Congratulations! You passed!

Grade received 100% **To pass** 80% or higher

Go to next item

1. Imagine you have an array with one object that represents a dessert. You would like to apply some transformation to the item to output a different structure using the `map` function as per the code below. What would be the value of the `newDesserts` variable?

1 / 1 point

```
const desserts = [
  {
    title: 'Chocolate Cake',
    description: 'Chocolate cake is a cake flavored with melted chocolate',
    calories: 500,
  }
];

const newDesserts = desserts.map((dessert) => {
  return {
    title: dessert.title.toUpperCase(),
    ...dessert,
    kCal: dessert.calories / 1000,
  };
});
```

○
```
[
  {
    title: 'CHOCOLATE CAKE',
    description: 'Chocolate cake is a cake flavored with melted chocolate',
    kCal: 0.5,
  }
]
```

◉
```
[
  {
    title: 'Chocolate Cake',
    description: 'Chocolate cake is a cake flavored with melted chocolate',
    calories: 500,
    kCal: 0.5,
  }
]
```

○
```
[
  {
    title: 'CHOCOLATE CAKE',
    description: 'Chocolate cake is a cake flavored with melted chocolate',
    calories: 500,
    kCal: 0.5,
  }
]
```

✓ **Correct**
That's correct, since the mapping output merges the previous object values after the `title` is re-defined, it has no effect and the `title` is still as before. Also, a new property is introduced, `kCal`.

2. How do you access dynamic data inside the JSX from the `render` function?

1 / 1 point

○ Using local state in the component.

○ Using component props.

[✓ Wrapping the variable in question with curly braces.]

Correct
That's correct, that's the way to access dynamic data in JSX.

3. What could be a potential problem of using a randomiser function that generates an integer number from 0 to 10 as a key for your list items, having a list of only eight items? Select all that apply

1 / 1 point

- [x] The randomiser function does not entirely guarantee that the keys it generates will be different per item and a collision could happen, having two items with the same integer as keys.

Correct
That's correct, since each value generated from the randomiser function is independent of the other, you could have key duplications.

- [x] There is no persistence of the keys generated since the moment the component re-renders the keys will vary and that could cause unexpected UI changes.

Correct
That's correct, when a re-render occurs, the randomiser could generate a different value per list item and that could cause issues with the internal state of the component.

- [ ] The randomiser function is a potential performance bottleneck since it has to run every re-render and it's an unnecessary computation.

4. The `todos` array contains a list of `todo` objects, where each object has an `id` property that is unique. Which of the following code snippets will throw a React warning when opening up the browser console? Select all that apply

1 / 1 point

- [x]

```
{todos.map((todo, index) => (
  <ToDo id={todo.id} />
))}
```

Correct
That's correct, since a key per list item is missing, React will throw a warning in the console.

- [ ]

```
{todos.map((todo, index) => (
  <ToDo key={index} id={todo.id} />
))}
```

- [ ]

```
{todos.map((todo, index) => (
  <ToDo key={todo.id} id={todo.id} />
))}
```

- [x]

```
{todos.map((todo, index) => (
  <ToDo key="myKey" id={todo.id} />
))}
```

Correct
That's correct, the keys are all the same for all items and React will throw a warning in the console regarding that problem.

5. What are the potential problems of using indexes as keys?

1 / 1 point

- [✓ If the order of items may change, that can negatively impact performance and may cause issues with component state.]
- An index is not guaranteed to be unique.
- The index is not persisted and will change the moment the component re-renders.

Correct
That's correct, indexes are discouraged when the order of the items may change.