Internet : développer en PHP

PREPA1
Développement Web
Langage PHP

31,5 h

Olivier Pons / 2013 - 2014

Objectif

Faire un site Internet en entierr

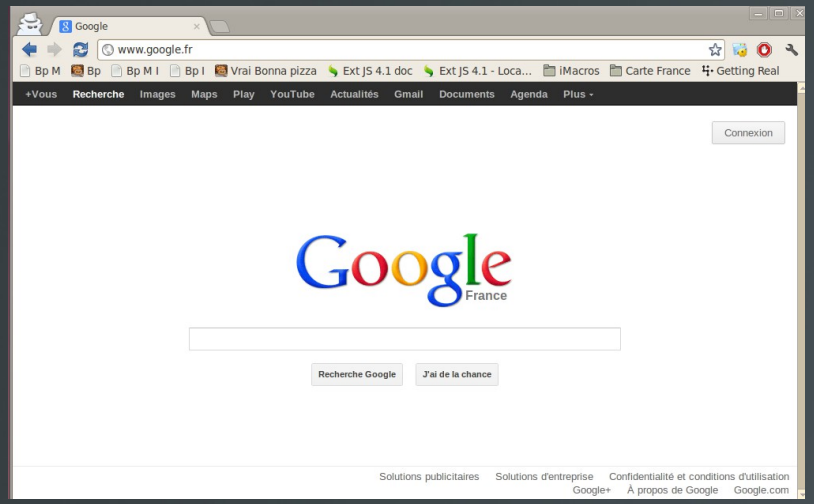
Sommaire

1 – Fondamentaux – DNS

```
fsf.com  =>  208.73.210.29
gnu.org  =>  140.186.70.148
...
```

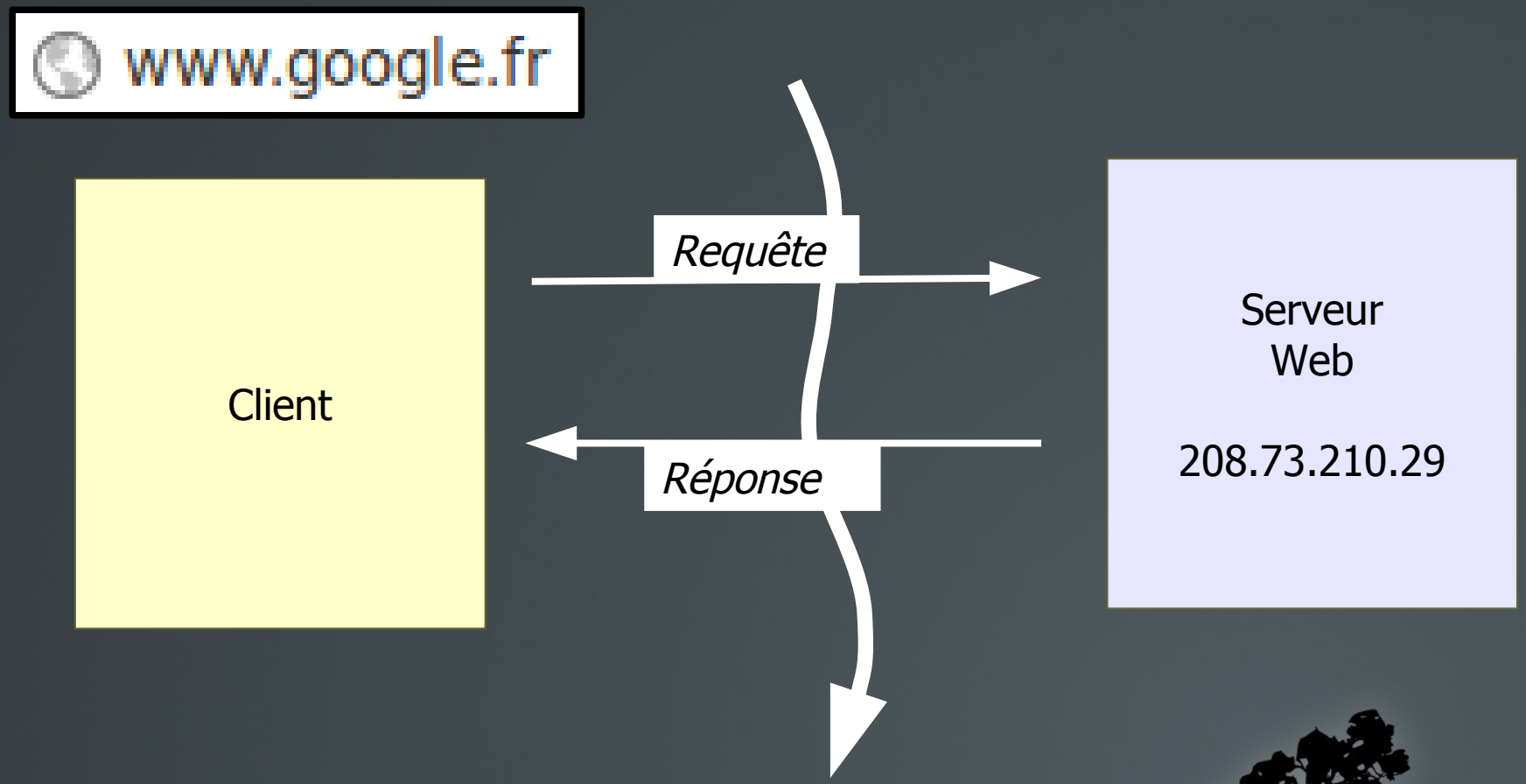
1 – Fondamentaux – DNS

www.google.fr



=> 208.73.210.29

1 – Fondamentaux – Client / Seveur



1 – Fondamentaux – Protocole

- HTTP: HyperText Transfer Protocol
- HTTP: les principales méthodes
  - GET URL : demander le contenu de la ressource
  - POST URL : envoi de données vers une application
- HTTP: le transport
  - Architecture Client-Serveur, mode « Pull »
  - Connections courtes, « Sans état » (stateless)

1 – Fondamentaux – Client / Seveur

Client

Requête

```
GET / HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT …
Accept: text/xml,application/xml, … ,image/png, …
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

Serveur Web

1 – Fondamentaux – Client / Seveur

Client

Réponse

```
HTTP/1.1 200 OK
Date: Sun, 14 Aug 2005 15:10:14 GMT
Server: Apache/1.3.23 (Win32)
Content-Type: text/html

<html>
<head></head>
<body><h1>Page</h1></body>
</html>
```

Serveur Web

1 – Fondamentaux – Echanges

(1) Client demande une page
(2) Serveur renvoie la page

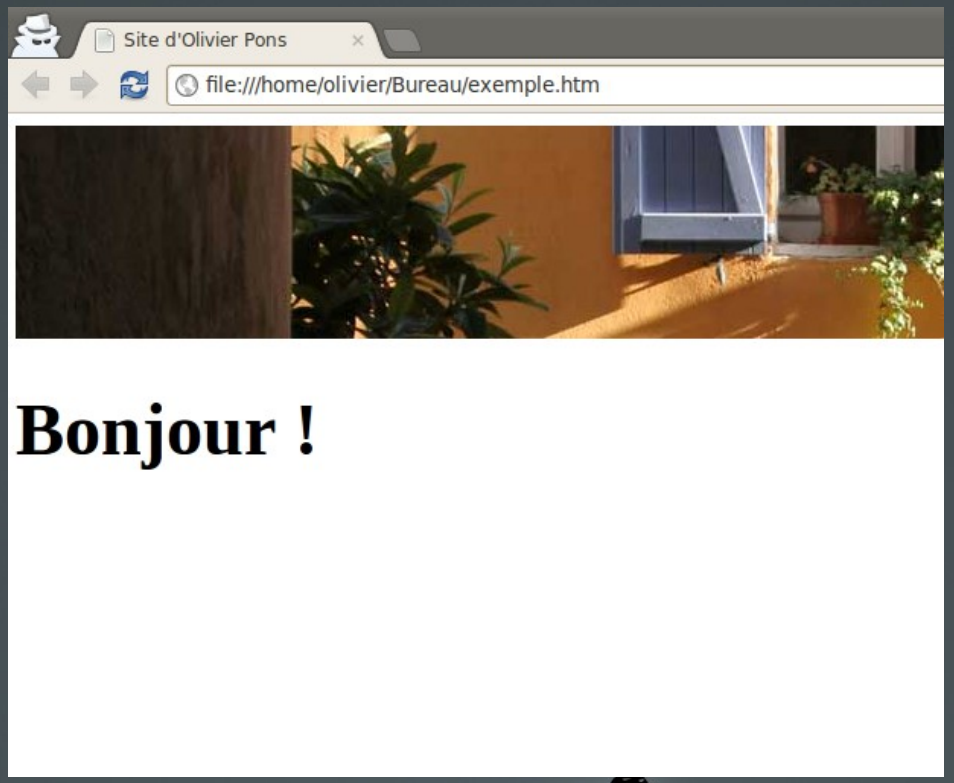
(Boucle)

Client demande ressource nécessaire à la page
Serveur renvoie la ressource

(Fin boucle)

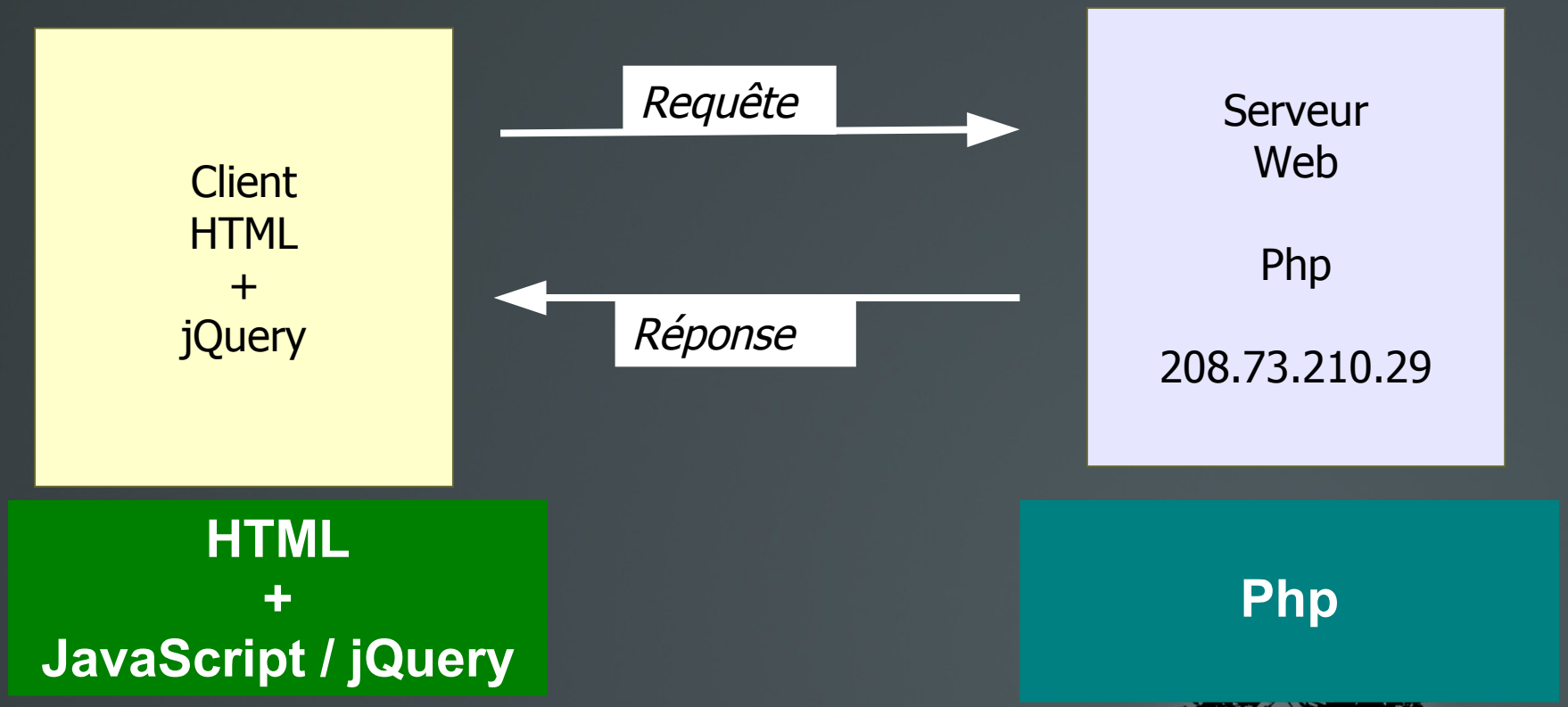
1 – Fondamentaux – HTML

```
<html>
  <head>
    ...
    <title>Site d'Olivier Pons</title>
    ...
  </head>
<body>
  <div>
    <img src="/header.jpg"
         alt="Texte si erreur"
         title="Titre image" />
  </div>
  <h1>Bonjour !</h1>
</div>
</body>
</html>
```



1 – Fondamentaux – AJAX

`http://monsiteweb.fr/post.php`



2 – VirtualHosts

```
>ping olivierpons.fr
PING olivierpons.fr (88.191.136.228) 56(84) bytes of data.

>ping keemy.com
PING keemy.com (88.191.136.228) 56(84) bytes of data.
```

1 – Hôtes virtuels

Le premier virtualhost est le virtualhost par défaut.

```
<VirtualHost *:80>
  ServerName keemy.fr
  ServerAlias www.keemy.fr
  DocumentRoot /var/www/keemy/
</VirtualHost>
```

1 - Fondamentaux

PHP: Hypertext Preprocessor3, plus connu sous son sigle PHP (acronyme récursif), est un langage de programmation libre4 principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP3, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté-objet.

En informatique, la programmation impérative est un paradigme de programmation qui décrit les opérations en termes de séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme.

http://fr.wikipedia.org/wiki/PHP

1 - Fondamentaux

- PHP : PHP Hypertext Preprocessor (version 5.4 à ce jour)
- Langage de script crée par Rasmus Lerdorf en 1995
- Langage impéératif et (plus récemment) orienté objet
- Principalement utilisé pour générer des pages Web
- Largement repandu sur Internet
- Site de référence : `www.php.net`
- Documentation : `www.php.net/MOT-CLE`
- Attention é ce qu'on peut lire sur Internet
- Utilisation possible en CLI (COMMAND LINE INTERFACE)

1 – Fondamentaux – Généralités

L'interpréteur ne s'intéresse qu'au code entre balises :

```
<?php /* mon code PHP */ ?>
```

Si le fichier termine par du PHP, la balise de fin est facultative

Ce qui est en dehors n'est pas traité et laissé tel quel
Instructions séparées d'un `;` (pas obligatoire pour la dernière)
Les commentaires sont `/* */` et `//`

1 – Fondamentaux – Généralités

Inclusion de fichiers multiple :

- `include $fichier`
- `require $fichier`

Inclusion de fichiers unique :

- `include_once $fichier`
- `require_once $fichier`

Utilise un garbage collector

1 – Fondamentaux – Les variables

Les variables n'ont pas besoin d'être déclarées
Noms de variables précédées d'un **$**
Leur type est déterminé à l'exécution selon leur utilisation
Exemple : **<?php $maVariable = 2; $maVariable = 3.2; ?>**
4 types scalaires :
- **int** : valeurs prises
- **float** : valeurs telles que **1.234** ou **1.2e3** mais aussi **NAN**
- **bool** : valeurs **TRUE** ou **FALSE** (non sensible à la casse)
- **string** : chaine de caractères non unicode

isset($var) pour tester l'existence
unset($var) pour effacer

1 – Fondamentaux – Les tableaux

Déclaration d'un tableau : **array()**

```
$tab = array();
```

- sans clé : **array()**

```
$tab = array( 1, 2, "Chaine", 1.5 );
```

Accès aux éléments :

```
echo $tab[0]." - ".$tab[1]." - ".$tab[2];
```

- avec clé : **array()**

```
$tab = array( "c1" => 1, "a" => 2, 0 => "Chaine");
```

Accès aux éléments :

```
echo $tab["c1"]." - ".$tab["a"]." - ".$tab[0];
```

1 – Fondamentaux – Les tableaux

Ce sont des tableaux associatifs : clef => valeur
Ils peuvent se créer dynamiquement :

```php
<?php
$tab[0] = 3;
$tab[1] = "toto";
?>
```

```php
<?php
$tab["test"] = 12;
$tab[ 15   ] = false;
?>
```

Ou bien être déclarés avec array() :

```php
<?php
$tab = array(3, " toto ");
?>
```

1 – Fondamentaux – Chaînes

`'string'` : interprétation de `'\\'` et `'\''` seulement
"`string`" : interprète en plus les `"$variable"`, les `"\n"`, . . .
Exemple : `<?php $maPhrase = "valeur:\n$maValeur"; ?>`
Fonction longueur :

- `strlen($str)` pour connaitre son nombre de caractères
- si en utf8, `strlen("aeià")` vaut `6`

- `mb_strlen($str)` pour connaitre la longueur (international)
- si en utf8, `mb_strlen("aeià")` vaut `4`

1 – Fondamentaux – Variables

2 types composés :
- **`array`** : tableau associatif à clef numérique ou textuelle
- **`object`** : objet instancié d'une classe

2 types spéciaux :
- **`resource`** : lien vers une ressource externe
- **`null`** : type avec une seule valeur **`NULL`** (non sensible à la casse)

Autres pseudo-types : void, mixed, callable, ...

1 – Fondamentaux – Variables - Transtypage

Une variable **`$x`** est considérée **`null`** si :
- On lui a assigné **`null`** comme valeur
- Elle n'a pas encore reçu de valeur
- Elle a été effacée avec **`unset($x)`**

Transtypage (type) **`$x`** et conversion **`settype($x,"type")`**
Contrôle de types avec **`is_bool($x)`**, **`is_string($x)`**, ...

1 – Fondamentaux – Variables - Dump

`print_r($x)` : le contenu d'un élément (tableaux)

`var_dump($x)` : infos sur des éléments (types)

`var_export($x)` : infos sur des éléments (types)

Dans une chaine : `$chaine = var_export($x, true);`

1 – Fondamentaux – Variables - Opérateurs

`[]`	accéder à un élement d'un tableau (chaînes y compris)
`$x++ $x++ ~$x`	Incrémenter, décrementer, négation bit à bit
`instanceof`	true si l'objet est une instanciation
`!$x`	Négation logique
`+ - * / %`	Opérateurs mathématiques classiques
`.`	Concaténation d'une chaine
`<< >>`	Décalages bit à bit
`< <= >= >`	Comparaisons d'inégalités
`== != === !==`	Comparaisons égalité / + type
`new clone`	Créer un nouvel objet / cloner

1 – Fondamentaux – Fonctions - Déclaration

```
function maFc()
{
}

function mcFc($parametre)
{
}
```

Les arguments sont passés par valeur
On peut les déclarer n'importe où
Passage par référence avec & dans la déclaration

1 – Fondamentaux – Fonctions - Déclaration

```
$a = 1;
function maFc()
{
    $a = 2;
}
maFc();
print $a."\n";
```

Qu'affiche cette fonction ? Pourquoi ?

1 – Fondamentaux – Fonctions - Déclaration

```
$a = 1;
function maFc()
{
    global $a;
    $a = 2;
}
maFc();
print $a."\n";
```

Qu'affiche cette fonction ? Pourquoi ?