

Université Claude Bernard Lyon 1



Licence Math-Informatique 1<sup>ère</sup> année

Partie 3

Olivier Glück

Université LYON 1 / Département Informatique

Olivier.Gluck@univ-lyon1.fr

<http://perso.univ-lyon1.fr/olivier.gluck>

# Copyright

- Copyright © 2017 Olivier Glück; all rights reserved
- Ce support de cours est soumis aux droits d'auteur et n'est donc pas dans le domaine public. Sa reproduction est cependant autorisée à condition de respecter les conditions suivantes :
  - Si ce document est reproduit pour les besoins personnels du reproducteur, toute forme de reproduction (totale ou partielle) est autorisée à la condition de citer l'auteur.
  - Si ce document est reproduit dans le but d'être distribué à des tierces personnes, il devra être reproduit dans son intégralité sans aucune modification. Cette notice de copyright devra donc être présente. De plus, il ne devra pas être vendu.
  - Cependant, dans le seul cas d'un enseignement gratuit, une participation aux frais de reproduction pourra être demandée, mais elle ne pourra être supérieure au prix du papier et de l'encre composant le document.
  - Toute reproduction sortant du cadre précisé ci-dessus est interdite sans accord préalable écrit de l'auteur.

# Plan du cours

- CM1 : Internet, les réseaux et le web
- CM2 : Pages HTML et feuilles de styles CSS
- CM3 : Web interactif, formulaires, pages dynamiques et PHP
- CM4 : Protocole HTTP, méthodes GET et POST
- CM5 : Les applications d'Internet
- CM6 : La couche transport : les protocoles TCP et UDP
- CM7 : Le protocole IP
- CM8 : Les protocoles Ethernet, ARP et ICMP. Synthèse des échanges entre un client et serveur Web



Université Claude Bernard  Lyon 1



# CM3 : Web interactif, formulaires, pages dynamiques et PHP

Le web interactif  
Formulaires HTML  
Programmation côté serveur en PHP  
Les bases du langage PHP

# Plan du CM3

- Le web interactif
  - Pages statiques et dynamiques, Programmation Web côté client, Programmation Web côté serveur
- Formulaires HTML
  - Principe, client passif/actif, La balise <FORM>, Les éléments d'un formulaire
- Programmation côté serveur en PHP
  - Qu'est-ce que PHP ? Interactions entre le client et le serveur, Fonctionnement de l'interpréteur, Un premier exemple
- Les bases du langage PHP
  - Types de données et variables, les opérateurs, les chaînes de caractères, les tableaux, instructions conditionnelles et boucles, fonctions, variables globales/locales, téléchargement de fichiers



Université Claude Bernard  Lyon 1



# Le Web interactif

Pages statiques et dynamiques  
Programmation Web côté client  
Programmation Web côté serveur

# Pages statiques et dynamiques

- Le HTML ne permet pas d'interactivité avec l'utilisateur
  - Les pages visualisées sont "statiques"
- Pages statiques
  - La page visualisée N fois sur le même navigateur donnera toujours le même résultat
- Pages dynamiques
  - La page visualisée dépend des manipulations de l'utilisateur
  - Elle est générée dynamiquement selon les actions de l'utilisateur dans la page
  - Nécessite de la programmation pour prendre en compte les actions
    - Programmation Web côté client : principalement Javascript
    - Programmation Web côté serveur : principalement PHP

# Programmation Web côté client

- Les exécutions associées ont lieu sur le poste client
- Le navigateur doit supporter ces exécutions...
- Des scripts embarqués dans la page HTML qui sont transférés depuis le serveur vers le client
  - "*HTML-embedded scripting*"
  - exemples : javascript, vbscript, ...
- Des applets
  - java, ActiveX, ...
  - le code de *l'applet* est envoyé au client
  - java tourne sur toutes les plate-formes
  - le navigateur doit néanmoins disposer d'une console java...
- Des *plugins* propriétaires



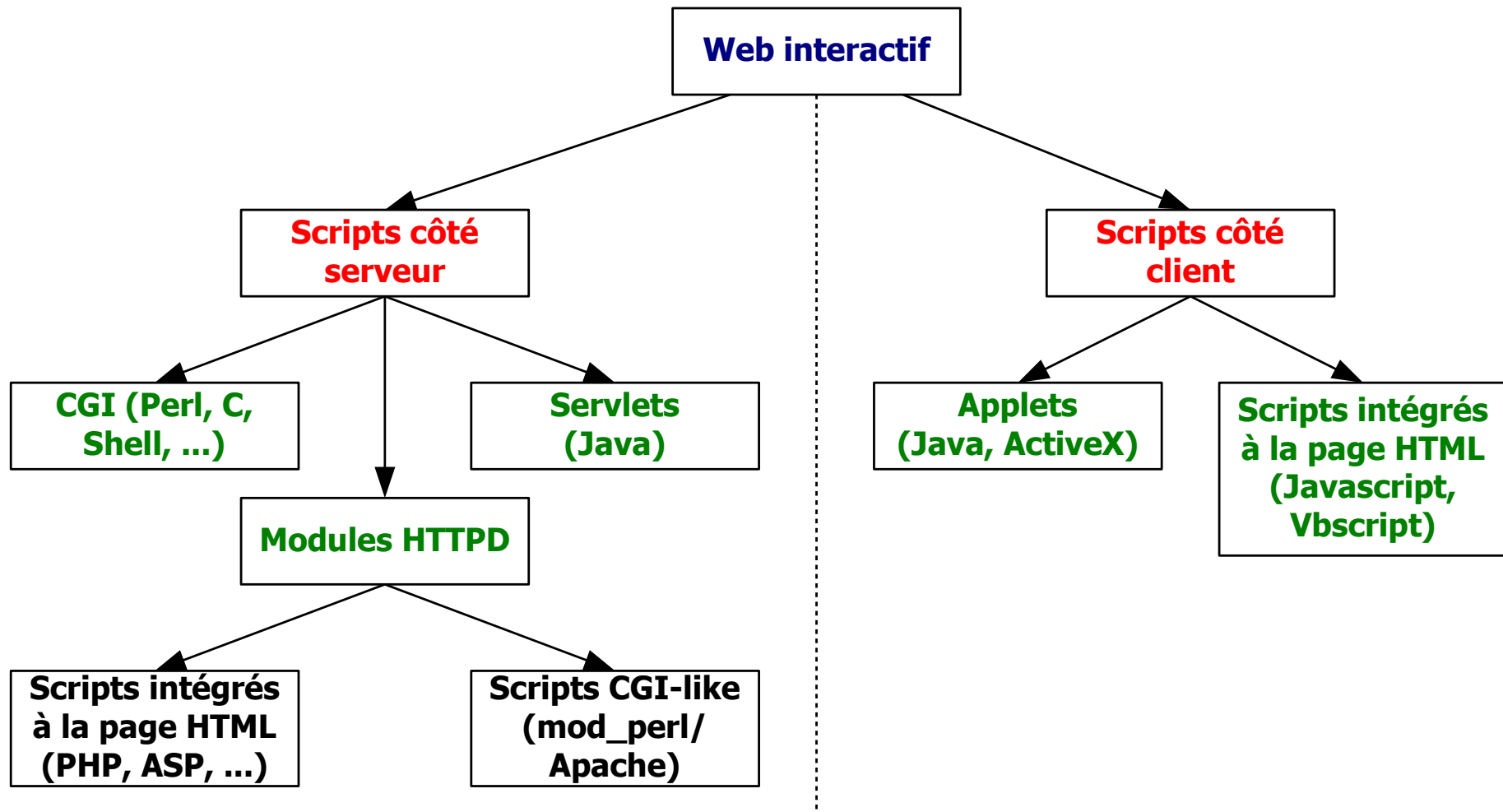
# Programmation Web côté serveur (1)

- Les exécutions associées ont lieu sur le serveur
- Le résultat de ces exécutions est une page HTML qui est renvoyée au navigateur client
- Basée sur l'interface CGI - *Common Gateway Interface*
  - Interface de communication qui définit le format d'échanges des données entre le client et le serveur HTTP
  - Les paramètres de l'utilisateur sont transmis par le serveur HTTP à un programme qui traite les requêtes et produit un résultat en HTML (page dynamique)
  - Les formulaires permettent de récolter les paramètres de l'utilisateur avant de les envoyer au script qui s'exécute sur le serveur
  - Scripts en PHP, Perl ou Shell, programmes C, Ada...

# Programmation Web côté serveur (2)

- Les interpréteurs intégrés au serveur web
  - Des scripts peuvent être insérés dans le code source de la page HTML et exécutés par le serveur web pour générer du code source HTML avant envoi au navigateur
    - Exemples : PHP, ASP, ...
  - Le serveur web peut exécuter des scripts non intégrés dans la page (CGI-like)
    - Exemples : *mod\_perl/Apache*
- Les servlets
  - Le pendant des Applets
  - En langage Java

# Programmation Web : récapitulatif



# Formulaires HTML

Principe, client passif/actif

La balise <FORM>

Les éléments d'un formulaire

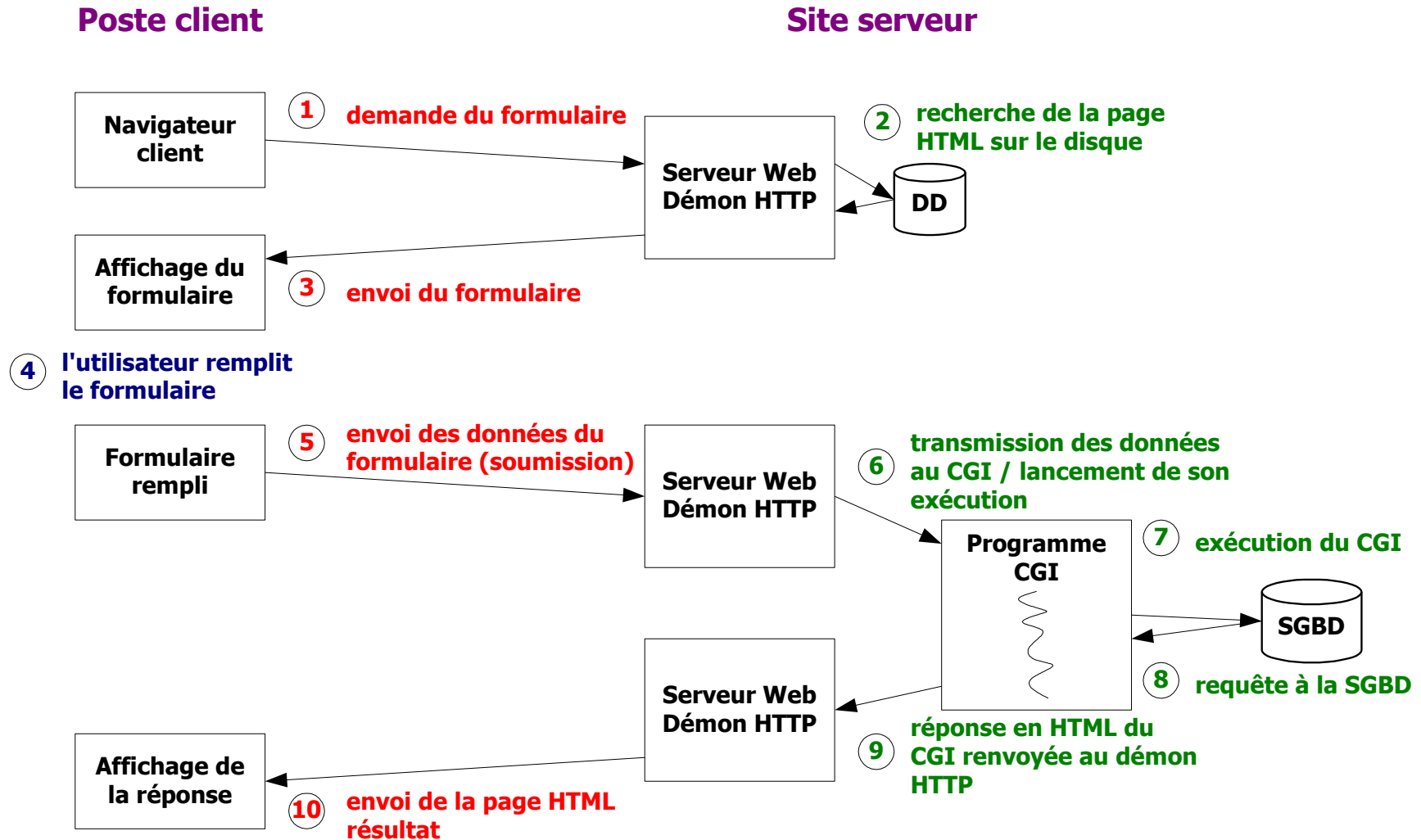
# Pourquoi des formulaires ?

- Apporte de l'inter-activité avec l'utilisateur en proposant des zones de dialogue : un formulaire n'est qu'une interface de saisie !
- Selon les choix de l'utilisateur, il faut y associer un traitement
  - sur le client avec JavaScript par exemple
  - sur le serveur par l'intermédiaire de CGI, PHP, ...
- Exemples typiques d'utilisation de formulaire
  - commandes, devis via Internet
  - moteurs de recherche
  - interactions avec une base de données

# Principe du formulaire

- On décrit à l'aide de balises HTML les différents champs de saisie
- Chaque zone est identifiée par un nom symbolique auquel sera associée une valeur par l'utilisateur
- Quand le formulaire est soumis, les couples (nom/valeur) de toutes les zones sont transmis dans la requête HTTP au serveur
- A chaque zone de saisie peut être associé un traitement sur le client par l'intermédiaire d'un événement JavaScript

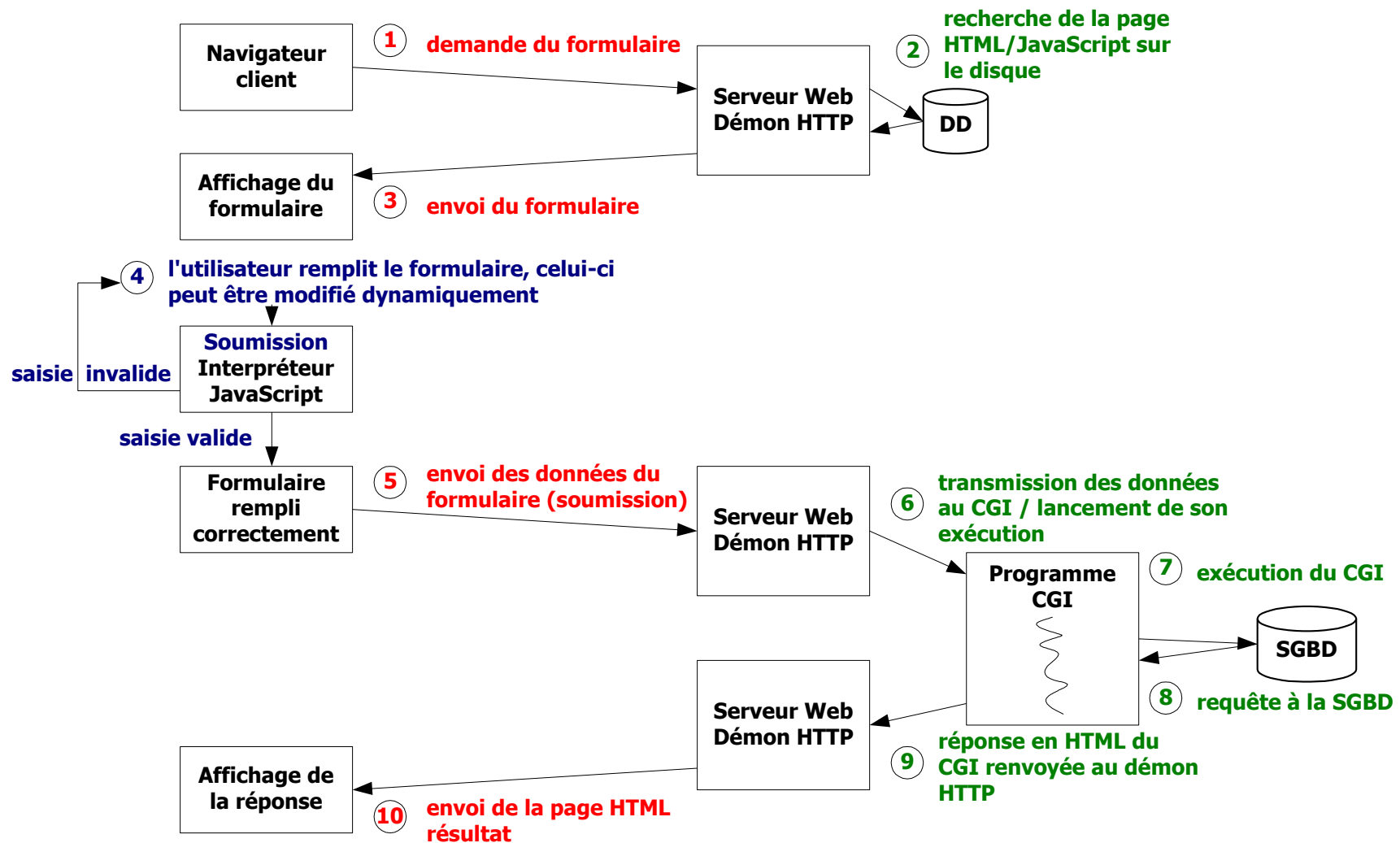
# Le client est passif



# Le client est actif

## Poste client

## Site serveur





# Les éléments d'un formulaire

- **Trois catégories**

- **input** : champs de saisie de texte et divers types de boutons
  - type="text" - zone de texte (type par défaut)
  - type="password" - zone de texte caché
  - type="checkbox" - cases à cocher
  - type="radio" - minimum 2, un seul sélectionnable
  - type="submit" - soumission du formulaire
  - type="reset" - bouton de remise à zéro des champs
  - type="button" - bouton associé à du code JavaScript
  - type="hidden" - bouton caché
  - type="file" - upload d'un fichier vers le serveur
- **select** : menus déroulants, listes à ascenseurs
  - size="1" - pop liste, 1 seul élément sélectionnable
  - size="n", n>1 - liste à choix multiples
- **textarea** : zone de saisie d'un texte "long"



# La balise <FORM> (1)

- <FORM>...</FORM> début et fin du formulaire
- Des champs de type input, select ou textarea ne seront visibles que s'ils sont à l'intérieur d'une balise <FORM>
- Attributs METHOD, ACTION, NAME, TARGET
  - **METHOD** : valeurs GET ou POST qui indiquent la façon dont les données sont transmises au script CGI
  - **ACTION** : URL du programme CGI qui sera exécuté quand l'utilisateur clique sur un bouton de soumission
  - **NAME** : distingue les différents formulaires
  - **TARGET** : cible dans laquelle la réponse du programme CGI sera affichée
- Les champs du formulaire sont transmis via l'URL (méthode GET) ou dans le contenu de la requête HTTP (méthode POST). Ils sont accessibles en PHP dans le tableau **\$\_GET** ou **\$\_POST**.

# La balise <FORM> (2)

## ■ Attribut ENCTYPE

- **ENCTYPE** : spécifie l'encodage utilisé pour l'envoi des données du formulaire dans le cas de la méthode POST (dans ce cas les données sont transmises dans le corps de la requête)
  - **ENCTYPE="application/x-www-form-urlencoded"** : valeur par défaut ; url-encode le contenu du formulaire de la même façon que par la méthode GET
  - **ENCTYPE="text/plain"** : le contenu du formulaire est envoyé en format texte lisible par le destinataire (action mailto: par exemple)
  - **ENCTYPE="multipart/form-data"** : permet d'expédier un fichier attaché dans le corps de la requête (<input type="file">)

# La balise <FORM> (3)

- Propriétés de l'objet FORM

- **action** : accès à l'attribut ACTION

```
<form name="f1" action="/cgi-bin/p1.cgi">...</form>
```

```
<script>document.f1.action="cgi-bin/p2.cgi"</script>
```

- **method** : accès à l'attribut METHOD

- **target** : accès à l'attribut TARGET

- **enctype** : type d'encodage des données transmises vers le serveur avec la méthode POST

- **elements** : accès aux objets du formulaires

- `elements.length` - nombre d'objets du formulaire

- `elements[n].name` - nom du énième+1 objet

- `elements[n].value` - valeur du énième+1 objet

# La balise <FORM> (4)

- Méthode de l'objet FORM : submit()
  - déclenche l'envoi du formulaire comme si l'utilisateur avait appuyé sur un bouton de soumission

```
<script>document.f1.submit()</script>
```


- Événement JS associé à l'objet FORM : onSubmit()
  - permet l'exécution de code JavaScript avant l'envoi du formulaire (vérification des saisies par exemple)

```
<form name="f1" method="post" action="/cgi-bin/p1.cgi" target="_blank" onSubmit="return verif_f1(this)">
```

# <INPUT type="TEXT">

- Attributs : NAME, VALUE, SIZE, MAXLENGTH
  - SIZE : taille d'affichage de la zone (en caractères)
  - MAXLENGTH : taille de remplissage de la zone (en carac.)

```
<INPUT TYPE="text" NAME="email" VALUE="entrez votre email ici" SIZE="30" MAXLENGTH="50" />
```
- Propriétés
  - name, value, defaultValue, type, form (le formulaire qui contient l'élément INPUT)
- Méthodes
  - focus(), blur(), select()
- Événements
  - onBlur, onChange, onFocus, onSelect



entrez votre email ici

# <INPUT type="PASSWORD">

- Attributs : NAME, VALUE, SIZE, MAXLENGTH

```
<INPUT type="PASSWORD" NAME="pass" VALUE="entrez votre  
passwd ici" SIZE="8" />
```

- Propriétés

- name, value, defaultValue, type, form

- Méthodes

- focus(), blur(), select()

- Pas d'événement associé



# <INPUT type="CHECKBOX">

- Cases à cocher permettant un choix multiple
- Attributs : NAME, VALUE, CHECKED



A screenshot of a web form with three checkboxes. The first checkbox is checked and labeled 'HTML'. The second checkbox is checked and labeled 'JS'. The third checkbox is unchecked and labeled 'CGI'.

```
<INPUT type="CHECKBOX" NAME="cours" VALUE="1" CHECKED />HTML<BR/>  
<INPUT type="CHECKBOX" NAME="cours" VALUE="2" CHECKED />JS<BR/>  
<INPUT type="CHECKBOX" NAME="cours" VALUE="3" />CGI<BR/>
```

- Propriétés
  - name, value, type, form, checked et defaultChecked (booléen)
- Méthode
  - document.f1.cours[1].click() - coche/décoche la case JS
- Événement
  - onClick - quand l'utilisateur coche la case



# <INPUT type="RADIO">

- Choix d'une et une seule option parmi n
- Attributs : NAME, VALUE, CHECKED

HTML  JS  CGI

```
<INPUT type="RADIO" NAME="cours" VALUE="1" />HTML
```

```
<INPUT type="RADIO" NAME="cours" VALUE="2" CHECKED />JS
```

```
<INPUT type="RADIO" NAME="cours" VALUE="3" />CGI
```

- Propriétés
  - name, value, type, form, checked et defaultChecked (booléen), index (donne le rang du bouton sélectionné), length
- Méthode
  - document.f1.cours[2].click() - sélectionne la case CGI
- Événement
  - onClick - quand l'utilisateur coche la case

# <INPUT type="SUBMIT">

- Envoi des données et exécution du programme CGI spécifié par l'attribut ACTION de <FORM>
- Attributs : NAME, VALUE
  - <INPUT type="SUBMIT" NAME="s" VALUE="login" />
  - <INPUT type="SUBMIT" NAME="s" VALUE="logout" />
  - VALUE permet de différencier le traitement à effectuer par le CGI s'il y a plusieurs boutons de soumission
- Propriétés
  - name, value, type, form
- Méthode
  - click() - soumet le formulaire
- Événement
  - onClick



# <INPUT> et <BUTTON>

- Il est possible de remplacer la balise <INPUT> par la balise <BUTTON>. Cela permet de distinguer la valeur transmise au serveur de la valeur qui s'affiche dans le bouton.

- Exemple d'utilisation de la balise <BUTTON>

```
<BUTTON type="SUBMIT" NAME="action" VALUE="com1">
```

Commenter !

```
</BUTTON>
```

```
<BUTTON type="SUBMIT" NAME="action" VALUE="com2">
```

Commenter !

```
</BUTTON>
```

Les deux boutons ont une valeur différente mais s'affichent de manière identique dans le navigateur.

# <INPUT type="RESET">

- Recharge tous les champs du formulaire à leur valeur par défaut

A rectangular button with a light gray background and a thin black border. The word "Effacer" is written in a dark gray, sans-serif font in the center of the button.

- Ne provoque pas l'exécution du CGI

- Attributs : NAME, VALUE

```
<INPUT type="RESET" NAME="raz" VALUE="Effacer" />
```

- Propriétés

- name, value, type, form

- Méthode

- click() - réinitialise le formulaire

- Événement

- onClick

# <INPUT type="BUTTON">

- N'a de sens que dans un contexte JavaScript
  - pas de comportement préprogrammé
  - ne permet pas de collecter une valeur dans le CGI
- Attributs : NAME, VALUE

```
<INPUT type="BUTTON" NAME="b1" VALUE="Aide"
onClick="AideEnLigne()" />
```
- Propriétés
  - name, value, type, form
- Méthode
  - click() - simule un click de l'utilisateur
- Événement
  - onClick



# <INPUT type="HIDDEN">

- Permet de transmettre des "variables" cachées au programme CGI
  - très utile par exemple pour transmettre des variables de formulaires en formulaires
  - invisible pour l'utilisateur
- Attributs : NAME, VALUE
  - `<INPUT type="HIDDEN" NAME="h1" VALUE="0" />`
- Propriétés
  - name, value, type, form
- Pas de méthode et pas d'événement puisque l'objet n'est pas visible

# <INPUT type="FILE">

- Permet de transmettre un fichier de la machine locale vers le serveur web (**upload**)
- Attributs : NAME, VALUE  
`<INPUT type="FILE" NAME="fichier" VALUE="Parcourir" />`
- Pour transmettre un fichier, dans <FORM>, il faut :  
`<FORM method="POST" action="page.php" enctype="multipart/form-data">`
- Pour limiter la taille du fichier transmis :  
`<INPUT type="hidden" name="MAX_FILE_SIZE" value="1048576" />`
  - A mettre avant `<INPUT type="FILE">`
  - La taille est en octets :  $1048576 = 1 \text{ Mo}$
- Les fichiers sont transmis au serveur dans le contenu de la requête HTTP (méthode POST). Ils sont accessibles en PHP dans le tableau `$_FILES`. Ils sont stockés dans un dossier temporaire sur le serveur. PHP fournit des fonctions pour manipuler les fichiers.

# <SELECT> et <OPTION> (1)

- Attributs de <SELECT> : NAME, SIZE, MULTIPLE
  - SIZE : taille de la *scrolled-list* (*pop-list* si 1)
  - MULTIPLE : autorise la sélection multiple si SIZE>1
- Attributs de <OPTION> : VALUE, SELECTED

```
<SELECT NAME="pop">
```

```
<OPTION VALUE="v1">1
```

```
<OPTION VALUE="v2" SELECTED>2
```

```
<OPTION VALUE="v3">3
```

```
</SELECT>
```

```
<SELECT NAME="mul" SIZE="3" MULTIPLE>
```

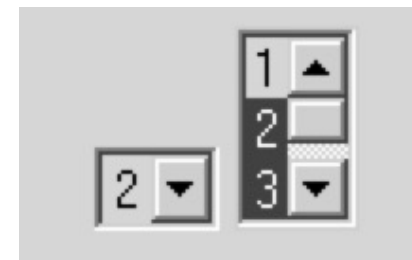
```
<OPTION VALUE="v1">1
```

```
<OPTION VALUE="v2" SELECTED>2
```

```
<OPTION VALUE="v3" SELECTED>3
```

```
<OPTION VALUE="v4">4
```

```
</SELECT>
```





## <SELECT> et <OPTION> (2)

- Propriétés d'un objet <SELECT>
  - `name`, `type` (*select/select-one/select-multiple*), `form`, `length`,
  - `selectedIndex` : rang de l'option sélectionnée ; dans le cas d'une liste multiple, rang de la première option sélectionnée
- Méthodes d'un objet <SELECT> : `focus()`, `blur()`
- Événements liés à un objet <SELECT> : `onFocus`, `onBlur`, `onChange`
- Propriétés relatives aux options
  - `defaultSelected`, `selected`, `text`, `value`  
`document.f1.mul.options[2].text` contient 3  
`document.f1.mul[2].text = 4;`
  - on peut dynamiquement modifier, ajouter, supprimer des items de la liste

# <SELECT> et <OPTION> (3)

```
<!-- ex_popup.html -->
<HTML><HEAD>
<SCRIPT>
function ChLang(f) {
  if (f.lingue.selectedIndex == 0) {
    f.choix.options[0].text="Rouge";
    f.choix.options[1].text="Vert";
    f.choix.options[2].text="Bleu";
  }
  else {
    f.choix.options[0].text="Red";
    f.choix.options[1].text="Green";
    f.choix.options[2].text="Blue";
  }
}
</SCRIPT>
</HEAD><BODY>
```

```
<FORM name="f1" METHOD="POST"
ACTION="/cgi-bin/p1.cgi">
<SELECT NAME="langue"
onChange="ChLang(this.form)">
  <OPTION VALUE="F">Français
  <OPTION VALUE="E">English
</SELECT>
<SELECT NAME="choix">
  <OPTION VALUE="R">Rouge
  <OPTION VALUE="V">Vert
  <OPTION VALUE="B">Bleu
</SELECT>
</FORM>
</BODY></HTML>
```



# <SELECT> et <OPTION> (4)

```
<!-- ex_scrollist.html -->
<HTML><HEAD>
<SCRIPT>
function to2(f) {
  r1=f.l1.selectedIndex;
  if (r1<0) return;
  opt=new Option();
  opt.text=f.l1.options[r1].text;
  opt.value=f.l1.options[r1].value;
  f.l1.options[r1]=null;
  r2=f.l2.length;
  f.l2.options[r2]=opt;
}
function to1(f) {...}
</SCRIPT>
</HEAD><BODY>
```

```
<FORM name="f1" METHOD="POST" ACTION="/
cgi-bin/p1.cgi">
<SELECT NAME="l1" SIZE="5">
  <OPTION VALUE="1">Français
  <OPTION VALUE="2">Anglais
  <OPTION VALUE="3">Italien
  <OPTION VALUE="4">Espagnol
  <OPTION VALUE="5">Allemand
</SELECT>
<INPUT type="BUTTON" VALUE=">>"
      onClick="to2(this.form)">
<INPUT type="BUTTON" VALUE="<<"
      onClick="to1(this.form)">
<SELECT NAME="l2" SIZE="5">
</SELECT></FORM>
</BODY></HTML>
```



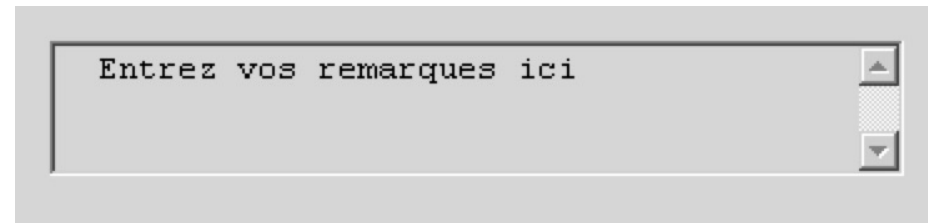
# <TEXTAREA>

- Zone de saisie de texte libre
- Attributs : NAME, ROWS, COLS

```
<TEXTAREA NAME="t1" ROWS="3" COLS="40">
```

Entrez vos remarques ici

```
</TEXTAREA>
```



- Propriétés
  - name, value, defaultValue, type, form, cols, rows
- Méthodes
  - focus(), blur(), select()
- Événements
  - onBlur, onChange, onFocus, onSelect

# Programmation côté serveur en PHP

Qu'est-ce que PHP ?  
Interactions entre le client et le serveur  
Fonctionnement de l'interpréteur  
Un premier exemple

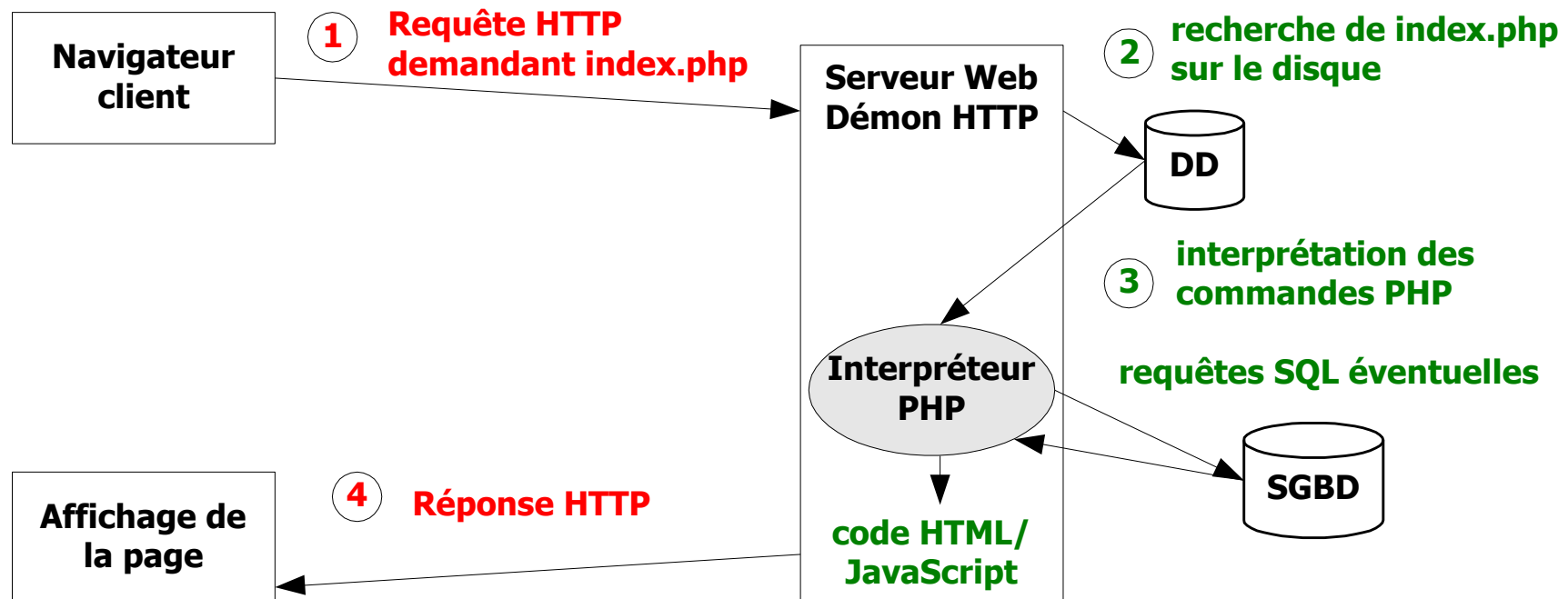
# Qu'est-ce que PHP ?

- "PHP is a **server-side, HTML-embedded, cross-platform** scripting language"
  - Langage interprété et indépendant de la plate-forme d'exécution,
  - qui s'exécute sur le serveur,
  - dont les instructions sont intégrées au code source d'une page HTML.
- Un fichier **.php** peut contenir du code HTML et du code PHP
- Le serveur lit les instructions PHP intégrées à la page HTML, interprète ces instructions et les remplace par le résultat de leur exécution (pages dynamiques)
  - Le client n'a pas accès au code source car il est interprété avant envoi
  - Le client ne reçoit que le résultat de l'exécution

# Interactions entre le client et le serveur

## Poste client

## Site serveur



# Fonctionnement de l'interpréteur PHP

- Un bloc PHP est un groupe de lignes encadré par deux balises `<?php ?>`
  - Toute ligne située à l'extérieur de ces balises n'est pas interprétée : elle est recopiée à l'identique pour être renvoyée au client
  - Toute ligne située à l'intérieur de ces balises est interprétée par le serveur comme une instruction PHP
- On peut mélanger les commentaires du C, C++ et shell !
  - `<?php /* commentaire type C */`
  - `// commentaire type C++`
  - `# commentaire type shell`
  - `?>`



# Un premier exemple

```
<!-- 1.php -->
<HTML><HEAD>
    <TITLE>Premier exemple PHP</TITLE>
</HEAD><BODY>
<H2>Voici les informations que je connais à votre sujet</H2>
<?php
/* $HTTP_USER_AGENT sera remplacée par sa valeur
avant l'envoi de la réponse au client */
echo $_SERVER['HTTP_USER_AGENT'];
?>
</BODY></HTML>
```

# Les bases du langage PHP

Les Types de données et variables, les opérateurs

Les chaînes de caractères

Les tableaux

Instructions conditionnelles et boucles

Fonctions, variables globales/locales

Téléchargement de fichiers

# Types de données et variables

- Les noms de variables sont précédés d'un **\$**

```
$i = 5;  
echo "$i";
```

- **Pas de déclaration, l'affectation détermine le type de la variable**
- PHP supporte les types de données suivants
  - Booléens : `true` ou `false`
  - Nombres entiers : `$i = (int) (7/3) # $i vaut 2`
  - Nombres à virgule flottante : `$i = 7/3 # $i vaut 2.33333333...`
  - Chaînes de caractères
  - Tableaux
  - Objets (programmation orientée objet)

# Les opérateurs (1)

- Opérateurs arithmétiques

- addition :  $\$a + \$b$

- soustraction :  $\$a - \$b$

- multiplication :  $\$a * \$b$

- division :  $\$a / \$b$

- modulo (reste de la division entière) :  $\$a \% \$b$

- ```
 $\$i = 7/3; echo "<br />\$i"; # affiche 2.333333333333333$ 
```

- ```
 $\$j = 7\%3; echo "<br />\$j"; # affiche 1$ 
```

- Concaténation de chaîne de caractère (.)

- ```
 $\$ch1 = "Bonjour "; \$ch2 = "Monsieur";$ 
```

- ```
 $\$ch = \$ch1.\$ch2$ 
```

- ```
 $echo \$ch . " Durand" . ' !';$ 
```

- ```
 $# affiche "Bonjour Monsieur Durand !"$ 
```

# Les opérateurs (2)

- Opérateurs logiques

- ET logique : `and`, `&&` (les deux sont possibles)
- OU logique : `or`, `||`
- XOR logique : `xor`
- NON logique : `!`

- Opérateurs d'affectation

- affectation avec le signe `=`

`$a = ($b = 4) + 1; # $b vaut 4 et $a vaut 5`

- les opérateurs combinés : `+=`, `-=`, `*=`, `/=`, `.=`, `&=`, `|=`

`$a += 1; # équivalent à $a = $a + 1;`

`$ch .= "!"; # équivalent à $ch = $ch . "! »;`

# Les opérateurs (3)

- Opérateurs de comparaison
  - égal à :  $\$a == \$b$
  - différent de :  $\$a != \$b$
  - inférieur à :  $\$a < \$b$
  - supérieur à :  $\$a > \$b$
  - inférieur ou égal à :  $\$a <= \$b$
  - supérieur ou égal à :  $\$a >= \$b$
- Opérateur ternaire
  - $(condition) ? (expr1) : (expr2);$
  - renvoie  $expr1$  si condition est vraie sinon renvoie  $expr2$
  - $\$max = (\$a >= \$b) ? \$a : \$b; \# \$max <- \max(\$a, \$b)$

# Chaînes de caractères (1)

- Chaînes de caractères

```
$ch1 = 'Bonjour'; $ch2 = "$ch1 Monsieur !\n";
```

- entre guillemets simples, rien n'est interprété sauf \ ' et \ \ qui échappent le caractère ' et le caractère \
- entre guillemets doubles, sont interprétés les variables, les caractères spéciaux \n \r \t \\$ \" \\
- **echo "\n";** provoque un saut de ligne dans le code HTML généré mais ne provoque de saut de ligne HTML (il faut utiliser <br />) !

- Accès aux caractères d'une chaîne et concaténation

```
$ch = 'Ce' . 'ci est une chaîne.'; echo $ch{2}; # affiche "c"
```

# Chaînes de caractères (2)

- Substitutions de chaînes

- **str\_replace(search, replace, str)** retourne une chaîne qui contient la chaîne str dans laquelle toutes les occurrences de search sont remplacées par replace

# la commande suivante affiche "Bonjour titi"

```
echo str_replace("toto", "titi", "Bonjour toto");
```

- Découpage de chaînes

- **explode(sep, str)** retourne un tableau de chaînes en scindant str à l'aide du séparateur sep
- **implode(sep, tab)** retourne une chaîne fabriquée par la concaténation des éléments du tableau et du séparateur sep entre chaque élément
- **parse\_str(str)** analyse une chaîne type QUERY\_STRING et fabrique les couples \$nom=valeur associés



# Chaînes de caractères (3)

- Comparaison et longueur
  - **strcmp(str1, str2)** comparaison binaire (comme en C)
  - **strlen(str)** retourne la longueur de str
- Web
  - **rawurlencode(str)/rawurldecode(str)** encode/décode la chaîne str selon la RFC1738 (%xy <-> caractère)
  - **urlencode(str)/urldecode(str)** idem mais adapté au formulaire (remplace également le + en espace...)
  - **parse\_url(str)** retourne dans un tableau les composants d'une URL (scheme, host, port, user, pass, path, query, fragment)  
<http://username:password@hostname/path?arg=value#anchor>
- Afficher la date et l'heure actuelle  

```
echo "Nous sommes le " . date("j m Y") . " et il est " . date("H \h i") . " mn";
```

# Tableaux (1)

- Les tableaux de PHP sont associatifs
  - l'index dans le tableau est appelé **clé**
  - la valeur associée à une clé est appelée **valeur**
  - un tableau est un ensemble d'associations clé/valeur
  - la clé peut être un entier ou une chaîne de caractères
- Création d'un tableau
  - soit directement en affectant des valeurs au tableau
    - `$tab[0] = 1;` # clé entière, valeur entière
    - `$tab[1] = "toto";` # clé entière, valeur de type chaîne
    - `$tab["toto"] = "titi";` # clé et valeur de type chaîne
    - `$tab["toto"][0] = 1;` # tableau à 2 dimensions
  - soit en utilisant la fonction **array()**
    - `$tab1 = array(1, "toto");` # 0=>1 et 1=>"toto"
    - `$tab2 = array("toto"=>1, "titi"=>"toto");` # la clé est donnée

# Tableaux (2)

- Nombre d'éléments d'un tableau
  - `sizeof($tab); count($tab);` # retourne le nombre d'éléments du tableau \$tab s'il existe, 1 si \$tab n'est pas un tableau, 0 si \$tab n'existe pas
- Suppression d'un élément
  - `unset($tab["toto"]);` # fonctionne aussi pour une variable
- Tris de tableaux
  - le tri peut se faire sur les clés et/ou les valeurs ; l'association clé/valeur peut être cassée
  - `asort()/arsort()` : trie le tableau par ordre croissant/décroissant de valeurs
  - `ksort()/krsort()` : trie le tableau par ordre croissant/décroissant de clés
  - `sort()` : trie le tableau par ordre croissant de valeurs et réassigne des clés (0,1,...) ; on perd l'association clé/valeur

# Tableaux (3)

- Le pointeur de tableau

- à chaque tableau correspond un pointeur interne qui est une référence sur l'élément courant
- `current($tab)` désigne l'élément courant
- `next($tab)` déplace le pointeur vers l'élément suivant
- `prev($tab)` déplace le pointeur vers l'élément précédent
- `end($tab)` déplace le pointeur sur le dernier élément
- `reset($tab)` déplace le pointeur sur le premier élément

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);  
$val = current($tab); echo "$val<br />"; # affiche "1<br />"  
$val = next($tab); echo "$val<br />"; # affiche "5<br />"  
asort($tab); end($tab); prev($tab); $val = prev($tab);  
echo "$val<br />"; # affiche "4<br />"
```

# Tableaux (4)

- Extraction d'éléments d'un tableau
  - **list()** permet d'extraire des valeurs d'un tableau  

```
$tab = array(1, 8, 5); sort($tab);  
list($v1, $v2) = $tab;  
echo "$v1 $v2"; # affiche "1 5"
```
  - **key(\$tab)** permet d'extraire la clé de l'élément pointé par le pointeur interne du tableau  

```
$tab = array("a"=>1,"b"=>8); next($tab);  
$clé = key($tab); $val = $tab[$clé];  
echo "$clé: $val"; # affiche "b: 8"
```
  - **extract(\$tab)** permet d'extraire d'un tableau toutes les valeurs, chaque valeur est copiée dans une variable ayant pour nom la valeur de la clé  

```
$tab = array("a"=>1,"b"=>8); extract($tab);  
echo "$b $a"; # affiche "8 1"
```

# Tableaux (5)

- Extraction d'éléments d'un tableau
  - **each(\$tab)** retourne la paire clé/valeur courante du tableau et avance le pointeur de tableau ; cette paire est retournée dans un tableau de 4 éléments : 0=>clé, 1=>valeur, key=>clé et value=>valeur
- Parcours de l'ensemble du tableau
  - avec **list()** et **each()**

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);  
reset ($tab);  
while (list ($k, $v) = each ($tab)) {  
echo "$k => $v<br />\n"; }  
#quand le pointeur dépasse la fin de $tab, each retourne false
```

# Tableaux (6)

- Parcours de l'ensemble du tableau
  - **foreach()** place le pointeur en tête du tableau et parcourt l'ensemble des éléments ; foreach() travaille sur une **copie** du tableau original

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);  
# parcours des valeurs uniquement : $v = 1, 5, 8, 4  
foreach ($tab as $v) { echo "valeur : $v<br />\n"; }  
# parcours des couples clé/valeur  
foreach ($tab as $k => $v) { echo "$k : $v<br />\n"; }  
# tableaux multidimensionnels  
$tab2[0][0] = "a"; $tab2[0][1] = "b";  
$tab2[1][0] = "y"; $tab2[1][1] = "z";  
foreach($tab2 as $t) { foreach ($t as $v) { echo "$v\n"; } }
```

# Instructions conditionnelles (1)

- if/then/else comme en C

```
if (condition1) {  
    # si condition1 est vraie  
} elseif (condition2) {  
    # si condition2 est vraie (et pas condition1)  
} elseif (condition3) {  
    # si condition3 est vraie (et ni condition1, ni condition2)  
} else {  
    # si ni condition1, ni condition2, ni condition3 ne sont vraies  
}
```



# Instructions conditionnelles (2)

- Le switch comme en C

```
switch(expr) {  
  case (val1) :  
    # à exécuter si expr vaut val1  
  break;  
  case (val2) :  
    # à exécuter si expr vaut val2  
  break;  
  default :  
    # à exécuter dans tous les autres cas  
}
```

# Les boucles

- Boucle "tant que" comme en C  
`while (condition) { # exécuté tant que la condition est vraie }`
- Boucle "Faire...tantque" comme en C  
`do { /* ... */ } while (condition);`
- Boucle "for" comme en C  
`for ($i = 1; $i <= 10; $i++) { # exécuté tant que ($i <= 10) est vraie }`
- L'instruction `break` permet de sortir d'une boucle
- L'instruction `continue` permet de passer directement à l'itération suivante de la boucle

# Les fonctions (1)

- Premier exemple

```
function max($a, $b) {  
    # corps de max()  
    if ($a>$b) { return $a; } else { return $b; }  
}  
# appel de la fonction max()  
$m = max(5, 8); # $m vaut 8
```

- Il est possible de retourner n'importe quel type de variable (tableau, objet, ...)
- Une fonction peut ne rien retourner
- PHP permet le passage d'arguments par valeur et par référence

# Les fonctions (2)

- Deux types de passage par référence
  - de façon permanente en ajoutant un **&** devant le nom de la variable dans la définition de la fonction
  - de façon ponctuelle en ajoutant un **&** devant le nom de la variable lors de l'appel de la fonction

```
function plus3($a) {           # passage par valeur
    $a += 3;
}                               # passage par référence ponctuel
function plus5(&$a) {         $y = 1; plus3(&$y); echo $y; # affiche "4"
    $a += 5;                   # passage par référence "forcé"
}                               $z = 1; plus5($z); echo $z; # affiche "6"
```

# Les variables globales/locales (1)

- Variables globales

```
<?php $a = 1; include "fonctions.php"; ?>
```

# \$a est globale ; elle est accessible dans fonctions.php

- Variables locales à une fonction

```
<?php
```

```
$a = 1; # portée globale
```

```
function affiche() {
```

```
    echo $a; # $a est locale à la fonction
```

```
}
```

```
affiche(); # n'affiche rien !
```

```
?>
```

# Les variables globales/locales (2)

- Accès à une variable globale dans un bloc
  - Déclarer la variable comme **global** dans le bloc (crée une référence locale sur la variable globale)
  - Ou utiliser le tableau associatif **\$GLOBALS**

```
<?php $a1 = 1; $a2 = 2; # portée globale
function affiche() {
    global $a1; # $a1 est la variable globale
    echo $a1." et ".$GLOBALS['a2'];
    $a1++;
}
affiche(); # affiche "1 et 2"
echo $a1; # affiche "2" ($a1 a été modifiée)
?>
```

# Les variables globales/locales (3)

- Variables statiques dans un bloc
  - Une variable statique a une portée locale à un bloc mais conserve sa valeur entre deux exécutions du bloc
  - Elles sont utiles pour les fonctions récursives

```
function up_to_10() {  
    static $cpt = 0;  
    $cpt++;  
    echo "$cpt ";  
    if ($cpt<10) up_to_10();  
}  
up_to_10(); # affiche 1 2 3 4 5 6 7 8 9 10
```

# Les variables prédéfinies

- Depuis PHP 4.1.0, des tableaux "**super-globaux**" sont prédéfinis
  - accessibles dans n'importe quel contexte d'exécution
  - infos sur les variables du serveur, les variables d'environnement, les variables de PHP, ...
- Le contenu de ces tableaux dépend de la version et de la configuration du serveur et de PHP
  - \$GLOBALS** : variables globales de l'exécution en cours
  - \$\_SERVER** : variables fournies par le serveur WEB
    - 'PHP\_SELF' (chemin du script sur le serveur),
    - 'argc', 'argv' (exécution en ligne de commande ou GET),
    - les variables d'environnement CGI, par exemple **\$\_SERVER['QUERY\_STRING']**
  - \$\_GET** : variables fournies par HTTP/méthode GET
  - \$\_POST** : variables fournies par HTTP/méthode POST
  - \$\_FILES** : variables fournies par HTTP suite à un téléchargement de fichier(s) par la méthode POST



# Téléchargement de fichiers (1)

- PHP offre la possibilité de recevoir des fichiers texte ou binaire en provenance du client et d'y associer un traitement
  - Réception par la méthode POST : une boîte de dialogue permet à l'utilisateur de sélectionner un fichier local

```
<FORM ENCTYPE="multipart/form-data" ACTION="..." METHOD="POST">  
  <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="1000" />  
  Envoyez ce fichier : <INPUT NAME="userfile" TYPE="file">  
  <INPUT TYPE="submit" VALUE="Envoyer le fichier">
```

```
</FORM>
```

- Le fichier téléchargé sera stocké temporairement dans un répertoire `$TMPDIR` sur le serveur
- `move_uploaded_file()` permet de déplacer un fichier téléchargé par PHP
- `is_uploaded_file()` permet de vérifier qu'un fichier a bien été téléchargé par la méthode POST

# Téléchargement de fichiers (2)

- Informations reçues par PHP dans **`$_FILES`**
  - le champ caché **`MAX_FILE_SIZE`** permet à PHP de faire des vérifications sur la taille du fichier téléchargé
  - **`$_FILES['userfile']['name']`** : nom original du fichier sur la machine du client web
  - **`$_FILES['userfile']['type']`** : type MIME du fichier, si le navigateur a fourni cette information
  - **`$_FILES['userfile']['size']`** : taille, en octets, du fichier
  - **`$_FILES['userfile']['tmp_name']`** : nom temporaire du fichier qui sera chargé sur la machine serveur
  - **`$_FILES['userfile']['error']`** : 0 en cas de réussite du téléchargement sinon code d'erreur (taille trop grande...)