

Introduction à JavaScript



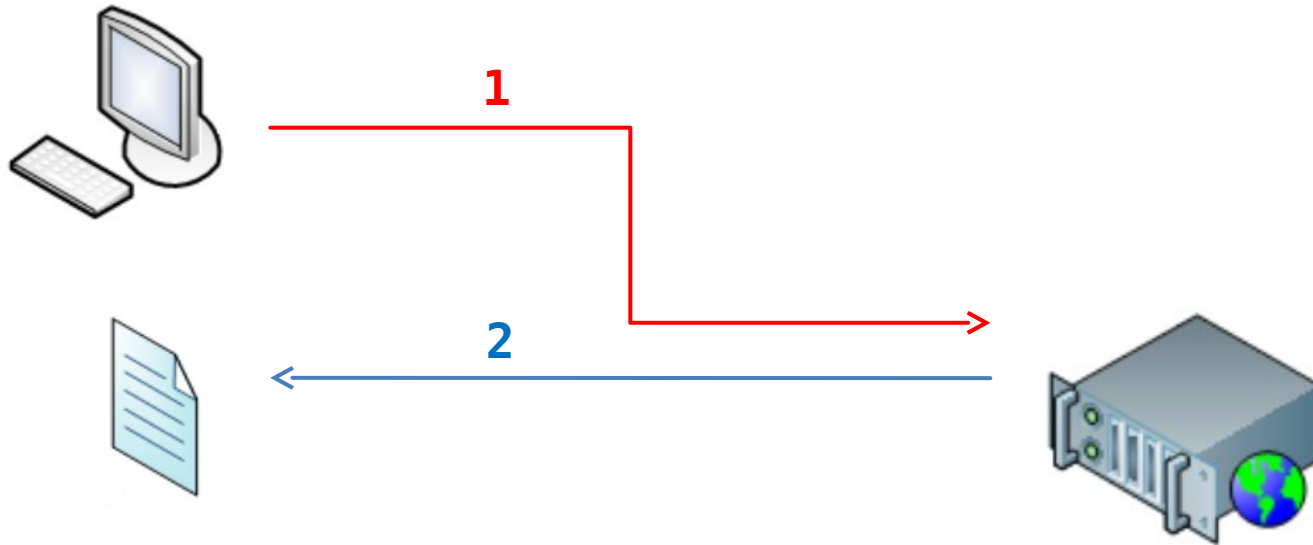
Plan du cours

VII.Asynchronisme en JavaScript

- 1 . Utilisation classique d'un serveur web
- 2 . Utilisation plus dynamique d'un serveur web
- 3 . Ciblage et Asynchronisme
- 4 . Technologies utilisées
- 5 . Exemple de fichier de données en format JSON
- 6a. Exemple de traduction JSON → JavaScript
- 6b. Exemple de traduction JavaScript → JSON
- 7a. L'interface XMLHttpRequest
- 7b. Méthodes et attributs d'un objet XMLHttpRequest

Asynchronisme en JavaScript

1. Utilisation classique d'un serveur web

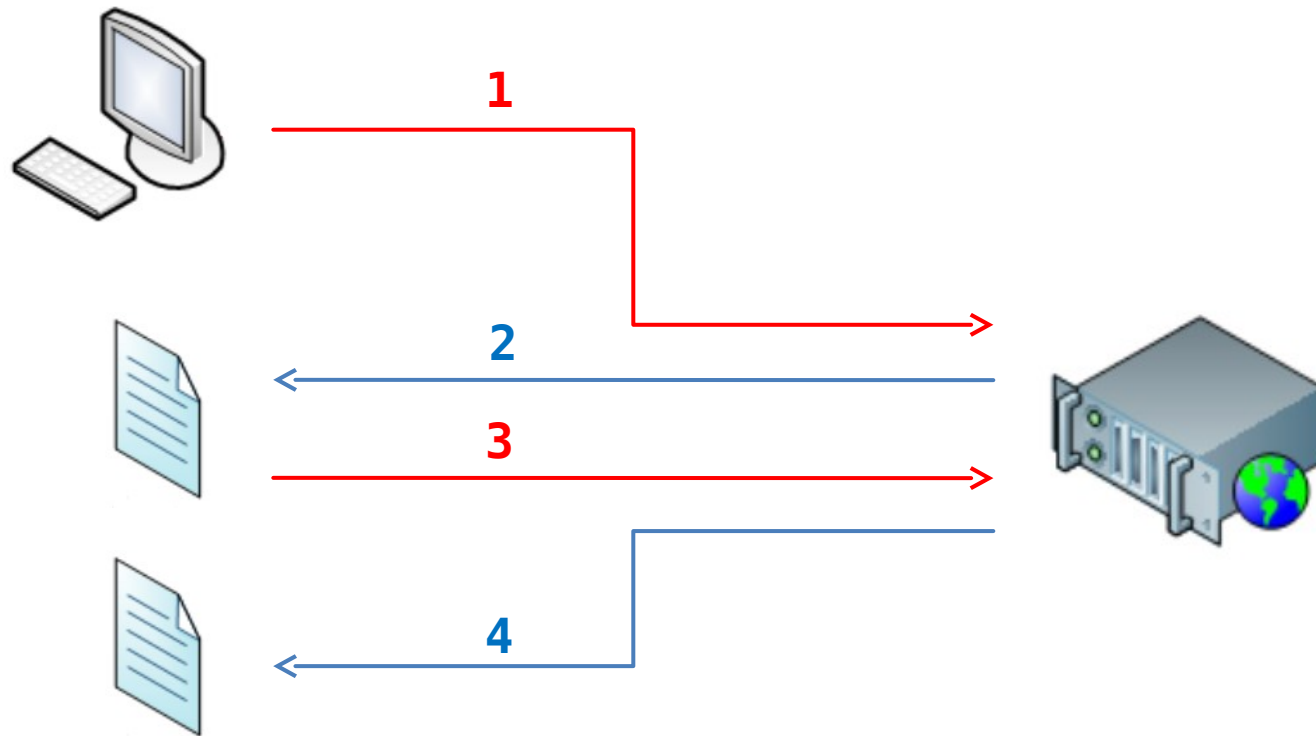


1 → requête http au serveur, par une URL

2 → réponse du serveur : html, scripts, images, css, ...

Asynchronisme en JavaScript

1. Utilisation classique d'un serveur web



1 → requête http au serveur, par une URL

2 → réponse du serveur : html, scripts, images, css, ...

3 → nouvelle requête du client (ex : par un lien dans la page chargée)


4 → nouvelle réponse du serveur = nouvelle page

Asynchronisme en JavaScript

1. Utilisation classique d'un serveur web

recherche par requête http classique

vosre recherche



Asynchronisme en JavaScript

1. Utilisation classique d'un serveur web

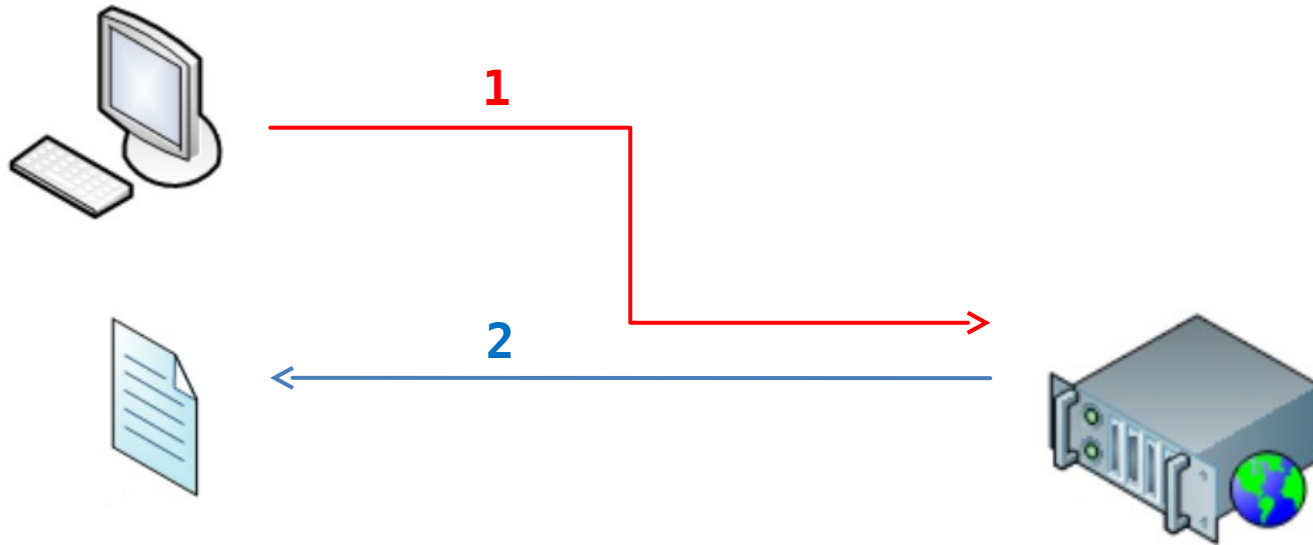
recherche par requête http classique

vosre recherche

- Toussieux
- Toulis-et-Attencourt
- Toulon-sur-Allier
- Toudon
- Touët-de-l'Escarène

Asynchronisme en JavaScript

2. Utilisation plus dynamique d'un serveur web

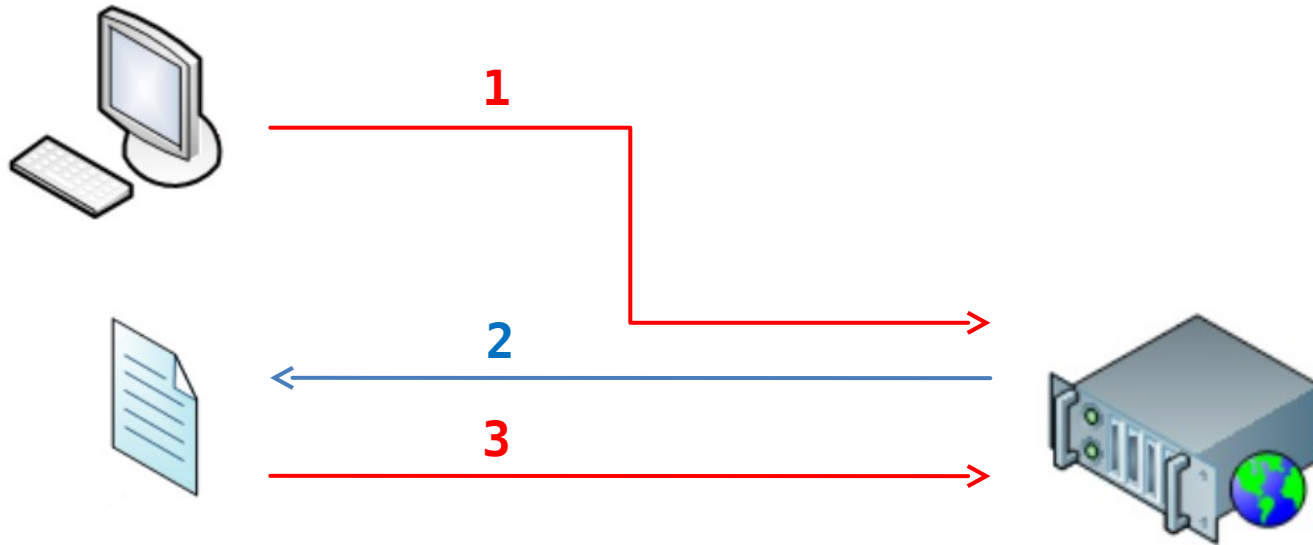


1 → requête http au serveur, par une URL

2 → réponse du serveur : html, scripts, images, css, ...

Asynchronisme en JavaScript

2. Utilisation plus dynamique d'un serveur web



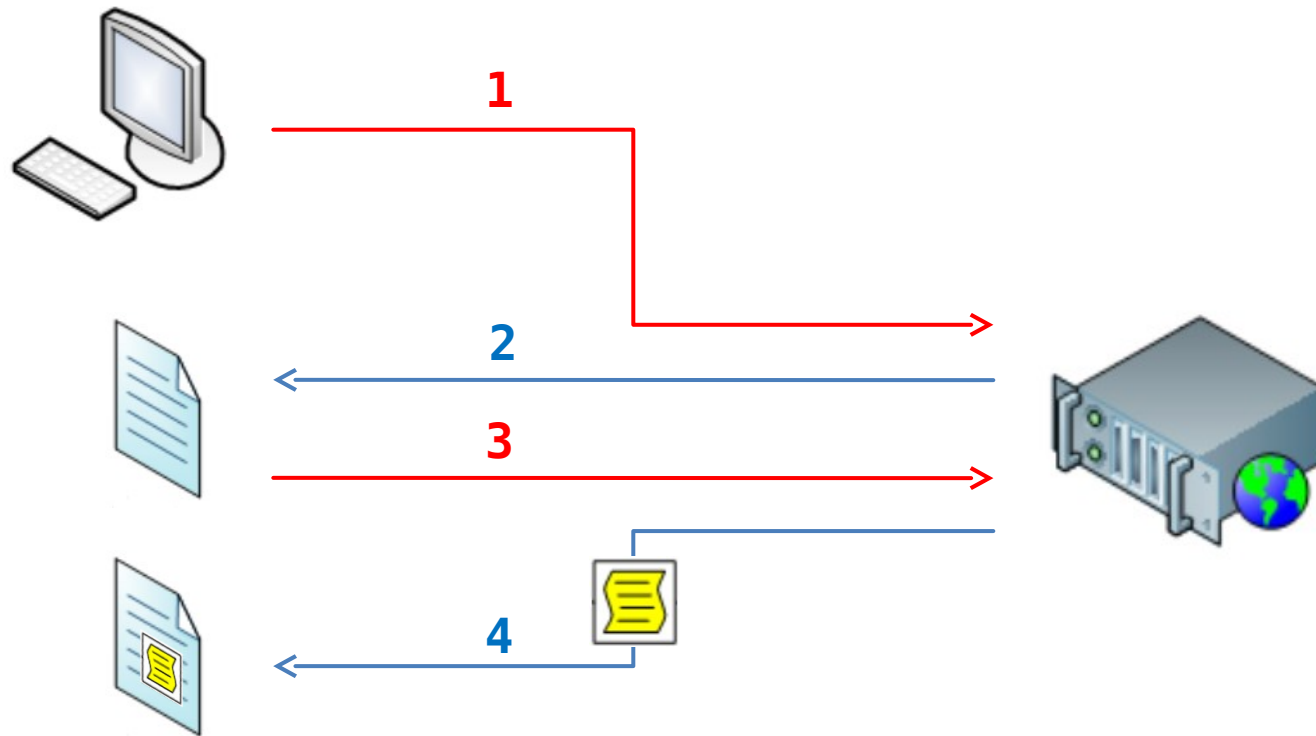
1 → requête http au serveur, par une URL

2 → réponse du serveur : html, scripts, images, css, ...

3 → requête de données du client

Asynchronisme en JavaScript

2. Utilisation plus dynamique d'un serveur web



1 → requête http au serveur, par une URL

2 → réponse du serveur : html, scripts, images, css, ...

3 → requête de données du client

4 → le serveur envoie les données, une partie de la page est actualisée

Asynchronisme en JavaScript

2. Utilisation plus dynamique d'un serveur web

recherche par requête AJAX

vosre recherche

- Talissieu
- Tenay
- Thézillieu
- Thil
- Thoiry

Asynchronisme en JavaScript

3. Ciblage et asynchronisme

- Ciblage : on peut affecter seulement une partie de la page, sans avoir tout à reconstruire (ex : la div qui accueille les noms de ville).

Asynchronisme en JavaScript

3. Ciblage et asynchronisme

- Ciblage : on peut affecter seulement une partie de la page, sans avoir tout à reconstruire (ex : la div qui accueille les noms de ville).
- Asynchronisme : on dissocie les tâches.
 - On lance la requête de données (ex : au changement de l'input de recherche, requête lancée).

Asynchronisme en JavaScript

3. Ciblage et asynchronisme

- Ciblage : on peut affecter seulement une partie de la page, sans avoir tout à reconstruire (ex : la div qui accueille les noms de ville).
- Asynchronisme : on dissocie les tâches.
 - On lance la requête de données (ex : au changement de l'input de recherche, requête lancée).
 - Entre le lancement de la requête et la réception complète des données, le reste du script n'est pas arrêté et continue de s'exécuter.

Asynchronisme en JavaScript

3. Ciblage et asynchronisme

- Ciblage : on peut affecter seulement une partie de la page, sans avoir tout à reconstruire (ex : la div qui accueille les noms de ville).
- Asynchronisme : on dissocie les tâches.
 - On lance la requête de données (ex : au changement de l'input de recherche, requête lancée).
 - Entre le lancement de la requête et la réception complète des données, le reste du script n'est pas arrêté et continue de s'exécuter.
 - Quand les données sont toutes arrivées, on lance le traitement prévu de ces données (ex : remplissage de la div). Une fonction appelée fonction callback s'en charge.

Asynchronisme en JavaScript

4. Technologies utilisées

- JavaScript pour l'objet qui gèrera la requête au serveur (objet XMLHttpRequest) ;

Asynchronisme en JavaScript

4. Technologies utilisées

- JavaScript pour l'objet qui gèrera la requête au serveur (objet XMLHttpRequest) ;
- PHP côté serveur pour communiquer avec la base de données (mais ce peut être un autre langage côté serveur).

Asynchronisme en JavaScript

4. Technologies utilisées

- JavaScript pour l'objet qui gérera la requête au serveur (objet XMLHttpRequest) ;
- PHP côté serveur pour communiquer avec la base de données (mais ce peut être un autre langage côté serveur).
- JSON comme support de communication (format de données) pour transmettre les données à JavaScript. JSON = JavaScript Object Notation.

Asynchronisme en JavaScript

4. Technologies utilisées

- JavaScript pour l'objet qui gèrera la requête au serveur (objet XMLHttpRequest) ;
- PHP côté serveur pour communiquer avec la base de données (mais ce peut être un autre langage côté serveur).
- JSON comme support de communication (format de données) pour transmettre les données à JavaScript. JSON = JavaScript Object Notation.
- L'ensemble de ces technologies est regroupé sous le nom AJAX (pour Asynchronous JavaScript And XML, car le format de données XML était plus utilisé avant).

Asynchronisme en JavaScript

7a. L'interface XMLHttpRequest

- Un objet implémentant cette interface JavaScript pourra matérialiser la communication entre le client et le serveur pour le lancement de la requête au serveur ;
- Il peut établir une communication synchrone ou asynchrone avec le serveur. Une communication en mode asynchrone n'est pas bloquante.
- Il est instancié ainsi :

```
> let xhr = new XMLHttpRequest();
```

- Il dispose de méthodes pour ouvrir une requête, l'envoyer, l'abandonner, connaître l'évolution de son statut, connaître le contenu de la réponse.

Asynchronisme en JavaScript

7b. Méthodes et attributs d'un objet XMLHttpRequest

a) la méthode open (ouvre la requête)

```
> xhr.open("GET", "http://www.monsite.fr/exemple.php", true);
```

↓
GET ou POST

↓
URL à visiter

↓
mode asynchrone

Asynchronisme en JavaScript

7b. Méthodes et attributs d'un objet XMLHttpRequest

a) la méthode `open` (ouvre la requête)

```
> xhr.open("GET", "http://www.monsite.fr/exemple.php", true);
```

↓
GET ou POST

↓
URL à visiter

↓
mode asynchrone

b) la méthode `send` (envoie la requête avec un corps de requête)

```
> xhr.send(contenu);
```

- si la méthode est GET, contenu est null;
- si la méthode est POST, contenu est soit null soit égal à une chaîne du type "param1=valeur1¶m2=valeur2&..."

Asynchronisme en JavaScript

7b. Méthodes et attributs d'un objet XMLHttpRequest

c) l'attribut readyState

Il indique l'état de réception des données :

- readyState = 0 : objet créé mais pas ouvert (avant open)
- readyState = 1 : après open, mais avant send
- readyState = 2 : après send, avant réception de données
- readyState = 3 : données en cours de réception
- readyState = 4 : données entièrement reçues

Asynchronisme en JavaScript

7b. Méthodes et attributs d'un objet XMLHttpRequest

c) l'attribut readyState

Il indique l'état de réception des données :

- `readyState = 0` : objet créé mais pas ouvert (avant `open`)
- `readyState = 1` : après `open`, mais avant `send`
- `readyState = 2` : après `send`, avant réception de données
- `readyState = 3` : données en cours de réception
- `readyState = 4` : données entièrement reçues

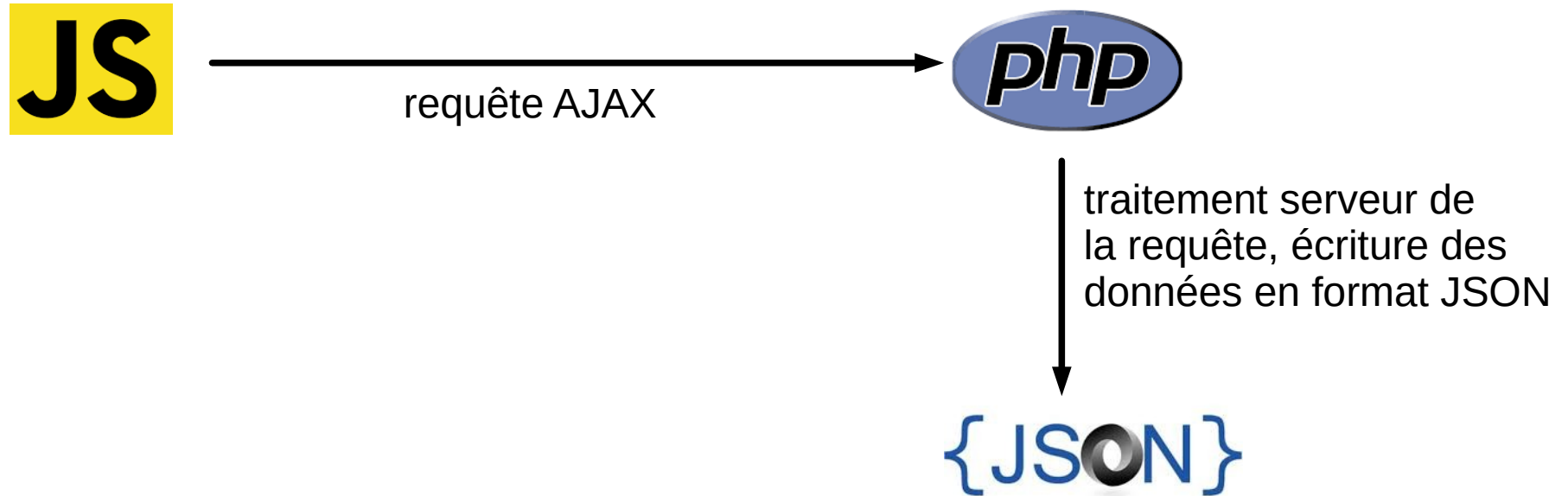
b) l'attribut responseText

Il contient, sous forme d'une chaîne de caractères, les données en réponse à la requête. Il n'est complet que si `readyState` est à la valeur 4.

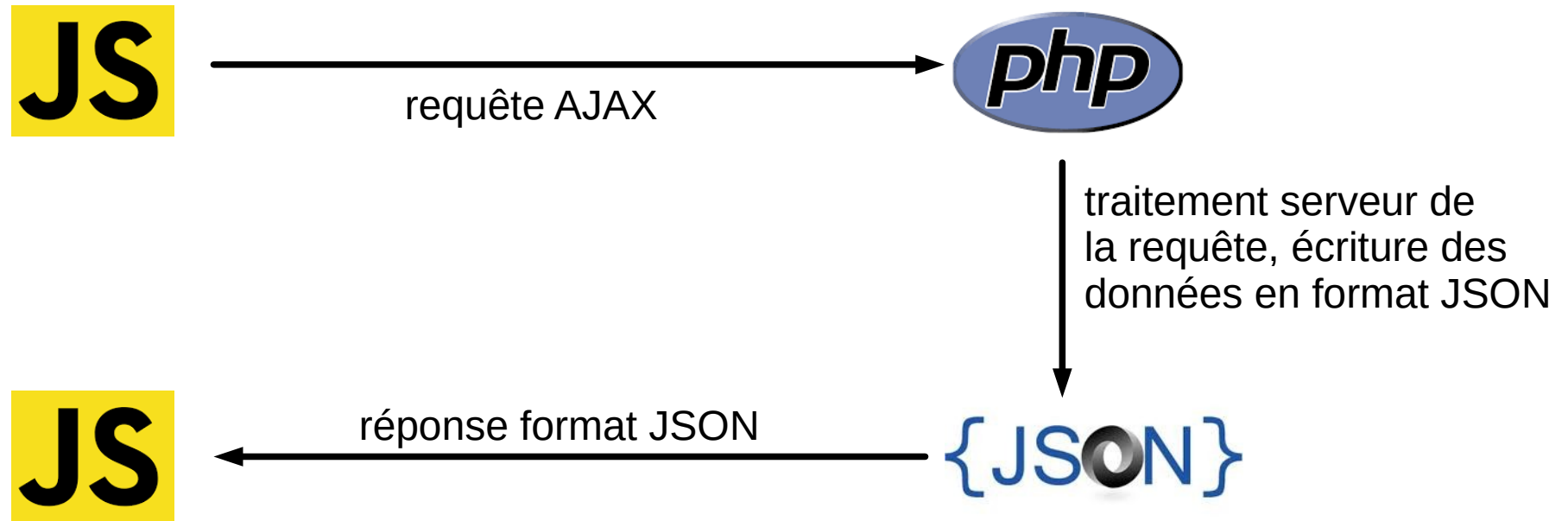
Asynchronisme en JavaScript



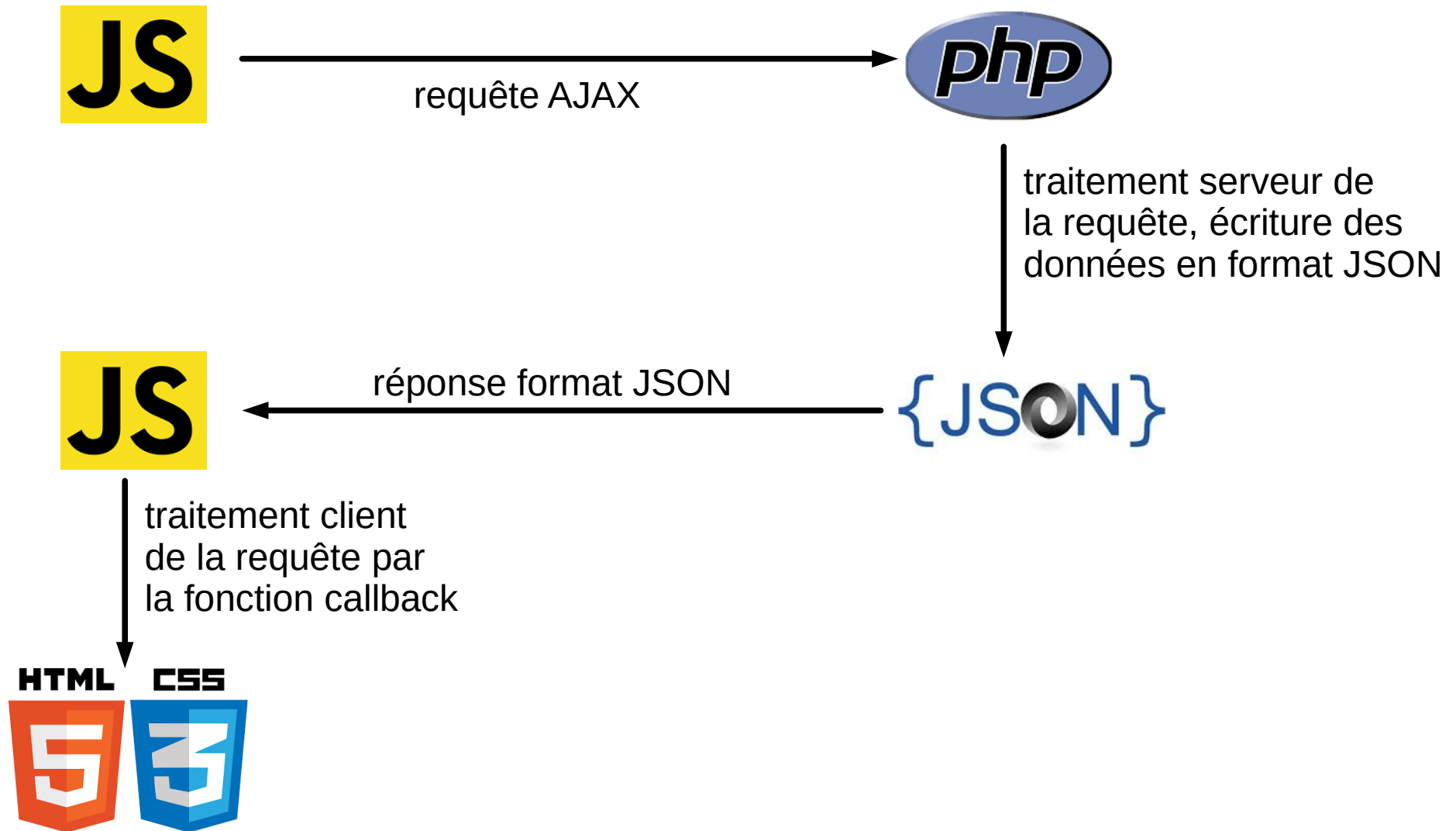
Asynchronisme en JavaScript



Asynchronisme en JavaScript

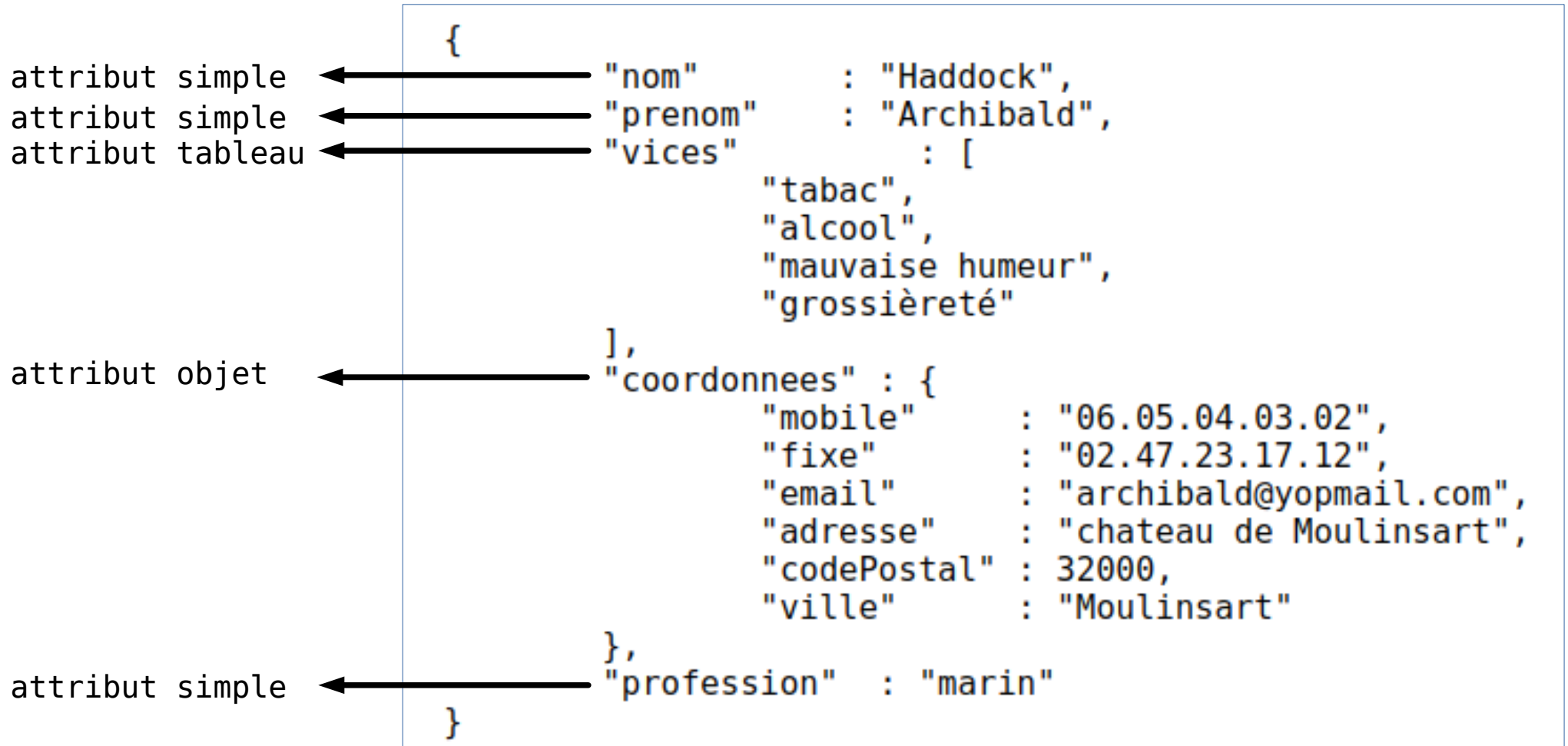


Asynchronisme en JavaScript



Asynchronisme en JavaScript

5. Exemple de fichier de données en format JSON



Asynchronisme en JavaScript

6a. Exemple de traduction JSON → JavaScript

```
> let chaine_B = '{"x":-2,"y":5,"couleur":"jaune","marqueur":"rond"}'  
< undefined  
  
> let B = JSON.parse(chaine_B)  
< undefined  
  
> B  
< ▼ {x: -2, y: 5, couleur: "jaune", marqueur: "rond"} ⓘ  
    x: -2  
    y: 5  
    couleur: "jaune"  
    marqueur: "rond"
```

Asynchronisme en JavaScript

6b. Exemple de traduction JavaScript → JSON

```
> class Point {  
    constructor(x,y,couleur,marqueur) {  
        this.x = x;  
        this.y = y;  
        this.couleur = couleur;  
        this.marqueur = marqueur;  
    }  
}  
< undefined  
  
> let A = new Point(5,-3,"rouge","croix")  
< undefined  
  
> A  
< ▶ Point {x: 5, y: -3, couleur: "rouge", marqueur: "croix"}  
  
> JSON.stringify(A)  
< '{"x":5,"y":-3,"couleur":"rouge","marqueur":"croix"}'
```