Module quiz: Components

1. When using a key for your list items, what's the best general choice? **1 / 1 ponto**

- ◯ Using an ID generated by a library that guarantees no duplications.
- ◉ Using an ID coming from the data, that points to the database ID.
- ◯ Using an index.

✓ **Correto**
That's correct, that's the preferred option.

2. Imagine you have a specification for rendering the following list item: **1 / 1 ponto**

`<li>Ice cream - 200 cal</li>`, where the name and the number of calories are coming as dynamic data. Select all the options that would correctly render the desired output:

- ☐ `<li>{Ice cream - 200 {item.cal}}</li>`
- ☑ `<li>{item.name} - {item.cal} cal</li>`

✓ **Correto**
That's correct, that would render the desired output using JSX syntax.

- ☑ `<li>{`${item.name} - ${item.cal} cal`}</li>`

✓ **Correto**
That's correct, that would render the desired output using JSX syntax and string interpolation.

3. Let's suppose you have a list of two items and a new item is added to the list. From the point of view of the React diffing algorithm, what's the most optimal position for the new element added? Select all that apply **1 / 1 ponto**

- ◯ The new element added at the beginning.
- ◯ The new element added in the 2nd position, in between the existing list items.
- ◉ The new element added at the end.

✓ **Correto**
That's correct, that's the most optimal path.

4. What are controlled components? **1 / 1 ponto**

- ◯ A set of components whose internal state is controlled by the DOM.
- ◉ A set of components whose internal state is controlled by React.
- ◯ A set of components whose value is determined by React refs.

✓ **Correto**
That's correct, state is delegated from the DOM to React local state.

5. What are the features you can still achieve with uncontrolled components? Select all that apply **1 / 1 ponto**

- ☑ One time value retrieval on submit using a React ref.

✓ **Correto**
That's correct, the ref allows you to access the internal state held by the DOM.

- ☑ Validation on submit.

✓ **Correto**
That's correct, in the submit handler function you can use refs to access all the input values and perform any custom client validation.

- [ ] Conditionally disabling the submit button.

6. When creating an API for context consumers via `useContext`, what's the argument you have to provide to the `useContext` call? **1 / 1 ponto**

- ○ The `Context.Provider` component.
- ◉ The Context object obtained via the `createContext` call.
- ○ The React state that defines the global state to be injected.

✓ **Correto**
That's correct, you always have to pass a context object as argument to the `useContext` hook.

7. Imagine the below component structure, where all components ComponentA, ComponentB and ComponentC are simple presentational components that hold no props or state: **1 / 1 ponto**

```
const App = () => {
  return(
    <AppContext.Provider>
      <ComponentA />
    </AppContext.Provider>
  );
};

const ComponentA = React.memo(() => <ComponentB />);
const ComponentB = () => <ComponentC />;
const ComponentC = () => null;
```

If the App component re-rendered for whatever reason, what would be the sequence of component re-renders that would take place?

- ○ App -> ComponentA -> ComponentB -> ComponentC.
- ○ App -> ComponentB -> ComponentC
- ◉ App

✓ **Correto**
That's correct, since ComponentA is wrapped with `React.memo`, it will prevent unnecessary updates within itself and down the tree, so only the App component would update.

8. Even though props and state are inherently different, what are areas where they overlap? Select all that apply. **1 / 1 ponto**

- ☑ Both props and state are plain JS objects.

✓ **Correto**
That's correct, they are both plain JS objects.

- ☑ Both props and state changes trigger a render update.

✓ **Correto**
That's correct, as soon as one or the other update, it will trigger a re-render.

- ☑ Both props and state are deterministic, meaning that your component always generates the same output for the same combination of props and state.

✓ **Correto**
That's correct, React components are pure functions when it comes to their rendering output.

9. When defining a JavaScript object as a piece of local React state that will be injected as context, what is the specific React hook that allows you to keep the same object reference in memory and prevent unnecessary re-renders if the object itself hasn't changed any values? **1 / 1 ponto**

- ○ `useCallback`
- ◉ `useMemo`
- ○ `useRef`

✓ **Correto**
That's correct, `useMemo` keeps in memory the previous object reference and if no changes occur in the object values, it will reuse the same reference on subsequent updates.

10. What are some possible examples of application features that are well suited to be defined as React context? Select all that apply **1 / 1 ponto**

- [x] The application theme.

  Correto
  That's correct, the theme is one example of global application state.

- [x] Locale preferences.

  Correto
  That's correct, local preferences is one example of global application state.

- [x] The current authenticated user.

  Correto
  That's correct, authenticated user is one example of global application state.

- [ ] The value of a text input.