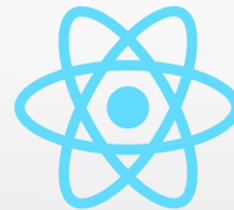




**CSA2**  
**Application Côté Client 2**

# Introduction à React



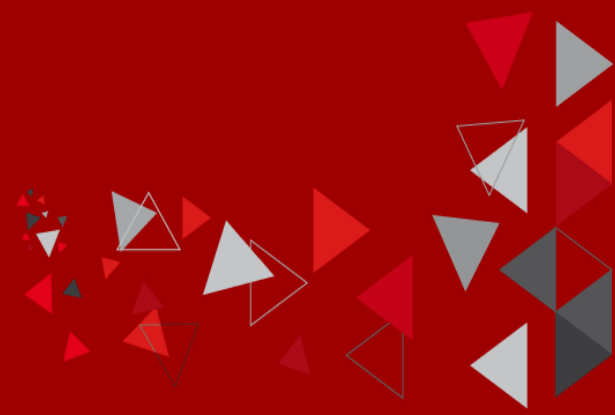
# Prérequis

☐ HTML / CSS

☐ JavaScript

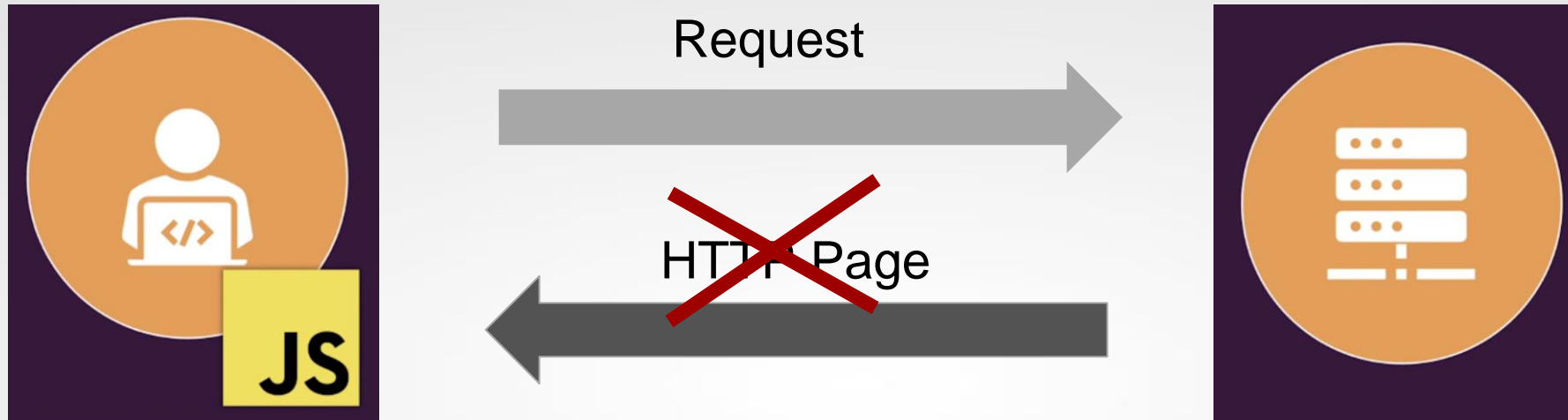


- ☐ Pourquoi le JavaScript côté FrontEnd?
- ☐ SPA vs MPA
- ☐ Bibliothèque vs Framework
- ☐ React?
- ☐ VDOM vs DOM
- ☐ Environnement de travail & installation



# Pourquoi le JavaScript côté FrontEnd?

# ► Pourquoi le JavaScript côté FrontEnd?

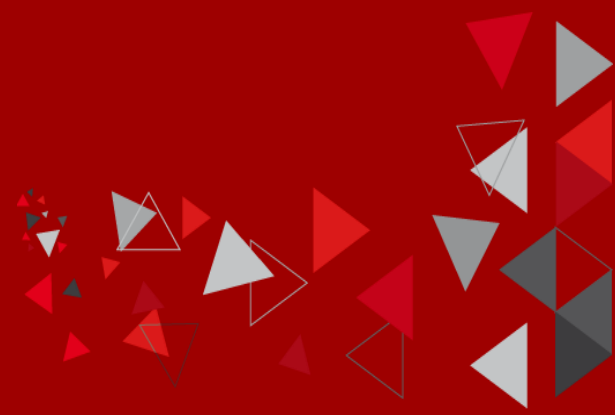


Pas besoin d'attendre une nouvelle page HTML comme réponse

# Pourquoi le JavaScript côté FrontEnd?

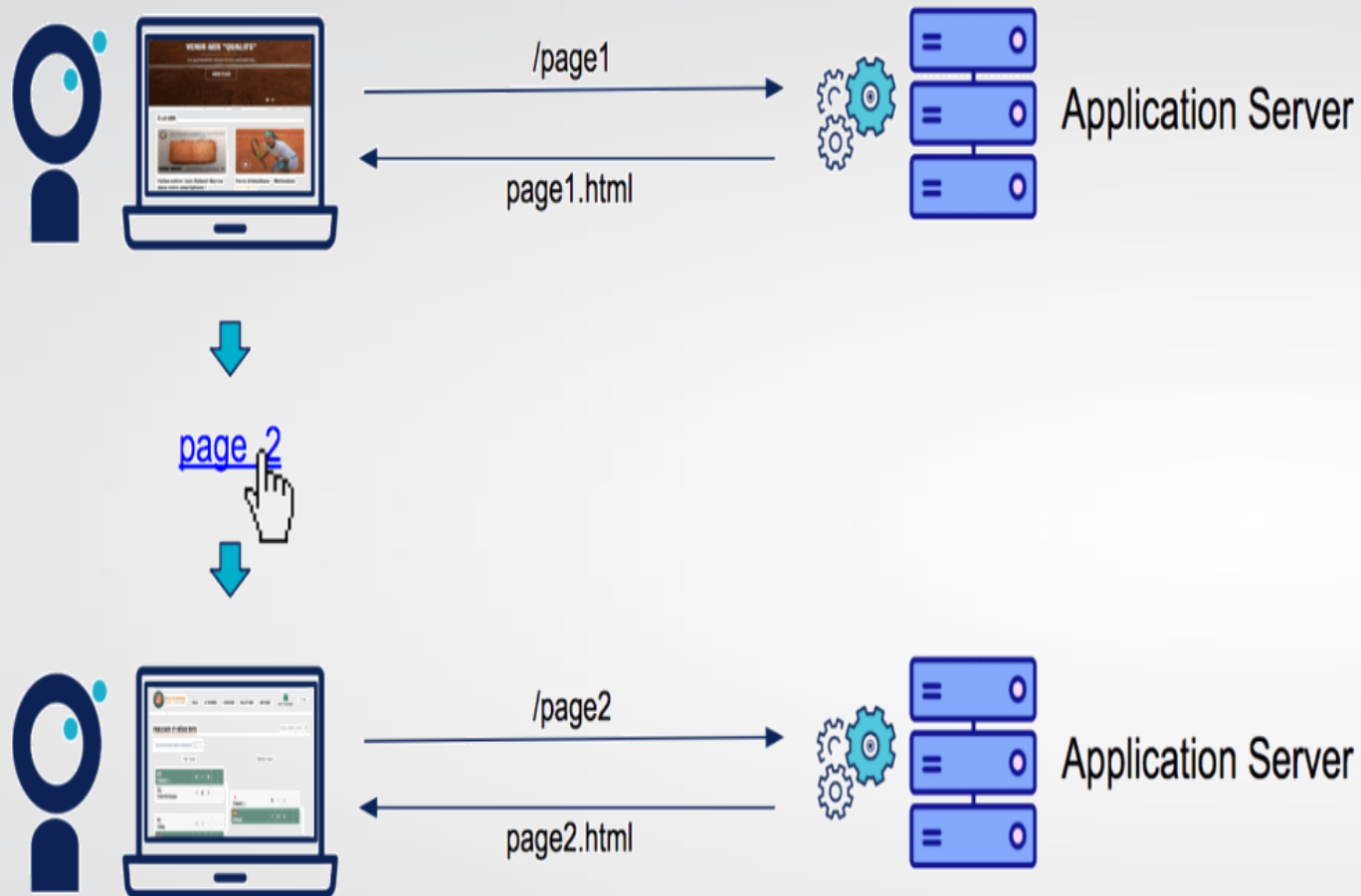


- ❑ JavaScript est un langage de programmation qui peut être utilisé de manière **asynchrone**, ce qui signifie que plusieurs tâches peuvent être exécutées simultanément sans bloquer l'exécution du code principal.
- ❑ Le JavaScript permet de développer des applications **monopage** qui ne nécessitent pas le rechargement de la page entière lorsque l'utilisateur interagit avec l'application.



# MPA vs SPA

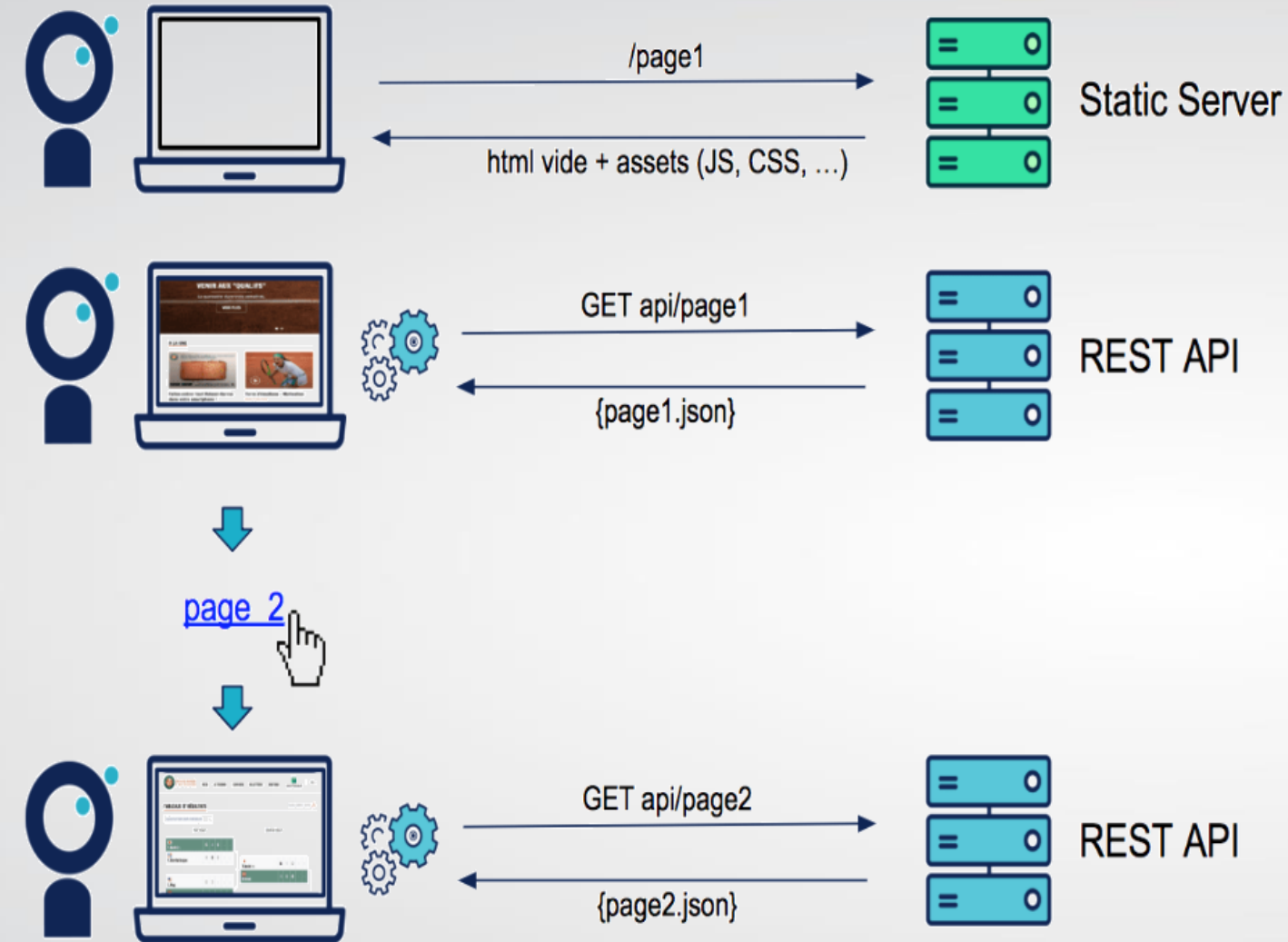
# ▶ MPA: Application multi-pages



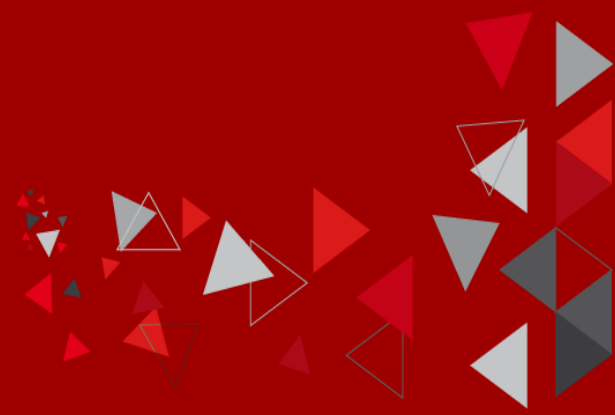
1. l'utilisateur demande la page `"/page1"`.
2. Le serveur récupère les données et alimente le template de la page.
3. le serveur répond à la page complète.
4. le navigateur télécharge les assets demandés : JavaScript, CSS, images.
5. lorsque l'utilisateur demande une nouvelle page, le processus reprend à la 2ème étape.



# SPA: Application monopage

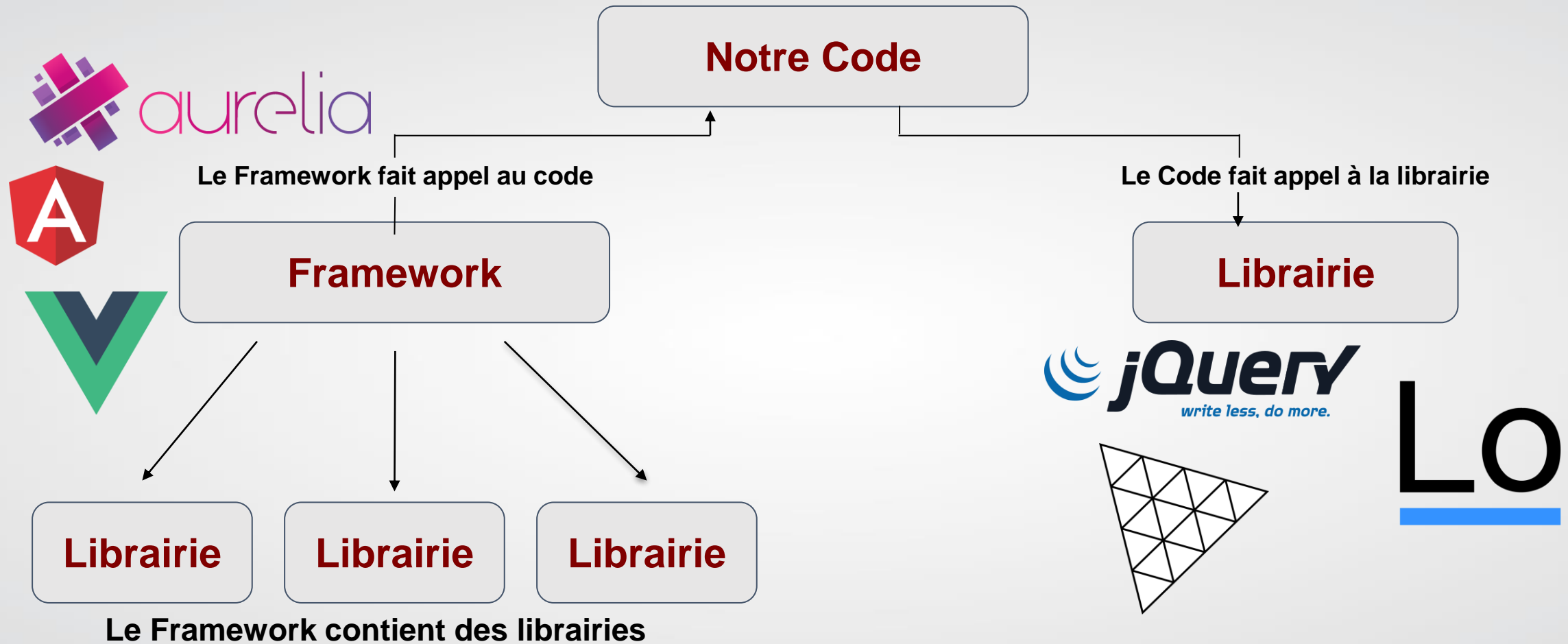


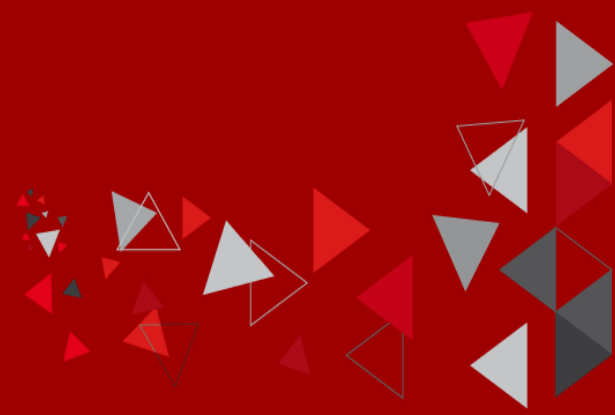
1. L'utilisateur demande la page à l'url `"/page1"`
2. Le serveur renvoie toujours le `index.html`
3. Le navigateur affiche le `index.html` (page blanche ou page de chargement)
4. Le navigateur va télécharger les assets : JavaScript, CSS, images
5. Le navigateur parse et exécute le JavaScript
6. Le JS construit le HTML et rend l'application interactive
7. Le JS fait des appels à l'API et complète le HTML
8. La navigation se fait entièrement côté navigateur
9. Le client fait directement des appels d'API pour récupérer ou mettre à jour les données supplémentaires.



# Classement des Frameworks/librairies

# Bibliothèque vs Framework





**React ?**



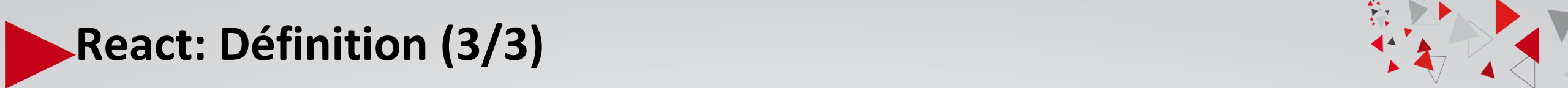
- ❑ Bibliothèque JavaScript (Libre - Open source) Ce n'est pas un Framework !
- ❑ Développée, par Facebook depuis 2013, par **Jordan Walke**.
- ❑ Utilisée pour la création d'interfaces web monopage (SPA).
- ❑ Basée sur une approche de programmation orientée composants (**Web Components**)



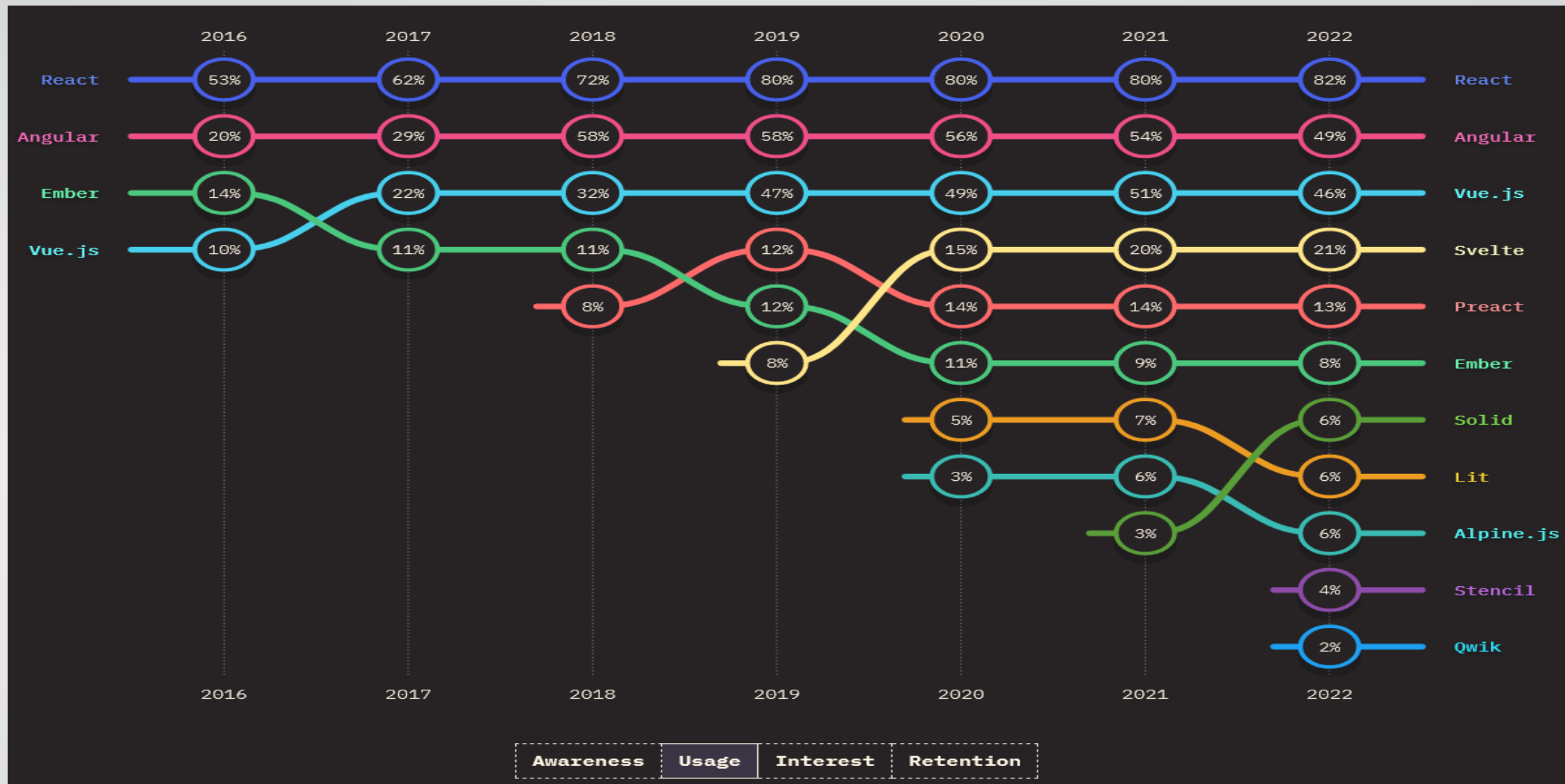
## React: Définition (2/3)



- ☐ React utilise la syntaxe JSX (JavaScript XML).
- ☐ React utilise le Virtual DOM (VDOM).
- ☐ Une librairie très légère.
- ☐ Il a une communauté active.



# React: Définition (3/3)



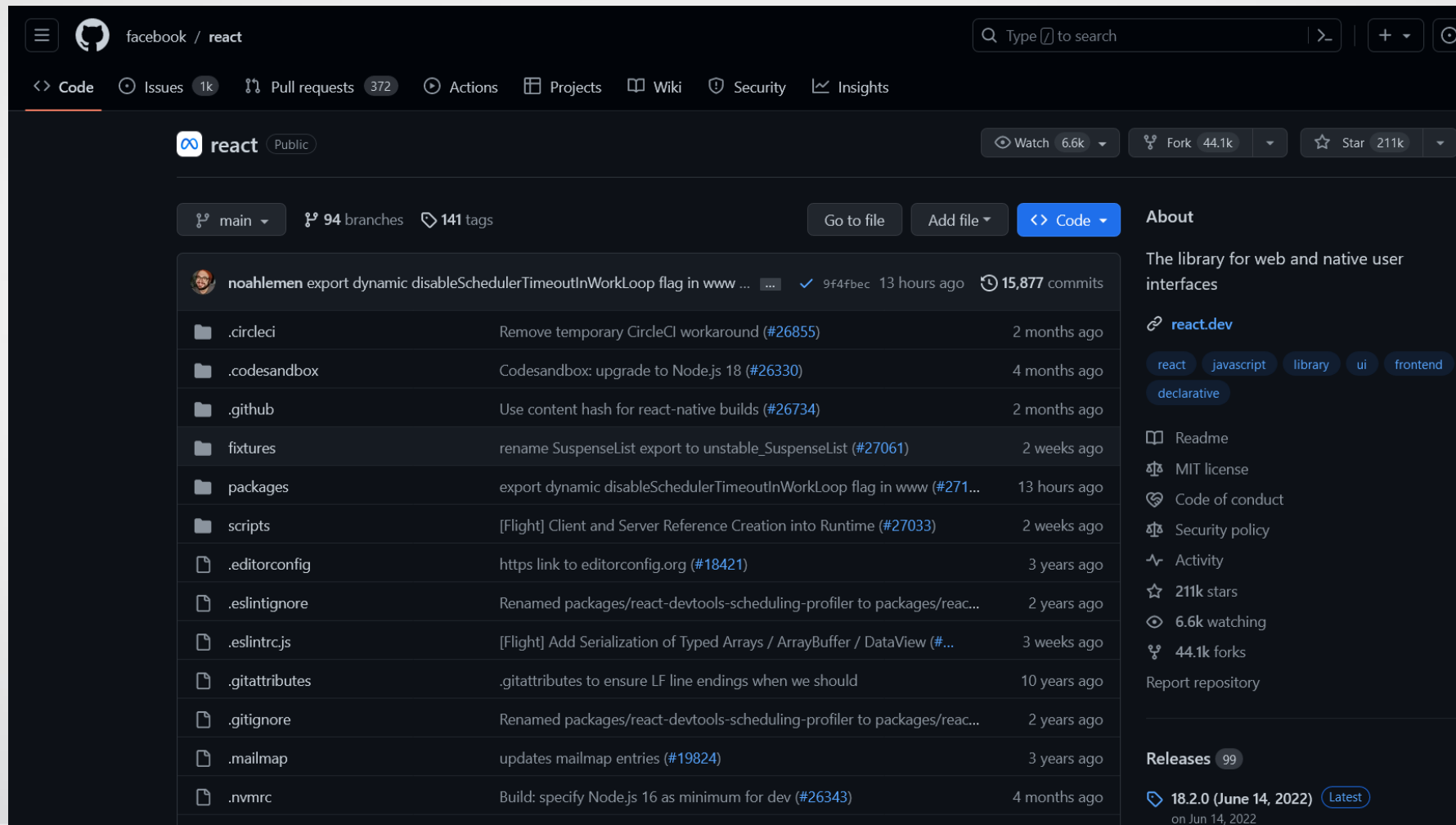
Classement des Frameworks Front-End les plus populaires suivant les ratios d'utilisation



# React: Librairie



- La taille de la librairie React est de 291 kb
- React a dans les environs de 1600+ contributeurs



The screenshot shows the GitHub repository for React, owned by facebook. The repository is public and has 6.6k watches, 44.1k forks, and 211k stars. The main branch is 'main' with 94 branches and 141 tags. The repository description is 'The library for web and native user interfaces'. The 'About' section lists links to react.dev, the README, MIT license, Code of conduct, and Security policy. The 'Releases' section shows the latest release is 18.2.0 (June 14, 2022). The file list includes .circleci, .codesandbox, .github, fixtures, packages, scripts, .editorconfig, .eslintignore, .eslintrc.js, .gitattributes, .gitignore, .mailmap, and .nvmrc.

File	Description	Time
.circleci	Remove temporary CircleCI workaround (#26855)	2 months ago
.codesandbox	Codesandbox: upgrade to Node.js 18 (#26330)	4 months ago
.github	Use content hash for react-native builds (#26734)	2 months ago
fixtures	rename SuspenseList export to unstable_SuspenseList (#27061)	2 weeks ago
packages	export dynamic disableSchedulerTimeoutInWorkLoop flag in www (#271...	13 hours ago
scripts	[Flight] Client and Server Reference Creation into Runtime (#27033)	2 weeks ago
.editorconfig	https link to editorconfig.org (#18421)	3 years ago
.eslintignore	Renamed packages/react-devtools-scheduling-profiler to packages/reac...	2 years ago
.eslintrc.js	[Flight] Add Serialization of Typed Arrays / ArrayBuffer / DataView (#...	3 weeks ago
.gitattributes	.gitattributes to ensure LF line endings when we should	10 years ago
.gitignore	Renamed packages/react-devtools-scheduling-profiler to packages/reac...	2 years ago
.mailmap	updates mailmap entries (#19824)	3 years ago
.nvmrc	Build: specify Node.js 16 as minimum for dev (#26343)	4 months ago



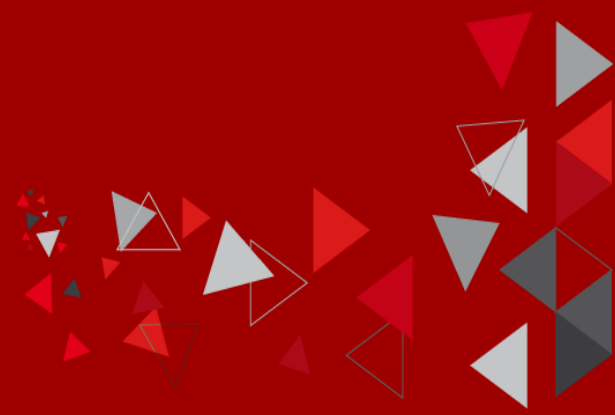
# ▶ React: Exemples d'applications



**NETFLIX**

**yahoo!**





# VDOM vs DOM

(Virtual Document Object Model vs Document Object Model)



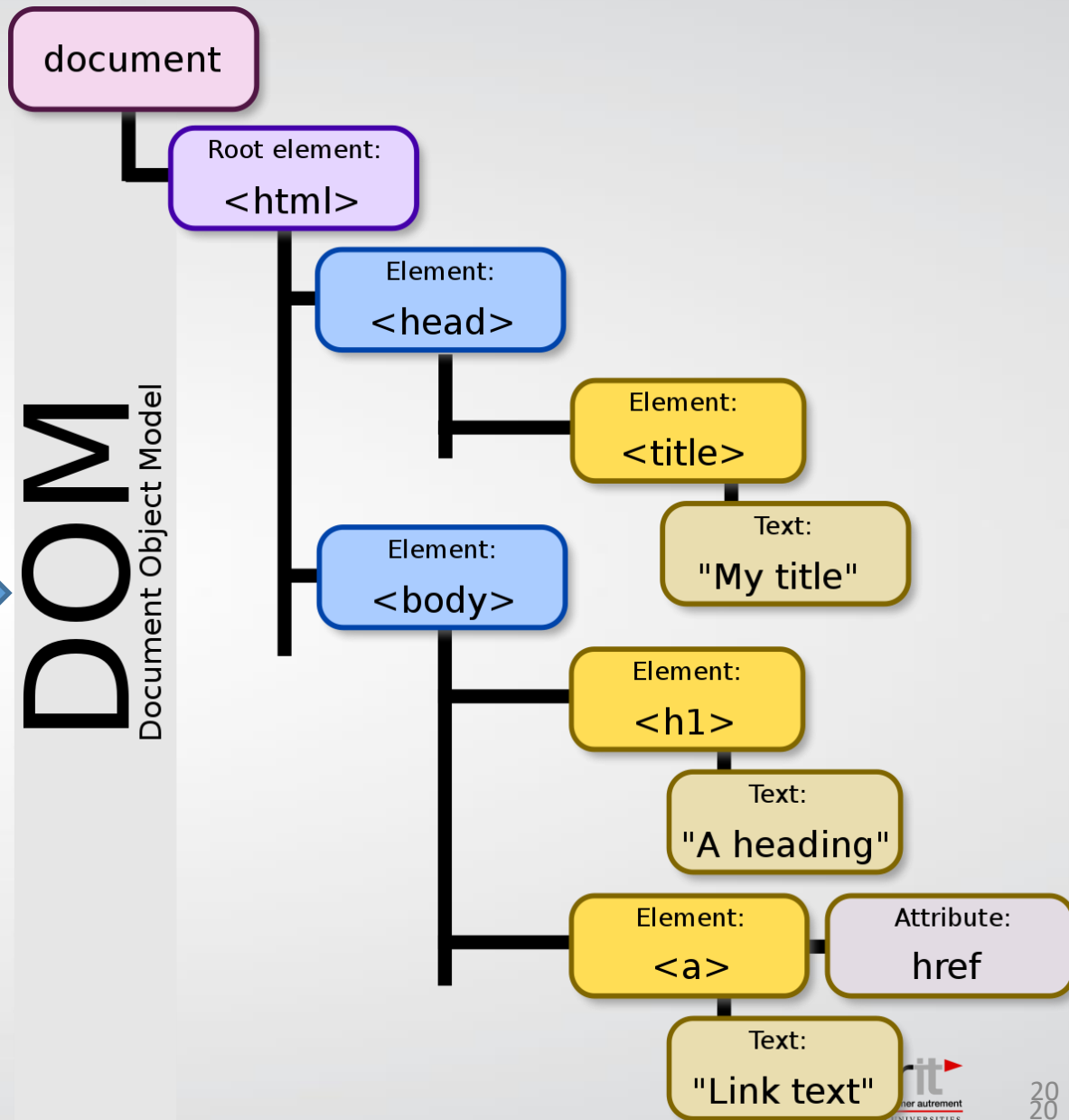
# DOM : Document Object Model

- Une structure arborescente créée par le navigateur.
- Facilite l'accès à la structure HTML.
- Le navigateur utilise le DOM pour appliquer le style et corriger les éléments.
- Le développeur utilise le DOM pour manipuler la page.
- Un DOM est un arbre constitué de nœuds.
- Chaque nœud est un objet pour lequel il existe des méthodes et des propriétés (lecture , modification, suppression)

# DOM: Document Object Model



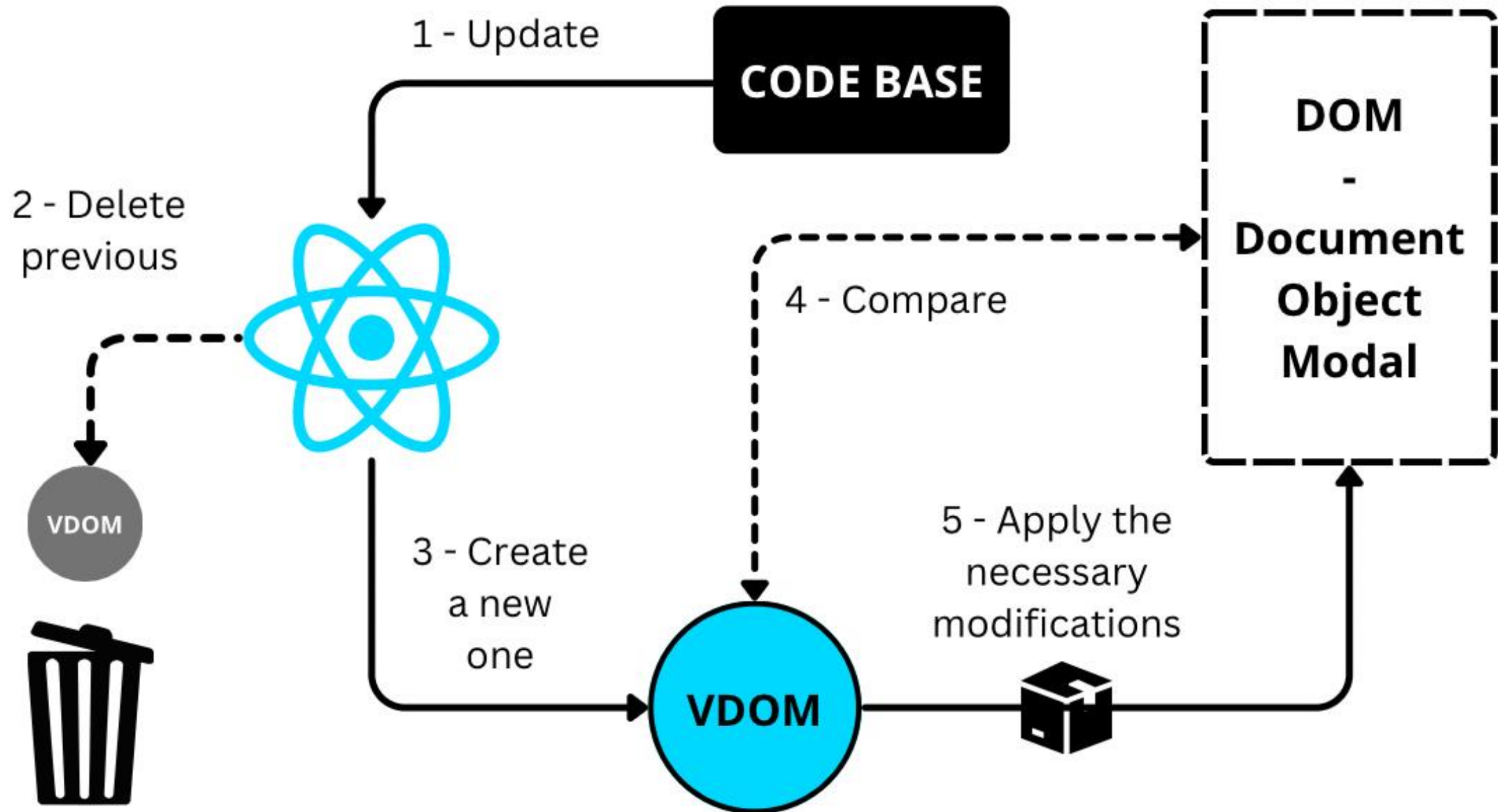
```
<html lang="en">
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="">Link text</a>
  </body>
</html>
```







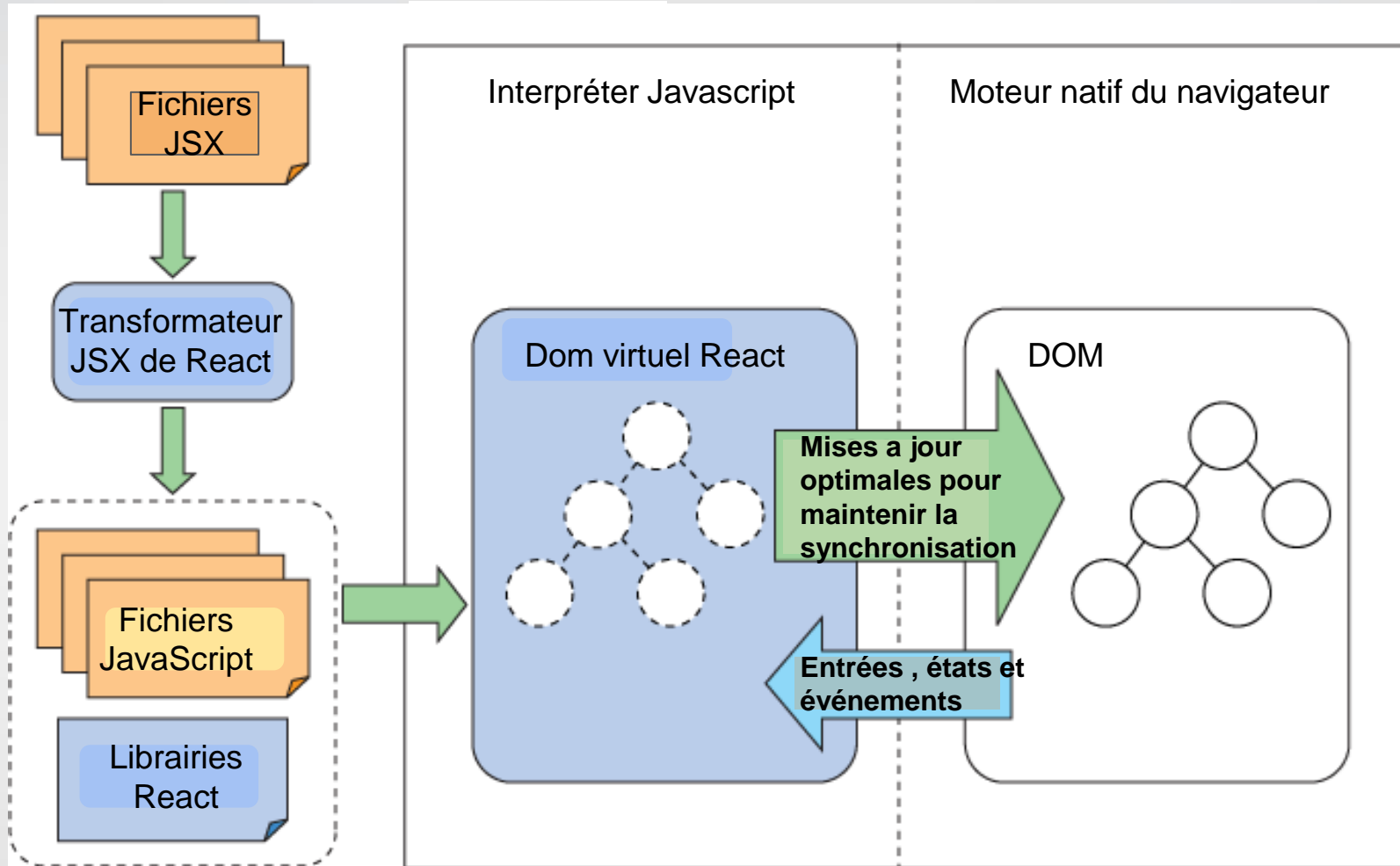
# VDOM : Virtual Document Object Model

- Le "Virtual DOM" (DOM virtuel) est une technique utilisée principalement dans les bibliothèques et les frameworks JavaScript
- L'idée derrière le Virtual DOM est d'améliorer les performances en minimisant le nombre de manipulations directes du DOM réel.
- Virtual DOM) est considéré comme une copie du DOM.
- Il possède toutes les propriétés que le véritable DOM, mais n'a pas la capacité d'écrire dans la page HTML.
- Le virtuel DOM gagne en vitesse et en efficacité du fait qu'il est léger.  
=> Un nouveau DOM virtuel est créé après chaque nouveau rendu.

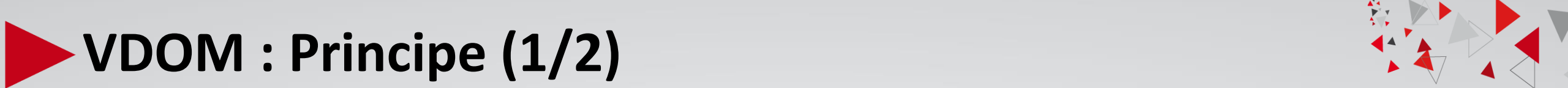


- 
- 
1. Quand l'application est mise à jour , React crée un nouvel arbre du VDOM avec toute les modifications.
  2. Il détecte les éléments ayant changé et ceux qui doivent être mise à jour en comparant le nouvel arbre VDOM avec le précédent en utilisant un algorithme de différenciation appelé "Reconciliation"
  3. À partir de cela, il génère un plan d'action qui détaille les mises à jour à appliquer sur le DOM.
  4. Enfin, React applique les mises à jour nécessaires au DOM réel en suivant le plan d'action généré avec React DOM.


# VDOM vs DOM: Schéma illustratif

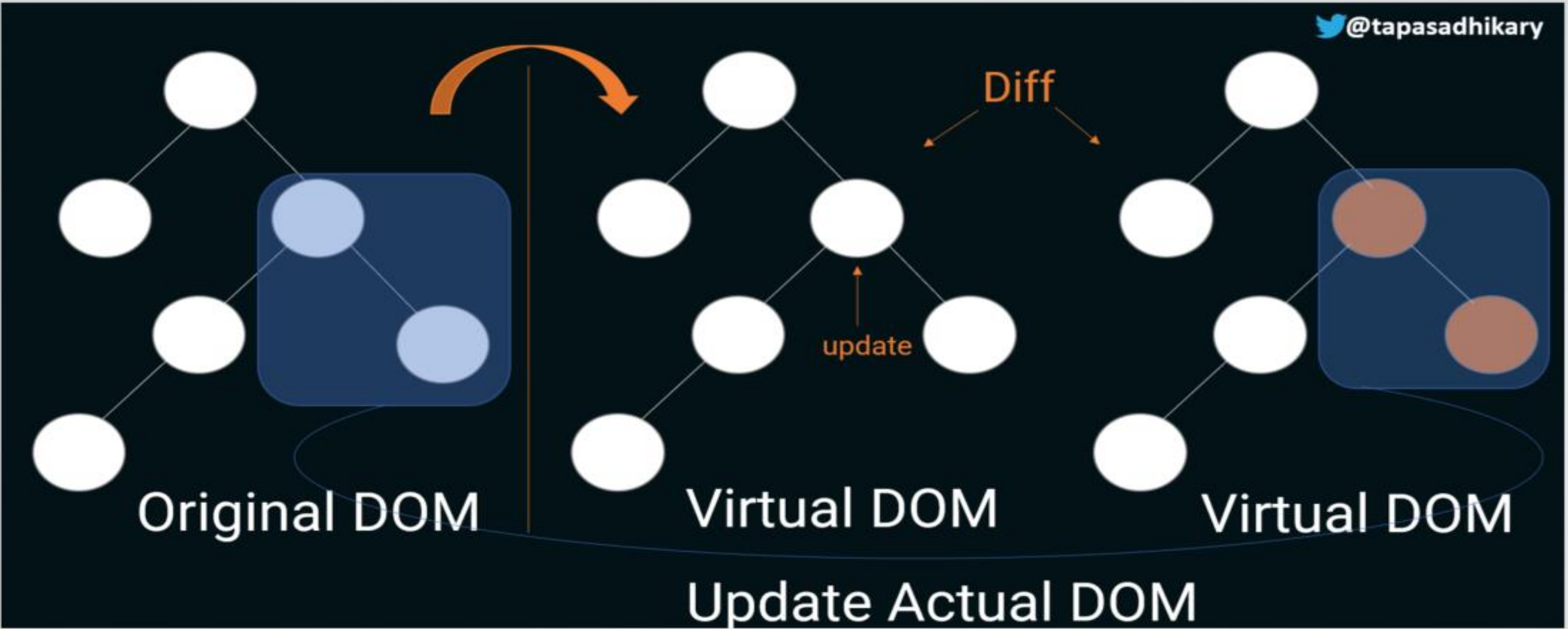






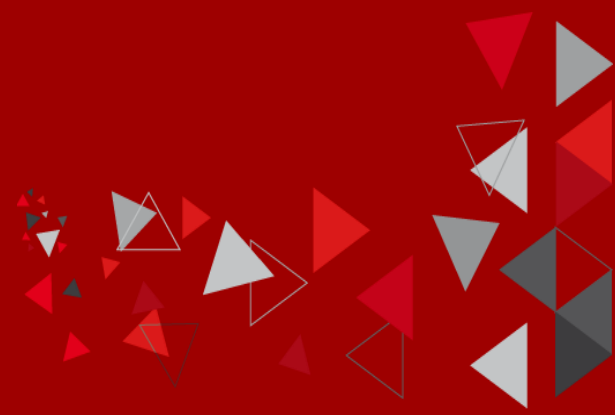
# VDOM : Principe (1/2)

 @tapasadhikary



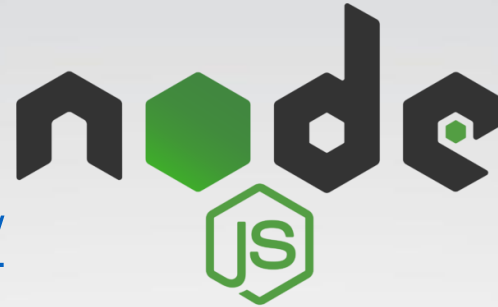
Les étapes de base de l'utilisation du Virtual DOM (VDOM) dans React sont les suivantes :

1. **Initialisation:** Lorsque React est chargé, il crée une copie de l'arbre de composants en mémoire, appelé Virtual DOM. Cela permet à React de gérer efficacement les mises à jour de l'interface utilisateur sans avoir à manipuler directement le DOM réel.
2. **Mise à jour de l'état:** Lorsque l'état d'un composant est mis à jour, React crée une nouvelle version de l'arbre de composants en mémoire, reflétant les changements de l'état.
3. **Comparaison:** React utilise un algorithme de différenciation pour comparer l'ancien VDOM avec le nouveau. Cela permet à React de déterminer quelles parties de l'arbre de composants ont effectivement changé.
4. **Mise à jour du DOM:** React ne met à jour que les parties du DOM réel qui ont effectivement été modifiées, en utilisant les informations fournies par l'algorithme de différenciation. Cela permet à React de limiter les manipulations inutiles sur le DOM et d'améliorer les performances de l'application.



# Environnement de travail & installation

# ► Environnement de travail & installation (1/2)



## ❑ Node JS

Site : <https://nodejs.org/en/download/>

- Node JS installe l'outil **npm** (Node Package Manager) qui permet de télécharger et installer des bibliothèques JavaScript.
- Il existe d'autres gestionnaires de modules:
  - npx: Node Package Execute
  - yarn: Yet Another Resource Negotiator



- Pour vérifier les versions installées :

> **node --version** ou **node -v** → (version de node)

> **npm --version** ou **npm -v** → (version de npm)

> **npm view react version** → (version de React)

# ► Environnement de travail & installation (2/2)



## ❑ Éditeur de texte

- Visual Studio Code (<https://code.visualstudio.com>)



## ❑ Extensions

- Installation des plugins (extensions) associés est souhaitable afin de faciliter le codage.  
(**Exemple** : Simple React Snippets sous VSC : imr, imrc, cc , ccc, ffc, ...)

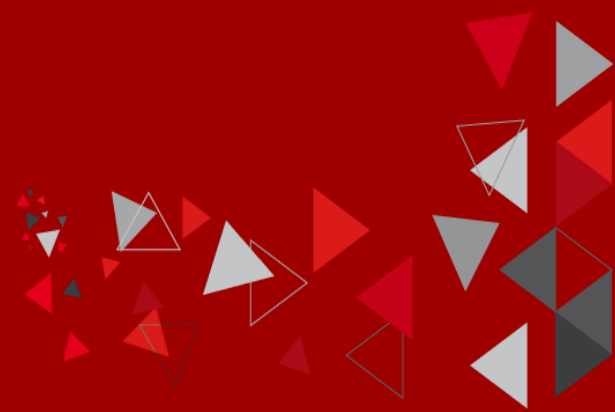
## ❑ React DevTools :



Extension sur chrome et Firefox qui permet d'examiner :

- Une arborescence de composants React dans les outils de développement du navigateur.
- Les props et l'état local des composants React

# NPM vs NPX / Yarn



# ▶ NPM vs NPX



❑ **npm** est l'acronyme de **Node Package Manager**

- NPM est utilisé pour installer les packages Node.js afin de les utiliser dans notre application.
- Permet des installations locales et globales :
  - **Localement** : Lorsqu'un package est installé localement, il est installé dans ./node\_modules/.bin/ du répertoire local du projet.
  - **Globalement** : un package global est installé dans le chemin de l'environnement utilisateur /usr/local/bin pour Linux et AppData%/npm pour Windows.



- ❑ **npx** est l'acronyme de **Node Package Execute**

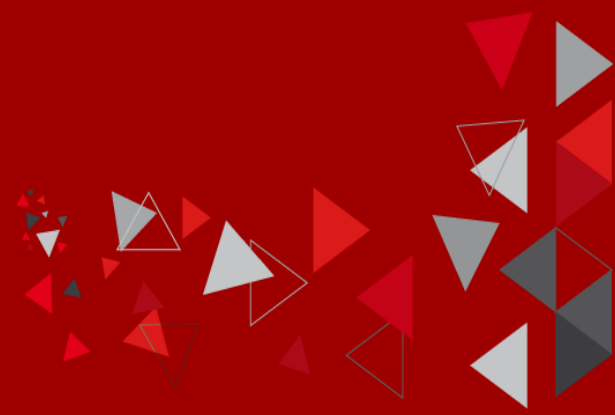
- Permet l'exécution de paquet même s' il n'existent pas localement ou globalement.
- NPX agit comme NPM, mais nous permet en plus
  - d'éviter la gestion des versions,
  - les problèmes de dépendance,
  - l'installation de packages non pertinents que nous voulons simplement vérifier.





- ❑ **yarn** est l'acronyme de **Yet Another Resource Negotiator**
  - Yarn installe simultanément les packages, c'est pourquoi Yarn est plus rapide que NPM..
  - A noter que YARN doit être installé manuellement à partir de NPM.





# Création d'un projet REACT

# ▶ Création d'un projet REACT (1/5)



- ❑ **Vite** est un outil qui se focalise sur la vitesse et la performance.
  - Vite se compose d'un serveur de développement offrant divers améliorations: Hot Module replacement, pre-bundling, prise en charge de typescript et importation dynamique.
  - Vite utilise Rollup lors du lancement de la commande qui regroupe le code (bundle).
  - Vite est préconfiguré pour gérer les assets en mode production.



**Vite**

Next Generation Frontend Tooling

# ▶ Création d'un projet REACT (2/5)



- ❑ Pour créer une nouvelle application en utilisant **Vite**

- **yarn create vite**

```
✓ Project name: ... vite-project-yarn
? Select a framework: » - Use arrow-keys. Return to submit.
> Vanilla
  Vue
  React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

```
? Select a variant: » - Use arrow-keys. Return to submit.
> TypeScript
  TypeScript + SWC
  JavaScript
  JavaScript + SWC
```

- SWC (Speedy Web Compiler) est un compilateur JavaScript/TypeScript. Le but principal de SWC est d'accélérer le processus de compilation du code JavaScript/TypeScript, ce qui peut être particulièrement utile dans des projets de grande envergure.

# ▶ Création d'un projet REACT (3/5)



- ❑ Pour créer une nouvelle application en utilisant **Vite**
  - **yarn create vite**

```
✓ Project name: ... vite-project-yarn
✓ Select a framework: » React
✓ Select a variant: » JavaScript
```

```
Scaffolding project in D:\
```

```
\vite-project-yarn ...
```

```
Done. Now run:
```

```
cd vite-project-yarn
yarn
yarn dev
```

```
Done in 141.71s.
```

# ▶ Création d'un projet REACT (4/5)



- ❑ Alternative 1 : Pour créer une nouvelle application en utilisant **Vite**
  - `npm create vite@latest`

```
npm create vite@latest
Need to install the following packages:
  create-vite@4.4.0
Ok to proceed? (y) y
✓ Project name: ... vite-project-npm
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in D:\Esprit\2022-2023\React\Cours 1.0\vite-project-npm

Done. Now run:

  cd vite-project-npm
  npm install
  npm run dev
```

# ▶ Création d'un projet REACT (5/5)



## ❑ Alternative 2 : Installer Vite

- `npm install -g create-vite`
- `create-vite nom-du-projet --template react` → Les -- sont nécessaires pour les versions de npm > à 7.

ou

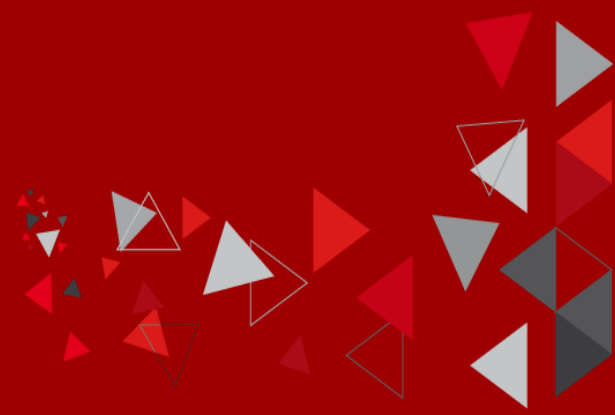
- `yarn global add create-vite`
- `yarn create vite react-yarn -- template react`

```
npm create vite@latest my-react-app -- --template react
```

```
Scaffolding project in D:\                                \my-react-app ...
```

```
Done. Now run:
```

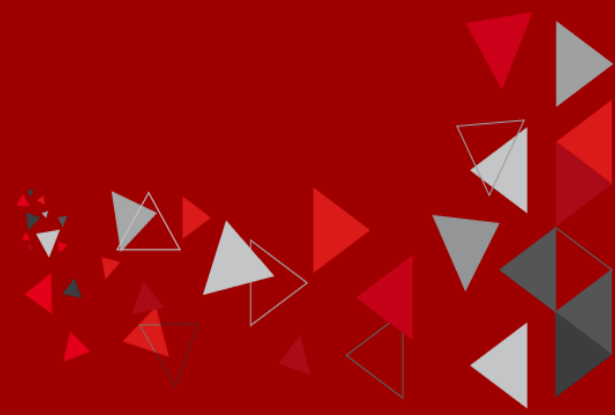
```
cd my-react-app
npm install
npm run dev
```



# **Atelier 0 :**

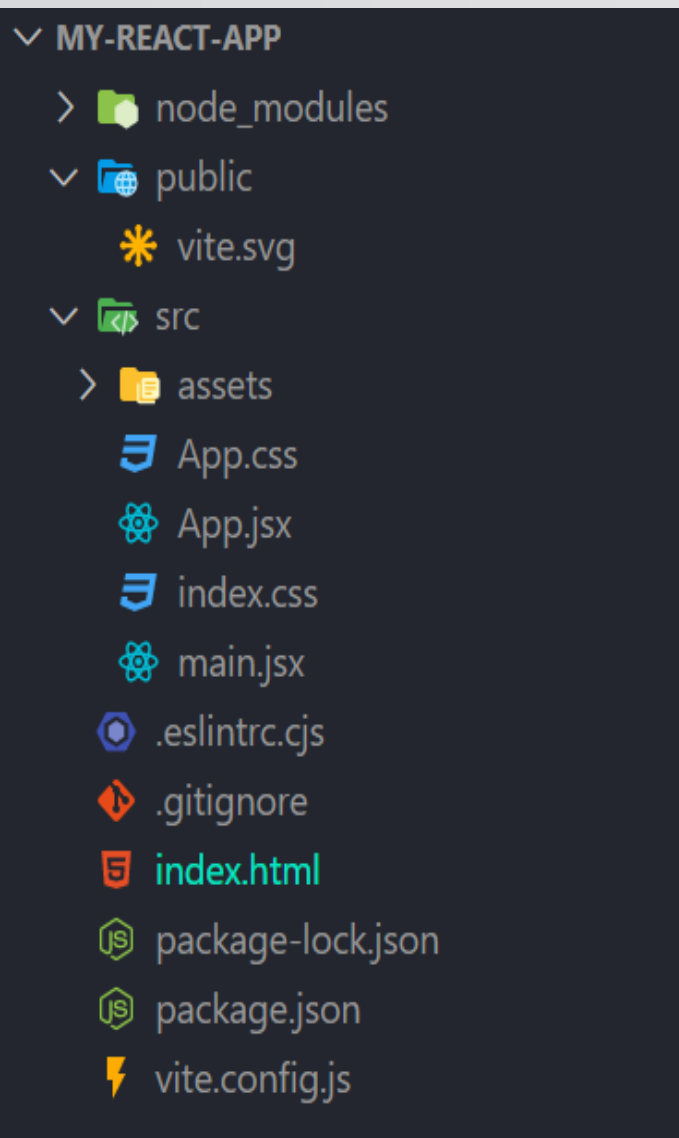
## ***Installation et configuration d'une application React***





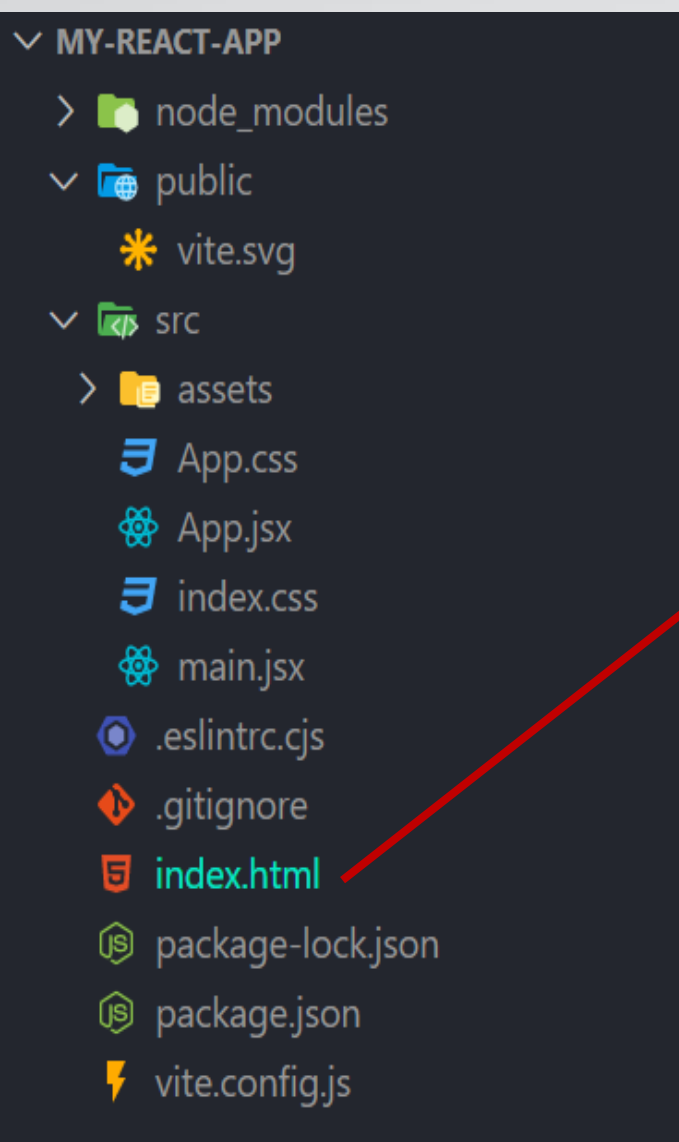
# Structure d'un projet React

# ► Structure d'un projet React (1/7)



- ❑ **Nodes\_modules:** contient toutes les bibliothèques JavaScript externes utilisées par l'application.
- ❑ **public:** contient les fichiers publics du projet.
- ❑ **src:.**
- ❑ **index.html:** est le point d'entrée contient le code de l'application de l'application
- ❑ **package.json:** est le fichier de configuration qui contient les métadonnées du projet ainsi que toutes les dépendances nécessaires pour créer et exécuter l'application.
- ❑ **vite.config.js:** contient les configurations du projet: plugins, serveur...

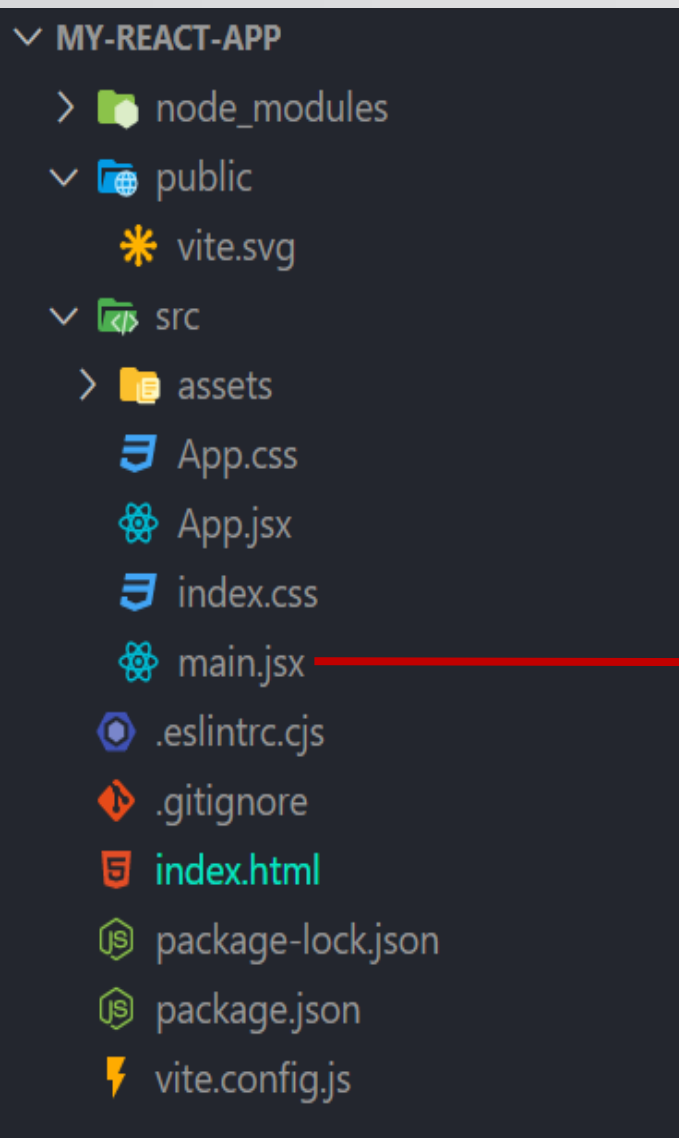
# ► Structure d'un projet React (2/7)



## Index.html

```
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>Vite + React</title>
8    </head>
9    <body>
10     <div id="root"></div>
11     <script type="module" src="/src/main.jsx"></script>
12   </body>
13 </html>
14
```

# ► Structure d'un projet React (3/7)



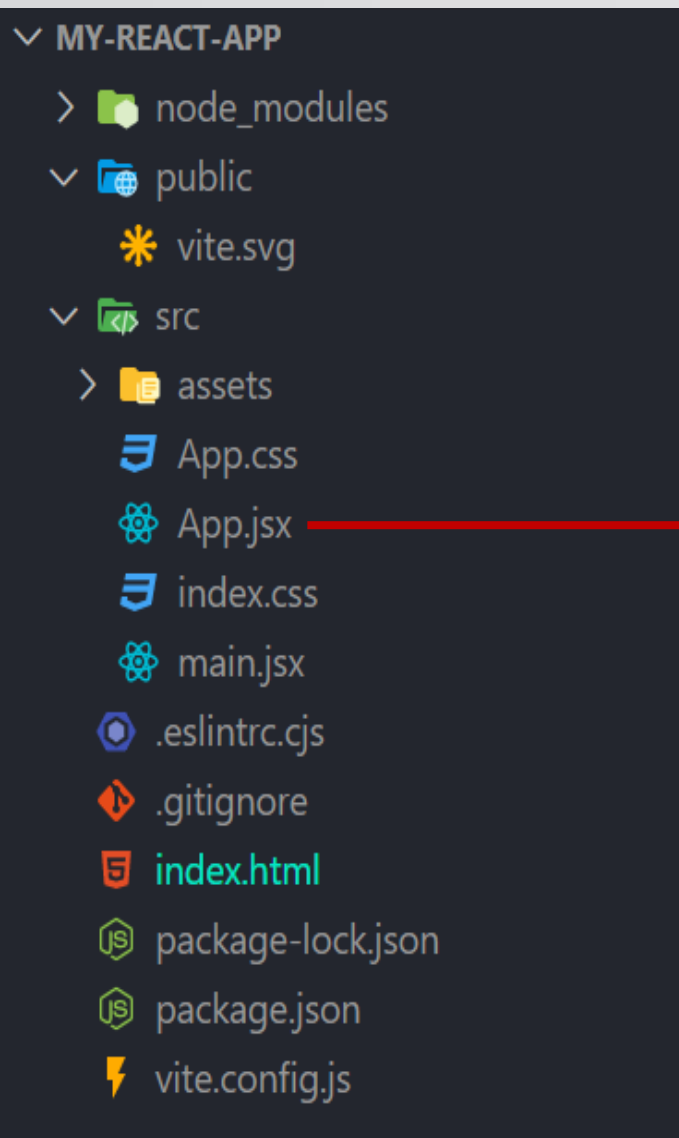
Index.html

```
9   <body>
10   <div id="root"></div>
11   <script type="module" src="/src/main.jsx"></script>
12 </body>
```

(2)

```
src > main.jsx
1   import React from 'react'
2   import ReactDOM from 'react-dom/client'
3   import App from './App.jsx'
4   import './index.css'
5
6   ReactDOM.createRoot(document.getElementById('root')).render(
7     <React.StrictMode>
8       <App />
9     </React.StrictMode>,
10 )
11
```

# ► Structure d'un projet React (4/7)



main.jsx

```
5
6   ReactDOM.createRoot(document.getElementById('root')).render(
7     <React.StrictMode>
8       <App />
9     </React.StrictMode>,
10  )
```

(3)

```
src > App.jsx
1   import { useState } from 'react'
2   import reactLogo from './assets/react.svg'
3   import viteLogo from '/vite.svg'
4   import './App.css'
5
6   function App() {
7     const [count, setCount] = useState(0)
8
9     return (
10    >   <div>...
31    </div>
32  )
33  }
34
35  export default App
36
```

# ► Structure d'un projet React (5/7)



## Index.html

```
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>Vite + React</title>
8    </head>
9    <body>
10     <div id="root"></div>
11     <script type="module" src="/src/main.jsx"></script>
12   </body>
13 </html>
14
```

- ❑ **<div id="root">** : est un élément HTML qui est utilisé dans les applications React. Il est le point d'entrée de l'application et sert de conteneur pour tous les autres éléments de l'interface utilisateur. Toutes les composantes React sont affichées à l'intérieur de ce div, ce qui permet aux développeurs de contrôler le rendu de l'application.

# ► Structure d'un projet React (6/7)

## main.jsx



main.jsx

src > main.jsx

```
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.jsx'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById('root')).render(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10  )
```

❑ **StrictMode** :est un outil pour mettre en évidence des problèmes potentiels dans une application.

- Ne rend aucune interface utilisateur visible.
- Il active des contrôles et des avertissements supplémentaires pour ses descendants.

# ► Structure d'un projet React (7/7)

## App.jsx



```
App.jsx ×  
src > App.jsx > ...  
1  import { useState } from 'react'  
2  import reactLogo from './assets/react.svg'  
3  import viteLogo from '/vite.svg'  
4  import './App.css'  
5  
6  function App() {  
7    const [count, setCount] = useState(0)  
8  
9    return (  
10 >    <...  
31    </>  
32  )  
33 }  
34  
35 export default App  
36
```



# Lancement du projet

- Pour lancer le projet :
  - **npm install**
  - **npm run dev**

```
C:\Windows\system32\cmd.exe

VITE v5.0.12  ready in 1243 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h + enter to show help
```





# Références



- ❑ [Installation | Yarn \(yarnpkg.com\)](https://yarnpkg.com)
- ❑ <https://fr.reactjs.org/docs/getting-started.html>
- ❑ <https://blog.greenroots.info/reactjs-virtual-dom-and-reconciliation-explain-like-im-five>