UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Cours de technologies Web
Introduction

Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

Présentation de l'UE

- **Objectif** : initiation aux technologies web et à la conception de sites web
  Langages informatique : HTML, CSS, PHP, et Javascript

- **Volume horaire :**
  - 24h cours + TD (12 séances de 2h)
  - 16h TP (8 séances de 2h)

- **Evaluation** : contrôle continu
  - conception de votre page web personnelle (mise en pratique HTML et CSS) : 1/3 de la note
    - 3 versions: HTML5/CSS3 avec 2 feuilles de style + Bootstrap
    - à préparer en parallèle des séances de cours
  - projet complet en binôme : 2/3 de la note
    - rapport technique : cahier des charges + étude préliminaire (fonctionnalités, architecture, moyens techniques utilisés, etc) → 1/3 de la note
    - développement (code à rendre) + séance de démonstration → 1/3 de la note

Références Bibliographiques

- http://fr.selfhtml.org/navigation/html.htm
- http://cyberzoide.developpez.com/html/

- http://www.w3.org/Style/CSS/
- http://www.alsacreations.com/

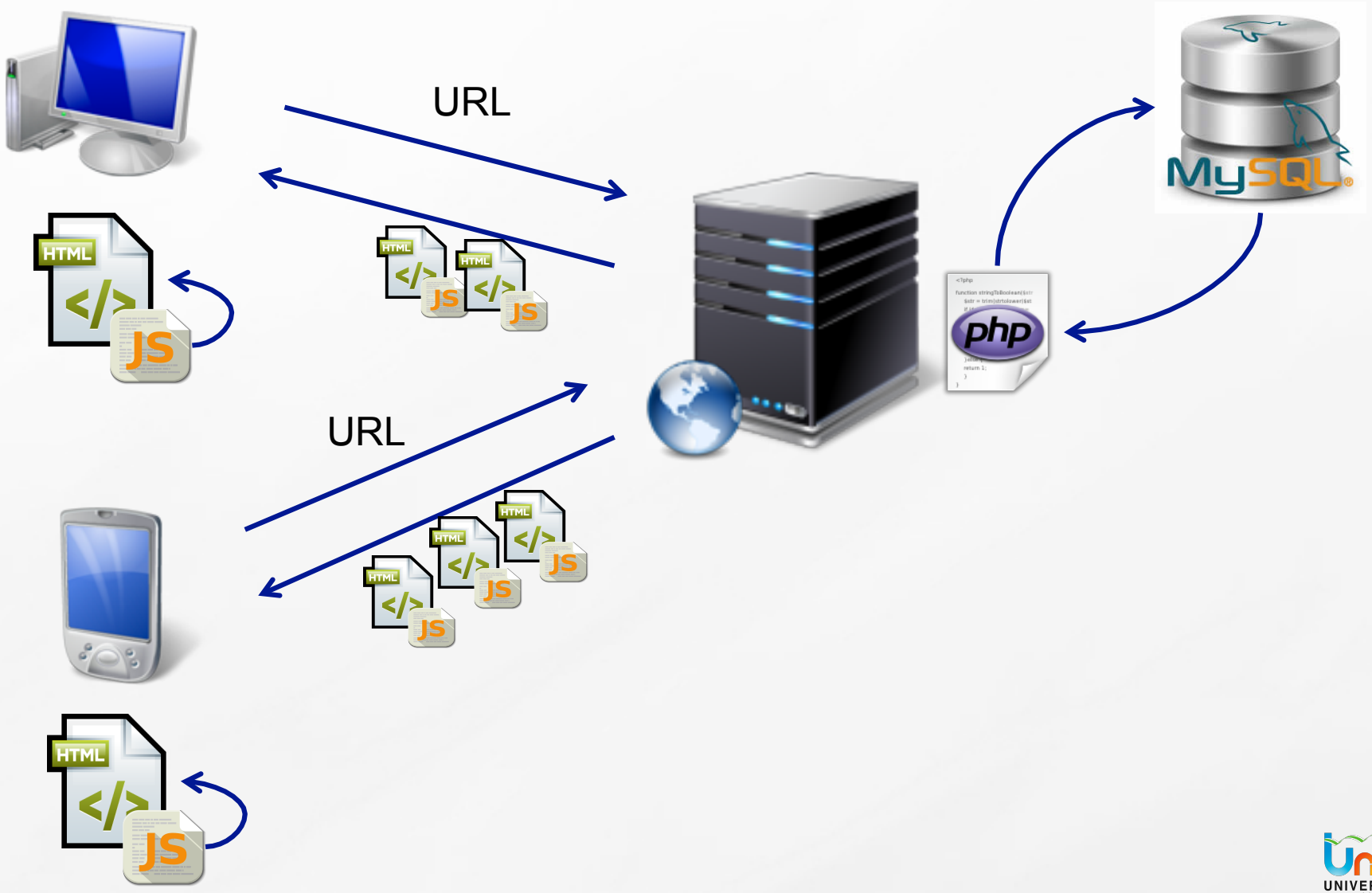
- http://www.php.net/manual/fr/
- http://www.phpfrance.com
- http://www.developpez.com/php/

- cours de Vincent Vanneste (Université du littoral), http://cours.armali.fr/public/javascript/index.html
- cours de Gilles Chagnon (Université Pierre et Marie Curie), http://www.gchagnon.fr/cours/dhtml
- cours d'Emmanuel Coquery (Université Lyon 1)
- cours d'Olivier Serre (Polytechnique)

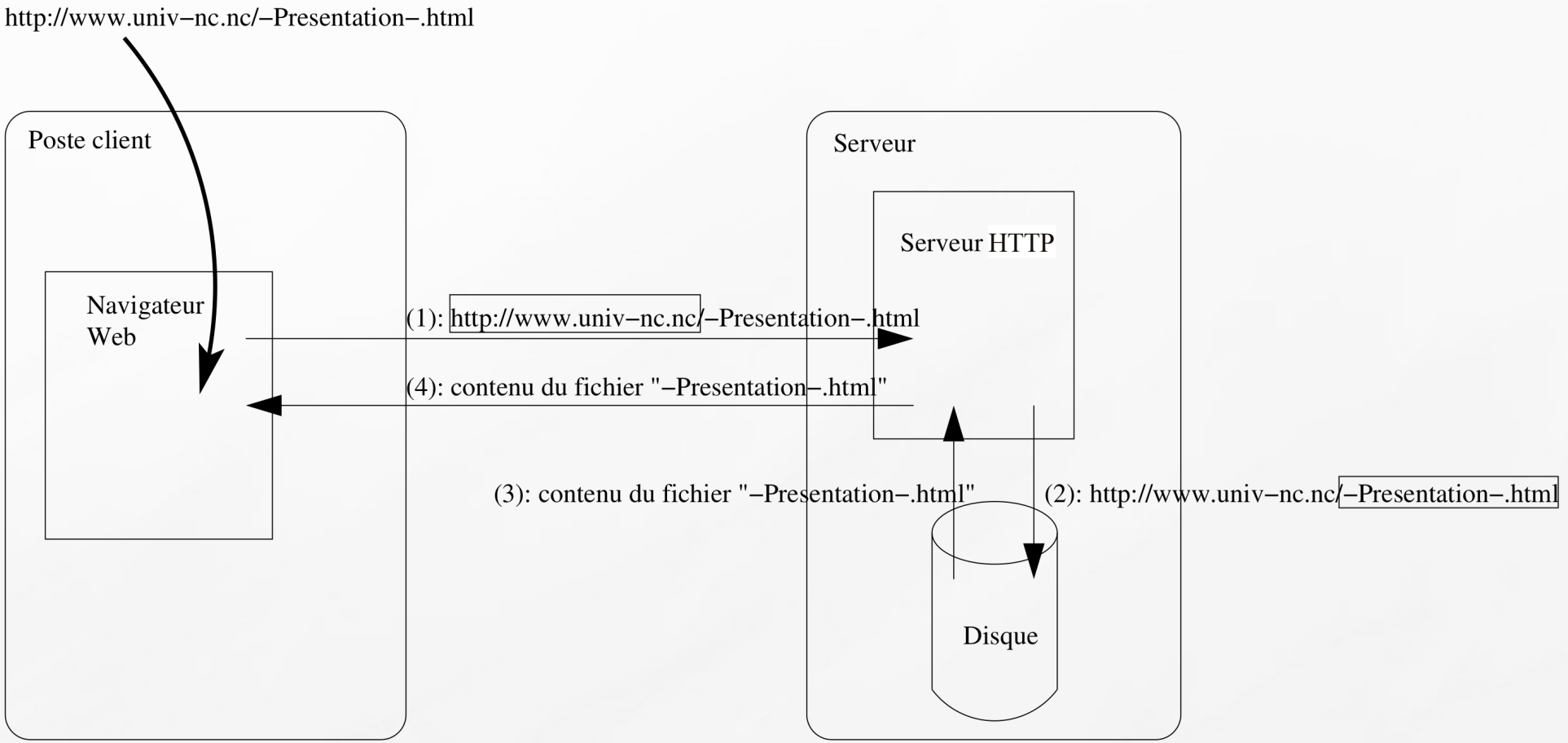
Les langages

- Un gros nombre de langages de programmation
  - HTML, CSS, JavaScript, PHP, PERL, CGI, Applet, Flash, …
- Les plus utilisés (et ceux étudiés dans le cadre du cours)
  - **HTML** (côté client)
    - contenu des pages web (textes, images, …)
  - **CSS** (côté client)
    - mise en forme des pages web
  - **JavaScript** (côté client)
    - ajouter des comportements dynamiques (p.ex. vérification de champs de saisie, ouverture de pop-up, etc) en réponse aux actions des utilisateurs
  - **PHP** (côté serveur)
    - générer dynamiquement des pages web (HTML) en fonction des interactions avec les utilisateurs
    - accéder aux bases de données
  - **SQL** (côté serveur)
    - interroger les bases de données

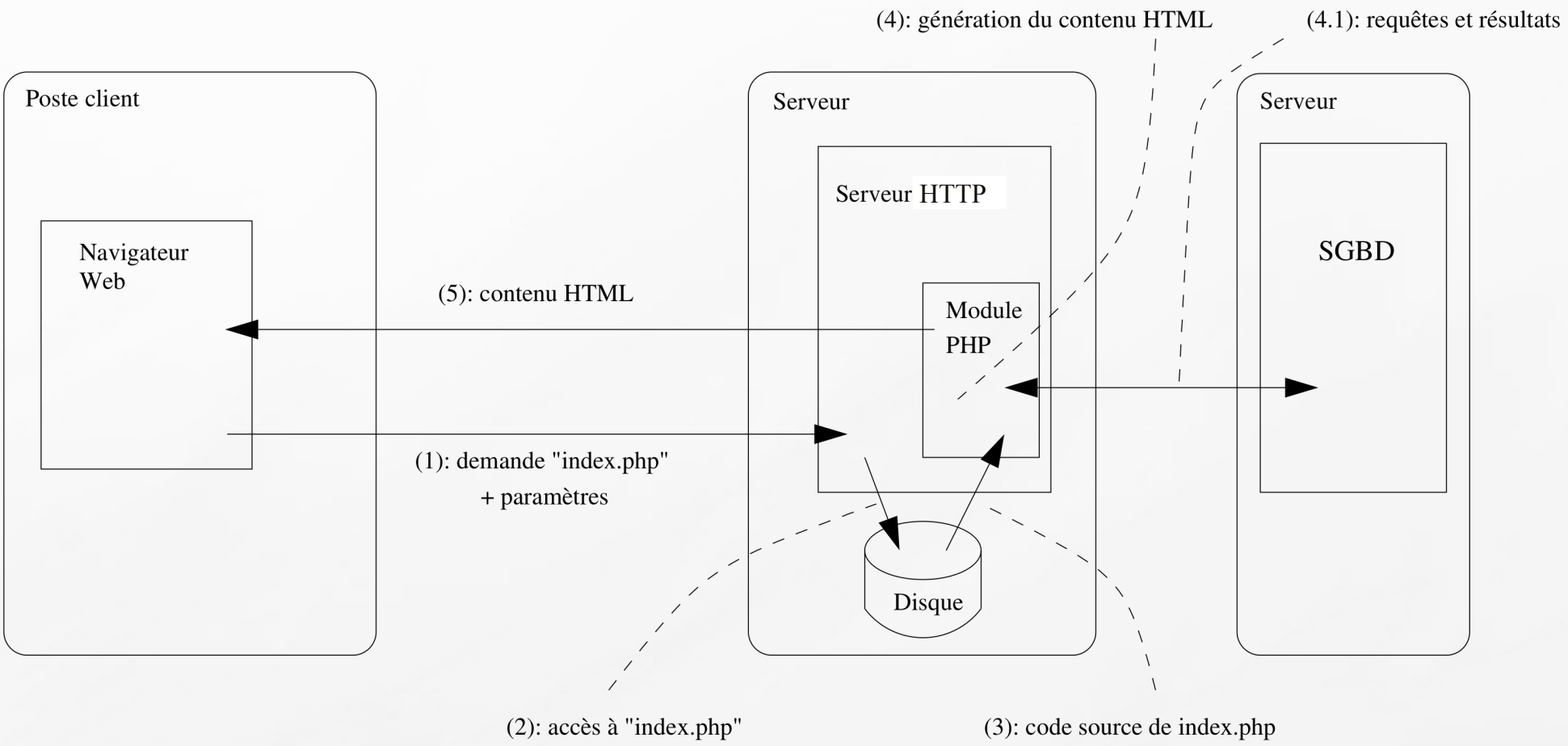
Principe de fonctionnement



Principe de fonctionnement: page web statique



Principe de fonctionnement: pages web dynamiques



Les outils pour éditer

- Les éditeurs WYSIWYG (What You See Is What You Get)
  - logiciels facilitant la création et la modification de pages HTML telle qu'elles apparaissent dans le navigateur Web
    - p.ex. Dreamweaver, BlueGriffon, KompoZer, Mozilla Composer
  - *avantages*: facilité d'utilisation et observation immédiate du rendu graphique
  - *inconvénients*: manque de maîtrise, risque d'incompatibilité avec des navigateurs, dédié en général aux pages web statiques
- Les éditeurs de texte
  - éditeur de texte classique avec coloration syntaxique en fonction du langage
    - p.ex. **Aptana Studio 3**, Notepad++, Quanta Plus, Bluefish
  - *avantage*: maîtrise des documents produits
  - *inconvénient*: nécessité d'avoir des connaissances en programmation Web

Les outils pour "compiler"

- HTML/CSS
  - pas vraiment de compilateur ➔ compilation ET exécution faite par le navigateur web
  - ➢ mais des outils existent pour vérifier la validité du code
    - ➢ **W3C Validator**: vérifie la conformité aux standard du W3C (organisme international de standardisation des technologies Web)
    - ➢ **W3C CSS Validation Service**: vérifie la conformité du feuille de style CSS avec les recommandations du W3C
  - ➢ possibilité d'utiliser des plugins du navigateur pour "inspecter" les éléments affichés
- PHP
  - interprétation et exécution du code faite par un moteur PHP intégré au serveur web
    - ex. de serveurs web: Apache, IIS
  - erreurs PHP insérées dans le code HTML et affichées par le navigateur
- JavaScript
  - possibilité d'utiliser des plugins du navigateur pour débugger

Développer en local puis mettre en ligne

- **Développer tout d'abord en local**
  - pages HTML, CSS, JavaScript
    - créer les fichiers sources dans un répertoire
    - puis les exécuter avec le navigateur
  - pages PHP
    - nécessité d'avoir un serveur web d'installé avec un module PHP
    - utiliser des environnements de développement tel que *Xampp* (serveur Apache, MySQL, …)

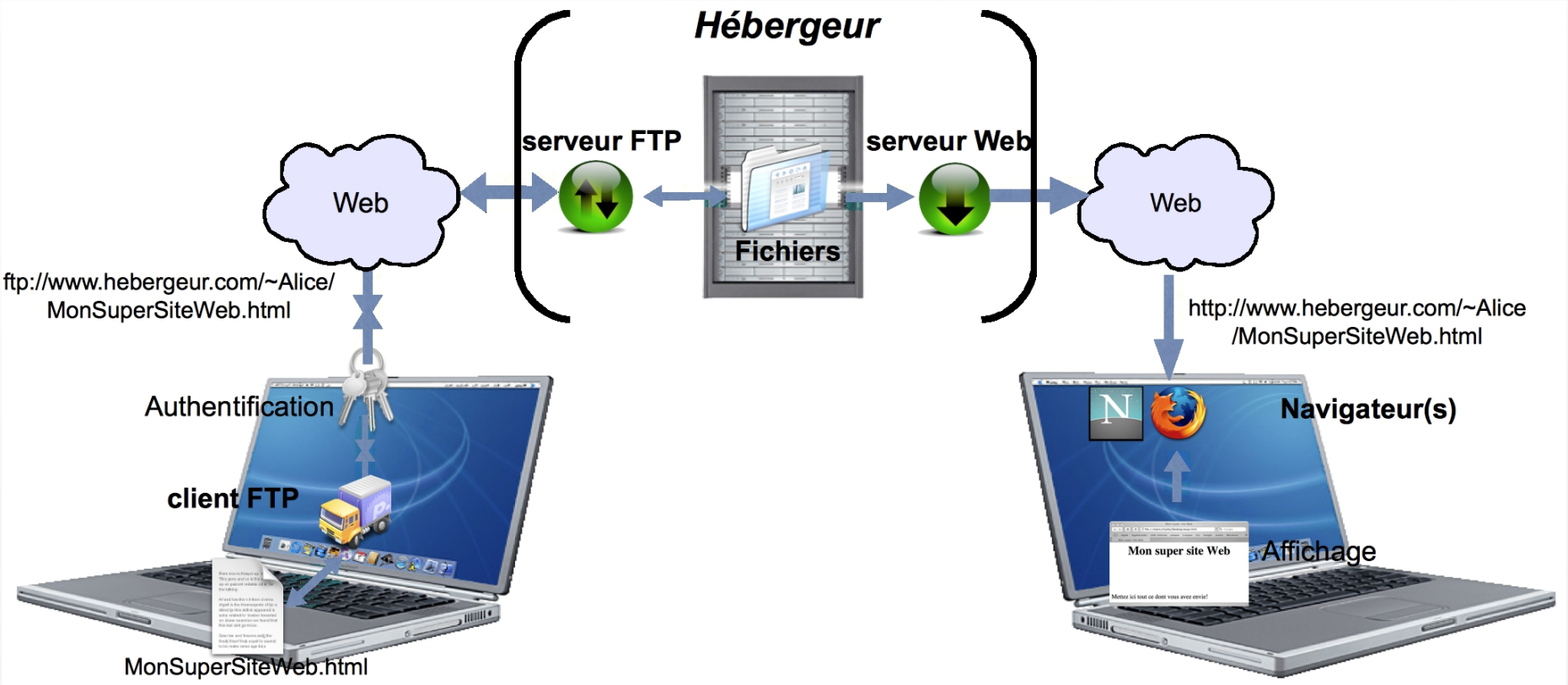
      création d'un nom de domaine virtuel par le serveur web: http://localhost ou http://127.0.0.1

      racine du serveur web = répertoire */htdocs* dans le répertoire où est installé Xampp

      juste copier les pages web dans un sous répertoire de */htdocs* et accéder à l'adresse *localhost/mon_repertoire/mon_fichier.php* avec le navigateur web

Développer en local puis mettre en ligne

- **Puis mettre en ligne**



UNIVERSITÉ de la NOUVELLE-CALÉDONIE

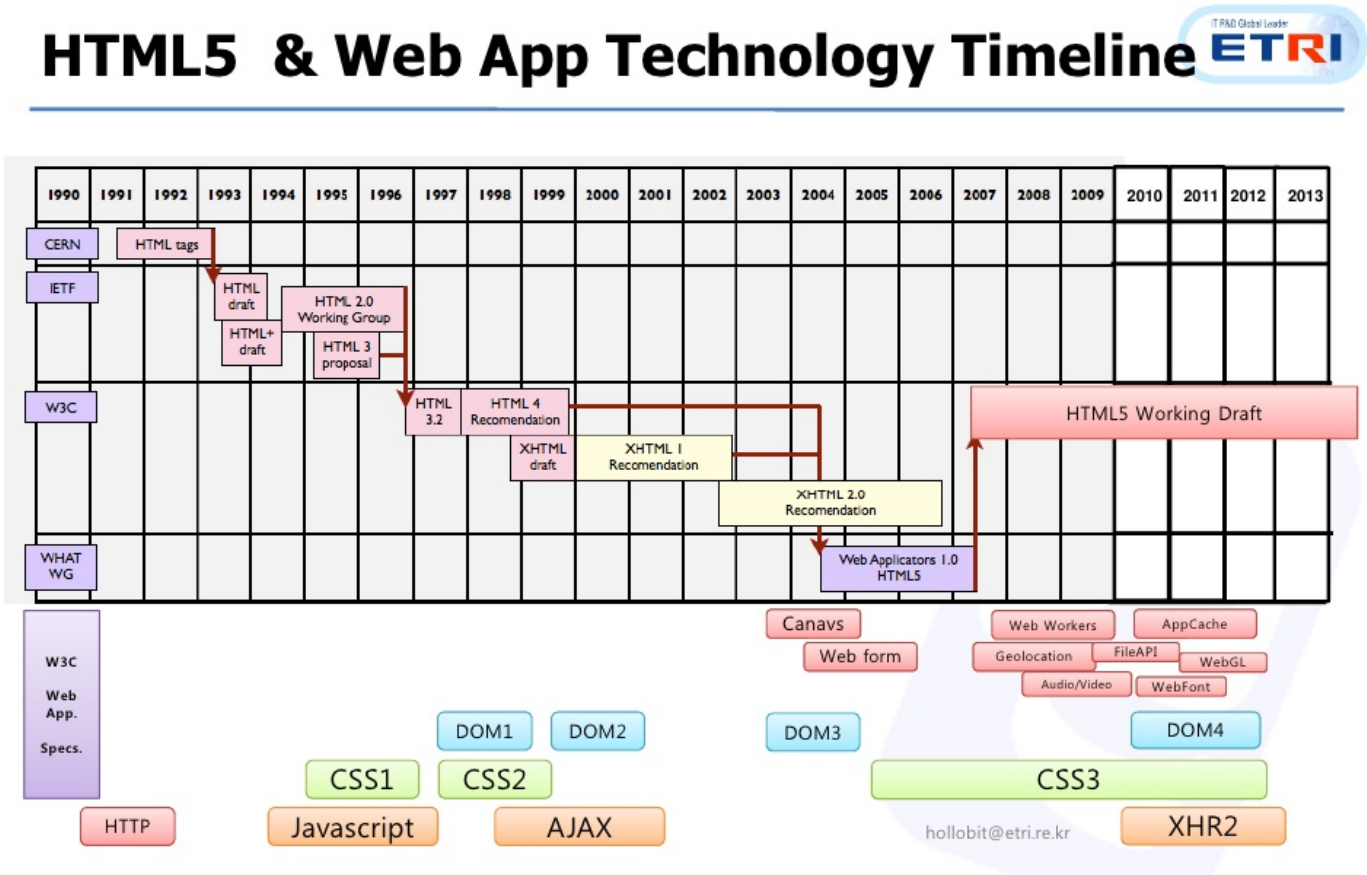
Cours de technologies Web
Introduction HTML5/CSS3

Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

Le HTML

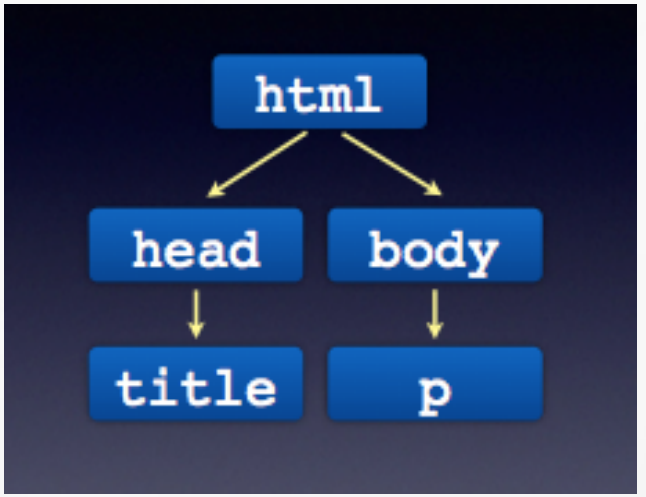


Inventé par Tim Berners-Lee au CERN dans les années 90

- HTML5 pas encore un standard du W3C !

HyperText Markup Language (HTML)

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                <title>mon titre</title>
        </head>

        <body>
                <p>Voici ma première page</p>
        </body>
</html>
```



- Un couple de balises encadrant un texte lui donne une signification particulière pour l'interprétation

- Notion de balises ouvrantes (**<nom_balise>**) et fermantes (**</nom_balise>**)
  - première ouverte, dernière fermée

Structure d'une page HTML

- **<html> … </html>** encadre tout le document HTML
- **<body> … </body>** encadre le contenu
- **<head> … </head>** encadre le titre et les méta-informations de la page
  - méta-informations utilisées par exemple par les moteurs de recherche

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                <title>mon titre</title>
                <meta name="keywords" content="cours, html, essai" />
                <meta name="description" content="Accueil" />
        </head>
 …
</html>
```

- Les commentaires sont délimités pas **<!--** et **-->**

Ajouter du texte dans la page

```
<body>
        <p>Bienvenu sur ma page personnelle !
Je suis maître de conférence en informatique à l'Université de la Nouvelle-Calédonie depuis le 1er septembre 2008.
        </p>
</body>
```

- ** **texte** ** texte forme un paragraphe (saut de ligne avant et après)

- **Attention**
  - **les espaces, les retours à la ligne ne "marchent pas"**
  - ➢ on doit les "coder"

    ** ** code un espace insécable

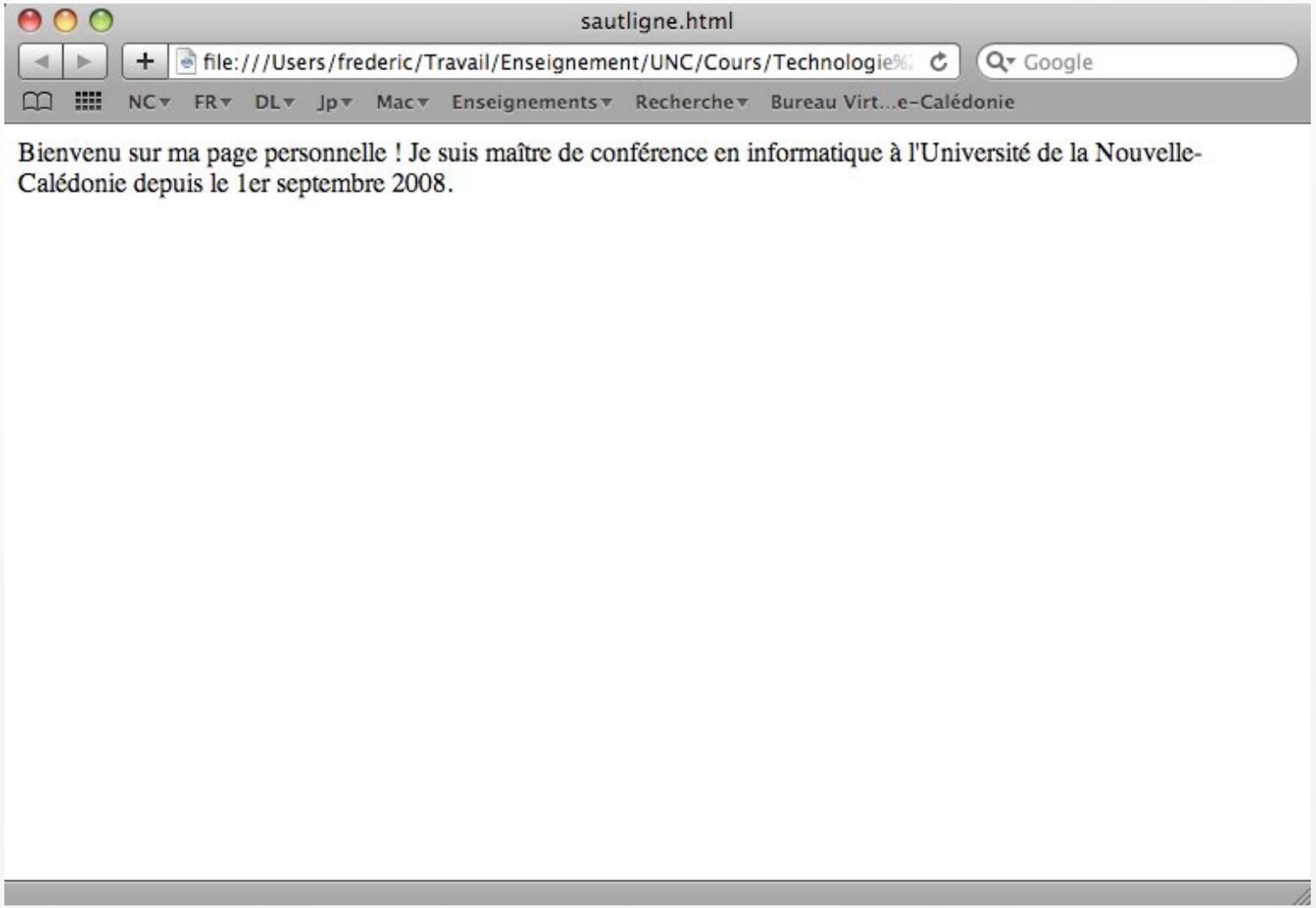
    **
** code un saut de ligne (*balise autonome*)

  - ➢ A ne pas utiliser pour faire de la mise en page !!!!
    - mise en page, styles définis par la feuille de style CSS

Ajouter du texte dans la page

```
<!DOCTYPE html>
<html lang="fr">
  <head>
            <title> Ma page personnelle </title>
            <meta name="keywords" content="page perso, flouvat, calédonie, université,
fouille de données" />
             <meta name="description" content="Page personnelle de Frédéric Flouvat" />
  </head>
  <body>
       <p>Bienvenu sur ma page personnelle !
       Je suis maître de conférence en informatique à l'Université de la Nouvelle-
       Calédonie                    depuis le 1er septembre 2008.
       </p>
  </body>
</html>
```

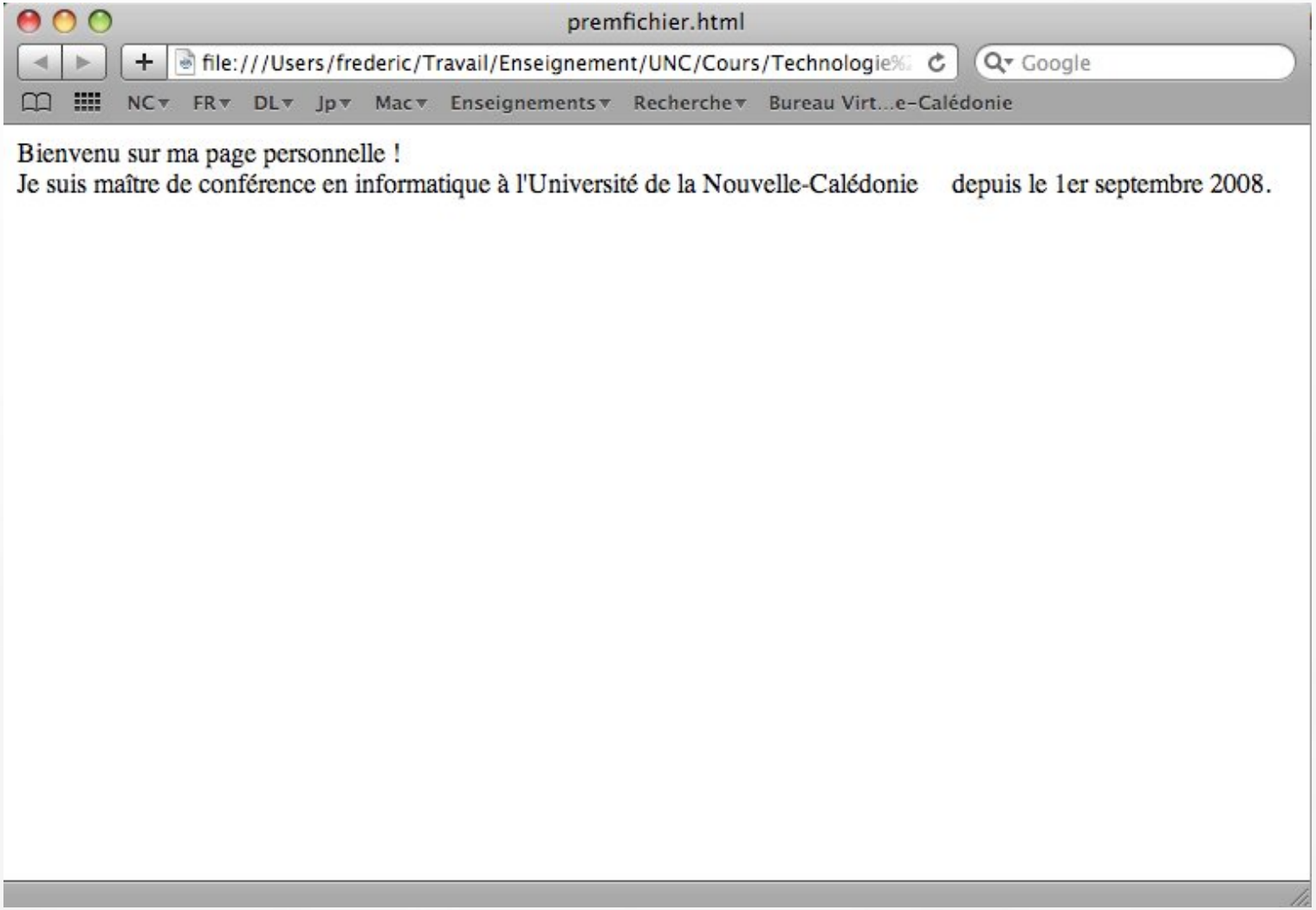
Ajouter du texte dans la page



Ajouter du texte dans la page

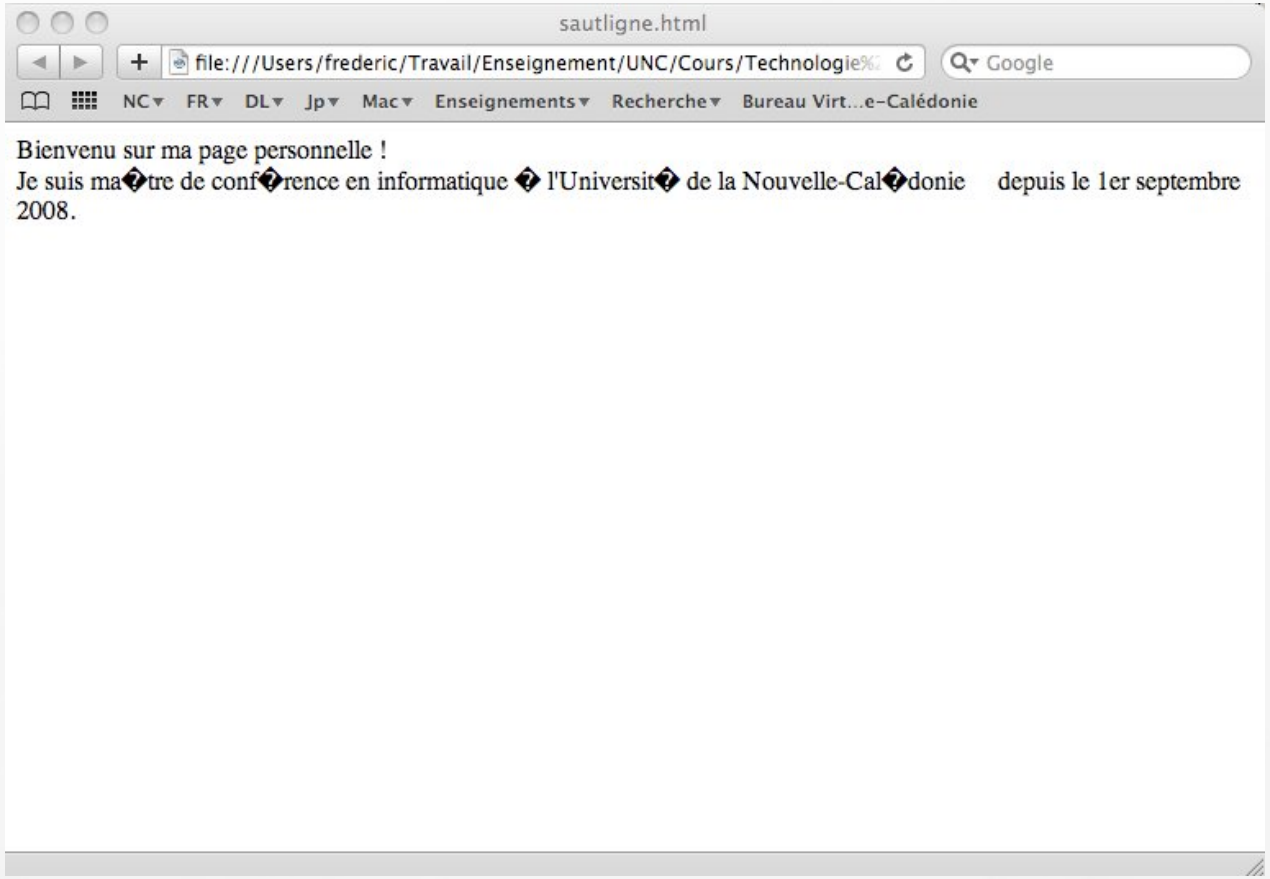
```
<!DOCTYPE html>
<html lang="fr">
  <head>
            <title> Ma page personnelle </title>
            <meta name="keywords" content="page perso, flouvat, calédonie, université,
fouille de données" />
             <meta name="description" content="Page personnelle de Frédéric Flouvat" />
  </head>
  <body>
       <p>Bienvenu sur ma page personnelle ! <br />
       Je suis maître de conférence en informatique à l'Université de la Nouvelle-
       Calédonie           depuis le 1er septembre 2008.
       </p>
  </body>
</html>
```

Ajouter du texte dans la page



Problèmes avec les caractères spéciaux (accents, cédilles...)

- Dans certains cas, on peut obtenir l'affichage suivant



alors que le code HTML est correcte …

Problèmes avec les caractères spéciaux (accents, cédilles...)

- Problème de reconnaissance de l'encodage du fichier
  - avec quel genre de caractères le fichier a-t-il été écrit ?
    - p.ex. Windows = Windows-Latin 1, MacOs = MacOS Roman
- Définir l'encodage du fichier dans les méta-informations du document HTML

```
<!DOCTYPE html>
<html lang="fr">
 <head>
     ...
     <meta http-equiv="Content-Type" content="text/html;charset=iso-8859-1" />

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

Problèmes avec les caractères spéciaux (accents, cédilles...)

- Certains caractères doivent tout de même être codés par des mots-clés

<	**<**
>	**>**
&	**&**
"	**"**

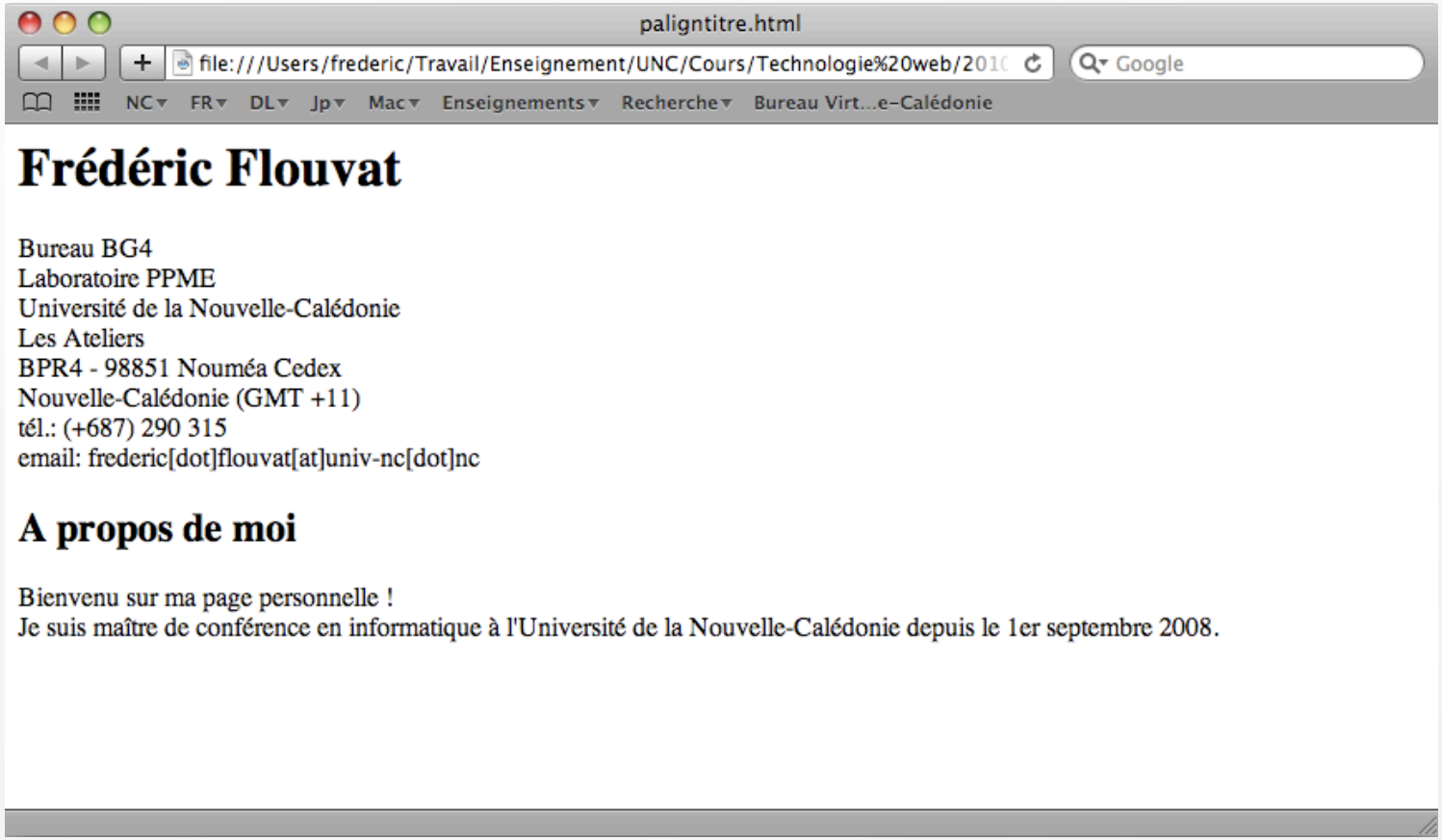
Ajouter des titres

- **<h1>**texte**</h1>** gros titre
- **<h2>**texte**</h2>** plus petit que h1
- …
- **<h6>**texte**</h6>** plus petit titre

```
<body>
  <h1>Frédéric Flouvat</h1>
  <p> Bureau BG4 <br />
      Laboratoire PPME <br />
      Université de la Nouvelle-Calédonie <br />
      Les Ateliers <br />
      BPR4 - 98851 Nouméa Cedex <br />
      Nouvelle-Calédonie (GMT +11) <br />
      tél.: (+687) 290 315 <br />
      email: frederic[dot]flouvat[at]univ-nc[dot]nc
  </p>

  <h2>A propos de moi</h2>
  <p> Bienvenu sur ma page personnelle ! <br />
      Je suis maître de conférence en informatique à l'Université de la Nouvelle-Calédonie  depuis le 1er
      septembre 2008.
  </p>
</body>
```

Ajouter des titres

- **<h1>**texte**</h1>** gros titre
- **<h2>**texte**</h2>** plus petit que h1
- …
- **<h6>**texte**</h6>** plus petit titre



Ajouter des listes

- Liste ordonnées (**o**rdered **l**ist)

```
<body>

  <ol>
        <li> premier élément </li>
        <li> deuxième élément </li>
  </ol>

</body>
```

1. premier élément
2. deuxième élément

- Liste non ordonnées (**u**nordered **l**ist)

```
<body>

  <ul>
        <li> premier élément </li>
        <li> deuxième élément </li>
  </ul>

</body>
```

- premier élément
- deuxième élément

Ajouter des tableaux

```
<table>
  <tr>
    <th>titre colonne 1</th> <th>titre colonne 2</th> <th>titre colonne 3</th>
  </tr>
  <tr>
    <td> cellule 1,1 </td> <td> cellule 1,2 </td> <td> cellule 1,3 </td>
  </tr>
 <tr>
    <td> cellule 2,1 </td> <td> cellule 2,2 </td> <td> cellule 2,3 </td>
  </tr>
</table>
```

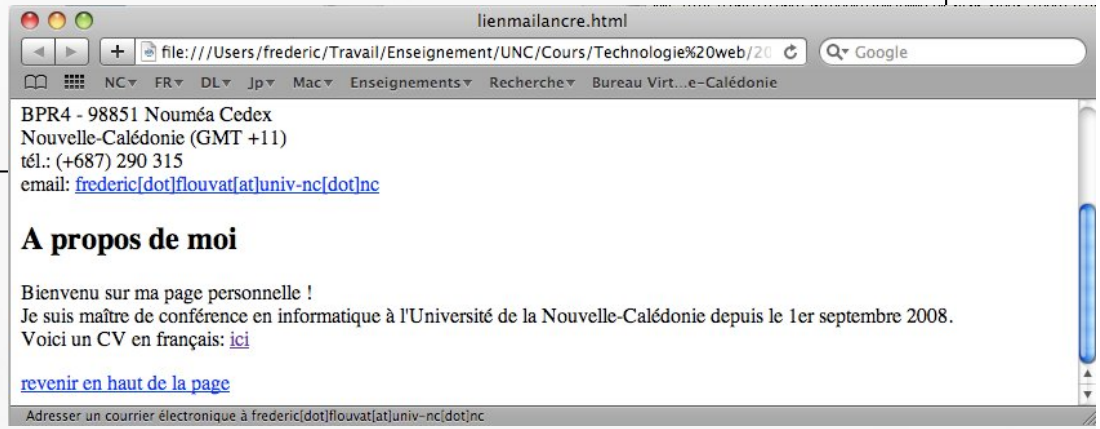
- Définition ligne après ligne
- Chaque ligne doit contenir le même nombre de cases

titre colonne 1	titre colonne 2	titre colonne 3
cellule 1,1	cellule 1,2	cellule 1,3
cellule 2,1	cellule 2,2	cellule 2,3

- Possibilité d'ajouter une légende : **<caption> … </caption>**
- Possibilité des cases
  - au niveau des lignes: attribut **rowspan** de **<td>**
  - au niveau des colonnes: attribut **colspan** de **<td>**
    - ex. **<td colspan="2" >** cellule 1,1 + 1,2 **</td> <td>** cellule 1,3 **</td>**

Ajouter un lien hypertexte

```
…
<h1> <a href="images/photo.jpg">Frédéric Flouvat</a>  </h1>
<p>
  Bureau BG4 <br />
  <a href="http://ppme.univ-nc.nc/">Laboratoire PPME</a> <br />
  <a href="http://www.univ-nc.nc/">Université de la Nouvelle-Calédonie</a> <br />
   …
</p>
….
```



- **href** : une adresse Web ou un chemin relatif vers un fichier
  - mettre des "/" et pas des "\"
  - jamais de chemin absolu (p.ex. "*c:\site\contacts.html*"), sinon ça ne marchera plus sur le serveur Web

- **<a>** : peut encadrer n'importe quel contenu (texte, liste, tableau, image,…)

Ajouter un lien hypertexte interne à une page

```
<body>
 <h1>
   <a id="debut">
          <a href="images/photo.jpg">Frédéric Flouvat</a>
   </a>
 </h1>
...
 <p>
   <a href="#debut">revenir en haut de la page</a>
 </p>
</body>
```

1. Définir un point d'ancrage **...**
   - invisible dans le navigateur
2. Définir un lien vers ce point d'ancrage **...**

- Possibilité de mixer lien externe et ancre
  - p.ex. ...

Ajouter des images

```
<body>
…
  <p>
    <img src="images/foot.jpg" alt="Affichage du bandeau UNC"/>
    <a href="#debut">revenir en haut de la page</a>
  </p>
</body>
```

- **alt**: texte à afficher à la place de l'image si celle-ci ne peut être affichée

- **Attention** à la taille des images
  - grosse image = temps de chargement de la page long

  - redimensionner *physiquement* les images (avec un logiciel de traitement d'images)
  - ne pas insérer directement de grosses images
    - faire 2 versions de l'image: une petite (taille "vignette") et une taille réelle
    - insérer dans la page la petite, puis y associer un lien hypertexte vers la grande

Ajouter du contenu audio et vidéo

```
<audio src="medias/son.mp3" controls></audio>

<video controls loop autoplay>
   <source src="medias/butterfly.mp4" type="video/mp4">
   <source src="medias/butterfly.ogv" type="video/ogg">
</video>
```

- **Attention**: syntaxe HTML5 non supportée par tous les navigateurs
  - cf. http://caniuse.com

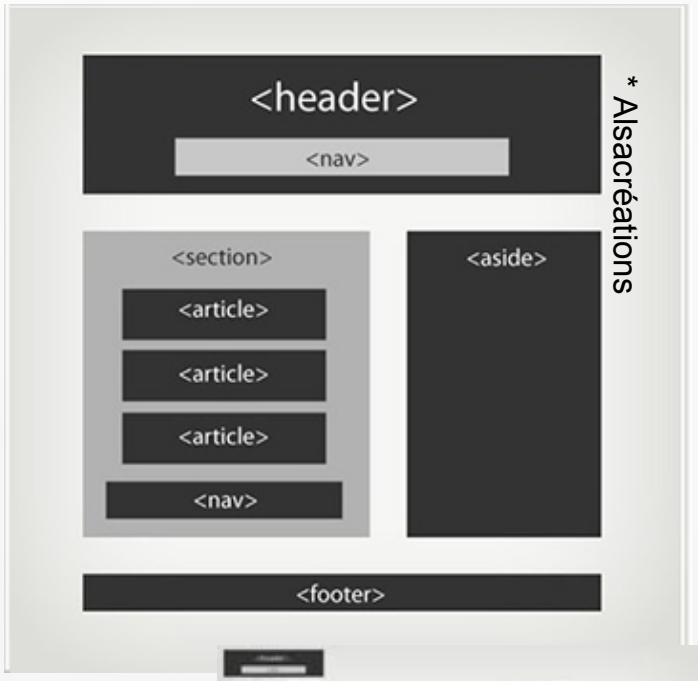
- Possibilité d'inclure des vidéos externes (p.ex. Youtube)
  - génération du code à partir de Youtube : *Partager* > *Intégrer*

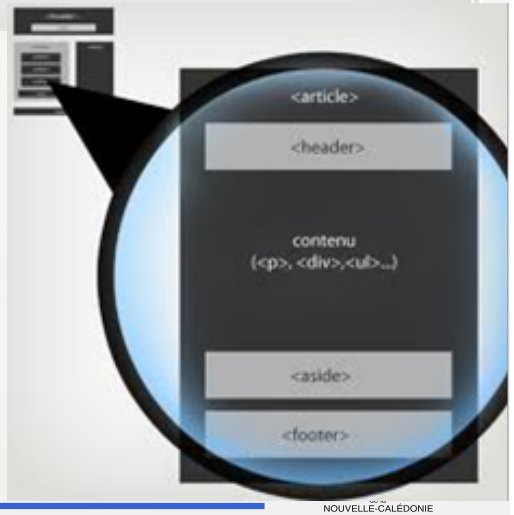
```
<iframe width="560" height="315" src="https://www.youtube.com/embed/2q2Wx5H6wkg"
frameborder="0" allowfullscreen> </iframe>
```

Les balises sémantiques du HTML5

- Apporter de la sémantique en regroupant les balises du document en fonction de leur contenu

<section>	une section d'un document regroupant des informations sur un même sujet
<article>	un article d'un document
<nav>	liens de navigations principaux
<aside>	contenu, information complémentaire
<header>	en-tête de page, introduction d'une section ou d'un article
<footer>	pied de page, conclusion d'une section ou d'un article
<figure> (<figcaption>)	illustration, diagramme, photos, … (avec possibilité de définir un titre)
…	…





- **Attention**: non compatible Internet Explorer < 9

```
<!--[if lt IE 9]>
  <script src="//html5shim.googlecode.com/svn/trunk/html5.js"> </script>
<![endif]-->
```

Remarque sur le style

- Possibilité d'inclure dans le HTML des informations de style
  - attribut **style**

```
<span style="color:#FF0000">Hello</span>
<span style="background-color:#FF0000">Hello</span>
<span style="font-family:cursive">Hello</span>
```

- **Fortement déconseillé !!!**

- **Mettre la mise en page dans une feuille de style externe pour séparer fond et forme ➔ CSS**

La validation du code HTML

- Vérifie que le document est bien écrit
  - site W3C validator (http://validator.w3.org/)
  - **https://validator.w3.org/nu/**

  - outils de développement intégrés aux navigateur
    - p.ex. en affichant le code source dans Firefox (regarder ce qui est en rouge)

- **Attention:** les pages non validées risquent d'être affichées différemment en fonction des navigateurs

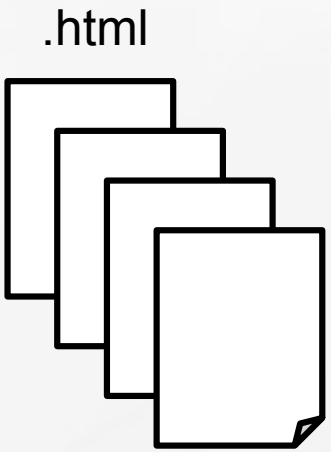
- *Rq*: lorsque le code est valide, pensez à ajouter l'icône de validation du W3C
  - http://www.w3.org/html/logo/

```
<a href="http://www.w3.org/html/logo/">
<img src="http://www.w3.org/html/logo/badge/html5-badge-h-solo.png"
width="63" height="64" alt="HTML5 Powered" title="HTML5 Powered">
</a>
```

Les feuilles de style CSS

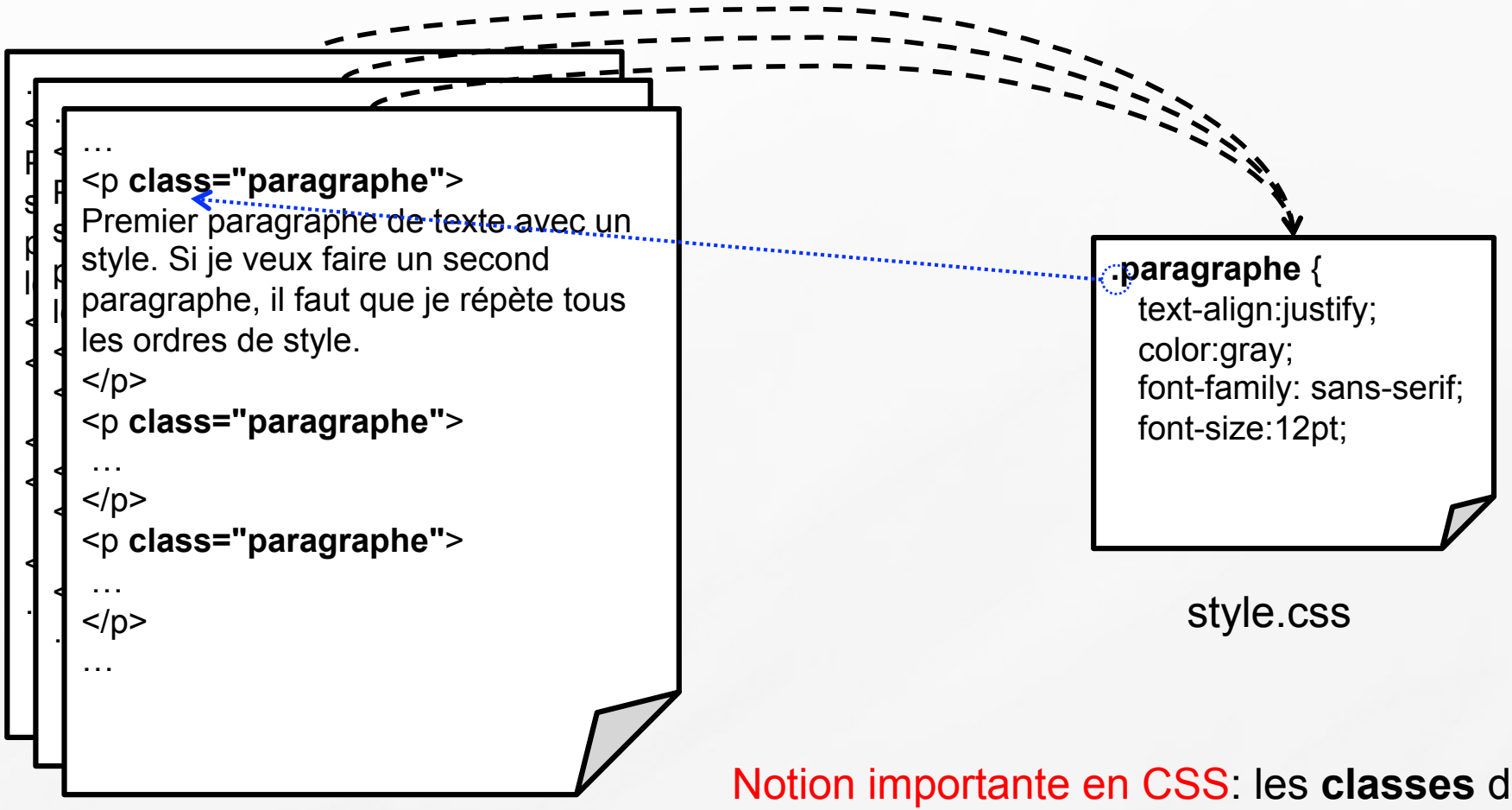
- CSS = Cascading Style Sheets
- **Séparer fond et forme**
  - 1 contenu, plusieurs affichages
  - faciliter le développement et la maintenance

```
<p style="text-align:justify; color:gray; font-family:sans-serif; font-size:12pt">
Premier paragraphe de texte avec un style. Si je veux faire un second paragraphe, il faut que je répète tous les ordres de style.
</p>
<p style="text-align:justify; color:gray; font-family:sans-serif; font-size:12pt">
 …
</p>
<p style="text-align:justify; color:gray; font-family:sans-serif; font-size:12pt">
 …
</p>
```



- beaucoup de copier/coller !!
  - nécessité de modifier toutes les pages en cas de mise à jours du style du site
- une seule mise en page par site
  - PC ? tablette ? mobile ?

Les feuilles de style CSS



Notion importante en CSS: les **classes** de style

- Les CSS par l'exemple : http://www.csszengarden.com/

Faire le lien avec la feuille de style

```
<html lang="fr">
  <head>
        <link rel="stylesheet" href="style.css">
        ...
  </head>
 …
</html>
```

- Possibilité d'ajouter l'attribut **media** pour associer une feuille de style à un support spécifique
  - valeur = all, braille, handheld, print, screen, tv, …

```
<head>
  <link rel="stylesheet" href="theme.css">
  <link rel="stylesheet" href="small.css" media="screen and (max-width: 800px)" >
  <link rel="stylesheet" href="print.css" media="print">
</head>
```

Redéfinir, regrouper et combiner les styles

- Dans le **HTML**,
  - Combiner des classes

```
<p class="aligneGauche styleAncien">
```

  - Associer une même classe à des éléments différents

```
<p class= "styleTexte">
 ...
<span class= "styleTexte">
```

- Dans le **CSS**,
  - Redéfinir le style par défaut des balises
    - pout tout le document

```
p {
    color: red ;
    font-family: helvetica ;
}
```

  - Possibilité de définir plusieurs informations en même temps pour certains attributs de style (p.ex. *font*)

```
h1 { font: bold 12pt helvetica; }
```

  - Décrire plusieurs styles en même temps

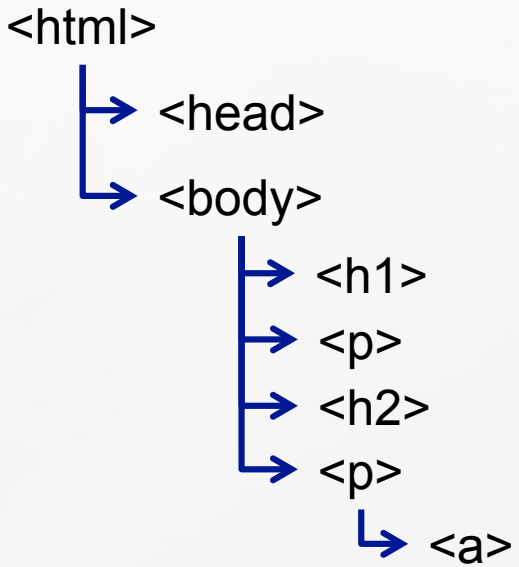
```
h1 { color : red; }

h2 { color : red; }
```

```
h1, h2 { color : red; }
```

Héritage et CSS

- Héritage en cascade des styles

```
<!DOCTYPE html>
<html lang="fr">
  <head>
                …
  </head>
  <body>
    <h1>Frédéric Flouvat</h1>
    <p> Bureau BG4 <br />
        Laboratoire PPME <br />
        Université de la Nouvelle-Calédonie <br />
        BPR4 - 98851 Nouméa Cedex <br />
        email: frederic[dot]flouvat[at]univ-nc[dot]nc
    </p>
    <h2>A propos de moi</h2>
    <p> Bienvenu sur ma page personnelle ! <br />
        Je suis maître de conférence en informatique à l'
        <a href="http://www.univ-nc.nc">Université de la
        Nouvelle-Calédonie</a> depuis le 1er septembre
        2008.
    </p>
    …
  </body>
</html>
```



Les styles non redéfinis sont répercutés dans tous les éléments contenus

Héritage et CSS

- Héritage en cascade des styles

```
<!DOCTYPE html>
<html lang="fr">
 <head>
            …
 </head>
 <body>
   <h1>Frédéric Flouvat</h1>
   <p> Bureau BG4 <br />
       Laboratoire PPME <br />
       Université de la Nouvelle-Calédonie <br />
       BPR4 - 98851 Nouméa Cedex <br />
       email: frederic[dot]flouvat[at]univ-nc[dot]nc
   </p>
   <h2>A propos de moi</h2>
   <p> Bienvenu sur ma page personnelle ! <br />
       Je suis maître de conférence en informatique à l'
       <a href="http://www.univ-nc.nc">Université de la
       Nouvelle-Calédonie</a> depuis le 1er
       septembre 2008.
   </p>
   …
 </body>
</html>
```

Objectif: mettre en bleu et en gras les liens qui sont dans un paragraphe, et laisser les autres en noir et en italique

```
body {
      color: black;
      font-family: Arial;
}

p {
      font-family: Helvetica;
}

a {
       color: blue;
       font-weight: bold;
 }
```

style.css

Problème: tous les liens hypertextes de la page seront modifiés

Héritage et CSS

- Héritage en cascade des styles

```
<!DOCTYPE html>
<html lang="fr">
 <head>
            …
 </head>
 <body>
   <h1>Frédéric Flouvat</h1>
   <p> Bureau BG4 <br />
       Laboratoire PPME <br />
       Université de la Nouvelle-Calédonie <br />
       BPR4 - 98851 Nouméa Cedex <br />
       email: frederic[dot]flouvat[at]univ-nc[dot]nc
   </p>
   <h2>A propos de moi</h2>
   <p> Bienvenu sur ma page personnelle ! <br />
       Je suis maître de conférence en informatique à l'
       <a href="http://www.univ-nc.nc">Université de la
       Nouvelle-Calédonie</a> depuis le 1er septembre
       2008.
   </p>
   …
 </body>
</html>
```

Solution:

style.css

```
body {
      color: black;
      font-family: Arial;
      font-weight: normal;
}

p {
      font-family: Helvetica;
}

a {
      font-style: italic;
}

p a {
       color: red;
       font-weight: bold;
 }
```

Limiter la portée d'un style

- Dans le **CSS**,
  - Restreindre le style à une balise

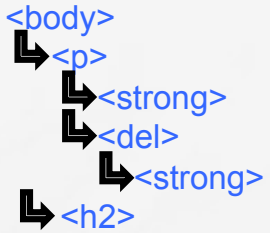
```
p.styleTexte { color : red ; }
```

```
<p class= "styleTexte">
  le texte est rouge
</p>
<p> le texte n'est pas rouge </p>
```

  - Appliquer un style uniquement s'il est imbriqué dans une balise (ou un autre style)

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

```
<p>La couleur <strong>rouge</strong> est appliquée
ici, mais aussi <del><strong>là</strong></del></p>
<h2> Mais pas là</h2>
```



  - Appliquer un style uniquement s'il est descendant direct d'une balise (ou un autre style)

```
p > strong { color : red ; }
```

```
<p>La couleur <strong>rouge</strong> est appliquée
ici, mais aussi <del><strong>là</strong></del></p>
<h2> Mais pas là</h2>
```

Limiter la portée d'un style

- Dans le **CSS**,
  - Appliquer un style uniquement s'il est suit immédiatement une balise (ou un autre style)

```
p + h2 { color : red ; }
```

```
<p>La couleur <strong>rouge</strong> est appliquée
ici, mais aussi <del><strong>là</strong></del></p>
<h2> Mais pas là</h2>
```

  - ***Rq:*** Possibilité de combiner tous ces sélecteurs contextuels

```
.texteBleu {
   color: blue;
}

ul.texteBleu li em {
   color:red;
}
```

```
<ul class= "texteBleu">
   <li> Hello <em>Red</em> </li>
</ul>

<ul>
   <li> Hello <em> Black </em> </li>
</ul>

<p class="texteBleu"> texte en <em> bleu </em></p>
```

Les pseudo-classes

- Ajouter un style particulier lorsqu'un élément est dans un certain état

- Liste de pseudo-classes associées aux liens (dans l'ordre de déclaration)
  - **:focus** style particulier lorsque le lien à le focus
  - **:link** style particulier lorsque pour les liens non visités
  - **:visited** style particulier lorsque pour les liens visités
  - **:hover** style particulier lorsque la souris passe sur un lien
  - **:active** style particulier lorsque le lien est activé

```
a:link {  color: rgb(255,80,0); }

a:visited {  color: rgb(255,80,0);  }

a:hover {  color: black; }

a:active {   color: green; }
```

id versus class

- **id="nom"**
  - élément unique dans la page
  - dans le fichier CSS référencé par **#nom**

```
#coordonnees {
   font-family: helvetica;
   font-size: 12 pt;
}
```

```
<section id="coordonnees">
    <p> Bureau BG4 <br />
        Laboratoire PPME <br />
        Université de la Nouvelle-Calédonie <br />
        BPR4 - 98851 Nouméa Cedex <br />
        email: frederic[dot]flouvat[at]univ-nc[dot]nc
    </p>
   …
</section>
```

- **class="nom"**
  - plusieurs éléments possibles
  - dans le fichier CSS référencé par **.nom**

```
.publications {
   font-family: arial;
   font-size: 10 pt;
}
```

```
<section class="publications">
   <article>
    …
   </article>
</section>
```

Quelques propriétés CSS

- Le fond de la page
  - background, background-attachment, background-color, background-image, background-position, background-repeat, background-clip, background-origin, background-size
- La police
  - font, font-family, font-size, font-style, font-variant, font-weight, @font-face, font-size-adjust, font-stretch
- Le texte
  - color, direction, letter-spacing, line-height, text-align, text-decoration, text-indent, text-transform, …
- Les listes
  - list-style-image, list-style-position, list-style-type, …
- …

- Remarque sur les **différentes unités**
  - en pourcentage **%**, en pixel **px,** en unité d'imprimerie **pt**, en fonction de la taille courante **em**
  - Attention: toujours préciser l'unité (sauf pour 0)

Le modèle d'affichage CSS

- Notion de **flux**: ordre d'affichage à l'écran = ordre apparition dans le code

- Deux catégories d'éléments en HTML, et donc deux types d'affichage
  - les éléments en "blocs"
    - les éléments blocs s'empilent les uns sur les autres (sur toute la largeur)
    - un élément bloc peut contenir d'autres éléments blocs et/ou lignes
    - p.ex. : article, aside, audio, figure, figcaption, footer, form, h1 ... h6, header, li, ol, p, section, nav, table, ul, video, **div …**

container générique souvent utilisé pour faire de la mise en page et structurer le document

  - les éléments en lignes
    - les éléments en ligne se placent les uns à coté des autres
    - un élément en ligne peut contenir d'autres éléments en ligne (mais pas blocs)
    - un élément en ligne doit être dans un bloc
    - p.ex. : a, img, br, em, strong, **span**,…

container générique permettant de regrouper du texte et de lui appliquer un style différent

```
<h2>A propos de moi</h2>
<p> Bienvenu sur ma page personnelle ! <br /> Je suis maître de conférence en informatique à l' <a href="http://www.univ-nc.nc">Université de la Nouvelle-Calédonie</a> depuis le 1er septembre 2008.
</p>
```

A propos de moi

Bienvenu sur ma page personnelle !
Je suis maître de conférence en informatique à l'Université de la Nouvelle-Calédonie depuis le 1er septembre 2008.

La mise en page du contenu

Les marges et les dimensions des éléments

- Les éléments en **lignes**
  - **marge intérieur**: *padding, padding-top, padding-bottom, padding-left, padding-right*

- Les éléments en **blocs**
  - **marge intérieure**: *padding, padding-top, padding-bottom, padding-left, padding-right*
  - **marge extérieure**: *margin, margin-top, margin-bottom, margin-left, margin-right*
  - **dimensions**: *width, height, min-height, max-height, min-width, max-width*

- Utiliser la propriété ***display*** pour modifier la classe d'un élément
  - *none, inline, block, inline-block, list-item, table-cell, …*

La position des éléments dans la page

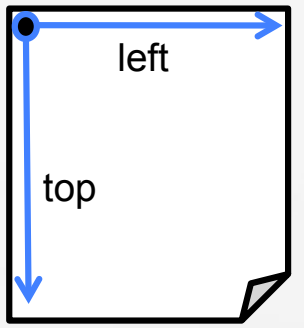
- Utiliser la propriété **position**
  - *static*: placement par défaut

```
div#toto {
   position: static;
}
```

  - *relative* (propriétés **top** et **left** à définir):
    - élément dans le flux de la page
    - décalé par rapport à la position qu'il devrait normalement occuper

```
div#toto {
   position: relative;
   top: 20px;
   left: 10px;
}
```

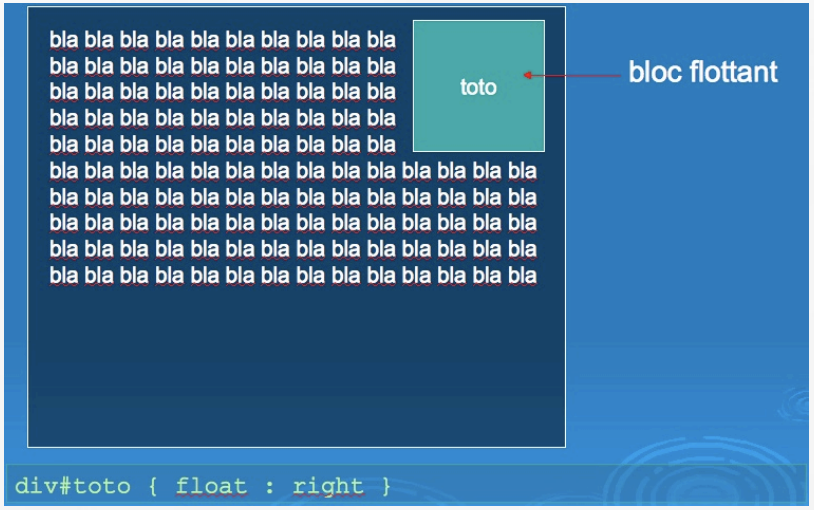
  - *absolute* (propriétés **top** et **left** à définir):
    - élément sorti du flux de la page
      sa position n'est plus déterminée par les éléments qui le précèdent et lui-même n'a plus d'influence sur la position des éléments qui le suivent
    - décalé par rapport à la position de son conteneur "parent"

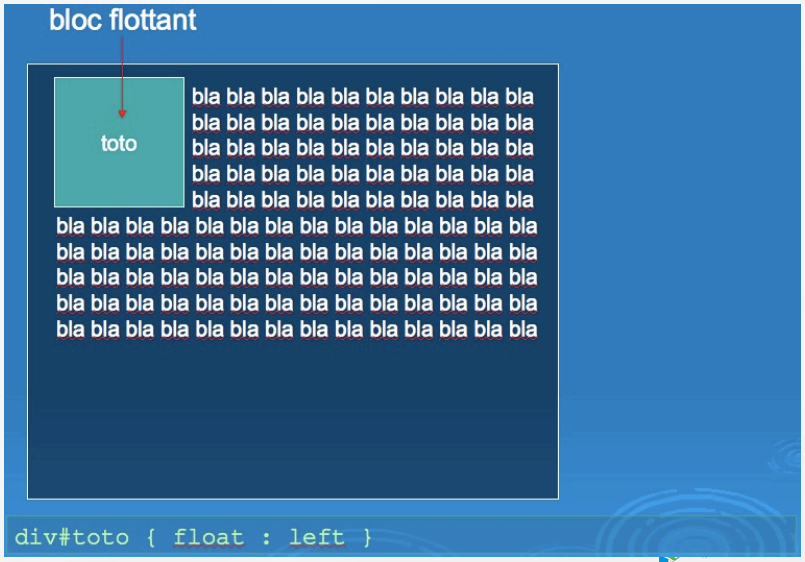


  - *fixed* (propriétés **top** et **left** à définir):
    - élément sorti du flux de la page
    - position fixe (comme épinglé) à l'écran
      positionnement par rapport à la fenêtre du navigateur

La position des éléments dans la page

- Définir un élément flottant par rapport à l'élément suivant
  - propriété **float**
    - *none, left, right*
  - si activé, élément enlevé partiellement du flux et placé à l'extrémité gauche/ droite de son conteneur





Guide de survie de la mise en page CSS

- Erreurs à ne pas commettre
  - trop utiliser la propriété **position** (dans une majorité des cas pas nécessaire)
  - tout positionner en relatif (uniquement pour faire un léger décalage)
  - tout positionner en absolu (risque de chevauchement des contenus et de débordement)
  - attention avec le positionnement fixe (différentes résolutions d'écran)

- Quelle propriétés utiliser? dans quels cas?
  - positionner: garder les éléments dans le flux
    - jouer sur les **margin** et **padding**
    - modifier le mode de rendu avec la propriété **display**
  - centrer des éléments
    - texte centré horizontalement ➔ **text-align**
    - bloc centré horizontalement ➔ marge automatique (valeur à *auto*)
    - centré verticalement le contenu d'un élément ➔ **display***:table-cell* et **vertical-align**:*middle*
  - placer des blocs côte-à-côte, des colonnes
    - préférer **display:** *inline-block* à **float**

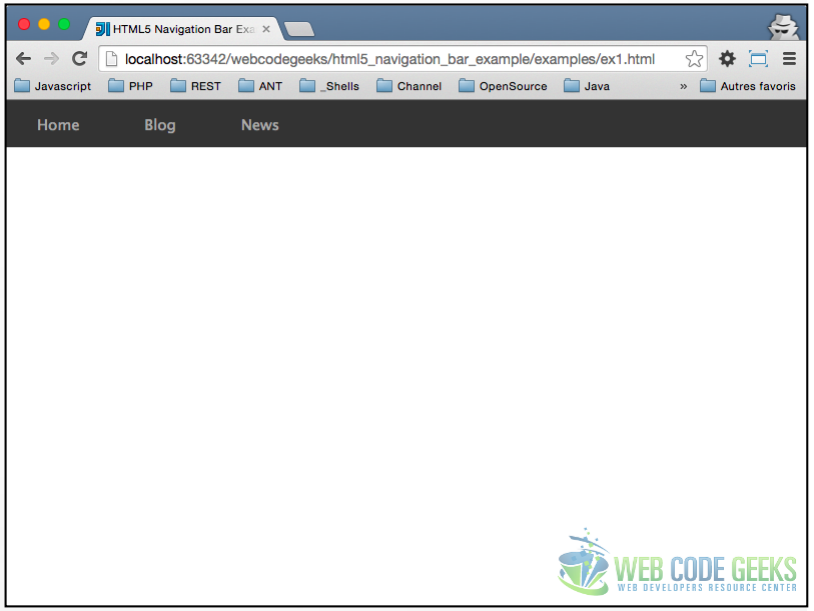
      mais ne marche pas avec les anciennes versions de IE (avant IE 8) et
      des problèmes de mise en page si des espaces "blancs" entre les éléments en ligne

```
<div class="container exemple"><!--
   --><div><p>Lorem Elsass ipsum bredele [...]</p></div><!--
   --><div><p>Lorem Elsass ipsum bredele [...]</p></div><!--
   --><div><p>Lorem Elsass ipsum bredele [...]</p></div><!--
--></div>
```

Exemple: faire un menu horizontal simple en CSS

HTML

```
…
<nav>
    <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="blog.html">Blog</a></li>
        <li><a href="news.html">News</a></li>
    </ul>
</nav>
…
```



CSS

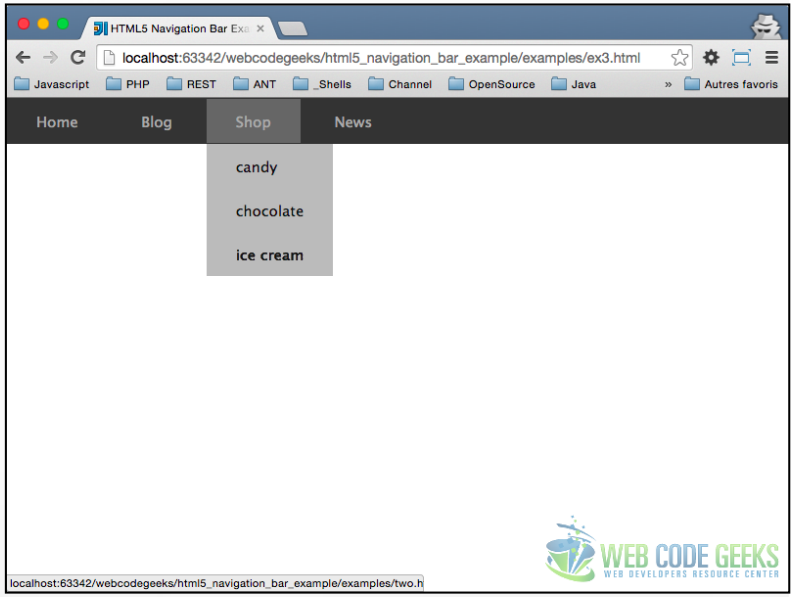
```
nav {
   background-color: #333;
   margin: 0;
   padding: 0;
   overflow: hidden;
}

nav ul{
   margin: 0;
   padding: 0;
}
```

```
nav ul li {
   display: inline-block;
   list-style-type: none;
}

nav > ul > li > a {
   color: #aaa;
   background-color:#333;
   display: block;
   line-height: 2em;
   padding: 0.5em 0.5em;
   text-decoration: none;
}
```

Exemple: faire un menu horizontal aves sous-menus

HTML

```
<nav>
    <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="blog.html">Blog</a></li>
        <li><a href="shop.html">shop</a>
                <ul>
                  <li><a href="candy.html">candy</a></li>
                  <li><a href="chocolate.html">chocolate</a></li>
                  <li><a href="icecream.html">ice cream</a></li>
                </ul>
        </li>
        <li><a href="news.html">News</a></li>
    </ul>
</nav>
```



Ajouter **en plus** dans le CSS

```
nav a:hover {
   background-color: #666;
}

nav li > ul{
   display : none;
}

nav li > ul li {
   display: block;
}
```

```
nav li:hover > ul{
   position:absolute;
   display : block;
   background-color: #aaa;
}

nav li > ul li a {
    color: #111;
    display: block;
    line-height: 2em;
    padding: 0.5em 2em;
    text-decoration: none;
}
```

Encore et toujours valider …

- Vérifier que la feuille de styles CSS est bien écrite
  - https://jigsaw.w3.org/css-validator/validator

- **Attention:** les pages non validées risquent d'être affichées différemment en fonction des navigateurs

- *Rq*: lorsque le code est valide, pensez à ajouter l'icône de validation du W3C
  - http://www.w3.org/html/logo/

```
<p>
  <a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
         src="http://jigsaw.w3.org/css-validator/images/vcss"
         alt="CSS Valide !" />
  </a>
</p>
```

Les frameworks et les prototypes

- Outils mis à disposition par des industriels et des consortiums de développeurs pour faciliter le développement d'applications web
  - tout en suivant des standards, des bonnes pratiques et en garantissant un maximum de compatibilité avec les principaux navigateurs (même les anciennes versions)

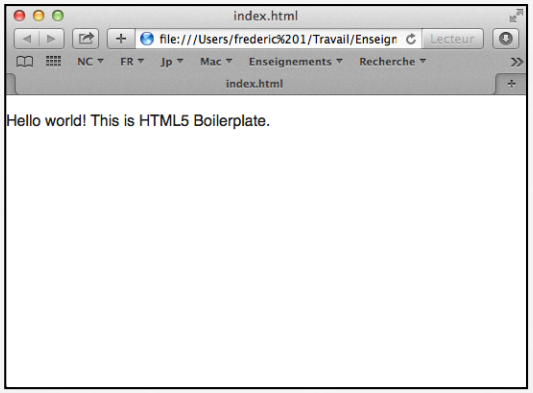
- Deux grandes familles d'outils
  - les prototypes = squelettes d'application, des modèles, permettant d'initier rapidement le développement de pages web
    - **HTML5 Boilerplate** (H5BP)

  - les frameworks de création d'interfaces = collections d'outils permettant de créer des interfaces "responsive" rapidement
    - **Twitter Bootstrap**, Zurb Foundation, Pure, …
      - limites: lourd, complexe, "moins de créativité"

HTML5 Boilerplate

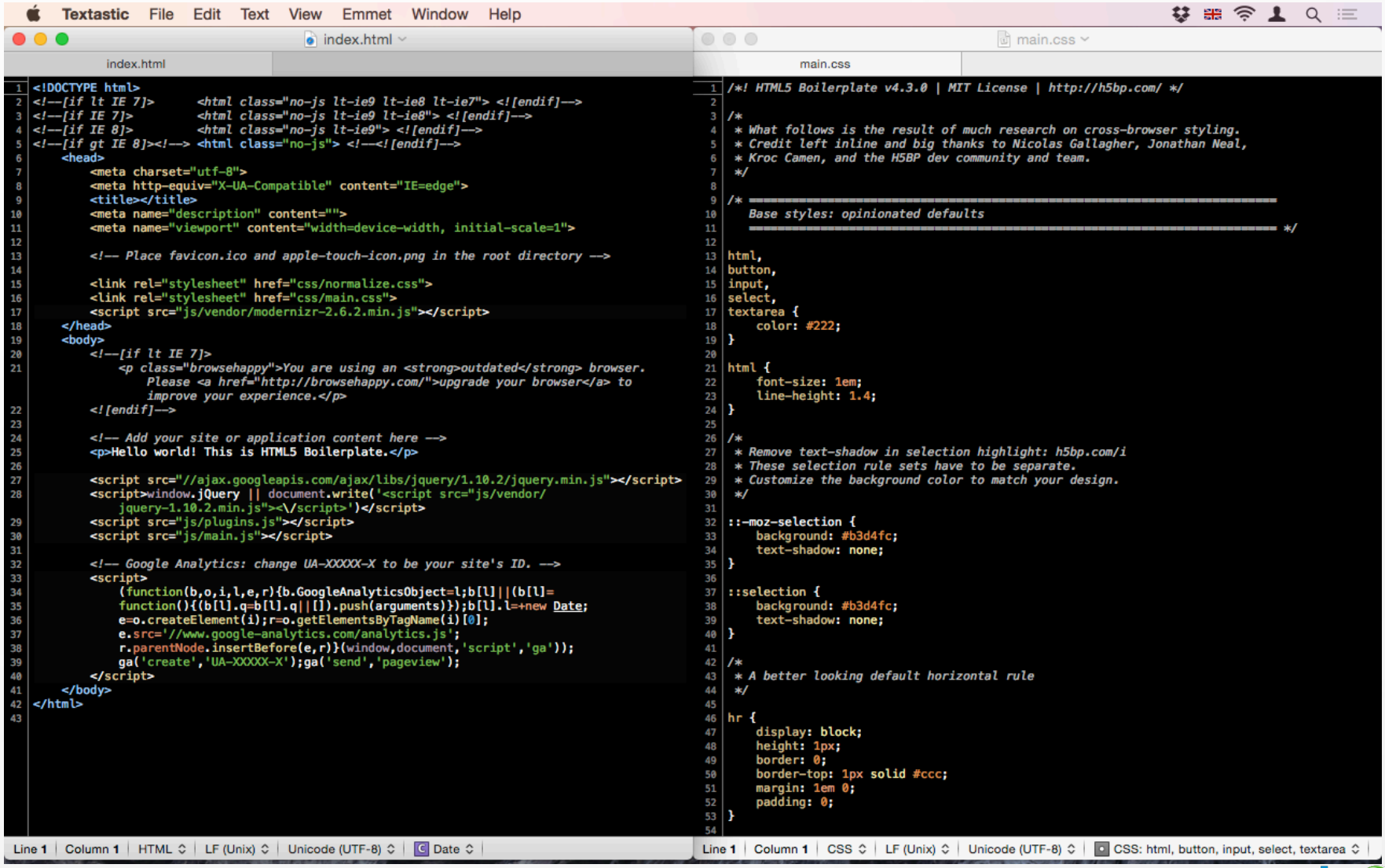
- Un modèle HTML5 permettant de démarrer (*starter template*)
  - fournit une structure de base pour les fichiers et les répertoires
    - HTML, CSS, JavaScript, .htaccess (configuration serveur Apache), …
  - permet aux pages de s'afficher sur les navigateurs "légers" des appareils mobiles, les anciens navigateurs et IE
  - ➢ un concentré de bonnes pratiques et d'optimisations

- Comment utiliser ?
  - télécharger simplement la dernière version sur https://html5boilerplate.com et décompresser
  - modifier les fichiers et insérer votre code




```
.
├── css
│   ├── main.css
│   └── normalize.css
├── doc
├── img
├── js
│   ├── main.js
│   ├── plugins.js
│   └── vendor
│       ├── jquery.min.js
│       └── modernizr.min.js
├── .htaccess
├── 404.html
├── index.html
├── humans.txt
├── robots.txt
├── crossdomain.xml
├── favicon.ico
└── [apple-touch-icons]
```

HTML5 Boilerplate

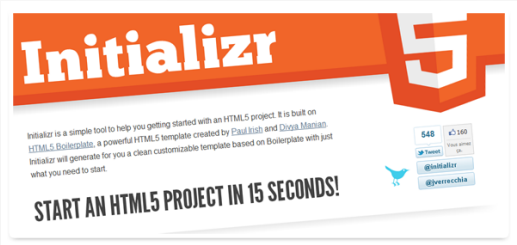


Bootstrap

- Un *front-end framework*
  - fournit des fichiers CSS, des composants JavaScript, des icones et des polices
  - permet d'avoir une interface responsive, compatible avec les différents navigateurs
  - basé sur un système de grilles

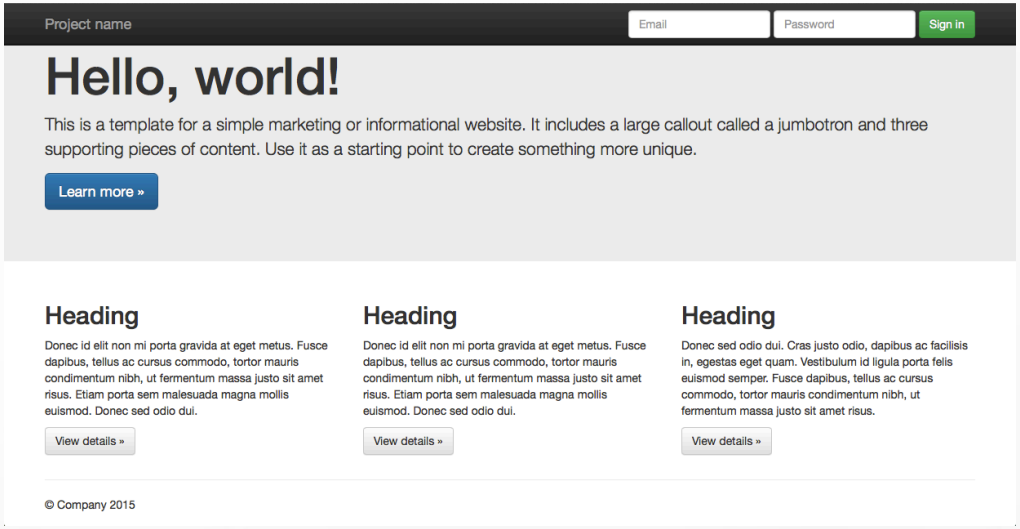
```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

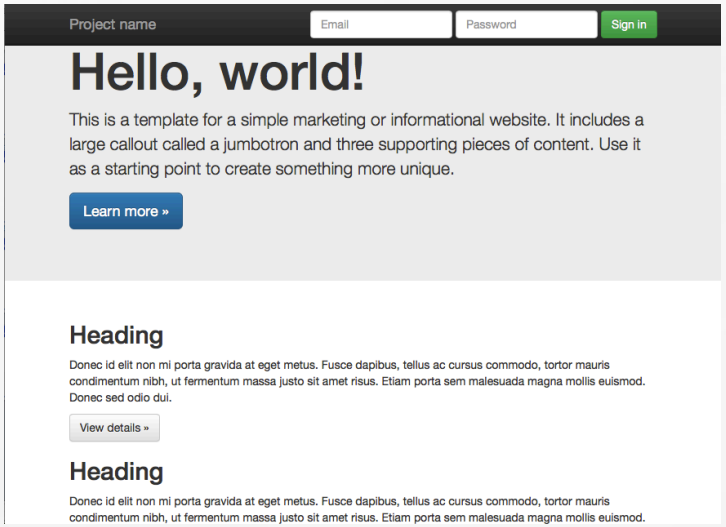
- Comment utiliser ?
  - télécharger et installer la dernière version sur http://getbootstrap.com
  - référencer un des fichier CSS et utiliser ses classes

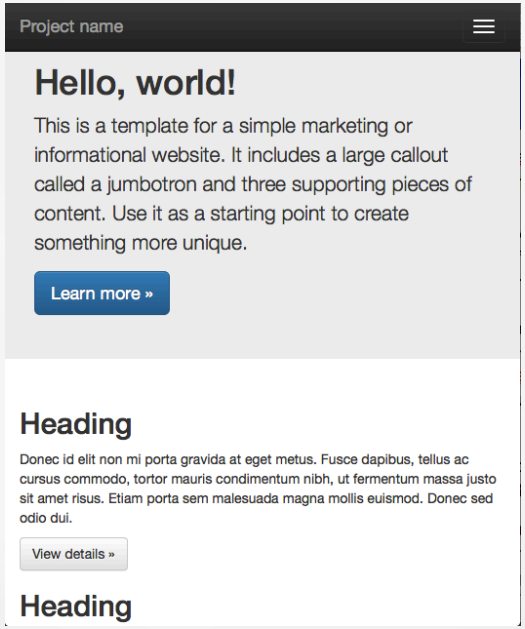


➢ Possibilité d'avoir une installation et un paramétrage simplifiés de HTML5 Boilerplate et de Bootstrap en utilisant **Initializr** sur http://www.initializr.com

Bootstrap


Project name
Email
Password
Sign in
Hello, world!
This is a template for a simple marketing or informational website. It includes a large callout called a jumbotron and three supporting pieces of content. Use it as a starting point to create something more unique.
Learn more »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading
Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.
View details »
© Company 2015


Project name
Email
Password
Sign in
Hello, world!
This is a template for a simple marketing or informational website. It includes a large callout called a jumbotron and three supporting pieces of content. Use it as a starting point to create something more unique.
Learn more »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod.


Project name
Hello, world!
This is a template for a simple marketing or informational website. It includes a large callout called a jumbotron and three supporting pieces of content. Use it as a starting point to create something more unique.
Learn more »
Heading
Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.
View details »
Heading

HTML5 Boilerplate et Bootstrap: quelques références

- https://html5boilerplate.com
- http://www.creativebloq.com/web-design/how-use-html-boilerplate-11513798
- http://www.sitepoint.com/introduction-html5-boilerplate/

- http://getbootstrap.com
- http://getbootstrap.com/getting-started/

- http://www.initializr.com
- http://www.html5-css3.fr/html5/initializr-generateur-template-html5-boilerplate
- http://www.sitepoint.com/boilerplate-bootstrap/

UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Cours de technologies Web
Introduction au PHP

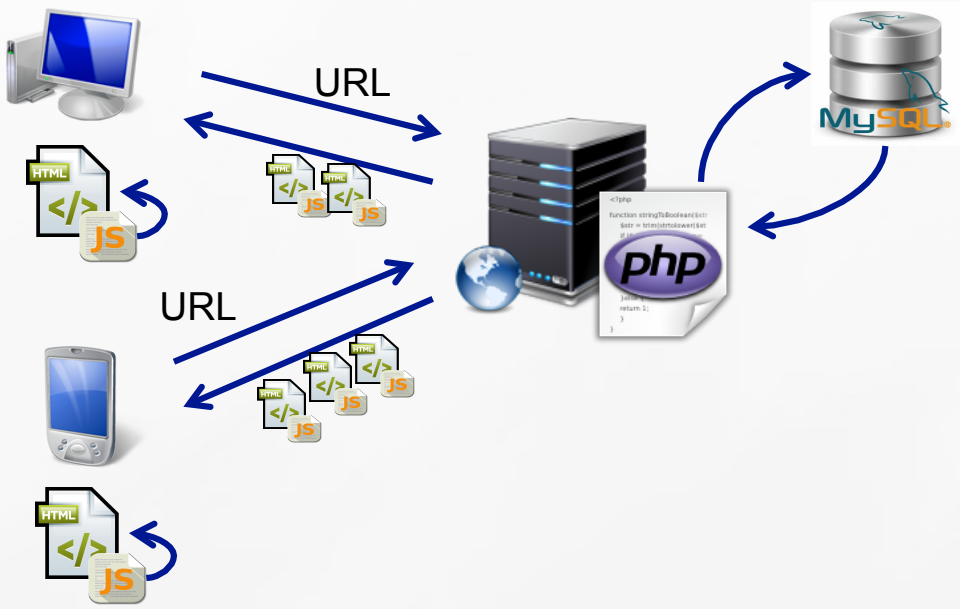
Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

Rappel

- Le PHP, un langage côté serveur
  - génère dynamiquement du HTML
  - accède aux bases de données
  - totalement invisible pour les clients



- Coder en PHP
  - nécessité d'avoir un serveur web d'installé avec un module PHP
  - utiliser des environnements de développement, tel que *Xampp,* pour travailler en local (http://localhost)

- Exécution du code PHP
  - interprétation et exécution du code faite par un moteur PHP intégré au serveur web
  - erreurs PHP insérées dans le code HTML et affichées par le navigateur
  - possibilité d'utiliser des debugger tel que phpdbg (intégré par défaut à partir de PHP 5.6)

Le PHP

- Langage de programmation dont l'objectif est de générer du HTML grâce à des ***echo*** ou des ***print***
  - une syntaxe similaire au C/C++/Java, mais un fonctionnement et une philosophie différents sur certains aspects (p.ex. les variables, les tableaux)

- Fichier source PHP = fichier texte avec l'extension **.php**
  - code PHP entre balises **<?php** …. **?>**
  - peut contenir directement du HTML

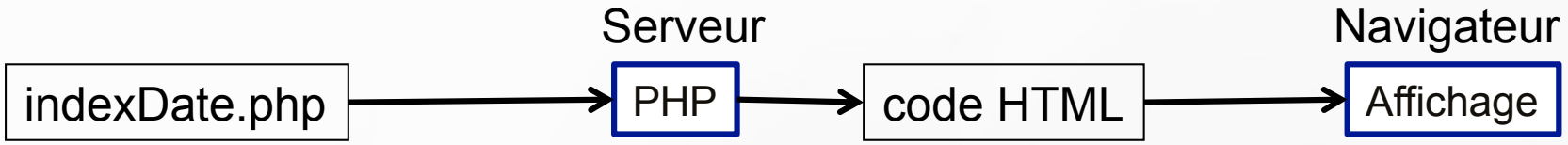
```
<!DOCTYPE html>
<html lang="fr">
        <head>
                …
        </head>
        <body>
                <p>
                    <?php
                        echo "Hello World" ;
                    ?>
                </p>
        </body>
</html>
```

Le PHP

fonction PHP retournant une chaîne de caractères représentant la date courante

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                …
        </head>
        <body>
                <p>Nous sommes le
                    <?php
                        echo  date("F j Y, G:i,") ;
                    ?>
                   à Nouméa.
                </p>
        </body>
</html>
```

Fonctionnement du PHP



indexDate.php

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Exemple 2 PHP</title>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    </head>
    <body>
        <p>Nous sommes le
        <?php
            echo  date("j F Y, G:i,") ;
        ?>
        à Nouméa.
        </p>
    </body>
</html>
```

Source de : http://localhost/cours-TechnoWeb/indexDate.php

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Exemple 2 PHP</title>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    </head>
    <body>
        <p>Nous sommes le
        25 June 2015, 3:30,        à Nouméa.
        </p>
    </body>
</html>
```

Exemple 2 PHP

localhost/cours-TechnoWeb/indexD

Les plus visités À la une Windows Live Hot...

Nous sommes le 25 June 2015, 3:29, à Nouméa.

Le langage et ses spécificités

- PHP offre les mêmes possibilités que des langages tels que C/C++/Java
  - des variables (locales et globales)
  - des tableaux
  - des structures de contrôle (**while**, **for**, **do**, **if**, **switch**)
  - des fonctions (avec passage des paramètres par valeur ou par adresse)
  - …
- Mais avec
  - une gestion des types très souples (langage faiblement typé)
  - une syntaxe spécifique pour les variables et les fonctions
  - des fonctions et des opérateurs spécifiques pour manipuler les chaînes de caractères, les types, les tableaux, …

Les similarités syntaxiques avec le C/C++/Java

- Instructions terminées par **;**
- Délimitation des blocs avec **{** … **}**
- Mise en commentaires avec *//* ou */** … ******/*
- Utilisation de structures de contrôles
  - **while(** *cond* **) {** … **}**
  - **do{** … **} while(** *cond* **);**
  - **for(** *init* **;** *cond* **;** *modif* **){** … **}**
  - **if(** *cond* **){** … **} else if(** *cond* **){** … **} else {** … **}**
  - **switch(** *expr* **){ case** *val_1***:** … **break; case** *val_2***:** … **break; default:** …**}**
- Définition des opérateurs
  - arithmétiques: +, -, *, /, %, ++, --
  - d'affectation: =, +=, -=, *=, /=, .=
  - de comparaison: ==, <, >,<=, >=, !=
  - logiques: &&, ||, ^, !

Les variables en PHP

- Pas de déclaration
  - type défini au moment de l'affectation
- Commence par **$**
  - ex: $i
- Affectation par valeur **=**
  - ex: $i = 1;
- Affectation par référence **=&**
  - ex.: $i =& $tmp ;
- Possibilité de définir des variables de variables (variables dites dynamiques) **$$** ou **${$...}**
  - variable prenant pour valeur une variable et l'utilisant comme nom d'une autre variable

```
$foo = 'original' ;
$bar = $foo ;
$bar = 'modifié' ;
echo $foo ;
```

```
$foo = 'original' ;
$bar =& $foo ;
$bar = 'modifié' ;
echo $foo ;
```

```
$var = 'bonjour' ;
$$var = 'monde' ;
echo $var ;
echo ${$var} ;
echo $$var ;
echo $bonjour ;
```

Les chaînes de caractères en PHP

- Deux syntaxes avec des interprétations différentes
  - chaînes de caractères délimitées par des guillemets simples (p.ex. *'chaîne'*)
    - ➢ contenu non interprété
    - – i.e. les variables apparaissant dans des guillemets simples ne sont pas remplacées
  - chaînes de caractères délimitées par des guillemets doubles (p.ex. *"chaîne"*)
    - – contenu interprété
    - – i.e. si une variable apparaît dans la chaîne, elle est remplacée par sa valeur

```
$age = 20 ;
echo ' j\'ai $age ans' ;
echo " j\'ai $age ans" ;
```

j'ai $age ans
j'ai 20 ans

- **Conseil:** guillemets simples ou doubles ?
  - utiliser les guillemets simples au niveau PHP
  - utiliser les guillemets doubles au niveau HTML

```
echo ' <div class="contenu"> blablabla </div> ' ;
echo ' j\'ai '.$age.' ans' ;
```

Les chaînes de caractères en PHP

- Afficher des caractères réservés: mettre \ devant
- Concaténer deux chaînes de caractères: **.**
- Accéder à un caractère var**[** index **]**
- Convertir en chaîne de caractères: **(string)** devant une variable ou fonction **strval(** *$var* **)**
- Autres fonctions utiles
  - avoir la longueur d'une chaîne de caractères: **strlen(** $var **)**
  - extraire une sous-chaîne: **substr(** $var, $start, $len **)** ou substr($var, $start **)**
    - extrait la sous-chaîne de $str commençant au caractère $start, et faisant éventuellement $len caractères de long

```
echo ' j\'ai '.$age.' ans' ;
$di = $age[0] ;
```

Les tableaux en PHP

- Allocation du tableau et ajout d'éléments fait dynamiquement par PHP

- Plusieurs types de tableaux
  - les tableaux indicés
    - affectation automatique d'indice si pas précisé (indice de la première case vide)

```
$tab = array() ;
$tab[0]='hello ';
$tab[] = 'world ';
```

  - les tableaux associatifs
    - pas d'indices entiers mais une clé en chaîne de caractères

```
$mymovies = array('Vertigo'=>'Hitchcock', 'Alien'=>'Scott');

$films = array();
$films['Burton'] = 'Beetlejuice' ;
$films['Burton'] = 'Sleepy hollow' ;
```

Les tableaux en PHP

- Plusieurs types de tableaux (suite)
  - les tableaux multi-dimensionnels (indicés et/ou associatifs)

```
$mymovies = array( 'Vertigo'=>array('Alfred','Hitchcock'), 'Alien'=>array('Ridley','Scott') );
$films = array();
$films['Burton'][1] = 'Beetlejuice' ;
$films['Burton'][5] = 'Sleepy hollow' ;
```

- Deux modes de parcours
  - parcours classiques avec une boucle **for** ou **while**
  - parcours spécifiques avec **foreach(...)**

```
foreach( $tab as $value )
{
 ...
}
```

```
foreach( $tab as $key => $value )
{
 ...
}
```

- Beaucoup de fonctions mises à dispositions
  - **unset** (effacer), **array_diff** (faire la différence entre deux tableaux), **sort** (trier)

Les tableaux en PHP

```php
<?php
  $tab = array();
  $tab['cle'] = 'Hello';
  $tab[2] = 'World';
  $tab[0] = 'Bye';
  echo 'Le tableau a '.count($tab).' élément(s):<ul>';
  foreach ($tab as $key=>$valeur) {
            echo '<li> '.$key.' associé à '.$valeur.'</li>';
  }
  echo '</ul>';
  var_dump($tab);
?>
```

affiche les informations d'une variable, ici le tableau (très utile lors du débogage)

```php
<?php
  $menu = array();
  $menu['Home'] = 'index.html';
  $menu['News'] = 'news.html';
  $menu['Contact'] = 'contact.html';
  echo "<ul>";
  foreach ($menu as $nom => $url ) {
            echo "<li><a href='$url'>$nom</a></li>";
  }
  echo '</ul>';
?>
```

Les fonctions en PHP

```
<?php
  function somme( $a, $b){
        return $a + $b;
  }

  echo somme(1,2) ;
  echo $a;
?>
```

pas besoin de déclarer le type de retour, ni le type des paramètres

pas les mêmes variables !!

```
$a = 1;
$b = 2;
function somme(){
        global $a, $b;
         res = $a + $b;
}
echo $res;
```

- Portée des variables
  - locale (par défaut)
    - locale à la fonction (si dans une fonction), locale à la page web (sinon)
  - globales
    - variable accessible en dehors des fonctions
  - statiques
    - portée locale mais conserve sa valeur lorsque le script appelle la fonction plusieurs fois

```
function test()
{
    static $count = 0;
    $count++;
    echo $count;
    if ($count < 10)
        test();

    $count--;
}
```

Les fonctions en PHP

- Le passage des paramètres
  - par valeur (par défaut)
    - une copie des paramètres est faite au moment de l'appel
    - modifications non conservées
  - par référence: **&** devant le paramètre
    - pas de recopie
    - modifications conservées

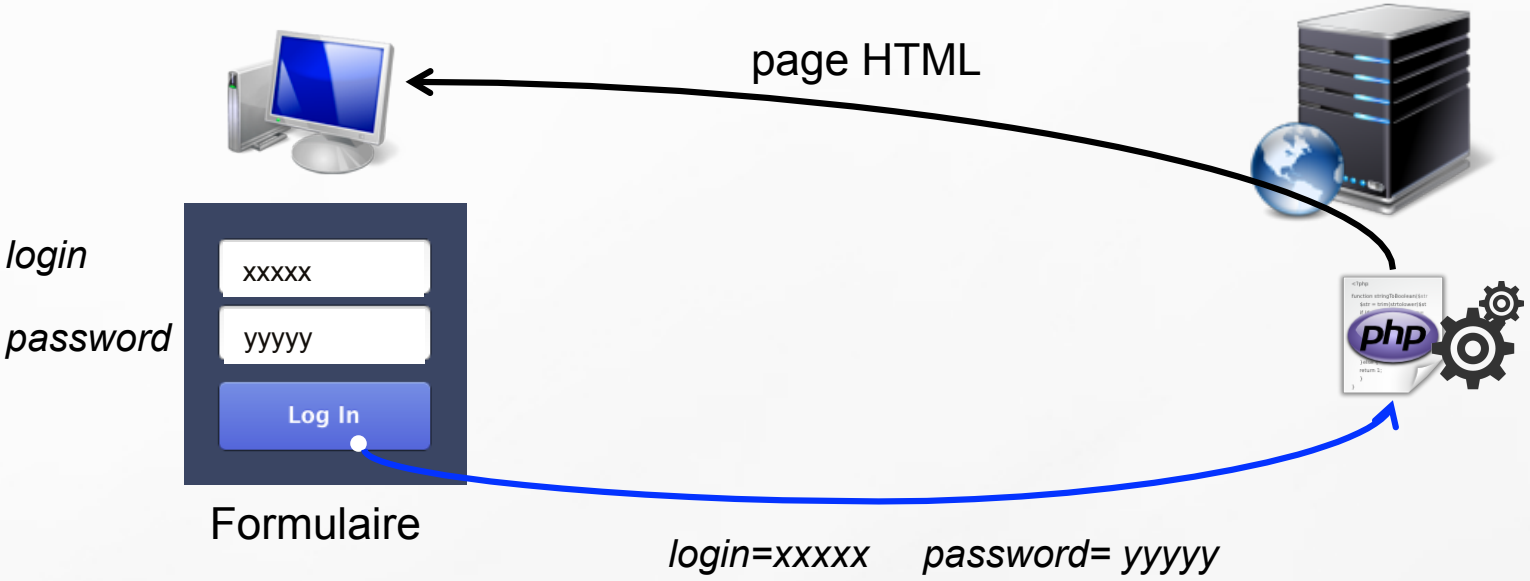
```
function add_some_extra( &$string )
{
    $string .= ', et un peu plus.';
}
$str = 'Ceci est une chaîne';
add_some_extra($str);
echo $str;
```

- Définir des valeurs par défaut

  function *nom_fonction(param0, $param1=valDef1, $param2, …)*

- Astuces:
  - possibilité d'avoir un nombre variable de paramètres en utilisant un tableau en paramètre
  - si le nom d'une variable est suivi de parenthèses, PHP recherchera une fonction de même nom, et essaiera de l'exécuter ➔ utile p.ex. pour faire des tableaux de fonctions

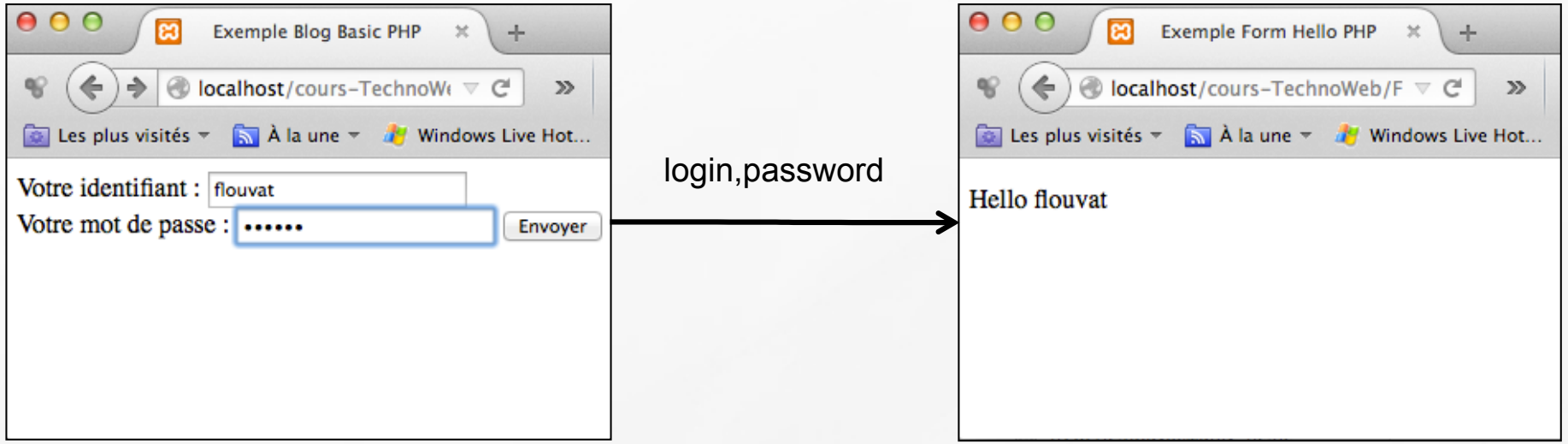
PHP et formulaires HTML: principe



- Saisie des données par un utilisateur dans un formulaire **HTML** et transmission au serveur web (pour traitement) au moment de la validation du formulaire
  - différents composants de saisie: champs textuels, listes déroulantes, cases à cocher, …
  - validation formulaire = clic bouton soumission du formulaire (*submit*)

- Traitement des données saisies en PHP

PHP et formulaires HTML: un premier exemple



index.**html**

hello.**php**

PHP et formulaires HTML: un premier exemple

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Blog Basic PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>
  <form method="post" action="hello.php">

   <label for="login"> Votre identifiant </label> :
   <input type="text" name="login" id="login" placeholder="defaut" />

   <br />

   <label for="password"> Votre mot de passe </label> :
   <input type="password" name="password" id="password" />

   <input type="submit" value="Envoyer">

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

index.html

login
password

hello.php

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Form Hello PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

   <p> Hello <?php echo $_POST['login']; ?> </p>

 </body>
</html>
```

PHP et formulaires HTML: les balises HTML et leurs attributs

- La balise principale : **<form>** … **</form>**
  - url de la page web recevant les données: attribut **action**
  - méthode de transmission des données: attribut **method="POST" ou "GET"**

- **Attention:** impossible d'inclure un formulaire dans un autre

- Par contre, possible d'avoir plusieurs formulaires dans une même page

```
<form method="…" action="…" >

  <label for="…">… </label>
  <input type="…" name="…" id="…" … />

  <label for="…">… </label>
  <textarea name="…" id="…" … >
            …
  </textarea>

  <label for="…">… </label>
  <select name="…" id="…" … >
            <option> … </option>
            <option> … </option>
            …
  </select>

  <button type="…" name="…" id="…" … >
  …
  </button>

</form>
```

PHP et formulaires HTML: les balises HTML et leurs attributs

- Les sous-balises de **form**
  - les principaux composants de saisie

    <input type="…" … />
    - avec **type**=*text, checkbox, radio, file, image, password, hidden, tel, email, search, number, datetime, date, submit, reset, button…*

    <textarea> … </textarea> (zone de texte)
    <select> … </select> (liste déroulante)

    et leurs attributs
    - **name**: nom de la variable dans laquelle la valeur saisie va être stockée
    - **id**: identifiant de la sous-balise
    - **required, placeholder, autofocus, autocomplete, pattern, list, maxlength, …**

```
<form method="…" action="…" >

  <label for="…">… </label>
  <input type="…" name="…" id="…" … />

  <label for="…">… </label>
  <textarea name="…" id="…" … >
            …
  </textarea>

  <label for="…">… </label>
  <select name="…" id="…" … >
            <option> … </option>
            <option> … </option>
            …
  </select>

  <button type="…" name="…" id="…" … >
  …
  </button>

</form>
```

PHP et formulaires HTML: les balises HTML et leurs attributs

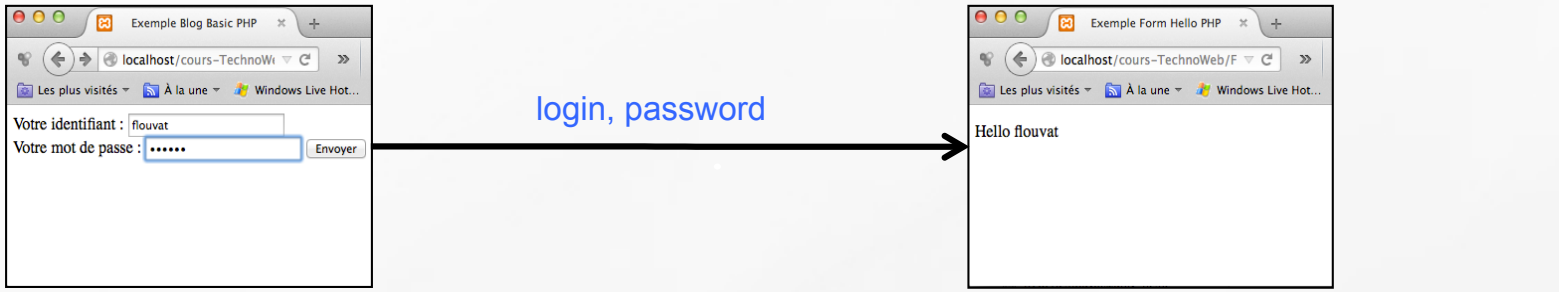
- Les sous-balises de **form** (suite)
  - Les étiquettes **<label> … </label>**
    - attribut **for**: identifiant du composant de saisie
  - **<button type="…"> … </button>** versus **<input type="…" />**
    - avec **type**=*submit*, *reset*, *button*
      - tous les deux, un bouton
      - seule différence: plus de possibilités pour modifier le style avec **button**
        - changer le texte, faire un bouton à partir d'une image, …
      - attention: des problèmes avec certaines versions de IE (p.ex. IE6) et button

```
<form method="…" action="…" >

  <label for="…">… </label>
  <input type="…" name="…" id="…" … />

  <label for="…">… </label>
  <textarea name="…" id="…" … >
            …
  </textarea>

  <label for="…">… </label>
  <select name="…" id="…" … >
            <option> … </option>
            <option> … </option>
            …
  </select>

  <button type="…" name="…" id="…" … >
  …
  </button>

</form>
```

PHP et formulaires HTML: retour sur la transmission des paramètres

- Deux méthodes de transmission des paramètres <form **method**="..." ... >...</form>
  - ***get*** **:** transmission des données dans l'adresse de la page

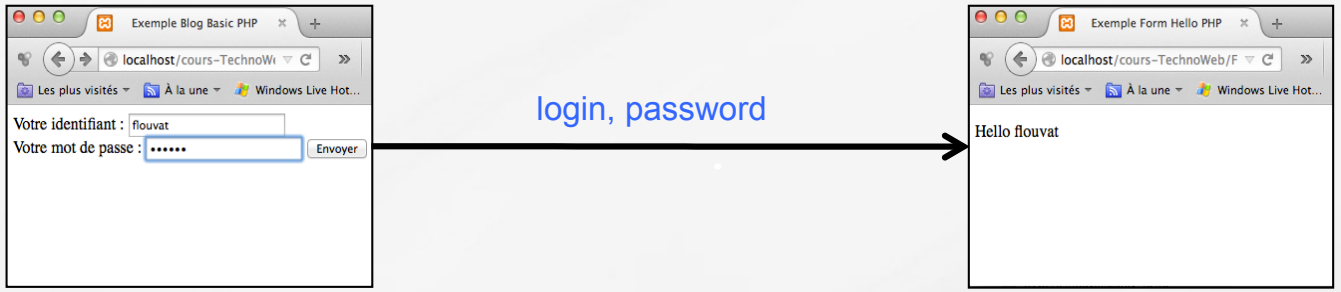
    *url***?***var1***=***val1***&***var2***=***val2* …



index.html

Hello.php**?login=**flouvat**&password=**truc

  - ***post*** : transmission des données séparément de l'adresse web
    - ➔ permet de cacher les informations transmises



index.html

Hello.php

PHP et formulaires HTML: récupération des données en PHP

- Principe:
  - l'interpréteur PHP initialise un tableau associatif qui associe à chaque nom de paramètre sa valeur

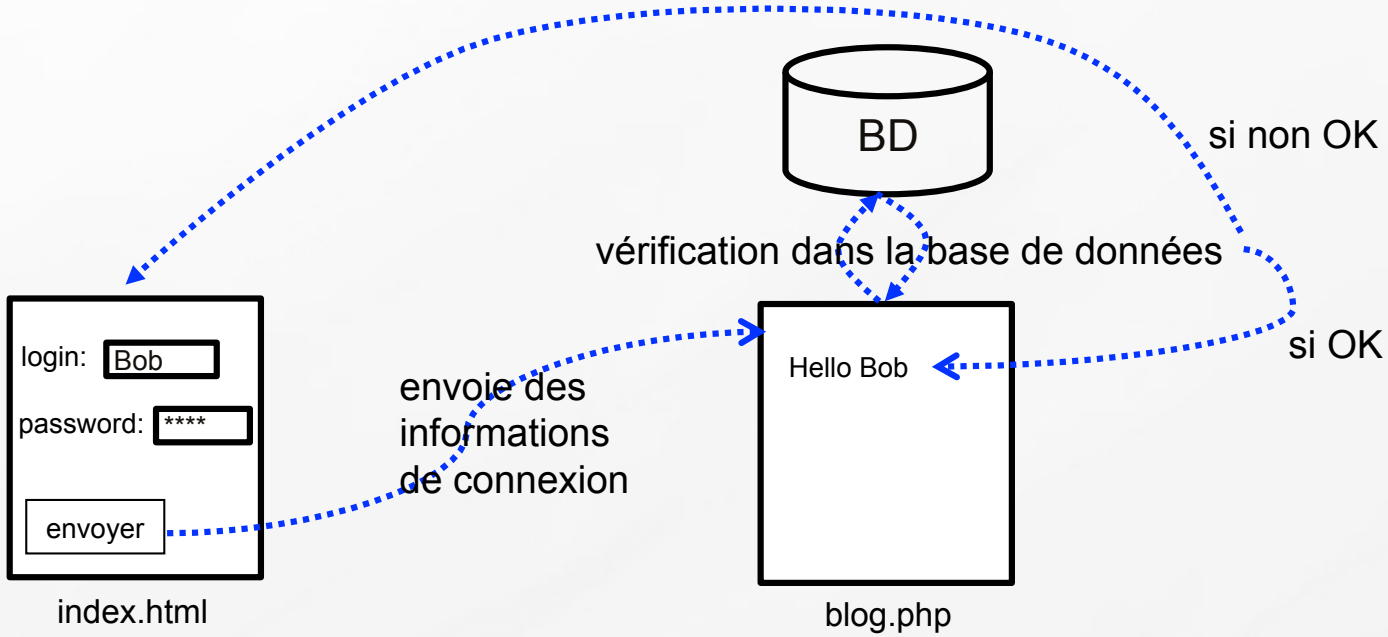
- Le nom du tableau initialisé dépend de la méthode:
  - **$_POST['**…**']** pour la méthode ***post***
  - **$_GET['**…**']** pour la méthode ***get***

données transmises
en mode post

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Form Hello PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

    <p> Hello <?php echo $_POST['login']; ?> </p>

 </body>
</html>
```

Bases de données et PHP: un premier exemple

- Ex.: accès à une bases de données pour vérifier un login et un mot de passe



Bases de données et PHP: un premier exemple

```php
<?php
   $link = mysqli_connect('localhost', 'root', '', 'blog_db');
   $query= 'SELECT login FROM Users WHERE login="'.$_POST['login'].'" and password="'.$_POST['password'].'"';
   $resultlogin = mysqli_query($link, $query );

    if( ! mysqli_num_rows( $resultlogin) ){
                mysqli_free_result( $resultlogin );
                header( "refresh:5;url=index.html" );
                echo 'Erreur de login et/ou de mot de passe (redirection automatique dans 5 sec.)';
                exit;
   }
?>

<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Form Hello PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

    <p> Hello <?php echo $_POST['login']; ?> </p>

 </body>
</html>

<?php  mysqli_close( $link );  ?>
```

1. Connexion à la base de données
2. Envoi d'une requête SQL
3. Récupération du nombre de résultats
   - Si ! OK alors
     - libération de la mémoire occupée par les résultats
     - redirection vers la page de connexion et affichage d'un message d'erreur
   - Sinon
     - affichage du reste de la page
4. Fermeture de le connexion

blog.php

Bases de données et PHP: les étapes

1) Connexion au SGBD et sélection de la base de données
   - fonction **mysqli_connect(** *$server, $user, $pwd, $database* **)** retourne la connexion à la base de données

```
$link = mysqli_connect('localhost', 'root', '', 'blog_db');
```

2) Envoi d'une requête SQL
   - fonction **mysqli_query(** *$link, $query* **)** retourne le résultat de la requête dans une structure de données **mysqli_result** (si requêtre SELECT) ou true/false (si requête INSERT, UPDATE, DELETE)
     - utiliser **mysqli_error**($link) pour récupérer les messages d'erreurs de MySQL

```
$resultall = mysqli_query($link, 'SELECT id, title FROM Post');
```

   - attention: pas de point-virgule à la fin de la requête

Bases de données et PHP: les étapes

3) Récupération et utilisation du résultat (un tuple à la fois)

- fonction **mysqli_fetch_assoc(** *$result* **)** retourne un tableau associatif qui correspond à la ligne récupérée (ligne en cours) ou NULL s'il n'y a plus de ligne

➢ nécessité d'utiliser une boucle pour parcourir s'il y a plusieurs tuples dans le résultat

```
<ul>
  <?php while ($row = mysqli_fetch_assoc($resultall) ): ?>
      <li>
            <?php echo $row['title']; ?>
      </li>
  <?php endwhile ?>
</ul>
```

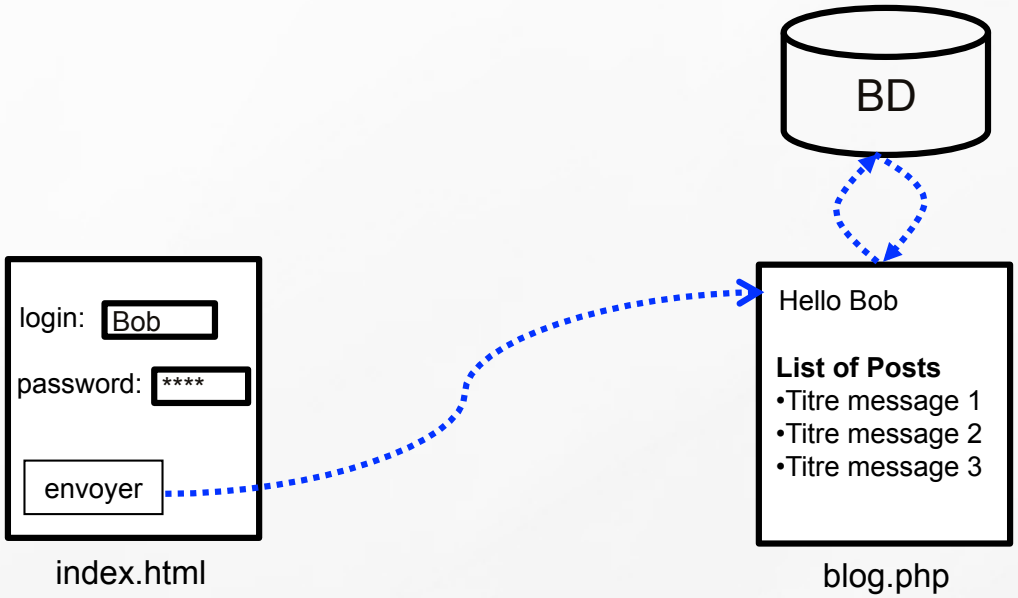
4) Fermeture de la connexion

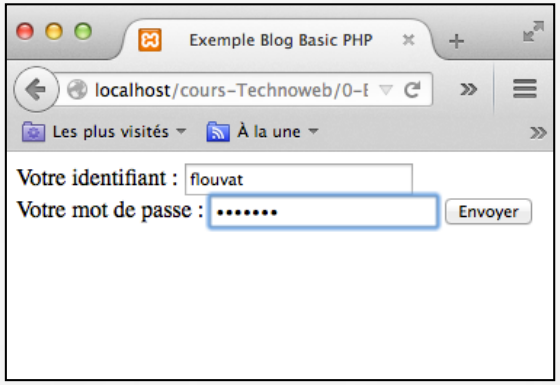
- fonction **mysqli_close**(*$link*) retourne true si la connexion a été fermée
  - automatique en fin de page

```
mysqli_close( $link );
```

Bases de données et PHP: exemple Blog

- Ex.: afficher la liste des messages d'un blog





Bases de données et PHP: exemple Blog

```
<?php
   $link = mysqli_connect('localhost', 'root', '', 'blog_db');
   $query= 'SELECT login FROM Users WHERE login="'.$_POST['login'].'" and password="'.$_POST['password'].'"';
   $resultlogin = mysqli_query($link, $query );

   if( mysqli_num_rows( $resultlogin) ){
      mysqli_free_result( $resultlogin );
      $resultall = mysqli_query($link, 'SELECT id, title FROM Post');
   }
   else{
      header( "refresh:5;url=index.html" );
      echo 'Erreur de login et/ou de mot de passe (redirection automatique dans 5 sec.)';
      exit;
   }
?>

<!DOCTYPE html>
<html lang="fr">
 <head>
…
 </head>
 <body>
   <p> Hello <?php echo $_POST['login']; ?> </p>
   <h1>List of Posts</h1>
   <ul>
      <?php while ($row = mysqli_fetch_assoc($resultall)): ?>
         <li><?php echo $row['title']; ?> </li>
      <?php endwhile ?>
   </ul>
 </body>
</html>

<?php    mysqli_close( $link );?>
```

Modifier blog.php

Transmettre des données d'une page à l'autre via un lien hypertexte

- Transmettre des variables et leurs valeurs directement dans un lien hypertexte
  - *page "origine":* créer un simple lien hypertexte et ajouter les variables à la fin de l'url
    - équivalent à une transmission de données via un formulaire en **get**

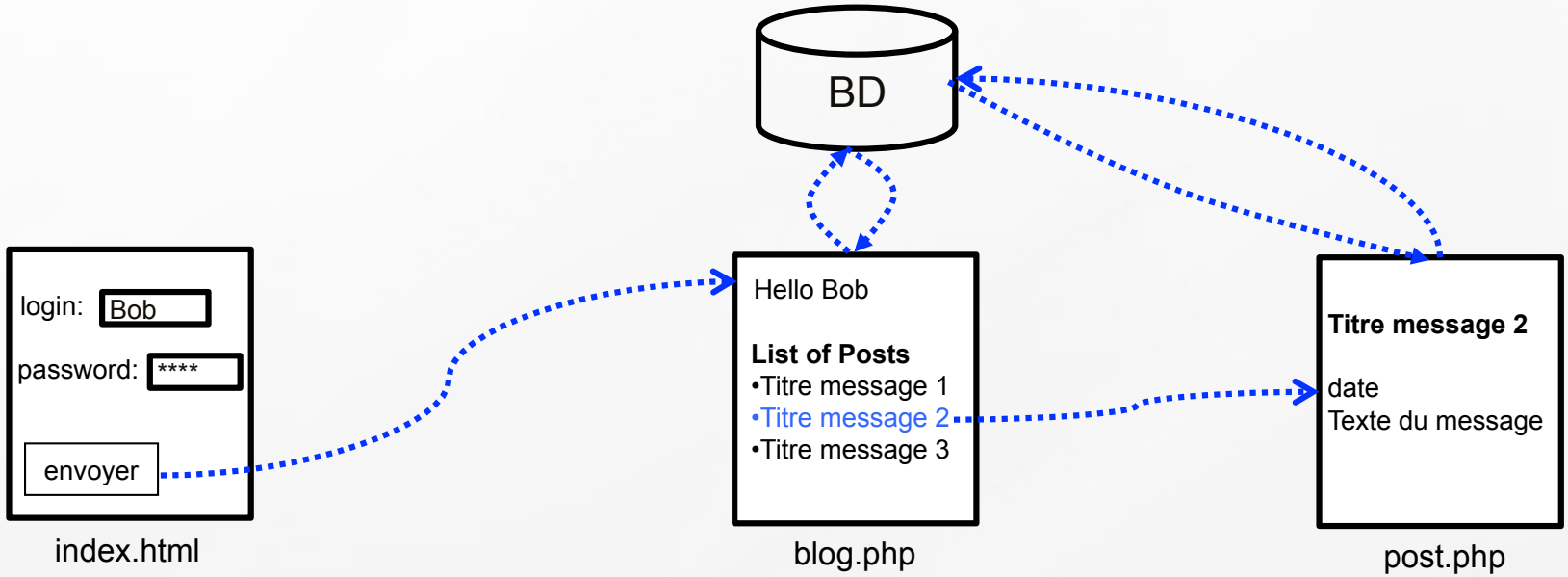
```
<a href="article.php?nom=John&prenom=Do">lien</a>
```

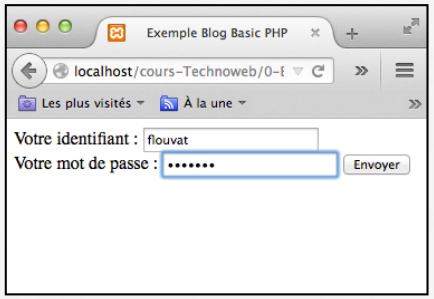
  - *page "destination":* utiliser le tableau $_GET[...] pour la valeur en PHP

```
<?php
   echo $_GET['nom'];
   echo $_GET['prenom'];
?>
```

Transmettre des données d'une page à l'autre via un lien hypertexte

- Ex.: extension du blog
  - lorsque l'on clique sur le titre d'un message ("*post*"), on est redirigé vers une page qui affiche le texte associé





Transmettre des données d'une page à l'autre via un lien hypertexte

Modifier blog.php

```
...
 <h1>List of Posts</h1>
   <ul>
      <?php while ($row = mysqli_fetch_assoc($resultall)): ?>
          <li>
                <a href="post.php?id=<?php echo $row['id']; ?>">
                    <?php echo $row['title']; ?>
                </a>
          </li>
      <?php endwhile ?>
   </ul>
...
```

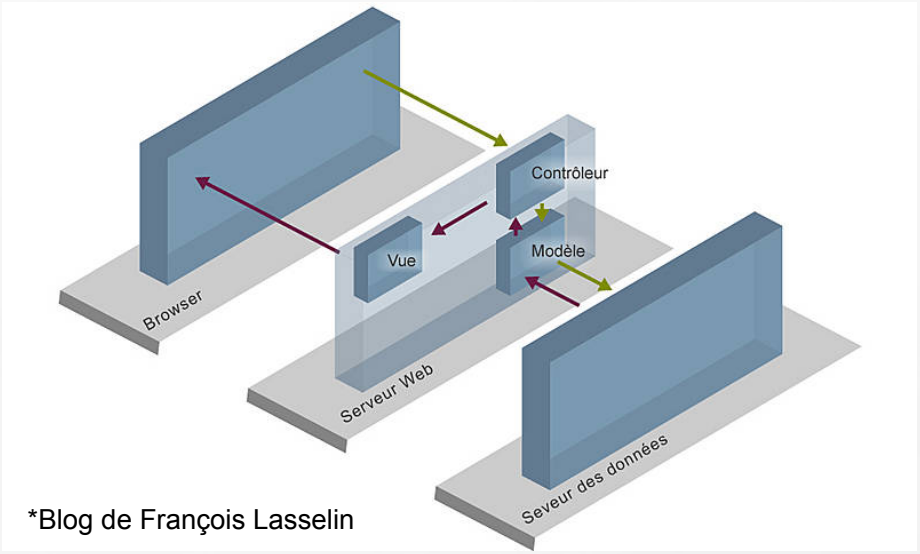
post.php

```
<?php
   $link = mysqli_connect('localhost', 'root', '', 'blog_db');
   $result = mysqli_query($link, 'SELECT * FROM Post WHERE id='.$_GET['id'] );
   $post = mysqli_fetch_assoc($result);
?>

<!DOCTYPE html>
<html lang="fr">
 <head>
   <title>Exemple Blog Basic PHP</title>
   <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

    <h1><?php echo $post['title']; ?></h1>
    <div class="date"> <?php echo $post['date']; ?> </div>
    <div class="body"> <?php echo $post['body']; ?> </div>

 </body>
</html>

<?php    mysqli_close( $link );?>
```

MVC et PHP: principe

- Le patron de conception (*design pattern*) Modèle-Vue-Contrôleur (MVC)
  - un modèle d'architecture logicielle = une solution générale permettant de résoudre un problème classique en développement logiciel
  - séparation de l'accès aux données (le modèle), de l'interface (la vue) et des traitements (le contrôleur)
  - ➢ simplifie la maintenance et l'évolution du logiciel

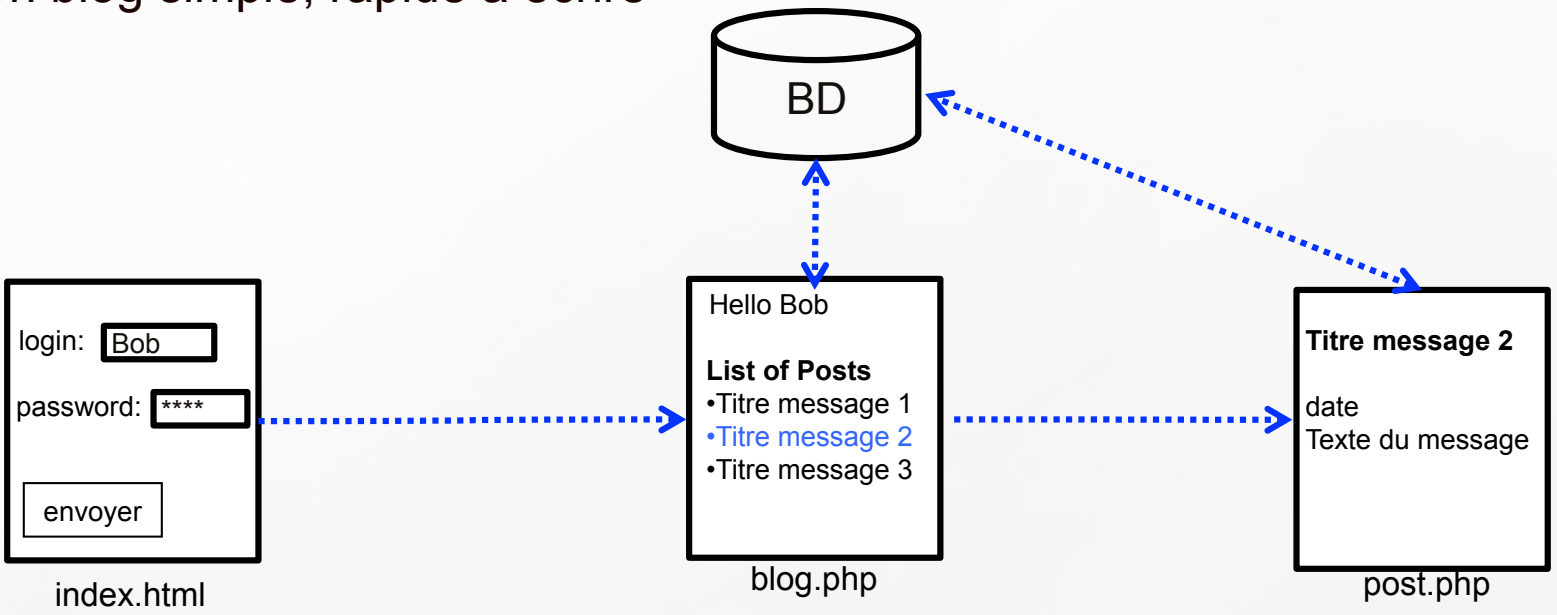
- Mise en œuvre un peu particulière dans le cas des pages web car c'est le navigateur qui gère l'affichage et non l'application



*Blog de François Lasselin

MVC et PHP: exemple du blog *

- Un blog simple, rapide à écrire



  - tout est fortement couplé/lié
  - ➢ difficile à faire évoluer

- ➢ Comment le rendre plus modulaire et évolutif ?
  - ➢ Appliquer l'architecture MVC

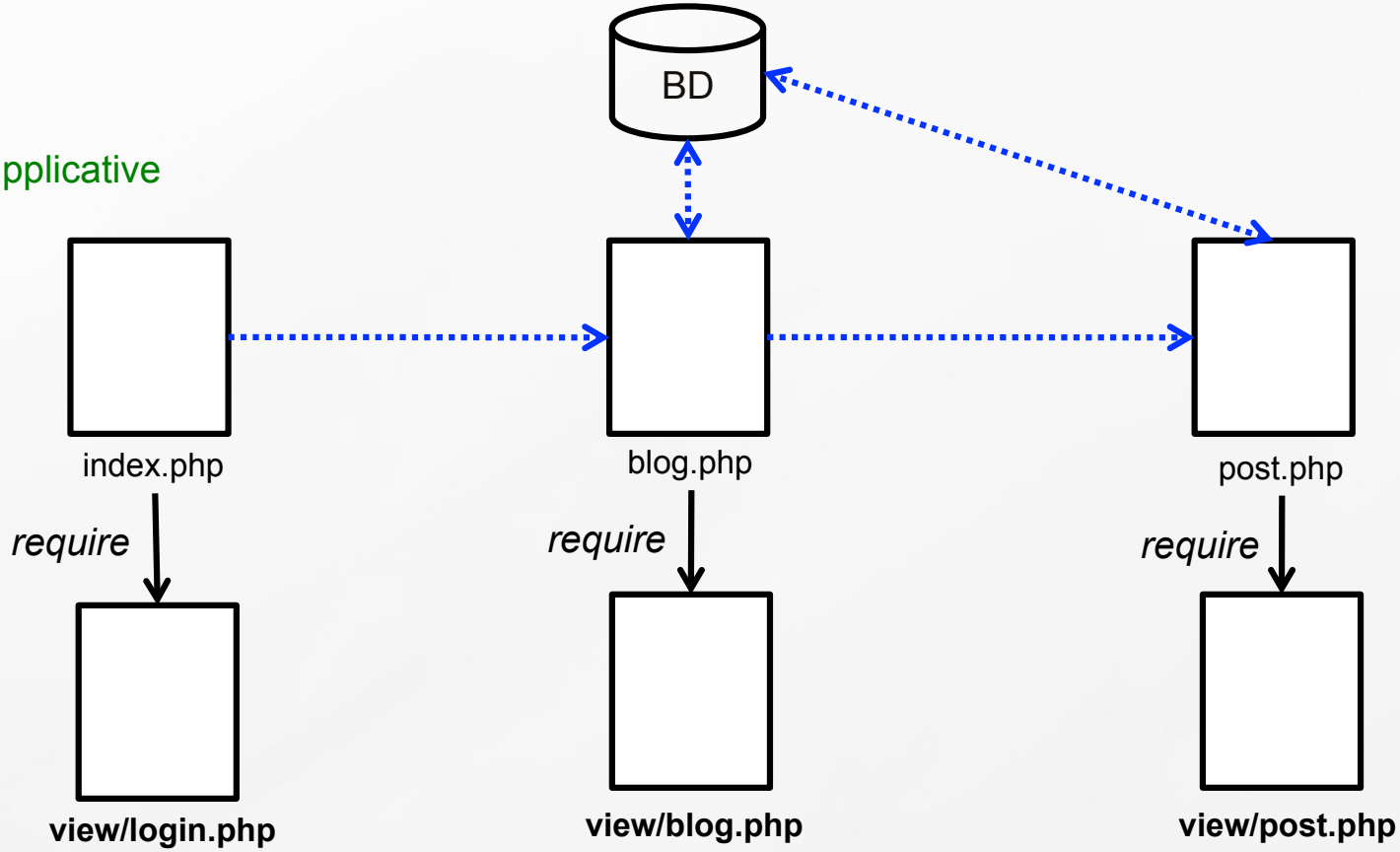
*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

- Etape 1: Isoler la présentation / l'interface

Fichiers chargés de la logique applicative
(appelée aussi **contrôleurs**)
• traite les entrées utilisateurs
• prépare une réponse

Fichiers chargés de la présentation HTML



*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

inclut le fichier chargé de l'affichage du login

```php
<?php
   require 'view/login.php';
?>
```

index.php

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Blog Basic PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>
  <form method="post" action="blog.php">
    <label for="login"> Votre identifiant </label> :
    <input type="text" name="login" id="login" placeholder="defaut" maxlength="12" required />
    <br />
    <label for="password"> Votre mot de passe </label> :
    <input type="password" name="password" id="password" maxlength="12" required />

    <input type="submit" value="Envoyer">
  </form>
 </body>
</html>
```

view/login.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

```php
<?php
   $link = mysqli_connect('localhost', 'root', '', 'blog_db');
   $query= 'SELECT login FROM Users WHERE login="'.
$_POST['login'].'" and password="'.$_POST['password'].'"';

   $resultlogin = mysqli_query($link, $query );

   if( mysqli_num_rows( $resultlogin) ){
      $login = $_POST['login'];
      mysqli_free_result( $resultlogin );
      $resultall = mysqli_query($link, 'SELECT id, title FROM Post');
      $posts = array();

      while ($row = mysqli_fetch_assoc($resultall)) {
         $posts[] = $row;
      }
   }
   mysqli_close( $link );

   // inclut le code de la présentation HTML
   require 'view/blog.php';
?>
```

blog.php

récupère les données et les stocke dans des variables utilisées dans la présentation

```php
<?php
   if( !isset( $login) ){
      header( "refresh:5;url=index.php" );
      echo 'Erreur de login et/ou de mot de passe (redirection
automatique dans 5 sec.)';
      exit;
   }
?>

<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Blog Basic PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>
   <p> Hello <?php echo $login; ?> </p>
   <h1>List of Posts</h1>
   <ul>
      <?php foreach( $posts as $post ) : ?>
       <li>
         <a href="post.php?id=<?php echo $post['id']; ?>">
           <?php echo $post['title']; ?>
         </a>
       </li>
      <?php endforeach ?>
   </ul>
 </body>
</html>
```

view/blog.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

```php
<?php
   $link = mysqli_connect('localhost', 'root', '', 'blog_db');
   $result = mysqli_query($link, 'SELECT * FROM Post WHERE id='.$_GET['id'] );

   $post = mysqli_fetch_assoc($result);
   mysqli_close($link);

    // inclut le code de la présentation HTML
   require 'view/post.php';
?>
```

post.php

```php
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title>Exemple Blog Basic PHP</title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

    <h1><?php echo $post['title']; ?></h1>
    <div class="date"> <?php echo $post['date']; ?> </div>
    <div class="body"> <?php echo $post['body']; ?> </div>

 </body>
</html>
```

view/post.php

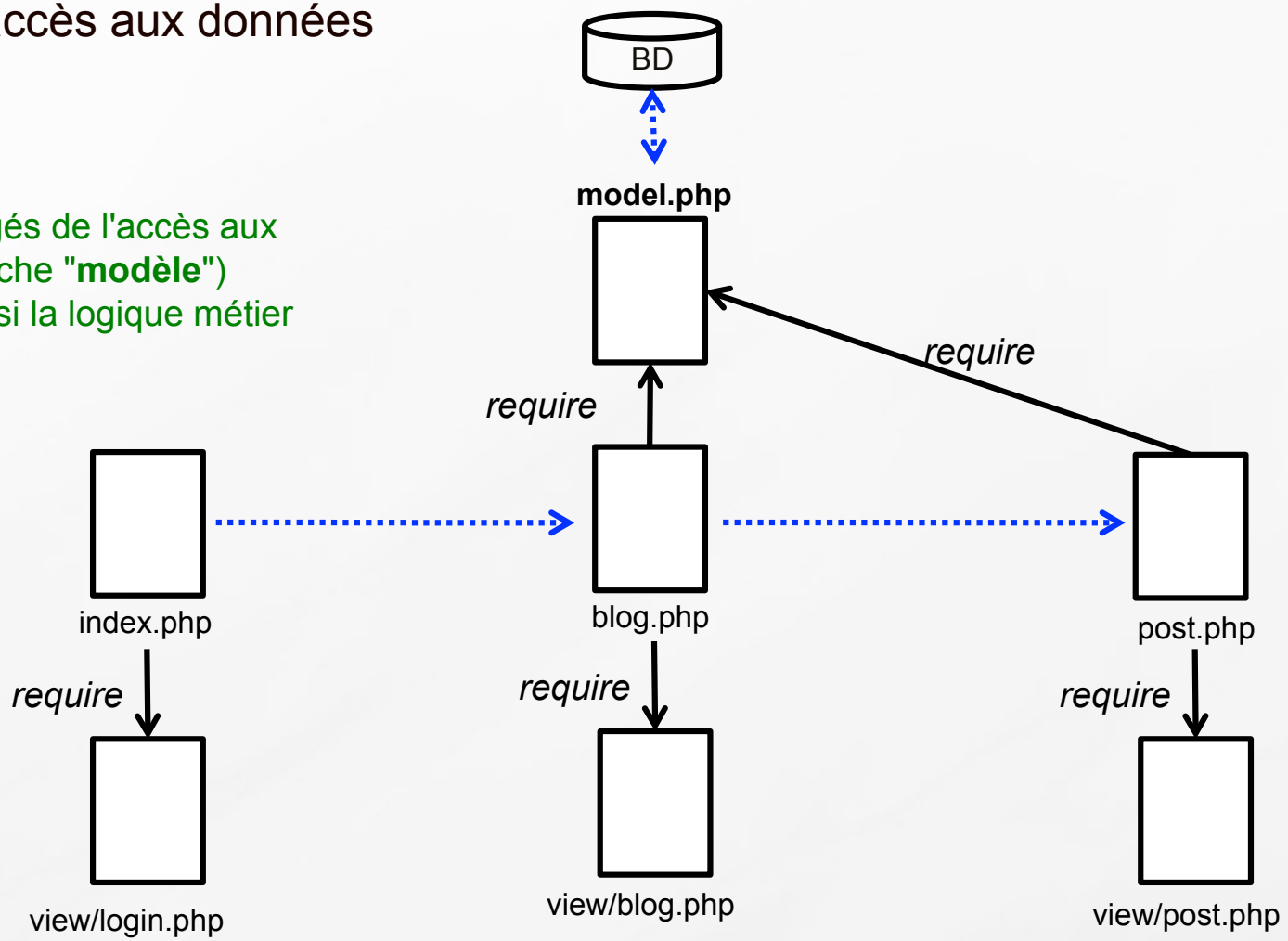
*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

- Etape 2: Isoler l'accès aux données

Fichiers chargés de l'accès aux données (couche "**modèle**")
• contient aussi la logique métier

➔ rôle des contrôleurs:
• récupérer les données de la couche modèle
• appeler la vue permettant d'afficher les données



*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

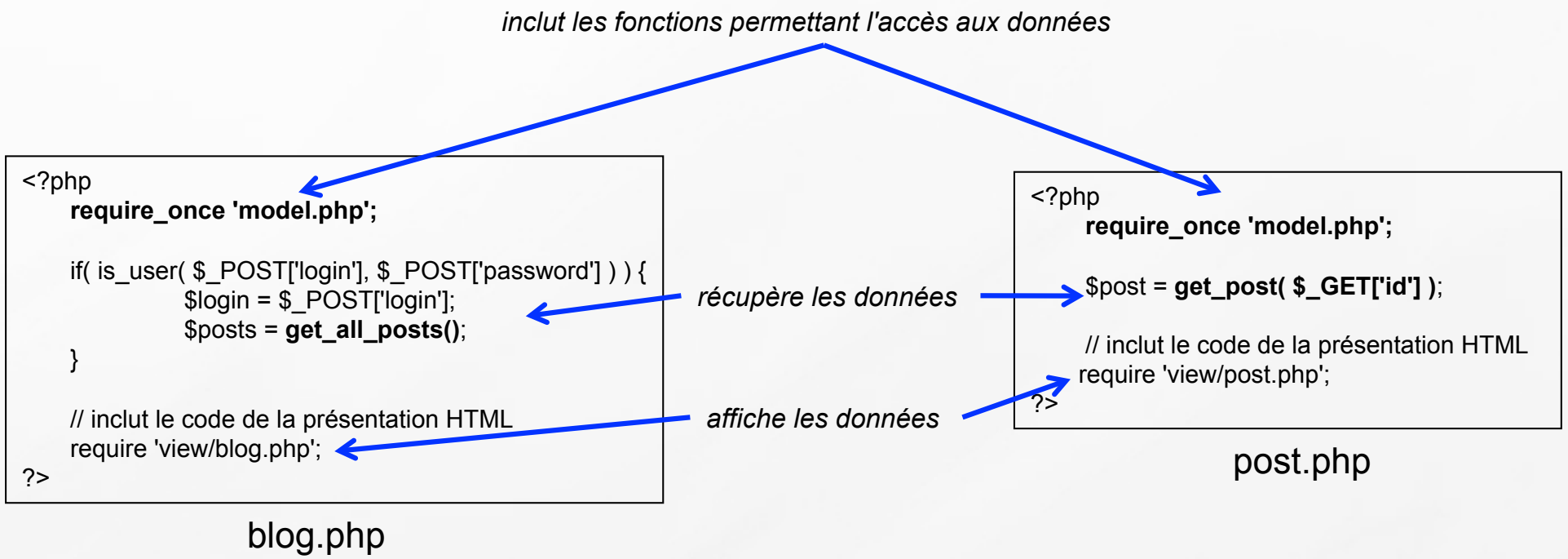
```php
<?php

   function open_database_connection()
   {
      $link = mysqli_connect('localhost', 'root', '', 'blog_db');
      return $link;
   }

   function close_database_connection($link)
   {
      mysqli_close($link);
   }

   function is_user( $login, $password )
   {
      $isuser = False ;
      $link = open_database_connection();

      $query= 'SELECT login FROM Users WHERE
login='".$login."' and password='".$password."'";
      $result = mysqli_query($link, $query );

      if( mysqli_num_rows( $result) )
         $isuser = True;

      mysqli_free_result( $result );
      close_database_connection($link);

     return $isuser;
   }
```

```php
function get_all_posts()
   {
      $link = open_database_connection();

      $resultall = mysqli_query($link,'SELECT id, title FROM Post');
      $posts = array();

      while ($row = mysqli_fetch_assoc($resultall)) {
         $posts[] = $row;
      }

      mysqli_free_result( $resultall);
      close_database_connection($link);

      return $posts;
   }

   function get_post( $id )
   {
      $link = open_database_connection();

      $id = intval($id);
      $result = mysqli_query($link, 'SELECT * FROM Post WHERE
id='.$id );
      $post = mysqli_fetch_assoc($result);

      mysqli_free_result( $result);
      close_database_connection($link);
      return $post;
   }
?>
```

model.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

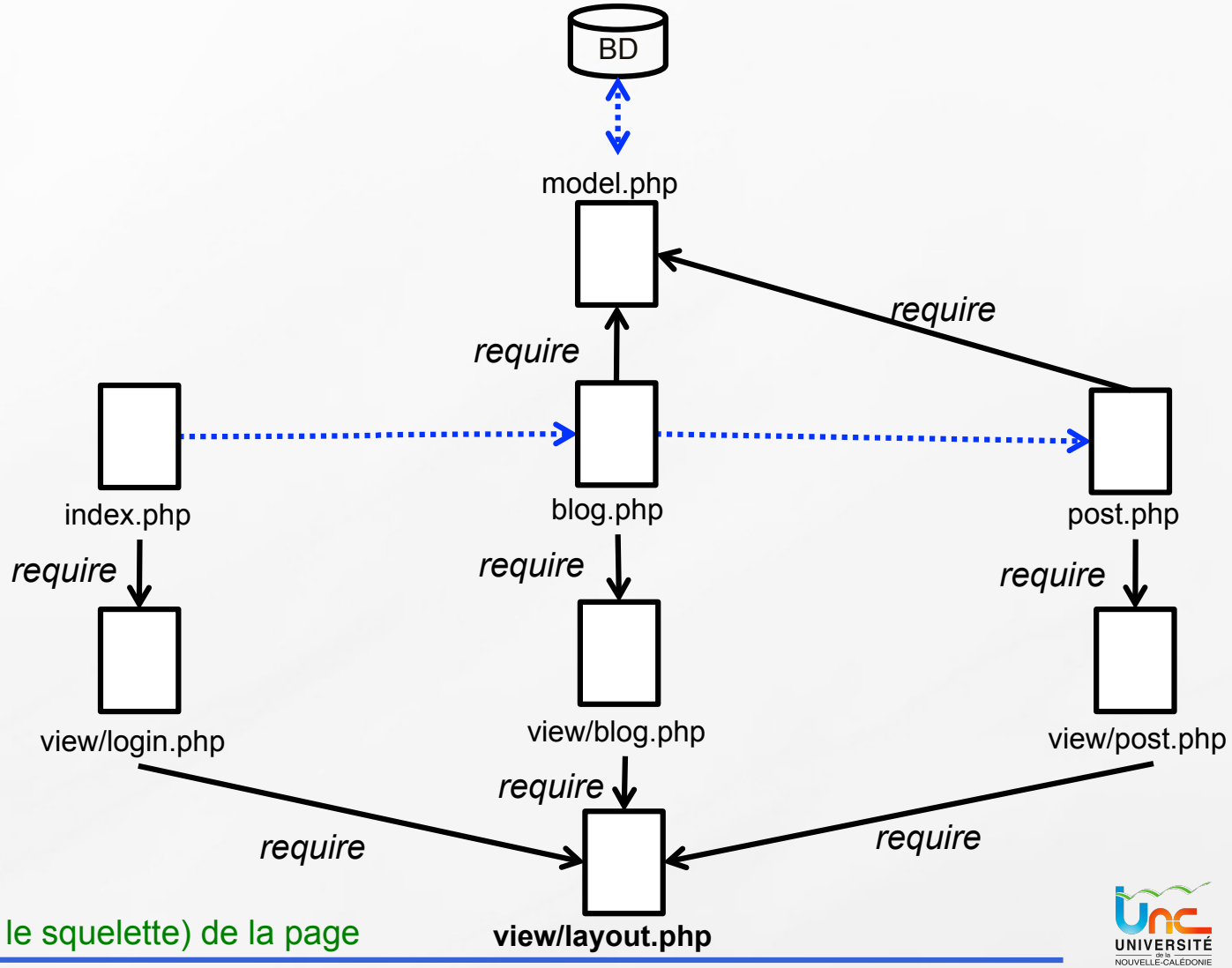
MVC et PHP: exemple du blog*



*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

- Etape 3: Isoler la mise en page de base de la page (le *layout*)



Fichier factorisant le layout (ie le squelette) de la page

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title><?php echo $title; ?></title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>
    <?php echo $content; ?>
 </body>
</html>
```

view/layout.php

squelette de la page :
- *un titre*
- *un contenu*

```
<?php $title= 'Exemple Blog Basic PHP: Connexion'; ?>

<?php ob_start(); ?>
   <form method="post" action="blog.php">
      <label for="login"> Votre identifiant </label> :
      <input type="text" name="login" id="login" placeholder="defaut" maxlength="12" required />
      <br />
      <label for="password"> Votre mot de passe </label> :
      <input type="password" name="password" id="password" maxlength="12" required />

      <input type="submit" value="Envoyer">
   </form>
<?php $content = ob_get_clean(); ?>

<?php include 'layout.php'; ?>
```

initialisation du titre utilisé dans le squelette

initialisation du contenu (le formulaire de login) et stockage en mémoire

affichage de la page

view/login.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

```
<?php
   if( !isset( $login) ){
      header( "refresh:5;url=index.php" );
      echo 'Erreur de login et/ou de mot de passe (redirection
automatique dans 5 sec.)';
      exit;
   }
?>

<?php $title= 'Exemple Blog Basic PHP: Blog'; ?>

<?php ob_start(); ?>
   <p> Hello <?php echo $login; ?> </p>
   <h1>List of Posts</h1>
   <ul>
      <?php foreach( $posts as $post ) : ?>
        <li>
          <a href="post.php?id=<?php echo $post['id']; ?>">
            <?php echo $post['title']; ?>
          </a>
        </li>
      <?php endforeach ?>
   </ul>
<?php $content = ob_get_clean(); ?>

<?php include 'layout.php'; ?>
```

view/blog.php

1. *initialisation du titre*
2. *initialisation du contenu*
3. *affichage de la page*

```
<?php $title= 'Exemple Blog Basic PHP: Post'; ?>

<?php ob_start(); ?>
   <h1><?php echo $post['title']; ?></h1>
   <div class="date"> <?php echo $post['date']; ?> </div>
   <div class="body"> <?php echo $post['body']; ?> </div>
<?php $content = ob_get_clean(); ?>

<?php include 'layout.php'; ?>
```

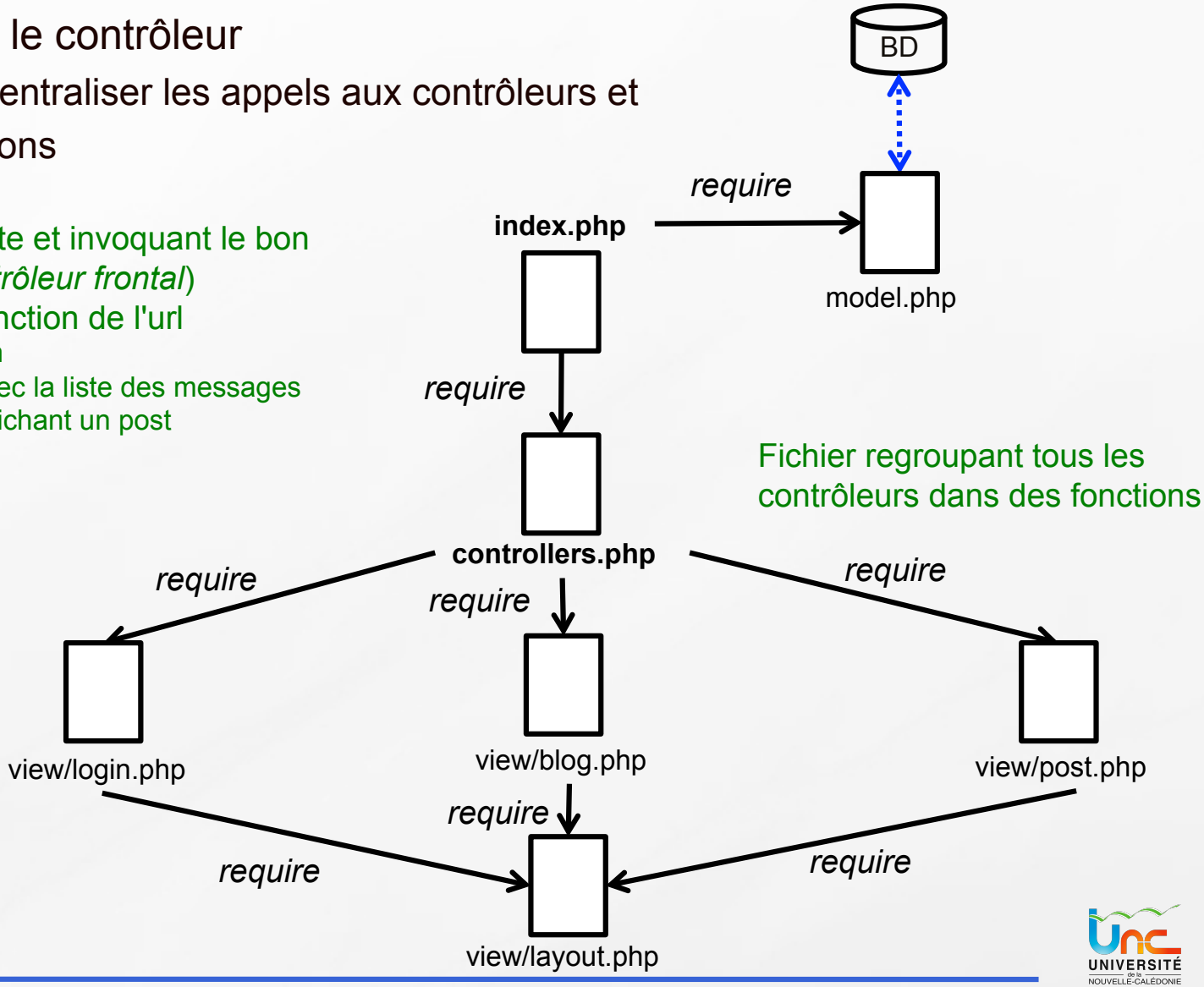
view/post.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

- Etape 4: Isoler le contrôleur
  - permet de centraliser les appels aux contrôleurs et les vérifications

Fichier traitant chaque requête et invoquant le bon contrôleur (appelé aussi *contrôleur frontal*)
• routage des requêtes en fonction de l'url
/index.php -> page de login
/index.php/blog -> page avec la liste des messages
/index.php/post -> page affichant un post



Fichier regroupant tous les contrôleurs dans des fonctions

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

MVC et PHP: exemple du blog*

```php
<?php

 function login_action()
 {
      require 'view/login.php';
 }

 function blog_action( $login, $password)
 {
      if( is_user( $login, $password ) )
         $posts = get_all_posts();
      else
         $login='';

     require 'view/blog.php';
 }

 function post_action($id)
 {
      $post = get_post($id);
      require 'view/post.php';
 }

?>
```

controllers.php

```php
<?php

  // charge et initialise les bibliothèques globales
  require_once 'model.php';
  require_once 'controllers.php';

  // route la requête en interne
  $uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

  if ('/myBlog/index.php' == $uri) {
     login_action();
  }
  elseif ( '/myBlog/index.php/blog' == $uri
           && isset($_POST['login']) && isset($_POST['password']) ){

     blog_action($_POST['login'], $_POST['password']);
  }
  elseif ('/myBlog/index.php/post' == $uri
           && isset($_GET['id'])) {

     post_action($_GET['id']);
  }
 else {
    header('Status: 404 Not Found');
    echo '<html><body><h1>My Page Not Found</h1></body></html>';
  }
?>
```

index.php

extraction du chemin de l'url

Si index.php -> contrôleur login

Si index.php/blog et login+password transmis -> contrôleur blog

Si index.php/post et id post transmis -> contrôleur post

Sinon affichage erreur 404

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

Sessions PHP et cookies

- Il peut être utile de conserver des informations d'une page sur l'autre lors de la visite d'un utilisateur
  - p.ex. pour se souvenir du login de l'utilisateur connecté, de panier d'achat en cours d'un utilisateur, des dernières pages visitées, etc

- Jusqu'ici, seuls moyens:
  - utiliser des paramètres et les transmettre pages après pages via des liens de navigation ou des formulaires (voire des formulaires cachés)
  - passer par une base de données
  - ➢ mécanismes relativement "lourds" d'un point de vue programmation

- ➢ Deux autres alternatives
  - les sessions
  - les cookies

Les sessions

- Permet de conserver des informations entre les différentes pages
  - durée de conservation par défaut: 24 min. (modifiable dans le fichier php.ini)

- En PHP, la session est une variable spéciale de type tableau associatif appelée **$_SESSION**

- Utilisation:
  - commencer chaque page par l'instruction **session_start()** (*avant génération du code HTML*)
  - utiliser la variable **$_SESSION** comme un tableau associatif classique

```
<?php
  session_start();
  …
  $_SESSION['login']='root';
  …
?>
```

  - lorsque l'utilisateur "se déconnecte", il est important de détruire la session
    - p.ex. pour éviter qu'une seconde personne utilisant le même ordinateur ne se fasse passer pour la première personne

```
<?php
  …
  session_destroy();
  …
?>
```

Les sessions: un exemple simple

- Sans session

```
…
<a href="msg.php?id=1&login=root">
                titre  message 1
</a>
…
```

listeMail.php

```
…
<?php
  echo $_GET['login'];
?>
…
<a href="msg.php?id=1&login=$_GET['login']">
                next page
</a>
 …
```

msg.php

- Avec session

```
<?php
 session_start();
  …
  $_SESSION['login']='root';
  …
?>

…
<a href="msg.php?id=1"> titre  message 1 </a>
...
```

listeMail.php

```
<?php
 session_start();
  …
  echo $_SESSION['login'];
  …
?>

…
<a href="msg.php?id=1"> next page </a>
...
```

msg.php

Les sessions: exemple du blog

- Vérifier l'authentification sur toutes les pages du site pour éviter les accès non connectés
  - supprimer les tests dans le contrôleur de la partie *blog* et mettre en place les tests dans le contrôleur frontal
    - utilisation de la fonction is_user
    - **vérification et enregistrement de session**
    - création d'une variable pour stocker le type d'erreur et passage en paramètre des contrôleurs
  - supprimer l'affichage du message d'erreur (et de la redirection) dans la vue de la partie "blog" et gérer l'affichage des erreurs dans la vue *layout*
- Généraliser l'affichage d'un message d'accueil "*Connecté en tant que …"* à toutes les pages du site
  - modifier les contrôleurs en leur passant en paramètre le login de l'utilisateur
  - supprimer l'affichage du message d'accueil dans la vue de la partie "blog" et gérer son affichage dans la vue "layout"
- Ajouter la fonctionnalité "Déconnexion" pour fermer une session utilisateur
  - ajouter dans le contrôleur frontal le cas "Déconnexion" via l'url "*/index.php/logout"*
  - ajouter dans la vue "layout" l'affichage d'un lien "Déconnexion" si la personne est connectée

Les sessions: exemple du blog

```php
<?php
    // charge et initialise les bibliothèques globales
    require_once 'model.php';
    require_once 'controllers.php';

    // démarrage de la session
    session_start();

    // récupération du nom de la page demandée
    $uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

    // vérification utilisateur authentifié
    if( !isset($_SESSION['login']) ) {

      if( !isset($_POST['login']) || !isset($_POST['password']) ) {
          $error='not connected';
          $uri='/myBlog/index.php' ;
       }
       elseif( !is_user($_POST['login'],$_POST['password']) ){
          $error='bad login/pwd';
          $uri='/myBlog/index.php' ;
       }
       else {
          $_SESSION['login'] = $_POST['login'] ;
          $login = $_SESSION['login'];
       }
    }
    else
       $login = $_SESSION['login'] ;
```

```php
    // route la requête en interne
    if ('/myBlog/index.php' == $uri ) {
       login_action($login,$error);
    }
    elseif ( '/myBlog/index.php/blog' == $uri ){
       blog_action($login,$error);
    }
    elseif ('/myBlog/index.php/post' == $uri && isset($_GET['id'])) {
       post_action($_GET['id'],$login,$error);
    }
    elseif('/myBlog/index.php/logout' == $uri ) {
       // fermeture de la session
       session_destroy();
       // affichage de la page de connexion
       login_action("","");
    }
    else {
       header('Status: 404 Not Found');
       echo '<html><body><h1>My Page Not Found</h1></body>
</html>';
    }

?>
```

index.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

Les sessions: exemple du blog

```php
<?php
   function login_action($login,$error)
   {
      require 'view/login.php';
   }

   function blog_action($login,$error)
   {
      $posts = get_all_posts();
      require 'view/blog.php';
   }

   function post_action($id,$login,$error)
   {
      $post = get_post($id);
      require 'view/post.php';
   }
?>
```

controllers.php

```php
<?php $title= 'Exemple Blog Basic PHP: Blog'; ?>

<?php ob_start(); ?>
   <p> Hello <?php echo $login; ?> </p>
   <h1>List of Posts</h1>
   <ul>
      <?php foreach( $posts as $post ) : ?>
       <li>
         <a href="post.php?id=<?php echo $post['id']; ?>">
          <?php echo $post['title']; ?>
         </a>
       </li>
      <?php endforeach ?>
   </ul>
<?php $content = ob_get_clean(); ?>

<?php include 'layout.php'; ?>
```

view/blog.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

Les sessions: exemple du blog

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <title><?php echo $title; ?></title>
  <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
 </head>
 <body>

   <p>
   <?php
      if( isset($error) ){
         switch( $error ) {
            case 'not connected':
               echo 'Veuillez svp vous authentifier';
               break;
            case 'bad login/pwd':
               echo 'Erreur de login et/ou de mot de passe';
               break;
         }
      }
      elseif( isset($login) )    {
         echo 'Connecté en tant que '.$login ;
         echo ' <a href="/myBlog/index.php/logout">Déconnexion</a>';
      }
   ?>
   </p>

   <?php echo $content; ?>
 </body>
</html>
```

view/layout.php

*cf Symphony2 versus PHP pur http://symfony.com/fr/doc/current/book/from_flat_php_to_symfony2.html

Les cookies

- Permet de conserver des informations entre les différentes pages sur de longues périodes et entre plusieurs visites
  - stocké dans un fichier sur la machine du client

- En PHP, les cookies sont une variable spéciale de type tableau associatif appelée **$_COOKIE**

- Utilisation:
  - utiliser la fonction

    setcookie($name,$value,$duration**)**
    (*avant toute génération de code HTML*)
  - utiliser la variable **$_COOKIE** comme un tableau associatif classique

  - pour détruire un cookie faire

    setcookie($name,' ', time() – 3600)

```
<?php
  setcookie('login_cookie', 'flouvat', time()+(86400*30))
   …
   echo $_COOKIE['login_cookie'];
   …
?>
```

30 jours

```
<?php
  …
  setcookie('login_cookie', '', time()-3600)
   …
?>
```

UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Cours de technologies Web
Introduction au JavaScript

Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

JavaScript n'est pas Java

- Langage différent du Java au niveau des concepts et de la syntaxe

JavaScript

- scripts intégrés dans le HTML
- interprété par le client
- faiblement typé

Java

- programmes autonomes fondés sur les concepts objets
- pseudo-compilé et exécuté du côté du serveur
- typage fort

- Exemples de fonctionnalités JavaScript
  - ajouter dynamiquement du texte dans une page Web
  - réagir à des événements
  - lire et écrire/créer des éléments HTML dynamiquement
  - vérifier, si nécessaire, les données envoyées au serveur
  - détecter le navigateur des visiteurs et modifier les paramètres
  - …

Insertion de code JavaScript dans un document HTML

- 3 modes d'insertion
  - dans la page entre les balises **<script> … </script>**
    - dans le <head> ou le <body>
    - ➢ inconvénient: mise à jour difficile si plusieurs pages

```
<!DOCTYPE html>
<html lang="fr">
 ...
  <script>
    alert("Hello");
  </script>
...
</html>
```

  - au niveau de certains attributs de balises (p.ex. **onclick**, **onmouseover**)
    - typiquement pour des attributs liés à des évènements
    - ➢ inconvénient: mise à jour de la page difficile

```
<img alt="image" src="images/image.gif" onclick="alert('Hello')"/>
```

  - **importer un fichier JavaScript externe via la balise <script src="...."></script>**

```
…
  <script  src="js/exemple.js"></script>
…
```

```
alert("Hello");
```

js/exemple.js

Syntaxe de base

- Un langage proche du PHP

- Les **similitudes** avec le PHP
  - langage sensible à la casse
  - typage faible des variables: déclaration et typage implicite à l'affectation, possibilité de changer de type ensuite
  - structures de contrôles similaires

```
for( var i =0 ; i < 10; i++ )
{
 ...
}
```

```
do{
 ...
}while( cond );
```

```
while( cond ){
 ...
}
```

```
if( cond )
{
 ...
}
else if( cond )
{
 ...
}
else
{
 ...
}
```

```
switch( expr )
{
 case VALEUR_1:
  ...
 break;
 case VALEUR_2:
  ...
 break;
 default:
  ...
 break;
}
```

  - les mêmes opérateurs: **+, =, ++, +=, ===,...**
  - commentaires: *// et /* */*

Syntaxe de base

- Les **différences** avec le PHP[1]
  - pas de **$** devant le nom des variables
  - portée des variables: par défaut globale (contraire du PHP), sauf si déclarée avec le mot-clé **var**

```
function foo(){
   var variableA = "value"; // Local variable with use of "var"
}
```

  - conversion de variables: utilisation de fonctions (p.ex. la fonction **parseInt**)
  - mot clé **true** en minuscule (uniquement)
  - pas de **foreach**
  - pas de tableaux associatifs
  - syntaxe légèrement différentes pour les tableaux indicés

```
var foo = []; // New empty array
var foo = ["a", "b", "c"]; // Numeric index

alert( foo[0] );
```

1. cf http://www.lullabot.com/articles/learning-javascript-php-comparison

Les fonctions en JavaScript

- Une implémentation différentes des autres langages
  - une fonction = une variable particulière (un objet) que l'on peut stocker dans un conteneur

```
var maFonction = function(a, b){      // a et b sont des paramètres
   var c = a + b;
   return c;          // valeur renvoyée (facultatif)
}

var d = maFonction(3,2) ;       // d vaut 5
```

  - possibilité de définir directement une fonction (mais toujours une variable)

```
maFonction (a, b){          // a et b sont des paramètres
...
}

var d = maFonction(3,2) ;      // d vaut 5
```

  - possibilité d'utiliser le tableau **arguments** pour accéder aux paramètres

```
var maFonction = function(){
    var c = arguments[0] + arguments[1];
     ...
}
```

Les objets en JavaScript

- Objet JavaScript = variable possédant des propriétés, et sur laquelle il est possible d'agir par l'intermédiaire de méthodes

```
var myObject = new Object();
myObject.name = "Andrew";
myObject.number = 42;
```

```
var myObject = {name: "Andrew", number: 42 } ;
myObject.tired = "Yes";
myObject.printName = function() { alert(this.name);
```

- Création des objets via une fonction

```
function dude (name, age) {
  this.name = name;
  this.age = age;
}

dude.setAge = function (x) { this.age = x; };

var guy = new dude("Andrew", 24);
guy.setAge(42);
```

Exemples d'objets prédéfinis

- L'objet **Array** (tableau)
  - création: **new Array() ;** , **new Array(*size*);** , ou **new Array(*val1*, *val2*, *val3*, ...);**
  - accès et modification avec **[ ]**
  - taille du tableau: propriété **.length**
  - quelques méthodes: **reverse()** (inverse les éléments), **concat(*val1*, *val2*, ...)** (ajoute des éléments la fin du tableau), **sort()** (trie le tableau), **push(*val*)** / **pop()** (ajoute/ supprime un élément en fin), ...

- L'objet **String** (chaîne de caractères)
  - taille de la chaîne de caractères: propriété **.length**
  - quelques méthodes: **charAt(*i*)** (i-ème caractère), **substring(*posDeb*, *posFin*)** (extrait la sous-chaîne entre *posDeb* et *posFin* non compris), **toUpperCase** (convertit en majuscules), **replace(*str1*, *str2*)** (remplace la sous chaîne *str1* par *str2*), **match(*regExp*)** (sélectionne les parties de la chaîne correspondant à l'expression régulière *regExp*), ...
  - concaténation: opérateur **+**

Exemples d'objets prédéfinis

- L'objet **Math** (calculs mathématiques)
  - quelques propriétés (constantes prédéfinies): **E**, **PI**, **SQRT2**, **LN2**, **LOG2E**
  - quelques méthodes: **sqrt(*nb*)** (racine carré), **random()** (nombre réel choisi au hasard entre [0,1[ ), **floor(*nb*)** (entier immédiatement inférieur), **sin(*nb*)** (sinus)

- L'objet **Date** (dates et heures)
  - création: **new Date()** (date et heure courante), **new Date(*année, mois, jour [heure, min, sec]*)** (initialiser la date et l'heure)
  - récupérer la date et/ou l'heure: méthodes **getDate()**, **getYear()**, **getMonth()**, **getHours()**, **getMinutes()**, **getSeconds()**
    - **Attention**: les secondes et les minutes sont notées de 0 à 59, les jours de la semaine de 0 (dimanche) à 6, les jours du mois de 1 à 31, les mois de 0 (janvier) à 11, et les années sont décomptées depuis 1900
  - de la même manière, **set...(*val*)** pour modifier une date/heure
  - comparaison avec les opérateurs de comparaisons classiques: **==**, **!=**, **<**, **>**, **<=**, **>=**
  - temps écoulé entre deux dates: utilisation de la méthode **getTime()** retournant le nombre de millisecondes écoulées depuis le 1/1/1970

JavaScript et HTML

- Principe:
  - séparation le fond (HTML) et le comportement (JavaScript)

- Utilisation classique:
  - sélectionner un élément HTML
    - p.ex. un *div* précis, tous les éléments d'une classe CSS, tous les liens, …
  - associer un comportement (une fonction JavaScript) à un événement sur cet élément
    - p.ex. au chargement de la page, sur un clic bouton, sur la passage de la souris, …
  - dans cette fonction,
    - sélectionner d'autres éléments HTML ou en créer de nouveaux,
    - changer leur style (p.ex. taille, visibilité, position, …)

Exemple

```
<!DOCTYPE html>
<html lang="fr">
 <head>
    <title>Premier exemple JavaScript</title>
    <meta http-equiv="Content-Type" content="text/
html;charset=utf-8" />

    <script  src="js/slideshow.js"></script>
 </head>
 <body>
    <img src="images/img1.jpg" alt="unc" id="slide" />
 </body>
</html>
```

PremierExempleJS.html

1. Sélectionner la page
2. Associer l'exécution de la fonction ***swapImg*** à l'événement ***onload*** du document (fin de chargement de la page)

```
listImg = Array();
i=0;

listImg[0]='images/img1.jpg';
listImg[1]='images/img2.jpg';
listImg[2]='images/img3.jpg';
listImg[3]='images/img4.jpg';

function swapImg(){
    var imageSlide=document.getElementById('slide');
    if( i < listImg.length -1 ) i++
    else i=0 ;
    imageSlide.src = listImg[i];
    setTimeout( swapImg, 2000);
}

window.onload=swapImg;
```

variables globales

1+2

js/slideshow.js

Sélectionner des éléments

- Méthodes pour accéder aux éléments d'une page (**document**):
  - **document.getElementById(*value*)** : retourne l'élément dont l'**id** vaut *value*
  - **document.getElementsByTagName(*value*)** : retourne un tableau contenant les éléments dont le ne nom de la balise est *value*

```
var images = document.getElementsByTagName('img');
images[0].style.width = '50px';
images[1].style.width = '100px';
```

- De nombreuses autres méthodes pour accéder aux éléments d'une page
  - **getElementsByName, getElementsByClassName, querySelectorAll, querySelector**

- Certaines sont aussi applicables à un élément et non au document en entier
  - p.ex. **getElementsByTagName**, **getElementsByClassName**, **querySelectorAll**, **querySelector**

```
<img alt="image"  src="image1.gif" />
<div id="zone">
   <img alt="image"  src="image2.gif" />
</div>
```

```
var zone = document.getElementById('zone');
var images = zone.getElementsByTagName('img');

images[0].style.width = '300px';
```

Modifier les éléments sélectionnés

- Accéder/Modifier les attributs d'une balise sélectionnée
  - accéder et modifier l'identifiant, ou la classe, de la balise : propriétés **id** ou **className**
  - accéder à un attribut : **getAttribute(** *nameAttribute* **)**
  - modifier un attribut : **setAttribute(** *nameAttribute*, *value* **)**
  - supprimer un attribut : **removeAttribute(** *nameAttribute* **)**

```
var lien = document.querySelector('#lien');
lien.setAttribute('href','www.univ-nc.nc');
```

- Accéder/modifier le contenu d'une balise: propriété **innerHTML**

```
var par1 = document.querySelector('#paragraphe1');
par1.innerHTML='Ceci est un texte <b>en gras</b>.';
```

- Accéder/modifier le contenu d'un champ de formulaire: propriété **value**

Modifier le style des éléments sélectionnés

- Accéder au style d'une balise
  - méthodes **getComputedStyle(** *element*, *pseudo* **).***propertyName* ou **getComputedStyle(** *element*, *pseudo* **)[***propertyName***]** : retournent la valeur de l'attribut de style *propertyName* de la balise référencée par *element*
    - *pseudo* = nom d'une pseudo-classe (p.ex. **hover**) ou **null** si non nécessaire

```
var element = document.getElementById('image1');          // un élément de largeur 100px
var largeur = getComputedStyle(element, null).width;     // largeur = "100px"
var l = parseInt(largeur,10);          // l = 100 , 10 correspond à la base
```

  - certaines propriétés permettent d'accéder directement à certains attributs de style d'une balise, p.ex. **offsetWidth**, **offsetHeight**, **offsetLeft**, **offsetTop**

```
var element2 = document.getElementById('image2');
var largeur2 = element2.offsetWidth;          // largeur complète (width+padding+border)
```

- Modifier le style d'une balise : propriété **style.***propertyName*

```
var element2 = document.getElementById('image2');
element2.style.width = 200 px;
```

Associer un comportement à un évènement

- Associer à des événements des fonctions JavaScript
  - un moyen classique d'ajouter de la dynamique aux pages Web

```
…
<img src="images/img1.jpg" alt="unc" id="image" />
…
<script  src="js/img.js"></script>
…
```

```
var image = document.querySelector('#image');
image.onclick = function(){ this.src = 'images/img2.jpg'; };
```

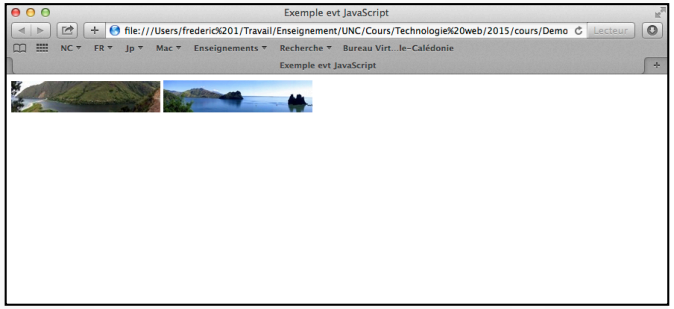
- Sélectionner l'élément puis lui associer un évènement (propriété = fonction)
  - ex. de propriétés **onload**, **onunload**, **onclick**, **ondblclick**, **onmousedown**, **onmouseup**, **onmouseover**, **onmousemove**, **onmouseout**, **onfocus**, **onblur**, **onkeypress**, **onkeydown**, **onkeyup**, **onsubmit**, **onreset**, **onchange**

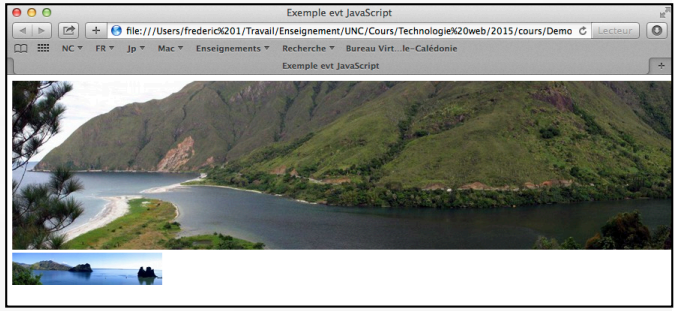
- **Attention:** code JavaScript toujours après les objets qu'il manipule (sinon les objets n'existent pas)

  ➔ Pour résoudre ce problème, utilisation de l'évènement: **onload**

Exemple galerie d'images minimaliste

```
<!DOCTYPE html>
<html lang="fr">
 <head>
    <title>Exemple evt JavaScript</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <script  src="js/galerie.js"></script>
 </head>
 <body>

    <img src="images/img1_small.jpg" alt="unc" class="galerie" />
   <img src="images/img3_small.jpg" alt="unc" class="galerie" />

 </body>
</html>
```





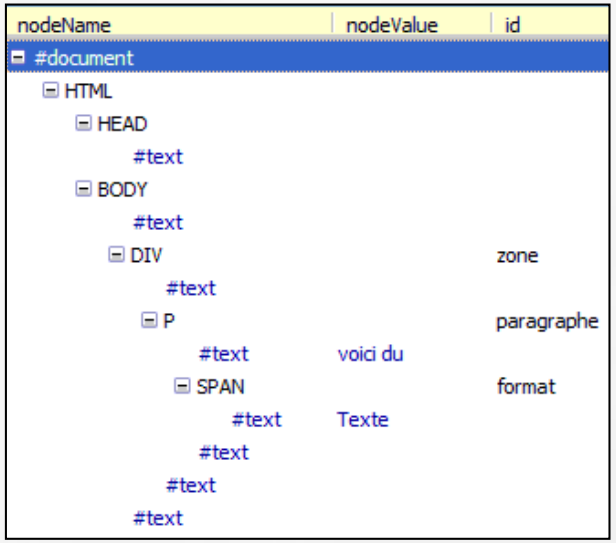

```
function increaseImg(){
    var images = document.getElementsByClassName('galerie');

    for( i=0; i< images.length ; i++)
        images[i].onclick = function(){
                                    if( this.src.search( '_small' ) != -1 )
                                            this.src = this.src.replace('_small.jpg','.jpg');
                                    else
                                            this.src = this.src.replace('.jpg','_small.jpg');
                                };
}
window.onload=increaseImg;
```

Le DOM – Document Object Model

- Définition du W3C
  - *The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.*

- Représentation des documents HTML sous forme arborescente
  - nœuds = éléments balise, texte et attributs
  - relation hiérarchique entre les nœuds
    - p.ex. l'élément **html** est l'enfant de **document** et le parent des éléments **head** et **body**

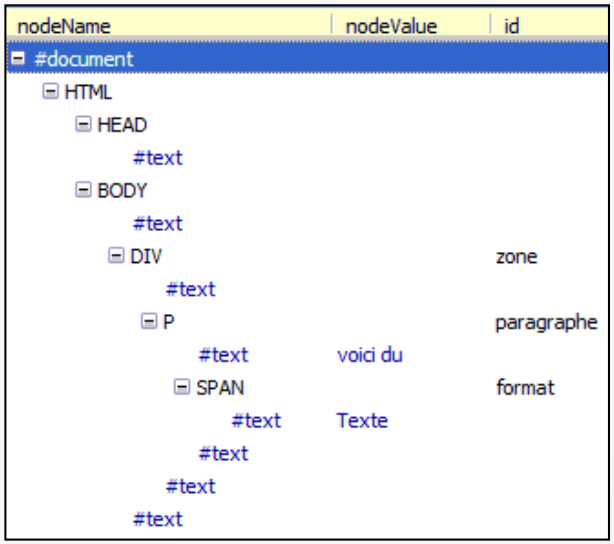
```
<html>
  <head> ... </head>
  <body>
    <div id="zone">
      <p id="paragraphe">
            voici du <span id="format" > Texte </span>
      </p>
    </div>
  </body>
</html>
```



Module DOM Inspector de Mozilla Firefox

Naviguer à travers les éléments de la page via le DOM

- Propriétés permettant de parcourir l'arborescence d'un document HTML
  - accès au noeud parent :
    element.**parentNode**
  - accès aux noeuds enfants :
    element.**childNodes** (retourne un tableau d'éléments),
    element.**fistChild** et *element*.**lastChild**
  - accès au noeud frère :
    element.**previousSibling** et *element*.**nextSibling**




```
var par = document.getElementById('paragraphe');
par.firstChild.style.color='red';
```

- Propriétés permettant d'avoir des informations sur un noeud de l'arborescence
  - nom du noeud courant (p.ex. #text, BODY) : *element*.**nodeName**
  - type du noeud courant (un code, p.ex. 3 pour du texte) : *element*.**nodeType**

Ajouter, supprimer, déplacer des éléments via le DOM

- Méthodes pour ajouter du contenu (créer des noeuds)
  - créer un élément : var element = **document.createElement(***tagName***)**
  - insérer le nœud *childElement* comme dernier enfant du nœud *element* : *element*.**appendChild(***childElement***)**
  - insérer *newElement* avant *referenceElement* dans *parentElement*: *parentElement*.**insertBefore(***newElement*, *referenceElement***)**

- Méthode pour supprimer du contenu (supprimer des noeuds)
  - supprimer un enfant d'un élément : *element*.**removeChild(***childElement***)**

- Méthode pour copier du contenu (copier des noeuds)
  - copier le nœud *element* dans la mémoire du navigateur :

    var clone = *element*.**cloneNode(***boolean***)**
    - le booléen indique si on copie ses descendants ou pas.

- Rq: le fichier HTML n'est pas modifié, tout est fait dans la mémoire du navigateur

Exemple: formulaire dynamique

```
<!DOCTYPE html>
<html lang="fr">
 <head>
    <title>Liste centres d'intérêt'</title>
    <meta http-equiv="Content-Type"
content="text/html;charset=utf-8" />
    <script  src="js/userProfil.js"></script>
 </head>
 <body>
    <h3>Centre(s) d'intérêt:</h3>
    <form action="profil.php" method="post" >
       <input type="text" name="mots[]"/>
       <button type="button"> Ajouter un
mot-clé</button> <br/>
       <input type="submit" value="ok" />
    </form>
 </body>
</html>
```

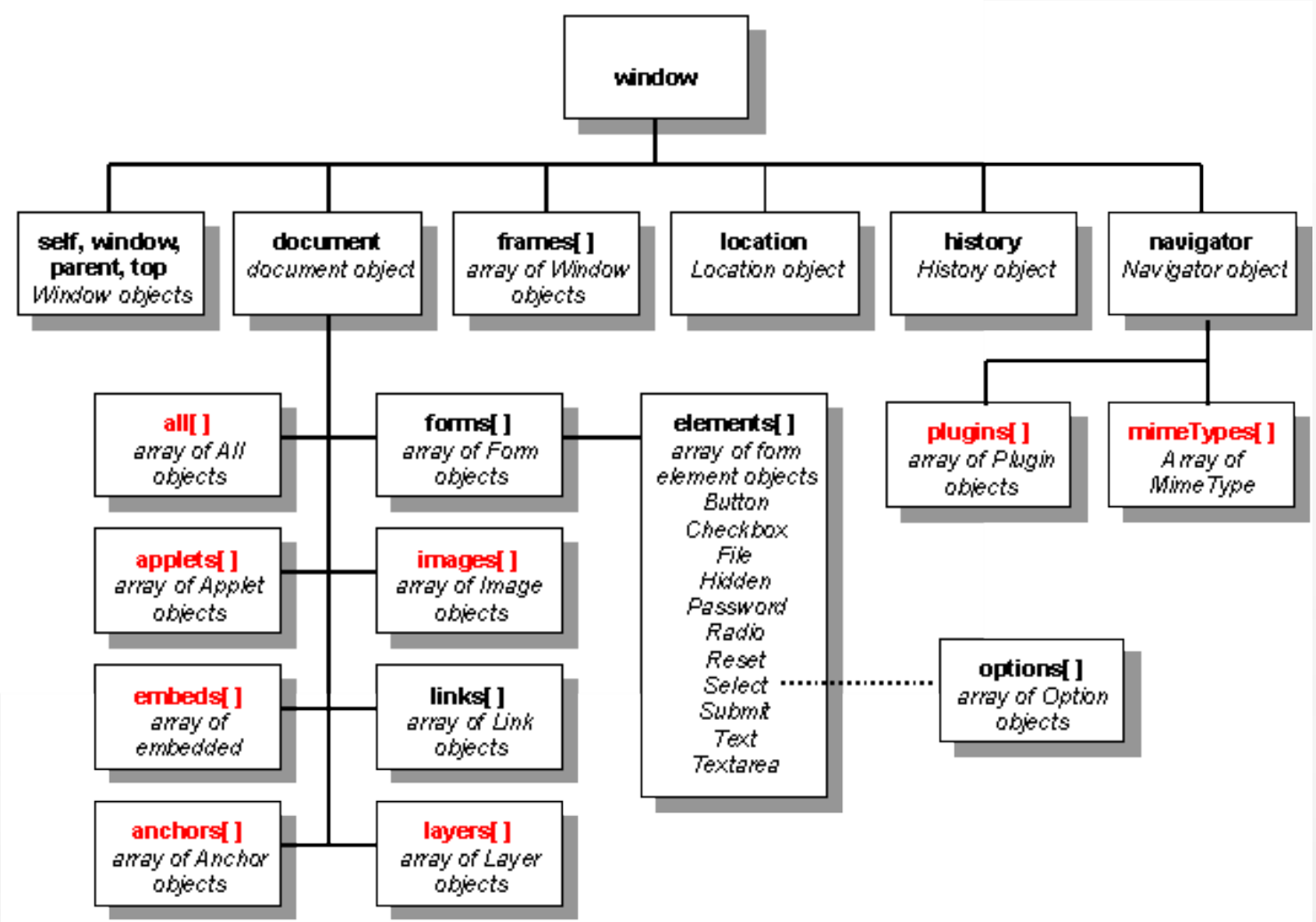

```
function addInput(){
    var listForms = document.getElementsByTagName('form');

    var copieInput = listForms[0].firstChild.cloneNode(true);
    copieInput.value=' ';
    var br = document.createElement('br');

    listForms[0].insertBefore( br, this);
    listForms[0].insertBefore( copieInput, this);
}

function mainController(){
    var listButtons = document.getElementsByTagName('button');
    listButtons[0].onclick = addInput ;
}

window.onload=mainController;
```

Récupération des valeurs en PHP:

```
<?php
      foreach($_POST['mots'] as $mot)
          echo $mot;
?>
```

Les objets du noyau JavaScript

- Manipulation des pages web possible grâce à un ensemble d'objets



Exemples d'objets du noyau JavaScript

- Objet **Window** (objet de plus haut niveau dans la hiérarchie)
  - représente la fenêtre du navigateur
- Objet **Document**
  - représente le document HTML, peut être utilisé pour accéder à tous ces éléments
- Objet **Location**
  - permet d'avoir des informations sur la "localisation" du document (propriétés .href, .hostname, .pathname, etc.).
- Objet **History**
  - permet de manipuler l'historique de navigation
- Objet **Navigator**
  - contient des informations sur le navigateur des clients
- Objet **Screen**
  - contient des informations sur l'écran d'affichage des clients

...

- Pour une description détaillée des propriétés et méthodes des objets JavaScript: http://www.howtocreate.co.uk/tutorials/javascript/javascriptobject

UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Cours de technologies Web
Un peu de sécurité

Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

Et la sécurité ?

- Comment pirater un site web ?
  - exploiter ses failles de sécurité
    - injection SQL
    - faille XSS
    - …
  - exploiter les failles de ses utilisateurs
- Qui ?
  - un "pirate" informatique
  - un bot, i.e. un programme informatique
- Comment se protéger ?
  - sécuriser l'accès à la base de données et masquer son fonctionnement
  - surveiller et sécuriser les données saisies par les utilisateurs ainsi que leur comportement
  - utiliser des protocoles de communication sécurisés lorsque cela est nécessaire

Les injections SQL

- Principe: se servir des champs de saisie (ou plus généralement des variables transmises) pour "injecter" du code SQL "malicieux"

- Permet de changer le résultat d'une requête, d'accéder au contenu de la base de données, voire de supprimer des tables

login inexistant dans la base de données!

Saisir:
" or 1 = 1#

$query= 'SELECT login FROM Users WHERE login='".$_POST['login']."' and password='".$_POST['password']."';

symbole MySQL permettant de mettre la suite en commentaire

Exécute la requête suivante:

SELECT login FROM Users WHERE login="truc" and password="" or 1 = 1#"

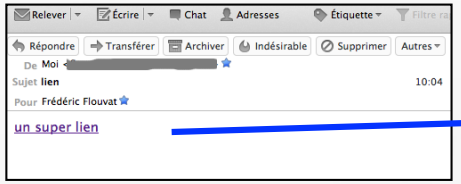
➢ retourne toujours un résultat

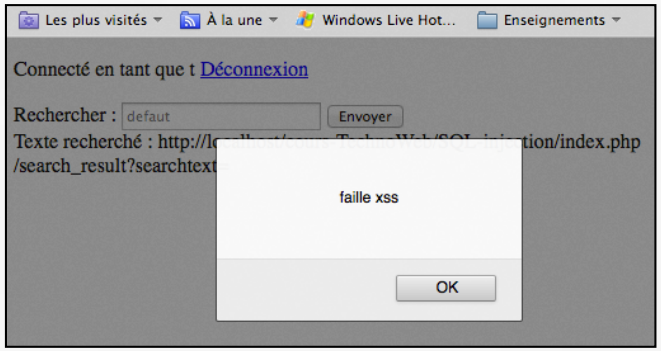
La faille XSS (*cross-site scripting*)

- Principe: injecter un script (exécuté côté client) dans des pages vues par d'autres utilisateurs
  - du code JavaScript, HTML, …

- Permet de rediriger vers un faux site (hameçonnage, *phishing*), de voler des informations enregistrées dans les cookies, de faire exécuter du code malicieux
  - failles non permanentes (par réflexion)

un mail avec un lien vers un site de confiance





http://sitesur.com/index.php/search_result?searchtext=<script language="javascript" src="http:/monsite.com/xss.js" </script>

  - failles permanentes
    - le lien ou le code malicieux est enregistré par le site dans un fichier ou dans la base de données, et est affiché/exécuté à chaque ouverture du site (p.ex. dans un post)

Les autres failles

- D'autres failles peuvent être exploitées *
  - la faille include
    - exploite une mauvaise utilisation de la fonction **include**

```
<?php include($_GET['fichier']); ?>
```

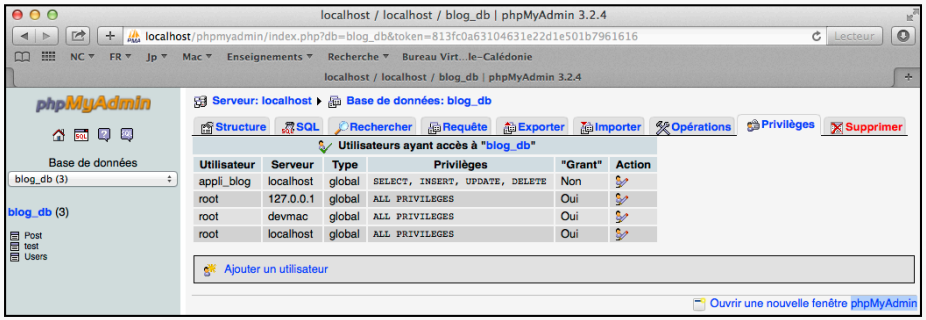
      - ➢ utilisée pour exécuter du code PHP situé sur une autre page, parcourir les répertoires du site, …
  - la faille upload
    - uploader du code PHP malicieux à la place d'une image, d'un fichier, …
    - vérifier l'extension du fichier ne suffit pas !
      p.ex. upload de "**backdoor.php\0.jpg**" fonctionnera
  - la faille CRLF (Carriage Return Line Feed)
    - insérer un retour à la ligne (%0A) dans un champ de saisie de texte
    - ➢ permet de récupérer le mot de passe d'un utilisateur en se mettant en copie (dans les champs de type "mot de passe oublié")
  - …

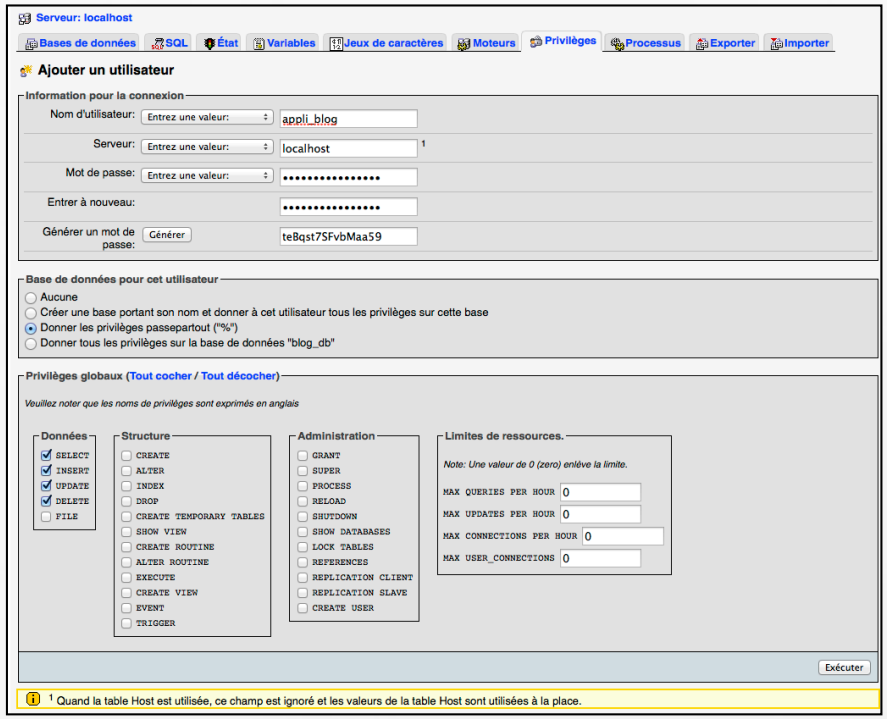
victime@service.com%0Apirate@service.com

* https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web

Sécuriser la BD: son accès et son fonctionnement

- Sécuriser l'accès à la base de données
  - créer des utilisateurs avec des droits limités dans la base de données
  - ➢ ne jamais connecter l'application web en "root" !





- Masquer son fonctionnement
  - ne pas afficher directement les erreurs SQL car cela peut donner des renseignements sur la base de données

Sécuriser la BD: son interrogation

- En PHP, utiliser les requêtes préparées et les paramètres liés de Mysqli (ou des Php Data Objects, i.e. PDO)
  - La fonction **mysqli_query** interdit l'exécution de plusieurs requêtes en même temps, mais n'empêche pas de modifier la requête
    - p.ex. évite juste l'injection de ***DROP TABLE Users;*** si l'utilisateur saisi ***" or 1; DROP TABLE Users; #*** mais accepte ***" or 1;***

➢ Intérêt des requêtes préparées

1. la requête est créée et envoyée au SGBD sans ses paramètres utilisateurs
   - p.ex. SELECT login FROM Users WHERE login=? and password=?
2. le SGBD compile et prépare le plan d'exécution de la requête
3. à l'exécution, l'application lie les valeurs des paramètres à la requête et le SGBD l'exécute

➢ impossible d'avoir des requêtes dans les paramètres

Sécuriser la BD: son interrogation

- A la place de **mysqli_query** et de **mysqli_fetch_assoc** faire:

1. créer une requête préparée avec **mysqli_prepare**(*$link*, *$query*)
   - les paramètres issus de données utilisateurs sont remplacées par des **?** dans la requête
2. lier les paramètres de la requête avec **mysqli_stmt_bind_param(***$stmt,$types, $var1,$var2,…* **)**
3. exécuter la requête avec **mysqli_stmt_execute(***$stmt***)**
4. lier les résultats (colonnes) à des variables avec **mysqli_stmt_bind_result(**$col1, $col2,…**)**
5. parcourir les résultats avec **mysqli_stmt_fetch(***$stmt***)**
6. libérer le résultat et/ou la requête avec **mysqli_stmt_free_result(** $stmt **)** **mysqli_stmt_close(** $stmt **)**

```
$link = mysqli_connect('localhost', 'user', 'pwd', 'blog_db');

$stmt = mysqli_prepare($link,'SELECT id, title FROM
Post WHERE login =?');


mysqli_stmt_bind_param($stmt, 's', $_POST['login']);



mysqli_stmt_execute($stmt);


mysqli_stmt_bind_result($stmt, $id, $title);


$userposts = array();
while ( mysqli_stmt_fetch($stmt) ) {
        $userposts[] = array( 'id' => $id, 'title' => $title ) ;
}
mysqli_stmt_free_result( $stmt );
mysqli_stmt_close( $stmt);
```

Vérifier toutes les données provenant de l'utilisateur

- Utiliser la fonction PHP **htmlspecialchars(**$string**)** pour filtrer les symboles du type <, & ou "
  - la fonction les remplace par leur "code" HTML (p.ex. **& "** ..)

- Vérifier le type des données passées en paramètres par l'utilisateur
  - p.ex. utilisation de la fonction PHP **intval(** $input **)** pour s'assurer qu'une valeur est un entier
  - ➢ http://www.php.net/manual/en/book.ctype.php

- Interdire les retours à la ligne dans certains champs de saisie (p.ex. pour les mails)
  - utiliser p.ex. la fonction PHP **str_replace**

```
$secure_str = str_replace(array('\n','\r',PHP_EOL),' ',$user_str);
```

- Renommer tous les fichiers uploader (du nom à l'extension)*

* https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web

Vérifier toutes les données provenant de l'utilisateur

- Mettre en place un système de jetons afin de garantir que les actions en cours sont bien faites dans le cadre de la session en cours *
  - générer aléatoirement un identifiant de session,
  - enregistrer cet identifiant dans la session et dans un champ **hidden** dans les formulaires
  - vérifier la correspondance à chaque modification
- Vérifier la durée de la "session"
  - stocker l'heure dans la session et dans champ **hidden** dans les formulaires
  - vérifier que la delta entre les deux ne dépasse pas une certaine limite
- Vérifier d'où viennent les données postées via la variable **$_SERVER[‘HTTP_REFERER’]**
- …

* https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web

Renforcer l'authentification des utilisateurs

- Sécuriser les mots de passe et ne pas les enregistrer en clair dans la base de données
  - au moins 10 caractères (lettres, chiffres, casse, symboles)

11 min. pour craquer e87a5de7
3 heures pour e8!a5d&7
3 jours pour A8!a5d&7
58 ans pour A8!a5d&712

(avec un PC "lambda")

  - le faire à la place de l'utilisateur si nécessaire …

```
$hash= password_hash($pwdGet, PASSWORD_DEFAULT);
INSERT INTO users VALUES ($login, $hash)
```

enregistrement

```
…
if( password_verify( $pwdGet, $hashInDB) ){
…
}
```

vérification

- Utiliser des captcha pour limiter les attaques des bots
  - mais les programmes de reconnaissances sont de plus en plus efficaces, et possibilité d'acheter des captcha (p.ex. 1000 pour 1$)

- Utiliser un protocole sécurisé (p.ex. HTTPS) lors de la phase d'authentification