

### Plan

- Internet
- HTML/CSS
- JavaScript/DOM
- AJAX/XMLHttpRequest
- Framework : frontend/backend
  - AngularJS/BootStrap
  - Framework PHP
- Langages des serveurs web
- SSL
- XML/JSON
- Architecture multi-tiers
  - AMP
  - J2EE
  - dotNET
- Framework de persistance et de mapping ORM
- Objets distribués
  - RMI
  - CORBA
- REST

### *qu'est ce que Internet*

- On compare souvent Internet à une *autoroute*.
- Par certains côtés, il ressemble davantage au réseau téléphonique.
  - Chaque abonné dispose d'un numéro unique permettant de le joindre.
  - Dans Internet, on appelle ce numéro « l'adresse IP » :
    - pour `www.insa-centrevaldeloire.fr`, par exemple, c'est le **195.221.38.254**
  - Chaque abonné est identifié par un nom qui lui est propre.
    - Dans Internet, on l'appelle « le nom de domaine ».
    - DNS, est le service qui associe à un nom de domaine une adresse IP

### *les protocoles TCP/IP*

- L'infrastructure matérielle n'a qu'un seul intérêt : permettre le fonctionnement du protocole TCP/IP.
  - À bien des égards, le TCP/IP **est** l'Internet.
  - Mais qu'est-ce qu'un protocole ?

Le protocole TCP/IP permet aux ordinateurs du réseau de communiquer

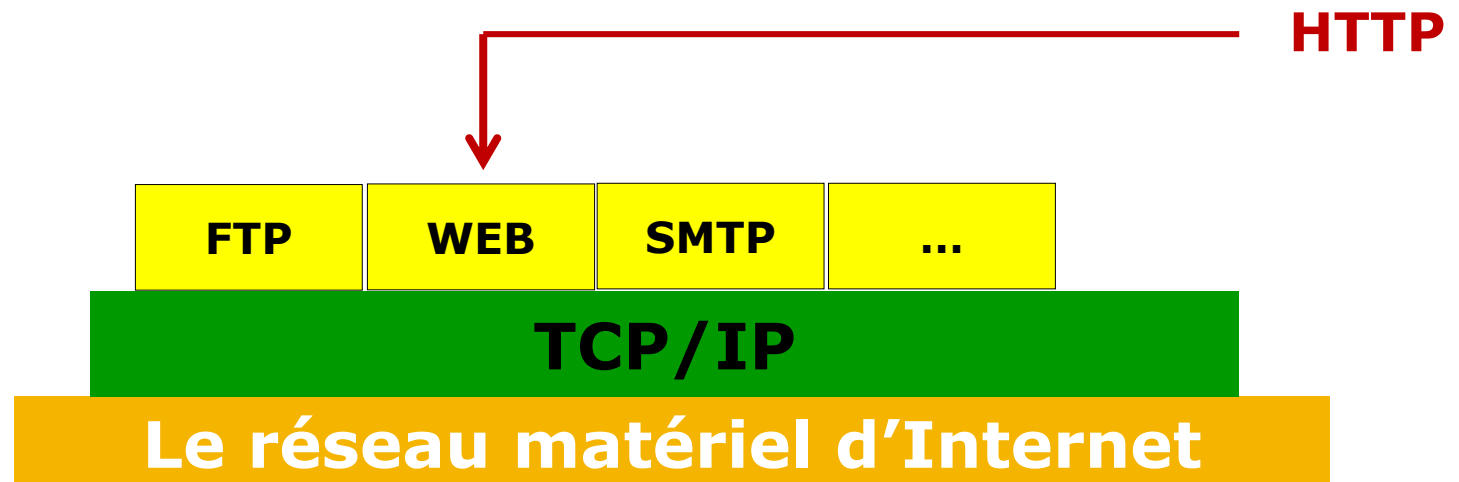


TCP/IP

Le réseau matériel d'Internet

### *le service web*

- Le Web repose sur un protocole appelé HTTP inventé par Tim Berners-Lee vers 1990-1991.
  - L'acronyme signifie « **Hyper Text Transfer Protocol** ».
  - Le HTTP est un protocole de type « client-serveur ».



### ***Le protocole HTTP***

- Le Web repose sur le protocole HTTP ;
  - C'est une application de type client-serveur ;
  - Le client est votre navigateur préféré, Firefox ou Internet Explorer
  - Le serveur est le logiciel qui vous accueille quand vous naviguez :
    - Par exemple, **Apache, IIS, Tomcat**

### ***Le protocole HTTP***

- Un protocole de type client/serveur
- **Le client:**
  - effectue une demande de service auprès du serveur (requête)
  - initie le contact (parle en premier), ouvre la session
- **Le serveur:**
  - est la partie de l'application qui offre un service est à l'écoute des requêtes clientes répond au service demandé par le client (réponse)



### ***Le protocole HTTP***

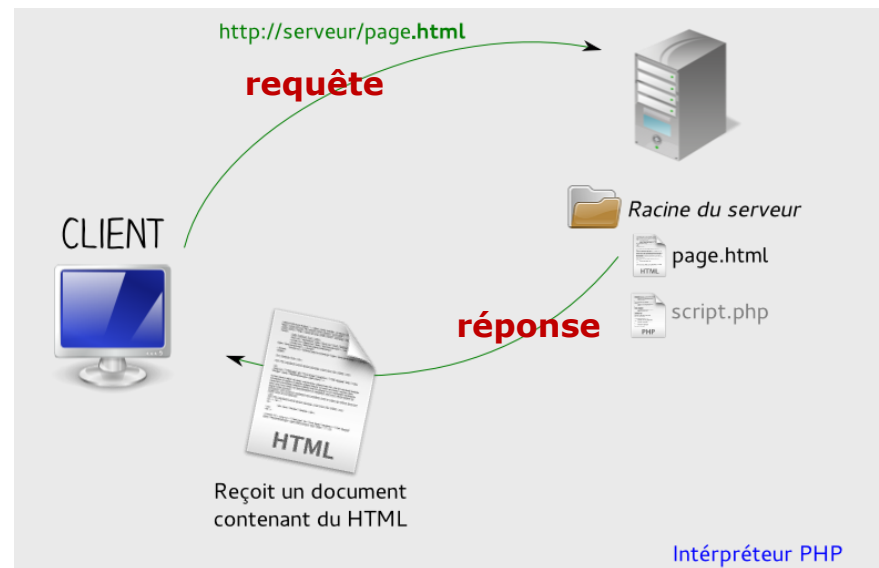
- Protocole standard
  - Dialogue entre le client Web/Serveur Web.
  - Sur un port spécifié (par défaut 80)
- Protocole de type déconnecté
  - Le serveur ne garde pas de contexte (pas de session utilisateur)

le protocole HTTP est le fondement du Web.



### Une requête HTTP

- En cliquant sur le lien, vous amorcez la séquence suivante :
  - Votre navigateur s’empare de l’adresse associée au lien et lance une requête à l’ordinateur correspondant sur Internet (« Envoie-moi tel fichier, voici mon adresse de retour »).
  - À destination, le serveur prend connaissance de la requête, localise le fichier demandé et l’expédie au demandeur pour remplir la commande.
- Fondamentalement, ce n’est pas plus compliqué que ça : un va-et-vient de **requêtes et de réponses entre logiciels clients et serveurs**.





### *Une requête HTTP*

- Le client envoie au serveur (requête HTTP)
  - Une demande de document :

**GET /index.html HTTP/1.0**

- Des informations sur sa configuration (en-tête optionnel):
    - Nom, numéro de version.
    - type de documents supportés ...
    - Données supplémentaires (facultatif)
- Exemple de requête

**PUT /page1.html HTTP/1.0 ↵**

**User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.7) Gecko/20050414 Firefox/1.0.3 ↵**

**Ligne blanche ↵**

### Une réponse HTTP

- Le serveur répond alors au client par
  - L'envoi d'une ligne d'état contenant : la version HTTP, le code d'état et sa description.  
**HTTP/1.0 200 OK**
  - L'envoi d'en-tête
    - Informations concernant le serveur et le document demandé
  - Les données formant le document demandé
  - Le serveur coupe la connexion

- Exemple de réponse : **HTTP/1.x 200 OK** ↵  
**Date:** Mon, 25 Apr 2005 04:25:17 GMT ↵  
**Server:** Apache/2.0.46 (Red Hat) ↵  
**X-Powered-By:** PHP/4.3.2 ↵  
**Content-Type:** text/html ↵  
**Ligne blanche** ↵  
**<html>**  
**<head><title> ...**

### *Commandes HTTP*

- Type de commandes HTTP supportées
  - **GET** : les données du client se trouvent dans l'URL.
  - **POST** : les données du client se trouvent dans la requête HTTP.
  - **HEAD** : Idem GET sauf que le serveur n'envoie aucune données (en-tête seulement)
  - **LINK, PUT, DELETE** .
- Trame HTTP:

Ligne de commande : commande, url, version de protocole

Entête de requête

Ligne vide

Corps de requête

### ***HTTP et Sessions***

- HTTP est un protocole sans état ( gestion de session absente):
  - Le client ouvre la connexion
    - Le serveur ferme la connexion
  - 1 transaction = 1 ressource transférée (v 1.0)
    - Aucune information gardée entre deux transactions
  - Le serveur "oublie" le client après chaque transaction
- Services nécessitant une gestion de session:
  - Login, l'authentification doit être faite une seule fois pour la durée de la session
  - Commerce électronique, gestion de panier de l'utilisateur
  - ...

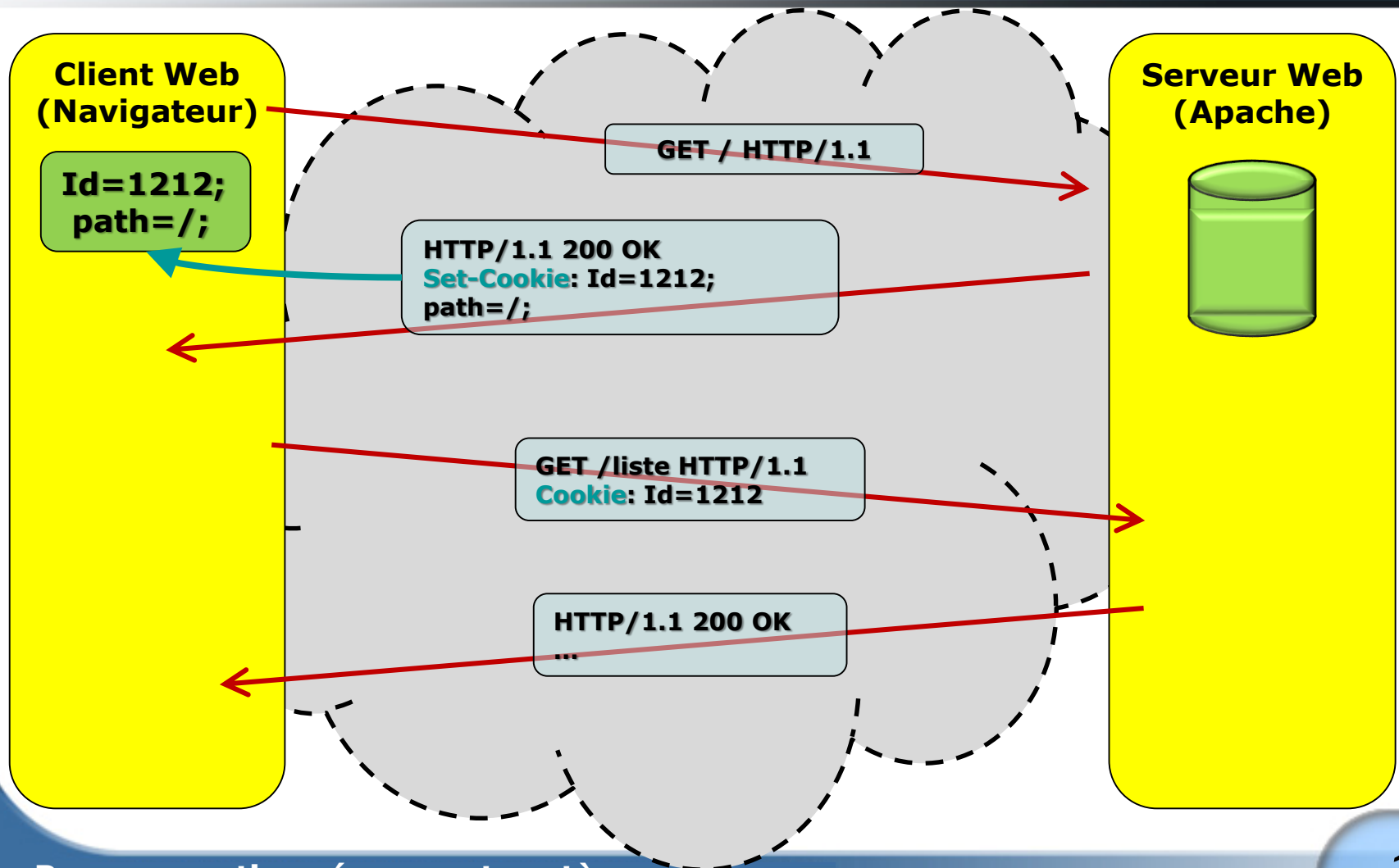


**cookie**

### ***Cookie HTTP***

- But :
  - Éviter que le serveur « oublie le client »
  - Maintenir un « mode connecté » (= session)
  - Rendre transparent un échange client / serveur
  - Exemple e-commerce : ajouter des articles au panier
- Serveur :  
**Set-Cookie: *var=val*; expires=*date*; path=*chemin*;  
domain=*domaine*]**
- Client :  
**Cookie: *var=val*;**

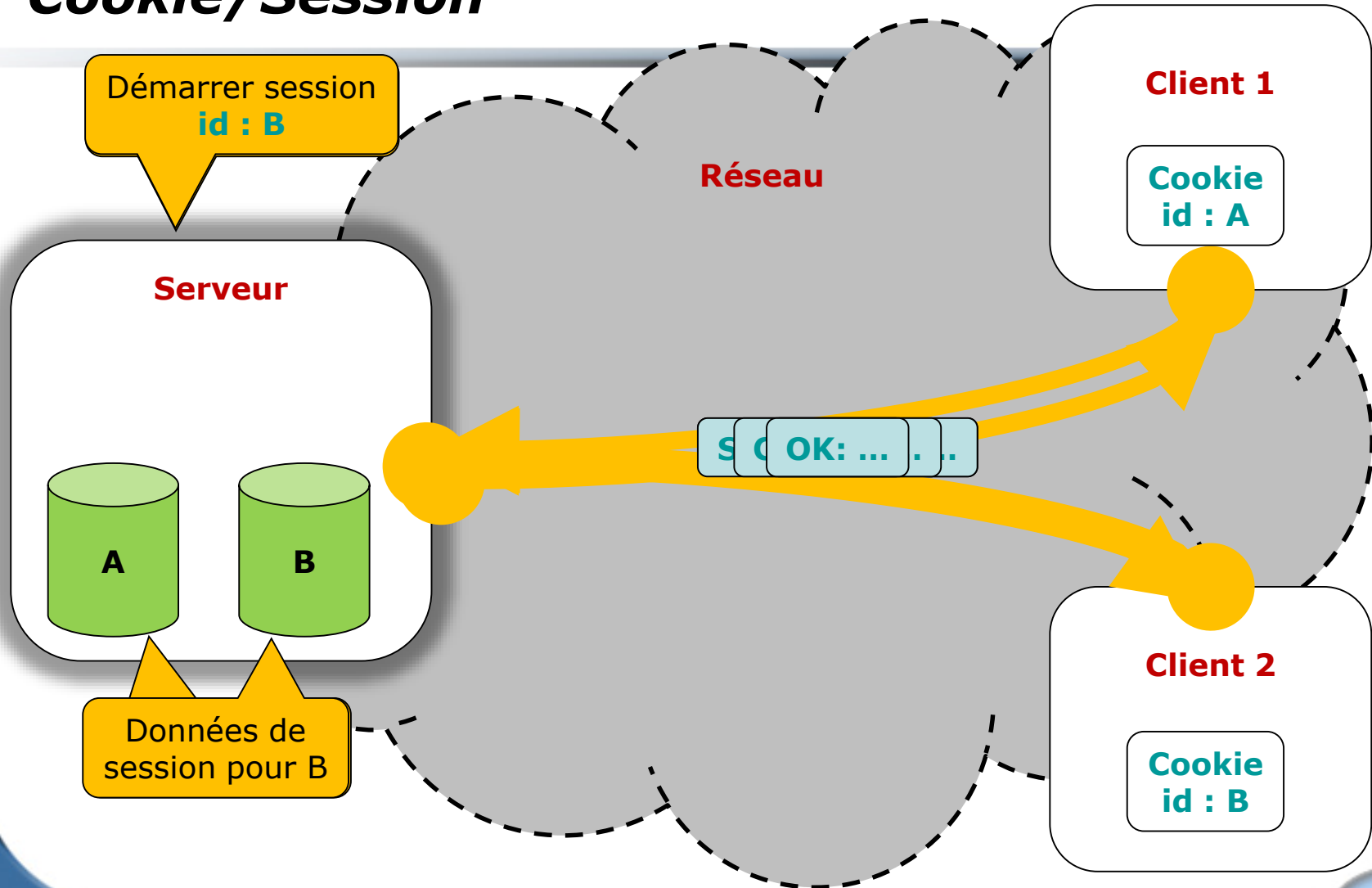
### *Cookie, principes des échanges*



### ***Cookie/Session pour simuler un mode connecté***

- Stockage sur le serveur de données associées à un client particulier
- Nécessite une identification unique pertinente et persistante des clients
  - Identifiant de session (MD5 128bits / SHA-1 160bits)
  - Persiste par paramètre d'URL ou cookie
- Évite l'échange permanent de données (en dehors de l'identifiant)
- Nécessite la linéarisation des variables pour leur stockage (fichier, BD, personnalisé)
- **Simule un mode HTTP connecté**

### Cookie/Session





### ***HTML***

- HTML: HyperText Markup Language
  - Structure arborescente
  - Balise de plus haut niveau :html
  - Puis deux parties head et body
  - Balises de contenu : h1, h2, div, pa, a, span, form,ul,li,input
- 1993 : HTML 1.0 . Langage de présentation de contenu
- 1995-1997: HTML 2.0
- 1997 : HTML 3.2 et 4.0
- 2000 : XHTML 1.0

### Un fragment de code HTML

- En rouge, entre crochets, vous voyez deux paires de balises `<u>` et `</u>`.
  - En HTML 3.2 ces balises marquent respectivement le début et la fin du soulèvement.
  - Et, de fait, sur la ligne interprétée, on voit que le **navigateur** a souligné les mots chacun et chacune comme le demande le marquage.

`<font size=+3>`

Bienvenue à

`<u>`chacun`</u>`

et à `<u>`chacune`</u>`  
d'entre vous!

`</font>`

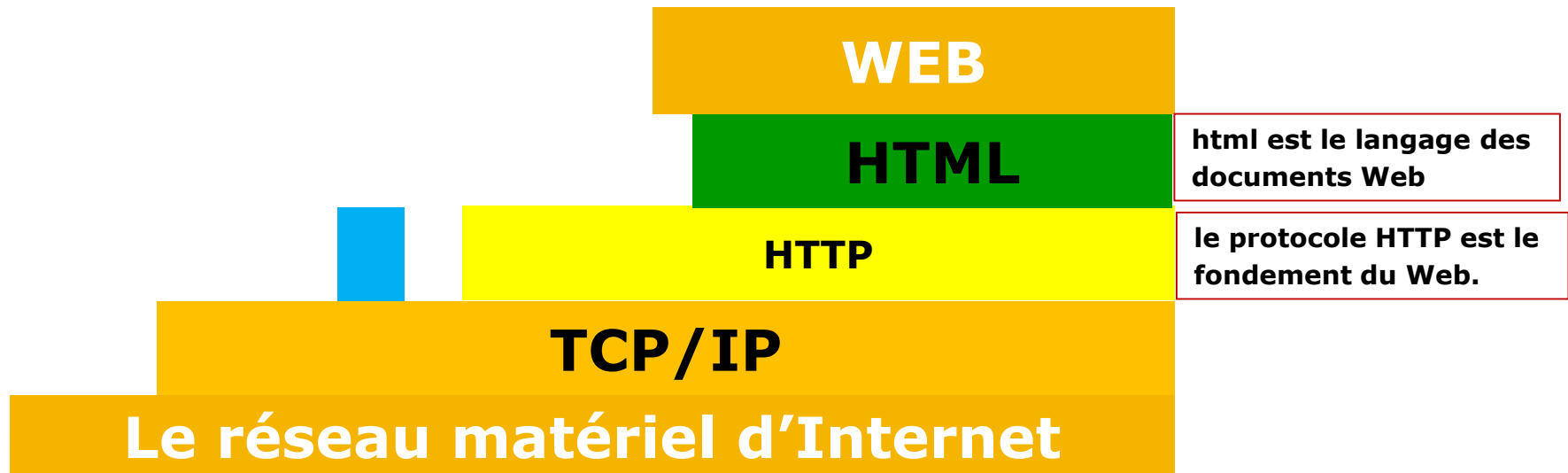
Chrome, Firefox, IE

Bienvenue à chacun et à chacune d'entre vous !

### *Code HTML plus complet*

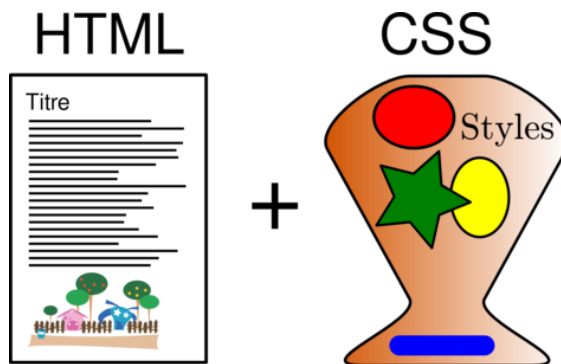
```
<html>
<body>
  <table border>
    <tr> <th>ID</th> <th>Name</th> <th>Department</th> </tr>
    <tr> <td>00128</td> <td>ABDALLAH</td> <td>Comp. Sci.</td> </tr>
  ....
</table>
  <form action="PersonQuery" method=get>
    Search for:
    <select name="persontype">
      <option value="student" selected>Student </option>
      <option value="instructor"> Instructor </option>
    </select> <br>
    Name: <input type=text size=20 name="name">
    <input type=submit value="submit">
  </form>
</body> </html>
```

### *Architecture web*



### *Les feuilles de style (feuilles CSS)*

- Elle consistait à séparer le plus possible, dans les pages Web, l'**apparence** et le **contenu**.
  - Toutes les indications concernant l'apparence ont été déplacées vers un document indépendant qu'on appelle une feuille de style, qui permet :
    - L'application de l'apparence des éléments de page associés aux balises
    - Le positionnement précis des éléments de la page
    - Une forme rudimentaire de superposition des éléments.



### *Les feuilles de style*

- CSS : Cascading Style Sheet
- 1996 : CSS 1.0
- Il est encapsulé dans une page HTML (ou dans un fichier lié) pour être interprété par le client.
- Les feuilles indiquent aux balises HTML leur comportement ou style



### Exemple CSS

```
<HTML>
  <HEAD>
    <TITLE>Le CSS</TITLE>
    <link rel="stylesheet" type="text/css" href="style.css">
  </HEAD>
  <BODY>
    <H1>Introduction</H1>
    ...
  </BODY>
</HTML>
```

.HTML

Le Résultat

Introduction

...

```
@charset "iso-8859-1";
```

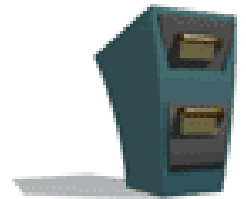
```
H1 {
```

Sélecteur simple

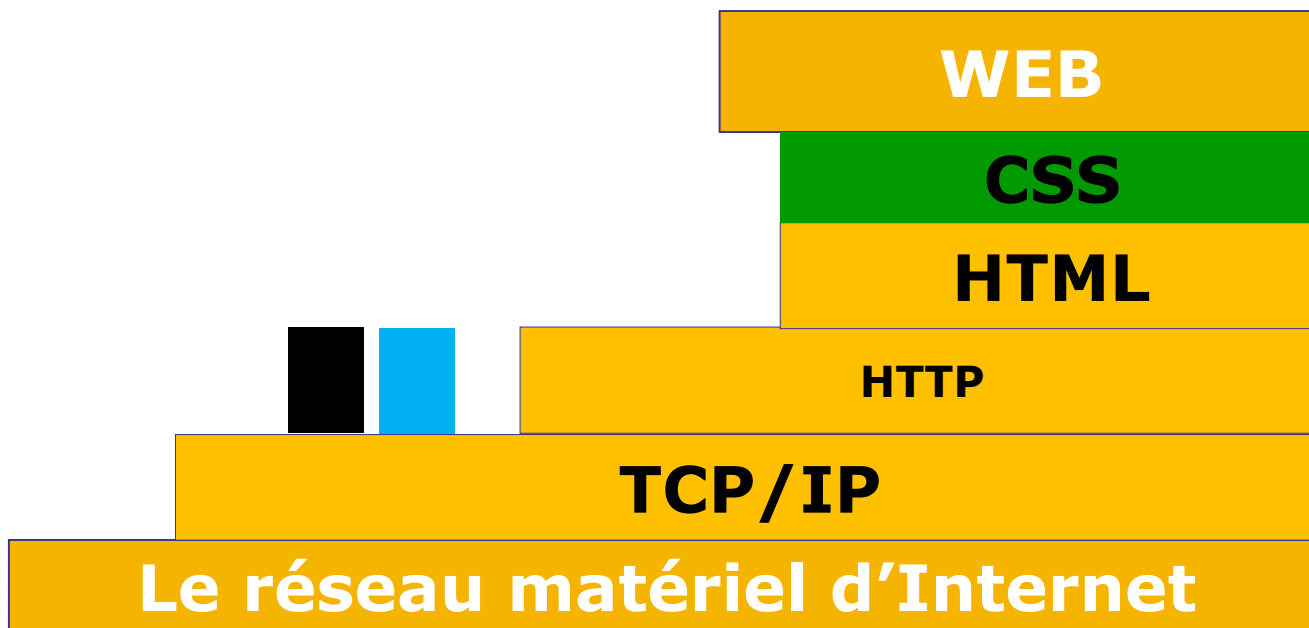
```
  color : red ;
```

```
}
```

style.css



### *Architecture web*



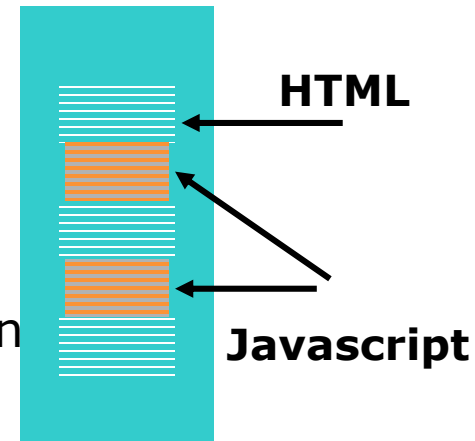
html est le langage des documents Web

le protocole HTTP est le fondement du Web.



### *Le web a évolué*

- Pression pour avoir un contenu dynamique
- Ajouter des possibilités au web
  - Son, vidéo, animation
- Programmation des pages web !
  - La plus réussie de ces formes de programmation est le **javascript**.
    - Quand le navigateur interprète le code HTML, il reconnaît les parties du texte qui sont des programmes rédigés en javascript.



### ***JAVASCRIPT et DOM***

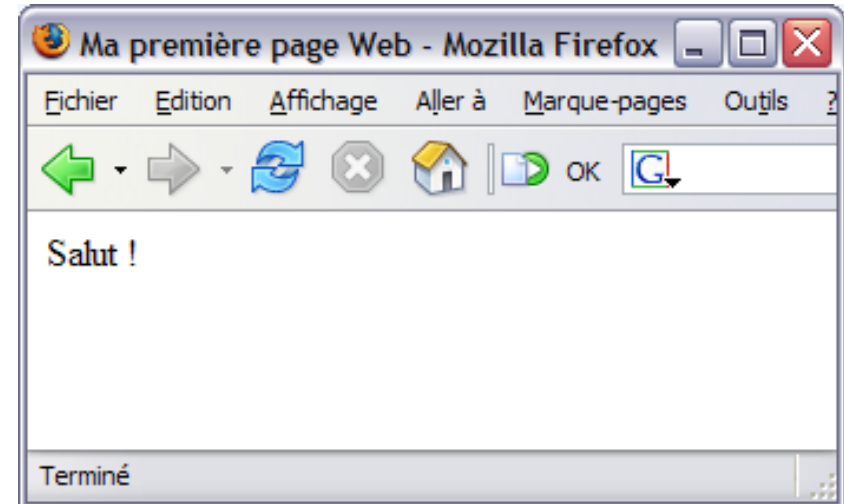
- **Objectif principal :**
  - introduire de l'interactivité avec les pages HTML
  - effectuer des traitements simples sur le poste de travail de l'utilisateur
- **Moyen :** introduire de petits programmes, appelés **SCRIPTS**, dans les pages HTML
- langage de programmation **structurée** qui concourt à **enrichir le HTML**
- JS est **intégré** complètement dans le **code HTML**
- JS contient des **gestionnaires d'événement**

### ***JAVASCRIPT et DOM***

- 1995: Naissance de **JavaScript**.
- **JS** est exécuté dans le navigateur
- W3C: normalisation DOM pour la manipulation des arbres HTML.
- **JS** est encapsulé dans une page HTML ou dans un fichier lié qui sera exécuté par le navigateur.
- On peut placer du code JS à 3 endroits :
  - Entre les balises `<SCRIPT>` et `</SCRIPT>`
  - Associé à une balise HTML qui gère un événement
  - Associé au pseudo-protocole *javascript:* dans une URL

### JAVASCRIPT et DOM

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



### *JavaScript*

- Les spécifications **ECMAScript** (**ES**) ont permis de pérenniser JavaScript
- JavaScript permet de contrôler quasiment tous les paramètres d'une page WEB
- C'est le seul langage, coté client, capable de changer dynamiquement l'aspect d'une page WEB
- Avec l'arrivée de l'objet XMLHttpRequest permettant le développement d'applications AJAX, JavaScript est devenu incontournable dans le développement d'interfaces WEB évoluées (WEB2.0)

### ***Les premiers « frameworks » JavaScript***

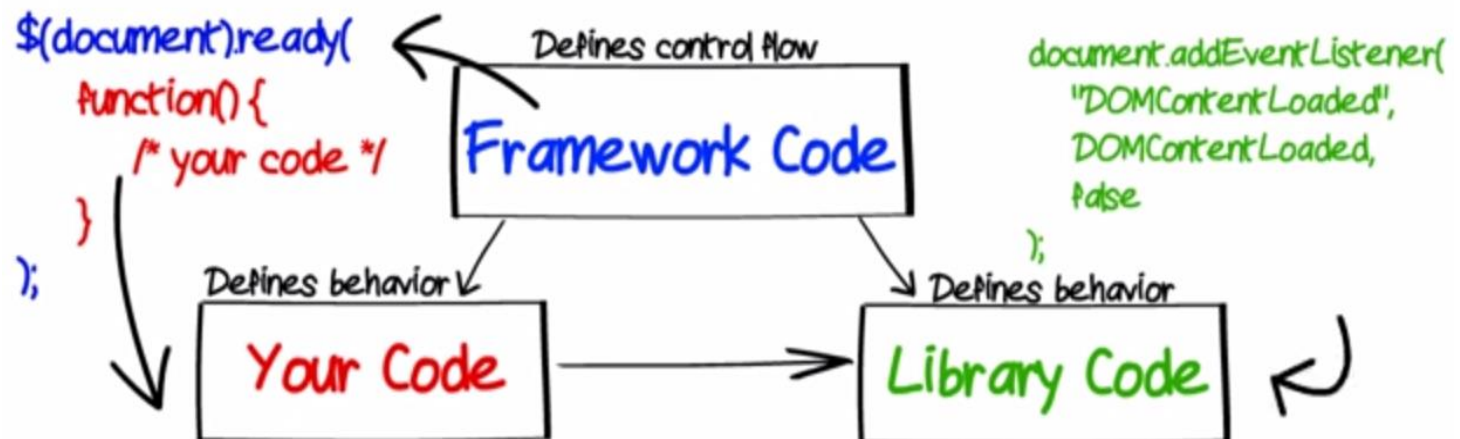
- Comme il était devenu difficile de coder du javascript pour tous les navigateurs, sont apparus des « Frameworks » permettant une spécification unique, indépendante du navigateur
  - PrototypeJS - [www.prototypejs.org](http://www.prototypejs.org)
    - [script.aculo.us](http://script.aculo.us)
  - Mootools - [mootools.net](http://mootools.net)
  - DoJo Toolkit - [www.dojotoolkit.org](http://www.dojotoolkit.org)
  - Yahoo UI - [developer.yahoo.com/yui/](http://developer.yahoo.com/yui/)
  - ExtJS - [www.extjs.com](http://www.extjs.com)
  - UIZE - [www.uize.com](http://www.uize.com)

### JavaScript, j'en profite pour

- Expliquer la différence entre un « framework » logiciel et une bibliothèque logicielle?
  - Fondamentalement, c'est celui qui est en charge du flot de contrôle et celui qui définit le comportement
  - Une image vaut .....:**

**Applait/applé**

#### Inversion of Control (IoC)



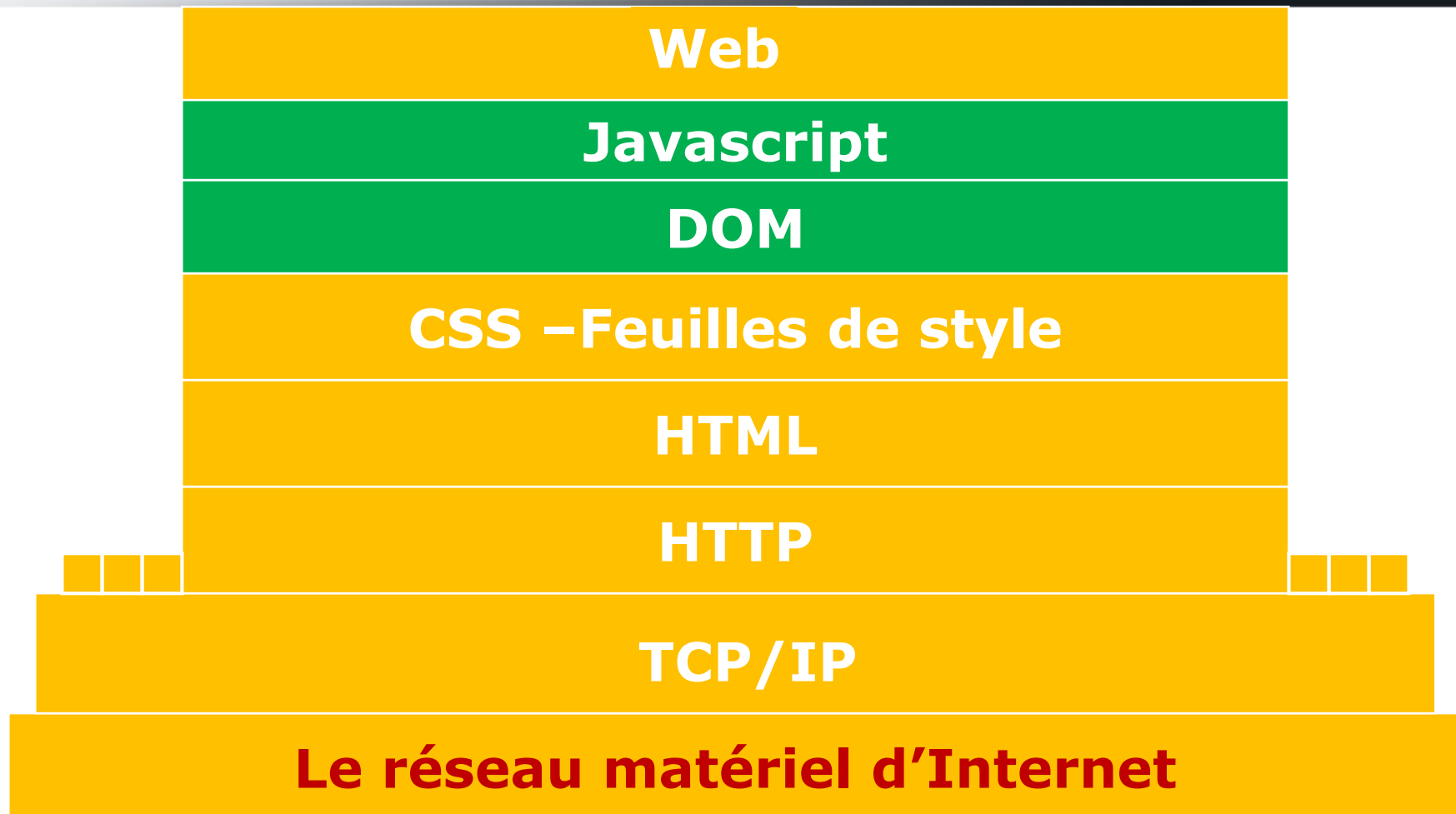
- Un framework c'est un « OS » applicatif
- Une bibliothèque, un ensemble de fonctions/méthodes/objets

### **DOM**

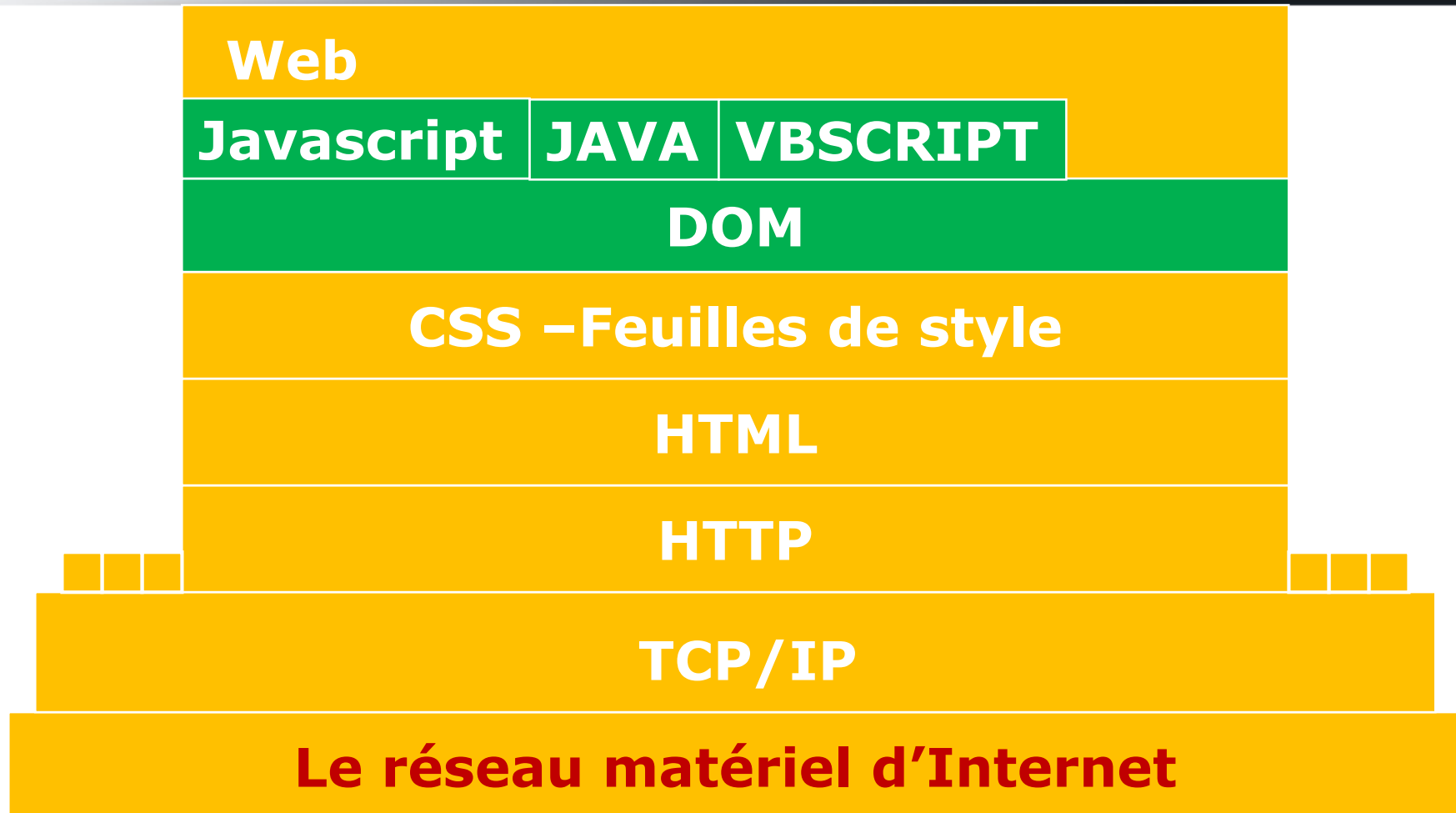
- DOM est une norme W3C indépendante de toute plate-forme et de tout langage de programmation.
- Selon DOM, un document est vu comme une **arborescence** avec une imbrication hiérarchique de ses composants.
- Pour JavaScript, DOM garantit l'accès aux composants d'une page Web.
- API (**Application Programming Interface**) pour la manipulation de HTML / XML :
  - Définit la structure logique des documents
  - Définit la façon d'y accéder, de la manipuler
    - ➔ Créer des documents
    - ➔ Parcourir leur structure
    - ➔ Ajouter, effacer, modifier des éléments
    - ➔ Ajouter, effacer, modifier leur contenu



### *Architecture web*



### *Architecture web*



### ***Interactivité***

- Le matériel d'ordinateur s'est standardisé ;
- Les pressions se sont faites nombreuses pour :
  - maîtriser l'apparence des pages Web;
  - ajouter des possibilités au Web :
    - le son, l'animation et le vidéo : le multimédia,
    - la programmation des pages Web,
    - **l'interactivité.**

### ***Interactivité***

- L'interactivité du Web repose sur la capacité « dynamique » du protocole HTTP ;
- Commençons donc par établir la distinction entre sites Web «statiques» et sites Web « dynamiques ».

### ***Le site web statique***

- Sur un tel site, les pages HTML envoyées ne changent pas.
  - Elles ont été créées à un moment donné par le concepteur du site Web et entreposées sur le disque du serveur.
  - Tous les usagers qui demandent une page donnée reçoivent exactement la même chose.
  - HTML + CSS

### *Le site web dynamique*

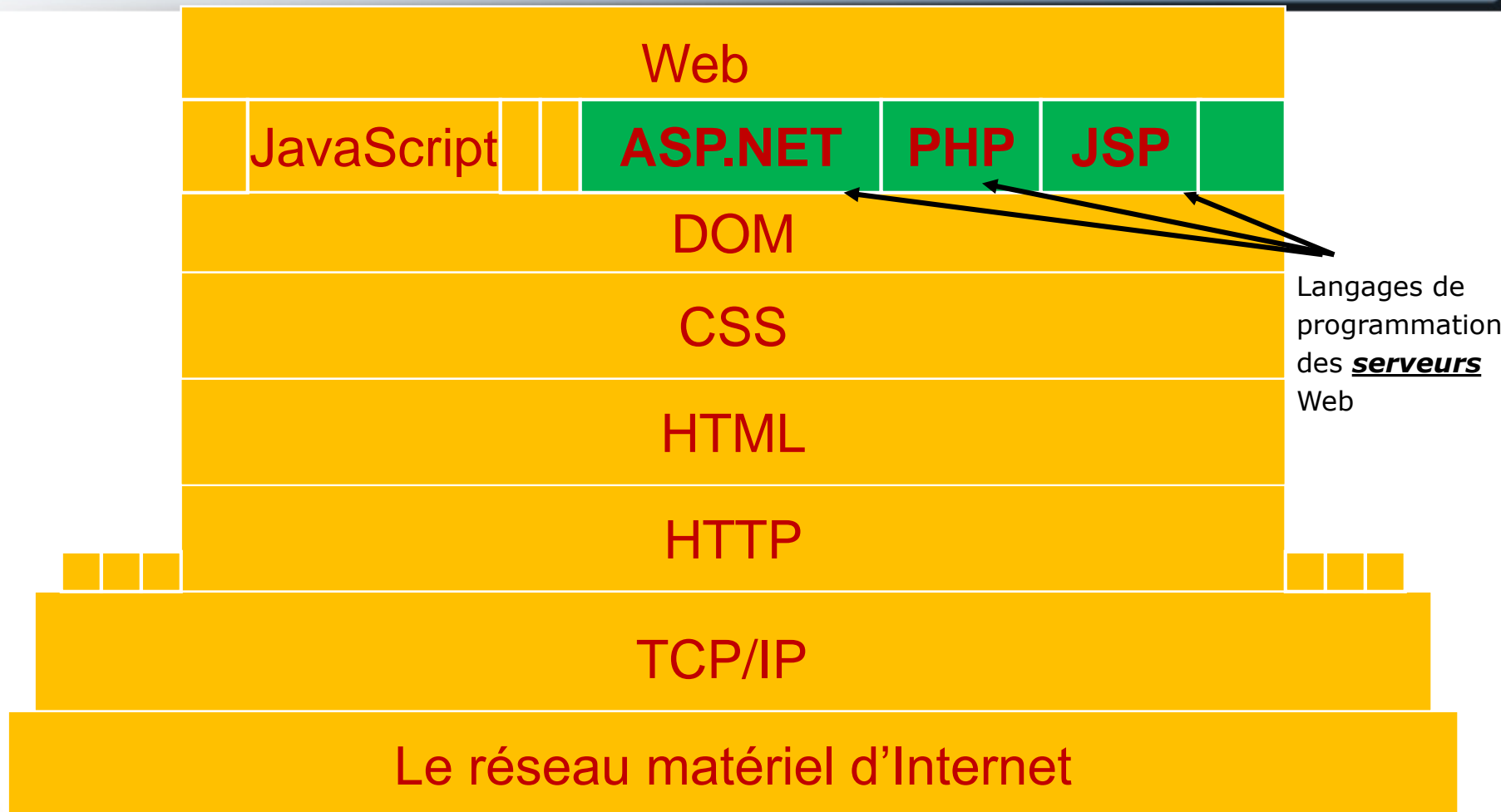
- Quand l'utilisateur demande une page php, par exemple, le serveur:
  - Lance l'exécution d'un programme qui cherche dans les bases de données, réunit les données qui lui sont nécessaires,
  - Génère une page HTML ad hoc qu'il transmet au client en réponse à sa demande.
- La page HTML complète n'existait pas avant la demande et elle a été créée de toutes pièces.



### ***Langages des serveurs web***

- Le programme s'exécute sur le **serveur**, À l'aide de langages comme :
  - le **ASP.NET** de Microsoft (active server pages), qui accède à une base de données;
  - le **PHP**, très populaire, surtout en milieu UNIX et Linux, associé à la base de données *open source* (code source libre) MySQL.
  - Le **JSP** , Java Server Pages, qui accède à une base de données;
  - etc...

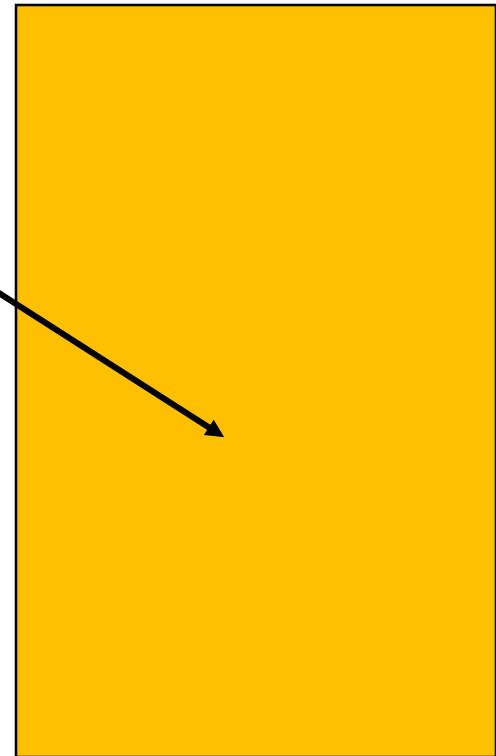
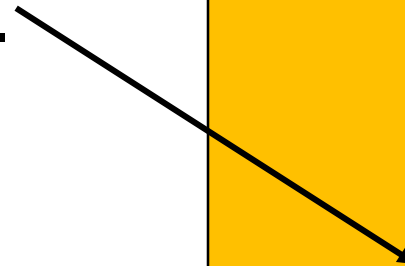
### Architecture web





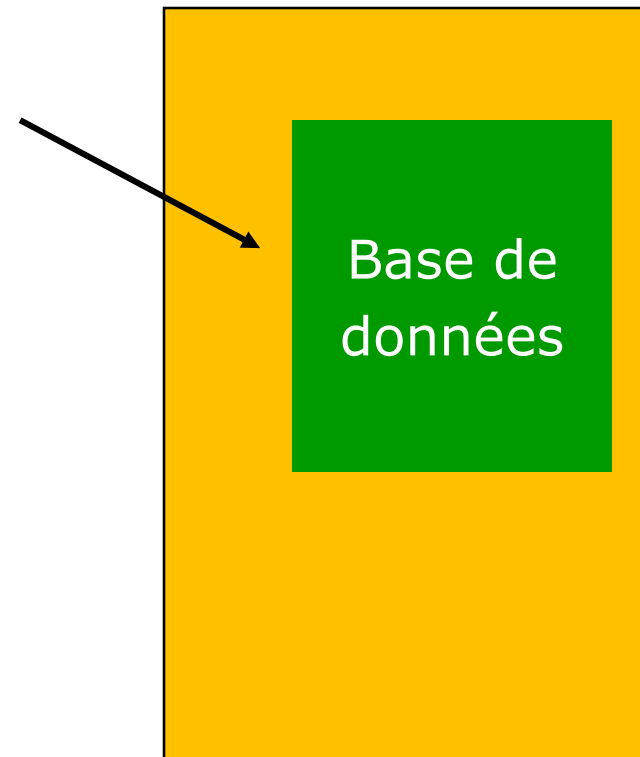
### *Exemple d'une application de E-Commerce*

Ordinateur de  
l'entreprise  
XYZ inc.



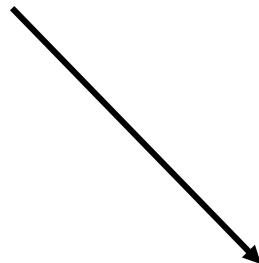
### *Exemple d'une application de E-Commerce*

- Comptes clients
- Inventaire
- Comptabilité
- Commandes
- Suivi de production
- Etc.



### *Exemple d'une application de E-Commerce*

Serveur HTTP

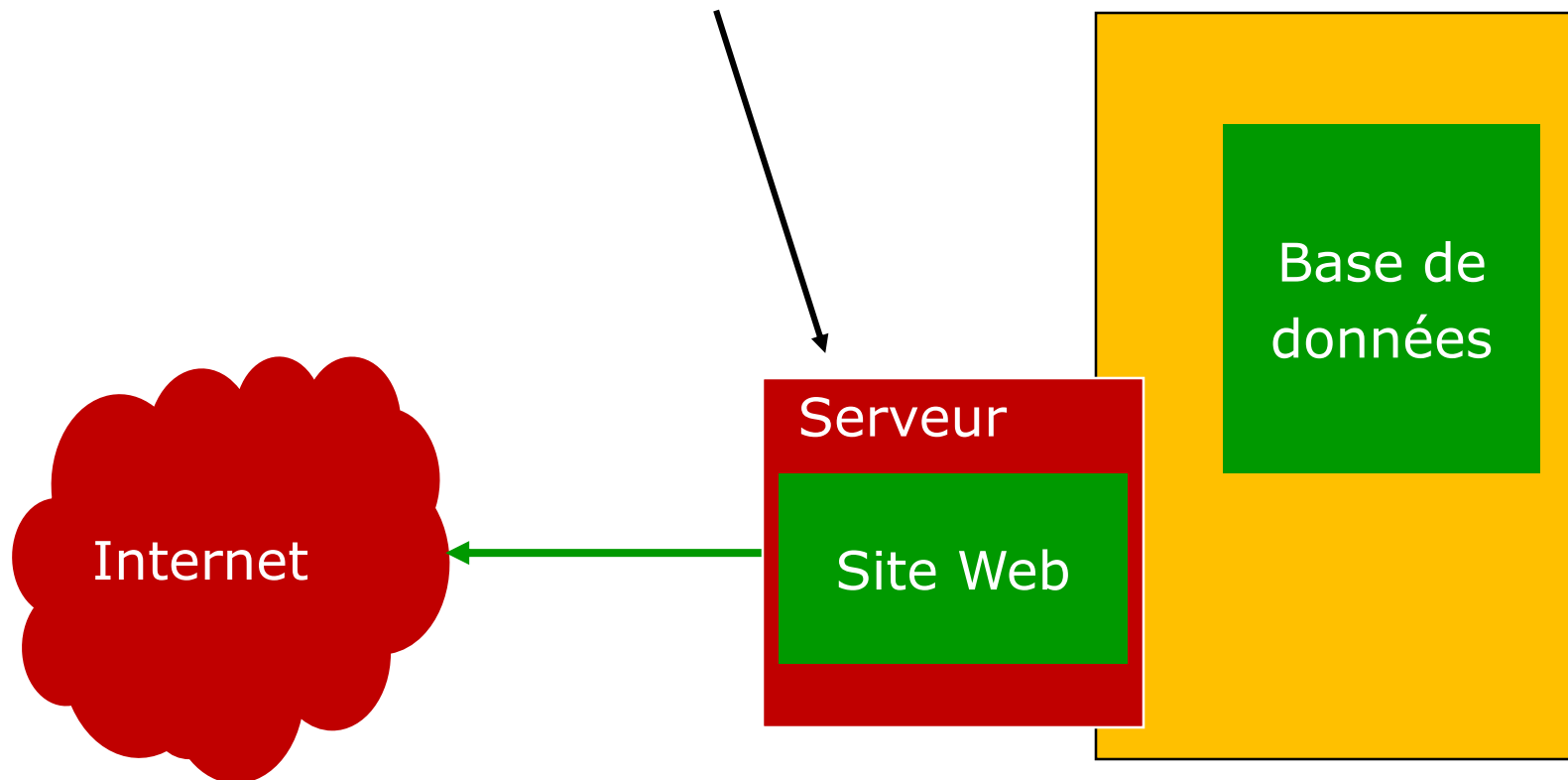


Site Web

Base de données

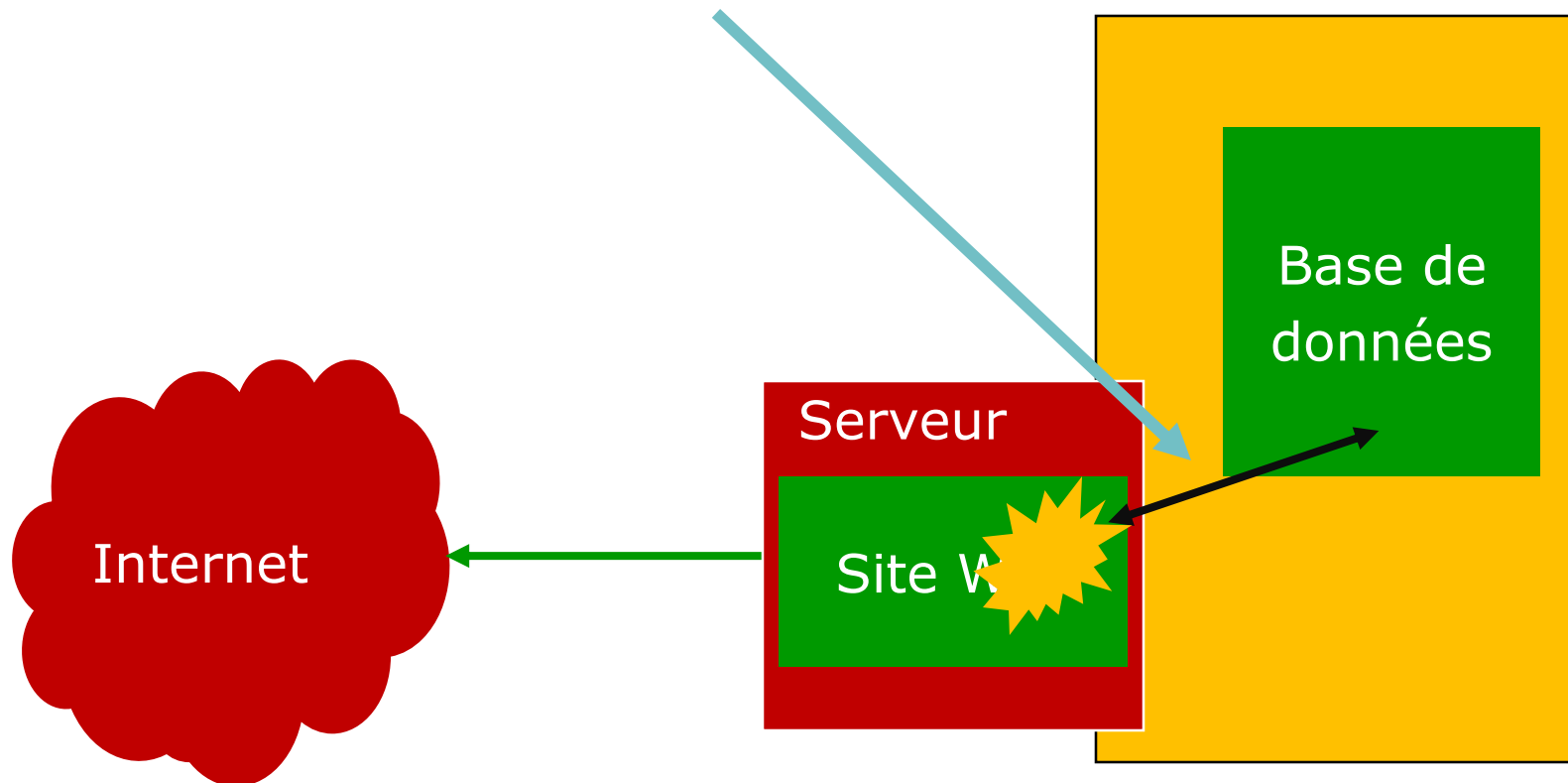
### *Exemple d'une application de E-Commerce*

Le serveur héberge le site Web  
de la compagnie, ouvert sur Internet.



### *Exemple d'une application de CE*

Le site étant dynamique,  
il est relié à la base de données de l'entreprise.



### Exemple d'une application de E-Commerce

#### PREMIÈRE VISITE ?

Vous n'êtes plus qu'à quelques clics de valider votre commande



Civilité \*



Madame



Monsieur

Nom \*

Prénom \*



J'ai une carte fnac

E-mail \*

@

Choisissez un mot de passe \*

8 caractères minimum dont au moins 1 chiffre et 1 lettre.



J'aimerais être alerté en avant-première des meilleures promotions et des exclusivités de la Fnac (promis, pas de spam).

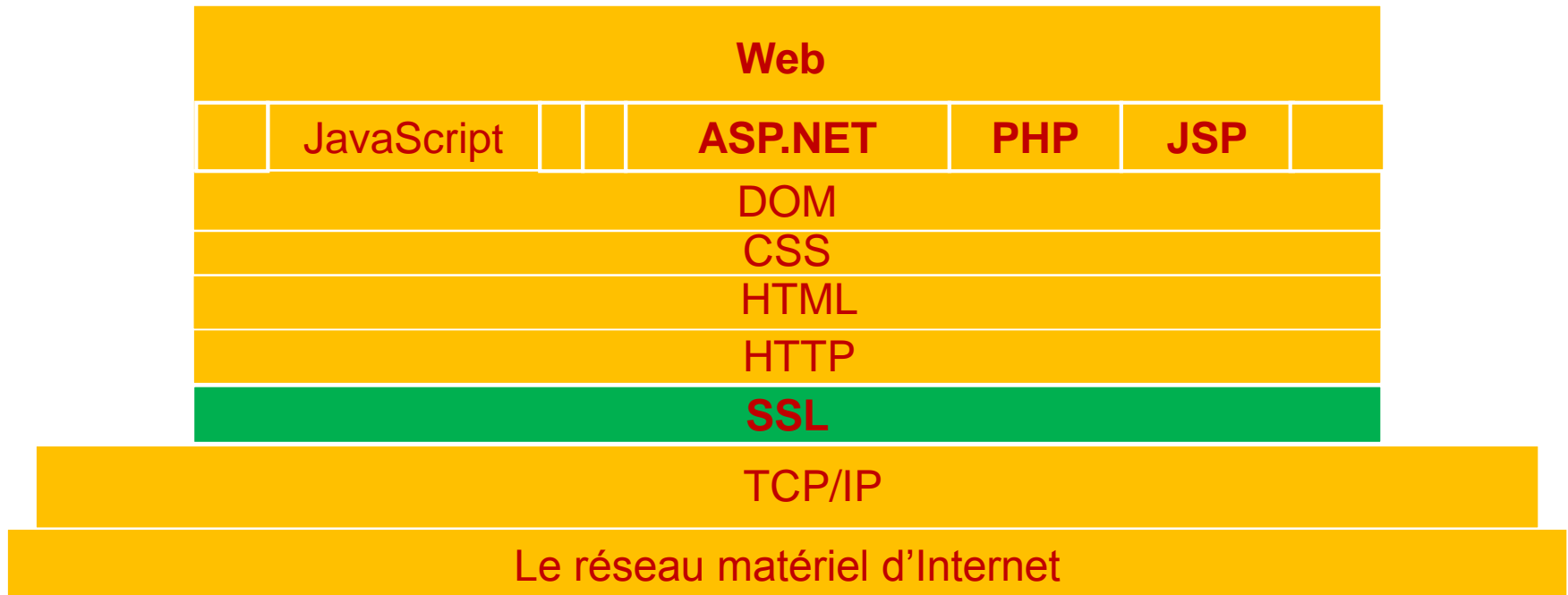
Annuler

Continuer

### *Site web sécurisé*

- SSL (**Secure Sockets Layers**, que l'on pourrait traduire par *couche de sockets sécurisée*) est un procédé de sécurisation des transactions effectuées via Internet. SSL assure 3 choses:
  - **Confidentialité**: Il est impossible d'espionner les informations échangées.
  - **Intégrité**: Il est impossible de truquer les informations échangées.
  - **Authentification**: Il permet de s'assurer de l'identité du programme, de la personne ou de l'entreprise avec laquelle on communique.

### *Architecture web*



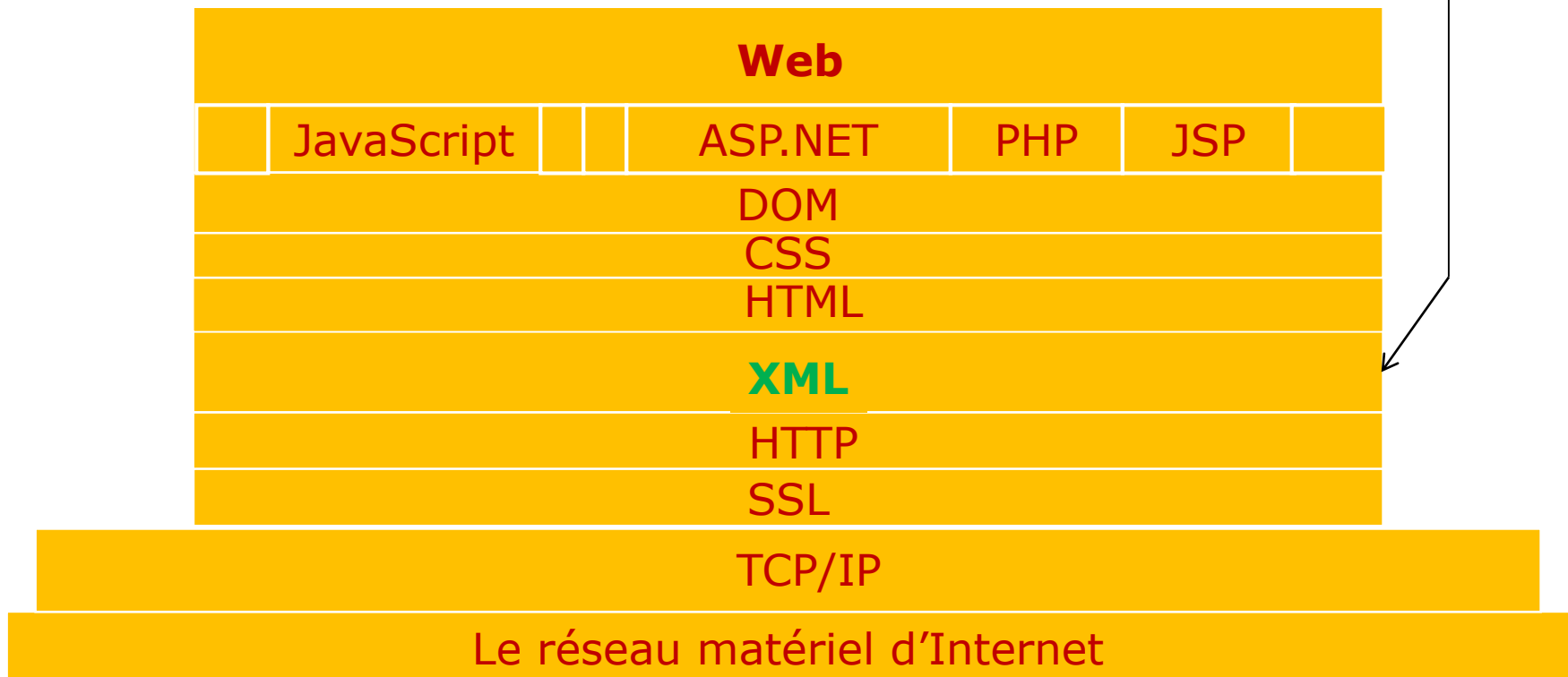


### ***Le problème du web***

- Les standards sont incomplets ;
  - On ne cesse d'innover et que nous devons sans cesse adapter nos protocoles aux nouvelles possibilités.
- Les standards ne sont pas entièrement respectés ;
  - Les entreprises se servent de la compatibilité et de l'incompatibilité comme des armes pour conquérir des parts de marché.
- Les bases sont fragiles;
  - le HTML constituait une base plutôt fragile pour tout ce qu'on souhaitait lui faire supporter.

### Architecture Web

Une refonte majeure :  
le XML



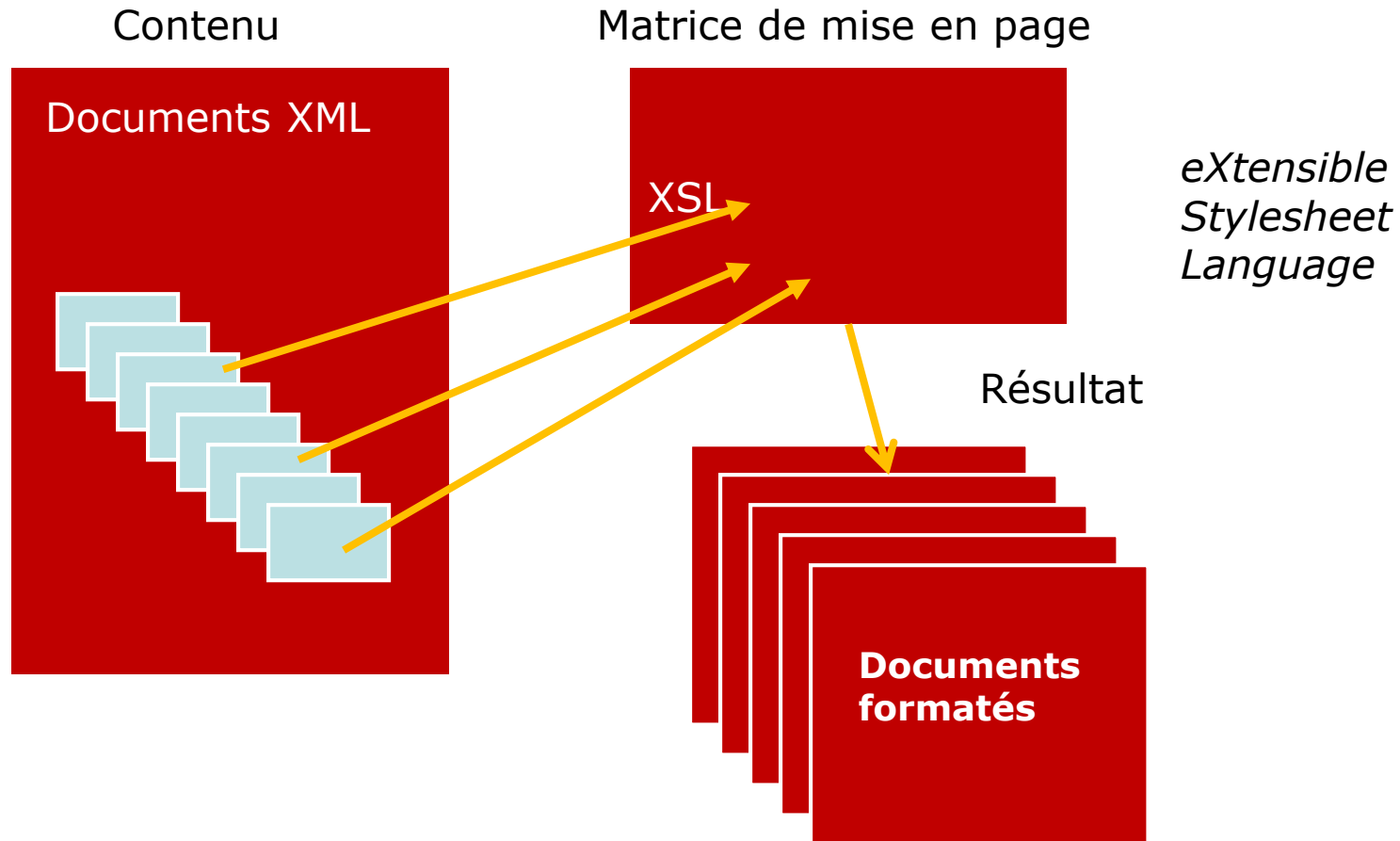
### ***XML...***

- c'est un système de balises, come HTML, qui s'emboîtent les unes dans les autres pour former des structures hiérarchiques complexes.
  - La différence c'est que, avec le XML, la structure concerne le contenu du document et non sa présentation.
  - Les balises, concernent strictement le contenu, et leur fonction est d'en distinguer les parties, comme les champs d'une base de données.

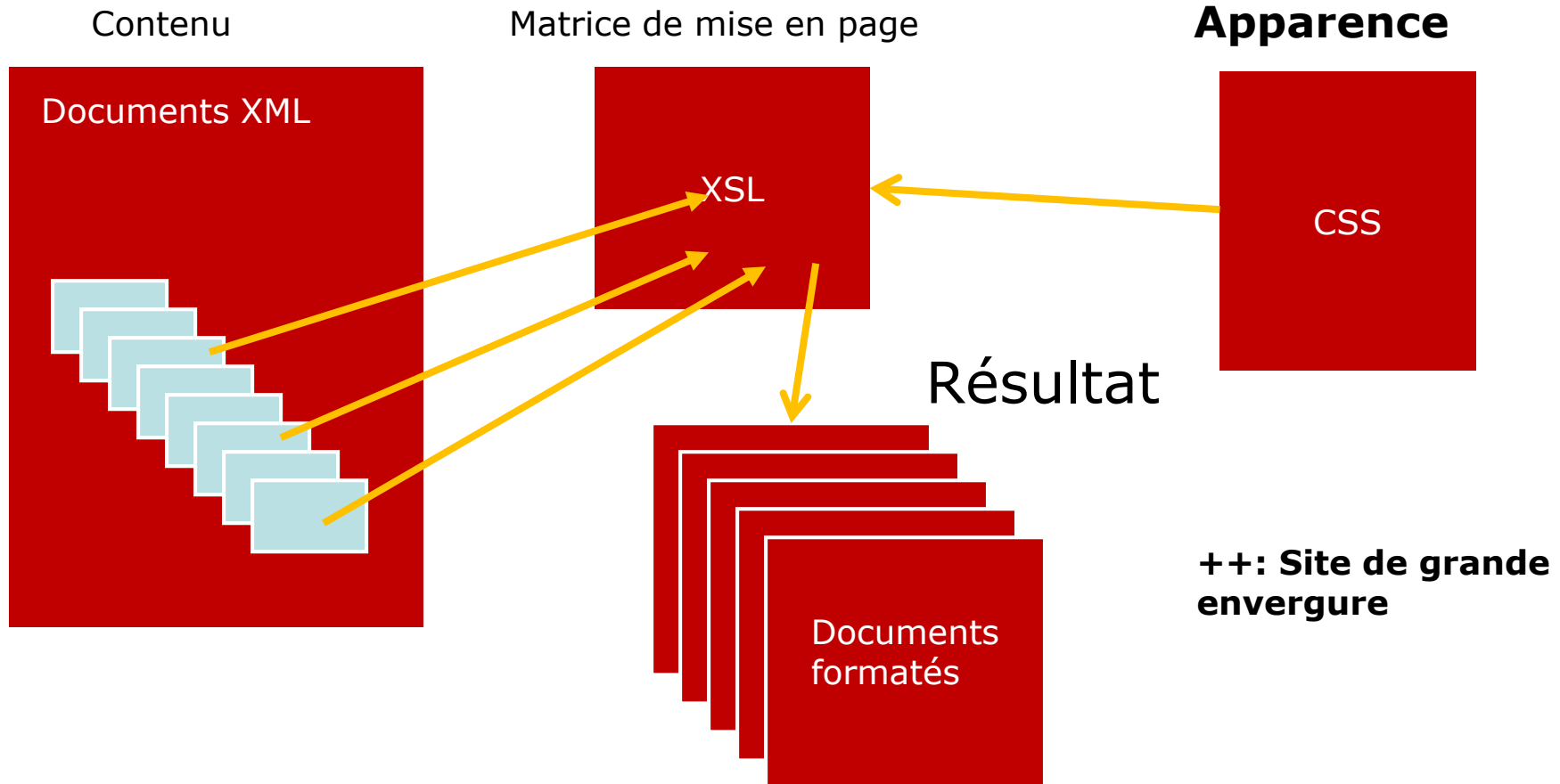
### ***XML...***

```
<livre>  
  <titre>  
    L'Ingénieux Hidalgo Don Quichotte de la Manche  
  </titre>  
  <auteur>  
    <nom>  
      Cervantès  
    </nom>  
    <prénom>  
      Miguel  
    </prénom>  
  </auteur>  
  <chapitre no=8>  
    <titre>  
      Du beau succès que le valeureux Don Quichotte  
    </titre>  
  </chapitre>  
</livre>
```


### Comment fonctionne XML



### Comment fonctionne XML



### ***JSON vs XML***

- XML est un langage verbeux
- Le XML est une **calamité** à parser en JavaScript
- **JavaScript Object Notation**, un format ouvert de données, qui utilise la notation des objets JavaScript pour transmettre de l'information structurée
- Structure JSON :
  - Paires clé / valeurs
  - Les valeurs peuvent être des listes de paires, des tableaux de paires, des objets, chaînes de caractères...
- Objet JavaScript  Doc au format JSON

### Exemple de format JSON

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Convert a string written in JSON format, into a JavaScript object.</h2>
```

```
<p id="demo"></p>
```

```
<script>
```

JSON : Paires clé / valeurs

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
```

```
var myObj = JSON.parse(myJSON);
```

```
document.getElementById("demo").innerHTML = myObj.name;
```

```
</script>
```

```
</body>
```

```
</html>
```

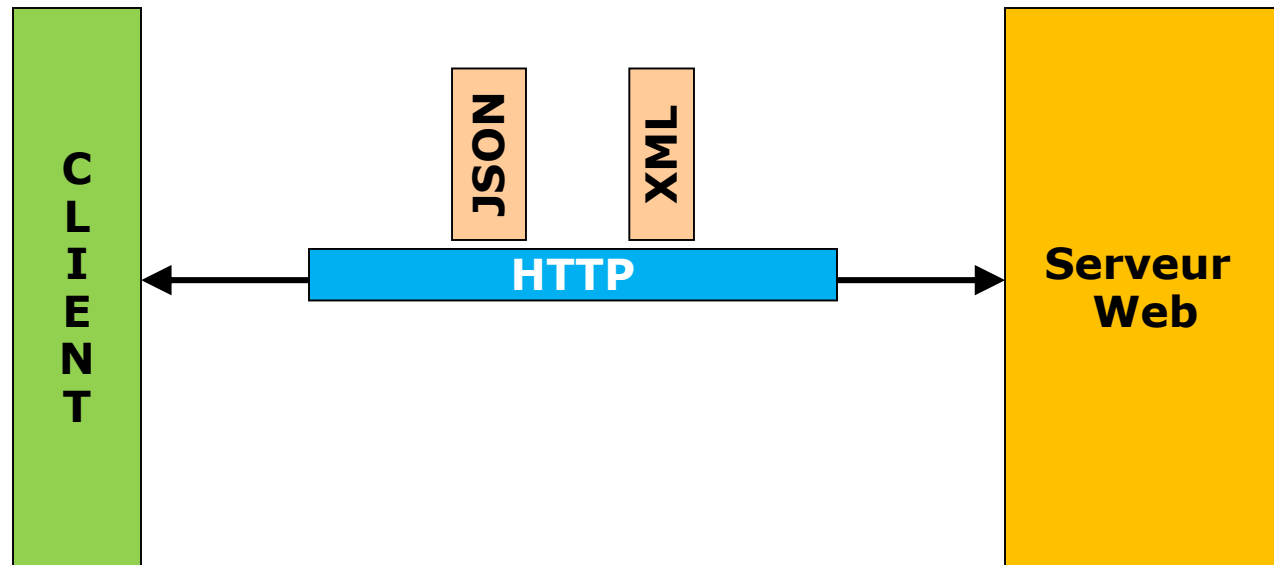
#### Le Résultat

Convert a string written in JSON format, into a JavaScript object.

John



### *JSON/XML/HTTP*



### JSON/XML

**XML**

```
<request>
  <method>people.get</method>
  <params>
    <userId>
      <id>@owner</id>
    </userId>
    <groupId>@self</groupId>
    <id>owner</id>
    <fields>
      <field>id</field>
      <field>name</field>
      <field>thumbnailUrl</field>
      <field>profileUrl</field>
      <field>id</field>
      <field>displayName</field>
    </fields>
  </params>
</request>
```

**281 Bytes**

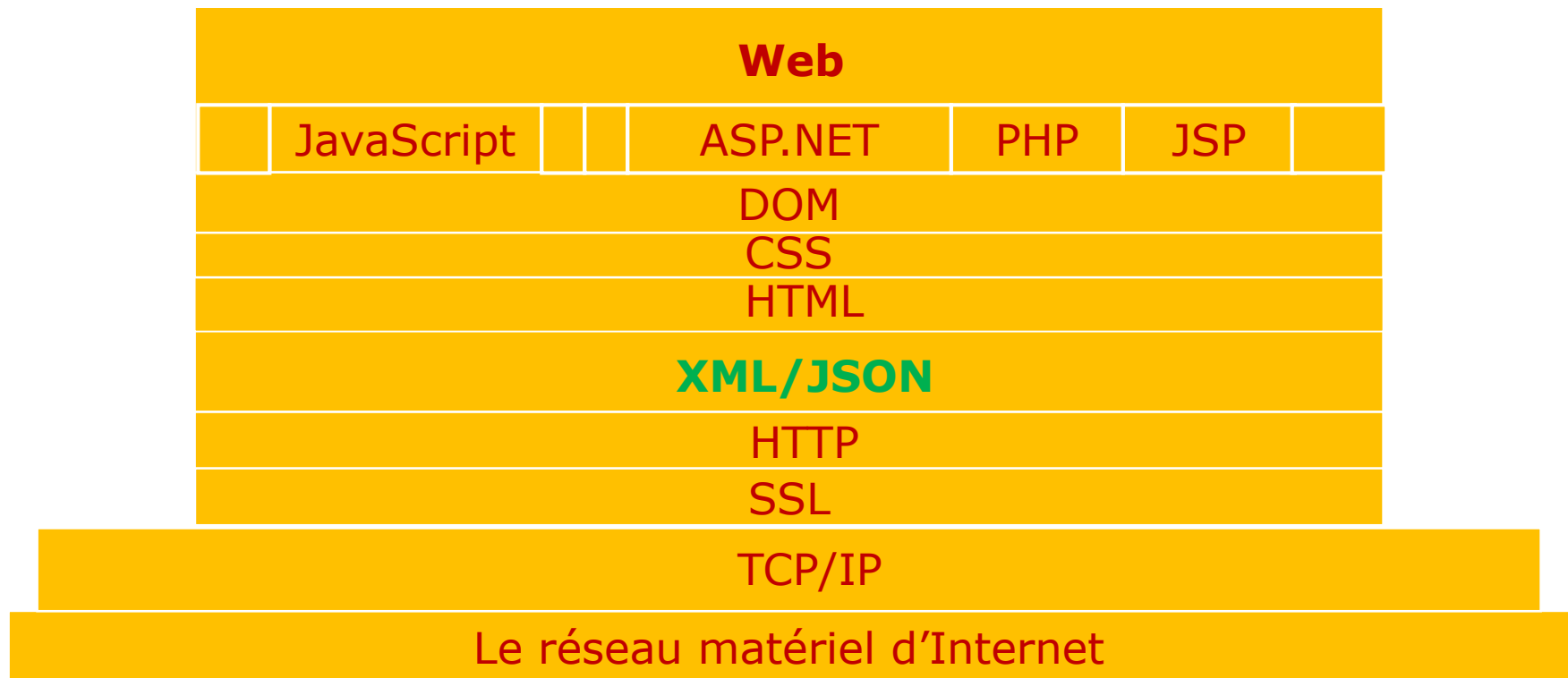
```
[{"method" : "people.get",
  "params" : {
    "userId" : ["@owner"],
    "groupId" : "@self",
    "id" : "owner",
    "fields" :
      ["id", "name", "thumbnailUrl", "profileUrl", "id", "displayName"]
  }
}]
```

**157 Bytes**

**JSON**



### Architecture Web

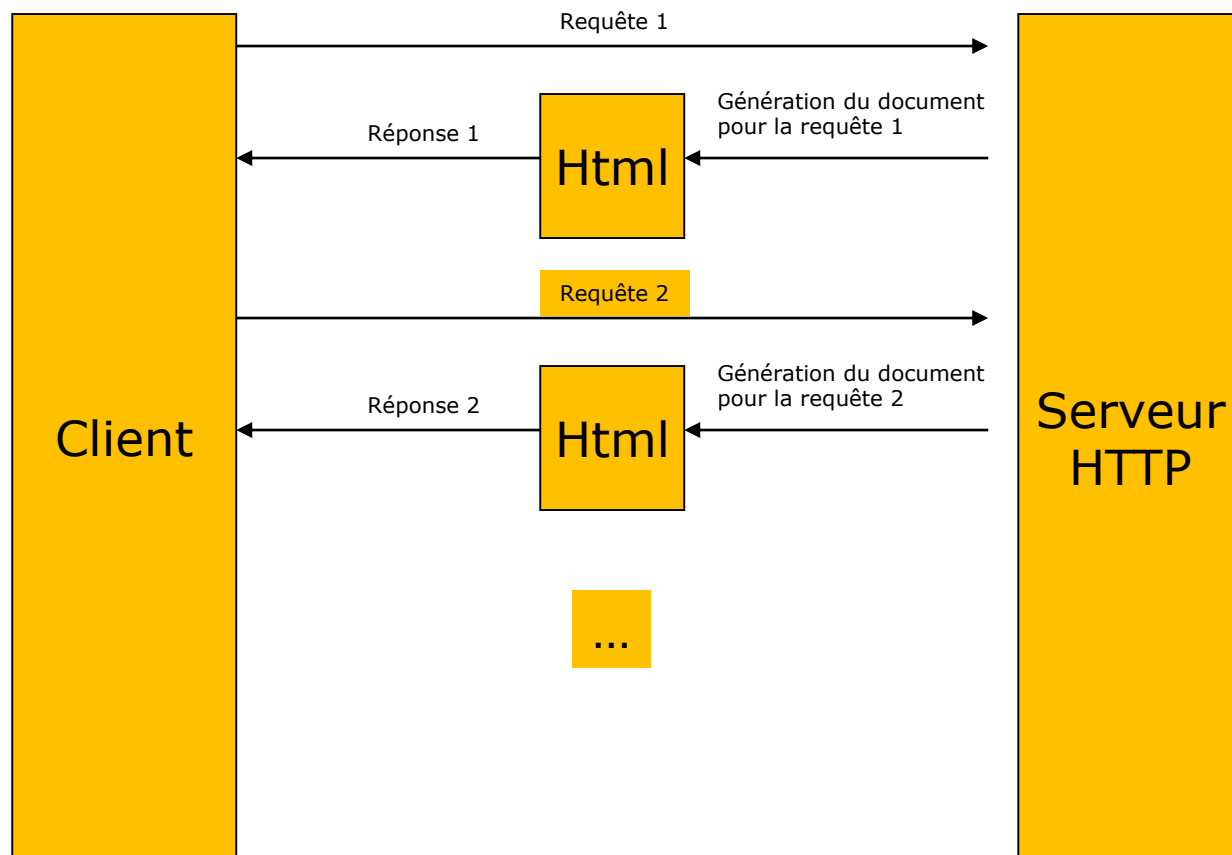


### ***Web réactif et ergonomique***

- Application WEB traditionnelle :
  - Le client envoie une requête HTTP
  - Le serveur renvoie une page
    - Cela consomme inutilement une partie de la bande passante, car une grande partie du code HTML est commun aux différentes pages de l'application
    - Le chargement d'une nouvelle page à chaque requête n'est pas ergonomique
- Demandes pour une architecture plus réactive
  - Interface plus riche
  - Interface personnalisable

**AJAX**

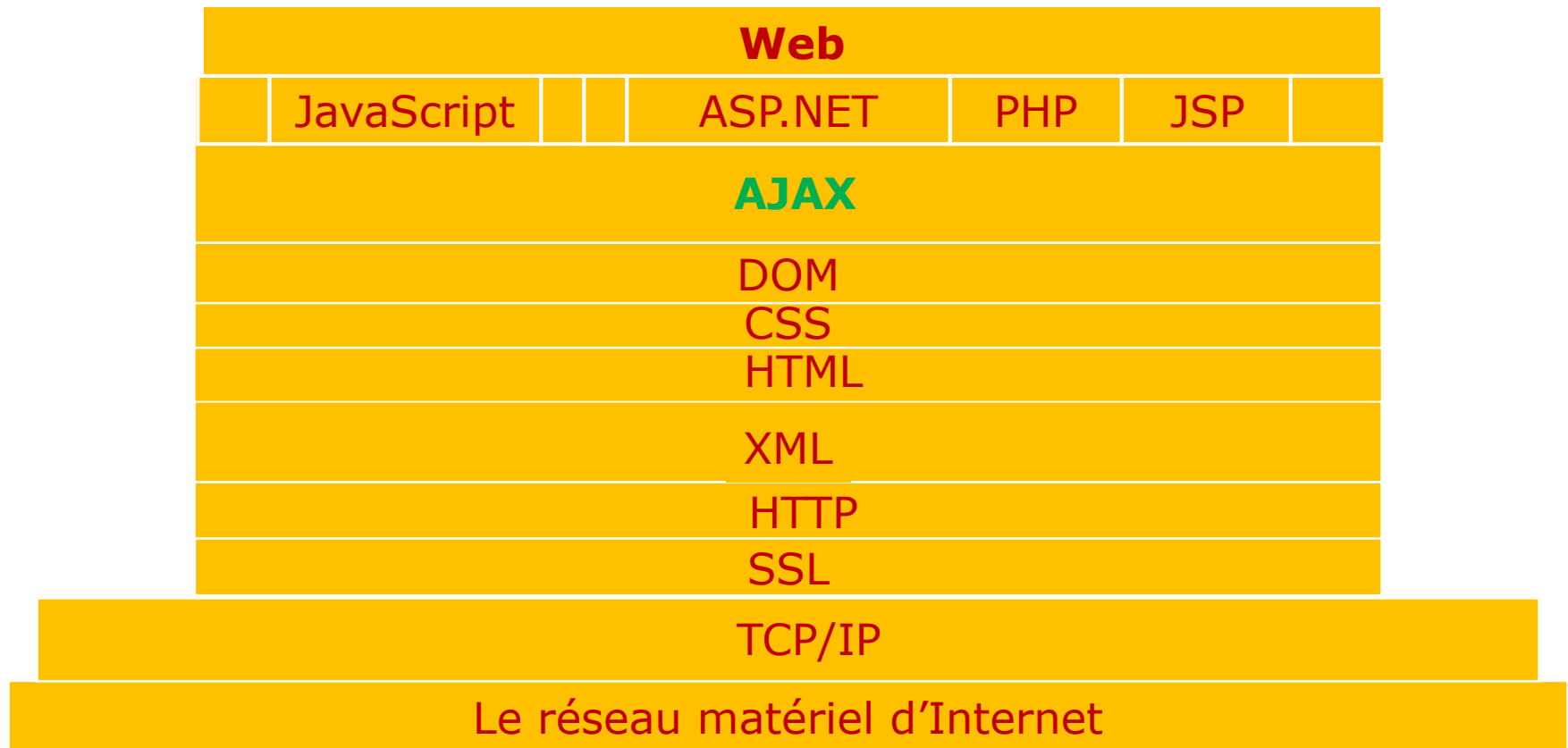
### *Web traditionnel*



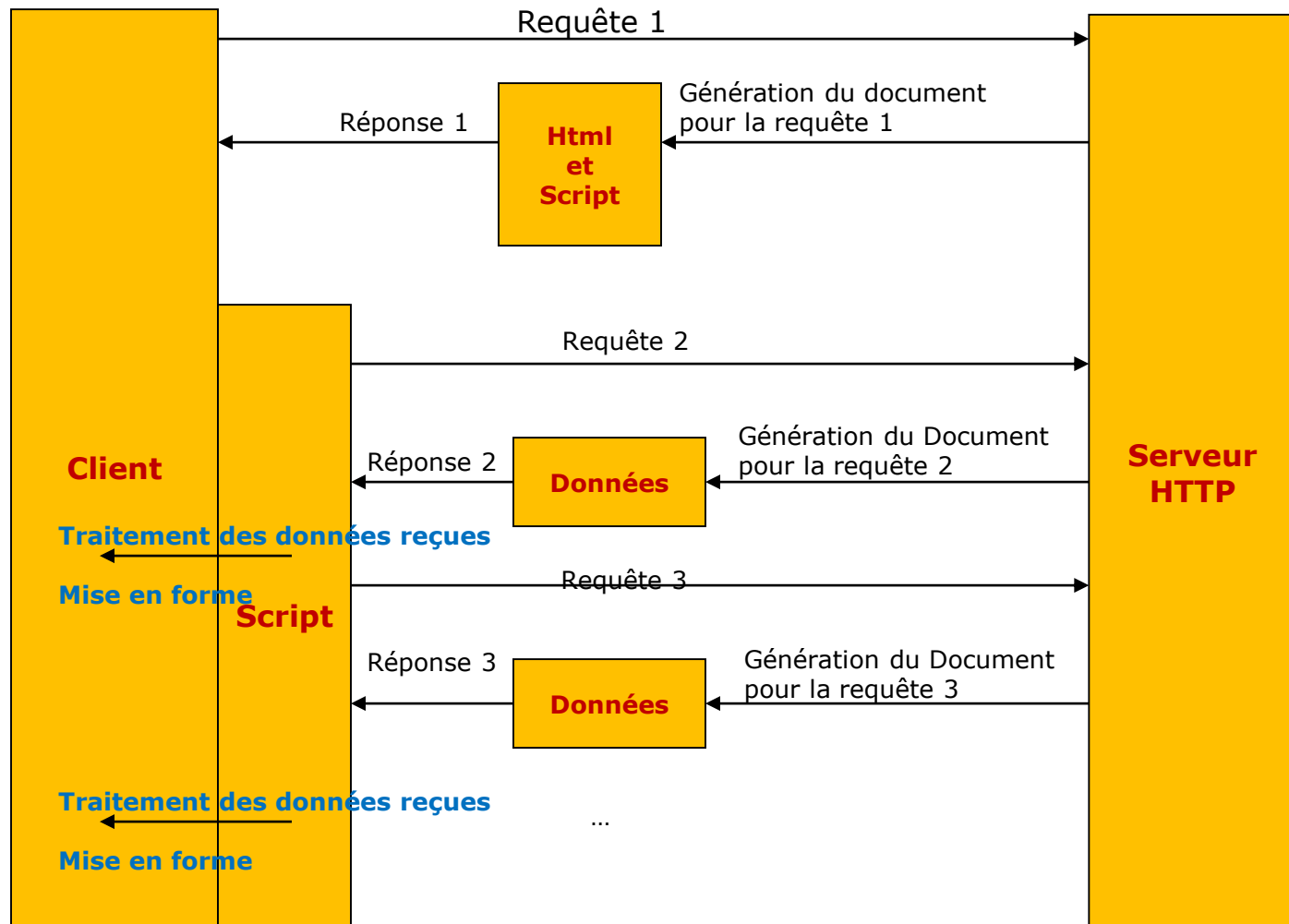
### ***AJAX vers Web 2.0***

- AJAX = **A**ynchronous **J**avaScript **A**nd **X**ML
- Utilisation **conjointe** d'un ensemble de technologies couramment utilisées sur le Web :
  - HTML (ou XHTML) et CSS pour la mise en forme
  - **DOM et JavaScript** pour afficher et interagir dynamiquement avec l'information présentée
  - XML, XSLT et l'objet **XMLHttpRequest** pour échanger et manipuler les données de manière asynchrone avec le serveur web

### Architecture Web



### Approche AJAX





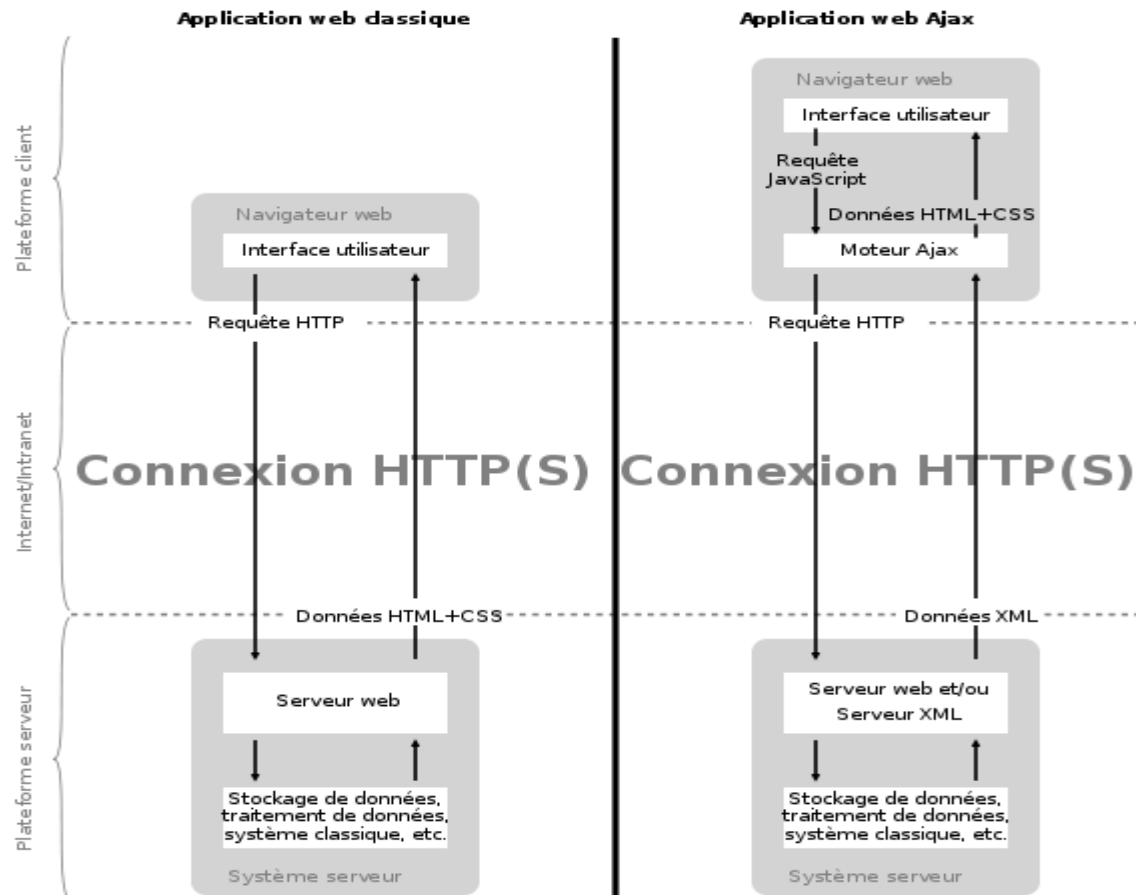
### ***Approche AJAX***

- Application AJAX :
  - Envoyer des requêtes au serveur HTTP pour ne récupérer que les données nécessaires
  - Utilisation de l'objet HTTP XMLHttpRequest
  - Utilisation la puissance des feuilles de style (CSS) et de Javascript côté client pour interpréter et mettre en forme la réponse du serveur
  - Permet au navigateur de modifier partiellement la page pour la mettre à jour sans avoir à la recharger
  - Applications plus réactives, meilleure ergonomie

### ***Approche AJAX***

- Nécessite de charger, sur la première page, une bibliothèque AJAX volumineuse
- Nécessite un navigateur compatible, autorisant le Javascript et le composant XMLHttpRequest
- Nécessite des tests minucieux car il existe de grandes différences entre les navigateurs

### Approche AJAX



### *XMLHttpRequest*

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
  <button type="button" « onclick="loadDoc('ajax_info.txt', myFunction)">Change Content
  </button>
</div>

<script>
function loadDoc(url, cFunction) {
  var xhttp;
  xhttp=new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
function myFunction(xhttp) {
  document.getElementById("demo").innerHTML = xhttp.responseText;
}
</script>
</body>
</html>
```

### ***HTML/CSS***

- Vous devez les maîtriser d'abord
- Vous n'avez pas le choix
- Facile à apprendre

### *Outils de base*

- Editeur de Texte : notepad 2 /Sublime Text /Atom.io/Brackets.io/IDE
- Editeur d'images : Photoshop/GIMP/Illustrator
- FTP/SSH : Putty/FileZilla
- Navigateurs: Chrome/Firefox

### ***Bases de JavaScript***

- JS de base
  - Types de données
    - String, Array, Object, Number
  - Structures de contrôle
    - Boucle for, While,
  - Fonctions, Opérateurs
  - Gestion des événements
  - JSON
  - **jQuery** : ssi vous sentez à l'aise avec JS!

### *Front End UI (Web Designer)*

- Choisir un **framework HTML/CSS** pour le développement front end
  - **Twitter Bootstrap**
  - Zurb Foundation
  - Pure
  - MUI

C'est un ensemble qui contient

- des codes [HTML](#) et [CSS](#),
- des formulaires, boutons,
- des libellés, icônes, miniatures, barres de progression
- des outils de navigation et
- autres éléments interactifs,
- ainsi que des extensions [JavaScript](#) en option



Bootstrap permet de s'adapter dynamiquement au format des supports depuis lesquels ils sont accédés ([PC](#), [tablette](#), [smartphone](#)).



### ***Back End, Server-Side***

- PHP : n'est pas le meilleur langage mais il a fait ses preuves
- Node.js : framework, nouveau arrivant, puissant...
- Ruby on rails : de moins en moins
- Python : facile à apprendre, pas très populaire

### ***Bases de données***

- Relationnel
  - MySQL/PostgreSQL
- NoSQL
  - MongoDB
  - CouchDB
  - Cassandra

**PHP/MySQL**

**Node.js/MongoDB**

### Outils

- Git, Github : contrôle de version, dépôt
- SSH
- Pré compilateurs CSS : Sass/LESS

**LESS** est un langage dynamique de génération de feuilles de style conçu par Alexis Sellier

LESS ajoute à CSS les mécanismes suivants : variables, imbrication, mixins, opérateurs et fonctions.

```
@color: #4D926F;  
  
#header {  
  color: @color;  
}  
  
h2 {  
  color: @color;  
}
```

La compilation du code LESS ci-dessus donne le code CSS suivant :

```
#header {  
  color: #4D926F;  
}  
  
h2 {  
  color: #4D926F;  
}
```

### Frameworks

- **JavaScript**

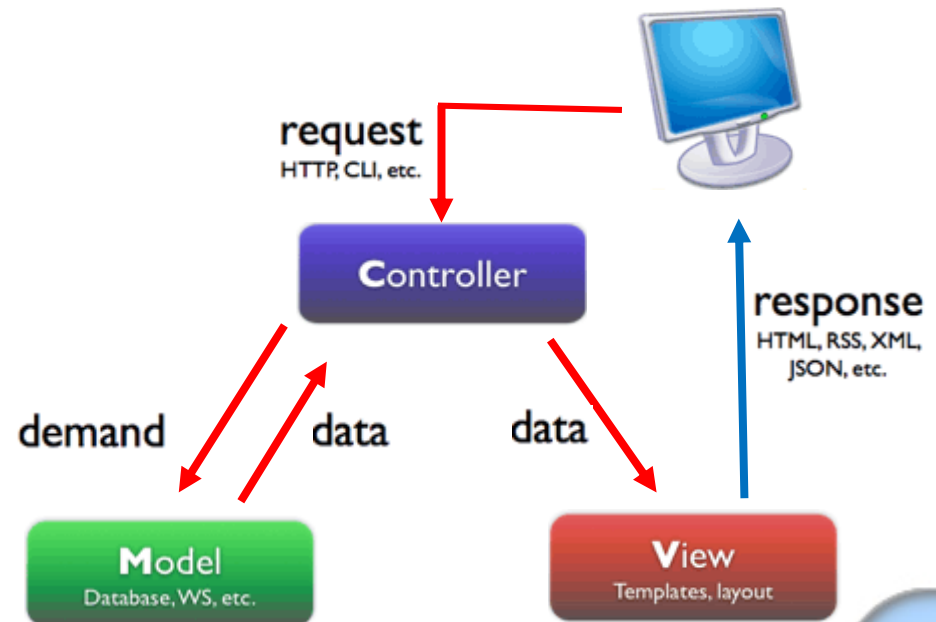
- **AngularJS**
- React
- Vue.js
- Express (backend)

#### MVC

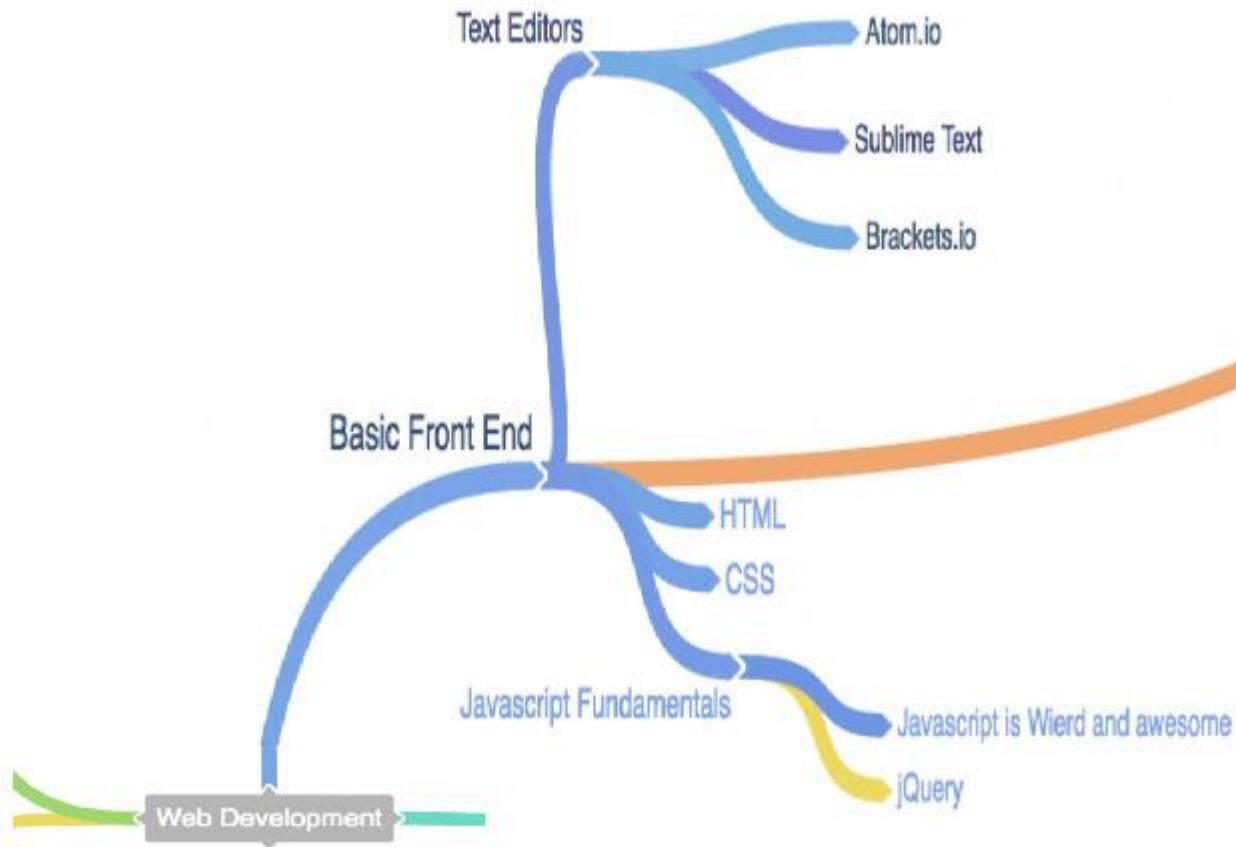
- Routing
- Data Mapping
- Helpers
- Data Binding
- Templating & UI

- **PHP**

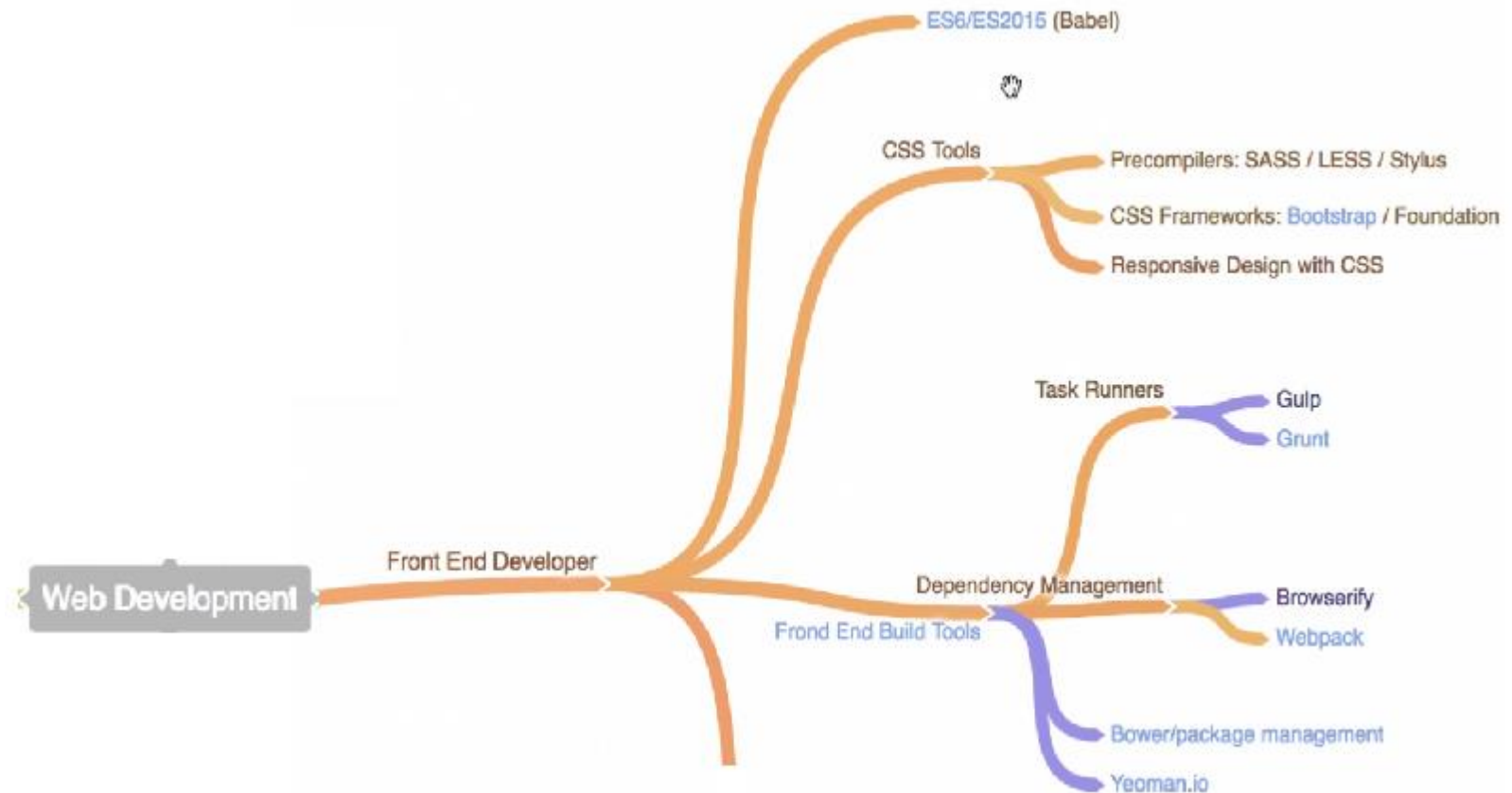
- Laravel
- Codeigniter
- Symfony
- CakePHP 2
- Yii
- Zend Framework 2



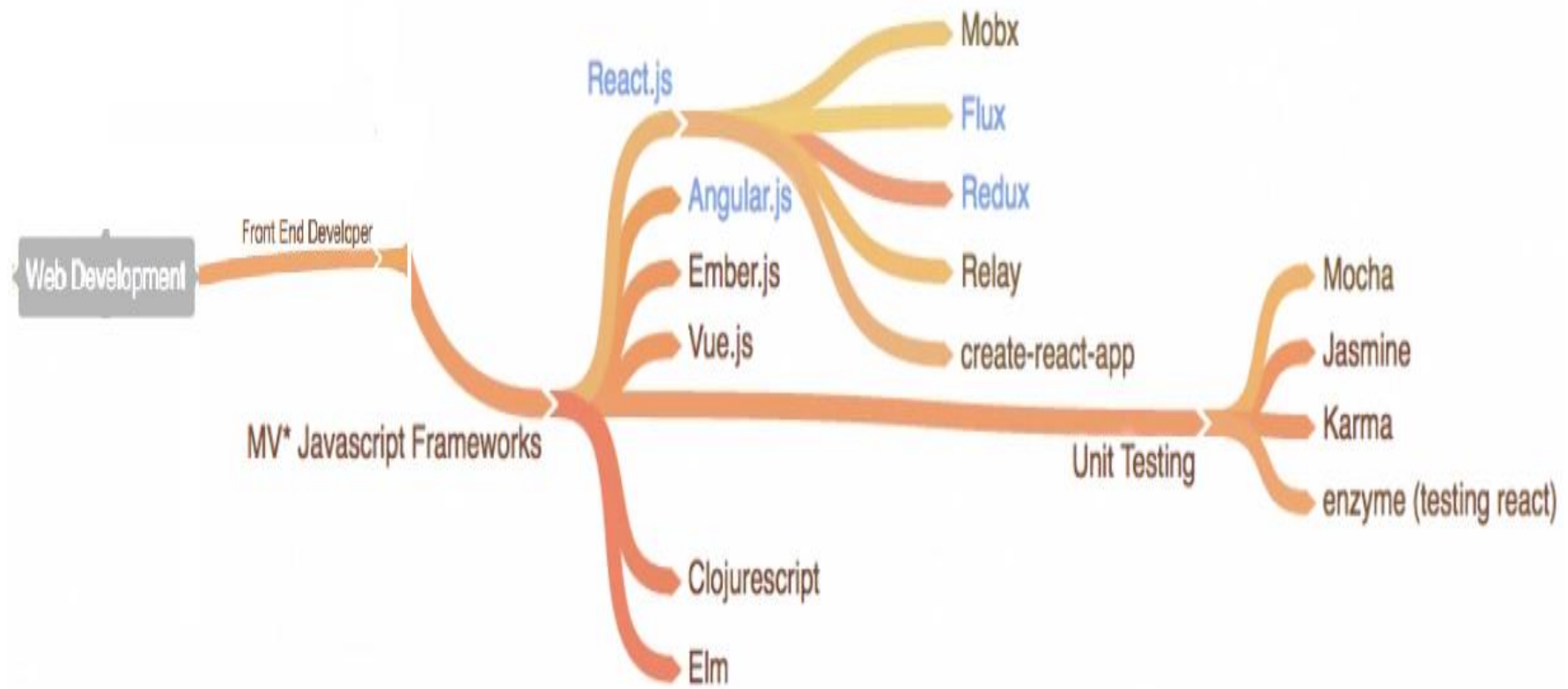
### *Développeur Web 2017,*



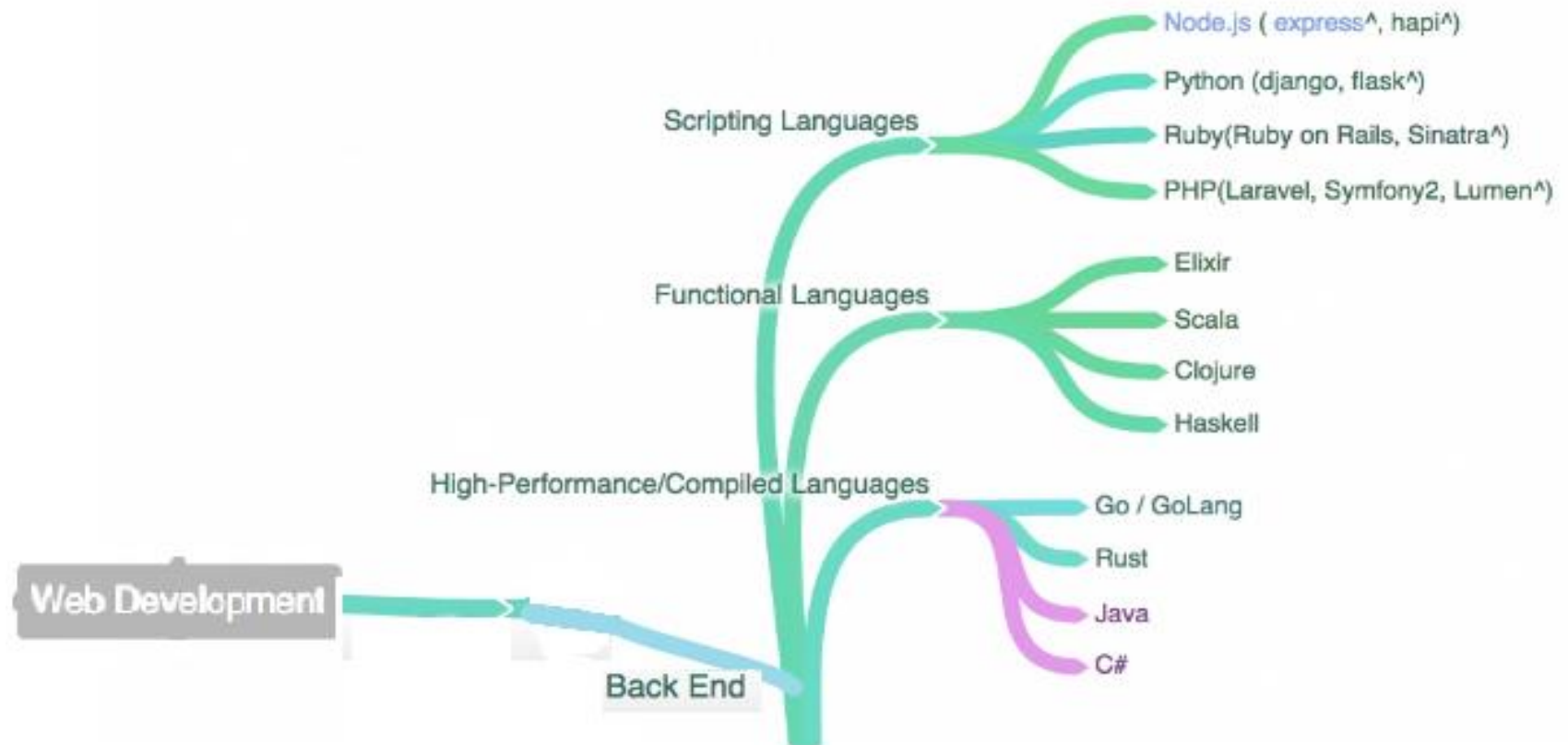
### Développeur Web 2017,



### Développeur Web 2017,

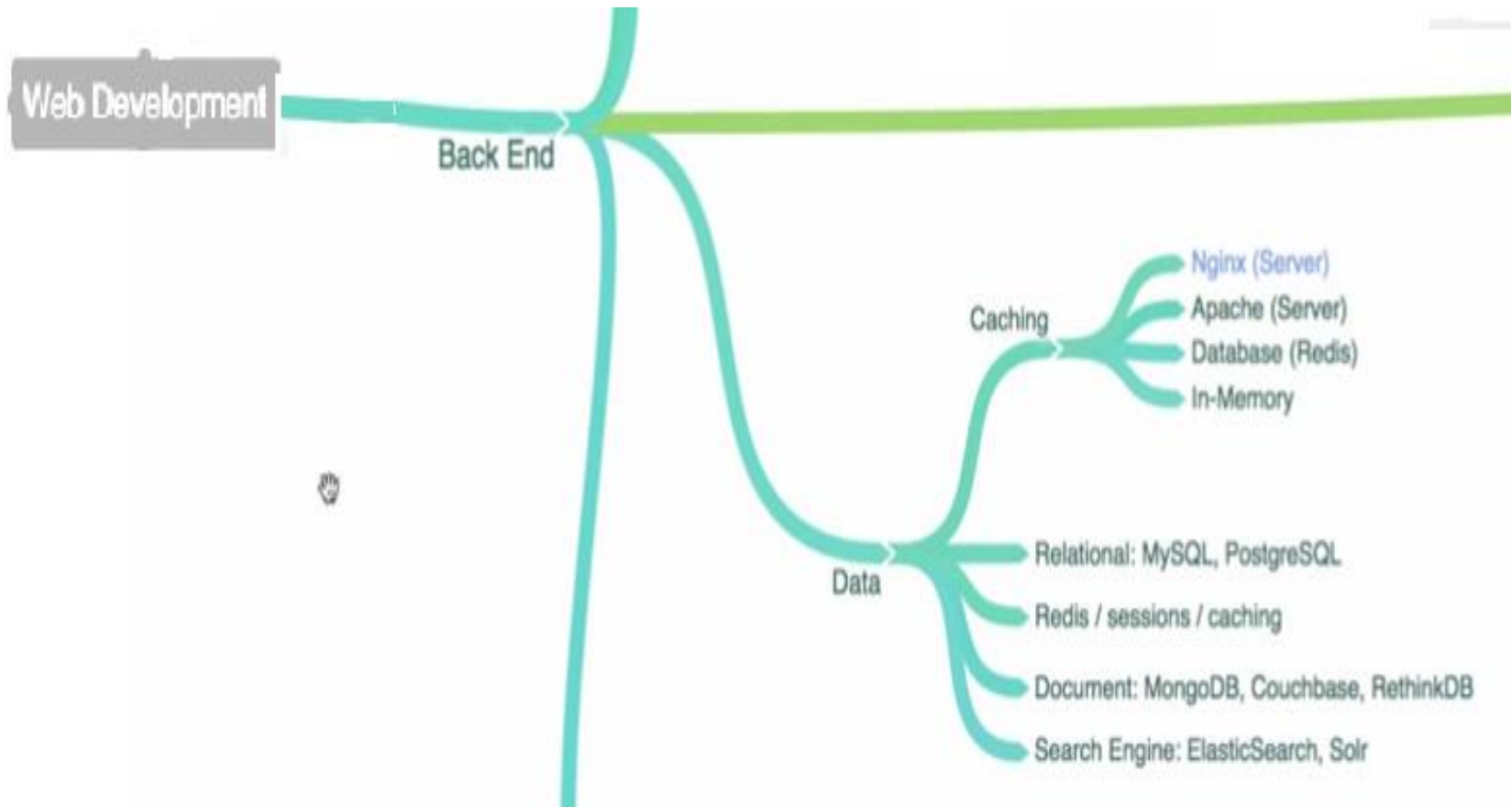


### Développeur Web 2017,

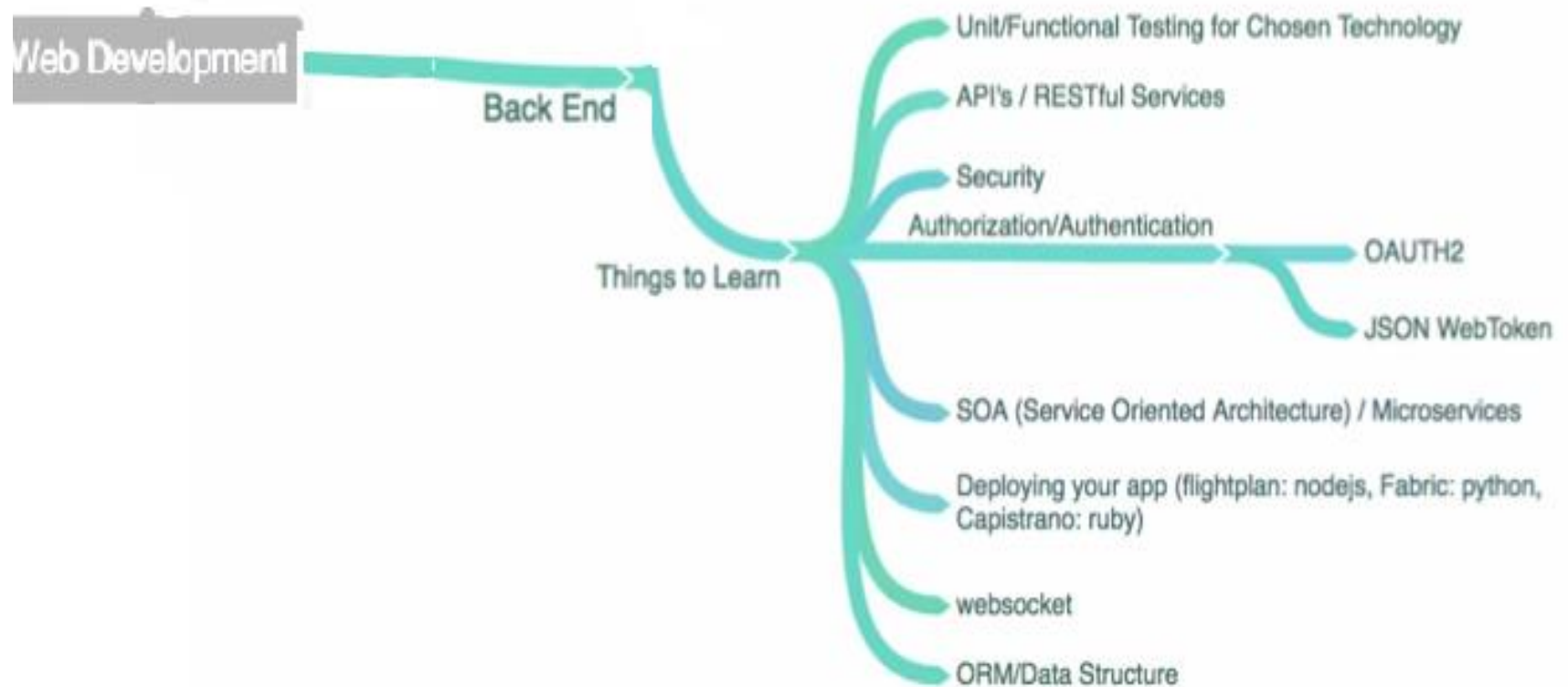




### Développeur Web 2017,



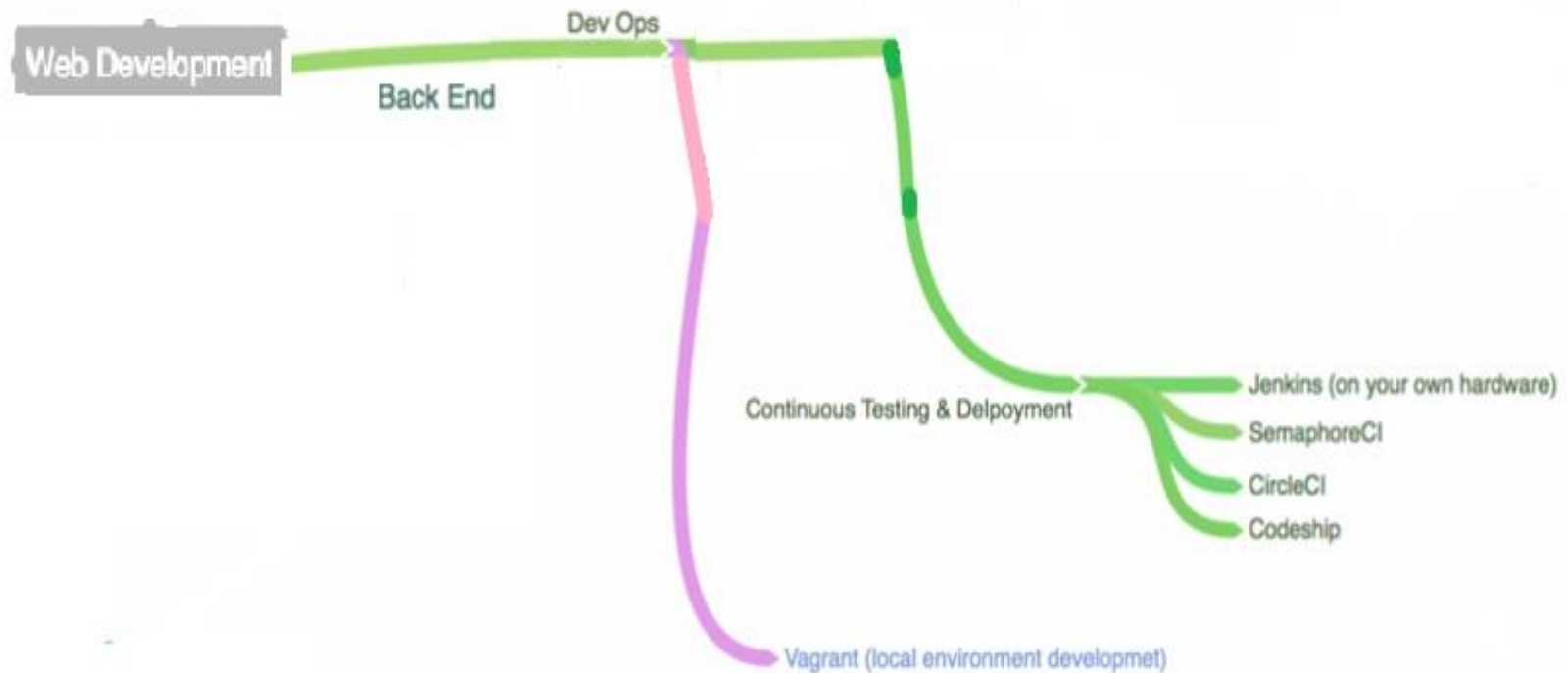
### Développeur Web 2017,



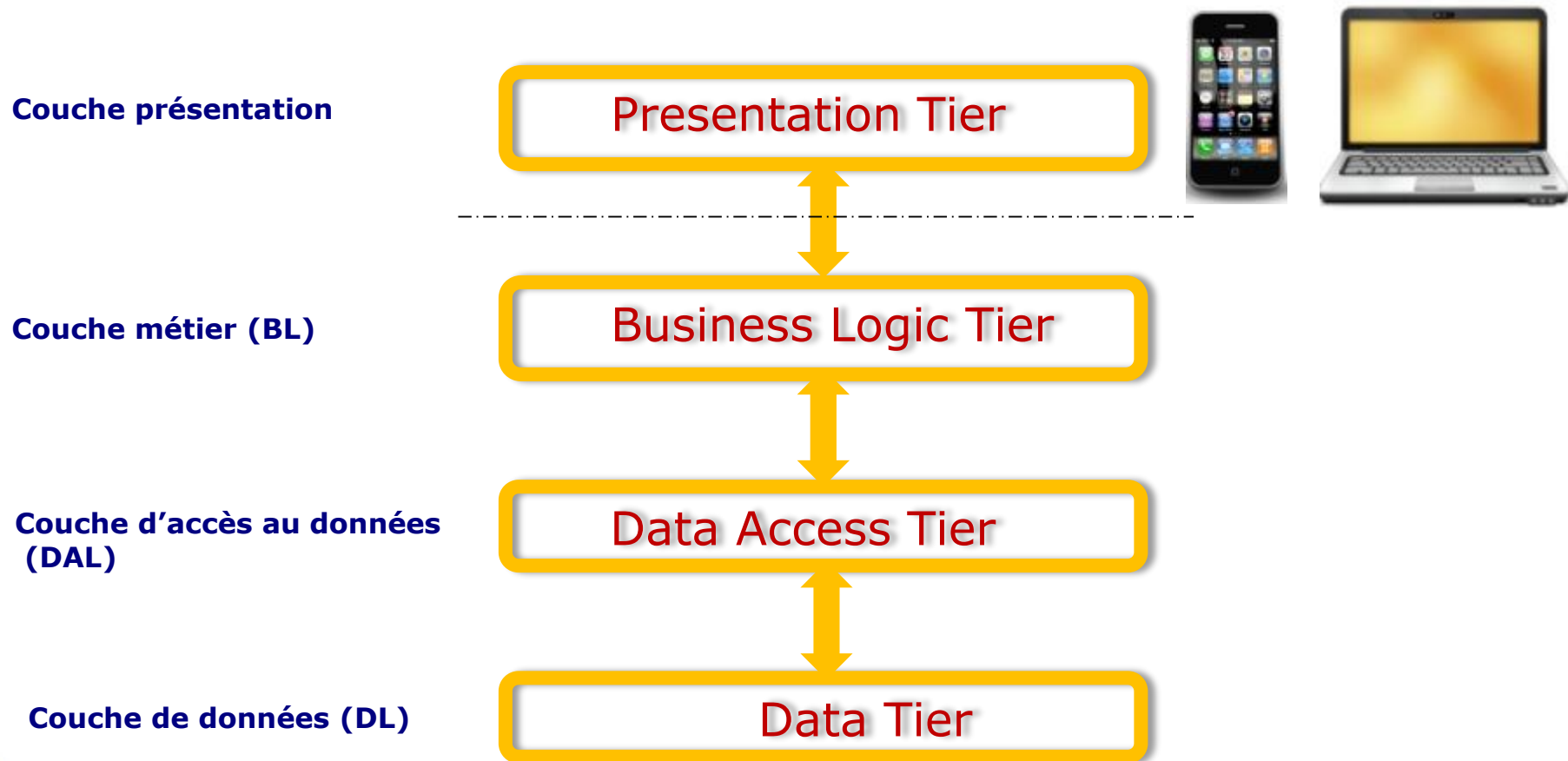
### Développeur Web 2017,



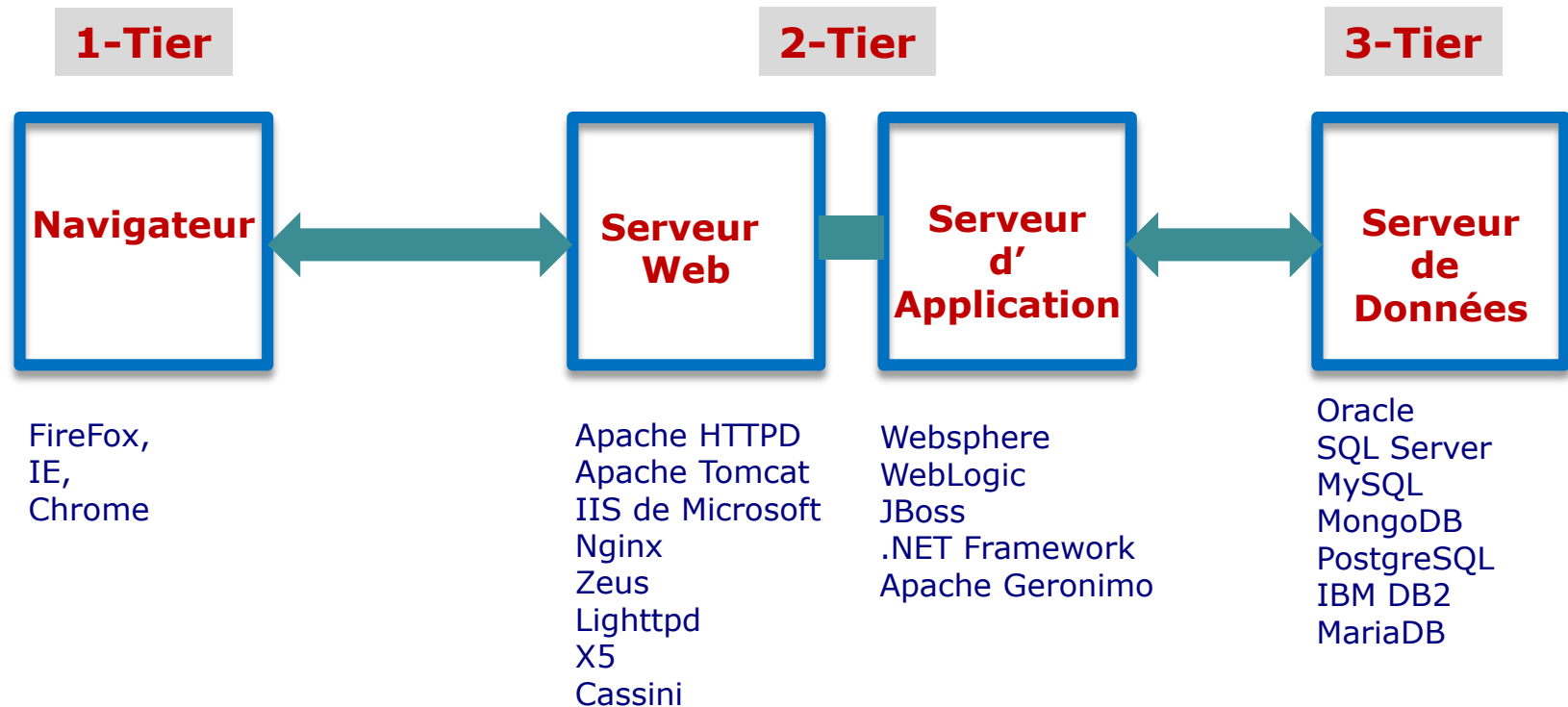
### Développeur Web 2017,



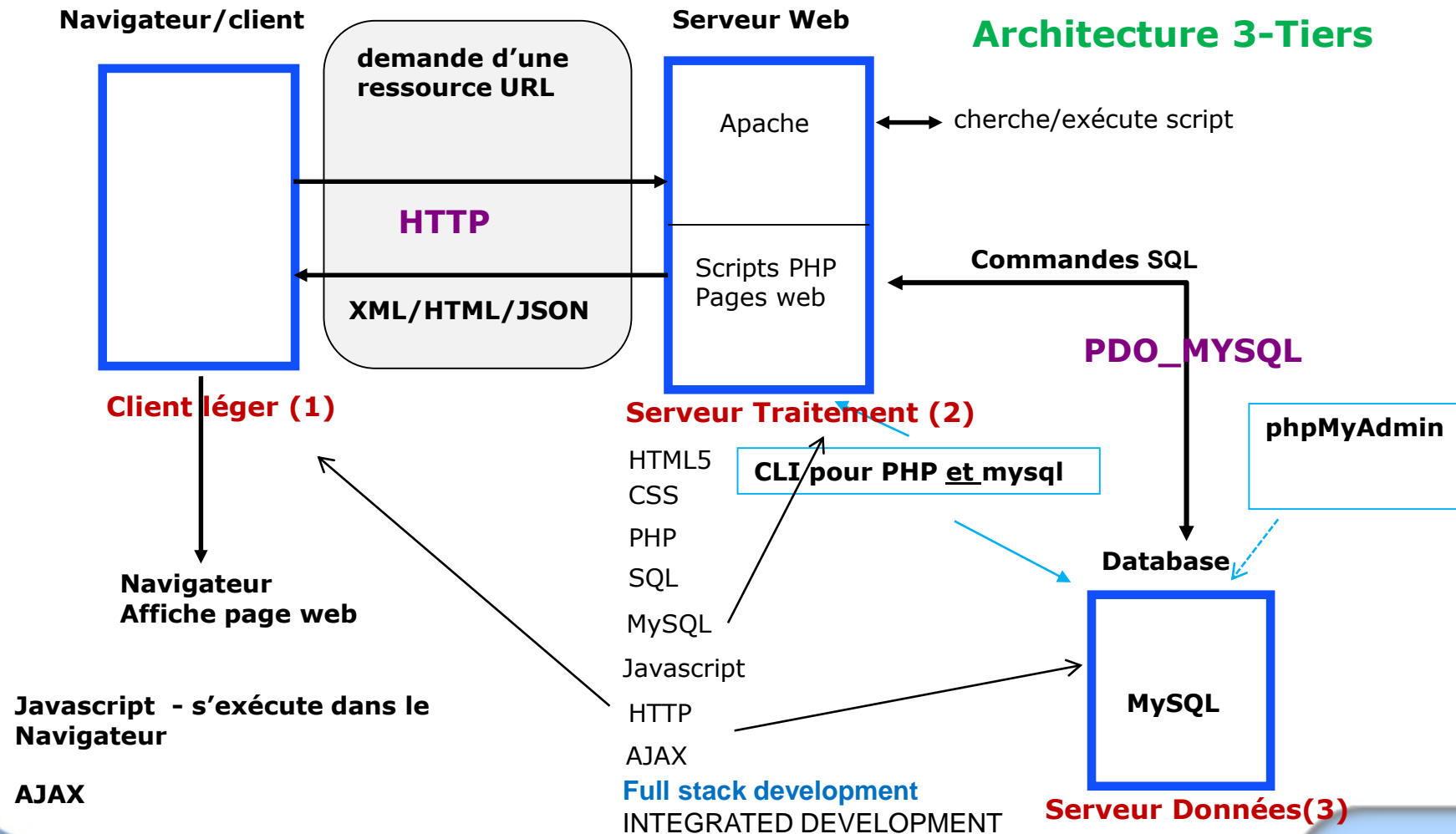
### *Architecture multi-tiers générique*



### Architecture Web multi-tiers générique

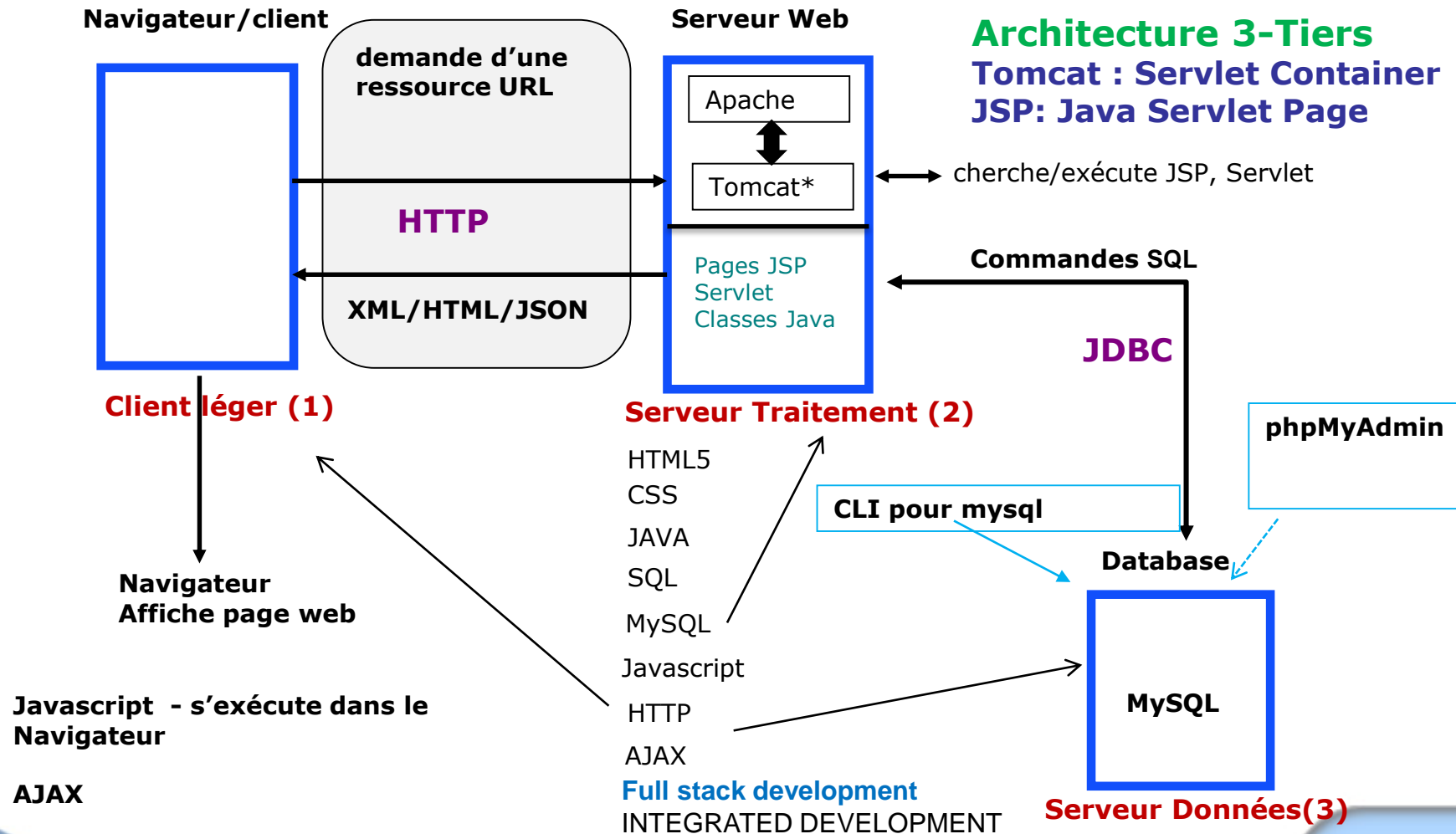


### Pile AMP (Apache, MySQL, PHP)



### Pile Apache, Tomcat, MySQL

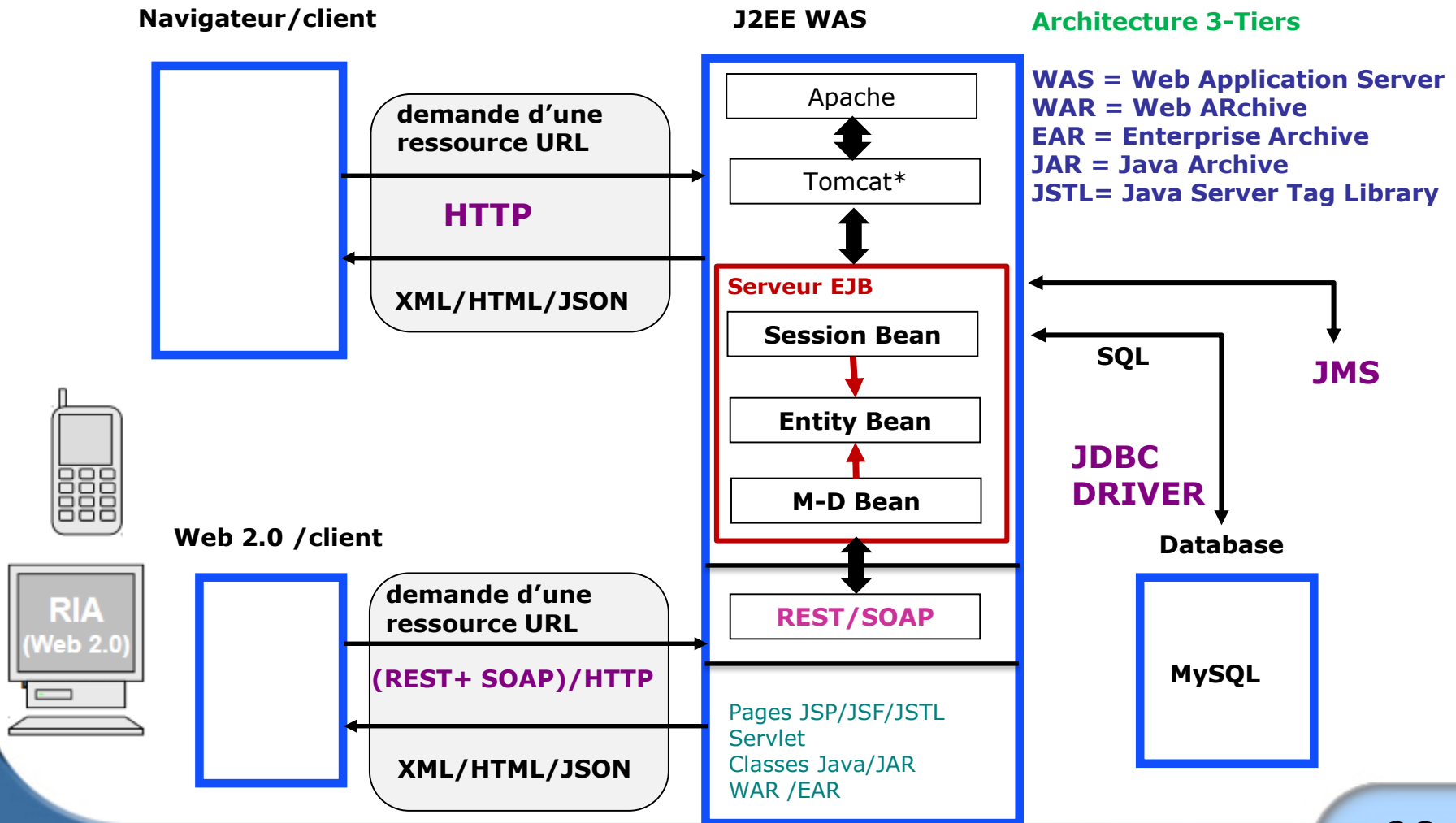
#### Architecture 3-Tiers Tomcat : Servlet Container JSP: Java Servlet Page





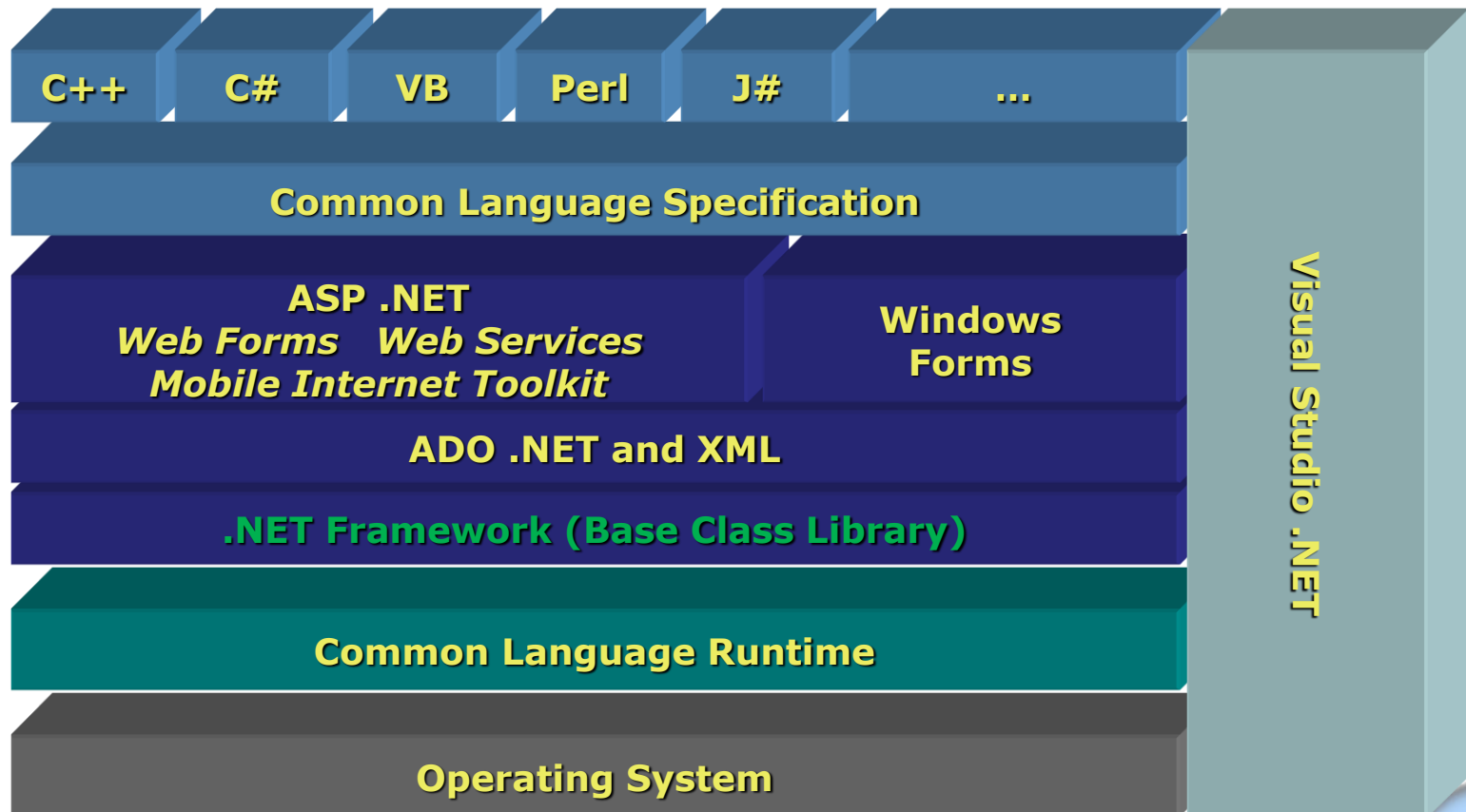


### J2EE

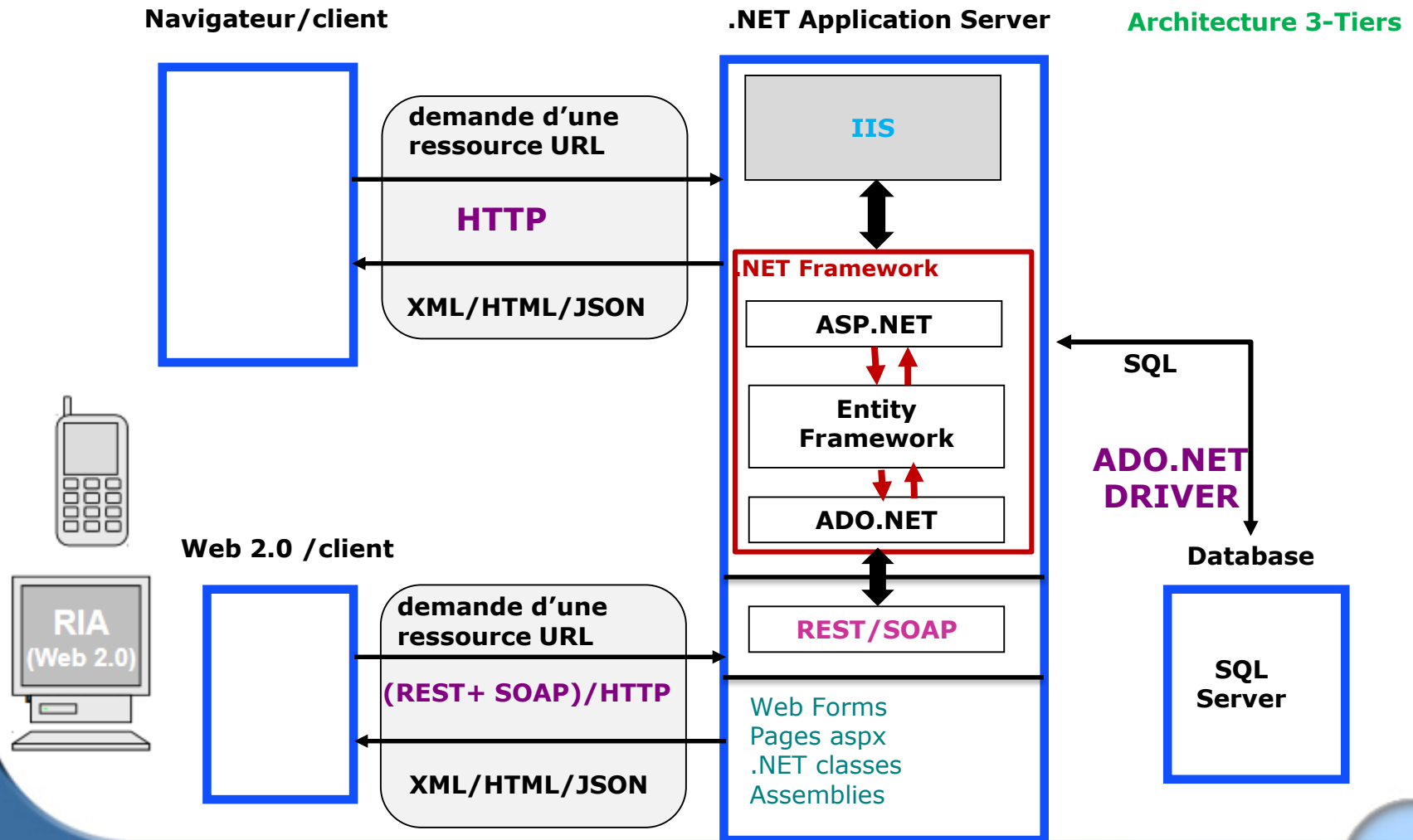


### Microsoft.NET

- NET est une plateforme complète pour développer, déployer et exécuter des Applications Web, Windows, Mobiles et serveur.
  - Modèle de programmation unifiée

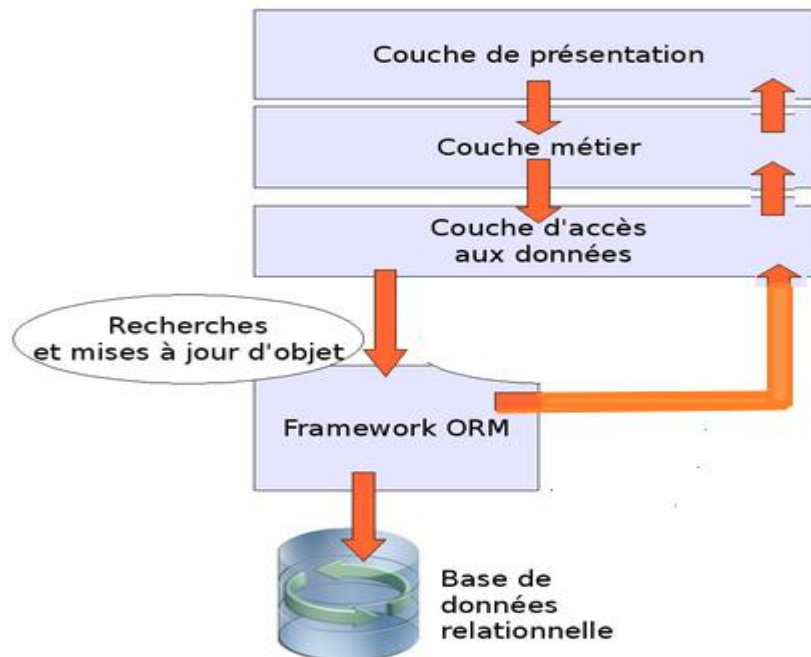


### Microsoft.NET



### *ORM, Object relational Mapping*

- ORM, Couche d'abstraction d'accès aux données
  - Ne plus écrire de requêtes mais des objets
  - La communication entre les mondes objet et relationnel suppose une transformation pour adapter la structure des données relationnelles au modèle objet.
  - Un ORM est une API permettant de gérer les requêtes en base de données. Il permet de s'affranchir du codage SQL, de la gestion des transactions et des connexions.



### ORM, Couche de persistance

#### Classe/Objet

```
@Table(name="EMP")
public class Employee {
    @Id
    private int id;
    @Column(name="EMP_NAME")
    private String name;
    private double salary;
    @Lob
    private byte[] pic;
    // getters & setters
    ...
}
```

#### Relation/Table

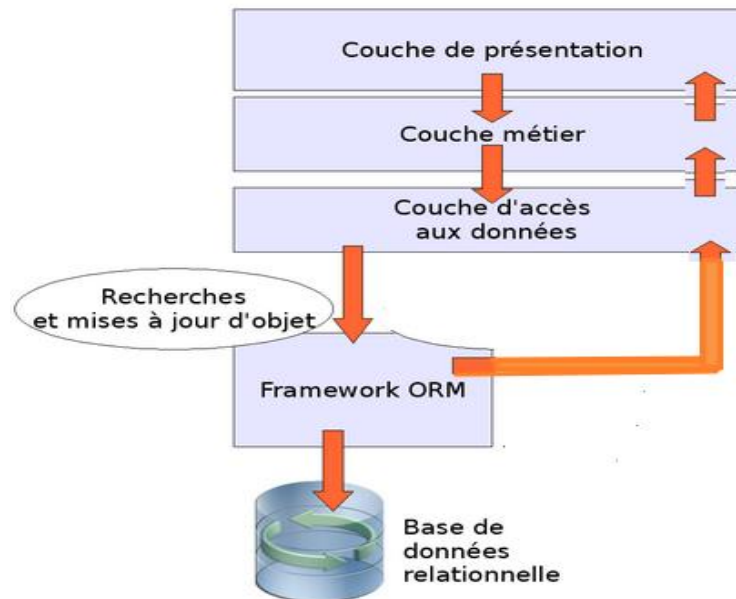
ID	EMPNAME	SALARY	PIC
			« BLOB »

Mapping O/R

- La persistance d'objets désigne le fait que des objets individuels peuvent survivre au processus de l'application.
- Ils peuvent être enregistrés dans un entrepôt de données et récupérés par la suite

### ORM Frameworks

- Java dispose des implémentations **Java Persistence API, Hibernate, Spring**;
- Python possède la librairie Python **Database Object** ;
- Perl dispose du module **Rose::DB::Object** ;
- .NET possède le portage d'Hibernate, un O.R.M du monde Java, et de **Entity Framework**;
- Ruby dispose d'un O.R.M dans son framework **Ruby On Rails**.
- PHP dispose de **Symfony**;



### ORM, JPA

#### Mapping en utilisant les annotations *versus*

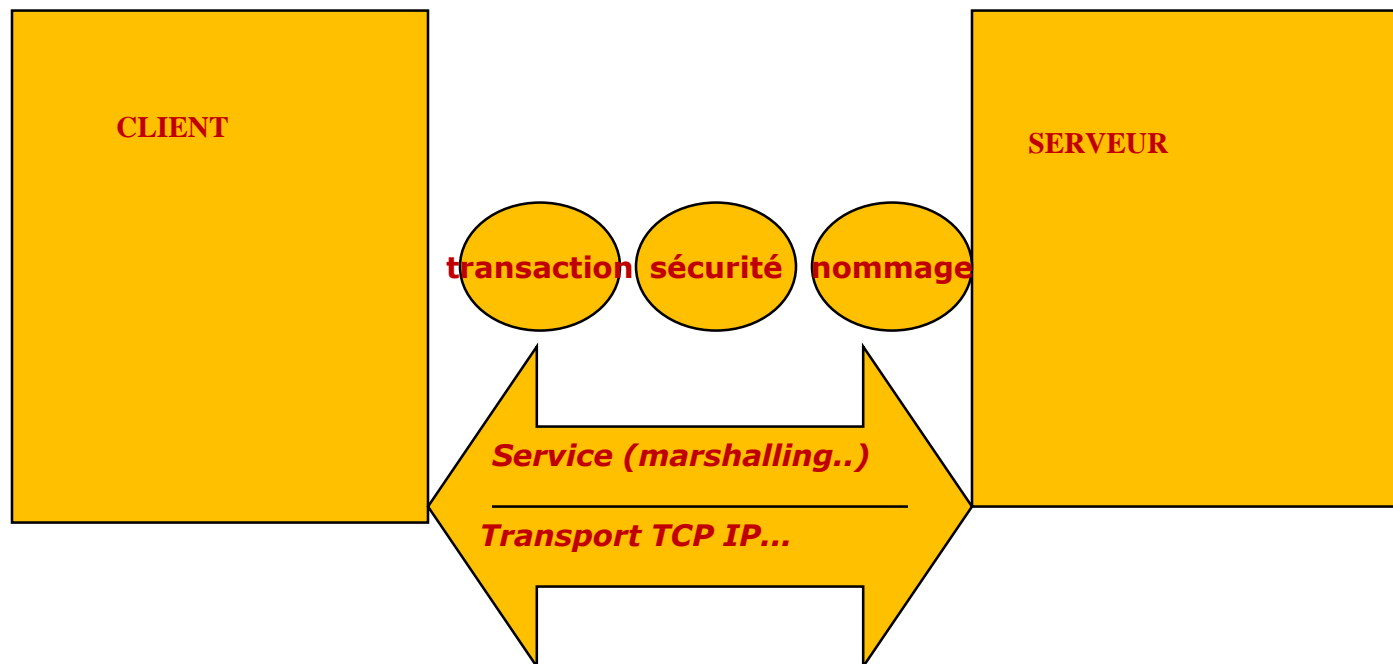
```
@Entity
public class Person {
    @Id
    @GeneratedValue
    private long id;
    @Column(name="LASTNAME")
    private String lastname;
    private String firstname;
    private Address address;
    public Person() {}
    public String getLastname() { return lastname;}
    public void setLastname(String lastname){
        this.lastname = lastname;
    }
    ...
}
```

#### Mapping décrit en XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<entity-mappings version="1.0">
    <entity class="fr.insa.Person">
        <attributes>
            <id name="id">
                <generated-value />
            </id>
            <basic name="lastname">
                <column name="LASTNAME" />
            </basic>
        </attributes>
    </entity>
</entity-mappings>
```

### *Les classiques*

- Il fût un temps avant les objets distribués:
  - des communications client-serveur à base:
    - sockets
    - RPC
    - couches OSI

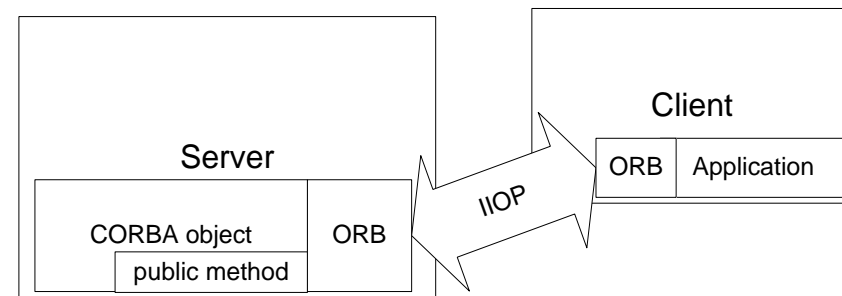
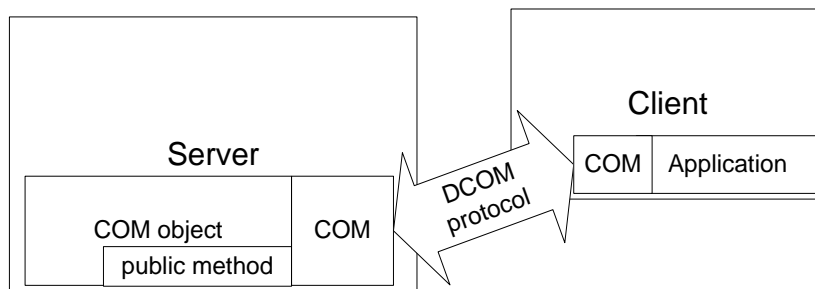
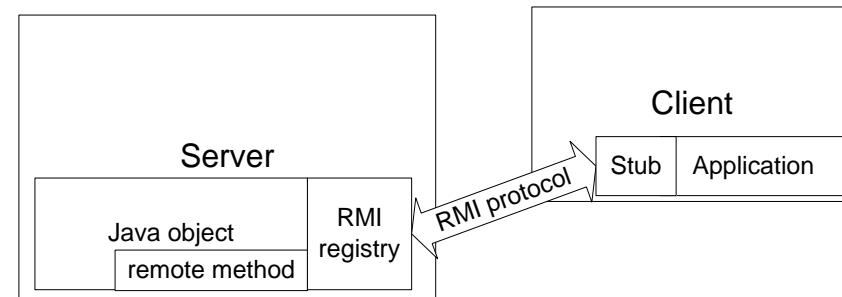
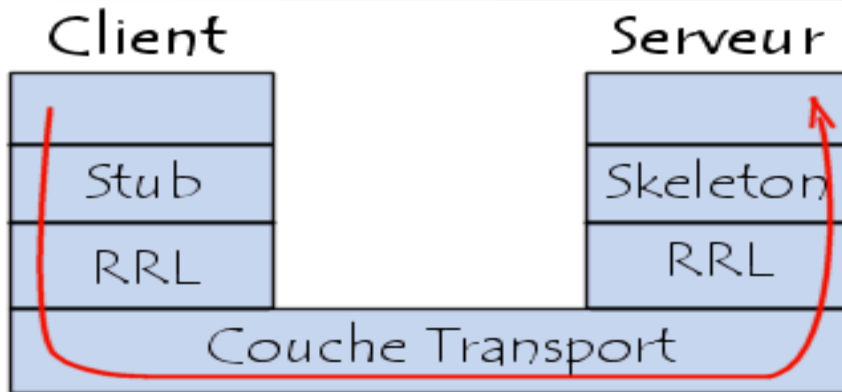




### ***Architectures « célèbres »***

- Les infrastructures distribuées
  - RMI : JRMI/JRMP
  - DCOM
  - CORBA/IIOP
- **Préparent l'avènement des Web services**

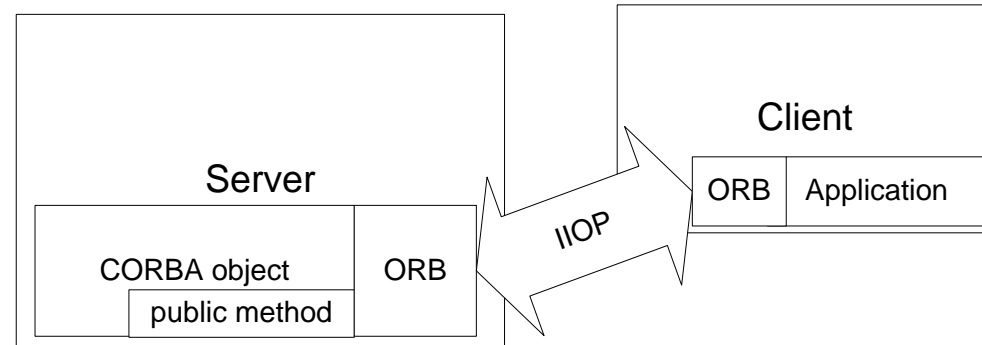
### Principes de communication



### ***CORBA, IDL***

#### **Interface Definition Language**

```
// IDL – file ticket.idl
typedef float Price;
struct Place {
    char row;
    unsigned long seat;
};
Interface TicketOffice {
    readonly attribute string name
    readonly attribute unsigned long numberOfSeats
    Price getPrice (in Place chosenPlace);
    boolean bookSingleSeat (in Place chosenPlace, in string creditCard);
};
```



### ***CORBA, IDL Compiler***

% **idl -B -S ticket.idl** (compilateur) :

- ticket.hh – C++ headers

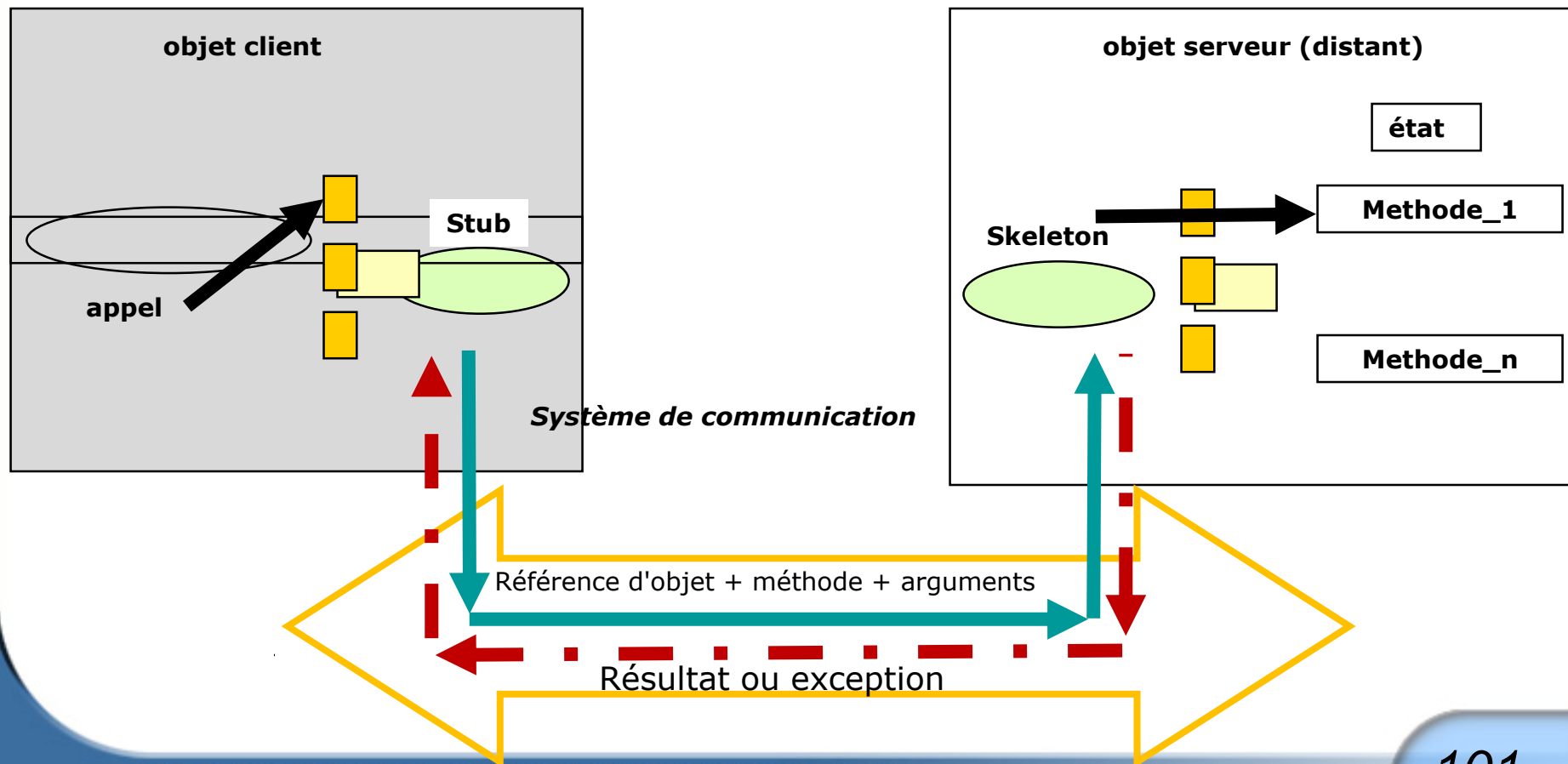
```
#include <CORBA.h>
typedef CORBA::Float Price;
Struct Place { CORBA::Char row; CORBA::ULong seat; };
Class TicketOffice: public virtual CORBA::Object {
public:
    Virtual char* name() throw (CORBA::SystemException);
    ...
    Class TicketOfficeBOAImpl { ... };
```
- ticketC.C // stubs for clients
- ticketS.C // skeleton for server
- TicketOffice\_i.h, .C // Outline of implementation

### **RMI**

- Remote Method Invocation
  - JRMP, Java Remote Method (Invocation) Protocol
  - le code d'un objet client invoque une méthode sur un objet distant
    - Utilisation d'un objet substitut dans la JVM du *client* : le **stub** (souche) de l'objet serveur
  - un objet de réception (Skeleton) effectue les actions suivantes :
    - décode les paramètres encodés
    - situe l'objet à appeler
    - invoque la méthode spécifiée
    - capture et encode la valeur de retour ou l'exception renvoyée par l'appel
  - Vision objet distribué de JAVA

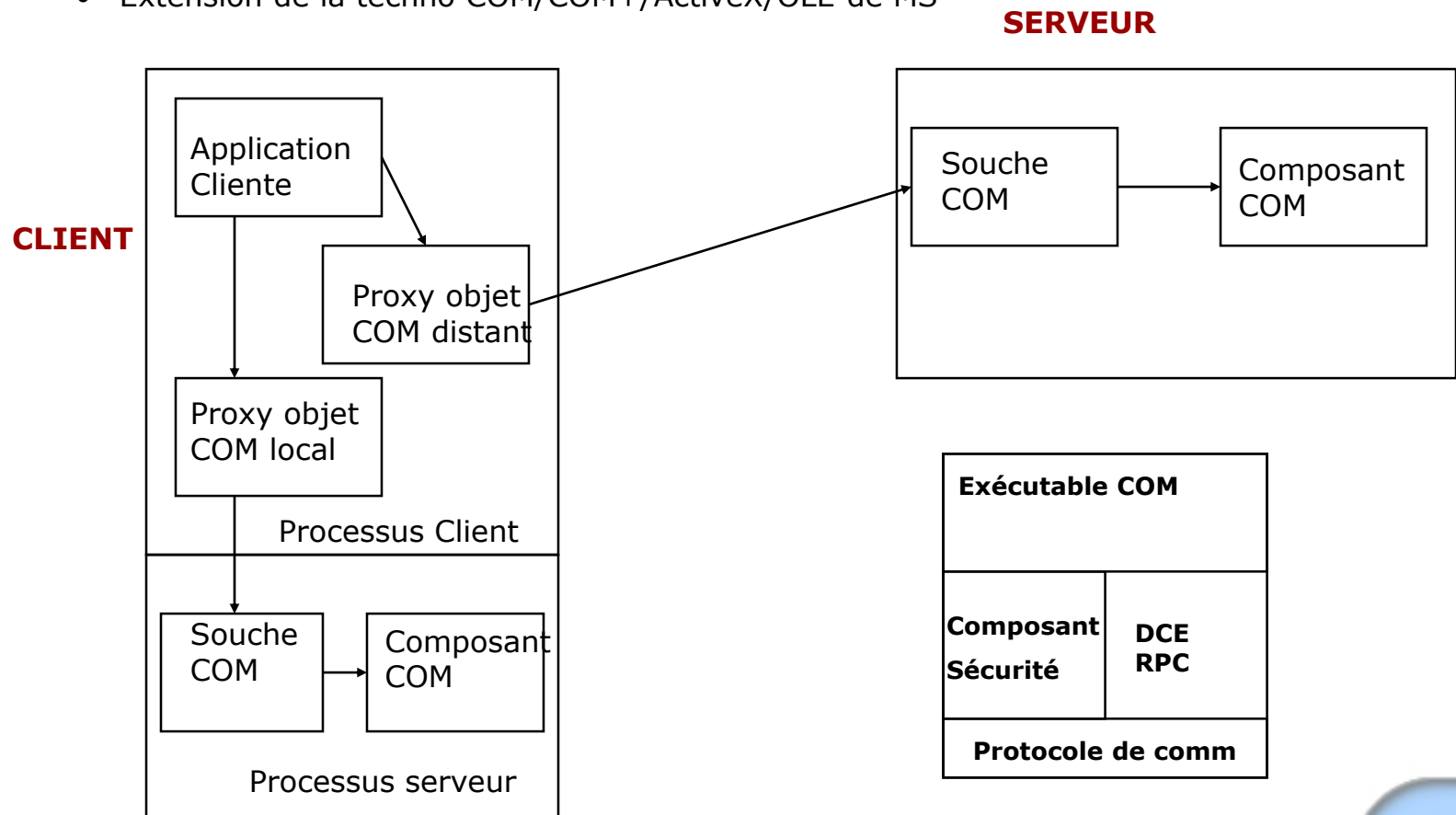
### RMI, mécanisme

Stub = souche(ou talon) client de l'objet serveur



### DCOM

- **DCOM, Distributed Component Object Model**
  - « Plomberie logicielle » de Microsoft pour les objets distribués
    - Extension de la techno COM/COM+/ActiveX/OLE de MS



## ***SOA, Service-Oriented Architecture***

- Architecture orientée Service:
  - Une architecture orientée services est une architecture logicielle s'appuyant sur un ensemble de services simples.
  - Elle permet de composer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants et de décrire finement le schéma d'interaction entre ces services.
  - Lorsque l'architecture SOA s'appuie sur des web services, on parle alors de WSOA, pour **Web Service-Oriented Architecture**.



### SOA

- Les services web sont les successeurs des architectures distribuées :
  - **DCOM, RMI, RPC, CORBA**
- Les services Web utilisent des standards et protocoles ouverts.
  - Les services Web sont basés sur le protocole **HTTP**.
  - Les données échangées entre un client et un service web sont au format texte (**XML/JSON**)
- Les technos WS : HTTP, XML, JSON, SOAP, REST, WSDL, UDDI....
  - **RESTful** est une implémentation particulière des WS

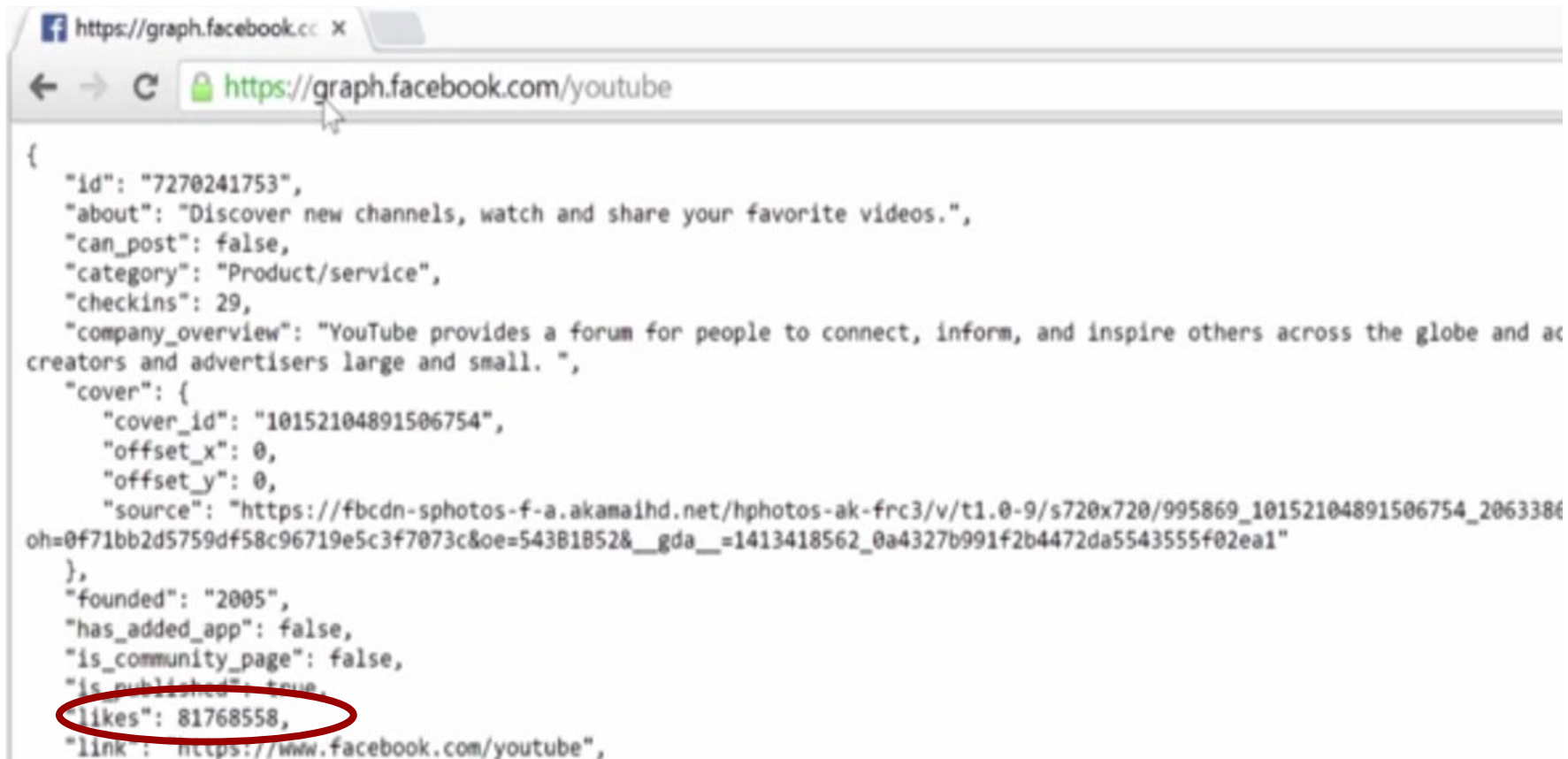
### *RESTFuI*

- Appel d'une page Web:



### RESTful

- Appel d'une ressource RESTful



The screenshot shows a web browser window with the address bar displaying `https://graph.facebook.com/youtube`. The page content is a JSON object representing the Facebook page for YouTube. The `likes` field is circled in red.

```
{
  "id": "7270241753",
  "about": "Discover new channels, watch and share your favorite videos.",
  "can_post": false,
  "category": "Product/service",
  "checkins": 29,
  "company_overview": "YouTube provides a forum for people to connect, inform, and inspire others across the globe and ac
creators and advertisers large and small. ",
  "cover": {
    "cover_id": "10152104891506754",
    "offset_x": 0,
    "offset_y": 0,
    "source": "https://fbcdn-sphotos-f-a.akamaihd.net/hphotos-ak-frc3/v/t1.0-9/s720x720/995869_10152104891506754_2063386
oh=0f71bb2d5759df58c96719e5c3f7073c&oe=543B1852&__gda__=1413418562_0a4327b991f2b4472da5543555f02ea1"
  },
  "founded": "2005",
  "has_added_app": false,
  "is_community_page": false,
  "is_published": true,
  "likes": 81768558,
  "link": "https://www.facebook.com/youtube",
}
```

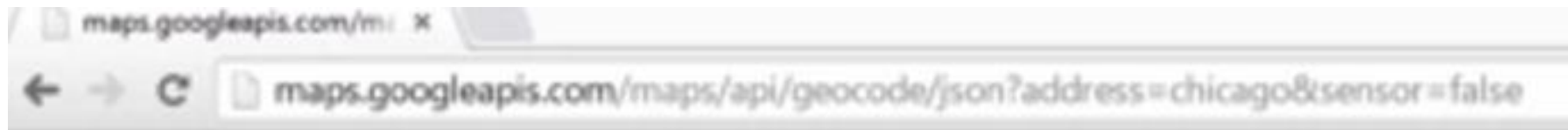
### *RESTful*

- Appel d'une ressource RESTful avec paramètres



### ***RESTful***

- **Constituants de l'appel d'une ressource RESTful**



SERVER = MAPS.GOOGLEAPIS.COM

RESOURCE = /MAPS  
/API  
/GEOCODE  
/JSON

### RESTful

- Appel d'une ressource RESTful avec paramètres



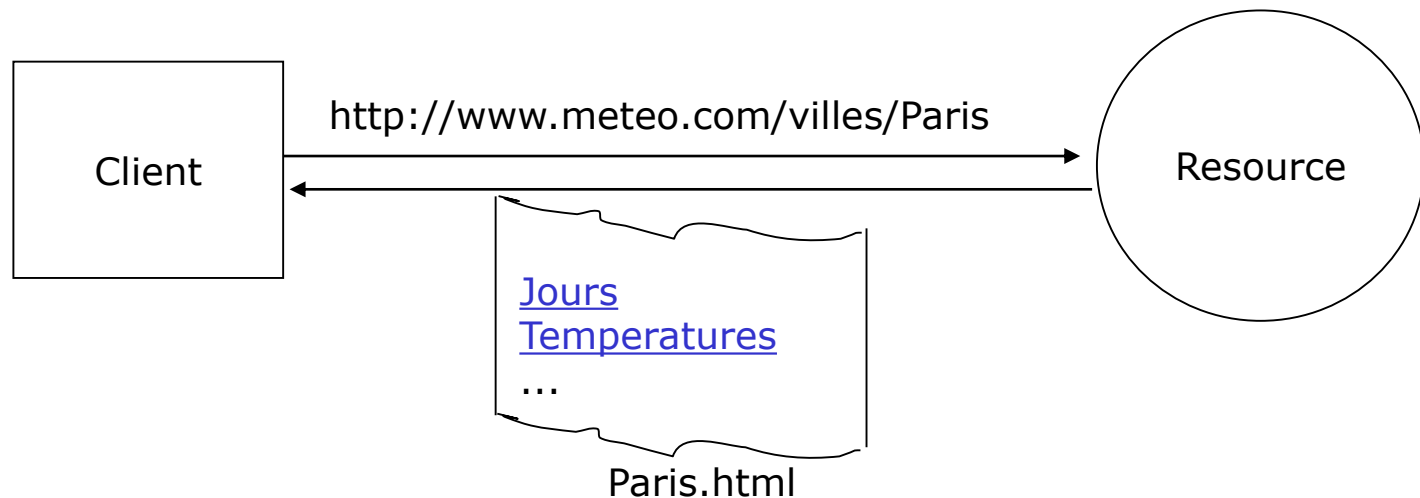
```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "Chicago",
          "short_name" : "Chicago",
          "types" : [ "locality", "political" ]
        },
        {
          "long_name" : "Cook County",
          "short_name" : "Cook County",
          "types" : [ "administrative_area_level_2", "political" ]
        },
        {
          "long_name" : "Illinois",
          "short_name" : "IL",
          "types" : [ "administrative_area_level_1", "political" ]
        },
        {
          "long_name" : "United States",
          "short_name" : "US",
          "types" : [ "country", "political" ]
        }
      ],
      "formatted_address" : "Chicago, Il, USA",
      "geometry" : {
        "bounds" : {
          "northeast" : {
            "lat" : 42.023131,
            "lng" : -87.52404399999999
          },
          "southwest" : {
            "lat" : 41.6443349,
            "lng" : -87.9402669
          }
        },
        "location" : {
          "lat" : 41.8781136,
          "lng" : -87.6297982
        },
        "location_type" : "APPROXIMATE",
      }
    }
  ]
}
```

### ***REST, c'est quoi?***

- Thèse de Roy Fielding en 2000
- Un style d'architecture
- Un ensemble de contraintes
  - Client /serveur
  - Sans états (Stateless)
  - Cache
  - Interface uniforme
- La plus connue des implémentations de REST est HTTP

### ***REST, les principes***

- Une ressource
- Un identifiant de ressource
- Une représentation
- Interagir avec les ressources
  - Exemple avec HTTP : GET, POST, PUT et DELETE



Une partie de la logique métier est effectuée par le client



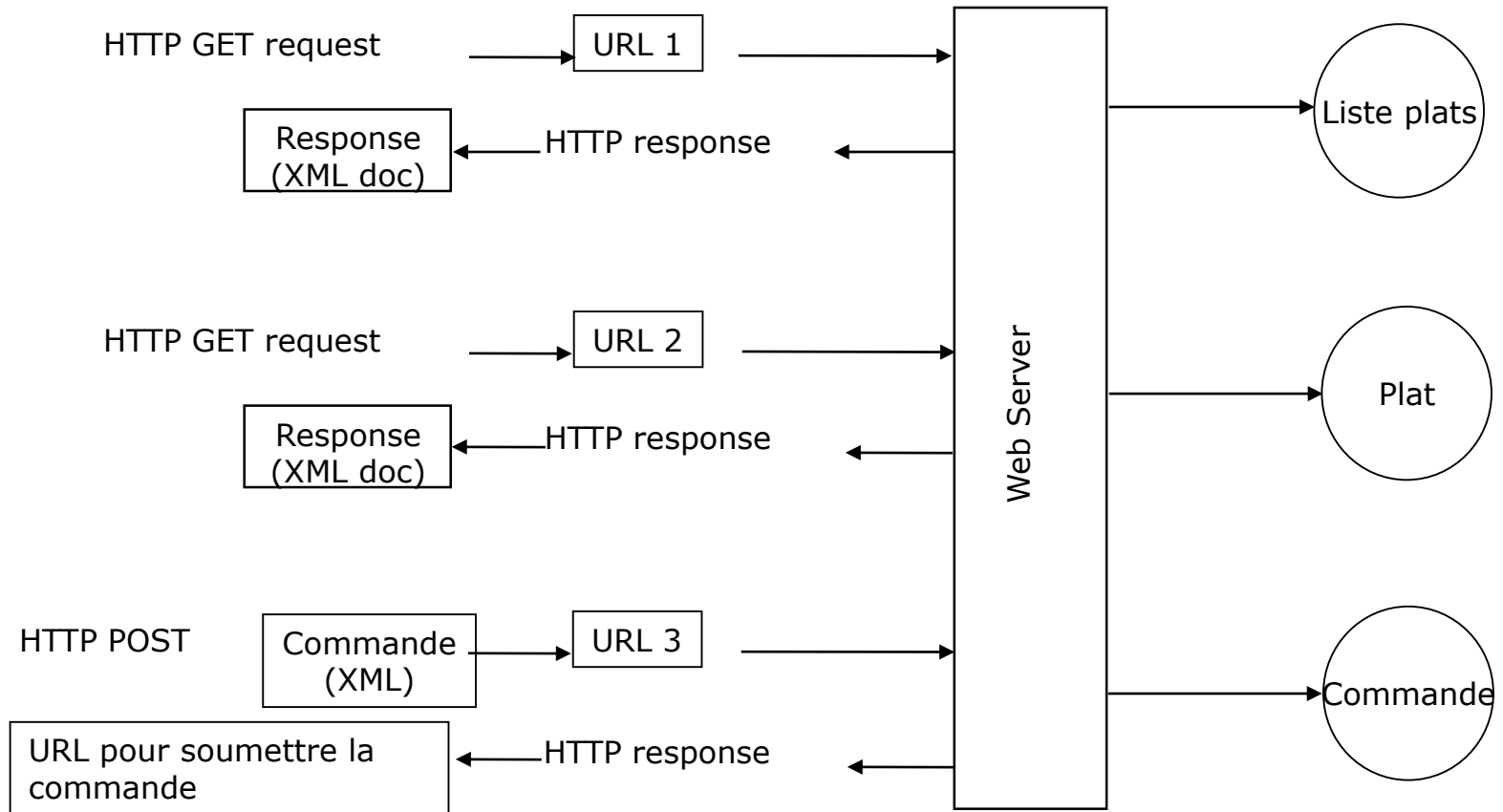
### ***REST, le service***

- Un service REST,
  - Identifier les ressources
  - Définir les URIs
  - Spécifier les méthodes des interfaces
  - Lier les ressources

### ***REST, un exemple***

- Un traiteur propose sur son site plusieurs services à ses clients :
  - Obtenir la liste des plats disponibles
  - Obtenir des informations sur un plat précis
  - Passer une commande

### *REST, un exemple*



### *REST, un exemple*

- La liste des plats est disponible à l'URL suivante :  
<http://www.monresto.com/plats/>
- Le client reçoit une réponse sous la forme suivante :

```
<?xml version="1.0"?>
  <p:Plats xmlns:p="http://www.monresto.com/"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <Plat id="0001" xlink:href="http://www.monresto.com/Plats/0001"/>
    <Plat id="0002" xlink:href="http://www.monresto.com/Plats/0002"/>
    <Plat id="0003" xlink:href="http://www.monresto.com/Plats/0003"/>
  [...]
</p:Plats>
```

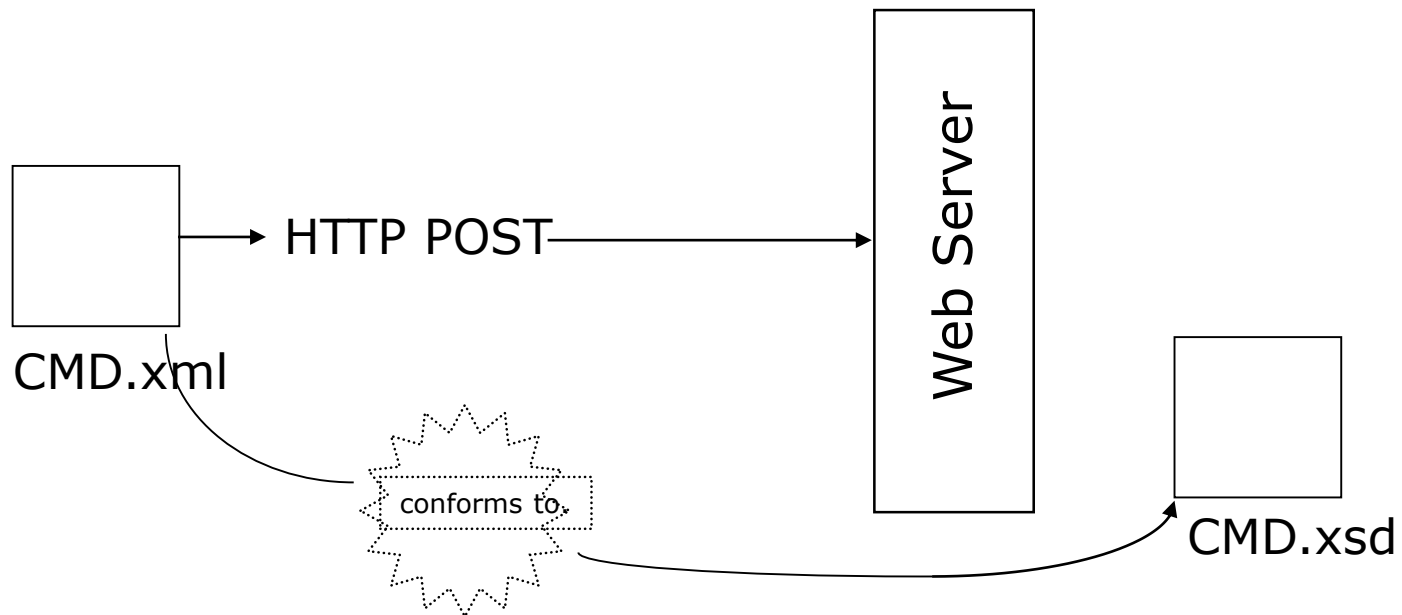
### *REST, un exemple*

- Les détails d'un plat se trouvent à l'URL :  
<http://www.monresto.com/plats/0002>
- D'où la réponse :

```
<?xml version="1.0"?>
  <p:Plat xmlns:p="http://www.monresto.com"
        xmlns:xlink="http://www.w3.org/1999/xlink">
    <Plat-ID>0002</Plat-ID>
    <Nom>Rouleaux de Printemps</Nom>
    <Description>Entrée</Description>
    <Details xlink:href="http://www.monresto.com/plats/00002/details"/>
    <CoutUnitaire monnaie="EUR">3</CoutUnitaire>
  </p:Plat>
```

### *REST, un exemple*

- Le service « Passer commande »
  - Créer une instance de « commande » conforme à un schéma



### *REST, un exemple*

- Le service « Passer une commande » répond par une URL vers la commande soumise

