

# HTML 5 et CSS 3

## Mathieu LACROIX

<https://www.lipn.univ-paris13.fr/~lacroix/>

# Principe du Web

# Définitions

---

**Réseau** : machines connectées ensemble

**Internet** : réseau connectant tous les réseaux

**Web** : application permettant de consulter des pages Web à l'aide d'un navigateur.

- Caractérisé par des hyperliens
- Distinction entre client et serveur
- Utilise Internet si nécessaire

**Navigateur** : logiciel de visualisation de pages Web

**HTML** : langage de balises permettant d'écrire des pages Web

# Qu'est-ce qu'une URL ?

---

protocole://adresse\_site/chemin\_repertoire/fichier

- **protocole** : langage utilisé pour la communication entre ordinateurs.  
Protocole pour le Web : http
- **adresse\_site** : Adresse du serveur (numéro IP ou nom de domaine) contenant le fichier
- **chemin\_repertoire** : répertoire où se trouve le fichier sur le serveur
- **fichier** : nom du fichier que l'on veut afficher

Exemples d'url :

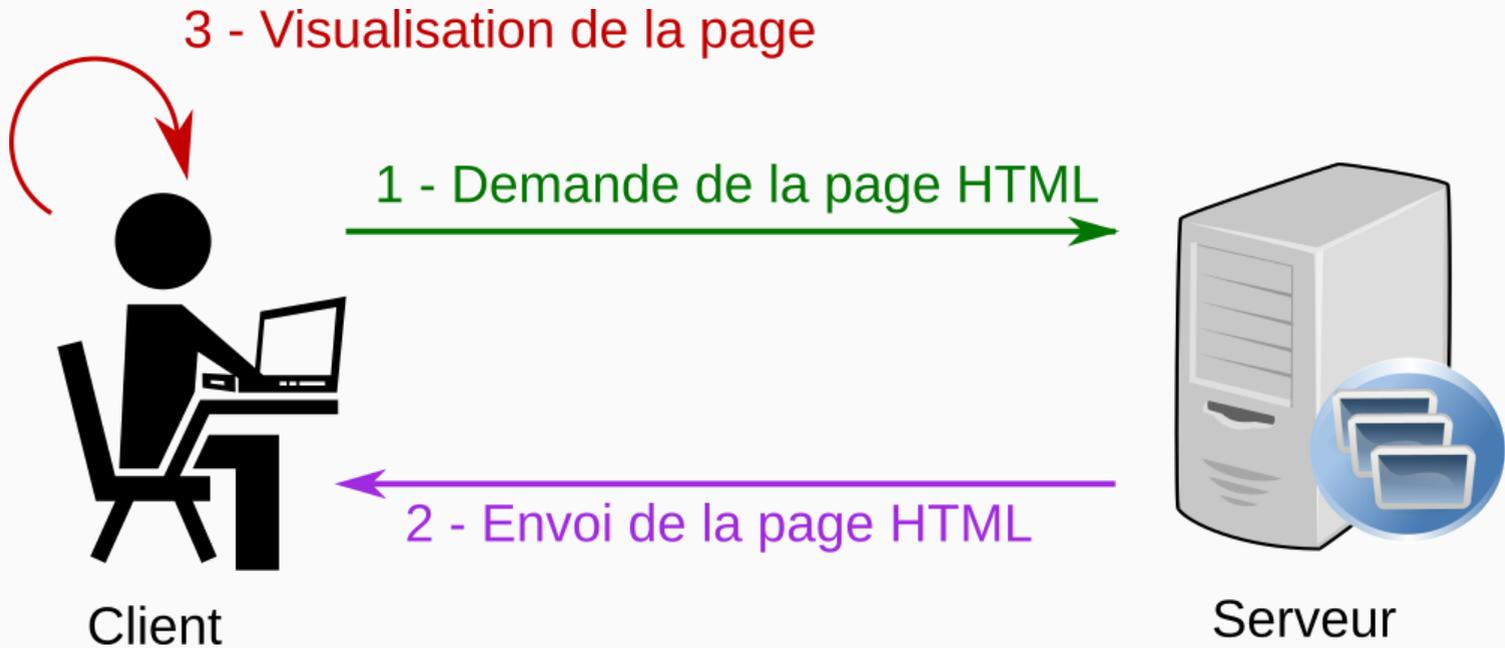
<http://lipn.univ-paris13.fr/~lacroix/Documents/Web/interaction.html>

<http://157.240.21.39/index.php>

<http://www.facebook.com/>

# Fonctionnement du Web

---



# Fonctionnement du Web

---

## Étape 1

```
GET /~lacroix/Documents/Web/interaction.html HTTP/1.1  
Host: lipn.univ-paris13.fr
```

# Fonctionnement du Web

---

## Étape 2

```
HTTP/1.1 200 OK
```

```
Content-Type: text/html
```

```
<!DOCTYPE html>
```

```
<html lang="fr">
```

```
<head>
```

```
  <title>Interaction client/serveur</title>
```

```
  <meta charset="utf-8">
```

```
</head>
```

```
<body>
```

```
<h1> Interaction client/serveur lors de la saisie d'une url  
  (statique) </h1>
```

```
...
```

# Fonctionnement du Web

## Étape 3

A screenshot of a Mozilla Firefox browser window. The title bar reads "Interaction client/serveur - Mozilla Firefox". The address bar shows the URL "www.lipn.univ-paris13.fr/~lacroix/Documents/IntroWeb/interaction.html". The browser interface includes a menu bar (Fichier, Édition, Affichage, Historique, Marque-pages, Outils, Aide), a tab bar with three tabs ("Positionnement", "Interaction client/serveur", "Positionnement"), and a toolbar with various icons for navigation and settings. The main content area is currently blank, and a mouse cursor is visible near the bottom center. The status bar at the bottom right shows the Zotero logo.

**Interaction client/serveur lors de la saisie d'une url (statique)**

La liste suivante présente les différentes étapes de l'interaction client/serveur lors de la saisie dans le navigateur d'une url dont le fichier porte l'extension "html" :

1. le client envoie un message au serveur pour lui demander de lui envoyer le fichier,
2. le serveur répond en envoyant le fichier demandé,
3. le navigateur interprète le code XHTML et affiche la page Web.

# Langage HTML

# Principe

---

- Définit la structure d'une page Web :
  - Paragraphes,
  - titres,
  - etc
- Ne fait pas la mise en forme de la page
- Langage de balises

```
<balise attribut1="valeur1" attribut2="valeur2"> ... </balise>  
<balise attribut1="valeur1" attribut2="valeur2" />
```

# Les principales balises

---

## Je suis un titre de niveau 1

## Je suis un titre de niveau 2 (titre de sous-section)

On peut aller jusqu'à des titres de niveau 6

Un paragraphe avec *emphase faible* et **emphase forte** et un retour à la ligne. On trace ensuite une ligne horizontale :

---

- liste non ordonnée, item 1
- liste non ordonnée, item 2

1. liste ordonnée, item 1
2. liste ordonnée, item 2



Paragraphe avec une image

On peut insérer des [hyperliens](#) pointant sur des pages Web/documents du site même site (chemin relatif) ou sur un autre serveur (chemin absolu).

On peut aussi insérer des tableaux :

### Balise Signification

p	paragraphe
h1	Titre de niveau 1
hr	ligne horizontale

# Les principales balises

---

```
<h1>Je suis un titre de niveau 1</h1>
```

```
<h2>Je suis un titre de niveau 2 (titre de sous-section)</h2>
```

```
<h6>On peut aller jusqu'à des titres de niveau 6</h6>
```

```
<p>Un paragraphe avec <em>emphase faible</em> et <strong>emphase forte</strong>  
et un retour <br/> à la ligne. On trace ensuite une ligne horizontale :</p>
```

```
<hr/>
```

```
<ul>
```

```
  <li>liste non ordonnée, item 1</li>
```

```
  <li>liste non ordonnée, item 2</li>
```

```
</ul>
```

```
<ol>
```

```
  <li>liste ordonnée, item 1</li>
```

```
  <li>liste ordonnée, item 2</li>
```

```
</ol>
```

# Les principales balises

---

`<p>Paragraphe avec une image`

```
</p>
```

```
<!-- Ici un commentaire -->
```

`<p>On peut insérer des <a href="https://galilee.univ-paris13.fr/">hyperliens</a> pointant sur des pages Web/documents du site même site (chemin relatif) ou sur un autre serveur (chemin absolu).</p>`

`<p>On peut aussi insérer des tableaux :</p>`

```
<table>
```

```
  <tr>  <th>Balise</th>  <th>Signification</th>  </tr>
```

```
  <tr>  <td>p</td>  <td>paragraphe</td>  </tr>
```

```
  <tr>  <td>h1</td>  <td>Titre de niveau 1</td>  </tr>
```

```
  <tr>  <td>hr</td>  <td>ligne horizontale</td>  </tr>
```

```
</table>
```

# Les principales balises

---

Pour diviser logiquement une page :

- **article** : regroupe un contenu ayant un sens propre indépendant  
Exemple : article de journal, commentaire, etc.
- **section** : regroupe des éléments ayant une même thématique.
- **header** : en-tête d'article, section, page, etc.
- **footer** : pied-de-page d'article, section, page, etc.
- **main** : partie principale de la page Web (1 seul)
- **nav** : menu de navigation.

# Structure d'une page Web

---

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>TITRE OBLIGATOIRE</title>
    <meta charset="utf-8" />
  </head>
  <body>
    ... code HTML de la page (titres, paragraphes, etc)...
  </body>
</html>
```

[Exemple complet](#)

# Représentation en arbre

---

```
<balise1> ... <balise2> .... </balise2> ... </balise1>
```

## Ascendants/descendants

- `balise2` est *descendant* de `balise1`
- `balise1` est *ascendant* de `balise2`

## Enfants/parents

- `balise2` est *enfant* de `balise1` s'il est descendant de `balise1` et qu'il n'existe pas d'élément ascendant de `balise2` et descendant de `balise1`
- `balise1` est *parent* de `balise2` si cette dernière est enfant de `balise1`

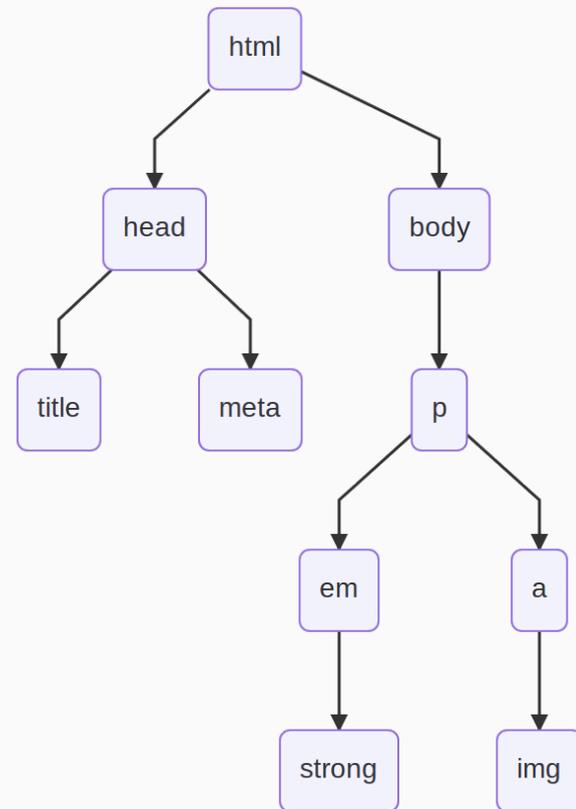
```
<p> texte <a href=""> texte <em> texte </em> texte </a> texte </p>
```

- `em` est enfant de `a` et descendant de `p`
- `a` est parent de `em` et enfant de `p`

# Représentation en arbre

- **Sommets** : balises HTML du document
- **Arcs**: arc b1 à b2 si b1 est parent de b2

```
<DOCTYPE html>
<html lang="fr">
<head>
  <title>Presentation</title>
  <meta charset="utf-8"/>
</head>
<body>
<p> Texte <em> avec emphase <strong> forte
</strong> </em> et hyperlien <a href="">
<img src="" alt=""/> </a>
</body>
</html>
```



# Balises `block` et `inline`

---

## Balises de type `inline`

- Balises `br`, `em`, `strong`, `img` et `a`
- Pas de retour à la ligne avant et après
- Prennent uniquement la place dont elles ont besoin

## Balises de type `block`

- Toutes les autres balises
- Retour à la ligne avant et après
- Prennent toute la largeur du conteneur parent

```
<p> du texte <em> sur la même </em> </p>  
<p> Nouveau bloc ! </p>
```

Du texte *sur la même* ligne.

Nouveau bloc !

# Règles à respecter

---

## Pas d'imbrication de balises

~~<balise1> ... <balise2> ... </balise1> ... </balise2>~~

## Enfants et parents possibles

Certaines balises sont interdites comme enfants ou parents d'une autre balise.

Exemples :

- un titre ne peut pas être fils de paragraphe et inversement
- un paragraphe ne peut pas être fils d'un autre paragraphe
- les seuls fils possibles pour `ul` et `ol` sont `li`
- les balises `inline`, excepté `a`, ne peuvent être parents d'une balise `block`

# Langage CSS

# Règles CSS

---

Langage CSS : ensemble de règles

```
selecteur {  
  propriete1: valeur;  
  propriete2: valeur;  
  ...  
}
```

Exemple :

```
h1 {  
  color: red;  
}
```

Lier le fichier CSS au HTML (dans la partie head):

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

# Propriétés CSS

# Police et couleurs

---

- `font-family` : police utilisée (importer des polices avec `@font-face`)
- `font-size` : taille de la police en pixels `px`, taille relative `em`
- `font-weight` : `bold` pour mettre en gras (`normal` sinon)
- `font-style` : `italic` ou `normal`
- `text-align` : `left`, `right`, `center` ou `justify`
- `text-decoration` : `underline` ou `none`
- `color` : couleur de police
  - Nom de couleur : `red`, `black`, etc
  - Code hexadécimal : `#123456`
  - `rgba(rouge,vert,bleu,transparence)` avec :
  - `rouge`, `vert` et `bleu` : nombres entre 0 et 255
  - `transparence` : nombre entre 0 et 1

## Exemple

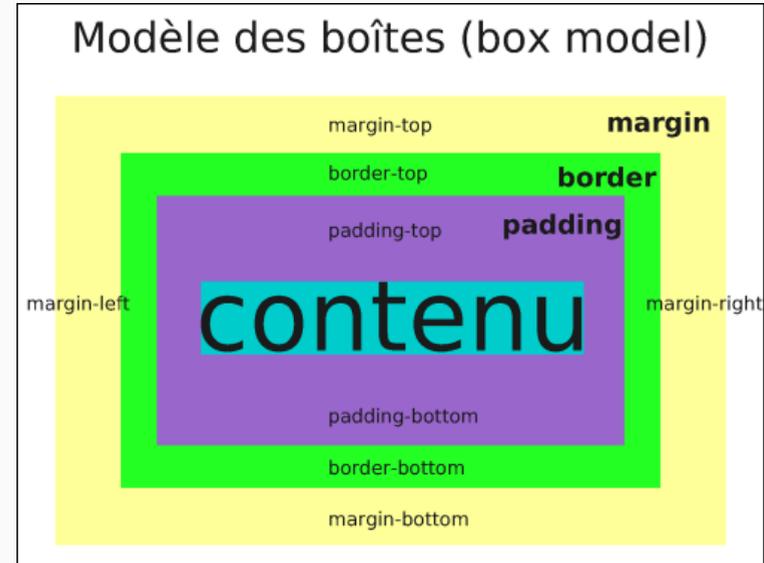
# Marges et bordures

- largeur : `width`
- hauteur : `height`
- marge intérieure : `padding`
- marge extérieure : `margin`
- bordure : `border: type size color;`
  - `type` : `solid`, `double`, `inset`, `outset`
  - `size` : épaisseur de la bordure
  - `color` : n'importe quelle couleur
- arrondis : `border-radius` : `taille;`

`padding`, `border` et `margin` peuvent être différents pour chaque côté (`-left`, `-right`, `-top` et `-bottom`)

`width` et `height` donnent uniquement la taille du contenu. Pour prendre en compte les marges intérieures et les bordures => `box-sizing: border-box.`

[Exemple](#)



# Arrière-plan

---

- `background-color` : couleur d'arrière-plan (n'importe quelle couleur)
- `background-image:url("fichier_image")` : image d'arrière-plan
- `background-repeat` : si l'image de fond se répète (`repeat`) ou pas (`no-repeat`)
- `background-position` : position de l'image par rapport à l'élément.
  - Positionnement vertical : `top`, `bottom`, `center`
  - Positionnement horizontal : `left`, `right`, `center`

```
background-position : bottom right;
```

## [Exemple](#)

# Listes et tableaux

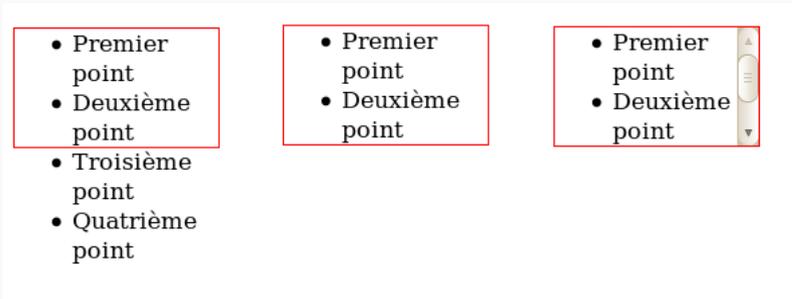
---

- `border-collapse: collapse` pour coller les cellules d'un tableau
- `list-style` : style des puces et numéros
  - liste non-ordonnée : `square`, `disc`, `circle`, `none`
  - liste ordonnée : `decimal`, `lower-roman`, `upper-roman`, `none`

# Dépassement et type de balises

---

- **overflow** : comportement en cas de dépassement
  - **visible** : ce qui dépasse est affiché
  - **hidden** : caché
  - **auto** : barre de défilement (scroll)



- **display** : modification du comportement d'affichage
  - **block** : s'affiche comme un élément de type **block**
  - **inline** : s'affiche comme un élément de type **inline**
  - **inline-block** : taille, marge mais pas de retour à la ligne
  - **none** : supprime l'élément

# Classes et identifiants

# Utilisation

---

Classes et identifiants permettent de ne modifier que certains éléments.

```
<balise class="nom_classe" id="nom_identifiant"> ... </balise>  
<balise class="nom_classe" id="nom_identifiant" />
```

```
.nom_classe{  
    ...  
}  
#nom_identifiant{  
    ...  
}
```

Les règles ne s'appliquent qu'aux éléments ayant la classe (et/ou l'identifiant).

L'identifiant est unique.

[Exemple](#)

# Pseudo-classes

---

Les pseudo-classes précisent un état de la cible.

- `:hover` : quand la cible est survolée par la souris
- `:nth-child(n)` : quand la cible est le  $n^{\text{ème}}$  enfant
- `:active`: quand la cible est cliquée
- `:visited` : quand la cible (hyperlien) a été visité

Exemple :

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

Quand on survole un paragraphe, il s'affiche en rouge.

[Exemple nth-child](#)

# Balises universelles

---

- Permettre de modifier l'apparence de code non délimité par des balises
- Deux balises universelles :
  - `div` : balise de type `block`
  - `span` : balise de type `inline`

## Exemple

```
<p><span class="bleu">Paris</span>, <span class="bleu">Lyon</span> et  
<span class="bleu">Marseille</span> sont les trois plus grandes villes de  
France.</p>
```

```
.bleu { color: blue;}
```

Paris, Lyon et Marseille sont les trois plus grandes villes de France.

# Combinaisons et modifications en cascade

# Combiner les sélecteurs

---

Combinaison	Explication
E	un élément E
*	n'importe quel élément
E F	un élément F descendant de E
E > F	un élément F enfant de E
E.ma_classe	un élément E ayant la classe ma_classe
E#id	un élément E ayant l'identifiant id
E, F	un élément E ou un élément F

## Exemples

[Exemple](#)

```
p .rouge { /* les éléments de classe rouge contenus dans un paragraphe */ }
```

```
p.rouge { /* les paragraphes de classe rouge */ }
```

```
p > em.rouge.it { /* les em de classe rouge et it enfants de paragraphes */ }
```

# Styles en cascade

---

Plusieurs règles peuvent s'appliquer à un même élément.

```
<div id="monId"> <p class="rouge"> ... </p> </div>
```

```
p {  
  font-size: small;  
}  
.rouge {  
  color: red;  
}  
#monId p {  
  text-align: right;  
}
```

=> Le paragraphe est écrit en petit, rouge et à droite.

# Priorité des règles

---

Lorsque plusieurs règles s'appliquent, si des valeurs sont différentes pour une propriété, on choisit la règle prioritaire.

- On associe un nombre à trois chiffres pour chaque sélecteur :
  - centaines : nombre d'identifiants dans le sélecteur,
  - dizaines : nombre de classes et pseudo-classes,
  - unités : nombre de balises.
- La règle prioritaire est celle dont le nombre associé au sélecteur est le plus grand.
- En cas d'égalité, on choisit celle écrite en dernier dans le fichier.

```
p{ color: yellow; } /* Priorité : 001 */  
.rouge { color: red; } /* Priorité 010 */  
#monId p { color: green; } /* Priorité 101 */
```

=> Le paragraphe est écrit en vert.

[Exemple](#)

# Héritage

---

**Propriété supportant l'héritage** : si la valeur de cette propriété n'est pas spécifiée pour un élément, celle-ci prend comme valeur celle de l'élément parent.

```
body{ color: red; }
```

=> Tout le document (sauf hyperliens) est en rouge.

```
<p class="rouge"> Du texte <em> mis en valeur </em> puis <strong> fortement  
mis en valeur. </strong> </p>
```

```
.rouge { color: red; }  
em { color: blue; }
```

- Du `texte` et `puis` => rouge d'après la règle 1
- `mis en valeur` => bleu d'après la règle 2
- `fortement mis en valeur` => rouge par héritage.

# Positionnement en CSS

# Propriété position

# Types de positionnement

---

## Positionnement statique (`static`)

- Les éléments sont affichés dans l'ordre où ils apparaissent dans le HTML (flux).
- Positionnement par défaut.

## Positionnement relatif (`relative`)

- déplace l'élément par rapport à la position qu'il avait dans le flux.
- `top`, `bottom`, `left`, `right` indiquent le **décalage** (`left:50px;` ajoute 50 pixels à gauche de l'élément)

## [Exemple](#)

# Types de positionnement

---

## Positionnement absolu (**absolute**)

- L'objet est retiré du flux.
- Il est ajouté par dessus les autres éléments.
- **top, bottom, left, right** indiquent la **distance** par rapport au premier ascendant qui n'a pas de positionnement statique (page Web si aucun).

[Exemple1](#) [Exemple2](#)

## Positionnement fixe (**fixed**)

- Similaire au positionnement absolu.
- La distance est toujours par rapport au navigateur.
- Insensible au scrolling.

[Exemple](#)

# Grilles

# Concept

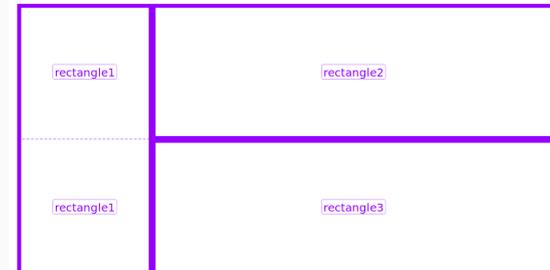
---

- Mise en page à l'aide de grilles (en CSS)
- Le conteneur se comporte comme une grille/tableau
- **Ses enfants** sont rangés dans les cases
- Pas de modification du code HTML

# Conteneur

- Indiquer que c'est une grille avec `display:grid`
- Indiquer le nombre de lignes (et les tailles) avec `grid-template-rows`
- Indiquer le nombre de colonnes (et les tailles) avec `grid-template-columns`
- Former des zones (rectangulaires) nommées avec `grid-template-areas`

```
.conteneur{  
  display: grid; /* C'est une grille */  
  /* Avec deux lignes de 100 pixels */  
  grid-template-rows: 100px 100px;  
  /* Avec deux colonnes (50 et 200 pixels) */  
  grid-template-columns: 50px 200px;  
  grid-template-areas:  
    "rectangle1 rectangle2"  
    "rectangle1 rectangle3";  
}
```



# Conteneur

---

## Remarques

- Il existe pour les tailles de colonnes et lignes l'unité `fr` (pour free space)

```
grid-template-columns: 1fr 3fr 200px;
```

- une colonne
- une colonne 3 fois plus large
- une colonne de 200px
  - ⇒ Les 4 premières colonnes prennent toute la largeur disponible.
- On peut utiliser `repeat(n, taille)` pour donner les tailles
- On peut ajouter un espace entre les lignes/colonnes avec `row-gap` et `column-gap` (anciennement `grid-row-gap` et `column-row-gap`)
- On n'est pas obligé de préciser `grid-template-rows` (les lignes prennent alors la place minimale nécessaire)
- Si une case n'est pas utilisée, dans `grid-template-areas`, on met un `.`

# Enfants sur la grille

---

Pour chaque enfant, on précise la zone dans laquelle il doit s'afficher avec `grid-area: nom_de_la_zone`

- Chaque enfant doit être affecté à une zone
- Toutes les zones doivent être utilisées

```
<section class="conteneur">
  <p>Paragraphe 1</p>
  <p>Paragraphe 2</p>
  <p>Paragraphe 3</p>
</section>
```

[Exemple 1](#)

[Exemple 2](#)

```
section > p:nth-child(1){
  grid-area: rectangle1;
}
section > p:nth-child(2){
  grid-area: rectangle2;
}
section > p:nth-child(3){
  grid-area: rectangle3;
}
```

# Responsive design

# Principe

---

- Affichage "confortable" du site quel que soit le support (taille)
- Pas de modification du fichier HTML sauf ajout dans le head :

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- Du code CSS s'applique pour certaines tailles d'écran seulement
- Utilisation des *media queries*

```
@media (condition){  
  /* Règles CSS qui sont prises en compte si la condition est vérifiée */  
}
```

condition est définie par `min-width:taille` et/ou `max-width:taille`

## Exemple

Pour une expérience utilisateur optimale, il faut un CSS compatible pour (presque) tous les navigateurs, quel que soit leur version.