SNT · LE WEB · SÉANCE 3

Qui fait quoi quand on visite une page Web ?

Une enquête à partir d'un simple F5

Client · serveur · HTTP · JavaScript · PHP · DOM

Pourquoi ma modification ne reste-t-elle pas ?



À faire formuler par les élèves : Où se trouve la version d'origine ? Qui la fournit ?

Une page pour plusieurs visiteurs

Si toute la page était seulement chez moi...

- Comment mon voisin pourrait-il consulter la même page ?
- Comment le site pourrait-il être disponible demain ?
- Qui conserverait la version de référence ?

Si toute la page était seulement chez moi…

- Comment mon voisin pourrait-il consulter la même page ?
- Comment le site pourrait-il être disponible demain ?
- Qui conserverait la version de référence ?

Idée construite ensemble

Une machine accessible par le réseau fournit la ressource demandée : c'est le **serveur**.
Mon navigateur joue le rôle de **client**.

Comment demander la page ?

CLIENT
Navigateur

SERVEUR
Site Web

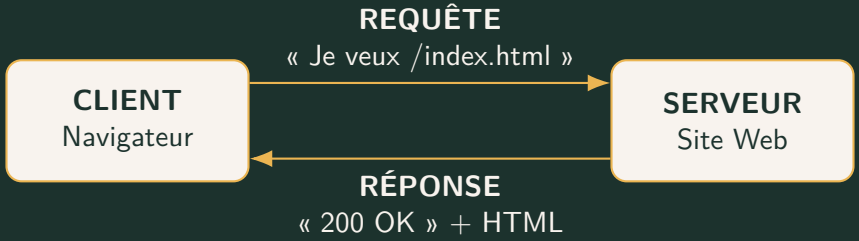
Émettre une hypothèse : qui parle en premier ?

Comment demander la page ?



Le client prend l'initiative de demander.

Comment demander la page ?



Le serveur répond à la demande.

Pourquoi faut-il des règles communes ?

Comment le serveur comprend-il la demande ?

Demande floue

« Bonjour, je voudrais la page, s'il vous plaît... »

Pourquoi faut-il des règles communes ?

Comment le serveur comprend-il la demande ?

Demande floue

« Bonjour, je voudrais la page, s'il vous plaît... »

Demande structurée

```
GET /index.html HTTP/1.1
```

Pourquoi faut-il des règles communes ?

Comment le serveur comprend-il la demande ?

Demande floue

« Bonjour, je voudrais la page, s'il vous plaît... »

Demande structurée

```
GET /index.html HTTP/1.1
```

HTTP = un protocole

Un ensemble de règles pour formuler les **requêtes** et les **réponses**.

À faire formuler par les élèves : À quoi sert le mot GET ? Et /index.html ?

La requête HTTP, pièce par pièce

Que signifie chaque morceau ?

`GET` `/search?q=snt` `HTTP/1.1`

Que signifie chaque morceau ?

`GET` `/search?q=snt` `HTTP/1.1`

Méthode
(action)

Que signifie chaque morceau ?

`GET` `/search?q=snt` `HTTP/1.1`

Méthode (action)

Ressource + paramètres

Que signifie chaque morceau ?

`GET` `/search?q=snt` `HTTP/1.1`

Méthode (action)	Ressource + paramètres	Version du protocole

Que signifie chaque morceau ?

`GET` `/search?q=snt` `HTTP/1.1`

Méthode (action)	Ressource + paramètres	Version du protocole

Autre exemple

`POST` sert souvent à envoyer les données d'un formulaire dans le corps de la requête.
POST ne chiffre pas les données.

Des renseignements accompagnent la demande

Que peut apprendre le serveur ?

Que peut apprendre le serveur ?

En-têtes HTTP Nom du site, logiciel utilisé, langue préférée, cookie éventuellement présent.

À faire formuler par les élèves : Quelles informations avons-nous données volontairement ?

La réponse du serveur

Comment savoir si la demande a réussi ?

200
Page reçue

404
Introuvable

403
Accès interdit

La réponse du serveur

Comment savoir si la demande a réussi ?

200
Page reçue

404
Introuvable

403
Accès interdit

Une réponse HTTP contient

Un **code d'état**, des **en-têtes** (ex. `Content-Type: text/html`) et, souvent, un **corps** contenant la ressource.

Le paradoxe de la connexion

Comment un site se souvient-il de moi ?

- Je me connecte au site.
- Je clique ensuite sur « Mon compte ».
- Le serveur reçoit une **nouvelle** requête HTTP.

Comment un site se souvient-il de moi ?

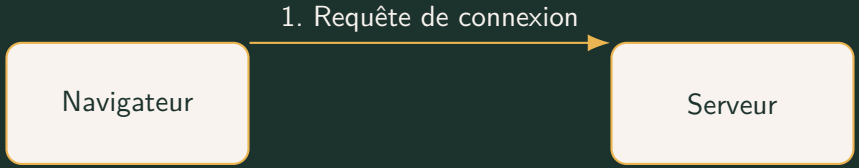
- Je me connecte au site.
- Je clique ensuite sur « Mon compte ».
- Le serveur reçoit une **nouvelle** requête HTTP.

Une solution : le cookie

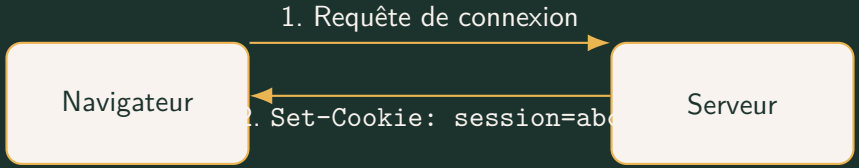
Le serveur peut demander au navigateur de conserver une petite **donnée**. Pour des requêtes ultérieures compatibles avec ses règles, le navigateur la renvoie.

À faire formuler par les élèves : Pourquoi cela aide-t-il le serveur à reconnaître une session ?

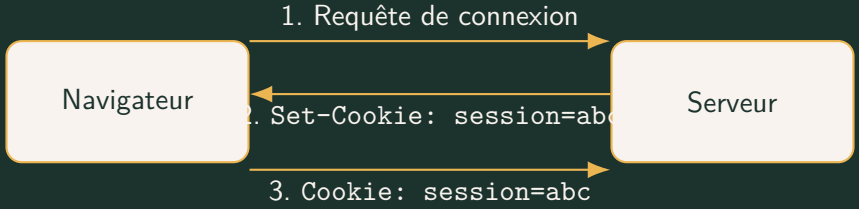
Deux requêtes, une session



Deux requêtes, une session



Deux requêtes, une session



Le cookie peut servir d'**identifiant de session**.

Passage à l'activité 3.1

Peux-tu lire une vraie structure HTTP ?

1. Repère la méthode et le chemin.
2. Décode les paramètres `q` et `lang`.
3. Repère trois informations envoyées dans les en-têtes.
4. Pour `Set-Cookie`, nous construirons la réponse ensemble.

Livret 3 · Activité 3.1

Une page, combien de requêtes ?

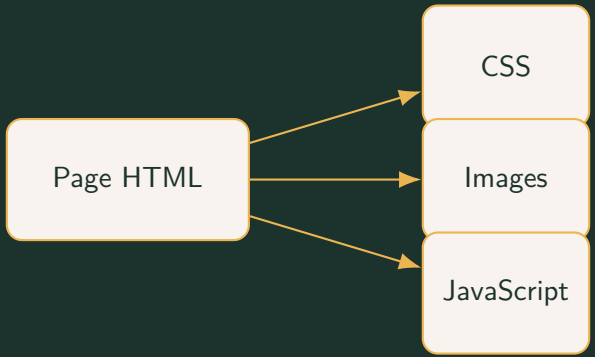
Une seule page Web = une seule requête ?

Page HTML

Question aux élèves : Combien de lignes verrons-nous dans Firefox ?

Une page, combien de requêtes ?

Une seule page Web = une seule requête ?



Question aux élèves : Combien de lignes verrons-nous dans Firefox ?

Où observer les requêtes ?

1. Ouvrir le site dans **Firefox**.
2. Appuyer sur **F12**, puis choisir **Réseau**.
3. Recharger avec **F5**.
4. Lire les colonnes **État**, **Méthode**, **Domaine**, **Type**.

Livret 3 · Activité 3.2

Qui affiche « champ obligatoire » ?

Formulaire de connexion

E-mail : _____

Mot de passe : _____

Veuillez remplir les champs

Qui affiche « champ obligatoire » ?

Formulaire de connexion

E-mail : _____

Mot de passe : _____

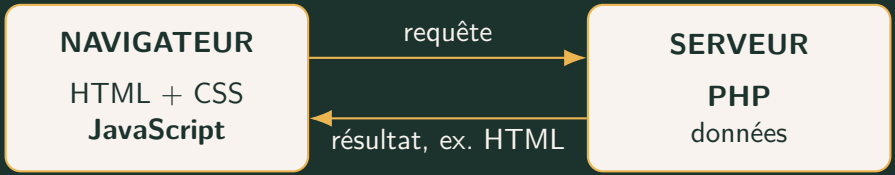
Veuillez remplir les champs

Hypothèse à vérifier

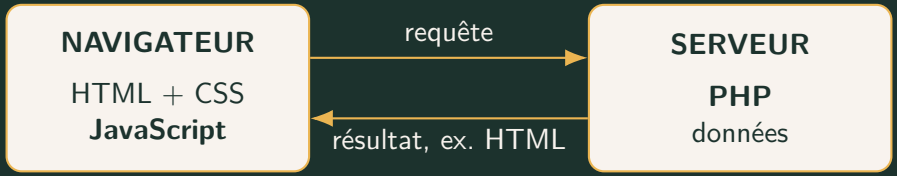
Un programme exécuté **dans le navigateur** peut réagir immédiatement à une action. Ici, le **JavaScript** contrôle le formulaire.

JavaScript et PHP : deux lieux d'exécution

Qui exécute le programme ?



Qui exécute le programme ?



Point essentiel

Ctrl+U peut montrer le JavaScript fourni au navigateur. Sur un serveur PHP normalement configuré, le visiteur reçoit le **résultat**, pas le source PHP.

Pourquoi vérifier côté serveur ?

Et si le navigateur mentait ?

- Le JavaScript peut avertir : « mot de passe manquant ».

Pourquoi vérifier côté serveur ?

Et si le navigateur mentait ?

- Le JavaScript peut avertir : « mot de passe manquant ».
- Mais l'utilisateur contrôle son navigateur et peut modifier son comportement.

Pourquoi vérifier côté serveur ?

Et si le navigateur mentait ?

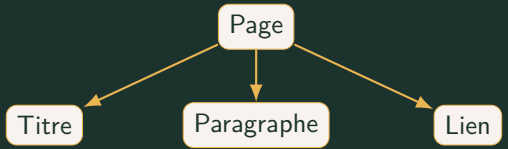
- Le JavaScript peut avertir : « mot de passe manquant ».
- Mais l'utilisateur contrôle son navigateur et peut modifier son comportement.
- Le serveur doit donc **vérifier lui-même** les données sensibles.

Exemple

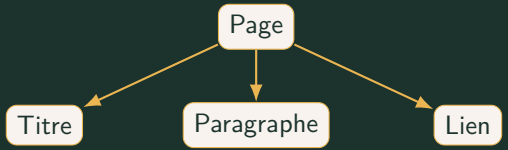
Le navigateur améliore l'expérience de saisie. Le serveur décide si l'authentification est autorisée.

Le DOM : la page représentée en mémoire

Qu'a modifié l'outil « Inspecter » ?



Qu'a modifié l'outil « Inspecter » ?



DOM

Le navigateur construit une représentation des éléments de la page. « Inspecter » peut modifier cette représentation **locale**, sans changer le fichier d'origine sur le serveur.

À faire formuler par les élèves : Pourquoi mon voisin ne voit-il pas le nouveau titre ?

Passage aux activités 3.3 et 3.4

Peux-tu retrouver les deux côtés ?

1. **3.3** : tester `connexion.html` et lire `<script>`.
2. Repérer `<form>`, `action` et `method`.
3. Ouvrir la **copie locale** de `connexion.php`.
4. **3.4** : classer chaque opération côté client ou serveur, avec une justification.

Vérification rapide

Quelle affirmation est correcte ?

A. POST chiffre automatiquement les mots de passe.

B. Modifier le DOM change le site pour tous les visiteurs.

C. Le navigateur peut envoyer une requête GET à un serveur.

Vérification rapide

Quelle affirmation est correcte ?

A. POST chiffre automatiquement les mots de passe.

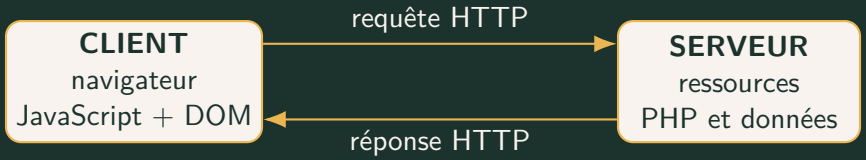
B. Modifier le DOM change le site pour tous les visiteurs.

C. Le navigateur peut envoyer une requête GET à un serveur.

Réponse : C

POST ne chiffre pas. Une modification locale du DOM ne modifie pas la page des autres utilisateurs. Le navigateur peut envoyer une requête `GET`.

Le modèle à retenir



Le navigateur demande. Le serveur répond.

Le code peut s'exécuter **de part et d'autre**.