

Couche application

HTTP (suite)

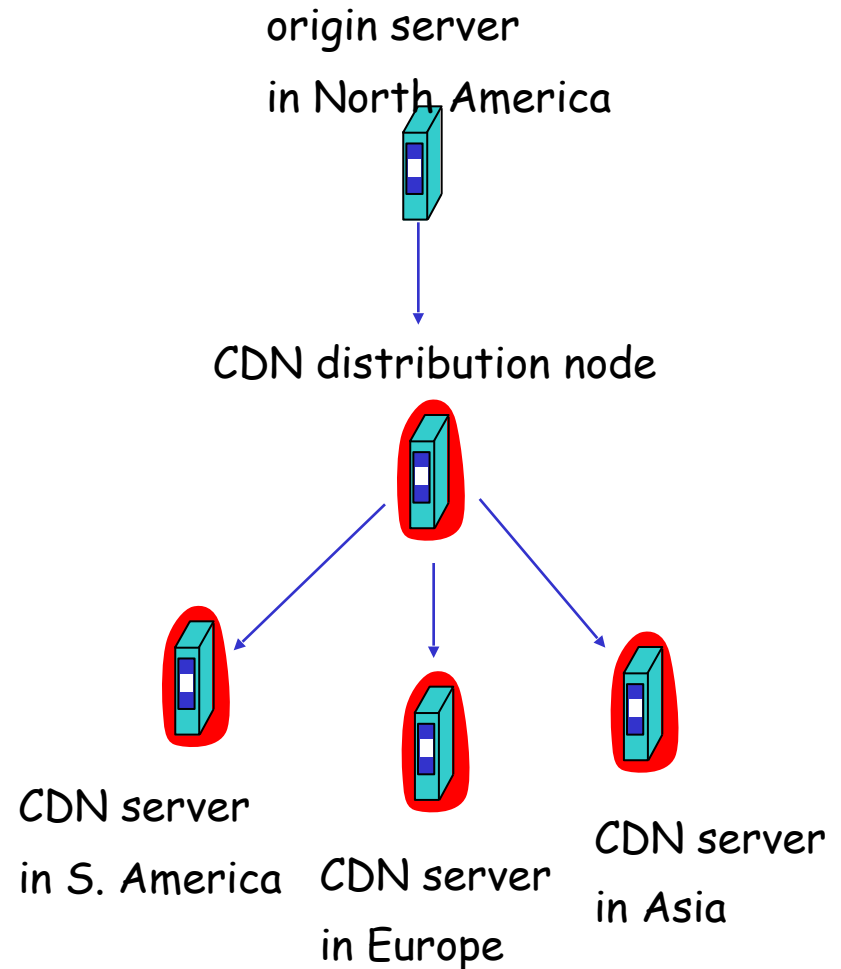
Streaming stored video

- » streaming over UDP
- » streaming over HTTP
 - » Client Application buffer (\neq TCP receive buffer)
 - » HTTP byte-range header (in the HTTP GET request message)
- » Dynamic Adaptive Streaming over HTTP (DASH)
 - » manifest files identify alternative streams and their respective URLs.

Content distribution networks (CDNs)

Content replication

- ❖ Challenging to stream large files (e.g., video) from single origin server in real time
- ❖ Solution: replicate content at hundreds of servers throughout Internet
 - content downloaded to CDN servers ahead of time
 - placing content “close” to user avoids impairments (loss, delay) of sending content over long paths
 - CDN server typically in edge/access network



More about CDNs

routing requests

- ❖ CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes
- ❖ when query arrives at authoritative DNS server:
 - server determines ISP from which query originates
 - uses “map” to determine best CDN server
- ❖ CDN nodes create application-layer overlay network

Static Web Pages

Static Web pages are simply files

- Have the same contents for each viewing

Can be visually rich and interactive nonetheless:

- HTML that mixes text and images
- Forms that gather user input
- Style sheets that tailor presentation
- Vector graphics, videos, and more (over) . . .

Static Web Pages

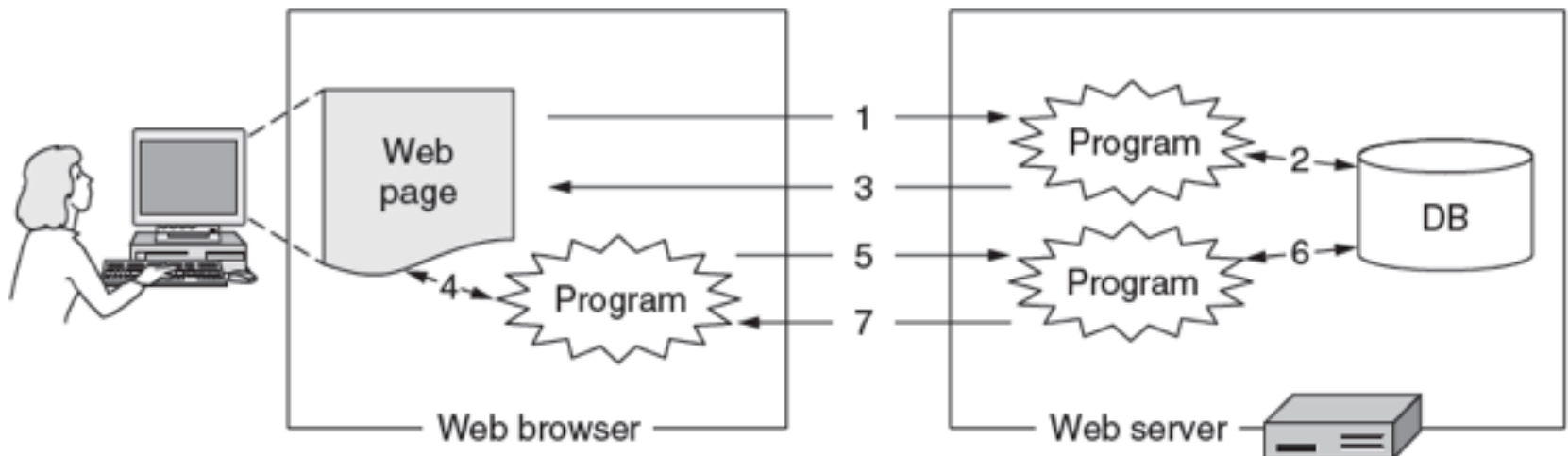
Progression of features through HTML 5.0

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	X	X	X	X	X
Images	X	X	X	X	X
Lists	X	X	X	X	X
Active maps & images		X	X	X	X
Forms		X	X	X	X
Equations			X	X	X
Toolbars			X	X	X
Tables			X	X	X
Accessibility features				X	X
Object embedding				X	X
Style sheets				X	X
Scripting				X	X
Video and audio					X
Inline vector graphics					X
XML representation					X
Background threads					X
Browser storage					X
Drawing canvas					X

Dynamic Pages & Web Applications

Dynamic pages are generated by programs running at the server (with a database) and the client

- E.g., PHP at server, JavaScript at client
- Pages vary each time like using an application




Dynamic Pages & Web Applications

Web page that gets form input and calls a server program

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

PHP server program that creates a custom Web page

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```



PHP calls

The diagram shows two arrows pointing from the text "PHP calls" to the PHP code snippets in the previous block. One arrow points to the line "Hello <?php echo \$name; ?>." and the other points to the line "Prediction: next year you will be <?php echo \$age + 1; ?>".

Resulting Web page (for inputs "Barbara" and "32")

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```


Dynamic Pages...

JavaScript program
produces result page
in the browser

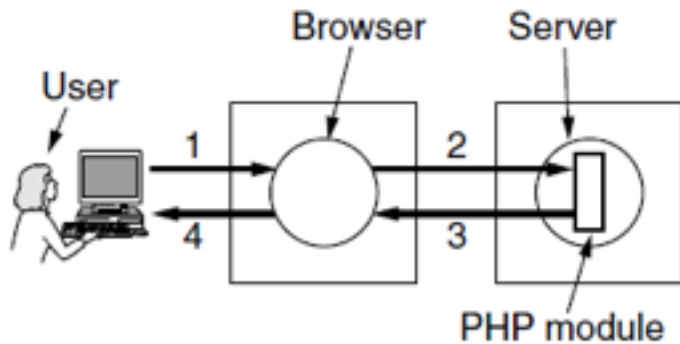
First page with form,
gets input and calls
program above

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
    var person = test_form.name.value;
    var years = eval(test_form.age.value) + 1;
    document.open();
    document.writeln("<html> <body>");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>

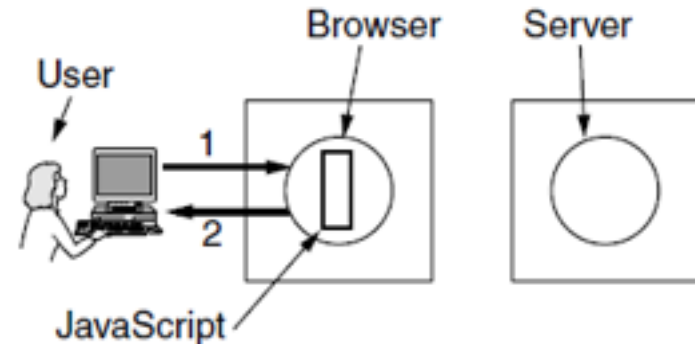
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Dynamic Pages & Web Applications

The difference between server and client programs



Server-side scripting with PHP



Client-side scripting with JavaScript

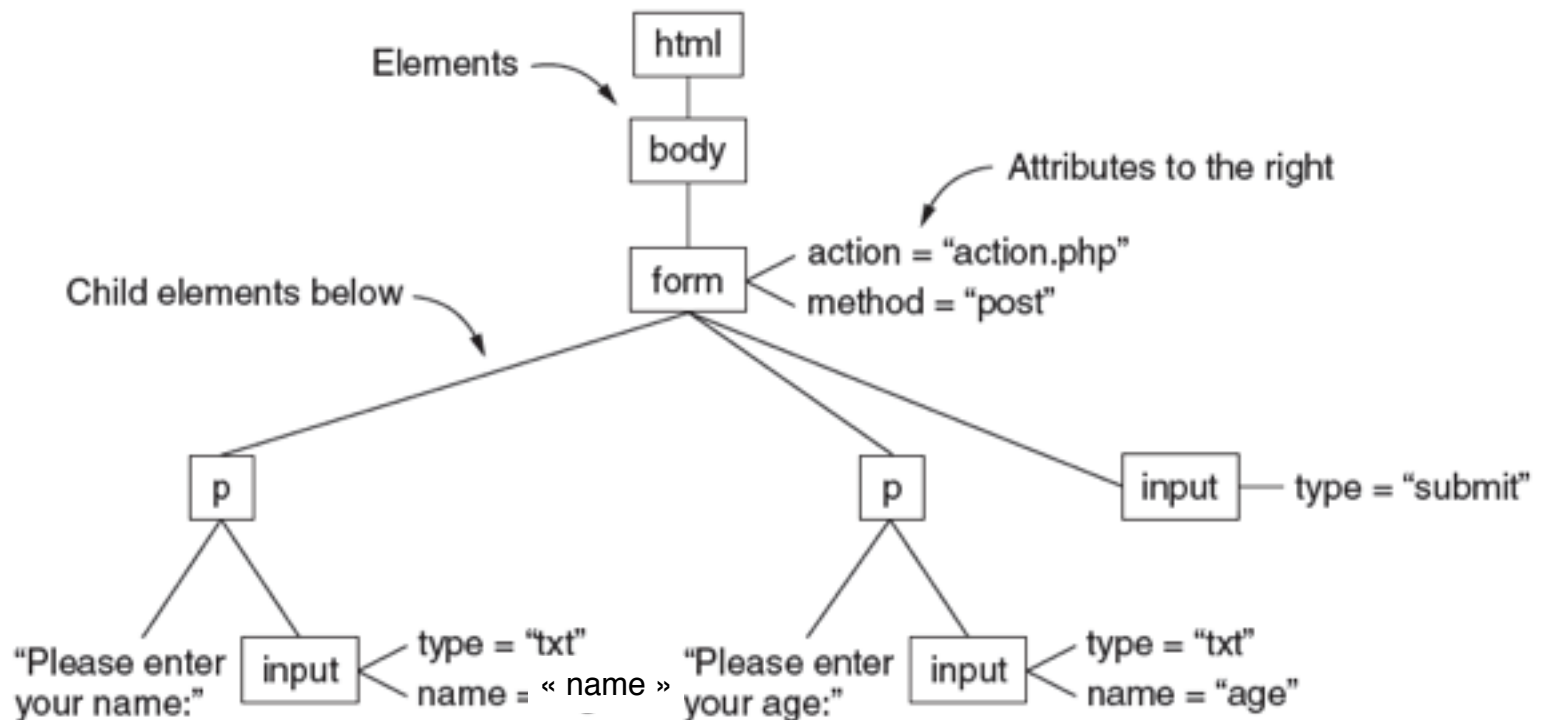
Dynamic Pages & Web Applications

Web applications use a set of technologies that work together, e.g. AJAX:

- HTML: present information as pages.
- DOM: change parts of pages while they are viewed.
- XML: let programs exchange data with the server.
- Asynchronous way to send and retrieve XML data.
- JavaScript as a language to bind all this together.

Dynamic Pages & Web Applications

The DOM (Document Object Model) tree represents Web pages as a structure that programs can alter



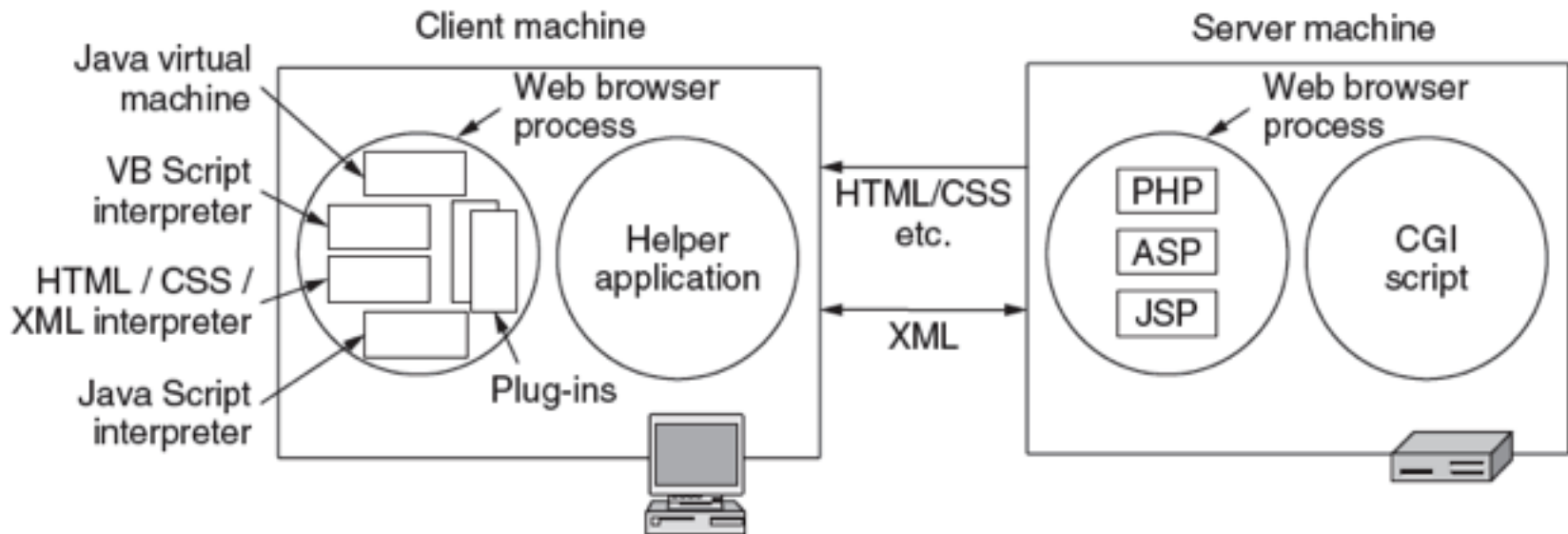
Dynamic Pages & Web Applications

XML captures document structure, not presentation like HTML. Ex:

```
<?xml version="1.0" ?>
<book_list>
  <book>
    <title> Human Behavior and the Principle of Least Effort </title>
    <author> George Zipf </author>
    <year> 1949 </year>
  </book>
  <book>
    <title> The Mathematical Theory of Communication </title>
    <author> Claude E. Shannon </author>
    <author> Warren Weaver </author>
    <year> 1949 </year>
  </book>
  <book>
    <title> Nineteen Eighty-Four </title>
    <author> George Orwell </author>
    <year> 1949 </year>
  </book>
</book_list>
```

Dynamic Pages & Web Applications

Web applications use a set of technologies, revisited:



The Mobile Web

Mobiles (phones, tablets) are challenging as clients:

- Relatively small screens
- Limited input capabilities, lengthy input.
- Network bandwidth is limited
- Connectivity may be intermittent.
- Computing power is limited

Strategies to handle them:

- Contents: servers provide mobile-friendly versions; transcoding can also be used
- Protocols: no real need for specialized protocols; HTTP with header compression sufficient

apache

- ❖ Pour pouvoir tester les exemples on utilisera apache
 - logiciel libre disponible sur la plupart des plateformes
 - Le serveur le plus fréquent
 - Prise en charge de nombreux modulesServeurs virtuels

- ❖ Pour obtenir Apache, MySQL, PHP.... (Mac)
<https://coolestguidesontheplanet.com/get-apache-mysql-php-and-phpmyadmin-working-on-macos-sierra/>
- ❖ Lancer Apache *sudo apachectl start* (*/usr/sbin/apachectl*)
- ❖ Arrêter *sudo apachectl stop* (et redemarrer *sudo apachectl restart*)
- ❖ Version `http -v`

Principes...

- ❖ Le serveur reçoit des requêtes http et renvoie des pages html dans des réponses http
 - Interprète les requêtes
 - Lance sur le côté serveur les applications concernées
 - Récupère les résultats et les transmet au client
- ❖ Configuration: httpd.conf (en général dans /etc/apache2 (et par catalogue .htaccess et htpasswd)
par users /etc/apache2/users/nomuser.conf

/etc/apache2/httpd.conf

```
#  
# ServerRoot: The top of the directory tree under which the server's  
# configuration, error, and log files are kept.  
#  
# Do not add a slash at the end of the directory path. If you point  
# ServerRoot at a non-local disk, be sure to specify a local disk on the  
# Mutex directive, if file-based mutexes are used. If you wish to share the  
# same ServerRoot for multiple httpd daemons, you will need to change at  
# least PidFile.  
#  
ServerRoot "/usr"
```

#

Listen: Allows you to bind Apache to specific IP addresses and/or
ports, instead of the default. See also the <VirtualHost>
directive.

#

Change this to Listen on specific IP addresses as shown below to
prevent Apache from glomming onto all bound IP addresses.

#

#Listen 12.34.56.78:80

Listen 80

....

/etc/apache2/users/remi.conf

```
$ more remi.conf
```

```
<Directory "/Users/remi/Sites/">
```

```
Options Indexes MultiViews
```

```
AllowOverride None
```

```
Order Allow, Deny
```

```
Allow from all
```

```
</Directory>
```

Principes

- ❖ Correspondance entre url et fichiers locaux
 - DocumentRoot:
 - Si DocumentRoot = /Library/WebServer/Documents
 - <http://localhost/un/deux.html> sera converti en /Library/WebServer/Documents/un/deux.html
 - Pages des utilisateurs
 - UserDir: ~/Sites
 - http://localhost/~user/file.html sera /Users/user/Sites/file.html
 - En plus des alias et des redirections

Pour voir...

- ❖ Sur cette machine:
 - `/etc/apache2/httpd.conf`
 - `ServerRoot: /usr`
 - `DocumentRoot: /Library/WebServer/Documents`
 - + fichier de configurations dans « extra »
 - `UserDir: Sites`

CGI

- ❖ Common Gateway Interface
- ❖ exécuter du code du côté serveur
- ❖ Passage de paramètre par la méthode POST ou la méthode GET
- ❖ Variables d'environnement

Pour Apache

❖ Les executables cgi (dépendant)

- ScriptAlias /cgi-bin /Library/WebServer/CGI-Executables/
- Pour <http://localhost/cgi-bin/exemples/treat.pl>
/Library/WebServer/CGI-Executables/exemples/treat.pl sera exécuté

❖ "Paramètres"

- POST: transmis sur l'entrée standard (STDIN)
- GET: variable de l'environnement QUERY_STRING

❖ STDOUT pour la réponse

- (au moins un MIME type header
 - Content-type: text/html et deux newline)

Exemple

❖ en shell: /Library/WebServer/CGI-Executables/
examples/examples/date.cgi

```
#!/bin/sh  
tmp=`/bin/date`  
echo "Content-type: text/html\n  
<HTML><HEAD><TITLE>Script Cgi</TITLE></HEAD><BODY>  
<CENTER>  
<H1>La date courante sur le serveur est</H1> $tmp  
</CENTER> </BODY> </HTML>"
```

L'URL affichera la date

<http://localhost/cgi-bin/examples/date.cgi>

Avec un formulaire:

```
<HTML><HEAD><TITLE>Formulaire simple</TITLE></HEAD>
<BODY>
<H2>Répondez aux questions suivantes</H2>
<FORM ACTION="http://localhost/cgi-bin/exemples/treat.pl" METHOD=GET>
Prénom : <INPUT TYPE="text" NAME=prenom SIZE=20><BR>
Nom : <INPUT TYPE="text" NAME=nom SIZE=20><BR>
Age : <SELECT NAME=age>
    <OPTION>- de 18 ans
    <OPTION>19 à 40 ans
    <OPTION>41 à 60 ans
    <OPTION>+ de 60 ans
</SELECT><BR>
<INPUT TYPE=submit VALUE="Envoyer"> <INPUT TYPE=reset VALUE="Remettre
à zéro">
</FORM>
</BODY>
http://localhost/un/formulaire.html
```

Résultat

- ❖ par la méthode *GET* codage des paramètres:
- ❖ Prenom=Remi&nom=Dupond&age=41+%C3%A0+60+ans
- ❖ le navigateur génère l'url:
[http://localhost/cgi-bin/exemples/treat.pl?
prenom=Remi&nom=Dupond&age=41+
%C3%A0+60+ans%C3%A0+60+ans](http://localhost/cgi-bin/exemples/treat.pl?prenom=Remi&nom=Dupond&age=41+%C3%A0+60+ans%C3%A0+60+ans)

Avec la méthode *POST*

<http://www.localhost.com/cgi-bin/exemples/treat.pl>

Prenom=Remi&nom=Dupond&age=41+%C3%A0+60+ans dans la
partie « entity body »

Traitement en perl

- ❖ /Library/WebServer/CGI-Executables/examples/treat.pl
- ❖ <http://localhost/cgi-bin/examples/treat.pl?prenom=Remi&nom=Dupond&age=41+%C3%A0+60+ans%C3%A0+60+ans>

Paramètres

- ❖ Les paramètres sont accessibles par l'intermédiaire de la variable d'environnement `QUERY_STRING`

```

#!/usr/bin/perl
# les donnees sont envoyees par methode GET
# donc on recupere les donnees dans la variable
# d'environnement QUERY_STRING
$buffer=$ENV{"QUERY_STRING"};
# Si POST:    $buffer= <STDIN>;

# on split la chaine de donnees en des paires name=value
local(@champs) = split(/&/, $buffer);
local($donnees) = " »;

# affichage du debut du code HTML
printf STDOUT "Content-type: text/html\n\n";
printf STDOUT "<HTML><HEAD>";
printf STDOUT "<TITLE>Reponse au questionnaire</TITLE>";
printf STDOUT "</HEAD>";
printf STDOUT "<BODY BGCOLOR=\"#ffffff\"> »;

```

```
printf STDOUT "<H1>R<E9>sultat du traitement de votre questionnaire</H1>";  
printf STDOUT "<H2>Chaine de donn<E9>es re<E7>ue par le programme</H2>";  
printf STDOUT "QUERY_STRING <STRONG>%s</STRONG>", $buffer;  
printf STDOUT "<H2>Liste des informations d<E9>cod<E9>es</H2>";  
printf STDOUT "<UL>";  
printf STDOUT "<BL>";
```

```
# recuperation et mise en forme des donnees  
# on parcourt la liste des paires name=value  
foreach $i (0 .. $#champs) {  
    # On convertit les plus en espaces  
    $champs[$i] =~ s/\+/ /g;
```


On separe chaque champ en une cle et sa valeur

```
($key, $val) = split(/=/,$champs[$i],2);
```

On convertit les %XX de leur valeur hexadecimale en alphanumerique

```
$key =~ s/%(..)/pack("c",hex($1))/ge;
```

```
$val =~ s/%(..)/pack("c",hex($1))/ge;
```

on affiche le resultat

```
printf STDOUT "<LI><STRONG>%s:</STRONG>%s\n",$key,$val;
```

```
}
```

```
printf STDOUT "</BL>";
```

```
printf STDOUT "</UL>";
```

```
printf STDOUT "</BODY>";
```

```
printf STDOUT "</HTML>";
```

Variables d'environnement

- ❖ **SERVER_SOFTWARE**
 - Le nom et la version du serveur HTTP répondant à la requête. (Format : nom/version)
- ❖ **SERVER_NAME**
 - Le nom d'hôte, alias DNS ou adresse IP du serveur.
- ❖ **GATEWAY_INTERFACE**
 - La révision de la spécification CGI que le serveur utilise. (Format : CGI/révision)

Variables...

- ❖ **SERVER_PROTOCOL**
 - Le nom et la révision du protocole dans lequel la requête a été faite (Format : protocole/révision)
- ❖ **SERVER_PORT**
 - Le numéro de port sur lequel la requête a été envoyée.
- ❖ **REQUEST_METHOD**
 - La méthode utilisée pour faire la requête. Pour HTTP, elle contient généralement « GET » ou « POST ».
- ❖ **PATH_INFO**
 - Le chemin supplémentaire du script tel que donné par le client. Par exemple, si le serveur héberge le script « /cgi-bin/monscript.cgi » et que le client demande l'url « http://serveur.org/cgi-bin/monscript.cgi/marecherche », alors PATH_INFO contiendra « marecherche ».
- ❖ **PATH_TRANSLATED**
 - Contient le chemin demandé par le client après que les conversions virtuel → physique aient été faites par le serveur.

Variables

- ❖ **SCRIPT_NAME**
 - Le chemin virtuel vers le script étant exécuté. Exemple : « /cgi-bin/script.cgi »
- ❖ **QUERY_STRING**
 - Contient tout ce qui suit le « ? » dans l'URL envoyée par le client. Toutes les variables provenant d'un formulaire envoyé avec la méthode « GET » sera contenue dans le QUERY_STRING sous la forme « var1=val1&var2=val2&... ».
- ❖ **REMOTE_HOST**
 - Le nom d'hôte du client. Si le serveur ne possède pas cette information (par exemple, lorsque la résolution DNS inverse est désactivée), REMOTE_HOST sera vide.
- ❖ **REMOTE_ADDR**
 - L'adresse IP du client.
- ❖ **AUTH_TYPE**
 - Le type d'identification utilisé pour protéger le script (s'il est protégé et si le serveur supporte l'identification).

Variables

- ❖ **REMOTE_USER**
 - Le nom d'utilisateur du client, si le script est protégé et si le serveur supporte l'identification.
- ❖ **REMOTE_IDENT**
 - Nom d'utilisateur (distant) du client faisant la requête. Le serveur doit supporter l'identification RFC 931. Cette variable devraient être utilisée à des fins de journaux seulement.
- ❖ **CONTENT_TYPE**
 - Le type de contenu attaché à la requête, si des données sont attachées (comme lorsqu'un formulaire est envoyé avec la méthode « POST »).
- ❖ **CONTENT_LENGTH**
 - La longueur du contenu envoyé par le client.

Variables

❖ HTTP_ACCEPT

- Les types de données MIME que le client accepte de recevoir.
- Exemple : text/*, image/jpeg, image/png, image/*, */*

❖ HTTP_ACCEPT_LANGUAGE

- Les langages dans lequel le client accepte de recevoir la réponse.
- Exemple : fr_CA, fr

❖ HTTP_USER_AGENT

- Le navigateur utilisé par le client.
- Exemple : Mozilla/5.0 (compatible; Konqueror/3; Linux)

<http://localhost/un/formulairePOST.html>

```
<HTML><HEAD><TITLE>Formulaire simple</TITLE></HEAD>
<BODY>
<H2>Repondez aux questions suivantes</H2>
<FORM ACTION="http://localhost/cgi-bin/exemples/affiche.pl"
METHOD=POST>
Prenom : <INPUT TYPE="text" NAME=prenom SIZE=20><BR>
Nom : <INPUT TYPE="text" NAME=nom SIZE=20><BR>
Age : <SELECT NAME=age>
    <OPTION>- de 18 ans
    <OPTION>19 à 40 ans
    <OPTION>41 à 60 ans
    <OPTION>+ de 60 ans
</SELECT><BR>
```

<http://localhost/un/formulairePOST.html>

```
<INPUT TYPE=submit VALUE="Envoyer"> <INPUT TYPE=reset  
VALUE="Remettre  
a zero">  
</FORM>  
</BODY>
```


<http://localhost/cgi-bin/examples/affiche.pl>

```
#!/usr/bin/perl
```

```
$buffer=$ENV{"SERVER_SOFTWARE"};  
$buffer1=$ENV{"SERVER_NAME"};  
$buffer2=$ENV{"GATEWAY_INTERFACE"};  
$buffer3=$ENV{"SERVER_PROTOCOL"};  
$buffer4=$ENV{"SERVER_PORT"};  
$buffer5=$ENV{"REQUEST_METHOD"};  
$buffer6=$ENV{"PATH_INFO"};  
$buffer7=$ENV{"PATH_TRANSLATED"};  
$buffer8=$ENV{"SCRIPT_NAME"};  
$buffer9=$ENV{"QUERY_STRING"};  
$buffer10=$ENV{"REMOTE_HOST"};
```

<http://localhost/cgi-bin/examples/affiche.pl>

```
$buffer11=$ENV{"REMOTE_ADDR"};  
$buffer12=$ENV{"AUTH_TYPE"};  
$buffer13=$ENV{"REMOTE_USER"};  
$buffer14=$ENV{"REMOTE_IDENT"};  
$buffer15=$ENV{"CONTENT_TYPE"};  
$buffer16=$ENV{"CONTENT_LENGTH"};  
$buffer17=$ENV{"HTTP_ACCEPT"};  
$buffer18=$ENV{"HTTP_ACCEPT_LANGUAGE"};  
$buffer19=$ENV{"HTTP_USER_AGENT"};
```

<http://localhost/cgi-bin/examples/affiche.pl>

affichage du debut du code HTML

```
printf STDOUT "Content-type: text/html\n\n";
```

```
printf STDOUT "<HTML><HEAD>";
```

```
printf STDOUT "<TITLE>Reponse au questionnaire</TITLE>";
```

```
printf STDOUT "</HEAD>";
```

```
printf STDOUT "<BODY BGCOLOR=\"#ffffff\">";
```

```
printf STDOUT "<H1>Var de l'environnement</H1>";
```

```
printf STDOUT "QUERY_STRING <STRONG>%s</STRONG>", $buffer;
```

```
printf STDOUT "<BR>";
```

```
printf STDOUT "QUERY_NAME <STRONG>%s</STRONG>", $buffer1;
```

```
printf STDOUT "<BR>";
```

```
printf STDOUT "GATEWAY_INTERFACE <STRONG>%s</STRONG>", $buffer2;
```

<http://localhost/cgi-bin/examples/affiche.pl>

```
printf STDOUT "<BR>";
printf STDOUT "SERVER_PROTOCOL <STRONG>%s</STRONG>", $buffer3;
printf STDOUT "<BR>";
printf STDOUT "SERVER_PORT <STRONG>%s</STRONG>", $buffer4;
printf STDOUT "<BR>";
printf STDOUT "REQUEST_METHOD <STRONG>%s</STRONG>", $buffer5;
printf STDOUT "<BR>";
printf STDOUT "PATH_INFO <STRONG>%s</STRONG>", $buffer6;
printf STDOUT "<BR>";
printf STDOUT "PATH_TRANSLATED <STRONG>%s</STRONG>", $buffer7;
printf STDOUT "<BR>";
printf STDOUT "SCRIPT_NAME <STRONG>%s</STRONG>", $buffer8;
printf STDOUT "<BR> »";
```

<http://localhost/cgi-bin/examples/affiche.pl>

```
printf STDOUT "QUERY_STRING <STRONG>%s</STRONG>", $buffer9;  
printf STDOUT "<BR>";  
printf STDOUT "REMOTE_HOST <STRONG>%s</STRONG>", $buffer10;  
printf STDOUT "<BR>";  
printf STDOUT "REMOTE_ADDR <STRONG>%s</STRONG>", $buffer11;  
printf STDOUT "<BR>";  
printf STDOUT "AUTH_TYPE <STRONG>%s</STRONG>", $buffer13;  
printf STDOUT "<BR>";  
printf STDOUT "REMOTE_USER <STRONG>%s</STRONG>", $buffer13;  
printf STDOUT "<BR>";  
printf STDOUT "REMOTE_IDENT <STRONG>%s</STRONG>", $buffer14;  
printf STDOUT "<BR> »";
```

<http://localhost/cgi-bin/examples/affiche.pl>

```
printf STDOUT "CONTENT_TYPE <STRONG>%s</STRONG>", $buffer15;  
printf STDOUT "<BR>";  
printf STDOUT "CONTENT_LENGTH <STRONG>%s</STRONG>", $buffer16;  
printf STDOUT "<BR>";  
printf STDOUT "HTTP_ACCEPT <STRONG>%s</STRONG>", $buffer17;  
printf STDOUT "<BR>";  
printf STDOUT "HTTP_ACCEPT_LANGUAGE <STRONG>%s</STRONG>", $buffer18;  
printf STDOUT "<BR>";  
printf STDOUT "HTTP_USER_AGENT <STRONG>%s</STRONG>", $buffer19;  
printf STDOUT "</BODY>";  
printf STDOUT "</HTML>";
```

On obtient

Var de l'environnement

QUERY_STRING Apache/2.4.9 (Unix)

QUERY_NAME localhost

GATEWAY_INTERFACE CGI/1.1

SERVER_PROTOCOL HTTP/1.1

SERVER_PORT 80

REQUEST_METHOD POST

PATH_INFO

PATH_TRANSLATED

SCRIPT_NAME /cgi-bin/exemples/affiche.pl

QUERY_STRING

REMOTE_HOST

REMOTE_ADDR ::1

On obtient

AUTH_TYPE

REMOTE_USER

REMOTE_IDENT

CONTENT_TYPE application/x-www-form-urlencoded

CONTENT_LENGTH 42

HTTP_ACCEPT text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

HTTP_ACCEPT_LANGUAGE fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP_USER_AGENT Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36

Compléments Javascript

- ❖ Code qui s'exécute du côté du client
 - calcul local
 - contrôle d'une zone de saisie
 - affichage d'alerte
 - fenêtres menus etc..
- ❖ Balise :
 <SCRIPT langage="JavaScript1.2">
 le code...
 </SCRIPT>

Exemple: bonjour

```
<HTML><HEAD>  
<TITLE>Très facile</TITLE>  
</HEAD>
```

```
<SCRIPT language="JavaScript1.2">  
function bonjour()  
{  
    alert ("Bonjour madame, bonjour monsieur");  
}  
</SCRIPT>
```

```
<BODY bgcolor="WHITE" onLoad="bonjour();">  
    <H1>Bonjour</H1>
```

```
</BODY></HTML>
```

<http://localhost/un/BjrJvs.html>

Un peu plus: minicalcul

```
<HTML>
<HEAD>
<TITLE>Petit calcul</TITLE>
</HEAD>
<BODY bgcolor='WHITE'>

<script language='JavaScript1.2' src='calcul.js'></script>
<script language='JavaScript1.2' src='fenetre.js'></script>
<script language='JavaScript1.2' src='ctrl.js'></script>

<CENTER><H1>Calcul</H1></CENTER>
```

Un petit exemple de formulaire.

```
<P>
```

Création d'une

```
<A href='#A' onClick='afficheDoc();'>fenêtre avec JavaScript</A>
```

Suite

```
<FORM ACTION='Simul.html' METHOD='POST' NAME='Simul'>
<CENTER>
<TABLE BORDER=3>
<TR><TD>Argument 1
  <TD> <INPUT TYPE='TEXT' SIZE=20 NAME='arg1' onChange='calcul();'></TR>
<TR><TD>* Argument 2
  <TD> <INPUT TYPE='TEXT' SIZE=20
    NAME='arg2' onChange='calcul();'>
</TR>

<TR><TD>Résultat=
  <TD> <INPUT TYPE='TEXT' SIZE=20
    NAME='res' >
</TR>
</TABLE>
<INPUT TYPE='BUTTON' VALUE='Vérifier' onClick='ctrl();'>
<INPUT TYPE='RESET' VALUE='Effacer tout'
  onClick=' if (!confirm("Vraiment vous voulez effacer ?")) exit;'>
</CENTER>
</FORM>
</BODY>
</HTML>
```

<http://localhost/un/Simul.html>

Carole Delporte

Fichiers js

❖ un/ctrl.js

❖ function ctrl()

```
{  
  if (isNaN(window.document.Simul.res.value ))  
  {  
    alert ("Valeur incorrecte : " +  
          document.Simul.res.value + "?");  
    document.forms[0].res.focus();  
  }  
}
```

❖ un/calcul.js

```
function calcul()  
{  
  v1=document.forms[0].arg1.value;  
  v2=document.forms[0].arg2.value;  
  document.forms[0].res.value = v2*v1 ;  
}
```

suite et fin

❖ un/fenetre.js

```
function afficheDoc()
{
  options = "width=300,height=200";
  fenetre = window.open(", 'MU', options);

  fenetre.document.open();
  manuel = "<HTML><HEAD><TITLE>Documentation</TITLE></HEAD>"
    + "<BODY bgcolor='white'>"
    + "Il n'y a pas besoin d'aide "
    + " c'est facile."
    + " Bonne chance !</BODY></HTML>";
  fenetre.document.write(manuel);
  fenetre.document.close();
}
```

Compléments: php

- ❖ php est un langage de script pour les serveurs webs
- ❖ de nombreuses fonctions permettent de traiter les requêtes http (en particulier des requêtes concernant des bases de données)
- ❖ ici on est du côté du serveur...

Exemple simple

```
<HTML> <HEAD>
<TITLE>Exemple très simple</TITLE>
</HEAD>
<BODY>
<H1>Exemple</H1>
le <?php date_default_timezone_set('Europe/Paris');
echo Date ("j/m/Y à H:i:s"); ?>
<P>

<?php
    echo "Client : " . $_SERVER['HTTP_USER_AGENT'] . "<BR>";
    echo "Adresse IP client:". $_SERVER['REMOTE_ADDR'] . "<BR>";
    echo "Server: " . $_SERVER['SERVER_NAME'];
?>

</BODY></HTML>
http://localhost/un/ExempleSimple.php
```


Résultat

Exemple

Le 10/10/2016 à 10:24:25

Client :Mozilla/5.0 (Macintosh; Intel Mac OS X
10_10_5) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/45.0.2454.101 Safari/537.36
Adresse IP client:::1
Server: localhost

Reçu par le client

```
<HTML> <HEAD>  
<TITLE>Exemple très simple</TITLE>  
</HEAD>  
<BODY>
```

```
<H1>Exemple</H1>
```

Le 10/10/2016 à 10:24:25

```
<P>
```

Client :Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36

Adresse IP client:::1

Server: localhost

```
</BODY></HTML>
```

Php

- ❖ On est ici côté serveur:
 - les balises `<?php>` `<?>` sont interprétées par le serveur (apache par exemple) et servent à générer la page html reçue par le client
- ❖ Mais surtout php permet
 - d'accéder aux variables d'environnement
 - d'utiliser de nombreuses fonctionnalités
 - sessions, paramètres etc.
- ❖ Php sert souvent d'interface pour MySql serveur simple de bases de données

Pour le serveur...

- ❖ tableaux associatifs prédéfinis
 - `$_SERVER`: environnement serveur
 - `REQUEST_METHOD`
 - `QUERY_STRING`
 - `CONTENT_LENGTH`
 - `SERVER_NAME`
 - `PATH_INFO`
 - `HTTP_USER_AGENT`
 - `REMOTE_ADDR`
 - `REMOTE_HOST`
 - `REMOTE_USER`
 - `REMOTE_PASSWORD`

Suite

❖ Autres tableaux

- `$_ENV` : environnement système
- `$_COOKIE`
- `$_GET`
- `$_POST`
- `$_FILES`
- `$_REQUEST` (variables des 4 précédents)
- `$_SESSION`
- `$GLOBALS` les variables globales du script

Cookies et php

<http://localhost/un/SetCookie.php>

```
<?php
// Est-ce que le Cookie existe ?
if (isset($_COOKIE['compteur']))
{
    $message = "Vous êtes déjà venu {$_COOKIE['compteur']} fois "
        . "me rendre visite<BR>\n";
    // On incrémente le compteur
    $valeur = $_COOKIE['compteur'] + 1;
}
else
{
    // Il faut créer le cookie avec la valeur 1
    $message = "Bonjour, je vous envoie un cookie<BR>\n";
    $valeur = 1;
}
// Envoi du cookie
SetCookie ("compteur", $valeur);
?>
```

Cookies et php (fin)

```
<HTML><HEAD>  
  <TITLE>Les cookies</TITLE>
```

```
</HEAD>  
<BODY>
```

```
<H1>Un compteur d'accès au site avec cookie</H1>
```

```
<?php echo $message; ?>
```

```
</BODY></HTML>
```

<http://localhost/un/SetCookie.php>

En utilisant les sessions

```
<?php
// La fonction session_start fait tout le travail
session_start();
?>
<HTML><HEAD>
  <TITLE>Les cookies</TITLE>
</HEAD>
<BODY>

<H1>Un compteur d'accès au site avec Session</H1>
```


Fin

```
<?php
if (!isset($_SESSION['cp']))
{
    $_SESSION['cp']=1;
    echo "C'est la première fois,votre id est:" .
    session_id()."<BR>";
}
else{
    $_SESSION['cp']++;
    echo "C'est votre ".$_SESSION['cp']." n-ième connexion";
    if($_SESSION['cp']>10){
        echo "on vous a trop vu."<BR>";
        session_destroy();
    }
}
?>
</BODY></HTML>
```

<http://localhost/un/SessionPHP.php>

session

- ❖ `session_start()`
- ❖ `session_destroy()`
- ❖ `session_id()`
 - on peut associer des variables à la session par le tableau associatif `$_SESSION`
 - elle sera accessible à chaque `session_start()` jusqu'au `session_destroy()` pour toute connexion qui fournit le `session_id()`.