



Introduction à React JS

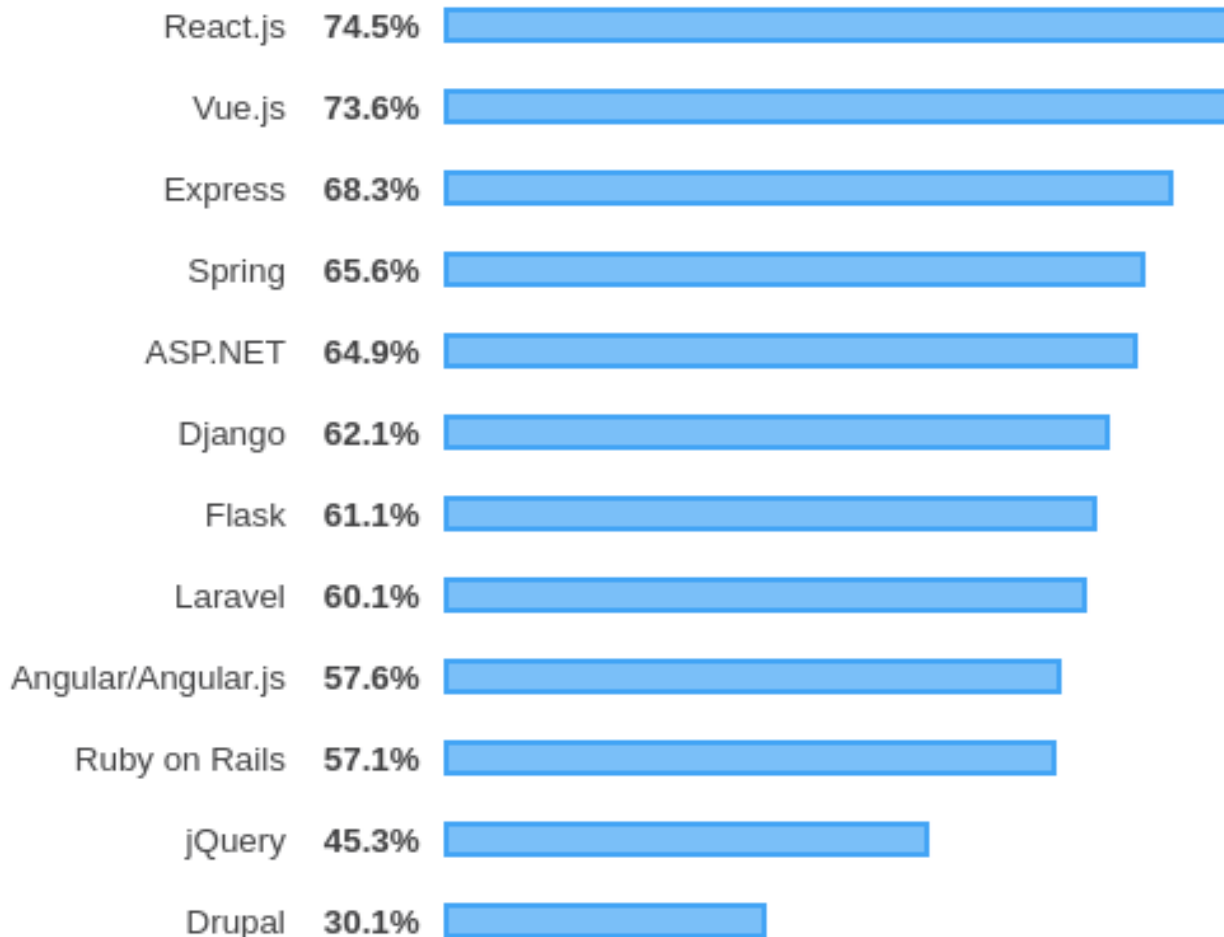
Qu'est qu'un framework ?

- Une librairie est une collection de fonctionnalités écrites dans un langage pour permettre la réutilisation de code déjà écrit par d'autres développeurs.
- Un framework est une collection de plusieurs librairies, une architecture minimale de projet et un ensemble de scripts pour vous simplifier le développement.

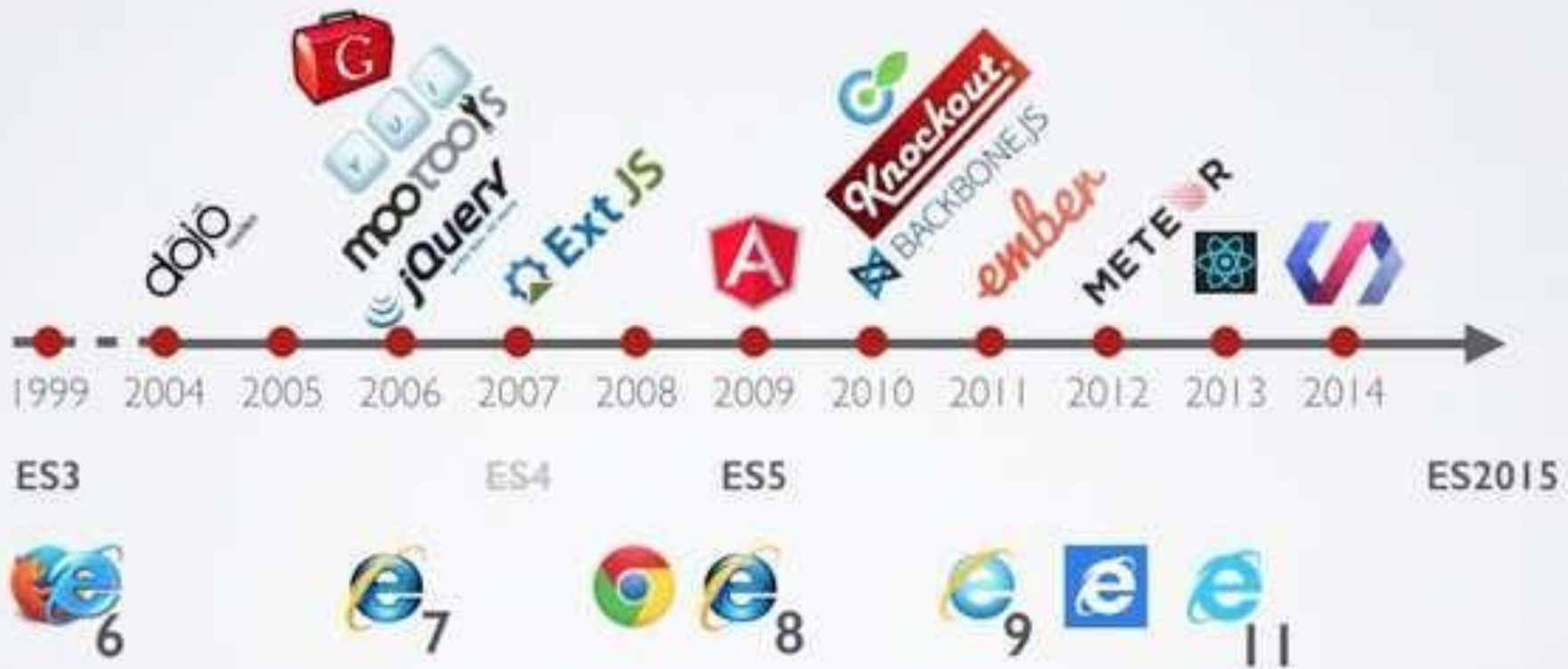
Framework Web

- Les frameworks sont devenus une partie essentielle du développement Web, car les normes des applications Web sont en constante évolution.
- C'est pourquoi l'utilisation de frameworks approuvés par des milliers de développeurs à travers le monde est une approche très judicieuse pour créer des applications Web riches et interactives.

Classement des frameworks web



Evolution des Frameworks javaScript



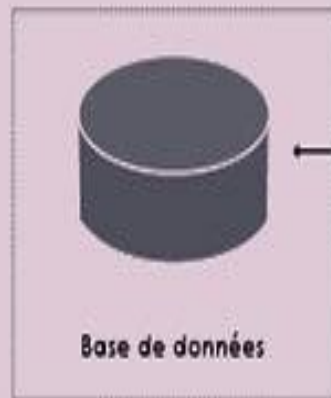
Frameworks Back-end et Front-end

- Toute application Web se compose d'une partie back end ou côté serveur et d'une partie front end.
- Par conséquent, il existe des Frameworks de développement web front-end et back-end.
- Les Frameworks web front-end sont responsables de l'interface utilisateur, c'est-à-dire de la partie visuelle d'un site web ou d'une application avec laquelle les utilisateurs finaux interagissent.
- Ils sont basés sur des langages de programmation front-end tels que HTML, CSS et JavaScript.
- Les Frameworks web back-end sont responsables de la partie cachée d'un site web ou d'une application avec laquelle les développeurs interagissent.
- Ils traitent du fonctionnement du serveur et de la base de données, de la logique et de l'architecture de la solution.
- Ces plateformes sont basées sur des langages de programmation back-end tels que .NET, Ruby, Python, Java et PHP.

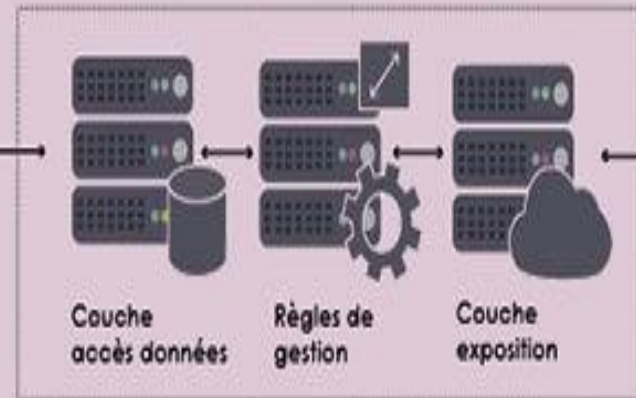
Front-end vs. Back-end

Environnement BACK-END

Couche données



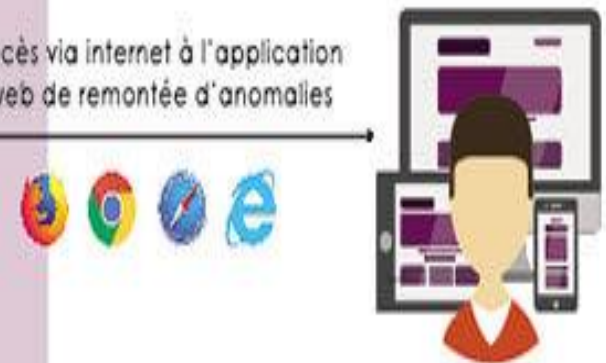
Couche métier



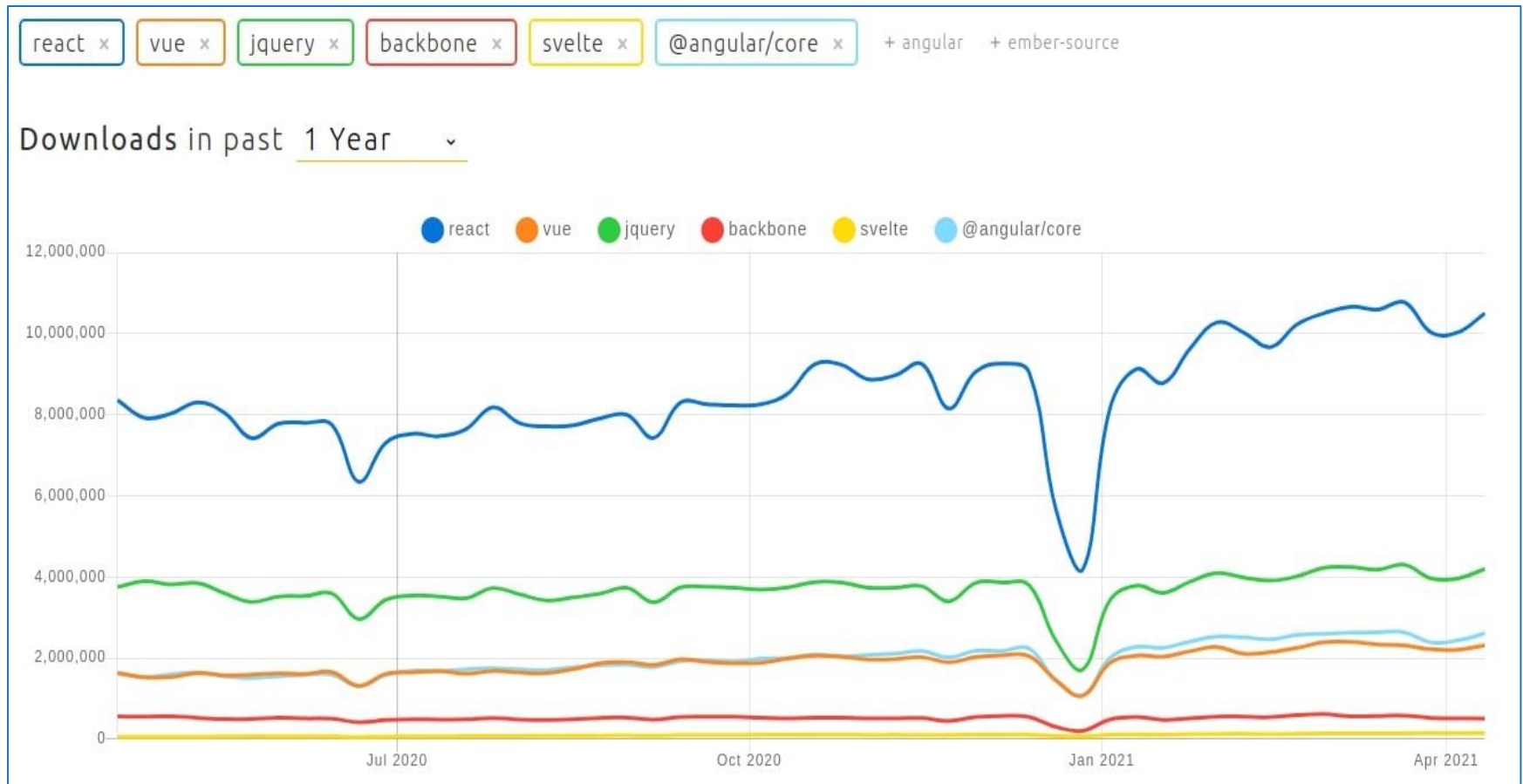
Environnement FRONT-END

Couche de présentation (IHM)

Accès via internet à l'application web de remontée d'anomalies



Classement Frameworks front-end

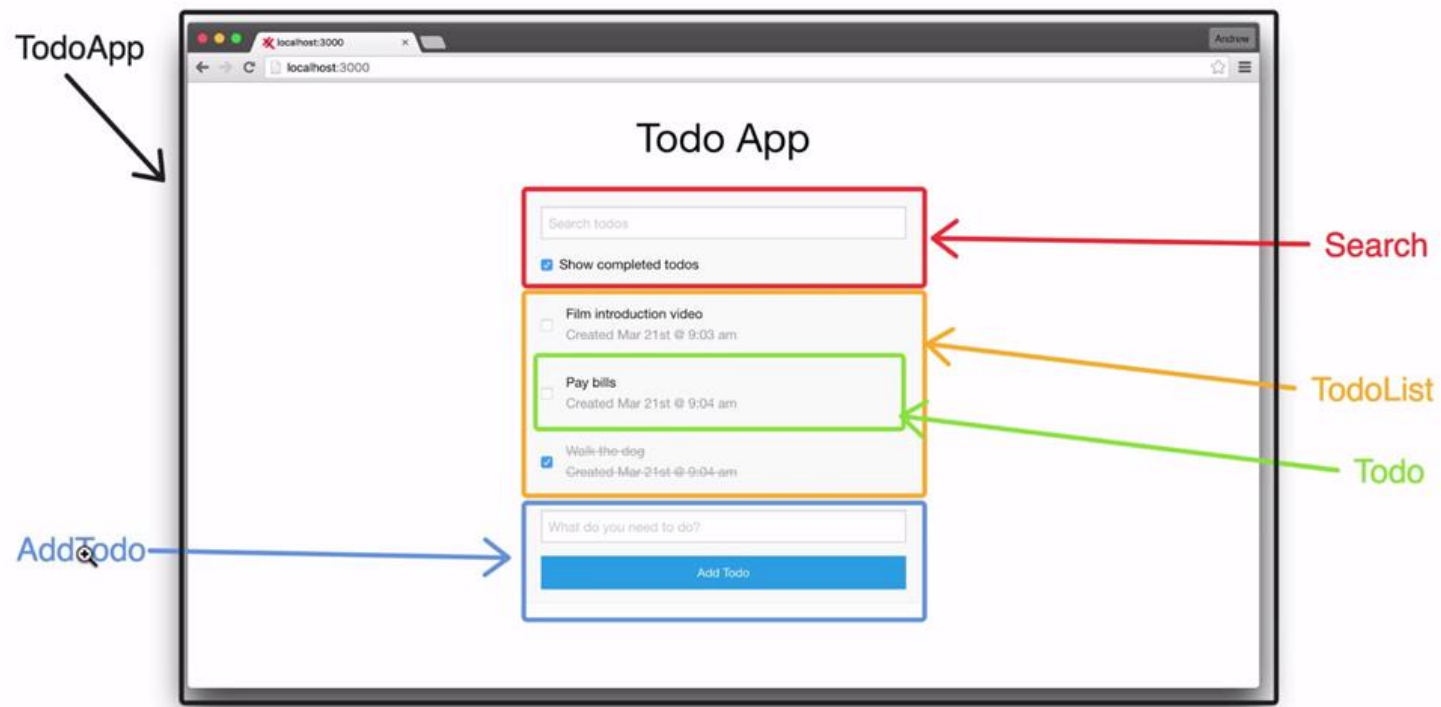


React : présentation générale

- Librairie JavaScript pour construire et gérer des interfaces utilisateurs (UI).
- Développée et pilotée par Facebook depuis 2013.
- Projet open-source, désormais distribué sous la licence MIT.
- React se concentre principalement sur la gestion de l'UI.
- Les autres couches applicatives (routage côté client, le stockage des données, etc.) sont laissées aux solutions complémentaires de son écosystème (Ex : React-Router, Reduc, Redux-offline, etc.).
- Devenue très populaire : La plupart des « gros sites » connus du web ont migré leurs interfaces web et leurs applications mobiles vers React et React Native : Atlassian, Dailymotion, Dropbox, Instagram, Netflix, Paypal, Twitter, Wordpress, Yahoo, etc.

L'approche Web Components

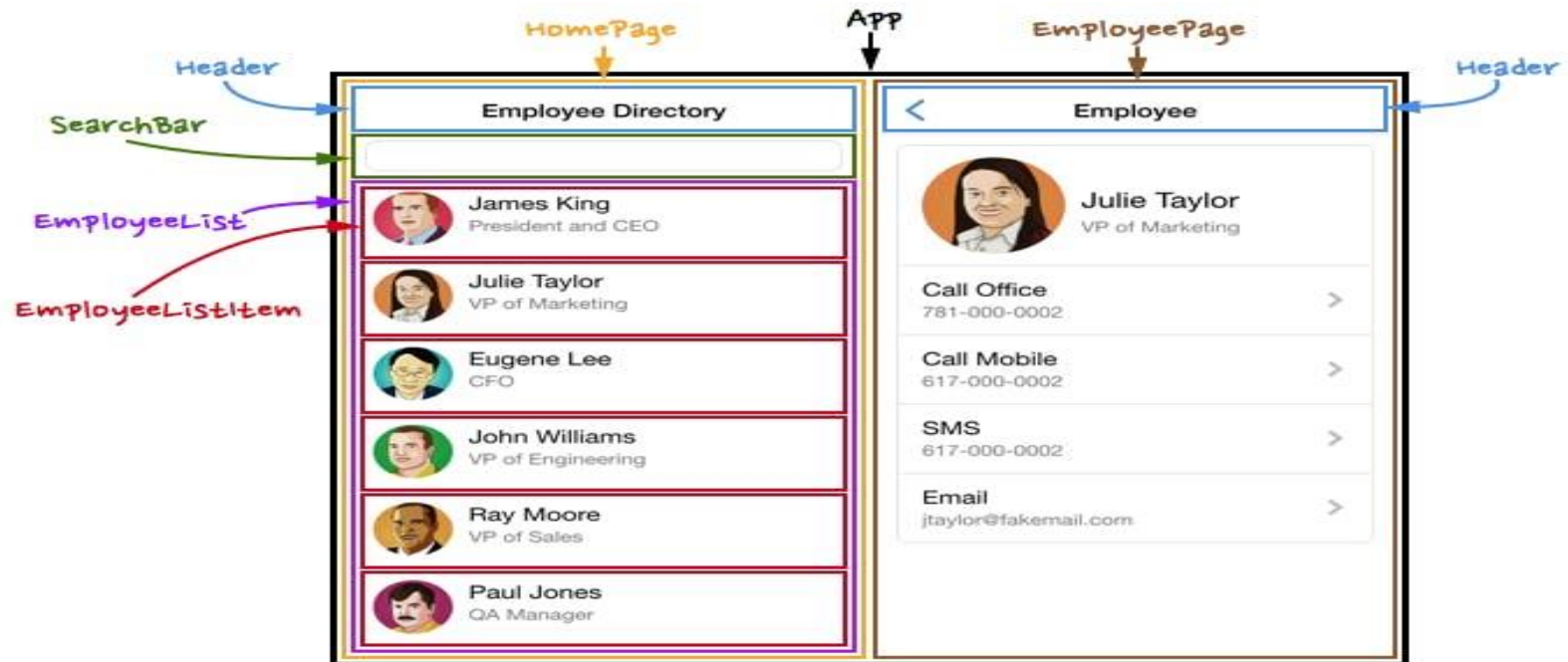
- React est basé sur l'approche Web Components qui considère une page web comme un ensemble de composants, c'est à dire de parties unitaires que l'on va pouvoir facilement exploiter, réutiliser et tester.



Composants réutilisables

11

- React permet aux développeur de créer des Components (Composants) correspondant aux parties de l'interface.
- Les composants peuvent être réutilisés ou combinés avec autres composants afin de créer une interface complète.



Environnement du travail et Installation

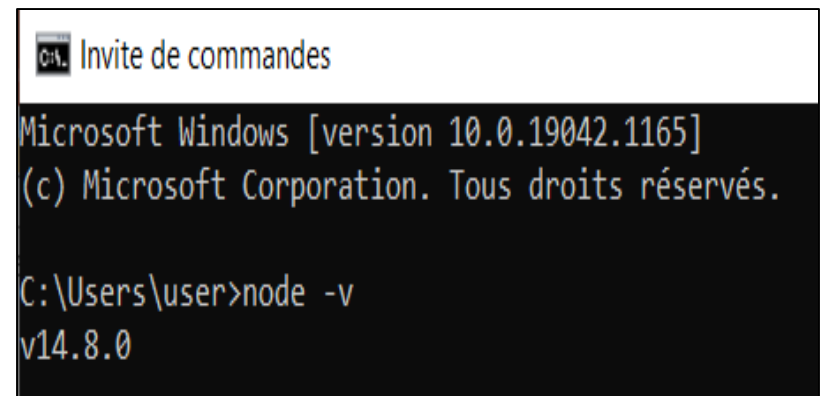
- Node JS installe l'outil npm (Node Package Manager) qui permet de télécharger et installer des bibliothèques JavaScript : gestionnaire de modules de Node.

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

- Puis sélectionner l'installateur Windows (.msi) et exécuter le fichier msi téléchargé. Enfin accepter toutes les étapes par défaut de l'installation

- Pour vérifier la version installée :

`node --version` ou `node -v`



```
Invite de commandes
Microsoft Windows [version 10.0.19042.1165]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\user>node -v
v14.8.0
```

- Nous aurons besoin de Node ≥ 8.10 et de npm ≥ 5.6 sur notre machine pour créer une application React.

Installation des outils

- Editeur :

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



ou encore IntelliJ, WebStorm, PHP Storm, etc.

- Installation des plugins (extensions) y associés est souhaitable afin de faciliter le codage :

Exemple : Simple React Snippets sous VSC



Simple React Snippets

Dead simple React snippets you will actually use

Burke Holland



EXTENSIONS: MARKETPLACE



React snippets



ES7 React/Redux/GraphQL/React-Native snippets

3.4M ★ 4.5

Simple extensions for React, Redux and Graphql in JS/TS with ES7 syntax
dsznajder

Install



React snippets

26K ★ 5

A set of snippets for React
runningcoder

Install

Extensions (Ctrl+Shift+X)

Snippets



Dead simple React snippets you will actually use
Burke Holland



snippets

ReactJS snippets created by GoBystrok.
GoBystrok ReactJS



React Snippets

17K ★ 4.5

Code snippets for React
NicholasHsiang

Install



React Snippets

11K

ReactJS, Redux and React Router code snippets with hooks support.
Ross

Install



React Snippets

2K ★ 5

react snippets



Simple React Snippets v1.2.3

Burke Holland | 1 124 962 | ★★★★★ (21)

Dead simple React snippets you will actually use

Uninstall ▼ ⚙️

⚠️ This extension has been disabled because the current workspace is not trusted.

Details Feature Contributions Changelog

Snippet	Renders
<code>imr</code>	Import React
<code>imrc</code>	Import React / Component
<code>imrs</code>	Import React / useState
<code>imrse</code>	Import React / useState useEffect
<code>impt</code>	Import PropTypes
<code>impc</code>	Import React / PureComponent
<code>cc</code>	Class Component
<code>ccc</code>	Class Component With Constructor
<code>cpc</code>	Class Pure Component
<code>sfc</code>	Stateless Function Component
<code>cdm</code>	componentDidMount

Create React App (CRA)

- ❑ Outil pour faciliter le développement d'applications web fondées sur React.
- ❑ Permet d'éviter les problèmes d'installation, de configuration et d'intégration.
- ❑ Permet la génération automatique d'un squelette applicatif.

Installation de React JS

```
C:\TPReact>npx create-react-app myfirstapp
```

- Cela va créer un dossier nommé myfirstapp et y installer tous les fichiers requis.
- npx est un outil destiné à compléter l'expérience d'utilisation des packages du registre npm. Il facilite l'utilisation des outils CLI et autres exécutables hébergés sur le registre.

Remarque :

Le nom de l'application doit être en minuscule

```
C:\TPReact>npx create-react-app myfirstapp
npx: installed 67 in 10.78s

Creating a new React app in C:\TPReact\myfirstapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[.....] \ fetchMetadata: sill resolveWithNewModule react-dom@17.0.2 checking installable status
```

Exécution de React JS

18

```
C:\TPReact>cd myfirstapp
```

```
C:\TPReact\myfirstapp>npm start
```

➡ L'application se lance par défaut en local sur <http://localhost:3000/>

```
Success! Created myfirstapp at C:\AppReact\myfirstapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

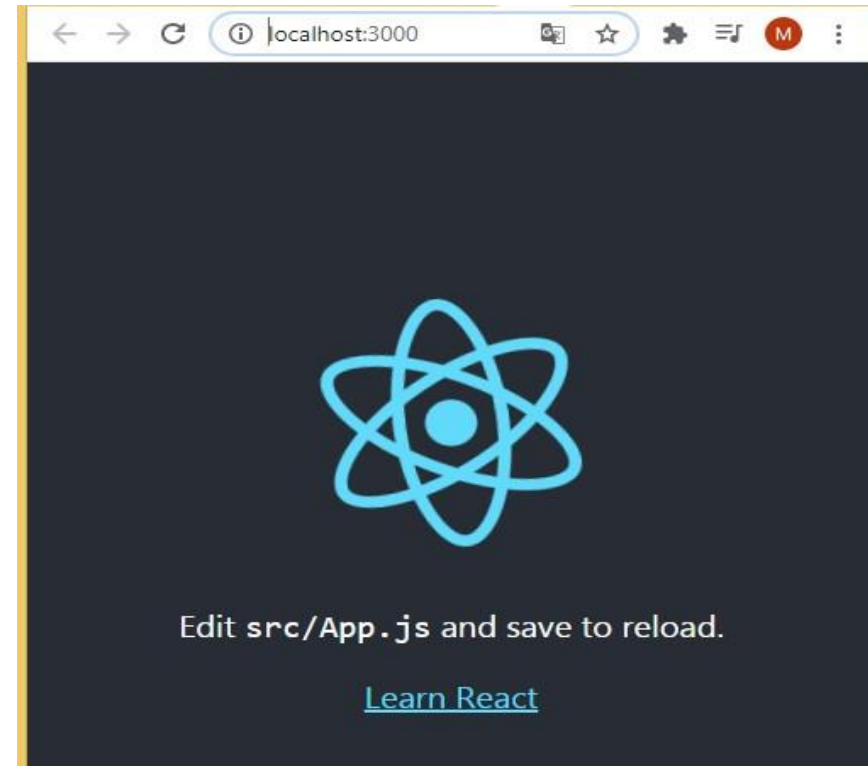
  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd myfirstapp
  npm start

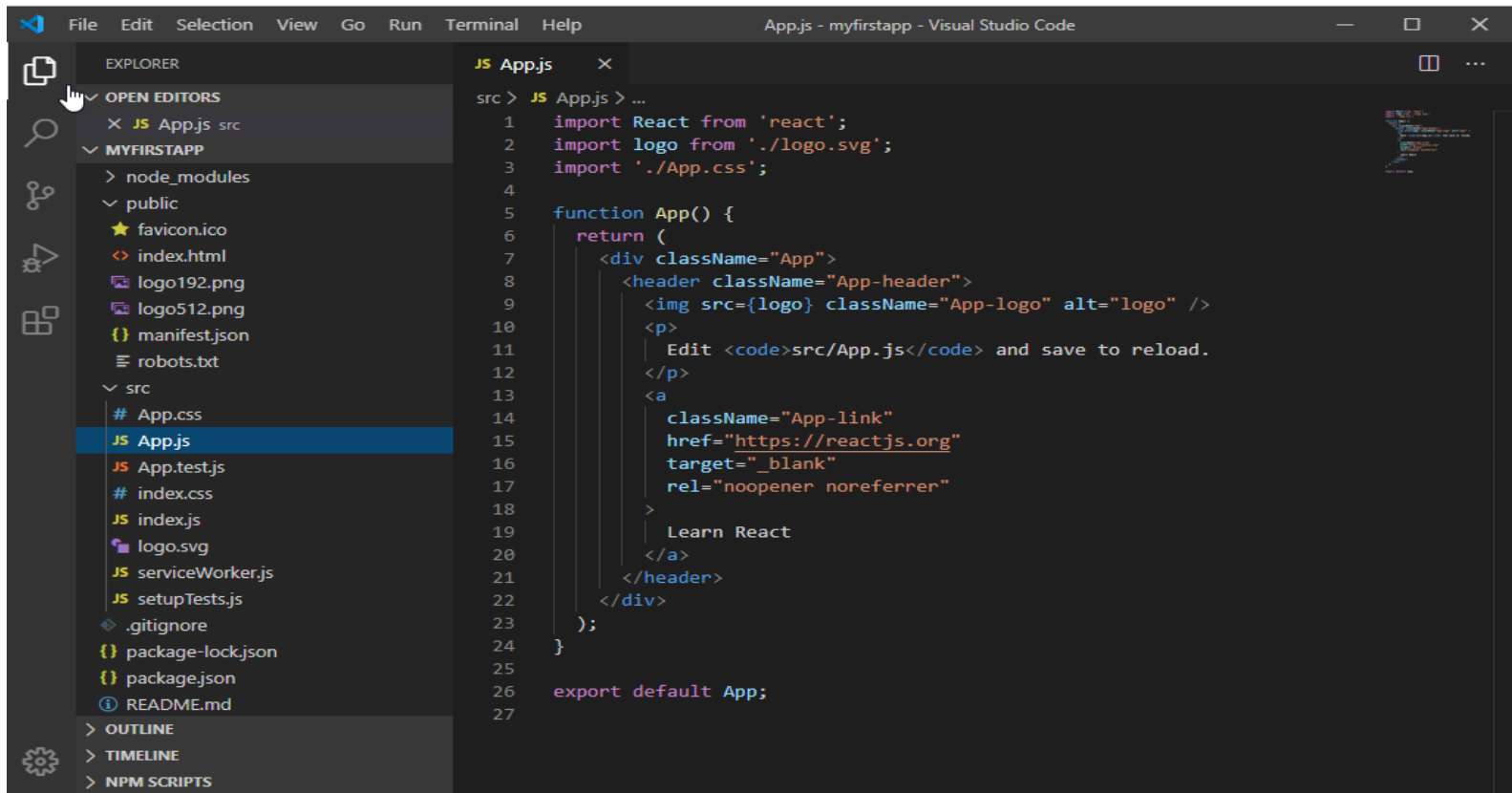
Happy hacking!
```



Ouvrir l'application React dans VSC

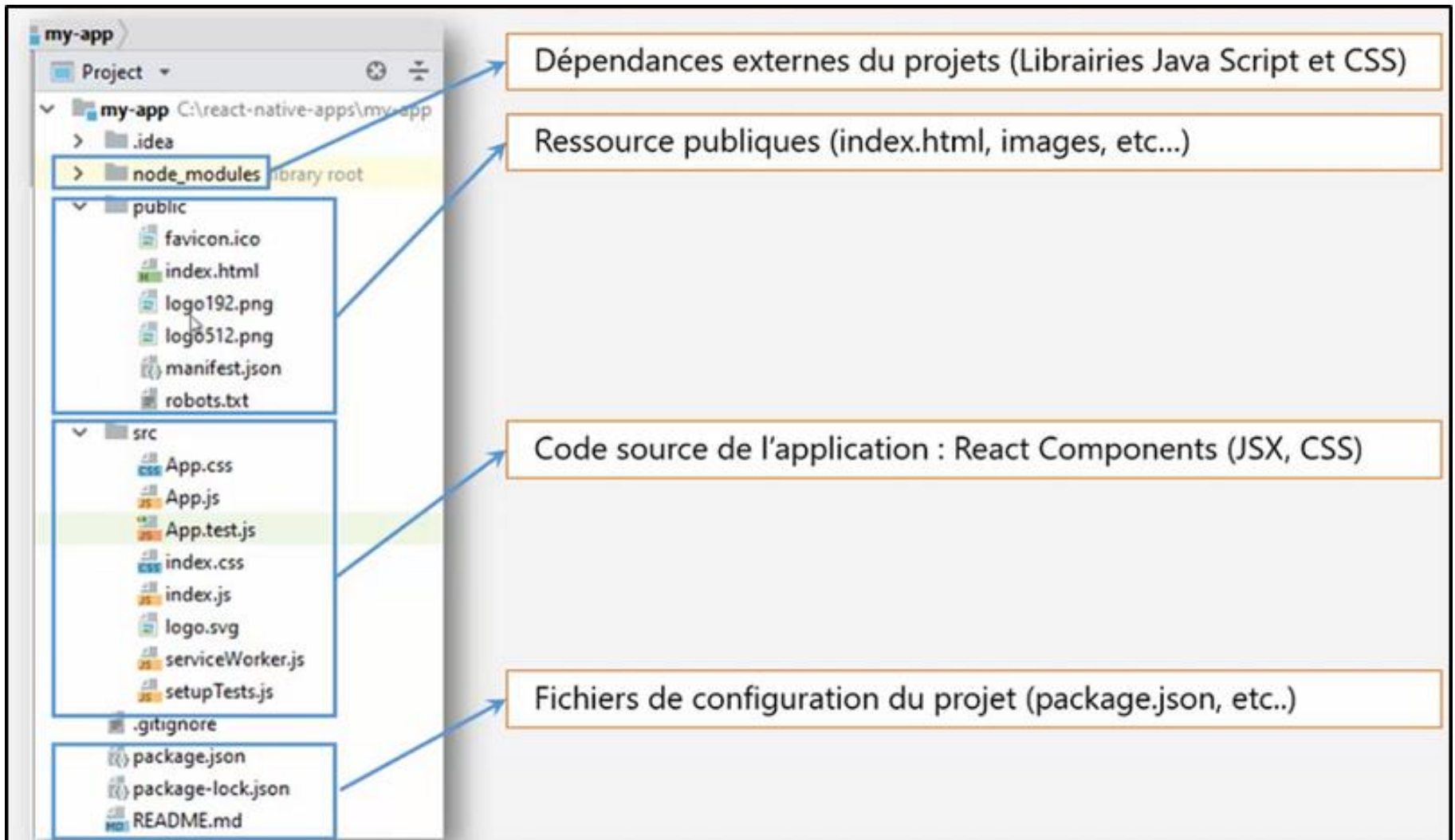
19

C:\AppRect\myfirstapp>code •



Structure d'un projet React

20



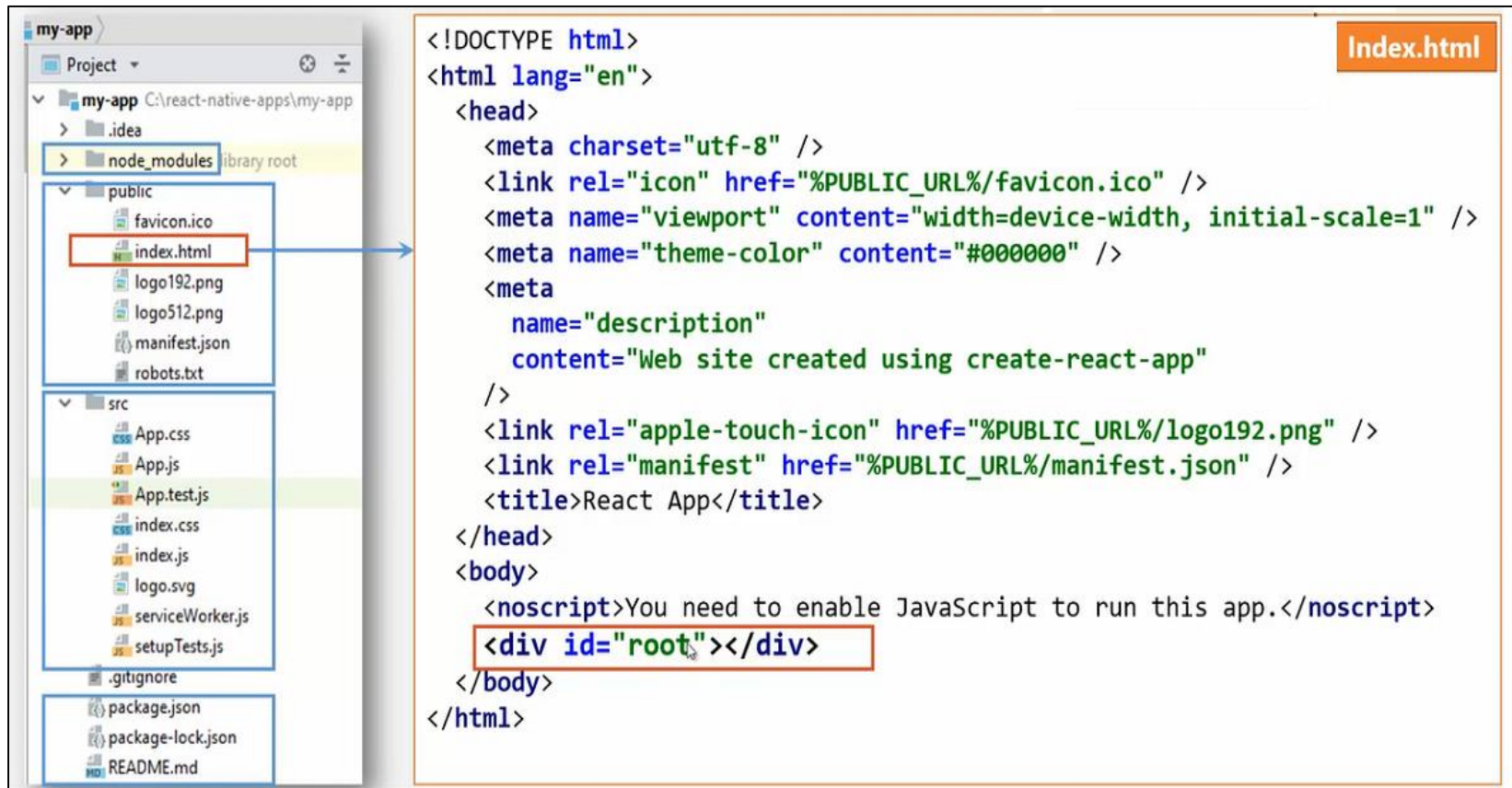
Principaux répertoires et fichiers

- **Nodes_modules** : Toutes les bibliothèques Java Script externes utilisées par l'application
- **index.html** : le point d'entrée de l'application
- **App.js** : le composant principal (root component)
- **Package.json** : contient les métadonnées du projet (titre, version, etc.) ainsi que toutes les dépendances nécessaires afin de pouvoir créer et exécuter une application React.
- Le répertoire **public/** : fichiers HTML, JSON et images de base (les racines de l'application).
- Le répertoire **src/** : fichiers sources (js, css, etc.) que nous créerons devront y être.

index.html

22

Si on ouvre le fichier public/index.html, on y trouve le contenu suivant :



The image shows a code editor with two panels. The left panel displays the file structure of a project named 'my-app'. The 'public' directory is expanded, showing files like 'favicon.ico', 'index.html' (highlighted with a red box), 'logo192.png', 'logo512.png', 'manifest.json', and 'robots.txt'. The 'src' directory is also expanded, showing files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'serviceWorker.js', and 'setupTests.js'. The right panel shows the content of the 'Index.html' file, which is a standard HTML document with a head section containing meta tags for charset, viewport, theme-color, and a description, and a body section containing a noscript message and a root div. The root div is highlighted with a red box.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

index.js / index.css

23



```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

serviceWorker.unregister();
```

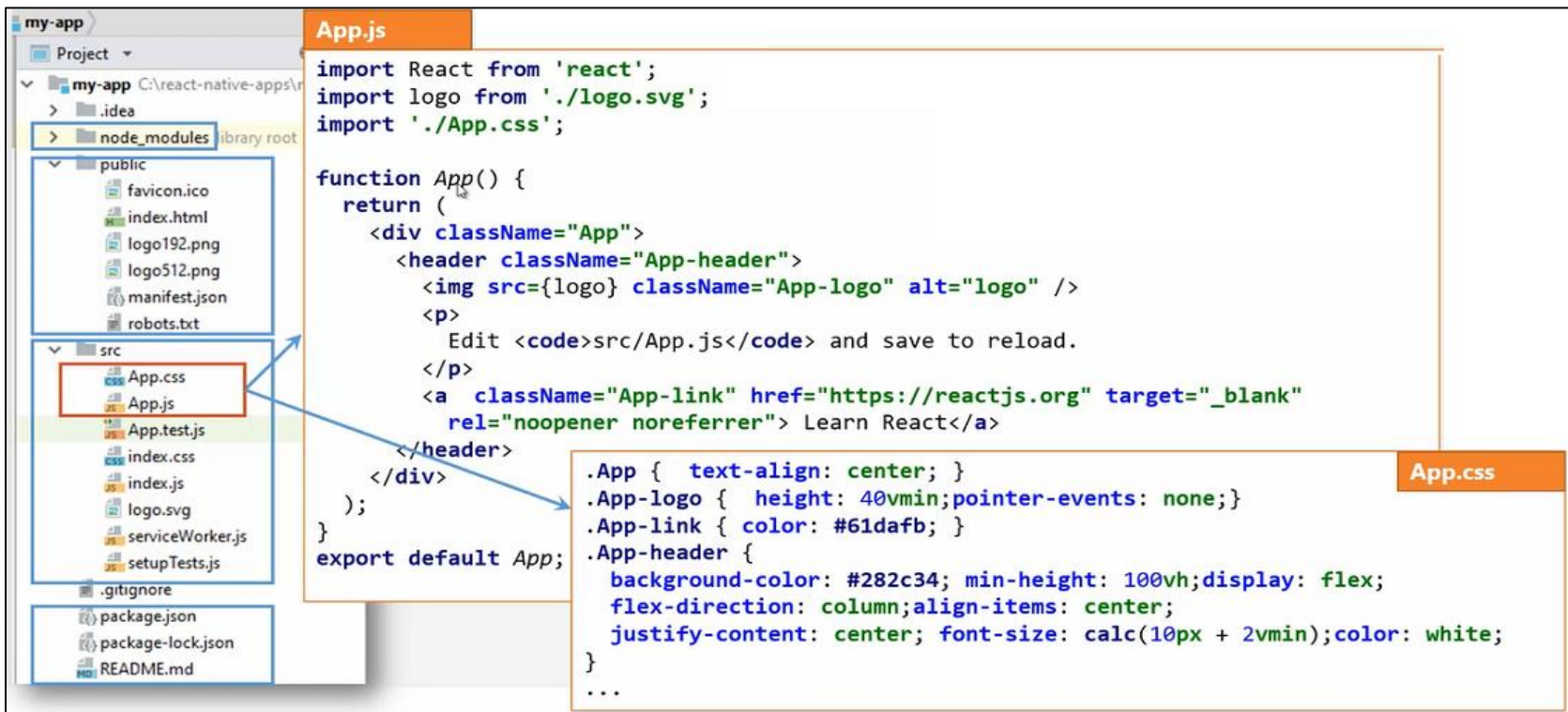
```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```


App.js

24

src/App.js est un exemple de composant React appelé «App» que nous obtenons par défaut lors de la création d'une nouvelle application :



The screenshot displays an IDE interface with a project explorer on the left and two code editors on the right. The project explorer shows a directory structure for 'my-app' with subdirectories 'public' and 'src'. The 'src' directory contains files including 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'serviceWorker.js', and 'setupTests.js'. The 'App.js' file is highlighted in the explorer and its content is shown in the top code editor. The 'App.css' file is also highlighted, and its content is shown in the bottom code editor. A blue arrow points from the 'App.css' file in the explorer to the bottom code editor.

```
App.js
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a className="App-link" href="https://reactjs.org" target="_blank"
          rel="noopener noreferrer"> Learn React</a>
      </header>
    </div>
  );
}
export default App;
```

```
App.css
.App { text-align: center; }
.App-logo { height: 40vmin;pointer-events: none;}
.App-link { color: #61dafb; }
.App-header {
  background-color: #282c34; min-height: 100vh;display: flex;
  flex-direction: column;align-items: center;
  justify-content: center; font-size: calc(10px + 2vmin);color: white;
}
...
```


Test d'un code

25

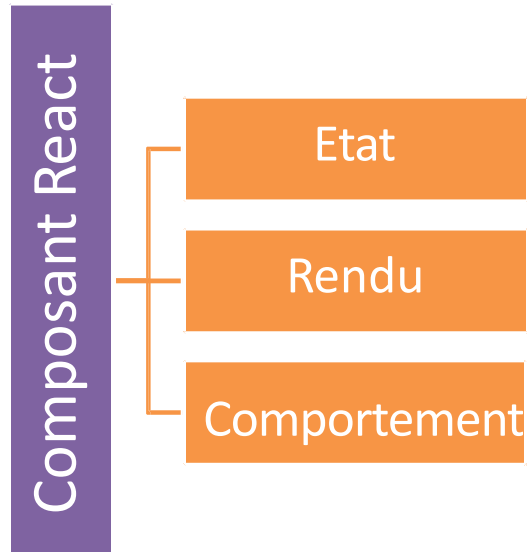
- ❑ Supprimez tous les fichiers source qui ne sont pas nécessaires :
 - ❑ Parcourez le dossier src dans le dossier my-app généré,
 - ❑ Supprimez les fichiers logo.svg, App.css et App.test.js.
- ❑ Dans le fichier src/App.js, remplacez le code existant par le code suivant :

```
import React, { Component } from 'react';
class App extends Component {
  render() {
    return (
      <div className="App">
        <h1>Hello World!</h1>
      </div>
    );
  }
}
export default App;
```



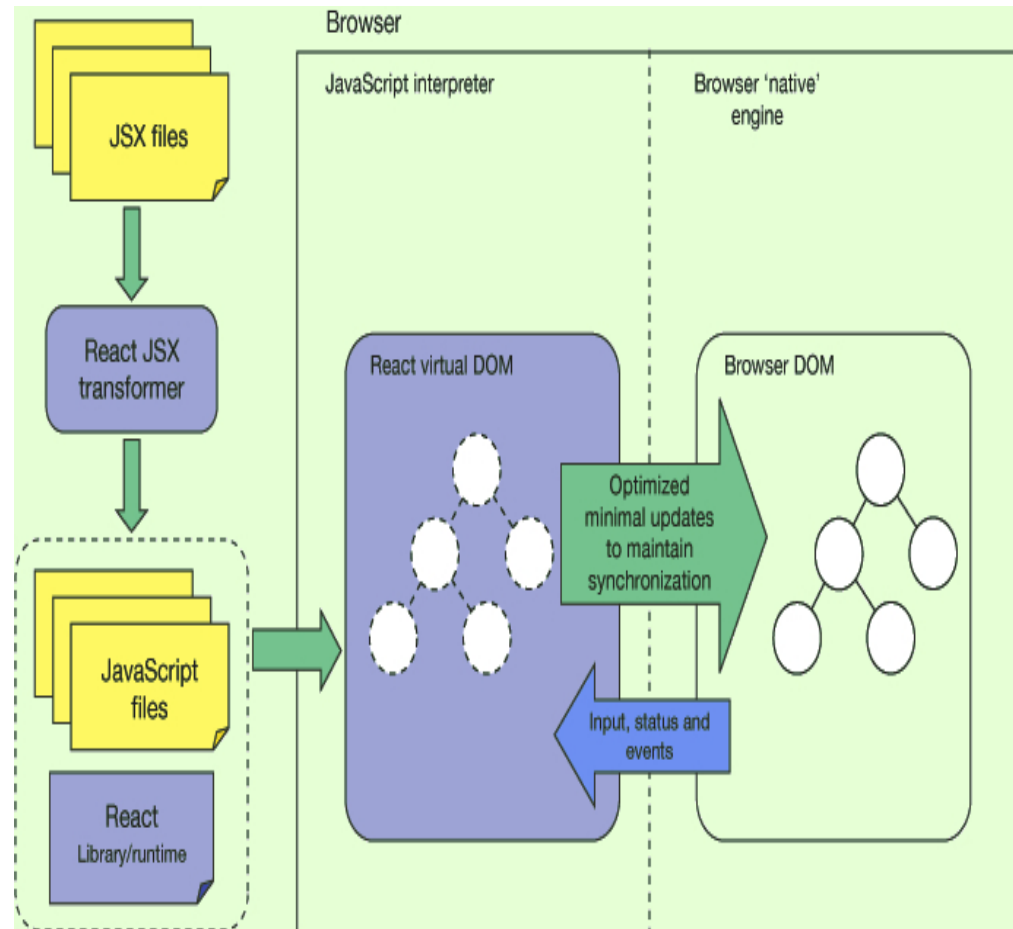
Les fondamentaux du React

- React
 - permet de créer des applications web de type SPA (Single Page Application)
 - basé sur une approche de programmation orientée composants (Web Components)
- Composant React est caractérisé par :
 - L'état du composant (State) qui représente le Modèle
 - Le rendu : Structure de la vue du composant (View)
 - Le comportement (Behaviour ou Controller)



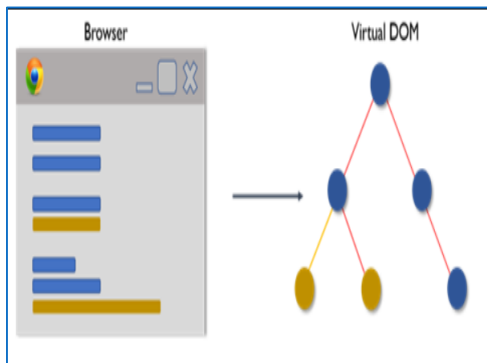
Architecture de React

- Le moteur de rendu JSX, permet de coupler dans un même fichier JSX, les trois aspects d'un composant React
- Le moteur de React génère un DOM virtuel qui sera transformé et synchronisé avec le DOM réel du Navigateur
- Le Virtuel DOM optimise les changements à apporter au niveau du Browser DOM. Ce qui lui permet d'accélérer l'aspect réactif du rendu.

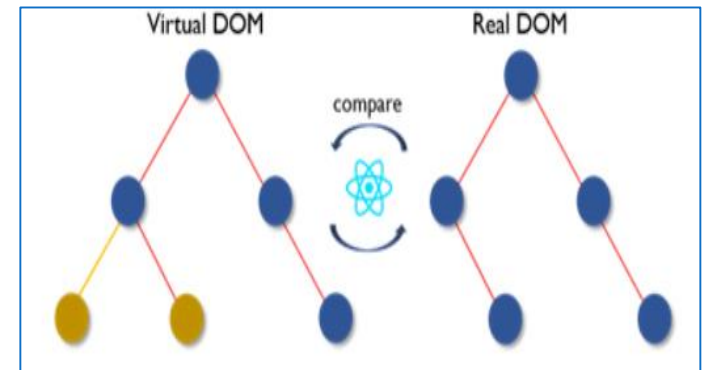


Virtual DOM

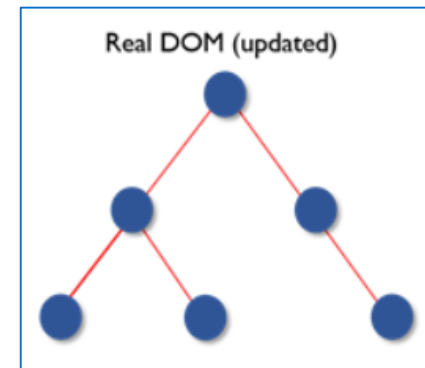
1 Chaque fois que des données du DOM réel changent, l'interface utilisateur entière est restituée en représentation Virtual DOM.



2 La différence entre la représentation DOM précédente et la nouvelle est calculée.



3 Une fois les calculs effectués, le DOM réel va mettre à jour uniquement les objets qui ont réellement changé.



Real DOM Vs Virtual DOM

Real DOM	Virtual DOM
Il se met à jour lentement.	Il se met à jour plus rapidement.
Peut directement mettre à jour le code HTML	Impossible de mettre à jour directement le code HTML.
Crée un nouveau DOM si l'élément est mis à jour.	JSX s'occupe de la mise à jour de l'élément
La manipulation DOM est très coûteuse.	La manipulation DOM est très facile.
Gaspillage de la mémoire.	Bonne gestion de la mémoire.

Principe de data binding

- Ce mécanisme consiste à lier la partie vue à la partie logique : si la source de données change, il est possible de faire en sorte que le contrôle soit automatiquement mis à jour.



- Grâce au data binding les éléments du code HTML seront liés au contrôleur JavaScript
- Ceci passe généralement par les étapes suivantes :
 - Détection des changements
 - La résolution des liaisons associées
 - La mise à jour du DOM

Data binding dans React

- React a une liaison de données à sens unique (one-way)
- Tout d'abord, l'état du modèle est mis à jour, puis il rend la modification de l'élément d'interface utilisateur.
- Toutefois, si vous modifiez l'élément d'interface utilisateur, l'état du modèle ne change pas. Toutes les modifications apportées à la liaison de données unidirectionnelle doivent être effectuées manuellement par le développeur, ce qui est bon en termes de maintenabilité et de tests.

