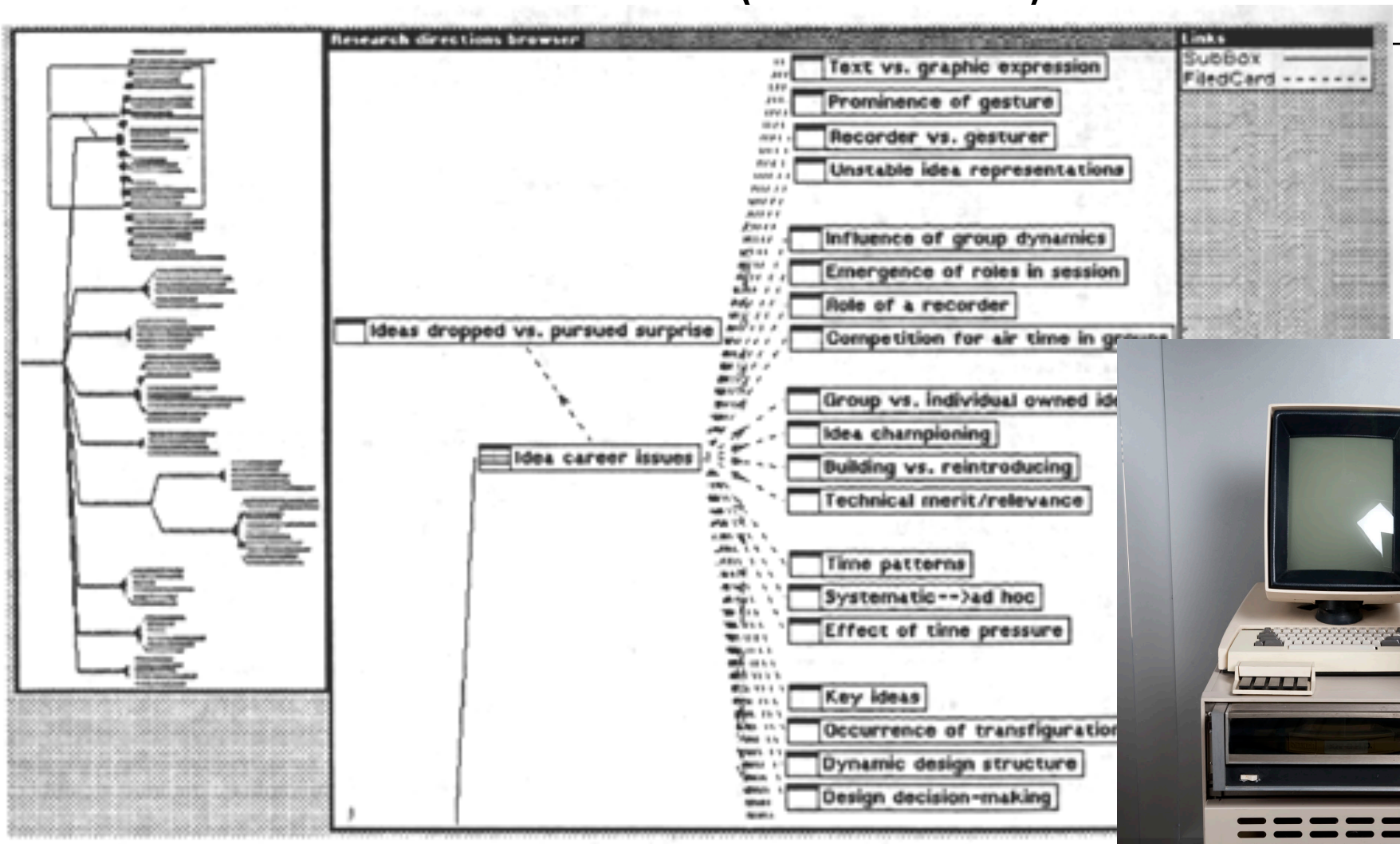
Chapitre 1 : Le WEB et l'origine des langages de script

Le Web …

- Qu'est-ce qui existait en 1989 ?

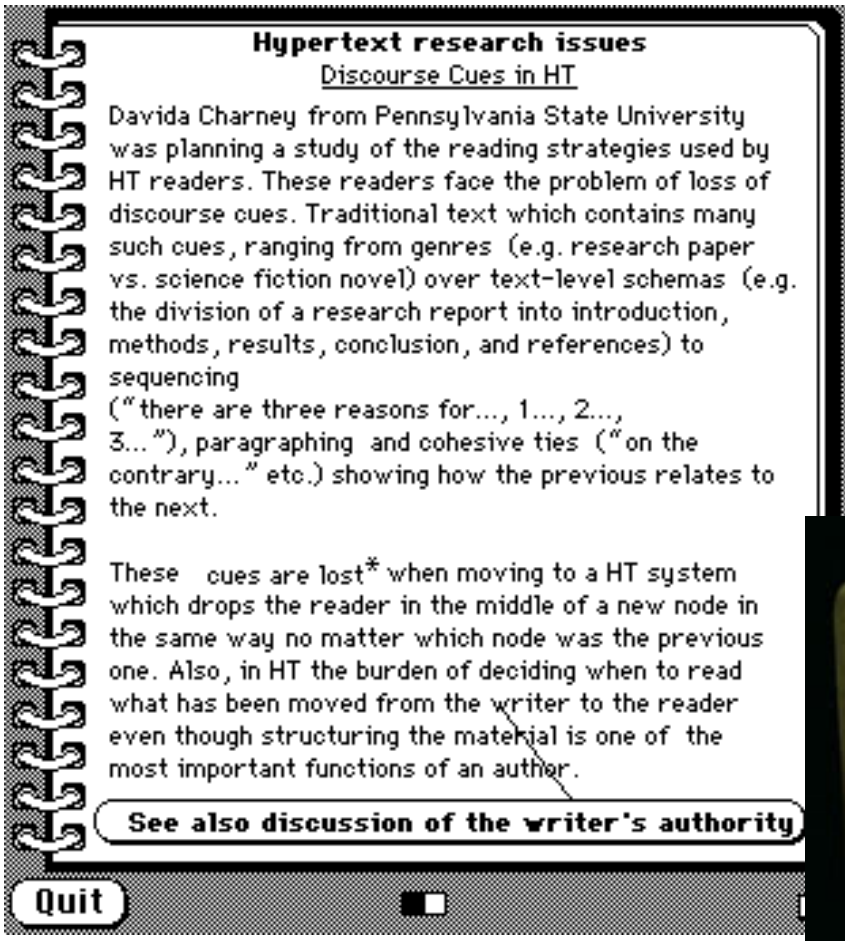
- IP : Internet Protocol
- FTP : Protocole de transfert de fichier
- Hypertexte : des logiciels et idées, NoteCards (Xerox 1984) hypercard (Apple 1987), …
- Plein de format de textes : PS, DVI, PDF, ..
- Langage de balises : SGML (1969, Charles Goldfarb, ISO en 1986), TEI (1987)

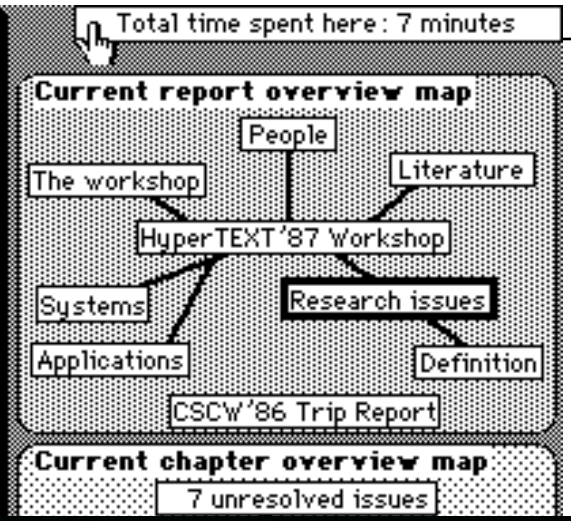
NoteCards (Xerox 1984)



Xerox Alto (1973) : métaphore du bureau : icones, fenêtres, souris …

Hypercard (Apple 1987)





Apple Lisa (1980)

L’idée de Tim Berners-Lee (1989)

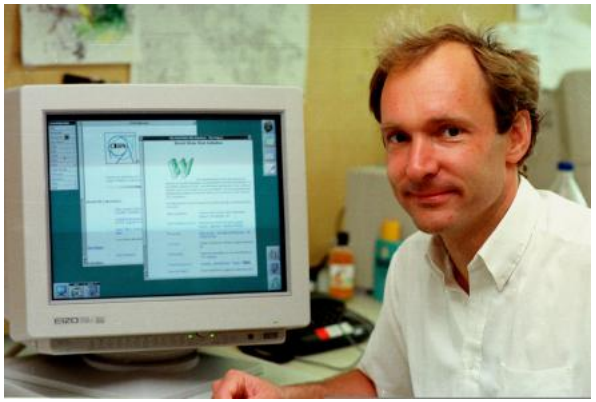
« Je n'ai fait que prendre le principe d’hypertexte et le relier au principe du TCP et du DNS et alors, boum ! ce fut le World Wide Web ! »

1) URL : référence mondiale unique

2) HTTP : protocole très simple

Sans état, textuel

3) HTML : inspiré de SGML, TEI, Hypertextes …



Premier Navigateur populaire (1993)

Modèle Client/Serveur

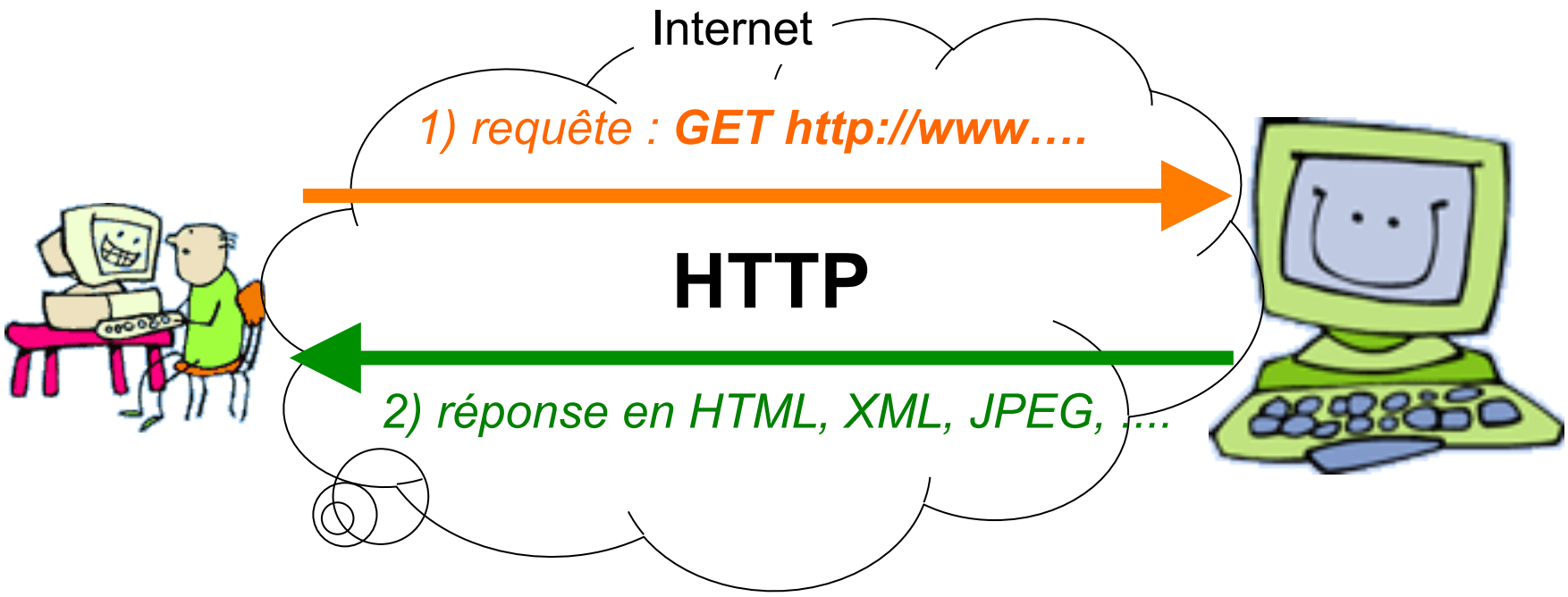
Principe de base : **Client/Serveur**

Machine Client
(ex : pc-dg-32)

Machine Serveur
(ex : www-etu-info.iut2.upmf.grenoble.fr)

un « **navigateur** »
(ex : Mozilla Firefox,Opéra, Safari, IE,…)

un « **serveur** web »
(ex : Apache, IIS)



Le protocole HTTP

Protocole HTTP (HyperText Transfert Protocol):

- utilisé pour transférer les documents hypertextes d'un serveur Web vers un client Web. Mais n'importe quel fichier peut être transmis.
- ces documents sont appelés aussi « ressources »

Chaque **ressource** est identifiée par une « URL »

URL: **U**niform **R**esource **L**ocator

URL

URL: **U**niform **R**esource **L**ocator

adresse identifiant toute ressource (document) du web.

Syntaxe d'une URL:

http://<nomserveur ou adresse IP>:<port>/<chemin ressource>

Exemples :

http://www.google.fr/

http://www.iut2.upmf-grenoble.fr/index.html

http://monsite:8080/ap5/page.html#chap1

http://216.58.206.67:80/search?source=hp&q=iut&oq=iut

http://www-etu-info.iut2.upmf-grenoble.fr/~pesty/tp1/tintin.html

D'autres protocoles possibles : https, ftp, ... files

Le protocole HTTP

- Format **texte**, très simple, codage **ASCII**
  - **GET** : demande une ressource par son URL (pas de modification)
  - **PUT**: inverse de get, place une ressource sur un serveur
  - **POST** : envoie de données binaires (encodées en texte) sur un serveur pour traitement (utilisée incorrectement à la place de PUT)
  - **DELETE** : détruit une ressource
  - **PATCH** : modification partielle d'une ressource
  - **HEAD**, **OPTION**, **CONNECT**, **TRACE** .....

PUT, DELETE, PATCH : nécessitent une accès avec privilère (mot de passe)

- Il a inspiré l'architecture **REST** : "representational state transfer"
  - style d'architecture pour les systèmes hypermédia distribués
  - créé par Roy Fielding dans sa thèse en 2000
    - client/serveur, sans état, identification ressource

HTTP 1.0 : la requête

Format requête :

```
Ligne de commande (Commande, URL, Version de protocole)
En-tête de requête
[Ligne vide]
Corps de requête
```

Exemple :

```
GET /page.html HTTP/1.0
Host: example.com
Referer: http://example.com/
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

HTTP 1.0 : la réponse

Format Réponse :

```
Ligne de statut (Version, Code-réponse, Texte-réponse)
En-tête de réponse
[Ligne vide]
Corps de réponse
```

Exemple :

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 2016 23:59:59 GMT
Server: Apache/0.8.4
Content-Type: text/html
Content-Length: 59
Expires: Sat, 01 Jan 2017 00:59:59 GMT
Last-modified: Fri, 09 Aug 2016 14:21:40 GMT

<TITLE>Exemple</TITLE>
<P>Ceci est une page d'exemple.</P>
```

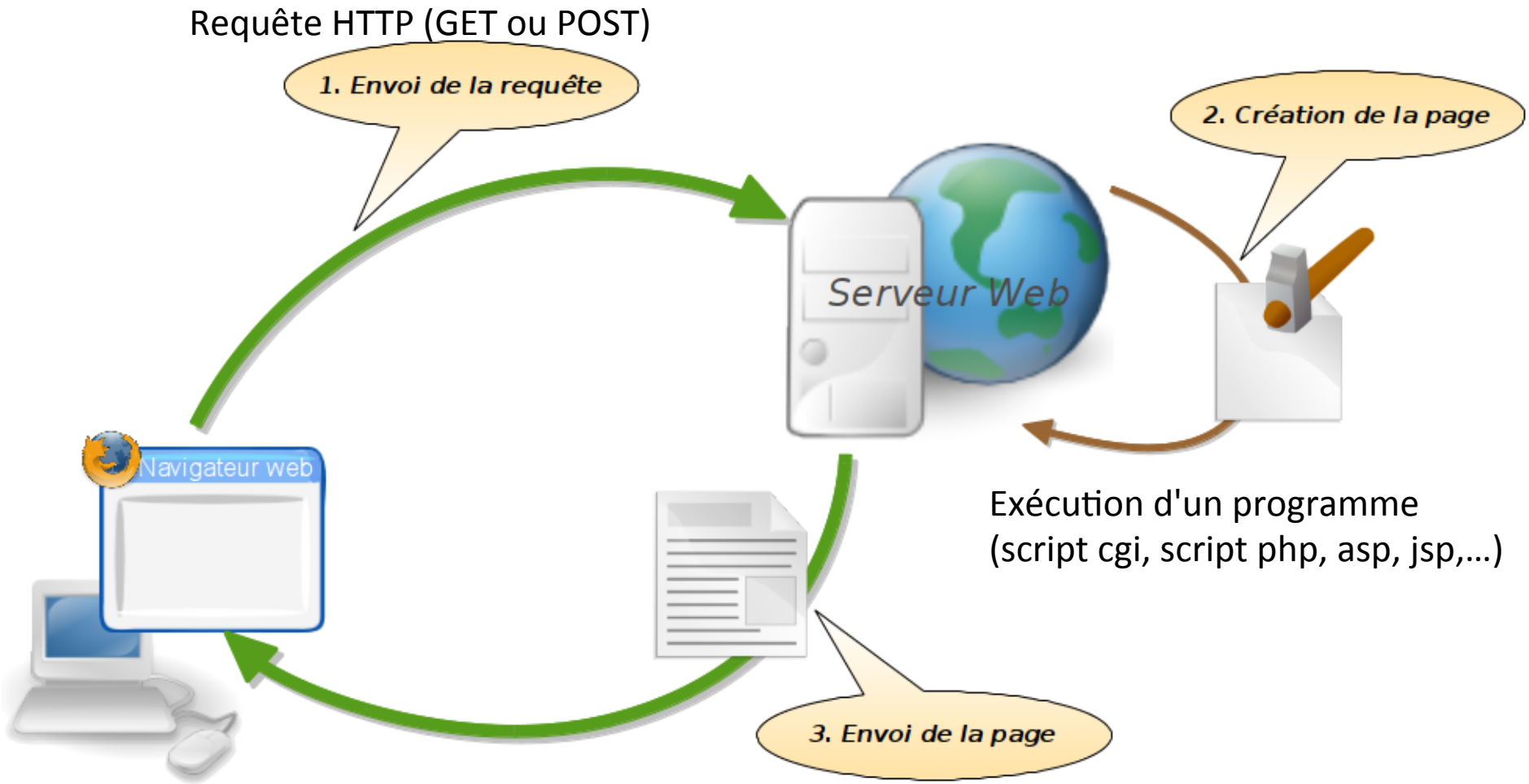
Pages HTML statiques vs dynamiques

- Contenu fixé à l'avance => manipulation de nombreuses pages pour la maintenance
- Aucun moyen d'action pour l'utilisateur pour sélectionner et/ou transmettre des informations au serveur => pas de « personnalisation » des pages selon l'utilisateur

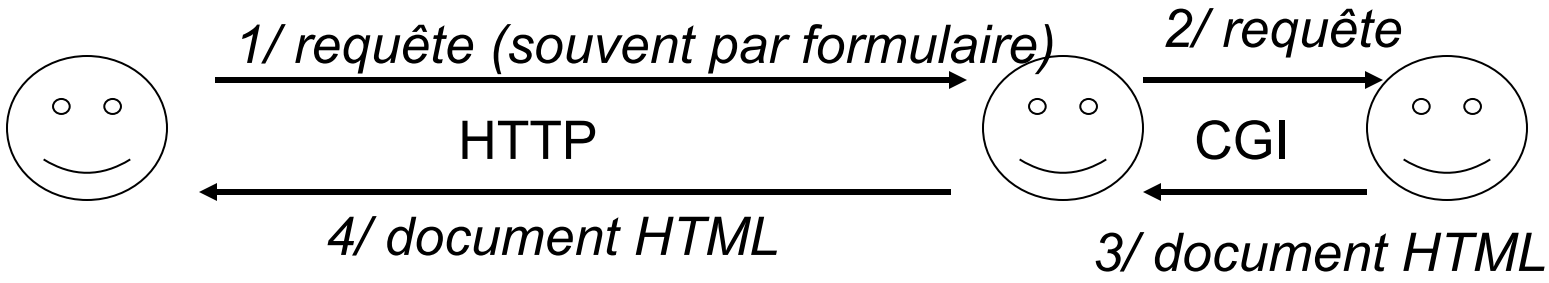
- Technique de base pour pallier ces inconvénients:
  - CGI (Common Gateway Interface)

- Vers des sites web dynamiques : créer des pages dynamiquement en fonction du contenu d'une base de données

Environnement d'exécution d'une application WEB



Principe CGI



Programme client
(ex: mozilla)

Programme
serveur
(ex: Apache)

« Programmes CGI »

en **C**
en **Perl**
script **shell**
...

CGI : norme pour l'interfaçage d'applications externes avec des serveurs d'information (serveur Web en particulier)

=> définit le mécanisme pour faire ***exécuter des programmes par un serveur Web*** à la demande du client

HTTP : Protocole **sans état**

• Echange HTTP : **connexion** au serveur http, **transfert** de la page web, puis **fermeture** de la connexion.

• A la connexion suivante, pas moyen de savoir ce qu'il s'est passé lors des précédents échanges avec le même client.

L'historique de la communication est perdu !

• Problème lors du développement d'une application quand le résultat d'une action dépend de l'historique :

exemple des boutons « mode » et « suivant » du blog

exemple du scénario de type « commerce électronique »

Gestion de l'état : quelques solutions

1. Chaque page est un état de l'interface (ex : mode)

2. Ajouter des paramètres aux « get » (ex : next_page)

3. Stocker l'état côté client (variables javascript, cookies)

4. Stocker l'état côté serveur (variable, fichier/bd) + identifiant (côté client)

   correspond au principe des **sessions** dans les applications Web :

   - Un identifiant est associé à un client (stocké sous forme de cookie ou échangé dans l'url)
   - L'état est stocké côté serveur et les informations concernant l'échange avec le client sont stockés côté serveur

Langage de script

- A l'origine : langage pour manipuler les fonctionnalités d'un **système** Informatique
- Langage de composition de commandes
- Automatiser des actions comme un *script de théâtre*
- Texte d'un programme interprété
- Raccourcir le schéma
  - édition ➔ compilation ➔ édition de lien ➔ exécution
  - édition ➔ exécution

Langage de scripts actuellement

- Langage de programmation dynamique
  - interprété ou semi compilé
  - typage dynamique
  - ajout de code en fonctionnement
  - toutes les fonctionnalités des autres langages
  - inspiré des principaux langages : C, C++, java
- Exemple :
  - shells : sh, bash, csh, tcsh .. (Unix)
  - AppleScript, VBScript
  - Mathlab
  - Javascript, PHP
  - Perl, Python; Ruby, Tcl/Tk ...

Langages de scripts pour le WEB

- Langages "côté serveur" :
  - Compilés : C, C++, ADA,…
  - Interprétés : Perl, JSP, ASP, PHP…
- pour :
  - exécuter une application
  - collecter des données de formulaire
  - consulter, mettre à jour,… des données d'une BD

- PHP (Hypertext Preprocessor) :
  - langage de scripts généraliste
  - Open Source
  - spécialement conçu pour le développement d'applications web
  - peut être intégré facilement au HTML