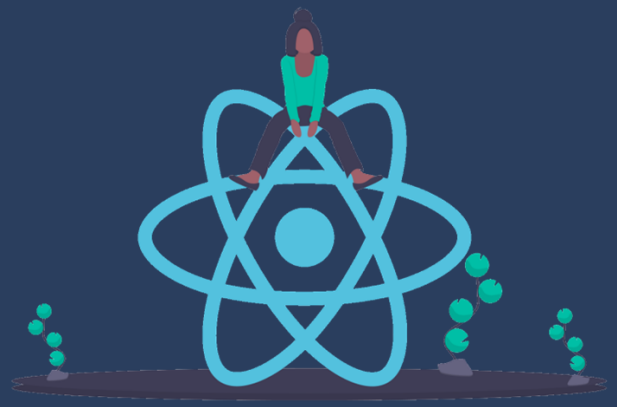
TD3

Frontend with React



https://www.jordansablon.fr/enseignement

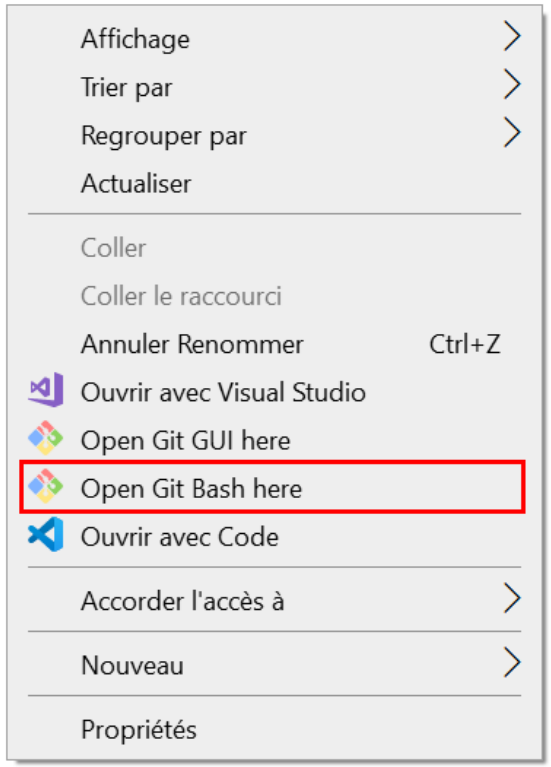
https://github.com/jsablonEnseignement/td3_sources

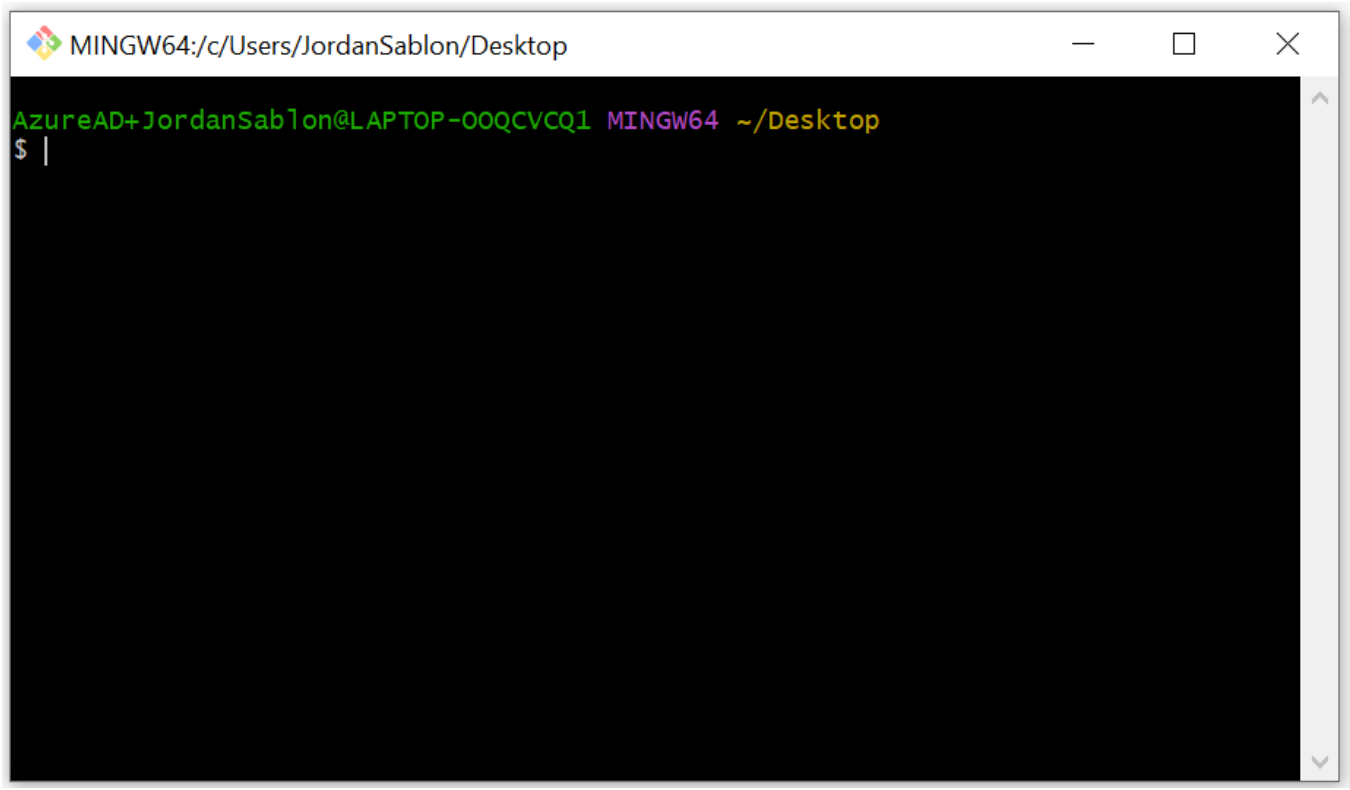
Sommaire

Récupération du code source du projet & installation des dépendances

1. Se positionner à l'emplacement de votre choix (où se trouvera le futur dossier du projet)

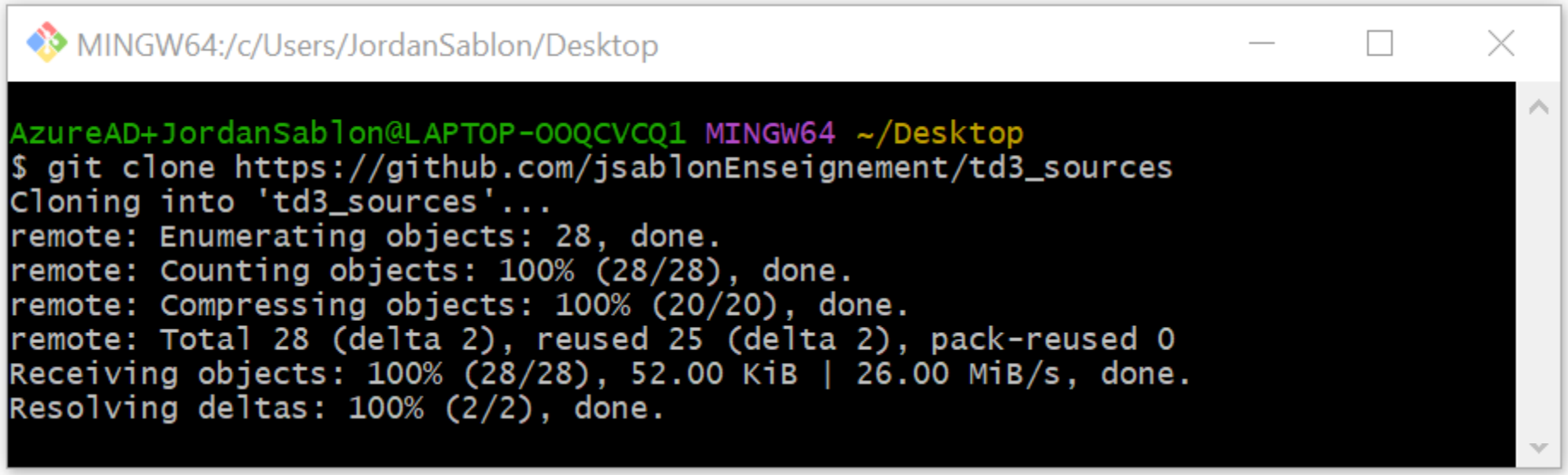
2. Ouvrir un invité de commande à l'emplacement précédent :





Récupération du code source du projet & installation des dépendances

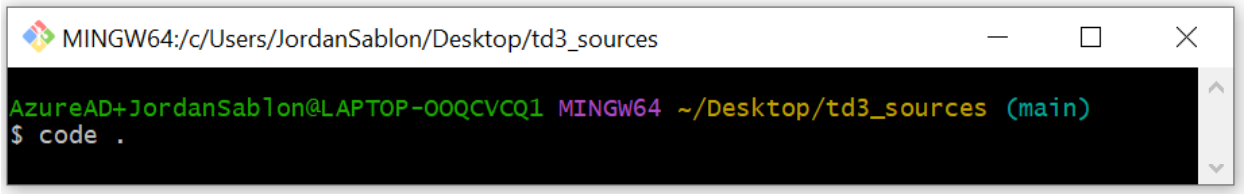
3. Cloner le répertoire Git : https://github.com/jsablonEnseignement/td3_sources

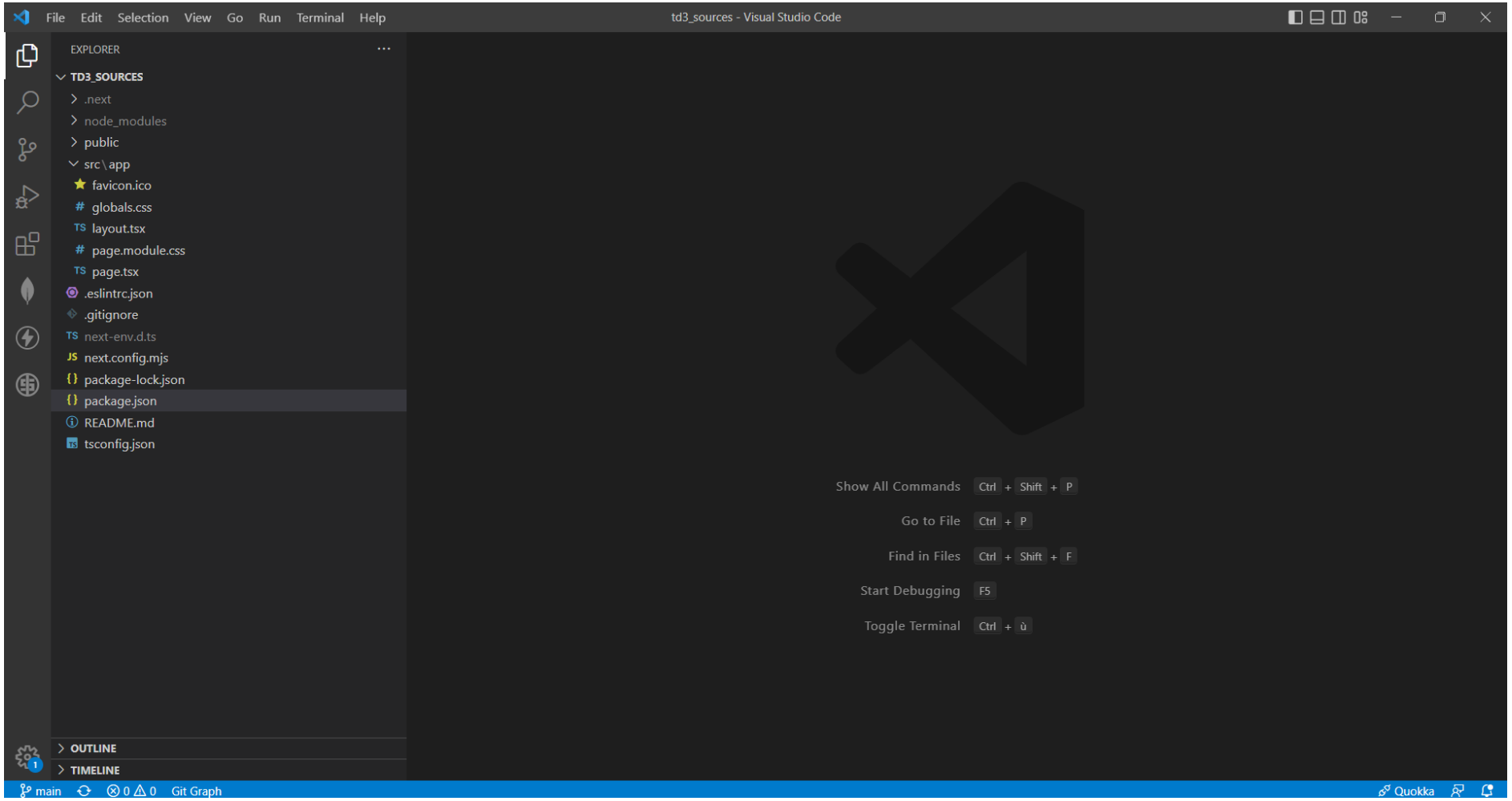


4. Se rendre dans le dossier précédemment cloné :

Récupération du code source du projet & installation des dépendances

5. Ouvrir le projet dans VSCode :





Récupération du code source du projet & installation des dépendances

Depuis le terminal de VSCode dans le dossier **td3_sources**, exécuter la commande **`npm i`** :

Les dépendances nécessaires au bon fonctionnement de l'application viennent d'être installées

Récupération du code source du projet & installation des dépendances

Depuis le terminal de VSCode, exécuter la commande **`npm run dev`** dans le dossier **td3_sources** afin de démarrer l'application React :

L'application est maintenant accessible via l'url http://127.0.0.1:3000/

Création d'un premier composant

Créer un dossier **/components** dans **/src/app** qui regroupera l'ensemble de nos composants

Créer un dossier **Demo** dans **/components**

Créer un fichier **Demo.tsx** qui contiendra le code du composant :

```
export const Demo = () => {
  return <>Hello from our first component!</>;
};
```

Importer et faire appel à ce composant depuis le fichier **page.tsx**

```
"use client";
import { Demo } from "./components/Demo/Demo";

export default function Home() {
  return (
    <Demo />
  );
}
```

Utilisation d'une variable d'état (*state*) et du hook *useState*

Depuis notre composant **Demo.tsx**, importer le hook ***useState***

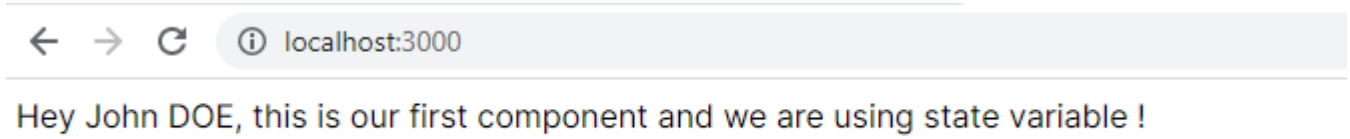
```
import { useState } from "react";
```

Initialisation du hook ***useState***

```
const [name, setName] = useState("John DOE");
```

Utilisation de la variable :

```
Hey {name}, this is our first component and we are using state variable!
```

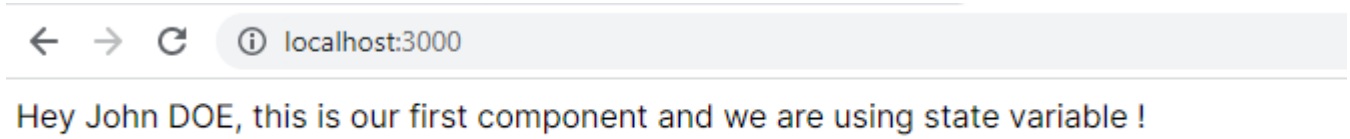


Utilisation d'une variable d'état (*state*) et du hook *useState*

Le composant **Demo.tsx** ressemble désormais à ceci :

```
import { useState } from "react";

export const Demo = () => {
  const [name, setName] = useState("John DOE");

  return <>Hey {name}, this is our first component and we are using state variable!
</>;
};
```

Il nous donne le rendu suivant :



Utilisation d'une variable d'état (*state*) et du hook *useState*

Ajout d'un champ de saisie ***input*** de type ***text***

```
<input type="text" defaultValue={name} />
```

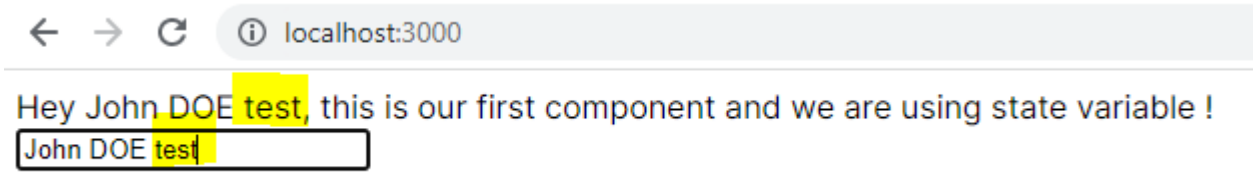
Ajout d'un évènement ***onChange*** afin de modifier la valeur

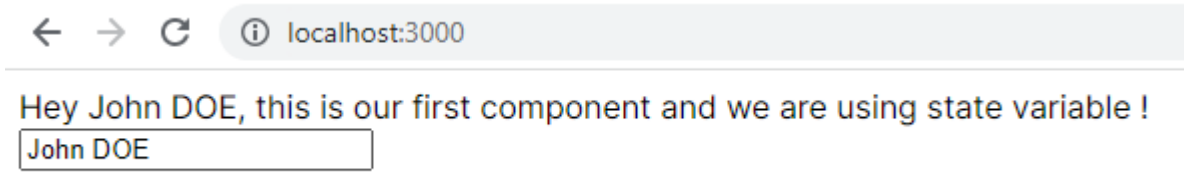
```
<input
    type="text"
    defaultValue={name}
    onChange={(e) => setName(e.target.value)}
/>
```

Utilisation d'une variable d'état (*state*) et du hook *useState*

Voici le code du composant et le rendu associé

```
import { useState } from "react";

export const Demo = () => {
  const [name, setName] = useState("John DOE");

  return (
    <>
      Hey {name}, this is our first component and we are using state variable!
      <input
        type="text"
        defaultValue={name}
        onChange={(e) => setName(e.target.value)}
      />

    </>;
};
```





Création d'un second composant & import / export

Création d'un fichier ***Hello.tsx*** qui contiendra le texte affiché du précédent composant :

```
export const Hello = () => {
  return (
    <>
      Hey John DOE, this is our first component and we are using state
variable
      !
    </>
  );
};
```

Création d'un second composant & import / export

Import du composant ***Hello.tsx*** depuis ***Demo.tsx***

```tsx
import { useState } from "react";
import { Hello } from "./Hello";

export const Demo = () => {
  const [name, setName] = useState("John DOE");
  return (
    <>
      <Hello />
      <br />
      <input
        type="text"
        defaultValue={name}
        onChange={(e) => setName(e.target.value)}
      />
    </>
  );
};
```

Utilisation des propriétés (*props*)

Ajout d'un paramètre ***name*** pour le composant ***Hello.tsx***

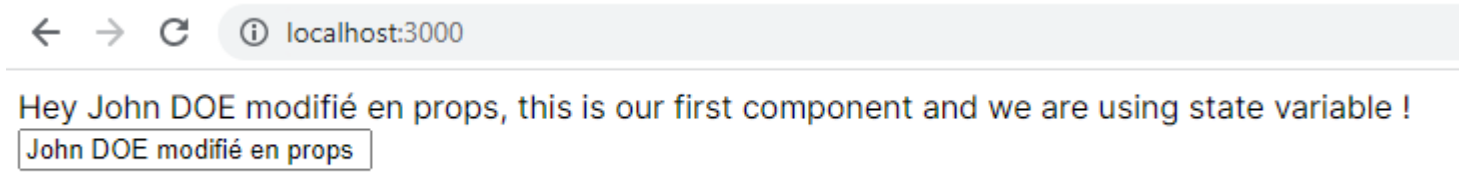
```tsx
type HelloProps = {
  name: string;
};

export const Hello = ({ name }: HelloProps) => {
  return (
    <>
      Hey {name}, this is our first component and we are using state
variable !
    </>
  );
};
```

→ On note la définition d'un type qui regroupe l'ensemble des paramètres d'entrée du composant ***Hello***

Utilisation des propriétés (*props*)

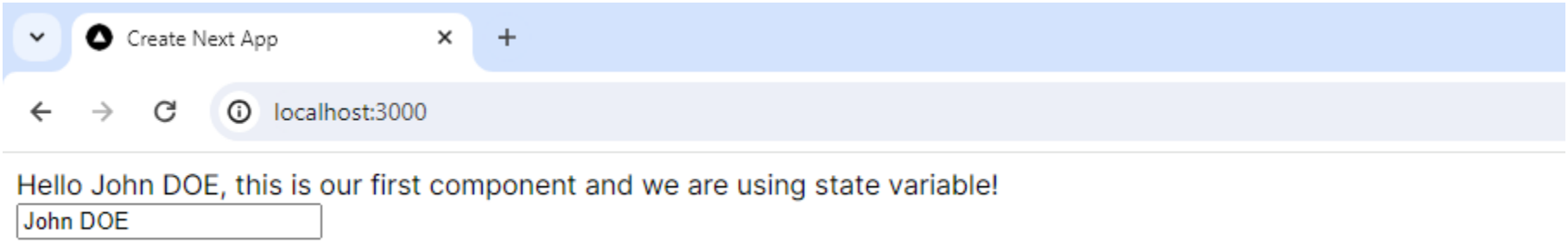
On passe la variable de state name comme paramètre du composant ***Hello***

```
<Hello name={name} />
```



Affichage du composant **<Hello />** seulement si *name* existe

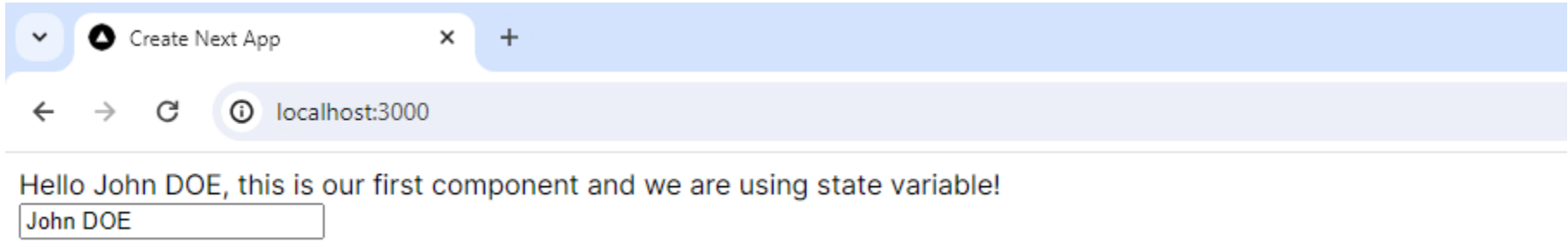
```
{name && <Hello name={name} />}
```



Conditionner l'affichage d'un composant

Il est également possible d'afficher un texte alternatif :

```
{name ? <Hello name={name} /> : "Veuillez renseigner un nom d'utilisateur"}
```



Utilisation du hook *useEffect*

Deux usages :

→ Exécution d'une action au chargement du composant (avant son rendu)

→ Exécution d'une action lors du changement d'état d'une ou plusieurs variable(s)

Syntaxe :

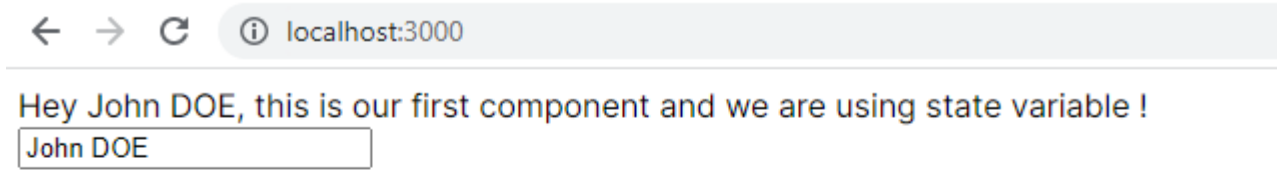
```
useEffect(() => {
        // some code here
}, []);
```

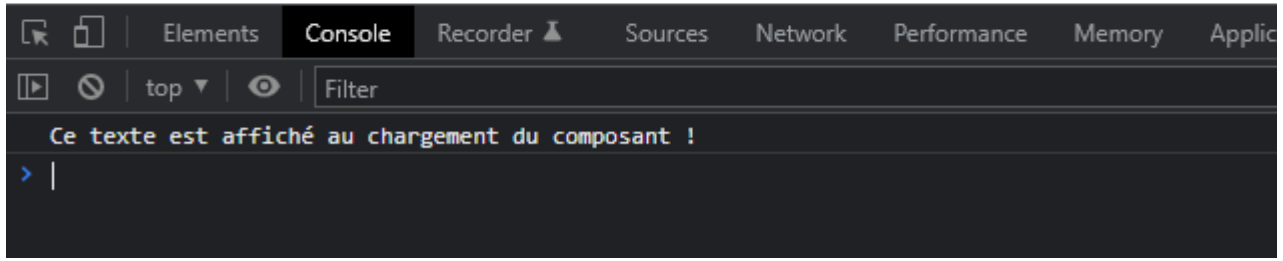
Utilisation du hook *useEffect*

Ajout d'un ***useEffect*** après la déclaration du ***useState*** dans le composant **Demo** afin d'écrire dans la console lors du chargement du composant.

Ouvrir les outils de développement du navigateur (F12) pour afficher la console.

```
useEffect(() => {
    console.log("Ce texte est affiché au chargement du composant !");
  }, []);
```

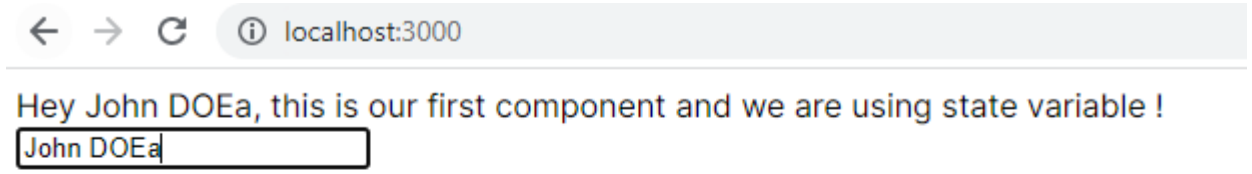


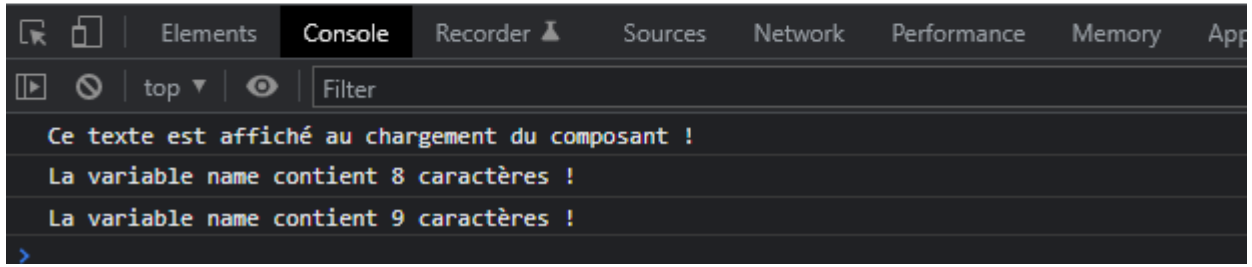


Utilisation du hook *useEffect*

Ajout d'un second ***useEffect*** d'afficher le nombre de caractère de la variable ***name***

```
useEffect(() => {
    console.log(`La variable name contient ${name.length} caractères !`);
  }, [name]);
```





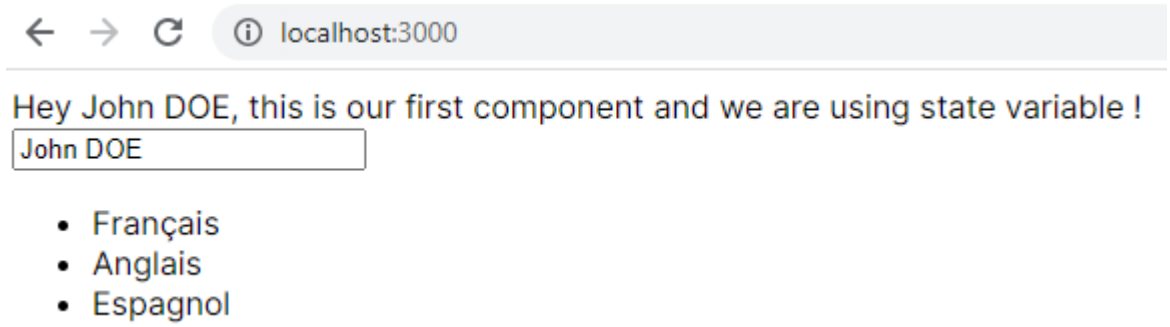
Affichage d'un composant au sein d'une itération

Ajout d'un tableau de valeur dans le composant **Demo.tsx**

```
  const languages = ["Français", "Anglais", "Espagnol"];
```

Affichage des valeurs sous forme de liste :

```
<ul>
  {languages.map((language) => (
    <li key={language}>{language}</li>
  ))}
</ul>
```



Affichage d'un composant au sein d'une itération

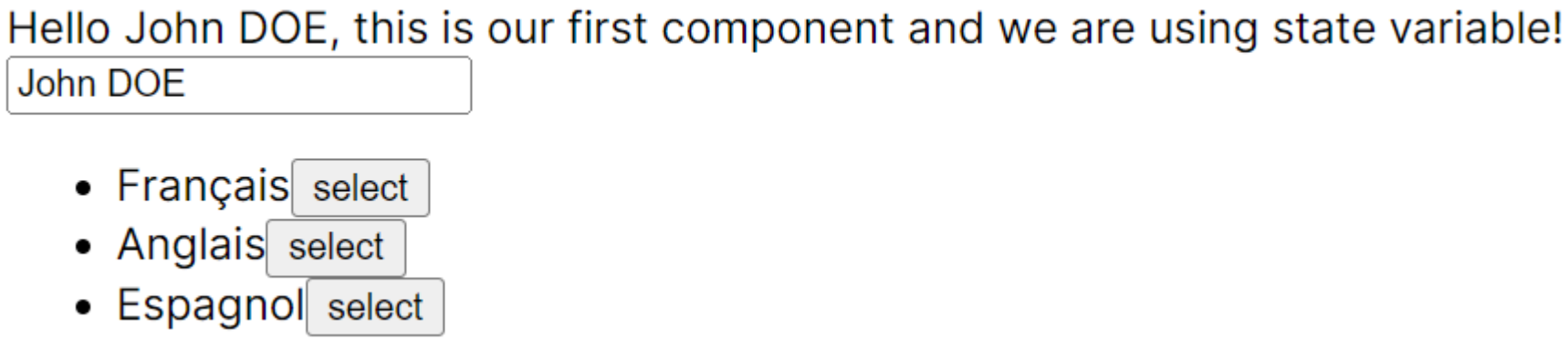
Voici désormais le code du composant :

```
import { useEffect, useState } from "react";
import { Hello } from "./Hello";

const languages = ["Français", "Anglais", "Espagnol"];

export const Demo = () => {
    const [name, setName] = useState("John DOE");

    useEffect(() => {
        console.log("Ce texte est affiché au chargement du composant !");
    }, []);

    useEffect(() => {
        console.log(`La variable name contient ${name.length} caractères !`);
    }, [name]);


    return <>
        {name ? <Hello name={name} /> : "Veuillez renseigner un nom d'utilisateur"}
        <br />
        <input type="text" defaultValue={name} onChange={(e) => setName(e.target.value)} />
        <ul>
            {languages.map((language) => (
                <li key={language}>{language}</li>
            ))}
        </ul>

    </>;
};
```

Affichage d'un composant au sein d'une itération

Ajout d'un bouton pour chaque élément :

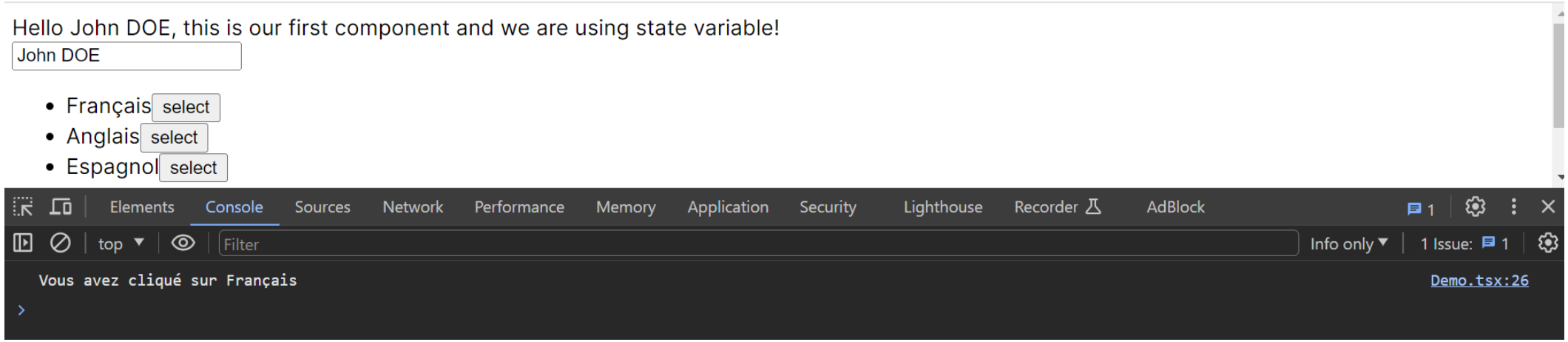
```
<ul>
    {languages.map((language) => (
        <li key={language}>
            {language}
            <button>select</button>
        </li>
    ))}
</ul>
```



Affichage d'un composant au sein d'une itération

Au clic sur ce bouton, on souhaite afficher la valeur de la ligne correspondante

```
<button onClick={() => console.log("Vous avez cliqué sur " + language)}>select</button>
```



Installation de la librairie

```
npm install antd
```

Import d'un composant (***Button***)

```
import { Button } from "antd";
```

Utilisation :

```
<Button type="primary">
   Ceci est un bouton de la librairie Ant Design
</Button>
```