



Formation PHP

Romain Bohdanowicz

Twitter : @bioub - <https://github.com/bioub>

<http://formation.tech/>



Introduction



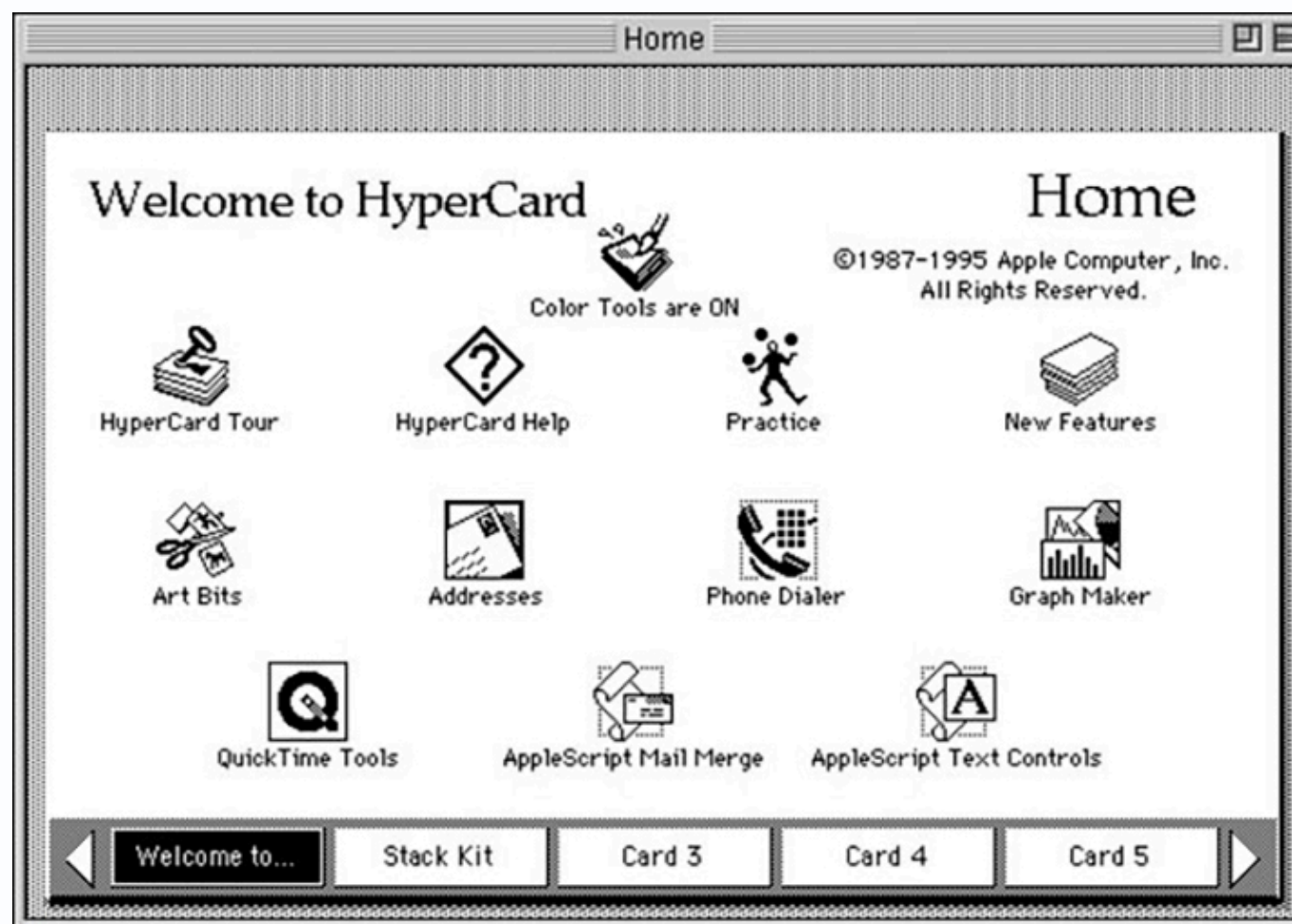
- Romain Bohdanowicz
Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle
- Expérience
Formateur/Développeur Freelance depuis 2006
Plus de 10 000 heures de formation animées
- Langages
Expert : HTML / CSS / JavaScript / PHP / Java
Notions : C / C++ / Objective-C / C# / Python / Bash / Batch
- Certifications
PHP 5 / PHP 5.3 / PHP 5.5 / Zend Framework 1
- Particularités
Premier site web à 12 ans (HTML/JS/PHP), Triathlète à mes heures perdues
- Et vous ?
Langages ? Expérience ? Utilité de cette formation ?

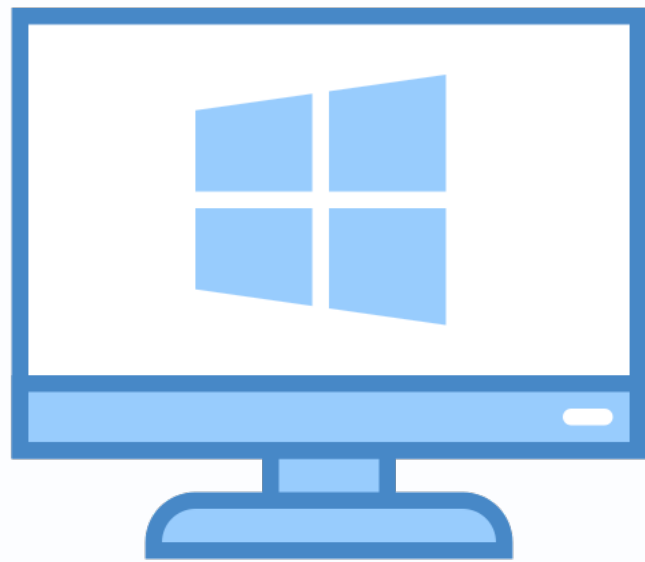


Technologies Web

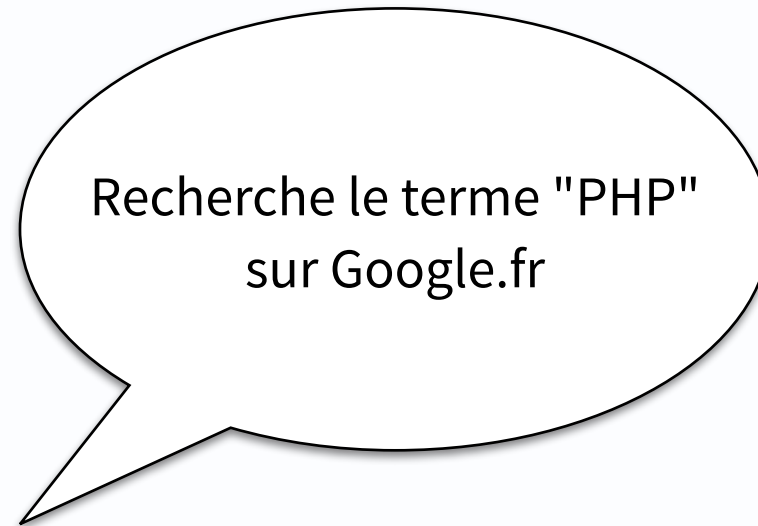


- ▶ Tim Berners-Lee, chercheur au CERN à Genève créé le web en 1989
- ▶ Il s'inspire d'un logiciel d'Apple appelé Hypercard, permettant de visualiser des "piles de cartes" en local, similaires aux pages web et aux liens
- ▶ L'idée est d'adapté le concept pour que les pages puissent être hébergées sur un serveur distant et consultables via le réseau

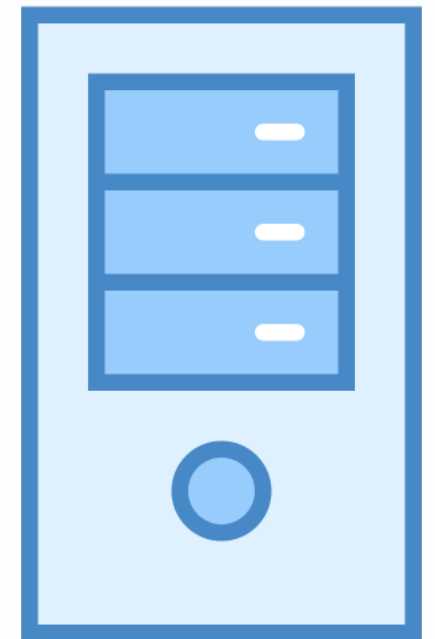




Client



Réseau



Serveur

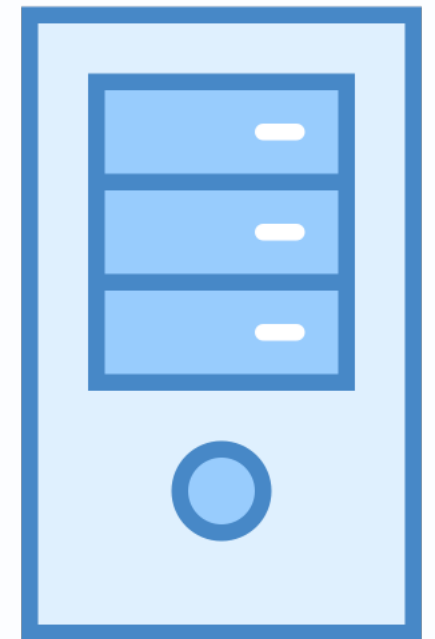


Client

Voici les résultats :

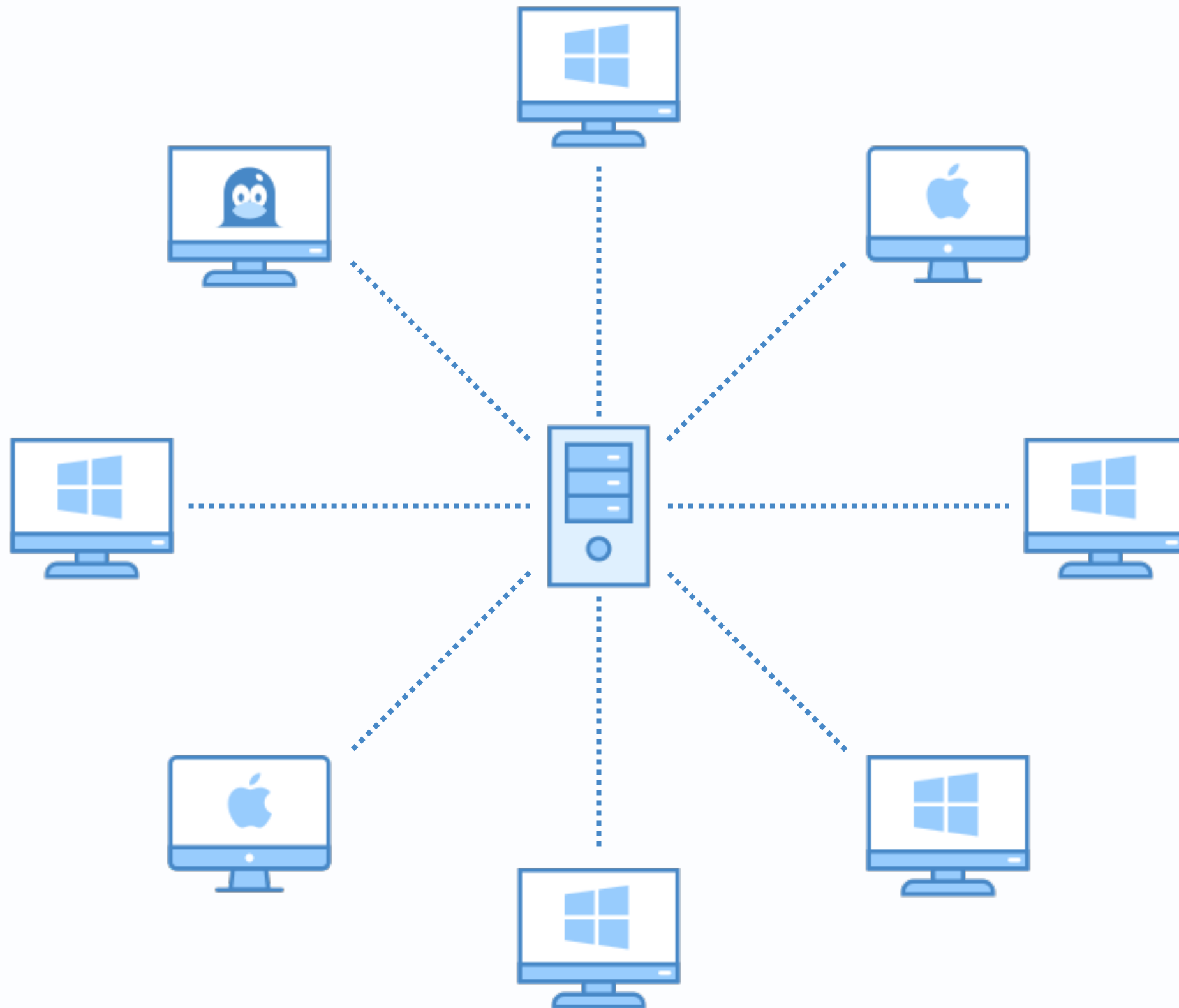
- Site officiel
- Wikipedia
- OpenClassrooms

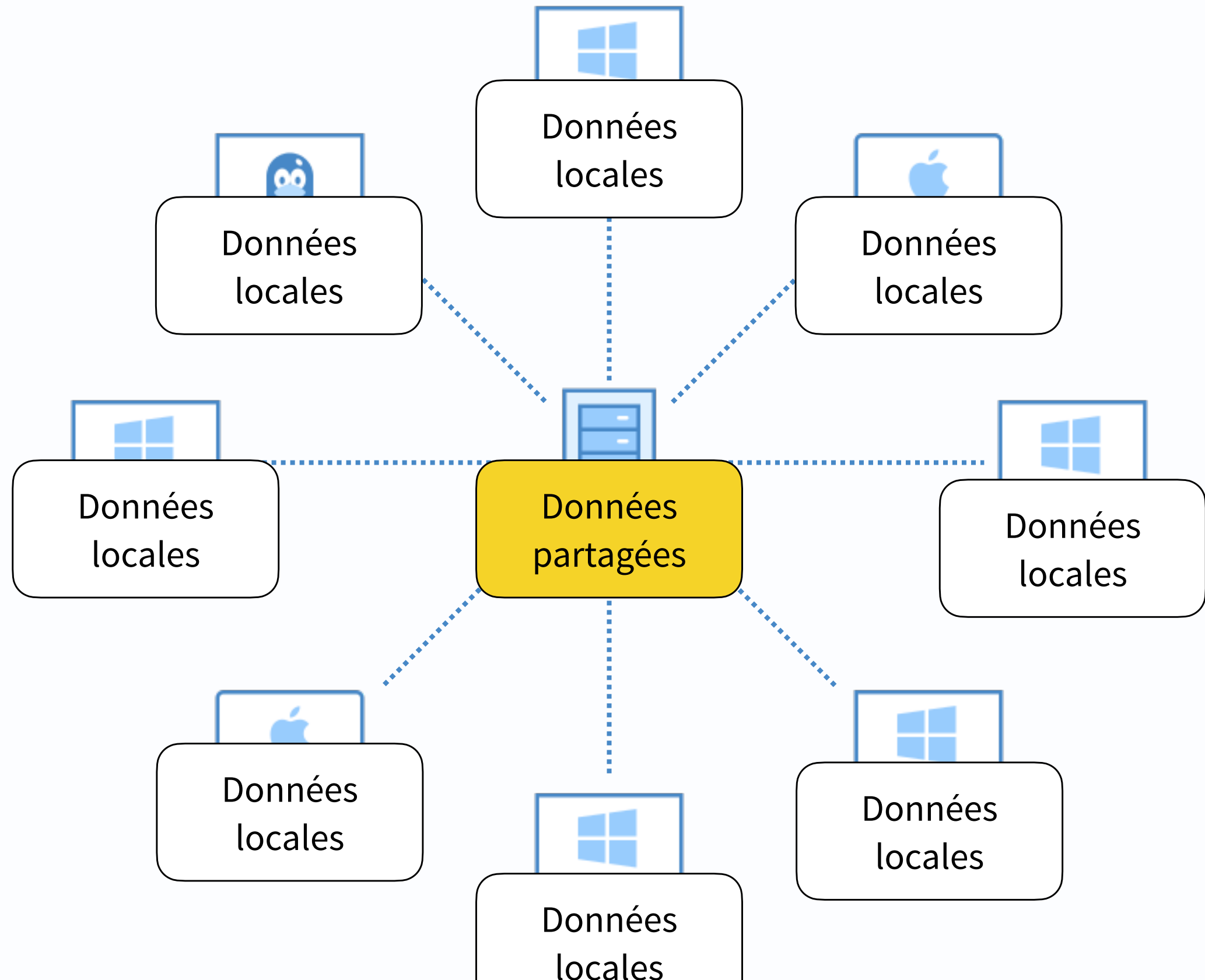
Réseau



Serveur

Technologies Web - Client / Serveur





Technologies Web - Technologies / Languages



HTML
CSS
JavaScript
...

Client

HTTP
JSON
XML
...

Réseau

PHP
JavaScript (Node.js)
Java
C#
Ruby
Python
Go
...

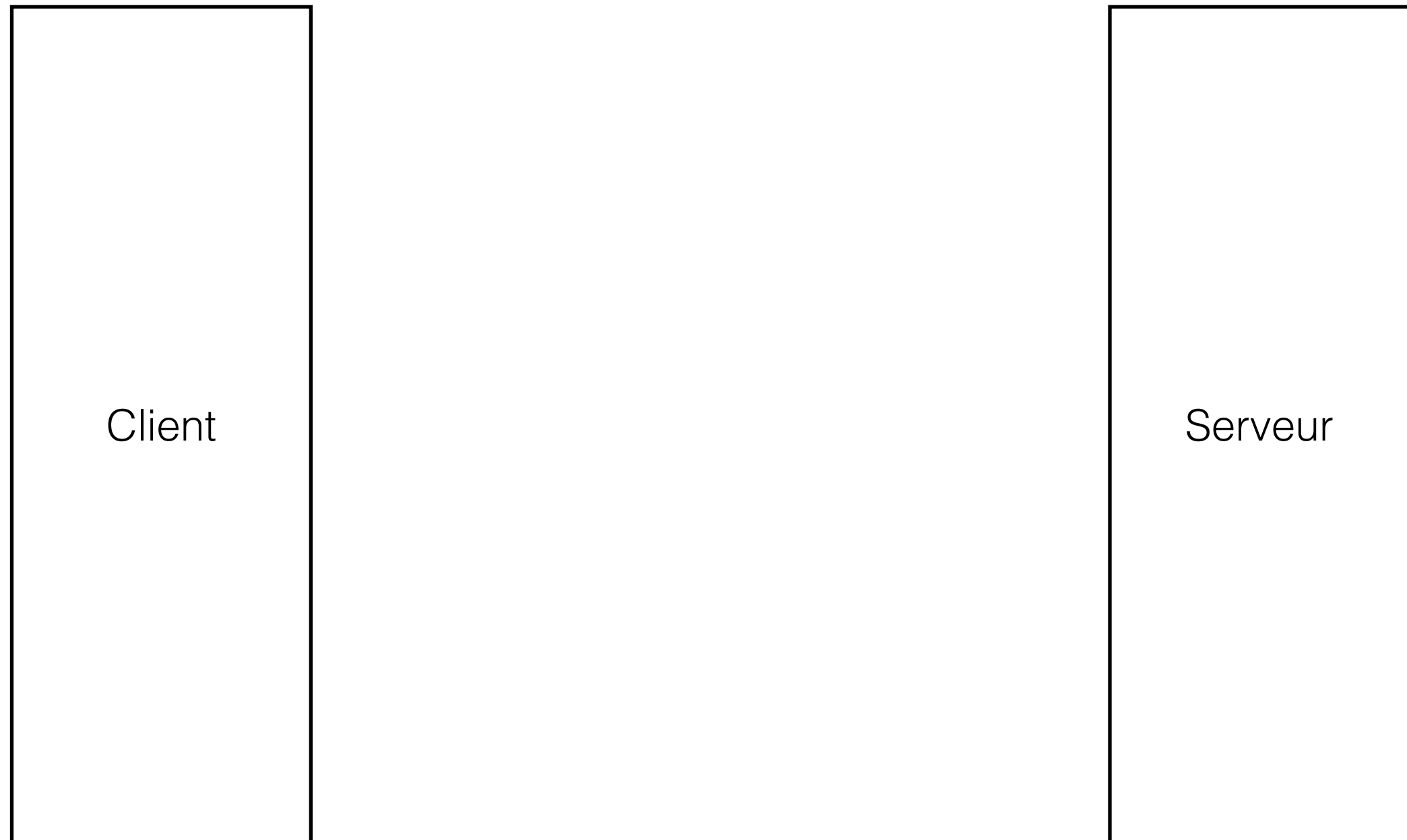
Serveur



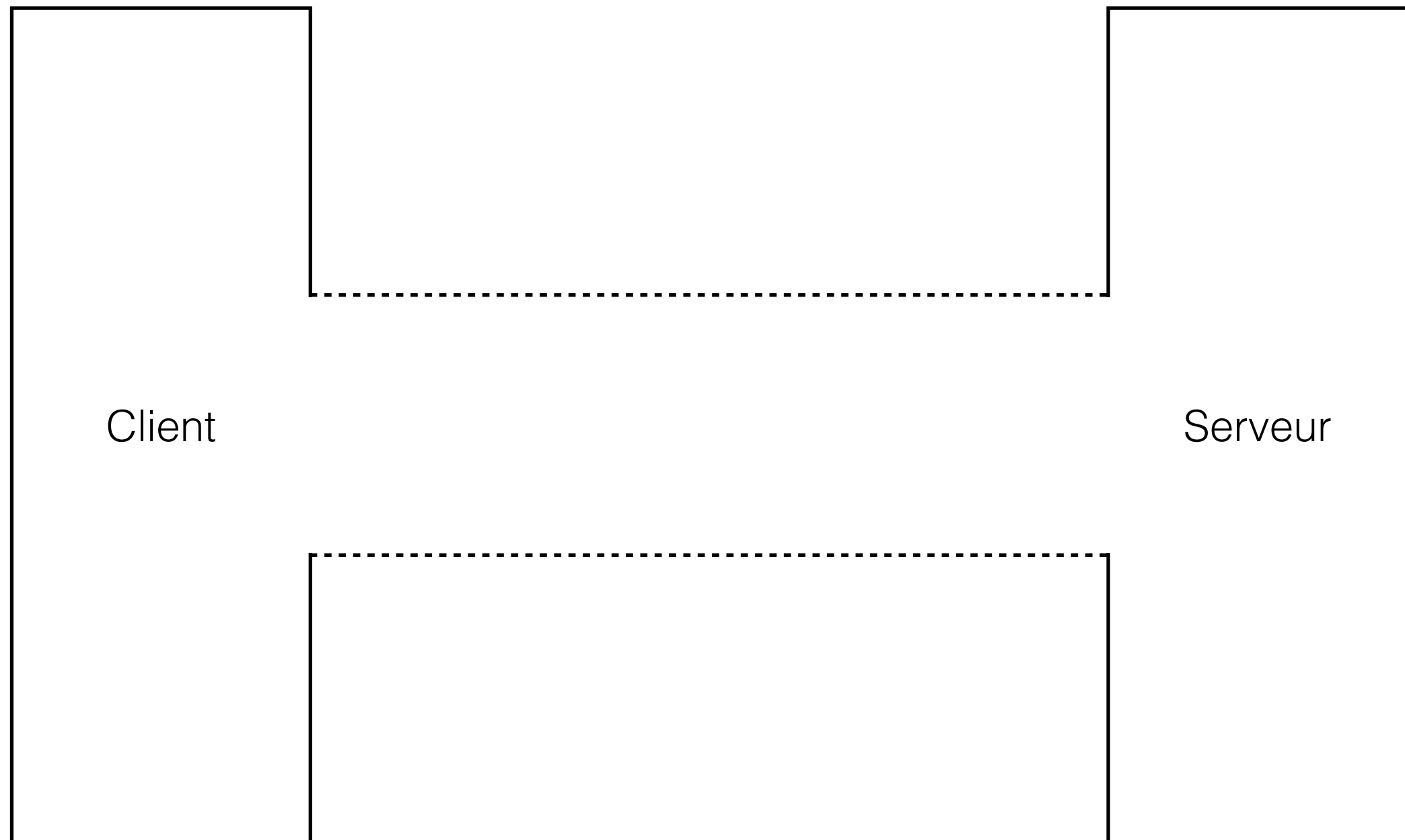
- Protocole réseau (norme réseau) permettant l'échange de données entre un client et un serveur web
- Créé en 1990 par Tim Berners-Lee alors chercheur au CERN
- Normé pour la première fois en 1996, HTTP/1.0 dans la RFC 1945
<https://tools.ietf.org/html/rfc1945>
- Nouvelles fonctionnalités dans HTTP/1.1, RFC 2616
<https://tools.ietf.org/html/rfc2616>
Version la plus répandue aujourd'hui
- Evolution majeure en 2015 avec HTTP/2, RFC 7540
<https://tools.ietf.org/html/rfc7540>



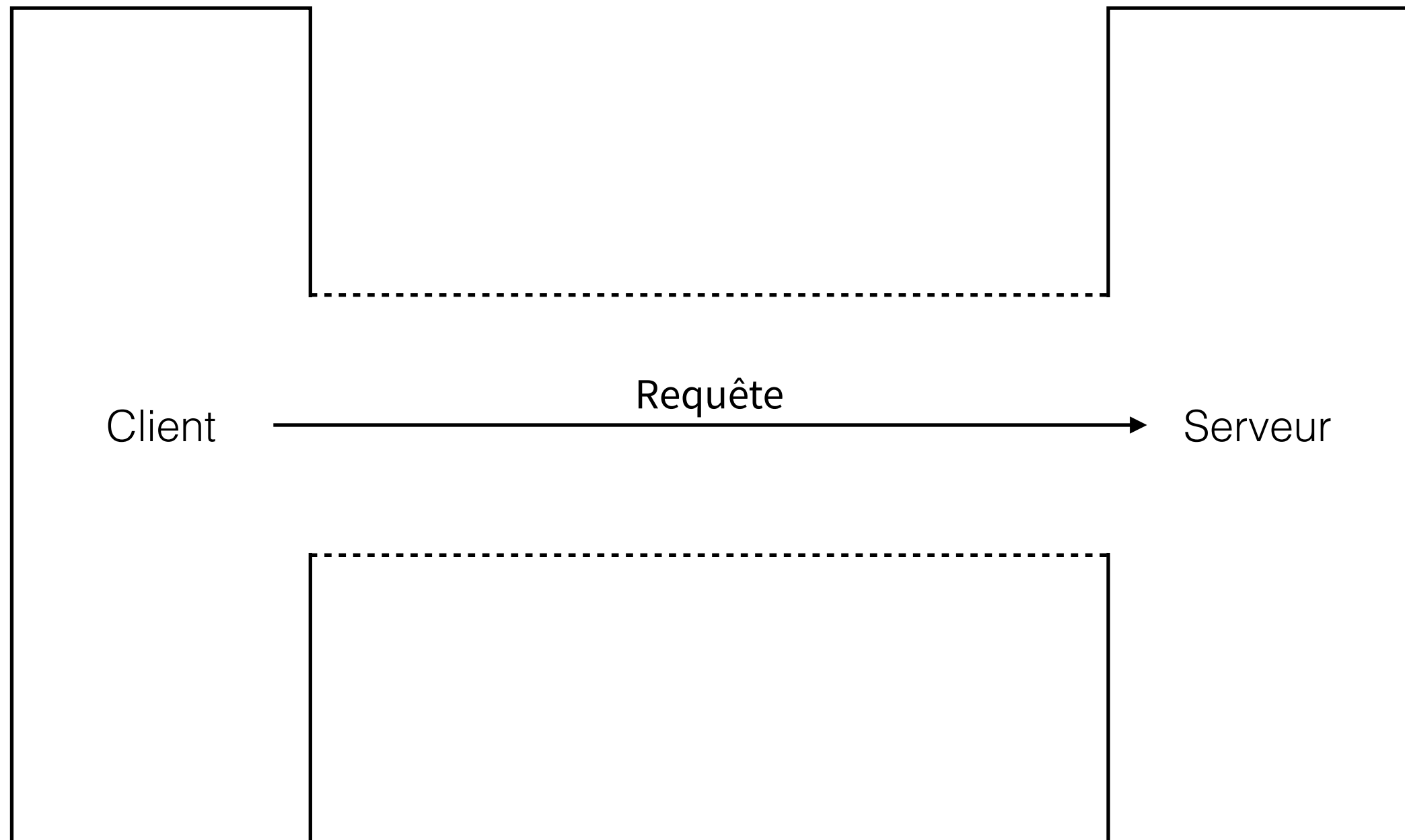
- Quatre étapes
 1. connexion du client HTTP
 2. envoi d'une requête
 3. réponse du serveur HTTP
 4. le serveur ferme la connexion pour signaler la fin de la réponse.
- La requête étant à l'initiative du client on parle de *Client Pull*, par opposition au *Serveur Push* où le serveur est à l'origine



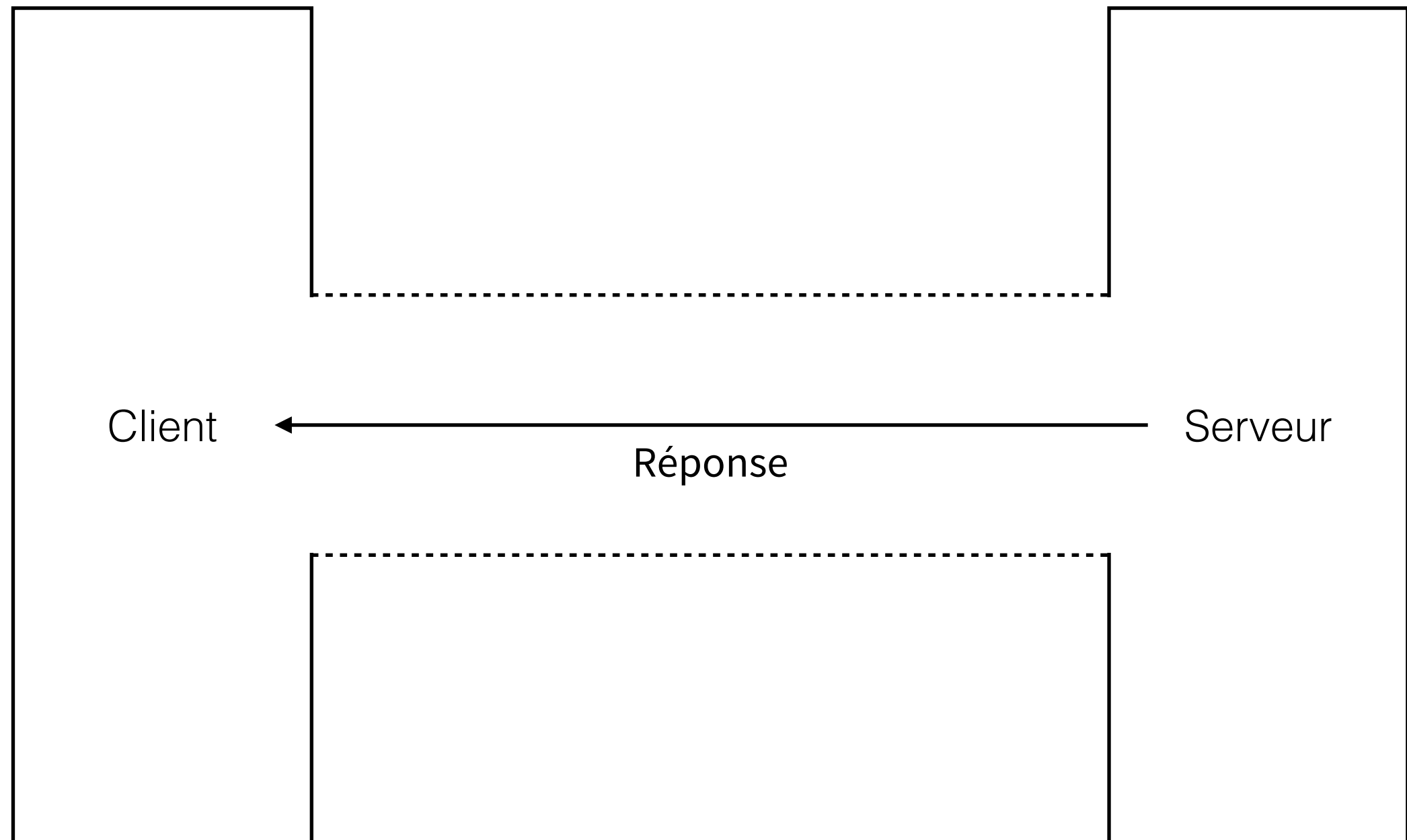
2 programmes souhaitent échanger des données via le réseau



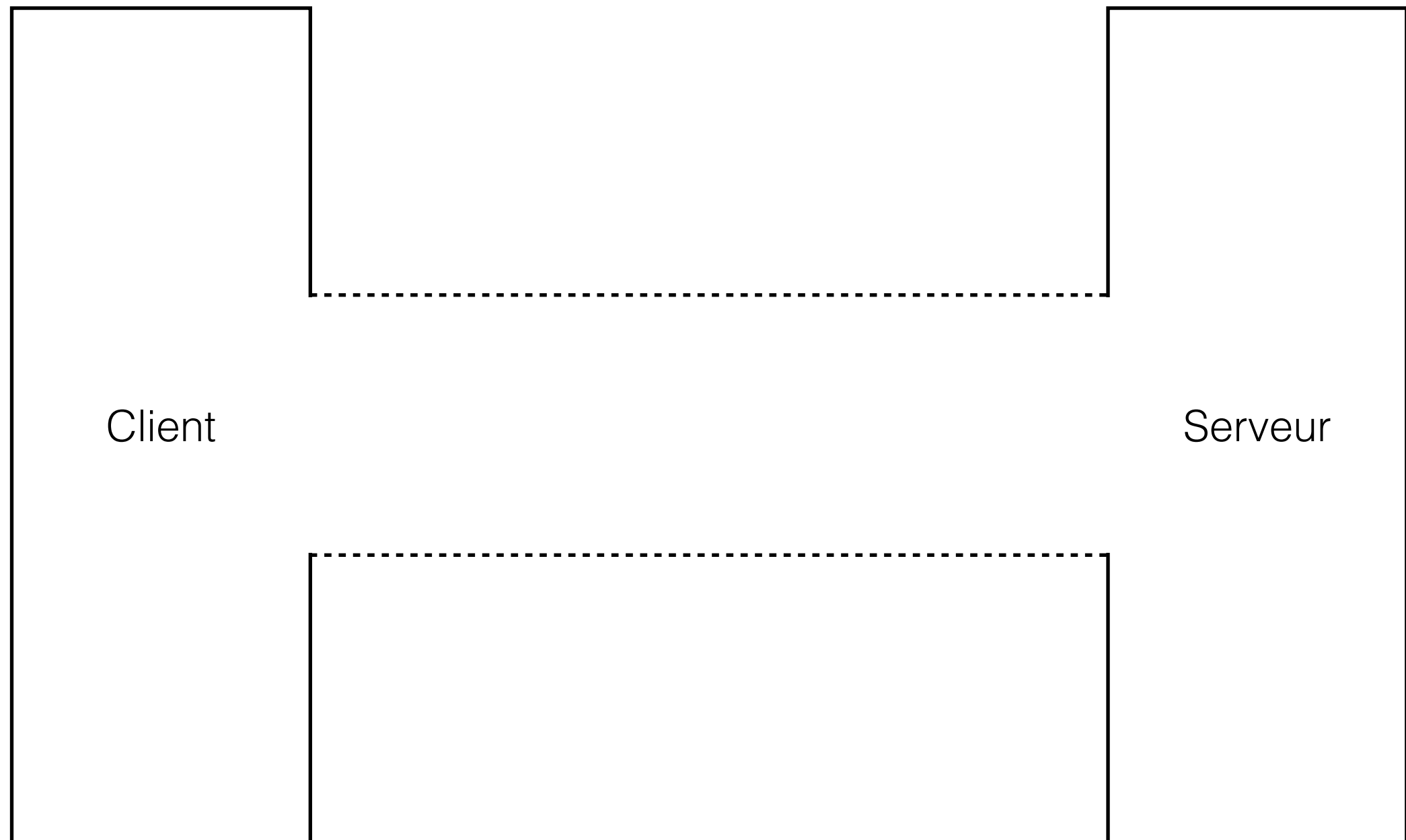
Le client ouvre une connection (socket) entre
lui et le serveur



Le client envoie une requête (request)



Le serveur traite la requête et émet une
réponse (response)



Le serveur ferme la connection



connexion du client HTTP

envoi d'une requête

réponse du serveur HTTP

le serveur ferme la connexion pour
signaler la fin de la réponse.

```
MacBook-Pro:~ romain$ telnet example.com 80
Trying 93.184.216.34...
Connected to example.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Sat, 11 Aug 2018 14:37:14 GMT
Etag: "1541025663"
Expires: Sat, 18 Aug 2018 14:37:14 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (dca/24D1)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1270

<!doctype html>
<html>
<head>
...
</body>
</html>
Connection closed by foreign host.
```

Technologies Web - URI



https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top

userinfo host port
scheme authority path query fragment

ldap://[2001:db8::7]/c=GB?objectClass?one

scheme authority path query

mailto:John.Doe@example.com

scheme path

news:comp.infosystems.www.servers.unix

scheme path

tel:+1-816-555-1212

scheme path

telnet://192.0.2.16:80/

scheme authority path

urn:oasis:names:specification:docbook:dtd:xml:4.1.2

scheme path



- Principaux langages compris par les navigateurs modernes
 - HTML : structure de la page (apprentissage facile)
 - CSS : mise en forme (apprentissage intermédiaire)
 - JavaScript : interactions dynamiques avec l'utilisateur ou le serveur (apprentissage difficile)
- D'autres moins connus :
 - SVG, RSS, ATOM, MathML, ...
- Langages du serveur :
 - PHP, Java, C#, Ruby, Python, Go, JavaScript...



PHP



- PHP 1 (1994)
Rasmus Lerdorf créer un ensemble d'outils CGI pour son site sur lequel il publie son CV, PHP veut alors dire Personal Home Page
- PHP 2 (1995)
PHP devient un langage de programmation à part entière, la syntaxe est inspirée du BASIC et les fonctions utilisateurs sont désormais possibles
- PHP 3 (1997)
Andi Gutmans et Zeev Suraski alors étudiant à Tel Aviv réécrivent entièrement le coeur de PHP, la syntaxe s'inspire du C. PHP devient un acronyme récursif - PHP: Hypertext Preprocessor.
- PHP 4 (1999)
Andi Gutmans et Zeev Suraski réécrivent le coeur de PHP qui se nomme désormais Zend Engine (puis la société Zend), introduction de la programmation orientée objet
- PHP 5 (2004)
Zend Engine 2, importante réécriture de la partie objet qui s'inspire désormais de Java : Classes Abstraites, Interfaces.
- PHP 7 (2015)
Pas de PHP 6 car une version a été en cours d'écriture puis abandonnée et des livres avaient été écrits. Principale nouveautés, le typage statique devient possible sur tous les types.

PHP - A quoi ça sert ?

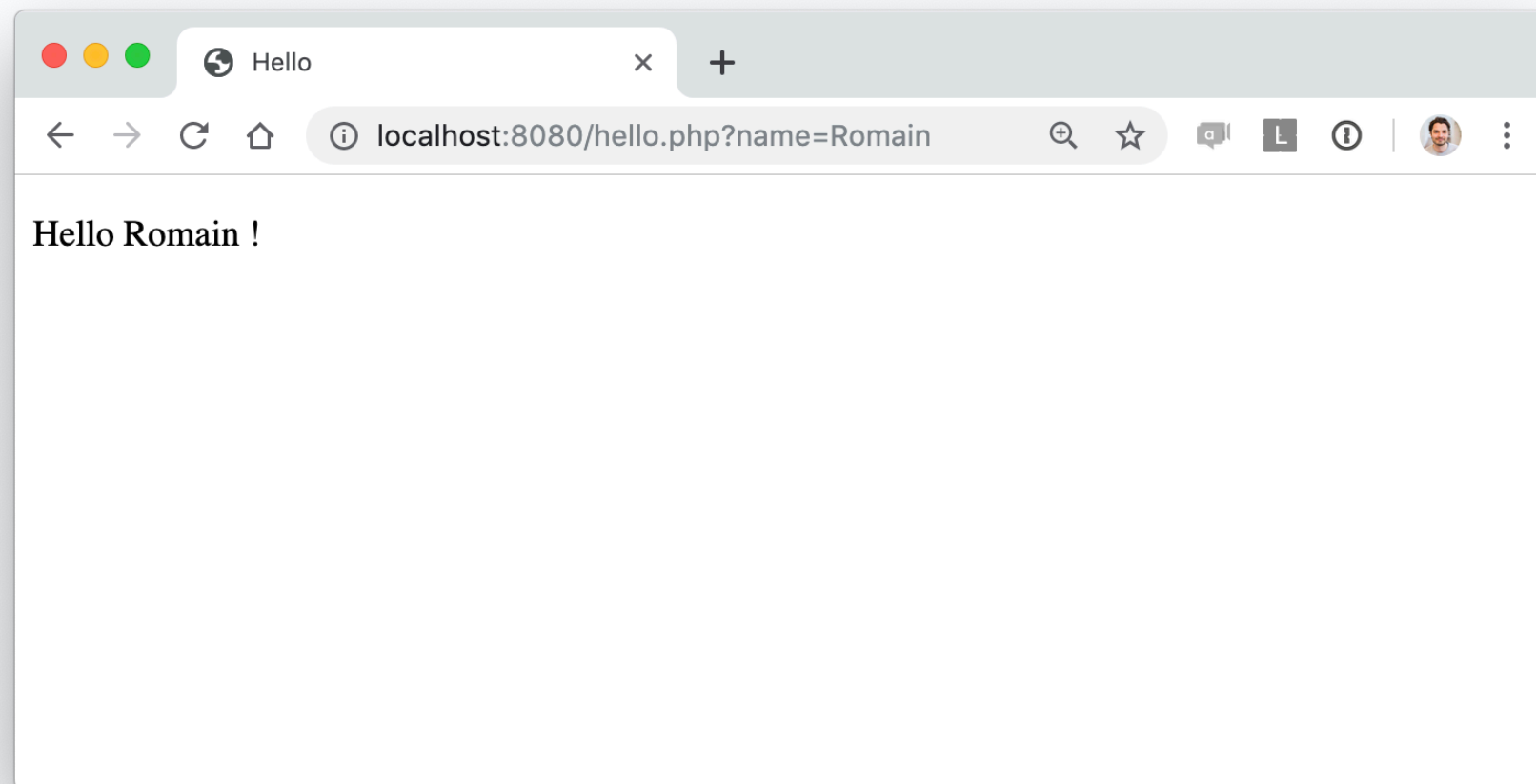


- Aujourd'hui on peut écrire 3 types de programmes en PHP
 - des scripts serveurs (la plupart du temps)
 - des scripts en ligne de commande (rarement)
 - des applications graphiques (quasiment jamais)

PHP - Exemple de script en ligne de commande (CLI)



```
<?php
$name = $_GET['name'] ?? '';
?>
<p>Hello <?=$name?> !</p>
```



PHP - Exemple de script en ligne de commande (CLI)



```
<?php
$name = $argc > 1 ? $argv[1] : '';
echo "Hello $name\n";
```

A terminal window with a title bar that reads "Slides — -bash — 53x10". The window contains the following text:

```
[$ php hello.php Romain
Hello Romain
$
```

PHP - Application Graphiques



- PHP permet l'écriture d'application graphiques avec PHP-GTK
- La documentation est inexistante
- Préférez d'autres langages si le besoin de fait ressentir



PHP - Syntaxe de base



- On écrit le code dans des balises PHP

```
<?php
function hello($name)
{
    return "Hello $name\n";
}

$names = ['Romain', 'Edouard'];

foreach ($names as $name) {
    echo hello($name);
}
```

- Il n'y a pas besoin d'utiliser la balise fermante en fin de fichier ?> (et c'est même recommandé)
- Pour afficher quelque chose, on appelle utilise la construction de langage *echo*



- Lorsque contenu de la balise PHP ne fait qu'un echo d'une valeur, on peut utiliser la syntaxe courte

- Ainsi

```
<p><?php echo 'Bonjour' ?></p>
```

- Est l'équivalent de

```
<p><?= 'Bonjour' ?></p>
```

PHP - Échappement depuis du HTML



- On peut utiliser PHP comme un moteur de template
- C'est à dire que dès qu'on écrit en dehors d'une balise PHP le contenu sera écrit sur la page

```
<?php
function hello($name)
{
    return "Hello $name\n";
}

$names = ['Romain', 'Edouard'];
?>
<ul>
    <?php foreach ($names as $name) : ?>
        <li><?= hello($name) ?></li>
    <?php endforeach; ?>
</ul>
```



- Une variable permet de mettre de côté une valeur pour plus tard
- On déclare une valeur en commençant par \$

```
<?php
$prenom = 'Romain';
$age = 33;
$estFormateur = true;

echo $prenom; // Romain
```



- En PHP une valeur peut être de type :
string, int, double, boolean, array, object (et tous ces dérivés), resource, null, callable
- Une expression peut être assimilée à "tout ce qui a une valeur" (valeur, résultat d'un calcul, appel de fonction avec un retour...)

```
<?php
var_dump('texte'); // string(5) "texte"
var_dump(123); // int(123)
var_dump(1.2); // double(1.2)
var_dump(true); // bool(true)
var_dump([]); // array(0) {}
var_dump(new stdClass()); // class stdClass#1 (0) {}
var_dump(fopen('/dev/null', 'r')); // resource(5) of type (stream)
var_dump(null); // NULL
var_dump(function () {}); // callable
```



- Les chaînes de caractères (string) représentent du texte
- L'opérateur `.` permet de concaténer deux chaînes entre elles (les assembler)
- L'opérateur `[]` permet d'accéder au caractère présent à un certain indice
- Les autres opérations se feront via des fonctions
<https://www.php.net/manual/fr/ref.strings.php>
- C'est le type le plus répandu, faites le tour des fonctions savoir ce qu'il est possible de faire !

```
echo 'Romain'; // Romain
echo 'Romain' . ' ' . 'Bohdanowicz'; // Romain Bohdanowicz
echo strtoupper('Romain'); // ROMAIN
echo 'Romain'[0]; // R
```




- Echappement

Si la chaîne de caractère contient son délimiteur il faudra l'échapper en le préfixant avec un antislash \

```
echo 'Je m\'appelle Romain';  
echo "<a href=\"https://formation.tech/\">formation.tech</a>";
```

- Pour éviter d'échapper on peut pour des cas complexes utiliser les syntaxes Nowdoc et Heredoc qui permettent de remplacer les délimiteurs (ici par STR)
- Attention, il ne faut pas de caractères après STR sur la partie ouvrante et avant ou après STR sur la fermante

```
// Nowdoc  
echo <<<'STR'  
<a href="https://formation.tech/">C'est mon site !</a>  
STR;  
// Heredoc  
echo <<<STR  
<a href="https://formation.tech/">C'est mon site !</a>  
STR;
```



- Les chaînes de caractères déclarées avec les guillemets et Heredoc sont interprétées
 - Les variables sont remplacées par leur contenu
 - Les caractères tabulation, retour à la ligne sont pris en compte

```
$prenom = 'Romain';  
echo 'Bonjour $prenom\n'; // Bonjour $prenom\n  
echo "Bonjour $prenom\n"; // Bonjour Romain
```

```
$url = 'https://formation.tech/';  
$site = "C'est mon site !";  
// heredoc  
echo <<<STR  
<a href="$url">$site</a>  
STR;
```



- Le type booléen peut contenir 2 valeurs
 - `true` (vrai)
 - `false` (faux)
- C'est le résultat d'un test
- On l'utilise pour activer ou non une partie du code

```
<?php
if (true) {
    echo 'sera affiché';
}

if (false) {
    echo 'ne sera pas affiché';
}
```



- Opérateurs de comparaison
- Leur résultat est de type booléen

```
$age = 33;  
var_dump($age > 33); // false  
var_dump($age >= 33); // true  
var_dump($age < 33); // false  
var_dump($age <= 33); // true  
var_dump($age == 33); // true  
var_dump($age === 33); // true  
var_dump($age != 33); // false  
var_dump($age !== 33); // false
```

- Opérateur == vs ===

```
$age = 33;  
var_dump($age == '33'); // true  
var_dump($age === '33'); // false
```



- Il existe 3 opérateurs permettant de combiner des booléens (opérateurs logiques)
- Par exemple on ne peut pas écrire

```
$age = 33;  
var_dump(0 < $age < 120); // PHP Parse error: syntax error, unexpected '<'
```

- Car l'expression `0 < $age` est de type booléen et ne pourra pas être comparée avec l'opérateur `<`
- A la place il faut séparer 2 comparaisons `0 < $age` et `$age < 120` et les regrouper avec un opérateur logique

```
$age = 33;  
var_dump(0 < $age && $age < 120); // true
```



- ET Logique - `&&`
Toutes les expressions doivent être vraies pour que l'ensemble soit vrai
- OU Logique - `||`
Une seule expression doit être vraie pour que l'ensemble soit vrai
- NON Logique - `!`
Inverse le booléen

```
var_dump(true && true); // true
var_dump(true && false); // false
var_dump(false && true); // false
var_dump(false && false); // false

var_dump(true || true); // true
var_dump(true || false); // true
var_dump(false || true); // true
var_dump(false || false); // false

var_dump(!true); // false
var_dump(!false); // true
```



- Les structures conditionnelles permet d'exécuter ou non certaines instructions, selon une condition (expression booléenne)

```
<?php
if (/*condition*/) {
    /* bloc d'instructions; */
}
```

- Exemple

```
if (true) {
    echo 'sera affiché';
}

if (false) {
    echo 'ne sera pas affiché';
}
```

- Lorsqu'il n'y a qu'une seule instruction, les parenthèses ne sont pas obligatoires (mais recommandées)

```
if (true) echo 'sera affiché';
```



- Un bloc `if` peut être lié à un bloc `else` qui sera évalué si l'expression est fausse

```
if (true) {  
    echo 'sera affiché';  
} else {  
    echo 'ne sera pas affiché';  
}  
  
if (false) {  
    echo 'ne sera pas affiché';  
} else {  
    echo 'sera affiché';  
}
```




- Si un bloc `else` contient à nouveau un `if`

```
if ($nb < 10) {  
    echo "$nb est inférieur à 10";  
} else {  
    if ($nb < 20) {  
        echo "$nb est compris entre 10 et 20";  
    } else {  
        echo "$nb est supérieur à 20";  
    }  
}
```

- Il sera plus léger de ne pas mettre les accolades du `else` :

```
if ($nb < 10) {  
    echo "$nb est inférieur à 10";  
} else if ($nb < 20) {  
    echo "$nb est compris entre 10 et 20";  
} else {  
    echo "$nb est supérieur à 20";  
}
```



- Un tableau permet de regrouper plusieurs valeurs
- Un tableau numérique permet de retrouver sa valeur par rapport à la position dans le tableau
- Un tableau associatif permet de retrouver sa valeur par rapport à une autre (une clé)
- Il peuvent être hybrides (numériques / associatifs), mais ce serait une mauvaise pratique

```
<?php
// tableau numérique
$nbs = [2, 3, 4];

// tableau associatif
$capitales = [
    'France' => 'Paris',
    'Allemagne' => 'Berlin',
    'Espagne' => 'Madrid',
];
```



- Pour accéder aux valeurs d'un tableau on utilise les crochets []

```
<?php
// tableau numérique
$nbs = [2, 3, 4];
echo $nbs[0]; // 2

// tableau associatif
$capitales = [
    'France' => 'Paris',
    'Allemagne' => 'Berlin',
    'Espagne' => 'Madrid',
];
echo $capitales['France']; // Paris
```

- Les fonctions sur les tableaux :
<https://www.php.net/manual/fr/ref.array.php>



- Les boucles permettent de répéter plusieurs fois un bloc d'instructions
- while (nombre de passages inconnus à l'avance, entre 0 et n passages)

```
$nb = mt_rand(0, 100);  
  
while ($nb) {  
    echo $nb;  
    $nb = (int) ($nb / 2);  
}
```

- do ... while (nombre de passages inconnus à l'avance, entre 1 et n passages)

```
do {  
    $nb = mt_rand(0, 100);  
} while ($nb % 2 === 1);
```

- for (nombre de passages connus à l'avance)

```
$nbs = [2, 3, 4];  
  
for ($i=0; $i<count($nbs); $i++) {  
    echo $nbs[$i];  
}
```



- foreach (boucle sur tous les éléments d'un tableau)

```
$capitales = [  
    'France' => 'Paris',  
    'Allemagne' => 'Berlin',  
    'Espagne' => 'Madrid',  
];  
  
foreach ($capitales as $capitale) {  
    echo $capitale;  
}  
  
foreach ($capitales as $pays => $capitale) {  
    echo $pays;  
    echo $capitale;  
}
```



- Les fonctions permettent d'exécuter plusieurs fois le mêmes bloc, mais pas forcément à la suite
- Elles permettent de factoriser des traitements (évitant de dupliquer du code)

```
function hello($name) {  
    return "Bonjour $name";  
}  
  
echo hello('Romain'); // Bonjour Romain  
echo hello('Edouard'); // Bonjour Edouard
```

```
function addition($a, $b) {  
    return $a + $b;  
}  
  
echo addition(1, 2); // 3
```



▸ Recherche dans un tableau

```
$contacts = [  
    ['prenom' => 'Steve', 'nom' => 'Jobs'],  
    ['prenom' => 'Bill', 'nom' => 'Gates'],  
];  
  
function findByPrenom($array, $prenom) {  
    foreach ($array as $elt) {  
        if ($elt['prenom'] === $prenom) {  
            return $elt;  
        }  
    }  
  
    return null;  
}  
  
echo findByPrenom($contacts, 'Steve')['nom']; // Jobs
```



- Créer une fonction qui affiche le plus grand des 2 nombres en paramètres
- Créer une fonction qui affiche le plus grand des nombres du tableau reçu en paramètre
- Créer une fonction qui retourne les éléments d'un tableau dans l'ordre inverse
- Créer une fonction qui inverse les clés et les valeurs d'un tableau associatif



MySQL



- MySQL est un Système de Gestion de Bases de Données Relationnel (SGBDR ou RDBMS en anglais)
- Une base de données relationnelles est un ensemble de tables, que l'on pourrait représenter sous la forme d'un tableau Excel, mais qui sont stockées sur le disque d'une manière rendant des opérations de lectures / écritures bien plus rapides
- Une tables contient des colonnes et des enregistrements

MySQL - Table



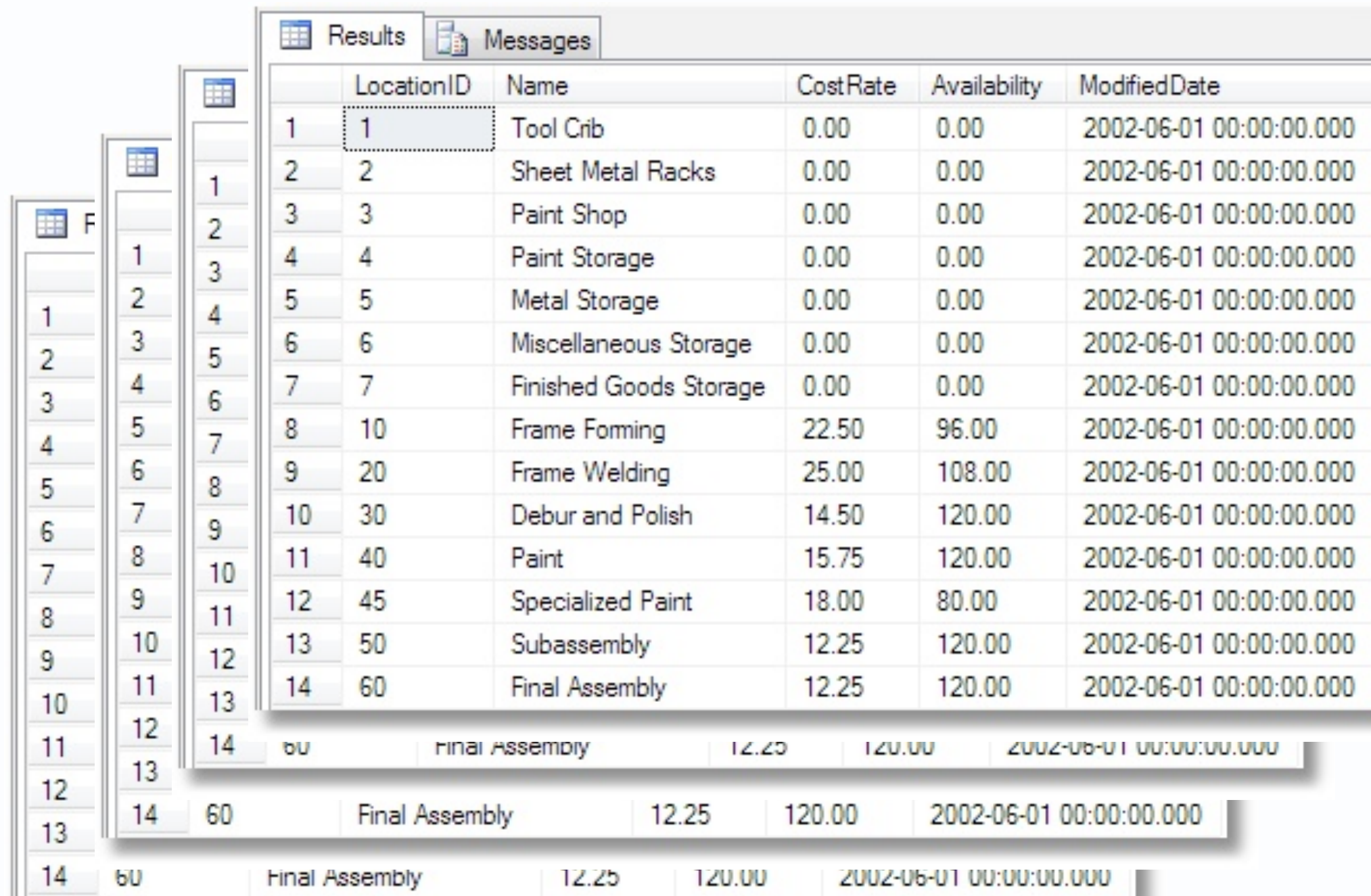
- Une table contient les enregistrements

Results		Messages			
	LocationID	Name	CostRate	Availability	ModifiedDate
1	1	Tool Crib	0.00	0.00	2002-06-01 00:00:00.000
2	2	Sheet Metal Racks	0.00	0.00	2002-06-01 00:00:00.000
3	3	Paint Shop	0.00	0.00	2002-06-01 00:00:00.000
4	4	Paint Storage	0.00	0.00	2002-06-01 00:00:00.000
5	5	Metal Storage	0.00	0.00	2002-06-01 00:00:00.000
6	6	Miscellaneous Storage	0.00	0.00	2002-06-01 00:00:00.000
7	7	Finished Goods Storage	0.00	0.00	2002-06-01 00:00:00.000
8	10	Frame Forming	22.50	96.00	2002-06-01 00:00:00.000
9	20	Frame Welding	25.00	108.00	2002-06-01 00:00:00.000
10	30	Debur and Polish	14.50	120.00	2002-06-01 00:00:00.000
11	40	Paint	15.75	120.00	2002-06-01 00:00:00.000
12	45	Specialized Paint	18.00	80.00	2002-06-01 00:00:00.000
13	50	Subassembly	12.25	120.00	2002-06-01 00:00:00.000
14	60	Final Assembly	12.25	120.00	2002-06-01 00:00:00.000

MySQL - Base de données



- Une base de données est contient un ensemble de tables



	LocationID	Name	CostRate	Availability	ModifiedDate
1	1	Tool Crib	0.00	0.00	2002-06-01 00:00:00.000
2	2	Sheet Metal Racks	0.00	0.00	2002-06-01 00:00:00.000
3	3	Paint Shop	0.00	0.00	2002-06-01 00:00:00.000
4	4	Paint Storage	0.00	0.00	2002-06-01 00:00:00.000
5	5	Metal Storage	0.00	0.00	2002-06-01 00:00:00.000
6	6	Miscellaneous Storage	0.00	0.00	2002-06-01 00:00:00.000
7	7	Finished Goods Storage	0.00	0.00	2002-06-01 00:00:00.000
8	10	Frame Foming	22.50	96.00	2002-06-01 00:00:00.000
9	20	Frame Welding	25.00	108.00	2002-06-01 00:00:00.000
10	30	Debur and Polish	14.50	120.00	2002-06-01 00:00:00.000
11	40	Paint	15.75	120.00	2002-06-01 00:00:00.000
12	45	Specialized Paint	18.00	80.00	2002-06-01 00:00:00.000
13	50	Subassembly	12.25	120.00	2002-06-01 00:00:00.000
14	60	Final Assembly	12.25	120.00	2002-06-01 00:00:00.000



- Les index sont des colonnes dont la valeur permet de retrouver rapidement le ou les enregistrements associés
- Un index va augmenter la taille de la base de données, il convient donc de ne pas choisir toutes les colonnes d'une table comme index

The diagram illustrates how an index works by showing two tables. The left table has columns for a country and a numeric index value. The right table has columns for an index value, a country, and a city. Colored arrows connect the index values in the left table to their corresponding rows in the right table, demonstrating how the index allows for quick retrieval of records.

Canada	4
France	2
Italy	1
Japan	5
Nigeria	6
Poland	7
Syria	3
Thailand	8

1	Italy	Roma
2	France	Paris
3	Syria	Damascus
4	Canada	Ottawa
5	Japan	Tokyo
6	Nigeria	Lagos
7	Poland	Warsaw
8	Thailand	Bangkok

MySQL - Clés primaires / clés étrangères



- Une clé primaire est une valeur permettant d'identifier de manière unique un enregistrement dans un table (généralement un entier auto-incrémenté dans MySQL)
- Une clé étrangère est une valeur qui reproduit une clé primaire dans une autre table, permettant ainsi de lier les enregistrements

MySQL - Créer sa base de données

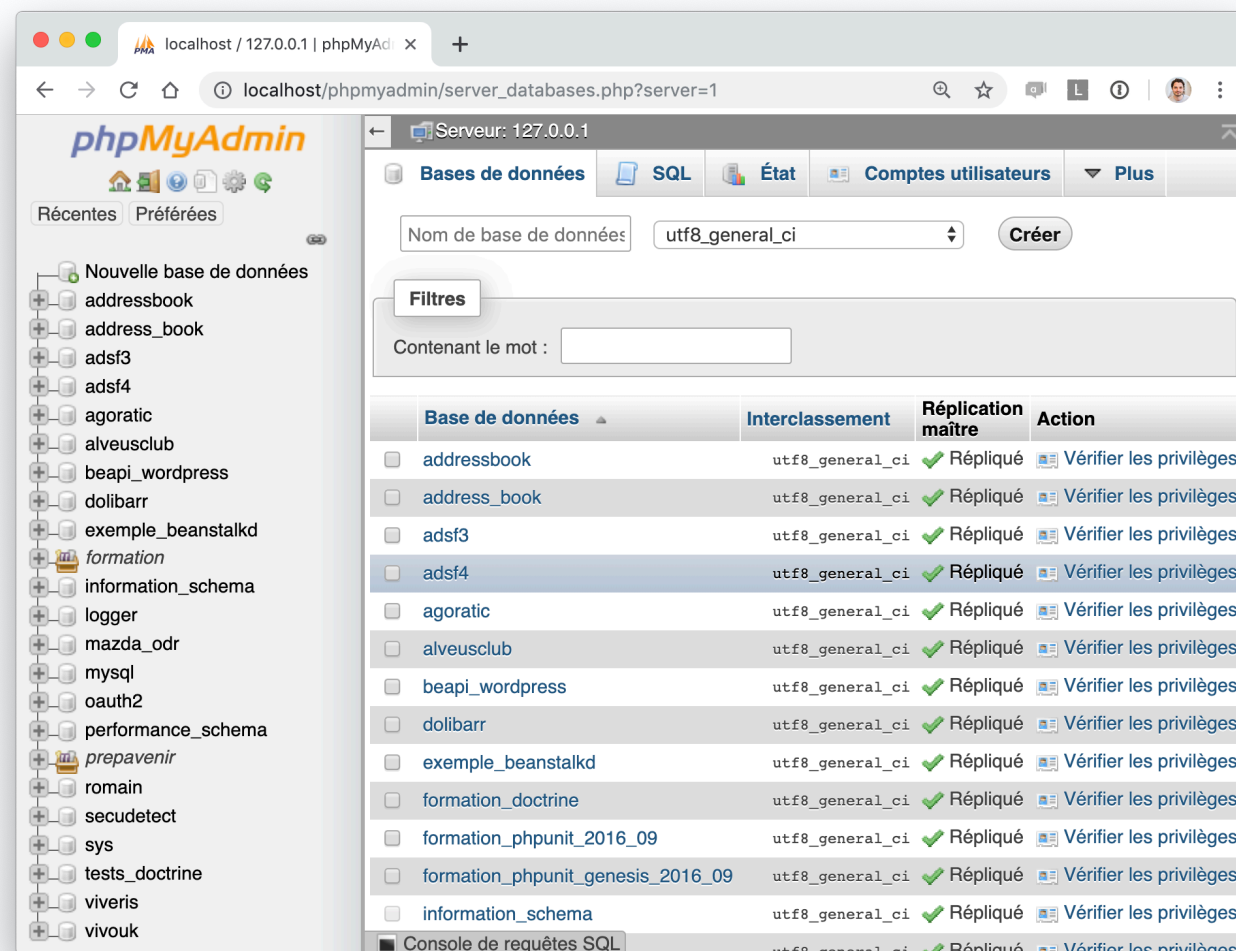


- Pour créer sa base de données on utilise en général des outils graphiques
 - phpMyAdmin est un outil PHP
 - MySQL Workbench est une application à installer avec la possibilité de créer via des schémas

MySQL - Créer sa base de données



- Pour créer sa base de données on utilise en général des outils graphiques
 - phpMyAdmin est un outil PHP
 - MySQL Workbench est une application à installer avec la possibilité de créer via des schémas





▸ Insertion

```
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)
```

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, ...)  
VALUES ('valeur 1', 'valeur 2', ...)
```

▸ Exemple

```
INSERT INTO contact (prenom, nom)  
VALUES ('Romain', 'Bohdanowicz')
```



- Suppression

```
DELETE FROM table  
WHERE condition
```

- Exemple

```
DELETE FROM contact  
WHERE id = 2
```



- Mettre à jour

```
UPDATE table  
SET nom_colonne_1 = 'nouvelle valeur'  
WHERE condition
```

- Exemple

```
UPDATE contact  
SET prenom = 'Romain'  
WHERE id = 3
```



▸ Recherche

```
SELECT nom_du_champ  
FROM nom_du_tableau  
WHERE condition
```

▸ Exemple

```
SELECT id, prenom, nom  
FROM contact  
WHERE prenom LIKE 'R%'
```



- Pour interagir avec une base MySQL en PHP on peut utiliser 2 APIs :
 - mysqli
Interface procédurale ou objet
 - PDO
Interface objet / abstraction pour d'autres types de SGBDR
- Historiquement il existait une API appelé mysql mais elle a disparu lors de l'arrivée de PHP 7



- Se connecter

```
$connection = mysqli_connect('localhost', 'root', '34fb4d', 'addressbook',  
'3306');  
  
if (!$connection) {  
    echo 'Erreur de connection';  
}
```

- La fonction *mysqli_connect* retourne soit un objet `mysqli`, soit une `false` si un problème a eu lieu lors de la connection
- L'objet `mysqli` représente la connection et devra être passé à toutes les fonctions suivantes



- Ecrire en base

```
$sql = <<<SQL
INSERT INTO contact (prenom, nom) VALUES ('Romain', 'Bohdanowicz')
SQL;

$result = mysqli_query($connection, $sql);

if ($result) {
    echo 'Enregistrement inséré';
}
```

- Pour les opérations d'écritures (INSERT, UPDATE, DELETE) mais aussi structurelles (CREATE, ALTER...) la fonction *mysqli_query* va retourner un booléen, soit true si la requête a fonctionné, false sinon
- Si la requête contient un id auto-incrémenté, on peut le récupérer via la fonction *mysqli_insert_id*

```
echo 'Dernier id : ' . mysqli_insert_id($connection);
```



- Retrouver une liste d'enregistrements

```
$sql = <<<SQL
SELECT prenom, nom FROM contact
SQL;

$result = mysqli_query($connection, $sql);
$contacts = mysqli_fetch_all($result, MYSQLI_ASSOC);

/*
[
    ['prenom' => 'Steve', 'nom' => 'Jobs'],
    ['prenom' => 'Bill', 'nom' => 'Gates'],
];
*/
```

- Le retour de *mysqli_query* peut être soit un objet *mysqli_result* soit false (en cas d'erreur)
- Pour transformer cet objet en tableau on utiliser *mysqli_fetch_all* (MYSQLI_ASSOC permet de dire que chaque enregistrement doit être mis dans le tableau sous forme de tableau associatif)



- Retrouver un d'enregistrement

```
$sql = <<<SQL
SELECT prenom, nom FROM contact WHERE id = 1
SQL;

$result = mysqli_query($connection, $sql);
$contact = mysqli_fetch_assoc($result);

/*
['prenom' => 'Steve', 'nom' => 'Jobs']
*/
```

- Pour retrouver un enregistrement on utilise la fonction *mysqli_fetch_assoc*



- Fermer la connection

```
mysqli_close($connection);
```

- Une fois le code terminé on ferme la connection avec *mysqli_close*



Architecture MVC

Architecture MVC - Introduction



- Afin de permettre d'obtenir du code plus réutilisable et plus maintenable, on peut mettre en place quelques bonnes pratiques au niveau de l'architecture
- L'architecture la plus répandue en PHP s'appelle MVC, pour Model View Controller (Modèle Vue Contrôleur)
- On appelle cela un Design Pattern (Patron de conception), c'est à dire une réponse à une problématique courante de programmation, établie comme étant une bonne pratique

Architecture MVC - Mauvais exemple



- Mauvais exemple
 - Mélange du code HTML avec le code d'accès au données
 - Pas de réutilisation possible du code SQL ou HTML

```
<?php
$connection = mysqli_connect('localhost', 'root', '', 'addressbook');
$query = 'SELECT prenom FROM contact';
$result = mysqli_query($connection, $query);
?>
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
  <?php foreach (mysqli_fetch_all($result, MYSQLI_ASSOC) as $contact) : ?>
    <p><?=$contact['prenom']?></p>
  <?php endforeach; ?>
</body>
</html>
<?php
mysqli_close($connection);
```

Architecture MVC - Séparer la logique applicative



- Séparer la logique applicative du rendu
 - Dans cet exemple le code PHP après `mysqli_close` ne concerne que le rendu (c'est à dire ici la création du HTML)
 - Le code du dessus est la logique applicative de la page (récupérer les infos en DB, valider les entrées, envoyer un email...)

```
<?php
$connection = mysqli_connect('localhost', 'root', '', 'addressbook');
$query = 'SELECT prenom FROM contact';
$result = mysqli_query($connection, $query);
$contacts = mysqli_fetch_all($result, MYSQLI_ASSOC);
mysqli_close($connection);
?>
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
  <?php foreach ($contacts as $contact) : ?>
    <p><?=$contact['prenom']?></p>
  <?php endforeach; ?>
</body>
</html>
```

Architecture MVC - Séparer le code métier (Modèle)



- Séparer le code métier (Modèle)
 - Ici on va chercher à rendre le code métier plus réutilisable
 - Les fonctions *dbConnect*, *dbGetAllContacts* et *dbClose* pourront être réutilisées sur d'autres pages

```
<?php
require_once 'model.php';
$connection = dbConnect();
$contacts = dbGetAllContacts($connection);
dbClose($connection);
?>
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
<?php foreach ($contacts as $contact) : ?>
  <p><?=$contact['prenom']?></p>
<?php endforeach; ?>
</body>
</html>
```

Architecture MVC - Séparer le code métier (Modèle)



- Le fichier *model.php* ressemblera à :

```
<?php
function dbConnect(): mysqli {
    $host = 'localhost';
    $user = 'root';
    $pass = '';
    $db = 'addressbook';
    return mysqli_connect($host, $user, $pass, $db);
}

function dbGetAllContacts(mysqli $connection) {
    $query = 'SELECT prenom FROM contact';
    $result = mysqli_query($connection, $query);
    return mysqli_fetch_all($result, MYSQLI_ASSOC);
}

function dbClose(mysqli $connection) {
    mysqli_close($connection);
}
```


Architecture MVC - Séparer le rendu (Vue)



- Séparer le rendu (Vue)
 - La génération du HTML pourrait également être faite dans un fichier séparé, la rendant ainsi réutilisable (page qui affiche tous les contacts ou bien issue d'une recherche...)

```
<?php
require_once 'model.php';
$connection = dbConnect();
$contacts = dbGetAllContacts($connection);
dbClose($connection);
include 'templates/contacts-list.phtml';
```

Architecture MVC - Séparer le rendu (Vue)



- Séparer le rendu (Vue)
 - La génération du HTML pourrait également être faite dans un fichier séparé, la rendant ainsi réutilisable (page qui affiche tous les contacts ou bien issus d'une recherche...)

```
<?php
require_once 'model.php';
$connection = dbConnect();
$contacts = dbGetAllContacts($connection);
dbClose($connection);
include 'templates/contacts-list.phtml';
```

- Le fichier template (vue)

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Exemple</title>
</head>
<body>
<?php foreach ($contacts as $contact) : ?>
  <p><?=$contact['prenom']?></p>
<?php endforeach; ?>
</body>
</html>
```

Architecture MVC - Séparer le rendu (Vue)



- On abouti ainsi à l'architecture MVC, on chaque fichier à sa responsabilité :
 - Le Modèle (Model)
 - Accès aux données
 - Validation des données
 - La Vue (View)
 - Echo
 - Boucles ou rendu conditionnels
 - Appel à des fonctions de formatages (date, nombres...)
 - Le contrôleur (Controller)
 - La logique de la page
 - Le chef d'orchestre qui fera les appels au modèle et la la vue
 - La gestion des erreurs
 - La gestion du code HTTP (Superglobales, erreurs 404...)
- Tous les frameworks PHP (Symfony, Zend, Laravel...) utilisent l'architecture MVC



Formulaires



- Les formulaires permettent à l'utilisateur de soumettre des données qui pourront ensuite être traitées côté serveur, exemple :
 - Recherche
 - Inscription
 - Upload
 - Login
- La balise HTML `<form>` va au moment de la soumission créer une requête HTTP de type GET ou POST qui pourra ensuite
- Pour que les valeurs soient transmises dans la requête il faut obligatoirement définir les attributs `name` des champs `input`, `select` et `textarea`
- PHP va créer des tableaux associatifs `$_GET`, `$_POST`, `$_FILES` avec les données à l'intérieur, on appelle ces variables des superglobales (accessibles de n'importe où)



- Les formulaires de recherche utilise la méthode GET et la superglobale \$_GET

```
<?php
$capitales = ["France" => "Paris", "Allemagne" => "Berlin",
    "Espagne" => "Madrid", "Belgique" => "Bruxelles"];

if (isset($_GET["pays"])) {
    $paysUrl = $_GET["pays"];

    if (isset($capitales[$paysUrl])) {
        $capitaleTrouvee = $capitales[$paysUrl];
    }
}
?>
<form method="GET">
    Pays :
    <select name="pays">
        <?php foreach ($capitales as $pays => $capitale) : ?>
            <option><?php echo $pays; ?></option>
        <?php endforeach; ?>
    </select>
    <button>Recherche</button>
</form>
<?php if (isset($capitaleTrouvee)) : ?>
    La capitale de <?=$paysUrl?> est <?=$capitaleTrouvee?>
<?php endif; ?>
```

Formulaires - Inscription / Login



- Les formulaires d'inscription ou de login utilisent la méthode POST et la superglobale `$_POST`

```
<?php
if (isset($_POST["prenom"], $_POST["nom"])) {
    $prenom = $_POST["prenom"];
    $nom = $_POST["nom"];

    // TODO Inscription
}
?>
<form method="POST">
    <p>Prénom : <input type="text" name="prenom"></p>
    <p>Nom : <input type="text" name="nom"></p>
    <button>Inscription</button>
</form>
```



- Les formulaires avec upload de fichier doivent être POST et utiliser l'attribut `enctype="multipart/form-data"`
- La superglobale `$_FILES` contient les infos : nom, type, tmp_name, error, size
- Il faut déplacer le fichier indiquer à tmp_name le temps de la requête dans un dossier permanent

```
<?php
if (isset($_FILES["fichier"])) {
    var_dump($_FILES["fichier"]);
    // [
    //     'name' => 'exemple.pdf',
    //     'type' => 'application/pdf',
    //     'tmp_name' => '/private/var/folders/nb/b3_dc91n22d5_lhsvyc07xkc0000gn/
T/php80Br7',
    //     'error' => 0,
    //     'size' => 153244,
    // ]
    move_uploaded_file($_FILES["fichier"]["tmp_name"], './upload/exemple.pdf');
}
?>
<form method="POST" enctype="multipart/form-data">
    <p>Photo : <input type="file" name="fichier"></p>
    <button>Upload</button>
</form>
```




- Les formulaires avec upload de fichier doivent être POST et utiliser l'attribut `enctype="multipart/form-data"`
- La superglobale `$_FILES` contient les infos : nom, type, tmp_name, error, size
- Il faut déplacer le fichier indiquer à tmp_name le temps de la requête dans un dossier permanent

```
<?php
if (isset($_FILES["fichier"])) {
    var_dump($_FILES["fichier"]);
    // [
    //     'name' => 'exemple.pdf',
    //     'type' => 'application/pdf',
    //     'tmp_name' => '/private/var/folders/nb/b3_dc91n22d5_lhsvyc07xkc0000gn/
T/php80Br7',
    //     'error' => 0,
    //     'size' => 153244,
    // ]
    move_uploaded_file($_FILES["fichier"]["tmp_name"], './upload/exemple.pdf');
}
?>
<form method="POST" enctype="multipart/form-data">
    <p>Photo : <input type="file" name="fichier"></p>
    <button>Upload</button>
</form>
```

Formulaires - Champs multiples



- Si un champs propose plusieurs valeurs (checkbox, select ou file multiple) il suffixer l'attribut name avec des []
- En PHP les données seront récupérées dans un tableau

```
<?php
if (isset($_POST['couleurs'])) {
    var_dump($_POST['couleurs']); // [ 'Rouge', 'Bleu' ]
}
?>
<form method="POST" enctype="multipart/form-data">
    <label><input type="checkbox" name="couleurs[]" value="Rouge"> Rouge</label>
    <label><input type="checkbox" name="couleurs[]" value="Vert"> Vert</label>
    <label><input type="checkbox" name="couleurs[]" value="Bleu"> Bleu</label>
    <button>Go</button>
</form>
```



Session & Cookies

Session & Cookies - Introduction



- Certaines données sont liées à un utilisateur et ne peuvent être stockées en base de donnée
- Deux façons de les stocker :
 - Cookies
 - Session
- La session est un fichier unique, associé à un utilisateur (données User, panier...)
- Les cookies sont stocké côté client, il faut y mettre des données non sensibles (langue, thème...)

Session & Cookies - Cookies



- Pour manipuler les cookies on utilise la superglobale `$_COOKIE`
- Les cookies sont associés à un domaine, on une date d'expiration (ou expirent à la fermeture du navigateur)
- On défini le cookie dans un entête HTTP, il faut donc un aller-retour vers le serveur pour pouvoir lire sa valeur

```
<?php
echo $_COOKIE['langue']; // FR

// défini un cookie valable 30 jours
setcookie($_COOKIE['langue'], 'FR', time() + (60 * 60 * 24 * 30));
```



- La session est un concept associant un cookie d'identification à un fichier côté back
- Le mécanisme peut être mis en oeuvre simplement en PHP avec la variable `$_SESSION` et les fonctions de sessions
<https://www.php.net/manual/fr/ref.session.php>
- Pour créer une session on utilise la fonction `session_start`, ce qui aura pour effet de créer un fichier de session sur le serveur associé à un identifiant
- Si un identifiant a été envoyé par le client
- La session expire au bout de 24 minutes sans activité ou bien à la fermeture du navigateur (configurable)

```
<?php
session_start();

$_SESSION['username'] = 'romain';
```



GD



- GD est une bibliothèque de gestion d'images écrite en C et accessible depuis PHP
- Elle permet la manipulation d'images aux formats
 - BMP
 - GIF
 - JPEG
 - PNG
 - WEBP
 - ...

GD - Manipulation d'une image



- On peut soit créer une image avec la fonction `imagecreate`
- Soit ouvrir une image existant avec les fonctions préfixées par `imagecreatefrom` (ex : `imagecreatefrompng`)

```
<?php
$im = imagecreate(100, 100);
$background_color = imagecolorallocate($im, 0, 0, 0);
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5, "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
```

- Les fonctions `imagepng`, `imagegif`, `imagejpeg`... permettent de créer l'image
- `imagedestroy` est à appeler pour libérer proprement la mémoire



- Pour écrire du texte avec GD on utilise la fonction `imagestring`

```
$text_color = imagecolorallocate($im, 233, 14, 91);  
imagestring($im, 1, 5, 5, "A Simple Text String", $text_color);
```

- Les paramètres sont :

- font

Peut être 1, 2, 3, 4, 5 pour les polices internes d'encodage Latin2 (où les plus grands nombres correspondent aux polices larges) ou n'importe quels identifiants de police de votre choix, enregistrées avec la fonction `imageloadfont()`.

- x et y

Coordonnées du point en haut à gauche

- string

La chaîne de caractères à écrire.

- color

Un identificateur de couleur créé avec `imagecolorallocate()`.



- On peut redimensionner une image avec les fonctions `imagecrop` ou `imagescale`

```
<?php
$im = imagecreatefrompng('example.png');
$size = min(imagesx($im), imagesy($im));
$im2 = imagecrop($im, ['x' => 0, 'y' => 0, 'width' => $size, 'height' => $size]
);
if ($im2 !== FALSE) {
    imagepng($im2, 'example-cropped.png');
    imagedestroy($im2);
}
imagedestroy($im);
```

GD - Changer de police



- Pour charger un fichier de police on utilisera la fonction `imageloadfont`

```
<?php
// Création d'une nouvelle image
$im = imagecreatetruecolor(50, 20);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);

// Définit l'arrière-plan en blanc
imagefilledrectangle($im, 0, 0, 49, 19, $white);

// Charge la police GD et écrit 'Bonjour !'
$font = imageloadfont('./04b.gdf');
imagestring($im, $font, 0, 0, 'Bonjour !', $black);

// Affichage sur le navigateur
header('Content-type: image/png');

imagepng($im);
imagedestroy($im);
```