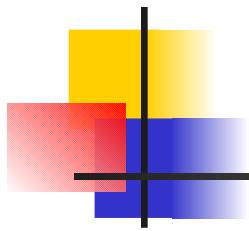


# AP5: Développement Web

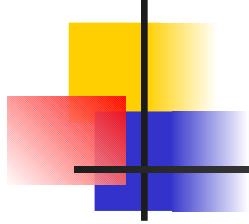
Université Cheikh Anta Diop, Dakar  
Ecole Supérieure Polytechnique  
Département Génie Informatique  
2ième année (1<sup>er</sup> cycle)  
2009-2010

Ibrahima FALL



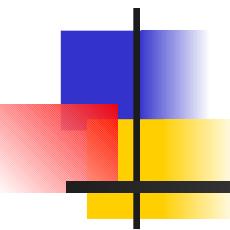
# FONCTIONNEMENT

- 15 séances de 2h
- 30 h : Cours + TD + TME
- **Evaluation**
  - **Un contrôle de connaissances (20%)**
  - **Des notes de TME (binomes) et d'interrogations écrites (10%)**
  - **Un projet commun (20%):**
    - Un rapport détaillé : rappel du sujet, algorithmique, choix (modules, ergonomie, ...)
    - Le tout envoyé par email ou rendu sur un support
    - Une démonstration sur machine
  - **Un DS final (50%)**
- **Attention: votre présence à toutes les séances est vivement souhaitée...  
votre respect des heures d'entrées/sorties également ...**

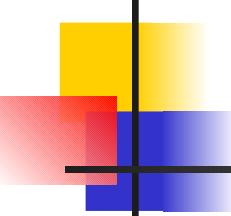


# PLAN DU COURS

1. Introduction/Généralités sur le WWW
2. Présentation **HTML, CSS**
3. Programme interprété par le client (**JavaScript**)
6. Programme interprété sur le serveur (**PHP**)
7. Accès bases de données (**MySQL**)



# ***INTRODUCTION/ GENERALITES SUR LE WWW***



# Introduction

**WWW : World Wide Web (la toile mondiale connectée en inter-réseaux)**

Né en 1989 au CERN

premier besoin ressenti : mettre en ligne des documents variés

**ex-tranet** : (tourné vers l'extérieur)

prestige, e-commerce, personnel distant

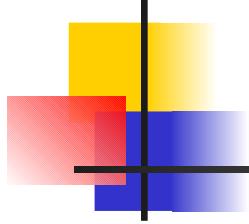
**in-tranet** : (tourné vers l'intérieur)

partage de la documentation interne, travail en groupes, ...

autres services de l'Internet :

la messagerie, le transfert de fichiers, les forums de discussion, la diffusion d'événements audio/vidéo en direct ...

*Principes du Web : « HTML + HTTP + URL », soit  
document hypertexte (pages) + protocole client/serveur + désignation des documents*



# Document hyper-texte

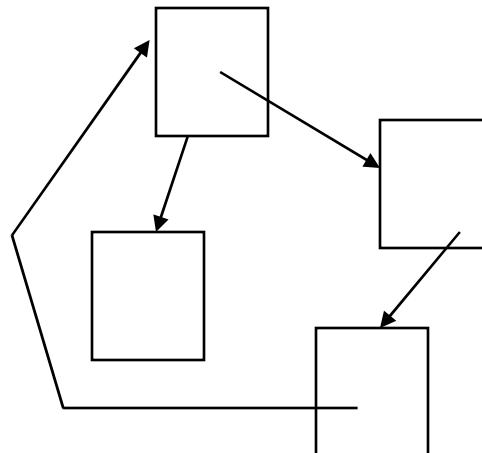
- Texte augmenté de liens de références à d'autres textes
- le texte est composite : informations textuelles, graphiques, audio, vidéo.

≠ Texte normal :

Organisation séquentielle

éventuellement des renvois sous forme de sommaire, index, note de bas de page

- Outils informatique de base : Edition sur un navigateur, ...  
(navigateur: IE, Mozilla, Safari, etc. : windows, mac, unix, ....)



# Edition de document hypertexte



*'Browser' ou navigateur de pages hypertextes, ...*

*Action exécutée : affichage d'un bandeau déroulant de texte*

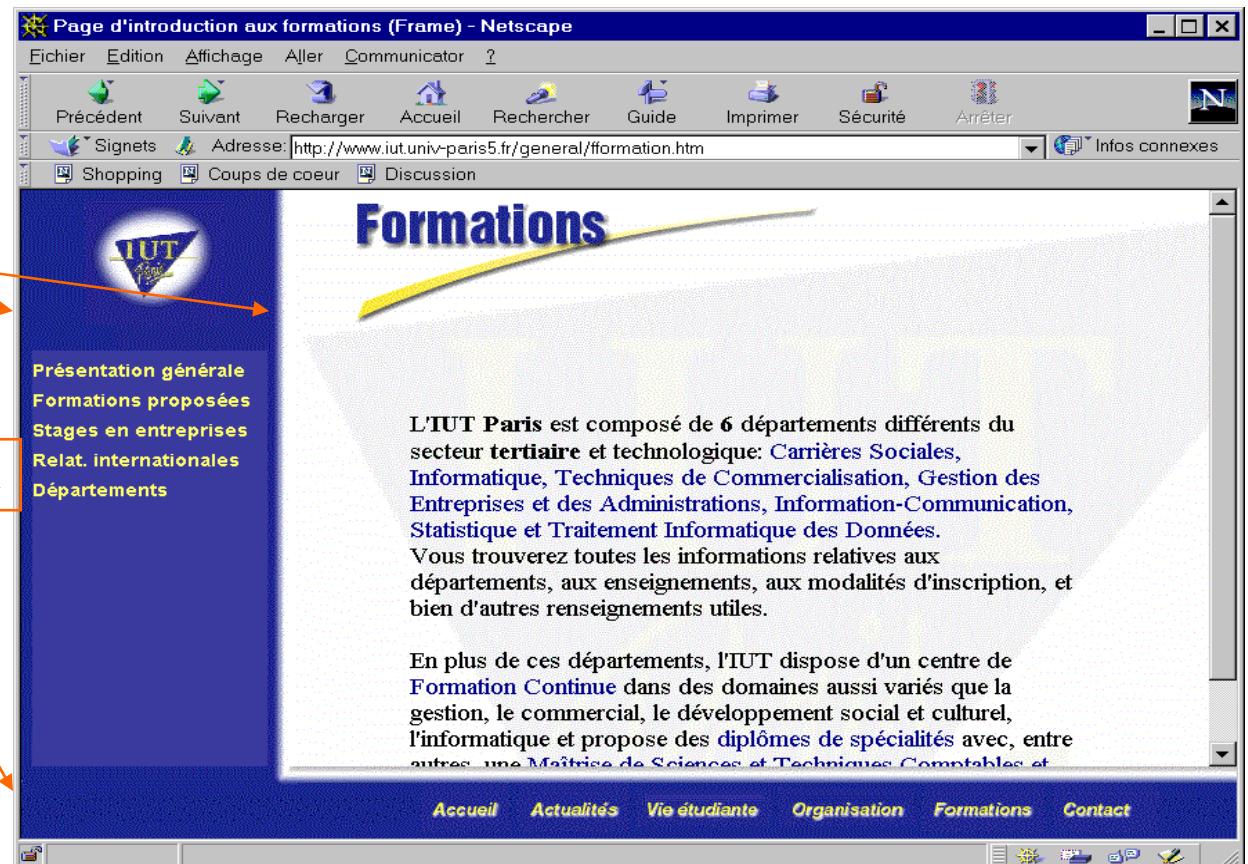
*Texte , image renvoyant vers d 'autres pages (liens)*

*Images de fond*

# Document hypertexte structuré

3 cadres  
(frames)

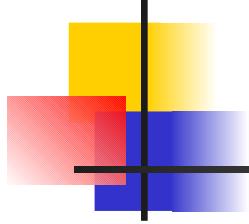
menu



HTML 3.2 :  
table, image cliquable  
& applet

HTML 4 :  
cadre  
& feuille de style (CSS)

XHTML :  
XML+CSS



# Balisage du document hypertexte

## HTML

HTML (v4) : HyperText Markup Language

permet de décrire un document et la façon dont il est formaté  
contient le document en lui même et des balises (ou tags)

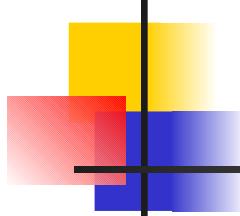
Ex : <b> ce texte est en gras </b>

↑    ↑  
balise début\_gras (bold)                                  balise fin\_gras

Instance de SGML (Standard Generalized Markup Language) :

standard international ISO 8879 de 1986

méta-langage qui permet de définir des classes de documents (exple DTD\_HTML)  
approche descriptive : dissocie l'organisation logique (point de vue de l'auteur)  
de sa structure physique (point de vue du typographe)



# Contenus des pages : type MIME

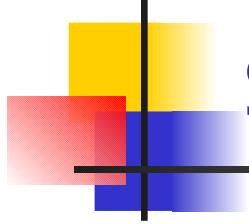
MIME : Multipurpose Internet Mail Extensions

Les navigateurs reconnaissent les types de données à afficher.

En fonction d'extensions associées, ils affichent eux mêmes le contenu ou font appel à un composant externe.

Exemple sous Netscape (voir préférences: Navigateur: Application)

Type de données	type/sous-type	extension	géré par
Plain text	text/plain	.txt .text .ini ...	mozilla
xml text	text/xml	.xml .xsl	i-explorer
real audio	audio/x-pn-realaudio	.ra .ram	realplayer
mpeg2 video	audio/x-mpeg2	.mpv2 .mp2v	mplayer2
gif image	image/gif	.gif	mozilla
jpeg image (compressée)	image/jpeg	.jpeg .jpg ...	mozilla



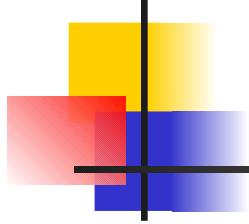
# Services associés aux navigateurs

Navigateur	courriel (e-mail)	forum (news...)	Discussion (chat....)	Son/vidéo	Edition page html (wysiwyg)
I. Explorer	Outlook	NetMeeting		NetVideo	FrontPage
Netscape	Messenger	Collabra	Conference	windows-media	Composer

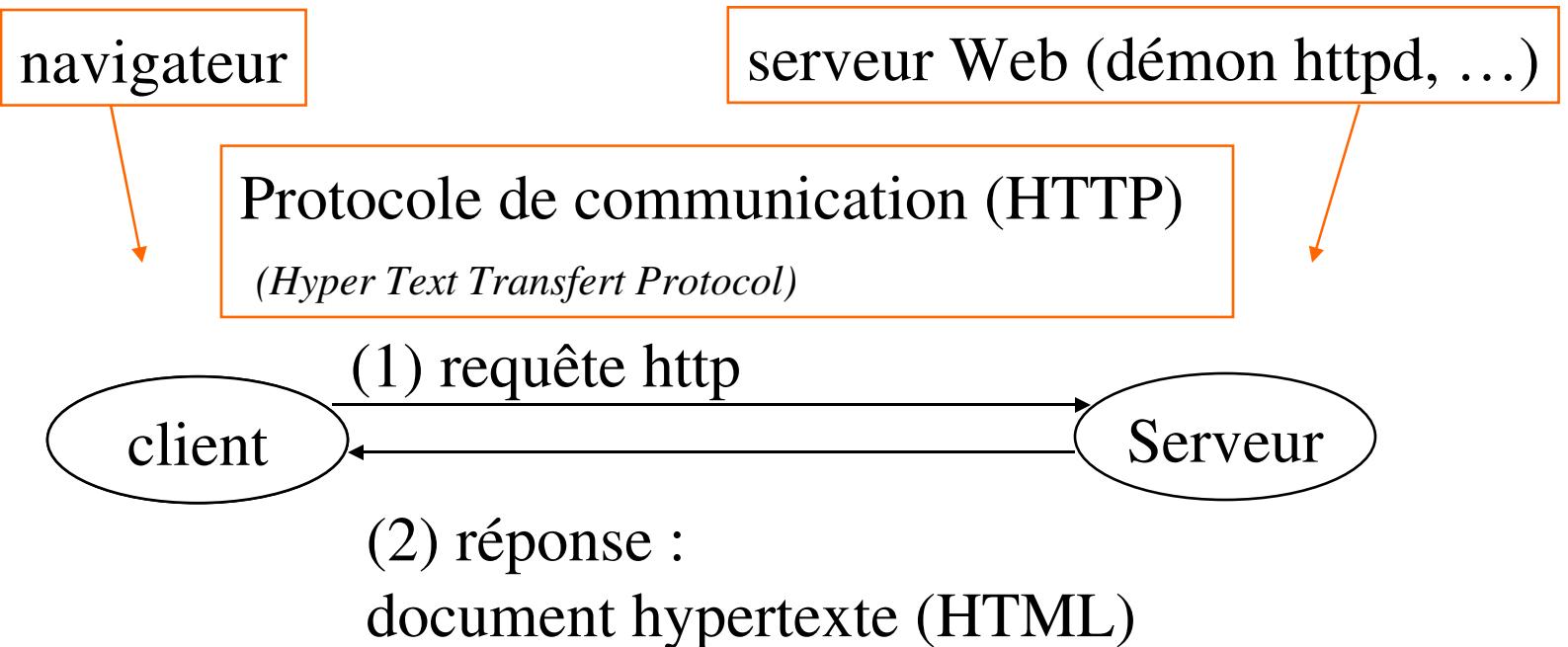
- Conférence de netscape : + conférence audio/vidéo, tableau blanc sur internet
- Active Desktop de microsoft : dossiers vus comme des pages internet  
Les applications d'Office de microsoft font appel en interne au noyau du navigateur ie

*News,Forum: accès à des zones d'échanges d'information (texte) sur un thème contrôlé par un modérateur*

*Chat : échange de texte en temps réel.*

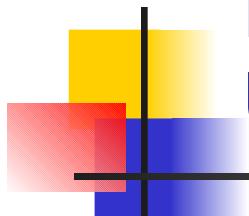


# Dialogue client-serveur



Version 1.1 du protocole http : requête/réponse de pages indépendantes  
(mode non connecté)

Version sécurisée : https

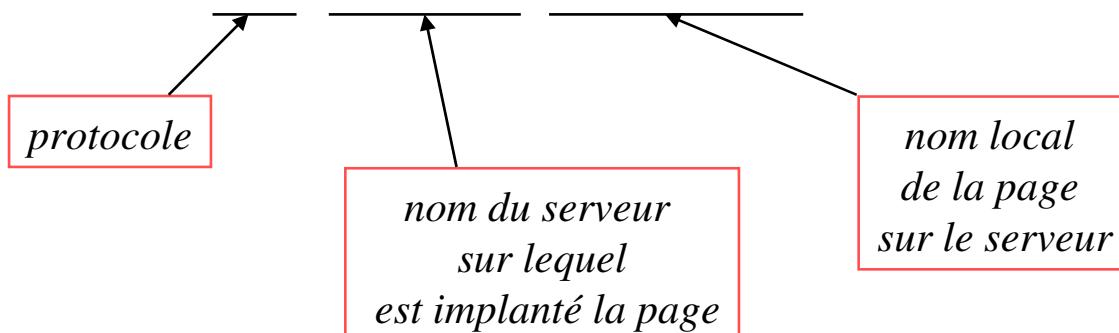


# Désignation des ressources sur le WEB : URL

**URL** Uniform Resource Locator

*désignation d'une page ou de toute autre ressource sur un internet (exécutable, fic script...) 3 parties + [des arguments] (arguments=options ou paramètres optionnels).*

*Ex simple : http://www.lip6.fr/welcome.html*



L'URL permet de répondre aux questions :

- . quelle est la page appelée ?
- . où est localisée cette page ?
- . comment peut-on y accéder ?
- . comment l'interpréter

Chaque page a une désignation unique :  
(pas d'ambiguïté possible)

: // et / séparateurs

# Correspondance des noms et des adresses d'internet (IP)



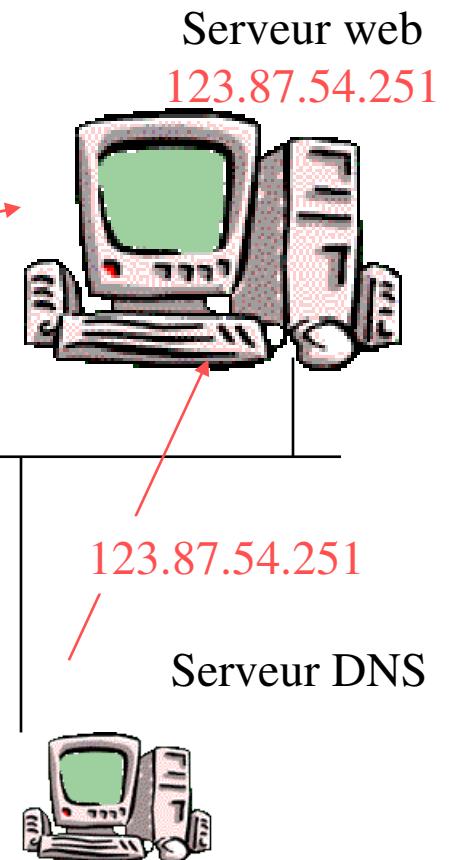
*Protocole TCP/IP :  
base pour tout protocole d 'internet*

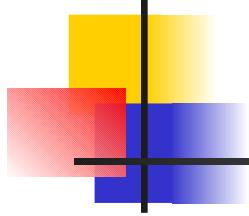


(1) adresse IP: 123.87.54.251

(1') nom: www.iut-paris5.fr

*Chaque machine sur internet  
est associée à un DNS  
qui fournit l'adresse IP de chaque nom  
(domain name system)*

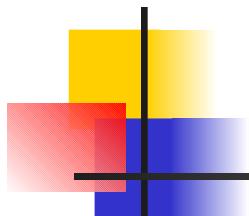




# Protocoles reconnus par les navigateurs

protocole	type	exemple
http	document hypertexte	<a href="http://www/lip6-src.fr/~ilie/index.htm">http://www/lip6-src.fr/~ilie/index.htm</a>
ftp	transfert de fichier	<a href="ftp://user:foo@research.digital.com/personal/latex.sty">ftp://user:foo@research.digital.com/personal/latex.sty</a>
file	accès au système de fichiers local	<a href="file:///C:/Site%20IUT%20Paris5/index.htm">file:///C:/Site%20IUT%20Paris5/index.htm</a>
news	accès à un forum de discussion accès à un article dans un forum de discussion	<a href="news:comp.lang.java news">news:comp.lang.java news</a> <a href="news:AB02224321122@comp.lang.java">news:AB02224321122@comp.lang.java</a>
mailto	envoi de courrier	<a href="mailto:Ibrahima.Fall@lip6.fr">mailto:Ibrahima.Fall@lip6.fr</a>
telnet	connexion à distance	<a href="telnet://euler.inria.fr">telnet://euler.inria.fr</a>
gopher	accès à une base de documents arborescents	<a href="gopher://gothic.tc.umn/11/Libraries">gopher://gothic.tc.umn/11/Libraries</a>
wais	accès à une base de documents indexés	<a href="wais://xenon.inria.fr/cuisine?tarte+tatin">wais://xenon.inria.fr/cuisine?tarte+tatin</a>

URL : fédère l'accès à des systèmes d'informations hétérogènes fonctionnant avec des protocoles différents



# URL : forme générale

```
protocole"://"  
[utilisateur[:":"motdepasse]@](nomladresseIP)[:port]  
["/"chemin]["/"nomdefichier][#ancre][?paramètres]
```

http://www.imag.fr/equipe/sirac/projet.html

(*nom du serveur*)

http://123.87.54.251/index.html

(*adresse IP du serveur*)

http://www.altavista.com/query.exe?iut+paris5

(*paramètre, +: blanc*)

http://www.info.projet/search?nom=ilie&prenom=jm

(*paramètres, &: et*)

http://xenon.inria.fr:8080/hello.html

(*n° port de comm. du serveur*)

http://milo.ecoledoc.lip6.fr/index.html#annuaire

(*ancre - pointeur interne*)

*Le format complet des URLs : norme RFC 1738 et 1808  
(voir <http://www.w3.org/Addressing/>)*

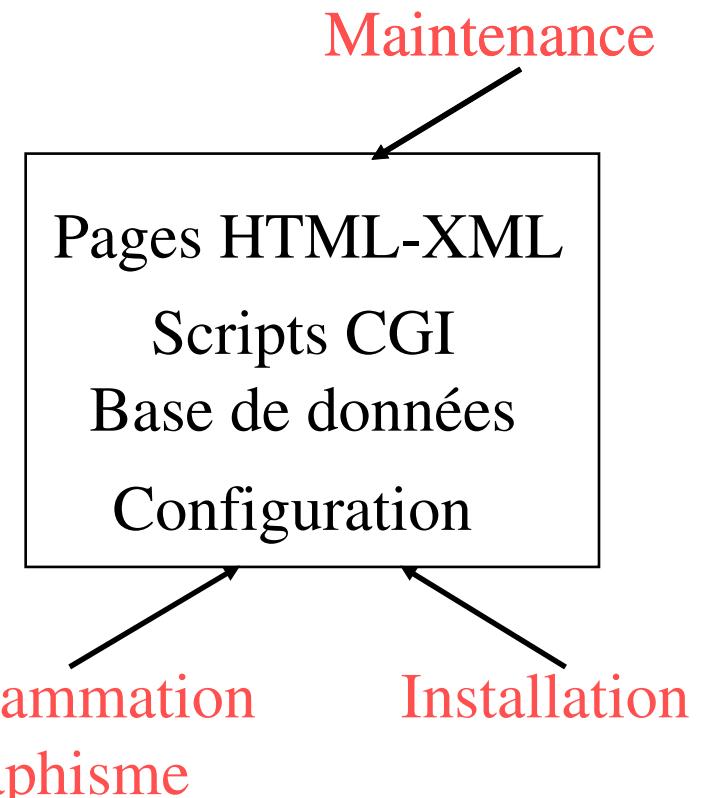
# Profession du responsable internet (web-master)

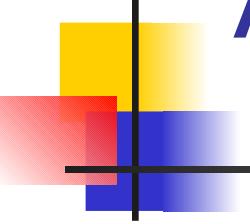
*Assure la veille technologique +  
administration de site web +  
création d'applications intranet/internet +  
travail d'équipe : webmaster+programmeurs-  
métier, graphistes spécialisés.*

Le client            Le serveur

Page statique : prédéfinie

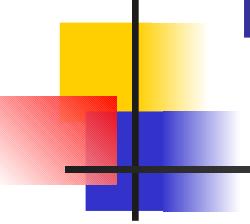
Page dynamique : créée par programme  
(exple : réponse via une requête à une BD)





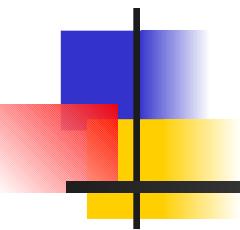
# A ne pas faire

- Un site bourré d'images de taille géante ou toutes animées.
- Un site monotone
- Un site qui ne respecte pas la configuration de l'utilisateur lambda
- Un site non homogène
- Un site à pointeur nul



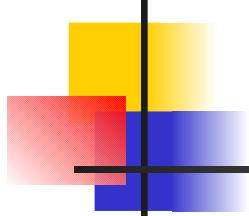
# L'idéal

- Un site performant mais compatible avec d'anciennes versions de browser (ie 5)
- Un site qui a un fil conducteur
- Un site dont l'interface est agréable
- Tenir compte de la pire des bandes passantes: le modem
- Un site dont le contenu mais non la forme change régulièrement



# ***LES DOCUMENTS HTML ET LEUR PRESENTATION***

---



# Documents HTML

## Présentation

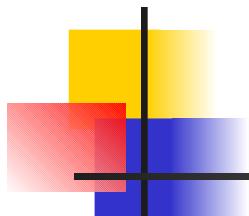
- composite texte - image - son - vidéo
- formatage simple

## structure du document

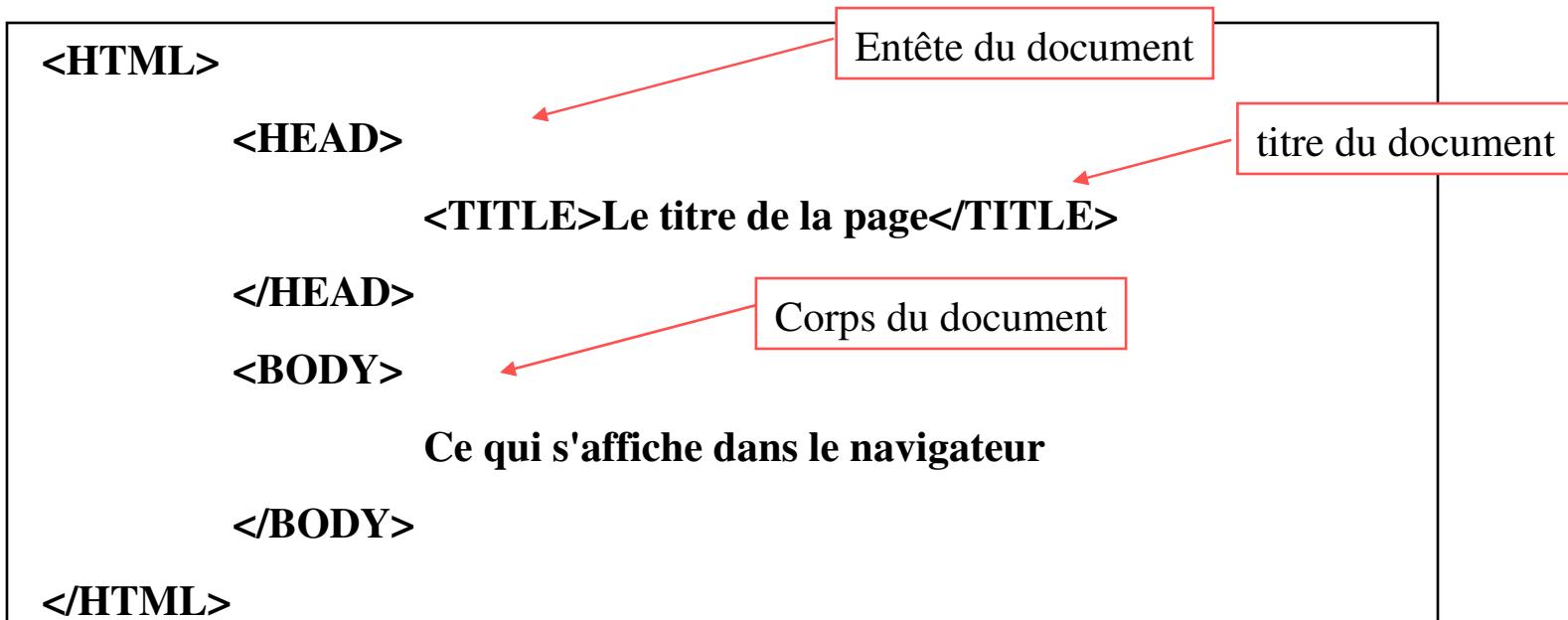
- repose sur les étiquettes (Tag)

## Apport

- l'interactivité avec les liens



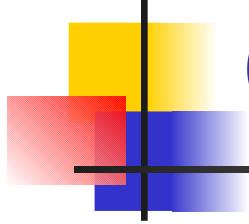
# Le cadre général



Remarque générale : <unTag> ... </fin\_untag>

Non sensible à la casse et peut contenir des options

Commentaires : <- - commentaire d'un fichier HTML -->



# Corps du document HTML

```
<BODY attr1="val1" ... attrn="valn"> ...corps... </BODY>
```

Attributs :

bgcolor : couleur de fond

text : couleur du texte

background : URL de l'image en fond d'écran

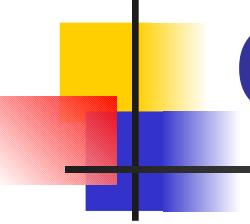
link : couleur des liens non encore visités

vlink : couleur des liens déjà visités

alink : couleur des liens lorsque l'on clique dessus

attributs optionnels  
sinon, valeur par défaut du navigateur

```
<body bgcolor="#FFFFFF" link="#FFFFFF"  
      vlink="#FFFFFF" alink="#FFFFFF"  
      BACKGROUND="images/FondIUT.jpg">
```



# Code des caractères

Jeu de caractères connu : ISO 8859-1 (Latin-1)

Contient tous les caractères accentués des langues latines (sauf œ)

Encodage des caractères possibles:

é	&acute;
è	&egrave;
<>	&lt; &gt;
&	&amp;

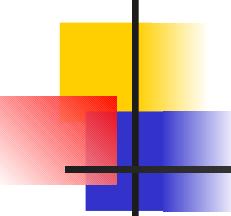
Le codage des caractères spéciaux est utile pour les protocoles ou quand on produit des pages par programme

*<h1>Les accents aigus &acute; passent</h1>  
<h1>Les accents graves &egrave; passent aussi</h1>*

Codage directe possible :

Espace	%20;
'	%39;
"	%34

<http://soleil:8080/simple%20essai.html>



# Gestion des couleurs

2 solutions :

#RRGGBB (en 3 codes hexa : Rouge\*Vert\*Bleu)  
noms en clair : green, yellow, purple, red, blue, ...  
(plus simple mais moins précis)

**Black = "#000000"**  
Silver= "#C0C0C0"  
Gray= "#808080"  
Maroon= "#800000"  
Red= "#FF0000"  
Purple= "#800080"  
Fuschia= "#FF00FF"  
**White= "#FFFFFF"**

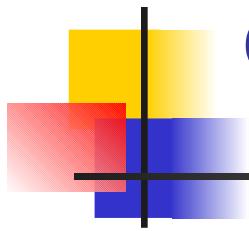
Green= "#008000"  
Lime= "#00FF00"  
Olive= "#808000"  
Yellow= "#FFFF00"  
Navy= "#000080"  
Blue= "#0000FF"  
Teal= "#008080"  
Aqua= "#00FFFF"

**Utilisation :** <FONT color = " "#FF0000 " ...>... </FONT>  
<BODY bgcolor = " "#00FF00 " ...> ... </BODY>  
<TD    bgcolor = " "#0000FF " ...>...</TD>

**Attention**  
on n'obtient pas forcément  
le même rendu suivant les navigateurs

on peut aussi utiliser  
un logiciel de gestion d'image  
pour connaître les valeurs  
de couleur RGB  
pour un objet prédéfini

[texte rouge]  
[corps fond vert]  
[cellule fond bleu]



# Gestion du fond d 'écran (background)

2 approches différentes:

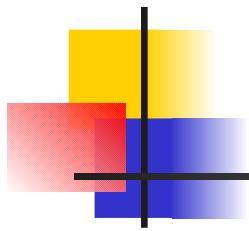
La couleur:

```
<BODY BGCOLOR="#$$$$$$">  
$$$$$$= code hexadécimal RGB
```

La texture:

```
<BODY BACKGROUND="URL">  
crée une mosaïque en fond
```

```
<BODY BACKGROUND="..../image/entete.gif">
```



# Structuration du document

<H?>...</H?>

6 niveaux différents de titre (H1..H6)

<P> paragraphe </P>

délimiteur de paragraphe

pour formater chaque paragraphe en modifiant l'alignement

<P align="left" | "center" | "right"> ... </P>

<HR> insère une ligne de séparation

Autres :

<CENTER> ...texte... </CENTER>      centrage d'un texte quelconque

<BLOCKQUOTE>                      mise en retrait du texte

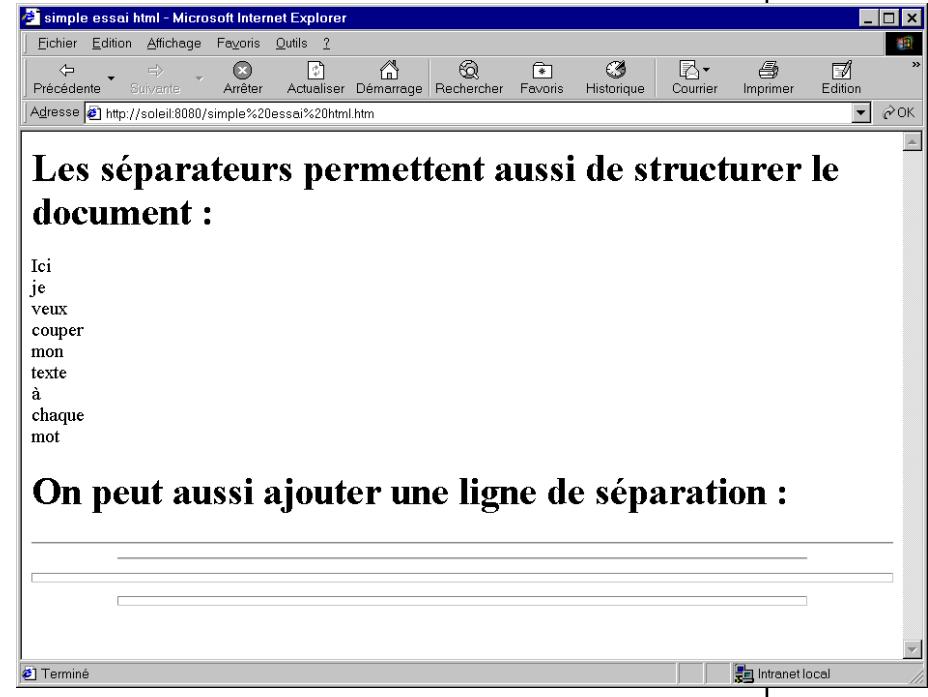
<BR>                              coupure forcée d'une ligne (au sein d'un paragraphe)

il existe aussi des styles prédéfinis (code, définition, citation ...)

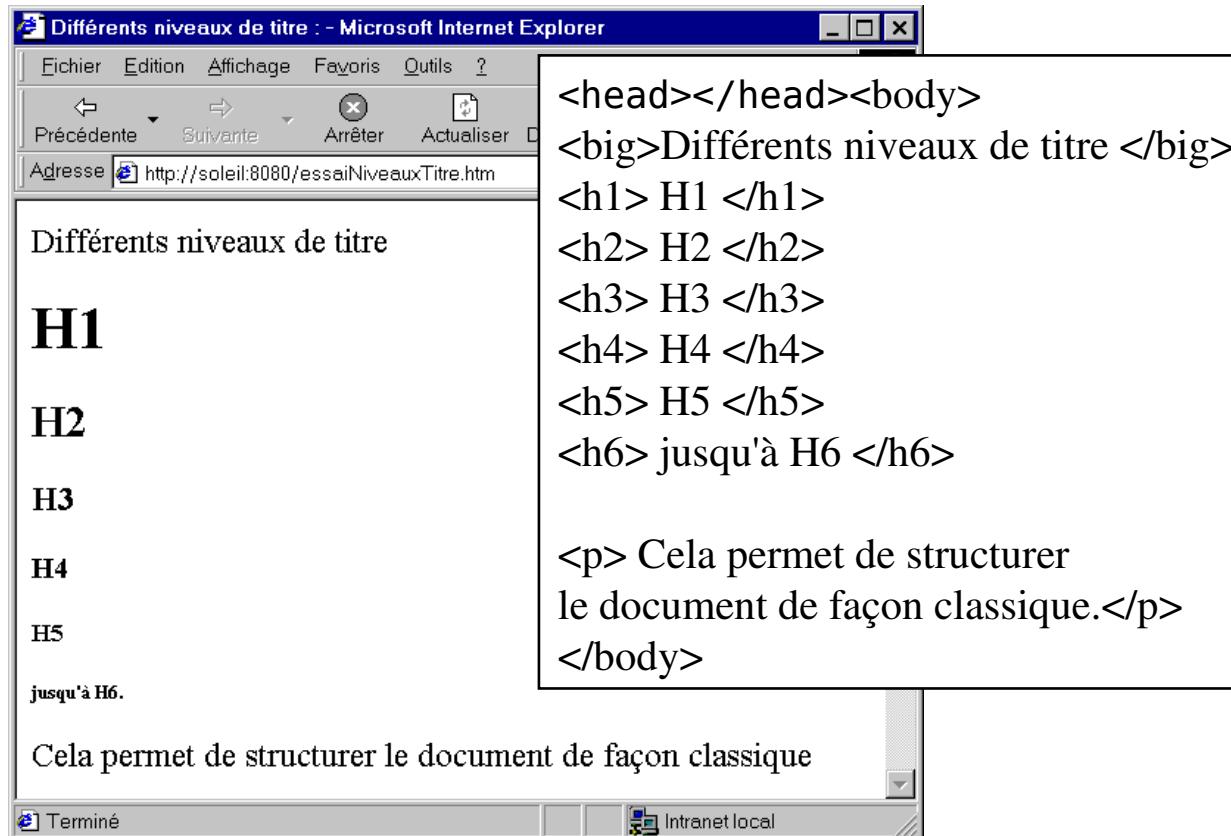
# Exemples de séparateurs

```
<head></head><body>
<h1>Les séparateurs permettent aussi de structurer le document :</h1>
Ici <br/>
je <br/>
veux <br/>
couper <br/>
mon <br/>
texte <br/>
à <br/>
chaque <br/>
mot <br/>

<h1>On peut aussi ajouter
une ligne de séparation : </h1>
<HR>
<HR width="80%" />
<HR size="10" />
<HR width="80%" size="10" /></body>
```



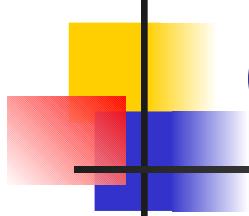
# Exemples de niveaux de titre



# Exemples de styles prédéfinis

The screenshot shows a Microsoft Internet Explorer window titled "Essai des styles prédéfinis - Microsoft I...". The window has a toolbar with "Fichier", "Edition", "Affichage", "Favoris", "Outils", and a help icon. Below the toolbar is a navigation bar with "Précédente", "Suivante", "Arrêter", and "Actualiser" buttons. The address bar shows the URL "http://soleil:8080/essaiStyle.htm". The main content area contains a sidebar with categories: "Citation", "Exemple", "Algorithme", "Définition", and "Variables". To the right of the sidebar is a large text area containing the following HTML code:

```
<body>
<p>    <CITE> Citation </CITE> </p>
<p>    <SAMPLE> Exemple </SAMPLE> </p>
<p>    <CODE> Algorithme </CODE> </p>
<p>    <DEFINITION> Définition </DEFINITION> </p>
<p>    <VAR> Variables </VAR> </p>
</body>
```



# Mise en valeur de la police de caractères

<BIG> ...texte... </BIG> "grande" police

<SMALL> ...texte... </SMALL> "petite" police

<FONT size="?"> ... </FONT> taille de police de 1 à 7 (7 est la plus grande)

<FONT size="+?">... </FONT> taille relative par rapport à la taille courante

<FONT size="-?">... </FONT> taille relative par rapport à la taille courante

<BASEFONT size="?"> initialise la taille courante (par défaut size="3")

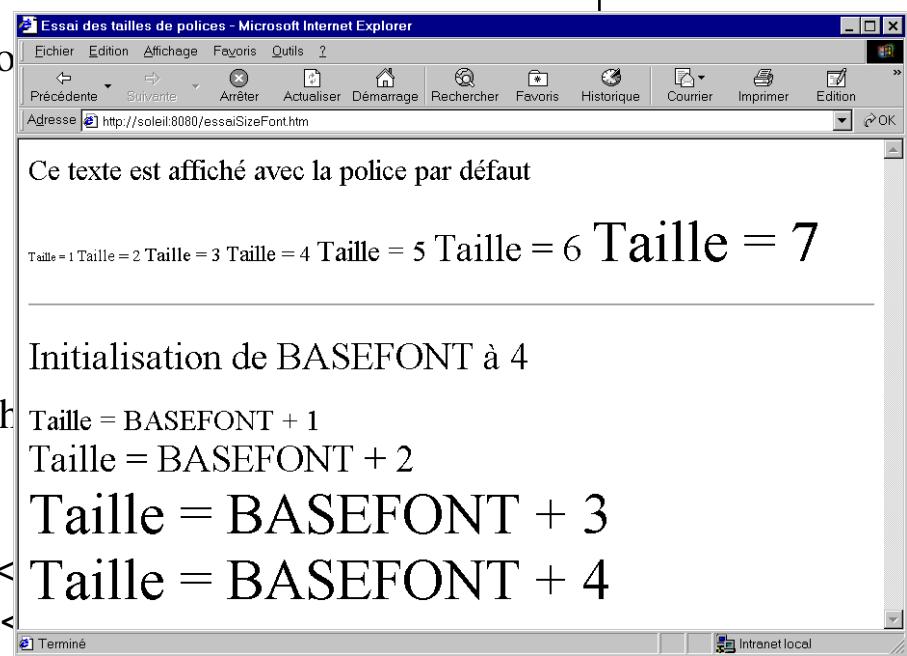
<B> ... gras... </B>, <I> ... italique... </I>, <U> ... souligné... </U>, etc

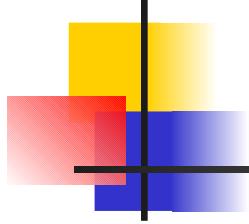
<PRE> ...texte préformaté (à ne pas reformater)... </PRE>

# Exemples : changement des tailles de police

```
<body>
<p><big><big> Ce texte est affiché avec la po
<p><FONT size="1">Taille = 1 </FONT>
<FONT size="2">Taille = 2 </FONT>
<FONT size="3">Taille = 3 </FONT>
<FONT size="4">Taille = 4 </FONT>
<FONT size="5">Taille = 5 </FONT>
<FONT size="6">Taille = 6 </FONT>
<FONT size="7">Taille = 7 </FONT> </p><h

<p><BASEFONT size="4">
<big><big>Initialisation de BASEFONT à 4 <
<FONT size="+1">Taille = BASEFONT + 1 <
<FONT size="+2">Taille = BASEFONT + 2 </FONT> <br>
<FONT size="+3">Taille = BASEFONT + 3 </FONT> <br>
<FONT size="+4">Taille = BASEFONT + 4 </FONT> </p>
</body>
```





# Les liens

**<A HREF="*url*"> lien - texte ou image </A>**

- permet d'aller vers un autre document en cliquant sur le texte ou l'image

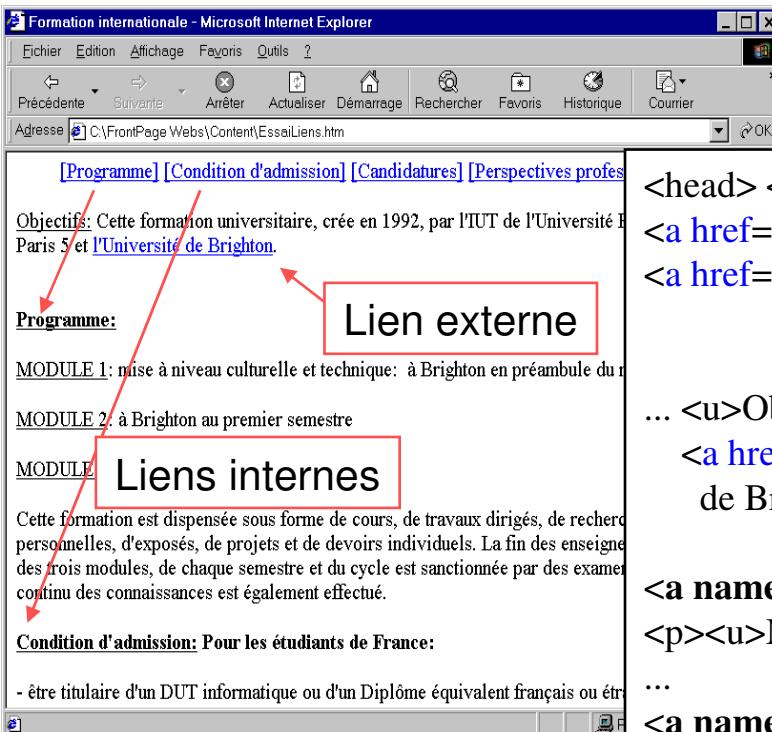
**<A HREF="*XXX#signet*"> texte ou image </A>**

- *XXX* est une url: lien vers un autre document
- *XXX=rien*: lien vers le même document
- "*#*" seul lien vers la position courante
- signet nom du pointeur dans le document

**Spécification des signets** (*bookmark ou pointeur dans le document*)

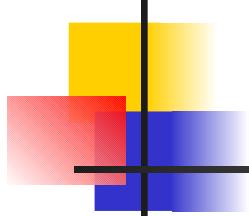
**<A NAME="signet"> texte </A>**

# Exemple de liens



```
<head> </head> <body>
<a href="EssaiLiens.htm#Programme">[Programme]</a>
<a href="EssaiLiens.htm#Admission">
    [Condition d'Admission]</a>
...
... <u>Objectifs:</u> Cette formation universitaire, ...
<a href="http://www.bton.ac.uk/">l'Université
    de Brighton</a>.

<a name="Programme">Programme</a> <br>
<p><u>MODULE1:</u><br>
...
<a name=" Admission">Condition d'admission</a> <br>
<p>Les dossiers de candidatures seront disponibles,
    au département informatique </p>
    ... </body>
```



# Les images simples

<IMG SRC="*url*">

<IMG SRC="maphoto.gif">

Attributs optionnels

ALIGN =

Insère l'image maphoto.gif

"top", "middle", "bottom", "left" ou "right"  
aligne l'image par rapport à la ligne  
HEIGHT=, WIDTH= hauteur, largeur de l'image

<IMG SRC="URL" ALT="explication concise pour la figure">

tolérance pour les browsers non graphiques

et explication pendant le chargement

(le survol de la forme fait apparaître le texte de ALT)

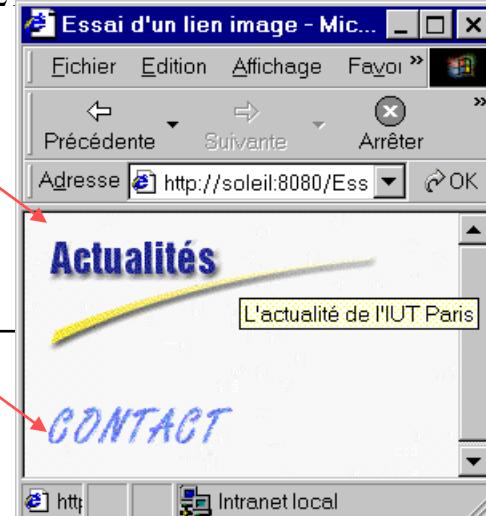
# Liens et images

```
<head> </head> <body>
<a href="general/factualites.htm">

</a><br><br>

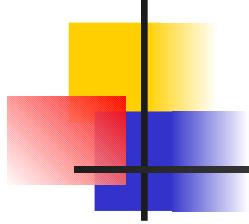
<a href="mailto:bedos@iut.univ-paris5.fr">

</a></body>
```



Note :

le second lien lance le gestionnaire de mail



# Les images "map"

<IMG SRC="*url*" ISMAP>  
indique où se trouve la carte

<MAP NAME="*nom de la carte*"> ...zones graphiques (shapes)... </MAP>



<AREA SHAPE="RECT" COORDS=",,,," HREF="*URL*">  
*permet de décrire les liens par zone*

# Les listes

3 types de listes <XXX>:

Non ordonnée:

XXX=UL

Ordonnée:

XXX=OL

attribut optionnel TYPE="DISC", "SQUARE" ou "CIRCLE"

Répertoire:

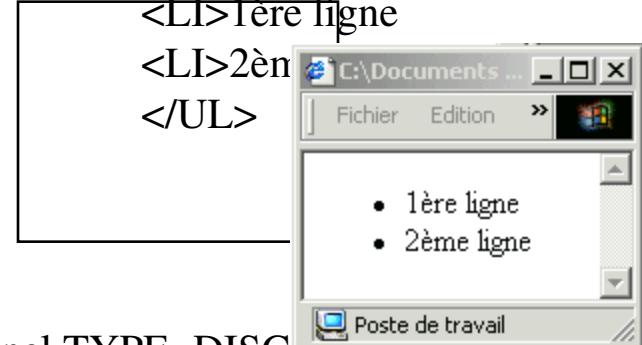
XXX=DIR (décalage)

<UL>

<LI>1ère ligne

<LI>2ème ligne

</UL>

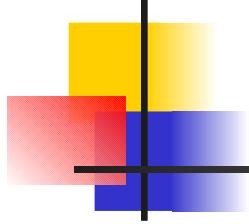


- 1ère ligne
- 2ème ligne

Attributs optionnels : <XXX TYPE="type\_graphique">

pour "UL" :      TYPE="DISC", "SQUARE" ou "CIRCLE"

pour "OL" :      TYPE="1" (décimal), "a" (minuscule), "A"(majuscule), "i" ou "I" (romain)

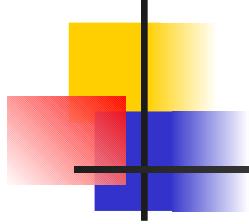


# Les tableaux (1)

```
<TABLE>
    <TR>
        <TD>contenu de la cellule 1-1</TD>
        <TD>contenu de la cellule 1-2</TD>
    </TR>
    <TR>
        <TD>contenu de la cellule 2-1</TD>
        <TD>contenu de la cellule 2-1</TD>
    </TR>
    ...
</TABLE>
```

| Ligne 1

| Ligne 2



# Les tableaux (2)

Les attributs:

Border: définit l'épaisseur du cadre

Cellspacing, Cellpadding: détermine l'espace entre le texte et le bord

Width: largeur du tableau (pixel ou %)

Colspan, rowspan: débordement d'une cellule sur la colonne ou la ligne suivante

Les attributs spécialisés:

**<TH> ..en-tête...</TH>**

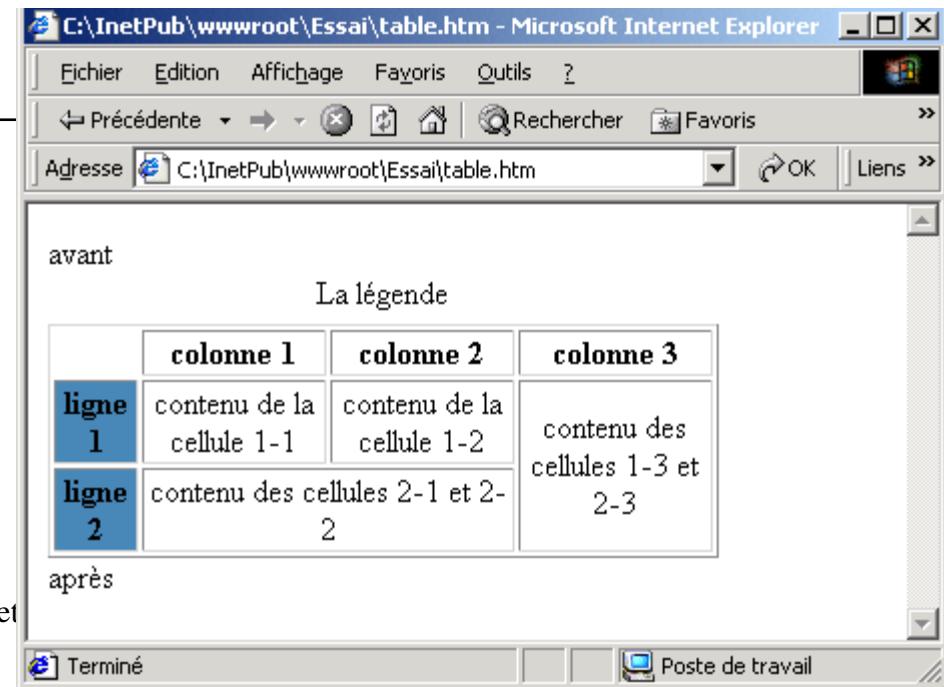
donne un titre au tableau

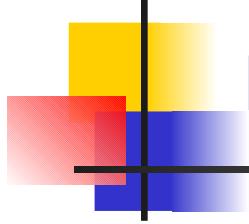
**<CAPTION> ... La légende:...</CAPTION>**

pour décrire de façon succincte le contenu du tableau

# Exemple de tableaux

```
<HTML><BODY>avant
<TABLE BORDER=1 WIDTH=80% >
  <CAPTION> La légende </CAPTION>
  <TR >
    <TH ></TH>
    <TH>colonne 1</TH>
    <TH>colonne 2</TH>
    <TH>colonne 3</TH></TR>
  <TR ALIGN="CENTER">
    <TH BGCOLOR="STEELBLUE">ligne 1</TH>
    <TD>contenu de la cellule 1-1</TD>
    <TD>contenu de la cellule 1-2</TD>
    <TD ROWSPAN=2 center>contenu des cellules 1-3 et
  <TR ALIGN="CENTER">
    <TH BGCOLOR="STEELBLUE">ligne 2</TH>
    <TD COLSPAN=2>contenu des cellules 2-1 et 2-2</TD></TR>
  </TR>
</TABLE>
après</BODY></HTML>
```





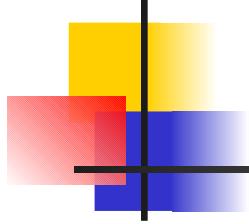
# Les cadres (frames) (1)

Le tag principal: FRAMESET  
remplace le tag BODY (<>...</>)

(pour le premier niveau)

Tag secondaire: NOFRAME  
indique le comportement à adopter dans le cas d'un browser NOFRAME

Tag secondaire: FRAME  
autant que de cadre du frameset  
décrit un cadre de la fenêtre et sa gestion



# Les cadres (2)

Les attributs de FRAMESET:

Rows, cols: découpage du cadre en plusieurs sous-cadres

FrameBorder: définit la bordure du cadre principal (yes | no)

Les attributs de FRAME:

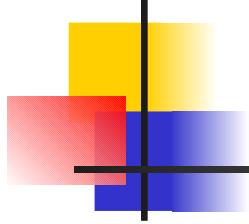
Src: Url de la page à charger par défaut dans cette fenêtre

Name: nom de la fenêtre

Scrolling: défilement ou non

Noresize: interdit de modifier la taille

Frameborder: bordure des fenêtres



# Les cadres (3)

```
<FRAMESET ...>
  <NOFRAME>
    Comportement pour les browsers noframe
  </NOFRAME>
  <FRAME...> Description par attribut du cadre 1
  ...
  <FRAME...> Description par attribut du cadre n
</FRAMESET>
```

Hierarchie de cadres :

On peut inclure des balises Frameset au sein d'un frameset

# Exemple de cadres

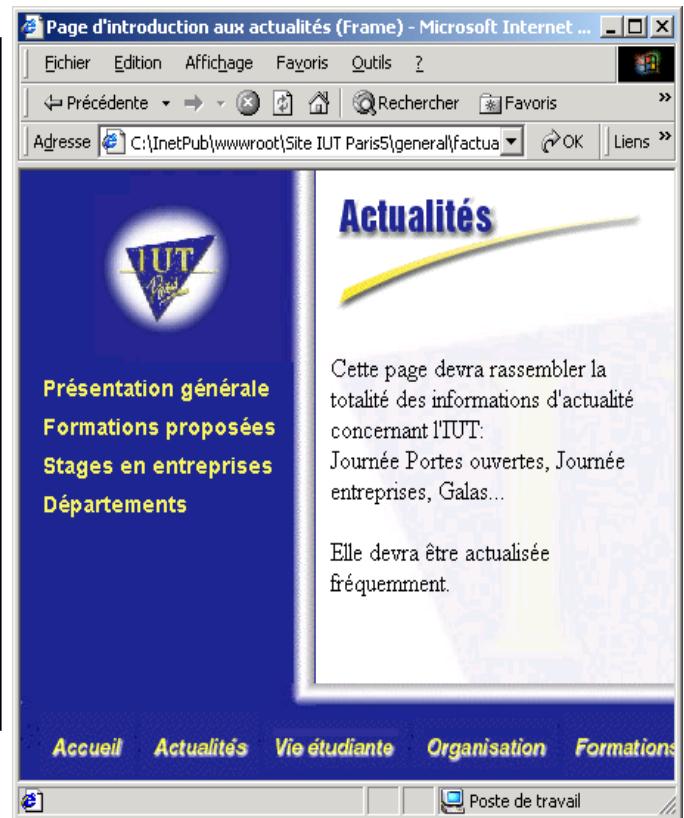
```
<html><head></head>

<frameset border="0" frameborder="0" rows="*,60">

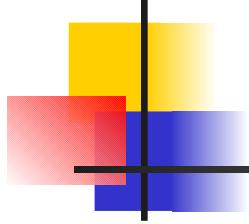
    <frameset border="0" frameborder="0" cols="200,*">
        <frame src="frame_iut.htm" scrolling="no">
        <frame src="actualites.htm" name= "fr1">
    </frameset>

    <frame src="../../site IUT Paris5/general/frame_bas.htm"
          scrolling="no">

    <noframes>
        <body> pas de frames ? </body></noframes>
</frameset></html>
```



*Ecrire en plus les pages citées dans les frameset*



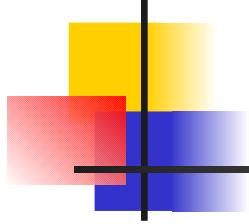
# Liens et cadres

```
<A HREF= "page" target="fenetre">
```

Indique le nom du cadre servant pour l'affichage de la page

Valeur prédéfinie de target:

- \_top : niveau le plus haut (peu éliminer ainsi un découpage en frame)
- \_parent niveau supérieur
- \_self même cadre
- \_blank nouvelle fenêtre



# Entête de pages HTML, tag META

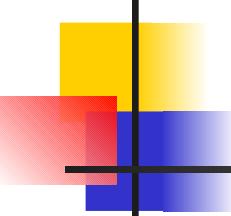
```
<META NAME="nom-de-la-propriété" CONTENT="valeur" >
```

author  
keywords  
description

Permet de décrire des propriétés générales du document  
Exploiter par les moteurs de recherche pour indéxer les pages et connaître les changements

```
<META HTTP-EQUIV="refresh/expires" CONTENT="précision" >
```

Force un changement de page : 'slide show', redirection de pages ...



# tag META

Les moteurs de recherche  
utilise les mots-cléfs pour indéxer les pages

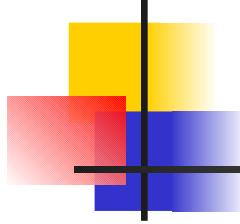
```
<HTML><HEAD><TITLE> mon entête préférée</TITLE>
<META NAME= "keywords" CONTENT="entête méta">
<META NAME= "authors" CONTENT="ifall">
<META NAME= "description" CONTENT="essais des tags de type méta">

<META HTTP-EQUIV= "refresh" CONTENT="5" ; URL=http://www.autre-page.html>

<META HTTP-EQUIV= "expires" CONTENT="Tues, 26 Sept 2002 09:30:02 gmt">
</HEAD><BODY></BODY></HTML>
```

Après la date d'expiration (26 sept2002) :  
oblige le gestionnaire du cache à recharger la  
page à partir du serveur web.

Après expiratión (5s) :  
Oblige le navigateur à redemander  
une URL (autre page à charger, ...)



# Modèle de document Cascading Style Sheet (css)

*Écrit une fois, Afficher partout* (HTML 4.0)

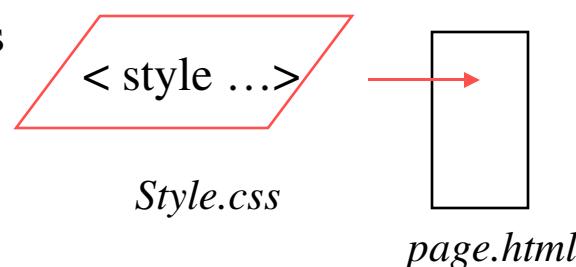
## Définition de styles réexploitables

définition a priori des attributs des balises (*règles css*)  
Réexploitation implicite ou par utilisation identifiant  
Héritage et surcharge en cas de multi définitions

## Localisation

En entête de page

Inclusion de fichiers définissant de styles



*Utilisation imposée dans XHTML*

# Règles de définition de styles

## Définition des styles

### Définition directe

```
<HEAD><STYLE TYPE="text/css"> ... tag_stylé ... </STYLE></HEAD>
```

### Définition de classes

```
tag_stylé.nomClasse {attribut: val; ...;}
```

### Définition pour des ID

```
tag#nomID {attribut: val; ...;}
```

```
*#nomID {attribut: val; ...;}
```

Dans le document HTML :

- Les classes (CLASS) diffèrentent les styles à appliquer pour un même type de balise
- Les identifiants (ID) sont associés à des balises (éventuellement de type différent)
- CLASS et ID peuvent jouer le même rôle mais on peut en combiner l'utilisation
- il y a héritage des styles en cas d'imbrication des balises

# Règles de définition de styles: exemple

```
<HTML><HEAD>

<STYLE TYPE="text/css">
H1 { color: blue; text-decoration: underline; }
H2 { color: red; text-decoration: none; }
</STYLE></HEAD>

<BODY>
<H1>Les titres de 1er
    niveau sont bleus et
    soulignés</H1><BR>
<H2>Les sous titres sont rouges
    et en italique</H2><BR>

<P STYLE="color: green; text-decoration: none;">
    Et les paragraphes sont verts, si je veux.
</P>

</BODY></HTML>
```



# Règles de définition de styles, exemple 2



<HTML><HEAD>

```
<STYLE TYPE="text/css">
  P { color: black; text-decoration: none; }
  <! définition de la classe plain pour la balise P>
  P.couleur {color: blue}
  <! définition de style pour toutes les balises d'ID=ital>
  *#ital {color: red; font-style: italic;} </STYLE></HEAD>
```

<BODY>

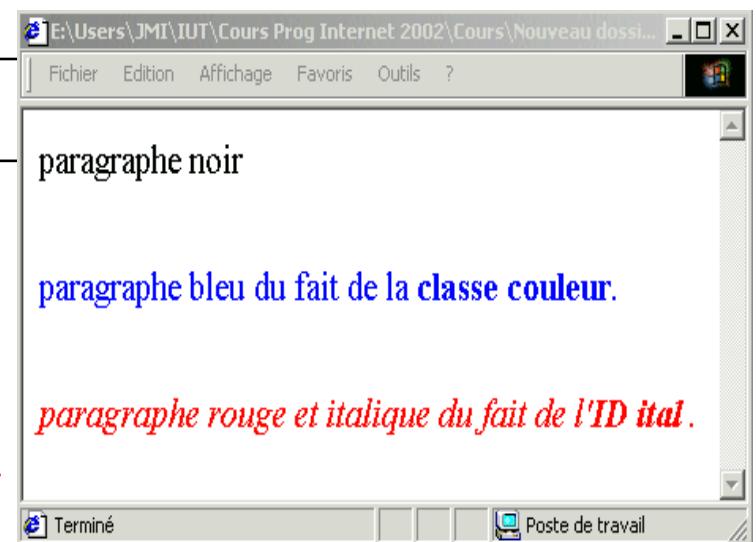
<P>paragraphe noir</P><BR>

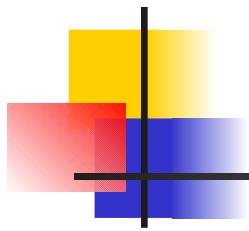
<P CLASS="couleur">

paragraphe bleu du fait de la <b>classe couleur</b>. </P><BR>

<P ID="ital">paragraphe rouge et italique du fait de l'<b>ID ital</b>. .</P>

</BODY></HTML>



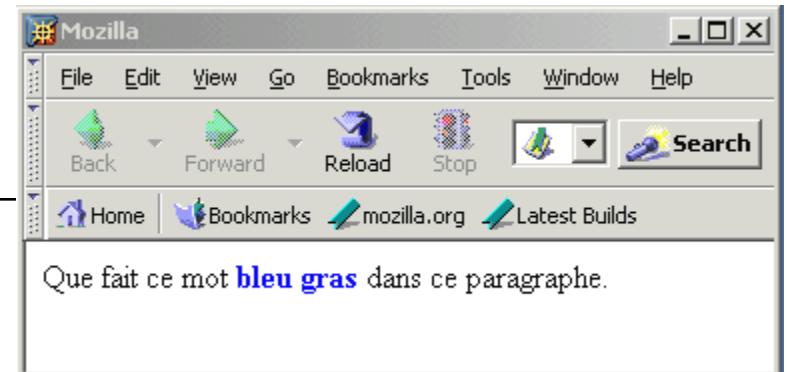


# Application de styles à un bloc de texte : l'élément SPAN

## Dans le document HTML

<SPAN> .... Texte à styliser .... </SPAN>

```
<HTML>
<BODY>
Que fait ce mot
<SPAN Style= "color:blue; font-weight:bold;"> bleu gras </SPAN>
dans ce paragraphe.
</BODY></HTML>
```



- La balise <DIV>... </DIV> permet aussi d'isoler un bloc de texte
- Le style d'une balise SPAN peut-être spécifié par l'utilisation de CLASS ou ID

# Inclusion de modèles de style

## Inclusion par balise LINK (dans l'entête)

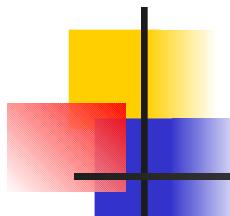
```
<HEAD> ...<LINK rel="style-sheet" type="text/css" href="une_url" >  
</HEAD>
```

```
<HTML><HEAD>  
<LINK rel=stylesheet type="text/css"  
      href="http://.../modele.css">  
</HEAD>  
  
<BODY>...</BODY></HTML>
```

## Inclusion par mot-clé @IMPORT

```
<HEAD><STYLE>  
  @IMPORT url ("une_url")  
</HEAD></STYLE>
```

```
<HTML><HEAD>  
<STYLE TYPE="text/css">  
  @IMPORT url (http://.../modele.css);  
...</STYLE> </HEAD>  
  
<BODY>...</BODY></HTML>
```



# Formatage des éléments de style

**bloc :** un document est vu comme un ensemble de blocs à formater  
chaque bloc est associé optionnellement à une hiérarchie de zones

margin

border

padding

Bloc contenu

*Padding : espace entre le contenu et les bords*

**paramètre de style : DISPLAY**  
= "block" //défini un niveau de bloc  
= "inline" //inclus dans le bloc  
*parent*  
= "none" //pas d'affichage

**Spécification par défaut des blocs**

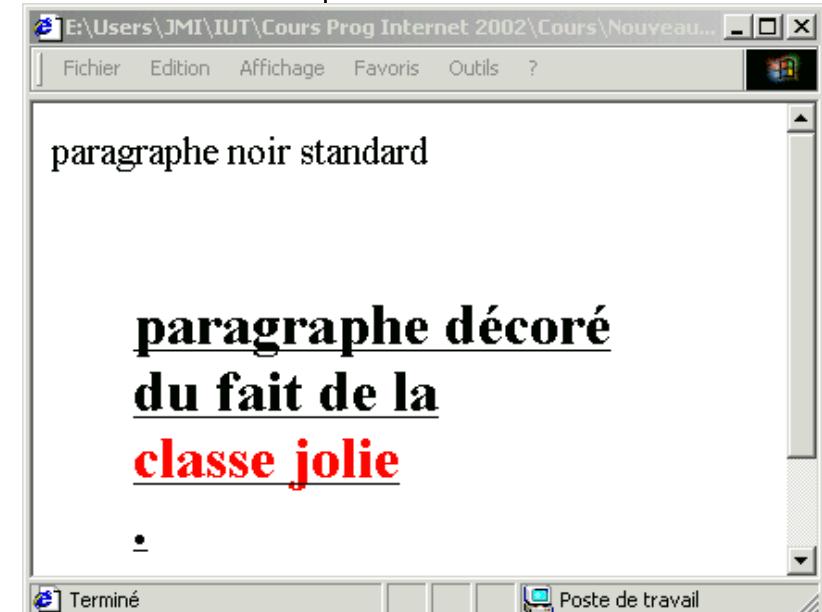
P -> bloc

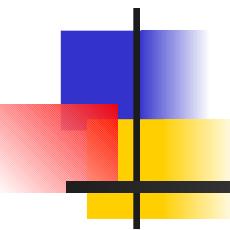
I ou B -> inline

# Formatage des éléments de style exemple

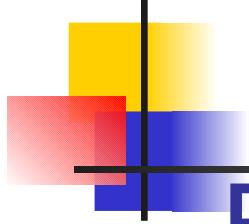
```
<HTML><HEAD>
<TITLE>Sélection des styles par class ou d'ID</TITLE>
<STYLE TYPE="text/css">
P {color: black; }
P.jolie { font-family: Times new roman;
          font-size:24pt;
          font-weight:bold;
          text-decoration:underline;
          display:block;
          padding:44}
B{color: red; display: block; }
</STYLE></HEAD>

<BODY>
<P>paragraphe noir standard</P>
<P CLASS="jolie">paragraphe décoré du fait de la <b>classe jolie</b>.</P>
</BODY></HTML>
```

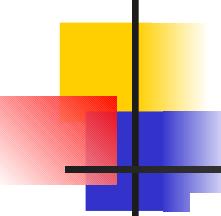




# ***SCRIPT CLIENT: JAVASCRIPT (SURVOL)***



# Elements du langage



# Introduction

## ■ JavaScript

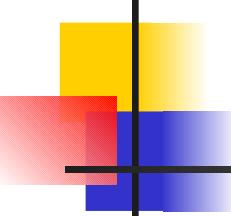
- Langage de script incorporé dans le HTML
- Historiquement, premier langage de script pour le Web
- Apporte des améliorations au HTML
  - HTML permet d'écrire
  - JavaScript permet de programmer, c'est-à-dire de gérer l'information

## ■ Qualités :

- Disponible sur les navigateurs actuels et gratuit

## ■ Défauts :

- Interprété et donc très lent, pas de débogueur



# Introduction

## ■ Code interprété ou compilé ?

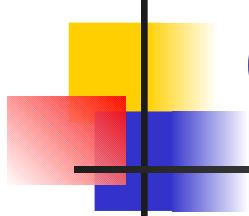
- Dès que le navigateur rencontre la balise `<script>` il passe la main à l'interpréte du langage appelé
- Votre navigateur interprétera votre script, puis l'exécutera

## ■ Que mettre dans le script ?

- Des variables et instructions, organisées selon votre algorithme, c'est-à-dire selon le résultat que vous souhaitez obtenir

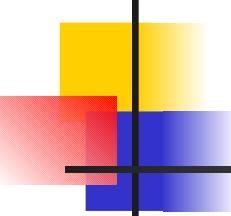
## ■ Où placer les scripts ?

- **Au début, dans le conteneur `<body>...</body>`**
  - Dès que vous serez capables de comprendre les fonctions, nous les placerons le plus souvent dans le conteneur `<head>...</head>`



# Commentaires

- Pour mettre en commentaires toute une ligne, on utilise le double slash:  
*// Tous les caractères derrière le // sont ignorés*
- Pour mettre en commentaire une partie du texte (éventuellement sur plusieurs lignes) on utilise le /\* et le \*/:  
*/\* Toutes les lignes comprises entre ces repères sont ignorées par l'interpréteur de code \*/*



# Lire/Ecrire

## ■ **prompt()**

- ouvre une boîte de dialogue avec une zone de saisie et 2 bouton : OK et Annuler, rend l'information lue

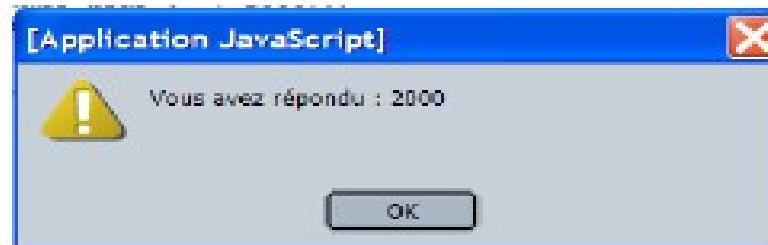
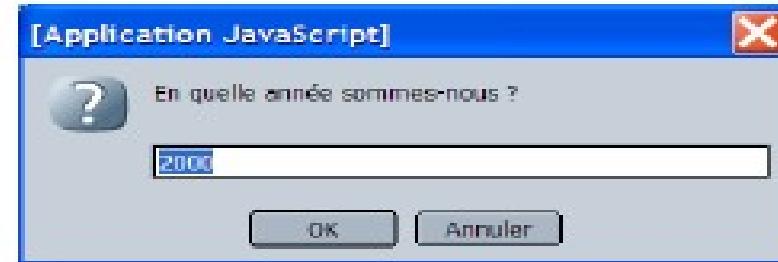
## ■ **alert ()**

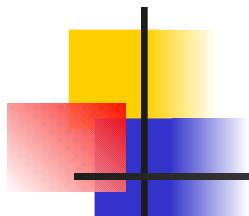
- permet d'écrire un message dans une fenêtre
- Exemple : [lire-ecrire.html](#)

```
<html>
<head><title>Programme In1</title></head>
<body>
  <script language="JavaScript">
    <!--
      annee=prompt('En quelle année sommes-nous ? ', 2000);
      alert("Vous avez répondu : " + annee);
    //-->
  </script>
</body>
</html>
```

# Lire/Ecrire)

## ■ Résultat



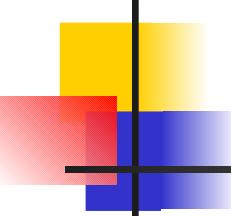


# Les cadres (2)

## ■ **document.write :**

- Permet d'écrire directement dans la fenêtre HTML
- Exemple : lire-ecrire2.html

```
<html>
  <head>
    <title>Programme Out1</title>
  </head>
  <body>
    <script language="JavaScript">
      <!--
        document.write('Vous avez le bonjour de
        JavaScript <br>');
      //-->
    </script>
  </body>
</html>
```



# Lire/Ecrire

- **document.write pour écrire le contenu de variable**

Exemple : lire-ecrire-var.html

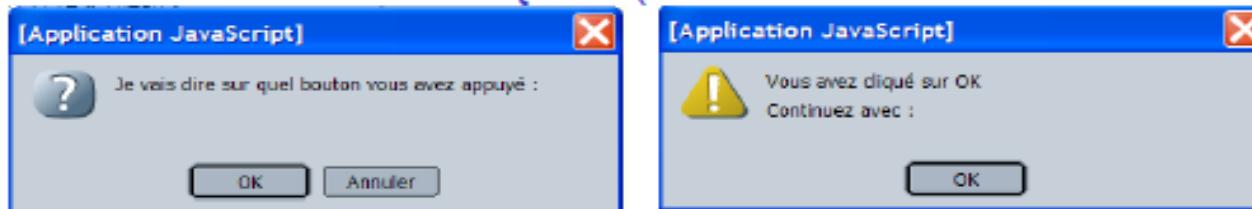
```
<html>
  <head><title>Programme Out2</title></head>
  <body>
    <script language="JavaScript">
      <!--
        var jour = 21;
        var mois = 'juin';
        document.write(jour + ' ' + mois + ' : solstice');
      //-->
    </script>
  </body>
</html>
```

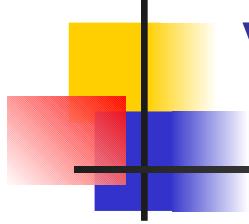
# Lire/Ecrire

## ■ Confirm ()

- Cette méthode ouvre une boîte de dialogue avec 2 boutons : OK et Annuler. Elle permet :
  - d'envoyer une information, de recevoir un booléen
- Exemple : lire-ecrire-confirm.html

```
<script language="JavaScript">  
    if (confirm('Je vais dire sur quel bouton vous  
    avez appuyé : '))  
        {alert(' Sur OK \n Continuez avec :') }  
    else {alert(' Sur Annuler \n Sortez avec Ok !') };
```

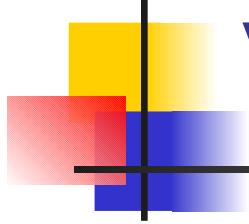




# Variables

## ■ Déclaration

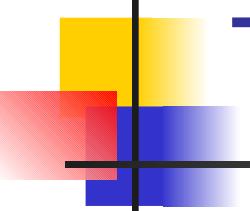
```
<script language="JavaScript">
    var date; // Déclaration sans affectation
    var compteur=0; // Déclaration avec affectation
    toto='coucou'; // Déclaration implicite par affectation
    var prem, second; // variables séparées par des virgules
</script>
```



# Variables

## ■ Déclaration

```
<script language="JavaScript">
    monNombre = new Number(); // Déclaration typée sans
                                affectation
    e = new Number(2.71828); // Déclaration typée avec
                            affectation
    var maChaine = new String(); //Déclaration de chaîne
    var toto = new Boolean(true); //Déclaration de booléen
</script>
```

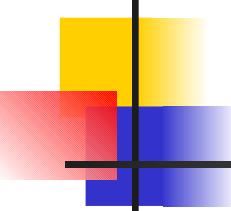


# Tableaux

## ■ Array

- Le type des éléments : nombres, chaînes, booléens, ...
- La dimension 1, 2, ou 3, ... : tab(7) ; tab(x,y) ; tab(A,B,C) ; ...
- Les indices : souvent des nombres entiers
- Exemple : tableau.html

```
<script language=JavaScript>
    // Tableau de chaînes, de dimension 1, indicé de 0 à 6 :
    Jour=new Array(7);
    Jour[0]='Dimanche' ;
    Jour[1]='Lundi' ;
    Jour[2]='Mardi' ; //...
    Jour[6]='Samedi' ;
    // En énumérant les éléments :
    jour=new Array('dimanche','lundi','mardi', ...
    , 'vendredi','samedi');
    document.write(Jour[1], ' ', Jour[0]); //affiche : Lundi
                                         dimanche
</script>
```



# Opérateurs

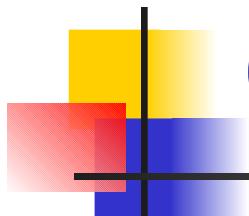
## ■ Arithmétiques

### Binaires :

+ Addition	$2 + 3$ ; compteur $+ 1$ . + opère aussi sur les chaînes. C'est un opérateur de concaténation : 'bon'+'jour'=='bonjour'
- Soustraction	$2 - 3$ ; rebours - 1
* Multiplication	$2 * 3$ ; $a * b$
/ Division	$2 / 3$ ; $a / b$ Attention : nombre/0 rend null
% Modulo	$13 \% 5$ ; Reste dans la division euclidienne $13 \% 5 = 3$

### Unaires :

- Opposé	<code>-monSolde</code>
++ Incrémentation	<code>i++</code> ; équivaut à <code>i = i + 1</code> (Pratique) <code>++i</code> ; l'incrémentation est faite avant d'utiliser la valeur de <code>i</code>
-- Décrémentation	<code>k--</code> ; équivaut à <code>k = k - 1</code> <code>--k</code> ; la décrémentation est faite avant d'utiliser la valeur de <code>k</code>



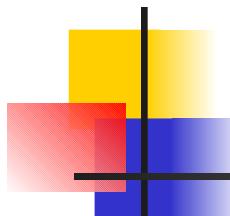
# Opérateurs

## ■ De comparaison

- La comparaison se fait entre deux objets de même type
  - renvoie un booléen : true ou false

Binaires :

<	Inférieur strict	$x < 2$ ; compteur < 101 'femme' < 'homme' renvoie true
>	Supérieur strict	$x > 2$ ; compteur > x 'chat' > 'chien' renvoie false
$\leq$	Inférieur large	$x \leq 2$ ;
$\geq$	Supérieur large	$x \geq 2$ ;
$\equiv$	Egal ("identique à")	$x \equiv 2$ ;
$\neq$	Different	$x \neq 2$ ;



# Opérateurs

## ■ Logique

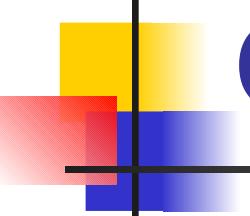
- Opèrent sur des booléens et renvoient un booléen

### Binaires :

&&	ET	bool1 && bool2 true uniquement dans le cas : true && true
	OU	bool1    bool2 false uniquement dans le cas : false    false

### Unaire :

!	NON	! bool1 ! true = false ; ! false = true
---	-----	--



# Opérateurs

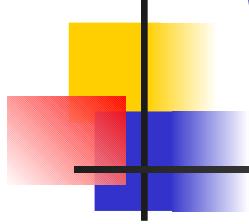
## ■ d'affectation

### Affectation simple :

=	Affectation simple	Total = HT+TVA ; JavaScript évalue HT+TVA puis le range dans Total Attention : 2 = toto+1 n'est pas une affectation, et pire n'a pas de sens. Quant à 2 == toto c'est un booléen vrai ou faux
---	--------------------	---

### Affectation arithmétique :

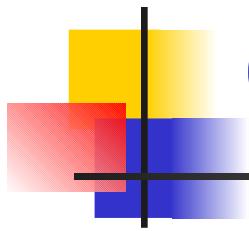
+=	Addition	x += y signifie x = x + y
-=	Soustraction	x -= y signifie x = x - y
*=	Multiplication	x *= y signifie x = x * y
/=	Division	x /= y signifie x = x / y
%=	Modulo	x %= y signifie x = x % y



# Conditionnelles

- **Syntaxe**

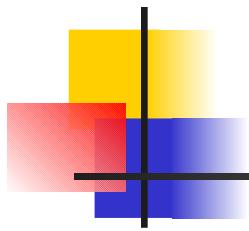
```
if (condition booléenne)
    {Instructions à exécuter si "oui"}
else
    {Instructions à exécuter si "non"};
```



# Choix

- Choix multiple

```
switch(x) {  
    case 1 : instructions 1; break;  
    case 2 : instructions 2; break;  
    ...  
    case n : instructions 3; break;  
    default : instructions 4; break;  
};
```



# Boucles

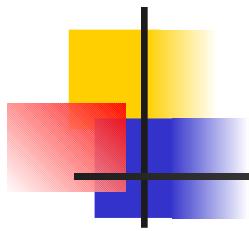
- **For**

```
for (valeur de départ ; contrôle pour sortie ; progression )  
{ Instructions à itérer }
```

- **while**

- Syntaxe :

```
while(condition) {  
    suite d'instructions;  
}
```



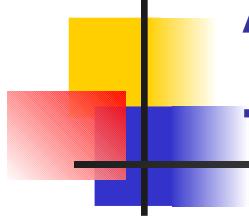
# Fonctions

- **Syntaxe**

```
function maFonction(x, toto) {  
    ... instructions; ...  
    return valeur du résultat;  
}
```

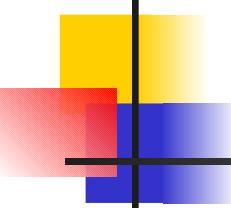
- **Exemple**

```
<script language="JavaScript">  
    function VolCylindre(r,h) {  
        pi=3.14159;  
        return pi*r*r*h;  
    }  
    document.write(VolCylindre(1,2));  
</script>
```



# Association avec formulaire

- Utilisation dans un formulaire
  - Schéma d'utilisation
    - Body :
      - ❖ Contient la définition du formulaire
      - ❖ Il fait appel aux variables et fonctions définies dans le head
    - Head :
      - ❖ Contient les fonctions



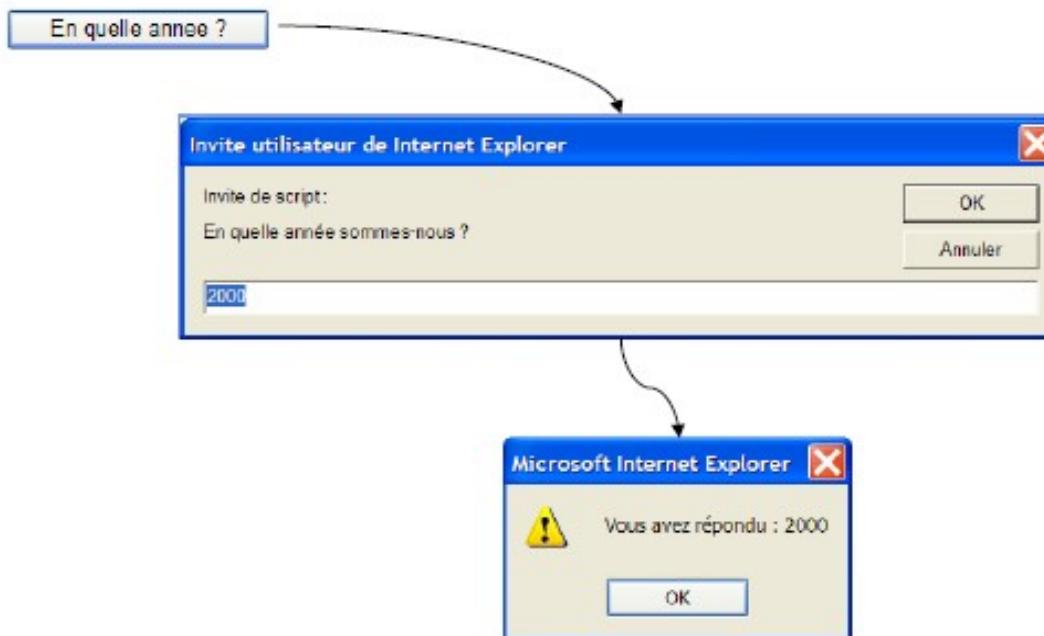
# Association avec formulaire

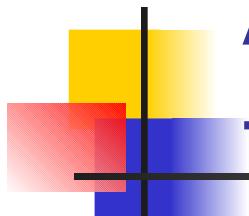
- Exemple 1 : appel à une fonction de lecture en cliquant sur un bouton du formulaire

```
<html>
    <head>
        <title>Programme In2</title>
        <script language="JavaScript">
            function lireAnnee() {
                annee=prompt('En quelle année sommes-nous ?', 2000);
                alert('Vous avez répondu : ' + annee) }
        </script>
    </head>
    <body>
        <form>
            <input type="button" value="En quelle année ? "
                  onClick="lireAnnee()">
        </form>
    </body>
</html>
```

# Association avec formulaire

- Exemple 1 :

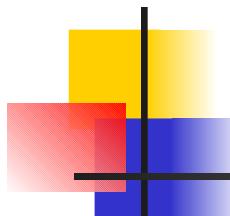




# Association avec formulaire

- Lire un nombre et écrire son double
  - Exemple 2 : Lire-ecrire-form.html
    - Saisie du nombre :  
`<input type="text" name="nbre" size="3">`
    - Traitement
      - ❖ `2*Number(document.lire.nbre.value);`

Entrez un nombre :   voici son double :



# Association avec formulaire

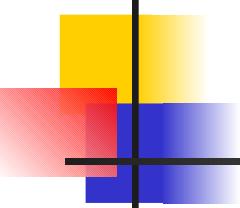
- Code source complet : ici le code js est dans le formulaire

```
<html>
<head><title>Programme In3</title></head>
<body>
    <form name="lire">
        Entrez un nombre :
        <input type="text" name="nbre" size="3">
        <input type="button" name="double" value="go"
               onClick="JavaScript:document.lire.aff.value=2*Num
               ber(document.lire.nbre.value);">
        voici son double : <input type="text" name="aff"
               size="8">
    </form>
</body>
</html>
```

Entrez un nombre :   voici son double :



# Interactions



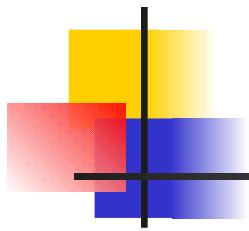
# Evènements

## ■ Présentation

- Les événements sont l'intérêt du JS en matière de programmation Web
- Ils donnent une **interactivité** à la page que vous consultez, ce qui n'existe pas avec le HTML, si on excepte bien entendu le lien hypertexte
- Le JS permet de réagir à certaines actions de l'utilisateur

## ■ Pour cela, on précise :

- L'événement (Event)
  - Clic de souris, survol de zones, chargement de la page...
- Le gestionnaire de l'événement (OnEvent)
  - `OnClick, onMouseOver...` : appel de fonction pour agir en conséquence



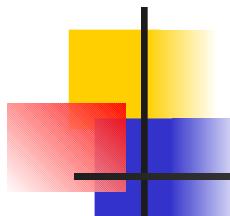
# Evènements

- Syntaxe du gestionnaire d'événement :

```
onEvent="Action_Javascript_ou_Fonction();"
```

- Lorsqu'il est utilisé dans un lien hypertexte, par exemple, la syntaxe sera la suivante :

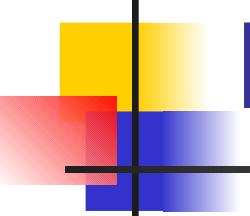
```
<A href="URL"  
"onEvenement='Action_Javascript_ou_Fonction();'"  
>Lien</a>
```



# Evènements

## ■ Le clic de souris

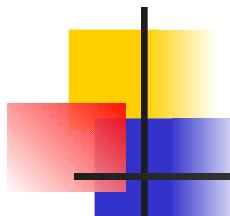
- **Gestionnaire d'événement**
  - onClick
- **Exemple :**
  - <input type="button" onClick="alert('vous avez cliqué sur le bouton') ;">
- **Balises supportées :**
  - <input type="button">
  - <input type="checkbox">
  - <input type="radio">
  - <input type="reset">
  - <input type="submit">
  - <a href..>



# Evènements

## ■ Le chargement

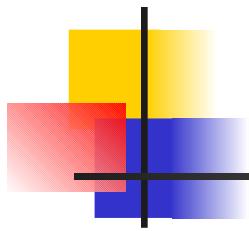
- **Gestionnaire d'événement**
  - . `onLoad`
- **Exemple :**
  - . `<body onLoad="alert('la page est chargée !') ;">`
- **Balises supportées :**
  - . `<body>`
  - . `<frameset>`
- **Effets :**
  - . Au chargement, réaliser tel événement



# Evènements

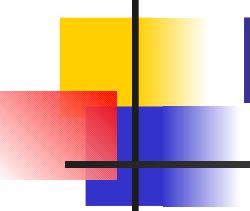
## ■ Error

- **Gestionnaire d'événement**
  - `onError`
- **Exemple :**
  - ``
- **Balises supportées :**
  - `<body>`
  - `<frameset>`
  - `<img>`
- **Effets :**
  - Prévient l'erreur au chargement



# Evènements

- **Abort**
  - **Gestionnaire d'événement**
    - `onAbort`
  - **Exemple :**
    - ``
  - **Balises supportées :**
    - `<img>`
  - **Effets :**
    - Prévient l'erreur au chargement



# Evènements

## ■ Le passage de la souris

- **Gestionnaire d'événement**

- onMouseOver

- **Exemples : onmouseover.html**

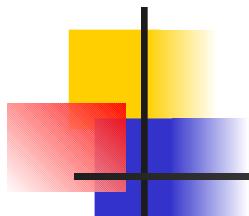
```
<div style="width:50; height:50; background:lightsteelblue;"  
    onMouseOver="alert('Le curseur entre dans la zone  
    bleue');"></div>  
<div> <P onMouseOver="this.style.color='red'"  
    onmouseout="this.style.color='black'"> Move the mouse pointer  
    over this text, then move it elsewhere in the document.</div>  
<a href="http://www.google.fr" onMouseOver="alert('Pour aller sur  
    google.fr, cliquer ici');">http://www.google.fr</a><a  
    href="http://www.google.fr" onMouseOver="alert('Pour aller sur  
    google.fr, cliquer ici') ;">http://www.google.fr</a>
```

- **Balises supportées :**

- Presque toutes

- **Effets :**

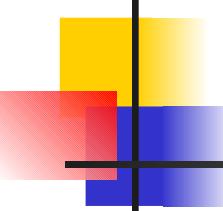
- Prévient quand on survole la cible : zone, texte, lien



# Evènements

## ■ Le passage de la souris

- **Gestionnaire d'événement**
  - onMouseOut
- **Exemples :**
  - <a href="http://www.google.fr" onMouseOut="alert('Vous ne voulez pas y aller ?');" ;>http://www.google.fr</a>
- **Balises supportées :**
  - Presque toutes
- **Effets :**
  - Prévient quand on s'éloigne de la cible

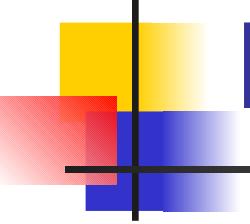


# Evènements

## ■ Le focus

- **Gestionnaire d'événement**
  - onFocus
  - se déclenche lorsque l'élément reçoit le focus (devient actif) soit par action de l'outil de pointage (souris), soit par la navigation tabulée (touches du clavier).  
L'événement onfocus est l'opposé de l'événement onblur.
- **Exemple :**

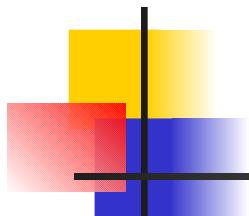
```
<input type="text" size="40" maxlength="50" name="pharma_name"
onFocus='javascript:this.style.backgroundColor="yellow"'
onMouseOver='javascript:this.style.background="yellow"'
onMouseOut='javascript:this.style.background="white";this.style.border="1"' onBlur='javascript:this.style.backgroundColor="white"'>
```
- **Balises supportées :**
  - Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
- **Effets : tester**



# Evènements

## ■ Le blur :

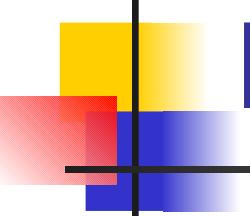
- Gestionnaire d'événement
  - onBlur
- Exemple :
  - <input type="text" onblur="this.size = 20" onfocus="this.size = 50">
- Balises supportées :
  - <input type="text">
  - <select>
  - <textarea>
  - <input type="password">
- Effets :
  - onBlur : Permet de savoir lorsque le champ perd le focus
  - OnFocus : Permet de savoir lorsque le champ a le focus



# Evènements

## ■ Les changements :

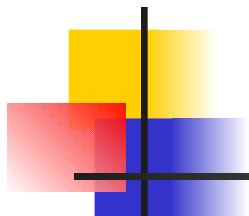
- **Gestionnaire d'événement**
  - onChange
- **Exemples :**
  - <input type="text" value="votre nom" name="nom" onChange="alert('vous avez changé votre nom')">
- **Balises supportées :**
  - <input type="text">
  - <select>
  - <textarea>
  - <input type="password">
- **Effets :**
  - Avertit du changement par rapport à ce qu'il y avait d'écrit dans la zone d'écriture



# Evènement

## ■ La sélection :

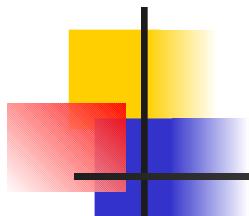
- **Gestionnaire d'événement**
  - onSelect
- **Exemples :**
  - <form>
  - Select text: <input type="text" value="Hello world!"
  - onselect="alert('You have selected some of the text.')">
  - <br /><br />
  - Select text: <textarea cols="20" rows="5"
  - onselect="alert('You have selected some of the text.')">
  - Hello world!</textarea>
  - </form>
- **Balises supportées :**
  - <input type="text">
  - <textarea>
- **Effets :**
  - Avertit de la sélection d'un champ



# Evènement

## ■ L'envoi :

- **Gestionnaire d'événement**
  - onSubmit
- **Exemple :**
  - <form name="testform" action="function()" onsubmit="alert('Hello  
' + testform.fname.value + '!')"> What is your name?<br />
  - <input type="text" name="fname" />
  - <input type="submit" value="Submit" />
  - </form>
- **Balises supportées :**
  - <input type="submit">
- **Effets :**
  - L'événement se produit quand le bouton de soumission est actionné



# Boucles

## ■ Le reset :

- **Gestionnaire d'événement**
  - onReset
- **Exemple :**
  - <input type="reset" value="Effacer" name="effacer" onSubmit="alert('On efface tout !');">
- **Balises supportées :**
  - <input type="reset">
- **Effets :**
  - En appuyant sur le bouton effacer, il remet dans la zone de texte : votre nom

votre nom  Effacer

# Exercice

- **Contrôle des champs de saisie**
  - Soit le formulaire suivant, proposer des fonctions permettant de contrôler la saisie de ces champs

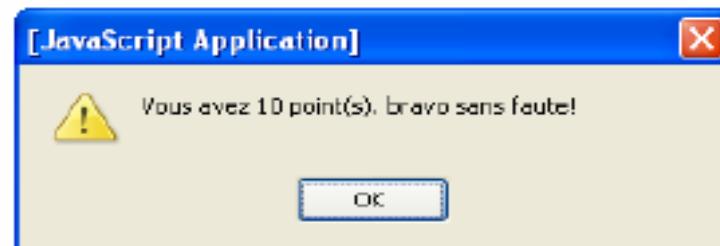
5 points par bonne réponse

Je m'appelle:

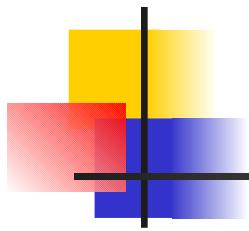
- Ed (bonne réponse)
- Rick
- Tom

Mon site s'appelle:

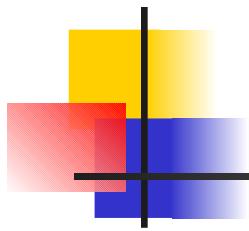
- Machin
- Javascript.Lab (bonne réponse)
- Truc



**RESULTATS**



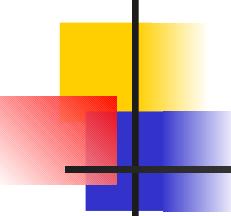
# Objets JS



# Introduction

## ■ Notion d'objet en Javascript

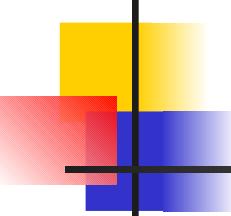
- Le Javascript traite les éléments qui s'affichent dans votre navigateur comme des objets, c'est-à-dire des éléments :
  - classés selon une hiérarchie pour pouvoir les désigner précisément
  - auxquels des propriétés et des actions (méthodes) sont associées



# Les objets

## ■ Comment JavaScript définit les objets ?

- Javascript divise la page du navigateur en éléments (objets), afin de permettre d'accéder à n'importe lequel d'entre-eux et de pouvoir les manipuler par l'intermédiaire de leurs propriétés
- On commence généralement par l'objet le plus grand (celui contenant tous les autres) puis on descend dans la hiérarchie jusqu'à arriver à l'objet voulu
  - L'objet le plus grand est l'objet fenêtre : **window**
  - Dans la fenêtre s'affiche une page, c'est l'objet **document**
  - Cette page peut contenir plusieurs objets, comme
    - ❖ des formulaires,
    - ❖ des images, etc.

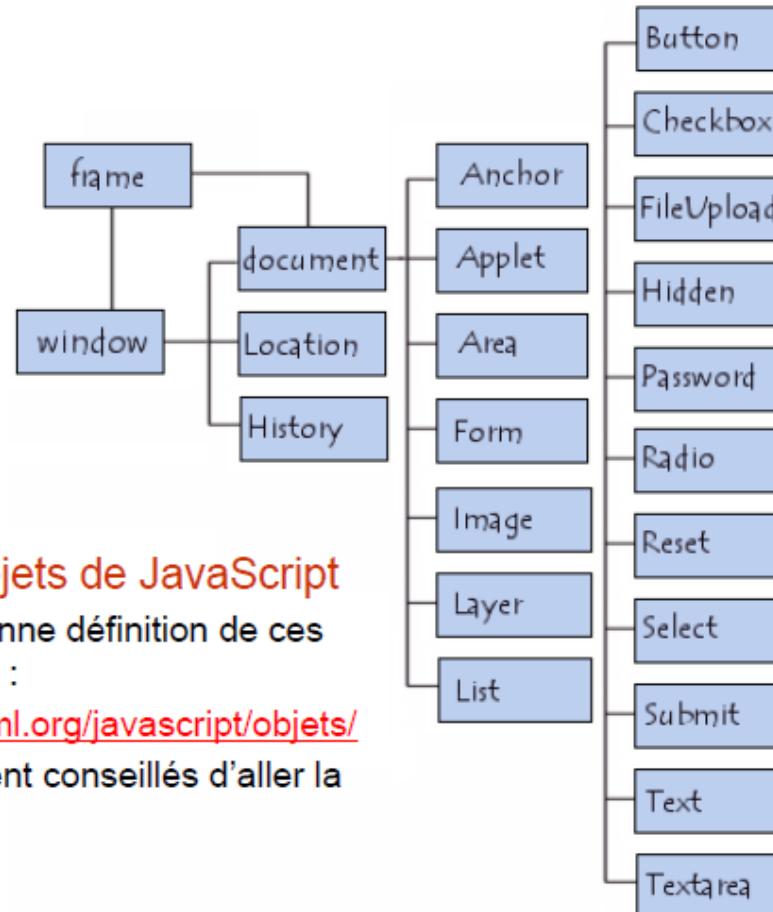


# Les objets

## ■ Les objets de base de JavaScript

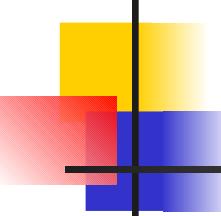
- navigator : qui contient des informations sur le navigateur de celui qui visite la page
- window : c'est l'objet où s'affiche la page, il contient donc des propriétés concernant la fenêtre elle-même mais aussi tous les objets-enfants contenus dans celle-ci
- location : contient des informations relatives à l'adresse de la page à l'écran
- history: c'est l'historique, c'est-à-dire la liste de liens qui ont été visités précédemment
- document : il contient les propriétés sur le contenu du document (couleur d'arrière plan, titre, ...)

# Les objets



## ■ Hiérarchie des objets de JavaScript

- On trouve une bonne définition de ces objets à l'adresse :
  - <http://fr.selfhtml.org/javascript/objets/>
- Vous êtes vivement conseillés d'aller la consulter

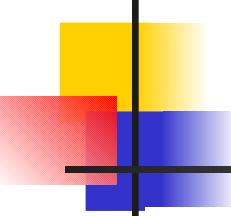


# Les objets

## ■ L'objet Navigator

- a plusieurs propriétés concernant votre navigateur
  - *appName* :
    - ❖ connaître le nom : *Netscape, IE, Mozilla*
  - *appVersion* :
    - ❖ connaître la version
  - *language* :
    - ❖ *FR, AN*
  - *platform* :
    - ❖ *windows, Linux...*
- Pour le savoir : exécuter :

```
<script language="Javascript">
    document.write(navigator.propriété);
</script>
```



# Les objets

## ■ L'objet Window

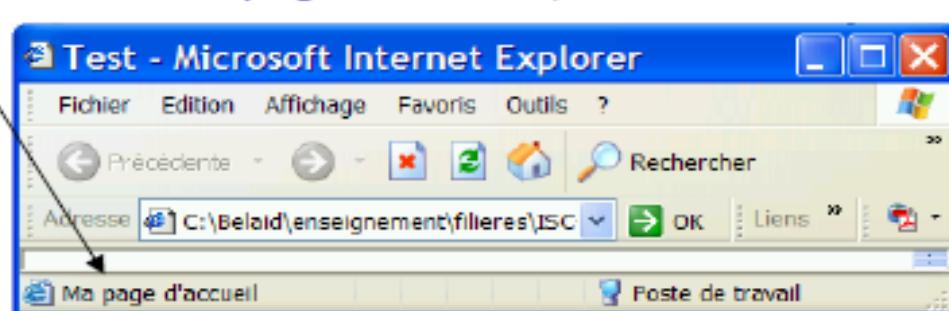
- est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web
- il contient donc :
  - l'objet document :
    - ❖ la page en elle-même
  - l'objet location :
    - ❖ le lieu de stockage de la page
  - l'objet history :
    - ❖ les pages visitées précédemment
  - l'objet frames :
    - ❖ les cadres (division de la fenêtre en sous-fenêtres)

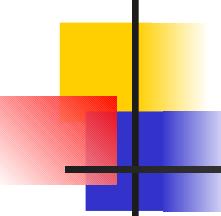
# Les objets

## ■ L'objet Window : Propriété defaultStatus

- affiche dans la barre d'état de la fenêtre d'affichage la valeur "Ma page d'accueil"
- Exemple : window0.html

```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
    window.defaultStatus = "Ma page d'accueil";
</script>
</head>
<body>
</body>
</html>
```



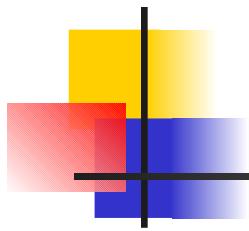


# Les objets

## ■ L'objet Window : la méthode open () :

- Cette fonction ouvre une nouvelle fenêtre, voici sa syntaxe :
  - `window.open("URL","nom_de_la_fenetre","options_de_la_fenetre")`
    - ❖ Ouvre « secondefenetre » et y affiche le fichier test.html.  
Secondefenetre peut être utilisé comme target pour l'affichage de l'extérieur
- Exemple : window01.html

```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
    function nouvellefenetre() {
        mafenetre = window.open("window0.html",
                               "secondefenetre", "width=300,height=200,scrollbars"); }
</script> </head>
<body>
<a href="javascript:nouvellefenetre()"> nouvelle fenêtre </a>
</body></html>
```

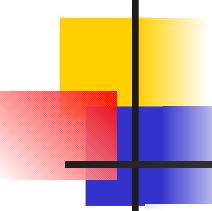


# Les objets

## ■ L'objet Window : la méthode close () :

```
<a href="javascript:mafenetre.close()">fermer la  
fenêtre</a>
```

- En cliquant sur ce lien, cela ferme la fenêtre précédemment ouverte avec le nom « mafenetre »

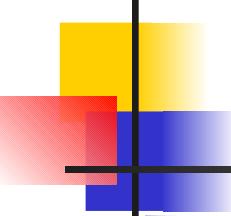


# Les objets

## ■ L'objet Window

- Vérification de l'état de la fenêtre
- Exemple : window2.html

```
<html><head><title>Test</title>
<script type="text/javascript">
<!--
    var InfoWin = window.open("fichier1.htm", "secondefenetre");
    function CheckOpen() {
        if(InfoWin.closed == true) alert("La fenêtre a été fermée"); else
        alert("La fenêtre est encore ouverte"); }
    //-->
</script>
</head>
<body>
    <a href="javascript:CheckOpen()">La fenêtre est-elle
    fermée?</a>
</body>
</html>
```

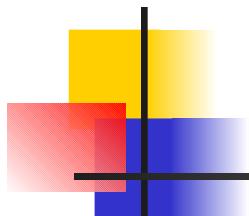


# Les objets

## ■ L'objet Window

- Fermeture automatique d'une fenêtre, après 2'

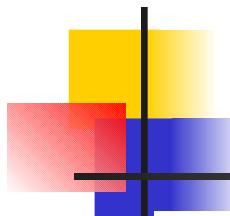
```
<html>
    <head>
        <title>Test</title>
        <script type="text/javascript">
            var InfoWin = window.open("exercice1.html",
                "secondefenetre");
            InfoWin.setTimeout("top.close()",2000);
        </script>
    </head>
    <body>
    </body>
</html>
```



# Les objets

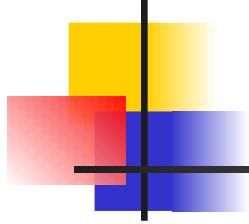
## ■ L'objet document

- L'objet document est un élément majeur, il va vous permettre de récupérer des informations d'un formulaire, créer des calques et les déplacer, écrire du texte....
- Propriétés :
  - `document.fgColor`, permet de récupérer et de changer la couleur du texte de votre page HTML  
`document.fgColor = "#993333";`
  - `document.bgColor`, permet de récupérer et de changer la couleur de fond de votre page HTML
  - `document.lastModified`, permet de savoir quand la page html a été modifiée
    - ❖ `document.lastModified;`



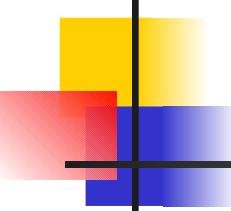
# Les objets

- `document.linkColor`
  - ❖ permet de récupérer et de changer la couleur des liens de votre page HTML
- `document.location`
  - ❖ permet de récupérer et changer l'url de votre page HTML, ce qui revient à charger une autre page HTML
  - `document.location = "URL/monDoc.HTML";`
- `document.write()`
  - ❖ permet d'écrire dans votre page HTML
- `document.images[ ]`
  - ❖ permet de récupérer et changer les images de votre page HTML
- `document.forms[ ]`
  - ❖ permet de récupérer et changer les informations de votre formulaire



# Les objets

- L'objet document :
  - Plusieurs exemples  
<http://fr.selfhtml.org/javascript/objets/document.htm>

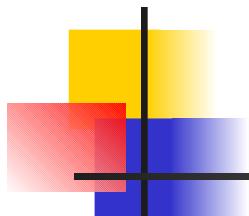


# Les objets

## ■ L'objet document : document1.html

- L'exemple fait un fondu enchaîné depuis le noir en passant par les nuances de gris jusqu'au blanc
- La fonction setColor() est appelée avec à chaque fois un délai de **20 millièmes** de seconde grâce à setTimeout

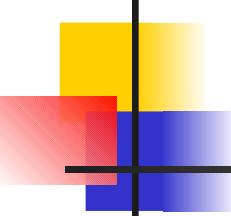
```
<script type="text/javascript">
var X = new
    Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F");
var x1 = 0, x2 = 0;
function setColor() {
    document.bgColor ="#" + X[x1] + X[x2] + X[x1] + X[x2] + X[x1] +
    X[x2]; x2 = x2 + 1;
    if(x2 % 16 == 0) { x2 = 0; x1 = x1 + 1; }
}
for(var i = 0; i < 255; ++i)
    { window.setTimeout("setColor()",20); }
</script>
```



# Les objets

## ■ L'objet : forms

- Avec l'objet **forms**, qui se trouve sous l'objet **document** dans la hiérarchie JavaScript, vous avez accès aux formulaires définis dans un fichier HTML
- Syntaxe :
  - `document.forms["nom_formulaire"].propriété`
  - `document.forms["nom_formulaire"].méthode`
- Plusieurs exemples sous :
  - <http://fr.selfhtml.org/javascript/objets/forms.htm>



# Les objets

## ■ Exemple : forms1.html

- Il s'agit d'accéder à la case à cocher pour modifier le contenu de la zone du texte en inscrivant : case cochée ou case non cochée
- La modification se fera par la fonction **ModifChamp()**;
- On déclare la case à cocher et la zone texte comme suit :

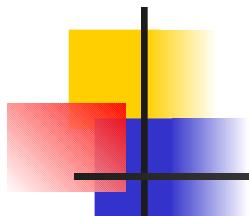
```
<form name="form1"> <br>
<input type="checkbox" name="checkbox"
       onClick="ModifChamp();"> <br>
<input type='TEXT' name='champ_text' value='Essai du
       javascript' size='24'>
</form>
```



Bouton coché



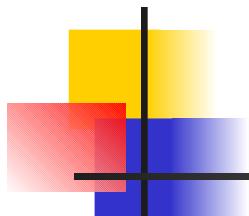
bouton non coché



# Les objets

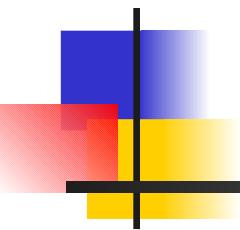
- Ensuite, on se réfère à la case à cocher et à la zone de texte à travers forms :

```
<script language="Javascript">  
function ModifChamp() {  
    if (document.forms["form1"].checkbox.checked) {  
        document.forms["form1"].champ_text.value='Bouton  
coché' }  
    else {  
        document.forms["form1"].champ_text.value='bouton  
non coché' } }  
</script>
```



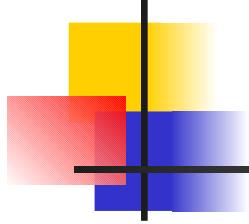
# Les objets

- **Le champ form a plusieurs propriétés :**
  - **Action ()**
    - Définit l'URL où le formulaire sera envoyé
  - **Elements**
    - Tableau représentant les éléments du formulaire
  - **Length**
    - Nombre d'éléments à l'intérieur du formulaire
  - **Method**
    - Définit le type d'envoi du formulaire (get ou post)
  - **Name**
    - Définit le nom du formulaire
  - **Target**
    - Définit la page (fenêtre ou frame) de réponse
  - **Parent**
    - Indique une fenêtre d'un cadre (frame)



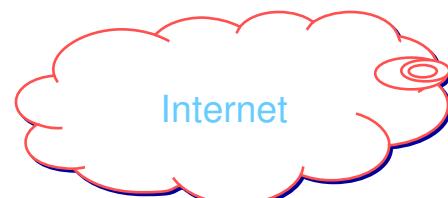
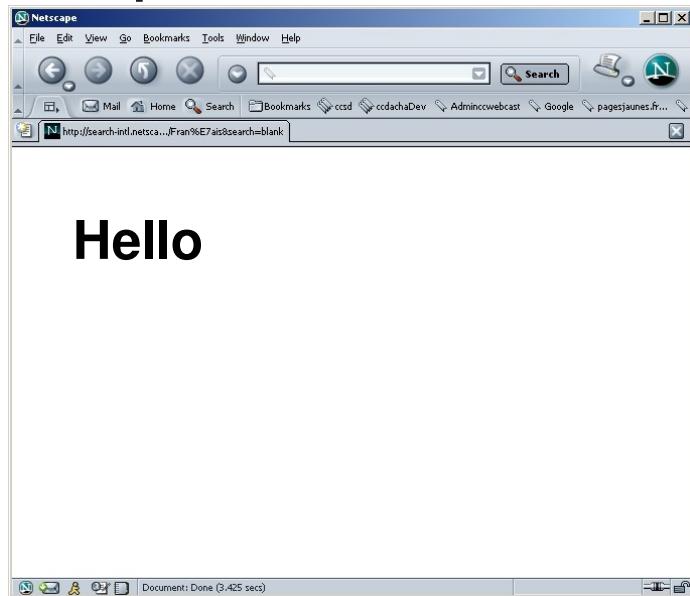
# ***SCRIPT SERVEUR: PHP***

---



# Introduction

# Le modèle HTML statique



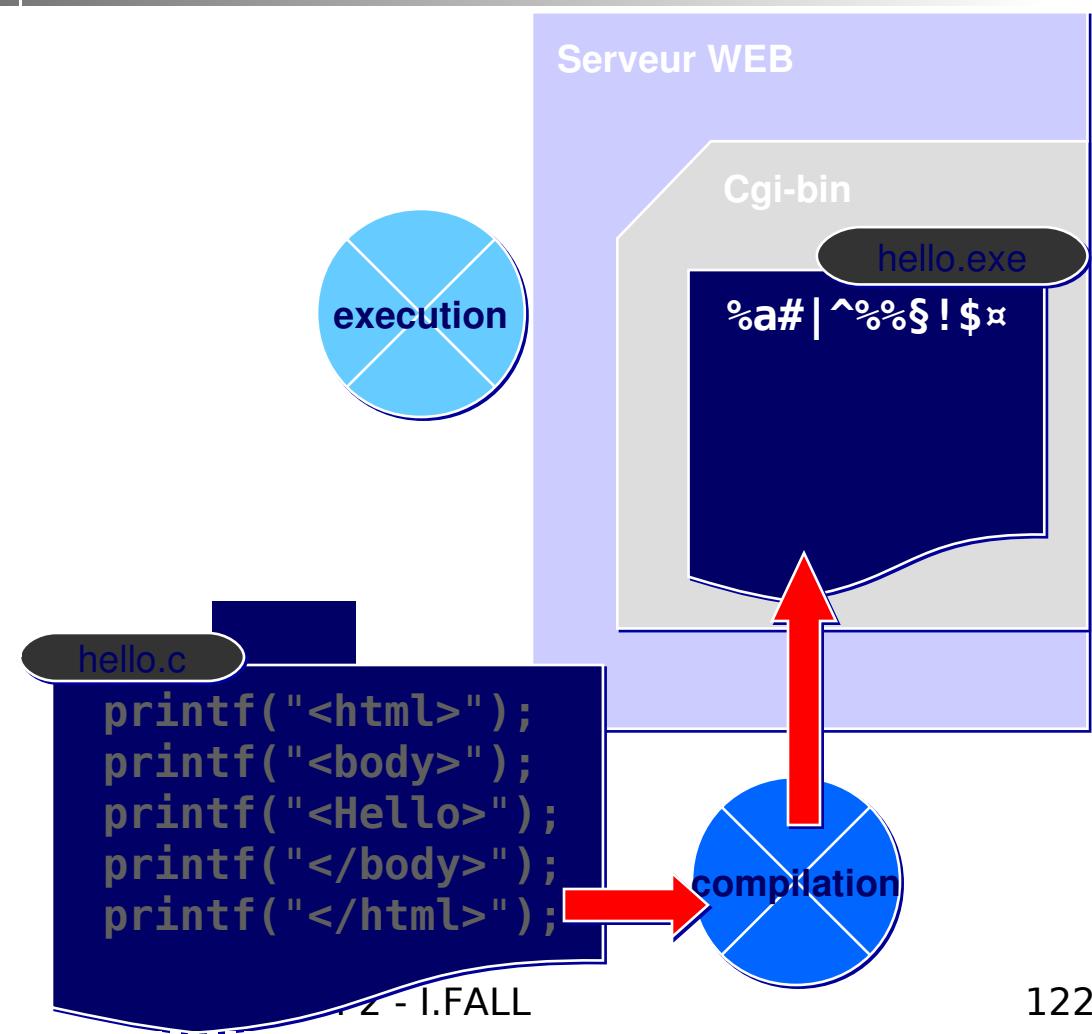
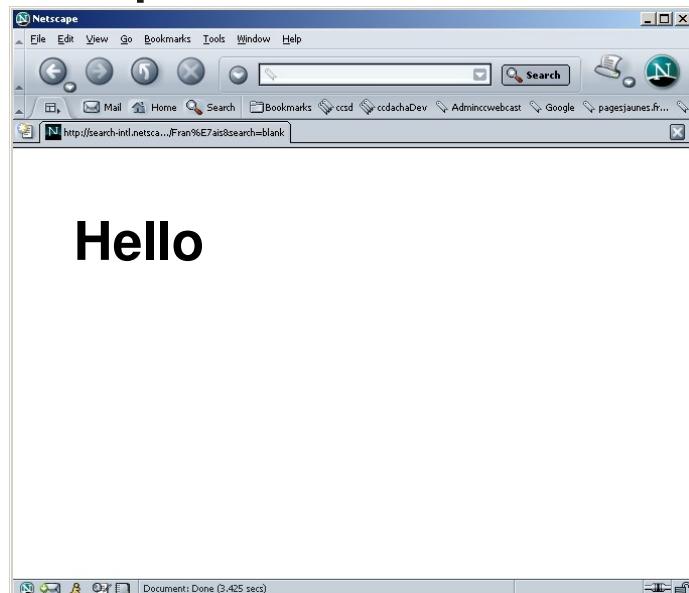
Serveur WEB

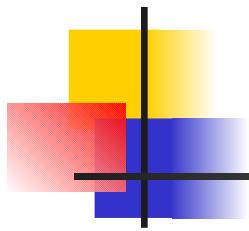
htdocs

Fichier hello.html

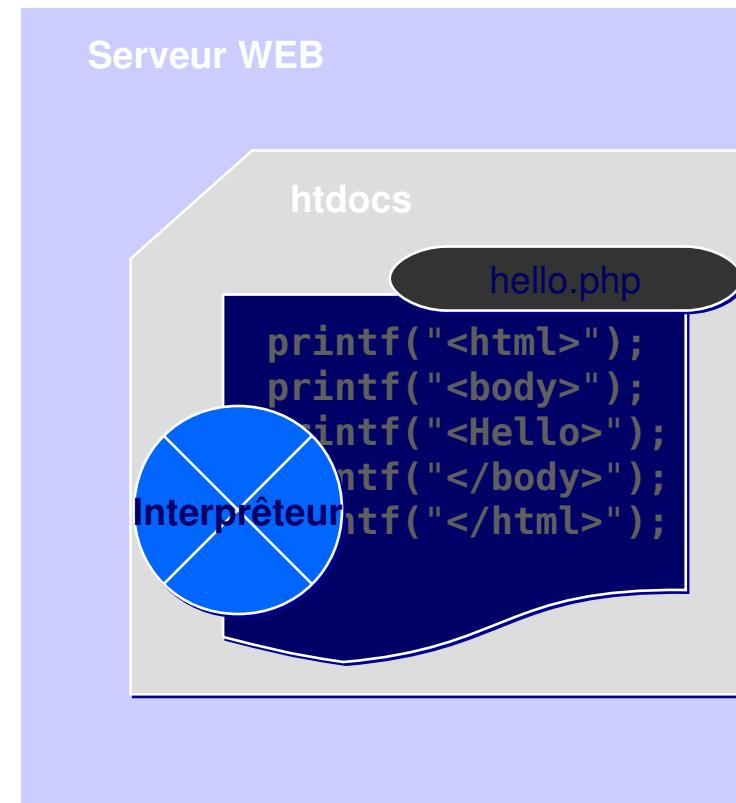
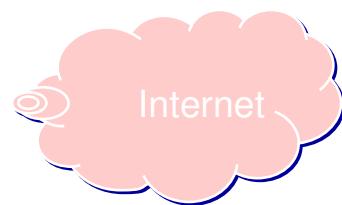
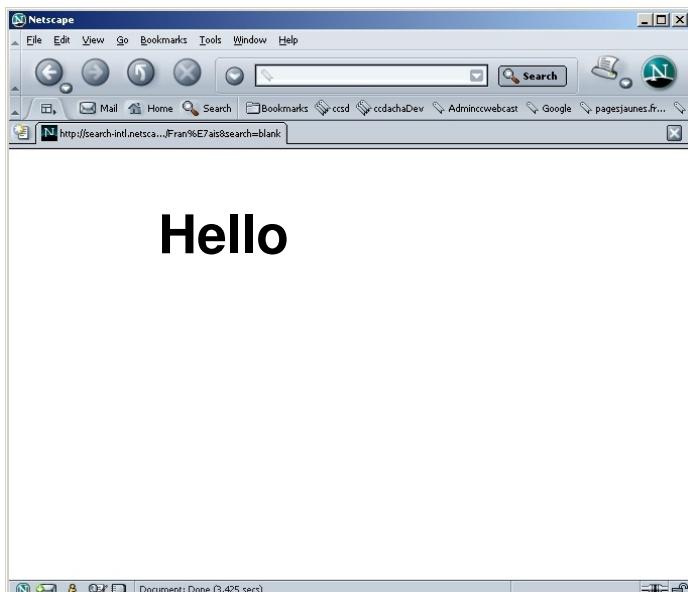
```
<html>
<body>
Hello
</body>
</html>
```

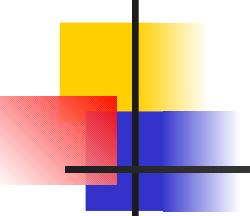
# Le modèle HTML dynamique avec les CGI





# Le modèle HTML dynamique avec un interpréteur

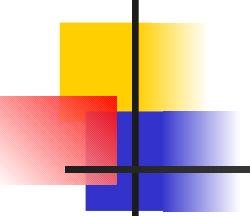




# Présentation fonctionnelle

Une application Web c'est :

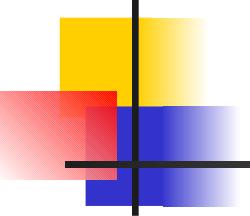
- Un formulaire inséré dans une page Web et permettant à un utilisateur de transmettre des données, de déposer une demande, etc.
- Un serveur traitant cette demande et envoyant une réponse, un accusé de réception, etc.
- *Exemples*
  - Les réservations de vols
  - La gestion de comptes bancaires
  - Le formulaire de déclaration d'impôts du Ministère des Finances
  - Les annuaires
  - Etc.



# Présentation technique

Une application Web c'est :

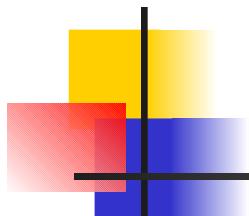
- Une page Web contenant un formulaire HTML envoyé au client
- Du code « autonome » s'exécutant sur un serveur HTTP
  - Le programme est auto suffisant, il calcule par exemple la racine carrée d'un nombre
- Ou du code s'exécutant sur un serveur HTTP et accédant à des services « non Web »
  - Accès à des bases de données
  - Accès à des serveurs LDAP
- ... Mais peut être aussi du code s'exécutant dans le navigateur
  - Validation, vérification des saisies
  - Ergonomie du formulaire



# Mise au point

Une application web nécessite alors

- Obligatoirement un serveur HTTP !
  - **Apache**, (éventuellement IIS)
- De quoi faire exécuter un programme sur le serveur
  - On peut programmer en C, en Fortran ou en assembleur : ce sont des CGI, ils ne sont pas indépendants de la plateforme d'exécution !
  - On préfèrera utiliser des interpréteurs : **PHP**, Java, Perl qui eux sont indépendants de la plateforme d'exécution
- Souvent une base de données
  - **MySQL**, PostgreSQL ou Oracle, etc.
- La possibilité de solliciter un interpréteur localisé sur le navigateur du client
  - Nestcape, IE, etc. et leur interpréteur commun : **JavaScript**

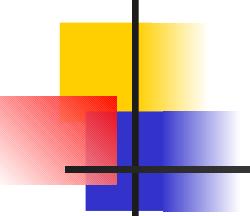


# Les scripts, C? ou S?

A bien distinguer

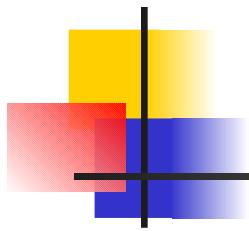
- Ce qui est exécuté sur le serveur : **le script PHP**
  - Tout comme les ASP (Vbscript), les JSP (servletsJava), les CGI (...)
- Et ce qui est exécuté sur le navigateur : **le script JavaScript**
  - Tout comme les applets (Java), les ActivX, les plugins propriétaires

**Ces scripts, PHP ou JS s'exécutent lorsque l'utilisateur agit sur un objet du formulaire : bouton, remplissage d'un champ, sélection dans une liste, ...**



# Les scripts, C? ou S?

- La programmation en JavaScript est de type **événementielle**. On écrira essentiellement de courtes fonctions réalisant des tests ou manipulant l'interface
  - Par exemple, on teste immédiatement, lors de la saisie du champ « âge du candidat », que les caractères entrés sont bien des chiffres et que la valeur est cohérente
- La programmation en PHP est plutôt **séquentielle**
  - Par exemple, on déroule les instructions permettant d'insérer dans la base de données les valeurs transmises depuis le formulaire

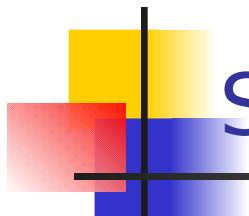


# Les scripts, C? ou S?

Toujours bien choisir

- Ce qui va s'exécuter sur le navigateur ...
- Et ce qui va s'exécuter sur le serveur



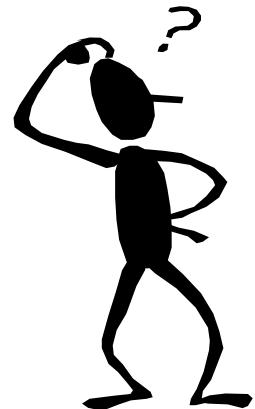


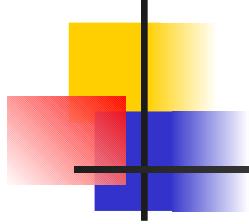
# Scripts: Recommandations

- Si vous faites un contrôle de saisie
  - Vous pouvez utiliser JavaScript sur le client
    - **Avantage** : rapidité, instantanéité, sanction immédiate, pas d'accès réseau
    - **Inconvénient** : si vous utilisez seulement un contrôle JS votre application doit s'assurer que JS n'est pas désactivé sur le navigateur
  - Vous pouvez utiliser PHP sur le serveur
    - **Avantage** : impossible de court-circuiter le contrôle
    - **Inconvénient** : accès serveur, pas de contrôle pas à pas possible, sanction a posteriori
  - Vous pouvez utiliser les deux
    - **Avantage** : rapidité et contrôle temps réel
    - **Inconvénient** : double codage

# Exercice

- Une calculette € : Par qui allez-vous faire effectuer la conversion ?
  - Par du code JavaScript sur le navigateur ?
  - Par du code PHP sur le serveur ?
- Et un script qui affiche l'heure ?
  - Navigateur ou serveur ?





# Intérêt des traitements côtés serveur

Création dynamique de pages web  
exploitation d'autres serveurs //SGBD, SGF

*Cas classique:*  
**Traitement d'un formulaire et réponse**

Transmission allégée vers le client //moins de scripts transmis,  
//souvent une simple page web

Test pour une adaptation aux versions des navigateurs

Test d'authentification des clients

Test de l'état du serveur

# Interprétation sur le serveur

init par le navigateur client : appel d'une URL (fichier html php ...)

☒ **explicite** (utilisateur)

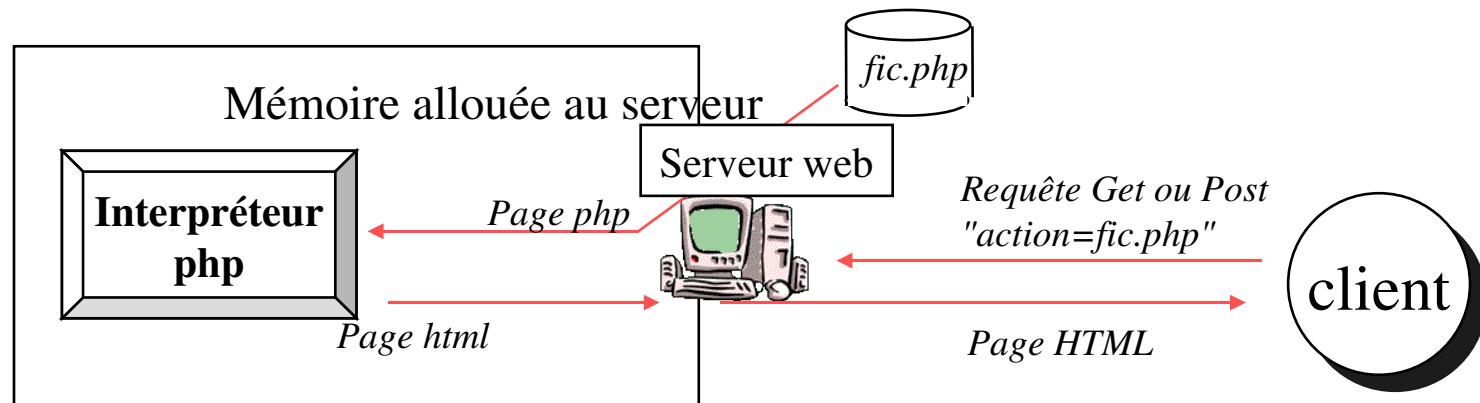
▪ **implicite** (traitement d'un formulaire)

Présentation  
d'une page web

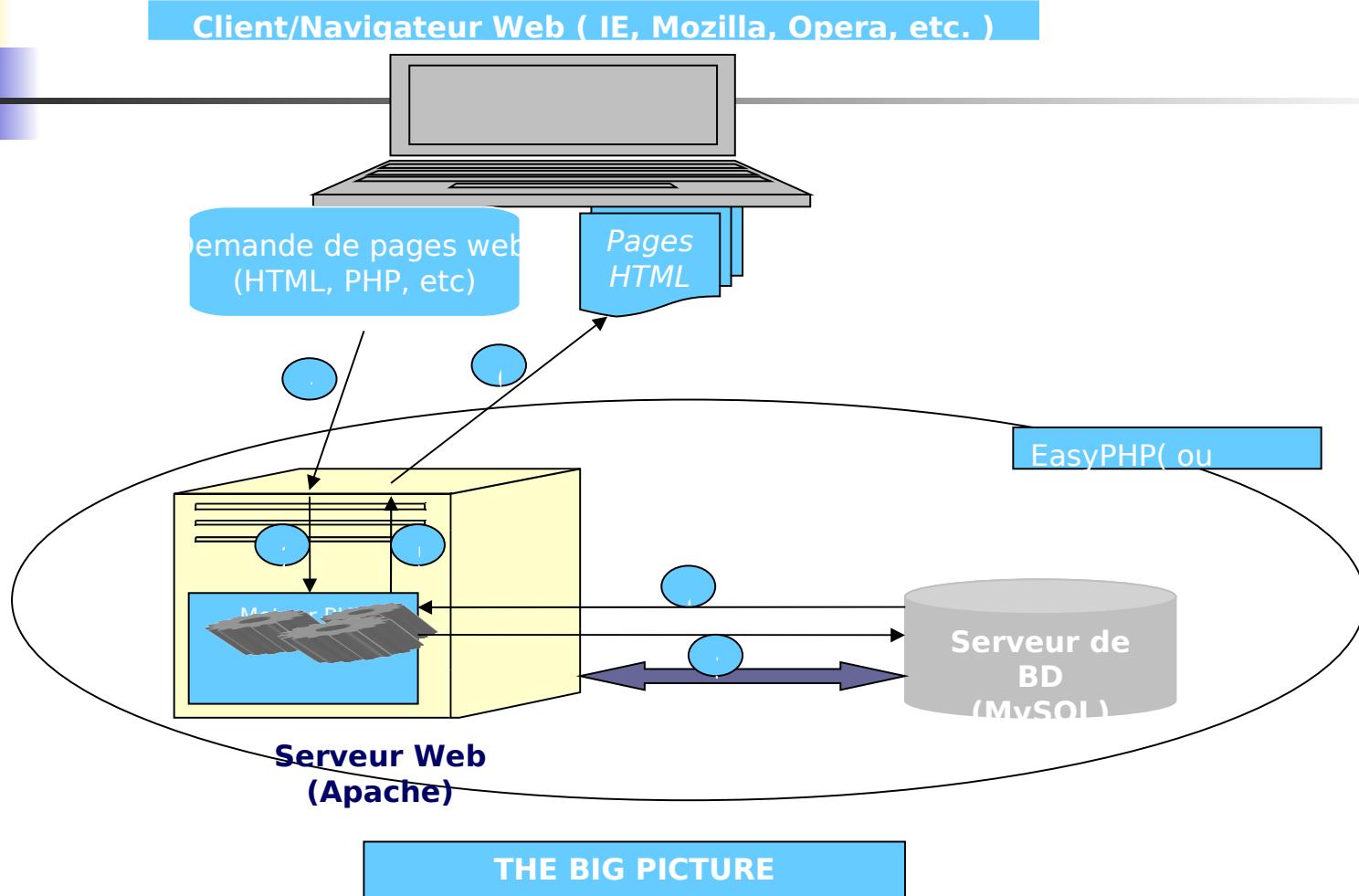
entrée : paramètres d'exécution, variables d'environnement du serveur web

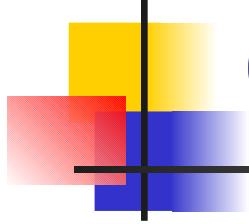
sortie : une page HTML à produire (print, echo, ...)

traitement : code classique (variables, instructions à réaliser sur le serveur)



# Schéma de fonctionnement général





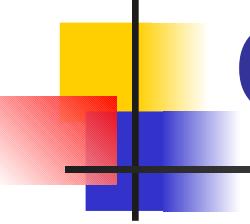
# Outils: EasyPHP

**EasyPHP est un outils Windows qui intègre le serveur web apache, un module PHP et un serveur MySQL.**

**Son installation est simple (moins d'une minute !!!) et aucune configuration spéciale ne lui est nécessaire pour fonctionner.**

**Il est libre et peut être téléchargé à  
[www.easypht.org](http://www.easypht.org)**

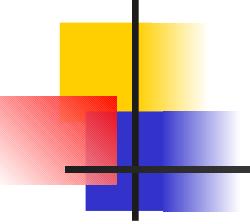
**Disponible en plusieurs versions**



# Outils: XAMPP

- XAMPP est une distribution d'Apache très simple à installer:**  
**just download, extract and start !!!!**
- Disponible en 4 versions: Windows, Linux, MacOS, Solaris**
- Contient MySQL, PHP et Perl**
- Documentation et fichiers disponibles sur le site des Apache Friends : <http://www.apachefriends.org/>**

Also can you get WAMP (Windows), Lamp(Linux), etc.

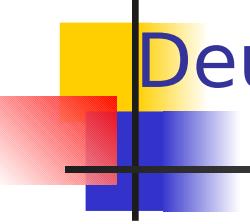


# Premier programme

- La conception du script PHP est réalisé avec un éditeur quelconque. Notepad, Context, Emacs, Vi (hé hé hé !) sont très bien pour cela.

```
<HTML>
<HEAD><TITLE>Mon premier programme</TITLE></HEAD>
<BODY>
<? echo "Ca marche !"; ?>
</BODY>
</HTML>
```

- Editer ce programme
- Copier le à la racine du serveur (htdocs, www, etc.)
- Appeler l'url <http://localhost/test.php> avec votre navigateur favoris pour voir le résultat
- L'extension de votre fichier doit être .php afin d'être interprété par le serveur (Apache par exemple)

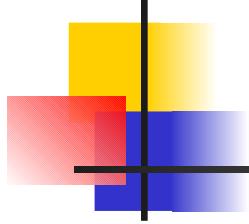


## Deuxième programme

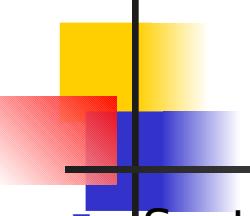
info.php

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

- Au résultat, vous recevrez une page contenant toutes les caractéristiques et options de votre interpréteur PHP



# Les bases du langage



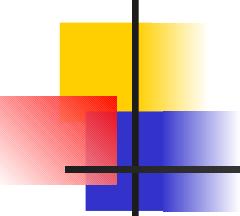
# Le langage

- Syntaxe héritée du C :

- Même séparateur pour les instructions : le ;
- Même types de commentaires

```
/* commentaires sur plusieurs lignes ...
...Fin des commentaires */
// une seule ligne de commentaires
```

- Tous les types de données : entiers, nombres à virgule flottante, chaînes de caractères, tableaux, objets...
  - Des entiers : 123, -24, 0677 (octal), 0x456 (hexadécimal)
  - Des flottants : 12.34, 1.2e3
  - Un booléen : true ou false
  - Une chaîne de caractères 'hello le monde'
  - ...

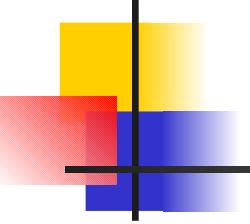


# Les variables

- Une variable en PHP commence par le caractère \$
  - \$\_nom, \$index, \$k2, \$annee\_de\_naissance
- Les variables sont sensibles à la casse
- Les variables n'ont pas à être déclarées
- Les variables ne sont pas typées, c'est le contexte qui décide.  
Langage faiblement typé (pas de déclaration nécessaire des types)

```
$var = 24; // $var représente un entier...
$var = 'mardi'; // ...puis une chaîne de caractères
```
- On peut faire des variables dynamiques (variable de variable)

```
$var = 'jour ';
$$var = 'mardi'
echo $jour; // imprimera mardi
```

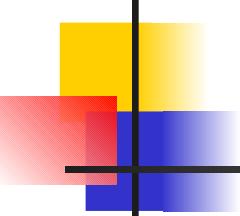


# Les constantes

- Il est inutile de revenir sur l'intérêt des constantes.
- Pour définir une constante il suffit d'utiliser l'instruction *define*

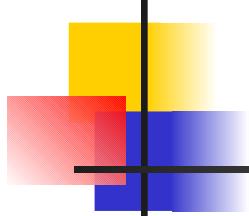
exemple :

```
define("MACONSTANTE", "Hello World") ;  
if (defined("MACONSTANTE")) {  
    print "La valeur de ma constante est :  
    ".MACONSTANTE //Attention, on accède au  
    contenu sans le $.  
}
```



# Les tableaux

- La déclaration d'un tableau se fait de la même manière que la déclaration d'une variable avec un indice se trouvant entre [ et ].
- `$tableau[0] = 1;` // on crée un tableau, et sa première valeur est 1
- Pour un tableau à deux dimensions, il suffit de mettre un second indice au moment de l'affectation.  
exemple :  
`$tableau[0][0] = 1;` // on crée un tableau, et sa première valeur est 1
- Il n'est pas obligatoire de préciser l'indice pour affecter une valeur.  
exemple :  
`$tableau[] = 1;` // équivaut à `$tableau[0] = 1;`  
`$tableau[] = 45;` // équivaut à `$tableau[1] = 45;`  
`$tableau[] = 6;` // équivaut à `$tableau[2] = 6;`
- La déclaration et l'initialisation d'un tableau peuvent également se faire par l'intermédiaire de la fonction 'array()'. Cette fonction permet de préciser les indices ainsi que les valeurs du tableau (à l'aide de l'opérateur =>).



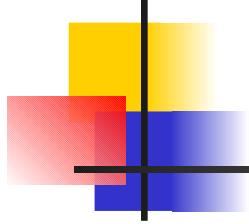
# Les tableaux

*Exemple :*

```
$tableau[ ] = array(0=>1, 1=>45, 2=>6);
```

```
$tableau[ ] = array("rouge"=>"red",
"vert"=>"green", "bleu"=>"blue");
```

- La navigation dans les éléments du tableau s'effectue à l'aide des fonctions 'next()' , 'prev()' et 'each()'. (voir la suite)
- Le nombre d'éléments d'un tableau peut être obtenu à l'aide de la fonction 'count()'.
- Le tri des tableaux est facilité par de nombreuses fonctions : asort(), ksort(), sort(), usort(), etc...



# Opérateurs

-Les opérateurs classiques sur les nombres sont disponibles :

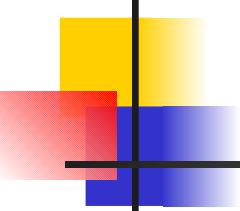
« + »

(addition), « - » (soustraction), « \* » (multiplication), « / »  
(division).

**Exple \$a=\$a1-\$a2;**

-Le seul opérateur sur les chaînes est la concaténation,  
symbolisée par un point (« . »).

**Exple \$a=\$a1.\$a2;**

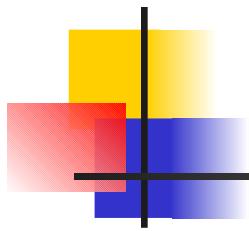


# Opérateurs Logiques

Ils permettent de combiner plusieurs tests entre eux.

exemple : Ici \$a et \$b peuvent prendre les valeurs booléennes vrai ou faux.

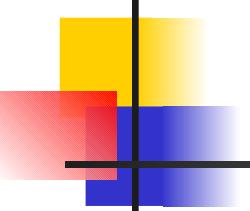
Opérateur	Exemple	Résultat
and ( <b>&amp;&amp;</b> )	\$a and \$b <i>\$a &amp;&amp; \$b</i>	vrai si \$a et \$b sont vrai tous les deux
or ( <b>  </b> )	\$a or \$b <i>\$a    \$b</i>	vrai si \$a est vrai ou \$b est vrai, ou encore si \$a et \$b sont vrai tous les deux
not ( <b>!</b> )	not \$a <i>!\$a</i>	vrai si \$a est faux



# Opérateurs de Comparaison

Ils permettent de comparer les valeurs de deux variables.  
exemple : Ici \$a et \$b sont du même type de variable.

Opérateur	Exemple	Résultat
<code>==</code>	<code>\$a == \$b</code>	vrai si \$a est égal à \$b
<code>!=</code> \$b	<code>\$a != \$b</code>	vrai si \$a est différent de \$b
<code>&lt;</code>	<code>\$a &lt; \$b</code>	vrai si \$a est inférieur \$b
<code>&gt;</code>	<code>\$a &gt; \$b</code>	vrai si \$a est supérieur \$b
<code>&lt;=</code> égal à \$b	<code>\$a &lt;= \$b</code>	vrai si \$a est inférieur ou égal à \$b
<code>&gt;=</code> égal à \$b	<code>\$a &gt;= \$b</code>	vrai si \$a est supérieur ou égal à \$b

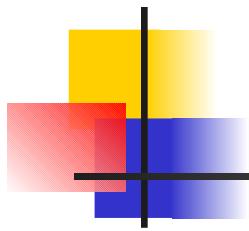


# L'opérateur ternaire

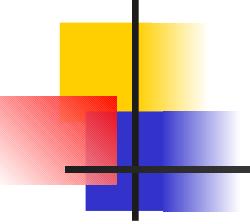
Cette instruction est uniquement une simplification de l'écriture *if... else ....*

```
echo ($nombre == 1) ? "$nombre est égal à 1" : "$nombre n'est pas  
égal à 1";
```

La condition doit obligatoirement se trouver entre parenthèses. Si la condition est vérifiée, la valeur de gauche est retournée, sinon, c'est celle de droite



### 3. INSTRUCTIONS DE CONTROLES

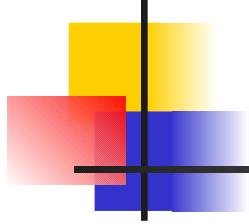


# Tests : if... then... else

Test de base que l'on trouve dans la majeure partie des langages. Si une condition est vrai alors on exécute des instructions sinon (facultatif) on en exécute d'autre. On peut changer de condition avec un 'elseif'.

syntaxe :

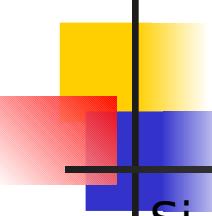
```
if ( condition1 ) {  
    Action 1  
} elseif ( condition2 ) {  
    Action 2  
}  
} else {  
    Action 3  
}
```



# Tests : if... then... else

exemple :

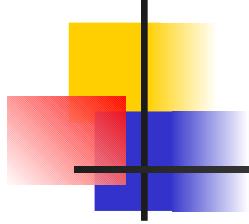
```
if ($a==$b) {  
    echo "A est égal à B";  
} elseif ($a > $b) {  
    echo "A est supérieur à B";  
} else {  
    echo "A est inférieur à B";  
}
```



# Tests : switch... case... default

Si les conditions successives ne portent que sur la valeur d'une variable, on pourra avantageusement remplacer le test 'if... elseif... else' par 'switch'. Dans ce test, la condition est associée à la valeur d'une variable, de plus l'instruction 'break' est primordiale à la fin de chaque bloc de conditions, sinon toutes les conditions seront vérifiées et exécutées. Exemple :

```
switch ($a) {  
    case $b:  
        echo "A est égal à B";  
        break;  
    case >$b:  
        echo "A est supérieur à B";  
        break;  
    default:  
        echo "A est inférieur à B";  
        break;  
}
```



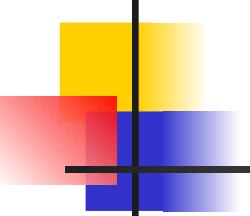
# Boucles

- *while ( condition ) {  
    Action;  
}*
- *do {  
    Action;  
} while (condition);*
- *for (expr1; expr2; expr3) {  
    Action;  
}*

```
$i=1;  
while ($i <= 10) {  
    echo "- $i -";  
    $i++;  
}
```

```
$i=1;  
do {  
    echo "- $i -";  
    $i++;  
} while ($i <= 10)
```

```
for ($i = 1; $i <= 10; $i++) {  
    echo "- $i -";  
}
```

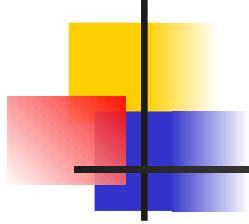


# Break;Continue

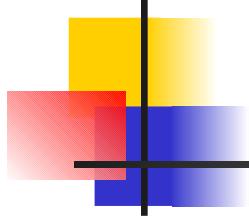
*Break* :Cette instruction permet de sortir de n'importe quelle boucle, à n'importe quel moment.

*Continue* : Cette instruction permet de ne pas exécuter le code contenu dans la boucle et de passer à l'itération suivante.

```
for ($i=1; $i<=10; $i++) {  
    if ($i<=5) {  
        echo $i;  
    }  
    else {  
        break;  
    }  
    echo "- ";  
}  
// cette boucle affichera : 1 - 2 - 3 - 4 - 5
```

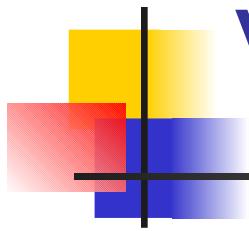


## 4. RETOUR SUR LES VARIABLES



# Variables d'environnement

- L'un des aspects fondamentaux d'une application construite sur une architecture de type intranet est l'utilisation des variables d'environnement du serveur, et notamment celles du serveur HTTP.
- Avec PHP toutes les variables d'environnement du serveur sont automatiquement reprises dans les scripts PHP en tant que variables globales. Ainsi il suffit de les utiliser directement dans le code.



# Variables d'environnement

## **exemples :**

→ Tableaux `$_SERVER /HTTP_SERVER_VARS`

```
$ip=$_SERVER['REMOTE_ADDR']
echo $ip; //@ IP de la machine cliente
```

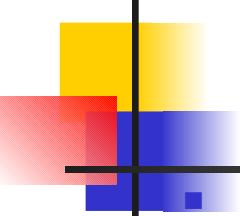
→ Vous pouvez donc, par exemple, obtenir le type de navigateur du visiteur de plusieurs manières :

```
<?
    echo "Avec HTTP_USER_AGENT : $HTTP_USER_AGENT<br>";
    echo "Avec getenv() : ".getenv("HTTP_USER_AGENT");
?>
```

Ce qui donne :

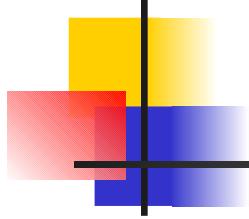
Avec `HTTP_USER_AGENT` : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)

Avec `getenv()` : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)



# Exemples de variables d'environnement

- **SERVER\_NAME** : Le nom du serveur hôte qui exécute le script suivant. Si le script est exécuté sur un hôte virtuel, ce sera la valeur définie pour cet hôte virtuel.
- **PHP\_SELF** : Le nom du fichier du script en cours d'exécution, par rapport au document root.
- **DOCUMENT\_ROOT** : La racine sous laquelle le script courant est exécuté, comme défini dans la configuration du serveur.
- **REMOTE\_ADDR** : L'adresse IP du client qui demande la page courante.
- **REMOTE\_PORT** : Le port utilisé par la machine cliente pour communiquer avec le serveur web.
- **SCRIPT\_FILENAME** : Le chemin absolu jusqu'au script courant.
- **SERVER\_PORT** : Le port de la machine serveur utilisé pour les communications. Par défaut, c'est '80'. En utilisant SSL, par exemple, il sera remplacé par le numéro de port HTTP sécurisé.
- **REQUEST\_URI** : L'URI qui a été fourni pour accéder à cette page. Par exemple : '/index.html'.



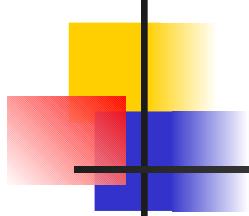
# Exemple simple: Récupérer son adresse IP

- Exemple : 213.154.81.180

```
<?
echo "$_SERVER['REMOTE_ADDR']";
?>
```

**C'est facile... on utilise seulement une variable d'environnement, ici \$REMOTE\_ADDR.**

**On aurait pu également utiliser getenv ("REMOTE\_ADDR"), c'est la même chose.**



# Variables issues de formulaires

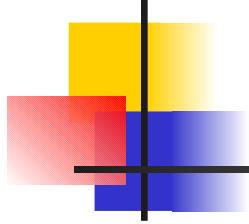
- Les variables sont issues de formulaires HTML, elles correspondent aux différents champs positionnés entre les balises <FORM> et </FORM> de ce formulaire. La page qui reçoit ces variables est celle qui est désignée par l'attribut ACTION de la balise <FORM>.

exemple (fichier *test.html*) :

```
<HTML>
<FORM ACTION="test.php" METHOD=POST>
<INPUT TYPE=hidden NAME="toto" VALUE="bonjour">
Nom : <INPUT TYPE=text NAME="nom">
<INPUT TYPE=submit VALUE="Envoyer">
</FORM>
</HTML>
```

# Variables issues de formulaires





# Différence de méthode de transmission entre GET et POST

Données du formulaire dans le paquet d'entête limité en taille, d'où petite requête

## 2 méthodes de transmission:

**GET** http://serveur/action.exe? nomChps1=val1&  
nomChps2=val2

**POST**

http://serveur/action.exe

données

données

Pour la méthode POST :  
Il est possible de rajouter  
(après un séparateur '?')  
des arguments  
pour le programme à exécuter

Paquets de données hors entête  
transfert fichier, grand formulaire

# Exemple de méthode GET

La soumission d'un formulaire au serveur peut être vue comme l'appel d'un programme avec une passation de paramètres.

<Form action=explore/search.php method=get>

université paris

Nom du champs

Valeur saisie mais codée

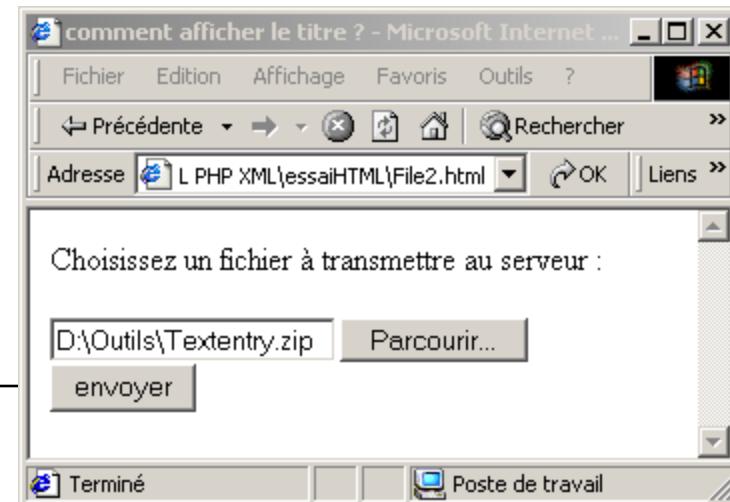
http://explore/search.php ? recherche=universit%E9+paris

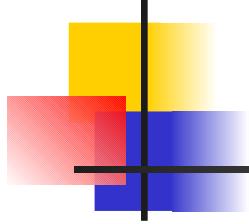
URL du script à lancer

- ☒ "+" : code URL du caractère d'espacement
- %E9 : code url du "é"

# Exemple de méthode POST : le chargement de fichier

```
<HTML><BODY> ...
<FORM action="upload.php" method="POST">
Nom: choisissez un fichier
<br><br>
<INPUT type="file" name="fic" EncType="multipart/form-data">
<INPUT type="submit" value="envoyer">
</FORM></BODY></HTML>
```



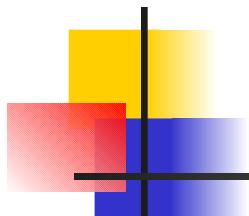


# Envoi d'un formulaire

Un bouton <INPUT> de type submit ou image au sein du formulaire  
note : l'appui du bouton peut être simulé en javascript par une action  
`nomForm.submit()`

```
<FORM Action="chemin/script ou exe"  
      Target="nom_fen"  
      Method="GET">  
  
... le formulaire...  
<INPUT type="submit" value="envoi"> ... </FORM>
```

Les données saisies sont transmises au script par une collection de paires :  
| - nom des champs de contrôle (`name`)  
| - valeur courante du champs (`value`)



# Variables PHP correspondant aux noms des champs d'un formulaire

variables automatiques : \$nomChps

- ☒ nécessite de paramétrier le serveur (fichier php.ini)  
\$register\_globals=on (pas par défaut)

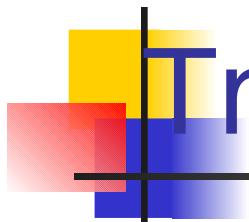
tableau de variables : (*meilleur contrôle pour la sécurité*)

- ☒ \$\_GET['nomChps'] méthode get
- \$\_POST['nomChps'] méthode post

```
<php //pour exploiter la syntaxe des variables automatiques si $register_globals=off  
    $nomChamps=$_GET['nomChps']  
    echo $nomChamps //ça marche !!!  
?>
```

Ancienne version php3 :

\$\_HTTP\_GET\_VARS['nom']  
\$\_HTTP\_POST\_VARS['nom']



# Traitement du formulaire

- Le formulaire est décrit dans une page HTML qui peut être statique ou dynamique...
- Le programme PHP décrit par le paramètre « action » de la balise « form », est invoqué lors du clic sur le bouton « submit »
- Le programme PHP doit récupérer les valeurs saisies dans les différents champs du formulaire pour les traiter
- C'est le nom, au sens HTML, défini par le paramètre « name », qui va être utilisé pour créer la variable PHP

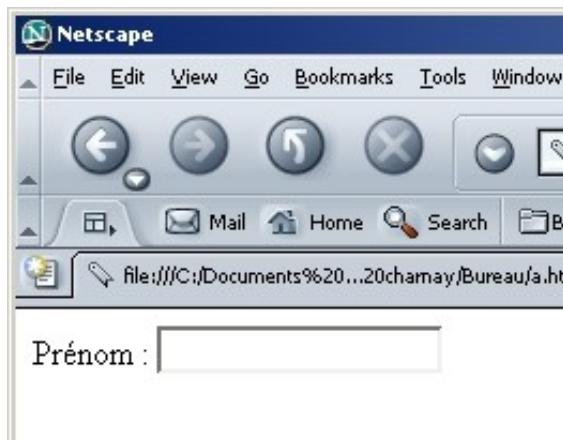
# Récupération de la valeur des champs -1

test.html

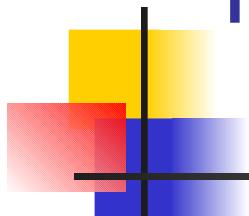
```
<html>
<form
    name=xyz
    action=test.php>
Prénom :
<input name=prenom>
</form>
</html>
```

test.php

```
<?
echo "Bonjour" . $prenom;
?>
```



⚠ Cette méthode n'est possible que si **register\_globals = on** dans le fichier de configuration de l'interpréteur PHP (php.ini)



# Récupération de la valeur des champs -2

- Pour des raisons de sécurité, la méthode précédente n'est pas conseillée...
- On utilisera le tableau associatif `$_REQUEST` dont les clés sont les noms des éléments HTML

Clé	Valeur	\$_REQUEST
nom	xavier	
prenom	blanc	
email	xblanc@lip6.fr	

```
$_REQUEST["prenom"] = xavier
```

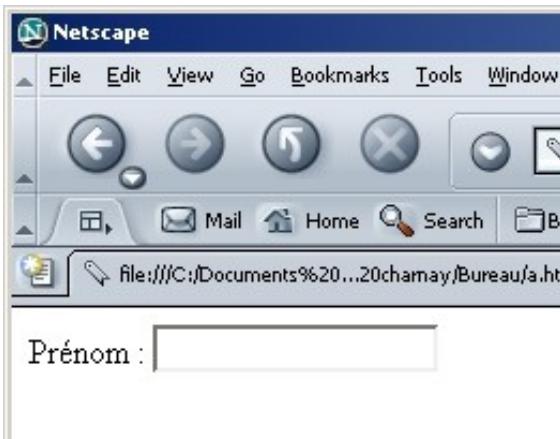
# Récupération de la valeur des champs -3

test.html

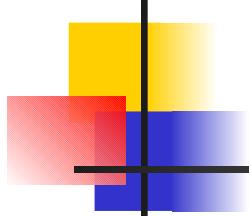
```
<html>
<form
    name=xyz
    action=test.php>
Prénom :
<input name=prenom>
</form>
</html>
```

test.php

```
<?
$prenom = $_REQUEST["prenom"];
echo "Bonjour". $prenom;
?>
```

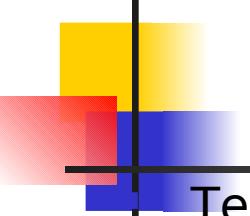


微笑 This method works regardless of whether the GET or POST method is chosen for form transmission.



# Have in mind !

- L'exécution du programme PHP sur le serveur ...
  - Effectue un certain nombre d'actions...
    - Vérification des données reçues, insertion d'enregistrements dans une base de données, récupération de fichiers, etc.
  - **Mais au final génère toujours de l'HTML à destination du client**
    - Message de confirmation, d'erreurs, résultats divers, etc.
    - Si on ne génère pas de code HTML le résultat retourné est ... une page vide !



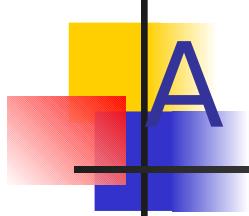
# Exercice

Tester « request.html » et « request.php »

- Vérifier la bonne exécution en POST et en GET (remarquer l'URL)
- Comprendre la fonction « isset » qui teste si une variable a été définie
  - En appelant directement le PHP
  - En passant par le formulaire HTML
- Faites la différence entre une variable non définie et une variable

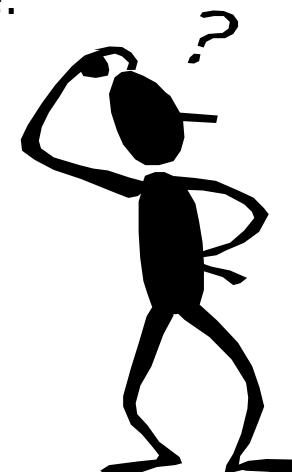
```
<html>
<form
    name=xyz
    action=request.php
    method=post>
Mail :
<input name=mail>
</form>
</html>
```

```
<?
if (isset($_REQUEST["mail"])){
    $mail = $_REQUEST["mail"];
    print ("<br>Adresse mail : $mail");
} else{
    print ("<p>mail non défini !");
}
?>
```



# A propos de cet exercice

- Imaginons que le champ « mail » soit obligatoire
- Qui devra s'assurer qu'il a bien été rempli ?
  - Le poste client avec du JavaScript ?
    - Et si JS n'est pas activé ? Doit-on interdire l'accès au formulaire ?
  - Le serveur devra aussi effectuer des vérifications, dans le cas où l'on admet que JS soit désactivé.

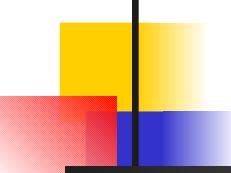


## La page HTML avec un script « client »

```
<html>
<head>
<script> //javaScript
function verifMail(){
    if (document.forms[0].mail.value.length < 6 ||
        document.forms[0].mail.value.indexOf("@") < 0 ||
        document.forms[0].mail.value.indexOf(".") < 0) {
        alert ("mail incorrect");
        return false;
    }
}
</script>
</head><body>
<form
    name="xyz"
    action= "recevoirMail.php"
    method="post"
    onSubmit="return verifMail();"
Mail :
<input name="mail">
</form>
</body></html>
```

lireMail.html

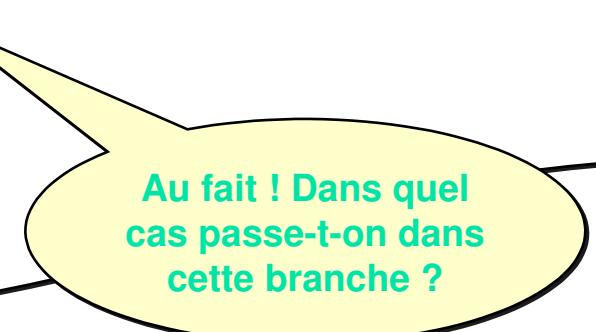
Bloque l'appel au programme PHP en cas d'erreur



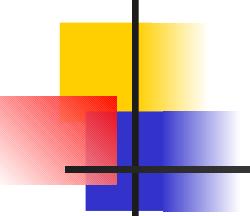
# Le script serveur PHP

recevoirMail.php

```
<?
if (isset($_REQUEST["mail"])){
    $mail = $_REQUEST["mail"];
    if (strlen($mail) < 6 || 
        ! strpos($mail, '@') || 
        ! strpos($mail, '.')){
        print ("<font color=red><b>Mail incorrect !</b></font> ");
        print ("<a href=lireMail.html>recommencez</a>");
    } else {
        print ("Adresse mail : $mail");
    }
} else{
    print ("<h1>INTERDIT !");
}
?>
```



Au fait ! Dans quel cas passe-t-on dans cette branche ?



# Conclusion

- Tester côté client est plus ergonomique
  - Sanction immédiate, pas de nécessité de regénérer le formulaire, on peut tester chaque saisie en temps réel (onBlur, onFocus)... mais impose que le client autorise JS
- Tester côté serveur marche à tous les coups
  - Mais ne tester que côté serveur, oblige à régénérer le formulaire, à conserver le contenu des champs qui étaient déjà remplis correctement...
- Tester des 2 côtés c'est parfait, mais c'est plus de travail !

# Le formulaire est-il toujours issu d'une page HTML statique ?

The screenshot shows a Netscape browser window with the title "RAPPORTS COMITÉ NATIONAL :: DÉPOSER - Netscape". The URL in the address bar is [http://evalcn.ccsd.cnrs.fr/index.php?action\\_todo=submit&RapportsCN\\_SESSID=c81ff6b0448fce281a3a](http://evalcn.ccsd.cnrs.fr/index.php?action_todo=submit&RapportsCN_SESSID=c81ff6b0448fce281a3a). The page header includes the CN logo, "COMITÉ NATIONAL DE LA RECHERCHE SCIENTIFIQUE", "Rapports chercheurs", and the CCSD logo. The menu bar has options like File, Edit, View, Go, Bookmarks, Tools, Window, Help. The toolbar includes icons for Back, Forward, Stop, Home, Mail, Search, and others. A search bar is present. The main content area shows a form for reporting research results. It includes fields for "Nom" (Name), "Prénoms" (First names), "Année de campagne" (Campaign year) with a dropdown menu showing "2003" and a list of options, "Code unité CNRS" (CNRS unit code), "Date de réunion du comité" (Committee meeting date), and "Fichier du rapport" (Report file). Below these fields are two buttons: "Déposer" (Upload) and "Annuler" (Cancel). A callout box highlights the dropdown menu for "Année de campagne" with the following text:  
**Non ! le formulaire est ici généré en PHP, les options de ce « select » sont issues d'une requête à une base de données**

A callout box highlights the dropdown menu for "Année de campagne" with the following text:

**Non ! le formulaire est ici généré en PHP, les options de ce « select » sont issues d'une requête à une base de données**

The dropdown menu shows the following options:

- Aucune Unité
- ESA8043 - Origine et structure de la bio...
- ESA8044 - Biologie des invertébrés marin...
- ESA8045 - Archéozoologie et histoire des...
- FR1577 - Institut de chimie moléculaire ...
- FR1739 - Institut federatif de recherche...
- FR1742 - Institut nancéien de chimie mol...
- FR1768 - Institut federatif de recherche...
- FR1769 - Institut de droit et d'économie...
- FR1818 - Milieux naturels et anthropisées...
- FR1878 - Institut Charles Gerhardt - Ins...
- FR1886 - Institut max mousseron - Instit...
- FR1981 - Centre de Recherche en Chimie M...
- FR2035 - Institut des sciences de la ter...
- FR2108 - Institut de Chimie de Rennes**
- FR2112 - Institut de Chimie Analytique
- FR2116 - Centre Armorican de Recherche ...
- FR2145 - Matériaux de Structure et Propr...

# Un seul fichier !

remarque.php

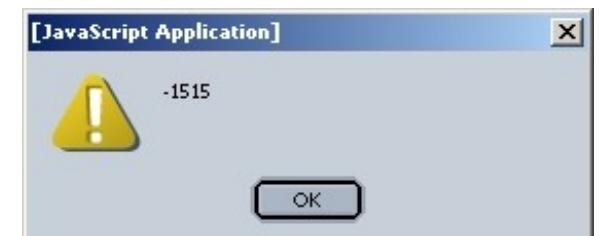
```
<HTML><HEAD><TITLE> tout en un </TITLE>
    <script langage=javascript>
mail = /^[a-zA-Z0-9]+[a-zA-Z0-9\.-_]+@[([a-zA-Z0-9\.-])+\.+([a-zA-Z0-9])+$/;
        function sMail(){
            ok=mail.test(document.forms[0].elements[1].value)
                if (!ok){
                    alert ("adresse mail invalide !");
                    return false;
                }
            }
        </script>
</HEAD><BODY BGCOLOR="#e3e3e3">
    <?PHP
    if ( !isset($_REQUEST["mail"])){
        ?>
    <form name=remarque action=remarque.php method=post
        onSubmit="return sMail();">
        Votre commentaire :<br>
        <textarea name=texte></textarea><br>
        Votre e-mail ?<br>
    <input name=mail><input type=submit value="envoyer">
        </form>
        <?
    } else {
        print ("Merci de votre commentaire<br> ");
print ("Nous vous r  pondrons &agrave; ".$_REQUEST["mail"]);
        }
    ?>
</BODY></HTML>
```

Remarquez l'all e-venue entre  
le code PHP et HTML et  
conditionnement de ce derni re  
par le if

# Code dans code

- Bien sûr, comme PHP génère du HTML, il peut aussi générer du JavaScript !

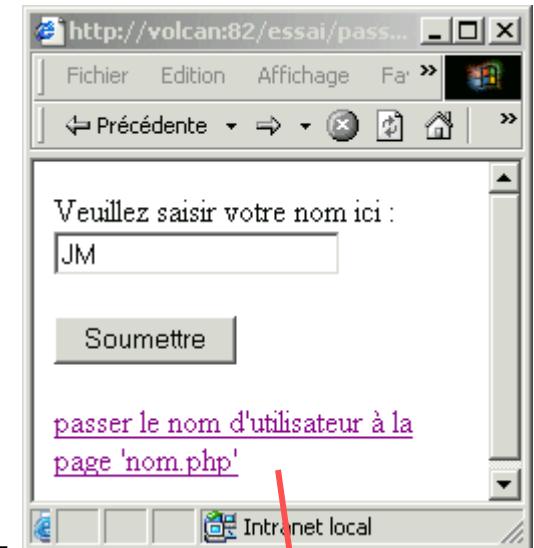
```
<?php  
...  
$erreur = fopen (...  
echo "<script language='JavaScript'>";  
echo "alert($erreur)";  
echo "</script>";  
...  
?>
```



# Passation de paramètres entre pages par un lien

```
<HTML><BODY><FORM Action= main.php> main.php
Veuillez saisir votre nom ici :<BR>
<INPUT TYPE=TEXT NAME=utilisateur><BR><BR>
<INPUT TYPE=SUBMIT VALUE="Soumettre">
</FORM>

<A id="idLien"
HREF="nom.php?U=<?php echo ($utilisateur); ?>" >
passer le nom d'utilisateur à la page 'nom.php' </A>
</BODY></HTML>
```



```
<HTML> <!-- cette page attend un paramètre de nom $U-->
<BODY>
    <P>Utilisateur : <?php echo (Utilisateur $U); ?> </P>
</BODY></HTML>
```

**nom.php**



## Inclusion de fichiers : request et include

- Les fichiers importés étant supposés du HTML, ils devront comporter des balises PHP

*require ("nom\_du\_fichier")* : remplace l'instruction par le contenu du fichier

*include ("nom\_du\_fichier")* : évalue le code contenu dans le fichier

require() et include() sont identiques sauf dans leurs façons de gérer les erreurs: la première génère une erreur fatale et la deuxième un warning

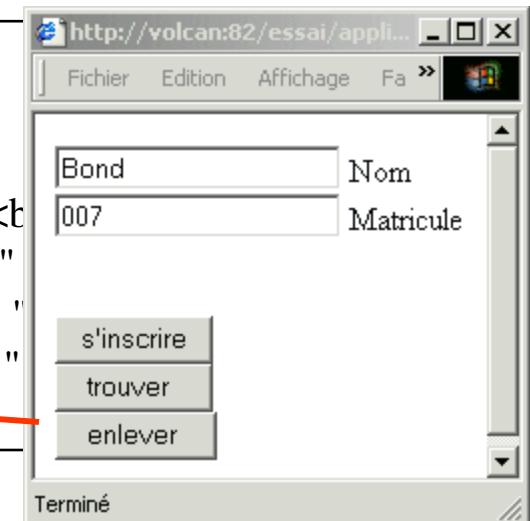
```
<?php  
include ('database.php');  
$cxDB=mysql_connect(HOST,USER,PWD);  
if (!$cxDB){  
?>
```

```
<?php  
define ("HOST","ccmysql.ucad.sn");  
define ("USER","leUser");  
define ("PWD","leMotDePasse");  
define ("BD", "resa");  
?>
```

database.php

# Exemple d'emploi des inclusions

```
<html><body>
<FORM action="main.php" METHOD=POST>
    <INPUT name="nom" type="text"> Nom <br>
    <INPUT name="mat" type="text"> Matricule <br>
    <BR><INPUT type="submit" name="choix" value = "s'inscrire"
    <BR><INPUT type="submit" name="choix" value = "trouver"
    <BR><INPUT type="submit" name="choix" value = "enlever"
</FORM></body></html>
```

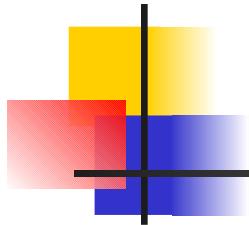


```
<html><body><h3>Bienvenue M. <?php echo ("$nom"); ?> </h3>
```

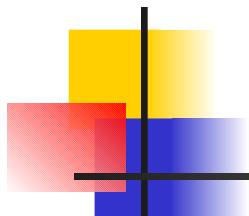
```
switch ($choix) {
case "s'inscrire" : include (' inscrire.php');break;
case "trouver" : include ('trouver.php');break;
case "enlever" :include ('enlever.php');break;
default :echo ("mais il nous faudrait autre chose !! "); exit();
}?><body><html>
```

structuration du code en trois fichiers  
dont un va être interprété à la soumission  
du formulaire

**main.php**



## 4. LES FONCTIONS



# Les Fonctions

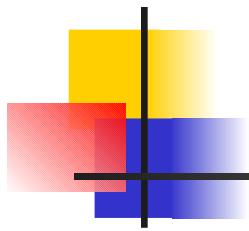
Comme dans tout langage structuré, en PHP, les fonctions sont la base d'une programmation claire et efficace. Une fonction est une sorte de sous-programme isolé du reste du code, exécutable à tout moment, depuis n'importe quelle partie du code principal ou n'importe quelle autre fonction, par simple appel. De plus, les avantages des fonctions sont :

La non répétition de la même séquence de code

De cet avantage découlent :

- Le gain de productivité.
- La meilleure lisibilité du code.
- La maintenance facilitée.

# Déclaration, paramètres, valeurs de retour



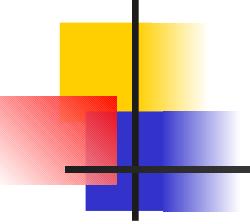
Une fonction est une séquence de code qui peut remplir n'importe quelle tâche. Tout code valide peut figurer dans le corps d'une fonction.

Déclaration :

La syntaxe de déclaration s'appuie sur le mot clé *function*. Ce mot clé est immédiatement suivi du nom de la fonction, puis de parenthèses (obligatoires) destinées à accueillir les éventuels paramètres à passer à la fonction.



# Déclaration, paramètres, valeurs de retour



exemple :

```
echo " Bonjour ! " ; function bonjour( ) {
```

```
}
```

Cette fonction ne fait qu'afficher 'bonjour' et ne retourne aucun résultat, on l'utilisera de la manière suivante :

```
Bonjour( ) ; // affiche 'bonjour' à l'écran
```

Pour que la fonction retourne un résultat, on utilisera le mot clé return

```
function bonjour2( ) {  
    return " Bonjour !" ;  
}
```

```
echo bonjour2( ) ;
```

bonjour affiche le résultat elle-même, alors qu'il faut afficher le résultat de bonjour2 pour obtenir une action similaire.



# Déclaration, paramètres, valeurs de retour

## Passage par valeur par référence :

Le passage des paramètres tel qu'on l'a vu précédemment est ce que l'on appelle le passage par valeur. Il existe une autre manière de procéder :le passage par référence. On passe à la fonction la référence (adresse mémoire) d'une variable existante, et la fonction modifie directement la valeur de cette variable.

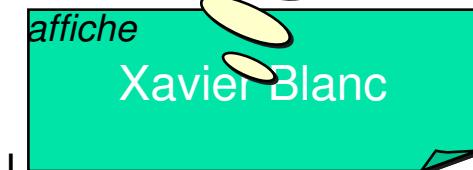
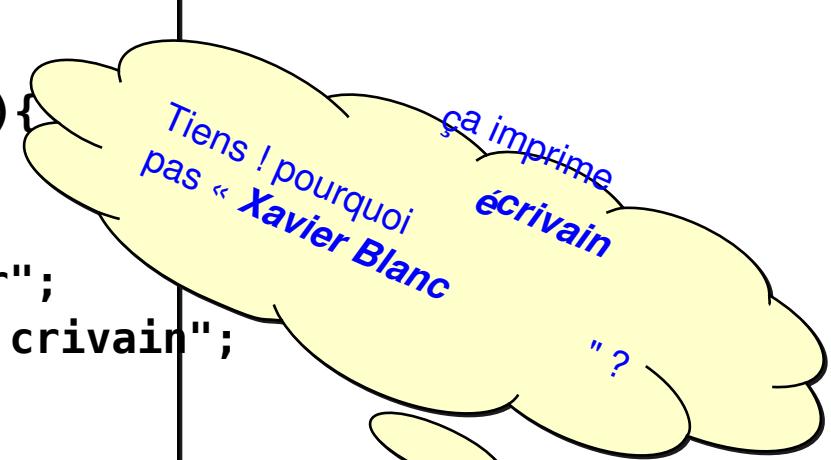
exemple :

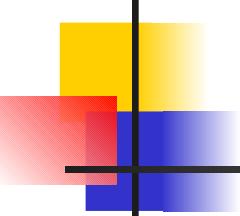
```
function bonjour(&$phrase) {  
    $phrase= " bonjour Toto Dupont " ;  
}  
$chaine = " Phrase qui va disparaître " ;  
bonjour($chaine) ;  
echo $chaine ; // affiche 'bonjour Toto Dupont' à l'écran
```

# Portée des variables

- La portée d'une variable dépend du contexte dans lequel elle est définie
  - Hors d'une fonction, elle est globale, mais doit être déclarée comme telle dans la fonction, sous peine de redéfinition
  - Dans une fonction elle est locale, sauf définition comme global

```
$nom = « Blanc»;
function jamaisTest(){
    global $nom;
    global $prenom;
    $prenom = « Xavier»;
    $titre = "&acute;crivain";
}
jamaisTest();
echo "$prenom $nom $titre";
```





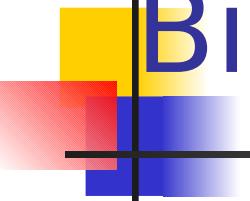
# Variables static

- Comme en C, la variable ne perd pas sa valeur à chaque appel du bloc
- Utilisé dans un contexte récursif

Mais, dans une  
seule exécution du  
programme !

```
function compte(){
    static $compteur = 0;
    $compteur++;
    while ($compteur <10){echo $compteur; compte();}
}
compte();
```

*affiche*  
123456789

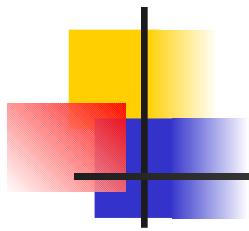


# Bibliothèques de fonctions

- La distribution PHP intègre des centaines de fonctions pour tout faire !
  - On n'a pas parlé du traitement des chaînes de caractères : il y a toutes celles du C (strxxx) plus des tas d'autres !
  - Evidement toutes les fonctions mathématiques
  - Il y a tout pour gérer des dates (c'est très très facile ...)
  - Tout pour gérer, les protocoles, les systèmes de fichiers, les SGBD ...
  - ... et tout pour faire du XML, XSL ...des web-services

**Dans les slides qui suivent nous parcourons un certain nombre de fonctions souvent utilisées**



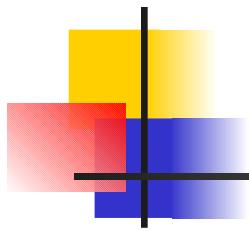


# Fonction Explode

- Il est fréquent d'avoir des fichiers contenant différents champs séparés par un délimiteur quelconque. Une fonction très utile dans ces cas là est la fonction *explode*. Sa syntaxe est la suivante

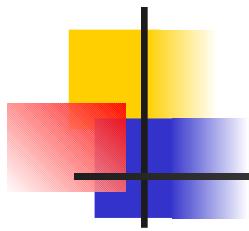
***explode (" caractère délimiteur ", chaîne de donnée)***

- *Blanc| Xavier | 36*
- *FALL | Ibrahima | 30*



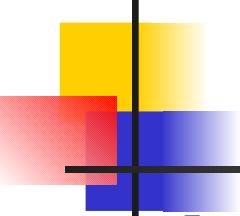
# Fonction Explode

```
<html> <head> <title>Exemples de Explode</title> </head> <body>
<?php
if (!file_exists(" test.txt ")) {
    print "<H3><BR>Erreur, fichier compteur manquant<BR>";
    exit;
} else {
    $fd = fopen($fic,"r");
    while (!feof($fd)) {
        $ligne = fgets($fd,255);
        $tab=explode("|",$ligne);
        print "Nom : $tab[0]<br>";
        print "Prénom : $tab[1]<br>";
        print "Age : $tab[2]<br>";
    }
    fclose($fd);
}
php?>
</body></html>
```



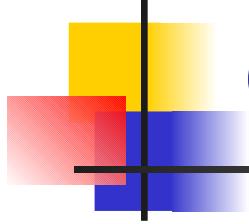
# Fonctions Mathématiques

- Elles y sont toutes !
  - Abs, cos, sin, tan, sqrt, exp, ...
  - pi( )



# Chaînes de caractères

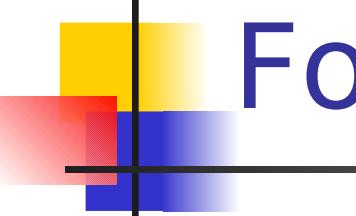
- ***strpos (chaîne, sous chaîne)*** : retourne la position de la sous chaîne dans la chaîne. Dans le cas où la chaîne existe en plusieurs exemplaires, c'est la position de la première occurrence qui est retournée. ***strrpos*** retourne quand à elle la position de la dernière occurrence.
- ***strstr (chaîne, sous chaîne)*** : retourne la portion de la chaîne à partir de la première occurrence de la sous chaîne.
- ***foreach (nom tableau)*** : A chaque appel, cette fonction retourne la valeur suivante du tableau.
- ***strlen (chaîne)*** : retourne la taille de la chaîne.
- ***strtolower|strtoupper (chaîne)*** : retourne la chaîne passée en paramètres an minuscules (resp. majuscules).
- ***str\_replace (car d'origine, car de destination, chaîne)*** : remplace le caractère d'origine par le caractère de destination dans la chaîne.
- ***g*** supprime les caractères invisibles (espaces,\n, ...) au début et à la fin de la chaîne.
- ***ereg(chaîne à chercher, chaîne)*** : retourne vrai si la chaîne à chercher (sous forme de chaîne ou sous forme d'expression régulière) est contenue dans chaîne.



# Achèvement d'une exécution

exit                      arrêt brutal de la page

die(\$ch)                arrêt avec une page-message



# Fonctions each() et list()

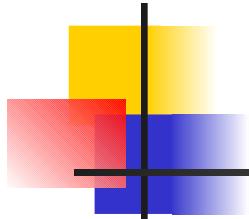
Ces fonctions sont aussi étroitement liées aux boucles.

## *each(..)*

Cette fonction permet de parcourir tous les éléments d'un tableau sans se soucier de ses bornes. Elle retourne la combinaison clé-valeur courante du tableau passé en paramètre, puis se positionne sur l'élément suivant, et cela du premier au dernier indice. Lorsque la fin du tableau est atteinte, each( ) retourne la valeur faux (false).

## *list(..)*

Cette fonction est très souvent associée à la fonction each(), elle permet d'affecter les éléments du tableau dans des valeurs distinctes.



# Parcours d'un tableau: fonctions each() et list()

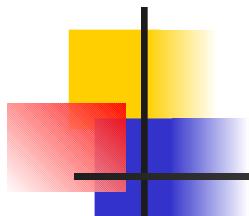
exemple :

```
$tableau = array("val1","val2","val3","val4"); //on crée un tableau  
           avec 4 valeurs  
while ($var = each($tableau)) {  
    echo "$var[0] : $var[1]";  
}
```

Ce code produit le résultat suivant :

0 : val1  
1 : val2  
2 : val3  
3 : val4

ici l'indice est affecté au premier élément de \$var et la valeur au deuxième élément \$var



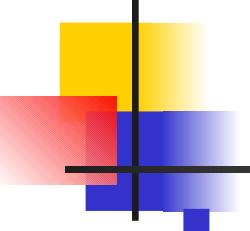
# Fonctions *each()* et *list()*

```
$tableau = array("val1","val2","val3","val4"); //on crée un
       tableau avec 4 valeurs
while (list($cle, $valeur)= each($tableau)) {
    echo "$cle : $valeur";
}
```

Attention : Il faut noter qu'une fois la boucle "while" terminée, le pointeur de tableau se trouve à la fin.

- Si vous voulez parcourir le tableau à nouveau, vous devez remettre le pointeur à la position zéro à l'aide de la fonction *reset()*

```
<?
    reset($tableau);
?>
```



# Trier un tableau

**Les fonctions sort() et rsort() permettent de trier un tableau par valeurs croissantes ou décroissante. Les indices sont changés.**

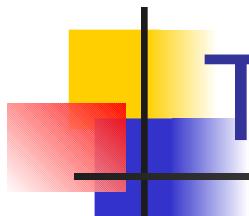
**void sort(tableau nomduTableau);**

- Petit exemple permettant de trier le tableau en ordre inverse.**

<?

```
$tableau = array(1 => 1, 2=>2, 3=>3);
rsort($tableau);
while(list($cle,$val) = each($tableau))
{
    echo "$cle : $val<br>";
}
```

?>.



# Trier un tableau

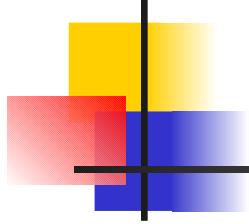
- On peut également trier un tableau par indice, à l'aide des fonctions `ksort()` et `krsort()`, qui s'utilisent de la même manière que les fonctions précédentes.

```
void ksort(tableau nomduetableau);
```

- Exemple: affichage d'une adresse IP inversée .

```
<?
```

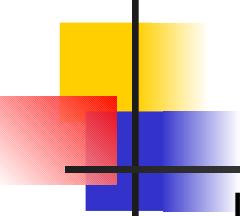
```
// recuperation de l'adresse ip
$ip = getenv("REMOTE_ADDR");
// on en crée un tableau
$array = explode(".", $ip);
// on trie le tableau par indice décroissant
// ce qui permet dans le cas présent de l'inverser
krsort($array);
// on re-transforme notre tableau inversé en chaîne
$rip = implode(".", $array);
?>
```



# Gestion de la date

- Fonction Date : nombreux paramètres

```
echo date("d-m-Y"); // affiche : << 16-11-2009 >>
```



# Gestion des fichiers

La fonction de base est la fonction *fopen()*. C'est elle qui permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire. Sa syntaxe est :

*entier fopen(chaine nomdufichier, chaine mode);*

Différents modes disponibles

*r* : ouverture en lecture seulement

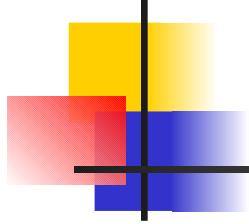
*w* : ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)

*a* : ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

*r+* : ouverture en lecture et écriture

*w+* : ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)

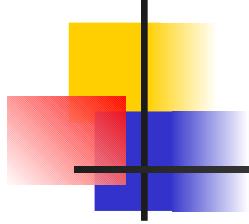
*a+* : ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas).



# Gestion des fichiers

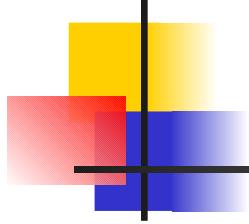
## **Exemples :**

```
$fp = fopen("../fichier.txt","r"); //lecture  
$fp = fopen("ftp://ucad.sn/pub/fichier.txt","w");  
          //écriture depuis début du fichier  
$fp =  
      fopen("http://www.nexen.com/fichier.txt","a");  
          //écriture depuis fin du fichier
```



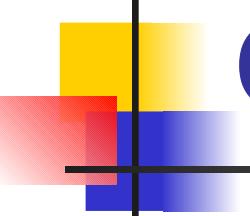
# Lecture dans un fichier

```
< ?  
$monfichier = fopen("monfichier.txt", "r") ; // ouverture en  
lecture  
if ( !($monfichier)) {  
    print(" Impossible d'ouvrir le fichier ") ;  
    exit ;  
}  
while ( !feof($monfichier) ) {  
    $ligne = fgets($monfichier,255); // 255 caractères max. ou bien  
    fin de ligne.  
    print "$ligne <BR>" ;  
}  
fclose ($monfichier) ;  
?>
```



# Écriture dans un fichier

```
< ?  
$monFichier = fopen("monfichier.txt"," w") ; // ouverture en écriture  
if ( !($monfichier) ) {  
    print(" Impossible de créer le fichier \n") ;  
    exit ;  
}  
  
fputs($monfichier, "ligne 1") ; // on écrit deux lignes  
fputs($monfichier, "ligne 2") ;  
  
fclose($monfichier) ; // on ferme le fichier, on libère les ressources  
?>
```



# Gestion des répertoires

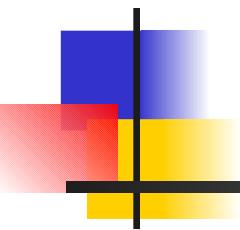
```
$bool = chdir ("nom_rep"); //positionne le répertoire courant  
$bool = mkdir ("nom_rep", int_mode);  
    //mode : permission d'accès unix (Windows l'ignore)  
$bool = rmdir ("nom_rep"); //si vide, détruit le répertoire
```

Ouvrir, afficher,  
positionner  
créer, détruire

```
$dd = opendir ("nom_rep"); //ouverture (par défaut le répertoire courant)  
$str = readdir ($dd); //lit la prochaine entrée du répertoire  
rewinddir ($dd); //se place sur la 1er entrée du répertoire  
closedir ($dd); //ferme
```

```
<?php...  
if (!mkdir ("/temp", 0700))  
echo "creation impossible";?>
```

```
<?php $rep=opendir("..")  
while ($nom_fich =readdir ("$rep")) echo "$nom_fic";  
  
chdir ("/temp"); ?>
```



# Envoyer des fichiers vers un serveur

# « upload » d'un fichier

- On se propose de transférer un fichier situé sur le poste du client vers le serveur
- C'est exactement, appliqué au protocole HTTP, un document attaché à un mail (protocole SMTP)
- Exemples
  - Transférer des photos numériques vers un centre de tirage
  - Transférer un article vers un site de conférences
  - Etc.

The screenshot shows a web interface for "FDI Service Numérique". At the top, there's a navigation bar with "Aide", the logo "FDI Service Numérique", and a "Print@FUJICOLOR" logo featuring a group of people. Below the bar, a green banner says "Pour importer vos images :". The main area contains a numbered list of steps:

1. Sélectionnez le nombre d'images que vous désirez importer.
2. Cliquez sur le bouton "Parcourir" et choisissez vos images.
3. Cliquez sur le bouton "Importer"
4. Une fois le téléchargement terminé, vos images apparaissent à l'écran sous forme de vignettes. Cliquez ensuite sur "Boutique Photo" ou "Tirage Express".

Below the list, there's a small image of a sunset and the text: "Transférez vos photos aisément à l'aide du procédé [Cliquer-Déplacer](#)". A link "Informations nécessaires" is also present. At the bottom, there's a section for selecting file counts (3) and three input fields for "Fichier 1", "Fichier 2", and "Fichier 3", each with a "Parcourir..." button. A red "Importer" button is at the bottom right.

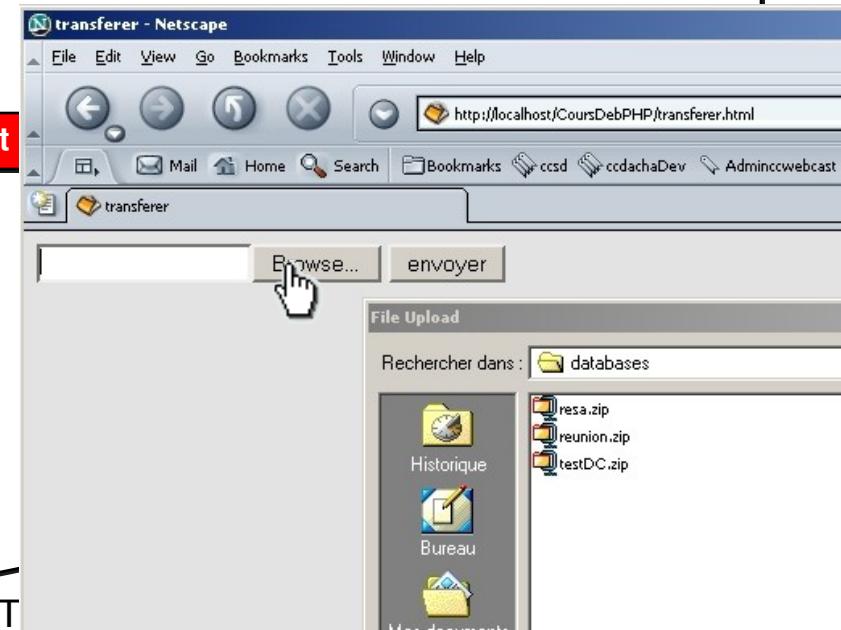
# La balise HTML <input type="file">

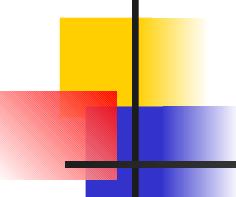
- Permet de « browser » le disque local pour sélectionner un fichier
- Permet « d'attacher » ce fichier à la suite des variables d'un formulaire, sous forme d'un message « multipart »

```
<HTML><HEAD><TITLE> transferer </TITLE></HEAD>
<BODY BGCOLOR="#e3e3e3">
<form name="transferer"
      ENCTYPE="multipart/form-data"
      action="stocke.php"
      method="post">
    <input name="monFichier"
          type="file">
    <input type="submit"
          value="envoyer">
</form>
</BODY>
</HTML>
```

transférer.html

Le serveur doit



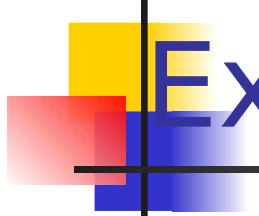


il n'y a ~ rien à faire !

# Sur le serveur...

stocke.php

```
<?php
define ("WEB", "http://localhost/CoursDebPHP/monDossier/");
define ("HTDOCS", "C:\\Program Files\\Apache Group\\Apache\\htdocs\\");
//
$dossier = HTDOCS."CoursDebPHP\\monDossier\\";
//
echo "Fichier original : ".$_FILES["monFichier"]["name"]."<br>";
echo "Fichier temporaire : ".$_FILES["monFichier"]["tmp_name"]."<br>";
echo "Taille du fichier : ".$_FILES["monFichier"]["size"]."octets";
//
copy ($_FILES["monFichier"]["tmp_name"],$dossier".$_FILES["monFichier"]["name"]);
//
echo "<p><a href=".WEB.$_FILES["monFichier"]["name"].">" ;
echo "Voir le fichier transf&eacute;r&eacute;</a>";
?>
```



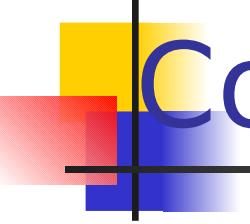
# Exercice

- Télécharger un texte et le stocker sur le serveur
- Proposer l'affichage de ce texte dans une page HTML en faisant surbriller toutes les zones de texte correspondant à un motif choisi par l'utilisateur
- *Cet exercice permettra de manipuler des fonctions manipulant des fichiers*
- *Cet exercice permettra de poser le problème des sessions*

# Correction -1

surbrille.php

```
<html><head>
<style type="text/css">.sur {background-color : Yellow}</style>
</head><body bgcolor="#e3e3e3>
<form name=chMotif action=surbrille.php>
motif &agrave; rechercher : <input name="motif">
<input type="submit" value="chercher">
<input type="hidden" name="leFichier">
</form>
<hr>
<?php
define ("WEB","http://localhost/CoursDebPHP/monDossier/");
define ("HTDOCS","C:\\Program Files\\Apache Group\\Apache\\htdocs\\");
$dossier = HTDOCS."CoursDebPHP\\monDossier\\";
```



# Correction -2

surbrille.php (suite)

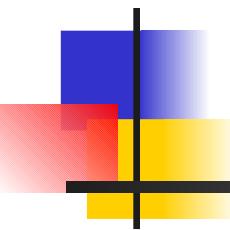
```
//Premier passage on vient du formulaire d'upload
//
if (isset($_FILES["monFichier"]["tmp_name"])){
    $fichier = $dossier.$_FILES["monFichier"]["name"];
    copy ($_FILES["monFichier"]["tmp_name"],$fichier);
    $fichier_2 = ereg_replace("\\\\","\\\\\\",$fichier);
    print("<script>document.chMotif.leFichier.value='".$fichier_2';</script>");
    $fd = fopen ($fichier,"r");
    while (!feof ($fd)) {
        $ligne = fgets($fd, 256);
        echo $ligne."<br>";
    }
    fclose ($fd);
} else {
```

# Correction -3

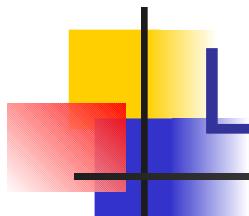
surbrille.php (suite et fin)

```
// Passage suivant on vient du formulaire de saisie du motif
$fichier = $_REQUEST["leFichier"];
$fichier = ereg_replace("\\\\\\\\","\\\\",$fichier);
$fd = fopen ($fichier,"r");
while (!feof ($fd)) {
    $ligne = fgets($fd, 256);
    $motif = $_REQUEST["motif"];
    $ligne = ereg_replace($motif,"<span class='sur'>$motif</span>",$ligne);
    echo $ligne."<br>";
}
fclose ($fd);
$fichier_2 = ereg_replace("\\\\","\\\\",$fichier);
print("<script>document.chMotif.leFichier.value='$fichier_2';</script>");

}
?>
<hr><body></html>
```

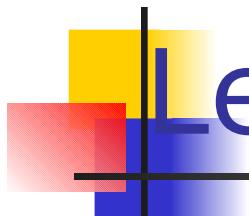


# Sessions et transactions



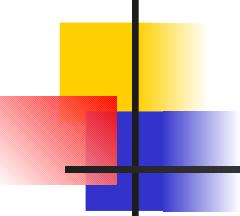
# Les transactions sur le Web

- Le problème
  - Vous vous connectez sur le serveur de votre banque avec nom et mot de passe
  - Vous circulez dans plusieurs pages (relevé de compte, virement, etc.)
  - Entre chaque page vous êtes déconnecté : **c'est la nature du Web !**
  - Or, vous ne redonnez pas votre nom et mot de passe à chaque page ...
  - ... comment ça marche ?



# Le mécanisme de sessions

- Le mode « déconnecté » du Web impose, si on veut implémenter la persistance d'informations entre les différentes pages, d'utiliser des mécanismes particuliers
  - Retransmettre les données nécessaires, de formulaire en formulaire, grâce à des champs cachés (<input type=hidden ...)
  - Utiliser un mécanisme de session, autorisant de stocker sur le serveur des données temporaires
    - Dans ce cas, de page en page, on ne transmet qu'un identifiant, celui-ci étant géré automatiquement par le système
    - On peut aussi éventuellement utiliser un *cookie* pour s'identifier



# Construire une session

- On appelle la fonction `session_start()`
- On demande la conservation de variables au travers d'un tableau associatif `http_session_vars["ma_variable"]`
- C'est tout ! (**ou presque...**)



# Détaillons ...

session.html

```
<html><body bgcolor="#e3e3e3">
<form name="tst" action="un.php">
Votre code :
<input name="login">
Votre mot de passe :<input type="password">
<input type="submit" value="Envoyer" />
</form></body>
```

émission post

```
<?
session_start();
$HTTP_SESSION_VARS["utilisateur"]=$_REQUEST["login"];
$HTTP_SESSION_VARS["pass_phrase"]=$_REQUEST["pwd"];
?>
<body bgcolor="#e3e3e3">
<form name="tst2" action="deux.php" method="get">
```

C:\temp\sess\_ebd66a6730f0c63670f686a1fc69f1c1  
utilisateur|s:4:"Emile";pass\_phrase|s:6:"tagada";

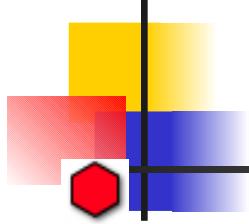
un.php

deux.php

```
<?
session_start();
print ("Bonjour ".$HTTP_SESSION_VARS["utilisateur"]);
print (" Votre mot de passe est : ".$HTTP_SESSION_VARS["pass_phrase"]);
print (" merci pour votre remarque : ".$_REQUEST["remarque"]);
session_destroy();
?>
```

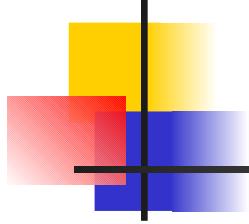
Lecture : on utilise le PHPSESSID pour ouvrir le bon fichier

tiquement



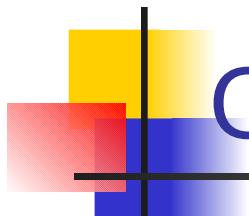
# Attention !

- A ne jamais mettre quelque chose avant l'instruction `session_start` !
  - Parce que php écrit un *header http* pour glisser, entre-autre, sa variable cachée
- La sanction serait un warning, éventuellement un fonctionnement anormal du code



# Exercices

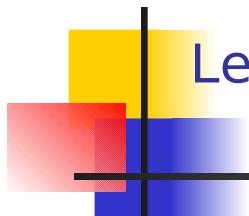
1. Faire un système qui compte combien de fois VOUS accédez à une page et non combien de fois cette page est accédée !
2. Imaginer le compteur classique qui compte le nombre d'accès à une page
3. Imaginez comment limiter le temps de stationnement sur une page



# Correction : compter les accès

comptons.php

```
<?php
session_start();
if (!isset($HTTP_SESSION_VARS['compteur']))
    $HTTP_SESSION_VARS['compteur']=1;
else
    $HTTP_SESSION_VARS['compteur']++;
?>
Vous &ecirc;tes venu
<font size="6" color ="red">
<? echo $HTTP_SESSION_VARS['compteur'] ?>
</font> fois
<form action=comptons.php>
<input type=submit value="continuer">
</form>
```



## Le stationnement limité ou la déconnexion automatique d'une application

- Définir une constante de temps maximum : MAXI\_INACTIF
- Ouvrir la session, prendre le temps d'arrivée : time() et le stocker en variable de session : \$HTTP\_SESSION\_VARS['temps']
- Lorsque l'on change de page dans l'application, la nouvelle page effectue :

```
if (time() - $HTTP_SESSION_VARS['temps'] > MAXI_INACTIF  
    on sort en temps dépassé
```

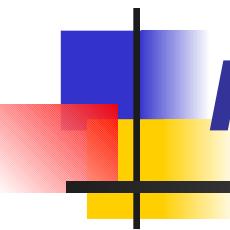
Else

on stocke le nouveau temps et on continue

# Correction :

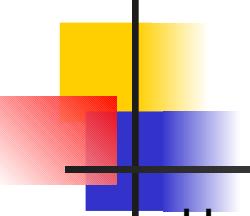
limite.php

```
<?php
define ("MAX_INACTIF",10);
session_start();
if (!isset($HTTP_SESSION_VARS['compteur'])){
    $HTTP_SESSION_VARS['compteur']=1;
    $HTTP_SESSION_VARS['temps']=time();
}else{
    if ((time()-$HTTP_SESSION_VARS['temps'])>MAX_INACTIF){
        session_destroy();
        die("Temps d&eacute;pass&eacute; !");
    }
    $HTTP_SESSION_VARS['compteur]++;
    $HTTP_SESSION_VARS['temps']=time();
}
?>
Vous &ecirc;tes venu
<font size="6" color ="red">
<? echo $HTTP_SESSION_VARS['compteur'] ?>
</font> fois
<form action=limite.php>
<input type=submit value="continuer"> avant 10 secondes
</form>
```



# ***INTERACTION PHP/MYSQL***

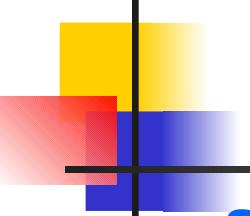
---



# Introduction aux bases de données

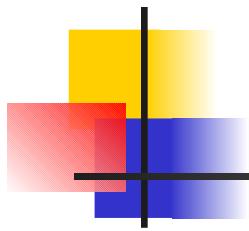
## Une base de données

- C'est un ensemble de données organisées
- Une base de données contient des tables qui décrivent les types des données
- Les tables contiennent à leur tour des enregistrements qui sont les vraies données
- En utilisant un identifiant commun entre les tables il est possible de les « relier » entre elles.
- Jargon
  - Une relation est une table comportant des colonnes (appelées aussi attributs) dont le nom et le type caractérise le contenu qui sera inséré dans la table.
  - Les enregistrements sont également appelés des tuples.



# SGBD

- **SGBD** : Système de Gestion de Bases de Données
- Fournit des méthodes **efficaces** pour gérer des données qui répondent à une structure (**un schéma**) précis.
- Rechercher des données : **requêtes**
- Ajouter, supprimer, modifier des données : **mises à jour**
- Traite de manière rapide de grandes quantités de données.
- De nombreux produits commerciaux (Oracle, Microsoft SQLServer, IBM DB-2, Microsoft Access. . . ) et libres (MySQL, PostGreSQL)

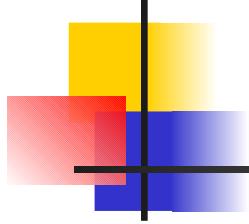


# Modèle relationnel

- Modèle le plus répandu et le plus classique.
- Les données sont organisées en des **tables**, chacune des colonnes représentant un **attribut** des données.

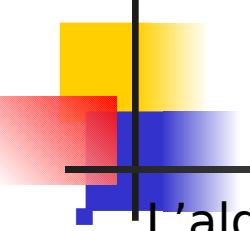
## Example

Prénom	Nom	Date de naissance
Jean	Dupont	1967-08-07
Pascale	Dupuis	1981-09-12
Alfred	Lambert	NULL



## Modèle relationnel (2)

- Chaque attribut (colonne) est **typé**.
- **SQL** (**S**tructured **Q**uery **L**anguage) : langage standard de requête et de mise à jour des données (petites variantes suivant les SGBD).



# Algèbre relationnelle

L'algèbre relationnelle regroupe toutes les opérations possibles sur les relations. Voici la liste des opérations possibles :

- **Projection** : on ne sélectionne qu'un ou plusieurs attributs d'une relation (on ignore les autres). Par exemple n'afficher que les colonnes *nom* et *prénom* de la table **Personnes**.
- **Jointure** : on fabrique une nouvelle relation à partir de 2 ou plusieurs autres en prenant comme pivot 1 ou plusieurs attributs. Par exemple, on concatène la table du carnet d'adresse et celle des inscrits à la bibliothèque en fonction du nom de famille (c'est typiquement du recouplement de fichiers).
- **Sélection** : on sélectionne tous les tuples ou bien seulement une partie en fonction de critères de sélection qui portent sur les valeurs des attributs. Par exemple n'afficher que les lignes de la table **Personnes** qui vérifient la condition suivante : le nom est « Dupuis ».
- Cette algèbre est facilement possible avec les commandes de MySQL (**SELECT... FROM... WHERE...**).

# Projection

Nom	Prénom	E-Mail	Laboratoire
Charnay	Daniel	charnay@in2p3.fr	Centre de Calcul
Macchi	Pierre-Etienne	macchi@in2p3.fr	Centre de Calcul

On projette la table  
*Personnes* sur les colonnes  
nom, prénom

SELECT nom,prénom FROM  
*Personnes*

Nom	Prénom
Charnay	Daniel
Macchi	Pierre-Etienne

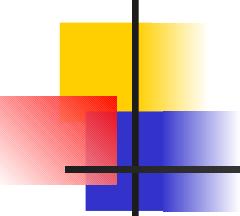
# Sélection

Nom	Prénom	E-Mail	Laboratoire
Charnay	Daniel	charnay@in2p3.fr	Centre de Calcul
Macchi	Pierre-Etienne	macchi@in2p3.fr	Centre de Calcul
Boutheri n	Bernard	Boutherin@isn.in2p 3.fr	ISN

On ne sélectionne que les tuples dont l'attribut laboratoire est égal à ISN

SELECT \* FROM Personnes WHERE Laboratoire="ISN"

Nom	Prénom	E-Mail	Laboratoire
Boutheri n	Bernard	Boutherin@isn.in2p 3.fr	ISN



# Jointure

*Personne*

Nom	Prénom	E-Mail	Labo
Charnay	Daniel	charnay@in2p3.fr	CC
Macchi	Pierre-Etienne	macchi@in2p3.fr	CC
Boutherin	Bernard	Boutherin@isn.in2p3.fr	ISN

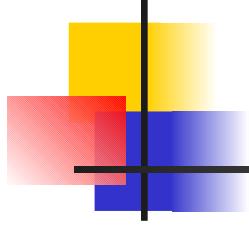
*Unité*

Ville	Labo
Villeurbanne	CC
Villeurbanne	CC
Grenoble	ISN

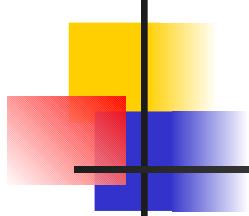
On joint les 2 tables grâce à la colonne Labo.

SELECT Personnes.Nom, Personnes.Prénom, Unité.Ville  
 FROM Personnes, Unité WHERE  
 Personnes.Labo=Unité.Labo

Nom	Prénom	Ville
Charnay	Daniel	Villeurbanne
Macchi	Pierre-Etienne	Villeurbanne
Boutherin	Bernard	Grenoble



# **Un SGBD : MySQL**



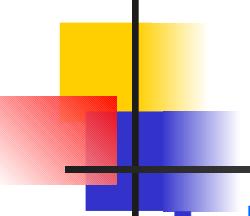
# Introduction

MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requête vers les bases de données exploitant le modèle relationnel.

Il en reprend la syntaxe mais n'en conserve pas toute la puissance puisque de nombreuses fonctionnalités de SQL n'apparaissent pas dans MySQL (sélections imbriquées, clés étrangères...)

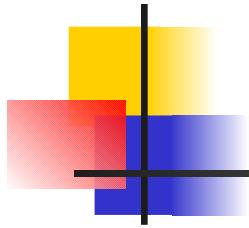
Le serveur de base de données MySQL est très souvent utilisé avec le langage de création de pages web dynamiques : PHP.

Il sera question ici des commandes MySQL utilisables via PHP dans les conditions typiques d'utilisation dans le cadre de la gestion d'un site web dynamique.

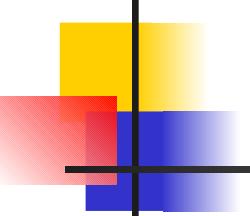


# Types de données

- **INT** : entier (42)
- **REAL** : nombre en virgule flottante (3.14159)
- **VARCHAR(N)** : chaîne de caractères ayant au plus N caractères ; les valeurs sont délimitées par des apostrophes ('Ceci est une chaîne').
- **TEXT** : long texte
- **DATE** : date (2005-11-08)
- **TIME** : temps (09:30:00)
- **NULL** : valeur spéciale
  - Dénote l'absence de valeur.
  - Différent de 0, de . . .
  - Une comparaison normale (=,<>) avec NULL renvoie toujours FAUX.
  - **IS NULL**, **IS NOT NULL** peuvent être utilisées pour tester une valeur.
  - Chacune des colonnes doit être déclarée comme acceptant ou non la valeur NULL.



# Administration en mode ligne



# Commandes en mode ligne

- **Moniteur MySQL :**

```
mysql -h [database_host] -u [utilisateur] -p [password]
```

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

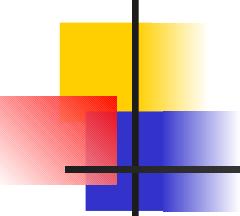
Your MySQL connection id is 67831 to server version: 3.23.54

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql> show databases;
```

- **Administration**

- [mysqladmin](#) : administration en mode ligne
- [mysqlimport](#) : import de données à partir de fichiers textes
- [mysqldump](#) : export des données dans un fichier texte



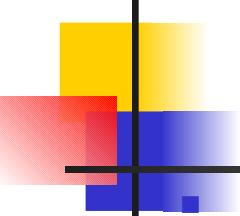
# Administration

## ■ Changer le mot de passe d'administration

- Lors de l'installation de MySQL, l'utilisateur root (administrateur) n'a pas de mot de passe.  
Pour le changer, il faut, au préalable, se positionner sur le répertoire /usr/bin/ et taper la commande  
***mysqladmin -u root password 'mot de passe'.***
- *Ou se connecter à MySQL, à la base MySQL.*
- *Taper, ensuite, la commande :*

***UPDATE user SET***

***password=password('nouveau\_mot\_de\_passe')  
WHERE user='root';***



# Envoyer des requêtes

Un exécutable (**mysql**) permet de se connecter à MySQL pour lui envoyer des requêtes.

**mysql -h host -u user -p pass db**

ou

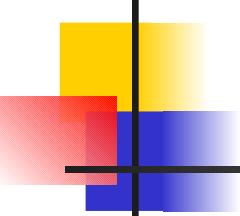
**mysql -h host -u user -p db** , puis le mot de passe

Chaque paramètre de la ligne de commande est facultatif. Par défaut :

- **host** = localhost
- **user** = root sous windows et sous linux

Une fois connecté, vous pouvez envoyer des commandes à MySQL.

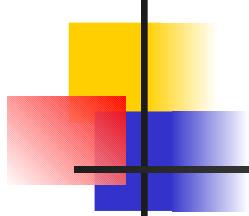
- **use** database; #changer de base de données
- **show** tables; #montrer les tables de la base
- **show** columns from table; #montrer les champs de la table
- **select \* from table;**
- etc..



# COMMANDES SQL DE BASE (1)

- Avant d'utiliser les fonctions PHP pour utiliser MySQL, il faut, au préliminaire, avoir quelques bases de SQL.
- Pour pouvoir envoyer des commandes sql à mysql, il faut d'abord lancer l'exécutable mysql.  
Sous [windows](#), il s'agit de [mysql.exe](#) qui doit se trouver dans [c:\mysql\bin\](#)
- Pour commencer, il faut créer une base de données et une table.
- ***Création de la base de données***

**CREATE DATABASE basel;**

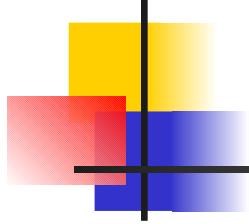


# COMMANDES SQL DE BASE (2)

- ***Création de la table***

```
CREATE TABLE test (
    id int(11) DEFAULT '0' NOT NULL auto_increment,
    nom varchar(20) NOT NULL,
    email varchar(50),
    dateheure datetime,
    PRIMARY KEY (id)
);
```

- Nous avons donc créé une table appelée ***test***, contenant 4 champs :
  - l'identifiant ***id*** qui permet de différencier les données. C'est un nombre entier qui s'auto-incrémente à chaque ajout.
  - Le ***nom*** et l'***email*** : chaînes pouvant contenir respectivement 20 et 50 caractères.
  - La date et l'heure ***dateheure***, stockée sous le format AAAA-MM-DD HH:MM:SS (par exemple 2006-03-01 22:54:03)



# Administration avec l'outil web phpMyAdmin

# PHPMyAdmin : Présentation générale

- Interface Web, écrite en PHP, de gestion d'une base de données MySQL
- Chaque opération réalisée est traduite en un ordre SQL, qui est affiché.

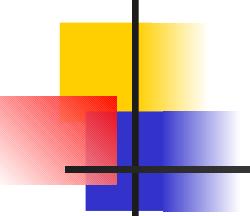
Cet outil permet de :

- créer de nouvelles bases
- créer/modifier/supprimer des tables
- afficher/ajouter/modifier/supprimer des tuples dans des tables
- effectuer des sauvegarde de la structure et/ou des données
- effectuer n'importe quelle requête
- gérer les privilèges des utilisateurs

The screenshot shows the PHPMyAdmin interface for the 'forum' database. The left sidebar lists databases: 'forum (4)' (selected), 'informatique', 'news', 'test', and 'uforum'. The main panel displays the structure of the 'mforum' table:

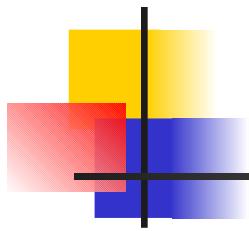
Champ	Type	Atributs	Null	Défaut	Extra	Action
<input type="checkbox"/> id	bigint(20)	UNSIGNED	Non		auto_increment	<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> title	tinytext		Non			<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> msg	text		Non			<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> hits	mediumint(8)	UNSIGNED	Non	0		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> thread_idx	bigint(20)	UNSIGNED	Non	0		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> author_idx	bigint(20)	UNSIGNED	Non	0		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>
<input type="checkbox"/> date	datetime		Non	0000-00-00 00:00:00		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Primaire</a> <a href="#">Index</a> <a href="#">Unique</a> <a href="#">Texte</a> <a href="#">entier</a>

Below the table structure, there are sections for 'Index' (with a 'Documentation' link), 'Espace utilisé' (with statistics like 'Données: 8 908 Octets'), and 'Statistiques' (with metrics like 'Format: dynamique'). At the bottom, there's a query editor with the command: 'SELECT \* FROM `mforum` WHERE 1'.



# Requêtes SQL: Manipulation d'une base de données

- Une fois, la table créée, on peut :
  - Insérer des données avec une requête **INSERT**
  - Sélectionner des données avec une requête **SELECT**
  - Retirer des données avec une requête **DELETE**
  - Mettre à jour des données avec une requête **UPDATE**
- **Remarque** : Les commandes de suppression et de mises à jour sont des variantes du **SELECT**.



# Requêtes SQL: Insérer des données (1)

- Pour insérer des données *Tuple* dans la table *Table*, on utilise une des requêtes suivantes :

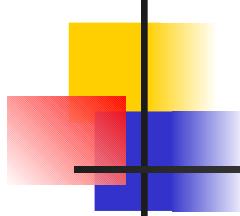
**INSERT INTO** *Table*

**VALUES** *Tuple*

**INSERT INTO** *Table nuplet\_d\_Attributs*

**VALUES** *nuplet\_de\_valeurs*

- Dans le dernier cas, les attributs sans valeurs seront initiés à NULL.

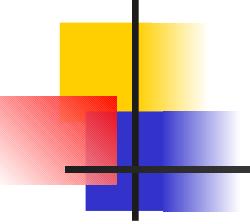


## Requêtes SQL: Insérer des données (2)

### Exemple

```
INSERT INTO Films  
VALUES ('Tootsie', 1982, 'US')
```

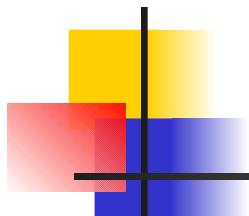
```
INSERT INTO Films (Titre, Annee)  
VALUES ('Wallace et Gromit le mystère du lapin-garou', 2005)
```



# Requêtes SQL: Sélectionner des données (1)

- Pour extraire de la base de données des informations, on utilise la commande **select**.
- **Syntaxe générale :**

```
SELECT [ DISTINCT ] attributs
      [ INTO OUTFILE fichier ]
      [ FROM relation ]
      [ WHERE condition ]
      [ GROUP BY attributs [ ASC | DESC ] ]
      [ HAVING condition ]
      [ ORDER BY attributs ]
      [ LIMIT [a,] b ]
```



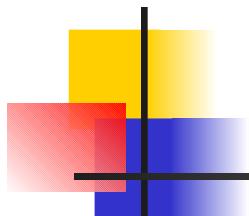
## Requêtes SQL: Sélectionner des données (2)

Pour sélectionner des données *nuplet\_d\_Attributs* dans la table *Table* selon une condition *Cond*, on utilise la requête suivante :

```
SELECT nuplet_d_Attributs
FROM Table
WHERE Cond
```

### Exemple

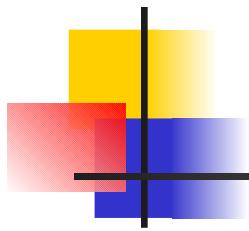
```
SELECT Titre
FROM Films
WHERE Annee > 1980
```



# Requêtes SQL: Sélectionner des données (3)

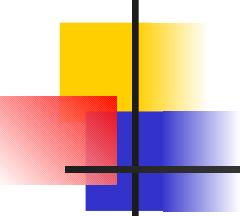
## SELECT

- \* pour sélectionner tous les attributs
- liste d'attributs séparés par une virgule
- FROM
  - liste de tables séparés par une virgule
- WHERE
  - clause optionnelle
  - condition basée sur les opérateurs : AND, OR, LIKE, =, <>, >, >=, etc.
- ORDER BY
  - clause optionnelle
  - noms de champs séparés par une virgule
  - Permet de définir l'ordre (**ASC**endant par défaut ou **DESC**endant) dans l'envoi des résultats.



## Requêtes SQL: Sélectionner des données (4)

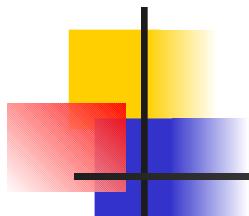
- **DISTINCT**
  - Permet d'ignorer les doublons de ligne de résultat
- **GROUP BY**
  - Permet de grouper les lignes de résultats selon un ou des attributs.
- **HAVING**
  - Définie un ou des critères de sélection sur des ensembles de valeurs d'attributs après groupement.
- **LIMIT**
  - Permet de limiter le nombre de lignes du résultats



## Requêtes SQL: Sélectionner des données (5)

### Exemple:

- Pour sélectionner tous les enregistrements d'une relation :  
**SELECT \* FROM *relation***
- Pour sélectionner toutes les valeurs d'un seul attribut :  
**SELECT *attribut* FROM *relation***
- Pour éliminer les doublons :  
**SELECT DISTINCT *attribut* FROM *relation***
- Pour trier les valeurs en ordre croissant :  
**SELECT DISTINCT *attribut* FROM *relation* ORDER BY *attribut ASC***
- Pour se limiter aux **num** premiers résultats :  
**SELECT DISTINCT *attribut* FROM *relation* ORDER BY *attribut ASC*  
LIMIT num**
- Pour ne sélectionner que ceux qui satisfont à une condition :  
**SELECT DISTINCT *attribut* FROM *relation* WHERE *condition* ORDER BY *attribut ASC* LIMIT num**



# Fonctions de comparaison de chaînes

Le mot clé **LIKE** permet de comparer deux chaînes.

Le caractère '**%**' est spécial et signifie : 0 ou plusieurs caractères.

Le caractère '**\_**' est spécial et signifie : 1 seul caractère, n'importe lequel.

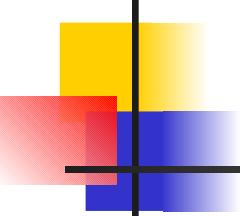
L'exemple suivant permet de rechercher tous les clients dont le prénom commence par 'Jean', cela peut être 'Jean-Pierre', etc... :

```
SELECT nom  
FROM clients  
WHERE prénom LIKE 'Jean%'
```

Pour utiliser les caractères spéciaux ci-dessus en leur enlevant leur fonction spéciale, il faut les faire précédé d'un antislash : '\\_'.

Exemple, pour lister les produits dont le code commence par la chaîne '**\\_XE**' :

```
SELECT *  
FROM produit  
WHERE code LIKE '\_XE%'
```



# Requêtes SQL: Retirer et modifier des données

- Pour retirer les données de la table *Table* qui correspondent à la condition *Cond*, on utilise la requête suivante :

**DELETE FROM *Table* WHERE *Cond***

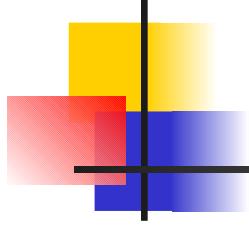
- Pour mettre à jour les données de la table *Table* qui correspondent à la condition *Cond*, on utilise la requête suivante :

**UPDATE *Table* SET *attribut* = *nouvelle***

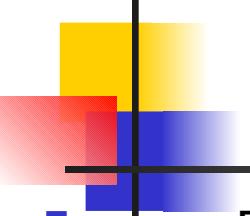
## Exemple

```
DELETE FROM Films  
WHERE Annee>2005
```

```
UPDATE Films  
SET Pays='RU'  
WHERE Pays='UK'
```



# Interface avec PHP



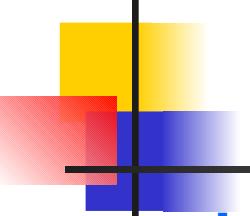
# Connexion au serveur des données

- Pour se connecter à une base depuis un script php, il faut spécifier un nom de **serveur**, un nom **d'utilisateur**, un **mot de passe** et un nom de **base**.
- Aucune connexion n'est possible sans authentification auprès du serveur de base de données.
- Les actions possibles de l'utilisateur sur la base à laquelle il se connecte dépendent des droits qui lui auront été fournis par l'administrateur de la base de données.

## Les fonctions de connexion :

**`mysql_connect($server,$user,$password)`** :

- permet de se connecter au serveur **\$server** en tant qu'utilisateur **\$user** avec le mot de passe **\$password**, retourne l'identifiant de connexion si succès, FALSE sinon



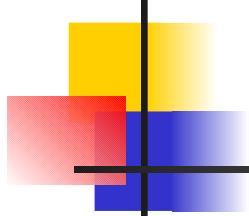
# Exemple (1)

## La déclaration des variables -non visibles aux visiteurs-

```
<?php  
$user= "root";  
$passwd="";  
$serveur=localhost;  
$bdd="ma_base";
```

## Connexion au serveur de données

```
<?php  
if( mysql_connect("localhost" , "root" , "") > 0 )  
{ echo "Connexion réussie ! " ;  
}else{  
echo "Connexion impossible ! " ;  
}  
?>
```



# Sélection de la base

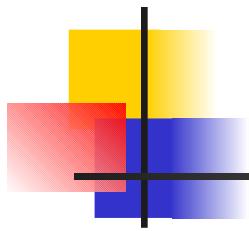
**mysql\_select\_db(\$base[\$id]) :**

- permet de choisir la base **\$base**, retourne TRUE en cas de succès, sinon FALSE

**mysql\_close([\$id]) :**

- permet de fermer la connexion

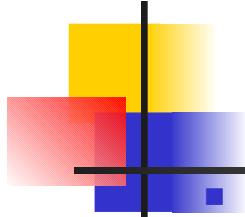
Les identifiants de connexion ne sont pas nécessaires si on ne se connecte qu'à une seule base à la fois, ils permettent seulement de lever toute ambiguïté en cas de connexions multiples.



# Exemple (2)

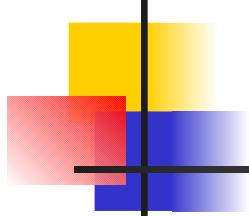
## *Selection de la base*

```
<?php  
if( mysql_select_db( "ma_base " ) == True )  
{  
    echo " Sélection de la base réussie ";  
} else  
{  
    echo" Sélection de la base impossible";  
}  
?>
```



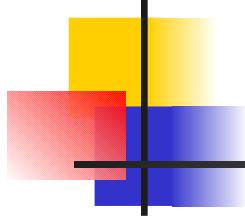
## Exemple (3) :

- ```
@mysql_connect("localhost","root","");
die("Echec de connexion au serveur.");
```
- **`@mysql_select_db("ma_base") or die("Echec de sélection de la base.");`**
  - Cet exemple est équivalent au précédent mais plus court à écrire. Le symbole **@** (arobase) permet d'éviter le renvoie de valeur par la fonction qu'il précède.
  - On pourra avantageusement intégrer ce code dans un fichier que l'on pourra joindre par **include()**. C'est aussi un moyen de sécuriser le mot de passe de connexion.



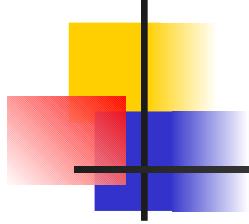
# Exécution d'une requête SQL: Creation et envoi de la requête

- int **mysql\_query** (string query, int link\_identifier )  
Contrairement aux autres fonctions (connexion et sélection), La valeur de retour est très importante, elle est utilisée pour retrouver les données rapatriées par une requête de sélection
- Stockage du résultat de la requête **mysql\_query** dans une variable **\$resultat**  
`$query = "SELECT nom,url FROM sites ORDER BY nom";  
$result = mysql_query($query);`



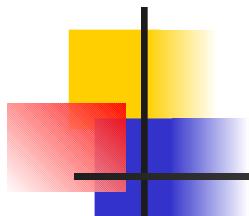
# Exécution d'une requête SQL: Traitement des résultats (1)

- Découper les lignes du résultat en colonnes avec `mysql_fetch_row`  
-ID, nom, pnom-  
`$row = mysql_fetch_row($result);`
- Affecter chaque colonne à une variable  
`$Nom = $row[0];`  
`$Url = $row[1];`
- Afficher les valeurs des variables  
`echo "<tr>\n<td><a href=\"$Url\">$Nom</a></td>\n<td>$Url</td>\n</tr>\n";`
- Utiliser une boucle pour répéter tant qu'il y a des lignes  
`while($row = mysql_fetch_row($result)){...`



# Exemple

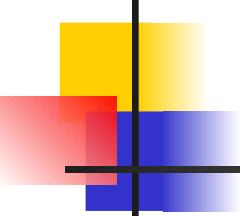
```
< ?  
while($row = mysql_fetch_row($result))  
{  
echo $row[0] ; //Afficher le champ Nom  
echo $row[1] ;//Afficher le champ Url  
echo <br>;  
}  
?>
```



# Exploitation des requêtes: Traitement des résultats (2)

- array **mysql\_fetch\_array** (int result, int result\_type ) Extrait la ligne sous forme d'un **tableau associatif**.
- **Exemple :**

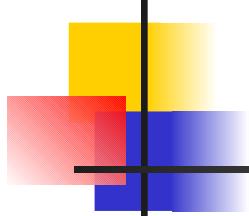
```
<?php
While($enregistrement = mysql_fetch_array($result))
{
    //Affiche le champ -prenom-
    echo $enregistrement["prenom"] ;
    //Affiche le champ -nom-
    echo $enregistrement["nom"] ;
    //Affiche le champ -date-
    echo $enregistrement["date"] ;
    echo « <br> »;
}
?>
```



# Exploitation des requêtes: Traitement des résultats (3)

- **object mysql\_fetch\_object (int result)** Retourne un objet (structure) correspondant à la ligne extraite de l'identificateur de résultat.
- **Exemple :**

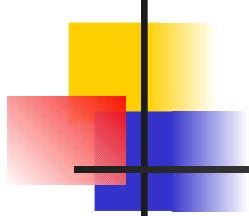
```
< ?php
$enregistrement = mysql_fetch_object($result);
        //Affiche le champ -prenom-
echo $enregistrement->prenom ;
        //Affiche le champ -nom-
echo $enregistrement->nom;
        //Affiche le champ -date-
echo $enregistrement->date ;
?>
```



# Exploitation des requêtes: Traitement des résultats (4)

- int **mysql\_num\_rows** (int *result*) Retourne le nombre d'enregistrements qui ont été retournés par la sélection.
- **Exemple :**

```
< ?php
    $requete = "SELECT * FROM membres" ;
    $result = mysql_query($requete);
    echo "il y a" .mysql_num_rows($result) .
membre(s)";
    while($enregistrement =
mysql_fetch_array($result))
    {
        echo $enregistrement["nom"]. " " .
$enregistrement["prenom"] ;
    }
?>
```



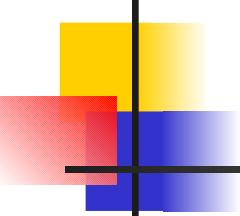
# Exploitation des requêtes: Traitement des résultats (5)

- int **mysql\_insert\_id** (int *link\_identifier* )

`mysql_insert_id()` retourne le dernier identifiant généré par un champs de type AUTO\_INCREMENTED.

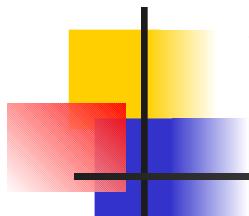
- **Exemple :**  
Suite à une requête d'insertion, on veut afficher le numéro auto incrémenté :  

```
< ?php  
echo "Votre numéro d'identifiant est : " .mysql_insert_id();  
?>
```



# Exploitation des requêtes: Traitement des résultats (6)

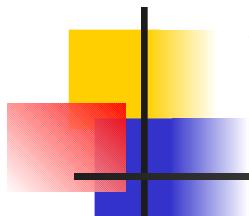
- int **mysql\_affected\_rows** (int *link\_identifier* )  
Cette fonction permet de fixer le nombre de lignes insérées, mises à jour ou supprimées par la précédente requête SQL (**INSERT**, **DELETE**, ... ou **UPDATE**) envoyée au serveur.
- **Exemple :**  
Suite à une commande "**UPDATE**", on voudrait savoir combien de lignes ont été modifiées :  
< ?  
echo **mysql\_affected\_rows()** ." enregistrement(s) modifiés";  
?>



# Exemple complet: Accès à une base de données via PHP -1

Le code

```
<html>
<head>
<title>Liens</title>
</head>
<body>
<table border="1" cellpadding="0" cellspacing="0">
<tr>
<th>Nom du site</th>
<th>URL</th>
</tr>
<?php
// Déclaration des paramètres de connexion
$host = la_machine;
// Généralement la machine est localhost
// c'est-a-dire la machine sur laquelle le script est hébergé
$user = votre_login;
$bdd = Nom_de_la_base_de_donnees;
$passwd = Mot_de_passe;
```

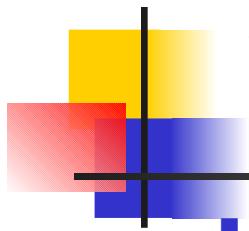


# Accès à une base de données via PHP

## -2

- Le code suite1

```
// Connexion au serveur  
mysql_connect($host, $user,$passwd) or die("erreur de  
connexion au serveur");  
mysql_select_db($bdd) or die("erreur de connexion a la  
base de donnees");  
// Creation et envoi de la requete  
$query = "SELECT nom,url FROM sites ORDER BY nom";  
$result = mysql_query($query);  
// Recuperation des resultats  
while($row = mysql_fetch_row($result)){  
$Nom = $row[0];  
$Url = $row[1];
```

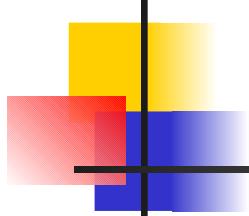


# Accès à une base de données via PHP

## -3

Le code suite2

```
echo "<tr>\n"
<td><a href=\"$Url\">$Nom</a></td>\n
<td>$Url</td>\n
</tr>\n";
}
// Deconnexion de la base de donnees
mysql_close();
?>
</tr>
</table>
</body>
</html>
```



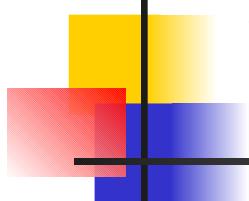
# Accès à une base de données via PHP méthode 2

- Regrouper les paramètres de connexion de la base dans un fichier **Connect.php**

```
<?php
Define (user, "personne");
Define(passwd, "perspass");
Define(serveur, "localhost");
Define(bdd, "personne");?>
```

- Créer le programme qui permet de se connecter à la base, exécuter une requête et afficher les résultats **pers.php**

```
<HTML><HEAD>
<TITLE>Connexion à MySQL</TITLE>
<LINK REL=Stylsheet HREF="personne.css" TYPE="text/css">
<BODY>
<H1> Requête sur la Table personne</H1>
```



# Accès à une base de données via PHP méthode 2

- **Partie php de pers.php**

```
<?php
Require("Connect.php");
$connexion = mysql_pconnect(serveur, user, passwd);
If(!connexion) { echo "Désolé, connexion à ".$serveur."impossible\n";exit; }
If (!mysql_select_db(base, $connexion))
{ echo "désolé...\n";exit; }
$resultat = mysql_query ("SELECT * FROM personne", $connexion);
if ($resultat){
While ($pers=mysql_fetch_object($resultat)){
Echo "$pers->ID, $pers->nom, $pers->pnom<BR>\n";} } else {
Echo "<B>erreur dans l'exécution de requête</B><BR>\n";
Echo "msg de sql:" . Mysql_error($connexion);}
?>
```

# Accès à une base de données

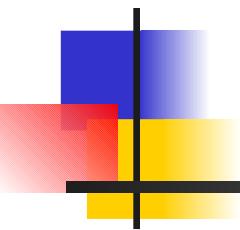
## Formulaire et PHP

### Partie formulaire myform.html

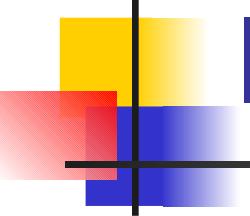
```
<FORM ACTION="MyPHP.php" METHOD=POST>
Ce formulaire vous permet d'indiquer des paramètre pour la recherche
<P>
NOM : <INPUT TYPE=TEXT SIZE=20 NAME='nom' VALUE='%'> <BR>
<P>
<INPUT TYPE=Submit VALUE='Rechercher'>
</FORM>
```

- **Partie php myPHP.php**

```
<?php
Require(Connect.php);
$requete="SELECT * FROM personne".' WHERE nom LIKE $nom ";
$connexion=mysql_pconnect(serveur, user, passwd);
mysql_select_db(bdd, $connexion); $resultat=mysql_query($requete, $connexion);
While ($pers=mysql_fetch_object($resultat)){
Ech
o '$pers->ID, $pers->nom, $pers->pnom<BR>\n';} } else {
Echo "<B>erreur dans l'exécution de requête</B><BR>\n";
Echo "msg de sql:" . Mysql_error($connexion);}
?>
```



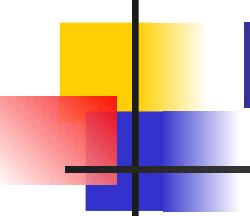
# UN PEU DE PHP5



# Introduction

## ■ Programmation objet

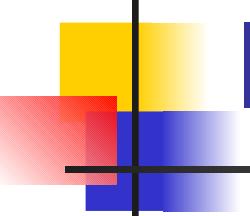
- Elle consiste à modéliser informatiquement un concept du monde réel en **entités informatiques**
- Ces entités informatiques sont appelées **objets**
- Exemple : compteBancaire
  - Propriétaire
  - Montant
  - Opérations possibles : débit, crédit...



# Introduction

## ■ Un objet

- est caractérisé par plusieurs notions :
- **L'identité** :
  - Nom qui **le distingue**
- **Les attributs** :
  - Données qui **le caractérisent**
  - Variables stockant des informations d'état de l'objet
- **Les méthodes** (appelées parfois *fonctions membres*) :
  - caractérisent son comportement
  - c'est-à-dire l'ensemble des actions (appelées *opérations*) que l'objet est à même de réaliser



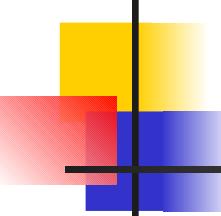
# Introduction

## ■ Notion de classe

- On appelle **classe** la **structure** d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet
- Un objet est donc "**issu**" d'une classe, c'est le produit qui sort d'un **moule**

## ■ Une classe est composée de deux parties :

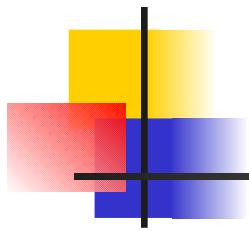
- **Les attributs** (parfois appelés *données membres*) :
  - il s'agit des données représentant l'état de l'objet
- **Les méthodes** (parfois appelées *fonctions membres*) :
  - il s'agit des opérations applicables aux objets



# Introduction

## ■ Exemple

- Si on définit la classe *voiture*
  - les objets *Peugeot 406*, *Renault 18* seront des instantiations de cette classe
  - Il pourra éventuellement exister plusieurs objets *Peugeot 406*, différenciés par leur numéro de série
  - Mieux :
    - ❖ Deux instantiations de classes pourront avoir tous leurs attributs égaux sans pour autant être un seul et même objet
    - ❖ C'est le cas dans le monde réel
      - deux T-shirts peuvent être strictement identiques et pourtant ils sont distincts



La suite...

Coming soon ...