

REACT

2. forduló



A kategória támogatója: TCS - Tata Consultancy
Services

Ismertető a feladathoz

Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Badge-eket** a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/10:00

Elért pontszám: 0/6

1. feladat 0/2 pont

Az alábbi komponens mit jelenít meg...

```
function Welcome(props) {  
  return <h1>Hello, {props.age + 1}</h1>;  
}
```

...ha így hivatkozunk rá:

```
<Welcome />
```

Válasz

- ☐ hibaüzenetet kapunk "Uncaught TypeError: Cannot read property."
- ☐ Hello,



Hello, NaN

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A **props** paraméter mindig értelmezve van, akkor is ha attribútumok nélkül, "üresen" hívjuk meg a komponenst.

Ezért a "hibaüzenetet kapunk "Uncaught TypeError: Cannot read property."" válasz **helytelen**.

Amennyiben "üresen" hívjuk meg a komponenst a **props.age** értéke **undefined** lesz amihez 1-at adva **NaN** kapunk:

```
undefined + 1 = NaN
```

Ezért a "Hello, " válasz **helytelen**.

És a "Hello, NaN" válasz **helyes**.

2. feladat 0/2 pont

Miért **nem** működik az alábbi komponens jól?

```
function DummyButton(props) {  
  return <button onClick={props.handler()}>Click</button>;  
}  
  
export default function App() {  
  const handler = () => {  
    console.log('working...')  
  }  
  return (  
    <>  
      <DummyButton handler={handler} />  
    </>  
  );  
}
```

Válaszok



A komponens jól működik



mert meghívjuk a **handler** függvényt ***props.handler()***

Ez a válasz helyes, de nem jelölted meg.



mert nincs ***() =>***-ben a ***props.handler()*** hívás

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A click eseményre történő feliratkozásnál az alábbi szintaxis (**props.handler()**) függvényhíváshoz vezet:

```
onClick={props.handler()}
```

Ezért amikor a komponens render-elődik, a handler függvény kiértékelődik és meghívódik, cserében a gombra történő kattintás nem fog működni.

Ezért a "A komponens jól működik" válasz **helytelen**.

Az **onClick** attribútum egy függvény referenciát vár, hogy helyesen működjön a komponens.

Ezért a "mert meghívjuk a **handler** függvényt ``props.handler()``" válasz helyes.

Illetve egy lehetséges javítási lehetőség, ha egy anonim függvénybe csavarjuk a hívást.

Ezért a "mert nincs `() =>` -ben a ``props.handler()`` hívás" válasz helyes.

3. feladat 0/2 pont

Adott az alábbi 2 komponens:

```
function CompA() {
  const [isSignedIn, setIsSignedIn] = useState(false);
  if (localStorage.getItem('signed-in') === 'yes') {
    setIsSignedIn(true)
  }
  return (<div>{String(isSignedIn)}</div>)
}
```

és

```
function CompB() {
  const [isSignedIn, setIsSignedIn] = useState(false);
  useEffect(() => {
    if (localStorage.getItem('signed-in') === 'yes') {
      setIsSignedIn(true)
    }
  }, [])
  return (<div>{String(isSignedIn)}</div>)
}
```

Az alábbi állítások közül melyek **igazak**?

Válaszok

- ☐ A 2 komponens egyformán viselkedik
- ☒ A 2 komponens csak akkor viselkedik egyformán, ha **``signed-in``** értéke nem **yes**
Ez a válasz helyes, de nem jelölted meg.
- ☐ Az első komponens végtelen ciklusba futhat, ha **``signed-in``** értéke nem **yes**
- ☒ Az első komponens végtelen ciklusba futhat, ha **``signed-in``** értéke **yes**
Ez a válasz helyes, de nem jelölted meg.
- ☐ A második komponensnél a **``useEffect``** használata felesleges

Magyarázat

A két komponens nem viselkedik egyformán.

Az első komponensben a **`useState()`** hívás mindig meghívódik, ha a **`signed-in`** értéke **yes**, ebben az esetben a komponens végtelen render ciklusba kerül. Amikor a **`signed-in`** értéke nem **yes** abban az esetben nem történik **`useState()`** hívás ezért az **`isSignedIn`** értéke **false** marad.

A második komponensben az üres függőségi listával meghívott **`useEffect`** hook csak a komponens legelső mount-oláskor fut le, ezért a benne meghívott **`useState()`** setter is maximum egyszer hívódik meg. Ezért az **`isSignedIn`** változó értéke a **`signed-in`** értékétől függően **true** vagy **false** lesz.

Ezért a

"A 2 komponens egyformán viselkedik" válasz helytelen.

"A 2 komponens csak akkor viselkedik egyformán, ha *signed-in* értéke nem yes" válasz helyes

"Az első komponens végtelen ciklusba futhat, ha *signed-in* értéke nem yes" válasz helytelen.

"Az első komponens végtelen ciklusba futhat, ha *signed-in* értéke yes" válasz helyes.

"A második komponensnél a *useEffect* használata" felesleges válasz helytelen.



[Legfontosabb tudnivalók](#)

[Kapcsolat](#)

[Versenyszabályzat](#)

[Adatvédelem](#)

© 2023 Human Priority Kft.

KÉSZÍTETTE 

Megjelenés

 Világos 