Technologies Web

Farah Benamara Zitoune
Maître de conférences
IRIT-UPS
benamara@irit.fr

Plan du cours

- Cours 1 : Introduction HTML/CSS
- Cours 2 : Introduction programmation web + javascirpt
- Cours 3 : Introduction PHP
- Cours 4 : Php et Base de données
- Cours 5 : Php session+Cookies
- Si on a le temps : Introduction à Ajax et problèmes de sécurité

Plan du cours 2

I. Architecture client-serveur

II. Page web statique vs. Page web dynamique

III. Notions de base du Javascript

IV. Javascript et formulaires HTML

Plan du cours 2

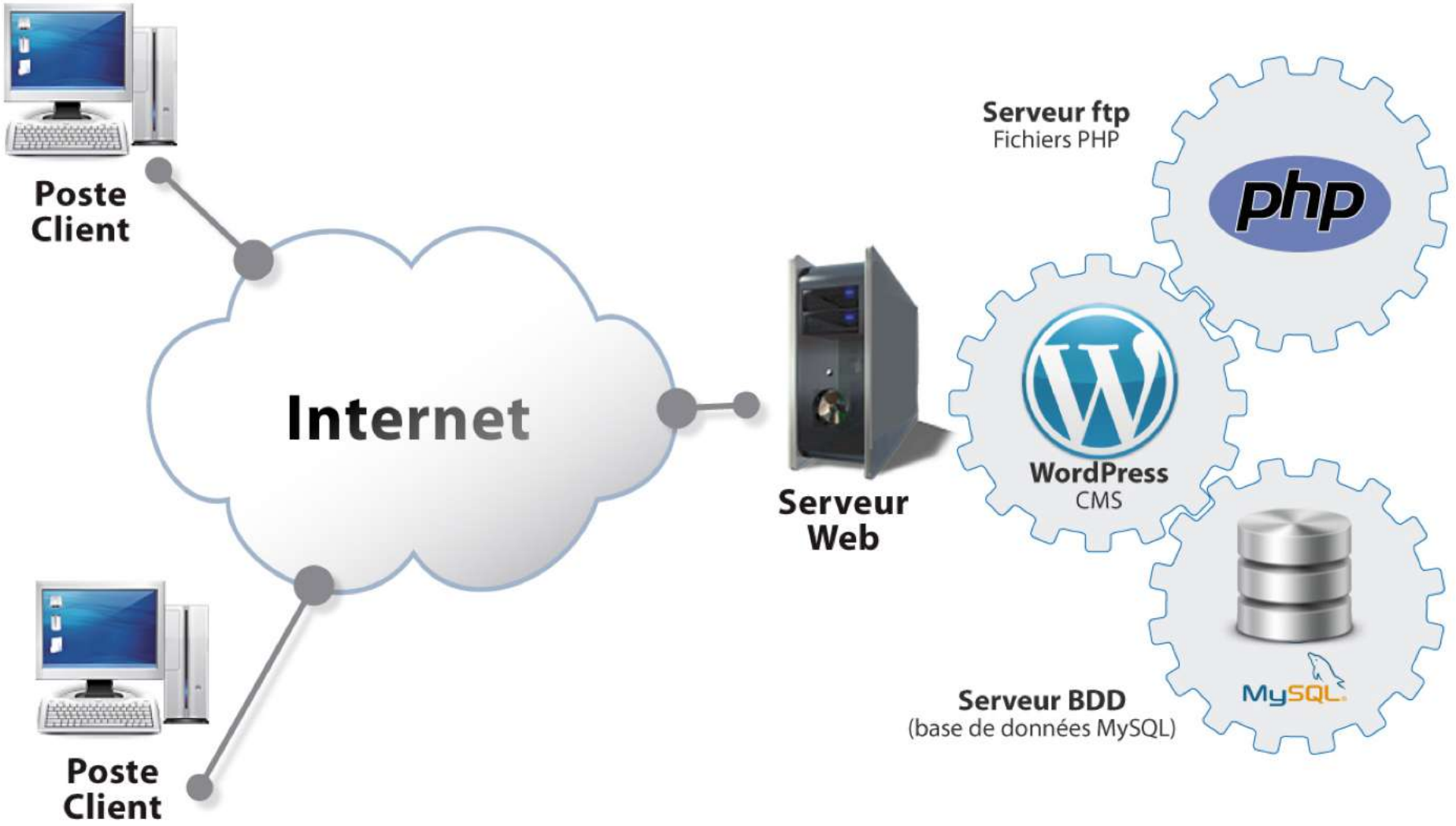
I. Architecture client-serveur

II. Page web statique vs. Page web dynamique

III. Notions de base du Javascript

IV. Javascript et formulaires HTML

Architecture client-serveur



Architecture client-serveur

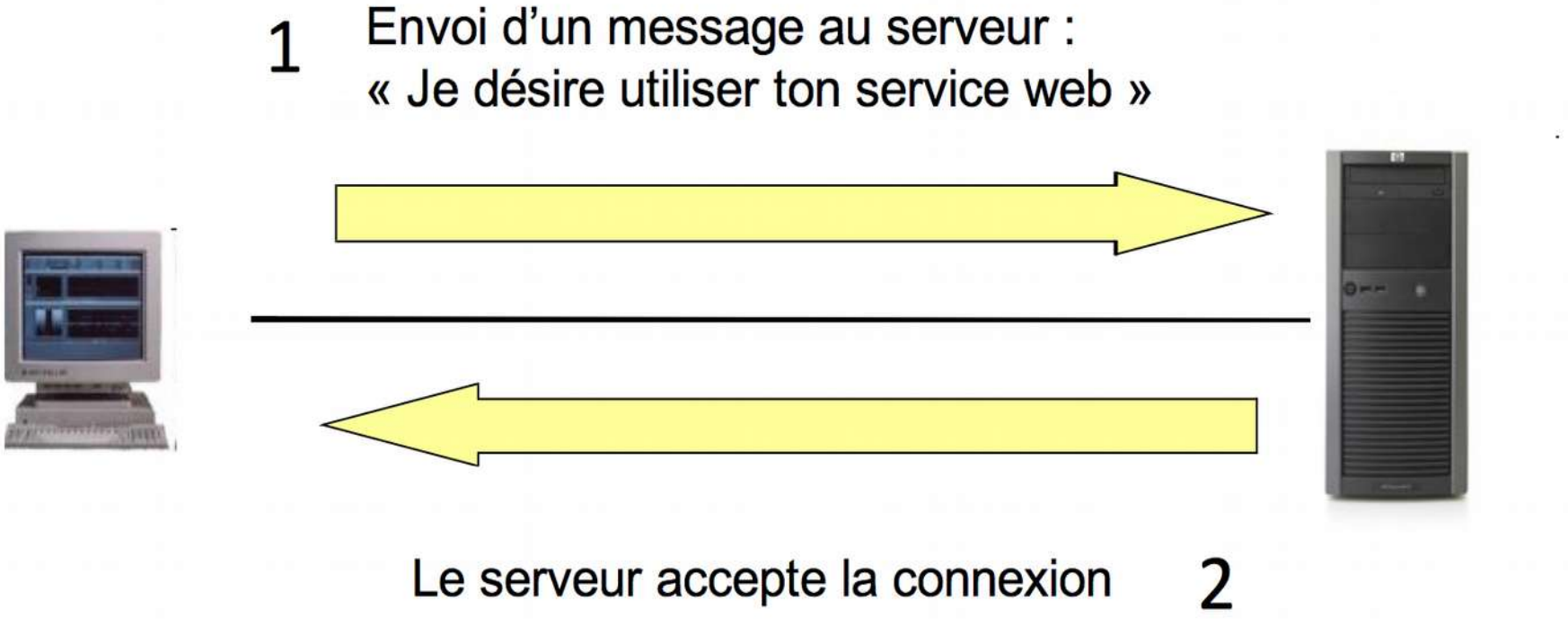
- Deux types d'ordinateurs sont utilisés
  - les serveurs
    - ordinateurs dotés de capacités supérieures à celles des PC (en terme de puissance de calcul, nombre d'entrées-sorties et de connexions réseau…)
    - peuvent répondre à un grand nombre de requêtes
    - partagent leurs ressources (fichiers, périphériques de stockage, périphériques d'impression…)
  - les clients
    - ordinateurs personnels ou appareils individuels (téléphone, tablette), mais pas systématiquement.
    - utilisent les ressources fournies par les serveurs

Architecture client-serveur

- Différents types de serveurs en fonction des besoins à servir
  - serveur web : publication de pages web demandées par des navigateurs
  - serveur de messagerie électronique : envoie des mails à des clients de messagerie
  - serveur de fichiers : stockage et consultation de fichiers sur le réseau
  - serveur d'application : utilisation d'un programme sur un serveur à partir de postes clients (gestion de fabrication, commerce, comptabilité, stock…).
  - serveur d'impression : permet le partage d'imprimantes

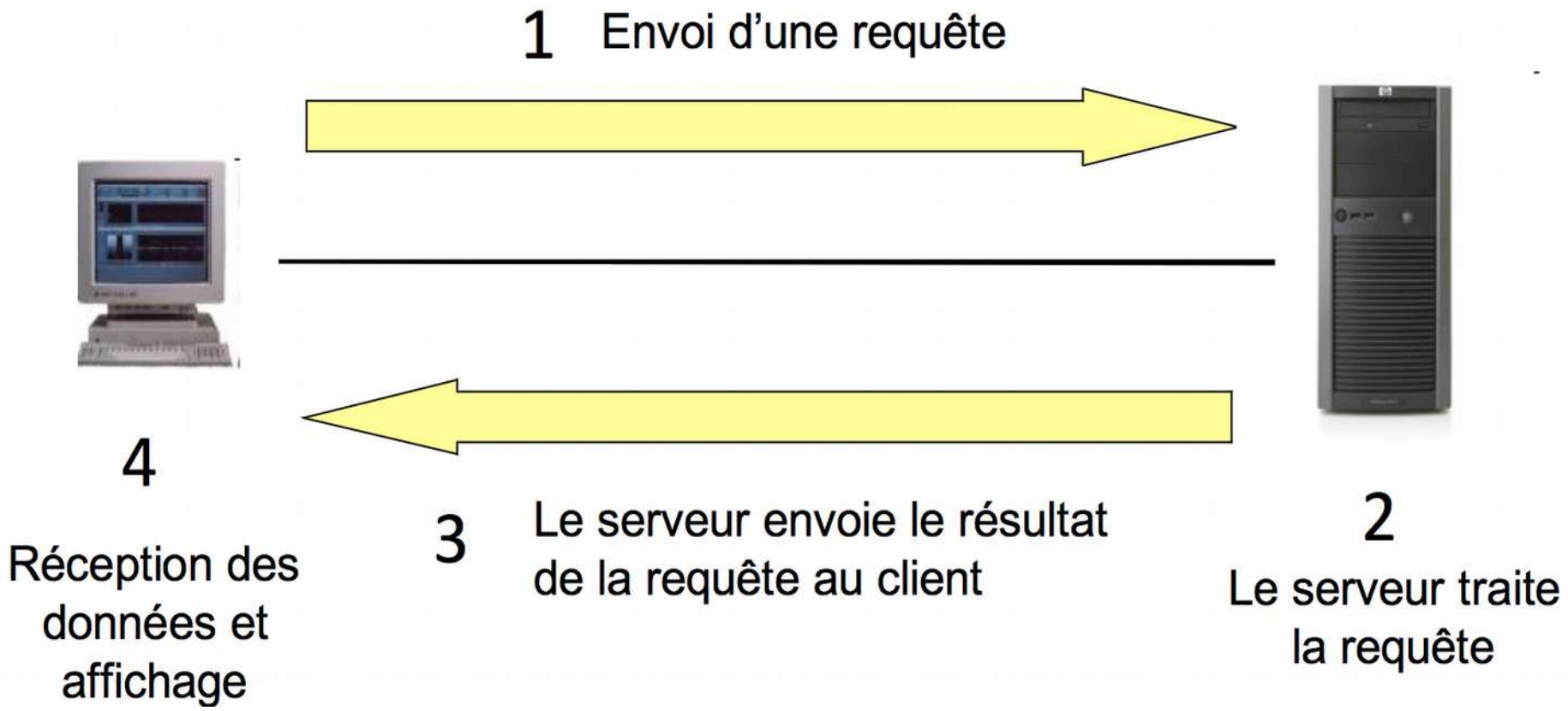
Architecture client-serveur

- Tentative de connexion : elle peut échouer : serveur en panne, service non autorisé ou non lancé, surcharge



Architecture client-serveur

- Traitement de la requête



Architecture client-serveur : le protocole HTTP

Hypertext Transfert Protocol (HTTP)

- protocole de communication client-serveur développé pour le World Wide Web
- couche application modèle OSI (cf. *cours réseau*) utilisé avec la couche transport TCP en général
- **Serveur HTTP** : apache, IIS, tomcat… (utilise par défaut le port 80)
- **Client HTTP** :
  - navigateurs web (Firefox, Konqueror, Internet Explorer, Safari…)
  - Robots d'indexation (GoogleBot, Slurp de Yahoo, MSNBot…)
- **Versions**
  - HTTP 0.9 : protocole de connexion
  - HTTP 1.0 (RFC 1945 et 822) : utilisation d'en-têtes spécialisées
  - HTTP 1.1 (RFC 2616) : meilleure gestion du cache (mémoire)

Plan du cours 2

I. Architecture client-serveur

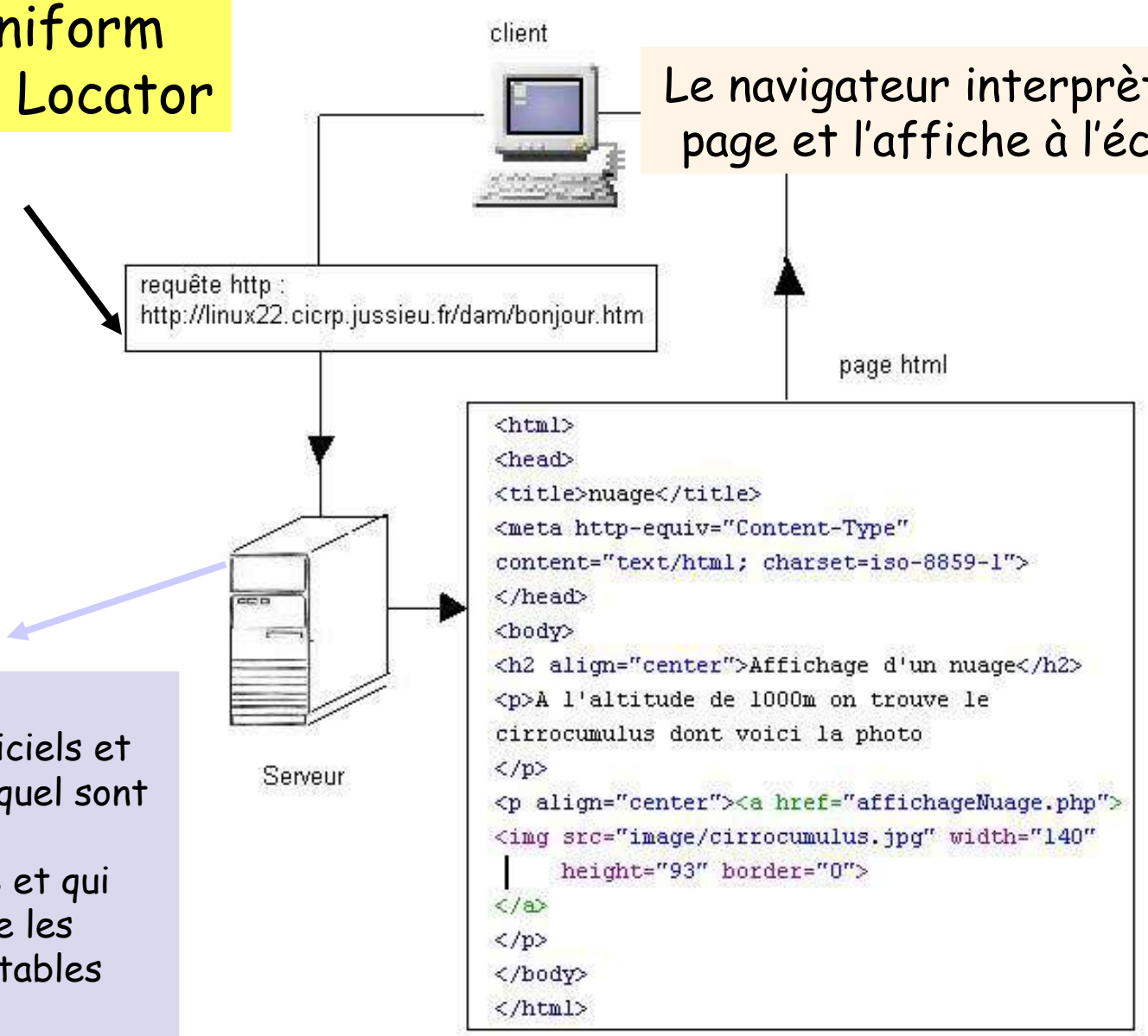
II. Page web statique vs. Page web dynamique

III. Notions de base du Javascript

IV. Javascript et formulaires HTML

Rappels sur l'architecture client serveur

URL: Uniform Resource Locator



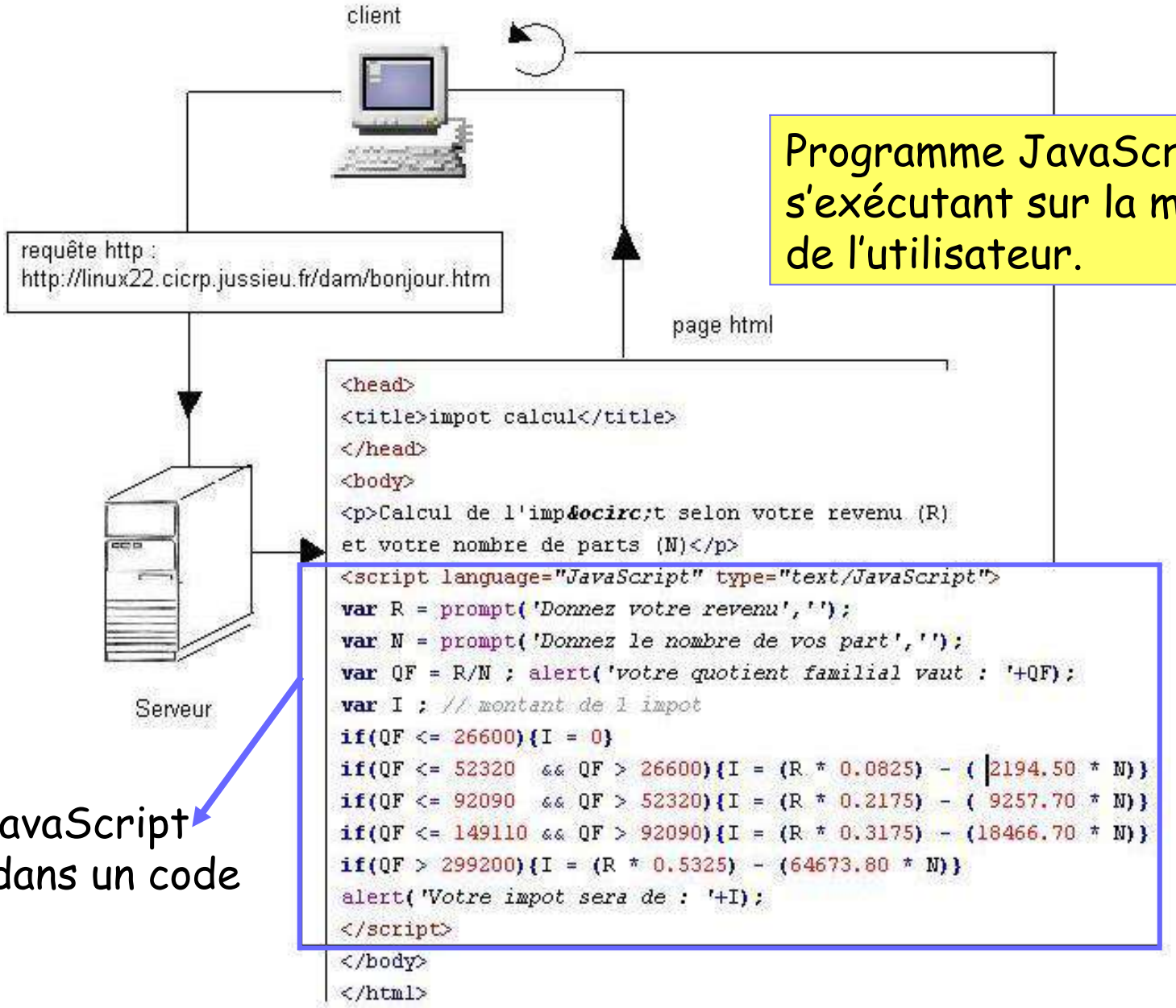
Ensemble des matériels, logiciels et liaisons sur lequel sont implémentés les sites WEB et qui permettent de les rendre consultables sur le WEB

Pages Web Statiques

Les pages web dynamiques

- Le serveur envoie, le client exécute
  - dépendance vis-à-vis du client
  - plus d'interactivité
- Le serveur exécute, le client reçoit
  - indépendance vis-à-vis du client (navigateur)
  - interactivité limitée

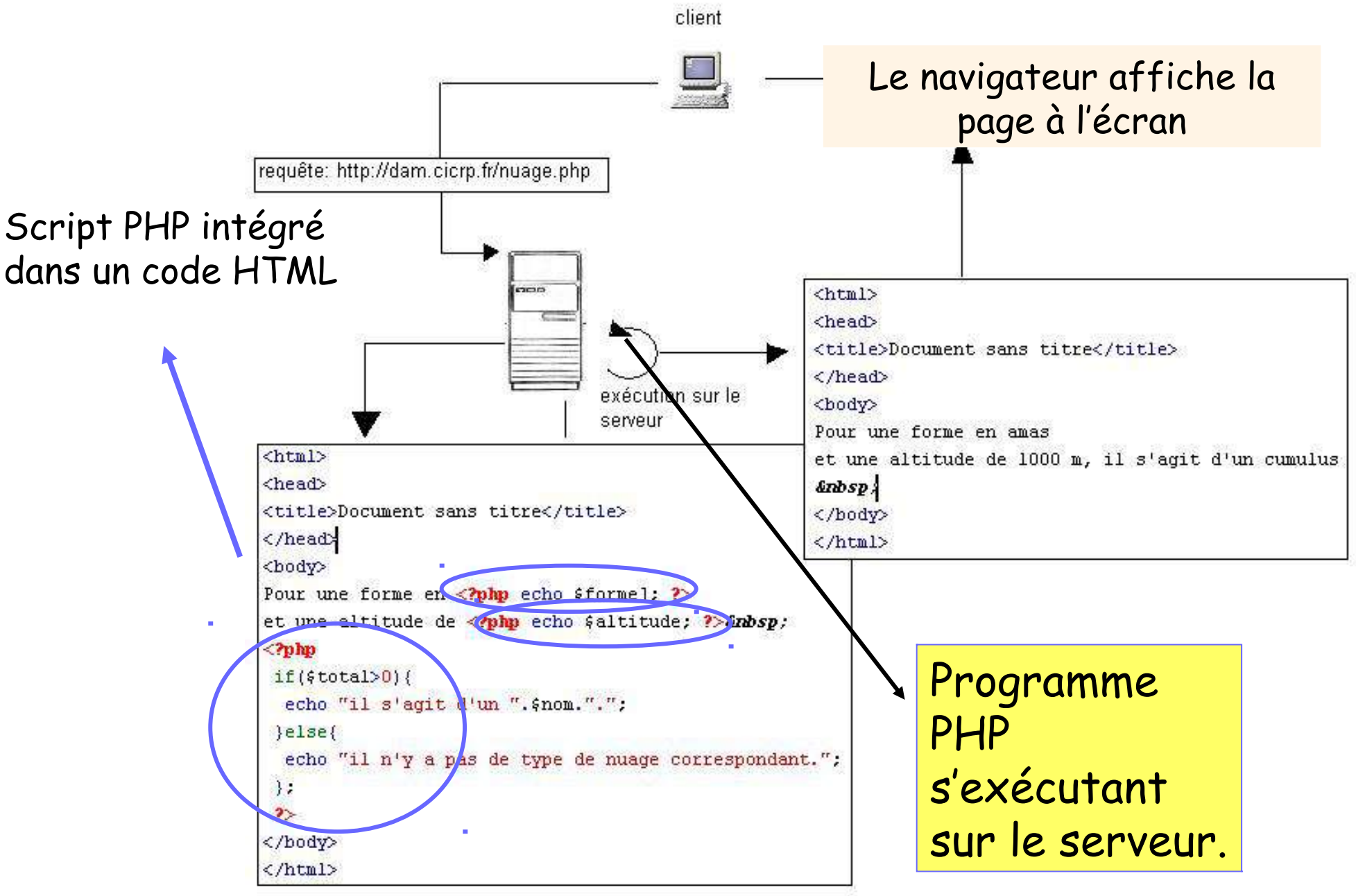
Le serveur envoie, le client exécute



Programme JavaScript s'exécutant sur la machine de l'utilisateur.

Script JavaScript intégré dans un code HTML

Le serveur exécute, le client reçoit



Les pages web dynamiques

- Pour réaliser des pages dynamiques, nous avons besoin de **langages de scripts**.
  - langage de programmation interprété et non compilé comme C++ ou JAVA
- Ces scripts sont conçus pour le développement d'applications web.
- Ils peuvent être intégrés facilement au code HTML

Intérêt du traitement côté serveur

- Réduction du temps de téléchargement puisque le client ne reçoit qu'une simple page HTML (=>diminution du trafic réseau)

- Absence de problème de compatibilité des navigateurs

- Offrir au client des données qui sont dans une base de données, etc.

Technologie pour les serveurs HTTP

CGI : Common Gateway Interface

- Standard industriel qui indique comment passer l'information du serveur HTTP au programme et comment en récupérer le contenu généré.
- Permet de passer des paramètres aux requêtes
- Exécution d'un programme sur le serveur
  - Les informations renvoyées au client sont statiques
  - Des requêtes successives permettent le dynamisme

Scripts côté serveur les plus utilisés

- Perl (Practical Extraction and Report Language)
- PHP (Hypertext Preprocessor)
- JSP (Java Server Pages)
- ASP (Active Server Pages)

Plan du cours 2

I. Architecture client-serveur

II. Page web statique vs. Page web dynamique

III. Notions de base du Javascript

IV. Javascript et formulaires HTML

Main lines

- Inclusion : specific markup

  <script>…</script>
- Main parameters :
  - Text type : **type**
  - Used language : **language**
- OO programming style…
- Call/execution :
  - Simple inclusion or ...
  - HTML event

1st example ;-)

```
<html>
<head>
   <title>My first Javascript</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Hello everybody!");
document.close();
</script>
</body>
</html>
```

- Exercise: center ;-)
- Create alert... with alert(...)
- Pb: we want to control when -> events

With an HTML event !

```
<html>
<head>
   <title>My second Javascript for a simple
   computation</title>
<script type="text/javascript" language="JavaScript">
    function compute(){ //function definition
    var x=1 ; var y = 2; //user defined variables
    var result = x + y;
    document.write("the result is " + result + "<br>");
    document.close();}
</script>
</head>
<body>
Click on the following button to run the script !
     <input type="button" value="calculate"
   OnClick="compute()">
     <!--call associated to an event -->
</body>
</html>
```

- **Pb: on veut nous même entrer les paramètres -> HTML FORM**

Plan du cours 2

I. Architecture client-serveur

II. Page web statique vs. Page web dynamique

III. Notions de base du Javascript

IV. Javascript et formulaires HTML

Les formulaires

Début/fin du formulaire

```
<form >
  Entrez votre prénom : <input type="text" name="prenom" value=”benamara”></input>
  <input type="submit" value="valider">
</form>
```

Les formulaires HTML: rappels

- Zone de saisie monolignes
  - *Syntaxe:* <input type="text" name="nom"></input>
  - *Attributs de la balise input*:
    - Size: taille de la zone de saisie
    - Maxlenght: taille maximale acceptée
    - Type: int, float, date, url, password
- Zone de saisie multlignes
  - *Syntaxe:* <textarea name="nom" cols=?? Rows=??>….Texte….</textarea>
  - *Attributs de la balise textarea:*
    - Cols: nombre de colonnes
    - Rows: nombre de lignes

Les formulaires HTML

- Liste d'options
  - *Syntaxe:*

```
<select name="nom">
    <option value="valeur1" >texte</option>
    <option value="valeur2" >texte</option>
  ...
  </select>
```

  - Eléments de liste sélectionnés par défaut.
    - Attribut **selected** dans la balise <option>
  - Liste à choix multiples
    - Attribut **multiple** dans la balise <select>

Les formulaires HTML

- Liste d'options: un exemple


```
<select name="pays">
 <optgroup label="Europe">
  <option value="fr" selected="selected">France</option>
  <option value="it">Italie</option>
 </optgroup>
 <optgroup label="Asie">
  <option value="ch">Chine</option>
 </optgroup>
</select>
```

Les formulaires HTML

- Boutons d'option ou boutons radio
  - *Syntaxe:*

```
<input type="radio" name="nom"   value="valeur"  {checked} >
```

- Cases à cocher
  - *Syntaxe*

```
<input type="checkbox" name="nom"  value="valeur"  {checked} >
```

Les formulaires HTML

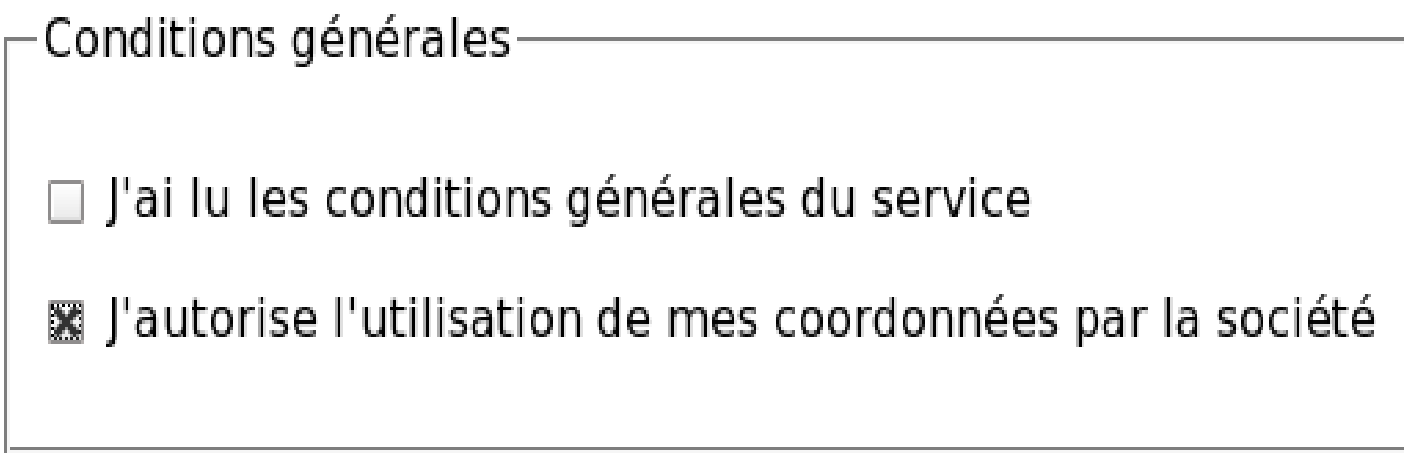
- Boutons d’option ou boutons radio
  - *Syntaxe:*

<input type="radio" name="nom" value="valeur" {checked} >

- Monsieur
- Madame
- Mademoiselle

```
<input type="radio" name="civil" value="m"> Monsieur
<input type="radio" name="civil" value="md"  checked> Madame
<input type="radio" name="civil" value="melle"> Mademoiselle
```

Les formulaires HTML




```
<fieldset>
  <legend>Conditions générales</legend>
  <p><label>
    <input type="checkbox" value="OK" name="condgen" />
    J'ai lu les conditions générales du service
  </label></p>

  <p><label>
    <input type="checkbox" value="OK" name="spam" checked="checked" />
    J'autorise l'utilisation de mes coordonnées par la société
  </label></p>
</fieldset>
```

Les formulaires HTML

- Les boutons d’action ou de commande
  - Bouton d’envoi
    - <input type="submit" name="nom" value="valeur">
  - Bouton d’annulation
    - <input type="reset" name="nom" value="valeur">

With forms!

```
<html>
<head>
<title>My second Javascript for a simple computation</title>
<script language="javascript" type="text/javascript">
function go(){
alert("hello "+maforme.votreprenom.value + " " +
   maforme.votrenom.value)
   }
</script>
</head>
<body>
<form name="maforme">
Enter your surname : <input type="text" name="votrenom"><br/>
Enter your firstname : <input type="text" name="votreprenom">
<input type="button" value="clic here" OnClick="go()">
</form>
</body>
</html>
```

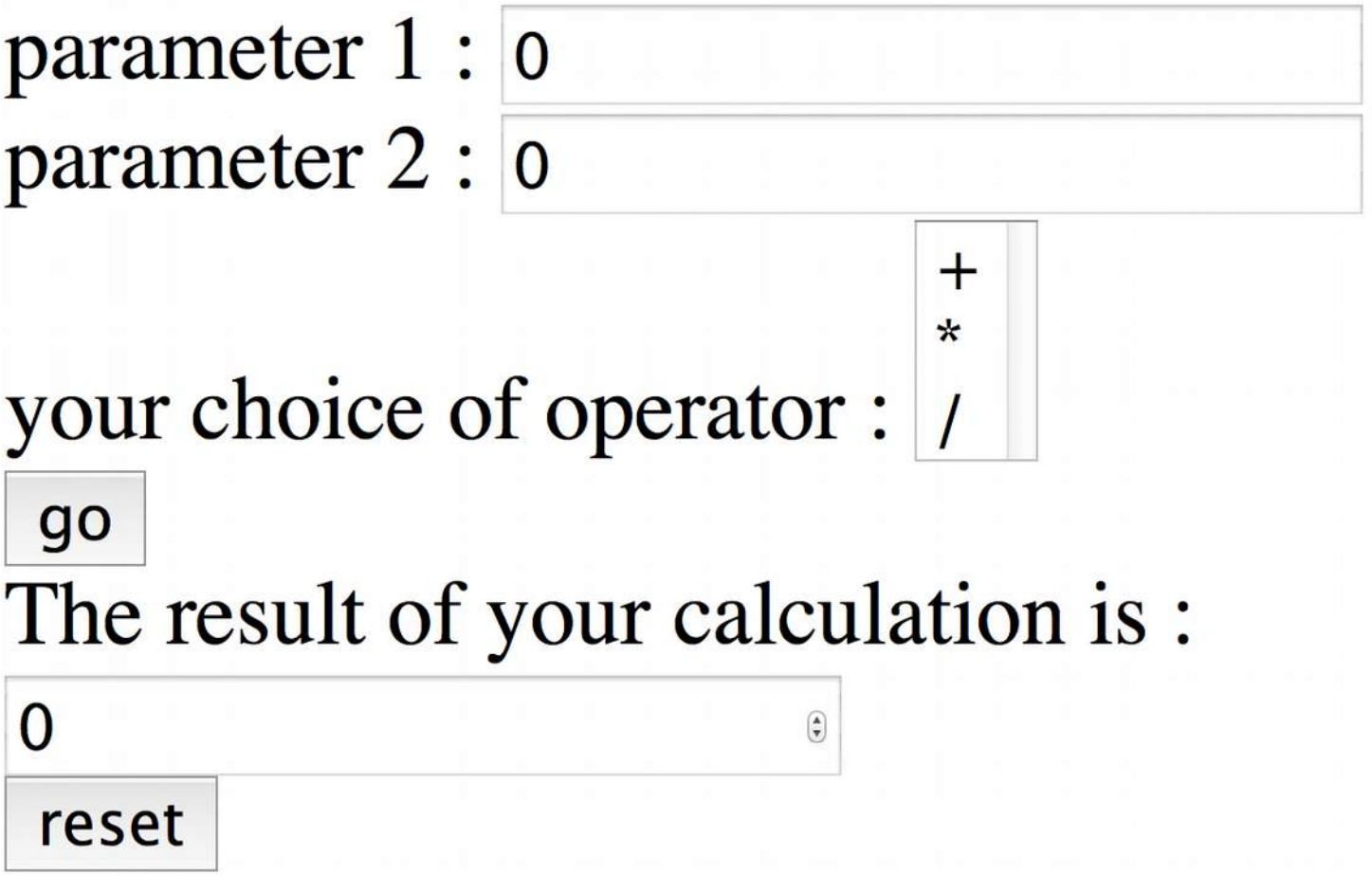
Pb: write inside the page?
Solution: getElementById()

Write inside the page

```
<html>
<head>
<title>My second Javascript for a simple computation</title>
<script language="javascript" type="text/javascript">
function go(){
document.getElementById('myarea').innerHTML = "Hello " +
   maforme.votreprenom.value + " " + maforme.votrenom.value;
  document.getElementById('mazone').innerHTML += '<br/>';
  }
</script>
</head>
<body>
<form name="maforme">
Enter your surname : <input type="text" name="votrenom"><br/>
Enter your firstname : <input type="text" name="votreprenom">
<input type="button" value="clic here" OnClick="go()">
</form>
<div id="myarea"></div><!-- preserve space -->
</body>
</html>
```

Exercice

Créer le formulaire suivant :



Exercice

Compléter l'action associée au bouton Go pour créer une calculette

```
<script language="javascript">
function add(form){ // a form parameter
var a=parseFloat(form.ope1.value);//transform
  text into real value
var b=parseFloat(form.ope2.value);
var result=a+b;
form.result.value=result; //met le resultat
  au bon endroit
}
....etc pour les autres opérateurs

function compute(form){
if (form.operator.value =="1") {add(form);}
else if (form.operator.value == "2")
  {mult(form);}
else div(form);
}
```

Parameters checking

- Objective: check before sending
  - To not overload the server
  - To not overload the network !
- Simple method:
  - HTML forms
  - For each parameter, validity checking
  - Event to control execution

Basic example

```
function check1(s){
var OK=true;
if (s=="")  {alert("put your name inside the box");OK=false;}
else {alert("OK")};
return OK
}
@@@@@ BODY
<form name="myform" action="mailto:benamara@irit.fr"
onSubmit="return check1(document.myform.myname.value)">
put your name inside the box : <input type="text"
name="myname" size="20">
<input type="submit" value="send">
</form>
```

JavaScript focus function

- **goal**: to improve the UX ;-)
  - Parameter checking
  - Focus on the first wrong parameter in a form
- **How**: using the `focus()` function
- **Algo**:
  - If notOK(param.value) {param.focus()}

Exercise

1. Use it for your previous HTML forms
2. Try the function select() !
3. Show me !

Exercise

1. Use it for your previous HTML forms

2. Try the function select() !
3. HTML5 <input required... > allows to force
4. not a standard (safari,etc.)