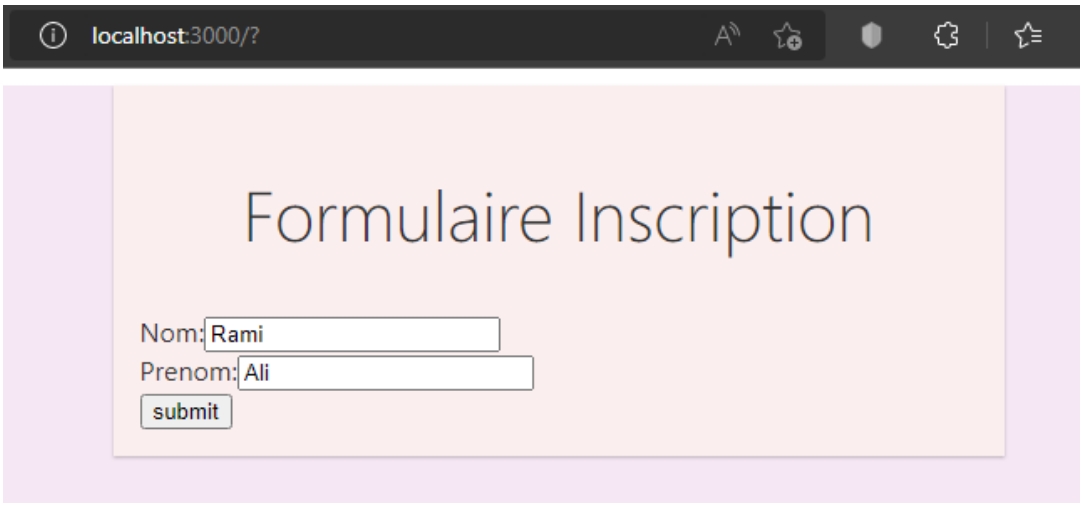
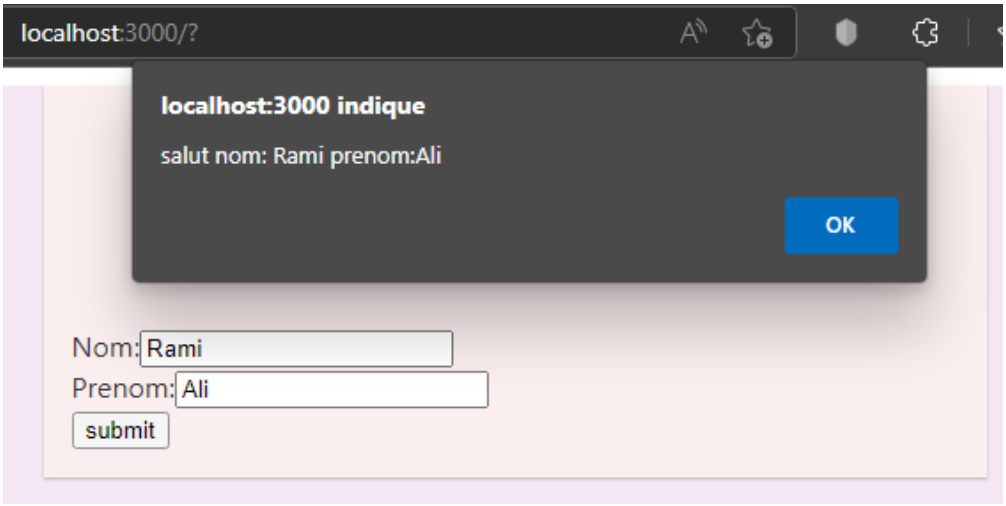
12.TP hooks

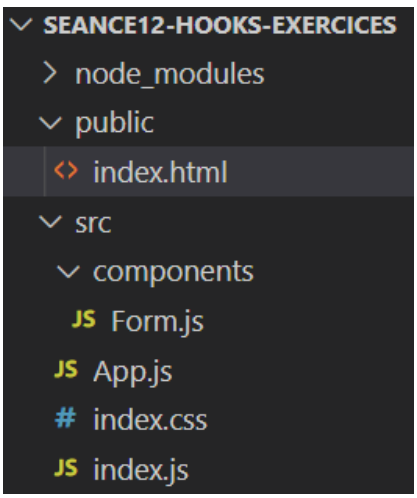
12.1. Exercice Formulaire inscription



Créer le composant Form qui permet d'afficher dans un message alert les informations nom et prenom



Structure de projet



Index.css

```
body {
    margin: 0;
    padding: 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;
  }
  .App {
    background-color: #f1daefa6;
    color: #333;
    min-height: 100vh;
  }
  .title {
    color:#333;
    font-size: 42px;
    font-weight: 300;
    margin: 0;
    padding: 32px;
    text-align: center;
  }
  .container list{
      padding: 16px;
      text-align: center;
    }

    .container {
      background-color: rgb(250, 238, 238);
      padding: 16px;
      box-shadow: 0 1px 2px rgba(0, 0, 0, 0.24);
      margin: 10px auto;
      margin-top: 10px;
      margin-bottom: 16px;
      max-width: 500px;
    }

    .container title {
      font-weight: 300;
    }

    .container list {
      padding: 16px;
      text-align: center;
    }
```

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
const element = document.getElementById("root");
const root = ReactDOM.createRoot(element);
root.render(<App />);
```

App.js

```
import React, { useState } from "react";
import Form from "./components/Form";
import '../src/index.css'

export default function App(){
    return (
    <div className="App">
    <Form/>
    </div>
    )
}
```

Form.js à compléter

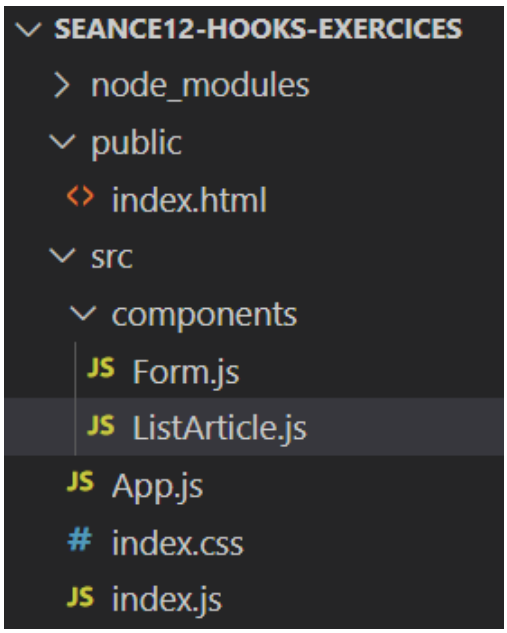
```
import React, { useState } from "react";
export default function Form(){
const [nom,setNom]=useState('')
const [prenom,setPrenom]=useState('')

function handlerOnChangeNom(event){
}
function handlerOnChangePrenom(event){
}
function handlerSubmit(event){
}
    return(
        <div className="container">
          <form onSubmit={handlerSubmit}>
        <div className="list">
         <h2 className="title">Formulaire Inscription</h2>
        <div>
        <label>Nom:</label><input type="text"  onChange={handlerOnChangeNom}
value={nom}/>
        </div>
        <div>
        <label>Prenom:</label><input
type="text"  onChange={handlerOnChangePrenom} value={prenom}/>
        </div>
        <div>
```

```
            <input type="submit" value="submit"/>
        </div>
        </div>
        </form>
        </div>
    )
}
```

12.2. Exercice gestion liste des articles

Le composant ListArticle permet d'ajouter un article dans une liste, l'interface affiche en bas les données de la liste.

Structure de projet



Index.css

Identique a celui de l'exercice 1 formulaire

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
const element = document.getElementById("root");
const root = ReactDOM.createRoot(element);
root.render(<App />);
```

App.js

```
import React, { useState } from "react";
import '../src/index.css'
import ListArticle from "./components/ListArticle";

export default function App(){
    return (
    <div className="App">
    <ListArticle/>
    </div>
    )
}
```

ListArticle.js à compléter

```
import React, { useState } from "react";
export default function ListArticle(){
    const [id,setId]=useState(0)
    const [designation,setDesignation]=useState('')
    const [prix,setPrix]=useState(0)
    const [articles,setArticles]=useState([])

function handlerAddArticle(){
}
function handlerOnChangeId(event){
}
function handlerOnChangeDesignation(event){
}
function handlerOnChangePrix(event){
    }
    return(
        <div className="container">
        <div className="list">
         <h2 className="title">Ajout d'un Article</h2>
        <div>
        <label>Id:</label><input type="text"  onChange={handlerOnChangeId}
value={id}/>
        </div>
        <div>
```

```
        <label>designation:</label><input
type="text"  onChange={handlerOnChangeDesignation} value={designation}/>
        </div>
        <div>
        <label>prix:</label><input type="text"  onChange={handlerOnChangePrix}
value={prix}/>
        </div>
        <div>
            <input type="button" value="Ajouter" onClick={handlerAddArticle}/>
        </div>
        <div>
            <h3>liste Articles</h3>
            <ul>



            </ul>
        </div>
        </div>
        </div>
          )
}
```

12.3. Exercice récupération des coordonnées de la souris

Récupération des coordonnées de la souris

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
const element = document.getElementById("root");
const root = ReactDOM.createRoot(element);
root.render(<App />);
```

App.js

```
import React, { useState } from "react";
import '../src/index.css'
import Coordonnee from "./components/Coordonnee";
export default function App(){
    return (
    <div className="App">
    <Coordonnee/>
    </div>
    )
}
```

Coordonnee.js à compléter

```
import React, { useEffect, useState } from "react";
export default function Coordonnee(){
const [x,setX]=useState(0)
const [y,setY]=useState(0)

useEffect(()=>{
window.addEventListener("mousemove",handleMouseMove)
return(()=>{window.removeEventListener("mousemove",handleMouseMove)})
},[])

function handleMouseMove(event){
    --------
}
    return(
        <div className="container">
            <h4 className="title">recuperation des coordonnes de la
souris</h4>
          <h5> coordonnes x:--- y: ----</h5>
        </div>
    )


}
```

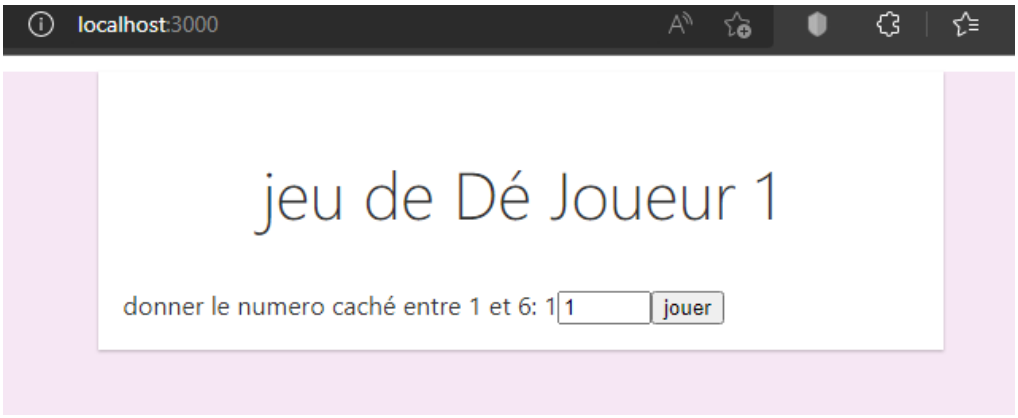
12.4. Exercice jeu de Dé

Le prince du jeu

Le premier joueur donne un nombre cache compris entre 1 et 6

Le deuxième joueur lance le Dé, si la face de Dé correspond au nombre caché la partie se termine en affichant le nombre d'essais, puis le programme demande d'initialiser le jeu

Les interfaces du jeu :



Le joueur 1 choisie un numéro entre 1 et 6 puis click sur le bouton jouer

Il s'affiche cette interface

Le deuxième joueur clique sur le bouton lancer jusqu'à la face de Dé correspond au numéro caché

Dans ce cas le numéro caché et 1

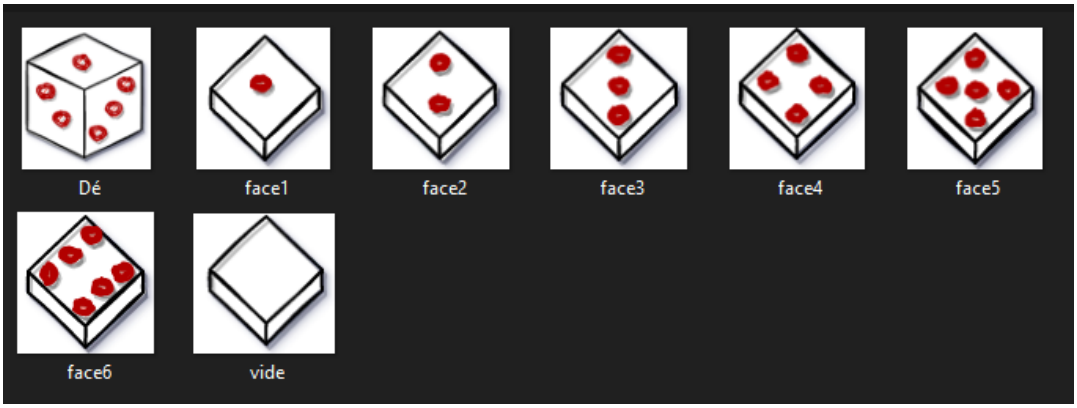
Et voila maintenant le joueur 2 a trouvé le numéro caché après 14 essais

Il s'affiche en suite le bouton initialiser qui permet d'initialiser le jeu le compteur prend la valeur 0

Et l'interface de joueur 1 s'affiche pour saisir un autre numéro caché.

Préparation du projet jeu Dé

Le dossier images qui se trouve dans le dossier public contient les images suivantes



Vous trouvez en format numérique le projet.

Essayez de tester le projet et de comprendre le code !!!

Pour exécuter le projet utiliser la console puis Se placer dans le dossier ProjetJeuDe

```
\TPs\ProjetJeuDe>npm install
```

Pour installer les dépendances

Puis `\TPs\ProjetJeuDe>npm start` pour lancer le jeu

Le travail à faire c'est de remplacer la classe composant JeuDe.js par un fonctionnel composant ☺

Allez bon courage