Lab 09: React Basics

Objective(s):

1. Learn JSX
2. Learn Rendering Elements
3. Components & Props
4. State & Lifecycle Methods

Lab Task(s):

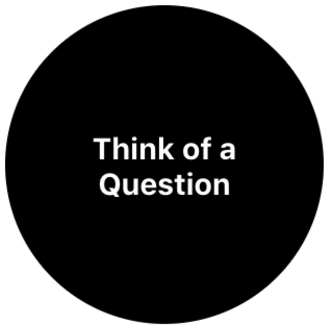
Exercises

1. Rebuild the Giphy API exercise (see Lab#8Exercise#8) with React.
   Once you finished rebuilding this application, include a "gif of the day", which should be a random gif generated, when the page loads. You will need to use a component life cycle method for this.
2. Create two components, **TodoList** and **Todo**. The TodoList component should contain an array called todos. The TodoList component should be responsible for listing all of the Todo components, which should each display the task necessary to complete.

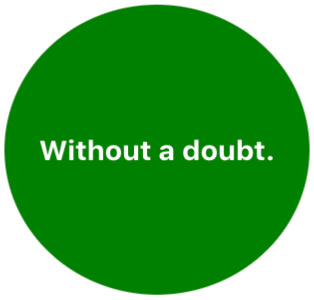
   You have full control of what you would like the UI of this application to be but think carefully about the components you will need as well as the props and state.

3. In this exercise, you'll build a simulation of a classic kid's toy, the Magic Eight Ball.

   This should appear as a black ball that initially reads "Think of a Question", like this:



When you click on the ball, it should choose a random answer & the matching color for that answer. For example, after clicking, it might look like this:



Step One: *EightBall* Component

This application will consist of two components:

App

A simple component that just renders an ***EightBall*** component.

EightBall

The component for the magic eight ball.

The ***EightBall*** should take a single property, ***answers***, which should be an array of objects, with each object having a key for ***msg*** and ***color***. For example, to use the answers from the classic commercial product, you could use these:

```
[
  { msg: "It is certain.", color: "green" },
  { msg: "It is decidedly so.", color: "green" },
  { msg: "Without a doubt.", color: "green" },
  { msg: "Yes - definitely.", color: "green" },
  { msg: "You may rely on it.", color: "green" },
  { msg: "As I see it, yes.", color: "green" },
  { msg: "Most likely.", color: "green" },
  { msg: "Outlook good.", color: "green" },
  { msg: "Yes.", color: "green" },
  { msg: "Signs point to yes.", color: "goldenrod"
},
  { msg: "Reply hazy, try again.", color:
"goldenrod" },
  { msg: "Ask again later.", color: "goldenrod" },
  { msg: "Better not tell you now.", color:
"goldenrod" },
  { msg: "Cannot predict now.", color: "goldenrod"
},
  { msg: "Concentrate and ask again.", color:
"goldenrod" },
  { msg: "Don't count on it.", color: "red" },
  { msg: "My reply is no.", color: "red" },
  { msg: "My sources say no.", color: "red" },
  { msg: "Outlook not so good.", color: "red" },
  { msg: "Very doubtful.", color: "red" },
]
```

The ***EightBall*** will need state to keep track of the current color and message text, and this state should initially be "black" and "Think of a Question".

Make it so that the ball chooses a random message when it is clicked on. This should change the background color of the ball and the message.

Further Study 1

Reset

Add a button below the ball that will reset the ball back to its initial state.

Record Keeping

As you ask questions to the ball, display counts of the number of times the eight ball shows up as each of the three different colors.

Further Study 2: Color Boxes

For this part, you should show a series of 16 boxes (a box is just square div with a background color). At the bottom of all of the boxes should be a button labeled "Change".

Initially, each box should have a background color chosen from a random list of colors.

When you click the button:

- it should select a random box
- it should change the background color of that random box to a new color from the possible colors list.

Before building anything, *think about the structure of your React app.* This entire thing shouldn't be one giant component.

Further Study 3

Default Properties

For both parts, there are opportunities to move some things into default properties:

- the list of possible colors for boxes
- the number of boxes (so it doesn't always have to be 16!)

Feedback on Changed Box

For the color boxes app, it can be tricky to tell which box changed when you clicked (particularly when it picks the same random color, so you can't see any difference!)

Change the application so that, when a box just changed, it shows "changed!" inside of the div. That text should go away after the next click.