

Eksamensdokument_gruppe 18

Link til GitHub-repositoriet

<https://github.com/Flowerafro/PokedexGR18>

Sikte på karakter

Vi har siktet på å oppfylle kravene for karakter C. Vi har også lagt til litt ekstra funksjonalitet utover minstekravet for karakter C.

Student-alias fra Github

MiriamKy - Miriam Kyrre

Flowerafro - Line Henriksen

AndreMooore - Andre Moore

Redegjørelser/forutsetninger

- I tillegg til komponenter fra minstekrav, skal applikasjonen ha komponentene
 - PokeCard, som viser et pokemonkort
 - TypeCard, som viser et typekort
- Et pokemonkort i home, type og søkeresultatet skal kunne klikkes på (enten hele spillkortet, eller med en knapp/lenke til et spill). Tips/hint: pokemon navnet kan brukes som slug og sendes med videre som en prop på PokeCard slik at man kan bruke det i en ny fetch for mer data inne på komponentet. Når linken åpnes, brukes Pokemon komponentet og skal fremvise informasjon om pokemonen som ble trykket på.
- Søkefunksjonen i menyen skal fungere, man skal kunne søke etter navnet til en pokemon og når man trykker på søk knappen skal man bli sendt til søkeresultat siden som enten viser resultatet av hva man søkte etter eller en tekststreng med beskjeden «Det du søker etter finnes ikke». Når man trykker på resultatet fra denne siden og linken åpnes, brukes Pokemon komponenten og man kommer til siden med informasjon om den gjeldene pokemonen.

Vi opplevde den markerte teksten i screenshot-en over som litt uklar. At pokemonkortet skulle kunne klikkes på "(enten hele spillkortet, eller med en knapp/lenke til et spill)". Her tolker vi at D-kravet gav oss valget mellom å route til pokemon-komponentet, eller lage en knapp som linker til et spill, hentet fra Api-et. Vi valgte å route til pokemon-komponentet ved klikk på hele knappen, fordi vi sikter på karakter C, der pokemon-komponentet har krav til innhold fra PokeApi. Det finnes heller ikke noen slik "knapp med lenke til et spill" i skissene eller videodemo, og som det står i oppgaveteksten:

Skissene i denne mappen er ment som veiledende og for en forståelse av noen av kravene i oppgaven. Dette inkluderer mer (men ikke mindre), mer/utfyllende funksjonalitet etc. PS: Fokuser på å oppnå kravene for hver karakteroppgave før dere legger til ekstra innhold/funksjonalitet. Kravene nevnt i oppgaven må oppnås for å nå ønsket karakter.

Påbegynt, men ikke inkludert funksjonalitet

1) Tomt søk

I tillegg til kravet om at det skal dukke opp en feilmelding dersom søket ikke stemmer overens med pokemon-navn i API, så ønsket vi å kode inn en feilmelding dersom søket er tomt. Siden det ikke var et krav i oppgaveteksten ble det skjøvet ned på prioriteringslisten og til slutt rakk vi ikke å implementere denne funksjonaliteten. Vi tenkte å legge inn en if-statement på selve handleSubmit-funksjonen eller på selve getResult-funksjonen, som ville sjekke om input-feltet var tomt. Vi rakk litt feilsøking/testing, men det ble aldri godt nok, så vi prioriterte dette vekk før innlevering.

2) Fjerne dokument “Pokemon” i Sanity Studio:

I Sanity studio hadde vi først lagt det opp slik at vi hadde to dokumenter; teams og pokemon. Tanken var at “teams” inneholdt informasjon om Team Valor, Team Instinct og Team Mystic, mens pokemon skulle inneholde informasjon om pokemon som tilhørte hvert team. Til dette oppsettet tenkte vi å bruke “reference”. Når vi begynte å sette opp schemaTypes så endte vi opp med å referere til pokemon i en array inne i dokumentet for teams. I siste liten så vi at vi ikke hadde fått fjernet det ubrukte dokumentet “Pokemon” fra Sanity Studio, ennå det var fjernet fra schema-Types i prosjektet vårt.

Kilder:

VS Code Copilot

De 3 neste screenshot-ene er fra VS Code Copilot hører til pokemon-komponentet. Se kommentar direkte i komponentet for hvordan forslaget ble brukt.

Prompt:

```

    <article className="pokemon-abilities">
      <h1>Abilities</h1>
      {itAll.abilities?.map((ability, index) => (
        <div key={index}>
          <h2>{ability.ability.name.charAt(0).toUpperCase() + ability.ability.name.slice(1)}</h2>
          <p>{ability.description}</p>
        </div>
      ))}
    </article>

  </>
  })
</section>
)}

```

In the description part, i want both the "Effect" and "Short effect" to show

Svar fra Copilot:

Vi endte med å ikke bruke denne funksjonen, fordi vi synes den virket mer komplisert enn den trenger å være, men den gav oss en retning:

To show both the "Effect" and "Short effect", you need to modify the `getItAll` function to fetch both pieces of data from the `abilityData`. Here's how you can do it:

```

const getItAll = () => {
  fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`)
    .then(response => response.json())
    .then(data => {
      const abilities = data.abilities;
      const promises = abilities.map(ability =>
        fetch(ability.ability.url)
          .then(response => response.json())
          .then(abilityData => {
            const effectEntry = abilityData.effect_entries.find(entry => entry.language.name === 'en');
            const effect = effectEntry.effect;
            const shortEffect = effectEntry.short_effect;
            return { ...ability, description: { effect, shortEffect } });
          })
      );
      Promise.all(promises)
        .then(abilitiesWithDescriptions => {
          setItAll({ ...data, abilities: abilitiesWithDescriptions });
        });
    });
}

```

```

<article className="pokemon-abilities">
  <h1>Abilities</h1>
  {itAll.abilities?.map((ability, index) => (
    <div key={index}>
      <h2>{ability.ability.name.charAt(0).toUpperCase() + ability.ability.name.s
      <p>Effect: {ability.description.effect}</p>
      <p>Short Effect: {ability.description.shortEffect}</p>
    </div>
  ))}
</article>

```

Team.jsx:

I dette komponentet vises PokeCard med navn, bilde og pokemonnummer fra som hentes fra API basert på navnet Pokemon har i Sanity. Koden skrevet her er basert på metoden vi brukte i Home.jsx, som igjen er veiledet av chat.AI.

Her er linken til chatten som hjalp Home.jsx og Team.jsx (ligger også som kommentar i Home.jsx):

<https://chat.openai.com/share/c9fa4901-bfc8-474c-b193-d48614373831>

SearchResults.jsx - loading

I dette komponentet har vi lagt inn en "loading state" for å imøtekomme kravet om feilmelding dersom et Pokemon-navn som søkes etter ikke finnes. For å få til dette var vi nødt til å lage en state til selve innlastingen av API-fetchen som er tilknyttet søket, som oppdateres når søket ikke gir resultater. Vi gjorde lignende i "boksøk-arbeidskravet" og da brukte vi denne siden som veiledning:

<https://rapidapi.com/guides/loading-state-react>

Kopi av Bing Chat:

Type-komponentet:

Du

```
import { useParams } from "react-router-dom" import { useState, useEffect } from "react"
export default function Type() { const {slug} = useParams() const [typeSlug, setTypeSlug] =
useState([]) const [pokemonUrl, setPokemonUrl] = useState([]) const getTypeSlug = async ()
=> { fetch(`https://pokeapi.co/api/v2/type/${slug}?limit=10`) .then(response =>
response.json()) .then(data => { setTypeSlug(data.pokemon); data.pokemon.forEach(poke =>
getTypeData(poke.pokemon.url)); }) .catch(error => console.error(error)) } const
getTypeData = async (url) => { fetch(`${url}?limit=1`) //Får mange objekter i konsollen, og
får ikke ut noe info, så må arbeides videre med :) .then(response => response.json())
.then(data => setPokemonUrl(prevState => [...prevState, data])) .catch(error =>
console.error(error)) } useEffect(() => { getTypeSlug() }, []) console.log(slug +
"pokemonUrl" + pokemonUrl) // Gjøre en ny fetch i dette komponentet som bare henter ut de
pokemon som stemmer med slugen som representerer typen som er klikket // Prøver meg her
på å hente verdien i slugen, men har ikke helt funnet ut av det ennå... return ( <>
{pokemonUrl?.map((pokeUrl) => { return ( {pokeUrl.pokemon.index.map((i) => <h2
key={i}>{i.pokemon.name}</h2>)} ) }} ) } </> ) }
```

Hei, jeg prøver å få tilgang på og skrive ut verdiene fra pokeAPI. Jeg har gjort 2 fetch som jeg samler i en state, og det er denne staten jeg skal bruke for å skrive ut verdiene. Jeg bare sliter med å få det til, kan du gi meg noen tips?

Copilot

Hei! Jeg ser at du prøver å hente data fra PokeAPI og skrive ut verdiene. Her er noen tips som kan hjelpe deg:

1. **Sjekk datastrukturen:** Først og fremst, sørg for at du forstår datastrukturen som returneres fra API-et. Du kan bruke `console.log(data)` rett etter at du har hentet data for å se hvordan dataene er strukturert.
2. **Oppdatering av tilstand:** Når du oppdaterer tilstanden med `setPokemonUrl(prevState => [...prevState, data])`, legger du til hele dataobjektet i arrayen. Hvis du bare vil ha spesifikk informasjon, som navnet på Pokémon, kan du endre dette til `setPokemonUrl(prevState => [...prevState, data.name])`.
3. **Rendering av data:** Når du prøver å render dataene, ser det ut til at du prøver å få tilgang til `pokeUrl.pokemon.index`. Hvis `pokemonUrl` er en array av Pokémon-navn, kan du endre dette til å bare være `pokeUrl` og deretter render hvert element i arrayen.

Her er en oppdatert versjon av koden din med disse endringene:

```
export default function Type() {

  const {slug} = useParams()

  const [pokemonNames, setPokemonNames] = useState([])

  useEffect(() => {

    fetch(`https://pokeapi.co/api/v2/type/${slug}?limit=10`)

      .then(response => response.json())

      .then(data => {

        data.pokemon.forEach(poke => {

          fetch(poke.pokemon.url)

            .then(response => response.json())

            .then(data => setPokemonNames(prevState =>
[...prevState, data.name]))

            .catch(error => console.error(error))

        });

      })

      .catch(error => console.error(error))

  }, [slug])
```

```

    return (
      <>
        {pokemonNames.map((name, index) => <h2 key={index}>{name}</h2>)}
      </>
    )
  }
}

```

Håper dette hjelper! Lykke til med prosjektet ditt. 😊

Du

Når jeg konsoll-logger data får jeg feilmelding at den er undefined?

Copilot

Det kan være flere grunner til at du får en `undefined` feilmelding når du prøver å konsoll-logge `data`. Her er noen mulige årsaker og løsninger:

1. **API-kallet mislykkes:** Hvis fetch-kallet ditt ikke returnerer data som forventet, vil `data` være `undefined`. Sørg for at API-endepunktet er riktig og at det returnerer data som forventet. Du kan legge til en `.catch` blokk etter `.then` for å fange eventuelle feil som oppstår under fetch-kallet.
2. **Asynkronitet:** JavaScript er asynkront, noe som betyr at koden din ikke nødvendigvis kjører i den rekkefølgen du forventer. Hvis du prøver å logge `data` før fetch-kallet er fullført, vil det være `undefined`. Sørg for at du logger `data` inne i `.then` blokken, etter at du har mottatt responsen.
3. **Datastruktur:** Hvis `data` er et objekt, og du prøver å få tilgang til en egenskap som ikke eksisterer, vil du få `undefined`. Sørg for at du forstår strukturen til dataene som returneres fra API-et, og at du får tilgang til egenskapene på riktig måte.

Hvis du fortsatt har problemer, kan du dele mer informasjon om feilen, så kan jeg hjelpe deg bedre. 😊

Du

```

const {slug} = useParams() const [typeSlug, setTypeSlug] = useState([]) const [pokemonUrl,
setPokemonUrl] = useState([]) const getTypeSlug = async () => {
  fetch(`https://pokeapi.co/api/v2/type/${slug}?limit=10`) .then(response => response.json())
  .then(data => { setTypeSlug(data.pokemon); data.pokemon.forEach(poke =>
    getTypeData(poke.pokemon.url)); }) .catch(error => console.error(error)) } const
  getTypeData = async (url) => { fetch(`${url}?limit=1`) .then(response => response.json())
    .then(data => setPokemonUrl(prevState => [...prevState, data])) .catch(error =>

```

```
console.error(error)) } useEffect(() => { getTypeSlug() }, []) // console.log(slug +  
"pokemonUrl" + pokemonUrl) console.log(pokemonUrl)
```

Denne koden gir en array med 186 objekter, selv om jeg prøver å sette en grense på hvor mange den skal hente ut i fetch, ved å skrive `limit=10` og `limit=1`. Men der ser ikke ut til at det fungerer? Hvorfor kan det skje?

Copilot

Hei! Det ser ut til at det er en misforståelse om hvordan `limit` parameteren fungerer i PokeAPI.

Når du bruker `limit=10` i det første API-kallet, begrenser du antall typer Pokémon som returneres. Men når du gjør et nytt API-kall for hver Pokémon type, henter du all informasjon om den spesifikke Pokémon, uavhengig av `limit` parameteren.

I det andre API-kallet ditt, `fetch(`${url}?limit=1`)`, ser det ut til at du prøver å begrense informasjonen som returneres for hver Pokémon. Men `limit` parameteren har ingen effekt her fordi du allerede henter data for en spesifikk Pokémon via dens unike URL. Derfor returnerer API-et all informasjon om den spesifikke Pokémon, uavhengig av `limit` parameteren.

Hvis du vil begrense antall Pokémon som vises, kan du gjøre det ved å endre hvordan du oppdaterer `pokemonUrl` tilstanden. For eksempel, du kan legge til en sjekk for å bare oppdatere tilstanden hvis lengden på `pokemonUrl` er mindre enn det ønskede antallet.

Håper dette hjelper! 😊

Du

```
fetch(`https://pokeapi.co/api/v2/type/${slug}?limit=10`)
```

Jeg tror ikke jeg begrenser antall typer som returneres her, da `slug` henter verdien i url-en, som i dette tilfellet er navnet på en type, og skal gjøre at det blir gjort en fetch på alle pokemon som stemmer med typen. Deretter må jeg gjøre en ny fetch for å få tilgang på mer informasjon per pokemon som blir fetch-et. Dette gjør jeg ved å fette på url-verdien i det første kallet. Kan det være en annen grunn til at jeg ikke har klart å begrense antallet objekter i arrayen som kommer ut?

Copilot

Beklager for misforståelsen. Du har helt rett. `slug` i din kode representerer typen Pokémon, og `limit=10` i det første API-kallet ditt begrenser antall Pokémon av den typen som returneres, ikke antall typer.

Når det gjelder det andre API-kallet ditt, `fetch(`${url}?limit=1`)`, så vil `limit=1` ikke ha noen effekt her, fordi du henter data for en spesifikk Pokémon via dens unike URL. Dette

API-kallet returnerer all informasjon om den spesifikke Pokémon, uavhengig av `limit` parameteren.

Hvis du får flere objekter i arrayen enn forventet, kan det være fordi hver Pokémon har mer enn ett objekt med data. Hvis du vil begrense antall objekter i arrayen, kan du prøve å velge bare de dataene du trenger fra hvert Pokémon-objekt, i stedet for å legge til hele objektet i arrayen.

For eksempel, hvis du bare er interessert i Pokémon-navnet, kan du endre

```
setPokemonUrl(prevState => [...prevState, data]) til setPokemonUrl(prevState => [...prevState, data.name]).
```

Håper dette hjelper! 😊

Pokemon-komponentet

Her har flere states og funksjoner fått nye navn etter chatten med Bing.

Du

```
const {slug} = useParams() const {itAll, setItAll} = useState([]) const getItAll = async () =>
{ fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`) .then(response => response.json())
.then(data => { setItAll(data.pokemon) console.log(data.pokemon) }) .catch(error =>
console.error(error)) } console.log(itAll)
```

This fetch does not give me an array, but it is saying undefined in the console. Can it be the way i am putting the data into the itAll-state by calling the setItAll-function and fetching data.pokemon?

Copilot

The issue seems to be with how you're trying to access the data from the fetch response. The `data.pokemon` property might not exist in the returned data, which is why it's undefined.

The PokeAPI returns a pokemon object directly, not under a `pokemon` property. So, you should be setting the state with the entire `data` object, not `data.pokemon`.

Here's how you can modify your code:

JavaScript

```
const {slug} = useParams()

const [itAll, setItAll] = useState([])

const getItAll = async () => {

    fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`)
```



```

    .then(response => response.json())

    .then(data => {

        setItAll(data) // set the entire data object

        console.log(data) // log the entire data object

    })

    .catch(error => console.error(error))

}

console.log(itAll)

```

Kode generert av kunstig intelligens. Se gjennom og bruk med forsiktighet. [Mer informasjon om vanlige spørsmål.](#)

Now, `itAll` should contain the data for the pokemon specified by `slug`. Remember that `itAll` is an object, not an array. If you want to access a specific property of the pokemon, you can do so with `itAll.propertyName`. For example, `itAll.name` will give you the pokemon's name.

I hope this helps! Let me know if you have any other questions. 😊

Du

Okay, youre saying tha `itAll` is an object, but is it right that it is placed in an array since i declare the state like this: `useState([])`? If not, can tou explain what the square brackets are doing to my code here?

Copilot

Good question! In the context of your code, `useState([])` is initializing `itAll` as an empty array. However, when you fetch data from the API and set the state with `setItAll(data)`, you're actually replacing the initial empty array with the data object you fetched. So, `itAll` becomes an object, not an array.

The square brackets in `const [itAll, setItAll] = useState([])` are using a feature of JavaScript called destructuring assignment. This allows you to unpack values from arrays or properties from objects into distinct variables.

In the case of `useState`, it returns an array with two elements: the current state value and a function to update it. By writing `[itAll, setItAll]`, you're assigning the first element of the array to `itAll` and the second element to `setItAll`.

So, to clarify, the square brackets are not making `itAll` an array. They are used for destructuring the array returned by `useState`.

If you want `itAll` to be an object, you should initialize it as an empty object like this: `useState({})`. But in your case, since you're fetching a single pokemon, it makes sense for `itAll` to be an object, not an array. So, you can keep your code as it is.

I hope this clears up your confusion! Let me know if you have any other questions. 😊

Du

```
import { useParams } from "react-router-dom" import { useEffect, useState } from "react"
import TypeCard from "../TypeCard" export default function Pokemon() { const {slug} =
useParams() const [itAll, setItAll] = useState([]) const getItAll = async () => {
fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`) .then(response => response.json())
.then(data => { setItAll(data); data.forEach(ability => getAbilities(ability.url)); }) .catch(error
=> console.error(error)) } const getAbilities = async (url) => { fetch(url) .then(response =>
response.json()) .then(data => setItAll(prevState => [...prevState, data])) .catch(error =>
console.error(error)) } useEffect(() => { getItAll() }, [slug]) console.log("itAll" + itAll) return (
<section id="pokemon"> {itAll.sprites && ( <> <section> <article
className="pokemon-img"> <h1>{itAll.name.charAt(0).toUpperCase() +
itAll.name.slice(1)}</h1> <img src={itAll.sprites.other?.['official-artwork'].front_default}
alt={itAll.name} /> </article> <article className="pokemon-abilities"> <h1>Abilities</h1>
{itAll.abilities?.map((ability, index) => ( <div key={index}>
<h2>{ability.ability.name.charAt(0).toUpperCase() + ability.ability.name.slice(1)}</h2>
<p>Effect: {ability.description.effect}</p> <p>Short Effect:
{ability.description.shortEffect}</p> </div> ))) </article> </section> <section> <article
className="pokemon-type"> <h1>Types</h1> {itAll.types?.map((type, index) =>
(<TypeCard key={index} name={type.type.name} />))} </article> <article
className="pokemon-stats"> <h1>Stats</h1> {itAll.stats?.map((stat, index) => ( <ul
key={index} className="stats"> <li>{stat.stat.name.charAt(0).toUpperCase() +
stat.stat.name.slice(1)}: </li> <progress value={stat.base_stat} max="100"></progress>
<li>{stat.base_stat}</li> </ul> ))) </article> </section> </> )} </section> )} }
```

I have changed the fetch in this component, so it does not communicate with the return and mapping of the html-print. How can i use this fetch to print map and print it?

Copilot

It seems like you're trying to fetch the abilities of a Pokemon and then display them. However, there's a mistake in your code. The `forEach` method is being called on `data`, but `data` is not an array, it's an object.

The abilities can be found in `data.abilities`, which is an array. Each ability is an object with a `url` property that you can use to fetch more details about the ability. However, the `getAbilities` function is currently updating the entire `itAll` state, which is not what you want. You want to update only the abilities part of the state.

Here's how you can modify your code:

```

import { useParams } from "react-router-dom"

import { useEffect, useState } from "react"

import TypeCard from "../TypeCard"

export default function Pokemon() {

  const {slug} = useParams()

  const [pokemon, setPokemon] = useState(null)

  const [abilities, setAbilities] = useState([])

  const getPokemon = async () => {

    fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`)

      .then(response => response.json())

      .then(data => {

        setPokemon(data);

        data.abilities.forEach(ability =>
getAbilityDetails(ability.ability.url));

      })

      .catch(error => console.error(error))

  }

  const getAbilityDetails = async (url) => {

    fetch(url)

      .then(response => response.json())

      .then(data => setAbilities(prevState => [...prevState, data]))

      .catch(error => console.error(error))

  }

  useEffect(() => {

    getPokemon()

  }, [slug])

  console.log("Pokemon: ", pokemon)

  console.log("Abilities: ", abilities)

```

```
// Rest of your component...

}
```

In this code, I've separated the state into `pokemon` and `abilities`. The `getPokemon` function fetches the basic Pokemon data and sets it to the `pokemon` state. It also initiates fetching of the details for each ability. The `getAbilityDetails` function fetches the details for a single ability and adds it to the `abilities` state.

Now, you can map over `abilities` in your JSX to display the details of each ability. Remember to adjust the rest of your component to use the new `pokemon` and `abilities` states. For example, replace `itAll.sprites` with `pokemon.sprites`, `itAll.abilities` with `abilities`, and so on.

Please note that the abilities fetched from the ability URL might not have a `description` property. You might need to adjust your code based on the actual structure of the ability data.

I hope this helps! Let me know if you have any other questions.

Du

```
export default function Pokemon() { const {slug} = useParams() const [details, setDetails] =
useState([]) const [abilities, setAbilities] = useState([]) const getDetails = async () => {
fetch('https://pokeapi.co/api/v2/pokemon/${slug}/') .then(response => response.json())
.then(data => { setDetails(data); data.abilities.forEach(ability =>
getAbilities(ability.ability.url)); }) .catch(error => console.error(error)) } const getAbilities =
async (url) => { fetch(url) .then(response => response.json()) .then(data =>
setAbilities(prevState => [...prevState, data])) .catch(error => console.error(error))
useEffect(()=> { getDetails() }, [slug]) console.log("Details" + details) return ( <section
id="pokemon"> {details.sprites && ( <> <section> <article className="pokemon-img">
<h1>{details.name.charAt(0).toUpperCase() + details.name.slice(1)}</h1> <img
src={Details.sprites.other?.['official-artwork'].front_default} alt={Details.name} /> </article>
<article className="pokemon-abilities"> <h1>Abilities</h1>
{details.abilities?.map((ability, index) => ( <div key={index}>
<h2>{ability.ability.name.charAt(0).toUpperCase() + ability.ability.name.slice(1)}</h2>
<p>Effect: {ability.description.effect}</p> <p>Short Effect:
{ability.description.shortEffect}</p> </div> ))} </article> </section> <section> <article
className="pokemon-type"> <h1>Types</h1> {details.types?.map((type, index) =>
(<TypeCard key={index} name={type.type.name} />))} </article> <article
className="pokemon-stats"> <h1>Stats</h1> {details.stats?.map((stat, index) => ( <ul
key={index} className="stats"> <li>{stat.stat.name.charAt(0).toUpperCase() +
stat.stat.name.slice(1)}: </li> <progress value={stat.base_stat} max="100"></progress>
<li>{stat.base_stat}</li> </ul> ))} </article> </section> </> )} </section> )} }
```

I have tried to modify my code manually with your advice, but can you adjust the rest of the component to use the details and abilities states? Also, feel free if i overlooked something elsewhere in the code :)

Copilot

Sure, I can help with that. Here's how you can adjust your component to use the details and abilities states:

```
import { useParams } from "react-router-dom"

import { useEffect, useState } from "react"

import TypeCard from "../TypeCard"

export default function Pokemon() {

  const {slug} = useParams()

  const [details, setDetails] = useState(null)

  const [abilities, setAbilities] = useState([])

  const getDetails = async () => {

    fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`)

      .then(response => response.json())

      .then(data => {

        setDetails(data);

        data.abilities.forEach(ability => getAbilities(ability.ability.url));

      })

      .catch(error => console.error(error))

  }

  const getAbilities = async (url) => {

    fetch(url)

      .then(response => response.json())

      .then(data => setAbilities(prevState => [...prevState, data]))

      .catch(error => console.error(error))

  }

  useEffect(() => {
```

```

    getDetails()

    }, [slug])

    console.log("Details: ", details)

    console.log("Abilities: ", abilities)

    return (

      <section id="pokemon">

        {details && (

          <>

            <section>

              <article className="pokemon-img">

                <h1>{details.name.charAt(0).toUpperCase() +
details.name.slice(1)}</h1>

                <img
src={details.sprites.other?.['official-artwork'].front_default}
alt={details.name} />

              </article>

              <article className="pokemon-abilities">

                <h1>Abilities</h1>

                {abilities.map((ability, index) => (

                  <div key={index}>

                    <h2>{ability.name.charAt(0).toUpperCase() +
ability.name.slice(1)}</h2>

                    <p>Effect: {ability.effect_entries[0].effect}</p>

                    <p>Short Effect:
{ability.effect_entries[0].short_effect}</p>

                  </div>

                )))}

              </article>

            </section>

            <section>

```

```

        <article className="pokemon-type">

        <h1>Types</h1>

        {details.types?.map((type, index) => (<TypeCard key={index}
name={type.type.name} />))}

        </article>

        <article className="pokemon-stats">

        <h1>Stats</h1>

        {details.stats?.map((stat, index) => (

        <ul key={index} className="stats">

                <li>{stat.stat.name.charAt(0).toUpperCase() +
stat.stat.name.slice(1)}: </li>

                <progress value={stat.base_stat} max="100"></progress>

                <li>{stat.base_stat}</li>

        </ul>

        )))}

        </article>

</section>

</>

)}

</section>

)

}

```

In this code, I've made the following changes:

- I've changed the initial state of `details` from an empty array to `null`. This is because the initial state should match the type of the state you're expecting to set later. In this case, you're expecting `details` to be an object, not an array.
- In the JSX, I've replaced `details.abilities` with `abilities` since you're now storing the abilities separately in the `abilities` state.
- I've adjusted the way you're accessing the ability name and effects based on the structure of the ability data. The ability name is now accessed directly as `ability.name`, and the effects are accessed from `ability.effect_entries[0]`.

- Please note that the `effect_entries` array might contain entries for multiple languages. In this code, I'm assuming that the first entry is in English. If this is not the case, you might need to filter the `effect_entries` array to find the entry for the desired language.

I hope this helps! Let me know if you have any other questions.

Du

```
export default function Pokemon() { const {slug} = useParams() const [details, setDetails] =
useState() const [abilities, setAbilities] = useState([]) const getDetails = async () => {
fetch('https://pokeapi.co/api/v2/pokemon/${slug}/') .then(response => response.json())
.then(data => { setDetails(data); data.abilities.forEach(ability =>
getAbilities(ability.ability.url)); }) .catch(error => console.error(error)) } const getAbilities =
async (url) => { fetch(url) .then(response => response.json()) .then(data =>
setAbilities(prevState => [...prevState, data])) .catch(error => console.error(error))
useEffect(() => { getDetails() }, [slug]) console.log("Details" + details) return ( <section
id="pokemon"> {details.sprites && ( <> <section> <article className="pokemon-img">
<h1>{details.name.charAt(0).toUpperCase() + details.name.slice(1)}</h1> <img
src={details.sprites.other?.['official-artwork'].front_default} alt={Details.name} /> </article>
<article className="pokemon-abilities"> <h1>Abilities</h1> {abilities?.map((ability, index)
=> ( <div key={index}> <h2>{ability.name.charAt(0).toUpperCase() +
ability.name.slice(1)}</h2> <p>Effect: {ability.effect_entries.effect}</p> <p>Short Effect:
{ability.effect_entries.shortEffect}</p> </div> ))} </article> </section> <section> <article
className="pokemon-type"> <h1>Types</h1> {details.types?.map((type, index) =>
(<TypeCard key={index} name={type.type.name} />))} </article> <article
className="pokemon-stats"> <h1>Stats</h1> {details.stats?.map((stat, index) => ( <ul
key={index} className="stats"> <li>{stat.stat.name.charAt(0).toUpperCase() +
stat.stat.name.slice(1)}: </li> <progress value={stat.base_stat} max="100"></progress>
<li>{stat.base_stat}</li> </ul> ))} </article> </section> </> )} </section> )} } This is now my
updated code. Can you run through it and check if it works? And i dont get anything in the
console.log??
```

Copilot

I'm sorry for any confusion, but as an AI text-based model, I don't have the capability to run or test code. However, I can certainly help you debug and improve your code!

Looking at your updated code, there are a few issues that might be causing problems:

1. The `useEffect` hook is currently inside the `getAbilities` function, which is not correct. It should be at the same level as your other hooks (`useState`, `useParams`).
2. In your `console.log("Details" + details)`, you're trying to concatenate an object (`details`) with a string, which will result in `[object Object]`. Instead, you should pass the object as a separate argument to `console.log`: `console.log("Details", details)`.

3. In your JSX, you're using `Details` instead of `details` in some places. JavaScript is case-sensitive, so `Details` and `details` are considered different variables.

Here's how you can fix these issues:

```
import { useParams } from "react-router-dom"

import { useEffect, useState } from "react"

import TypeCard from "./TypeCard"

export default function Pokemon() {

  const {slug} = useParams()

  const [details, setDetails] = useState()

  const [abilities, setAbilities] = useState([])

  const getDetails = async () => {

    fetch(`https://pokeapi.co/api/v2/pokemon/${slug}/`)

      .then(response => response.json())

      .then(data => {

        setDetails(data);

        data.abilities.forEach(ability => getAbilities(ability.ability.url));

      })

      .catch(error => console.error(error))

  }

  const getAbilities = async (url) => {

    fetch(url)

      .then(response => response.json())

      .then(data => setAbilities(prevState => [...prevState, data]))

      .catch(error => console.error(error))

  }

  useEffect(() => {

    getDetails()

  }, [slug])

}
```

```

console.log("Details", details)

return (
  <section id="pokemon">

    {details?.sprites && (
      <>

        <section>

          <article className="pokemon-img">

            <h1>{details.name.charAt(0).toUpperCase() +
details.name.slice(1)}</h1>

            <img
src={details.sprites.other?.['official-artwork'].front_default}
alt={details.name} />

          </article>

          <article className="pokemon-abilities">

            <h1>Abilities</h1>

            {abilities?.map((ability, index) => (
              <div key={index}>

                <h2>{ability.name.charAt(0).toUpperCase() +
ability.name.slice(1)}</h2>

                <p>Effect: {ability.effect_entries[0].effect}</p>

                <p>Short Effect:
{ability.effect_entries[0].short_effect}</p>

              </div>

            ))}

          </article>

        </section>

        <section>

          <article className="pokemon-type">

            <h1>Types</h1>

            {details.types?.map((type, index) => (<TypeCard key={index}
name={type.type.name} />))}


```

```

</article>

<article className="pokemon-stats">

<h1>Stats</h1>

{details.stats?.map((stat, index) => (

<ul key={index} className="stats">

    <li>{stat.stat.name.charAt(0).toUpperCase() +
stat.stat.name.slice(1)}: </li>

    <progress value={stat.base_stat} max="100"></progress>

    <li>{stat.base_stat}</li>

</ul>

))}

</article>

</section>

</>

)}

</section>

)

}

```

I hope this helps! Let me know if you have any other questions.

Schema-types og props til pokeCard:

Du

Jeg har en schema-type med teams som dette:

```

export const teams = { title: "Teams", name: "teams", type: "document", fields: [ { title: "Tittel",
name: "title", type: "string" }, { title: "Bilde", name: "image", type: "image" }, { title: "Slug",
name: "slug_url", type: "slug", options: { source: "title", slugify: input =>
input.toLowerCase().replace(/s+/g, '-') } }, { title: "Pokemon", name: "pokemon", type:
"array", of: [{type: "pokemon"}] } ] }

```

... og en schema type med pokemon som dette:

```

export const pokemon = { title: "Pokemon", name: "pokemon", type: "document", fields: [ {
title: "Navn", name: "name", type: "string" }, { title: "Nummer", name: "number", type:
"number" }, ] }

```

Hvordan kan jeg hente ut en array med pokemon som tilhører teamet stemmer med slugen i url på komponentet?

Copilot

For å hente ut en array med Pokémon som tilhører et team basert på slugen i URL-en, kan du følge disse trinnene:

1. Hent teamet basert på slugen:

- Først må du finne det aktuelle teamet ved å bruke slugen fra URL-en. Du kan gjøre dette ved å søke i databasen eller ved å bruke en funksjon som tar slugen som parameter og returnerer det tilsvarende teamet.

2. Hent Pokémon-arrayen fra det valgte teamet:

- Når du har teamet, kan du få tilgang til Pokémon-arrayen. I ditt tilfelle er Pokémon-arrayen en del av teamet og er definert som `teams.pokemon`.
- Du kan hente ut Pokémon-arrayen ved å bruke `team.pokemon` der `team` er objektet du har hentet fra databasen.

Her er et eksempel i JavaScript (Node.js) som viser hvordan du kan implementere dette:

JavaScript

```
// Anta at du allerede har hentet teamet basert på slugen fra URL-en
```

```
const team = /* Hent teamet fra databasen eller annen kilde */;
```

```
// Hent Pokémon-arrayen fra teamet
```

```
const pokemonArray = team.pokemon;
```

```
// Nå har du tilgang til Pokémon-arrayen og kan jobbe med den videre
```

```
console.log(pokemonArray);
```

Kode generert av kunstig intelligens. Se gjennom og bruk med forsiktighet. [Mer informasjon om vanlige spørsmål](#).

Husk å tilpasse koden til ditt rammeverk eller teknologistack. Dette eksempelet gir deg en generell idé om hvordan du kan hente ut Pokémon-arrayen basert på teamets slug i URL-en.



Du

Jeg bruker react og skal bruke fetch-funksjon for å hente dataene

Copilot

Flott! Når du bruker `fetch`-funksjonen i React, kan du hente data fra en ekstern API eller en backend-server. Her er et eksempel på hvordan du kan bruke `fetch` for å hente Pokémon-data basert på teamets slug i URL-en:

JavaScript

```
import React, { useState, useEffect } from 'react';
function TeamComponent({ match }) {
  const [team, setTeam] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // Hent teamet basert på slugen fra URL-en
    const slug = match.params.slug; // Anta at du har tilgang til slugen via React Router
    fetchTeamBySlug(slug);
  }, [match.params.slug]);

  const fetchTeamBySlug = async (slug) => {
    try {
      const response = await fetch(`/api/teams/${slug}`); // Endre URL til din
      backend-endepunkt
      if (response.ok) {
        const data = await response.json();
        setTeam(data);
      } else {
        console.error('Feil ved henting av teamdata');
      }
    } catch (error) {
      console.error('Noe gikk galt:', error);
    } finally {
      setLoading(false);
    }
  };

  if (loading) {
    return <p>Laster inn...</p>;
  }

  if (!team) {
    return <p>Teamet ble ikke funnet.</p>;
  }

  // Hent Pokémon-arrayen fra teamet
  const pokemonArray = team.pokemon;
```

```

    return (
      <div>
        <h1>{team.title}</h1>
        <ul>
          {pokemonArray.map((pokemon) => (
            <li key={pokemon.name}>{pokemon.name}</li>
          ))}
        </ul>
      </div>
    );
  }
}
export default TeamComponent;

```

Kode generert av kunstig intelligens. Se gjennom og bruk med forsiktighet. [Mer informasjon om vanlige spørsmål](#).

I dette eksempelet antar jeg at du har en backend-server som gir deg teamdata basert på slugen. Du kan tilpasse URL-en og datastrukturen etter ditt behov. Lykke til med implementasjonen! 😊

Du

Ok, du skal få mer kontekst. Schema typene representerer strukturen i Sanity, der jeg har lagret noe av dataene som skal hentes. Basert på pokemon-navnene som hentes i sanity, skal jeg gjøre et API-kall eller fetch som henter detaljer om disse pokemon. Så skal jeg sende props til et annet komponent kalt pokeCard, med informasjon om disse detaljene, som skal være navn, bilde og pokemon-nummer.

Copilot

Takk for den ekstra konteksten! Det gir meg et bedre bilde av hva du prøver å oppnå. La oss bryte ned oppgaven i mindre trinn:

1. Hent Pokémon-navn fra Sanity:

- Først må du hente Pokémon-navnene fra Sanity-databasen. Dette kan gjøres ved å lage en forespørsel til Sanity API eller bruke en backend-server som har tilgang til dataene.
- Du kan hente Pokémon-navnene som en liste eller et array.

2. Gjør et API-kall for å hente Pokémon-detalljer:

- Når du har Pokémon-navnene, kan du bruke dem til å gjøre et API-kall (for eksempel til en Pokémon API) for å hente detaljer om hver Pokémon.
- API-kallet bør returnere informasjon som navn, bilde-URL og Pokémon-nummer.

3. Send props til PokeCard-komponenten:

- Når du har Pokémon-detaljene, kan du sende dem som props til PokeCard-komponenten.
- PokeCard-komponenten kan deretter bruke disse propsene til å vise informasjonen, inkludert navn, bilde og Pokémon-nummer.

Her er et eksempel på hvordan du kan implementere dette i React:

JavaScript

```
// TeamComponent.js
import React, { useState, useEffect } from 'react';
import PokeCard from './PokeCard'; // Anta at du har en PokeCard-komponent
function TeamComponent({ match }) {
  const [pokemonNames, setPokemonNames] = useState([]);
  const [loading, setLoading] = useState(true);
  useEffect(() => {
    // Hent Pokémon-navnene fra Sanity eller annen kilde
    fetchPokemonNames();
  }, []);
  const fetchPokemonNames = async () => {
    try {
      // Gjør en API-forespørsel til Sanity eller backend-serveren
      // for å hente Pokémon-navnene
      const response = await fetch('/api/pokemon/names'); // Endre URL etter behov
      if (response.ok) {
        const data = await response.json();
        setPokemonNames(data);
      } else {
        console.error('Feil ved henting av Pokémon-navn');
      }
    } catch (error) {
      console.error('Noe gikk galt:', error);
    } finally {
      setLoading(false);
    }
  };
  if (loading) {
    return <p>Laster inn...</p>;
  }
  return (
    <div>
      {pokemonNames.map((pokemonName) => (
```

```

        <PokeCard key={pokemonName} name={pokemonName} />
      )}
    </div>
  );
}

```

export default TeamComponent;

Kode generert av kunstig intelligens. Se gjennom og bruk med forsiktighet. [Mer informasjon om vanlige spørsmål](#).

I **PokeCard**-komponenten kan du bruke props som **name**, **imageUrl** og **pokemonNumber** for å vise informasjonen for hver Pokémon. Husk å tilpasse koden til din spesifikke implementasjon og API-endepunkter. Lykke til! 😊

Du

Hva gjør jeg hvis API-et ikke tillater å fetche flere pokemon enn en om gangen?

Copilot

Hvis API-et ikke tillater å hente flere Pokémon samtidig, kan du følge denne tilnærmingen for å hente Pokémon-detalljer én etter én:

1. Iterer gjennom Pokémon-navnene:

- Hent Pokémon-navnene fra Sanity eller annen kilde.
- Lag en løkke (for eksempel en **map**-funksjon) som itererer gjennom hvert Pokémon-navn.

2. Gjør individuelle API-kall for hver Pokémon:

- Inne i løkken, for hvert Pokémon-navn:
 - Gjør et API-kall for å hente detaljer om den spesifikke Pokémonen.
 - Lagre informasjonen (navn, bilde-URL, Pokémon-nummer) for hver Pokémon.

3. Send props til PokeCard-komponenten:

- For hver Pokémon, send de hentede detaljene som props til PokeCard-komponenten.
- PokeCard-komponenten kan deretter vise informasjonen for hver Pokémon.

Her er et eksempel på hvordan du kan implementere dette:

JavaScript

```

// TeamComponent.js
import React, { useState, useEffect } from 'react';
import PokeCard from './PokeCard'; // Anta at du har en PokeCard-komponent

```



```

function TeamComponent({ match }) {
  const [pokemonNames, setPokemonNames] = useState([]);
  const [loading, setLoading] = useState(true);
  useEffect(() => {
    // Hent Pokémon-navnene fra Sanity eller annen kilde
    fetchPokemonNames();
  }, []);
  const fetchPokemonNames = async () => {
    try {
      // Gjør en API-forespørsel til Sanity eller backend-serveren
      // for å hente Pokémon-navnene
      const response = await fetch('/api/pokemon/names'); // Endre URL etter behov
      if (response.ok) {
        const data = await response.json();
        setPokemonNames(data);
      } else {
        console.error('Feil ved henting av Pokémon-navn');
      }
    } catch (error) {
      console.error('Noe gikk galt:', error);
    } finally {
      setLoading(false);
    }
  };
  if (loading) {
    return <p>Laster inn...</p>;
  }
  return (
    <div>
      {pokemonNames.map((pokemonName) => (
        <PokeCard key={pokemonName} name={pokemonName} />
      ))}
    </div>
  );
}
export default TeamComponent;

```

Kode generert av kunstig intelligens. Se gjennom og bruk med forsiktighet. [Mer informasjon om vanlige spørsmål.](#)

I `PokeCard`-komponenten kan du bruke props som `name`, `imageUrl` og `pokemonNumber` for å vise informasjonen for hver Pokémon. Husk å tilpasse koden til din spesifikke implementasjon og API-endepunkter. Lykke til! 😊