Après exercices, revenir à l'exemple de count

```
export default function App() {
    const [count, setCount] = React.useState(0)

    function add() {
        setCount(prevCount => prevCount + 1)
    }

    function subtract() {
        setCount(prevCount => prevCount - 1)
    }

    /**
     * Challenge:
     * - Create a new component called `Count`
     *    - It should receive a prop called `number`, whose value
     *      is the current value of our count
     *    - Have the component render the h2.count element below
     *      and display the incoming prop `number`
     * - Replace the h2.count below with an instance of
     *   the new Count component, passing the correct value
     *   to its `number` prop.
     * - After doing this, everything should be working the
     *   same as before.
     */

    return (
        <main className="container">
            <div className="counter">
                <button
                    className="minus"
                    onClick={subtract}
                    aria-label="Decrease count"
                >-</button>

                <h2 className="count">{count}</h2>

                <button
                    className="plus"
                    onClick={add}
                    aria-label="Increase count"
                >+</button>
            </div>
        </main>
    )
}
```

parler de rendering (console.log dans App et dans count)

reprendre exemple card profile

```
import React from "react"
import avatar from "./images/user.png"
import starFilled from "./images/star-filled.png"
import starEmpty from "./images/star-empty.png"

export default function App() {
    const [contact, setContact] = React.useState({
        firstName: "John",
        lastName: "Doe",
        phone: "+1 (212) 555-1212",
        email: "itsmyrealname@example.com",
        isFavorite: false
    })

    let starIcon = contact.isFavorite ? starFilled : starEmpty

    function toggleFavorite() {
        setContact(prevContact => ({
            ...prevContact,
            isFavorite: !prevContact.isFavorite
        }))
    }

    /**
     * Challenge: Move the star image into its own component (Star)
     * - It should receive a prop called `isFilled` that it
     *   uses to determine which icon it will display. (You'll
     *   need to import the 2 star icons into that new component first).
     * - Import and render that component, passing the value of
     *   `isFavorite` to the new `isFilled` prop.
     * - Don't worry about the abiliity to flip this value quite yet.
     *   Instead, you can test if it's working by manually changing
     *   `isFavorite` in state above.
     */

    return (
        <main>
            <article className="card">
                <img
                    src={avatar}
                    className="avatar"
                    alt="User profile picture of John Doe"
                />
                <div className="info">
                    <button
                        onClick={toggleFavorite}
                        aria-pressed={contact.isFavorite}
                        aria-label={contact.isFavorite ? "Remove from
favorites" : "Add to favorites"}
                        className="favorite-button"
                    >
                        <img
```

```
                            src={starIcon}
                            alt={contact.isFavorite ? "filled star icon" :
"empty star icon"}
                            className="favorite"
                        />
                    </button>
                    <h2 className="name">
                        {contact.firstName} {contact.lastName}
                    </h2>
                    <p className="contact">{contact.phone}</p>
                    <p className="contact">{contact.email}</p>
                </div>

            </article>
        </main>
    )
}
```

oncliCk=toggle favorite dans App component

Passing data around React

(diagramme)

App.jsx

```
import React from "react"
import Header from "./Header"
import Body from "./Body"

export default function App() {
    return (
        <main>
            <Header />
            <Body />
        </main>
    )
}
```

Body.jsx

```
import React from "react"

export default function Body() {
    return (
        <section>
            <h1>Welcome back, ___!</h1>
        </section>
    )
}
```

Header.jsx

```
import React from "react"
import avatar from "./icons/user.png"

export default function Header() {
    const [userName, setUserName] = React.useState("Joe")

    return (
        <header>
            <img src={avatar} />
            <p>{userName}</p>
        </header>
    )
}
```

index.css

```
* {
    box-sizing: border-box;
}

body {
    margin: 0;
    background-color: whitesmoke;
}

header {
    height: 65px;
    box-shadow: 0px 2.98256px 7.4564px rgba(0, 0, 0, 0.1);
    display: flex;
    justify-content: flex-end;
    align-items: center;
    padding-inline: 20px;
    background-color: #dce6fd
}

header > img, header > p {
    cursor: pointer;
}

section {
    padding: 20px;
}
```

```
    /**
     * Challenge:
     * Raise state up a level and pass it down to both the
     * Header and Body components through props.
     */
```

Sound pad challenge

Partie I :

```
import pads from "./pads"

export default function App() {
    /**
     * Challenge part 1:
     * 1. Initialize state with the default value of the
     *    array pulled in from pads.js
     * 2. Map over that state array and display each one
     *    as a <button> (CSS is already written for you)
     *    (Don't worry about using the "on" or "color"
     *    properties yet)
     */
    return (
        <main>
            <div className="pad-container">
                {/* <button>s go here */}
            </div>
        </main>
    )
}
```

pads.js

```
export default [
    {
        id: 1,
        color: "#F18D8B",
        on: true
    },
    {
        id: 2,
        color: "#F5C280",
        on: false
    },
    {
        id: 3,
        color: "#EEEC79",
        on: true
    },
    {
        id: 4,
        color: "#64ED98",
        on: true
    },
    {
        id: 5,
        color: "#63DEED",
        on: false
    },
    {
        id: 6,
        color: "#877FED",
        on: false
    },
    {
        id: 7,
        color: "#A57FE9",
        on: false
    },
    {
        id: 8,
        color: "#F289C1",
        on: true
    },
]
```

Solution :

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    const buttonElements = pads.map(pad => (
        <button key={pad.id}></button>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Dynamic styles

style= en HTML

```
<html>
    <head>
        <link rel="stylesheet" href="/index.css">
    </head>
    <body style="background-color: red">
        <div id="root"></div>
        <script src="/index.jsx" type="module"></script>
    </body>
</html>
```

exemple en JS vanille :

```
document.getElementById("something").style.backgroundColor = ""
```

exemple en React :

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)
    const styles = {
        backgroundColor: "red"
    }
    const buttonElements = pads.map(pad => (
        <button key={pad.id}></button>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Exemple (Dark Mode)

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    /**
     * Challenge: use a ternary to determine the backgroundColor
     * of the buttons
     * If darkMode is true, set them to "#222222"
     * If darkMode is false, set them to "#cccccc"
     */

    const buttonElements = pads.map(pad => (
        <button key={pad.id}></button>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Solution :

```
export default function App({ darkMode }) {
    const [pads, setPads] = React.useState(padsData)

    const styles = {
        backgroundColor: darkMode ? "#222222" : "#cccccc"
    }
    /**
     * Challenge: use a ternary to determine the backgroundColor
     * of the buttons
     * If darkMode is true, set them to "#222222"
     * If darkMode is false, set them to "#cccccc"
     */

    const buttonElements = pads.map(pad => (
        <button style={styles} key={pad.id}></button>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Pads challenge part 2

index.css :

```
* {
    box-sizing: border-box;
}

body {
    background-color: #1C1917;
}

main {
    display: flex;
    justify-content: center;
    align-items: center;
}

.pad-container {
    display: grid;
    grid-template-columns: repeat(4, 100px);
    grid-template-rows: repeat(2, 100px);
    gap: 10px;
}

button {
    height: 100px;
    width: 100px;
    border: 3px solid white;
    border-radius: 5px;
    cursor: pointer;

}
```

```
import padsData from "./pads"

export default function App() {
    const [pads, setPads] = React.useState(padsData)

    const buttonElements = pads.map(pad => (
        <button key={pad.id}></button>
    ))

    /**
     * Challenge part 2:
     * 1. Create a separate component called "Pad" and
     *    replace the `button` above with our <Pad /> component
     * 2. Pass the Pad component a prop called `color` with the
     *    value of the same name from the `padsData` objects
     * 3. In the Pad component, apply an inline style to the <button>
     *    to set the backgroundColor of the button.
     *
     * (We'll deal with the "on" property soon)
     */
    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Solution :

App.jsx

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    const buttonElements = pads.map(pad => (
        <Pad key={pad.id} color={pad.color} />
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Pad.jsx :

```
export default function Pad(props) {

    return (
        <button style={{backgroundColor: props.color}}></button>
    )
}
```

Part III

```
export default function Pad(props) {
    /**
     * Challenge part 3:
     * Our buttons got turned off by default! Update the code
     * so if the button is "on", it has the className of "on".
     */
    return (
        <button
            style={{backgroundColor: props.color}}
        ></button>
    )
}
```

Change this in index.css

```
button {
    height: 100px;
    width: 100px;
    border: 3px solid white;
    border-radius: 5px;
    cursor: pointer;
    opacity: 0.1;
}

button.on {
    opacity: 1;
}
```

solution :

Pad.jsx

```
export default function Pad(props) {

    return (
        <button
            style={{backgroundColor: props.color}}
            className={props.on ? "on" : ""}
        ></button>
    )
}
```

App.jsx

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    const buttonElements = pads.map(pad => (
        <Pad key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
```

```
                <div className="pad-container">
                    {buttonElements}
                </div>
            </main>
        )
    }
```

Montrer exemple avec &&

```
export default function Pad(props) {

    return (
        <button
            style={{backgroundColor: props.color}}
            className={props.on && "on"}
        ></button>
    )
}
```

PART IV

Option 1 : local state

```
export default function Pad(props) {
/**
 * Challenge: Create state controlling whether
 * this box is "on" or "off". Use the incoming
 * `props.on` to determine the initial state.
 *
 * Create an event listener so when the box is clicked,
 * it toggles from "on" to "off".
 *
 * Goal: clicking each box should toggle it on and off.
 */

    return (
        <button
            style={{backgroundColor: props.color}}
            className={props.on ? "on" : undefined}
        ></button>
    )
}
```

Solution :

```
export default function Pad(props) {
    const [on, setOn] = React.useState(props.on)

    function toggle() {
        setOn(prevOn => !prevOn)
    }

    return (
        <button
            style={{backgroundColor: props.color}}
            className={on ? "on" : undefined}
            onClick={toggle}
        ></button>
    )
}
```

NOM : DERIVED STATE (STATE dérivé)

Problème : Out of sync with the parent state : 2 different sources of truth

Ca marche, mais imaginons d'ajouter une feature turn all off :

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    function turnAllPadsOff() {
        console.log("Turning off")
        setPads(prevPads => prevPads.map(pad => ({
            ...pad,
            on: false
        })))
    }

    const buttonElements = pads.map(pad => (
        <Pad key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
            <button className="all-off" onClick={turnAllPadsOff}>Turn All
Off</button>
        </main>
    )
}
```

Option 2 : shared state

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    /**
     * Challenge: Create a toggle() function that logs
     * "clicked!" to the console
     *
     * Pass that function down to each of the Pad components
     * and set it up so when they get clicked, the function runs
     */

    const buttonElements = pads.map(pad => (
        <Pad key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Solution :

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    /**
     * Challenge: Create a toggle() function that logs
     * "clicked!" to the console
     *
     * Pass that function down to each of the Pad components
     * and set it up so when they get clicked, the function runs
     */

    function toggle() {
        console.log("Clicked!")
    }

    const buttonElements = pads.map(pad => (
        <Pad toggle={toggle} key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

```
export default function Pad(props) {
    const [on, setOn] = React.useState(props.on)

    return (
        <button
            style={{backgroundColor: props.color}}
            className={on ? "on" : undefined}
            onClick={props.toggle}
        ></button>
    )
}
```

Exemple :

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    function toggle(id) {
        // map over the pads array, and if the current item has
        // the same id as the one passed to this function, then
        // flip its `
    }

    const buttonElements = pads.map(pad => (
        <Pad toggle={toggle} key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}

export default function Pad(props) {
    const [on, setOn] = React.useState(props.on)

    return (
        <button
            style={{backgroundColor: props.color}}
            className={on ? "on" : undefined}
            onClick={() => props.toggle(id)}
        ></button>
    )
}
```

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    function toggle(id) {
        console.log(id)
        /**
         * Challenge:
         * Call setPads to update the state of the one pad that was
         * clicked. Map over the previous pads array, and if the current
         * item you're iterating over has the same id as the `id` passed
         * to this function, then return a new object with the `on` value
         * set to the opposite of what it was before.
         * Otherwise (if the ids don't match), just return the previous
         * item as it was, unchanged.
         */
    }

    const buttonElements = pads.map(pad => (
        <Pad toggle={toggle} id={pad.id} key={pad.id} color={pad.color} on={pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Solution :

App.jsx

```
export default function App() {
    const [pads, setPads] = React.useState(padsData)

    function toggle(id) {
        setPads(prevPads => prevPads.map(item => {
            return item.id === id ? {...item, on: !item.on} : item
        }))
    }
    const buttonElements = pads.map(pad => (
        <Pad toggle={toggle} id={pad.id} key={pad.id} color={pad.color} on=
{pad.on}/>
    ))

    return (
        <main>
            <div className="pad-container">
                {buttonElements}
            </div>
        </main>
    )
}
```

Pad.jsx :

```
export default function Pad(props) {
    return (
        <button
            style={{backgroundColor: props.color}}
            className={props.on ? "on" : undefined}
            onClick={() => props.toggle(props.id)}
        ></button>
    )
}
```