Langages de l'internet

Le web dynamique

Bruno Mascret

CPE Lyon
Techniques et langages de l'Internet

Plan

Introduction
Auteurs

Cours : **Bruno Mascret, Oscar Figueiredo, David Odin, Geoffroy Charollais**

TPs : **Bruno Mascret**

Intervenants : **Bruno Mascret, Mohamed Sallami**

Le Web dynamique

- Ajout de code exécutable à une page web

Deux possibilités :

- Exécution côté serveur
  - CGI
  - Server APIs (Apache modules, ISAPI)
  - ASP, PHP, JSP
  - Serveurs d'application (Zope, *groupware)
- Exécution côté client
  - Applets Java
  - ECMAscript, Javascript
  - JScript,
  - VBScript,
  - etc.

Le Web dynamique

- Ajout de code exécutable à une page web

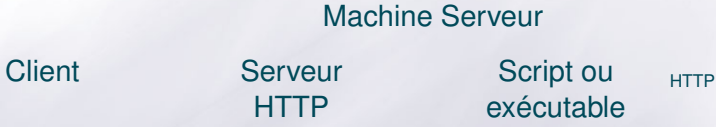
Deux possibilités :

- Exécution côté serveur
  - CGI
  - Server APIs (Apache modules, ISAPI)
  - ASP, PHP, JSP
  - Serveurs d'application (Zope, *groupware)
- Exécution côté client
  - Applets Java
  - ECMAscript, Javascript
  - JScript,
  - VBScript,
  - etc.

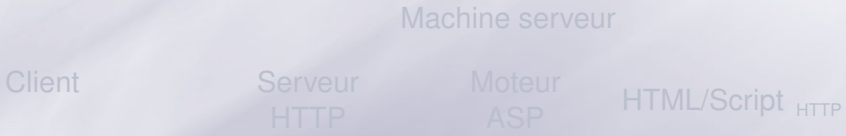
Plan

Modes d'exécution côté serveur

- CGI : Common Gateway Interface

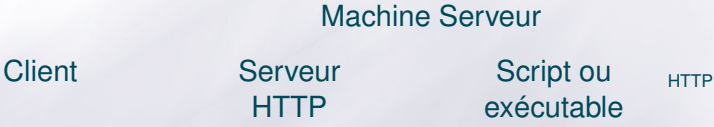


- Active Server Pages (ASP), PHP

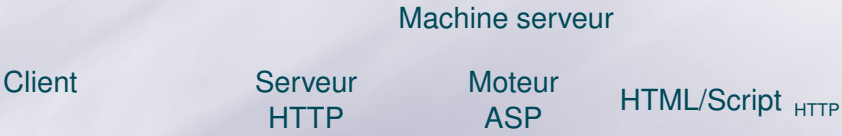


Modes d'exécution côté serveur

- CGI : Common Gateway Interface

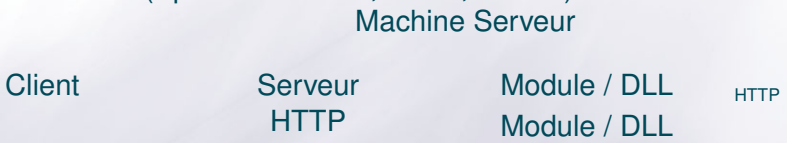


- Active Server Pages (ASP), PHP



Modes d'exécutions côté serveur (2)

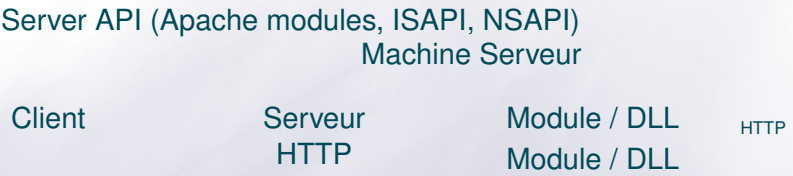
- Server API (Apache modules, ISAPI, NSAPI)



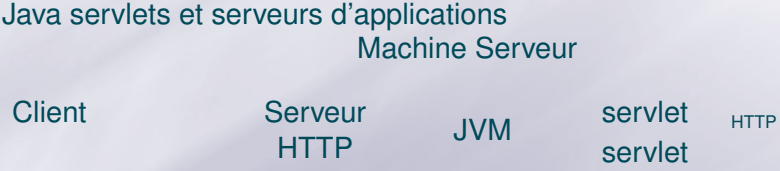
- Java servlets et serveurs d'applications

Modes d'exécutions côté serveur (2)

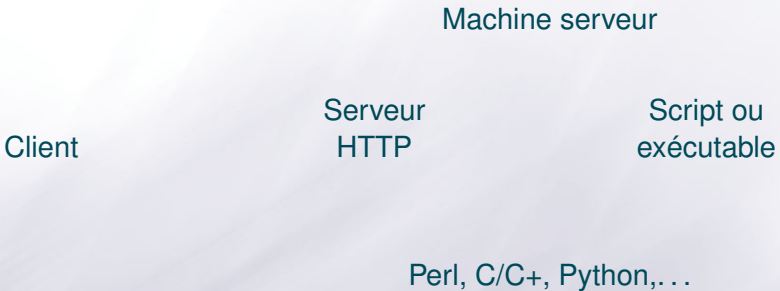
- Server API (Apache modules, ISAPI, NSAPI)



- Java servlets et serveurs d'applications



CGI : Common Gateway Interface



- CGI spécifie :
  - Que le serveur HTTP et le processus de traitement communiquesnt via `stdin`/`stdout`
  - Un certain nombre de variables d'environnement passées par le serveur HTTP au processus de traitement

Variables d'environnement CGI

SERVER_SOFTWARE	Nom et version du logiciel serveur HTTP
SERVER_NAME	Nom d'hôte ou adresse IP du serveur
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.0
SERVER_PORT	Port TCP du serveur Web
REQUEST_METHOD	`GET`, `HEAD` ou `POST`
SCRIPT_NAME	Nom virtuel du script invoqué
QUERY_STRING	La ligne d'arguments de la requête (sans décodage)
REMOTE_HOST	Le nom d'hôte du client
REMOTE_ADDR	L'adresse IP du client
AUTH_TYPE	Méthode d'authentification (si applicable)
REMOTE_USER	Utilisateur distant (si disponible)
REMOTE_IDENT	Identification de l'utilisateur distant par le serveur
CONTENT_TYPE	Type de données reçues (requête `POST`)
CONTENT_LENGTH	Longueur des données reçues (requête `POST`)
HTTP_*	Lignes d'entête HTTP reçues du client

CGI : document renvoyé

- Par défaut le serveur renvoie le document généré tel quel.
- Il est donc indispensable d'inclure au moins l'entête de type de contenu

```
Content-type: text/html

```

Ligne vide indispensable

```
<html>
  <head>
    <title>Ok</title>
  </head>
  <body>
    <p>Merci pour tout</p>
  </body>
</html>
```

CGI : document renvoyé

- Par défaut le serveur renvoie le document généré tel quel.
- Il est donc indispensable d'inclure au moins l'entête de type de contenu

```
Content-type: text/html
```

Ligne vide indispensable

```
<html>
  <head>
    <title>Ok</title>
  </head>
  <body>
    <p>Merci pour tout</p>
  </body>
</html>
```

Critique de CGI

- Mécanisme simple et universel
- Indépendance du langage du processus de traitement
- Isolation du processus de traitement du serveur web
- Le lancement d'un processus externe à chaque requête est très lourd (ce qui peut se contourner en utilisant des modules du serveur web comme mod_perl ou FastCGI).

PHP

- Langage de scriptage côté serveur
- Fonctionnellement analogue à ASP (Microsoft)
- Multi plates-formes (même si plutôt lié au départ à Linux et Apache)
- Pilier du quatuor Linux-Apache-MySQL-PHP (LAMP)
- Syntaxe mélant C et Perl
- Documentation très fournie (et en français !) sur http :// www.php.net/manual/fr/
- Très nombreuses bibliothèques
  - Connexion bases de données
  - Paiement “sécurisé”
  - XML
  - Réseau
  - Accès OS
  - etc. (185 bibliothèques aujourd’hui)

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.
PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)
Les apports de PHP 5 :

- plus rapide
- nouvelles fonctionnalités (tableau, base de données InterBase, internationnalisation, flux, date, etc.)
- intégration de SQLite
- nouveau modèle objet (utilisation de références)
- quelques (rares) incompatibilités

PHP : généralités sur le langage

- Syntaxe
  - Sensible à la case
  - instructions terminées par des ;
  - Commentaires C (/* */), C++ (//) ou shell (#)
  - Variables préfixées par $
  - Variables non typées
  - Déclaration automatique à l'affectation
- Types de données

[arrows=->,levelsep=2cm,nodesep=1mm]Scalaire

booléen	entier	flottant	chaîne
TRUE	16	0.017	"test"
False	020	17.0e-3	'un "a"'
	0x10		'$variable'
			"$valeur"

[arrows=->,levelsep=2cm,nodesep=1mm]composé

objet	tableau	indexé	associatif
$inet->web=42;		$a[0]=12;	$j["jan"]=31

PHP : généralités sur le langage

- Syntaxe
  - Sensible à la case
  - instructions terminées par des ;
  - Commentaires C (/* */), C++ (//) ou shell (#)
  - Variables préfixées par $
  - Variables non typées
  - Déclaration automatique à l'affectation
- Types de données

[arrows=->,levelsep=2cm,nodesep=1mm]Scalaire

booléen	entier	flottant	chaîne
TRUE	16	0.017	"test"
False	020	17.0e-3	'un "a"'
	0x10		'$variable'
			"$valeur"

[arrows=->,levelsep=2cm,nodesep=1mm]composé

objet	tableau	indexé	associatif
$inet->web=42;		$a[0]=12;	$j["jan"]=31

PHP : généralités sur le langage

- Syntaxe
  - Sensible à la case
  - instructions terminées par des ;
  - Commentaires C (/* */), C++ (//) ou shell (#)
  - Variables préfixées par $
  - Variables non typées
  - Déclaration automatique à l'affectation
- Types de données

[arrows=->,levelsep=2cm,nodesep=1mm]Scalaire

booléen	entier	flottant	chaîne
TRUE	16	0.017	"test"
False	020	17.0e-3	'un "a"'
	0x10		'$variable'
			"$valeur"

[arrows=->,levelsep=2cm,nodesep=1mm]composé

objet	tableau	indexé	associatif
$inet->web=42;		$a[0]=12;	$j["jan"]=31

PHP : généralités sur le langage

- Syntaxe
  - Sensible à la case
  - instructions terminées par des ;
  - Commentaires C (/* */), C++ (//) ou shell (#)
  - Variables préfixées par $
  - Variables non typées
  - Déclaration automatique à l'affectation
- Types de données

[arrows=->,levelsep=2cm,nodesep=1mm]Scalaire

booléen	entier	flottant	chaîne
TRUE	16	0.017	"test"
False	020	17.0e-3	'un "a"'
	0x10		'$variable'
			"$valeur"

[arrows=->,levelsep=2cm,nodesep=1mm]composé

objet	tableau	indexé	associatif
$inet->web=12;		$a[0]=12;	$j["jan"]=31

PHP : généralités sur le langage

- Syntaxe
  - Sensible à la case
  - instructions terminées par des ;
  - Commentaires C (/* */), C++ (//) ou shell (#)
  - Variables préfixées par $
  - Variables non typées
  - Déclaration automatique à l'affectation
- Types de données

[arrows=->,levelsep=2cm,nodesep=1mm]Scalaire

booléen: `TRUE` `False`

entier: `16` `020` `0x10`

flottant: `0.017` `17.0e-3`

chaîne: `"test"` `'un "a"'` `'$variable'` `"$valeur"`

[arrows=->,levelsep=2cm,nodesep=1mm]composé

objet: `$inst->mb=12;`

tableau

indexé: `$a[0]=12;`

associatif: `$j["jan"]=31`

Opérations de base

- Opérations classiques sur les entiers et les flottants,
- Les opérateurs sont en général les mêmes qu'en C,
- Opérations simples sur les chaînes de caractères (concaténation par .), transformées en un entier en cas d'opérations arithmétiques,
- Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- Opérateur de référence : & : `$a = &$b;`

Opérations de base

- Opérations classiques sur les entiers et les flottants,
- Les opérateurs sont en général les mêmes qu'en C,
- Opérations simples sur les chaînes de caractères (concaténation par .), transformées en un entier en cas d'opérations arithmétiques,
- Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- Opérateur de référence : & : `$a = &$b;`

Opérations de base

- Opérations classiques sur les entiers et les flottants,
- Les opérateurs sont en général les mêmes qu'en C,
- Opérations simples sur les chaînes de caractères (concaténation par .), transformées en un entier en cas d'opérations arithmétiques,
- Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- Opérateur de référence : & : `$a = &$b;`

Opérations de base

- Opérations classiques sur les entiers et les flottants,
- Les opérateurs sont en général les mêmes qu'en C,
- Opérations simples sur les chaînes de caractères (concaténation par .), transformées en un entier en cas d'opérations arithmétiques,
- Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- Opérateur de référence : & : `$a = &$b;`

Opérations de base

- Opérations classiques sur les entiers et les flottants,
- Les opérateurs sont en général les mêmes qu'en C,
- Opérations simples sur les chaînes de caractères (concaténation par .), transformées en un entier en cas d'opérations arithmétiques,
- Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- Opérateur de référence : & : `$a = &$b;`

PHP : classes

Exemple

```
class test
{
  var $str = "Hello world!";
  function init($str)
  {
    $this->str = $str;
  }
}
$class = new test;
print $class->str;

$class->init("Coucou");
print $class->str;
```

Héritage : par le mot clef `extends`

```
class A { var $str = "Hello world!"; }

class B extends A { var $bla = 'foo'; }

$objet = new B;
print $objet->str;
```

Pas d'héritage multiple, ni de visibilité, ni de destructeurs, mais opérateur de classes : `::`

PHP : classes

Exemple

```
class test
{
  var $str = "Hello world!";
  function init($str)
  {
    $this->str = $str;
  }
}
$class = new test;
print $class->str;

$class->init("Coucou");
print $class->str;
```

Héritage : par le mot clef `extends`

```
class A { var $str = "Hello world!"; }

class B extends A { var $bla = 'foo'; }

$objet = new B;
print $objet->str;
```

Pas d'héritage multiple, ni de visibilité, ni de destructeurs, mais opérateur de classes : ::

PHP : classes

- Exemple

```
class test
{
  var $str = "Hello world!";
  function init($str)
  {
    $this->str = $str;
  }
}
$class = new test;
print $class->str;

$class->init("Coucou");
print $class->str;
```

- Héritage : par le mot clef `extends`

```
class A { var $str = "Hello world!"; }

class B extends A { var $bla = 'foo'; }

$objet = new B;
print $objet->str;
```

- Pas d'héritage multiple, ni de visibilité, ni de destructeurs, mais opérateur de classes : :

Constructeurs

Un constructeur est simplement une fonction (méthode) qui a le même nom que la classe.

```
<?php
class A {
   function A() {
    echo "Je suis le constructeur de A.<br />\n";
   }
   function B() {
    echo "Je suis une fonction standard appelee B dans la classe A.<br />\n";
    echo "Je ne suis pas le constructeur de A.<br />\n";
   }
}

class B extends A {
}
// Cette syntaxe va appeler B() comme constructeur.
$b = new B;
?>
```

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- Constructeurs et destructeurs (__construct et __destruct)
- Visibilité (public, private, protected)
- Méthodes statiques
- Constantes de classes
- Classes abstraites
- Interfaces
- Méthodes finales
- Clonage
- Introspection
- ...

PHP : structures de contrôles

```php
if ($hue==1) {
  $color = "blue";
} elseif ($hue==2) {
  $color = "green";
} else {
  $color = "unknown";
}
```

```php
if ($hue==1):
  $color = "blue";
elseif ($hue==2):
  $color = "green";
else:
  $color = "unknown";
endif;
```

```php
while ($i--) {
  print $i;
}
```

```php
while ($i--):
  print $i;
endwhile;
```

```php
do {
  print $i;
} while ($i < 12);
```

```php
do {
  print $i;
} while ($i < 12);
```

```php
for ($i=0; $i<10; $i++) {
  print $i;
}
```

```php
for ($i=0; $i<10; $i++):
  print $i;
endfor;
```

```php
$tableau = array('05' => "abc", '35' => "def");
foreach ($r as $clef => $valeur)
  {
    print $clef . '␣' . $valeur . '<br␣/>';
  }
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello !";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello !";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello ";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello !";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello !";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note
```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, include_once et require_once
- Variables globales non visibles des fonctions, mot clé `global`

```
function test() {
  global $var;
  echo $var;
}
$var = "Hello world";
test();
```

=> Hello world

- Certaines variables sont *superglobales*
- Mot clef `static` comme en C.
- Passage de paramètres par valeur ou par référence

```
function sqr(x) {
  $x = $x * $x;
  return $x;
}
```

```
$x = 2;
sqr($x);
print "$x<br />";
sqr(&$x);
print $x;
```

=> 2 4

- Variables dynamiques

```
$foo = "hello !";
$bar = "foo";
print $$bar;
```

=> hello !

- Constantes

```
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; // affiche "Constante" et une note.
```

PHP : particularités (2/2)

- Les fonctions peuvent être appelées par leur nom :

```
function salut()
{
  echo 'bonjour';
}
$a = 'salut';
$a(); // affiche bonjour
```

- Les arguments de fonctions peuvent être par valeur, par référence et/ou avec une valeur par défaut (comme en C++)

PHP : particularités (2/2)

- Les fonctions peuvent être appelées par leur nom :

```
function salut()
{
  echo 'bonjour';
}
$a = 'salut';
$a(); // affiche bonjour
```

- Les arguments de fonctions peuvent être par valeur, par référence et/ou avec une valeur par défaut (comme en C++)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- Initialisez **toutes** vos variables !
- Vérifiez le contenu des variables fournies par l'utilisateurs (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- Initialisez **toutes** vos variables !
- Vérifiez le contenu des variables fournies par l'utilisateurs (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- Initialisez **toutes** vos variables !
- Vérifiez le contenu des variables fournies par l'utilisateurs (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- Initialisez **toutes** vos variables !
- Vérifiez le contenu des variables fournies par l'utilisateurs (chemin, injections sql, etc.)

Intégration PHP/HTML

- Insertion de code PHP dans HTML

```
<html>
  <head>
    <title>Configuration PHP</title>
  </head>

  <body>
    <h1>Configuration PHP</h1>

    <?php phpinfo(); ?>
  </body>
</html>
```

```
<html>
  <head>
    <title>Configuration PHP</title>
  </head>

  <body>
    <h1>Configuration PHP</h1>

    <? phpinfo(); ?>

  </body>
</html>
```

- Variables prédéfinies
  - Variables génériques (similaires à CGI : $HTTP_USER_AGENT, $REMOTE_ADDR, ...
  - Arguments de requête (POST, GET et COOKIES) dans des tableaux globaux :

```
$_GET['mavariable'],
$_POST['mavariable'],
$_COOKIE['moncookie']
```

PHP : gestion de formulaire

```
...

<h1>Formulaire PHP</h1>

<?php if (empty($_POST['nom']) || empty($_POST['age'])) :
        if ($_POST['submit']=="Envoyer")
           echo "<p>Formulaire incomplet, merci de completer:</p>";
?>
  <form action="<?php echo $SCRIPT_NAME ?>" method="post">
    <p>
      Votre nom: <input type="text" name="nom"
                        value="<?php echo $_POST['nom'] ?>" /><br />
      Votre age: <input type="text" name="age"
                        value="<?php echo $_POST['age'] ?>" /><br />
      <input type="submit" name="submit" value="Envoyer" />
    </p>
  </form>
<?php else : ?>
  <p>Merci pour votre reponse.</p>
  <p>Nom: <?php echo $_POST['nom']; ?></p>
  <p>Age: <?php echo $_POST['age']; ?></p>
<?php endif; ?>
```

Plan

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.nom="toto"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.nom="toto"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.nom="toto"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.name="point"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.name="point"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.name="point"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : caractéristiques générales

- Syntaxe dérivée de Java et de C++
- Variables non typées
- Seules les variables locales **doivent** être déclarées (`var`)
- Types de données
  - scalaire
    - nombre (`12`, `13.34`, `6.02e23`)
    - booléen (`true`, `false`)
    - chaîne (`"test"`, `"un 'a'"`)
    - regexp (`/foo/i`)
  - composé
    - tableau (`a[0]=12;b=[1,true]`)
    - objet (`o.x=12; o.name="point"`)
- Tous les types de données sont associés à un type objet fondamental : `Number`, `Boolean`, `String`, `Array`, `Object`, `RegExp`, `Dates`, `Function`, `Math`

Javascript : objets et tableaux

- **Objet** : collection de données et fonctions (propriétés) indexée par une clef symbolique

```
var o = new Object();
var now = new Date();
var pt = {x:212, y:38}
function pt_translate (dx, dy)
{
  this.x += dx
  this.y += dy;
}
pt.translate = pt_translate
```

- **Tableau** : collection de données indexée par des entiers 0...n

```
var a=new Array()
var b=new Array(10)
var c=new Array(1,2,3)
var d=[1,2,3]
var e=[1,true,[1,2],{x:1,y:2},"Hello"]
```

Javascript : chaînes de caractères

Opérateurs

```
var str1 = "Javascript"
var str2 = "en 10 lecons"
var str = str1 + " " + str2

str += " faciles !"

if (reponse == "oui")
  ...

if (mot1 < mot2)
  alert (mot1 + " est avant " + mot2 + " dans le dictionnaire");

if (reponse != pass)
  alert ("Mauvais mot de passe")
```

Toute chaîne est un objet String

```
if (pass.length < 8)
  alert("Mot de passe trop court")

debut = reponse.substring(0,3)
```

Javascript : constructions originales

- `for...in` : listage des propriétés d'un objet

```
function dump_props (obj, objName)
{
  var result = ""
  for (var i in obj)
    {
      result += objName + "." + i + " = " + obj[i] + "<br />"
    }
  retult += "<br />"
  return result
}
```

- `var` : déclaration de variables

```
var num_hits = 0, cust_no = 0
```

- `with` : sélection de portée sur un objet

```
var a, x, y, r = 10
with (Math)
  {
    a = PI * r * r
    x = r * cos (PI)
    y = r * sin (PI / 2)
  }
```

Intégration Javascript / HTML

- L'élément `script` : intégration du code Javascript

```
<script type="text/javascript" src="http://host/myscript.js" />

<script type="text/javascript">
<!--
   function alert(msg)
     {
       alert("ALERTE : " + msg)
     }
//-->
</script>
```

- Exécution du code Javascript
  - Lors de l'affichage du document pour les balises `script`
  - Sur événement grace aux attributs HTML d'événements

Attributs HTML d'événements

`onabort`	`img`	Interruption de chargement
`onblur`, `onfocus`	`form`, `frame`, window	Gain et perte de focus
`onchange`	`select`, `text`, `textarea`	Modification de contenu
`onclick`	élément de formulaire, lien	Clic sur l'élément
`ondblclick`	élément de formulaire, lien	Double-clic sur l'élément
`onerror`		Erreur survenue lors du chargement
`onkeydown`, `onkeypress`, `onkeyup`	document, bouton, lien	Événement clavier
`onload`	`img`, window	Fin de chargement de page
`onunload`	window	Changement de page
`onmousedown`, `onmouseup`	document, bouton, lien	Bouton de la souris
`onmouseover`, `onmouseout`	la plupart des éléments	Survol d'un élément

W3C Document Object Model (DOM)

- Voir le cours XML
- Récupérer éléments et attributs

```
paragraphs = document.getElementsByTagName("p")
element = paragraphs[0].childNodes[2]
field = document.getElementById("creditcard")
field_type = field.getAttribute("type")
```

- Modifier le texte d'un élément

```
score = document.getElementById ("score")
score.firstChild.nodeValue = 12
```

- Ajouter un élément

```
heading = document.createElement ("h1");
text = document.createTextNode("Conclusion");
heading.appendChild(text);
document.body.appendChild(heading);
```

Browser Object Model

- Extension non standardisée de DOM
- Permet d'accéder
  - aux fenêtres affichées, au navigateur
  - à des objets utilitaires, comme le XMLHttpRequest.

[treemode=R,levelsep=4cm,treesep=2mm,nodesep=3mm]window location history frames navigator appName appVersion

Javascript : validation de formulaire

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
  <title>validation de formulaire par Javascript</title>
    <script type="text/javascript">
      <!--
      function validateForm() {
        if (document.getElementById("nom").value=='') {
          alert("Entrez votre nom");
          return false;
        }
        return true;
      }
      // -->
    </script>
  </head>

  <body>
    <h1>Essai JS</h1>
    <form action="POST" onSubmit="return validateForm()">
      <p>
        Votre nom:<input type="text" name="nom" size="32" /><br />
        <input type="submit" value="Valider" />
      </p>
    </form>
  </body>
</html>
```

Javascript : modification d'image au survol

```
<a href="target.html"
  onmouseout="document.getElementById('butc').src='contact.png'"
  onmouseover="document.getElementById('butc').src='contactp.png'">
  <img id="butc"
       width="140" height="53"
       alt="[Contact␣Information]"
       src="contact.png" />
</a>
```

Javascript : manipulation de styles CSS (1/3)

CSS

```
.hinted {
  color: blue;
}
.hintbox {
  position: relative;
}
.hint {
  display: none;
  width: 10em;
  position: absolute;
  left: 0;  top: -2em;
  background-color: #ffffcc;
  font-size: 75%;
  padding: 0.5em;
}
```

Javascript : manipulation de styles CSS (2/3)

Javascript

```
function toggle(id,display) {
  elem = document.getElementById(id);
  if (display)
    elem.style.display = 'block';
  else
    elem.style.display = 'none';
}
```

Javascript : manipulation de styles CSS (3/3)

XHTML

```
<p>Petit exemple
  <span class="hinted"
  onmouseover="toggle('xhtml',true)"
  onmouseout="toggle('xhtml',false)">XHTML</span>
  <span class="hintbox">
  <span class="hint" id='xhtml'>eXtensible Hypertext Markup
      Language</span>
  </span>
  et Javascript : une infobulle d'aide contextuelle.</p>
```