red;  
3 | }  
4 | body {  
5 |   color: rgb(255,0,0);  
6 | }
```

Les propriétés d'arrière-plan

background-color

Pour choisir la couleur d'arrière-plan d'un élément

background-image

Pour spécifier l'image d'arrière-plan d'un élément

```
1 p {  
2   background-color: yellow;  
3 }
```

```
1 p {  
2   background-image: url(  
3     shadecks.jpg);  
}
```

Texte avec un fond coloré



Les propriétés d'arrière-plan

background-color

Pour choisir la couleur d'arrière-plan d'un élément

background-image

Pour spécifier l'image d'arrière-plan d'un élément

```
1 p {  
2   background-color: yellow;  
3 }
```

```
1 p {  
2   background-image: url(  
3     shadecks.jpg);  
}
```

Texte avec un fond coloré



Les propriétés d'arrière-plan

background-color

Pour choisir la couleur d'arrière-plan d'un élément

background-image

Pour spécifier l'image d'arrière-plan d'un élément

```
1 p {  
2   background-color: yellow;  
3 }
```

```
1 p {  
2   background-image: url(  
3     shadocks.jpg );  
}
```

Texte avec un fond coloré



Les propriétés d'arrière-plan

background-color

Pour choisir la couleur d'arrière-plan d'un élément

background-image

Pour spécifier l'image d'arrière-plan d'un élément

```
1 p {  
2   background-color: yellow ;  
3 }
```

```
1 p {  
2   background-image: url(  
3     shadocks.jpg ) ;  
}
```

Texte avec un fond coloré



Les propriétés d'arrière-plan

background-color

Pour choisir la couleur d'arrière-plan d'un élément

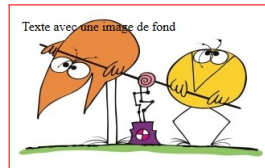
background-image

Pour spécifier l'image d'arrière-plan d'un élément

```
1 p {  
2   background-color: yellow ;  
3 }
```

```
1 p {  
2   background-image: url (  
3     shadocks.jpg ) ;  
}
```

Texte avec un fond coloré



Les fontes (polices de caractères)

font-size

permet d'indiquer la taille de la fonte

- taille absolue : en px, cm, mm, pt
- taille relative : en % ou en em

ex : `font-size: 16px;`

Les fontes (polices de caractères)

font-size

permet d'indiquer la taille de la fonte

- **taille absolue** : en **px**, **cm**, **mm**, **pt**
- **taille relative** : en % ou en em

ex : `font-size: 16px;`

Les fontes (polices de caractères)

font-size

permet d'indiquer la taille de la fonte

- taille absolue : en px, cm, mm, pt
- taille relative : en % ou en em

ex : `font-size: 16px;`

Attributs possibles pour les caractères

Attributs de fontes

- `font-family` : par exemple Times, Arial, Verdana, serif
- `font-style` : normal | italic
- `font-variant` : normal | small-caps
- `font-weight` : normal | bold " bolder " ou " lighter "

Normal

Italique

PETITES CAPITALES

gras

Attributs possibles pour les caractères

Attributs de fontes

- **font-family** : par exemple Times, Arial, Verdana, serif
- font-style : normal | italic
- font-variant : normal | small-caps
- font-weight : normal | bold " bolder " ou " lighter "

Normal

Italique

PETITES CAPITALES

gras

Attributs possibles pour les caractères

Attributs de fontes

- `font-family` : par exemple Times, Arial, Verdana, serif
- `font-style` : `normal` | `italic`
- `font-variant` : `normal` | `small-caps`
- `font-weight` : `normal` | `bold` " `bolder` " ou " `lighter` "

Normal

Italique

PETITES CAPITALES

gras

Attributs possibles pour les caractères

Attributs de fontes

- `font-family` : par exemple Times, Arial, Verdana, serif
- `font-style` : `normal` | `italic`
- `font-variant` : `normal` | `small-caps`
- `font-weight` : `normal` | `bold` " `bolder` " ou " `lighter` "

Normal

Italique

PETITES CAPITALES

gras

Attributs possibles pour les caractères

Attributs de fontes

- `font-family` : par exemple Times, Arial, Verdana, serif
- `font-style` : `normal` | `italic`
- `font-variant` : `normal` | `small-caps`
- `font-weight` : `normal` | `bold` " `bolder` " ou " `lighter` "

Normal

Italique

PETITES CAPITALES

gras

Attributs possibles pour les textes

text-decoration

`text-decoration` : `overline` | `underline` | `line-through` (rayé) | `blink` | `none` (par défaut).

surligné

souligné

~~rayé~~

clignotant

Attributs possibles pour les textes

text-align

`text-align : left | right | center | justify`

alignement à gauche

alignement à droite

alignement centré

alignement justifié

`vertical-align : top | middle | bottom`

généralement pour les cellules de tableaux (selon navigateur)

Exemple de style

Ex : balise strong stylée

Style css :

```
1 strong {  
2   font-family: Arial , sans-serif ;  
3   font-size: 20px ;  
4   font-weight: bold ;  
5   text-decoration: underline ;  
6   font-variant: small-caps ;  
7   color: #AA0055 ;  
8 }
```

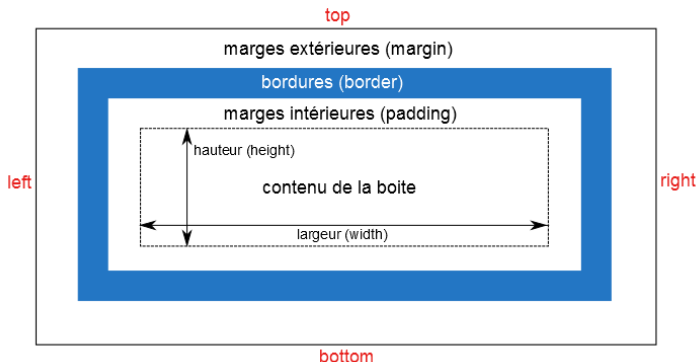
Code html :

```
1 Pour faire <strong> un style important</strong>
```

Pour faire UN STYLE IMPORTANT

Les bordures et les marges

- Les éléments de type bloc peuvent avoir des marges et des bordures.



Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- `border-style` et `border-[top, right, bottom, left]-style`

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- **border-color** et **border-[top, right, bottom, left]-color**
ex : **border-left-color: red;**
- **border-width** et **border-[top, right, bottom, left]-width**
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- **border-style** et **border-[top, right, bottom, left]-style**

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- `border-style` et `border-[top, right, bottom, left]-style`

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- **`border-style` et `border-[top, right, bottom, left]-style`**

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- `border-style` et `border-[top, right, bottom, left]-style`

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- `border-style` et `border-[top, right, bottom, left]-style`

```
1 p {  
2   border-style: solid ;  
3   border-width: thick ;  
4   border-color: green ;  
5 }  
6 ou  
7 p {  
8   border: solid thick green ;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

Voici un texte entouré

et un autre

Les propriétés de bordures

border

- `border-color` et `border-[top, right, bottom, left]-color`
ex : `border-left-color: red;`
- `border-width` et `border-[top, right, bottom, left]-width`
valeurs possibles : `thin` | `medium` | `thick` | une longueur (px, cm, etc.)
- `border-style` et `border-[top, right, bottom, left]-style`

```
1 p {  
2   border-style: solid;  
3   border-width: thick;  
4   border-color: green;  
5 }  
6 ou  
7 p {  
8   border: solid thick green;  
9 }
```

```
1 <p>Voici un texte entouré</p>  
2 <p>et un autre</p>
```

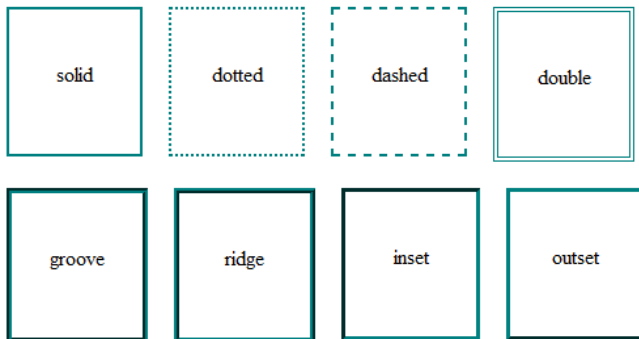
Voici un texte entouré

et un autre

Les propriétés de bordures

Les types de bordures :

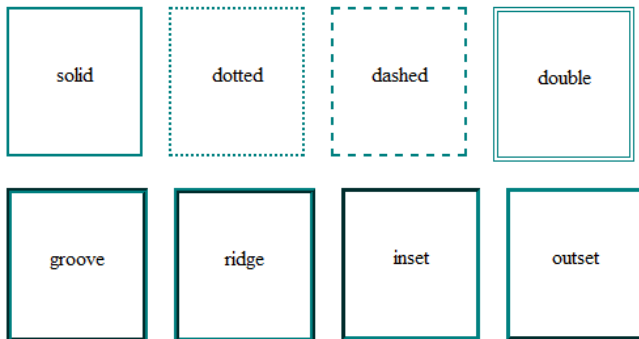
- **none** : pas de bordure (par défaut);
- et les autres ...



Les propriétés de bordures

Les types de bordures :

- none : pas de bordure (par défaut) ;
- et les autres ...



Les propriétés des marges


margin

- margin-top, margin-right, margin-bottom, margin-left
- valeur : une longueur | un pourcentage (de la largeur de la boîte) | auto (calcul) | inherit

Ex : sans marge

```
1 | <p>un paragraphe sans marge</p>
```

```
1 | p {  
2 |   background-color: lightgrey ;  
3 | }
```



un paragraphe sans marge

Les propriétés des marges


margin

- **margin-top, margin-right, margin-bottom, margin-left**
- valeur : une longueur | un pourcentage (de la largeur de la boîte) | auto (calcul) | inherit

Ex : sans marge

```
1 | <p>un paragraphe sans marge</p>
```

```
1 | p {  
2 |   background-color: lightgrey ;  
3 | }
```



un paragraphe sans marge

Les propriétés des marges


margin

- margin-top, margin-right, margin-bottom, margin-left
- valeur : une longueur | un pourcentage (de la largeur de la boîte) | auto (calcul) | inherit

Ex : sans marge

```
1 | <p>un paragraphe sans marge</p>
```

```
1 | p {  
2 |   background-color: lightgrey ;  
3 | }
```



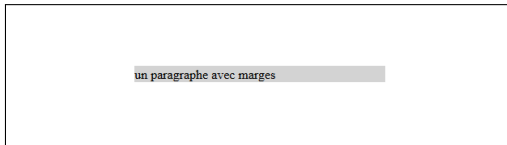
un paragraphe sans marge

Les propriétés des marges

Autre exemple : avec marges

```
1  p {  
2    background-color: lightgrey ;  
3    margin-top: 50px ;  
4    margin-bottom: 50px ;  
5    margin-right: 150px ;  
6    margin-left: 150px ;  
7  }  
8  ou  
9  p {  
10   background-color: lightgrey ;  
11   margin: 50px 150px 50px 150px ;  
12 }
```

```
1  <p>un paragraphe avec marges</p>
```



idem pour le padding (padding-top, padding-right, padding-bottom, padding-left)

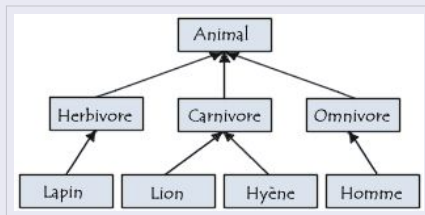
Plan

Plan

Notion d'héritage

Héritage

- **Au sens général** : transmission de quelque chose, d'une génération à une autre
- **Au sens informatique** : permet de transmettre les caractéristiques de certains "objets" (parent) vers un autre (enfant) qui en dérive
 - permet d'organiser les objets en structure hiérarchique

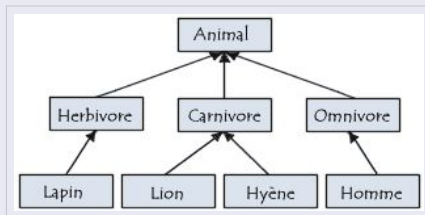


- **Au sens des CSS** : mécanisme qui permet, lors de la déclaration d'un nouveau style, d'y inclure les caractéristiques d'un autre

Notion d'héritage

Héritage

- **Au sens général** : transmission de quelque chose, d'une génération à une autre
- **Au sens informatique** : permet de transmettre les caractéristiques de certains "objets" (parent) vers un autre (enfant) qui en dérive
 - permet d'organiser les objets en structure hiérarchique

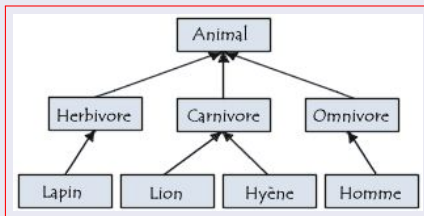


- **Au sens des CSS** : mécanisme qui permet, lors de la déclaration d'un nouveau style, d'y inclure les caractéristiques d'un autre

Notion d'héritage

Héritage

- **Au sens général** : transmission de quelque chose, d'une génération à une autre
- **Au sens informatique** : permet de transmettre les caractéristiques de certains "objets" (parent) vers un autre (enfant) qui en dérive
 - permet d'organiser les objets en structure hiérarchique

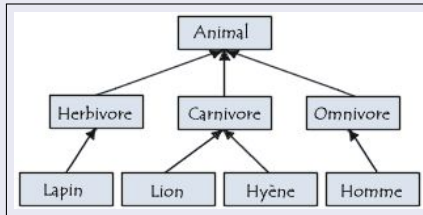


- **Au sens des CSS** : mécanisme qui permet, lors de la déclaration d'un nouveau style, d'y inclure les caractéristiques d'un autre

Notion d'héritage

Héritage

- **Au sens général** : transmission de quelque chose, d'une génération à une autre
- **Au sens informatique** : permet de transmettre les caractéristiques de certains "objets" (parent) vers un autre (enfant) qui en dérive
 - permet d'organiser les objets en structure hiérarchique

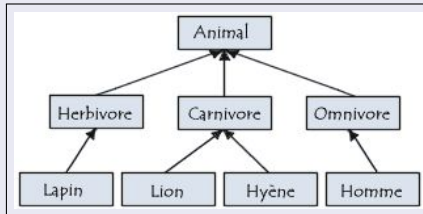


- **Au sens des CSS** : mécanisme qui permet, lors de la déclaration d'un nouveau style, d'y inclure les caractéristiques d'un autre

Notion d'héritage

Héritage

- **Au sens général** : transmission de quelque chose, d'une génération à une autre
- **Au sens informatique** : permet de transmettre les caractéristiques de certains "objets" (parent) vers un autre (enfant) qui en dérive
 - permet d'organiser les objets en structure hiérarchique



- **Au sens des CSS** : mécanisme qui permet, lors de la déclaration d'un nouveau style, d'y inclure les caractéristiques d'un autre

Notion d'héritage

- Par défaut, un style appliqué à un élément html sera également appliqué aux éléments qu'il contient (enfants)

Ex :

- Style css :

```
1 body{color:blue;}
```

- Code html :

```
1 <body>
2   voici une page
3   <h1>avec un titre</h1>
4   <p>voici un paragraphe</p>
5   <p>et un <em>autre</em></p>
6 </body>
```

- Le texte dans le corps (body) est en bleu y compris dans les balises h1, p, etc.

Notion d'héritage

- Par défaut, un style appliqué à un élément html sera également appliqué aux éléments qu'il contient (enfants)

Ex :

- Style css :

```
1 | body{color:blue;}
```

- Code html :

```
1 | <body>
2 |   voici une page
3 |   <h1>avec un titre</h1>
4 |   <p>voici un paragraphe</p>
5 |   <p>et un <em>autre</em></p>
6 | </body>
```

- Le texte dans le corps (body) est en bleu y compris dans les balises h1, p, etc.

Notion d'héritage

- Par défaut, un style appliqué à un élément html sera également appliqué aux éléments qu'il contient (enfants)

Ex :

- Style css :

```
1 | body{color:blue;}
```

- Code html :

```
1 | <body>
2 |   voici une page
3 |   <h1>avec un titre</h1>
4 |   <p>voici un paragraphe</p>
5 |   <p>et un <em>autre</em></p>
6 | </body>
```

- Le texte dans le corps (body) est en bleu y compris dans les balises h1, p, etc.

Notion d'héritage

- Par défaut, un style appliqué à un élément html sera également appliqué aux éléments qu'il contient (enfants)

Ex :

- Style css :

```
1 body{color:blue;}
```

- Code html :

```
1 <body>
2   voici une page
3   <h1>avec un titre</h1>
4   <p>voici un paragraphe</p>
5   <p>et un <em>autre</em></p>
6 </body>
```

- Le texte dans le corps (body) est en bleu y compris dans les balises h1, p, etc.

Regroupement d'éléments

- Rappel : élément = balise HTML
 - ex : p, span, div, body, h1 etc.

Regroupement d'éléments

- Ex : Les titres de niveaux 1, 2 et 3 sont écrits en rouge

```
1 h1, h2, h3 {  
2   color: red;  
3 }
```

- Ex : Les paragraphes et titres h1 ont une marge de 10px

```
1 p, h1 {  
2   margin: 10px;  
3 }
```

Regroupement d'éléments

- Rappel : élément = balise HTML
 - ex : p, span, div, body, h1 etc.

Regroupement d'éléments

- Ex : Les titres de niveaux 1, 2 et 3 sont écrits en rouge

```
1 h1, h2, h3 {  
2   color: red;  
3 }
```

- Ex : Les paragraphes et titres h1 ont une marge de 10px

```
1 p, h1 {  
2   margin: 10px;  
3 }
```

Regroupement d'éléments

- Rappel : élément = balise HTML
 - ex : p, span, div, body, h1 etc.

Regroupement d'éléments

- Ex : Les titres de niveaux 1, 2 et 3 sont écrits en rouge

```
1 | h1, h2, h3 {  
2 |     color: red;  
3 | }
```

- Ex : Les paragraphes et titres h1 ont une marge de 10px

```
1 | p, h1 {  
2 |     margin: 10px;  
3 | }
```

Regroupement d'éléments

- Rappel : élément = balise HTML
 - ex : p, span, div, body, h1 etc.

Regroupement d'éléments

- Ex : Les titres de niveaux 1, 2 et 3 sont écrits en rouge

```
1 | h1, h2, h3 {  
2 |     color: red;  
3 | }
```

- Ex : Les paragraphes et titres h1 ont une marge de 10px

```
1 | p, h1 {  
2 |     margin: 10px;  
3 | }
```

Regroupement d'éléments

- Rappel : élément = balise HTML
 - ex : p, span, div, body, h1 etc.

Regroupement d'éléments

- Ex : Les titres de niveaux 1, 2 et 3 sont écrits en rouge

```
1 | h1, h2, h3 {  
2 |     color: red;  
3 | }
```

- Ex : Les paragraphes et titres h1 ont une marge de 10px

```
1 | p, h1 {  
2 |     margin: 10px;  
3 | }
```

Sélecteurs contextuels

Sélecteurs contextuel

Mettre un *style particulier* sur une balise qui se trouve *dans une autre balise*

Ex : Tous les `span` dans des `p` apparaitront en rouge souligné
Style css :

```
1 | p span {  
2 |   color: red ;  
3 |   text-decoration: underline ;  
4 | }
```

Code html :

```
1 | <p>blabla et <span>bla</span></p>  
2 | <span>bla et bla</span>
```

Affichage :

blabla et bla

bla et bla

Sélecteurs contextuels

Autre ex : tous les liens a contenus dans un paragraphe p seront gris

Style css :

```
1 p a {  
2   color: gray;  
3 }
```

Code html :

```
1 <p> mon paragraphe avec <a>un lien</a> surtout !</p>
```


Classes de style

Une classe

- permettent d'affecter des styles différents à des balises identiques
- Dans le style css

```
1 élément.maclasse {  
2   propriété1: valeur1;  
3   propriété2: valeur2;  
4 }
```

- Dans le code html

```
1 <élément class="maclasse">  
2   un texte avec du style  
3 </élément>
```

Classes de style

Une classe

- permettent d'affecter des styles différents à des balises identiques
- Dans le style css

```
1 élément.maclasse {  
2     propriété1: valeur1;  
3     propriété2: valeur2;  
4 }
```

- Dans le code html

```
1 <élément class="maclasse">  
2     un texte avec du style  
3 </élément>
```

Classes de style

Une classe

- permettent d'affecter des styles différents à des balises identiques
- Dans le style css

```
1 | élément.maclasse {  
2 |   propriété1: valeur1 ;  
3 |   propriété2: valeur2 ;  
4 | }
```

- Dans le code html

```
1 | <élément class="maclasse">  
2 |   un texte avec du style  
3 | </élément>
```

Classes de style

Une classe

- permettent d'affecter des styles différents à des balises identiques
- Dans le style css

```
1 | élément.maclasse {  
2 |     propriété1: valeur1;  
3 |     propriété2: valeur2;  
4 | }
```

- Dans le code html

```
1 | <élément class="maclasse">  
2 |     un texte avec du style  
3 | </élément>
```

Les classes

Ex : Paragraphes p de classe special écrits en bleu centré

```
1 p.paragraphebleu{  
2   color: blue;  
3   text-align: center;  
4 }
```

```
1 <p> bla bla bla et bla bla </p>  
2 <p class="paragraphebleu"> bla bla bla et bla bla </p>
```

bla bla bla et bla bla

bla bla bla et bla bla

Classes de style

ex : style css

```
1 p.monstyle {  
2     font-family: Arial;  
3     font-size: 20pt;  
4 }
```

Code html

```
1 <p class="monstyle">un paragraphe stylé</p>  
2 <p>un paragraphe sans style</p>
```

Affichage :

un paragraphe stylé

un paragraphe sans style

Classes de style

Plusieurs éléments peuvent utiliser une même classe de style :

ex :

```
1 h1.rouge, a.rouge {  
2   color: red;  
3 }
```

```
1 <h1>Titre</h1>  
2 <h1 class="rouge">Titre rouge</h1>  
3 <a>Lien</a>  
4 <a class="rouge">Lien rouge</a>
```

Titre

Titre rouge

Lien **Lien rouge**

Classes de style

Un élément HTML peut avoir plusieurs classes :

ex :

```
1 p.rouge {  
2   color: red;  
3 }  
4 p.important {  
5   font-weight: bold;  
6 }
```

```
1 <p class="rouge important">Paragraphe important  
   rouge</a>  
2 <p>Paragraphe sans style</a>
```

Paragraphe important rouge

Paragraphe sans style

Classe universelle

Classe universelle

- On ne précise pas de balise
- classe pouvant être utilisée dans n'importe quelle balise!

Ex : Eléments de classe `special2` écrits en blanc sur fond noir

```
1 .fondnoir {  
2     color:white;  
3     background-color:black;  
4     padding:10px;  
5 }  
6 .bleu {  
7     color: blue;  
8 }
```

Affichage :

un titre

un paragraphe d'introduction

un sous-titre

```
1 <h1 class="fondnoir">un titre</h1>  
2 <p class="bleu">un paragraphe d'introduction</p>  
3 <h2 class="fondnoir">un sous-titre</h2>
```

Classe universelle

Classe universelle

- On ne précise pas de balise
- classe pouvant être utilisée dans n'importe quelle balise!

Ex : Eléments de classe `special2` écrits en blanc sur fond noir

```
1 .fondnoir {  
2     color:white;  
3     background-color:black;  
4     padding:10px;  
5 }  
6 .bleu {  
7     color: blue;  
8 }
```

Affichage :

un titre

un paragraphe d'introduction

un sous-titre

```
1 <h1 class="fondnoir">un titre</h1>  
2 <p class="bleu">un paragraphe d'introduction</p>  
3 <h2 class="fondnoir">un sous-titre</h2>
```

Classe universelle

Classe universelle

- On ne précise pas de balise
- classe pouvant être utilisée dans n'importe quelle balise!

Ex : Eléments de classe `special2` écrits en blanc sur fond noir

```
1 .fondnoir {  
2     color:white;  
3     background-color:black;  
4     padding:10px;  
5 }  
6 .bleu {  
7     color: blue;  
8 }
```

Affichage :

un titre

un paragraphe d'introduction

un sous-titre

```
1 <h1 class="fondnoir">un titre</h1>  
2 <p class="bleu">un paragraphe d'introduction</p>  
3 <h2 class="fondnoir">un sous-titre</h2>
```

Classe universelle

Classe universelle

- On ne précise pas de balise
- classe pouvant être utilisée dans n'importe quelle balise!

Ex : Eléments de classe `special2` écrits en blanc sur fond noir

```
1 .fondnoir {  
2   color:white;  
3   background-color:black;  
4   padding:10px;  
5 }  
6 .bleu {  
7   color: blue;  
8 }
```

Affichage :

un titre

un paragraphe d'introduction

un sous-titre

```
1 <h1 class="fondnoir">un titre</h1>  
2 <p class="bleu">un paragraphe d'introduction</p>  
3 <h2 class="fondnoir">un sous-titre</h2>
```

Classe universelle

Classe universelle

- On ne précise pas de balise
- classe pouvant être utilisée dans n'importe quelle balise!

Ex : Eléments de classe `special2` écrits en blanc sur fond noir

```
1 .fondnoir {  
2   color:white;  
3   background-color:black;  
4   padding:10px;  
5 }  
6 .bleu {  
7   color: blue;  
8 }
```

Affichage :

un titre

un paragraphe d'introduction

un sous-titre

```
1 <h1 class="fondnoir">un titre</h1>  
2 <p class="bleu">un paragraphe d'introduction</p>  
3 <h2 class="fondnoir">un sous-titre</h2>
```

Les identifiants

id

- identifiant unique au sein de la page
- fonctionne exactement de la même manière que class

MAIS ne peut être utilisé qu'une fois dans le code d'une page html

- servent notamment à localiser des éléments HTML grâce à javascript

```
1 <element id="monid">  
2   Style pour cet élément  
3 </element>
```

Les identifiants

id

- identifiant unique au sein de la page
- fonctionne exactement de la même manière que class

MAIS ne peut être utilisé qu'une fois dans le code d'une page html

- servent notamment à localiser des éléments HTML grâce à javascript

```
1 | <element id="monid">  
2 |   Style pour cet élément  
3 | </element>
```

Les identifiants

id

- identifiant unique au sein de la page
- fonctionne exactement de la même manière que class

MAIS ne peut être utilisé qu'une fois dans le code d'une page html

- servent notamment à localiser des éléments HTML grâce à javascript

```
1 | <element id="monid">  
2 |   Style pour cet élément  
3 | </element>
```


Les identifiants

id

- identifiant unique au sein de la page
- fonctionne exactement de la même manière que class

MAIS ne peut être utilisé qu'une fois dans le code d'une page html

- servent notamment à localiser des éléments HTML grâce à javascript

```
1 | <element id="monid">  
2 |   Style pour cet élément  
3 | </element>
```

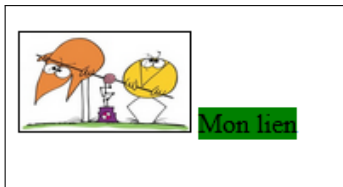
Les identifiants

Ex :

```
1 #logo {  
2     width:100px;  
3     border:1px solid;  
4 }  
5 #lien1 {  
6     background-color: green;  
7 }
```

```
1   
2 <a id="lien1">Mon lien</a>
```

Affichage :



Identifiant

Ex : apparaît sur un fond jaune et encadré en bleu

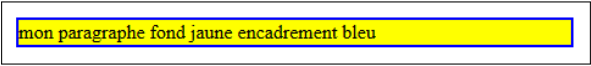
```
1 #fondjaune {  
2     background-color:yellow;  
3     border:blue 2px solid;  
4 }
```

Utilisation de ce style dans une page html :

Ex :

```
1 <p id="fondjaune">  
2     mon paragraphe fond jaune encadrement bleu  
3 </p>
```

Affichage :



mon paragraphe fond jaune encadrement bleu

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - En définissant une réaction à un événement
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - **En définissant une réaction à un événement**
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - En définissant une réaction à un événement
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - En définissant une réaction à un événement
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - En définissant une réaction à un événement
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes

- Permettent d'affiner le style appliqué à certaines balises
 - En définissant une réaction à un événement
 - Ou bien à la position relative de la balise au sein des autres balises.
 - *Attention : le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes*
- Différents types de pseudo-classes
 - dynamiques, de lien, de langue, first-child, de page ...

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - `hover`
 - `focus`
 - `active`
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - `hover`
 - `focus`
 - `active`
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - **hover**
 - focus
 - active
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - hover
 - focus
 - active
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - hover
 - focus
 - **active**
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

- Modifient le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier)
- 3 types différents :
 - hover
 - focus
 - active
- *Attention : ne sont pas reconnues de la même façon par tous les navigateurs*

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```


Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- `hover` : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- `focus` : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- `active` : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes dynamiques

Les pseudo-classes dynamiques

- **hover** : Style lors d'un survol par le curseur de souris

ex :

```
1 | a:hover {font-decoration: underline;}
```

- **focus** : style lorsque le focus lui est donné (ex : lors d'un clic dans une case de formulaire)

ex :

```
1 | textarea:focus {color: #FF0000;}
```

- **active** : style lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche)

ex :

```
1 | a:active {color: #FF0000;}
```

Les pseudo-classes de lien

Les pseudo-classes de lien

- **link** : style des liens hypertextes n'ayant pas encore été consultés

```
1 | a:link {color: green;}
```

- **visited** : style des liens hypertextes que le client a déjà visité

```
1 | a:visited {color: red;}
```

Les pseudo-classes de lien

Les pseudo-classes de lien

- `link` : style des liens hypertextes n'ayant pas encore été consultés

```
1 | a:link {color: green;}
```

- `visited` : style des liens hypertextes que le client a déjà visité

```
1 | a:visited {color: red;}
```


Plan

Plan

Comportement des éléments HTML

Comportement inline

Les éléments se placent toujours les uns à côté des autres afin de rester dans le texte (ex : liens, images)

- seules les propriétés de mise en forme s'appliquent (police, couleurs)
- propriétés généralement héritées



Comportement des éléments HTML

Comportement inline

Les éléments se placent toujours les uns à côté des autres afin de rester dans le texte (ex : liens, images)

- seules les propriétés de mise en forme s'appliquent (police, couleurs)
- propriétés généralement héritées



Comportement des éléments HTML

Comportement inline

Les éléments se placent toujours les uns à côté des autres afin de rester dans le texte (ex : liens, images)

- seules les propriétés de mise en forme s'appliquent (police, couleurs)
- propriétés généralement héritées

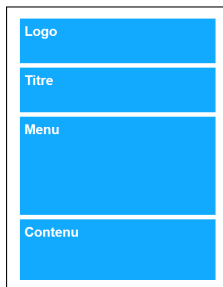


Comportement des éléments HTML

Comportement bloc

Les éléments occupent toute la largeur possible avec saut de ligne entre chacun (ex : `div`, `p`, `h1`, `li` etc.).

- on peut définir des marges et bordures

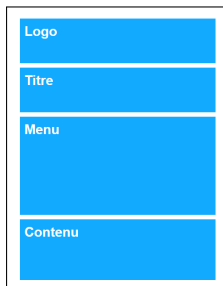


Comportement des éléments HTML

Comportement bloc

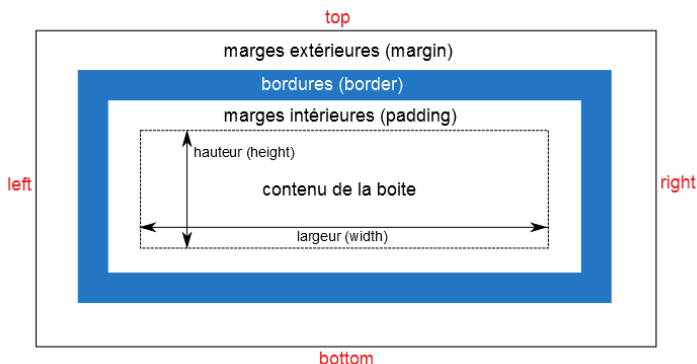
Les éléments occupent toute la largeur possible avec saut de ligne entre chacun (ex : div, p, h1, li etc.).

- on peut définir des marges et bordures



Description d'une boîte

Une boîte a une marge extérieure (margin), un marge intérieur (padding) ...



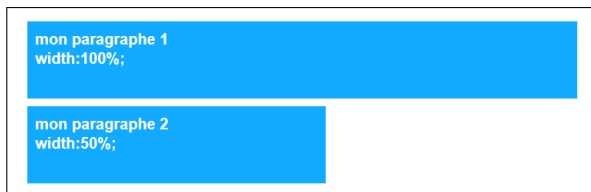
Le style des boîtes

Propriétés qu'on peut préciser sur un élément bloc !

width

largeur en pourcentage (%) ou pixels (px)

Ex : largeur en %



height

hauteur en pourcentage (%) ou pixels (px)

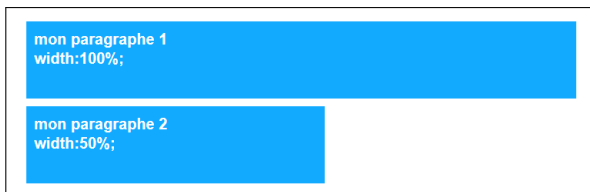
Le style des boîtes

Propriétés qu'on peut préciser sur un élément bloc !

width

largeur en pourcentage (%) ou pixels (px)

Ex : largeur en %



height

hauteur en pourcentage (%) ou pixels (px)

Le style des boîtes

border

Permet de spécifier le style de la bordure de la boîte (déjà vu !) `border-top`, `border-right`, `border-bottom`, `border-left`
`border-style` : `solid`, `dashed`, ..., `border-color` ...
ex : `border : 10px solid #00FFAA ;`

margin

marges extérieures, en pixels (px)

padding

marges intérieures

background-color

Définit la couleur interne, valable sur le contenu, les marges intérieures, jusqu'au bordures

Le style des boîtes

border

Permet de spécifier le style de la bordure de la boîte (déjà vu !) `border-top`, `border-right`, `border-bottom`, `border-left`
`border-style` : `solid`, `dashed`, ..., `border-color` ...
ex : `border : 10px solid #00FFAA ;`

margin

marges extérieures, en pixels (px)

padding

marges intérieures

background-color

Définit la couleur interne, valable sur le contenu, les marges intérieures, jusqu'au bordures

Le style des boîtes

border

Permet de spécifier le style de la bordure de la boîte (déjà vu !) `border-top`, `border-right`, `border-bottom`, `border-left`
`border-style` : `solid`, `dashed`, ..., `border-color` ...
ex : `border : 10px solid #00FFAA ;`

margin

marges extérieures, en pixels (px)

padding

marges intérieures

background-color

Définit la couleur interne, valable sur le contenu, les marges intérieures, jusqu'au bordures

Le style des boîtes

border

Permet de spécifier le style de la bordure de la boîte (déjà vu !) `border-top`, `border-right`, `border-bottom`, `border-left`
`border-style` : `solid`, `dashed`, ..., `border-color` ...
ex : `border : 10px solid #00FFAA ;`

margin

marges extérieures, en pixels (px)

padding

marges intérieures

background-color

Définit la couleur interne, valable sur le contenu, les marges intérieures, jusqu'au bordures

Propriété overflow

overflow

- `overflow: visible | hidden | scroll | auto ;`
- agit lorsque le contenu est plus grand que l'espace disponible pour l'élément

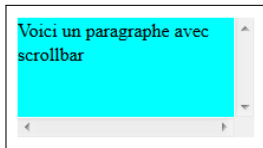
Ex :

```
1 div {  
2     width: 200px;  
3     height: 100px;  
4     overflow: scroll;  
5 }
```

Code html :

```
1 <div>Voici un paragraphe avec scrollbar</div>
```

Affichage :



Propriété overflow

overflow

- **overflow: visible | hidden | scroll | auto ;**
- agit lorsque le contenu est plus grand que l'espace disponible pour l'élément

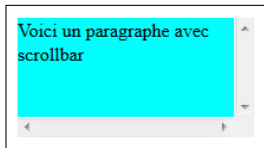
Ex :

```
1 div {  
2     width: 200px;  
3     height: 100px;  
4     overflow: scroll;  
5 }
```

Code html :

```
1 <div>Voici un paragraphe avec scrollbar</div>
```

Affichage :



Propriété overflow

overflow

- `overflow: visible | hidden | scroll | auto ;`
- agit lorsque le contenu est plus grand que l'espace disponible pour l'élément

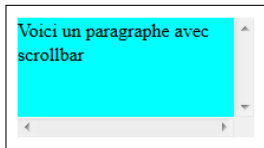
Ex :

```
1 div {  
2     width: 200px;  
3     height: 100px;  
4     overflow: scroll;  
5 }
```

Code html :

```
1 <div>Voici un paragraphe avec scrollbar</div>
```

Affichage :



Propriété display

display

Permet de forcer le comportement d'un élément

- 1 élément *inline* peut avoir un rendu *bloc* ou 1 élément bloc un rendu *inline*

Propriété display

display

Permet de forcer le comportement d'un élément

- 1 élément *inline* peut avoir un rendu *bloc* ou 1 élément *bloc* un rendu *inline*

Propriété display

Ex : bloc \rightarrow inline

```
1 | li {  
2 |     display: inline;  
3 | }
```

Code html :

```
1 | <ul>  
2 |   <li><a href="page1.html">HTML</a></li>  
3 |   <li><a href="page2.html">CSS</a></li>  
4 |   <li><a href="page3.html">JavaScript</a></li>  
5 | </ul>
```

Affichage sans le css :

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

Affichage avec le css :

[HTML](#) [CSS](#) [JavaScript](#)

Propriété display

Ex : inline → bloc

```
1 | strong {  
2 |     display: block;  
3 | }
```

Code html

```
1 | Voici <strong> une balise </strong> <strong> en ligne </  
   | strong> qui s'affiche en <strong> bloc </strong>
```

Affichage sans le css :

Voici **une balise en ligne** qui s'affiche en **bloc**

Affichage avec le css :

Voici
**une balise
en ligne**
qui s'affiche en
bloc

Balises universelles bloc/inline

2 balises universelles = sans aucune signification particulière !

- ` ` : balise inline à placer dans paragraphe de texte, pour sélectionner certains mots uniquement.
- `<div> </div>` : balise bloc, qui entoure un bloc de texte. Utilisée pour créer des boîtes !

Balises universelles bloc/inline

2 balises universelles = sans aucune signification particulière !

- ` ` : balise inline à placer dans paragraphe de texte, pour sélectionner certains mots uniquement.
- `<div> </div>` : balise bloc, qui entoure un bloc de texte. Utilisée pour créer des boîtes !

Positionnement des boites

Propriété position

- `relative` : par rapport à sa position calculée
- `absolute` : par rapport au document ou à un parent

Propriétés à éviter car difficile à gérer complètement !

Coordonnées de positionnement : `top`, `bottom`, `left`, `right` en pixels (px)

Meilleure solution

définir correctement les agencements à l'aide de marges et d'éléments flottants, pour mieux s'adapter au poste client

Positionnement des boites

Propriété position

- **relative** : par rapport à sa position calculée
- **absolute** : par rapport au document ou à un parent

Propriétés à éviter car difficile à gérer complètement !

Coordonnées de positionnement : top, bottom, left, right en pixels (px)

Meilleure solution

définir correctement les agencements à l'aide de marges et d'éléments flottants, pour mieux s'adapter au poste client

Positionnement des boites

Propriété position

- `relative` : par rapport à sa position calculée
- `absolute` : par rapport au document ou à un parent

Propriétés à éviter car difficile à gérer complètement !

Coordonnées de positionnement : `top`, `bottom`, `left`, `right` en pixels (px)

Meilleure solution

définir correctement les agencements à l'aide de marges et d'éléments flottants, pour mieux s'adapter au poste client

Positionnement des boites

Propriété position

- `relative` : par rapport à sa position calculée
- `absolute` : par rapport au document ou à un parent

Propriétés à éviter car difficile à gérer complètement !

Coordonnées de positionnement : `top`, `bottom`, `left`, `right` en pixels (px)

Meilleure solution

définir correctement les agencements à l'aide de marges et d'éléments flottants, pour mieux s'adapter au poste client

Positionnement des boites

Ex : Position absolue

```
1 h2 {  
2   position: absolute;  
3   left: 100px;  
4   top: 150px;  
5 }
```

Code html :

```
1 <h2>mon titre</h2>  
2 <p>avec la position absolue, un élément peut être placé  
   n'importe où sur la page</p>
```

Affichage sans le css :

mon titre

avec la position absolue, un élément peut être placé n'importe où sur la page

Affichage avec le css :

avec la position absolue, un élément peut être placé n'importe où sur la page

mon titre

Positionnement des boites

Ex : Position absolue

```
1 h2 {  
2   position: absolute;  
3   left: 100px;  
4   top: 150px;  
5 }
```

Code html :

```
1 <h2>mon titre</h2>  
2 <p>avec la position absolue, un élément peut être placé  
   n'importe où sur la page</p>
```

Affichage sans le css :

mon titre

avec la position absolue, un élément peut être placé n'importe où sur la page

Affichage avec le css :

avec la position absolue, un élément peut être placé n'importe où sur la page

mon titre

Positionnement des boites

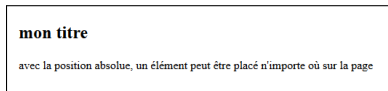
Ex : Position absolue

```
1 h2 {  
2   position: absolute;  
3   left: 100px;  
4   top: 150px;  
5 }
```

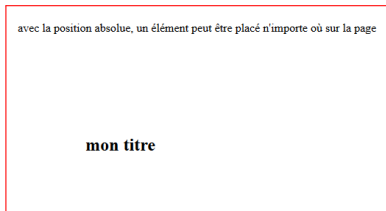
Code html :

```
1 <h2>mon titre</h2>  
2 <p>avec la position absolue, un élément peut être placé  
   n'importe où sur la page</p>
```

Affichage sans le css :



Affichage avec le css :



Dimension des pages

Principalement en fonction de la définition d'affichage.

- *largeur* : Choisir la plus petite largeur d'écran supportée par le plus grand nombre
- *Longueur* : ne pas dépasser 3 à 5 hauteurs d'écran.
- Autre solution : Détecter la résolution des visiteurs à l'aide d'un script javascript, et rediriger le visiteur vers une page de la bonne largeur.

Dimension des pages

Principalement en fonction de la définition d'affichage.

- *largeur* : Choisir la plus petite largeur d'écran supportée par le plus grand nombre
- *Longueur* : ne pas dépasser 3 à 5 hauteurs d'écran.
- Autre solution : Détecter la résolution des visiteurs à l'aide d'un script javascript, et rediriger le visiteur vers une page de la bonne largeur.

Dimension des pages

Principalement en fonction de la définition d'affichage.

- *largeur* : Choisir la plus petite largeur d'écran supportée par le plus grand nombre
- *Longueur* : ne pas dépasser 3 à 5 hauteurs d'écran.
- Autre solution : Détecter la résolution des visiteurs à l'aide d'un script javascript, et rediriger le visiteur vers une page de la bonne largeur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - **Contrôle de l'alignement des éléments**
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

Les grilles

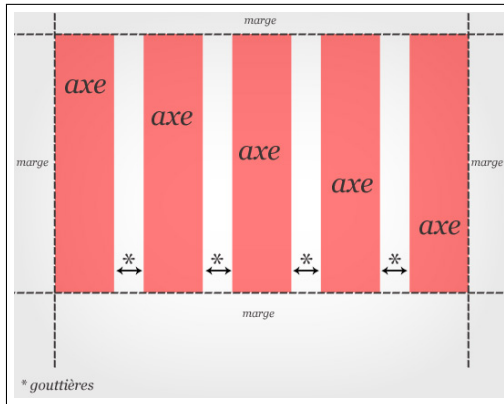
- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - **Contrôle des contenus qui se chevauchent**
 - le point de départ d'une grille est le conteneur.

Les grilles

- Grille = ensemble de lignes horizontales et verticales qui se croisent !
 - Les éléments sont placés sur la grille en fonction de ces rangées/lignes et colonnes
 - Hauteur et largeur des rangées et colonnes à taille fixe ou variable
 - Placement des éléments grâce au numéro ou nom de la ligne/colonnes
 - Il y a un algorithme de placement automatique pour les éléments non placés
 - Création de colonnes/rangées automatique pour du contenu
 - Contrôle de l'alignement des éléments
 - Contrôle des contenus qui se chevauchent
 - le point de départ d'une grille est le conteneur.

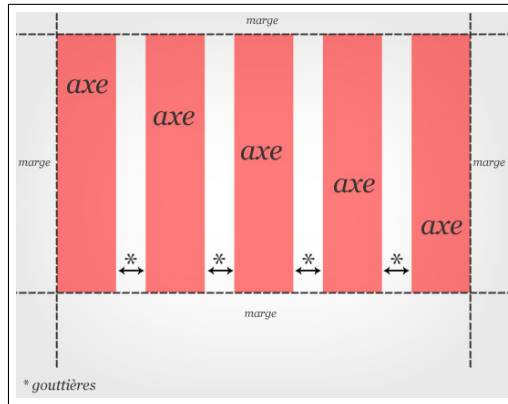
Les grilles

- Les grilles comportent généralement :
 - de repères (axes) verticaux et/ou horizontaux, séparés par des gouttières
 - de marges externes (pour les éventuels bords de la fenêtre) et internes (de chaque axe).



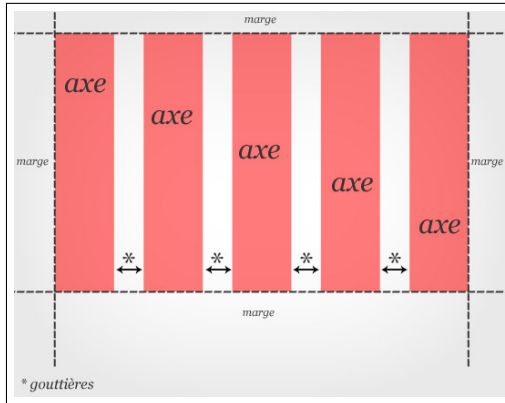
Les grilles

- Les grilles comportent généralement :
 - de repères (axes) verticaux et/ou horizontaux, séparés par des gouttières
 - de marges externes (pour les éventuels bords de la fenêtre) et internes (de chaque axe).



Les grilles

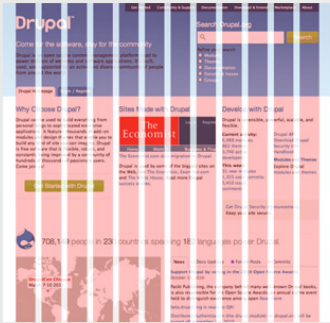
- Les grilles comportent généralement :
 - de repères (axes) verticaux et/ou horizontaux, séparés par des gouttières
 - de marges externes (pour les éventuels bords de la fenêtre) et internes (de chaque axe).



Les grilles

- Les grilles verticales

On peut choisir le nombre de colonnes à insérer dans une largeur fixe.

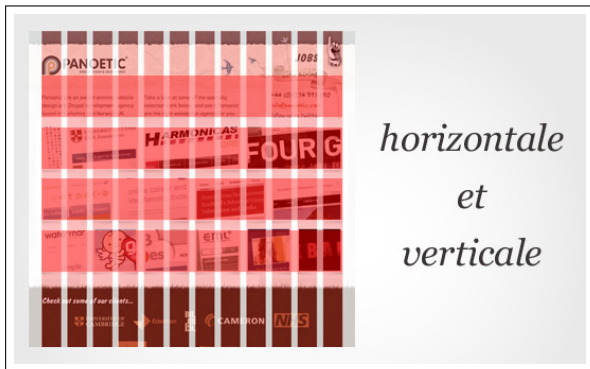


The screenshot displays the Drupal homepage, which is a classic example of a 12-column grid layout. The layout is divided into several vertical columns of varying widths, totaling 12 columns. The top navigation bar is a single row of 12 columns. The main content area is divided into three main sections: a left sidebar (3 columns), a main content area (6 columns), and a right sidebar (3 columns). The text '12 colonnes' is overlaid on the right side of the image.

12
colonnes

Les grilles

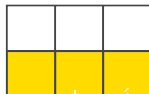
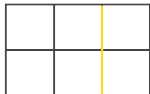
- Les grilles verticales et horizontales



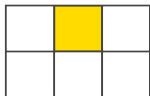
Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (lignesitué entre 2 lignes de grille de rangées)

Piste (ou plage) de grille : espace adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne



zone : espace entouré par 4 lignes de grille



Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (ligne de grille de rangées)



Piste (ou plage) de grille : espace situé entre 2 lignes de grille adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne



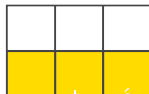
zone : espace entouré par 4 lignes de grille



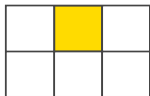
Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (ligne de grille de rangées)

Piste (ou plage) de grille : espace situé entre 2 lignes de grille adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne



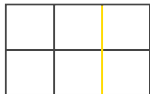
zone : espace entouré par 4 lignes de grille



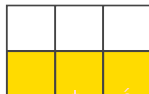
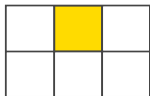
Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (lignesitué entre 2 lignes de grille de rangées)

Piste (ou plage) de grille : espace adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne



zone : espace entouré par 4 lignes de grille



Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (lignesitué entre 2 lignes de grille de rangées)

Piste (ou plage) de grille : espace adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne

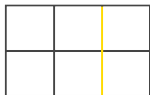


zone : espace entouré par 4 lignes de grille

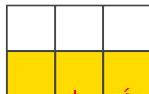


Les grilles

Ligne : verticale (ligne de grille de colonnes) ou horizontale (lignesitué entre 2 lignes de grille de rangées)



Piste (ou plage) de grille : espace adjacentes (ce sont les colonnes ou les rangées de la grille)



Cellule : espace situé entre 2 lignes de rangée et 2 lignes de colonne

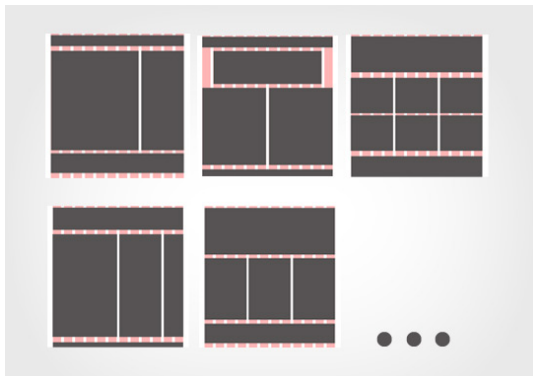


zone : espace entouré par 4 lignes de grille



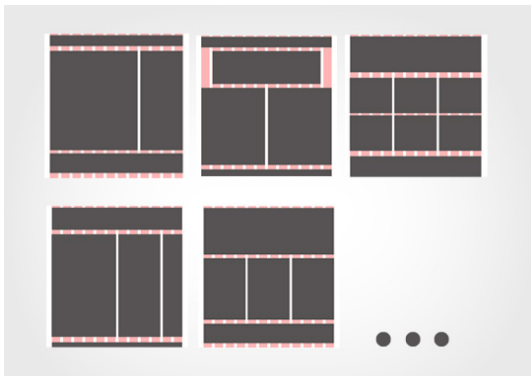
Les grilles

- Agencement des grilles (blocs)
 - Après avoir choisi 1 type de grille, il faut la "remplir" en disposant les blocs au bon endroit (images, textes, etc.).
 - Exemples de combinaisons :



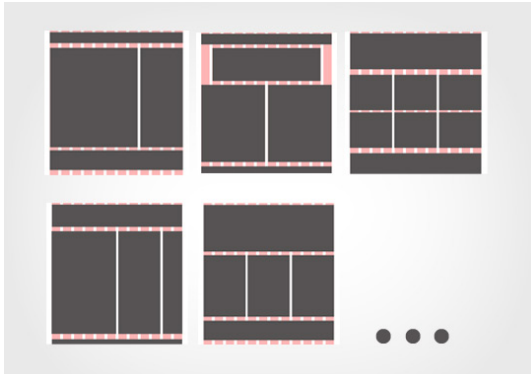
Les grilles

- Agencement des grilles (blocs)
 - Après avoir choisi 1 type de grille, il faut la "remplir" en disposant les blocs au bon endroit (images, textes, etc.).
 - Exemples de combinaisons :



Les grilles

- Agencement des grilles (blocs)
 - Après avoir choisi 1 type de grille, il faut la "remplir" en disposant les blocs au bon endroit (images, textes, etc.). ‘
 - Exemples de combinaisons :



Les grilles

● Exemple de page html :

```
1 <div class="wrapper">
2   <header>L 'en-tête</header>
3   <nav>
4     <ul>
5       <li><a href="">Titre 1</a></li>
6       <li><a href="">Titre 2</a></li>
7       <li><a href="">Titre 3</a></li>
8     </ul>
9   </nav>
10  <article>
11    <h1>Titre 1</h1>
12    <p>
13      bla bla bla ...
14    </p>
15  </article>
16  <aside>Infos 1</aside>
17  <aside>Infos 2</aside>
18  <aside>Pub</aside>
19  <footer>Pied de page</footer>
20 </div>
```


Les grilles

- Le conteneur (wrapper = "enveloppeur")

- est créé par la propriété `display:grid` ou `display:inline-grid` sur un élément

- Dans le css, `.wrapper` est déclaré comme conteneur de grille !

```
1 .wrapper {  
2   display: grid;  
3 }
```

- Affichage de l'exemple (avec l'inspecteur de grille de firefox) :

L'en-tête
<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3
Titre 1
bla bla bla ...
Infos 1
Infos 2
Pub
Pied de page

Les grilles

- Le conteneur (wrapper = "enveloppeur")
 - est créé par la propriété `display:grid` ou `display:inline-grid` sur un élément
 - Dans le css, `.wrapper` est déclaré comme conteneur de grille !

```
1 .wrapper {  
2   display: grid;  
3 }
```

- Affichage de l'exemple (avec l'inspecteur de grille de firefox) :

L'en-tête
<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3
Titre 1
bla bla bla ...
Infos 1
Infos 2
Pub
Pied de page

Les grilles

- Le conteneur (wrapper = "enveloppeur")
 - est créé par la propriété `display:grid` ou `display:inline-grid` sur un élément
 - Dans le css, `.wrapper` est déclaré comme conteneur de grille !

```
1 .wrapper {  
2   display: grid;  
3 }
```

- Affichage de l'exemple (avec l'inspecteur de grille de firefox) :

L'en-tête
<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3
Titre 1
bla bla bla ...
Infos 1
Infos 2
Pub
Pied de page

Les grilles

- Le conteneur (wrapper = "enveloppeur")
 - est créé par la propriété `display:grid` ou `display:inline-grid` sur un élément
 - Dans le css, `.wrapper` est déclaré comme conteneur de grille !

```
1 .wrapper {  
2   display: grid;  
3 }
```

- Affichage de l'exemple (avec l'inspecteur de grille de firefox) :

L'en-tête
<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3
Titre 1
bla bla bla ...
Infos 1
Infos 2
Pub
Pied de page

Les grilles

- Les pistes

- `grid-template-columns` et `grid-template-rows` pour définir des colonnes et des rangées

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 100px 200px 100px;  
4 }
```

Affichage avec 3 colonnes :

L'en-tête	<ul style="list-style-type: none">• <u>Titre 1</u>• <u>Titre 2</u>• <u>Titre 3</u>	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Les grilles

- Les pistes

- `grid-template-columns` et `grid-template-rows` pour définir des colonnes et des rangées

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 100px 200px 100px;  
4 }
```

Affichage avec 3 colonnes :

L'en-tête	<ul style="list-style-type: none">• <u>Titre 1</u>• <u>Titre 2</u>• <u>Titre 3</u>	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Les grilles

- L'unité fr : fr = fraction de l'espace disponible dans le conteneur de la grille, permet d'avoir des grilles flexibles en fonction de l'espace disponible
- Ex :

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4 }
```

L'en-tête	<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Les grilles

- L'unité fr : fr = fraction de l'espace disponible dans le conteneur de la grille, permet d'avoir des grilles flexibles en fonction de l'espace disponible
- Ex :

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4 }
```

L'en-tête	<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Les grilles

- **repeat()** : permet de répéter des pistes
grid-template-columns: 1fr 1fr 1fr; peut s'écrire
grid-template-columns: repeat(3, 1fr);
 - ex : 1 colonne de 20px de large, puis 6 fois une piste de 1fr, et 1 colonne de 20px de large
On peut utiliser des dimensions absolues et relatives en même temps !

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 20px repeat(6, 1fr) 20px;  
4 }
```

Les grilles

- **repeat()** : permet de répéter des pistes
grid-template-columns: 1fr 1fr 1fr; peut s'écrire
grid-template-columns: repeat(3, 1fr);
 - ex : 1 colonne de 20px de large, puis 6 fois une piste de 1fr, et 1 colonne de 20px de large
On peut utiliser des dimensions absolues et relatives en même temps !

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 20px repeat(6, 1fr) 20px;  
4 }
```

Les grilles

- `grid-auto-rows` et `grid-auto-columns` : Si un élément est placé en dehors de la grille définie avec `grid-template...`, la grille ajoute alors des colonnes ou rangées automatiquement
- `minmax()` : pour donner une taille minimum, qui s'agrandit pour s'adapter au contenu
- ex : rangées au min. 50px et au max. auto

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5 }
```

L'en-tête	<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pod d...		

Les grilles

- `grid-auto-rows` et `grid-auto-columns` : Si un élément est placé en dehors de la grille définie avec des `grid-template...`, la grille ajoute alors des colonnes ou rangées automatiquement
- `minmax()` : pour donner une taille minimum, qui s'agrandit pour s'adapter au contenu
- ex : rangées au min. 50px et au max. auto

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5 }
```

L'en-tête	<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Les grilles

- `grid-auto-rows` et `grid-auto-columns` : Si un élément est placé en dehors de la grille définie avec des `grid-template...`, la grille ajoute alors des colonnes ou rangées automatiquement
- `minmax()` : pour donner une taille minimum, qui s'agrandit pour s'adapter au contenu
- ex : rangées au min. 50px et au max. auto

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5 }
```

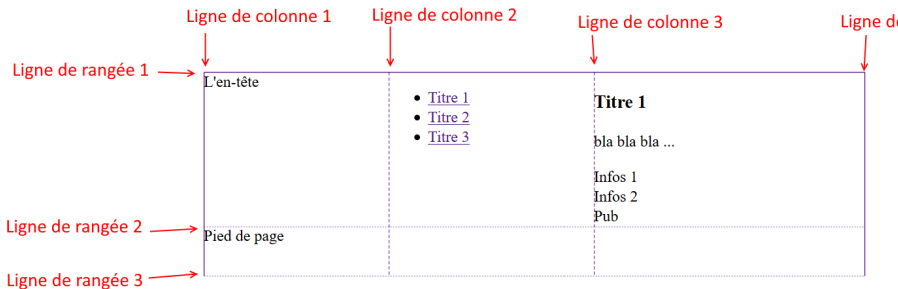
L'en-tête	<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	Titre 1 bla bla bla ...
Infos 1	Infos 2	Pub
Pied de page		

Exemple de code html

```
1  ...
2  <div class="wrapper">
3    <header class="tete">L'en-tête</header>
4    <nav class="menu">
5      <ul>
6        <li><a href="">Titre 1</a></li>
7        <li><a href="">Titre 2</a></li>
8        <li><a href="">Titre 3</a></li>
9      </ul>
10   </nav>
11   <section class="partiel">
12     <article class="art1">
13       <h1>Titre 1</h1>
14       <p>bla bla bla ... </p>
15     </article>
16     <aside class="info1">Infos 1</aside>
17     <aside class="info2">Infos 2</aside>
18     <aside class="pub">Pub</aside>
19   </section>
20   <footer class="pied">Pied de page</footer>
21 </div>
22 ...
```

Gérer le placement dans la grille

- Placement des zones en fonction des lignes de la grille avec `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`



Gérer le placement dans la grille

● ex : Placement de l'en-tête et du menu

```
1 .wrapper { ... }
2 .tete {
3   background-color:yellow;
4   grid-column-start: 2;
5   grid-column-end: 4;
6   grid-row-start: 1;
7 }
8 .menu {
9   background-color:lightgray;
10  grid-column-start: 1;
11  grid-row-start: 1;
12  grid-row-end: 3;
13 }
```

<ul style="list-style-type: none">• Titre 1• Titre 2• Titre 3	L'en-tête	
	Titre 1 bla bla bla . Infos 1 Infos 2 Pub	Pied de page

Gérer le placement dans la grille

- on ajoute le placement de la section et du pied de page

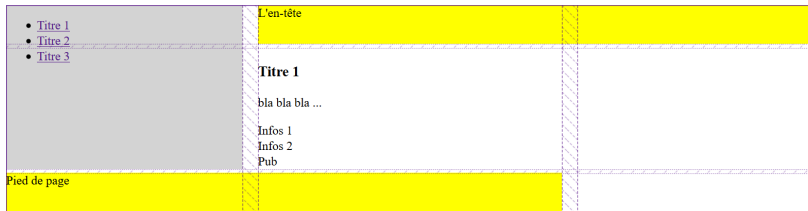
```
1  ...
2  .partie1 {
3      grid-column-start: 2;
4      grid-column-end: 4;
5      grid-row-start: 2;
6  }
7  .pied {
8      background-color: yellow;
9      grid-column-start: 1;
10     grid-column-end: 3;
11     grid-row-start: 3;
12     grid-row-end: 3;
13 }
```

<ul style="list-style-type: none">Titre 1Titre 2Titre 3	L'en-tête	
	Titre 1 bla bla bla . Infos 1 Infos 2 Pub	
	Pied de page	

Gérer le placement dans la grille

- Les gouttières = espacement entre les colonnes ou les rangées
- `grid-column-gap` et `grid-row-gap`

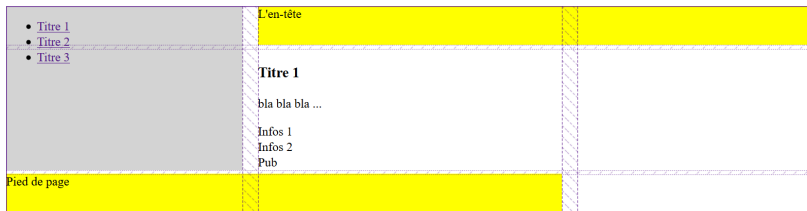
```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5   grid-column-gap: 20px;  
6   grid-row-gap: 5px;  
7 }
```



Gérer le placement dans la grille

- Les gouttières = espacement entre les colonnes ou les rangées
- `grid-column-gap` et `grid-row-gap`

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 2fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5   grid-column-gap: 20px;  
6   grid-row-gap: 5px;  
7 }
```

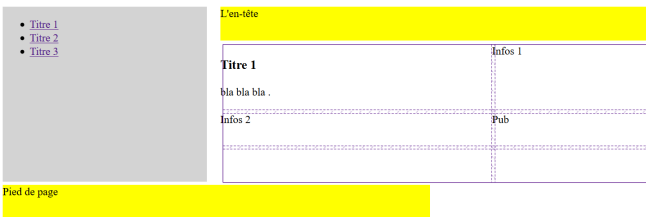


Gérer le placement dans la grille

- Grilles imbriquées

- 1 grille imbriquée n'est pas liée à la grille qui la contient
- Elle n'hérite pas des grid-gap ni des lignes de la grille parent donc les lignes ne s'alignent pas.

```
1 .partiel1 {  
2   ...  
3   display: grid;  
4   grid-template-columns: 3fr 1fr;  
5   grid-template-rows: 2fr 1fr 1fr;  
6   grid-gap: 5px;  
7 }  
8 ...
```

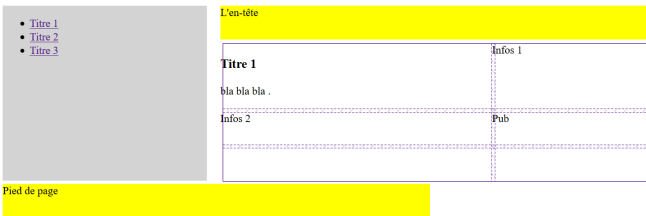


Gérer le placement dans la grille

- Grilles imbriquées

- 1 grille imbriquée n'est pas liée à la grille qui la contient
- Elle n'hérite pas des grid-gap ni des lignes de la grille parent donc les lignes ne s'alignent pas.

```
1 .partie1 {  
2   ...  
3   display: grid;  
4   grid-template-columns: 3fr 1fr;  
5   grid-template-rows: 2fr 1fr 1fr;  
6   grid-gap: 5px;  
7 }  
8 ...
```

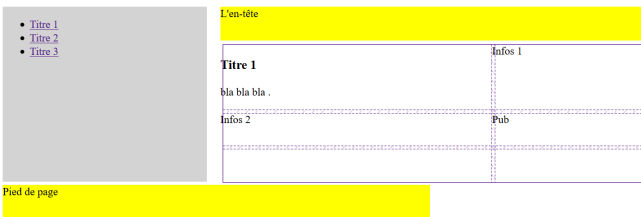


Gérer le placement dans la grille

- Grilles imbriquées

- 1 grille imbriquée n'est pas liée à la grille qui la contient
- Elle n'hérite pas des `grid-gap` ni des lignes de la grille parent donc les lignes ne s'alignent pas.

```
1 .partie1 {  
2   ...  
3   display: grid;  
4   grid-template-columns: 3fr 1fr;  
5   grid-template-rows: 2fr 1fr 1fr;  
6   grid-gap: 5px;  
7 }  
8 ...
```



Gérer le placement dans la grille

- Affichage final (avec la grille .partie1 apparente)

L'en-tête		
<u>Titre 1</u> <u>Titre 2</u> <u>Titre 3</u>		
Infos 1	Titre 1 bla bla bla ...	Pub
Infos 2		
Pied de page		

Gérer les zones de la grille

- Un autre moyen de placer les zones : `grid-area`
on donne un nom à chacune des zones, par exemple `tt` pour `tete`, `mn` pour `menu`, `pt1` pour `partie`, `ft` pour `pied`!

```
1 .tete {  
2     background-color:yellow;  
3     grid-area: tt;  
4 }  
5 .menu {  
6     background-color:lightgray;  
7     grid-area: mn;  
8 }  
9 .partie1 {  
10    grid-area: pt1;  
11    ...  
12 }  
13 .pied {  
14    background-color:yellow;  
15    grid-area: ft;  
16 }
```


Gérer le placement dans la grille

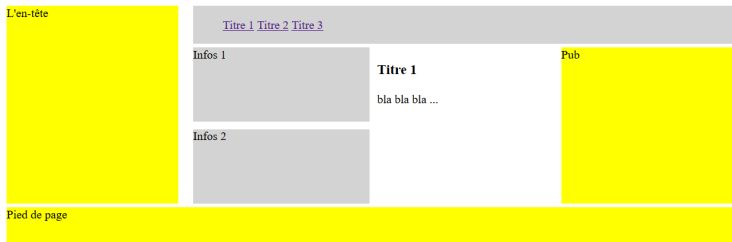
- Un autre moyen de placer les zones : grid-area
Puis on modifie le wrapper en indiquant où se trouve chaque zone dans les rangées et colonnes !
Ici, on a créé 8 colonnes et des rangées automatiques.

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;  
4   grid-auto-rows: minmax(50px, auto);  
5   grid-column-gap: 20px;  
6   grid-row-gap: 5px;  
7   grid-template-areas:  
8     "tt  tt  mn  mn  mn  mn  mn  mn"  
9     "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1 "  
10    "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1 "  
11    "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1 "  
12    "ft  ft  ft  ft  ft  ft  ft  ft";  
13 }
```



Gérer le placement dans la grille

- Résultat sans la grille



Gérer le placement dans la grille

- On peut utiliser les grilles pour créer des tableaux, agencer les images et le texte d'une manière spéciale etc.

Plan

Plan

Qu'est-ce que le RWD

Responsive Web Design (RWD) = Web adaptatif, conception d'une interface auto-adaptable.

Technologie CSS3 media queries

Extension de la règle `@media` pour adapter la mise en page du site suivant l'outil utilisé pour le consulter.

ex : ordinateur, tablette, smartphone, télévision, imprimante ...

- Permet d'utiliser des règles CSS différentes en fonction des caractéristiques du terminal de consultation

Qu'est-ce que le RWD

Responsive Web Design (RWD) = Web adaptatif, conception d'une interface auto-adaptable.

Technologie CSS3 media queries

Extension de la règle @media pour adapter la mise en page du site suivant l'outil utilisé pour le consulter.

ex : ordinateur, tablette, smartphone, télévision, imprimante ...

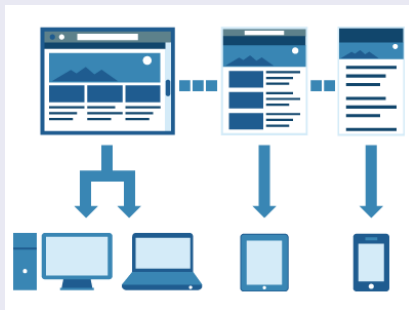
- Permet d'utiliser des règles CSS différentes en fonction des caractéristiques du terminal de consultation

Qu'est-ce que le RWD ?

Mobile First (le mobile en premier)

Construction d'une interface pour terminaux mobiles avant celle pour un ordinateur

- Contrainte + forte : faible espace d'affichage
=> épurer au maximum les éléments visibles par l'utilisateur final.

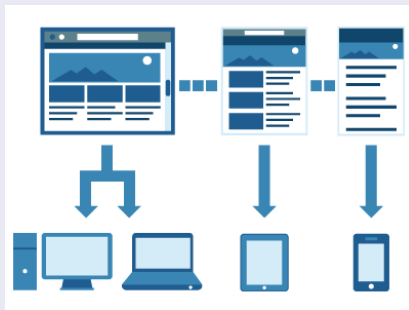


Qu'est-ce que le RWD ?

Mobile First (le mobile en premier)

Construction d'une interface pour terminaux mobiles avant celle pour un ordinateur

- **Contrainte + forte : faible espace d'affichage**
=> épurer au maximum les éléments visibles par l'utilisateur final.



Les media queries

- Utiliser la balise @media

- ex : si le media est une imprimante :

- ne pas imprimer les parties nav, aside et footer
- texte du corps de la page en noir

```
1 @media print {  
2     nav, aside, footer {  
3         display:none;  
4     }  
5     body {  
6         color:black;  
7     }  
8 }
```

Les media queries

- Utiliser la balise @media
- ex : si le media est une imprimante :
 - ne pas imprimer les parties nav, aside et footer
 - texte du corps de la page en noir

```
1 @media print {  
2   nav, aside, footer {  
3     display:none;  
4   }  
5   body {  
6     color:black;  
7   }  
8 }
```

Les media queries

- Utiliser la balise @media
- ex : si le media est une imprimante :
 - ne pas imprimer les parties nav, aside et footer
 - texte du corps de la page en noir

```
1 @media print {  
2     nav, aside, footer {  
3         display:none;  
4     }  
5     body {  
6         color:black;  
7     }  
8 }
```

Les media queries

- Utiliser la balise @media
- ex : si le media est une imprimante :
 - ne pas imprimer les parties nav, aside et footer
 - **texte du corps de la page en noir**

```
1 @media print {  
2     nav, aside, footer {  
3         display:none;  
4     }  
5     body {  
6         color:black;  
7     }  
8 }
```

Les media queries

Des opérateurs logiques

- and, only, not, "ou" obtenu en listant différentes media queries à la suite, séparées par des virgules

Ex : pour les écrans de largeur $<$ à 640 pixels (max-width : 640px)

Code html :

```
1 <link rel="stylesheet" media="screen and (max-width: 640px)" href="smallscreen.css" type="text/css"/>
```

Style css :

```
1 @media screen and (max-width: 640px) {  
2   .bloc {  
3     display: block;  
4     clear: both;  
5   }  
6 }
```

Les media queries

Des opérateurs logiques

- and, only, not, "ou" obtenu en listant différentes media queries à la suite, séparées par des virgules

Ex : pour les écrans de largeur $<$ à 640 pixels (max-width : 640px)

Code html :

```
1 <link rel="stylesheet" media="screen and (max-width: 640px)" href="smallscreen.css" type="text/css"/>
```

Style css :

```
1 @media screen and (max-width: 640px) {
2   .bloc {
3     display: block;
4     clear: both;
5   }
6 }
```

Les media queries

Autres exemples :

```
1  /* Sur tous types d'écran, quand la largeur de la
   *   fenêtre est comprise entre 1024px et 1280px */
2  @media all and (min-width: 1024px) and (max-width: 1280
   *   px) { ... }
3
4  /* Sur les téléviseurs */
5  @media tv { ... }
6
7  /* Sur tous types d'écrans orientés verticalement */
8  @media all and (orientation: portrait) { ... }
```


Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- **aspect-ratio** : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- **`height` : hauteur de la zone d'affichage (fenêtre).**
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- **`orientation` : orientation du périphérique (portrait ou paysage).**
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- **media** : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - **`print` : impression;**
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Fonctionnalités

peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - **`projection` : projecteur;**
 - `all` : tous les types d'écran.

Fonctionnalités

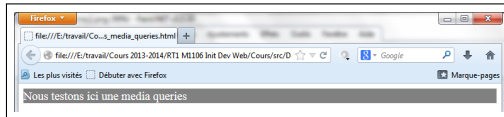
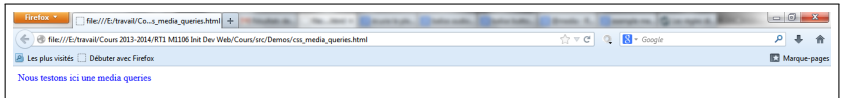
peuvent être préfixés par `min-` et `max-` pour définir des valeurs minimales ou maximales à respecter.

- `aspect-ratio` : ratio du périphérique de sortie (par exemple 16/9)
- `resolution` : résolution du périphérique (en dpi, dppx, ou dpcm)
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
 - `screen` : écran "classique";
 - `handheld` : périphérique mobile;
 - `print` : impression;
 - `tv` : télévision;
 - `projection` : projecteur;
 - `all` : tous les types d'écran.

Les media queries

Exemple :

```
1 p { color: blue; }  
2  
3 @media screen and (max-width: 1024px) {  
4   p { color: white;  
5       background-color: gray;  
6       font-size: 14pt; }  
7 }
```



Redéfinir une grille avec des media queries

- Adaptation facile pour différentes résolutions
- Redéfinition de la position des objets sur la grille et/ou de la grille elle-même
- Rappel du code html de l'exemple précédent :

```
1  ...
2  <body>
3  <div class="wrapper">
4    <header class="tete">L'en-tête</header>
5    <nav class="menu">
6      <ul>
7        <li><a href="#">Titre 1</a></li>
8        <li><a href="#">Titre 2</a></li>
9        <li><a href="#">Titre 3</a></li>
10     </ul>
11   </nav>
12   <section class="partiel">
13     <article class="art1">
14       <h1>Titre 1</h1>
15       <p>bla bla bla ... </p>
16     </article>
17     <aside class="info1">Infos 1</aside>
18     <aside class="info2">Infos 2</aside>
19     <aside class="pub">Pub</aside>
20   </section>
21   <footer class="pied">Pied de page</footer>
22 </div>
23  ...
```

Redéfinir une grille avec des media queries

- Adaptation facile pour différentes résolutions
- Redéfinition de la position des objets sur la grille et/ou de la grille elle-même
- Rappel du code html de l'exemple précédent :

```
1  ...
2  <body>
3  <div class="wrapper">
4    <header class="tete">L'en-tête</header>
5    <nav class="menu">
6      <ul>
7        <li><a href="#">Titre 1</a></li>
8        <li><a href="#">Titre 2</a></li>
9        <li><a href="#">Titre 3</a></li>
10     </ul>
11   </nav>
12   <section class="partie1">
13     <article class="art1">
14       <h1>Titre 1</h1>
15       <p>bla bla bla ... </p>
16     </article>
17     <aside class="info1">Infos 1</aside>
18     <aside class="info2">Infos 2</aside>
19     <aside class="pub">Pub</aside>
20   </section>
21   <footer class="pied">Pied de page</footer>
22 </div>
23  ...
```

Redéfinir une grille avec des media queries

- Adaptation facile pour différentes résolutions
- Redéfinition de la position des objets sur la grille et/ou de la grille elle-même
- Rappel du code html de l'exemple précédent :

```
1  ...
2  <body>
3  <div class="wrapper">
4    <header class="tete">L'en-tête</header>
5    <nav class="menu">
6      <ul>
7        <li><a href="">Titre 1</a></li>
8        <li><a href="">Titre 2</a></li>
9        <li><a href="">Titre 3</a></li>
10     </ul>
11   </nav>
12   <section class="partie1">
13     <article class="art1">
14       <h1>Titre 1</h1>
15       <p>bla bla bla ... </p>
16     </article>
17     <aside class="info1">Infos 1</aside>
18     <aside class="info2">Infos 2</aside>
19     <aside class="pub">Pub</aside>
20   </section>
21   <footer class="pied">Pied de page</footer>
22 </div>
23 ...
```


Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones `tete`, `menu`, `pied`
 - la "décoration" des zones `art1`, `info1`, `info2`, `pub`
 - les `grid-area` de la "grille" seront valables pour les 2 présentations
 - par contre, les `grid-area` de la "sous-grille" `.partie1` sont supprimés car ils ne sont pas communs

```
1 .tete {
2     background-color:yellow;
3     grid-area: tt;
4 }
5 .menu {
6     background-color:lightgray;
7     grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones tete, menu, pied
 - la "décoration" des zones art1, info1, info2, pub
 - les grid-area de la "grille" seront valables pour les 2 présentations
 - par contre, les grid-area de la "sous-grille" .partie1 sont supprimés car ils ne sont pas communs

```
1 .tete {
2   background-color:yellow;
3   grid-area: tt;
4 }
5 .menu {
6   background-color:lightgray;
7   grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones tete, menu, pied
 - la "décoration" des zones art1, info1, info2, pub
 - les grid-area de la "grille" seront valables pour les 2 présentations
 - par contre, les grid-area de la "sous-grille" .partie1 sont supprimés car ils ne sont pas communs

```
1 .tete {
2     background-color:yellow;
3     grid-area: tt;
4 }
5 .menu {
6     background-color:lightgray;
7     grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones tete, menu, pied
 - la "décoration" des zones art1, info1, info2, pub
 - les grid-area de la "grille" seront valables pour les 2 présentations
 - par contre, les grid-area de la "sous-grille" .partie1 sont supprimés car ils ne sont pas communs

```
1 .tete {
2     background-color:yellow;
3     grid-area: tt;
4 }
5 .menu {
6     background-color:lightgray;
7     grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones tete, menu, pied
 - la "décoration" des zones art1, info1, info2, pub
 - les grid-area de la "grille" seront valables pour les 2 présentations
 - par contre, les grid-area de la "sous-grille" .partie1 sont supprimés car ils ne sont pas communs

```
1 .tete {
2     background-color:yellow;
3     grid-area: tt;
4 }
5 .menu {
6     background-color:lightgray;
7     grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On souhaite avoir un style différent selon qu'on ait sur un écran "large" ou un écran "mobile" type smartphone
- On réfléchit d'abord à ce qui va être **commun** aux 2 types d'écrans!
 - les zones tete, menu, pied
 - la "décoration" des zones art1, info1, info2, pub
 - les grid-area de la "grille" seront valables pour les 2 présentations
 - par contre, les grid-area de la "sous-grille" .partie1 sont supprimés car ils ne sont pas communs

```
1 .tete {
2   background-color:yellow;
3   grid-area: tt;
4 }
5 .menu {
6   background-color:lightgray;
7   grid-area: mn;
8 }
9 .art1{ }
10 .info1{ background-color:lightgray; }
11 .info2{ background-color:lightgray; }
12 .pub{ background-color:yellow; }
13 .pied{ background-color:yellow; grid-area: ft; }
```

Redéfinir une grille avec des media queries

- On définit ensuite ce qui va être la règle à appliquer sur les écrans de taille supérieure à 500px.
 - On ajoute dans le css la règle : @media all and (min-width: 500px) ...
 - Cette règle contient le .wrapper et les .partie1, .art1, etc. qui seront différents selon l'écran

```
1 @media screen and (min-width: 500px){
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
5     grid-auto-rows: minmax(50px, auto);
6     grid-column-gap: 20px;
7     grid-row-gap: 5px;
8     grid-template-areas:
9       "tt  tt  mn  mn  mn  mn  mn  mn"
10      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
11      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
12      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
13      "ft  ft  ft  ft  ft  ft  ft  ft";
14  }
15  .partie1 {
16    grid-area: pt1;
17    display: grid;
18    grid-auto-columns: minmax(100px, auto);
19    grid-auto-rows: minmax(100px, auto);
20    grid-gap: 10px;
21  }
22  .art1 { grid-area: 1 / 2 / 3 / 3; }
```

Redéfinir une grille avec des media queries

- On définit ensuite ce qui va être la règle à appliquer sur les écrans de taille supérieure à 500px.
 - On ajoute dans le css la règle : `@media all and (min-width: 500px) ...`
 - Cette règle contient le `.wrapper` et les `.partie1`, `.art1`, etc. qui seront différents selon l'écran

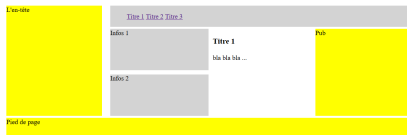
```
1 @media screen and (min-width: 500px){
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
5     grid-auto-rows: minmax(50px, auto);
6     grid-column-gap: 20px;
7     grid-row-gap: 5px;
8     grid-template-areas:
9       "tt  tt  mn  mn  mn  mn  mn  mn"
10      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
11      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
12      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
13      "ft  ft  ft  ft  ft  ft  ft  ft";
14  }
15  .partie1 {
16    grid-area: pt1;
17    display: grid;
18    grid-auto-columns: minmax(100px, auto);
19    grid-auto-rows: minmax(100px, auto);
20    grid-gap: 10px;
21  }
22  .art1 { grid-area: 1 / 2 / 3 / 3; }
```


Redéfinir une grille avec des media queries

- On définit ensuite ce qui va être la règle à appliquer sur les écrans de taille supérieure à 500px.
 - On ajoute dans le css la règle : @media all and (min-width: 500px) ...
 - Cette règle contient le .wrapper et les .partie1, .art1, etc. qui seront différents selon l'écran

```
1 @media screen and (min-width: 500px){
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
5     grid-auto-rows: minmax(50px, auto);
6     grid-column-gap: 20px;
7     grid-row-gap: 5px;
8     grid-template-areas:
9       "tt  tt  mn  mn  mn  mn  mn  mn"
10      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
11      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
12      "tt  tt  pt1 pt1 pt1 pt1 pt1 pt1"
13      "ft  ft  ft  ft  ft  ft  ft  ft";
14  }
15  .partie1 {
16    grid-area: pt1;
17    display: grid;
18    grid-auto-columns: minmax(100px, auto);
19    grid-auto-rows: minmax(100px, auto);
20    grid-gap: 10px;
21  }
22  .art1 { grid-area: 1 / 2 / 3 / 3; }
```

Redéfinir une grille avec des media queries



Redéfinir une grille avec des media queries

- Enfin on définit ce qui va être la règle à appliquer sur les écrans mobiles

- 1 On met `@media handheld ...`
- 2 On fait une copie du `.wrapper` et on le modifie : `1fr = 1` colonne dans la grille
- 3 On modifie le `grid-template-areas` en 1 colonne

```
1 @media handheld{
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr;
5     grid-auto-rows: minmax(20px, auto);
6     grid-row-gap: 5px;
7     grid-template-areas:
8       "tt"
9       "mn"
10      "a1"
11      "i1"
12      "i2"
13      "pub"
14      "pt1"
15      "ft";
16   }
17   .art1{ grid-area: a1; }
18   .info1{ grid-area: i1; }
19   .info2{ grid-area: i2; }
20   .pub{ grid-area: pb; }
21 }
```

Redéfinir une grille avec des media queries

- Enfin on définit ce qui va être la règle à appliquer sur les écrans mobiles

- 1 On met `@media handheld ...`
- 2 On fait une copie du `.wrapper` et on le modifie : `1fr = 1` colonne dans la grille
- 3 On modifie le `grid-template-areas` en 1 colonne

```
1 @media handheld{
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr;
5     grid-auto-rows: minmax(20px, auto);
6     grid-row-gap: 5px;
7     grid-template-areas:
8       "tt"
9       "mn"
10      "a1"
11      "i1"
12      "i2"
13      "pub"
14      "pt1"
15      "ft";
16   }
17   .art1{ grid-area: a1; }
18   .info1{ grid-area: i1; }
19   .info2{ grid-area: i2; }
20   .pub{ grid-area: pb; }
21 }
```

Redéfinir une grille avec des media queries

- Enfin on définit ce qui va être la règle à appliquer sur les écrans mobiles
 - 1 On met `@media handheld` ...
 - 2 On fait une copie du `.wrapper` et on le modifie : `1fr = 1 colonne` dans la grille
 - 3 On modifie le `grid-template-areas` en 1 colonne

```
1 @media handheld{
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr;
5     grid-auto-rows: minmax(20px, auto);
6     grid-row-gap: 5px;
7     grid-template-areas:
8       "tt"
9       "mn"
10      "a1"
11      "i1"
12      "i2"
13      "pub"
14      "pt1"
15      "ft";
16   }
17   .art1{ grid-area: a1; }
18   .info1{ grid-area: i1; }
19   .info2{ grid-area: i2; }
20   .pub{ grid-area: pb; }
21 }
```

Redéfinir une grille avec des media queries

- Enfin on définit ce qui va être la règle à appliquer sur les écrans mobiles
 - 1 On met @media handheld ...
 - 2 On fait une copie du .wrapper et on le modifie : 1fr = 1 colonne dans la grille
 - 3 On modifie le grid-template-areas en 1 colonne

```
1 @media handheld{
2   .wrapper {
3     display: grid;
4     grid-template-columns: 1fr;
5     grid-auto-rows: minmax(20px, auto);
6     grid-row-gap: 5px;
7     grid-template-areas:
8       "tt"
9       "mn"
10      "a1"
11      "i1"
12      "i2"
13      "pub"
14      "pt1"
15      "ft";
16   }
17   .art1{ grid-area: a1; }
18   .info1{ grid-area: i1; }
19   .info2{ grid-area: i2; }
20   .pub{ grid-area: pb; }
21 }
```

Redéfinir une grille avec des media queries

L'en-tête

- [Titre 1](#)
- [Titre 2](#)
- [Titre 3](#)

Titre 1

bla bla bla ...

Infos 1

Infos 2

Pub

Pied de page