REACT.JS

1. forduló

accenture

A kategória támogatója: Accenture

RENDELKEZÉSRE ÁLLÓ IDŐ:

10:00

Ismertető a feladathoz

Fontos információk:

A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.

Kérjük, hogy a feladatok szövegeit **ne másold** és a böngésződ fejlesztő eszközét/ konzolját se nyisd meg feladatmegoldás közben! Mindkettő kizárást vonhat maga után.

Minden forduló után a **megoldások csütörtök reggel 8 órakor** lesznek elérhetőek.

A megoldásokkal kapcsolatos esetleges **észrevételeket a megoldások megjelenését követő kedd éjfélig** várjuk.

A több válaszlehetőségű feleletválasztós kérdéseknél járnak **részpontszámok, ha egyik rossz választ sem jelölöd be**.

Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat.

Minden feladatsornak van egy **becsült minimum megoldási ideje** (legalább a feladat elolvasási ideje). Aki ennél rövidebb idő alatt oldja meg, kizárható a versenyből.

Az első kategória után, amelynek a feladatlapját megoldod a fordulóban, kapni fogsz egy 2-3 perc alatt kitölthető **kérdőívet**. Az ezekből összeállított piackutatás legfontosabb eredményeit a díjátadót követően Veled is megosztjuk majd. Formáljuk közösen a piacot!

ELső forduló

Üdvözlünk a React of the Seas óceánjáró fedélzetén!

A következő 7 hétben együtt fogunk utazni a frontend fejlesztés tengerén.

Minden héten a React.js egy-egy nagyobb témaköre lesz a kikötő.

Felhasznált idő: 01:47/10:00 **Elért pontszám:** 0/6

1. feladat 0/1 pont

Mi lesz az App komponens által megjelenített szöveg a böngészőben?

```
class MessageCounter extends React.Component {
  render() {
    const { count } = this.props;
    return <div>{count && <h1>Messages: {count}</h1>}</div>;
  }
}

class App extends React.Component {
  render() {
    <div className="App">
      <MessageCounter count={NaN} />
    </div>;
  }
}
```

Válasz

- ○ Semmi, üres output
- ○ Messages: NaN
- ○ Futás idejű hiba: **TypeError: Conversion from 'NaN' to 'number' is not allowed.**
- ◉ NaN

Magyarázat

React-ban a hamis jellegű értékek közül csak a boolean értékek (true, false), null, undefined nem eredményeznek output-ot. A többi hamis jellegű érték mindig megjelenik az output-ban mint pl. a 0 és a NaN. Ha szeretnénk, hogy helyesen működjön a komponens akkor a **count > 0** feltételt kell nézni a komponensben:

return {count > 0 && <h1>Messages: {count}</h1>} ;

2. feladat 0/1 pont

Az alábbi komponens a következő szöveget jeleníti meg :

Value: 0 Increment: 0

Mi lesz az App komponens által megjelenített szöveg a böngészőben, miután a felhasználó egyszer klikkel az Inc gombon?

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: 0,
      increment: 0,
    };
  }

  incrementAll = () => {
    this.setState({ increment: this.state.increment + 1 });
    this.setState({ value: this.state.increment + this.state.value });
  };

  render() {
    return (
      <div>
        Value: {this.state.value} Increment: {this.state.increment}
        <button onClick={this.incrementAll}>Inc</button>
      </div>
    );
  }
}
```

Válasz

- ○ **Value: 0 Increment: 0**
- ◯ **Value: 0 Increment: 1**
- ○ **Value: 1 Increment: 0**
- ○ **Value: 1 Increment: 1**

Magyarázat

A setState metódus aszinkron, ezért a második hívásnál még nincs beállítva a this.state.increment az új értékre, ezért a value 0 marad. A setState metódus rendelkezik egy másik overload-al ebben az esetben egy függvényt vár, ahol megkapjuk az aktuális state értékét, így a helyesen működő kód így nézne ki:

```
this.setState(state => ({ value: state.increment + state.value }))
```

3. feladat 0/2 pont

Melyik állítások igazak az alábbiak közül a **props**-re?

Válaszok

- ☑ Nem szabad az **this.props** adattagjait felülírni mert ez futás idejű hibát eredményez
- ☐ Csak egyszerű típusú értékei lehetnek: strings, numbers, booleans, undefined és null

- [x] Lehetőség van arra, hogy megváltozása esetén a komponens ne renderelődjön újra
- [ ] A **props** neveinek tetszőleges, érvényes JavaScript változó nevet adhatunk

Magyarázat

A "Csak egyszerű típusú értékei lehetnek" válasz hamis, mert bármilyen összetett típusú props-ok is megengedettek pl. Object, tömb, és függvény.

A "**props** neveinek tetszőleges, érvényes JavaScript változó nevet adhatunk" válasz hamis, mert kettő speciális props név nem használható: a **key** és a **ref**.

4. feladat 0/2 pont

Milyen állítások igazak az alábbiak közül a komponensek életciklus eseményeire?

Válaszok

- [x] A **render** metódus nem tartalmazhat mellékhatást és DOM manipulációt
- [x] Életciklus eseményei csak osztály (class) típusú komponenseknek lehetnek, funkció (function) típusú komponenseknél Hook-kat kell használni
- [ ] A **componentDidMount** megfelelő hely AJAX hívások indítására, de a DOM-ot nem módosíthatja
- [ ] A **componentWillUnmount** egy elavult esemény és helyette a **componentDidUnmount** eseményt kell használni

Magyarázat

A "A **componentDidMount** megfelelő hely AJAX hívások indítására, de a DOM-ot nem módosíthatja" válasz hamis, mert DOM módosításra is lehetőség van a **componentDidMount**.

A "A **componentWillUnmount** egy elavult esemény és helyette a **componentDidUnmount** eseményt kell használni" válasz hamis, mert **componentDidUnmount** esemény nem létezik, ezért a **componentWillUnmount**-t kell használni.