

Projet

Construction d'un site Web

Sites statiques avec le langage HTML

Construction d'un site Web

- **Comment ça marche le Web ?**
 - **WWW (World Wide Web)** a été créé par Tim Berners-Lee au CERN au début des années **1990**



Le client **demande** une **ressource Web** au serveur.
Le **navigateur** se charge d'afficher la ressource.

Construction d'un site Web

- C'est quoi une **ressource Web** ?
 - Toute sorte de **document accessible par le Web**
 - Page Web, images, vidéos, documents PDF...
- Un **site Web** regroupe alors **plusieurs ressources**
 - Différentes technologies / langages sont possibles
 - **HTML, CSS, JavaScript, PHP...**
 - On distingue les ressources **statiques** et **dynamiques**
 - Une **ressource dynamique** peut se **modifier dynamiquement** en fonction de la requête, des informations disponibles, etc.
 - Une **ressource statique** n'est **pas mise à jour automatiquement** lorsque les informations changent

Construction d'un site Web

- **Site Web statique**

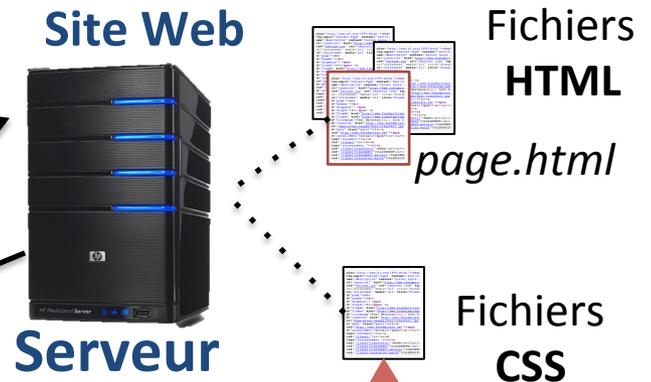
- Les pages ne sont pas mise à jour automatiquement

On demande une ressource (**page Web**) par son **URL**



Requête
http://serveur/page.html

Réponse
(contenu page.html)



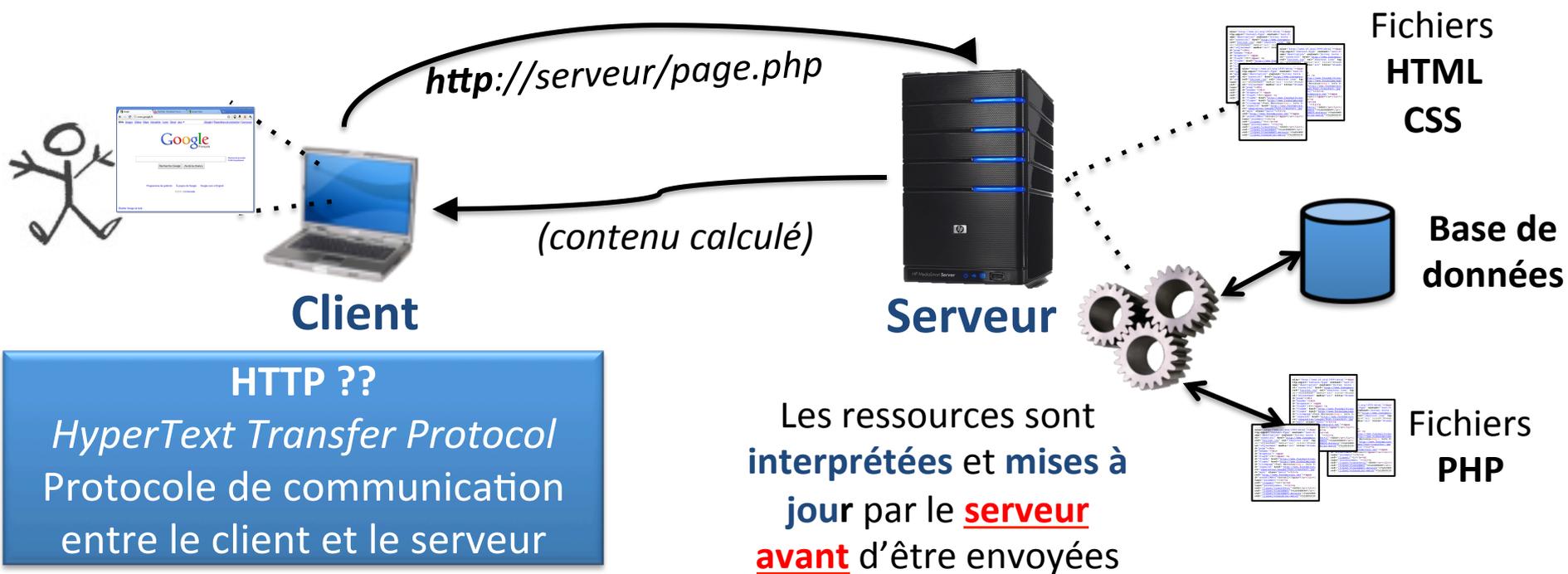
Le **navigateur interprète** et **affiche** le contenu de la **page**

Chaque **ressource Web** est identifiée par une **adresse URL : Unified Resource Location**

Construction d'un site Web

- **Site Web dynamique**

- **Mise à jour automatique** des pages lorsque les informations ont changées
- Le **contenu** des pages Web **est calculé** dynamiquement



- **HTML (HyperText Markup Language)**

- **Historique**

- Née avec le Web dans le années 1990
- **Normalisation** en 1995 avec HTML 2.0 et la **W3C** (*World Wide Web Consortium*)
- Années 2000 jusqu'à aujourd'hui : **HTML 4.01**
- Aujourd'hui et **demain** : **HTML 5**

- **Mais c'est quoi le HTML ?!**

- C'est le langage des pages Web, reconnu par les navigateurs
- Un **langage de balise**, pas un langage de **programmation**
- Il permet de **structurer le contenu** d'une page Web
- Hypertexte, car il permet de créer des liens vers d'autres pages ou d'autres parties d'une page

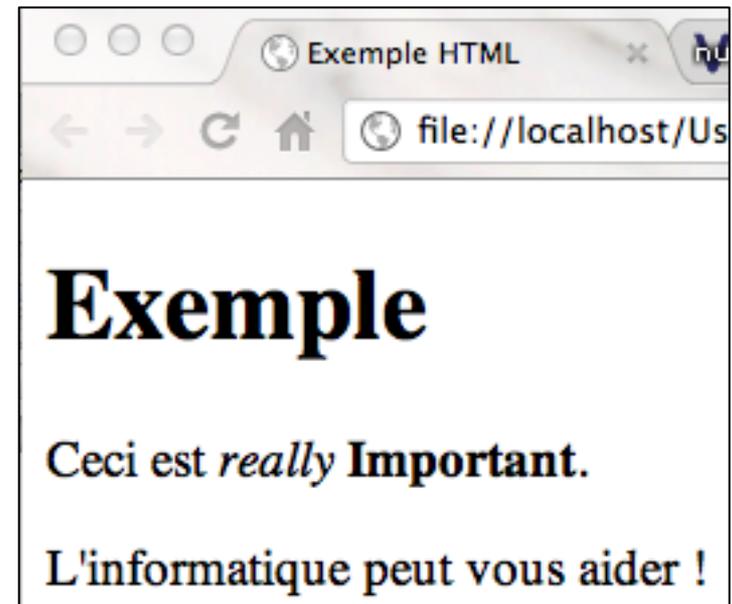
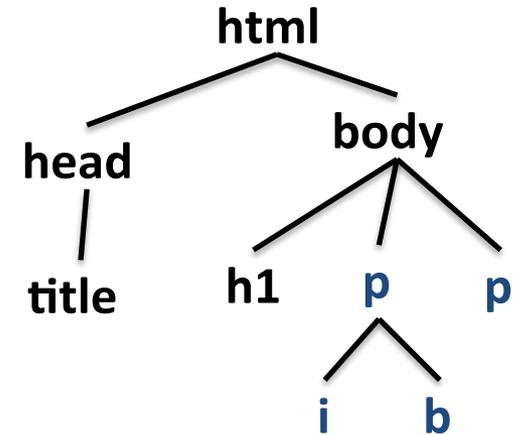
HTML

- Dans HTML, le texte est structuré par des balises, formant une structure arborescente
 - Une balise est un couple
< nomBalise > texte </ nomBalise >
 - <nomBalise>** est une *balise ouvrante*
 - </nomBalise>** est une *balise fermante*
 - Le **navigateur** Web va **lire** les **balises** et **afficher** le **texte** en conséquence
 - <i>** really **</i>** *really*
 - ** Important **** **Important**

HTML

Chaque balise ouverte
doit être fermée
<balise> ... </balise>

```
<html>
<head>
  <title> Exemple HTML </title>
</head>
<body>
  <h1>Exemple</h1>
  <p>Ceci est <i>really</i>
  <b>Important</b>. </p>
  <p> L'informatique peut vous
  aider ! </p>
</body>
</html>
```



HTML

- Structure d'un document HTML

```
<!DOCTYPE html>
```

Indication « idiome » HTML

```
<html>
```

```
<head>
```

```
<meta name="author" content="Manuele Kirsch Pinheiro" />
```

```
<title> Exemple HTML </title>
```

```
</head>
```

Entête (head)
Informations générales
sur le document

```
<body>
```

```
<h1>Exemple</h1>
```

```
<p>Ceci est <i>really</i>
```

```
<b>Important</b>. </p>
```

```
<p> L'informatique peut vous aider ! </p>
```

```
</body>
```

Corps (body)
Contenu du document

```
</html>
```

- Élément **DOCTYPE**

- Indique au navigateur quelle version de HTML a été utilisée

- **HTML 4.01**

- Couramment compris par tous les navigateurs

- ```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
http://www.w3.org/TR/html4/loose.dtd>
```

- **HTML 5**

- ```
<!DOCTYPE html>
```

- En cours de définition
 - Reconnu uniquement par les navigateurs les plus récents (Google Chrome 16.0, Firefox 9.0, Internet Explorer 9...)

- **Éléments de l'entête (head)**

- Informations **complémentaires** sur le document

```
<head>  
  <meta name="author"  
        content= "Manuele" />  
  <title> Exemple HTML </title>  
</head>
```

- **Ce n'est pas le contenu du document**, donc ces informations ne sont **pas affichées** dans la page

- Typiquement, informations pour les moteurs de recherche

- **Balises**

- **<titre> ... </titre>** : titre du document

- **<meta ... />** : métadonnées (descriptions) sur le document

- **<link ... />, <style> ... </style>** : styles (on verra plus tard)

- **Éléments de l'entête (head)**

Ouverture et fermeture de la balise

`<meta name="author" content="auteur" />`

Attributs associés à la balise

Précisions sur une balise

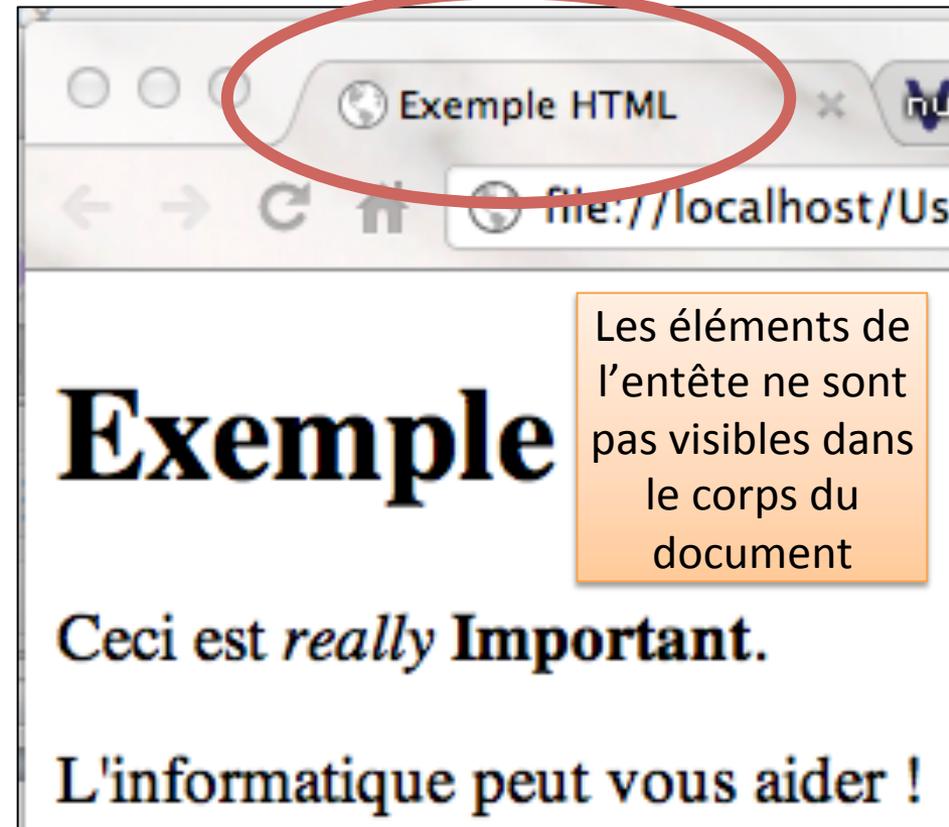
Chaque balise possède son ensemble d'attributs

`<balise attribut = "valeur" ... >`

`<meta name="description" value="..." />`

`<meta charset="ISO-8859-1">`

`< title > Exemple HTML </ title >`



- **Éléments du corps (body)**

- Contenu du document
- Partie rendue visible par les navigateurs

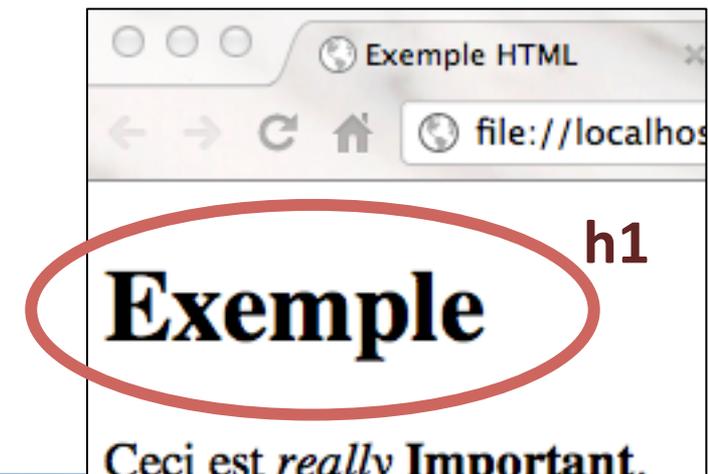
- **Balises** : il y en a plein...

- Titres : **<h1>**, **<h2>** ... **<h6>**
- Paragraphe et saut de ligne : **<p>** et **
**
- Citations et mises en valeur : ****, **<i>**, **<blockquote>**...
- Images et liens : ****, **<a ...>** ...
- Listes : ****, ****, ****
- Tableaux : **<table>**, **<tr>**, **<td>**...
- Organisation du document : **<div>**, **<section>**...

```
<body>
  <h1>Exemple</h1>
  <p>Ceci est <i>really</i>
    <b>Important</b>. </p>
  <p> L'informatique peut vous
aider ! </p>
</body>
```

- **Éléments du corps (body)**
- **Les titres : h1, h2, h3, h4, h5, h6**
 - Les éléments **hx** permettent de définir des **titres** de **différents niveaux**
 - **h1 correspond au titre principal**
 - Ils doivent apparaître dans l'ordre (**h1 avant h2**) avec un **seul titre principal (h1)**

```
<body>
  <h1>Exemple</h1>
  ...
</body>
```



- **Éléments du corps (body)**
- **Paragraphe, saut de ligne et cia...**
 - La balise `<p> ... </p>` indique un paragraphe
 - La balise `
` fait un simple saut de ligne
 - Les balises `...` et `<i>...</i>` mettent un texte en relief (en gras ou en italique)
 - La balise `<blockquote>...</blockquote>` permet de citer une autre page Web
 - `<blockquote cite="http://source/"> citation </blockquote>`
 - La balise `<hr />` permet d'établir une séparation (ligne horizontal) dans le document

HTML

• Éléments *body*

<html>

<head> ... **</head>**

<body>

<h1>Exemple h1**</h1>**

<h2>Exemple h2**</h2>**

<p>Ceci est un paragraphe avec un **<i>**terme technique**</i>** et un **
** ****mot-clé****. **</p>**

<blockquote

cite="http://fr.wikipedia.org/wiki/Hypertext_markup_language">

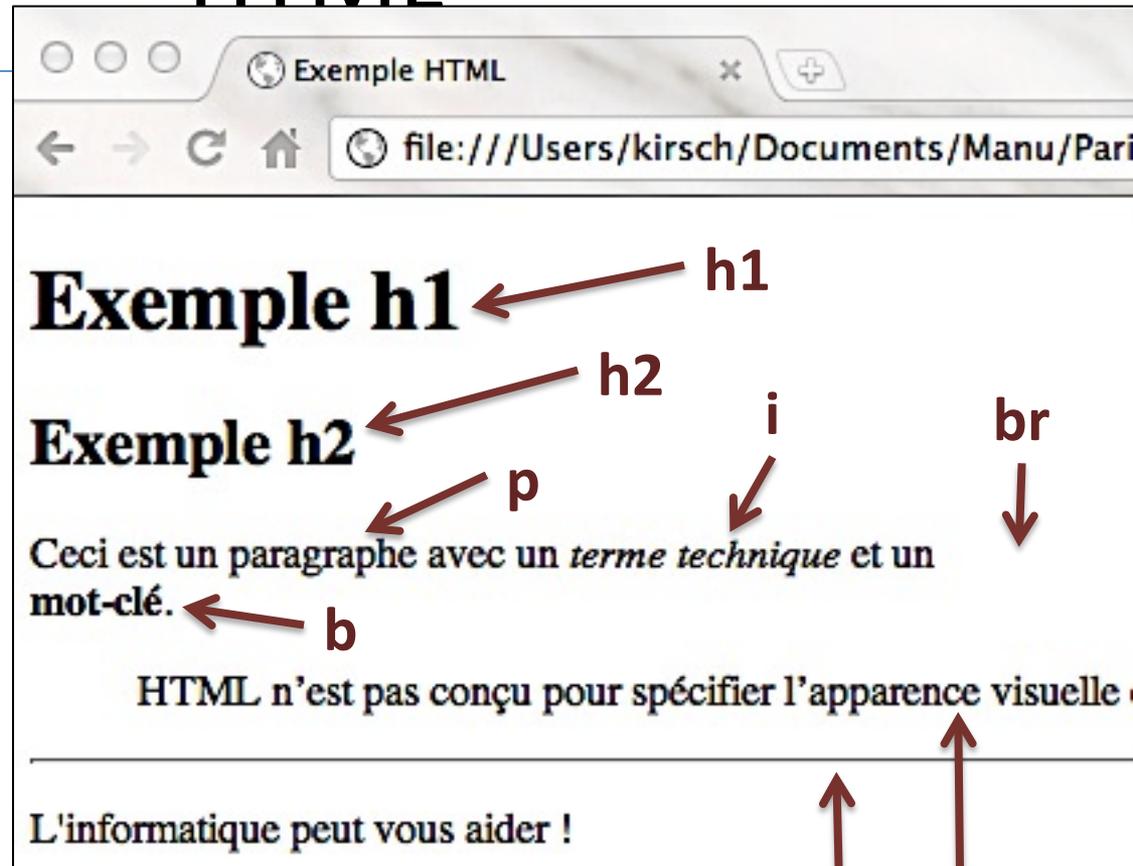
HTML n'est pas conçu pour spécifier l'apparence visuelle exacte des documents. **</blockquote>**

<hr/>

<p> L'informatique peut vous aider ! **</p>**

</body>

</html>



hr

blockquote

Informatique

Modélisation UML

Objectifs de la séance :

HTML

Images, liens, tableaux, listes

- **HTML**

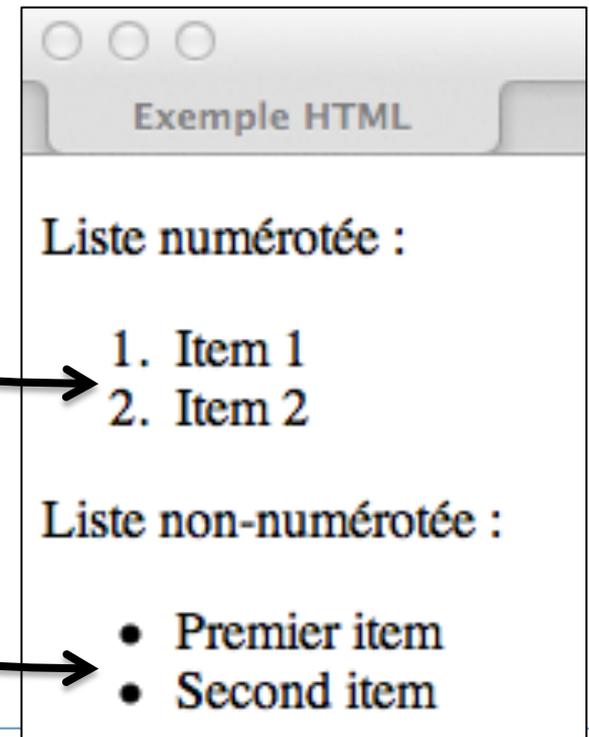
- Langage de balises, permettant la structuration des pages Web
- Organisation en balises
 - `<balise attr="valeur"> ... </balise>`
- Organisation du document
 - Entête : **head**
 - Corps du document : **body**
- Différentes types de balises possibles
 - Listes, tableaux, images, liens...

HTML : listes

- Plusieurs types de listes sont possibles
 - Listes numérotés : ` ... `
 - Listes non-numérotés : ` ... `
 - Peu importe la liste, un seul moyen d'indiquer les éléments : ` ... `

```
<ol>
  <li> Item 1 </li>
  <li> Item 2 </li>
</ol>

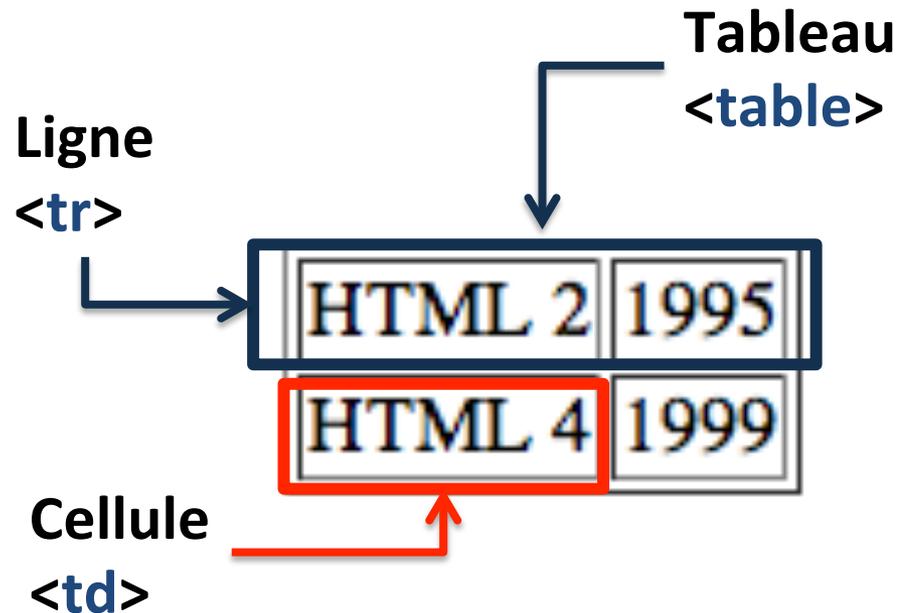
<ul>
  <li> Premier item </li>
  <li> Second item </li>
</ul>
```



HTML : tableaux

- Pour créer un tableau en HTML, on va combiner plusieurs balises :
 - **table**, **tr**, **td**, **caption**, **th**, **thead**, **tbody**

```
<table border="1">
  <tr>
    <td>HTML 2</td>
    <td>1995</td>
  </tr>
  <tr>
    <td>HTML 4</td>
    <td>1999</td>
  </tr>
</table>
```



HTML : tableaux

```
<table border="1">
```

```
<caption>Historique du HTML </caption>
```

caption : légende

```
<thead>
```

thead : Entête du tableau

```
<tr>
```

```
<th> Version</th>
```

th : Cellule de l'entête

```
<th> Année </th>
```

```
</tr>
```

```
</thead>
```

tbody : corps du
tableau

```
<tbody>
```

```
<tr>
```

```
<td>HTML 2</td> <td>1995</td>
```

```
</tr>
```

```
<tr>
```

```
<td>HTML 4</td> <td>1999</td>
```

```
</tr>
```

```
</tbody>
```

```
</table>
```

Exemple HTML

Historique du
HTML

Version	Année
HTML 2	1995
HTML 4	1999

HTML : images

- Insertion d'images dans le texte : balise **img**

```

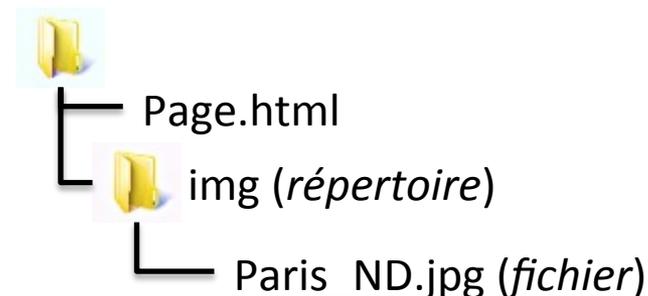
```

Taille optionnelle

```
width="80"  
height="70"
```

Texte alternatif
infobulle

src : où se trouve l'image
Soit elle est avec la page Web
src= ".\img\Paris_ND.jpg »



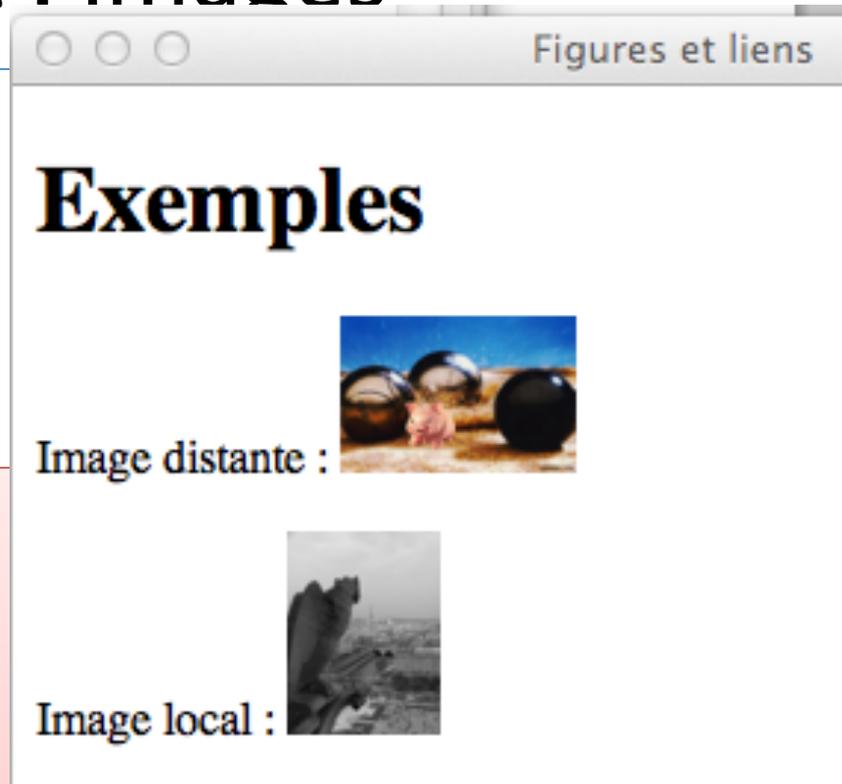
Soit elle est sur un serveur :

```
src="http://lsteffene1.fr/images/petanque-cochonnet.jpg"
```

HTML : images

- Balises HTML : Images

```
<html>
<head> ... </head>
<body>
  <h1>Exemples </h1>
  <p>Image distante :  </p>
  <p>Image local :  </p>
</body>
</html>
```



HTML : liens

- L'usage des **liens** permet de relier une page Web à d'autres pages, voir à d'autres points dans la page

** lien visible**

- L'attribut **href** indique vers où aller lorsqu'on clique sur le lien
 - Lien local :
** vers autre page **
 - Lien distant:
** ailleurs **
 - Envoyer un mail :
envoyer mail

HTML : liens

On attribue un identificateur
<balise id="identificateur">

```
<h1 id="debut">Liens </h1>
```

```
<p>Lien vers <a href="http://epi.univ-paris1.fr">  
'EPI </a></p>
```

```
<p>Lien vers <a href="coursHTML-5.html">  
exemple tableaux </a></p>
```

```
<p>Envoyer un mail à  
<a href="mailto:moi@mail.com"> moi </a></p>
```

```
<p> .... </p>
```

```
<p> <a href="coursHTML-7.html#debut"> Retourner  
au début </a> </p>
```

Liens

Lien vers [l'EPI](#)

Lien vers [exemple tableaux](#)

Envoyer un mail à [moi](#)

texte texte bla bla bla bla texte te
texte bla bla bla bla texte texte bl

Lien vers l'identificateur

[Retourner au début](#)

- **Légende**

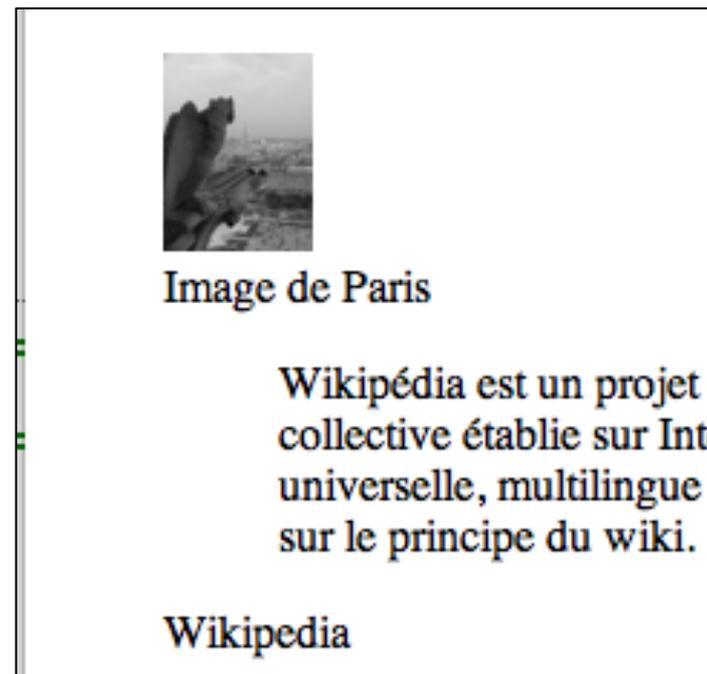
- En HTML5, on peut attacher une légende à une figure, un morceau de texte...

<figure> ... **<figcaption>** *Légende* **</figcaption>** **</figure>**

```

<figure>
  
  <figcaption> Image de Paris</figcaption>
</figure>

<figure>
  <blockquote cite="http://fr.wikipedia.org/"> ....
  </blockquote>
  <figcaption>Wikipedia</figcaption>
</figure>
    
```



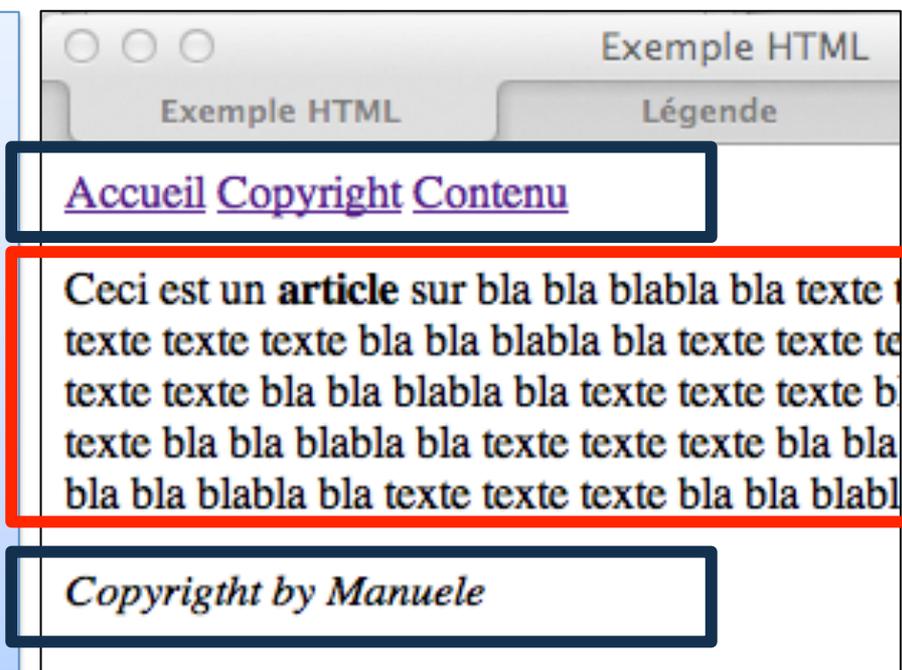
HTML : organisation du document

- **Organisation du document**

- On peut structurer le contenu en plusieurs blocs
- **Bloc de contenu : `div`**

`<div>` *bloc de contenu* `</div>`

```
<div id="menu">
  <a href="index.html">Accueil</a>
  <a href="#foot">Copyright</a>
  <a href="#article">Contenu</a>
</div>
<div id="article">
  <p>Ceci est un <b>article</b>... </p>
</div>
<div id="foot">
  <p><i>Copyrightt by Manuele </i></p>
</div>
```





- **Organisation du document**

- Nouvelle balises HTML5 : *header*, *footer*, *article*, *section*, *nav*, *aside*

```
<header> ... </header>
```

```
<nav> ... </nav>
```

```
<section>
```

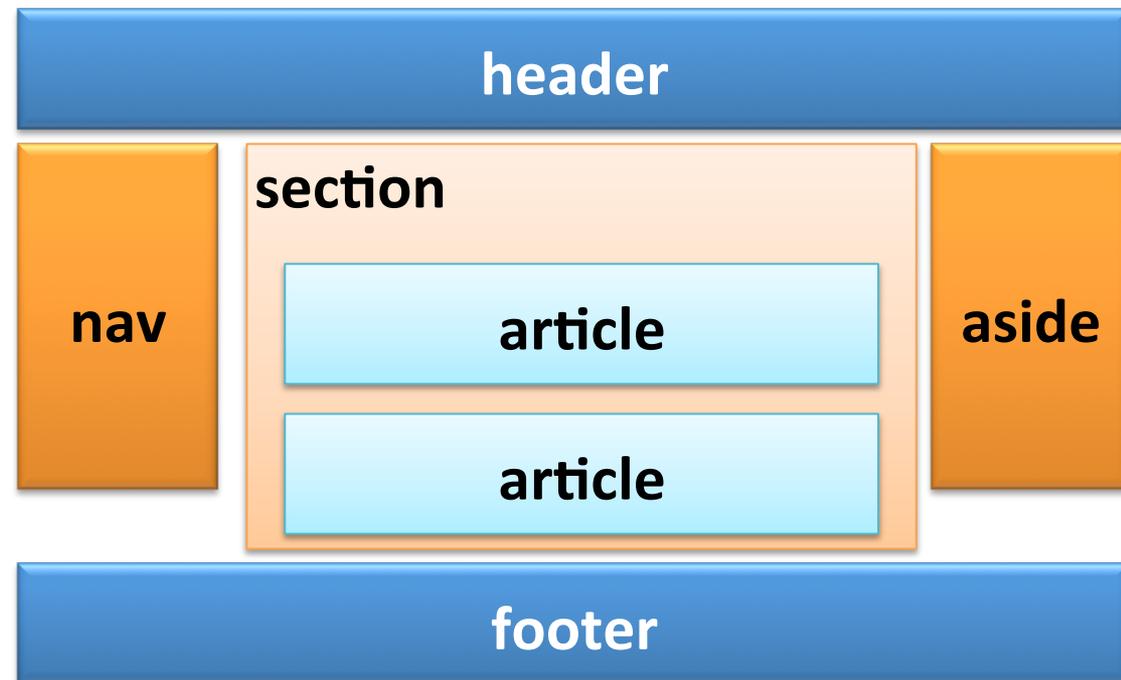
```
  <article>... </article>
```

```
  <article> ... </article>
```

```
</section>
```

```
<aside>...</aside>
```

```
<footer> ... </footer>
```



HTML : organisation du document

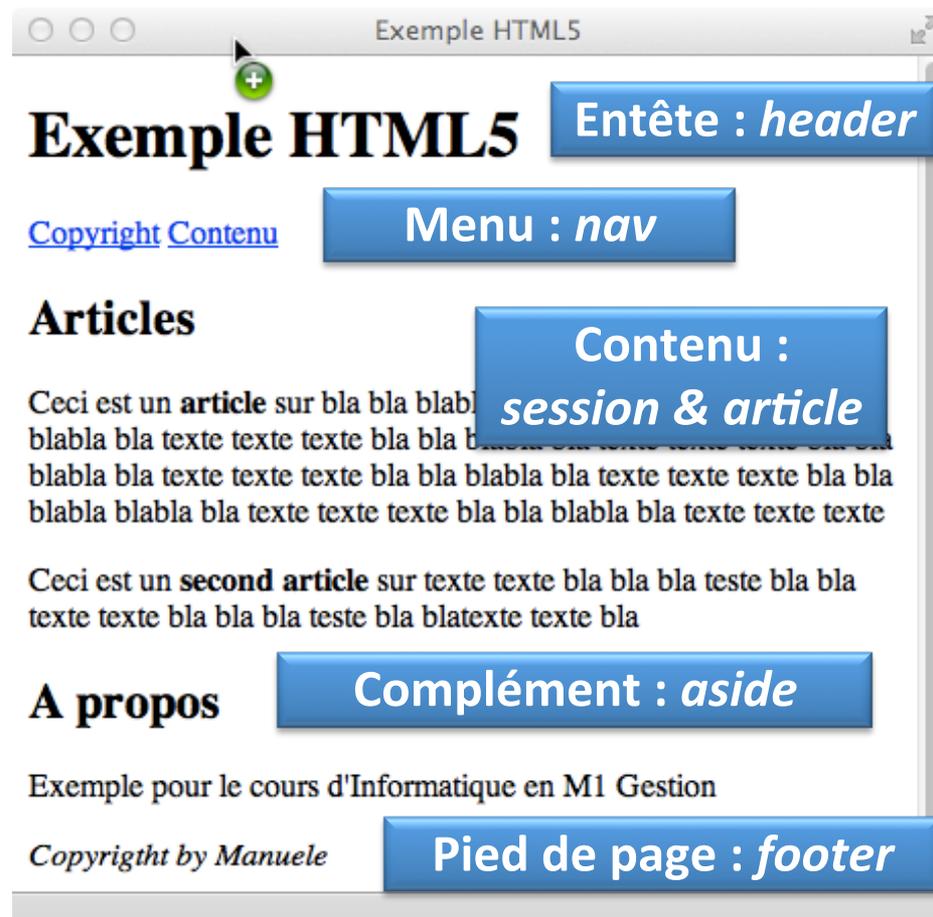
Et la mise en forme ?!
C'est le rôle de CSS !!!



```

<body>
<header> <h1>Exemple HTML5</h1> </header>
<nav>
  <a href="#foot">Copyright</a> ...
</nav>
<section id="cont">
  <h2>Articles</h2>
  <article>
    <p>Ceci est un <b>article</b> ...</p>
  </article>
  <article> <p> ... </p> </article>
</section>
<aside>
  <h2>A propos</h2> <p> ... </p>
</aside>
<footer id="foot"> <p><i>...</i></p>
</footer>
</body>

```



HTML : il y a beaucoup plus...



- HTML5 propose des nouvelles balises pour contenu multimédia
 - video, audio, canevas...

```
<figure>
```

```
<video controls>
```

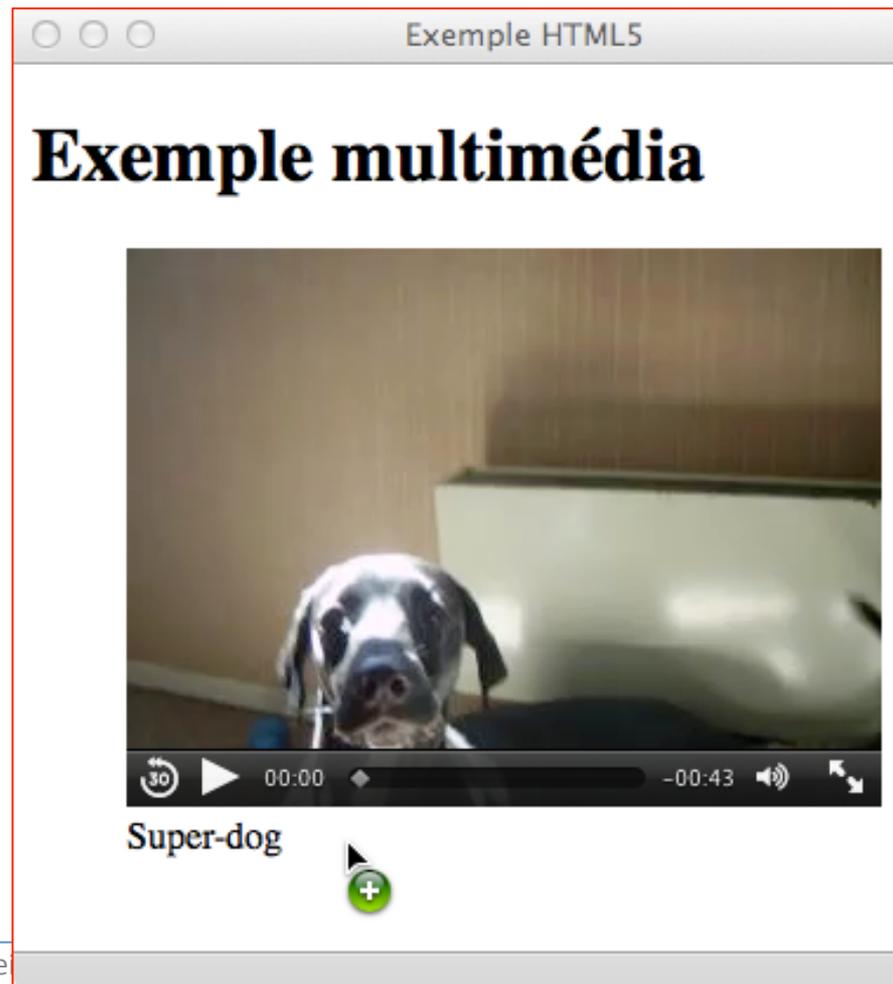
```
<source src="video/IMG4706.mp4"  
type="video/mp4"/>
```

Votre navigateur ne supporte pas la balise
<i>video</i>.

```
</video>
```

```
<figcaption>Super-dog</figcaption>
```

```
</figure>
```



Informatique

Modélisation UML

Objectifs de la séance :

**Mise en page de sites Web avec
CSS**

- **CSS : *Cascading Style Sheet***

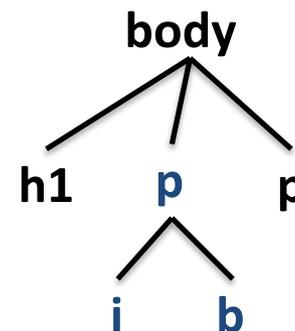
- Mise en forme des balises HTML

- Couleurs, fond, polices...

- Décorer les balises avec des **propriétés de mise en forme**

- Effet en cascade

- La décoration s'applique à la balise et aux balises à l'intérieur de celle-ci
- *Ex.: si on décore **p** en **bleue**, on décore aussi **i** et **b** en **bleue***



- **CSS n'est pas un langage de balise**

- **Propriétés :**

propriété : valeur ;

text-align: center ;

- **Sélecteurs :**

Sélecteur { liste de propriétés }

*h1 { text-align: center ;
color : red ; }*

- **Feuilles de style :** ensemble de sélecteurs

CSS : décoration

- **Décoration des balises** : la méthode simple et bête
 - On peut **décorer directement une balise** dans le document HTML
 - Attribut *style*
`<h1 style="text-align: center; color: red;"> ... </h1>`
 - On peut décorer **une partie du texte** HTML
 - Balise HTML ` ... `
`<p> Ceci est un texte en
 gras </p>`

CSS : décoration

- Exemple



```

<html>
<head>... </head>
<body>

  <h1 style="text-align: center;
           color: red;" >
    Exemple CSS </h1>
  <p> Ceci est un texte en
    <span style="font-weight: bold;">
      gras </span>.
  </p>

</body>
</html>

```

- Problème :

- Devoir répéter les styles partout dans le document

CSS: décoration

- **Décoration des balises** : la méthode simple, mais moins bête
 - On peut définir **l'ensemble de sélecteurs** pour un document dans l'entête du document HTML (`<head> ... </head>`)
 - Balise HTML `<style>`

Définition de la mise en forme pour **tous** les `<h1>` et les `<p>` du document

```
<head> ....  
  <style type="text/css">  
    h1 { text-align: center ;  
         font-size: large ;  
        }  
    p  { text-align: justify;  
         font-size: normal; }  
  </style>  
</head>
```


CSS : décoration

- Décoration des balises : la bonne méthode
 - Création d'une feuille de style dans un document à part
 - Balise `<link ...>` dans l'entête `<head>... </head>`
`<link rel="stylesheet" href="styles.css" />`

```
...  
<link ...  
href="styles.css" />  
...
```

Document HTML

```
h1 {  
  text-align: center ;  
  font-size: large ;  
}  
...
```

Document CSS

CSS : décoration

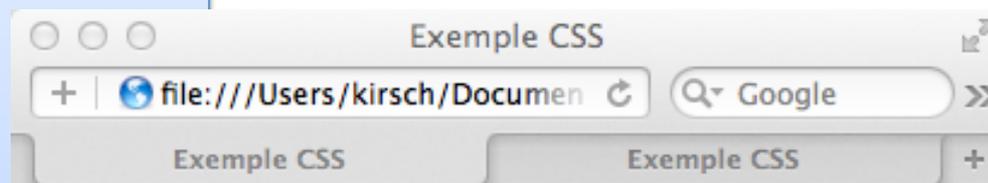
- Exemple

CoursCSS-3.html

```
<head> ...
  <title> Exemple CSS</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <h1> Exemple CSS </h1>
  <p> Ceci est un paragraphe justifié
    ...</p>
  <p> ...</p>
</body>
```

style.css

```
h1 { text-align: center ;
      font-size: large ;
      color: blue;      }
p { text-align: justify;
     font-size: 12pt;  }
```



Exemple CSS

Ceci est un paragraphe justifié. blabla blabla texte blabla blabla
blabla texte blabla blabla blabla texte blabla blabla blabla texte
blabla blabla blabla texte blabla blabla blabla texte blabla
blabla blabla texte blabla blabla blabla texte blabla blabla
blabla texte blabla

bla bla bla texte texte bla blabla bla bla bla texte texte bla
blabla bla bla bla texte texte bla blabla bla bla bla texte texte
bla blabla bla bla bla texte texte bla blabla

• Tous ensemble...

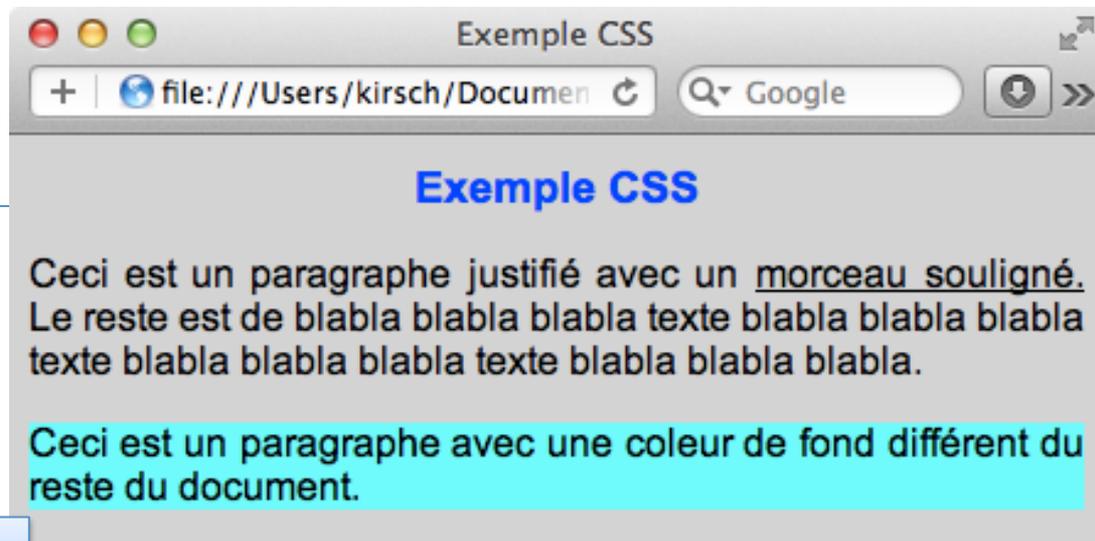
Application en cascade :

Le style qui s'applique à
body s'applique à ses
descendants...

css/monstyle.css

```
body { font-family: Arial, sans-serif;  
       text-align: justify;  
       color: black; }  
p { font-size: 12pt; }  
h1 { font-size: large ;  
     color: blue;  
     text-align: center ; }
```

... par contre, on peut
toujours redéfinir un style.
C'est la **surcharge**.



```
<head> ...  
<link rel="stylesheet" href="css/monstyle.css" />  
<style type="text/css">  
    body { background-color: lightgray; }  
</style>  
</head>  
<body>  
    <h1> Exemple CSS </h1>  
    <p> Ceci est un paragraphe justifié avec un  
<span style="text-decoration: underline">  
morceau souligné. </span> ... </p>  
    <p style="background-color: cyan"> Ceci est un  
paragraphe ...</p>  
</body>
```

CSS : propriétés

- Quelques propriétés
 - **Texte** : *text-align, text-decoration, text-indent, font-weight, font-family, font-style, font-size*
 - Indications de taille : 120% , 12**pt** , 5**px** , 10**em**
 - **Couleurs et fond** : *background-color, background-image, background-repeat*
 - Indication couleurs : *red*, **RGB(255,0,0)**, **RGBA(255, 0, 0, 0.5)**
 - Répétition fond : *repeat-x, repeat-y, no-repeat*
 - **Listes** : *list-style-type, list-style-image*
 - Styles listes **ul** : *disc, circle, square, none*
 - Styles listes **ol** : *decimal, lower-roman, upper-roman, lower-latin, none...*
 - Listes avec une image : **URL(image.jpg)**

CSS : sélecteurs

- Différents types de sélecteurs possibles

Sélecteur pour une balise

```
Balise { propriété: valeur; ... }  
body { font-family: Arial, sans-serif; color: black; }
```

Sélecteur pour certaines occurrences d'une balise

```
Balise.classe { propriété: valeur; ... }  
p.italique { font-style: italic; }  
<balise class="classe">...  
<p class="italique"> ... </p>
```

Sélecteur pour une « classe » quelque soit la balise

```
.classe { propriété: valeur; ... }  
.boite { background-color: rgba(125,125,125, 0.5); }  
<balise class="classe">...  
<p class="boite"> ... </p>
```

Sélecteur à appliquer dans un certain identificateur

```
#identificateur { propriété: valeur ... }  
#liste { font-weight: bold; }  
<balise id="identificateur">...  
<ol id="liste"> ... </p>
```

CSS : sélecteurs

- Feuille CSS : *selecteurs.css*

```
body { font-family: Arial, sans-serif;  
        text-align: justify;  
        color: black;  
    }
```

```
ol { list-style-type: lower-roman; }
```

```
h1 { text-align: center ;  
      font-size: large ;  
      color: rgb(0, 0, 255);  
    }
```

```
p.italique { font-style: italic;  
             text-align: center;  
            }
```

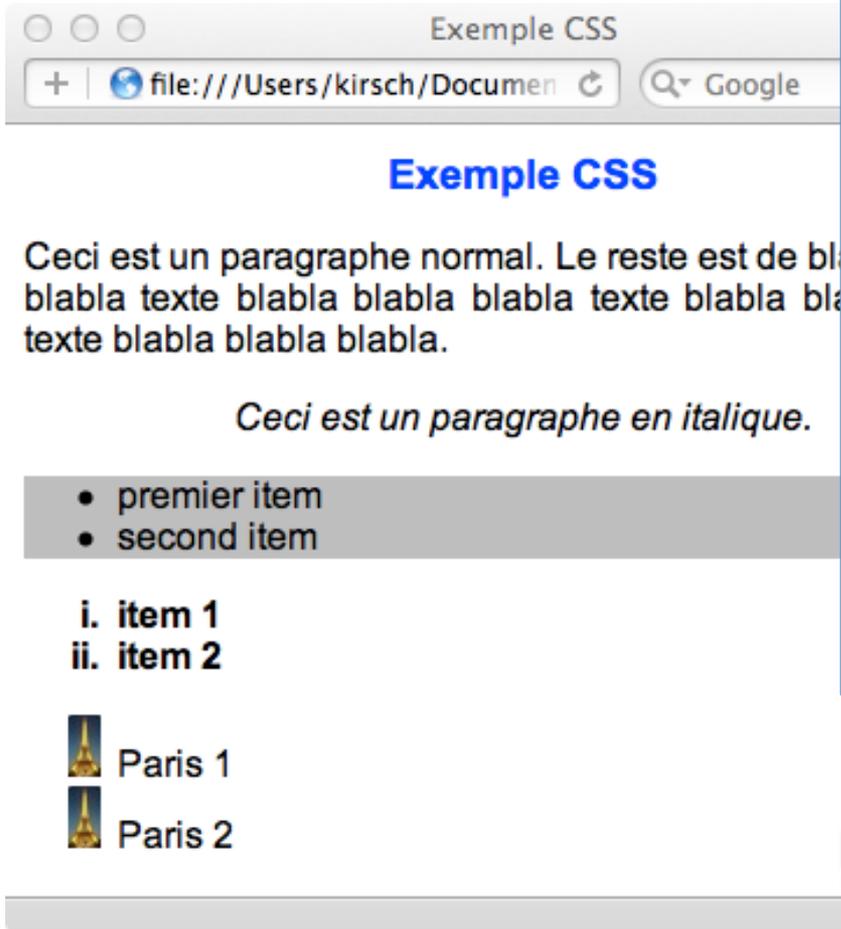
```
ul.paris {  
    list-style-image: url("../img/tour.jpg");  
}
```

```
.boite {  
    background-color: rgba(125,125,125, 0.5);  
}
```

```
#liste { font-weight: bold; }
```

CSS : sélecteurs

- Page HTML



```

<head> ...
  <link rel="stylesheet"
        href="css/selecteurs.css" />
</head>
<body>
  <h1> Exemple CSS </h1>
  <p> Ceci est un paragraphe normal. ... </p>
  <p class="italique"> Ceci est un ... </p>
  <ul class="boite">
    <li> ... </li>
    <li> ... </li>
  </ul>
  <ol id="liste">
    <li> item 1 </li>
    <li> item 2 </li>
  </ol>
  <ul class="paris">
    <li> Paris 1 </li>
    <li> Paris 2 </li>
  </ul>
</body>

```

CSS : tableaux

- **Mise en forme des tableaux : propriétés de base**
 - **Épaisseur de bordure** : ***border-width*** et ***border-XX-width***
(XX = top, bottom, right, left)
border-width: 2px; border-bottom-width: 4px;
 - **Style de bordure** : ***border-style*** et ***border-XX-style***
 - Valeurs : *none, dotted, dashed, solid, double...**border-style: dotted; border-bottom-style: solid;*
 - **Couleur de bordure** : ***border-color*** et ***border-XX-color***
border-color: black;
- **Autres propriétés intéressantes :**
 - **Séparation des bordures** : ***border-collapse***
border-collapse: collapse; ou border-collapse: separate;
 - **Espacement des bordures** : ***border-spacing***
border-spacing: 10px;

CSS : tableaux

- Exemple : tableaux.css

```
table.type1 { border-style: solid;
border-width: 2px;
border-color: black;
border-collapse: collapse;
text-align: center;
width: 50%; }
```

```
table.type1 th { border-bottom-style: solid;
border-bottom-width: 4px;
background-color: rgb(230,230,230); }
```

```
table.type1 td { border-style: dotted;
border-width: 2px ;
color: black;
font-style: italic; }
```

Colonne1	Colonne 2
<i>data 1</i>	<i>data 2</i>
<i>data 3</i>	<i>data 4</i>

Définition de **table** classe « **type1** » :
bordures collées noir de 2px

Définition « **table.type1 th** » :
balise <th> dans une <table class="type1">
Bordure inférieur solide en 4px, fond gris

Définition « **table.type1 td** » :
balise <td> dans une <table class="type1">
Bordure pointillée en 2px, police en italique

CSS : tableaux

- Exemple : tableaux.css

```
table.type2 {
  border: 1px solid black;
  border-collapse: separate;
  border-spacing: 10px;

  margin: auto;
  empty-cells: hide;
  border-radius: 15px;
}
```

```
table.type2 td {
  width: 120px;
  border: 1px solid red;
}
```

Tableau de type2

Colonne1	Colonne 2
data 1	data 2
data 3	data 4
data5	data6

Définition de **table** classe « **type2** » :
Bordures solides noir de 1px
Cellules séparée, espacement 10px

Emplacement de la table « **margin: auto** »
Cellules vides cachées « **empty-cells** »
Coins arrondis « **border-radius** »

Définition « **table.type2 td** » :
balise <td> dans une **<table class="type2">**
Bordure rouge solide de 1px
Taille de la cellule 120px

CSS : tableaux

- Le code HTML

```

<head> ...
  <link rel="stylesheet" href="css/selecteurs.css" />
  <link rel="stylesheet" href="css/tableaux.css" />
</head>
<body> ...
  <table class="type1">
    <caption> Tableau de type1 </caption>
    <thead>
      <tr> <th> ... </th> <th> ... </th> </tr>
    </thead>
    <tbody>
      <tr> <td> data 1 </td> <td> data 2 </td> </tr>
      <tr> <td> data 3 </td> <td> data 4 </td> </tr>
    </tbody>
  </table>
  ...
  
```

Exemple CSS

file:///Users/kirsch/Docume

Exemple CSS

Tableaux CSS

Tableau de type1

Colonne1	Colonne 2
data 1	data 2
data 3	data 4

Tableau de type2

Colonne1	Colonne 2
data 1	data 2
data 3	data 4
data5	data6

CSS : tableaux

- Le code HTML

...

<table class="type2">

<caption> Tableau de type2 </caption>

<thead>

<tr> <th> ...</th> <th> ...</th> </tr>

</thead>

<tbody>

<tr> <td> data 1 </td> <td> data 2 </td> </tr>

<tr> <td> data 3 </td> <td> data 4 </td> </tr>

<tr> <td rowspan="2"> data5</td>

<td> data6</td> </tr>

<tr> <td> </td> </tr>

</tbody>

</table> ...

Tableaux CSS

Tableau de type1

Colonne1	Colonne 2
data 1	data 2
data 3	data 4

Tableau de type2

Colonne1	Colonne 2
data 1	data 2
data 3	data 4
data5	data6

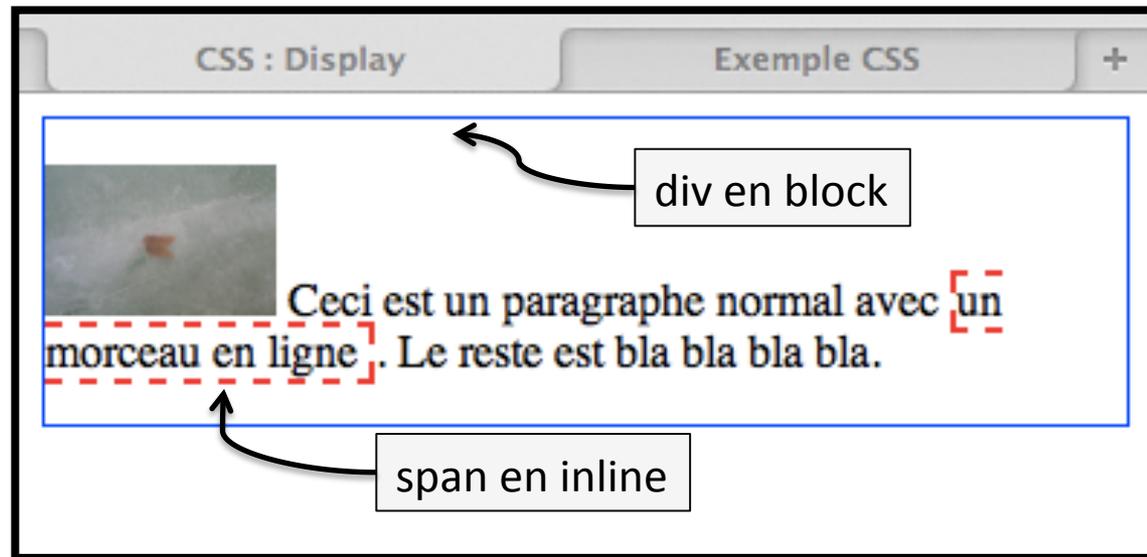
<td rowspan="2">

Cellule vide
<td></td>

CSS : mise en boîte

- **Propriété Display** : balises en **block** ou **inline**
 - Selon la propriété Display, les balises se présentent comme un **bloc**, ou sont rendues sur une **même ligne**

```
<style type="text/css">
.bloc { border: 1px solid blue;
        display: block;
      }
.ligne { border: 2px dashed red;
         display: inline;
      }
</style>
```



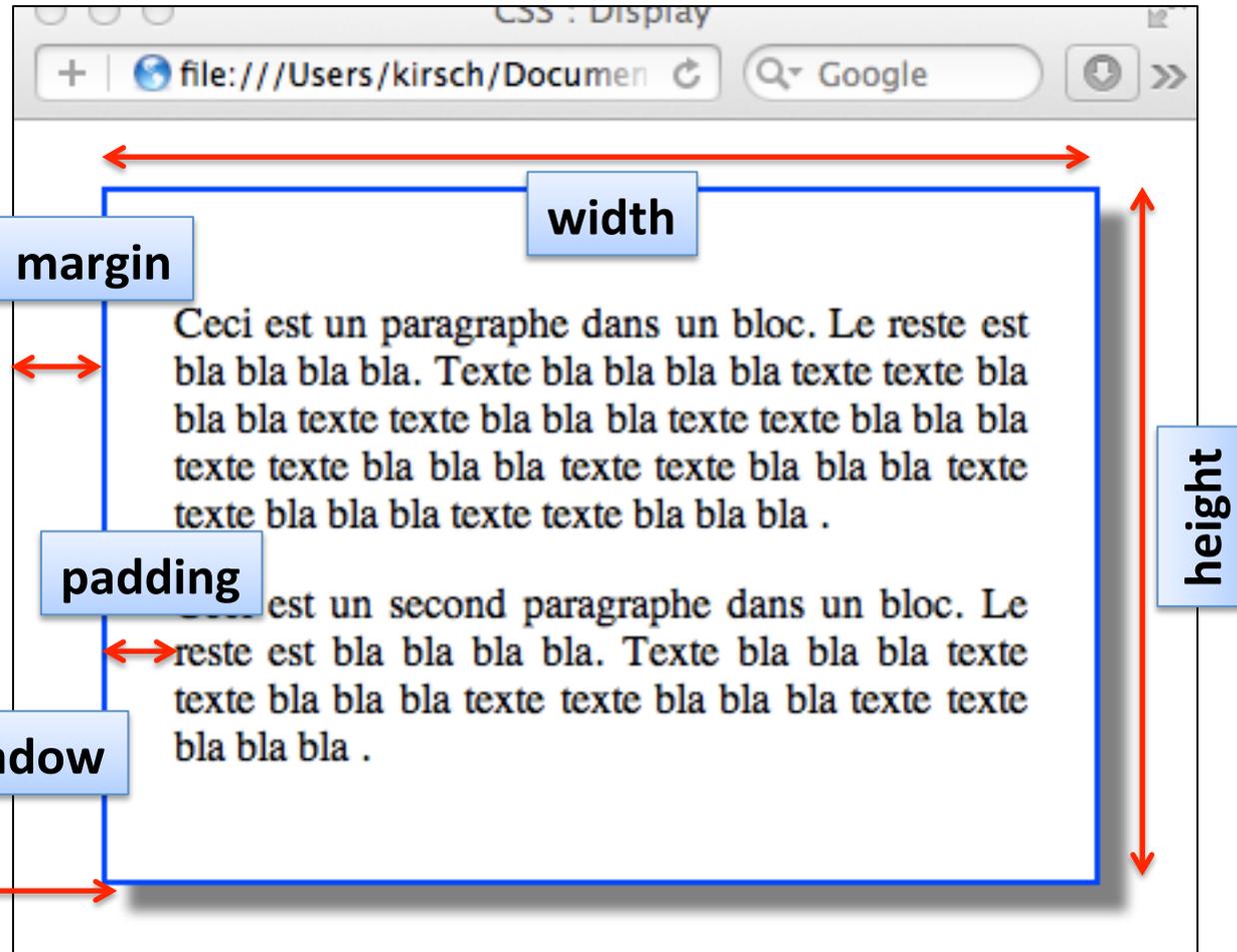
```
<div class="bloc"> <p> Ceci est ...
  <span class="ligne"> un morceau en ligne </span>. ... </p>
</div>
```

CSS : mise en boîte

- Les distances dans les blocs

```
div {
  border: 2px solid blue;
  display: block;
  margin: 25px;
  padding: 25px;
  width: 75%;
  box-shadow: 10px 10px
              5px gray;
  text-align: justify;
}
```

box-shadow:
offset-hor offset-vert
ombre couleur;

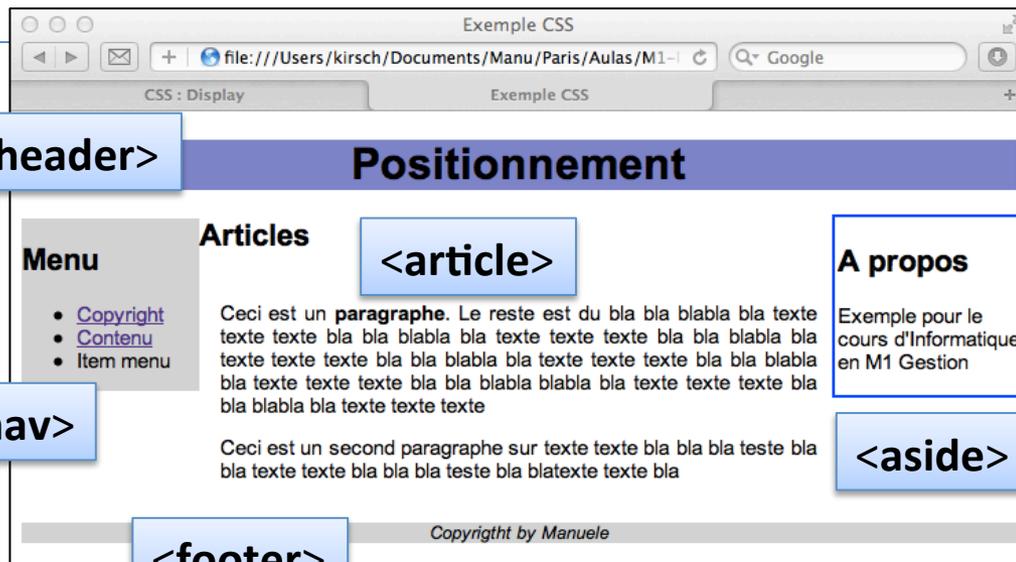


CSS : mise en boîte

- **Positionnement des blocs**
 - Position fixe dans la page : « ***position : absolute*** »
 - Propriétés ***top, bottom, left, right***
 - Position relative : « ***position : relative*** »
 - Positionnement par rapport aux éléments alentours
 - Flottement : propriété ***float***
 - L'élément flotte à droite (« ***float : right*** ») ou à gauche (« ***float : left*** ») des autres éléments

CSS : mise en boîte

- Exemple :



<header>

```
header { display: block;
... }
```

```
nav { display: block;
float: left;
width: 18%;
background-color: lightgray; }
```

<nav>

<article>

```
article { display: block;
margin: auto;
padding: 3px;
width: 60%;
... }
```

```
footer { clear: both;
background-color: lightgray;
text-align: center;
font-size: 10pt; }
```

<footer>

A propos
Exemple pour le cours d'Informatique en M1 Gestion

<aside>

```
aside {
display: block;
position: absolute;
top: 20%;
right: 2px;
width: 18%;
padding: 2px;
border : 2px solid blue;
}
```

CSS : pseudo-classes

- Il existe certains sélecteurs spéciaux

- **a:link** → liens

```
a:link { color: rgb(0,0,250); }
```

- **a:visited** → liens visités

```
a:visited { color: rgb(135,45,225); }
```

- **a:active** → liens actifs

- **Balise:hover** → passage de la souris

```
a:hover , p:hover {  
  background-color:  
    rgb(120,135,155);  
  color: yellow; }
```

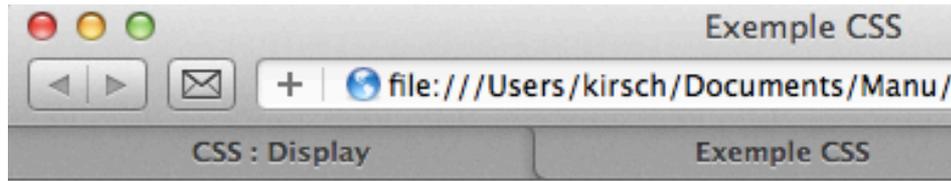
- **Balise:first-letter** → première lettre

- **Balise:nth-child(odd | even)** → n^{ème} balise pair/impair

```
p:nth-child(even) {  
  background-color: rgb(200,200,200); }
```

```
p:first-letter {  
  font-size: 20pt;  
  font-family: cursive; }
```

CSS : pseudo-classes

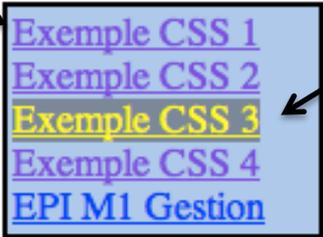


```
div.contenu { display: block;
float: right;
width: 70%;
margin: 3px;
font-size: 11pt; }
```

a:visited

Pseudo-classes CSS

a:hover



a:link

```
div.menu { display: block;
float: left;
width: 20%;
padding: 2px;
border: 2px solid black;
background-color:
rgb(170,200,235); }
```

Ceci est un paragraphe. Le reste est de blabla blabla blabla texte blabla blabla blabla texte blabla blabla blabla texte blabla blabla blabla.

Ceci est un paragraphe. Le reste est de blabla blabla blabla texte blabla blabla blabla texte blabla blabla blabla.

Ceci est un paragraphe. Le reste est de blabla blabla blabla texte blabla blabla blabla texte blabla blabla blabla.

p:nth-child(even)

p:first-letter

CSS : Conseils utiles

- Éviter de mettre des attributs de style directement dans la page
 - créer autant de feuilles de styles que nécessaire
- Possibilités :
 - une feuille de style générale pour le site
 - polices utilisées, couleurs de bases, tailles, forme des pages...
 - des feuilles de style par catégorie de page dans le site
- Penser et préparer le site avant de coder les pages

Informatique

Modélisation UML

Objectifs de la séance :

Création d'un site Web dynamique
PHP

- **PHP** est un langage de programmation utilisé pour la construction de sites Web dynamiques
 - **Pages PHP** : pages Web qui contiennent de PHP
 - On va mélanger le PHP au code HTML / CSS
 - Le code PHP va être analysé par le serveur
 - Le résultat va être une nouvelle page Web mise à jour automatiquement par le code PHP

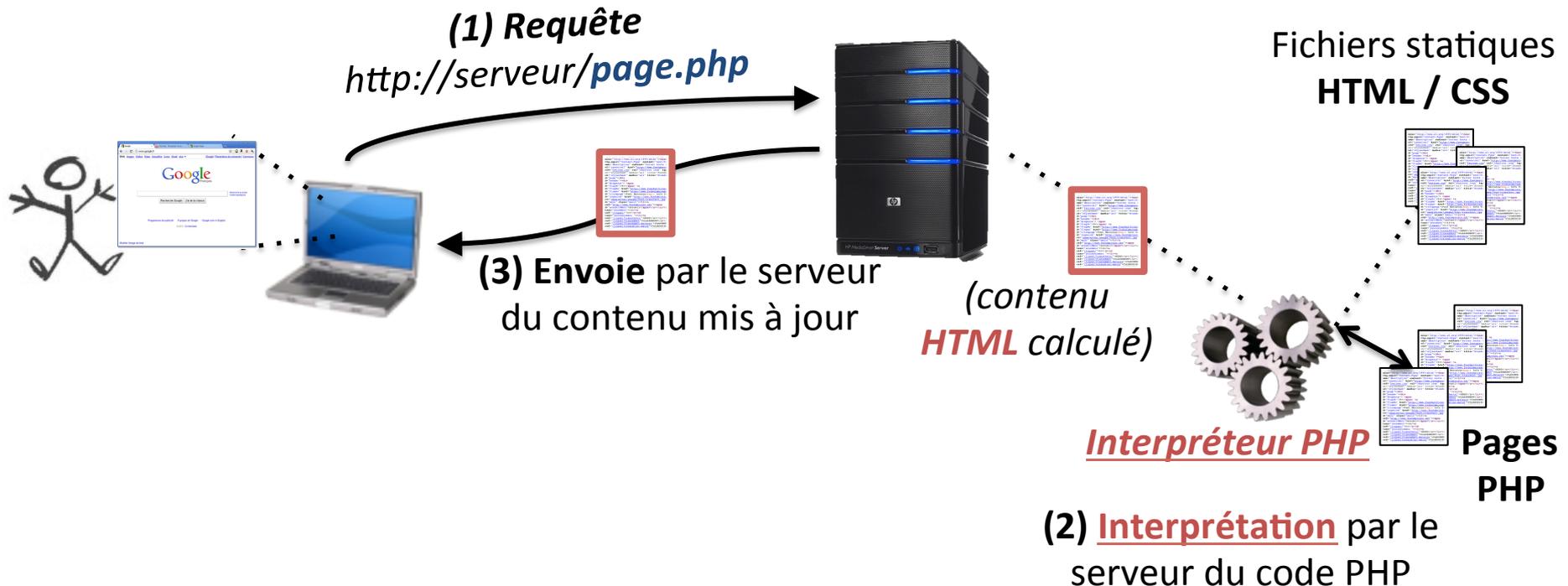
coursPHP-1.php

Le code PHP est à l'intérieur de la balise
`<?php ... ?>`
ou entouré par la balise
`<script language="php">`
`... </script>`

```
<html> ...  
  <?php  
    date_default_timezone_set("Europe/Paris");  
    echo "<p style='font-style: italic;'> Paris, le "  
      .date('d / m / Y')."</p>" ;  
  ?>  
... </html>
```

• Cycle de vie d'une page PHP

- 1) Le **client** envoie la requête au serveur
- 2) Les pages PHP sont analysées par le serveur, le code PHP est interprété
- 3) Le **contenu** de la page est **mise à jour automatiquement** et envoyé au client



PHP



```
...  
<h1> Bienvenue sur le site PHP </h1>  
<p style='font-style: italic;'> Paris, le  
28 / 02 / 2012</p>  
<p> Il est 18:45:12 .</p>  
...
```



<http://serveur/courPHP-1.php>



coursPHP-1.php

```
...  
<h1> Bienvenue sur le site PHP </h1>  
<?php  
date_default_timezone_set("Europe/Paris");  
echo "<p style='font-style: italic;'> Paris, le "  
    .date('d / m / Y')."</p>" ;  
?>  
<script language="php">  
    echo "<p> Il est ".date("H:i:s")." .</p>" ;  
</script>  
...
```

Contenu **HTML/CSS**
calculé



Interpréteur PHP

Page **PHP** originelle
(PHP & HTML / CSS)

- Exemple code PHP

coursPHP-2.php

Toute instruction PHP se termine par un « ; »

Portion de code PHP
<?php ... ?>

```
<!DOCTYPE html>
<?php
date_default_timezone_set("Europe/Paris");
$today = date("d-m-Y H:i:s");
$variable = "PHP5";
```

```
?>
<html> <head> ... </head>
<body>
<h1> Exemple PHP </h1>
<p>Contenu statique : ça ne change pas </p>
```

La séquence */*...*/* délimite un **commentaire**, visible pour l'auteur, invisible pour le client

Portion de code PHP

```
<?php
/* ce contenu va être interprété par le serveur */
echo "<p>Contenu en $variable </p>";
echo "<p> Aujourd'hui c'est le $today </p>";
```

L'instruction « **echo** » permet d'écrire dans le document final

```
?>
</body> </html>
```

- Code une fois interprété **par le serveur...**

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1> Exemple PHP </h1>
    <p>Contenu statique : ça ne change pas </p>
    <p>Contenu en PHP5 </p> <p> Aujourd'hui
c'est le 28-02-2012 19:38:54 </p>
  </body>
</html>
```

Résultat des
instructions « echo »

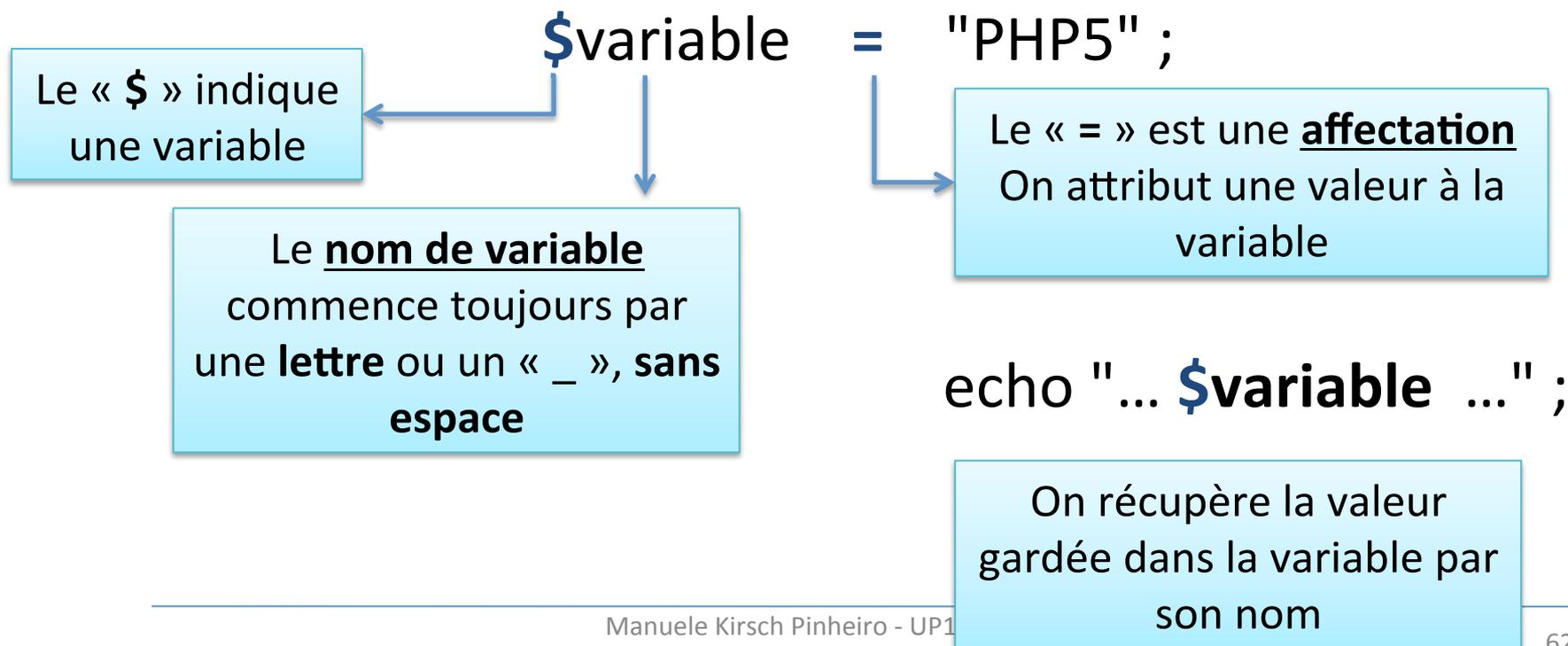


Ce qui le client voit...



- **La notion de variable**

- Une variable est **un conteneur de valeur**
- On peut lui affecter une valeur, qu'on va utiliser plus tard



- **La notion de variable : les types des données**
 - Les variables peuvent garder de valeurs de différents types
 - Nombres entiers (**integer**) : 25
 - Nombres décimaux (**double** ou **float**) : 2.25
 - Chaînes de caractères (**string**) : « 1 super chaîne ! »
 - Logique (**boolean**) : « true » (1) ou « false »
 - La fonction **gettype(\$variable)** permet de savoir quelle type de valeur contient la variable
 - \$entier = 25; gettype(\$entier) → integer
 - \$decimal = 2.25; gettype(\$decimal) → double
 - \$chaine = "1 super chaîne !"; gettype(\$chaine) → string
 - \$bool = true; gettype(\$bool) → boolean

- Exemple :

```
<?php
```

```
$entier = 25;
```

```
$decimal = 2.25;
```

```
$chaine = "1 super chaîne !";
```

```
$boolean = true;
```

```
echo "<li>" . gettype($entier) . ": $entier </li>";
```

```
echo "<li>" . gettype($decimal) . ": $decimal </li>";
```

```
echo "<li>" . gettype($chaine) . ": $chaine </li>";
```

```
echo "<li>" . gettype($boolean) . ": $boolean </li>";
```

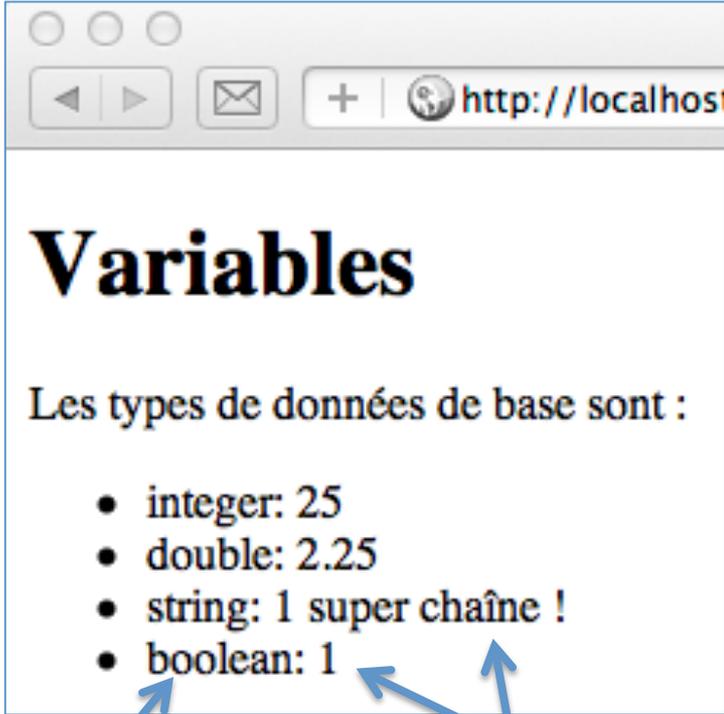
```
?>
```

Définition d'une variable

On récupère la valeur de la variable **\$boolean**

gettype() informe le type de la variable

Valeur de chaque variable



- **Opérateurs**

- Différents opérateurs permettent de manipuler des valeurs, qu'ils soient dans les variables ou pas

Opérateurs mathématiques	Opérateurs String	Opérateurs de comparaison	Opérateurs logiques
+ - * / %	. (concaténation)	== != <=< >=>	(OR) && (AND) ! (not)

<?php

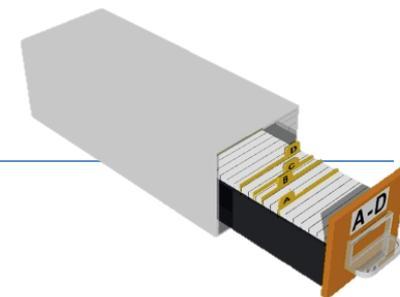
```
$a = 2 + 3 ;
$b = 4 - $a ;
$nom = "Toto";
echo "Salut " . $nom;
echo "<p> 4 - $a vaut $b </p>";
```

?>

Exemple avec les opérateurs :

Salut Toto

4 - 5 vaut -1



- **Tableaux**

- Variables capables d'enregistrer plusieurs valeurs d'un type

- **Tableaux à indice :**

- Chaque position est identifiée par un numéro (commençant par 0)

- \$tableau [0] = "A";
- \$tableau [1] = "B";
- \$tableau [3] = "Fin";
- \$tableau [] = "Suite";

Attention a définir toutes les positions avant de les utiliser ou il y aura un **message d'erreur.**



- **Tableaux associatifs :**

- Chaque position reçoit un identifiant (un label)

- \$tableauAssoc ["Prenom"] = "Jean";
- \$tableauAssoc ["Nom"] = "Dupont";



```

<head> ...
    <style>... </style>
</head>
<body> ...
    <h2>Tableaux à indice </h2>
    <table>
    <?php
        $tableau [0] = "A";
        $tableau [1] = "B";
        $tableau [3] = "Fin";
        $tableau [] = "Suite";

        echo "<tr> <td>". $tableau[0] . "</td> <td>". $tableau[1]
            . "</td> <td>". $tableau[2] . "</td><td>". $tableau[3]
            . " </td><td>". $tableau[4] . "</td></tr>";
    ?>
</table>
...
    
```

Tableaux en PHP

localhost:8888/exemplesPHP/coursPHP-6.php

Tableaux

Tableaux à indice

Notice: Undefined offset: 2 in /Users/kirsch/Documents/Sites/exemplesPHP/coursPHP-6.php on line 29

Alice	B		Fin	Suite
-------	---	--	-----	-------

Message d'erreur car le contenu de la position 2 (\$tableau[2]) n'a pas été défini auparavant.

Contenu de la position 4 (\$tableau[4])

```

...
<h2>Tableau associatif </h2>
<table>
  <tr> <th> Nom </th> <th>Prénom </th> </tr>

  <?php
    $tableauAssoc ["Prenom"] = "Jean";
    $tableauAssoc ["Nom"] = "Dupont" ;

    echo "<tr> <td>" . $tableauAssoc ["Nom"] . "</td>" ;
    echo "<td>" . $tableauAssoc ["Prenom"] . " </td></tr>" ;
  ?>

</table>
</body>

```

Tableau associatif

Nom	Prénom
Dupont	Jean

Informatique

Modélisation UML

Objectifs de la séance :

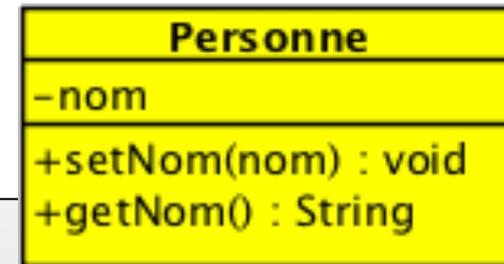
Classes & Objets

PHP orienté objets

- **PHP 5 est un langage « orienté objets »**
- Manipulation de classes et d'objets

Visibilité :
private \$nom

Définition de classe
class Personne



```
<?php
class Personne {
    private $nom;
```

Définition d'un attribut

Opération :
public function...

```
    public function setNom ( $nouvNom )
    { $this->nom = $nouvNom; }
```

Accès à un attribut
\$this->attribut

```
    public function getNom () {
        return $this->nom;
    }
}
```

Retourner une valeur
return valeur ;

PHP orienté objets

• Classes & Objets

– Création d'un objet : `$obj = new classe() ;`

```
...  
<?php  
$toto = new Personne();  
  
$toto->setNom("Toto");  
  
echo "<p> ... " . $toto->getNom() . "</p>";  
  
$toto->nom = "blablaba";  
echo " <p> " . $toto->nom . " </p> ";  
?>
```

Création d'un objet
`$toto = new Personne ()`

Accès aux opérations
publiques
`$toto->setNom("Toto")`
`$toto -> getNom ()`

Impossible d'accéder aux
attributs **privés**



PHP orienté objets

```
<html> <head> ...  
  <?php  
    class Personne {  
      private $nom;  
      ...  
    } //fin classe Personne  
  ?>  
</head>  
<body> ...  
  <?php  
    $toto = new Personne();  
    $toto->setNom("Toto");  
    echo "<p> Objet <i>Personne</i> : " . $toto->getNom() . "</p> ";  
    ...  
    echo " <p> " . $toto->nom . " </p> ";  
  ?>  
</body> </html>
```

Classes et Objets en PHP

Classes et objets en PHP

Objet *Personne* : Toto

Teste de l'accès (ça ne doit pas marcher) :

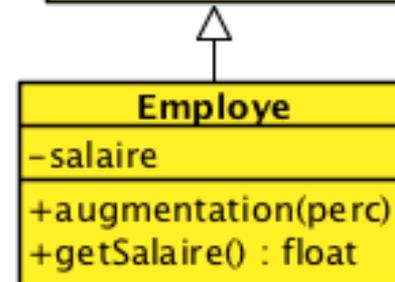
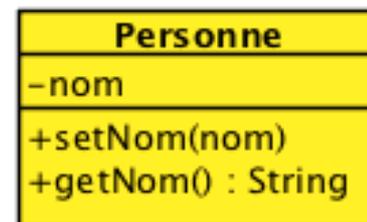
Fatal error: Cannot access private property Personne::\$nom in
/Users/kirsch/Documents/Sites/exemplesPHP/coursPHP-2013-1.php
on line 35

Erreur car l'attribut
« nom » est privé !!

PHP orienté objets

- **Classes & objets :**

- **Héritage :** `class SousClasse extends SuperClasse`



```
class Employe extends Personne {
    private $salaire = 1000;

    public function augmentation ($perc) {
        if ($perc > 0) {
            $this->salaire = $this->salaire +
                $this->salaire*$perc;
        }
    }

    public function getSalaire()
    { return $this->salaire; }
}
```

La classe **Employe** hérite de la classe **Personne**
class Employe extends Personne

On rajoute un nouvel attribut
private \$salaire
 Et des nouvelles opérations
public function augmentation
public function getSalaire

Fichier Employe.php

PHP orienté objets

- **Classes & objets**

```
<?php
include "Employe.php" ;

$toto = new Employe();

$toto->setNom("Toto");
$toto->augmentation(0.10);

echo "<i> nom </i> : " . $toto->getNom() ;
echo " <i> salaire </i> : " . $toto->getSalaire() . " € </p>";
?>
```

On importe la définition des classes
Employe et **Personne**

Toto est un **Employé**, il est donc
une **Personne**

Toto possède un **salaire (Employe)**,
mais aussi un **nom (Personne)**

La classe **Employe** hérite tous les
attributs et **opérations** de **Personne**

PHP orienté objets

• Classes & objets

```
<html> <head> . . . </head>
<body> <h1>Objets en PHP</h1>
<?php
  include "Employe.php" ;

  $toto = new Employe();

  $toto->setNom("Toto");
  $toto->augmentation(0.10);

  echo "<p>Objet Employe : </p> <ul>" ;
  echo "<li> <i> nom </i> : " . $toto->getNom() . "</li>";
  echo "<li> <i> salaire </i> : " . $toto->getSalaire() . " € </li>";
  echo "</ul>" ;
?>
</body> </html>
```



PHP orienté objets

- **Classes & objets**
 - Méthode **constructeur** : **__construct**
 - **Redéfinition** d'une opération

```
class Manager extends Employe {  
    private $bonus ;  
    function __construct ($bon)  
    { $this->bonus = $bon; }  
  
    public function getSalaire() {  
        return parent::getSalaire() + $this->bonus;  
    }  
    public function setBonus ($nouvBon) { ... }  
    public function getBonus () { ... }  
}
```

Le **constructeur** est appelé chaque fois qu'un objet est créé (**new**)

Redéfinition de l'opération **getSalaire**
parent::getSalaire correspond à l'opération **getSalaire** définie par la super-classe (**Employe**)

PHP orienté objets

- Classes & objets

```
<?php
require "Manager.php" ;

$toto = new Manager(400);

$toto->setNom("Toto");
$toto->augmentation(0.10);

echo "<p><i>Manager</i> : ". $toto->getNom()
. ", salaire ". $toto->getSalaire() . " € "
. ", bonus " . $toto->getBonus() . "</p>";

?>
```

Appel au **constructeur** :

```
function __construct ($bon)
{ $this->bonus = $bon; }
```

Appel à l'opération **getSalaire**
de la classe **Manager**

← → ↻ 🏠 🌐 localhost:8888/exe

Objets en PHP

Manager : Toto, salaire 1500 € , bonus 400

Informatique

Modélisation UML

Objectifs de la séance :
Formulaires HTML & PHP

PHP est un langage pour le Web

- **Communication entre le client (navigateur) et le serveur (php)**
 - Les **formulaires en HTML** permettent de recueillir des données auprès de l'utilisateur
 - Les données sont ensuite **communiquées** à un **programme**
 - Le navigateur **envoie les données** récoltées par les formulaires au serveur



Formulaires HTML

- Un **formulaire HTML** est défini par la balise **<form ...> ... </form>**

– Tous les éléments sont à l'intérieur de la balise **<form name="nomFormulaire"**

action="page.php" method="get | post" > </form>

action : à qui on envoie les données

method: comment on envoie les données

– Les **champs du formulaire** sont introduits par différents balises :

- **<input type="..." name="..." value="..." id="..." />**
- **<textarea name="..." id="..." cols="..." rows="..." > ... </textarea>**
- **<select name="..." id="..." size="..." >**
<option value="..." > ... </option> </select>

Formulaires HTML

The screenshot shows a web browser window with the URL `http://localhost:8888/exem`. The form contains the following elements:

- Nom:** A text input field containing "votre nom".
- Produit:** A dropdown menu showing "Super Kdo".
- Opinion:** A text area containing "Votre opinion sur nos produits".
- Buttons:** "Envoyer" (submit) and "Nettoyer" (reset).

Arrows connect the following HTML code snippets to their respective elements in the form:

- `<input type="text" name="nomClient" value="votre nom" size="40" maxlength="150" />` points to the "Nom" text input.
- `<select name="produit"> <option value="SuperKdo"> Super Kdo </option> ... </select>` points to the "Produit" dropdown menu.
- `<textarea name="opinionClient" cols="40" rows="5" > Votre opinion sur nos produits </textarea>` points to the "Opinion" text area.
- `<input type="submit" value="Envoyer" />` points to the "Envoyer" button.
- `<input type="reset" value="Nettoyer" />` points to the "Nettoyer" button.

input type="submit" se charge d'envoyer les données du formulaire

Formulaires HTML

```
<form name="formClient" action="coursPHP-7.php" method="POST" >
```

```
<label for="nom">Nom</label>
```

```
<input type="text" id="nom" name="nomClient"  
value="votre nom" size="40" maxlength="150" /> <br/>
```

À qui les
données sont
envoyées

```
<label>Produit</label>
```

```
<select name="produit">
```

```
<option value="SuperKdo">Super Kdo</option>
```

```
<option value="MegaTruc">Mega Truc</option>
```

```
<option value="BabyFun">Baby Fun</option>
```

```
</select> <br/>
```

input type="text"
Zone de saisie

select ... option
Liste de sélection
d'options

```
<label>Opinion</label>
```

```
<textarea name="opinionClient" cols="40" rows="5" >
```

```
Votre opinion sur nos produits </textarea> <br/>
```

textarea
Zone de texte

```
<input type="submit" value="Envoyer" class="bouton" />
```

```
<input type="reset" value="Nettoyer" class="bouton" />
```

```
</form>
```

input type="submit"
Input type="reset"
Boutons d'envoi et de
reset du formulaire

Formulaires HTML & PHP

```
<form name="formClient" action="coursPHP-7.php"
      method="POST" >
  <label for="nom">Nom</label>
  <input type="text" id="nom"
        name="nomClient" value="votre nom"
        size="40" maxlength="150" /> <br/>
```

```
...
<?php
  $nom = $_POST["nomClient"];
  $op = $_POST["opinionClient"];
  $prod = $_POST["produit"];
  echo "<p>Merci de votre participation, $nom ! </p>";
  echo "<p>Votre produit est : <i> $prod </i> </p>";
  echo "<p> Votre opinion est : <i> $op </i> </p>";
?>
```

Formulaires HTML & PHP

Formulaire Get

- **Méthode GET**

- Les données sont envoyées dans l'**URL** du programme
- Limitée à 256 octets
- *Déconseillé*

esPHP/coursPHP-8.php?client=toto&mdp=password

```
<form name="formGet"
      action="coursPHP-8.php"
      method="GET">
  <label>Nom</label>
  <input type="text" name="client" size="20" /> <br/>
  <label>Mot de passe </label>
  <input type="password" name="mdp" size="10"/>
  <br/>
  <input type="submit" value="Envoyer" />
</form>
```

Données reçues :
Bienvenue, *toto* !

```
<?php
  echo "<p>Bienvenue, <i>". $_GET["client"] . "</i> ! </p>";
?>
```

Formulaires HTML & PHP

- Exemple

Vos données

Nom :

Email :

Homme Femme

Vos produits

Produit préféré :

Suggestions :

```
<form name="..." action="coursPHP-9.php" method="POST">
<fieldset>
  <legend>Vos données </legend>
  <label >...</label> <input type="text" name="nom" ... /> <br/>
  <label >...</label> <input type="email" name="email" ... /><br/>
  <input type="radio" name="sexe" value="Homme" />Homme
  <input type="radio" name="sexe" value="Femme" /> Femme<br/>
</fieldset>
```

```
<fieldset> <legend>Vos produits </legend>
<label>...</label>
<select name="produit">
  <option value="SuperKdo">...</option>
  <option value="MegaTruc"> Mega Truc</option>
  <option value="BabyFun"> ... </option>
</select> <br/>
<label>...</label>
<textarea name="opinion" ... > ... </textarea>
</fieldset>
```

Formulaires HTML & PHP

- Exemple

```
<body>
  <h1>Récapitulatif </h1>
  <ul>
  <?php
    echo "<li> Nom : " . $_POST["nom"] . "</li>" ;
    echo "<li> Email : " . $_POST["email"] . "</li>" ;
    echo "<li> Sexe : " . $_POST["sexe"] . "</li>" ;
    echo "<li> Produit préféré : " . $_POST["produit"] . "</li>" ;
    echo "<li> Suggestion : " . $_POST["opinion"] . "</li>" ;
  ?>
  </ul>
</body>
```



Récapitulatif

- *Nom : Manuele*
- *Email : mkirschpin@univ-paris1.fr*
- *Sexe : Femme*
- *Produit préféré : MegaTruc*
- *Suggestion : bla bla bla bla*

Informatique

Modélisation UML

Objectifs de la séance :

Instructions de contrôle en PHP

Fonctions

- **Instructions de contrôle**
 - Instructions pour gérer le flot d'exécution
 - **Instructions conditionnelles**
 - Elles conditionnent l'exécution
 - Semblables à un **nœud de Décision** (diagramme activités)
 - *if... else ..., switch ... case ...*
 - **Instructions de boucle**
 - Elles permettent la **répétition** d'un bloc d'instructions
 - *for ... , foreach ... , while ... , do... while*

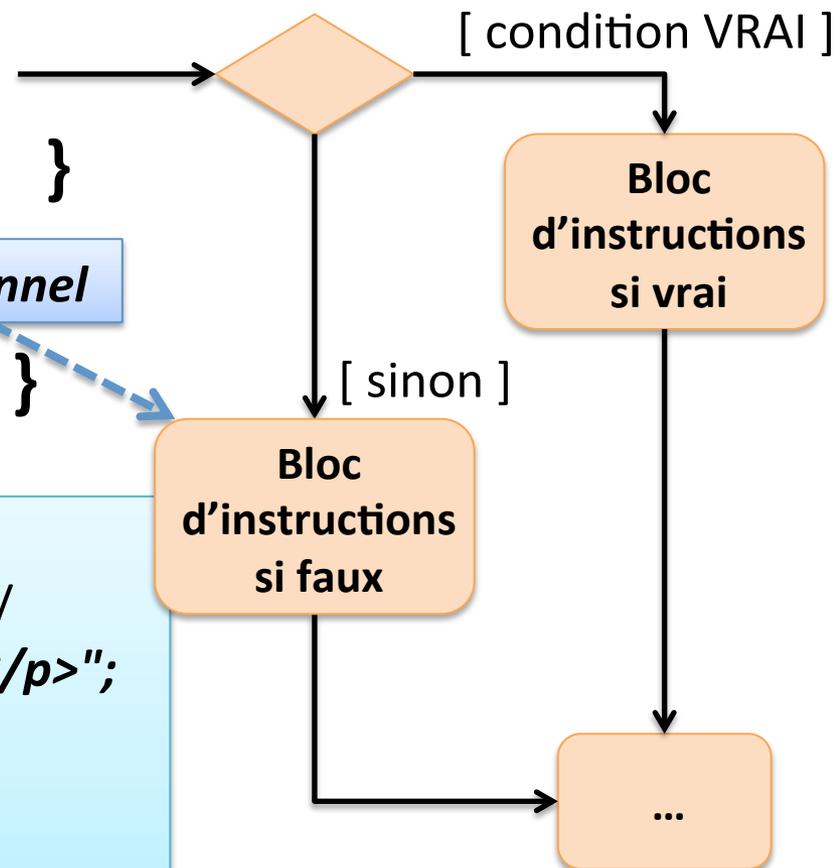
- Instructions conditionnelles **if ... else...**

```
if ( condition )
```

```
{ bloc d'instructions si vrai ; }
```

```
else
```

```
{ bloc d'instructions si faux ; }
```



```
if ( $qte >= 100)
```

```
{ $remise = 0.10; /* remise de 10 % offerte */  
  echo "<p>Vous avez une remise de 10% ! </p>";  
}
```

```
else {
```

```
  $remise = 0.05;  
  echo "<p>Vous avez une remise de 5% </p>";  
}
```

- **Instructions conditionnelles if ... else...**

- Les données pour la condition peuvent venir d'un formulaire

formExemple11.html

```
<form name="..." method="POST"
  action="coursPHP-11.php" >
...
<select name="prix">
  <option value="10">
    Super Kdo - 10€ </option>
  ...
</select>
...
<input type="number" size="10"
  name="qte" />
...
<input type="submit" value="Devis" />
</form>
```

<?php

coursPHP-11.php

```
$qte = $_POST["qte"];
$prixunit = $_POST["prix"];
$remise = 0;

if ( $qte >= 100)
{ $remise = 0.10; /* remise de 10 % offerte */
  echo "<p>Vous avez une remise de 10% ! </p>";
}

$prix = $prixunit * $qte
        - ($prixunit * $qte * $remise);
echo "<p> Pour un prix de <i> $prixunit </i>
  l'unité et <i> $qte </i> unités, vous avez à
  régler <i> $prix </i></p>";

?>
```

PHP

Préparer votre devis

Produit : Super Kdo - 10€

Quantité : 100

Devis

Devis

Vous avez droit à une remise de 10% !

Pour un prix de 10 l'unité et 100 unités, vous avez à régler 900

```
<form name="..." method="POST"
    action="coursPHP-11.php" >
    <label>Produit : </label>
    <select name="prix">
        <option value="10" >Super ... </option>
    ... </select> <br/>
    <label >Quantité : </label>
    <input name="qte" type="number"
        size="10" /> <br/>
    <input type="submit" value="Devis" />
</form>
```

```
<?php
    $qte = $_POST["qte"];
    $prixunit = $_POST["prix"];
    $remise = 0;
    ....
    if ( $qte >= 100)
    { $remise = 0.10;
      echo "<p>Vous avez .... </p>";
    }
    ... ?>
```

- **Instructions conditionnelles if ... else...**

- Les blocs if... else ... peuvent contenir n'importe quelle instruction, y compris d'autres blocs if... else ...

if (*condition1*)

{ *bloc d'instructions si condition1 vraie ;* }

elseif (condition2)

{ *bloc d'instructions si condition2 vraie ;* }

else

{ *bloc d'instructions si les conditions sont fausses ;* }

http://localhost:8888/ex

Préparer votre devis

Produit :

Quantité :

```

<form name="..." method="POST"
  action="coursPHP-12.php" >
...
<select name="prix"> ... </select>
...
<input type="number" ... name="qte"/>
...
<input type="submit" value="Devis" />
</form>
  
```

Devis

Prix unitaire : 10 , Quantité : 100 , Remise : 10 %

Total à régler : 900

PHP

```

<?php
  $qte = $_POST["qte"];
  $prixunit = $_POST["prix"];

  if ( $qte >= 100)
    { $remise = 0.10 ; }
  elseif ( $qte >= 50 )
    { $remise = 0.05 ; }
  else
    { $remise = 0 ; }

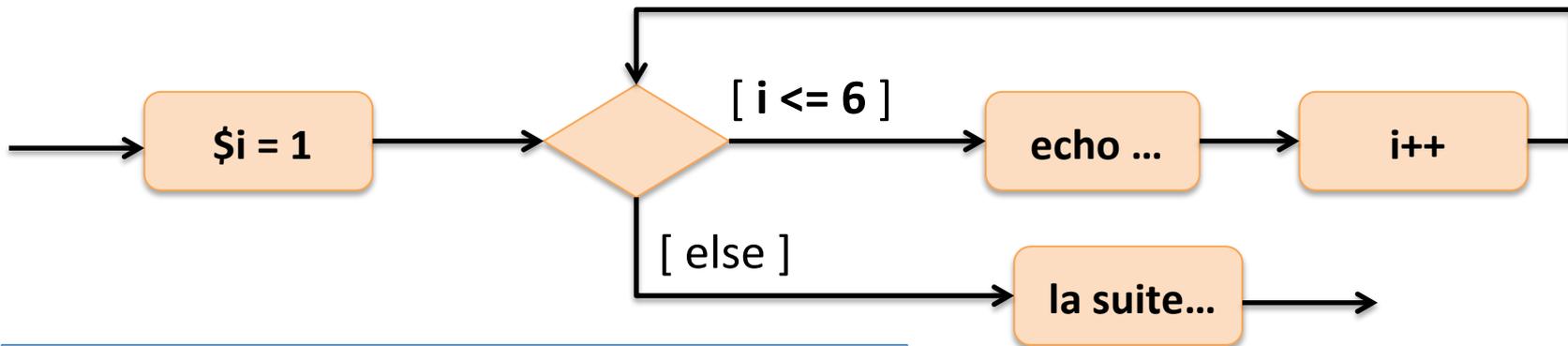
  $prix = $prixunit * $qte
    - ($prixunit * $qte * $remise) ;

  echo "<p> Prix unitaire : <i> $prixunit </i>,
    Quantité : <i> $qte </i>,
    Remise : <i> " . $remise*100 . "</i> % </p>";
  echo "<p><i>Total à régler : </i>
    <b> $prix </b></p>";
  
```

- **Instructions de boucle : for**

- La boucle **for** permet de **répéter** (un certain nombre de fois) l'exécution d'un bloc d'instructions

for (*initialisation* ; *condition* ; *incrémentation*)
 { *bloc d'instructions à répéter* ; }



```

for ( $i = 1 ; $i <= 6 ; $i++)
{
    echo "<h$i> Titre niveau $i </h$i>";
}
  
```

$\$i++ \rightarrow \$i = \$i + 1$

- Instructions de boucle : for

```
<?php
  for ( $i = 1 ; $i <= 6 ; $i++)
  {
    echo "<h$i> Titre niveau $i </h$i>";
  }
?>
```

Titre niveau 1

Titre niveau 2

Titre niveau 3

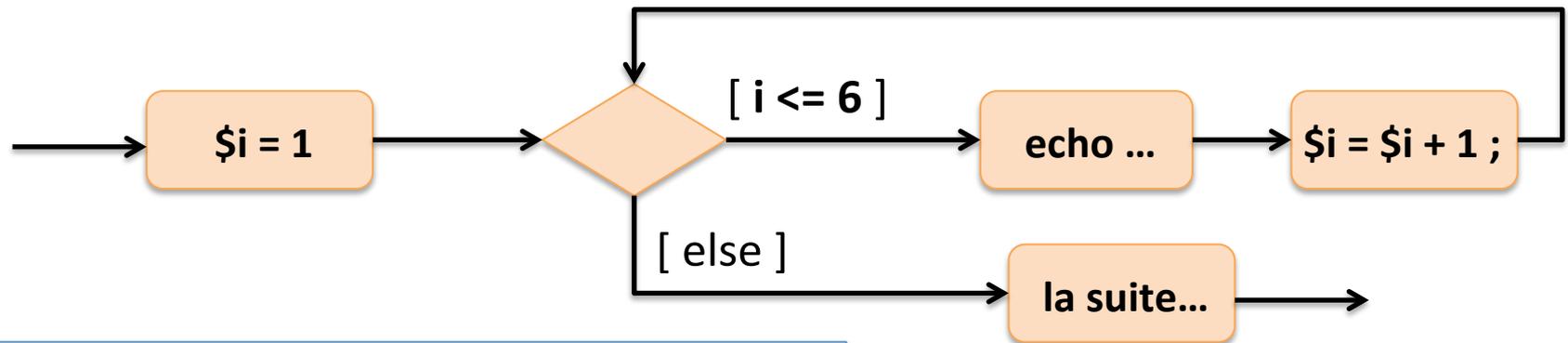
Titre niveau 4

Titre niveau 5

Titre niveau 6

- **Instructions de boucle : while**

- La boucle **while** permet de continuer à réaliser un bloc d'opérations **tant qu'une condition soit vraie**



```

$i = 1 ;
while ( $i <= 6 ) {
    echo "<h$i> Titre niveau $i </h$i>";
    $i = $i + 1;
}
  
```

- Instructions de boucle : while

On donne une **valeur initiale** à la variable **\$i**

```
<?php
```

```
$i = 1 ;
```

```
while ( $i <= 6 ) {
```

```
    echo "<h$i> Titre niveau $i </h$i>";
```

```
    $i = $i + 1;
```

```
}
```

```
?>
```

Tant que **\$i** ne dépasse pas la valeur 6

On met à jour la **valeur** de la variable **\$i**

Titre niveau 1

Titre niveau 2

Titre niveau 3

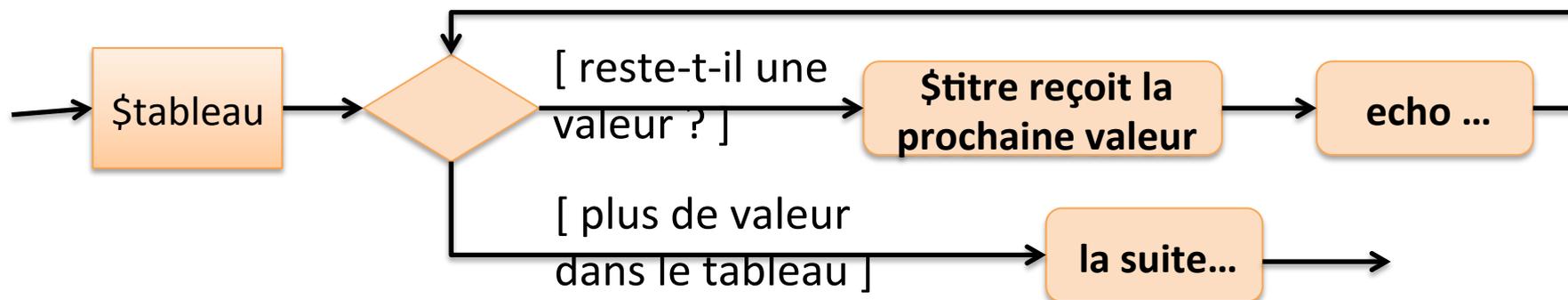
Titre niveau 4

Titre niveau 5

Titre niveau 6

• Instructions de boucle : foreach

- La boucle **foreach** permet de **répéter** un bloc d'instructions pour **chaque valeur dans un tableau**



```

foreach ($tableau as $titre ) {
    echo "<$titre> Titre $titre
    </$titre>";
}
  
```

- Instructions de boucle : foreach

```
<?php
```

On définit un tableau

```
$tableau = array("h1", "h2", "h3",  
                "h4", "h5", "h6");
```

```
foreach ($tableau as $titre ) {
```

```
    echo "<$titre> Titre $titre  
        </$titre>";
```

Pour chaque valeur
dans le tableau

```
}
```

```
?>
```

Titre h1

Titre h2

Titre h3

Titre h4

Titre h5

Titre h6

• Instructions de boucle : foreach

- ça fonctionne aussi pour les tableaux associatifs

```
<?php
    $tableau = array ("nom" => "Dupont" ,
        "prenom" => "Jean" ,
        "adresse" => "qq part à Paris" );

    foreach ($tableau as $cle=>$valeur) {
        echo "<li> $cle : $valeur </li>" ;
    }
?>
```

On définit un tableau
associatif : clé => valeur

Pour chaque pair
\$cle => \$valeur
dans \$tableau

- *nom : Dupont*
- *prenom : Jean*
- *adresse : qq part à Paris*

• Instructions de boucle : boucles imbriquées

- Il est possible d’imbriquer des boucles les unes dans les autres

```
<table>
<?php
for ( $lin = 1 ; $lin <= 9 ; $lin++) {
    echo "<tr> ";
    for ( $col = 1 ; $col <= 9 ; $col++) {
        echo "<td> "
            . ($col * $lin) . "</td>";
    }
    echo "</tr>";
}
?>
</table>
```

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

- **Fonctions**

- PHP offre une large panoplie de fonctions

- Exemple : **isset(\$var)** → TRUE si \$var est connue

- Exemple : **empty(\$var)** → TRUE si \$var est vide (ou vaut 0)

- On peut aussi écrire les nôtres

- (même en dehors des classes)

- **function nomFonction (\$paramètre , ...) { instructions }**

```
function salutation ( $nom ) {  
    echo "<h1>Bienvenue, $nom ! </h1>";  
    echo "<p class=droite>Aujourd'hui, nous sommes le " .date('d / m / Y'). "</p>" ;  
}
```

```
...  
<form name="..." method="POST"  
    action="coursPHP-15.php" >  
<label >Nom : </label>  
<input type="text" name="client"  
    size="25"/>  
...  
<input type="submit" value="OK" />  
</form>
```

```
<?php  
function salutation ( $nom ) {  
    date_default_timezone_set("Europe/Paris");  
    echo "<h1>Bienvenue, $nom ! </h1>";  
    echo "<p class=droite>Aujourd'hui..."  
        . date('d / m / Y'). "</p>" ;  
}  
  
if ( isset ( $_POST["client"] )  AND  
    ! empty ( $_POST["client"] ) ) {  
    salutation ( $_POST["client"] ) ;  
}  
else {  salutation ("cher client") ; }  
?>
```

Identification

Nom :

Mot de passe :

OK

Nom :

Mot de passe :

OK

Bienvenue, manuele !

Aujourd'hui, nous sommes le 04 / 03 / 2012

Bienvenue, cher client !

Aujourd'hui, nous sommes le 04 / 03 / 2012



- **Importation des fichiers**

- Incorporer le contenu d'un fichier dans une page PHP
- But : réutilisation des fichiers, uniformisation du site

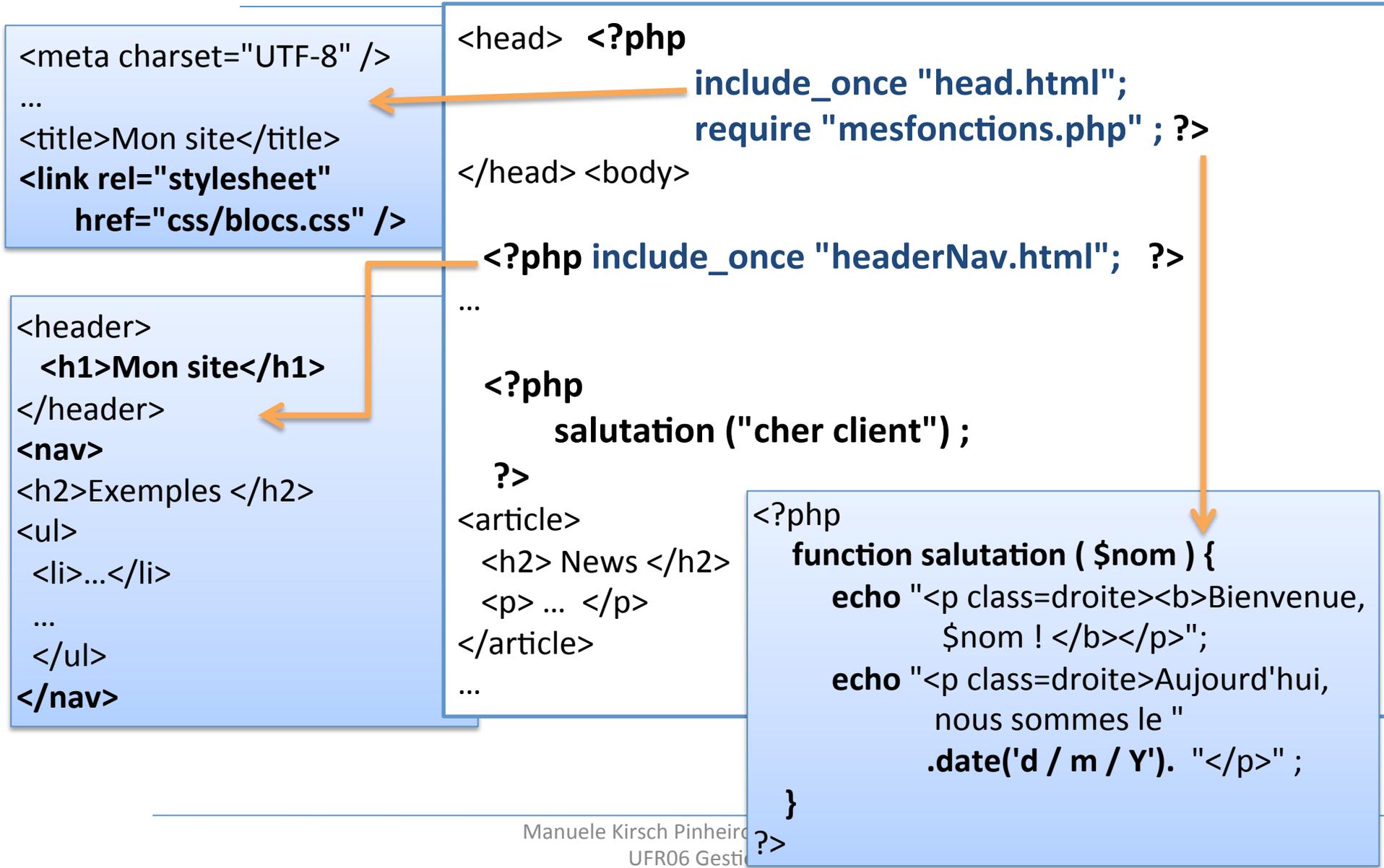
- **include "fichier" et include_once "fichier"**

- **include** remplace la ligne par le contenu du fichier
- **include_once** fait ça **une seule fois** (même dans une boucle)

- **require "fichier" et require_once "fichier"**

- idem **include**, mais si le fichier n'existe pas, on a une **erreur**

PHP



```
include_once "head.html"
```

```
<head> ...  
  <title>Mon site</title>  
  <link rel="stylesheet"  
    href="css/blocs.css" />  
</head>  
<body>  
<header> <h1>Mon site</h1> </header>  
<nav>  
<h2>Exemples </h2>  
<ul>  
  <li>...</li>  
  ...  
</ul>  
</nav>  
<section>  
<p class=droite><b>Bienvenue, cher client ! </b></p><p  
class=droite>Aujourd'hui, nous sommes le 22 / 03 / 2014 </p>  
...
```

```
include_once "headerNav.html";
```

```
require "mesfonctions.php" ;  
salutation ("cher client") ;
```



Informatique

Modélisation UML

Objectifs de la séance :

PHP & les bases de données

- **Accès aux bases de données à partir de PHP**
 - PHP-MySQL sont très utilisés pour les sites Web
 - Différents « bibliothèques » disponibles
 - **mysqli** et PDO
- **Etales pour l'utiliser une base des données**
 - 1) connexion au serveur MySQL
 - 2) envoi des requêtes SQL (select, insert into...)
 - 3) récupération des résultats
 - 4) fermeture de la connexion

- **Connexion à un serveur MySQL à travers mysqli**

- Toute la communication avec la BdD passe par un **objet** de la classe « **mysqli** »
- La **connexion** s'effectue à la **création de cet objet (new)**

```
$idcon = new mysqli ( $host, $user, $mdp, $bdd );
```

*objet identifiant
de la connexion*

nom du serveur

utilisateur autorisé à accéder à la base

mot de passe

base de données

- **Toute connexion ouverte doit être fermée**

```
$bool = $idcon->close ();
```

*on demande à l'objet
mysqli de fermer la
connexion*

PHP

```
<?php
```

```
$host = "localhost";  
$user = "root";  
$mdp = "root";  
$bdd = "clientsBD";
```

Astuce : placer ces informations dans un fichier et faire **require** (ou **include**) "fichier"

```
$mysqli = new mysqli ($host, $user, $mdp, $bdd );
```

Création de l'objet connexion

```
if ( $mysqli->connect_errno ) {  
    die ("<p> Impossible de connecter à $bdd : "  
        . $mysqli->connect_error . " </p>" );  
}
```

L'attribut **connect_errno** de indique si la connexion a bien été établie

```
else {  
    echo "<p> Connecté au serveur $host,  
        à la base $bdd </p>";  
    $mysqli->close();  
}
```

En cas de problème, on arrête avec la fonction **die**.

Fermeture de la connexion

```
?>
```

- **Envoie de requêtes à une base de données**

```
$result = $mysqli->query ($sql) ;
```

Résultat de la requête *exécution de la requête sur l'objet connexion* *Requête SQL à exécuter*

- Requête SQL :

- S'il s'agit d'un **SELECT**, le **résultat** correspond aux **données** fournies par la requête (objet **mysqli_result**)
- Sinon (**INSERT, UPDATE, DELETE...**), le résultat sera **TRUE** si la requête est **bien exécutée**, **FALSE** sinon

PHP

```

<form name="formNouveauClient"
  action="coursPHP-18.php"
  method="POST">

  ...
  <input type="text" name="nom" ... />
  ...
  <input type="text" name="email" ... />
  ...
  <input type="submit" value="Envoyer" />
</form>
    
```

Formulaire

Nouveau client

Nom

Email

Adresse

id	nom	email	adresse
1	Manuele Kirsch Pinheiro	mkirschpin@univ-paris1.fr	90 rue Tolbiac Paris

Sélection : Modifier Effacer Exporter

Nombre de lignes: En-têtes à chaque

Connexion BdD

Insertion de données

connecté !

Vous êtes le client numéro 1

```
<?php
if ( ! empty($_POST["nom"]) AND ! empty($_POST["email"]) ) {
    require "connexion.php";
    $mysqli = connexion() ;

    $nom = $_POST["nom"];
    $email = $_POST["email"];
    $adr = $_POST["adresse"];
    $id = '\N'; /* auto-increment */

    $sql = "INSERT INTO client (id, nom, email, adresse)
           VALUES ( '$id', '$nom', '$email', '$adr' ) ";

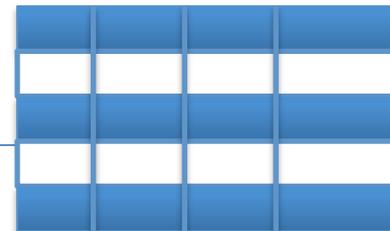
    $result = $mysqli->query ($sql) ;

    if ( ! $result ) { echo "<p>Désolée, ... </p>"; }
    else {
        echo "<p> Vous êtes le client numéro <i> "
            . $mysqli->insert_id . "</i></p>";
    }
    $mysqli->close() ;
}
... ?>
```

```
<?php
function connexion() {
    $host = "localhost";
    $user = "uml";
    $mdp = "uml";
    $bdd = "clientsBD";

    $mysqli = new mysqli ($host,
                          $user, $mdp, $bdd) ;

    if ( $mysqli->connect_errno ) {
        die ("<p> Impossible ..."
            . $mysqli->connect_error . " </p>" );
    }
    return $mysqli ;
} ?>
```



• Récupération des données

```
$result = $mysqli->query ("SELECT * FROM table");
```

- Les requêtes SELECT fournissent des données
- On récupère le résultat (ligne à ligne) à l'aide des opérations **fetch_***
- Chaque appel à **fetch_*** retourne la **prochaine ligne**
 - Ligne dans un tableau à indice : `$result->fetch_row ()` ;
 - Ligne dans un tableau associatif : `$result->fetch_assoc ()` ;
 - **Ligne dans un objet** : `$result->fetch_object ()` ;

```
<?php
require "connexion.php" ;
$mysqli = connexion();

$sql = "SELECT id, nom, email, adresse
        FROM client ORDER BY nom " ;
$result = $mysqli->query ($sql) ;

if ( ! $result ) { echo "<p> Desolée ... </p>" ; }
else { ...
    while ( $ligne = $result->fetch_object() ) {
        ...
        echo "<td>" . $ligne->id . "</td>";
        echo "<td>" . $ligne->nom . "</td>";
        echo "<td>" . $ligne->email . "</td>";
        echo "<td>" . $ligne->adresse . "</td>";
        ...
    } ...
} ?>
```

```
<?php
function connexion() {
    ...
    $mysqli = new mysqli ($host,
                          $user, $mdp, $bdd) ;
    ...
    return $mysqli ;
} ?>
```

On exécute la requête avec l'opération **\$mysqli->query**

L'opération **\$result->fetch_object** récupère la prochaine ligne, FAUX s'il n'y reste plus de lignes.

Chaque **attribut de la requête** devient un **attribut de l'objet \$ligne**

Clients			
id	Nom	Email	Adresse
1	Manuele Kirsch Pinheiro	mkirschpin@univ-paris1.fr	90 rue Tolbiac, 75013 Paris

La même requête avec récupération des informations

- ... par tableau à indice

```
...
$sql = "SELECT id, nom, email, adresse
        FROM client ORDER BY nom ";
$result = $mysqli->query ($sql);
...
while ( $ligne = $result->fetch_row() ) {
    ...
    echo "<td>" . $ligne[0] . "</td>";
    echo "<td>" . $ligne[1] . "</td>";
    echo "<td>" . $ligne[2] . "</td>";
    echo "<td>" . $ligne[3] . "</td>";
    ...
}...
```

ça commence toujours par **0**

- ... par tableau associatif

```
...
$sql = "SELECT id, nom, email, adresse
        FROM client ORDER BY nom ";
$result = $mysqli->query ($sql);
...
while ( $ligne = $result->fetch_assoc () ) {
    ...
    echo "<td>" . $ligne['id'] . "</td>";
    echo "<td>" . $ligne['nom'] . "</td>";
    echo "<td>" . $ligne['email'] . "</td>";
    echo "<td>" . $ligne['adresse'] . "<td>";
    ...
}...
```

chaque attribut est accessible
par son **nom**

- Autres informations peuvent être récupérées d'un objet **mysqli_result** (`$result = $mysqli->query (...)`)
 - **Combien de lignes et colonnes** on peut récupérer
 - `$nblignes = $result->num_rows ;`
 - `$nbcol = $result->field_count ;`
 - **Les noms des colonnes (attributs)** dans le résultat
 - `$colonnes = $result->fetch_fields() ;`

PHP

```
...  
$sql = "SELECT id, nom, email, adresse  
        FROM client ORDER BY nom " ;  
$result = $mysqli->query ($sql) ;  
  
...  
echo "<p> Nous avons " . $result->num_rows . " clients. </p>";  
echo "<p> Il y a " . $result->field_count . " attributs par client. </p> " ;  
...
```

A partir de l'objet **\$result**, on peut récupérer le nombre de lignes (attribut **num_rows**) et de colonnes par ligne (attribut **field_count**).

On peut aussi récupérer les **colonnes**. Chaque colonne est un **objet** et l'attribut **name** donne son nom.

La ligne aussi est un objet dont les **attributs** correspondent aux **colonnes**. On peut utiliser un **foreach** pour accéder à la **valeur des attributs**.

```
...  
$titres = $result->fetch_fields() ;  
foreach ($titres as $colonne) {  
    echo "<th> " . $colonne->name . " </th>" ;  
}  
while ( $ligne = $result->fetch_object() ) {  
    echo "<tr>" ;  
    foreach ( $ligne as $colonne=>$val ) {  
        echo "<td> " . $val . " </td>" ;  
    }  
    echo "</tr>" ;  
}...
```

Informatique

Modélisation UML

Objectifs de la séance :
Mécanismes de sessions

- **Mécanisme de sessions**
 - Chaque visite à un site / page est indépendante
 - Les **sessions** permettent de conserver les informations des visiteurs **entre les pages**
 - Les informations sur les sessions sont stockées sur le **serveur**
- **Fonctionnement général**
 - 1) Ouverture de session : **session_start ()**
 - Chaque utilisateur reçoit un identifiant transmis entre les pages
 - 2) Définition des variables de sessions (données)
 - Les variables de session sont transmises de page à page
 - **\$_SESSION["variable"] = valeur ;**
 - 3) Fermeture de session : **session_destroy()**

PHP

localhost:8888/exem

Identification

Login :

Mot de passe :

Login & mdp
différents de uml

localhost:8888/exemp

Desolé !

Page accessible uniquement aux membres.

Login & mdp corrects
(uml /uml)

localhost:8888/exemplesPHP/coursPHP-23.php

- [Accueil membre](#)
- [Deconnexion](#)

Bienvenue, cher uml

```
<form name="..."
  action="coursPHP-23.php"
  method="POST">
  <label >Login : </label>
  <input type="text" name="login" maxlength="15" /> <br/>
  <label >Mot de passe : </label>
  <input type="password" name="mdp" maxlength="15" />
  <br/>
  <input type="submit" value="OK" />
</form>
```

PHP

```
<?php session_start(); ?>
```

```
<html>
```

```
<head> ... </head>
```

```
<body>
```

```
<?php
```

```
...
```

```
$login = $_POST["login"] ;
```

```
$mdp = $_POST["mdp"];
```

```
if ( $login == "uml" AND $mdp == "uml" ) {
```

```
    $_SESSION["login"] = $login ;
```

```
    ...
```

```
    echo "<h1>Bienvenue, cher $login </h1>" ;
```

```
}
```

```
else { echo "<h1>Desolé ! </h1>";
```

```
    echo "<p> Page accessible uniquement aux membres. </p>";
```

```
}
```

```
?>
```

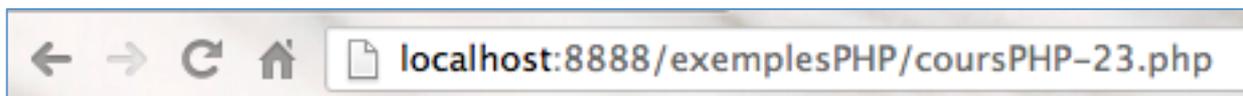
```
</body> </html>
```

Ouverture d'une session
(au début de chaque page)

Définition des variables de session
\$_SESSION["var"]

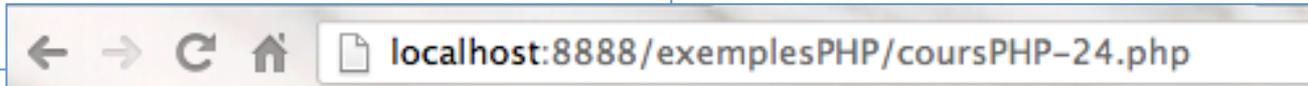
Les variables de session contiennent les informations qui passeront de page en page.

PHP



- [Accueil membre](#)
- [Deconnexion](#)

Bienvenue, cher uml



- [Accueil membre](#)
- [Deconnexion](#)

Mon site

Client **uml** : Ceci est une page pour les abonnés

```
<?php session_start(); ?>
```

```
<html> <head>... </head>
```

```
<body>
```

```
<?php
```

```
if ( isset( $_SESSION["login"] ) AND ! empty( $_SESSION["login"] ) ) {
    $login = $_SESSION["login"] ;
```

```
...
```

```
echo "<p>Client <b> $login </b> : Ceci est un
```

```
}
```

```
else { echo "<h1>Desolé ! </h1>";
```

```
echo "<p> Il s'agit d'une page privée !! Il faut être membre. </p>";
```

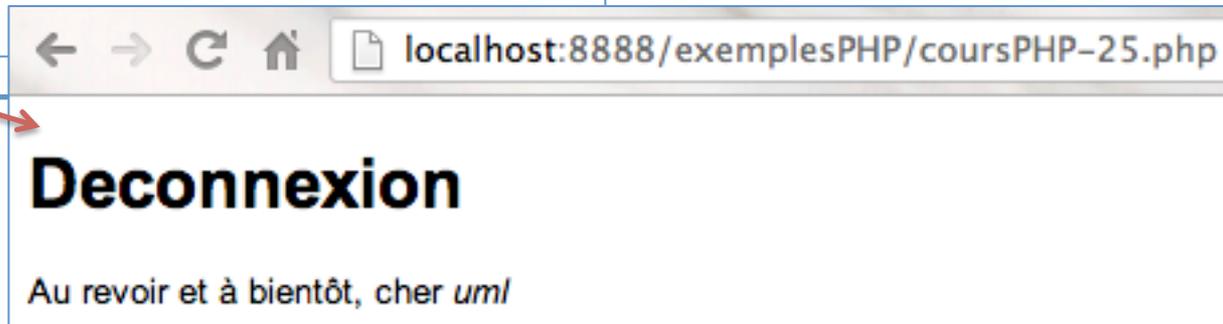
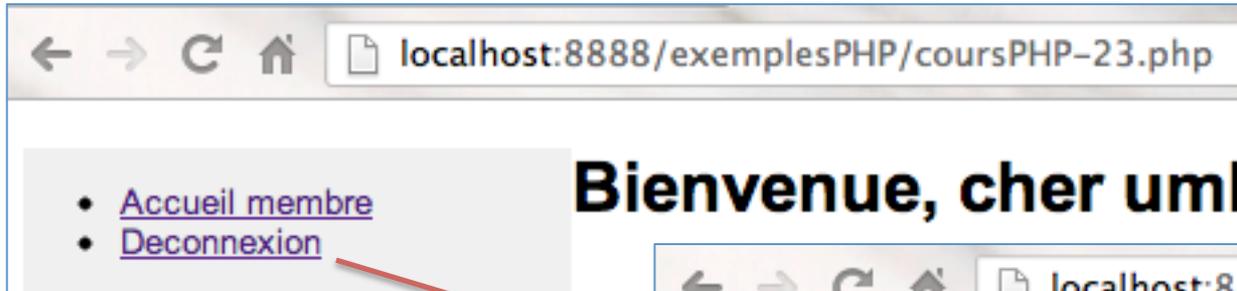
```
}
```

```
?>
```

```
...
```

Usage des variables de session

`$_SESSION["var"]`



```
<?php session_start(); ?>
<html> <head>... </head>
<body>
<?php
    if ( isset( $_SESSION["login"] ) AND ! empty( $_SESSION["login"] ) ) {
...
        unset($_SESSION["login"]);
        session_destroy();
    }
    else {    echo "<h1>Desolé ! </h1>";
            echo "<p> Pas de connexion active. </p>";
    }
?>
...
```

Fermeture de la session
`session_destroy()`

Ne pas oublier de vider les variables de session
`unset($_SESSION["var"])`

- **Mécanisme de sessions**

- Base pour la gestion de **panier** dans les sites de e-commerce
- Les produits choisis par le client sont enregistrés en tant que **variables de session**
- On peut y garder des **objets SIMPLES**

```
class LigneProduit {  
    public $nom ;  
    public $qte ;  
  
    /* constructeur */  
    function __construct( $nom ) {  
        $this->nom = $nom;  
        $this->qte = 1;  
    }  
}
```

Contenu du panier est gardé dans les variables de session.
Tableau contenant des objets LigneProduit.
Chaque **\$_SESSION[\$produit]** contient un objet.

PHP

```
function ajouterProduit($produit) {
```

```
    $qte = 0;
```

```
    if ( ! isset ( $_SESSION[$produit] ) ) {
```

```
        $_SESSION[$produit] = new LigneProduit($produit);
```

```
        $qte = $_SESSION[$produit]->qte
```

```
    }
```

```
    else { // produit déjà là, augmenter alors sa quantité
```

```
        $objet = $_SESSION[$produit] ;
```

```
        $objet->qte = $objet->qte + 1;
```

```
        $qte = $objet->qte ;
```

```
    }
```

```
    return $qte;
```

```
}
```

Chaque produit choisi est identifié par un « id » (ici le nom).

\$_SESSION[\$produit]
va contenir un objet **LigneProduit**

S'il n'y a aucun
\$_SESSION[\$produit] ,
on va créer un nouveau objet
LigneProduit

S'il y a déjà un
\$_SESSION[\$produit] ,
on va juste augmenter la valeur de
l'attribut « qte » dans l'objet
LigneProduit

PHP

```
function supprimerProduit($produit) {
```

```
    $qte = 0 ;
```

```
    if ( isset( $_SESSION[$produit] ) ) {
```

```
        $objet = $_SESSION[$produit] ;
```

```
        $objet->qte = $objet->qte - 1;
```

```
        $qte = $objet->qte;
```

```
        if ( $qte <= 0 ) { //on supprime le produit
```

```
            unset($_SESSION[$produit]);
```

```
        }
```

```
    }
```

```
    return $qte;
```

```
}
```

Lorsqu'on veut supprimer un produit, on va réduire sa quantité dans l'objet **LigneProduit**

On récupère l'objet **LigneProduit** gardé dans **\$_SESSION[\$produit]**

On réduit sa quantité d'une unité

S'il n'en reste plus (la **quantité** a atteint **0 unités**), on **supprime** le produit de la session

On peut récupérer le contenu du panier en récupérant le contenu de la variable de session **\$_SESSION**

Pour chaque **objet LigneProduit** gardé dans **\$_SESSION**

```
function afficherPanier() {  
    echo "<table>";  
    foreach($_SESSION as $objet) {  
        echo "<tr><td> " . $objet->nom . " </td> <td> "  
            . $objet->qte . " </td> </tr> ";  
    }  
    echo "</table>";  
}
```

Nos Produits

[Terminer](#)

[Voir panier](#)

Produit	Prix		
Autocollant Autocollant au symbole du club	15.00 €	<input data-bbox="1213 432 1257 465" type="button" value="+"/>	<input data-bbox="1290 432 1335 465" type="button" value="-"/>
T-shirt Official T-shirt officiel du club	35.00 €	<input data-bbox="1213 515 1257 548" type="button" value="+"/>	<input data-bbox="1290 515 1335 548" type="button" value="-"/>
T-shirt baby-look T-shirt au symbole du club	25.00 €	<input data-bbox="1213 598 1257 631" type="button" value="+"/>	<input data-bbox="1290 598 1335 631" type="button" value="-"/>

Exemple de Panier

[Produits](#)

Ajouter T-shirt Official au panier : quantite 1

[Terminer](#)

T-shirt Official	1
------------------	---

[Voir panier](#)

Exemple de Panier

[Produits](#)

Ajouter autocollant au panier : quantite 1

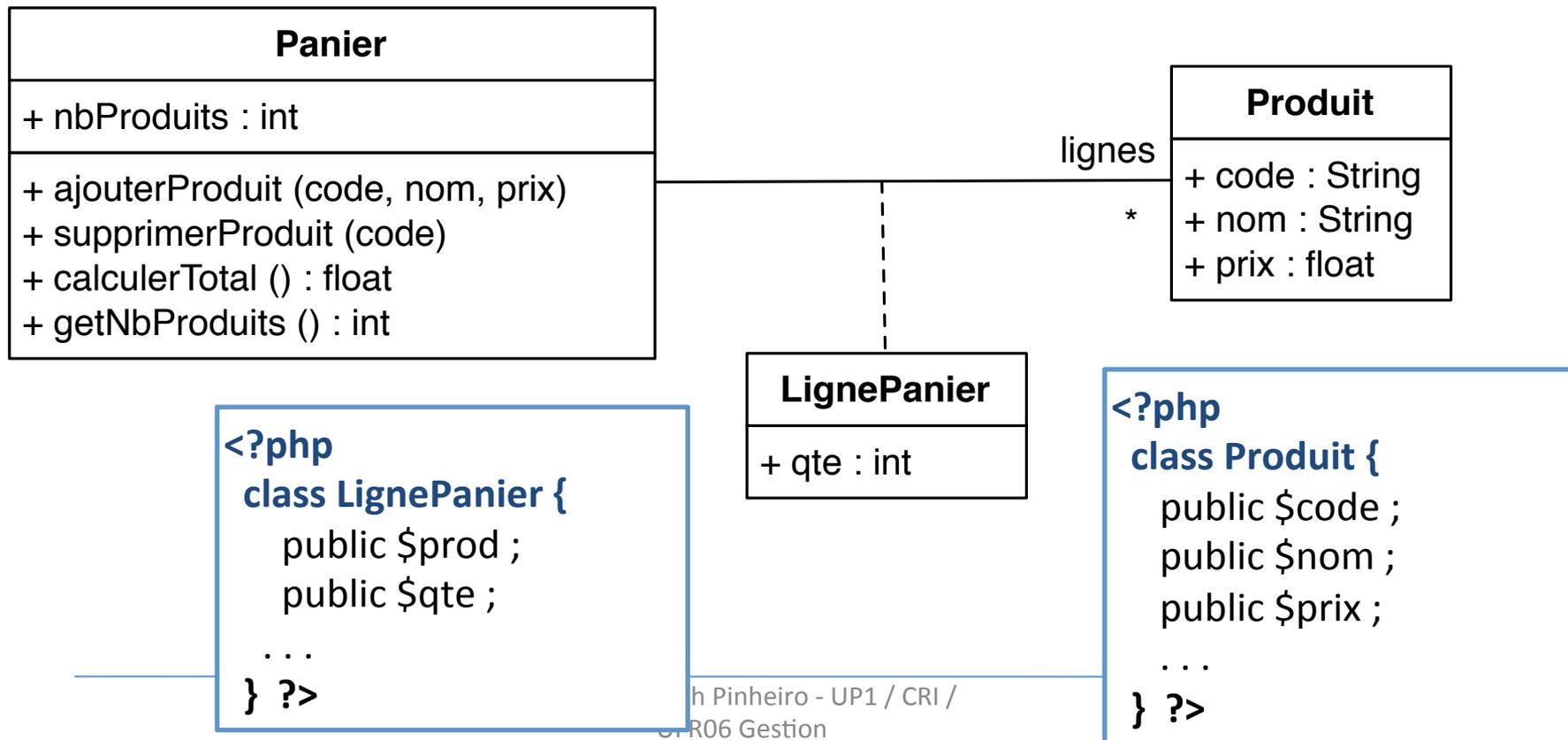
[Terminer](#)

T-shirt Official	1
autocollant	1

[Voir panier](#)

PHP : Panier avancé

- Voici un exemple avancé de Panier qui utilise les classes en PHP et la notion de session
- Le panier est modélisé par une **classe Panier**



PHP : Panier avancé

```
class Panier {  
    public $lignes ;  
    public $nbProduits ;
```

Chaque Ligne de Panier est gardée dans un tableau associatif
`$this->lignes[$code] => $LignePanier`

```
function __construct() {  
    $this->nbProduits = 0 ;  
}
```

On commence avec zéro produits dans le panier

```
...  
function ajouterProduit($code, $nom, $prix) {  
    if ( $this->nbProduits == 0 ) {  
        $prod = new Produit($code, $nom, $prix);  
        $lp = new LignePanier($prod);  
        $this->lignes[$code] = $lp;  
        $this->nbProduits = 1;  
    }  
    ...  
}
```

On va créer le tableau lors de l'ajout du premier produit au panier

PHP : Panier avancé

```
function ajouterProduit($code, $nom, $prix) {  
    if ( $this->nbProduits == 0 ) { . . . }  
    else {  
        if ( isset ( $this->lignes[$code] ) ) {  
            $lp = $this->lignes[$code] ;  
            $qte = $lp->qte;  
            $lp->qte = $qte + 1;  
        }  
        else {  
            $prod = new Produit($code, $nom, $prix);  
            $lp = new LignePanier($prod);  
  
            $this->lignes[$code] = $lp;  
            $this->nbProduits = $this->nbProduits + 1;  
        }  
    }  
}
```

Pour ajouter, on vérifie si le produit est déjà dans le panier

S'il y est, on le **récupère** et on met à jours la **quantité**

S'il n'y est pas, on va y **ajouter** une **nouvelle** ligne de panier

PHP : Panier avancé

```
function ajouterProduit($code, $nom, $prix) {
```

```
    if ( isset ( $this->lignes[$code] ) ) {
```

```
        $lp = $this->lignes[$code] ;  
        $lp->qte = $lp->qte - 1 ;
```

```
        if ( $lp->qte < 1) {  
            unset($this->lignes[$code]);  
            $this->nbProduits = $this->nbProduits - 1;
```

```
        }
```

```
    }
```

```
}
```

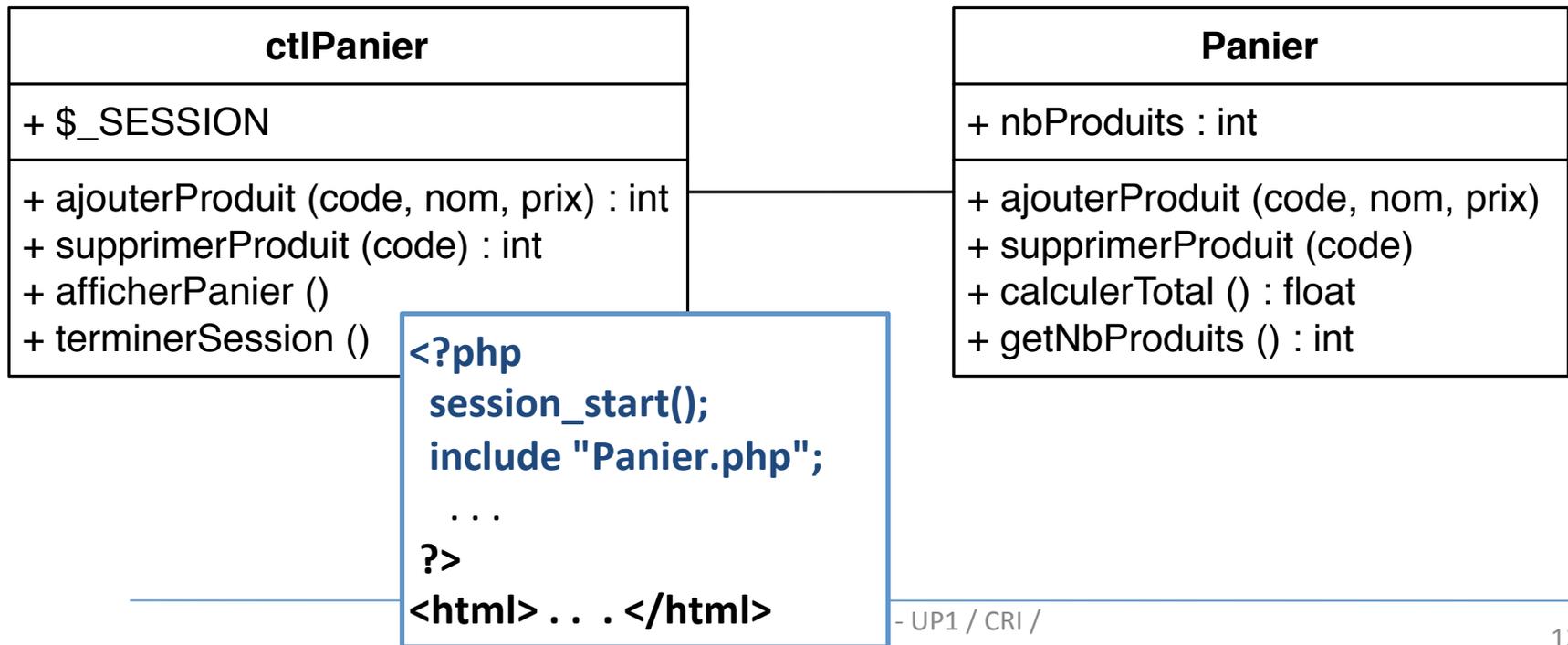
On ne supprime que si le produit est dans le panier

S'il y est, on met à jour la quantité, en **supprimant 1 unité**

Par contre, s'il ne reste plus rien (qte < 1), on **supprime la ligne de panier** du tableau

PHP : Panier avancé

- C'est un objet **Panier** que notre site va manipuler
- Une page « **ctlPanier.php** » va ainsi gérer le panier
- Pour cela, elle va devoir garder un objet **Panier** dans **\$_SESSION**



PHP : Panier avancé

- Or un objet **Panier** est un objet **complexe**
- Pour le garder dans **\$_SESSION**, il va falloir le « compacter » : c'est la **sérialisation**
 - `$_SESSION["panier"] = serialize($panier)`
 - `unserialize ($_SESSION["panier"])`

```
function ajouterProduit($produit, $nom, $prix) {  
...  
    $panier = unserialize($_SESSION["panier"]);  
    $panier->ajouterProduit($produit, $nom, $prix);  
    $_SESSION["panier"] = serialize($panier);  
...  
}
```

Pour ajouter ou supprimer un produit au panier, on va le **recupérer**, le **modifier** puis le **remettre** dans la session

```
function supprimerProduit($produit) {  
...  
    $panier = unserialize($_SESSION["panier"]);  
    $panier->supprimerProduit($produit);  
    $_SESSION["panier"] = serialize($panier);  
...  
}
```