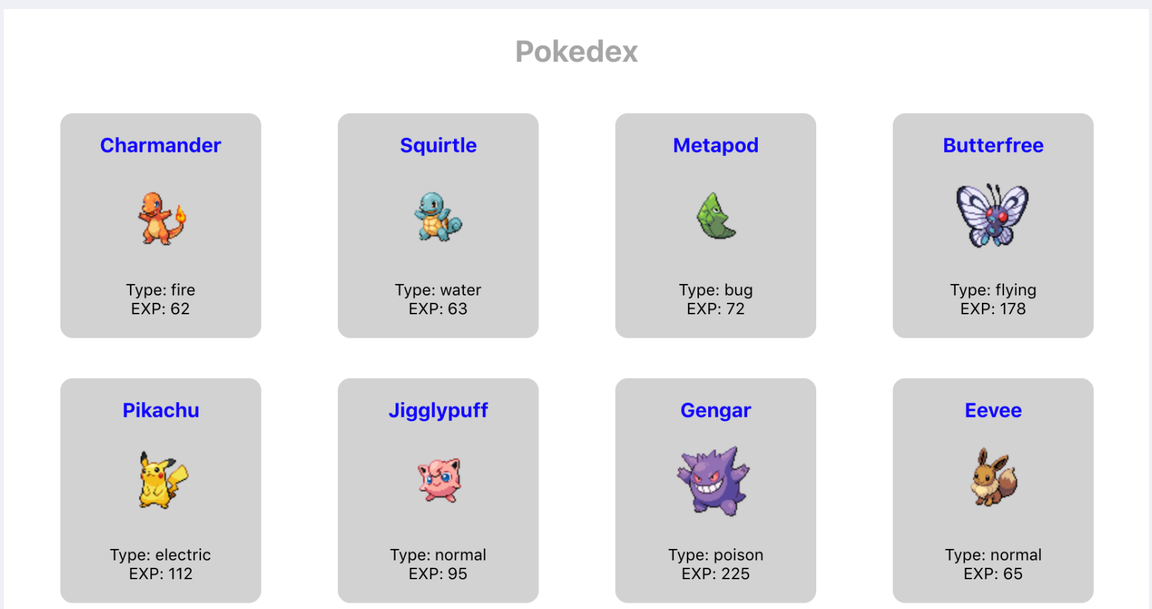
React Props: Pokedex

This exercise lets you pratice using React components and properties.

Create a pokemon application (a “pokedex”) that displays an interface that looks like this:



To create the pokedex, you should use 3 components:

App

> This should just render a single Pokedex.
>
> (It’s common for the top-level app to not have direct logic in it, but to render the top application object — this becomes useful when you build sites that compose several different parts together.)

Pokecard

> Shows a single Pokemon, with their name, image, and type.

Pokedex

> Is provided, via props, an array of objects describing different pokemon, and renders a sequence of ***Pokecard*** components.

Use the defaultProps feature of ***Pokedex*** to provide a default list of Pokemon characters to show. You can use this list for a good set of defaults:

```
[
  {id: 4, name: 'Charmander', type: 'fire', base_experience: 62},
  {id: 7, name: 'Squirtle', type: 'water', base_experience: 63},
  {id: 11, name: 'Metapod', type: 'bug', base_experience: 72},
  {id: 12, name: 'Butterfree', type: 'flying', base_experience: 178},
  {id: 25, name: 'Pikachu', type: 'electric', base_experience: 112},
  {id: 39, name: 'Jigglypuff', type: 'normal', base_experience: 95},
  {id: 94, name: 'Gengar', type: 'poison', base_experience: 225},
  {id: 133, name: 'Eevee', type: 'normal', base_experience: 65}
]
```

For each pokemon, their image source should be:
https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/${id}.png.

Further Study: More Pokemon!

Pokegame Component

Modify your component hierarchy so that ***App*** renders a component called ***Pokegame***. ***Pokegame*** should take your list of 8 pokemon and randomly assign them into two hands of 4 cards each. It should then render two ***Pokedex*** components, one for each hand.

Once you’ve got this working, modify your ***Pokegame*** so that it also calculates the total experience for each hand of pokemon. It should pass this total to the ***Pokedex***.

Next, have the ***Pokegame*** component determine which hand is the “winner,” where the winning hand is the one with the higher total experience. Then modify the ***Pokedex*** component one more time so that it accepts a prop of ***isWinner***. If the ***Pokedex*** is the winning one, it should display the message “THIS HAND WINS!” at the bottom of the deck.

Now when you load the page, you should see two different hands with a randomly changing winner every time you refresh.

Styling

Add styling to your components — perhaps you can do interesting things when hovering over a Pokecard, or have them smoothly transition into the page, all via CSS?

Further Study: Basic Blackjack!

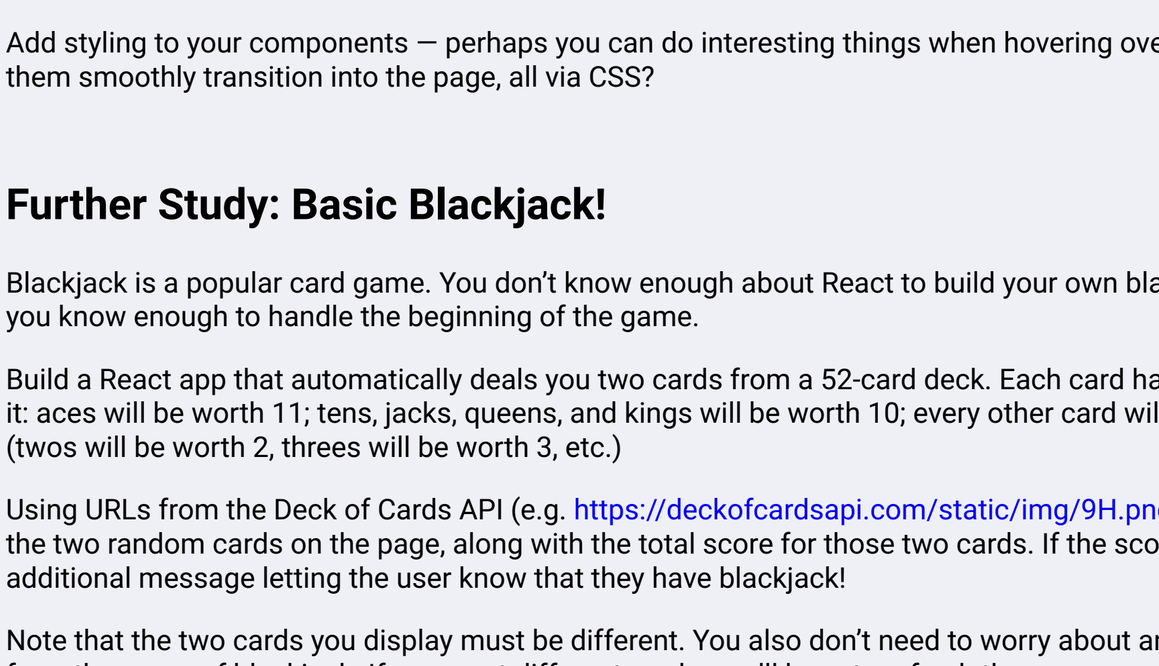
Blackjack is a popular card game. You don’t know enough about React to build your own blackjack game yet, but you know enough to handle the beginning of the game.

Build a React app that automatically deals you two cards from a 52-card deck. Each card has a score assigned to it: aces will be worth 11; tens, jacks, queens, and kings will be worth 10; every other card will be worth its value (twos will be worth 2, threes will be worth 3, etc.)

Using URLs from the Deck of Cards API (e.g. https://deckofcardsapi.com/static/img/9H.png), show images of the two random cards on the page, along with the total score for those two cards. If the score is 21, show an additional message letting the user know that they have blackjack!

Note that the two cards you display must be different. You also don’t need to worry about any other functionality from the game of blackjack. If you want different cards, you’ll have to refresh the page.

Here’s an image of what the app might look like:



Spend some time talking with your partner about the component design for this application, if you decide to give it a shot.

Solution

View our Solution