REACT.JS

5. forduló

accenture

A kategória támogatója: Accenture

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Felhasznált idő: 01:50/15:00 **Elért pontszám:** 0/12

1. feladat 0/2 pont

Adott az alábbi App komponens és context definíció:

```
const MyContext = React.createContext({
  theme: "dark",
});

function Consumer() {
  const context = useContext(MyContext);

  return <div className={context.theme ?? "light"}>Hello</div>;
}

function App() {
  return (
    <MyContext.Provider>
      <Consumer />
    </MyContext.Provider>
  );
}
```

Mi lesz az értéke a className-nek a Consumer komponensen belüli div-nek?

Válasz

- "dark"
- undefined

- ◯ TypeError: Cannot read properties of undefined
- ◯ "light"

Magyarázat

Mivel a Provider nem adott értéket a value properynek ezért a **context** változó értéke **undefined** lesz. Ugyanis hiába adtunk meg a **createContext**-nél egy default értéket az csak akkor kerül felhasználásra, ha Consumer-t a megfelelő Provider-en *kívül* használjuk. Mivel a **contextundefined** ezért **TypeError**-t kapunk amikor a theme tulajdonságot akarjuk elérni rajta.

2. feladat 0/2 pont

Az alábbi Redux middleware-t melyik kódrészelettel kell kiegészíteni, hogy a console.log az action végrehajtása utáni állapotot írja ki?

```
function middleWare(param1) {
    return param2 => param3 => {

        //...
        console.log(afterState);

        return returnValue
    }
}
```

Válasz

- ◯
```
const returnValue = param2(param1)
const afterState = param3.getState();
```
- ◯
```
const returnValue = param2(param3)
const afterState = param1.getState();
```
- ◯
```
const returnValue = param1(param2)
const afterState = param3.getState();
```
- ◯
```
const returnValue = param3(param2)
const afterState = param1.getState();
```

Magyarázat

Egy middleware szignatúrája így néz ki: middleware: **({ getState, dispatch }) => next => action**. Ezeket ha megfeleltetjük a param1, param2, param3 paramétereknek akkor a következő kódot kapjuk, ami a helyes implementáció:

```
const returnValue = next(action);
const afterState = store.getState();
```

3. feladat 0/6 pont

Melyik állítások **NEM** igazak az **useReducer** hook-ra?

Válaszok

- [x] kötelező minden **dispatch** hívásnál megadni az actionak a type adattagot, nélküle futásidejű hibát dob a React
- [ ] a **reducer**-nek pure függvénynek (nem lehet mellékhatása) ajánlott lennie
- [x] a **dispatch** függvényt nem lehet hívni async metódusból
- [x] az **useReducer**-ot nem lehet saját hook-ban felhasználni

Magyarázat

Az "kötelező minden dispatch hívásnál megadni" válasz nem igaz, tehát a kérdésre a helyes válasz, mert nem kötelező a type property használata ez csak konvenció, a React nem ellenőrzi és nem dob hibát

A "a **reducer**-nek pure függvénynek ajánlott lennie" válasz nem igaz, tehát a kérdésre a helyes válasz: a reducer-nek pure-nak kell lennie, mert ha megváltoztatja a kapott állapot-ot akkor az hibákhoz vezethet.

A "a **dispatch** függvényt nem lehet hívni **async** metódusból" válasz helytelen: nincsen ilyen megkötés, a dispatch korlátozás nélkül hívható bárhonnan.

A "az **useReducer**-ot nem lehet saját hook-ban felhasználni" válasz nem igaz, tehát a kérdésre a helyes válasz, mert nincsen ilyen megkötés, bármely beépített hook-ot lehet saját hook-ban felhasználni.

4. feladat 0/2 pont

Az alábbi Redux store-t létrehozó kódrészletek közül melyik a helyes (nem dob kivételt és betartja a Redux konvenciókat)?

Válasz

```
const store = createStore(
  combineReducers({
    users: (state = [], action) => {
      switch (action.type) {
        case "ADD":
          return state.users.push(action.payload);
        default:
          return state;
      }
    },
  }),
  {
    users: [],
    counter: 0,
  }
);
```

```
const store = createStore(
  combineReducers({
    users: (state = [], action) => {
      switch (action.type) {
        case "ADD":
          return [...state, action.payload];
        default:
          throw new Error("Not supported");
      }
    },
  }),
  {
    users: [],
    counter: 0,
  }
);
```

```
const store = createStore(
  combineReducers({
    users: (state = [], action) => {
      switch (action.type) {
        case "ADD":
          return [...state, action.payload];
        default:
          return state;
      }
    },
  }),
  {
    users: [],
    counter: 0,
  }
);
```

```
const store = createStore(
  combineReducers({
    users: (state, action) => {
      switch (action.type) {
        case "ADD":
          return [...state, action.payload];
        default:
          return state;
      }
    },
  }),
  {
    users: [],
    counter: 0,
  }
);
```

Magyarázat

A **state.user.push(action.playload)** válasz helytelen, mivel módosítja a state értékét és nem egy új state objektumot ad vissza.

A **thow new Error("Not Supported")** válasz helytelen, mivel a combineReducers elvárja, hogy ha ismeretlen action-t kap akkor a state-el térjen vissza, ami itt nem teljesül. Ezért a **createStore** kivételt fog dobni.

A **user: (state, action) =>** válasz helytelen, mivel a combineReducers elvárja, hogy ha üres state-el hívják meg akkor a kezdeti state-t adja vissza, ami itt nem teljesül. Ezért a **createStore** kivételt fog dobni.