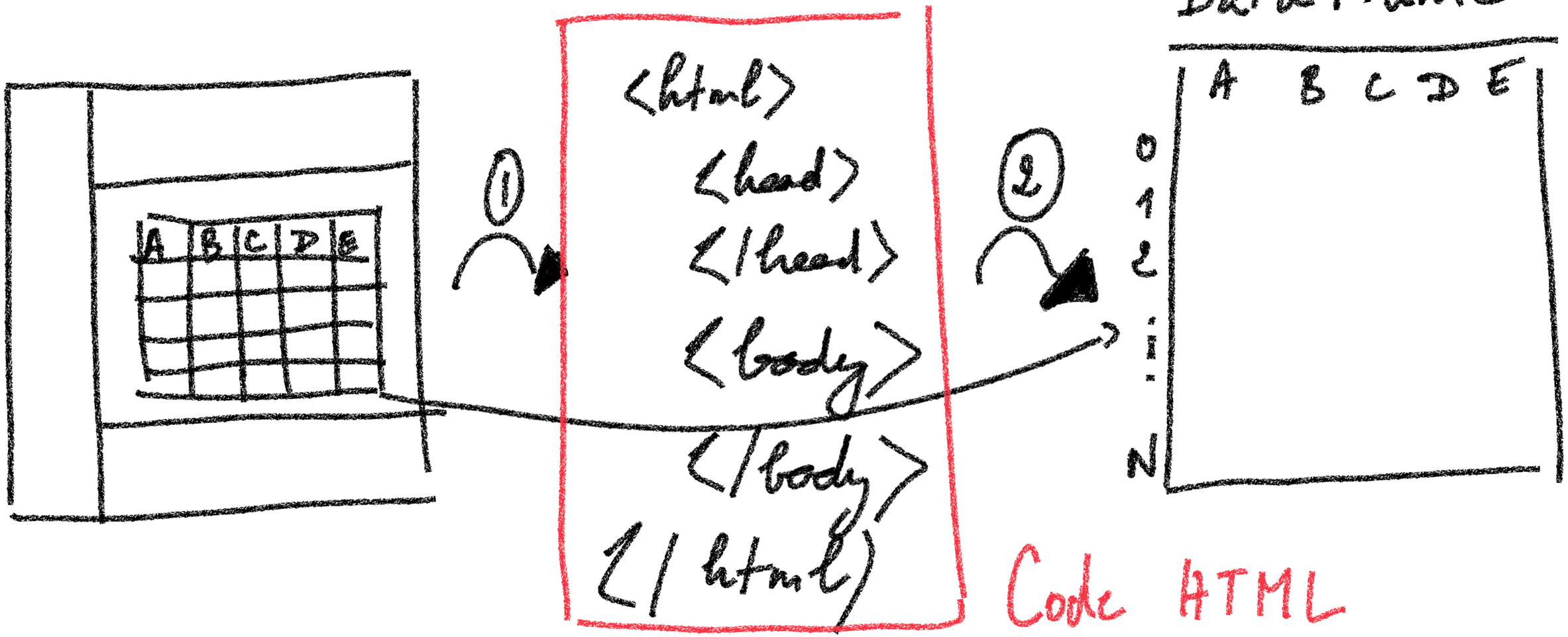
**Web Scraping** Aspirer les données d'une page web de manière structurée.



## ① Aspirer le code source :

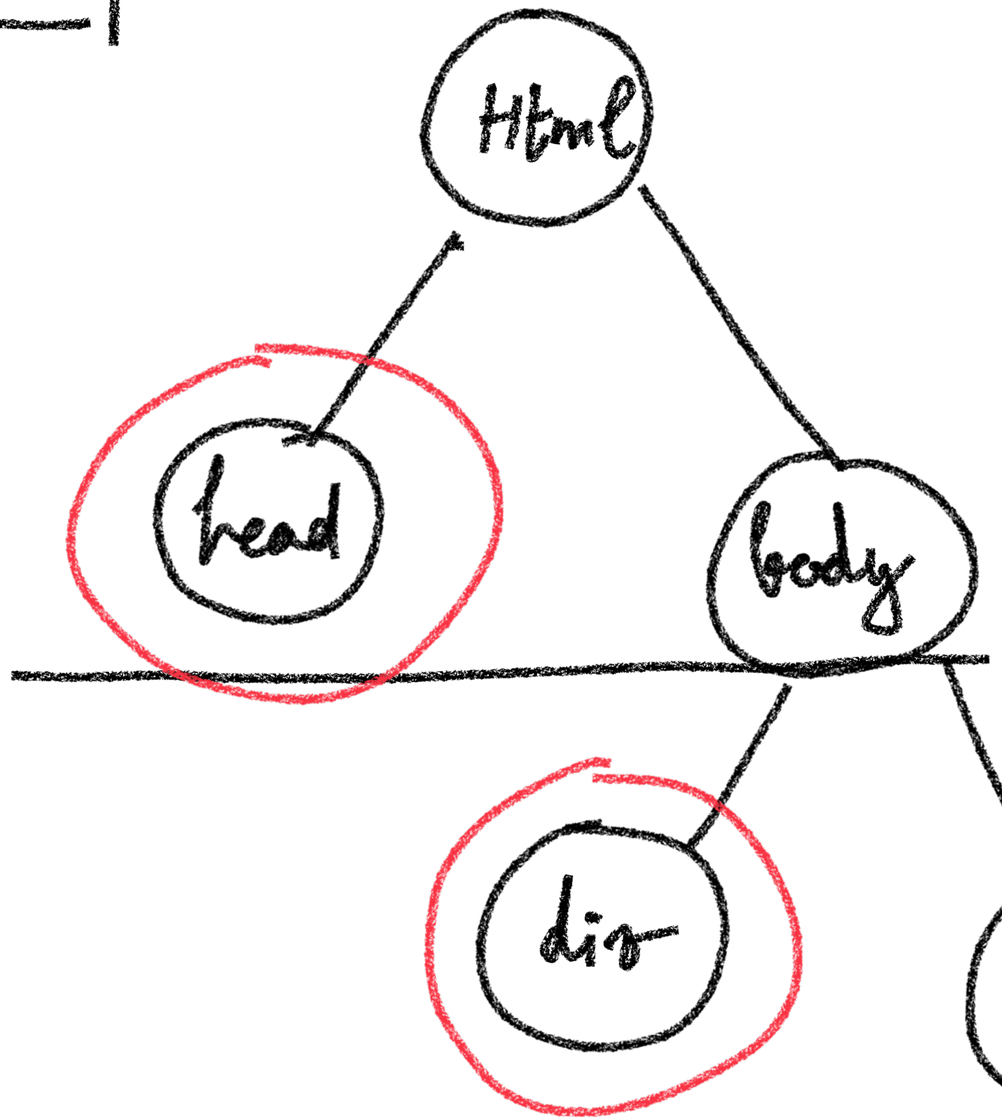
(= Récupérer le code source sous la forme d'un string)

Requête GET de l'URL de la page

```
> res : requests.get("http://x.y.z/a/b/c")
> res.content
```

# Dom HTML

## Document - Object Model = ARBRE




<html>

→ <head>

...

→ </head>

→ <body>

→ →  ... 

→ ...

→ </body> </html>

## (2a) Analyser / Parser le code HTML

Beautiful Soup

```
soup = BeautifulSoup(res.content, "html")
```

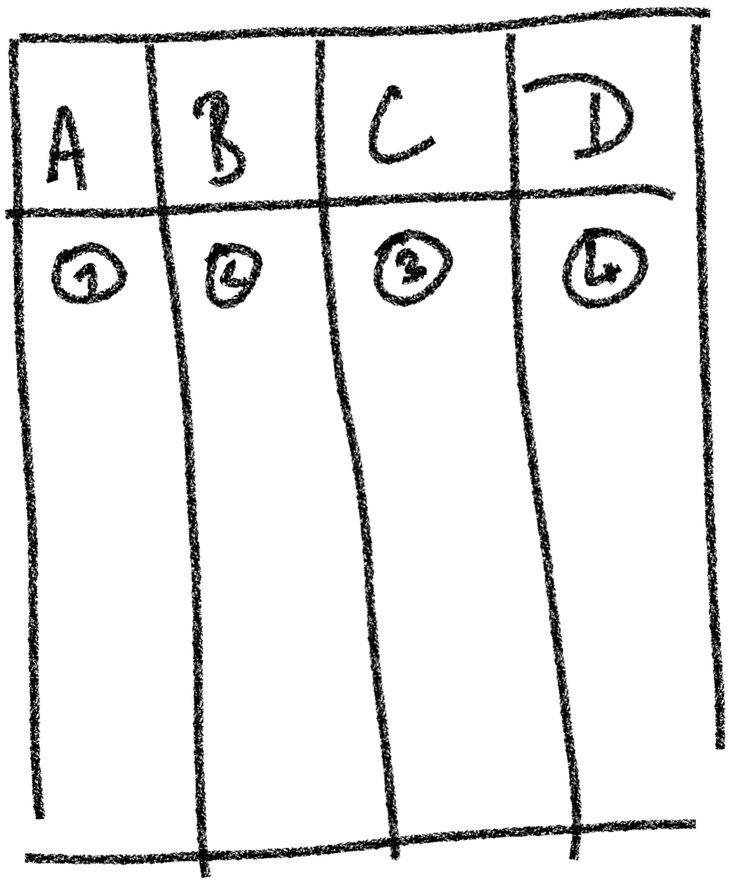
### (2b) Recherche ciblée d'informations :

| | | |
|---|---|---|
| soup. | select_one(...) | : renvoie la première balise qu'il trouve |
| | select_all(...) | : renvoie la liste de toutes les balises qui matchent |

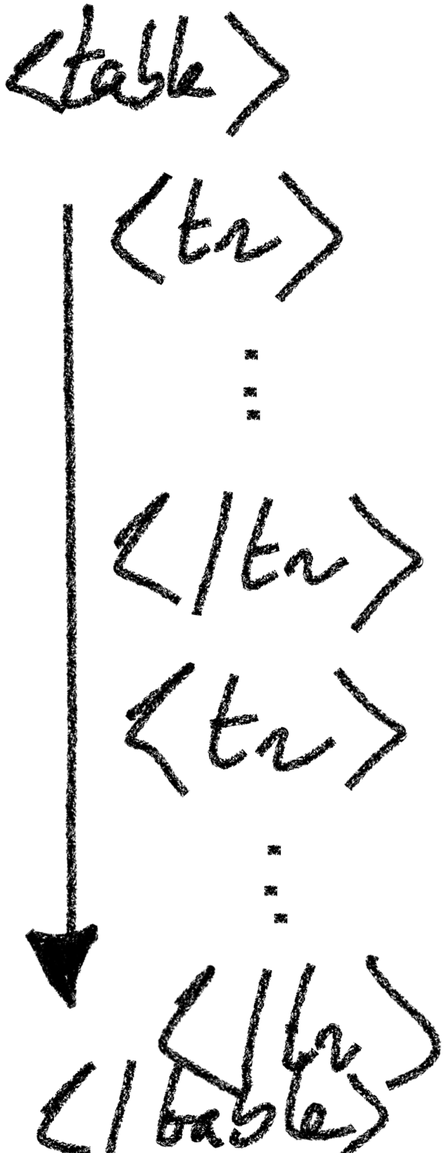
Soup.select* ⇒ un objet de type BeautifulSoup

↳ .select*

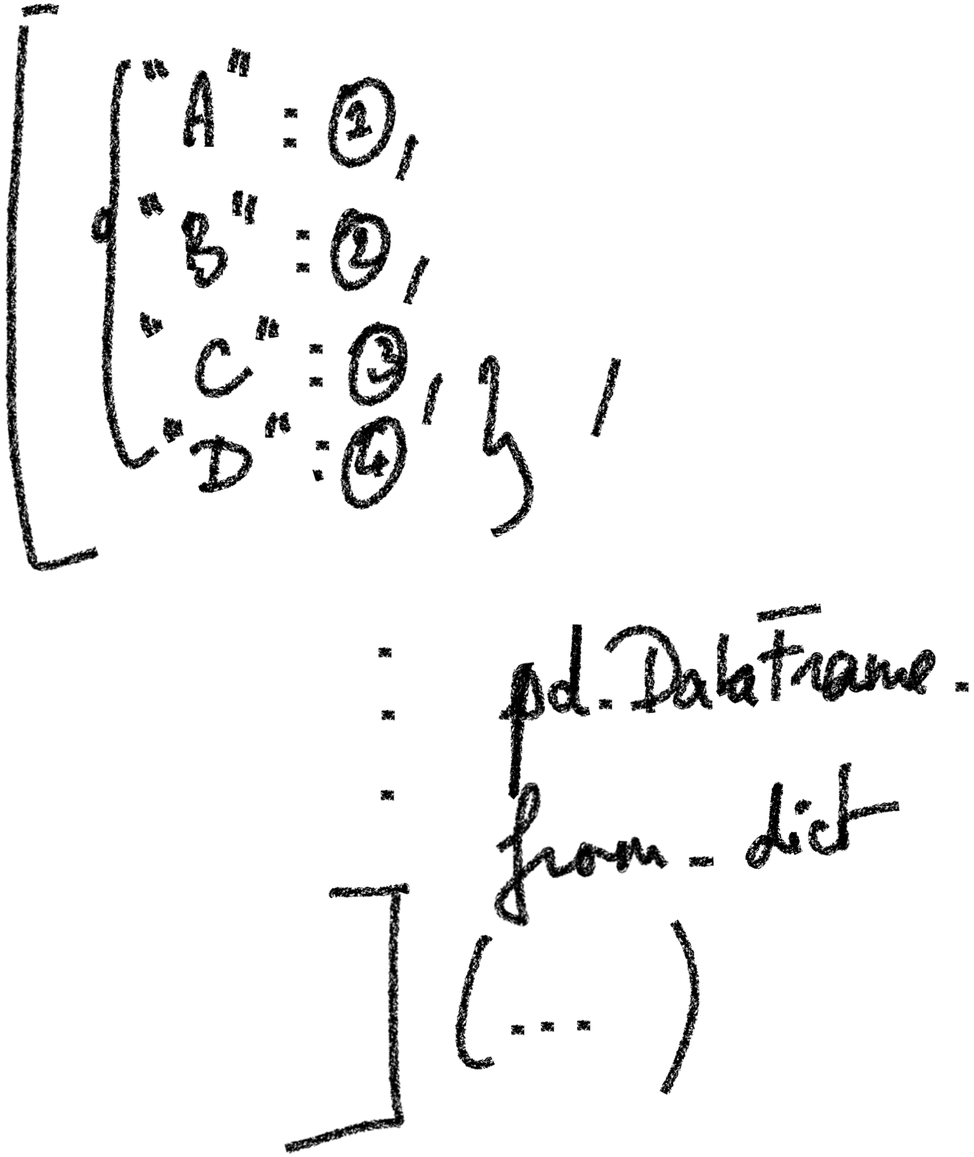
# (2c) Construire un DataFrame :




⇒




⇒




pd.DataFrame.from_dict(...)

# ① Limitations légales :

https://www.x.y/robots.txt

```
User-Agent : *
Allowed :
/a/b

Disallowed :
/c/d
```

User Agent = Navigateur

# ① Limitations techniques :

## ②a Limites de quota de requêtes :

Par ex : une seule requête / 10 s.

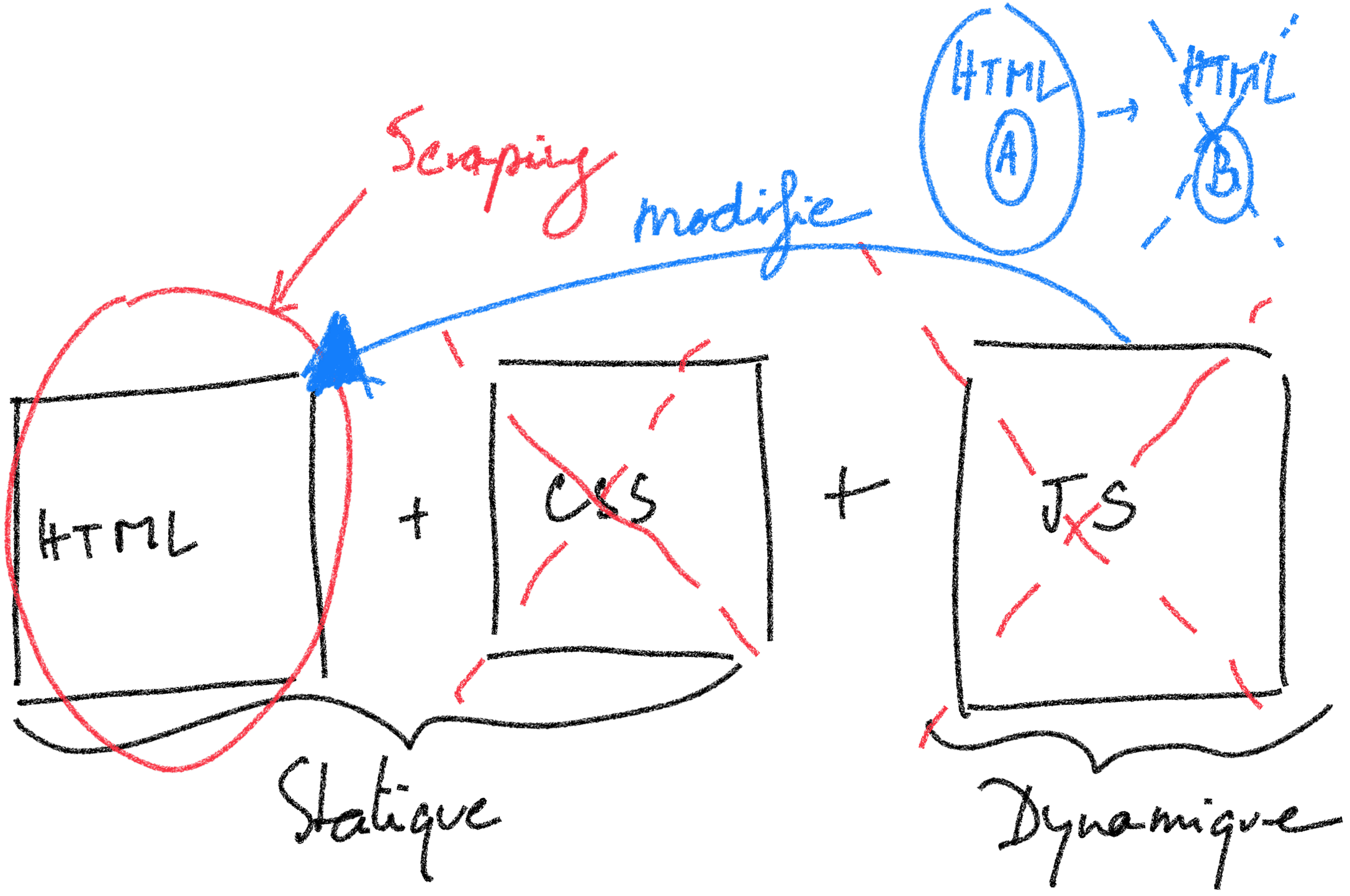
robots.txt

Ferme d'IP

## ②b Limites de scalabilité :

i) : changer de manière aléatoire les ids / class des balises (leboncoin.fr)

iii): mettre l'essentiel du contenu de la page dans le JavaScript (et non dans le HTML)


HTML
A
HTML
B
Scraping
modifie
HTML
+
CSS
+
JS
Statique
Dynamique

# Scraping ++:

## Headless Browsers (Selenium)

```
brows = ChromeBrowser(...)
    brows.go("a/b/c")
```