Technologies du web

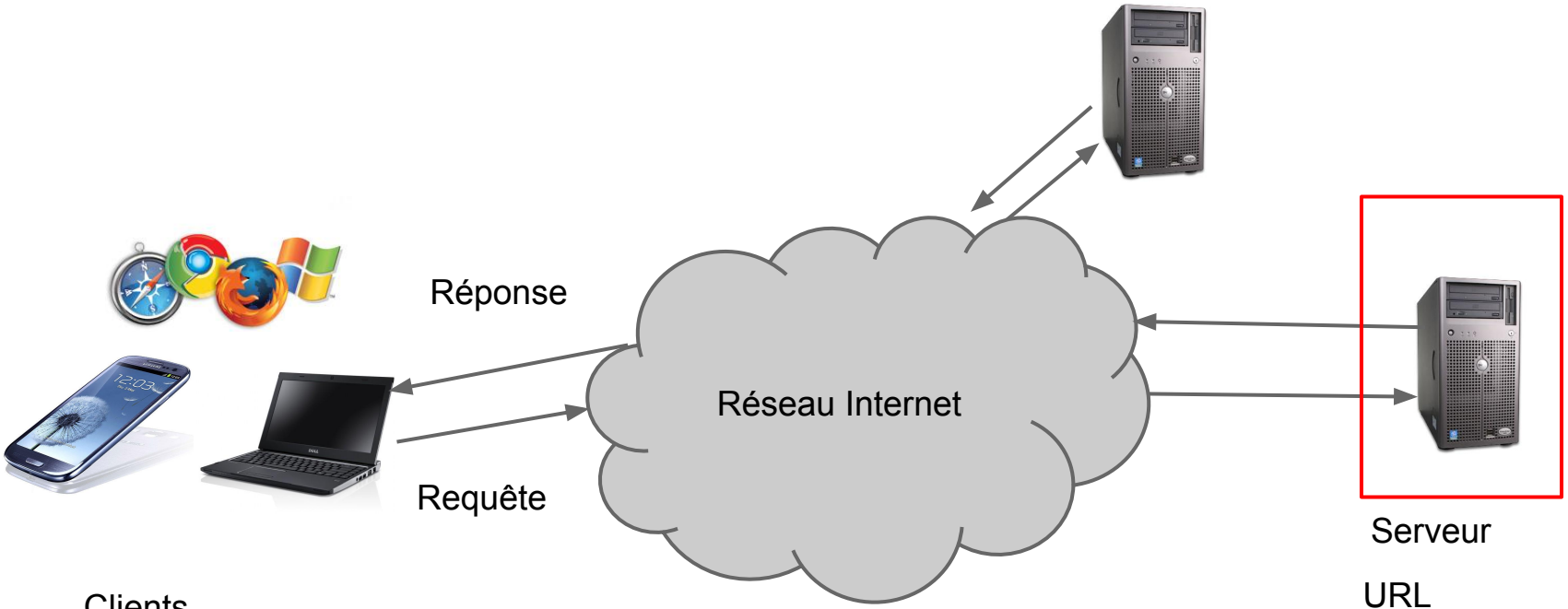
3 - Serveur Web

Architecture Web

- Le navigateur interprète et affiche *des données codées en*
  - *HTML / CSS / Javascript*

- D’où proviennent ces données ?

Architecture client / serveur



Clients
Navigateur
HTML / CSS / Javascript

Serveur

- Une machine / ordinateur connecté au réseau (internet)

- Fonction : **répondre à des requêtes provenant de ce réseau**.

Comment trouver un serveur

- Un serveur est identifié par
  - Une adresse IP, ex : 216.58.210.195 (IPv4)
  - Un nom de domaine, ex : www.google.fr

216.58.210.195 ⇔ www.google.fr

Serveur Web

- Machine répondant aux requêtes du **protocole HTTP ou HTTPS (version encryptée)**

- Qui délivre du contenu affichable dans un navigateur (mais pas que)
    - HTML / CSS / Javascript
    - Contenu multimedia Images / Videos / PDF...

Protocole

- **HTTP** Hypertext Transfer Protocol
- Protocole de transport de données au dessus de **TCP/IP**

- Transporte des caractères de texte (HTML)
- Transporte des fichiers

Autres protocoles : HTTPS, FTP, File

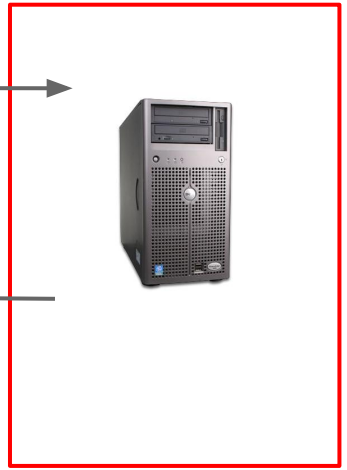
Logiciel serveur Web

- Écoute et attend les requêtes HTTP
- Analyse la requête
- Renvoie un résultat (valeur de retour)

    - Statique : Fichiers sur le serveur
    - Dynamique : Calculé par un programme

Protocol HTTP

Requête port 80
Header HTTP
(+ Corps du message)



Réponse
Header HTTP
(+ Corps du message)

Message HTTP

- Header : Méta informations de la requête
  - A ne pas confondre avec les headers HTML

- Corps : Contenu du message
  - Code HTML
  - Données binaires (etc fichier image, video, pdf etc…)

HEADER HTTP - Request

- Meta informations de la requête
  - **Host** : URL demandée
  - **Method** : GET / POST / PUT / DELETE
  - **User-Agent** : Type de client web (ex firefox, safari, IOS…)
  - **Content-Type** : Type de contenu de la requête
  - **Authorization**
  - **Cookie**
  - **Accept** : Type de retour attendue
  - **...**

Type de contenu ou Type MIME

- Champ **content- type**
  - text/plain
  - text/html
  - image/png
  - video/mp4
  - application/javascript
  - application/json
  - ...

Protocole HTTP / HTTPS

Actions principales

* **GET : récupère une donnée/page**
* **POST : envoie des données au serveur**
* DELETE
* PUT
* HEAD

HTTP Header - Response

- Status
  - 200 : OK
  - 404 : NOT FOUND
  - 304 : Non modifié
  - 500 : Internal Error
  - 403 : Forbidden
  - …
- Content-Type
- Cache-Control
- ...

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

Visualiser les headers

```
curl -X GET http://www.google.fr -vI
```

Requête sous forme d’URL

- Protocole
  - ex http://
- Nom du serveur
  - ex www.google.fr, localhost
- Chemin séparé par des slashes
  - ex /le_chemin/du_fichier
- Paramètres
  - ex ?p1=1&p2=3&p4=hello

Requête sous forme d’URL

Nom de domaine DNS :

http://www.google.fr

Ressource :

http://fr.wikipedia.org**/wiki/Hypertext_Transfer_Protocol**

Paramètres:

https://www.google.fr/search**?q=toto**

https://www.google.fr/search**?q=titi&tbm=isch**

URLs relatives

Dans un lien (balise **<a>**) ou dans une image **<img>**, on peut spécifier une url relative à l'url courante.

./images/photo.jpg

au lieu de

http://monserveur/images/photos.jpg

Localhost

Localhost = Ma machine

Localhost = 127.0.0.1

Ex : http://localhost/monsite/index.html

Logiciel serveurs

- Apache HTTP server

- NGINX (Reverse proxy)
- Varnish (Gestion de Cache)
- Apache Tomcat (Java J2EE)
- ...

Fonctionnalités d’Apache HTTP

- Ecouter les ports HTTP/HTTPS en attendant des connexions (ports 80 et 443)
- Servir des fichiers
- Rediriger des requêtes (Reverse proxy)
- Filtrer des requêtes
- Répartir la charge
- Réécrire des requêtes (module rewrite)
- Gérer des droits d’accès

Servir des fichiers statiques

- On définit dans un fichier de configuration le répertoire où trouver les fichiers sur le serveur, par défaut : **/var/www**

=> La racine du site (root)

- **Toutes les urls seront relatives à cette racine**

Accès aux fichiers

http://monserver.com/monsite/fichier.html

-> /var/www/monsite/fichier.html

http://localhost/monsite/images/ma_photo.png

-> /var/www/monsite/images/ma_photo.png

Index.html / index.php

Par defaut, si aucun nom de fichier est sélectionné, Apache renvoie, si présent, le fichier *index.html* ou *index.php*

http://monserver.com/monsite

-> /var/www/monsite/index.html

public_html

Dans son répertoire personnel, Apache définit un répertoire par défaut pour héberger ses fichiers web.

http://localhost/~sdufour/

=> /home/sdufour/public_html/index.html

Apache Virtual Host

Définit la configuration d’un site

Un serveur peut héberger plusieurs sites (avec des noms de domaines différents)

- Où trouver les fichier (root)
- Le port à écouter (par défaut 80)
- Les droits d’accès

Fichier de configuration Apache

```
NameVirtualHost *:80
<VirtualHost *:80>

        ServerName monserver.com
        DocumentRoot /var/www/

        <Directory /var/www/>
                Order allow,deny
                Allow from all
        </Directory>
</VirtualHost>
```

Fichier .htaccess

Définition de règles d’accès aux ressources s’appliquant dans un répertoire particulier

```
AuthUserFile /exemple/.htpasswd # contient les login:pwd
AuthName "Accès protégé" # Authentification
AuthType Basic
Require valid-user
DirectoryIndex index.php index.html # fichier par defaut
Options -Indexes    # ne pas afficher le contenu du répertoire

Redirect permanent /dossier1 http://www.monsite.com/dossier2/
```

Données Statiques / Dynamiques

Statiques : le résultat correspond au contenu de fichiers HTML/CSS/... **stockés** sur le serveur.

Dynamiques : le résultat est **généré à la volée** par un programme en fonction de la requête.

Servir du contenu dynamique

- Selon **des données externes** (base de données)
- Selon l’utilisateur
- Selon la date
- Selon le contexte
- Selon le contenu de la requête (formulaire)
- ...

Contenu dynamique

- Besoin d'un langage (+ framework ?)
  - **PHP** *(Symphony, CakePHP, ...)*
  - Java/J2EE *(Struts, Play!, Spring, ...)*
  - Ruby *(RubyOnRails)*
  - Python *(Django)*
  - Node.JS *(Express)*



PHP - Généralités

- *Crée en 1994, dérivé du langage Perl*
- *Personal Home Page Tools/Form Interpreter*
- *1997 : PHP 3 PHP Hypertext Preprocessor*
- *2004 : PHP 5*
- *2016 : **PHP 7***
  - Langage de script interprété côté serveur
  - Le plus souvent couplé à un serveur Apache HTTP
  - Adapté à la **génération de HTML**, au traitement de formulaires et à l'**accès aux bases de données**

PHP et Apache HTTP

- Le logiciel serveur va automatiquement exécuter le code PHP si le fichier demandé a une extension *.php (mod_php)*

Fichier PHP

C’est un fichier HTML amélioré

Une partie du code écrit dans ce fichier sera interprété et **exécuté par le serveur**

Le code PHP est dans une balise spéciale:

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

Fichier .php

```php
<?php
    // ceci est du code PHP
    echo 'Bonjour ça va ?';
    echo 'Il fait beau aujourd’hui';
?>
```

Generation HTML

```
<p>Ceci est un paragraphe hors bloc PHP</p>

<?php
    // generation de HTML
    echo '<h1>Bonjour ça va ?</h1>';
    echo '<h2>Il fait beau aujourd'hui</h2>';
?>

<p>Ceci est un paragraphe hors bloc PHP</p>
```

Interpretation du PHP

Hors balise **<?php …. ?>**, le texte est rendu tel quel.

Entre les balises **<?php … ?>** le code est analysé et **exécuté sur le serveur**. Le texte résultant est alors renvoyé au navigateur. (grâce à l'instruction *echo*)

Commentaires en PHP

```php
<?php
    // commentaire style C
    # commentaire style shell
    /* commentaire multi
    ligne */

    echo 'ceci n’est pas un commentaire';
?>
```

Variables PHP

```
<?php
  $a = -5;      # ceci est un commentaire de fin de ligne
  $resultat = $a + 8;

  echo "a : $a<br/>";
  echo "b : $resultat<br/>";
?>
```

Chaines de caractères PHP

```
<?php
  $message = "une chaine de caracteres";
  $message2 = $message . " (exemple 2)";   # concatenation
  $message3 = "$message (exemple 3)";   # substitution
  $message4 = '$message (exemple 4)';  # sans substitution



  echo "message : $message<br/>";
  echo "message2 : $message2<br/>";
  echo "message3 : $message3<br/>";
  echo "message4 : $message4<br/>";
?>
```

Les operations arithmétiques

```
$a = 1;
$b = 2;
$d = $a + 1;
$d = $b − $a;
$d = $a * $b;
$e = 5 / $a ;
$r = $d % 5;
$f = (5 * $a) - ($b * $a + 2);
$a += 2; $b -= 5;
$d++; ++$c ;
```

Condition

```
<?php
  $a =  8;
  $b = -2;

  if ($a == $b) {
    echo "a et b sont égaux";
  } else {
    echo "a et b sont différents";
  }
  echo "<br/>";
?>
```

Les opérations logique et de comp.

```
$a < $b
$a > $b
$a <= $b
$a >= $b
$a == $b
$a != $b
$a === $b  # comparaison de la valeur ET du type
$a !== $b  # comparaison de la valeur ET du type
($a > $b) && ($a > 0) # ET logique
($a < 0 || !$a) # OU et NON logique
```

Boucle

```
<?php
  for ($i = 0; $i < 10; $i++) {
    echo "$i<br/>";
  }
?>
```

Tableau simple (Array)

- Un Array contient une liste de valeur
  - indexée par leur position

```
<?php
  // liste de couleurs
  $couleurs = array("rouge", "vert", "bleu", "jaune");
  echo $couleurs[0] . "<br/>";
  echo $couleurs[2] . "<br/>";
  print_r($couleurs); // affiche le tableau
?>
```

Tableau associatif

- Un Array peut contenir aussi un tableau associatif
  - association clé / valeur

```
<?php
$ages = array("jean"=>20, "vincent"=>22, "julie"=>21);
echo $ages["vincent"]; // 22
 ?>
```

Parcours de tableaux simples

```
<?php
  $couleurs = array("rouge", "vert", "bleu", "jaune");
  foreach ($couleurs as $val) {
    echo "$val<br/>";
  }
?>
```

Parcours de tableaux associatifs

```php
<?php
  $ages = array("jean"=>20, "vincent"=>22, "julie"=>21);
  foreach ($ages as $key => $val) {
    echo "$key : $val<br/>";
  }
?>
```

Bibliothèque de fonctions

PHP fournit un grand nombre de fonctions prédéfinies (mathématiques, fichiers, temps, accès aux données...).

```
nom_de_la_fonction(p1, p2, …);
```

exemple : **date('Y-m-d H:i:s');**

Variables globales

PHP prédéfinit des variables globales contenant des données concernant la requête.

exemple : $_SERVER["REMOTE_ADDR"];

Exemples

- Affiche l’heure de la requete
  - date('Y-m-d H:i:s');

- Affiche l’adresse IP du navigateur :
  - $_SERVER["REMOTE_ADDR"];

Paramètres d’URL

Il est possible d’utiliser dans le code PHP les paramètres d’URL grâce à la variable globale $GET

Exemple:

- http://monserveur.com/page.php**?name=toto**
- echo $_GET["name"]; // affiche toto

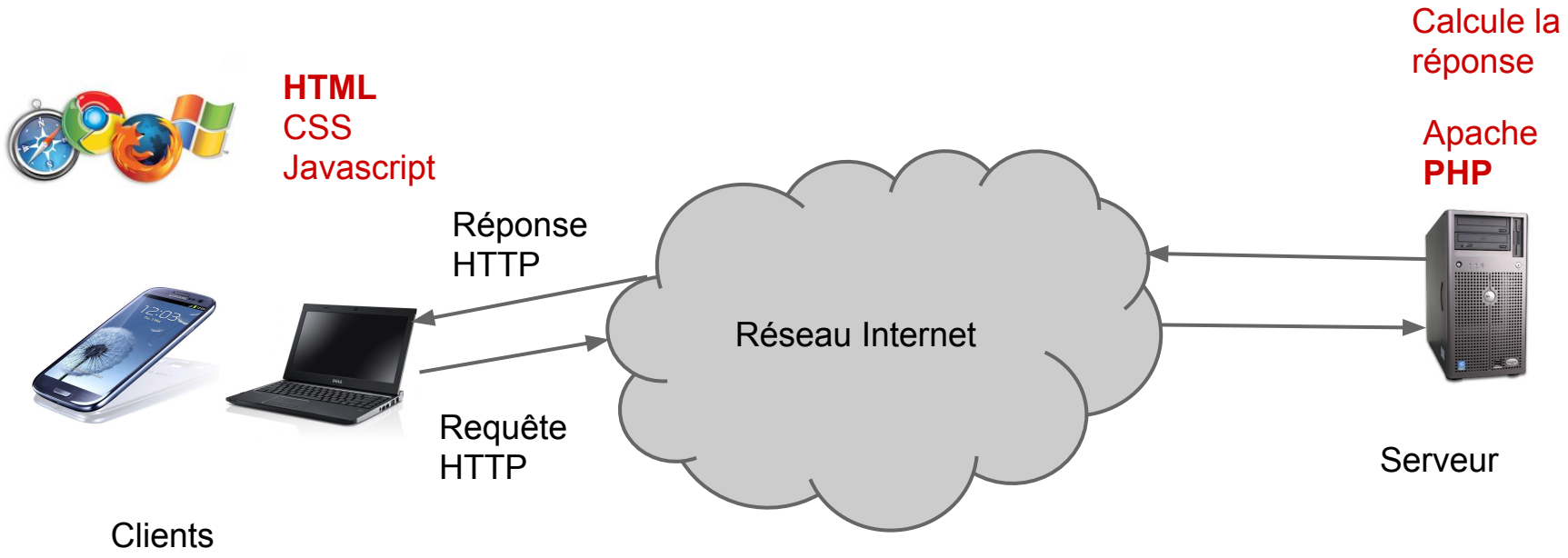
Exemple complet

```
<html>
     <head>
          <meta charset="UTF-8">
     </head>
     <body>
          <h1>Il est actuellement:</h1>
          <php?
              echo date('Y-m-d H:i:s');
          ?>
     </body>
</html>
```

Documentation PHP

- http://php.net/manual/fr/
- Utiliser google

Architecture client / serveur



Détails d’une requête

1. L’utilisateur fait une requête depuis son navigateur (url, lien…)
2. La requête est envoyé vers le serveur désigné (DNS/IP / Protocol HTTP)
3. Le logiciel serveur HTTP recoit et analyse la requête
4. Il charge la ressource designée depuis le repertoire ‘root’ du site
5. Si la ressource est un fichier PHP, le code est executé
6. Le résultat du traitement est renvoyé par le serveur vers le client (HTTP)
7. Le navigateur recoit le résultat HTML/CSS/Javascript
8. Le navigateur analyse ce résulat et l’affiche

Conclusion

- Un serveur web délivre du contenu via internet grace au **protocole HTTP**

- Le contenu peut être
  - soit des fichiers statiques HTML etc…
  - soit dynamique et généré par un programme écrit dans un langage comme PHP