**W9 -** PRACTICE

*Going deeper with states*

## *At the end of his practice, you should be able to…*

* Reflect on **your usage of states**
* Style widgets dynamically
* **Pass functions** as value to props
* **Handles objects** in states
  + Treat state as **read-only**
  + Copy objects with the **spread syntax**

## *How to work?*

* Download **the start code** from the Google classroom
* For each exercise you can either:
  + Run npm install
  + Or move an existing node\_modules to the exercise folder *(fastest option!)*

## *How to submit?*

* **Create a repository on GitHub** with the name of this practice:

Ex: C2-S6 PRACTICE

* **Push your final code** on this GitHub repository (if you are lost, [follow this tutorial](https://www.datacamp.com/tutorial/git-push-pull) )
* Finally submit on **Google classroom** your GitHub repository URL

Ex: https://github.com/thebest/ C2-S6 PRACTICE.git

## *Are you lost?* Queueing a Series of State Updates – React

*You can read the following documentation to be ready for this practice:*

<https://www.w3schools.com/react/react_events.asp>

<https://react.dev/learn/updating-objects-in-state>

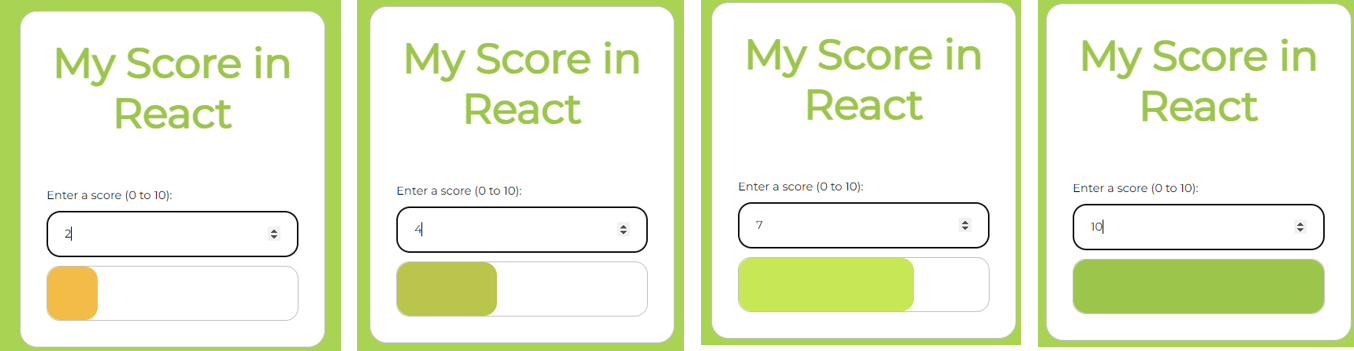
# EXERCISE 1

You can click on this link to see the expected results for this exercise:

<https://react-s11-ex1.vercel.app/>

The goal of this app is to update the progress bar, according to the value entered in the input field:

* + - Values range are from 0 to 10
    - Progress bar color changes depending on the score (see bellow)



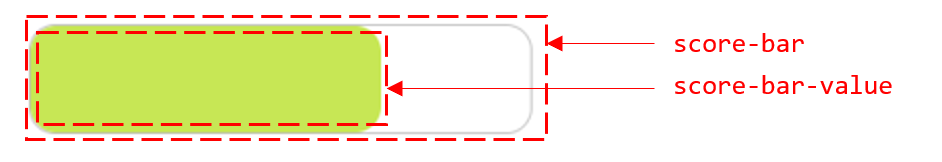
*Depending on the score value (0 to 10) the progress bar with and color shall be updated*

**Step 1:** What is/are the state(s) you plan to use for this exercise? Complete the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State Name | Type | Role | Used for… | Changed when… |
| score | number | Tracks the user's input score value | Use to store the enter score | User enters a new number in the input field between 0-10 |
|  |  |  |  |  |

**Step 2:** Update the **progress bar width** according to the score value

*Tips: The progress has 2 div: the border rectangle and the filled rectangle. Which one should you update?*



+ the one that should update is use score bar value.

**Step 3:** Update the **progress bar color** according to the score value

# EXERCISE 2

You can click on this link to see the expected results for this exercise:

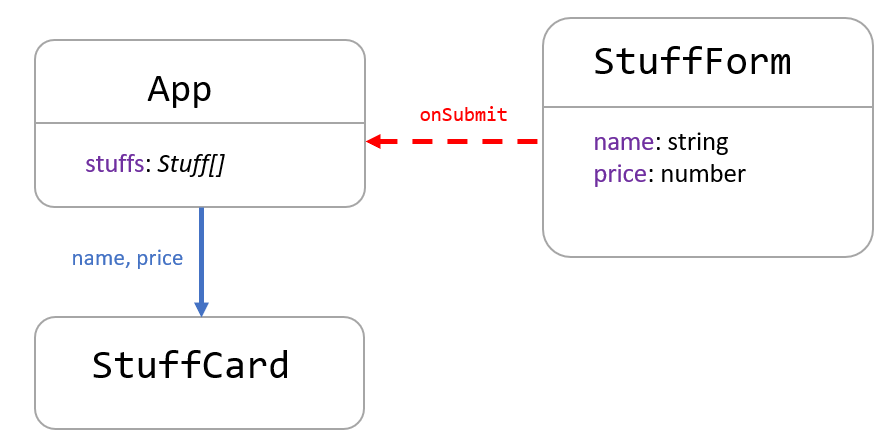
<https://react-s11-ex2.vercel.app/>

*The user can specify an object name and price. By clicking on Add, this object shall be added to the list.*

**Step 1:**

Look at the start code. We have 3 components.

* App manages a state with the list of objects
* StuffCard displays 1 object *(nothing to change on this component!)*
* StuffForm manages the form and send an event when user has click on Add



Complete the **StuffFrom** to handle the input values with the following states:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | Type | Component | Used to… | Changed when… |
| name | String | StuffFrom | Store the entered name | Input name had changed |
| price | number | StuffFrom | Store the entered price | Input price had changed |

**Step 2:**

When user clicks on Add button, the StuffFrom component sends an event to the App component

Complete the code to send this event, with the entered name and price as parameter

Note: For now, just log something on the console to check you can manage the event on App component. For example:

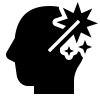
A new object named Piano, price 15$ will be added to the list

**Step 3:**

The last step is to update the state containing the stuff, with the new object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | Type | Component | Used to… | Changed when… |
| stuff | Array | App | Storing all stuffs | Add a new stuff |

First read the following articles about **objects** and **arrays** in state:



* <https://react.dev/learn/updating-objects-in-state>
* <https://react.dev/learn/updating-arrays-in-state>

**Q1** – Why states containing an array or an object **should be treated as read only**?

* React uses reference equality to determine if a state has changed. If you modify an array or object directly (mutate it), React won't detect the change because the reference remains the same. This can lead to components not re-rendering when they should.

**Q2** – Why do we need to **create a new array** when we want to add a created object to a state array?

* Creating a new array ensures that React detects the state change through reference equality checking. When we create a new array, React sees that the reference has changed and knows it needs to re-render the component.

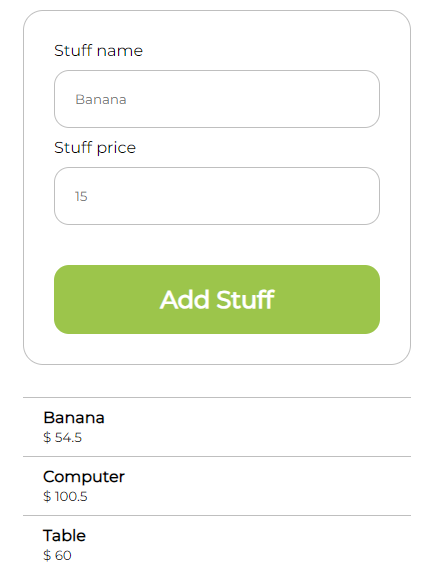
**Q3** – What is the **spread operator**? How can you use it to clone the state array and add the new created

object?

* The spread operator (...) allows us to expand an array into individual elements. In our case, we use it to create a new array by copying all elements from the existing array and adding the new object

**Q4** – Then complete the code to update the state array with the new object.

*The finished app will look like this:*



# EXERCISE 3

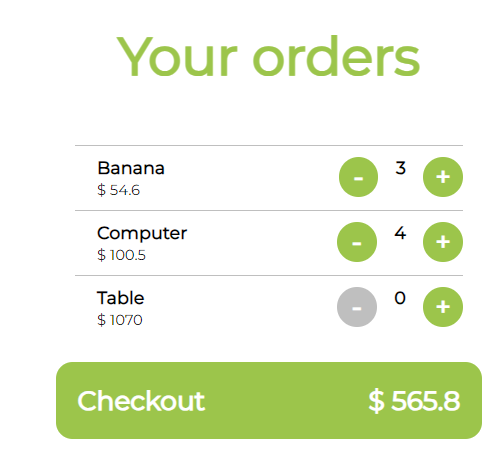
You can click on this link to see the expected results for this exercise:

<https://react-s11-ex3.vercel.app/>

*This app allows the user to adjust the quantity of items to buy and to compute the total price.*

*Note: the minus button is disabled when the quantity is equal to 0.*

*The finished app will look like this:*



**Step 1:**

Look at the start code. We have 3 components.

* App manages a state with the list of items
* Order Card handle the quantity to order for 1 item
* CheckoutButton displays the total sum *(nothing to change on this component!)*

**Q1 -** What is/are the state(s) you plan to use for this exercise? Complete the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State Name | Type | Component | Used for… | Changed when… |
| order | Array/object | app | Storing the list of orders | A new order is added, updated, or deleted |
|  |  |  |  |  |

* *Tips: Is the total price a state?*

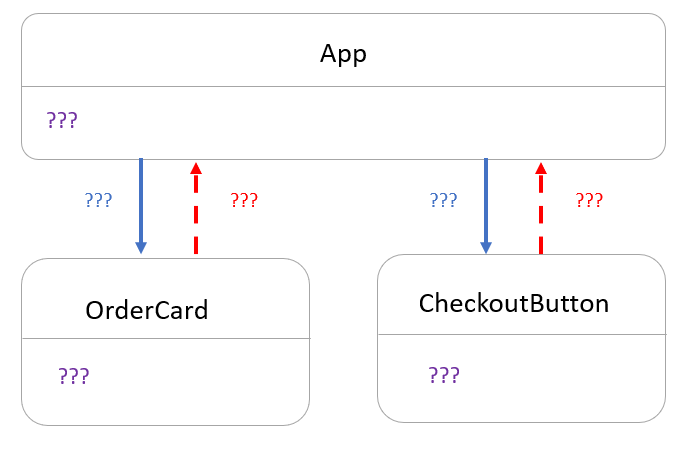
+the total price is not a state

**Q2 -** What are the interactions you plan to define between the components?

Complete the diagram: *purple=states, blue=props data, red=prop event*

* *Tips: what’s happen when you press + or – button on an item card? Which component will handle the state update?*

1. OrderCard triggers the callback (onIncrease/onDecrease)
2. App component updates the orders state
3. Changes flow down to update both OrderCard display and total in CheckoutButton



**Q3 –** Update the code to display each item using OrderCard components (name, unit price and current quantity)

*Note: For now, the + and – button do not work!*

**Q4 –** Update the code to **send the right total price** to the CheckoutButton

*Tips: It’s always good to separate our code with clear functions!*

**Q5 –** When user click on + or - the quantity of the related item must change, and also the total price

* Which state needs to be updated? Which component manages this state?

+ the orders state need to be update

+the app component manage this state.

*Tips: you don’t know how to update a state array?*

* Read again this [document](https://react.dev/learn/updating-arrays-in-state) (section Replacing items in an array)
* Understand how to use the non-mutating methods map and spread operation to perform this operation.

**Q6 –** When quantity is equal to 0, the **minus button shall be disabled** on the order card.

To effectively disable the button, we need to display it gray but also prevent any action when clicking on it.

Therefore, when quantity is equal to 0:

* The minus button background is gray (#bfbfbf)
* No action can be done when clicking on minus button