

Test 1:

Create a button which is pulsating – growing, then shrinking, then growing again (height and width changing) etc - from min size to max size (defined).

- The button size updates every second
- The button reverses direction automatically when max height or max width is reached (whichever first)
- User can click the button to change the direction at any time(if it was growing, after click it will be shrinking and the other way around)
- Opposite direction to what the button is doing must be displayed in the button as an indicator
- Styles are not important, but some are good.
- Time to complete: 15-20 min

```
import React, { useEffect, useState } from "react";

const MIN_WIDTH = 40;
const MIN_HEIGHT = 20;
const MAX_WIDTH = 200;
const MAX_HEIGHT = 100;
const STEP = 10;

export default function App() {
  return (
    <div style={/* center the button on the page */}
      <PulsatingButton />
    </div>
  );
}

const PulsatingButton = () => {
  // write your code here
}
```

Test 2:

Display the list of users from the API as a list of UI cards with users data in it.

- API endpoint: <https://randomuser.me/api?results=number>
- Arbitrary simple design of your choice
- Use Flexbox model
- Time to complete: 20-30 min

Test 3:

Given a nested object of an unknown depth, render the keys and values with indent and keys in bold like presented below. Object data is in the code.

Example final result:

gender: male

name:

title: Mr

first: Bastião

last: Nogueira

...

- Arbitrary simple design (indent, padding, colors etc)
- Without using JSON.stringify()
- Time to complete: 15 min

```
import React from "react";

const data = {
  gender: "male",
  name: {
    title: "Mr",
    first: "Bastião",
    last: "Nogueira"
  },
  location: {
    street: {
      number: 5109,
      name: "Rua São Paulo "
    },
    city: "Vespasiano",
    state: "Amapá",
    country: "Brazil",
    postcode: 62751,
    coordinates: {
      latitude: "-76.4479",
      longitude: "-155.4439"
    },
  },
}
```

```
    timezone: {
      offset: "0:00",
      description: "Western Europe Time, London, Lisbon, Casablanca"
    }
  },
  email: "bastiao.nogueira@example.com",
  login: {
    uuid: "2a68cf19-1703-4490-9f4b-98fdb147a960",
    username: "crazypanda622",
    password: "viagra",
    salt: "hBdmsLbY",
    md5: "e1f4a3af328eae964a052382c6a9e5dc",
    sha1: "26ca4cb3132e779d2cb2d1d9c020ac9e3899023b",
    sha256:
"45dd8922afcdcaf7fa3881d98808e060d847de4f1e89fad81cd5e4e99aebb68e"
  },
  dob: {
    date: "1973-03-03T22:26:36.701Z",
    age: 49
  },
  registered: {
    date: "2022-04-22T08:28:48.821Z",
    age: 0
  },
  phone: "(71) 2465-3078",
  cell: "(93) 0334-0189",
  id: {
    name: "CPF",
    value: "361.632.104-22"
  },
  picture: {
    large: "https://randomuser.me/api/portraits/men/85.jpg",
    medium: "https://randomuser.me/api/portraits/med/men/85.jpg",
    thumbnail: "https://randomuser.me/api/portraits/thumb/men/85.jpg"
  },
  nat: "BR"
};
```

```
function App() {
  return (
    <div style={{ margin: "auto", width: "80%", paddingTop: 50 }}>
      <NestedObject data={data} />
    </div>
  );
}

const NestedObject = ({ data }) => {
  // write your code here
}
```

Test 4:

The modal dialog is supposed to accept user input such as name, surname and email, and by clicking Add button the list of all added users is display below.

- Complete the application by creating a reducer and its actions to support the functionality by using useReducer() instead of useState()
- The final solution is not necessarily shorter or better than using useState, the point is to test redux knowledge.
- Time to complete: 20-30 min

Final result:

Add user

Add

List of users:

Add user

Add

List of users:

Email: evgenij.koronin@digib.com

Full Name: Evgenij Koronin

```
import React, { useReducer } from "react";
```

```
function App() {
```

```
  return (
```

```
    <div
```

```
      style={{
```

```
        display: "flex",
```

```
        justifyContent: "center",
```

```
        paddingTop: 50
```

```
      }}
    >
```

```
      <UserForm />
```

```
    </div>
```

```
  );
```

```
}
```

```
const UserForm = () => {
```

```
  // useReducer();
```

```
  const onSubmit = (e) => {
```

```
    // write your code here
```

```
  };

```

```
  return (
```

```
    <div>
```

```
      <form
```

```
        style={{
```

```
          display: "flex",
```

```
          flexDirection: "column",
```

```
          border: "solid",
```

```
          padding: 10
```

```
        }}
      <form
        onSubmit={onSubmit}
      >
```

```
        <h3>Add user</h3>
```

```
        <input
```

```
          value={form.email}
```

```
          type="text"
```

```

        onChange={}
        style={{ marginBottom: 5 }}
        placeholder="Email"
      />
      <input
        value={form.name}
        type="text"
        onChange={}
        style={{ marginBottom: 5 }}
        placeholder="Name"
      />
      <input
        value={form.surname}
        type="text"
        onChange={}
        style={{ marginBottom: 5 }}
        placeholder="Surname"
      />

      <button style={{ alignSelf: "center" }} onClick={onSubmit}>
        Add
      </button>
    </form>
    <br />
    <b>List of users: </b>
    // accounts here
  </div>
);
};

export default App;

```

Test 5:

The current application shows a list of users online and offline, their status randomly changes every 5 sec. The app is not optimized and suffers from props drilling.

- The aim is to refactor the app to use the React Context so that all app components would not receive any props but rather use the context

Example:

```
<UserList />
<UserStatus />
<ActionButton />
```

- The phrase “and I’m all alone” should be displayed against the users if he/she is the only one online
- The status should display the number of users online
- Time to complete: 15-20 min

Final result:

Nicolas: OFFLINE

Mary: OFFLINE

Julia: OFFLINE

John: ONLINE and I'm all alone...

Jorge: OFFLINE

There are currently 1 users online

Randomize now!

```
import React, { useEffect, useState } from "react";

function App() {
  const [users, setUsers] = useState({
    Nicolas: true,
    Mary: true,
    Julia: true,
    John: true,
    Jorge: true
  });
}
```

```

const randomizeOneUser = (users) => {
  const names = Object.keys(users);
  const random = Math.floor(Math.random() * names.length);
  const newUsers = { ...users };
  newUsers[names[random]] = !users[names[random]];
  return newUsers;
};

const randomizeUsers = () => {
  setUsers(randomizeOneUser);
};

useEffect(() => {
  const interval = setTimeout(randomizeUsers, 5000);
  return // ???
}, [users]);

return (
  <>
    <UserList users={users} />
    <UserStatus users={users} />
    <ActionButton action={randomizeUsers} />
  </>
);
}

const UserList = ({ users }) => (
  <div style={{ padding: 20 }}>
    {Object.keys(users).map((key) => (
      <User name={key} status={users[key]} users={users} />
    ))}
  </div>
);

const User = ({ name, status, users }) => (
  <p>
    `{${name}: ${status ? "ONLINE" : "OFFLINE"}}` {"and I m all
alone..."} /* update this */
  </p>

```



```
);

const UserStatus = ({ users }) => <p>There are currently X users
online</p>; // update status with a real number

const ActionButton = ({ action }) => (
  <button type="text" onClick={action}>
    Randomize now!
  </button>
);

export default App;
```

Test 6: (optional)

Create a HOC component withUser() which injects user prop into the conference Badge, which displays the user's name.

- The badge is supposed to work also without the HOC
- Time to complete: 10 min

```
const currentUser = {
  name: "Michael"
};

// write our code here

const Badge = ({ user, salutation }) => {
  return (
    <main style={{ border: "solid 1px blue", borderRadius: "20px",
textAlign: "center" }}>
      <header>
        <h1>{salutation}</h1>
        <p>My Name Is</p>
      </header>
      <div>
        <p style={{ fontSize: "32pt", fontWeight: "bold"
}}>{user.name}</p>
      </div>
    </main>
  );
};
```

```
};  
  
const BadgeWithUser = withUser(currentUser)(Badge);  
  
const Application = () => <BadgeWithUser salutation="Good day" />;  
  
export default Application;
```