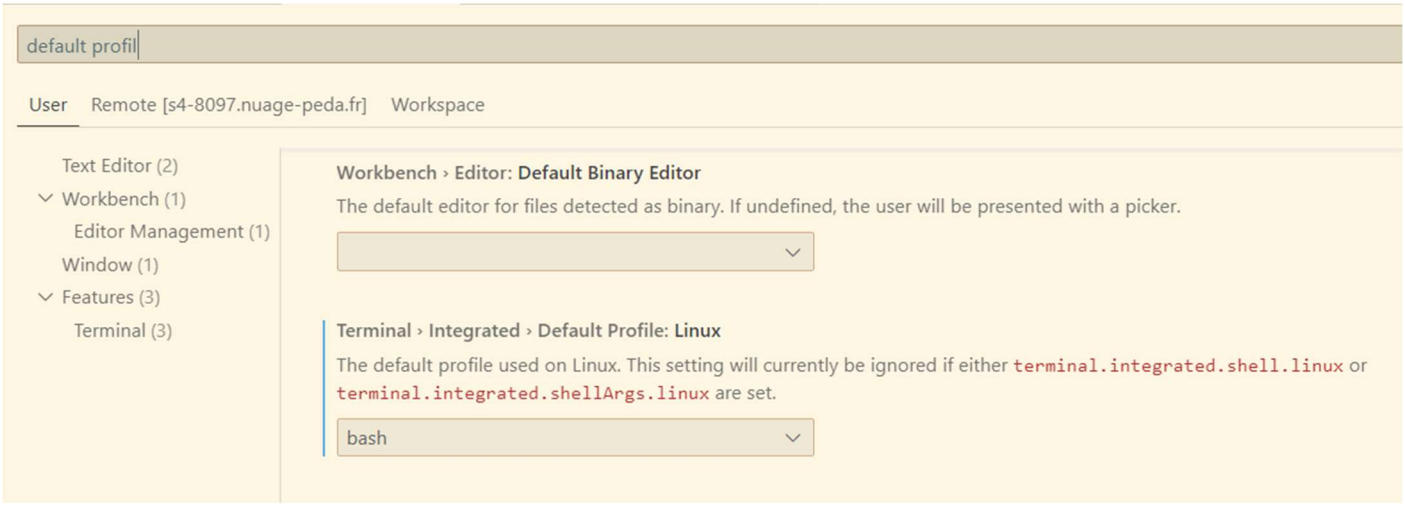
Doc de Parisot Clément

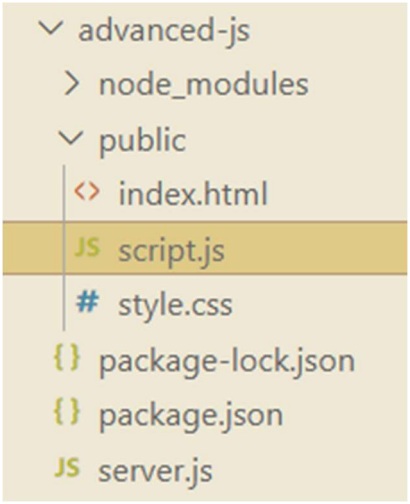
Config du terminal :



Check des versions :

```
login8097@REACT-8097:~/react-course$ node -v
v20.11.1
login8097@REACT-8097:~/react-course$ npm -v
10.2.4
```

Pour un projet JS mis sur un server, architecture cible :



Code du server :

```
const express = require('express');
const app = express();
const path = require('path');
const router = express.Router();
app.use(express.static(__dirname + '/public'));
router.get('/', function (req, res) {
    res.sendFile(path.join(__dirname + '/public/index.html'));
});
//add the router
app.use('/', router);
app.listen(process.env.port || 3000);
console.log('Running at Port 3000');
```

JS :

Déclaration de variable :

```
const bonjour = 4;
let salut = 7;
var nePasUtiliser = "Pas bien"

```

Template String :

```
const ajouteUnBonjour = (nom) => `Bonjour, ${nom} !`;

console.log(ajouteUnBonjour`John`);
```

- Pratique quand il faut mettre des variables dans un string
- Pratique pour formater des strings

Copie de données :

```
const fruits = {
  ukouko: 5,
  ukouka: 5,
  ukougk: 5,
  ukouoa: 5,
};

const fruits2 = {
  ...fruits,
  ukouko2: 5,
  ukouka2: 5,
  ukougk2: 5,
  ukouoa2: 5,
};
```

Toutes les valeurs de fruits sont dans fruits2

- Pratique pour changer des valeurs à la volés
- Pratique pour copier en ajoutant des infos
- Fonctionne aussi avec les tableaux (attention à l'ordre)

Falsy et Truthy :

```
if (!undefined && !null && !"" && !0 && !NaN) {
  console.log("tout est faux");
}
console.log(undefined, null, "", 0, NaN);
```

Toutes ces valeurs sont fausses, les autres sont vraies

- Pratique pour savoir si un string continent des caractères
- 0 = false

```
const rien = "";

const jeVeuxUnNom = rien || "Pas de nom";
console.log(jeVeuxUnNom);
```

-
- ^ Utile pour mettre un remplacement en cas de non-input !

Sortir des éléments d'objets :

```
const { ukouko } = fruits;
const { ukouko:ukoukoMaisAutre } = fruits2;
console.log(ukouko);
console.log(ukoukoMaisAutre);
```

- Sortir des éléments de librairies importées
- Renommer des variables pour la facilité d'utilisation
- Fonctionne aussi avec les tableaux (Attention à l'ordre)

```
const notes = [10, 6, 13, 18];
const notesTab = notes.map((note) => note + 2);
console.log(notesTab);
const caPasseOuCaCasse = notes.map((note) => note >= 10);
console.log(caPasseOuCaCasse);
```

Map et création d'un filtre (notes en dessous de 10 sont marquées « false »)

```
function filter(array, value) {
  return array.map((val) => (val == value ? null : val));
}

console.log(filter(fruitsArray, "concombre"));
```

Avec une vraie fonction + condition ternaire

ForEach :

```
function isColumnClicked(x) {
  let allclicked = 0;
  cases[x].forEach((carre) => (allclicked += carre.clicked));
  return allclicked == taille;
}
```

Passage par toutes les cases d'une rangée dans le morpion

Fetch :

```
const setBody = async () => {
  body = await fetch("https://jsonplaceholder.typicode.com/posts/1")
    .then((response) => response.json())
    .then((json) => {
      body = json.body;
      console.log("got value in body");
    })
    .catch((e) => console.log(e));
};
setBody();
```

On récup le json et on le met dans une variable

INSTALLATION DE REACT

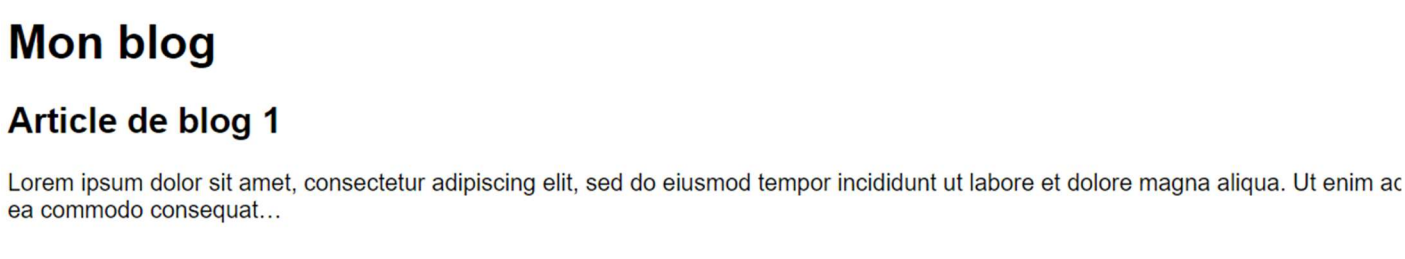
Directement dans le HTML :

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>React Kiment</title>
    <link rel="stylesheet" href="./style.css" />
  </head>
  <body>
    <div id="root"></div>

    <div class="container">…
    </div>

    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

Fonctionne mais ne donne pas accès à tout le confort accessible



Hiérarchie simple :

```js
function setupReact() {
  const container = document.getElementById("root");
  const root = ReactDOM.createRoot(container);

  const titre = React.createElement("h1", {}, "Mon blog");
  const secondTitre = React.createElement("h2", {}, "Article de blog 1");
  const p = React.createElement(
    "p",
    {},
    "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna alic
  );

  const div = React.createElement("div", {}, [secondTitre, p]);
  const gene = React.createElement("div", {}, [titre, div]);
  root.render(gene);
```

Installation d’un projet Vite :

```
create-vite@5.2.2
Ok to proceed? (y) y
✓ Project name: … react-learn
? Select a framework: › - Use arrow-keys. Return to submit.
›   Vanilla
? Select a framework: › - Use arrow-keys. Return to submit.
    Vanilla
? Select a framework: › - Use arrow-keys. Return to submit.
    Vanilla
? Select a framework: › - Use arrow-keys. Return to submit.
    Vanilla
? Select a framework: › - Use arrow-keys. Return to submit.
    Vanilla
✓ Select a framework: › React
✓ Select a variant: › JavaScript
```

```
✓ Project name: … react-learn
✓ Select a framework: › React
✓ Select a variant: › JavaScript + SWC

Scaffolding project in /var/www/html/react-course/react-learn...

Done. Now run:
```

Ne pas oublier d’ajouter ça pour le nuage péda :

Lancer le server :

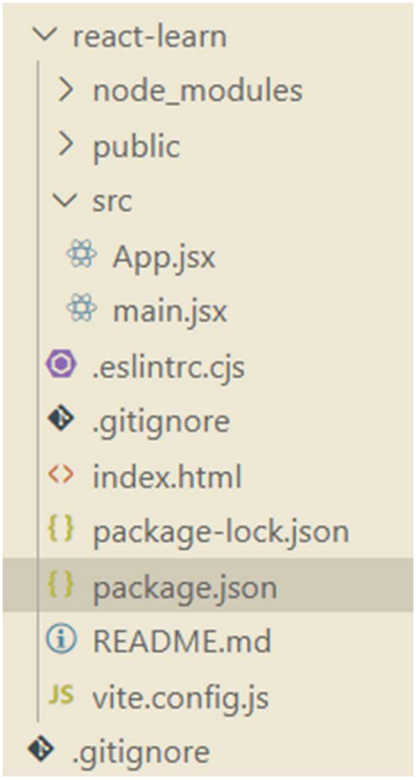
```
login8097@REACT-8097:~/react-course/react-learn$ npm run dev

> react-learn@0.0.0 dev
> vite --port 3000


  VITE v5.1.6  ready in 193 ms

  ➜  Local:   http://localhost:3000/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Architecture cible pour un projet propre :



TodoList en JSX :

TodoList :

Element 1 ☐

Element 2 ☐

Element 3 ☐

Element 4 ☐

```jsx
function App() {
  return (
    <>
      <h1>TodoList :</h1>

      <div>
        <p>Element 1 <input type="checkbox" name="" id="" /></p>
        <p>Element 2 <input type="checkbox" name="" id="" /></p>
        <p>Element 3 <input type="checkbox" name="" id="" /></p>
        <p>Element 4 <input type="checkbox" name="" id="" /></p>
      </div>
    </>
  );
}
export default App;
```

Styles

Possibilité de mettre des styles directement dans les éléments :

```
const Check = ({ index }) => {
  return (
    <div>
      <label htmlFor={"check-" + index} style={{padding:"10px", backgroundColor:"red"}}>Element {index}</label>
      <input type="checkbox" name={"check-" + { index }} />
    </div>
  );
};
```

- Pas de généralisation du style (élément par élément)
- Mais peut servir pour les éléments particuliers

Solution alternative :

```
const styles = {
  texte:{
    backgroundColor:"red",
    padding:"10px"
  }
}

const Check = ({ index }) => {
  return (
    <div>
      <label htmlFor={"check-" + index} style={styles.texte}>Element {index}</l
      <input type="checkbox" name={"check-" + { index }} />
    </div>
```

Mettre les styles dans une constante pour les utiliser autre part

Importation du css à partir d'un fichier :

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

function App() {
  const [nombreCoches, setNombreCoches] = useState();
  const [checks, setChecks] = useState([]);

  const Check = ({ index }) => {
    return (
      <div>
        <label htmlFor={"check-" + index} className="texte">
          Element {index}
```

Utilisation de style.module.css :

```
import app from "./app.module.css";

function App() {
  const [nombreCoches, setNombreCoches] = useState();
  const [checks, setChecks] = useState([]);

  console.log(app);

  const Check = ({ index }) => {
    return (
      <div>
        <label htmlFor={"check-" + index} className={app["test-green"]}>
```

- Importation du css sous forme de tableau
- Portée locale

Utilisation en camel case :

Exemple de la portée globale de App.css par rapport à app.module.css :

App.jsx | main.jsx

react-learn > src > main.jsx

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <App />
    <p className="text-green">Hello green !</p>
    <p className="text-red">Hello red !</p>
  </React.StrictMode>
);

```

app.module.css

react-learn > src > app.module.css > .text-greer

```css
.testGreen {
    color: green;
}

.text-green {
    color: lightgreen;
}
```

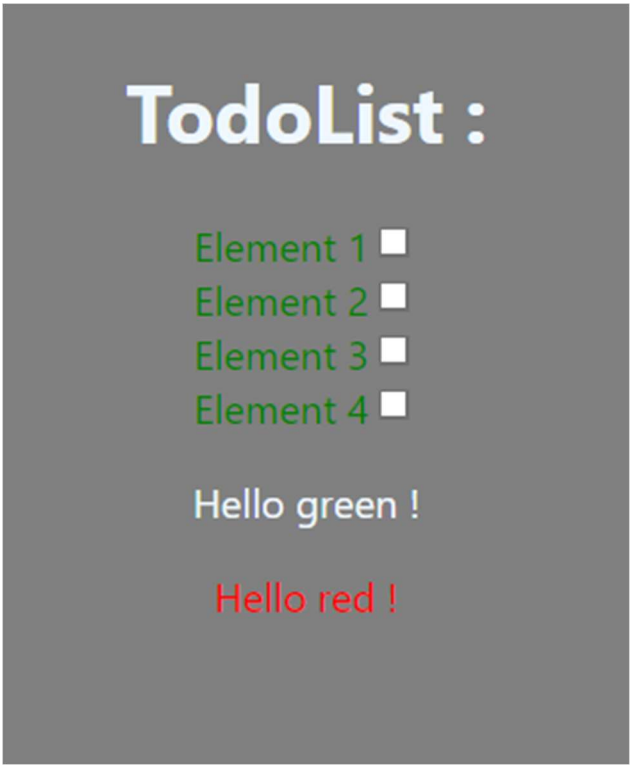
App.css

react-learn > src > App.css > .text-red

```css
}

.texte {
  background-color: red;
  padding: 10px;
}

.text-red {
  color: red;
}
```



Création d’un vrai style pour la todolist :

J’ai utilisé un mélange de Bootstrap pour poser les bases, et de css pour « fine tune »

```
const Check = ({ index }) => {
  return (
    <div className="check">
      <div className="two-col">
        <div>
          <input
            type="checkbox"
            name={"check-" + { index }}
            className="form-check-input"
          />
          <label htmlFor={"check-" + index}>Element {index}</lab
        </div>
        <div className="buttons">
          <button className="btn btn-primary">
            <i className="fa-solid fa-trash"></i>
          </button>
          <button className="btn btn-primary">
            <i className="fa-solid fa-pen"></i>
          </button>
        </div>
      </div>
    );
```

```
}

.todolist label {
  margin: 7px;
}

.todolist button {
  background-color: #efefef;
  border-color: #efefef;
  color: black;
  margin-left: 5px;
}

.two-col {
  display: grid;
  grid-template-columns: 1fr 1fr;
  margin: 5px;
}

.buttons {
  text-align: right;
}
```

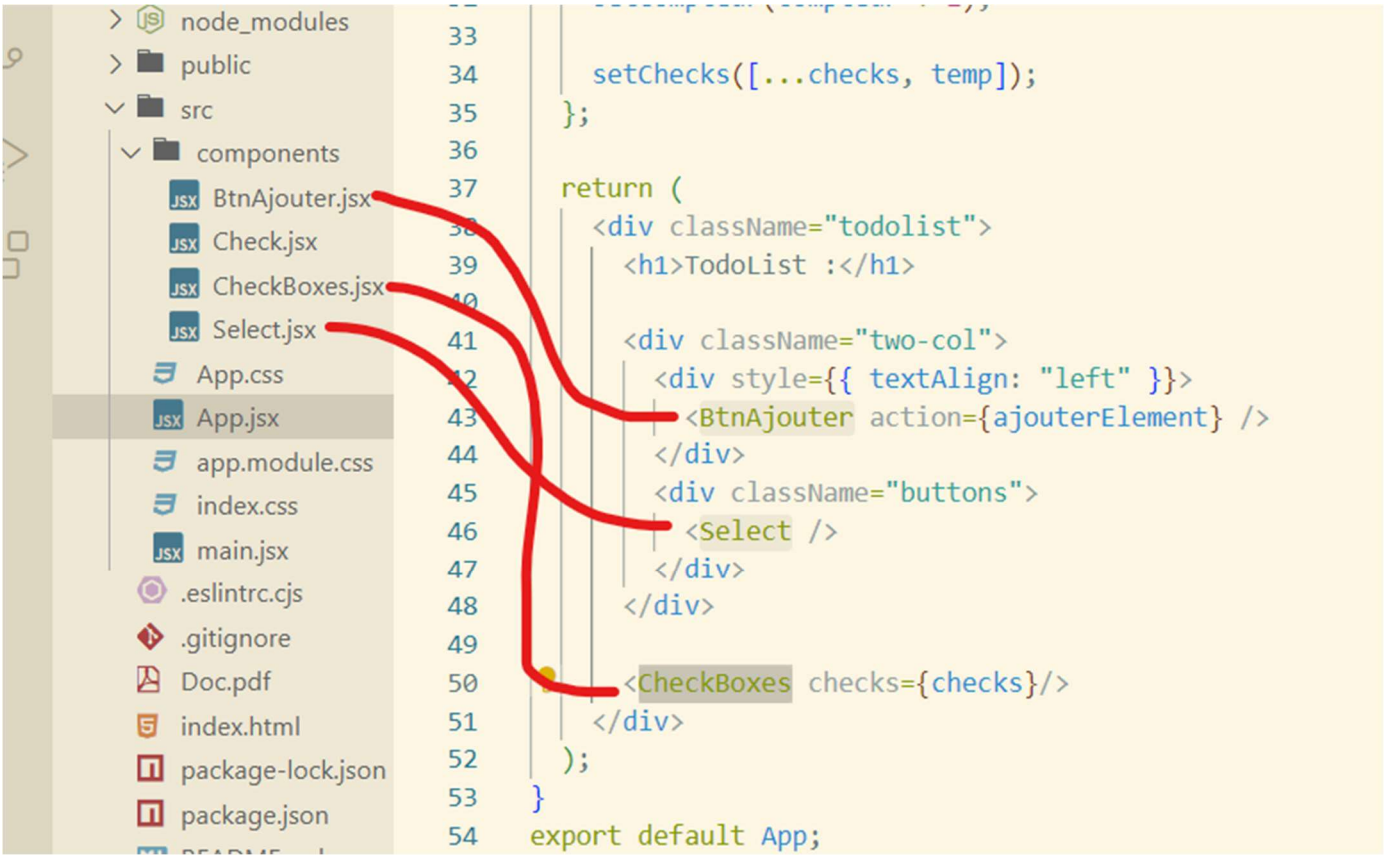
```
eturn (
  <div className="todolist">
    <h1>TodoList :</h1>

    <div className="two-col">
      <div style={{ textAlign: "left" }}>
        <button className="btn btn-primary" style={{backgroundColor
      </div>
      <div className="buttons">
        <select name="" id="" className="form-select" style={{backg
          <option>Toutes</option>
        </select>
      </div>
    </div>

    <div className="checkboxes">
      <Check index={1} />
      <Check index={2} />
      <Check index={3} />
      <Check index={4} />
    </div>
  </div>
;
```

Création de composents :

```
import BtnAjouter from "./components/BtnAjouter";
import Select from "./components/Select";
import CheckBoxes from "./components/CheckBoxes";
```



Un exemple :

App.jsx | Check.jsx | Select.jsx | BtnAjouter.jsx

react-learn > src > components > Check.jsx > default

```
export default ({ index }) => {
  return (
    <div className="check">
      <div className="two-col">
        <div>
          <input
            type="checkbox"
            name={"check-" + { index }}
            className="form-check-input"
          />
          <label htmlFor={"check-" + index}>Element {index}</
        </div>
        <div className="buttons">
          <button className="btn btn-primary">
            <i className="fa-solid fa-trash"></i>
          </button>
          <button className="btn btn-primary">
            <i className="fa-solid fa-pen"></i>
          </button>
        </div>
      </div>
    </div>
  );
};

```

Remplacement des boutons par le composant Boutton :

```jsx
<div className="buttons">
  <Boutton
    label={<i className="fa-solid fa-trash"></i>}
    textColor={"black"}
    backgroundColor={"#efefef"}
  />

  <Boutton
    label={<i className="fa-solid fa-pen"></i>}
    textColor={"black"}
    backgroundColor={"#efefef"}
  />
</div>
```

```jsx
export default ({ action }) => (
  <Boutton
    label={"Ajouter +"}
    textColor={"white"}
    backgroundColor={"#626ff1"}
    action={action}
  />
);
```

Le composant :

```jsx
export default ({ backgroundColor, label, textColor, action }) =>
  return (
    <button
      className="btn btn-primary"
      style={{ backgroundColor: backgroundColor, color: textColor
      onClick={action}
    >
      {label}
    </button>
  );
};
```

Fonction pour ajouter des éléments :

```jsx
const ajouterElement = () => {
  const temp = [];

  temp.push(<Check index={compteur} key={compteur} />);
  setCompteur(compteur + 1);

  setChecks([...checks, temp]);
};

return (
  <div className="todolist">
    <h1>TodoList :</h1>

    <div className="two-col">
      <div style={{ textAlign: "left" }}>
        <BtnAjouter action={ajouterElement} />
      </div>
      <div className="buttons">
        <Select />
      </div>
```

Fonctionnalités de la TodoList :

- Ajouter
- Supprimer
- Modifier
- Changer de page vers le morpion