le cnam

Développement Web
Panorama

P.Graffion

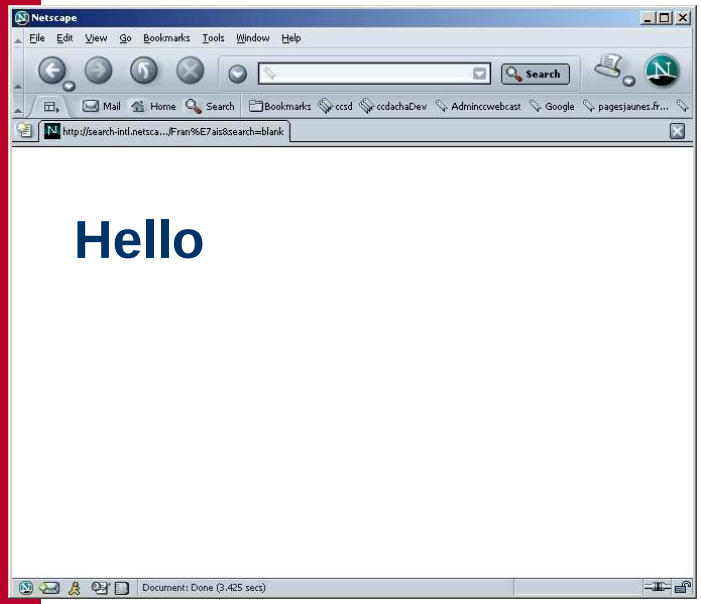
Construire une application Web

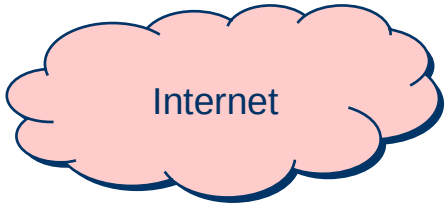
HTML et HTML dynamique
Protocole HTTP
Application Web
Scripts clients et scripts serveurs
Traitement de formulaire

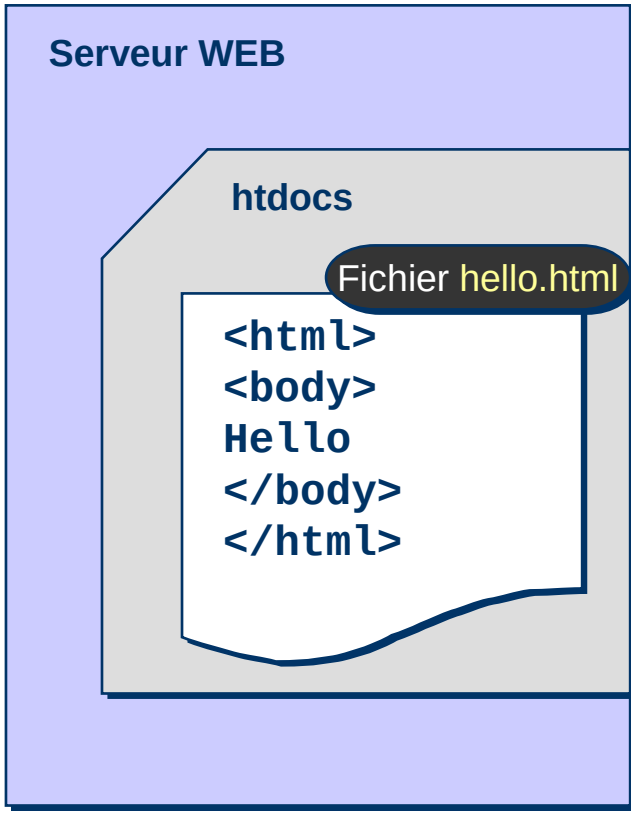
le cnam

HTML et HTML dynamique, rappels

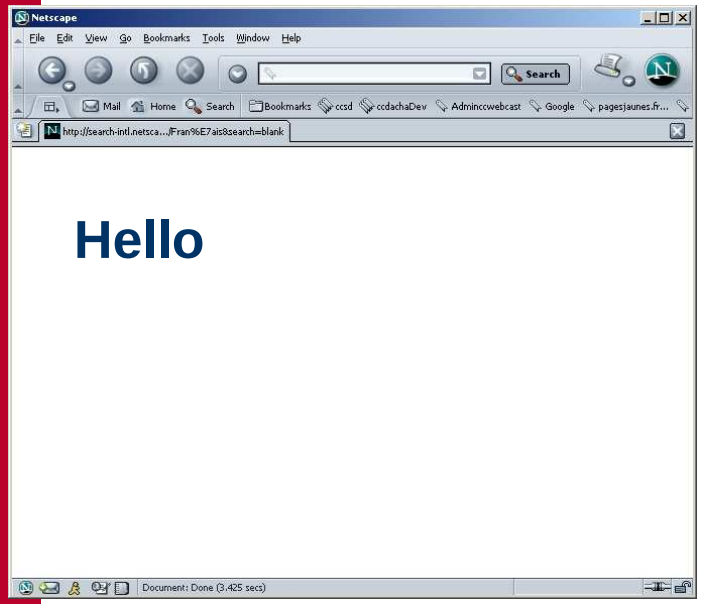
Le modèle HTML statique

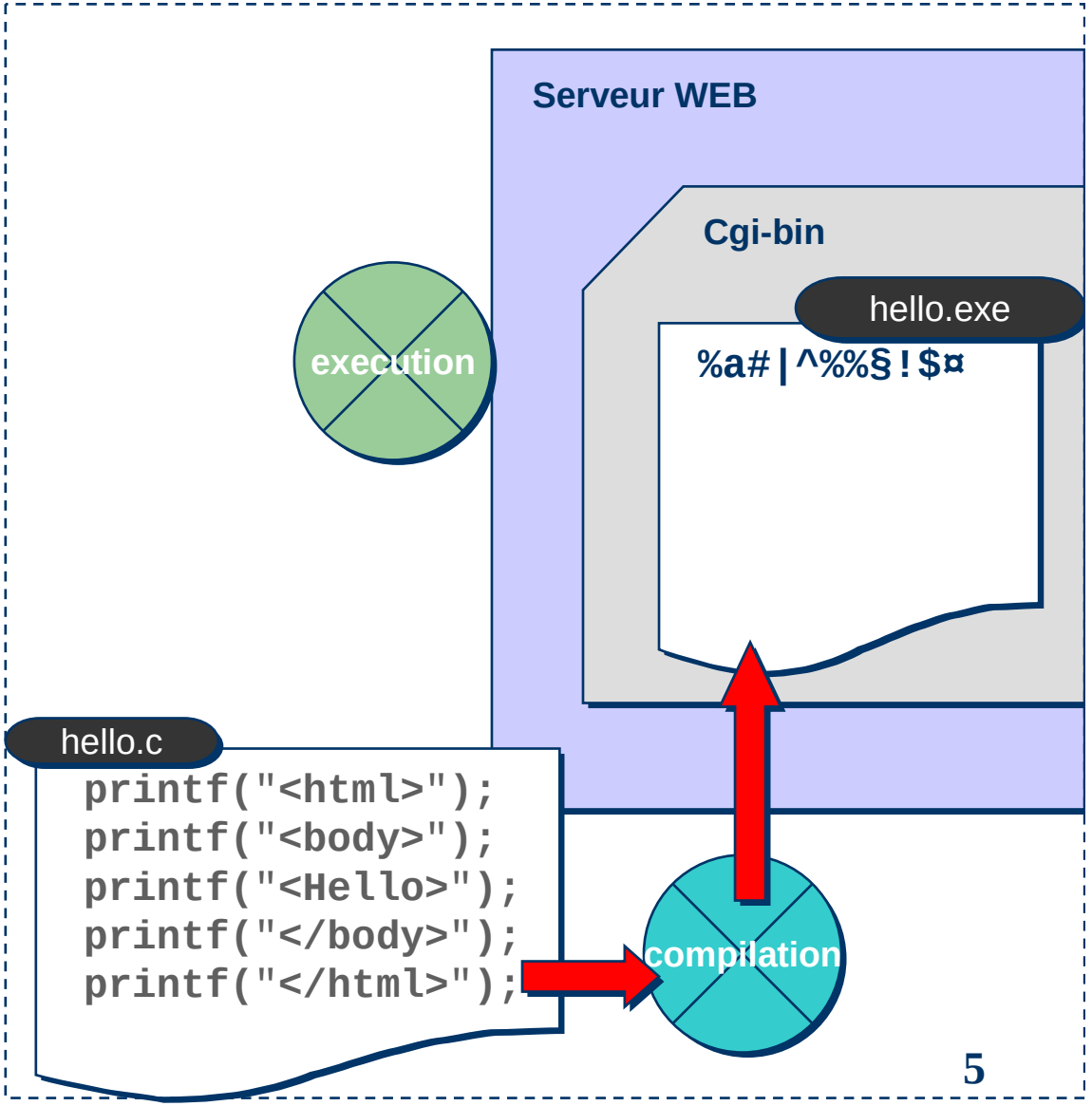




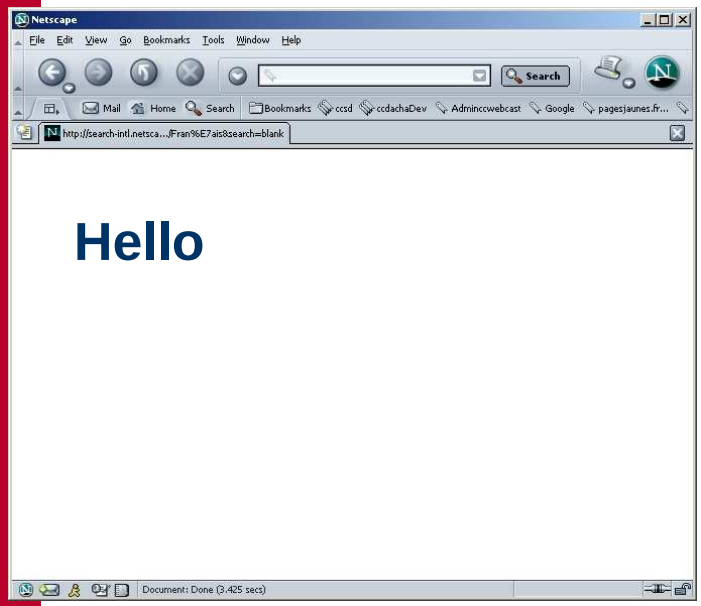


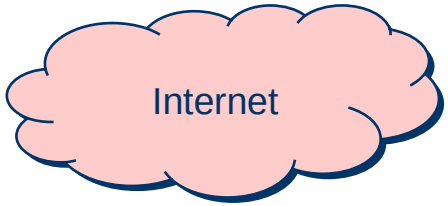
Le modèle HTML dynamique avec les CGI

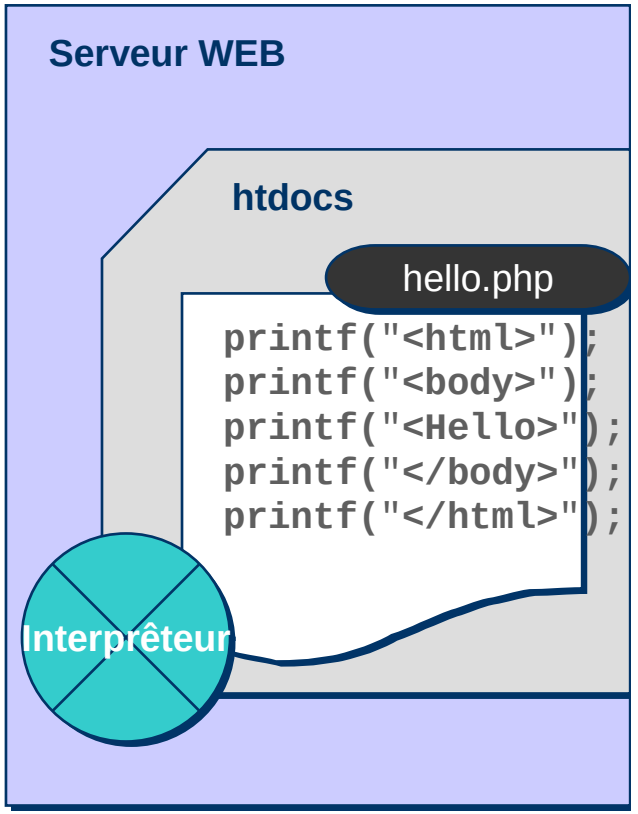




Le modèle HTML dynamique avec un interpréteur







le cnam

Le protocole HTTP

Le protocole HTTP

Transfert de messages avec en-tête décrivant le contenu du message (codage type MIME)

Permet un transfert de fichier (html) localisé par une URL entre un navigateur **(client)** et un **serveur** web (httpd)

RFC1945 (version 1.0,1.1)

http 1.0: Protocole sans état ☹ : pas d'informations gardée entre deux transactions

Le client

ouvre une connection

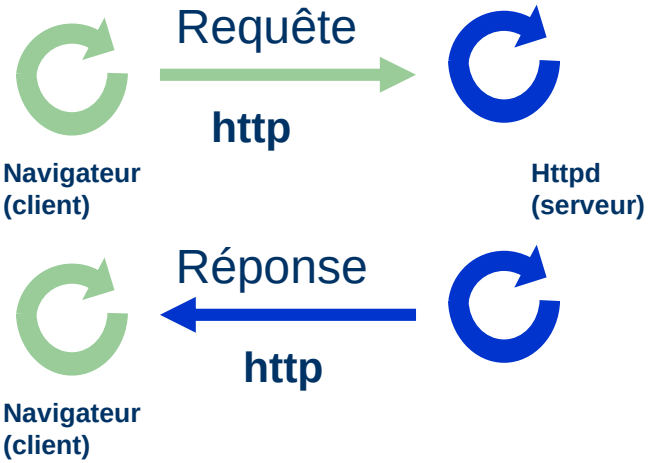
envoie une requête http au serveur

Le serveur

renvoie une réponse HTTP, qui contient d'une manière générale la ressource qui a été demandée par le client.

ferme la connection.

Evolution avec http1.1 (connexion persistante)



HTTP : format d'une requête

- 1ère ligne:Trois parties
  - La méthode (en MAJUSCULE)
    - GET : obtenir une ressource
    - HEAD : tester une ressource

    on obtient des infos sur la ressource (les en têtes (date péremption,taille,existence,…) mais pas la ressource
    - POST : envoyer des données au serveur

    corps de message comprenant le type, la taille des données et les données à traiter par le serveur
  - Chemin (request-URI): url de la ressource ou nom du programme serveur (POST)
  - Version du protocole : HTTP/x.x
    - En 1.1: connection persistante :

    le client envoie des requêtes successives terminées dont la dernière comporte l'en-tête *connection: close* et le serveur ferme la connection quand il traite cette dernière

    Avantage: économie de ressources (ouvrir et fermer une connection TCP/IP est consommateur!!)
- Des lignes d'en-tête (optionnelles): *Header-Name:value*

HTTP : exemple de requête

GET http://www.cnam.fr HTTP/1.0
Accept : text/html
If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

Ex d'en_tête: (46 en http1.1)
From: donne l'e-mail de la personne contrôlant le navigateur. Cela peut poser des problèmes de respect de la vie privée
Referer: URL de l'objet qui amène la requête (URL de la page dans laquelle il y a le lien)
User-Agent: l'identifiant du navigateur. Sert pour adapter la réponse au navigateur
Authorization: permet à un client de s'authentifier auprès du serveur

If-Modified-Since: permet de faire des GET conditionnels

POST http://www.cnam.fr/formpost.php HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Ligne blanche
nom=Dazy&voisins=aaaa

La méthode GET peut aussi être utilisée pour transmettre des données de petite taille :

GET http://www.cnam.fr/formget.php?nom=Berger&voisins=bbbbb HTTP/1.0

HTTP : format d'une réponse

- La ligne initiale d'une réponse HTTP, appelée ligne de statut, est composée de trois parties: la version HTTP, le code statut de la réponse, et une phrase d'explication qui décrit le code statut.
  - On pourra par exemple avoir:
    - HTTP/1.0 200 OK pour indiquer le succès de l'opération ou:
    - HTTP/1.0 404 Not Found
  - Les status
    - 1xx indique un message d'information
    - 2xx indique un message de succès
    - 3xx redirige le client vers une autre URL
    - 4xx indique une erreur du côté du client
    - 5xx indique une erreur du côté du serveur
- **Les champs d'en-tête de la réponse**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la réponse**: il contient le document demandé

HTTP : exemple de réponse

HTTP/1.0 200 OK

Date : Mon, 20 feb 2006 14:37:12 GMT

Server : Microsoft-IIS/2.0

Content-Type : text/HTML

Content-Length : 1234

Last-Modified : Mon, 20 Feb 2006 08:25:13 GMT

Ligne blanche

CORPS de la réponse (html) (1234 caractères)

le cnam

Qu’est ce qu’une application Web?

Fonctionnellement une application Web c'est :

- Un formulaire inséré dans une page Web et permettant à un utilisateur de transmettre des données, de déposer une demande
- Un serveur traitant cette demande et envoyant une réponse, un accusé de réception
- *Exemples*
  - Les réservations de la SNCF
  - La gestion de compte bancaire à la BNP
  - Le formulaire de déclaration d'impôts du Ministère des Finances
  - Etc.

Techniquement une application Web c'est :

- Une page Web contenant un formulaire HTML envoyé au client
- Du code « autonome » s'exécutant sur un serveur HTTP
  - Le programme est auto suffisant, il calcule par exemple la racine carrée d'un nombre
- Ou du code s'exécutant sur un serveur HTTP et accédant à des services « non Web »
  - Accès à des bases de données
  - Accès à des serveurs LDAP
- … Mais peut être aussi du code s'exécutant dans le navigateur
  - Validation, vérification des saisies
  - Ergonomie du formulaire

Une application Web nécessite donc

- Obligatoirement un serveur HTTP !
  - **Apache**, (éventuellement IIS)
- De quoi faire exécuter un programme sur le serveur
  - On peut programmer en C, en Fortran ou en assembleur : ce sont des CGI, ils ne sont pas indépendants de la plateforme d'exécution !
  - On préfèrera utiliser des interpréteurs : **PHP**, Java, Perl qui eux sont indépendants de la plateforme d'exécution
- Souvent une base de données
  - **MySql**, Postgres ou Oracle, etc.
- La possiblité de solliciter un interpréteur localisé sur le navigateur du client
  - Nestcape, IE et leur interpréteur commun : **JavaScript**

le cnam

Scripts clients et scripts serveurs

Scripts et scripts : distinguons bien…

- Ce qui est exécuté sur le serveur : le script PHP
  - Tout comme les ASP (Vbscript), les servlets (Java), les CGI (…)
- Et ce qui est exécuté sur le navigateur : le script JavaScript
  - Tout comme les applets (Java), les ActiveX, les plugins propriétaires

Ces scripts, PHP ou JS s'exécutent lorsque l'utilisateur agit sur un objet du formulaire : bouton, remplissage d'un champ, sélection dans une liste, …

Types de scripts

- La programmation en JavaScript est de type événementielle. On écrira essentiellement de courtes fonctions réalisant des tests ou manipulant l'interface
  - Par exemple, on teste immédiatement, lors de la saisie du champ « âge du candidat », que les caractères entrés sont bien des chiffres et que la valeur est cohérente
- La programmation en PHP est plutôt séquentielle
  - Par exemple, on déroule les instructions permettant d'insérer dans la base de données les valeurs transmises depuis le formulaire

Choisissons …

- Ce qui va s'exécuter sur le navigateur …
- Et ce qui va s'exécuter sur le serveur

Recommandations

- Si vous faites un contrôle de saisie
  - Vous pouvez utiliser JavaScript sur le client
    - Avantage : rapidité, instantanéité, sanction immédiate, pas d'accès réseau
    - Inconvénient : si vous utilisez seulement un contrôle JS votre application doit s'assurer que JS n'est pas désactivé sur le navigateur
  - Vous pouvez utiliser PHP sur le serveur
    - Avantage : impossible de court-circuiter le contrôle
    - Inconvénient : accès serveur, pas de contrôle pas à pas possible, sanction a posteriori
  - Vous pouvez utiliser les deux
    - Avantage : rapidité et contrôle temps réel
    - Inconvénient : double codage

le cnam

Quels langages ?

Langages pour le programmeur

Combien de langages pour le programmeur : 2, 3 ou 4 ?

- On considère qu'il connaît le HTML
- L'interpréteur PHP et le langage PHP !
- Si l'on veut accéder à une base de données, il faudra alors connaître un peu de SQL
- Et si on veut exécuter du code « côté client », on devra connaître le JavaScript

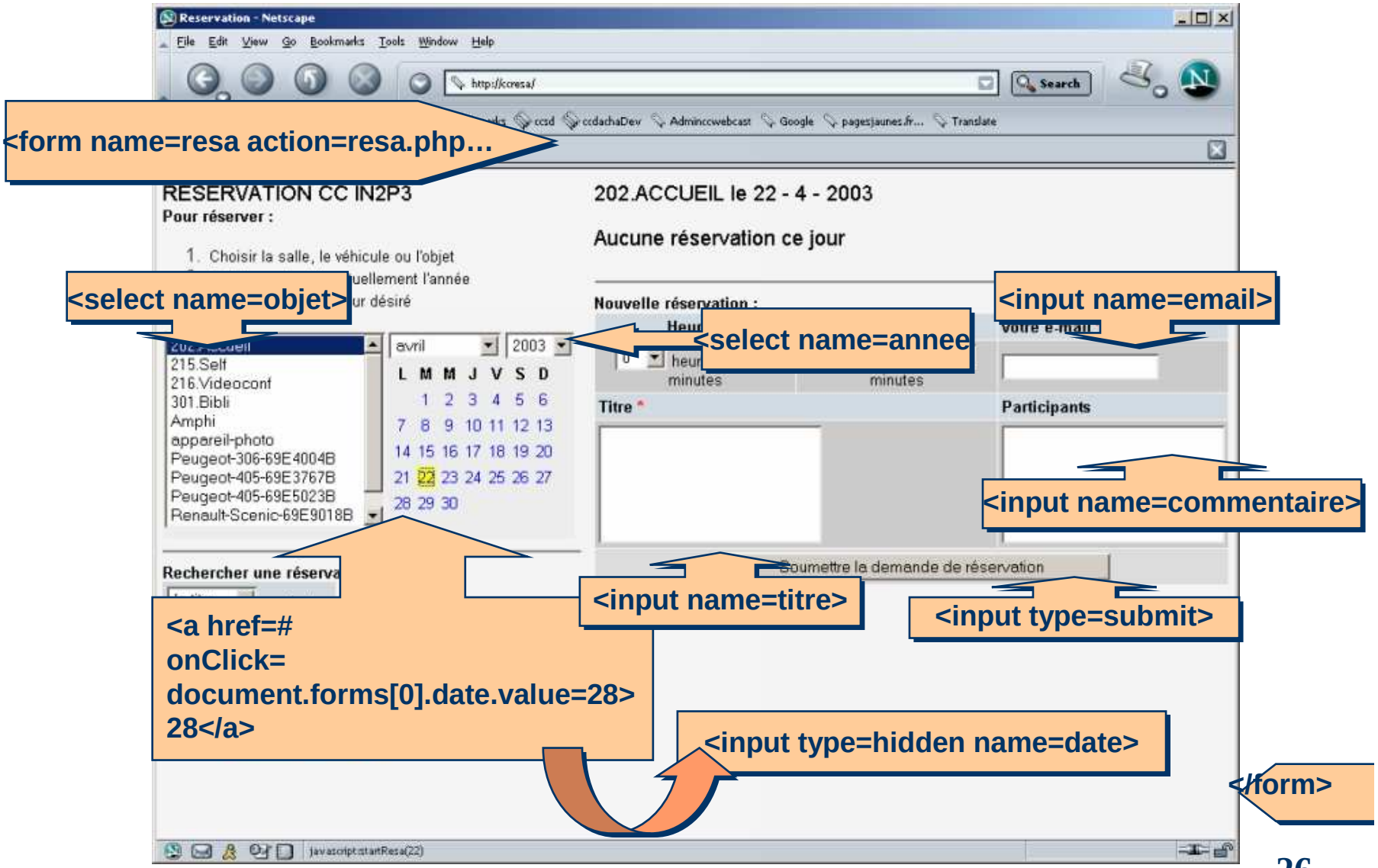
Les langages

- 2 langages de programmation : PHP et JavaScript
  - Syntaxe proche du C, de Java, de Perl
  - Les codes sont insérés dans la page HTML
    - Le code PHP est interprété sur le serveur **avant** l'envoi de la page
      - Le code PHP n'est donc pas visible
    - Le code JS est interprété par le navigateur, soit à réception, soit sur un événement
      - Le code JS peut être visualisé comme le code HTML
  - Pas de réels outils de *debug*
- 1 langage de composition de page Web : HTML
- 1 langage d'interrogation de la base de données : SQL

le cnam

Traitement de formulaire

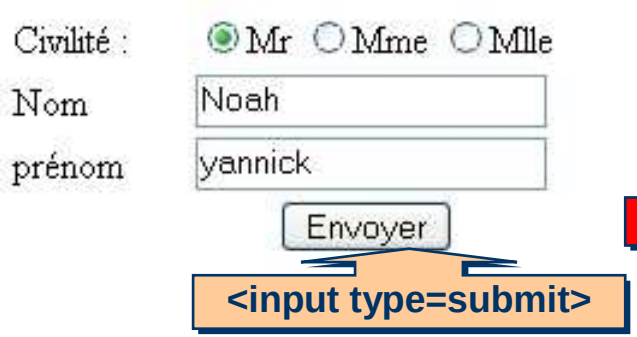
Un formulaire …



Traitement d'un formulaire

- Le formulaire est décrit dans une page HTML qui peut être statique ou dynamique…
- Le programme PHP décrit par le paramètre « action » de la balise « form », est invoqué lors du clic sur le bouton « submit »
- Le programme PHP doit récupérer les valeurs saisies dans les différents champs du formulaire pour les traiter
- C'est le nom, au sens HTML, défini par le paramètre « name », qui va être utilisé pour créer la variable PHP

Récupération de la valeur des champs



submit

edit.php

```
<?
$prenom = $_REQUEST["prenom"];
echo "Bonjour".$prenom;
?>
```

edit.html

```
<form
   name=xyz
   action=edit.php>
Pr&eacute;nom :
<input name=prenom>
…
</form>
```

☺ **Cette méthode fonctionne quelque soit la méthode (GET ou POST) choisie pour la transmission du formulaire.**

Page HTML avec un script « client »

editMail.html

```
<html>
<head>
<script> //javaScript
function verifMail(){
  if (document.forms[0].mail.value.length < 6 ||
      document.forms[0].mail.value.indexOf("@") < 0 ||
      document.forms[0].mail.value.indexOf(".") < 0) {
      alert ("mail incorrect");
      return false;
  }
  else return true;
}
</script>
</head><body>
<form
   name="xyz"
   action= "recevoirMail.php"
   method="post"
   onSubmit="return verifMail();" >
Mail :
<input name="mail">
… </form>
</body></html>
```

Bloque l'appel au programme PHP en cas d'erreur

Script serveur PHP

recevoirMail.php

```
<?
if (isset($_REQUEST["mail"])){
        $mail = $_REQUEST["mail"];
        if (strlen($mail) < 6 ||
            ! strpos($mail,'@') ||
            ! strpos($mail,'.') ){
               print ("<font color=red><b>Mail incorrect !</b></font> ");
               print ("<a href=lireMail.html>recommencez</a>");
         } else {
            print ("Adresse mail : $mail");
         }
}else{
        print ("<h1>INTERDIT !");
}
?>
```

Au fait ! Dans quel cas passe-t-on dans cette branche ?

Conclusion

- Tester côté client est plus ergonomique
  - Sanction immédiate, pas de nécessité de regénérer le formulaire, on peut tester chaque saisie en temps réel (onBlur, onFocus)… mais impose que le client autorise JS
- Tester côté serveur marche à tous les coups
  - Mais ne tester que côté serveur, oblige à régénérer le formulaire, à conserver le contenu des champs qui étaient déjà remplis correctement…
- Tester des 2 côtés c'est parfait, mais c'est plus de travail !