Formulaires HTML

1.Introduction

- La présence d’éléments de formulaires HTML fait distinguer une application web d’un site web.

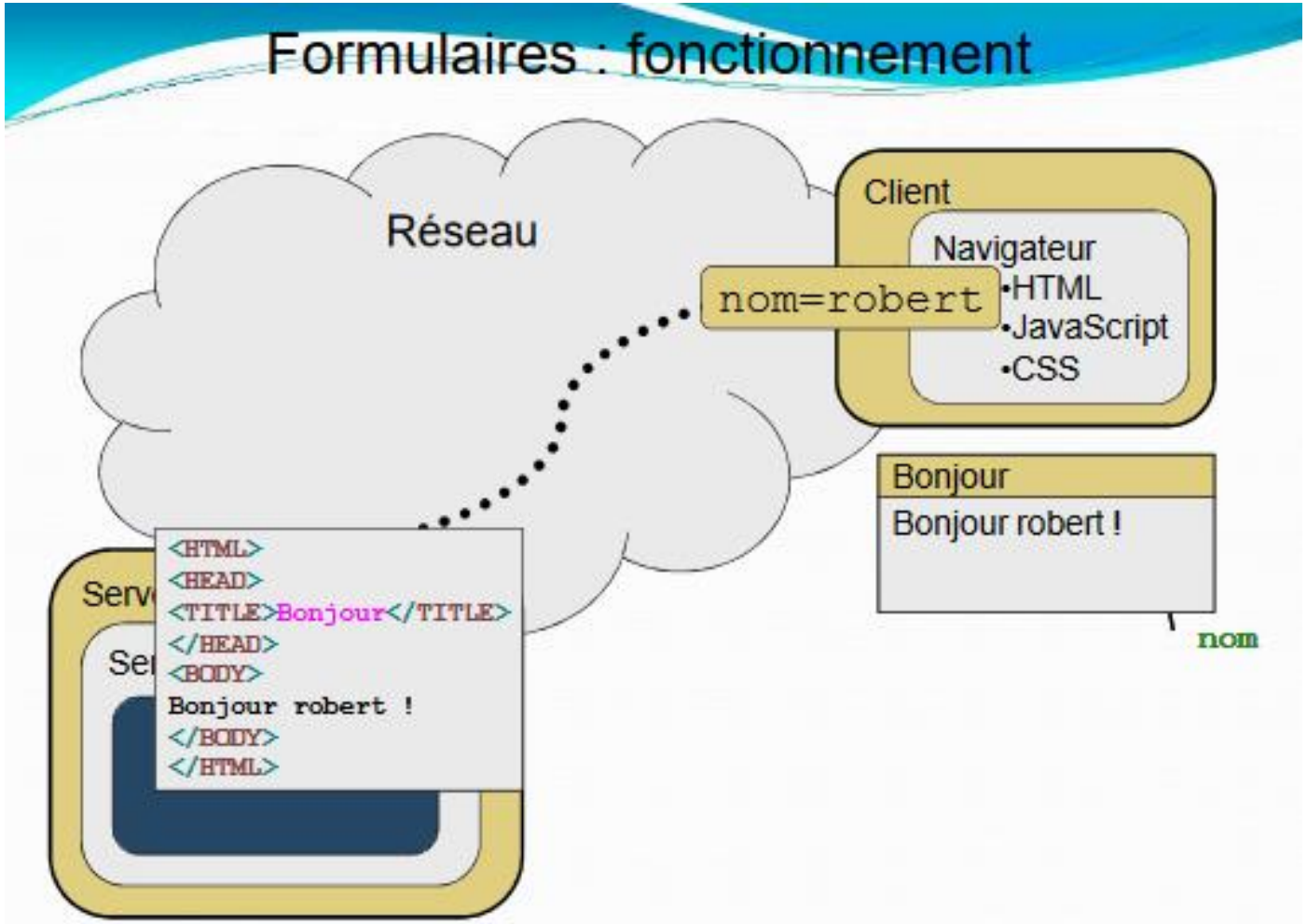
- Un formulaire HTML est une partie de page web qui accepte des entrées de l’utilisateur.

Un formulaire peut contenir :

- Des champs de saisie de texte;
- Des listes de sélections;
- Des boutons;
- Des cases à cocher;
- Des boutons d’options.
- Un bouton de soumission du formulaire et son contenu au serveur web.

Traitement du formulaire par le serveur web

- Récupération des valeurs des champs de formulaire;
- Traitements des données soumises via un langage;
- Génération d'une nouvelle page web;
- Renvoie de la page générée au client soumissionnaire du formulaire.



2.Définition de formulaire

BALISES	ATTRIBUTS	FONCTIONS
<FORM>… </FORM>		Balise qui permet de regrouper plusieurs éléments d'un formulaire (boutons, champs de saisie,...) et qui possède certains attributs obligatoires.
<FORM ATTRIBUTS>… </FORM>	NAME = "nom"	Un nom qui permet de différencier plusieurs formulaires sur une même page.
	ACTION = "http://url" ACTION = "mailto:url"	Envoyer les données du formulaire vers un adresse de site Web Envoyer les données du formulaire vers une adresse E-Mail.
	METHOD = "POST" METHOD = "GET"	Pour un envoi caché des informations vers une destination (CGI ou Email). La méthode "POST" envoie le contenu du formulaire séparément de l'URL. La méthode ‘’GET’’ envoi le contenu du formulaire et l’enregistre dans la variable d’environnement standard QUERY_STRING sur le serveur doté d’un protocole HTTP.
	ENCTYPE = "?"	"**text/plain**": Formulaire classique. Pour envoyer les informations par E-Mail. "**multipart/form-data**": Si le formulaire doit contenir un fichier attaché. "**application/x-www-form-urlencoded**": Si on a recours à un programme CGI.
	TARGET = "?"	Définit dans quelle fenêtre ou frame le résultat du formulaire sera affiché.
		Valeur **Action** nom_cadre Affiche la cible dans le cadre indiqué par le nom._ Self Affiche la cible dans le même cadre que le lien_ Parent Affiche la cible dans le cadre lié de niveau supérieur _blank Affiche la cible dans une nouvelle fenêtre sans cadre _top Affiche la cible dans la fenêtre entière du navigateur sans cadre

Exemple de page avec formulaire

```
<HTML>
<HEAD>
<TITLE>SELECT -4</TITLE>
</HEAD>
<BODY BGCOLOR="white" TEXT="black">

<H3>Formulaire </H3>
<FORM ACTION="select_4.htm“ method=“post”>

 </FORM>
</BODY>
</HTML>
```

Résultat

La balise form

- Les balises <FORM> et </FORM> définissent l'espace du formulaire.

- Elles possèdent trois attributs:
- - **ACTION** qui définit vers quelle URL envoyer le contenu du formulaire
- - **METHOD** qui définit le mode de transmission: GET ou POST
- - **ENCTYPE** qui définit le type de contenu: chaîne ou fichier

Résultat

L’attribut action

La propriété **ACTION** définit vers quelle **URL** (Universal Resource Location) envoyer le contenu du formulaire.

Ex :

<FORM ACTION="http://www.monsite.fr/traitement.php">

</FORM>

Pour renvoyer le formulaire à lui même, on écrira **action="#"** (voire on écrira rien du tout).

Résultat

L’attribut method

La propriété **METHOD** définit le mode de transmission, GET ou POST:

- **GET** : une chaîne est placée à la fin de l'URL après un **caractère " ? "** sous forme d'associations ***nomChamp=val*** où *nomChamp* est le nom du champ dans le formulaire et *val* la valeur saisie.

Nb: le caractère **" + "** remplace les espaces et le caractère **" & "** sépare les associations nomChamp=val.

- **POST** : la chaîne est transmise séparément de l'URL.

Ex :

<FORM ACTION="http://www.monsite.fr/traitement.php" METHOD="GET">

</FORM>

Résultat

L’attribut enctype

La propriété **ENCTYPE** définit le **type d'encodage** des données du formulaire qui doit être utilisé pour la transmission au serveur:

- **application/x-www-form-urlencoded**

C'est l'option par défaut qui prévoit que les champs du formulaire sont transmis sous la forme d'une liste de paires **nom=valeur.**

- **multipart/form-data**

Cette option doit être utilisée pour transmettre des fichiers.

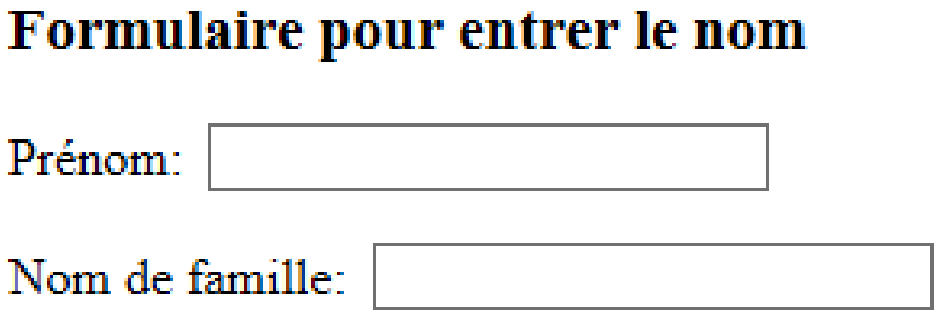
Résultat

3.Entrée d’une ligne de texte

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**text**"	Indique une boîte d'une seule ligne permettant l'entrée du texte
	NAME = "nom"	Identifier la boîte de texte pour pouvoir répérer/identifier le contenu
	SIZE = nombre	Spécifier la largeur de la boîte en nombre de caractères
	MAXLENGTH = n	Spécifier le nombre maximum de caractères permis dans la boîte Si ce nombre est plus grand que SIZE alors le texte se défilera à l'horizontale
	VALUE = "texte"	Indiquer quoi écrire ou spécifier le début du texte à entrer
	READONLY	Convertit le "champ de saisie" en "champ de sortie" non modifiable

Exemple 2 « text »

```
<HTML>
   <HEAD>
   <TITLE>INPUT -1</TITLE>
   </HEAD>
   <BODY BGCOLOR="white" TEXT="black">

   <H3>Formulaire pour entrer le nom</H3>
   <FORM ACTION="input_2.htm">
   <P>Prénom:
   <INPUT NAME="prénom" TYPE="text" SIZE="25" MAXLENGTH="40">
   <P>Nom de famille:
   <INPUT NAME="nom" TYPE="text" SIZE="25" MAXLENGTH="40">
   </FORM>
   </BODY>
</HTML>
```



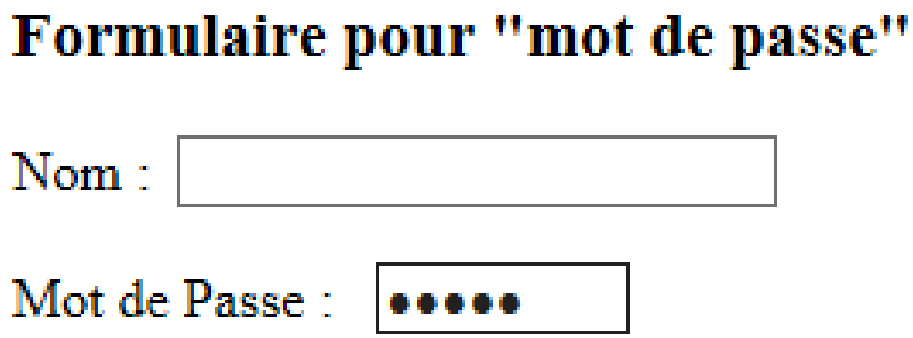
Résultat

4.Entrée d’un « Mot de Passe »

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**password**"	Indentique à une boîte de texte sauf que l'écriture est remplacée par des * * Les mots de passe sont, malgré la saisie cachée, transmis en clair sur Internet
	NAME = "nom"	Donner un nom pour identifier/répérer le contenu de la boîte de dialogue.
	SIZE = nombre	Spécifier la largeur de la boîte en nombre de caractères
	VALUE = "texte"	Indiquer le mot de passe qui apparaîtra sous forme ***** (Pas sécuritaire!!)

Exemple 3 « Password »

```
<HTML>
  <HEAD>
  <TITLE>PASSWORD -2</TITLE>
  </HEAD>
  <BODY BGCOLOR="white" TEXT="black">
    <H3>Formulaire pour "mot de passe"</H3>
    <FORM ACTION="password_2.htm">
    <P>Nom :
        <INPUT NAME="nom" TYPE="text" SIZE="25"    MAXLENGTH="20">
    <P>Mot de Passe :
        <INPUT NAME="nom" TYPE="password" SIZE="8" MAXLENGTH="5">
    </FORM>
  </BODY>
</HTML>
```



Résultat

5. Entrée d’un choix par « Bouton Radio »

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**radio**"	Disposera des boutons où un seul choix pourra être sélectionné. Tous les boutons dans un même groupe doivent avoir le même nom.
	NAME = "nom"	Identifie à quel groupe de boutons le 'bouton-radio' appartient.
	VALUE = "texte"	Indiquer le choix sélectionné. Peut être différent du choix affiché.
	CHECKED	Indique un choix par défaut parmi les options proposées.
	DISABLED	Montre les choix mais ne permet pas de faire, ou de modifier, la sélection.

Exemple 4 « Bouton Radio »

```
<HTML>
<HEAD>
<TITLE>RADIO-2</TITLE>
</HEAD>
<BODY BGCOLOR="white" TEXT="black">

<FORM ACTION="radio_2.htm">
<INPUT TYPE="radio" NAME="reponse" VALUE="O"> oui
<INPUT TYPE="radio" NAME="reponse" VALUE="N" CHECKED> non
</FORM>
</BODY>
</HTML>
```

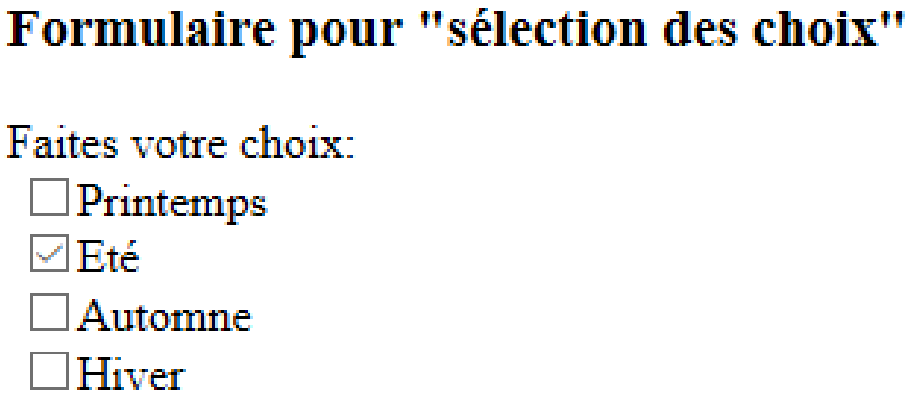
Résultat

6. Entrée de plusieurs choix par « Cases à Cocher »

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**checkbox**"	Disposera des cases où plusieurs choix pourront être sélectionnés. Toutes les cases dans un même groupe doivent avoir le même nom (NAME).
	NAME = "nom"	Identifie à quel groupe de cases la 'case à cocher' appartient.
	VALUE = "texte"	Indiquer le(s) choix sélectionné(s). Peut être différent du choix affiché.
	CHECKED	Indique un choix pré-définie parmi les options proposées.
	DISABLED	Montre les choix mais ne permet pas de faire, ou de modifier, la sélection.

Exemple 5 « Cases à Cocher »

```
<HTML>
   <HEAD>
   <TITLE>CHECKBOX-2</TITLE>
   </HEAD>
   <BODY BGCOLOR="white" TEXT="black">

   <H3>Formulaire pour "sélection des choix"</H3>
   <FORM ACTION="checkbox_2.htm">
   <P>Faites votre choix:<BR>
   <INPUT TYPE="checkbox" NAME="4S" VALUE="1">Printemps<BR>
   <INPUT TYPE="checkbox" NAME="4S" VALUE="2" CHECKED>Eté<BR>
   <INPUT TYPE="checkbox" NAME="4S" VALUE="3">Automne<BR>
   <INPUT TYPE="checkbox" NAME="4S" VALUE="4">Hiver<BR>
   </FORM>
   </BODY>

</HTML>
```



Résultat

Groupe lié

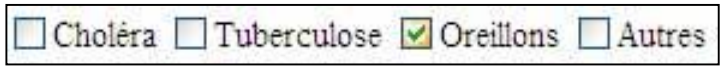
En donnant **le même attribut NAME** à plusieurs cases à cocher, on indique au navigateur que ces champs doivent être groupés dans la fenêtre d'affichage.

Choléra : <INPUT TYPE="checkbox" NAME="**maladie[]**" **VALUE**="C">

Tuberculose : <INPUT TYPE="checkbox" NAME="**maladie[]**" **VALUE**="T">

Oreillons : <INPUT TYPE="checkbox" NAME="**maladie[]**" **VALUE**="O" CHECKED>

Autres : <INPUT TYPE="checkbox" NAME="**maladie[]**" **VALUE**="A">



Résultat

7. Bouton « Envoyer »

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**submit**"	Envoyer le formulaire à l'adresse indiquée dans <FORM ACTION="...">
	VALUE = "texte"	Permet de modifier le texte affiché sur le bouton. (Par défaut = Submit)

8. Bouton « Reset »

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**reset**"	Remise à zéro de toutes les entrées du formulaire.
	VALUE = "texte"	Permet de modifier le texte affiché sur le bouton. (Par défaut = Reset)

<INPUT TYPE="submit" VALUE="Valider">

<INPUT TYPE="reset" VALUE="Tout effacer">

9.Bouton général

BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**button**"	Utilisé pour placer une bouton cliquable à l'écran. Sans l'utilisation de 'fonctions JavaScript', le bouton ne peut rien faire.
	NAME = "nom"	Donner un nom unique au bouton pour identifier le clic de la souris.
	VALUE = "texte"	Permet de placer un ou des mots descriptifs sur le bouton.

10. Bouton image

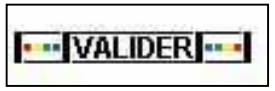
BALISES	ATTRIBUTS	FONCTIONS
<INPUT OPTIONS…>	TYPE = "**image**"	Identique au "Submit" mais le bouton est remplacé par une image. Les coordonnées x et y de l'emplacement où l'utilisateur a cliqué sur l'image seront également envoyées avec les autres informations du formulaire.

Exemple de type image

Pour remplacer un bouton de type SUBMIT par une image, vous disposez de la commande IMAGE :

Ex:

<INPUT TYPE="image" SRC="bouton.gif">

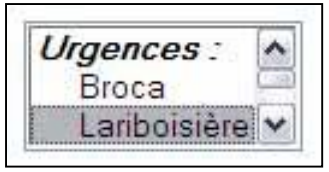


11. Liste déroulante

BALISES	ATTRIBUTS	FONCTIONS
<SELECT>… </SELECT>		Permet d'inclure une liste déroulante dans laquelle on pourra faire un choix. Les balises <OPTION> et </OPTION> permettent de définir les choix.
<SELECT OPTIONS>… </SELECT>	NAME = "nom"	Assignera un nom à l'item sélectionné par l'utilisateur.
	SIZE = "n"	Indique le nombre d'items à afficher dans la fenêtre en tout temps. Si l'attribut n'est pas spécifié, une liste déroulante est utilisé. (Size = 1)
	MULTIPLE	Permet de sélectionner plus qu'un item en enfonçant le bouton 'Ctrl' ou 'Shift' Si l'attribut n'est pas indiqué, un seul item de la liste pourra être sélectionné.
	DISABLED	Permet de créer une liste désactivée, c'est-à-dire affichée en grisée.

Exemple 6 « liste »

```
<SELECT>
  <OPTGROUP LABEL="Urgences>
   <OPTION VALUE="A">Broca
   <OPTION VALUE="B" SELECTED>Lariboisière
  </OPTGROUP>
</SELECT>
```



L'attribut **MULTIPLE** (sans valeur associée) autorise à sélectionner plusieurs valeurs dans la liste :

```
<SELECT NAME ="lieu" SIZE="3" MULTIPLE>
  <OPTGROUP LABEL="Urgences>
   <OPTION VALUE="A">Broca
   <OPTION VALUE="B" SELECTED>Lariboisière
  </OPTGROUP>
</SELECT>
```

12. Items d’une liste déroulantes

BALISES	ATTRIBUTS	FONCTIONS
<OPTION>… </OPTION>		Permet de détailler les items d'une liste déroulante.
<OPTION>… </OPTION>	VALUE = "texte"	Indiquera la phrase qui sera envoyé lors de la sélection de l'item.
	SELECTED	Permet de pré-identifier un item de la liste déroulante.

13. Entrée de textes dans un boîte

BALISES	ATTRIBUTS	FONCTIONS
<TEXTAREA>… </TEXTAREA>		Boite de texte éditable multi-lignes et multi-colonnes. Si le texte dépasse la limite verticale de la boîte, une barre de déroulent verticale apparaîtra.
	NAME = "nom"	Identifier la boîte de texte pour pouvoir repérer/identifier le contenu
	COLS = nombre	Spécifier la largeur de la boîte de texte en nombre de caractères
	ROWS = nombre	Spécifier le nombre de lignes pour la hauteur de la boîte de texte
	VALUE = "texte"	Permet de pré-définir du texte qui sera envoyé par défaut au script si le champ de saisie n'est pas modifié par une frappe de l'utilisateur.
	READONLY	Spécifie que la boîte est en mode lecture seulement.

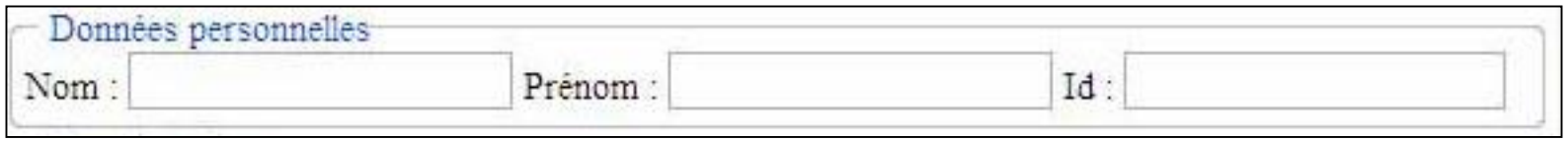
14. La balise Fieldset

La balise **<FIELDSET>** est uniquement décorative : elle permet d'entourer un groupe de commande par un léger filet gris.

On peut donner un titre à ce groupe à l'aide de la balise **<LEGEND>**.

Ex:

```
<FIELDSET>
<LEGEND>Données personnelles</LEGEND>
  Nom : <INPUT TYPE="text" NAME="nom" SIZE="20">
  Prénom : <INPUT TYPE="text" NAME="prenom" SIZE="20">
  Id : <INPUT TYPE="password" NAME="identifiant" SIZE="20">
</FIELDSET>
```



14. La balise Textarea

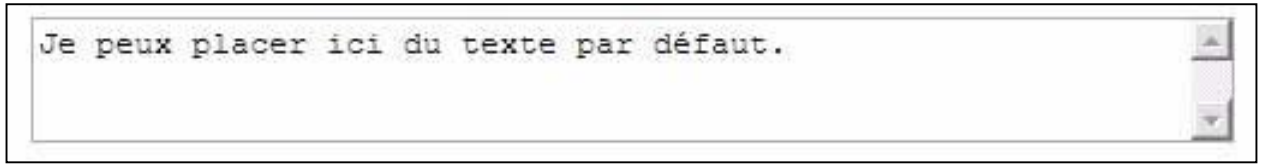
La balise **TEXTAREA** fournit à l'utilisateur une zone dans laquelle il peut rentrer du texte.

Cette commande est encadrée par une **balise ouvrante et fermante**.

Elle possède les attributs suivants:

- **NAME**
- **ROWS** : indique la **hauteur** de la zone de texte en nombre de ligne
- **COLS** indique la **largeur** de la zone de texte (en nombre de caractères).

```
<TEXTAREA NAME="commentaire" ROWS="3" COLS="60">
  Je peux placer ici du texte par défaut.
</TEXTAREA>
```

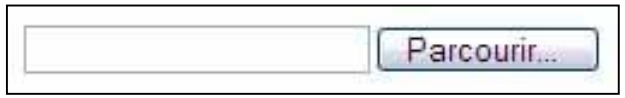


15. Le type file

La commande de type **FILE** permet de transmettre des fichiers par l'intermédiaire du formulaire.

Le champ doit alors contenir le chemin d'accès au fichier sur l'ordinateur du client.

Ex: <INPUT TYPE="file" SIZE="40" NAME="telFichier">



Une astuce permet de limiter la taille maximale du fichier à transmettre en utilisant une commande HIDDEN portant le nom MAX_FILE_SIZE et ayant pour valeur le nombre d'octets maximal.

Ex: <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="52000">

Comment traiter les données de formulaire ?

Un formulaire peut être traité :

- Soit au niveau du client – c'est le navigateur qui va travailler sur le formulaire

Exemple : Vérification de la validité des champs, apparition d'infos bulles etc…

Langage : JAVASCRIPT

Inconvénient : Pas d'interaction possible avec une base de données directement,

Un formulaire peut être traité :

- Soit au niveau du serveur – c'est le serveur qui va travailler sur le formulaire

Exemple : Vérification de la validité des champs, apparition d'infos bulles, interrogation d'une base de données

Langage : PHP, ASP, ….

Inconvénient : Rechargement de la page, plus de données qui circulent sur le réseau

DOM = Document Object Model

- API (Application Programming Interface) pour la manipulation de HTML / XML
- Définit la structure logique des documents
- Définit la façon d'y accéder, de la manipuler

➔Créer des documents

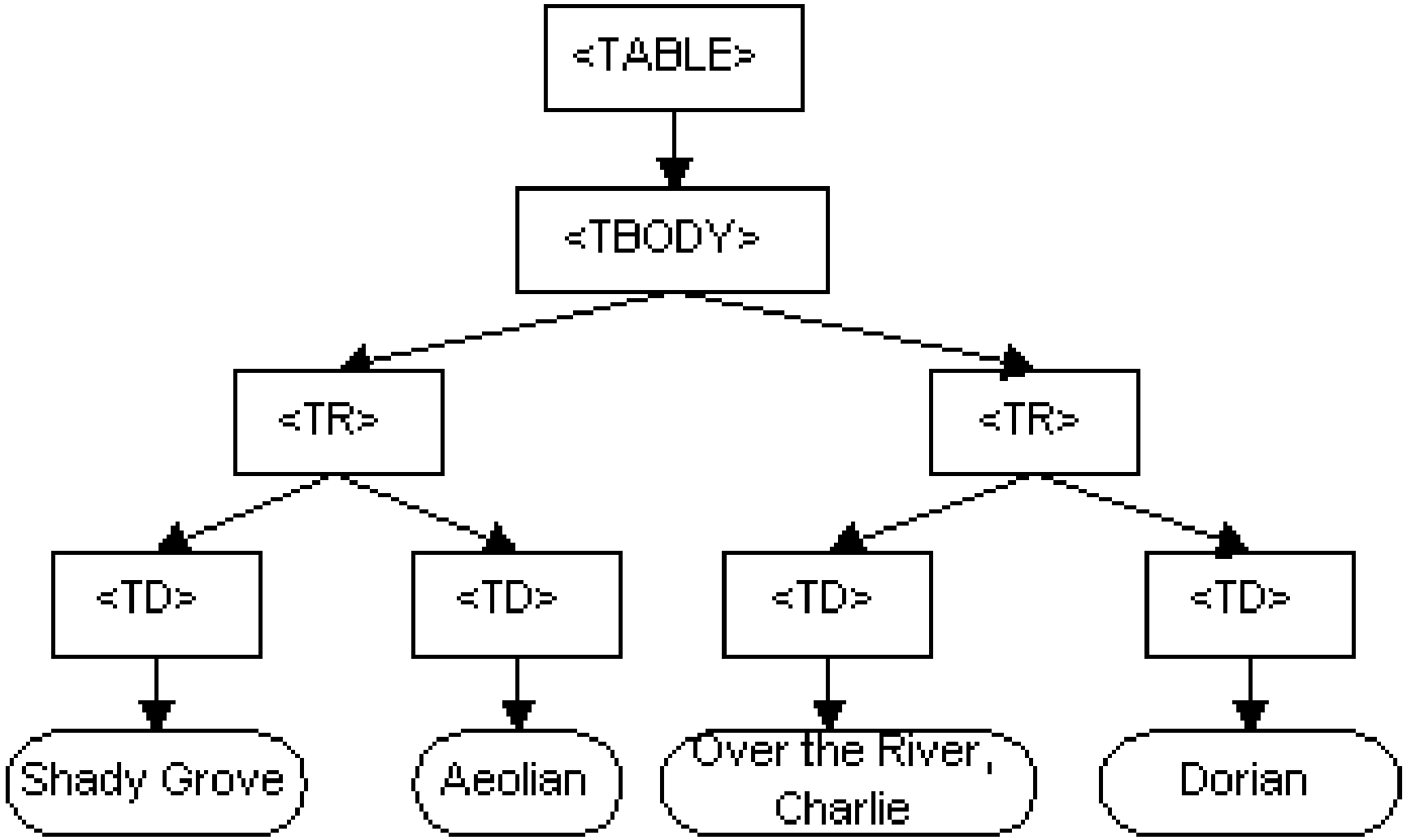
➔Parcourir leur structure

➔Ajouter, effacer, modifier des éléments

➔Ajouter, effacer, modifier leur contenu

Qu’est-ce que le DOM ?

```
<tab
<tbo
<tr>

</tr
<tr>

</tr
</tb
</ta
```



Qu’est-ce que le DOM ?

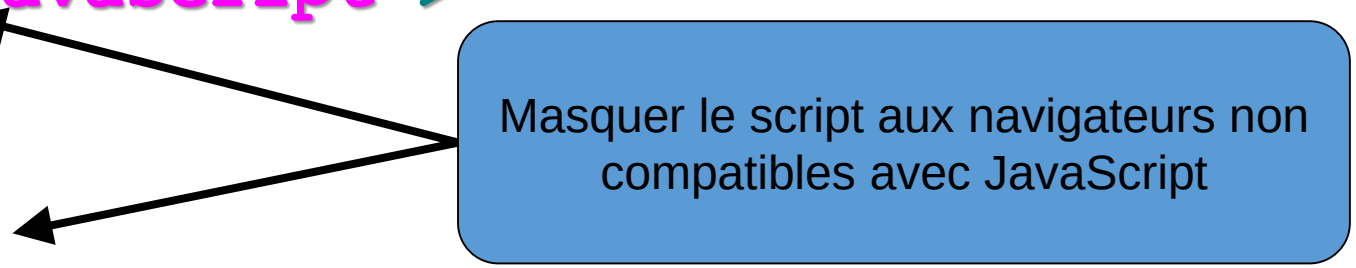
- Représentation arborescente du document
- Modèle objet (structure + méthodes)
- Permet la manipulation du document
- Une implémentation : JavaScript…
- … Des implémentations :
  - JavaScript IE
  - JavaScript Mozilla / Firefox
  - JavaScript Opera
  - …

JavaScript : Principe

- Langage de script objet
- Syntaxe style C / C++ / Java
- Sensible à la casse
- N’est PAS du Java
- Exécuté par le client Web
- Peut être désactivé sur le client
- Nombreux objets pour la manipulation HTML
- Gestion des événements HTML
- Rendre les pages dynamiques (HTML+CSS+JS)
- Haut niveau d’incompatibilité…

JavaScript : Balise script

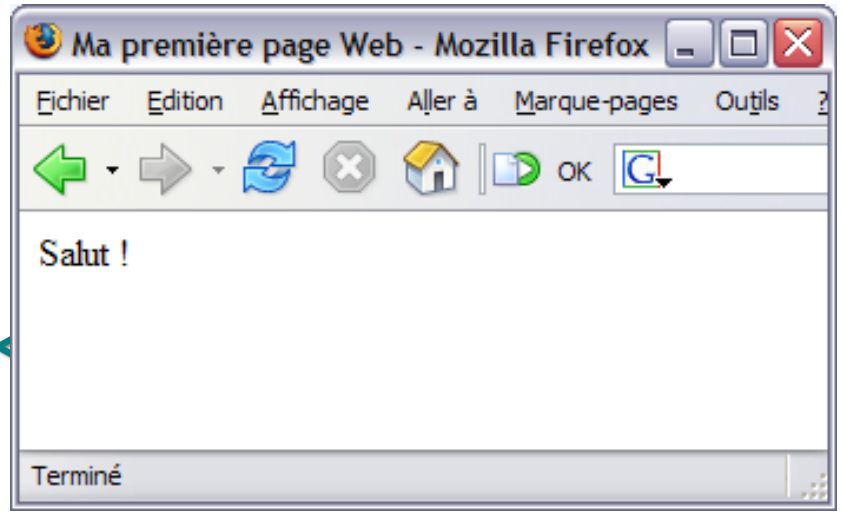
```
<script type="text/javascript"
  language="JavaScript">
<!--
script
// -->
</script>
```




```
<script type="text/javascript" language="JavaScript"
  src="URI">
</script>
```

JavaScript : Exemple

```
<html>
 <head>
  <title>Ma première page Web<
 </head>
 <body>
 <script type="text/javascript" language="JavaScript">
  <!--
   document.writeln("Salut !") ;
  // -->
 </script>
</body>
</html>
```



Variables

- Déclaration de variables facultative
- Variables non typées à la déclaration

```
var nom_variable ;
```

- Typage dynamique à l'affectation
- Types gérés:
  - Nombres (`10, 3.14`)
  - Booléens (`true`, `false`)
  - Chaînes ("Salut !", 'Salut !')
  - `null`
  - `undefined`

Structures conditionnelles

```
if (condition)
{
 instructions ;
}
[ else
{
 instructions ;
} ]
```

Structures conditionnelles

```
switch (expression)
{
   case étiquette :
      instructions ;
      break ;
   case étiquette :
      instructions ;
      break ;
   default :
      instructions ;
}
```

Structures itératives

```
while (condition)
{
  instructions ;
}

do
{
  instructions ;
}
while (condition) ;
```

Structures itératives

```
for (instr ; condition ; instr)
{
 instructions ;
}

for (variable in objet)
{
 instructions ;
}
```

Commentaires

```
// Commentaire ligne

/* Commentaire multi-lignes */
```

Fonctions

- Valeur de retour non typée
- Arguments non typés

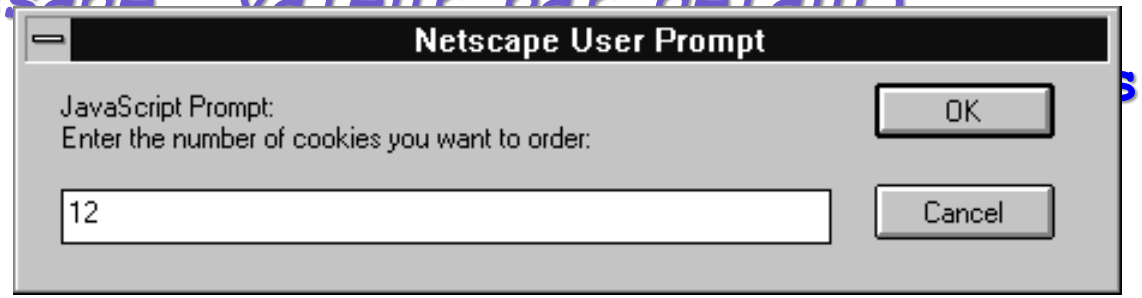
```
// Déclaration
function ma_fonction(arguments)
{
   instructions ;
   return quelque_chose; // ou pas…
}

ma_fonction(12) ; // Appel
```

Objets prédéfinis

- **window**
  - **alert(*messag*
  // Message d'

  - **confirm(*messa***
  // Message de **true ou false**

  - **prompt(*message, valeur par défaut*)**
  // Boîte de **saisie**



Objets prédéfinis

- **document**
  - **write(*message*)**

```
// Ecrire dans le document
```

  - **writeln(*message*)**

```
// Ecrire dans le document (retour à la ligne)
```

Chaînes : Propriétés & Méthodes

- Propriétés
  - **length**
  - …
- Méthodes
  - **charAt(*index*)**
  - **charCodeAt(*index*)**
  - **concat(*chaine2*, *chaine3*, …)**
  - **fromCharCode(*code1*, *code2*, …)**
  - **indexOf(*aiguille*[, *index*])**
  - **lastIndexOf(*aiguille*[, *index*])**
  - **match(*expr_reg*)**

Chaînes : Exemples

```
var s = "Bon anniversaire Benjamin" ;
document.write(s.charAt(2)) ;
➔ n
document.write(s.charCodeAt(2)) ;
➔ 110
document.write(s.concat(" du groupe C12")) ;
➔ Bon anniversaire Benjamin du groupe C12
document.write(String.fromCharCode(49, 50)) ;
➔ 12
document.write(s.indexOf("Benjamin")) ;
➔ 17
document.write(s.lastIndexOf("a")) ;
➔ 21
document.write(s.match(/Benjamin$/)) ;
➔ Benjamin                    (null si non trouvé)
```

Chaînes : Méthodes

- Méthodes
  - **replace(*expr_reg, nouvelle_chaine*)**
  - **search(*expr_reg*)**
  - **slice(*debut*[, *fin*])**
  - **split(*separateur*[, *limite*])**
  - **substr(*debut*[, *taille*])**
  - **substring(*debut, fin*)**
  - **toLowerCase()**
  - **toUpperCase()**
- Opérateurs
  - **+**

Chaînes : Exemples

```
var s = "Bon anniversaire Benjamin" ;
document.write(s.replace(/i/g, 'I')) ;
➔ Bon annIversaIre BenjamIn
document.write(s.search(/n{2}/i)) ;
➔ 5
document.write(s.slice(17)) ;
➔ Benjamin
document.write(s.split(" ")) ;
➔ Bon,anniversaire,Benjamin
document.write(s.substr(4, 12)) ;
➔ anniversaire
document.write(s.substring(4, 16)) ;
➔ anniversaire
document.write(s.toUpperCase()+s.toLowerCase()) ;
➔ BON ANNIVERSAIRE BENJAMINbon anniversaire benjamin
```

Propriétés & Fonctions supérieures

- Propriétés
  - **`Infinity, NaN, undefined`**
- Fonctions
  - **`eval(`*`chaine`*`)`**
  - **`isFinite(`*`nombre`*`)`**
  - **`isNaN(`*`objet`*`)`**
  - **`parseFloat(`*`chaine`*`)`**
  - **`parseInt(`*`chaine`*`)`**
  - **`escape(`*`chaine`*`)`**
  - **`unescape(`*`chaine`*`)`**

Propriétés & Fonctions supérieures

```
document.write(eval("Math.pow(3+2, 2)")) ;
➔ 25
document.write(isFinite(Math.log(0))) ;
➔ false
document.write(isNaN("abcd")) ;
➔ true
document.write("12.34"+2) ;
➔ 12.342
document.write(parseFloat("12.34")+2) ;
➔ 14.34
document.write(escape("Bon anniversaire")) ;
➔ Bon%20anniversaire
document.write(unescape("Bon%20anniversaire")) ;
➔ Bon anniversaire
```

Tableaux

- Objet Array
- Déclaration

```
var tab1 = new Array(taille) ;
var tab2 = new Array(1, "a", 9, …) ;
                index ➔  0   1   2   …
```

- Utilisation

```
window.alert(tab2[0]) ; // 1
tab2[2] = 6 // 6 remplace 9
```

- Accroissement automatique de la taille

```
var tab1 = new Array(2) ;
tab1[200] = 5 ;
```

Tableaux

- Parcours

```
var tab2 = new Array(1, "a", 9) ;
tab2[200] = 12 ;
for (i in tab2)
 window.alert("tab2[" + i + "] = "
                    + tab2[i]) ;
// tab2[0] = 1
// tab2[1] = a
// tab2[2] = 9
// tab2[200] = 12
```

Tableaux : Propriétés & Méthodes

- Propriétés
  - **length**
  - …
- Méthodes
  - **concat(*tab2*, *tab3*, …)**
  - **join(*sépar*)**
  - **pop()**
  - **push(*val1*, *val2*, …)**
  - **shift()**
  - **unshift(*val1*, *val2*, …)**
  - **slice(*début*[, *fin*])**

Contrôle de formulaires

- Vérifier la cohérence de la saisie
- Contrôles sur le client
- Évite les transmissions client / serveur
- Contrôles possibles:
  - Présence de valeur
  - Numérique / Chaîne
  - Expressions régulières
- Événement **`onSubmit`**

Contrôle de formulaires

```
<html><head><title>Contrôle</title>
<script type="text/javascript">
function verif() {
  if (document.formu.txt.value != '')
       return window.confirm('Envoyer ?') ;
  return false ; }
</script></head><body>
<form name="formu" action="URI" method="GET" onSubmit="return
  verif() ;">
  <input type="text" name="txt">
  <input type="submit" value="Envoyer">
</form></body></html>
```

Formulaires : Propriétés & Méthodes

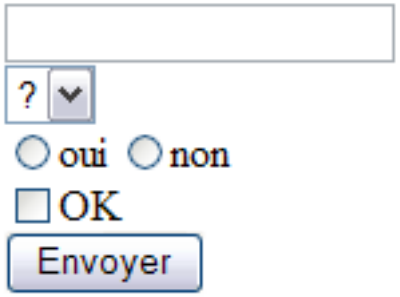
- Propriétés
  - **`action`**
  - **`elements`**
  - **`encoding`**
  - **`length`**
  - **`method`**
  - **`name`**
  - **`target`**
- Méthodes
  - **`reset()`**
  - **`submit()`**

Objets *commandes* de formulaires

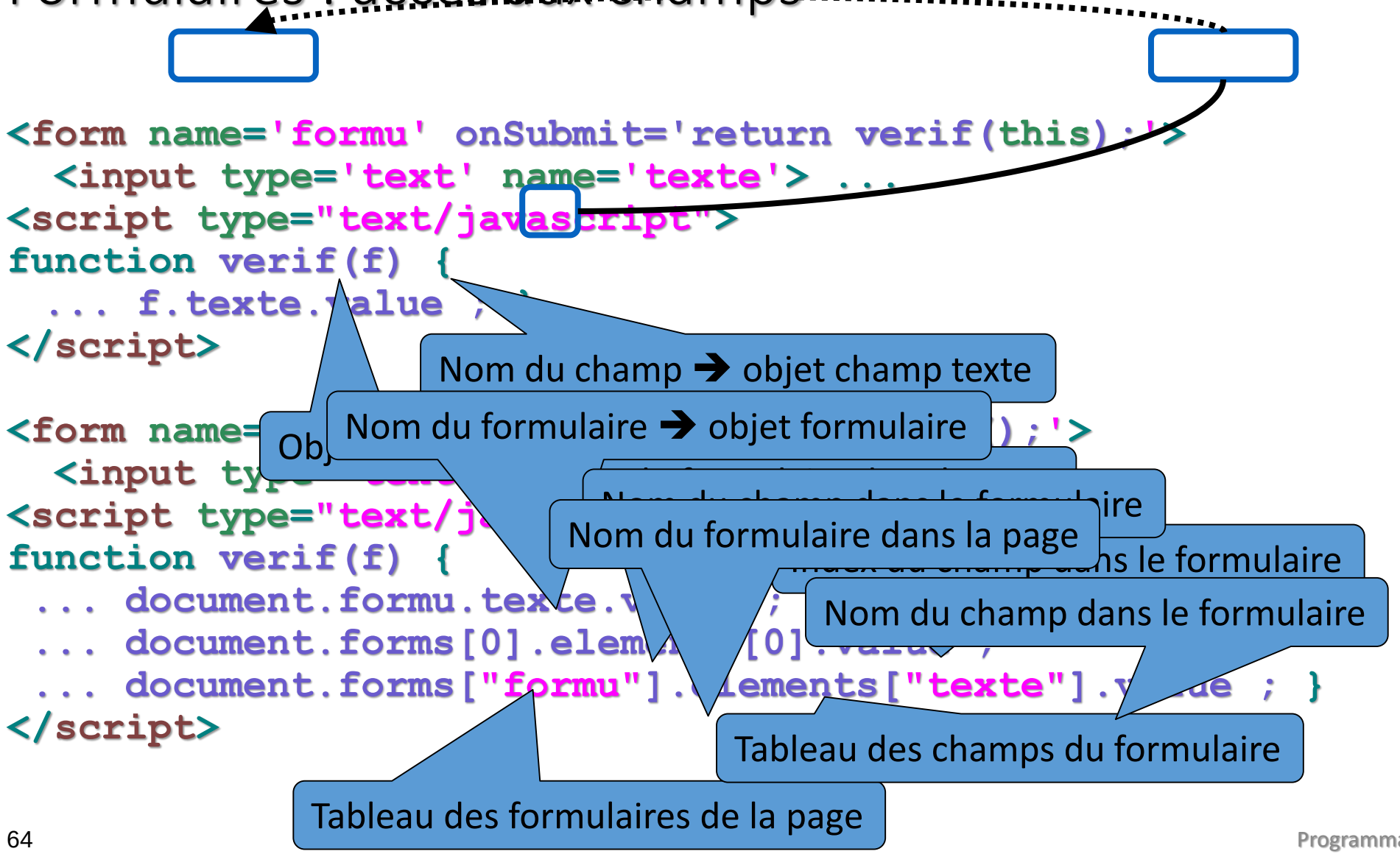
- **Text**
- **Textarea**
- **Hidden**
- **Password**
- **CheckBox**
- **Radio (/!\ tableau de /!\)**
- **Submit / Reset**
- **Select**
- **Option**
- **FileUpload**

Formulaires : Exemple

```
<form name='formu' onSubmit='return verif(this
  <input type='text' name='texte'><br>
  <select name='sel'>
    <option>?
    <option value=1>Un
    <option value=2>Deux
</select><br>
  <input type='radio' name='rad' id='rad1'>
  <label for='rad1'>oui</label>
  <input type='radio' name='rad' id='rad2'>
    <label for='rad2'>non</label><br>
  <input type='checkbox' name='chk' id='chk1'>
    <label for='chk1'>OK</label><br>
  <input type='submit' value='Envoyer'>
</form>
```



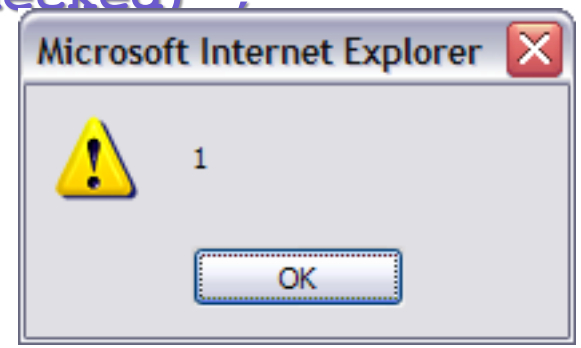
Formulaires : accès aux champs

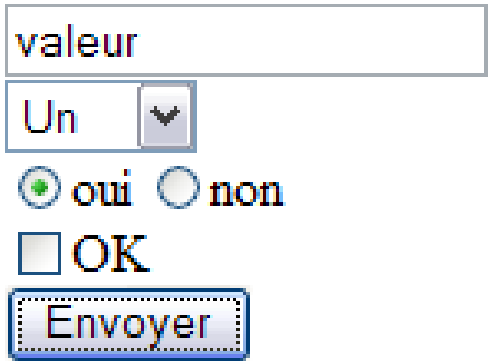



```
<form name='formu' onSubmit='return verif(this);'>
  <input type='text' name='texte'> ...
<script type="text/javascript">
function verif(f) {
  ... f.texte.value ;
</script>

<form name=
  <input ty
<script type="text/j
function verif(f) {
  ... document.formu.texte.v
  ... document.forms[0].elem
  ... document.forms["formu"].elements["texte"].v
</script>
```

Formulaires : Exemple

```
<script type="text/javascript">
function verif(f)
{
   window.alert(f.texte.value) ;
   window.alert(f.sel.selectedIndex) ;
   window.alert(f.sel[f.sel.selectedIndex].text) ;
   window.alert(f.sel[f.sel.selectedIndex].value) ;
   window.alert(f.rad[0].checked) ;
   window.alert(f.chk.checked) ;
   return false ;
}
</script>
```





Dates : Propriétés & Méthodes

- Méthodes
  - Constructeur
  - getDay(), attention de 0 (dimanche) à 6 (samedi)…
  - getDate() / setDate()
  - getMonth() / setMonth(), attention de 0 à 11…
  - getHours()/ setHours()
  - getMinutes() / setMinutes()
  - getTime() / setTime()
  - getYear() / setYear() / getFullYear() / setFullYear()
  - parse()

Dates : Exemples

```
var today = new Date()
document.write(today) ;
var birthday = new Date("December 17, 1995 03:24:00")
document.write(birthday) ;
birthday = new Date(95,11,17)
document.write(birthday) ;
birthday = new Date(95,11,17,3,24,0)
document.write(birthday) ;
document.write(birthday.getDay()) ;
document.write(birthday.getDate()) ;
birthday.setDate(25) ;
document.write(birthday) ;
```

Tue Nov 02 2010 00:11:36 GMT+0100

Sun Dec 17 1995 03:24:00 GMT+0100

Sun Dec 17 1995 00:00:00 GMT+0100

Sun Dec 17 1995 03:24:00 GMT+0100

0

17

Mon Dec 25 1995 03:24:00 GMT+0100

Dates : Exemples

```
document.write(birthday.getMonth()) ;
birthday.setMonth(10) ;
document.write(birthday) ;
document.write(birthday.getYea
birthday.setYear(96) ;
document.write(birthday) ;
document.write(birthday.getFul
birthday.setFullYear(2010) ;
document.write(birthday) ;
document.write(Date.parse("Aug 9, 1995")) ;
```

```
11
Sat Nov 25 1995 03:24:00 GMT+0100
95
Mon Nov 25 1996 03:24:00 GMT+0100
1996
Thu Nov 25 2010 03:24:00 GMT+0100
807919200000
```

Images : Propriétés & Méthodes

- Propriétés
  - **`complete`**
  - **`width`**
  - **`height`**
  - **`name`**
  - **`src`**
  - …
- Méthodes
  - **`constructeur`**
    - **`Image()`**
    - **`Image(`*`largeur, hauteur`*`)`**

Images: Exemples

```
<script type="text/javascript">
// Une image
rouge = new Image(100, 100) ;
rouge.src = 'rouge.gif' ; // Préchargement

// Une autre image
vert = new Image(100, 100) ;
vert.src = 'vert.gif' ; // Préchargement

// Modification de la 13e image de la page Web
document.images[12].src = rouge.src ;

// Modification de la 15e image de la page Web
document.images[14].src = rouge.src ;
</script>
```

Relations entre code HTML et DOM

- Deux visions, normalement concordantes :
  - Le code HTML produit par le concepteur
  - L'interprétation qui en faite par le navigateur
- Discordances possibles :
  - Erreurs dans le code (code non valide)
  - Balises non supportées (HTML 5, par exemple)
  - Modifications de la page par JavaScript

- Comment explorer l'état réel de l'interprétation du code HTML / JavaScript par la navigateur ?

Solutions pour modifier le DOM

- innerHTML
  - Construire une chaîne de caractères contenant du code HTML
  - Affecter cette chaîne de caractères à l'attribut **`innerHTML`** d'un élément de la page
  - Le navigateur interprète le contenu de la chaîne de caractères affectée pour modifier la structure du document
- DOM « pur »
  - Construire de nouveaux éléments logiques du document avec des méthodes JS
  - Construire les liens de parenté entre ces éléments

`innerHTML` : exemple

- Identifier un élément HTML
  `<div id="un_id"></div>`
- Accéder à un élément
  `e = document.getElementById("un_id");`
- Construire une chaîne contenant du HTML
  `s = "Voici <b>un texte</b>";`
- Modifier le contenu de l'élément
  `e.innerHTML = s;`
- Interprétation « automatique » par le navigateur du nouveau contenu pour modifier le document

Gestion des événements

- Interactions HTML / JavaScript
- Possibilité d'associer du code JavaScript à certains événements dans la page Web
- Evénements
  - OnMouseOver
  - OnMouseOut
  - OnClick
  - OnKeyDown
  - OnFocus
  - OnBlur
  - …

Mise en place des événements

```
<a href='URI_cible'
  onMouseOver="code_javascript1"
  onMouseOut="code_javascript2">Support</a>
```

```
<a href='URI_cible'
  onMouseOver="a=1"
  onMouseOut="b=2">Support</a>
```

```
<a href='URI_cible'
  onMouseOver="allumer()"
  onMouseOut="eteindre()">Support</a>
```

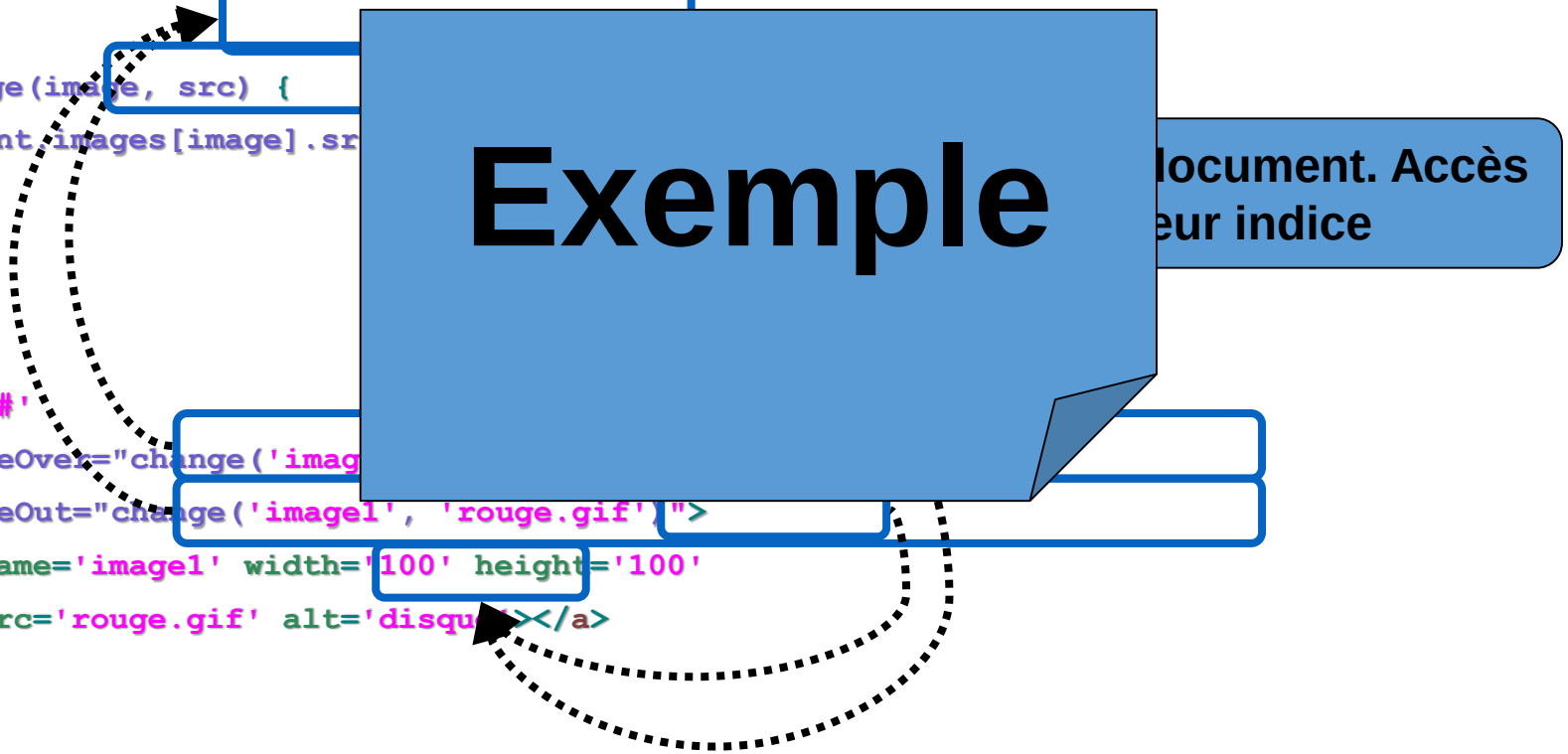
Événement onKeyUp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>    <head> <title>onKeyUp</title>
  <script type='text/javascript' language='JavaScript'>
  <!–
  function maj() {
         document.f.mail.value =
                   document.f.pren
                   +document.f.nom
  // -->
  </script>
  </head> <body>
  <form name='f'>
  Nom    <input type='text' name='nom'     onKeyUp='maj()'><br>
  Prenom <input type='text' name='prenom' onKeyUp='maj()'><br>
  Login  <input type='text' name='mail' disabled>
  </form>
</body> </html>
```

Exemple

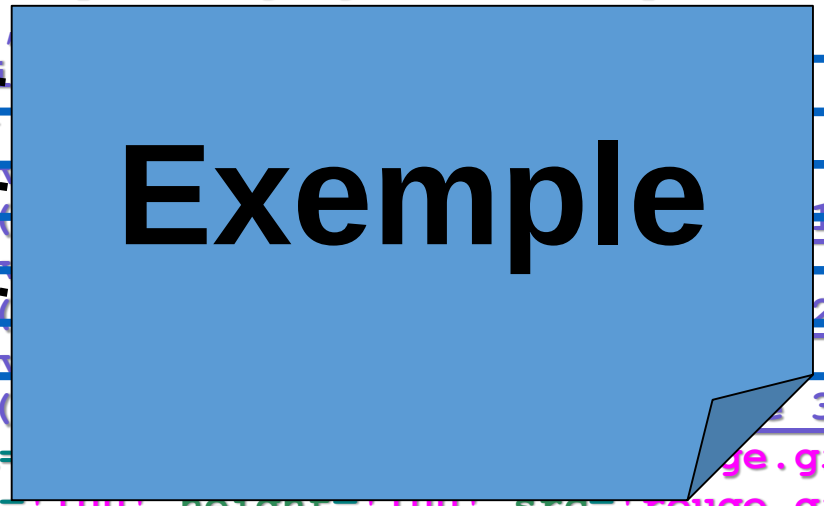
Événement onMouseOver / Out

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>    <head> <title>Rollover</title>
   <script type='text/javascript' language='JavaScript'>
   <!--
   function change(image, src) {
          document.images[image].sr
   // -->
   </script>
   </head>
   <body>
        <a href='#'
          onMouseOver="change('imag
          onMouseOut="change('image1', 'rouge.gif')">
          <img name='image1' width='100' height='100'
               src='rouge.gif' alt='disqu'></a>
   </body>
</html>
```

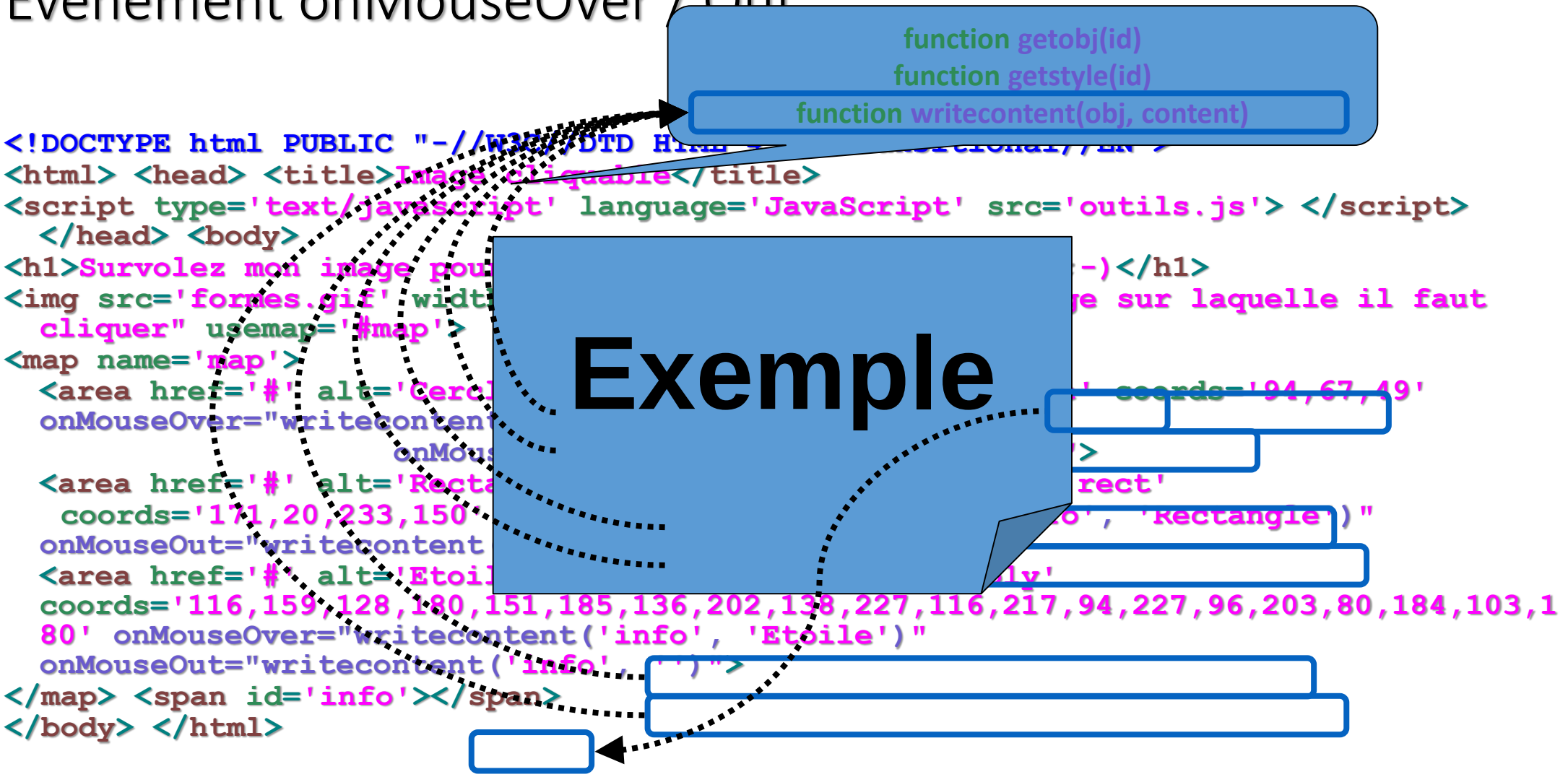


Événement onMouseOver / Out

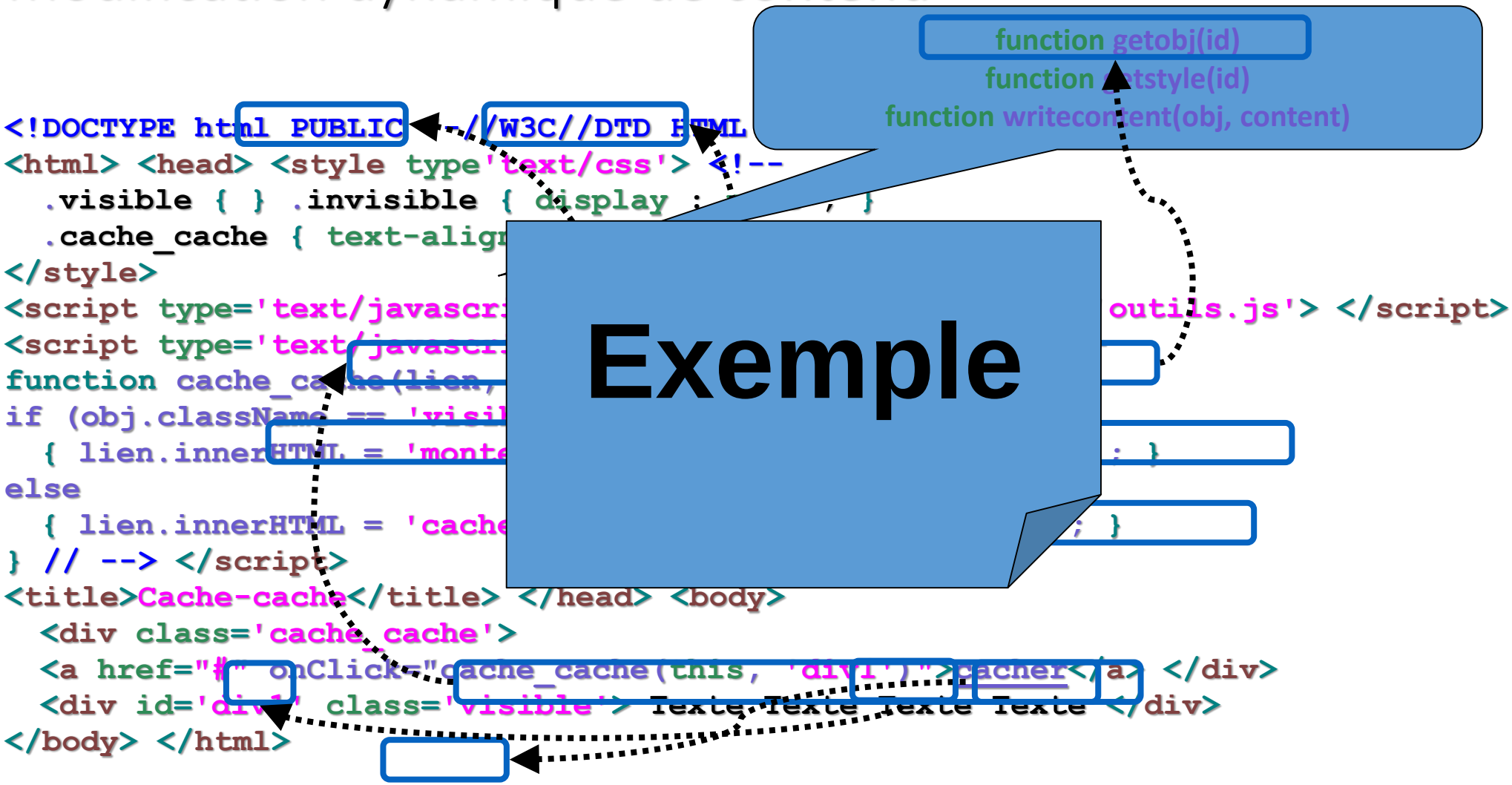
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html> <head> <title>Rollover</title>
  <script type='text/javascript' language='JavaScript'>
     function change(image,
          document.images[i
  </script> </head> <body>
  <li><a href='#' onMouseO
        onMouseOut="change(
  <li><a href='#' onMouseO
        onMouseOut="change(
  <li><a href='#' onMouseO
        onMouseOut="change(
  <img name='image1' width=
  <img name='image2' width='100' height='100' src='rouge.gif'       alt='disque'>
  <img name='image3' width='100' height='100' src='rouge.gif'       alt='disque'>
</body> </html>
```



Événement onMouseOver / Out

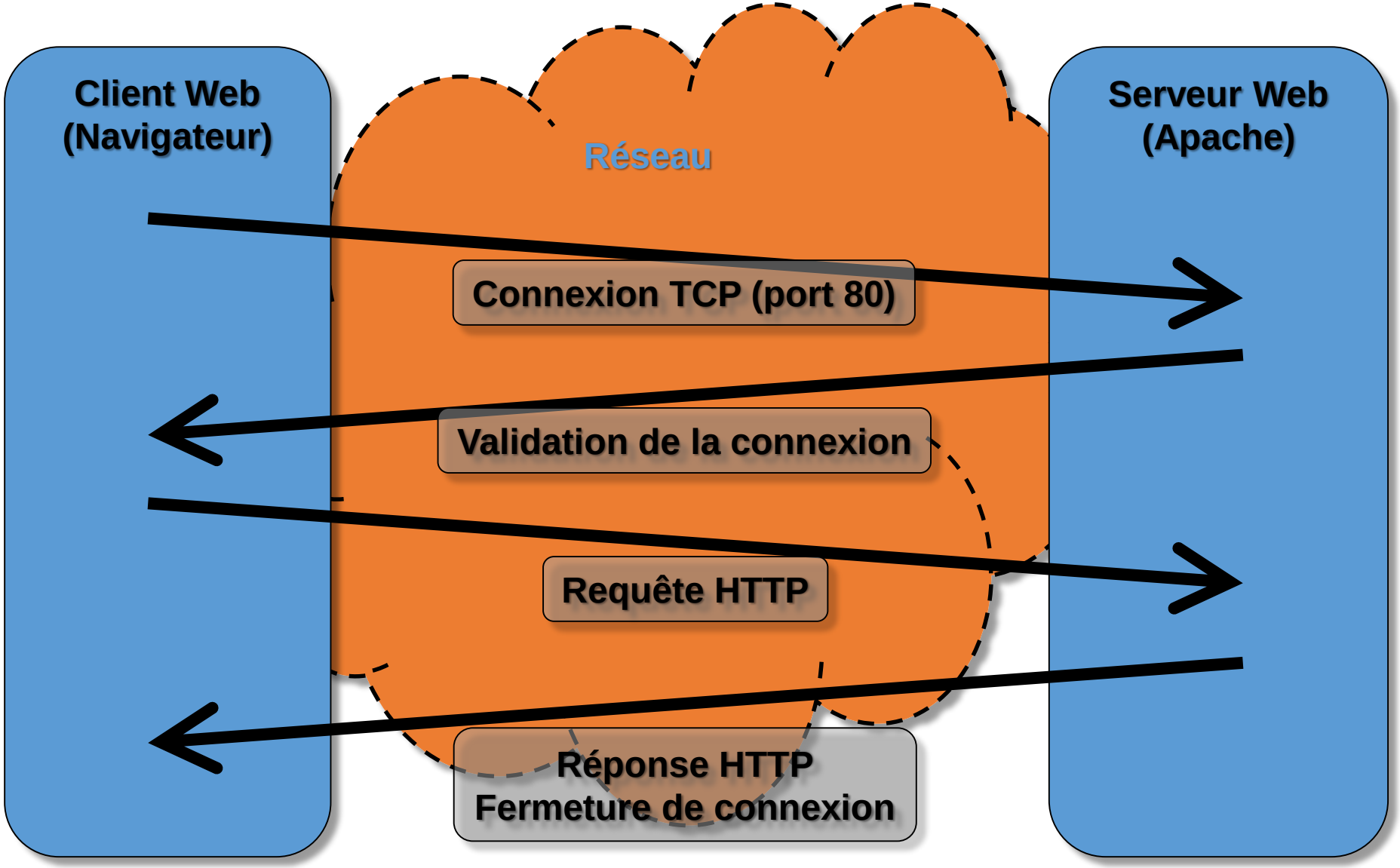


Modification dynamique de contenu



Protocole HTTP

Utilisation de HTTP/1.0



Méthodes de requête HTTP

HEAD

demande des informations concernant la ressource

GET

demande des informations et la ressource désignée

POST

envoi de données (formulaire vers le serveur) et demande la ressource désignée

PUT

enregistrement du corps de la requête à l ’URL indiquée

DELETE

suppression de la ressource désignée par l ’URL

Exemple de requête HTTP

```
GET / HTTP/1.0 ↵
Ligne blanche ↵
```

```
HEAD /page1.html HTTP/1.0 ↵
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
  en-US; rv:1.7.7) Gecko/20050414 Firefox/1.0.3 ↵
Ligne blanche ↵
```

États des réponses HTTP

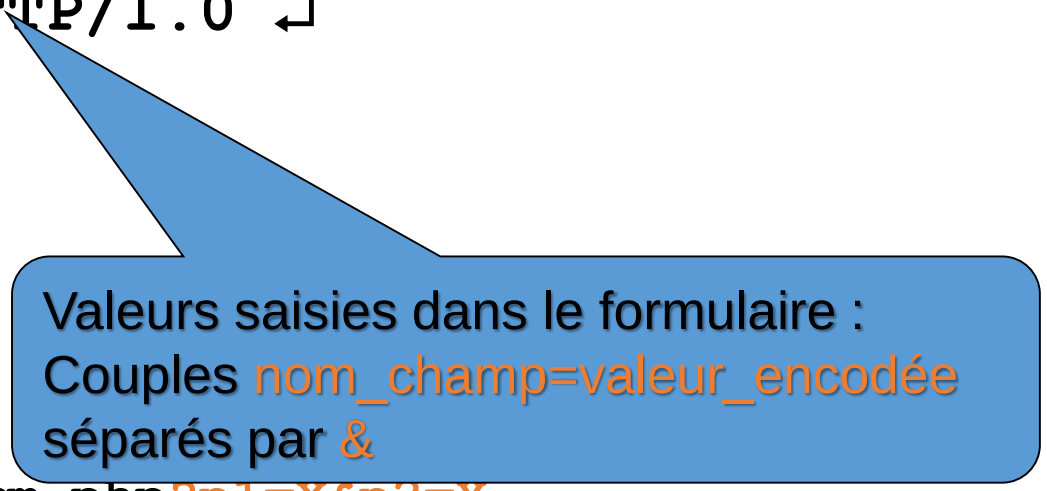
- Codes à 3 chiffres + phrase
- 1er chiffre : classe de réponse
  - `1xx` : Information (HTTP 1.1)
  - `2xx` : Succès
    - `200 OK`
  - `3xx` : Redirection
    - `304 NOT MODIFIED`
  - `4xx` : Erreur client
    - `403 FORBIDDEN`
    - `404 NOT FOUND`
  - `5xx` : Erreur serveur
    - `500 INTERNAL ERROR`

Soumission de formulaires

```
<form action="form.php" method="GET">
```

Requête HTTP :

```
GET form.php?p1=X&p2=Y HTTP/1.0 ↵
Ligne blanche ↵
```



Traduit dans l'URL : `http://abc/form.php?p1=X&p2=Y`

Peut donc être mis en favori

Soumission de formulaires

```
<form action="form.php" method="POST">
```

Requête HTTP :

```
POST form.php HTTP/1.0 ↵
Content-Type: application/x-www-form-urlencoded ↵
Content-Length: 9 ↵
Ligne blanche ↵
p1=X&p2=Y
```

Valeurs saisies dans le formulaire :
Couples nom_champ=valeur_encodée
séparés par &

NON traduit dans l'URL

Ne peut donc PAS être mis en favori

Programmation Web : PHP

PHP: Langage de script pour le Web

- Qu'est-ce que PHP ?
  - Langage de script. Principalement utilisé côté serveur
  - Créé en 1994-1995 par Rasmus Lerdorf
  - Acronyme initial : **P**ersonal **H**ome **P**age
  - Acronyme récursif : **P**HP: **H**ypertext **P**reprocessor
  - Extension utilisée sur ≈80% des serveurs Web
  - Langage multi plate-forme (UNIX / Windows…)
  - Open Source
  - Versions actuelles (*source w3techs.com, 03/14*) :
    - PHP5 >97%
    - PHP4 <3% - PHP3 <0,1%

Langage de script ?

- Langage impératif
- Langage interprété
- Pas de compilation
- Le code source « est » le programme
- Typage faible
- Programmation orientée objet possible

Utilité et utilisation de PHP

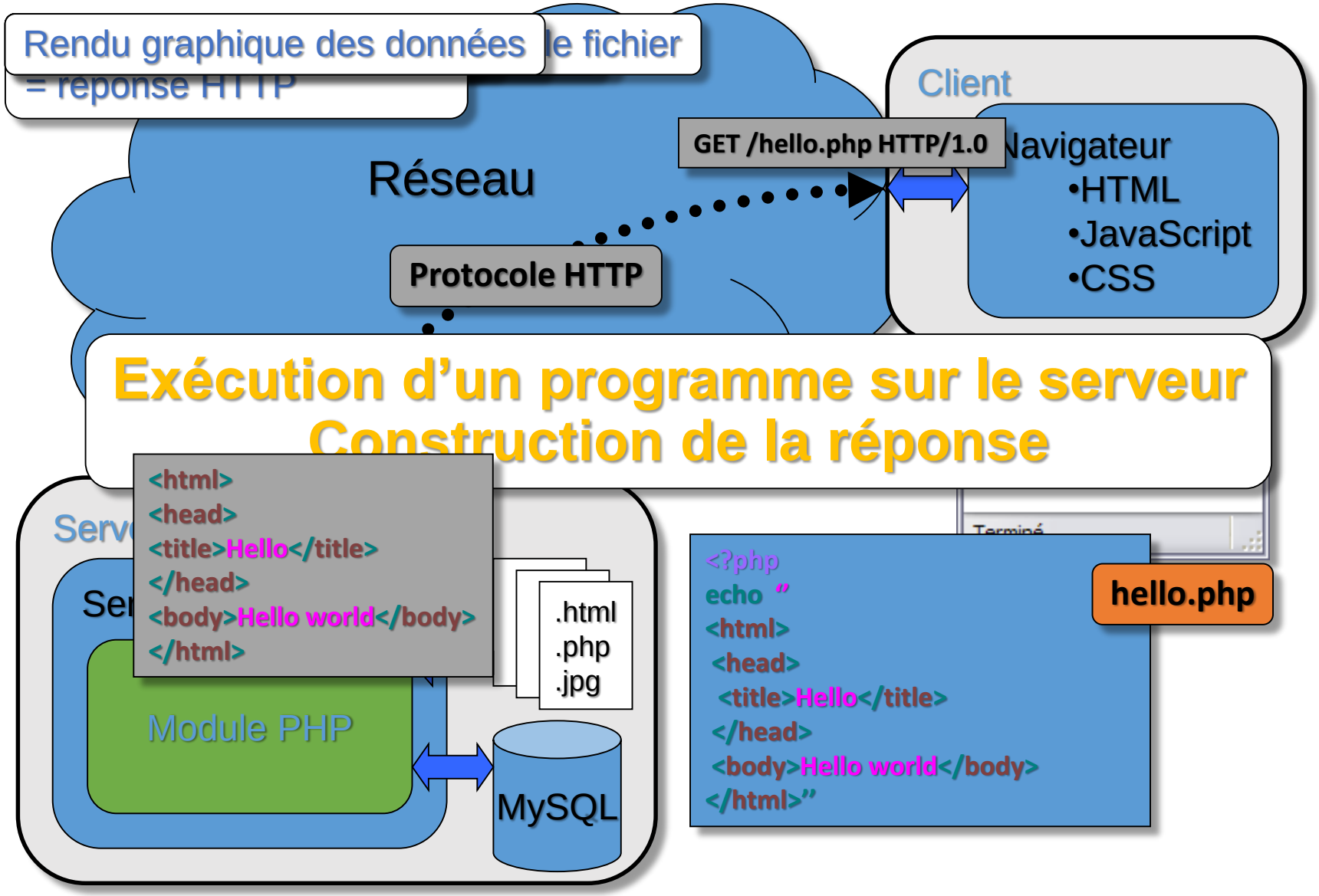
- Création de pages Web « dynamiques », fabriquées à la volée, construite à la demande
- Interface entre un serveur Web et des bases de données
- Création d'applications Web

Principales fonctionnalités de PHP

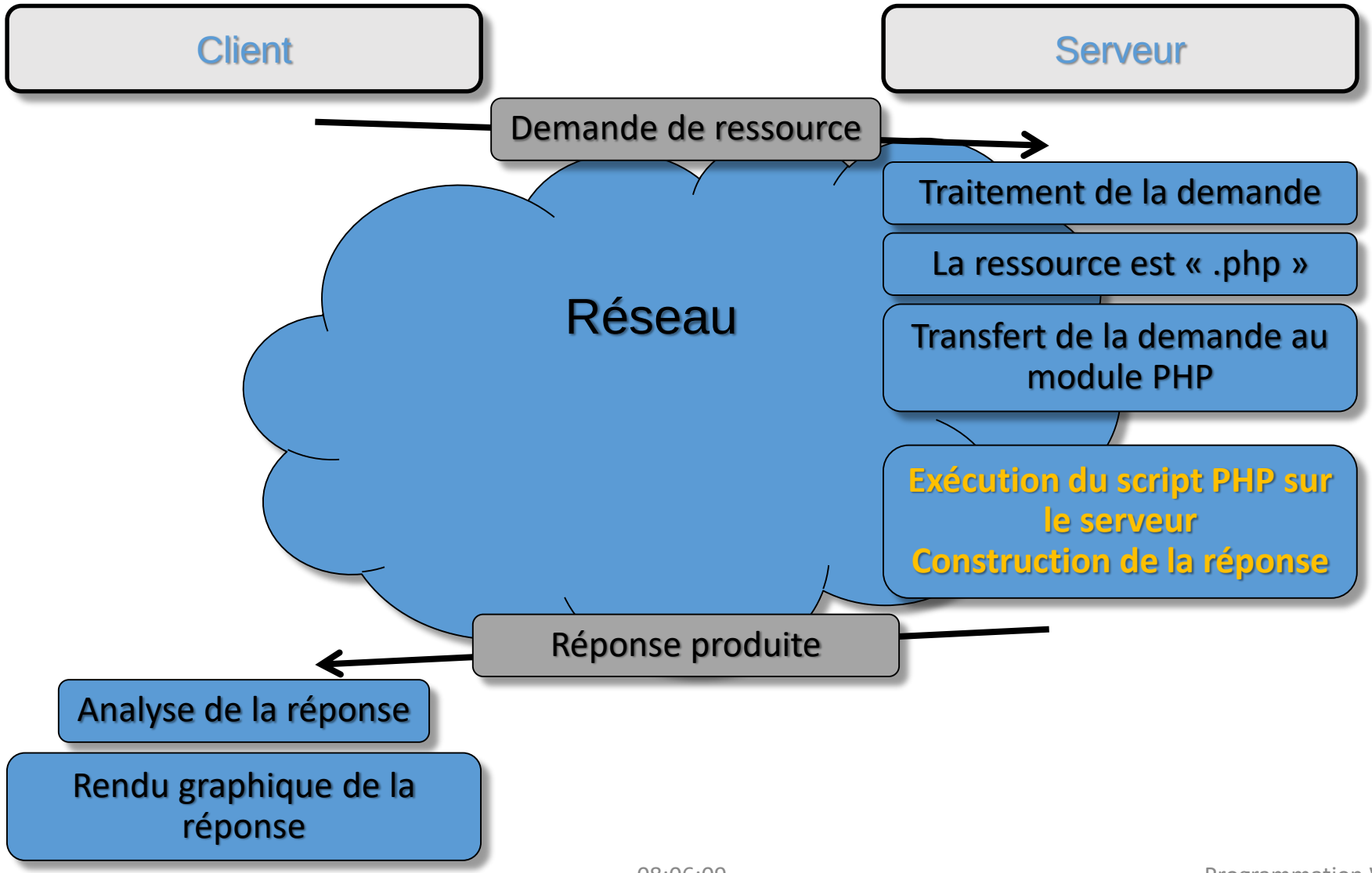
- Manipulation de chaînes et tableaux
- Calendrier / dates / heures
- Fonctions mathématiques
- Accès au système de fichiers
- Manipulation d'images
- HTTP / FTP / IMAP
- Bases de données (Oracle, MySQL, …)
- XML
- …

Fonctionnement

Fonctionnement de PHP



Fonctionnement de PHP



Fonctionnement de PHP au dpt INFO

A- Édition de ~cutron01/public_html/a.php

B- Enregistrement de ~cutron01/public_html/a.php

1- http://cutron01/a.php

2- Requête HTTP

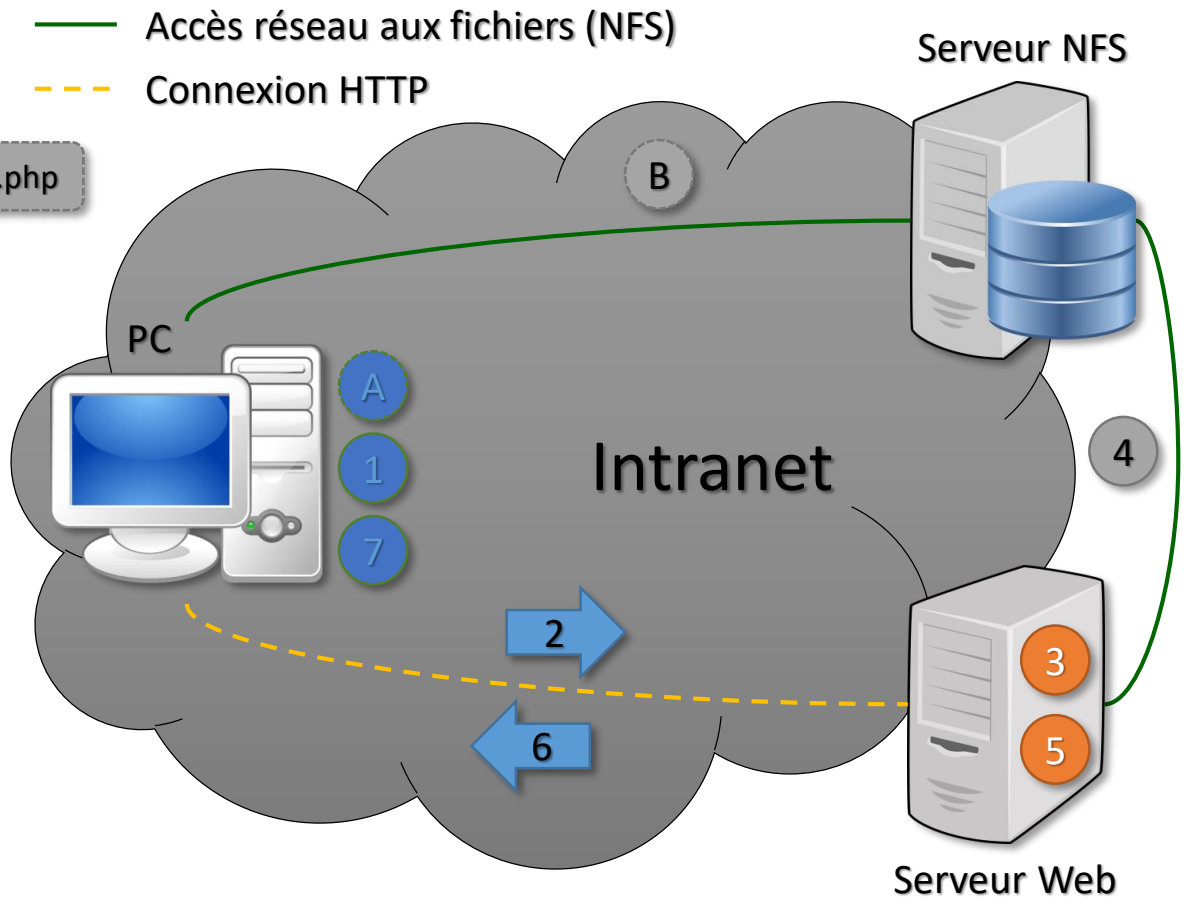
3- Traitement de la requête

4- Accès à ~cutron01/public_html/a.php

5- Exécution de a.php

6- Réponse HTTP produite par a.php

7- Affichage dans le navigateur



Programme en PHP

Délimitation du code PHP dans le fichier **`.php`** :

- `<?php Code PHP ?>`
- `<script language="PHP">`
  `Code PHP`
  `</script>`
- `<? Code PHP ?>`
- `<% Code PHP %>`

Fermeture optionnelle et déconseillée

Confusion avec JavaScript
➔ à bannir !!

sh
asp

Dépend de la configuration du serveur
➔ à bannir !!

Eléments de syntaxe PHP

- La syntaxe de PHP est celle de la famille « C » (C, C++, Java, …)

- Chaque instruction se termine par « ; »

- Commentaires:

```
/* jusqu'au prochain */
// jusqu'à la fin de la ligne
# jusqu'à la fin de la ligne
```

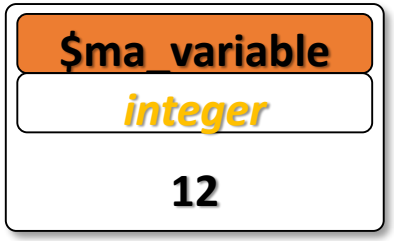
Variables et types

Les variables et les types de données

- Tout identificateur commence par « `$` »
- Les affectations sont réalisées grâce à « `=` »
- Les types sont :
  - Numérique entier : `12` ou réel : `1.54`
  - Chaîne: `"Hello"` ou `'Bonjour'`
  - Booléen: `true, false` (PHP 4)
  - Tableau: `$tab[2]=12`
  - Objet (PHP4, PHP5)
  - Ressource
  - `NULL`
- Les variables ne sont pas explicitement déclarées

Les variables et les types de données

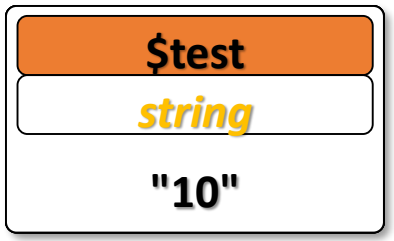
- La « déclaration » d'une variable correspond à sa première affectation
- Le type d'une variable est dynamique et est déterminé par la valeur qui lui est affectée
- Une variable possède donc :
  - Un nom (commençant par **$**)
  - Un type dynamique
  - Une valeur



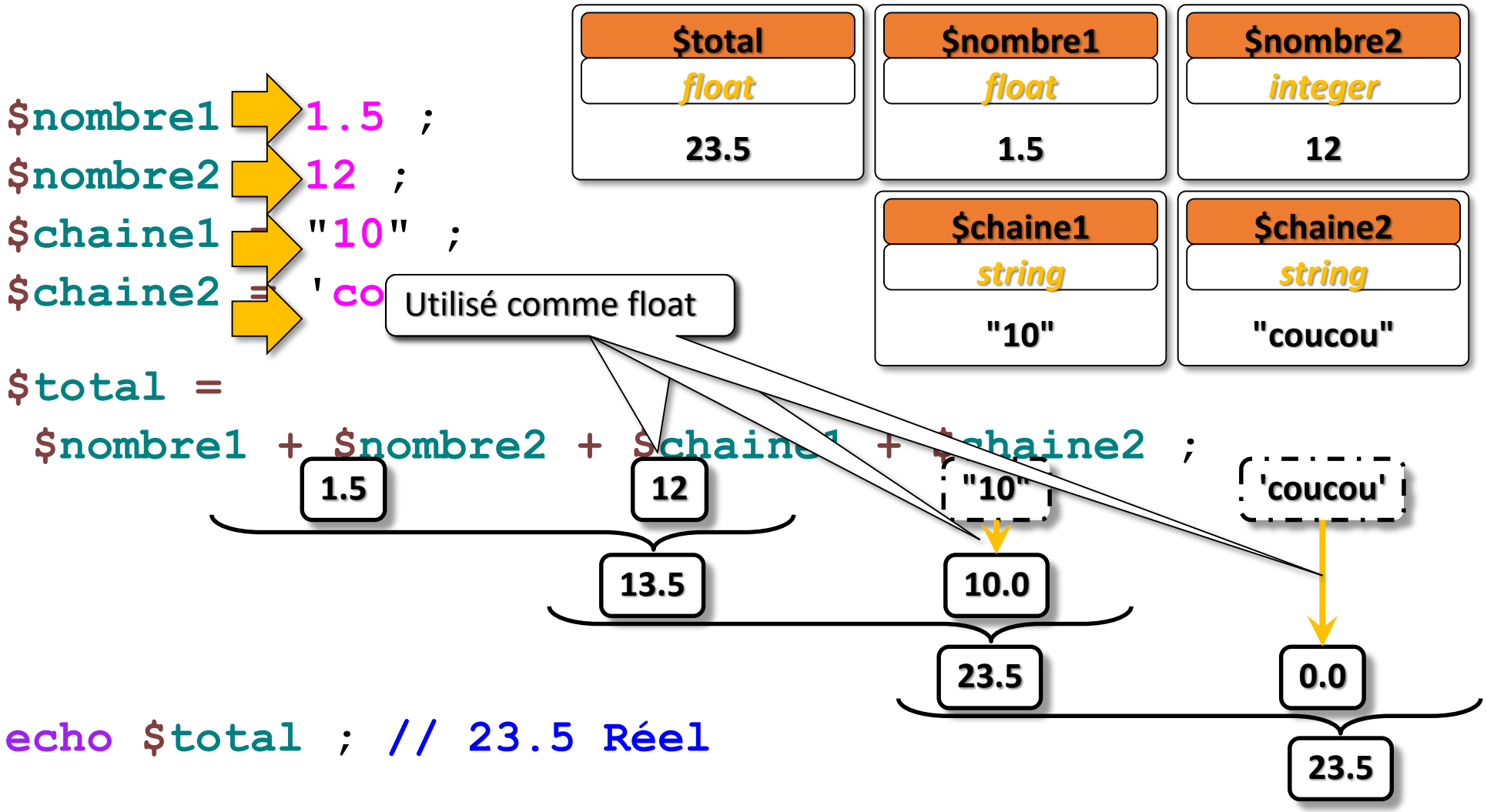
- Adaptation possible du type selon le contexte sans modification du type intrinsèque

Typage à l'affectation. Exemple

```
// Pas de déclaration préalable de variable

$test = 1.5 ;

$test = 12 ;

$test = array() ;

$test = "10" ;
```



Typage en fonction du contexte. Exemple

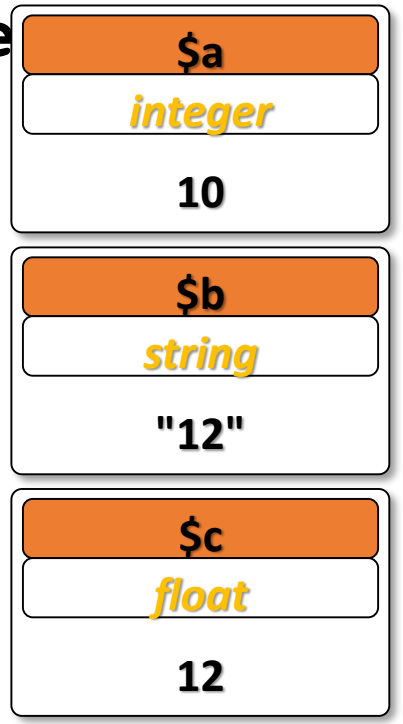



```
echo $total ; // 23.5 Réel
```

Modification de types

- Le type désiré peut être précisé : cast
- **`$variable = (nom_du_type) vale`**

```
$a = (integer) "10" ;

$b = (string) 12 ;

$c = (float) $b ;
```



Définition de constantes

```
<?php
define("ma_constante", "Bonjour à tous");
```

nom

valeur

Définition d'une constante

```
echo ma_constante
```

Utilisation de la constante

Chaînes de caractères

Substitution de variables dans les chaînes

- Guillemets doubles

```
$a="chaîne" ;
$b="voici une $a";
```

chaîne

voici une chaîne

- Guillemets simples

```
$a='chaîne' ;
$b='voici une $a';
```

chaîne

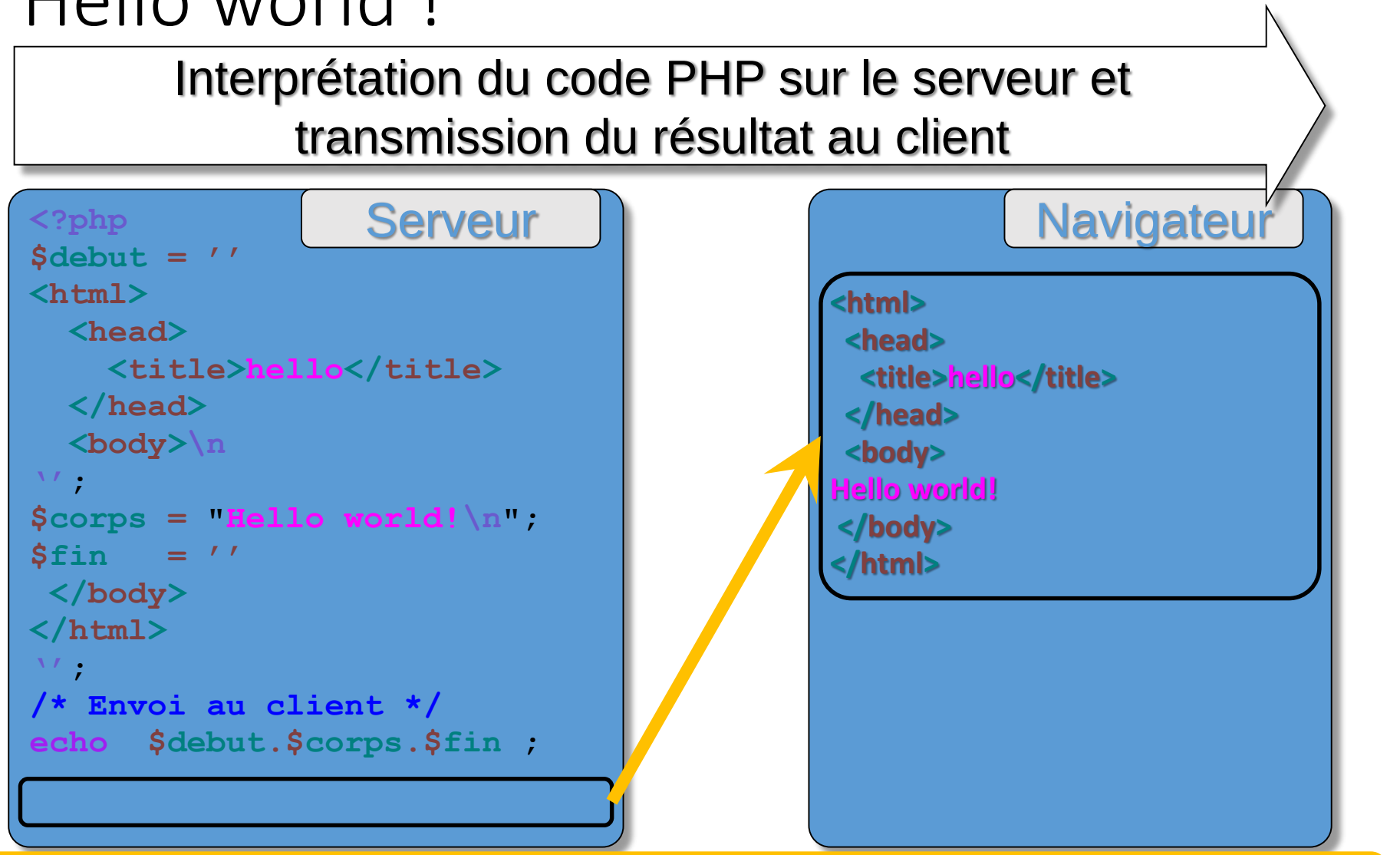
voici une $a

La commande echo

- Naïvement, permet d'envoyer du texte au navigateur du client (« écrire » la page au format HTML résultant de l'interprétation de PHP)
  - `echo "Bonjour" ;`
  - `$nom="Marcel" ; echo "Bonjour $nom" ;`

- Plus exactement, permet d'envoyer des octets au navigateur du client
  - Contenu HTML, XML, CSS, JavaScript, …
  - Données d'une image
  - Contenu d'un ficher PDF, Flash, etc.

Hello world !


Interprétation du code PHP sur le serveur et transmission du résultat au client
Serveur
<?php
$debut = ''
<html>
<head>
<title>hello</title>
</head>
<body>\n
'';
$corps = "Hello world!\n";
$fin = ''
</body>
</html>
'';
/* Envoi au client */
echo $debut.$corps.$fin ;
Navigateur
<html>
<head>
<title>hello</title>
</head>
<body>
Hello world!
</body>
</html>
Impossible de voir le code PHP depuis le navigateur !!

Opérateurs

Concaténation de chaînes

- Permet d'assembler plusieurs chaînes
- Réalisé grâce à l'opérateur point : « . »

```
"Bonjour " . "Marcel"
```

→ vaut `"Bonjour Marcel"`

```
$nb = 6*2 ;
"Acheter " . $nb . " oeufs"
```

→ vaut `"Acheter 12 oeufs"`

Les opérateurs arithmétiques

`$a + $b`	Somme
`$a - $b`	Différence
`$a * $b`	Multiplication
`$a / $b`	Division
`$a % $b`	Modulo (Reste de la division entière)

Les opérateurs d'in- et de dé-crémentation pré- et post-fixés

`$a++`	Retourne la valeur de `$a` puis augmente la valeur de `$a` de 1
`++$a`	Augmente la valeur de `$a` de 1 puis retourne la nouvelle valeur de `$a`
`$a--`	Retourne la valeur de `$a` puis diminue la valeur de `$a` de 1
`--$a`	Diminue la valeur de `$a` de 1 puis retourne la nouvelle valeur de `$a`

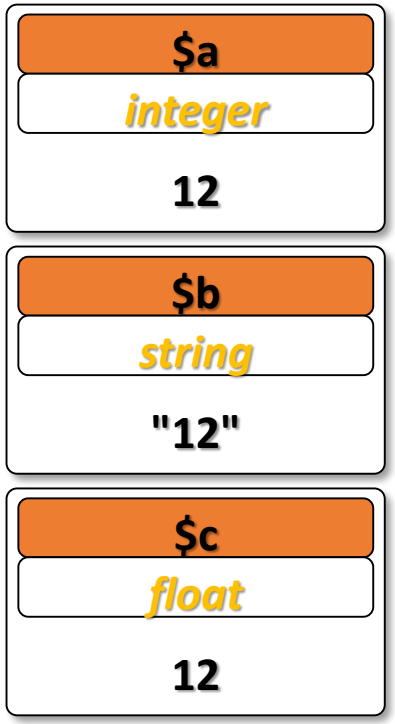
Les opérateurs de comparaison

`$a == $b`	Vrai si égalité entre les valeurs de `$a` et `$b`
`$a != $b`	Vrai si différence entre les valeurs de `$a` et `$b`
`$a < $b`	Vrai si `$a` inférieur à `$b`
`$a > $b`	Vrai si `$a` supérieur à `$b`
`$a <= $b`	Vrai si `$a` inférieur ou égal à `$b`
`$a >= $b`	Vrai si `$a` supérieur ou égal à `$b`
`$a === $b`	Vrai si `$a` et `$b` identiques (valeur et type)
`$a !== $b`	Vrai si `$a` et `$b` différents (valeur ou type)

Comparaison large / stricte

```php
$a = 12 ; $b = "12" ; $c = 13.0 ;

var_dump($a ==  $b) ;
var_dump($a ==  $c) ;
var_dump($c ==  $b) ;
var_dump($a !=  $b) ;
var_dump($a !=  $c) ;
var_dump($c !=  $b) ;
var_dump($a === $b) ;
var_dump($a === $c) ;
var_dump($c === $b) ;
var_dump($a !== $b) ;
var_dump($a !== $c) ;
var_dump($c !== $b) ;
```

```
boolean true
boolean false
boolean false
boolean false
boolean true
boolean true
boolean false
boolean false
boolean false
boolean true
boolean true
boolean true
```



Les opérateurs logiques

`[Expr1]` **`and`** `[Expr2]`	Vrai si `[Expr1]` et `[Expr2]` sont vraies
`[Expr1]` **`&&`** `[Expr2]`	idem
`[Expr1]` **`or`** `[Expr2]`	Vrai si `[Expr1]` ou `[Expr2]` sont vraies
`[Expr1]` **`||`** `[Expr2]`	idem
`[Expr1]` **`xor`** `[Expr2]`	Vrai si `[Expr1]` ou `[Expr2]` sont vraies mais pas les deux
`!` `[Expr1]`	Vrai si `[Expr1]` est non vraie

Les opérateurs sur bits

`$a & $b`	ET binaire
`$a | $b`	OU binaire
`$a ^ $b`	XOR binaire
`~ $a`	Inversion bit à bit
`$a << $b`	`$a` décalé à gauche de `$b` rangs
`$a >> $b`	`$a` décalé à droite de `$b` rangs

Précédence des opérateurs

Opérateurs
new
[
++ --
! ~ - (int) (float) (string) (array) (object) @
* / %
+ - .
<< >>
< <= > >=
== != === !==
&
…

Précédence des opérateurs

Opérateurs
…
^
|
&&
||
? :
= += -= *= /= .= %= &= |= ^= <<= >>=
and
xor
or

En cas de doute,
utilisez les parenthèses ;-)

Structures de contrôle

Structure de contrôle Si…Alors…Sinon…

```
if (condition)
{
 /* Bloc d'instructions exécuté si la
 condition est vraie */
}
else
{
 /* Bloc d'instructions exécuté si la
 condition est fausse */
}
```

Structure de contrôle Tant que… faire…

```
while (condition)
{
 /* Bloc d'instructions répété tant que la
 condition est vraie */
}
```

```
do {
 /* Bloc d'instructions exécuté une fois
 puis répété tant que la condition est
 vraie */
} while (condition) ;
```

Structure de contrôle Tant que… faire…

```
for (avant; condition; fin_chaque_itération)
{ /* Bloc d'instructions répété tant que la
  condition est vraie */
}
```

Équivalent à :

```
avant ;
while (condition)
{ /* Bloc d'instructions répété tant que la
  condition est vraie */
  fin_chaque_itération ;
}
```

Structure de contrôle selon…

```
switch (val)
{
  case v1:
    instructions exécutées si val==v1
  case v2:
  case v3:
    instructions exécutées si val==v2
                              ou si val==v3
                              ou si val==v1
  …
  default:
    instructions dans tous les cas
}
```

- Fonctionne exactement comme une série de **`if/else`** avec comparaison large
- **`val`**, **`v1`**, **`v2`**, … peut être de type **`integer`**, **`float`**, **`string`**, **`NULL`**

L'instruction **break**

Permet de sortir d'une structure de contrôle

```
switch (val)
{
  case v1:
     instructions exécutées si val==v1
     break ; /* Sortie du switch si val==v1 */
  case v2:
  case v3:
     instructions exécutées si val==v2
                           ou si val==v3
                           ou si val==v1
     break ; /* Sortie du switch si val==v2 ou val==v3*/
  …
  default:
    instructions exécutées dans tous les cas
                        si val!=v1 et val!=v2 et val!=v3
}
```

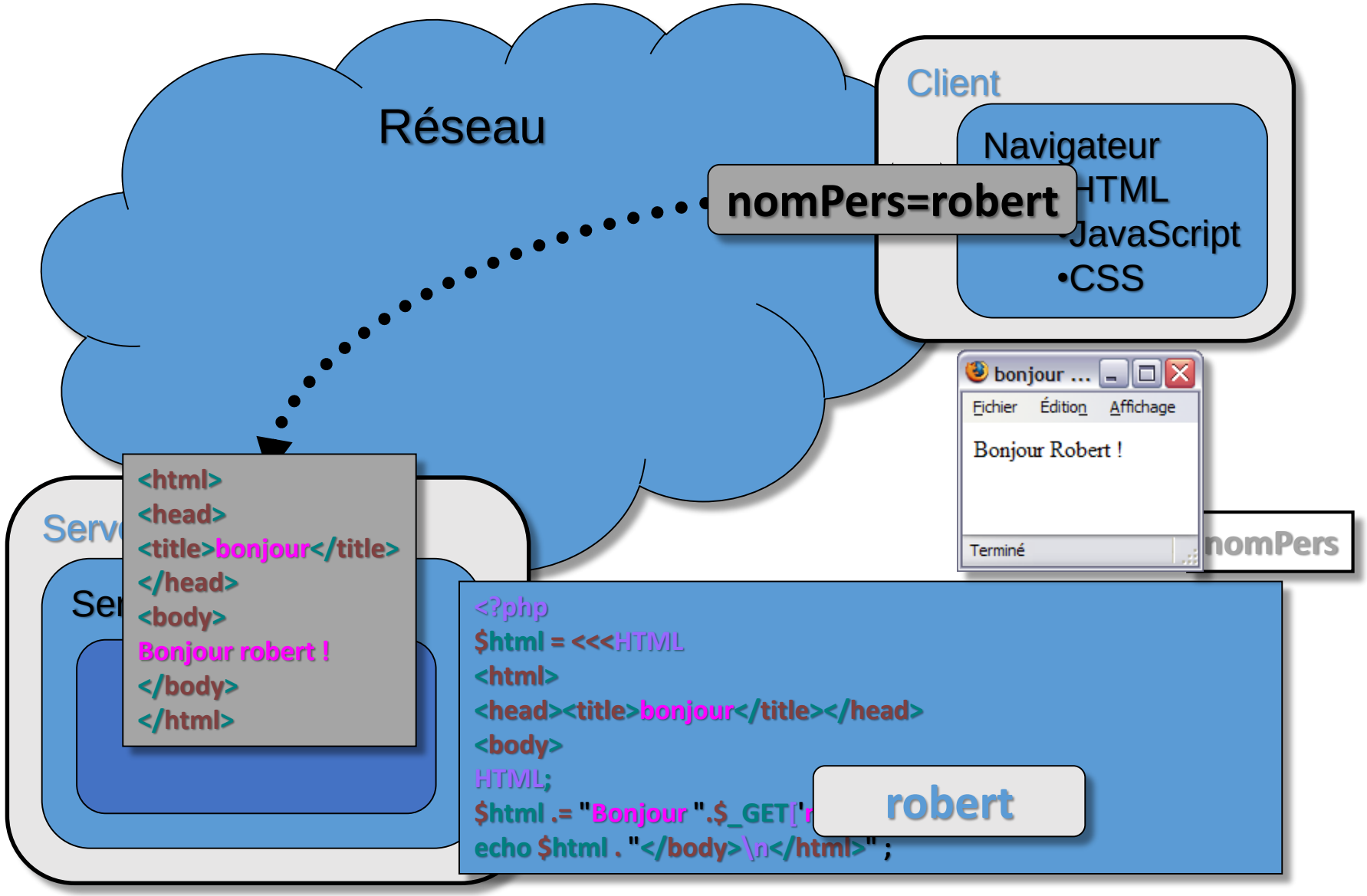
Cas rendus exclusifs si **break** présent pour chaque **case**

Données de formulaires

Traitement des données de formulaires

- PHP permet de traiter les données saisies grâce à un formulaire HTML si le champ **ACTION** du formulaire désigne une page PHP du serveur.
- Après récupération par le serveur Web, les données sont contenues dans l'une des variables superglobales de type tableau associatif **$_GET** ou **$_POST** (ou **$_REQUEST**).
- La valeur peut être trouvée grâce à une clé qui porte le même nom que le champs du formulaire de la page HTML de saisie.

Traitement des données de formulaires



Exemple – Formulaire HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <title>formulaire</title>
</head>
<body>
 <form action="valide1.php" method="get">
   Nom: <input type="text" name="nomPers">
 <input type="submit" value="Envoyer">
 </form>
</body>
</html>
```

Exemple – Traitement en PHP

```php
<?php
$html = <<<HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>bonjour</title>
</head>

<body> HTML;
  if (isset($_GET['nomPers']))
  {
      if (!empty($_GET['nomPers']))
      {
          $html .= "Vous avez saisi '"
                  .$_GET['nomPers']."'\n" ;
      }
      else
          $html .= "Aucune valeur saisie\n";
  }
  else
      $html .= "Utilisation incorrecte\n" ;
echo $html . "</body>\n</html>" ;
```

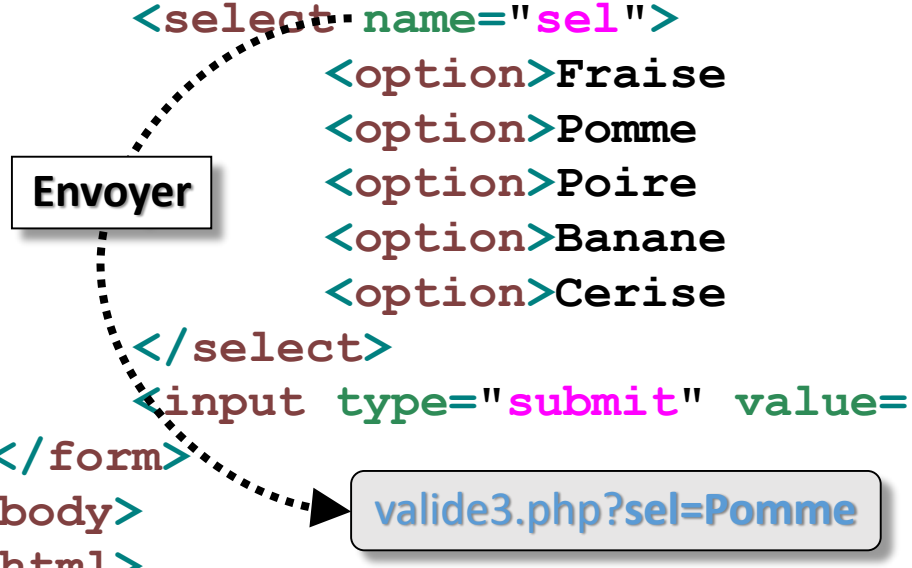
$_GET['nomPers'] est-il défini ?

$_GET['nomPers'] est-il vide ?

Formulaires contenant des champs « SELECT »

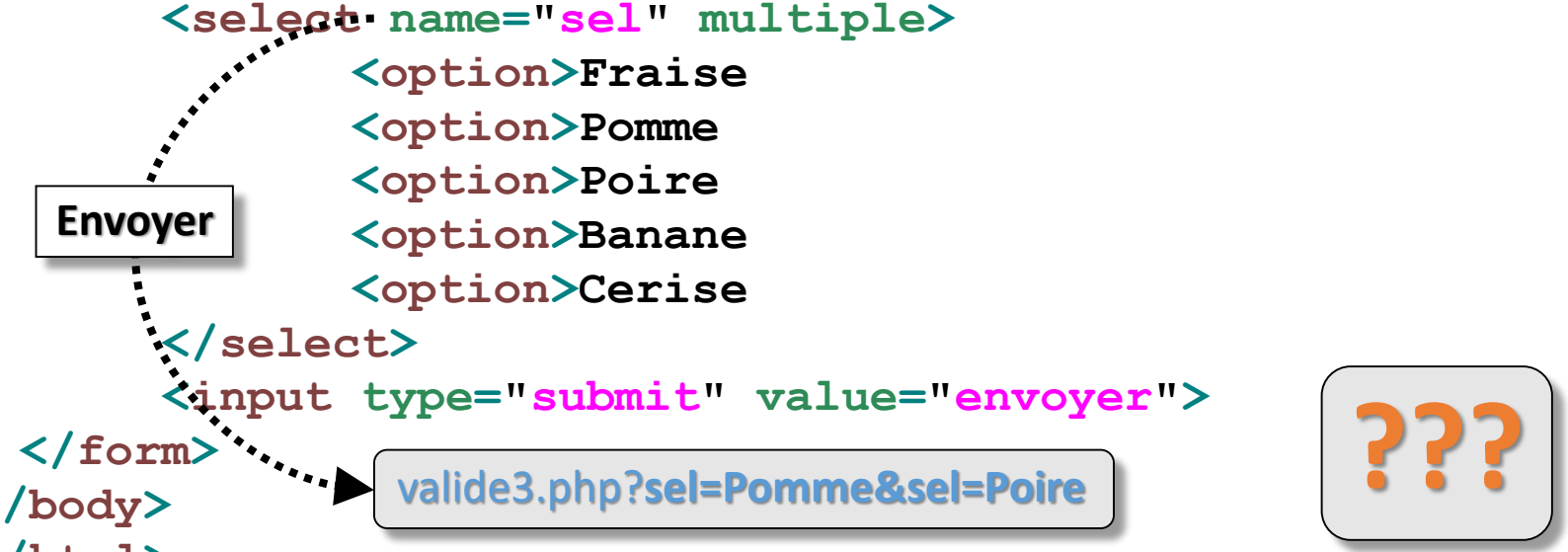
Formulaires contenant des champs « SELECT unique»

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
        Choisissez des fruits: 
        <select name="sel">
              <option>Fraise
              <option>Pomme
              <option>Poire
              <option>Banane
              <option>Cerise
        </select>
        <input type="submit" value="envoyer">
  </form>
</body>
</html>
```



Formulaires contenant des champs « SELECT multiple»

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
        Choisissez des fruits: 
        <select name="sel" multiple>
              <option>Fraise
              <option>Pomme
              <option>Poire
              <option>Banane
              <option>Cerise
        </select>
        <input type="submit" value="envoyer">
  </form>
</body>
</html>
```



Formulaires contenant des champs « SELECT multiple»

```
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
       Choisissez des fruits: 
       <select name="sel[]" multiple>
              <option>Fraise
              <option>Pomme
              <option>Poire
              <option>Banane
              <option>Cerise
       </select>
       <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

Envoyer

valide3.php?sel%5B%5D=Pomme&sel%5B%5D=Poire

valide3.php?sel[]=Pomme&sel[]=Poire

Tableaux

Principes généraux

- Un tableau PHP est en réalité une
- Une carte associe des valeurs à des clés
- Un tableau PHP peut être vu comme :
  - Une série de valeurs (peu importe leur type)
  - Chaque valeur est étiquetée, repérée par une clé (entier ou chaîne)
  - Ils sont dits

Clé	0	1	2	3
Valeur	12	15.2	"42"	NULL

Clé	"début"	12	"a"	"valeur"
Valeur	false	3	16	"bonjour"

Tableaux « classiques » : indexés

- Création / initialisation:

```
$tab1=array(12, "fraise", 2.5) ;

$tab2[] = 12 ;
$tab2[] = "fraise" ;
$tab2[] = 2.5 ;

$tab3[0] = 12 ;
$tab3[1] = "fraise" ;
$tab3[2] = 2.5 ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5

Tableaux associatifs : syntaxe

```
$tab5[ 'un'   ] = 12 ;
$tab5['trois'] = "fraise" ;
$tab5["deux" ] = 2.5 ;
$tab5[  42    ] = "el5" ;
```

Clé	Valeur
"un"	12
"trois"	"fraise"
"deux"	2.5
42	"el5"

```
$tab6 = array(  'un'   => 12,
                'trois' => "fraise",
                "deux"  => 2.5,
                   42   => "el5") ;
```

Structure de contrôle « Pour chaque… »

Des tableaux associatifs ne peuvent être parcourus grâce aux indices des cases contenant les éléments…

```
foreach ($tableau as $element)
{
 /* Bloc d'instructions répété pour chaque
 élément de $tableau */
 /* Chaque élément de $tableau est accessible
 grâce à $element */
}
```

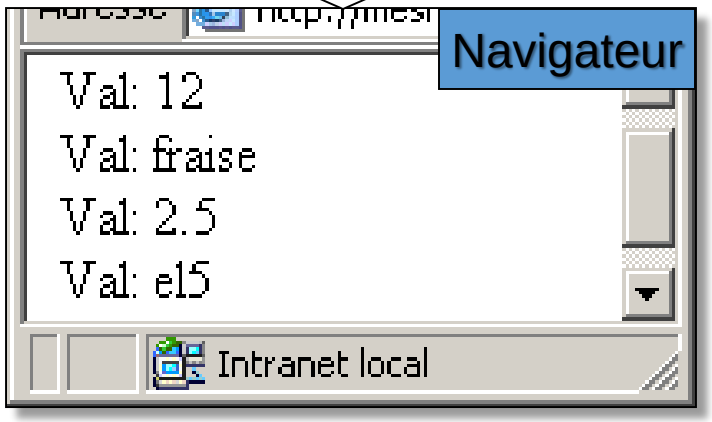
Parcours de tableau : **foreach**

PHP

```
…
$tab4[0] = 12 ;
$tab4[1] = "fraise" ;
$tab4[2] = 2.5 ;
$tab4[5] = "el5" ;
foreach($tab4 as $v)
{
   echo "<p>Val: $v\n";
}
…
```

HTML

```
…
<p>Val:12\n
<p>Val:fraise\n
<p>Val:2.5\n
<p>Val:el5\n
…
```



Tableaux associatifs

- Tableaux dont l'accès aux éléments n'est plus réalisé grâce à un index (0,1, …) mais grâce à une clé de type entier ou chaîne.
- Exemples de clés:

```
$tab['un']          = 12 ;
$tab[205]           = "bonjour" ;
$tab["la valeur"] = 3.0 ;
```

- Création

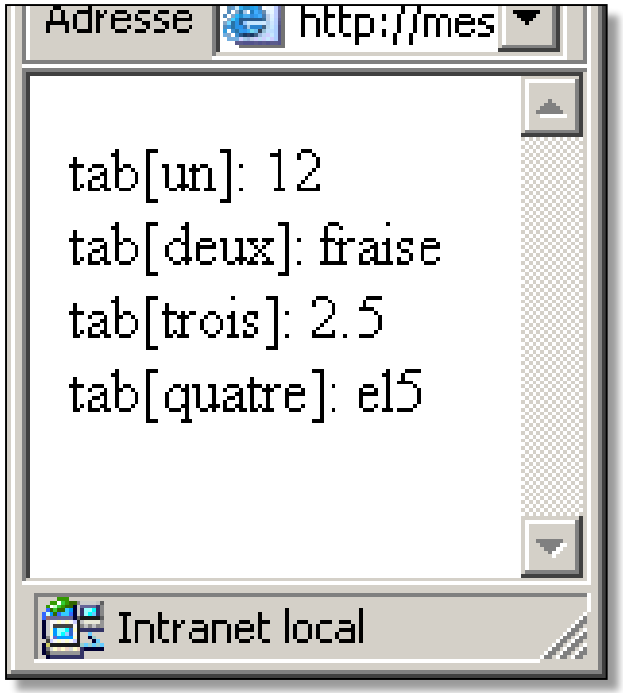
```
$tab = array(cle1 => val1,
             cle2 => val2,
             …);
```

Structure de contrôle « Pour chaque… »

```
foreach($tableau as $cle => $element)
{
 /* Bloc d'instructions répété pour
     chaque élément de $tableau */
 /* Chaque élément de $tableau est
     accessible grâce à $element */
 /* La clé d'accès à chaque élément est
     donnée par $cle */
}
```

Parcours de tableau

```php
<?php
$html = ''
<html>
  <head><title>foreach clé</title>
  </head>
<body>
'';
$tab6 = array('un'      => 12,
              'deux'    => "fraise",
              "trois"   => 2.5,
              "quatre" => "el5") ;
foreach ($tab6 as $cle => $val)
{
  $html .= "<p>tab[$cle]: $val\n" ;
}
echo $html . "</body>\n</html>" ;
```

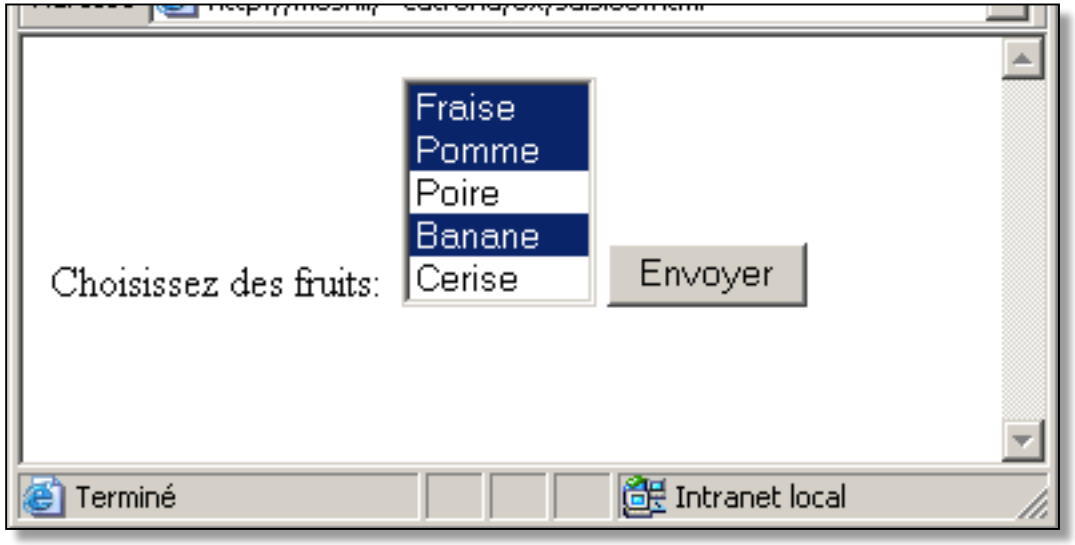


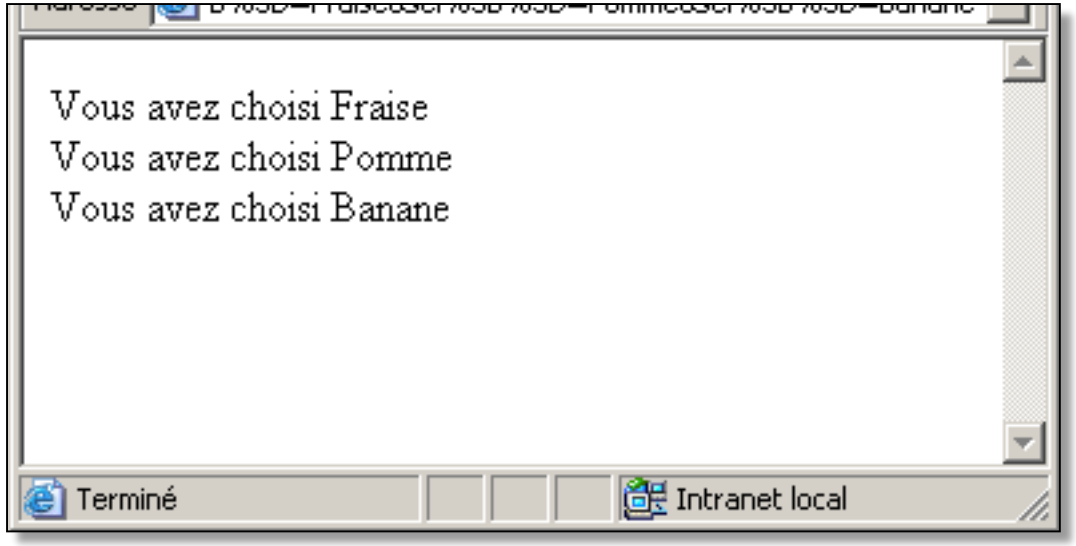
Traitement des données des champs « SELECT »

```php
<?php
$html = <<<HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Liste de fruits</title>
</head>
<body>
HTML;
  if (isset($_GET['sel']) && !empty($_GET['sel']))
  {/* La variable $_GET['sel'] est définie
      et elle n'est pas vide */
       foreach($_GET['sel'] as $fruit)
           $html .= "<p>Vous avez choisi $fruit\n" ;
  }
  else
      $html .= "<p>Vous n'avez pas choisi de fruit\n" ;
echo $html . "</body>\n</html>" ;
```

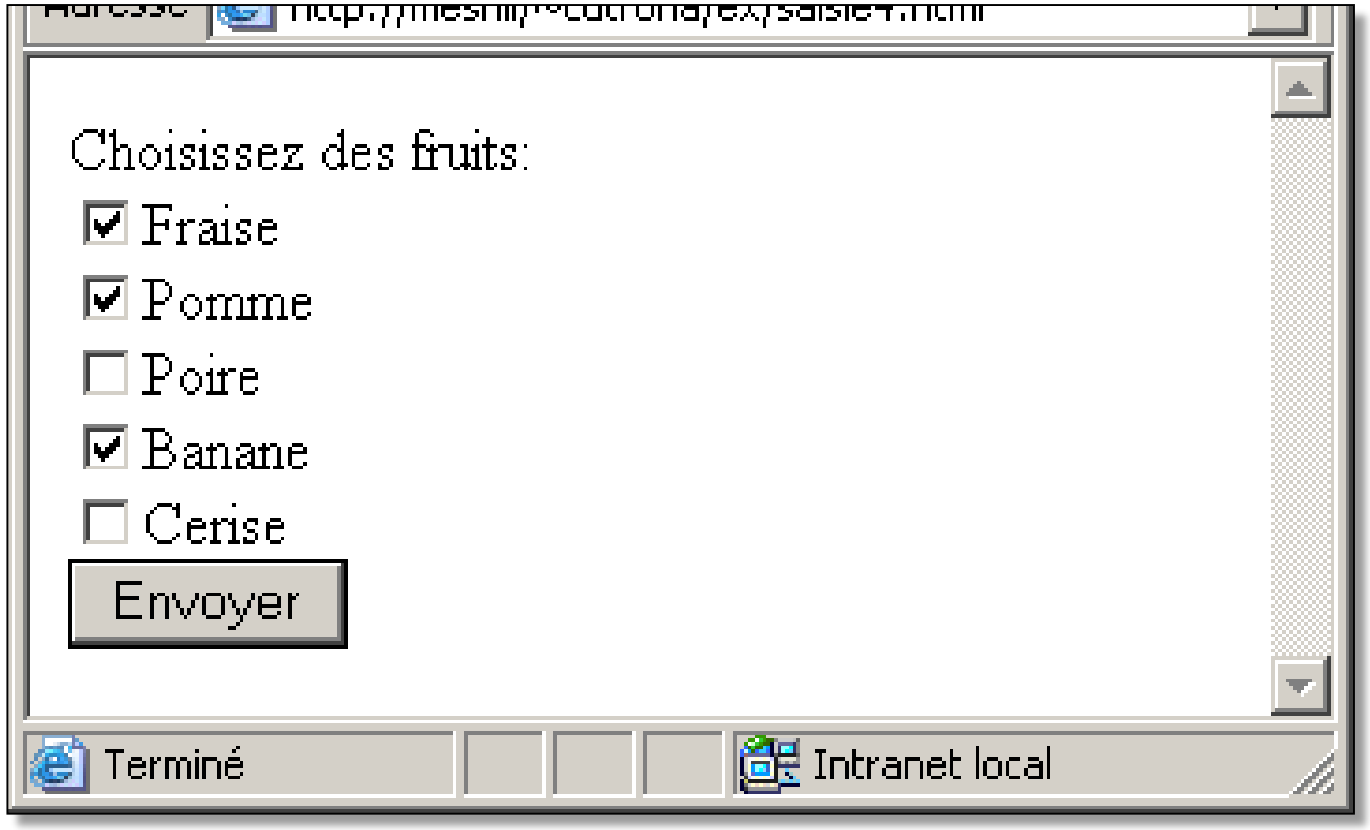
$_GET['sel'] est un tableau

Résultat


Fraise
Pomme
Poire
Banane
Cerise
Choisissez des fruits:
Envoyer
Terminé
Intranet local


Vous avez choisi Fraise
Vous avez choisi Pomme
Vous avez choisi Banane
Terminé
Intranet local

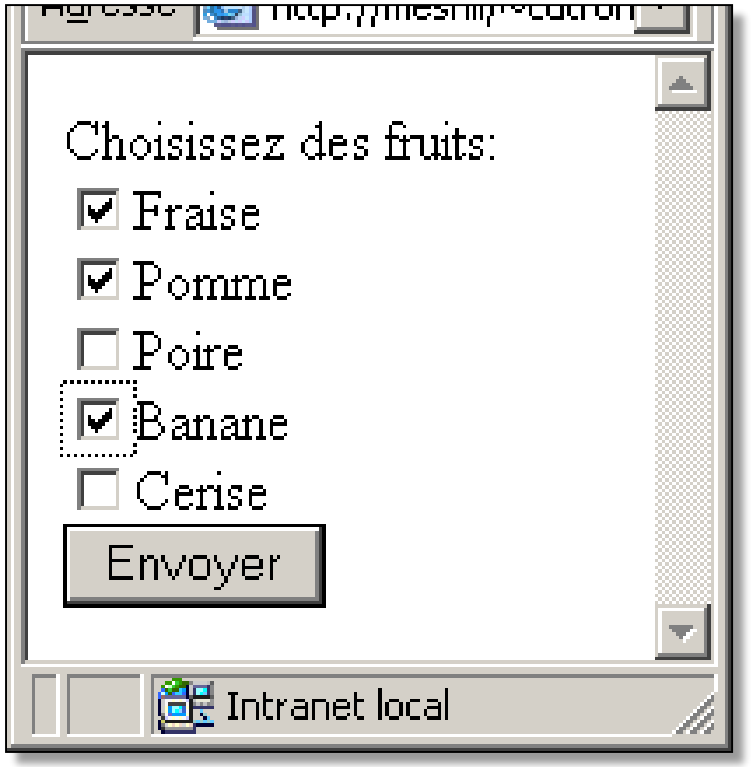
Formulaires contenant des champs « CHECKBOX »

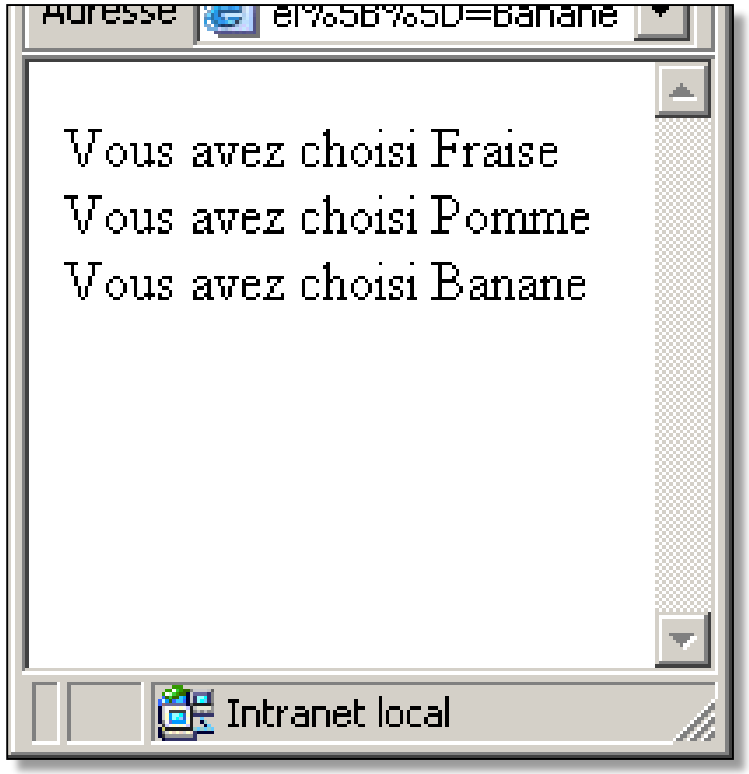


Formulaires contenant des champs « CHECKBOX »

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form name="formu" action="valide3.php" method="get">
  Choisissez des fruits :<br>
  <p><input type="checkbox" name="sel[]" value="Fraise">Fraise
  <p><input type="checkbox" name="sel[]" value="Pomme" >Pomme
  <p><input type="checkbox" name="sel[]" value="Poire" >Poire
  <p><input type="checkbox" name="sel[]" value="Banane">Banane
  <p><input type="checkbox" name="sel[]" value="Cerise">Cerise
  <p><input type="submit" value="Envoyer">
  </form>
</body>
</html>
```

Résultat

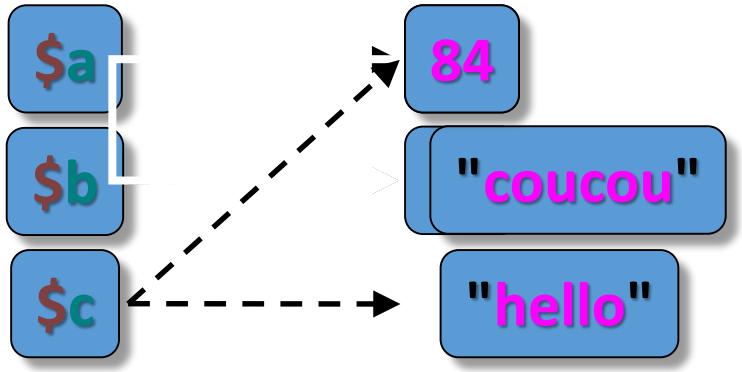

Choisissez des fruits:
Fraise
Pomme
Poire
Banane
Cerise
Envoyer
Intranet local


Vous avez choisi Fraise
Vous avez choisi Pomme
Vous avez choisi Banane
Intranet local

Références

```
$a = 12 ;
$b = $a ;
$c = &$a ;
$b = "coucou" ;
$c = 84 ;
echo "$a : $a" ;
echo "$b : $b" ;
echo "$c : $c" ;
unset($c) ;
$c = "hello" ;
```

```
$a : 84
$b : coucou
$c : 84
```



Fonctions

Fonctions utilisateur

- Description d'une fonctionnalité dépendant éventuellement de paramètres et retournant éventuellement un résultat
- Définition

```
function moyenne($a,$b)
{
    return ($a+$b)/2. ;
}
```

- Utilisation

```
$resultat = moyenne(2,4) ;
echo $resultat ; // vaut 3
```

Fonctions utilisateur

- Valeur de retour

```
function moyenne($a,$b)
{   …   }
```

Typage faible de PHP :
Aucune information

- Arguments

```
function moyenne( $a, $b)
{   …   }
```

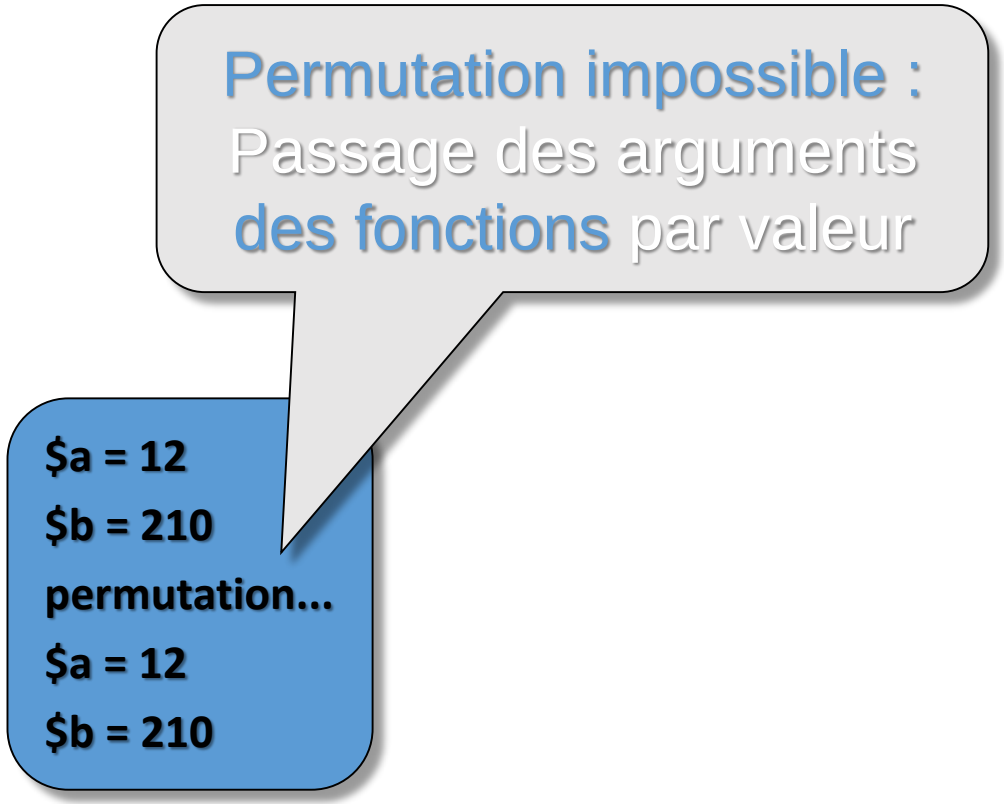
- Variables définies dans une f…on

➔ Visibles et acc…tion

Typage faible de PHP :
Aucune information

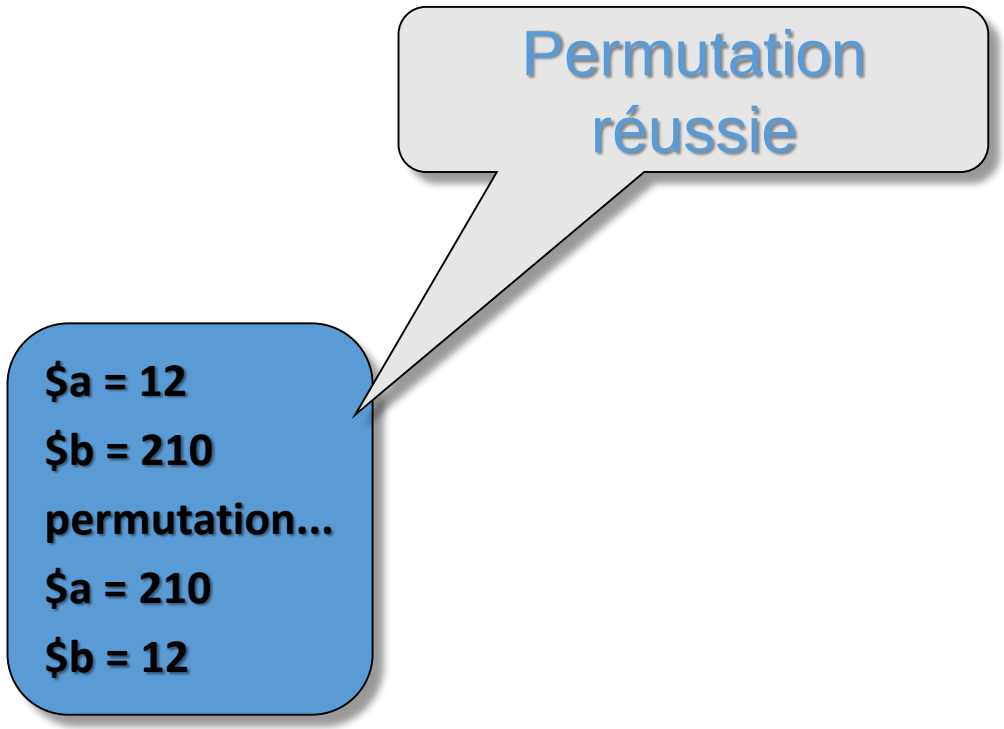
Mode de passage des arguments (types natifs)

```php
<?php
function permutation($x, $y) {
  echo "permutation..." ;
  $t = $x ;
  $x = $y ;
  $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
```



Mode de passage des arguments (types natifs)

```php
<?php
function permutation(&$x, &$y) {
  echo "permutation..." ;
  $t = $x ;
  $x = $y ;
  $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
```



Arguments par défaut des fonctions

- Valeur par défaut d'un argument s'il n'a pas été défini lors de l'appel de la fonction

```
function bonjour($nom="inconnu")
{ echo "Bonjour cher $nom" ; }
```

- Utilisation

```
bonjour() ;
```

Bonjour cher inconnu

```
bonjour("Marcel") ;
```

Bonjour cher Marcel

Définition de fonctions fréquemment utilisées

- Certaines fonctions sont utilisées dans plusieurs scripts PHP
- Comment faire pour ne pas les définir dans chacune des pages ?
- Utilisation de :
  - `include("fichier") ;`
  - `require("fichier") ;`
  - `include_once("fichier") ;`
  - `require_once("fichier") ;`
- Permet d'inclure le contenu de ***fichier*** dans le script courant

include et require

Fichier mafonction.php

```
<?
function mafonction($arg)
{
  …
}
```

Problème avec include :
- produit un **warning**
- le script continue

Problème avec require :
- produit un **fatal error**
- le script s'arrête

Fichier utilisation1.php

```
…
require("mafonction.php")
mafonction(true) ;
…
```

Fichier utilisation2.php

```
…
include("mafonction.php")
…
$var=false ;
mafonction($var) ;
…
```

Fichier utilisation3.php

```
…
require("mafonction.php")
…
```

Gestion des erreurs

- Dans certains cas, il n'est ni possible ni utile de poursuivre l'exécution du code PHP (variables non définies, valeurs erronées, échec de connexion, …)
- Arrêt brutal de l'exécution du code:
  - **`die`**(*`message`*)
  - **`exit`**(*`message`*)

  Envoie *`message`* au navigateur et termine l'exécution du script courant

Gestion des erreurs – (Mauvais) Exemple

PHP

```
<?php
$html = <<<HTML
<html>
<head>
<title>die-exit</title>
</head>
<body>
HTML;
if (!isset($val)) {
  die($html."problème val") ;
  /* Au delà de ce point,
     fin du script */
}
$html .= <<<HTML
Choix: $val
</body>
</html>
```

HTML

```
<html>
<head>
<title>die-exit</title>
</head>
<body>
problème val
```

HTML non valide…

Navigateur

problème val

Gestion de l'affichage des erreurs

- `int err` … `int`

Ancien niveau d'erreur

Sur un serveur en production, **toute erreur affichée** donne des indices sur les scripts et **rend le site vulnérable**

php.ini

display_errors boolean

Constante
E_ERROR
E_WARNING
E_PARSE
E_NOTICE
E_CORE_ERROR
E_CORE_WARNING
E_COMPILE_ERROR
E_COMPILE_WARNING
E_USER_ERROR
E_USER_WARNING
E_USER_NOTICE
E_ALL
E_STRICT

Débogage

Opérateur de contrôle d'erreur en phase de développement

Fichier absent

```
$v = file("dummy.txt")
    or die("Problème de lecture") ;
```

Warning: file(dummy.txt): failed to open stream: No such file or directory in **dummy.php** on line **68**
Problème de lecture

```
$v = @file("dummy.txt")
  or die("Problème de lecture") ;
```

Problème de lecture