

---

# Chapitre 8

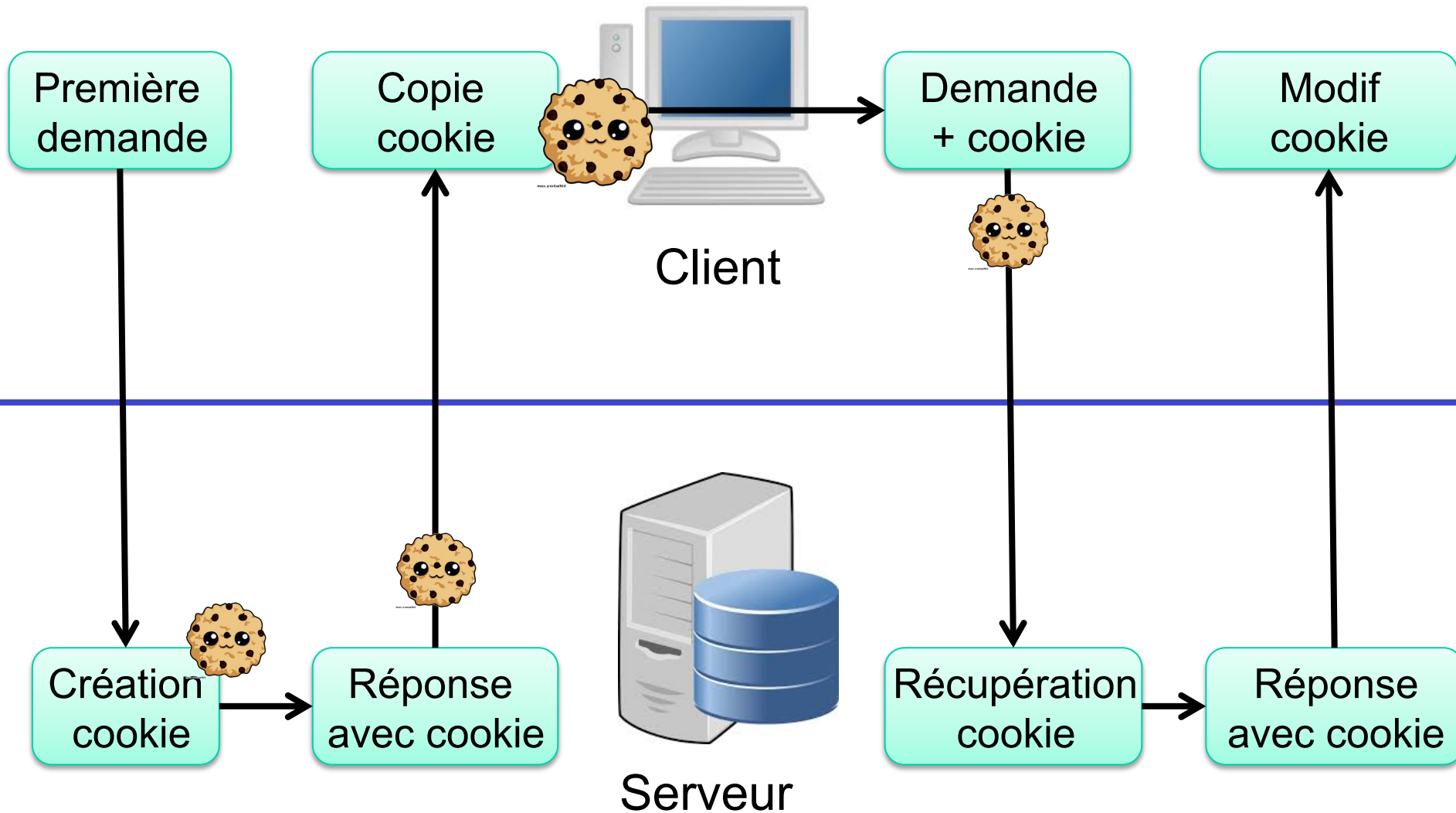
## Cookies et Sessions

# Cookies

---

- Retenir des informations sur un utilisateur.
- Solution pour garder un état : HTTP est un protocole sans état
- Intégré au protocole HTTP : introduit par Netscape..
- Permet d'identifier une navigation de manière unique.
  
- C'est le serveur qui envoie la valeur initiale du cookie
  - cookie copié chez le client
- Le navigateur doit alors **systematiquement renvoyer** le cookie
  
- Sans donnée d'expiration : crée dans la mémoire vive
  - Sinon sous forme d'un fichier
  
- Stockage non sécurisé : souvent un simple fichier "texte"
  - Pas de donnée sensibles, confidentielles, etc ...

# Cookie : fonctionnement



# Cookie : en PHP

---

- Création et envoie de cookie :
  - Créer un header HTTP manuellement (déconseillé ...)
  - ou utiliser : `setcookie(nom, valeur)`
- Le cookie est dans l'entête HTTP
  - Utiliser la fonction avant la sortie de tout code HTML
- Lecture d'un cookie
  - `$_COOKIE[nom]` : récupère sa valeur
  - Ce tableau est en lecture seule
- Lister la tableau `$_COOKIE` pour avoir la liste complète des cookies envoyés par le client.
- Effacer un cookie : utiliser `setcookie(nom)` sans valeur.
  - Penser aussi à enlever le cookie dans `$_COOKIE` par un `unset()`

# Cookie : PHP

---

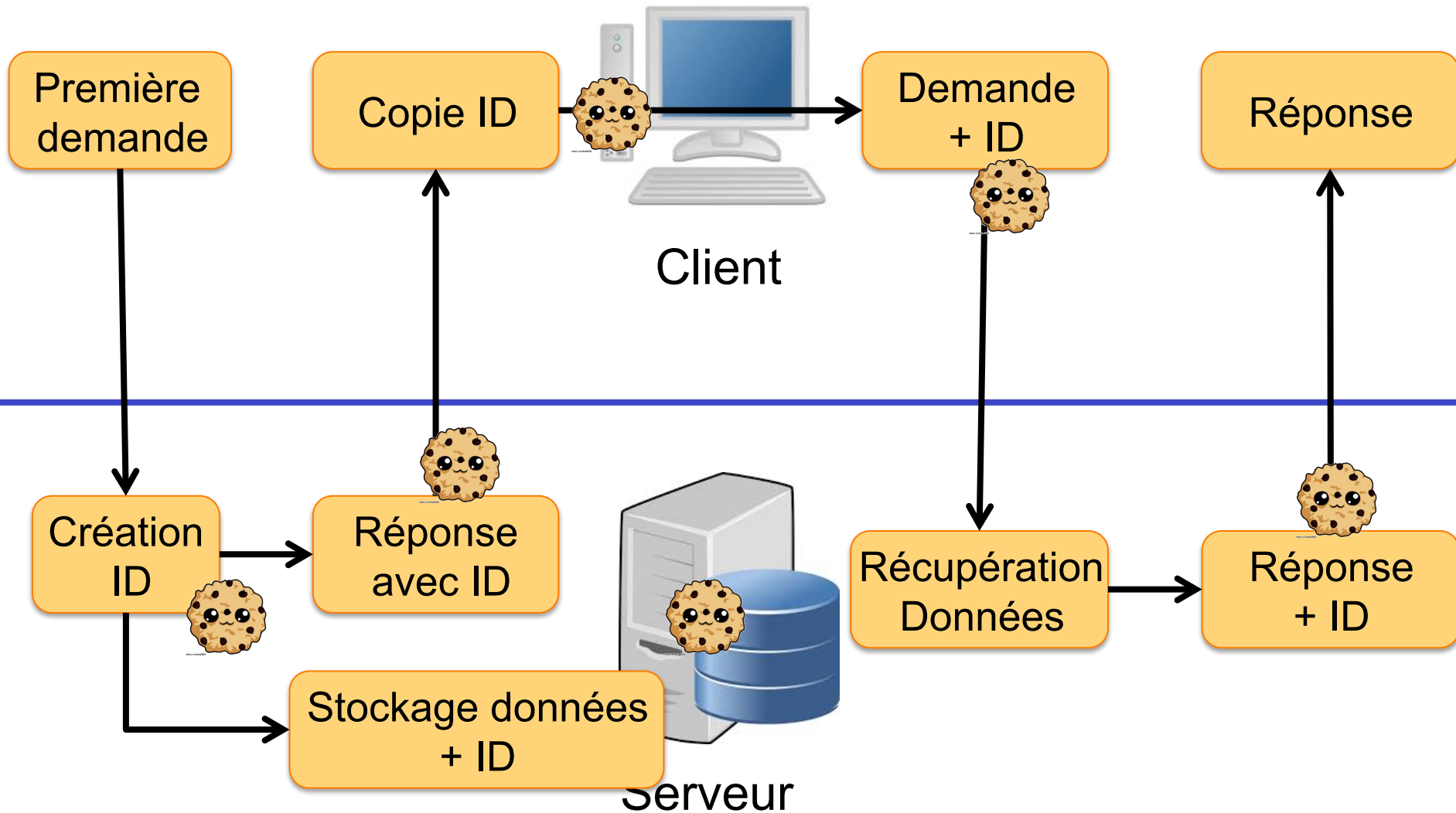
- Modifier un cookie :
  - Il suffit de renvoyer une nouvelle valeur avec le même nom.
- Date d'expiration :
  - Utiliser un troisième paramètre avec une date limite
  - Ex : `setcookie('nbVisite',$visite,mktime(0,0,0,12,31,2037));`
  - NB: timestamp Unix => max en 2037 !
- Stocker autre chose que des chaines dans le cookie
  - Sérialiser une donnée complexe : tableau ou objet
  - `$chaine = serialize($data);`
  - `setcookie('data',$chaine);`
  - dé-sérialiser à la réception :
  - `$data = unserialize($_COOKIE['data']);`

# Les sessions

---

- Inconvénients des cookies :
  - Stockage chez le client
  - Taille limitée
  - Modification possible par le client
- Sessions
  - Stockage d'informations sur le serveur : plus sûr !
  - Construction automatique d'un identifiant unique
  - Données correspondant à cet identifiant stockées dans le serveur
  - Pas de limitation de quantité de données à stocker
  - Mais ... session perdue avec la fermeture du navigateur

# Session : fonctionnement



# Les sessions en PHP

---

- Besoin de démarrer explicitement la gestion des sessions  
  `session_start();`
- Si aucune session ouverte
  - envoie d'un identifiant au client (id créé aléatoirement)
  - initialisation du tableau `$_SESSION`
- Si une session ouverte
  - Lecture de l'identifiant fournit par le client
  - Chargement des données correspondantes dans `$_SESSION`
- Les sessions peuvent utiliser un cookie pour stocker l'identifiant
  - Par défaut crée un cookie de nom 'PHPSESSID'
  - Mettre `session_start();` avant tout code HTML
- Modifier les données de la session
  - simplement modifier le tableau associatif



# Durée et ID d'une session

---

- Sur le client
  - Le temps de fonctionnement du navigateur
- Sur le serveur : dépend de la configuration
  - Par défaut durée sur le serveur de 1h30
- Suppression des données de session sur le serveur
  - `session_destroy();`
    - Attention : obligatoirement après un `session_start();`
    - N'efface pas les données de `$_SESSION`
- Forcer un id de session
  - `session_id()` : retourne l'id de la session en cours
  - `session_id(id)` : force l'identifiant et change de session
    - Si l'id n'existe pas, cela crée une nouvelle session

# Concurrence d'accès

---

- Les fichiers des données de sessions sur le serveur sont protégées par un mécanisme de verrou (voir Système d'Exploitation).
- Possibilité d'accéder aux données en lecture seule sans fermer les verrou  
`session_readonly();` à la place de `session_start();`
- Possibilité de forcer l'enregistrement des données de la session avant la fin du script  
`session_write_close();`

# Sécurité des sessions

---

- Si le client n'autorise pas les cookies sur son navigateur
  - PHP teste la disponibilité des cookies par le header HTTP
  - Si non disponible : ajout automatique d'une variable PHPSESSID dans toutes les URL
  - Risque plus grand pour la sécurité : No de session visible dans l'URL !
- Les fichiers sur le serveur ne sont pas cryptés
  - Risque de vol de données sur le serveur
  - Ne jamais stocker des informations 'sensibles'
- Les ID dans le HTTP ne sont pas cryptés
  - Risque de vol de session si analyse du flot internet
- L'id de l'utilisateur est en clair chez le client
  - Risque de vol de session, mais risque faible si l'ID de session est simplement en mémoire vive dans le code du navigateur