Introduction to React: API Usage and React Hooks

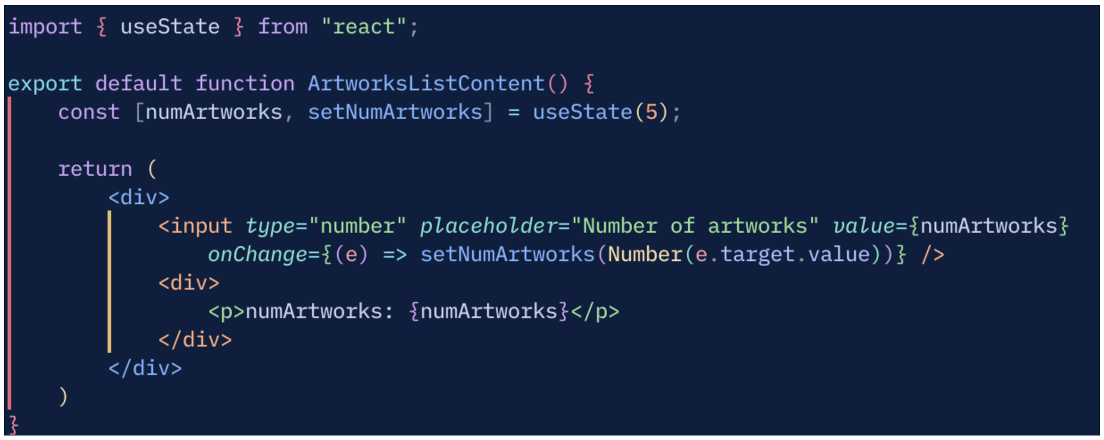
In this discussion we will be creating a React application.We will cover how to use React hooks to manage state and how to retrieve and display from an API endpoint.

Task 1: Initialize the project

1. Follow last weeks instructions to initialize a new React project with the same options via your preferred method, and open the project with your preferred editor

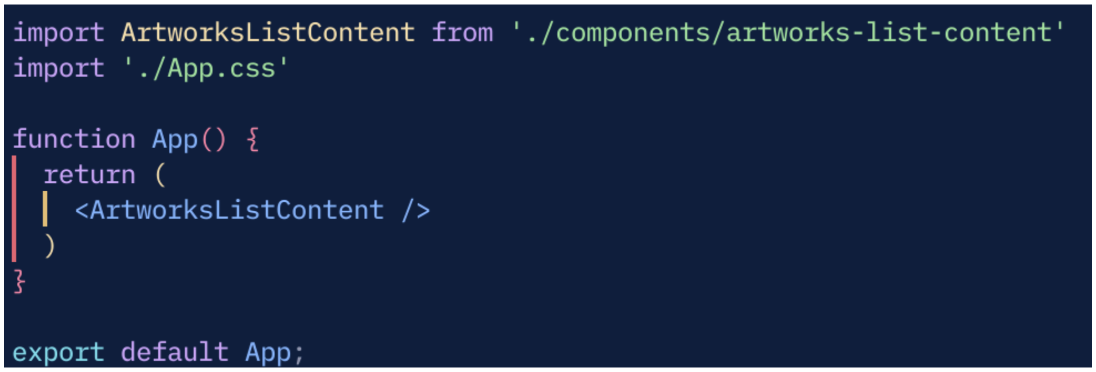
Task 2: Create a component to display pieces of artwork

1. Create a new directory with the path **src/components** and create a file within that directory called **artworks-list-content.tsx**
   a. Ultimately, the file should have the path **src/components/artworks-list-content.tsx**
2. Now, fill in some initial content to be displayed by this component
   a. Here we define a variable and its setter using the useState hook. We also display the state of the variable.



```tsx
import { useState } from "react";

export default function ArtworksListContent() {
    const [numArtworks, setNumArtworks] = useState(5);

    return (
        <div>
            <input type="number" placeholder="Number of artworks" value={numArtworks}
                onChange={(e) => setNumArtworks(Number(e.target.value))} />
            <div>
                <p>numArtworks: {numArtworks}</p>
            </div>
        </div>
    )
}
```

3. Edit the **App.tsx** file to import and display this component

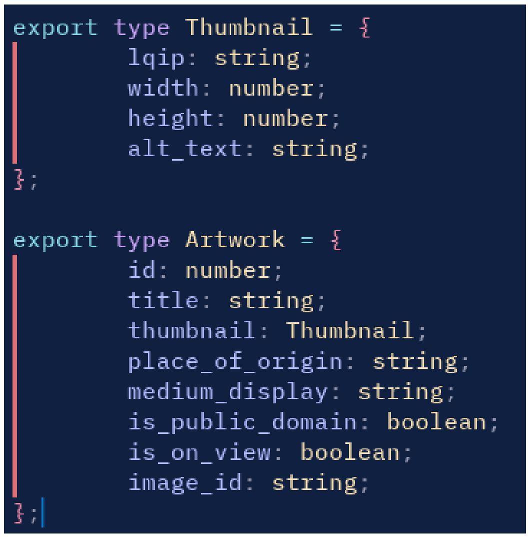


```tsx
import ArtworksListContent from './components/artworks-list-content'
import './App.css'

function App() {
  return (
    <ArtworksListContent />
  )
}

export default App;
```

4. Run **npm run dev** in your terminal and confirm everything is working
   a. When you visit the page, you should see a number and it should change as you input different values

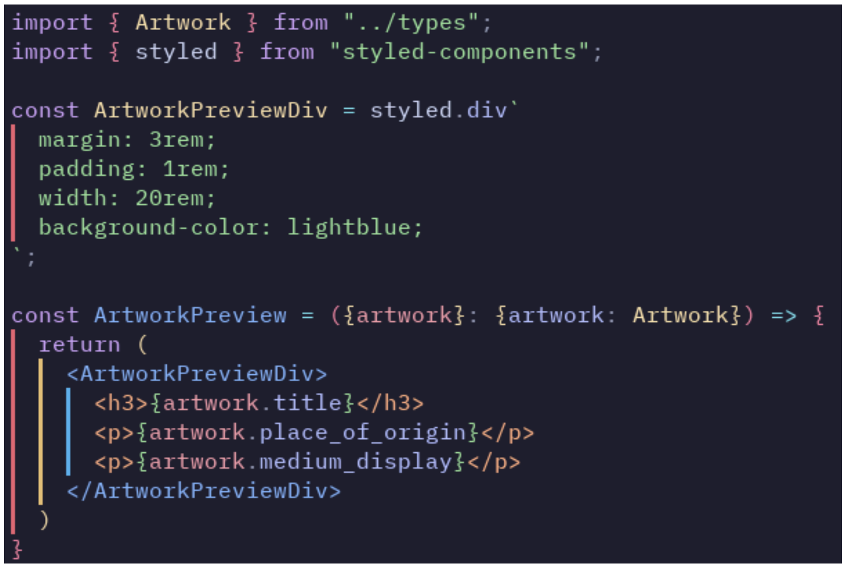
Task 3: Fetching data from an API endpoint

1. Create a **types.ts** file at the path **src/types.ts**. Here we will define the data types we expect to receive from the API
    a. We will be using the Art Institute of Chicago's API. You can visit https://api.artic.edu/api/v1/artworks to see the expected output. Include any relevant fields you want to use. Below is an example. Be sure to include **id** as a field.



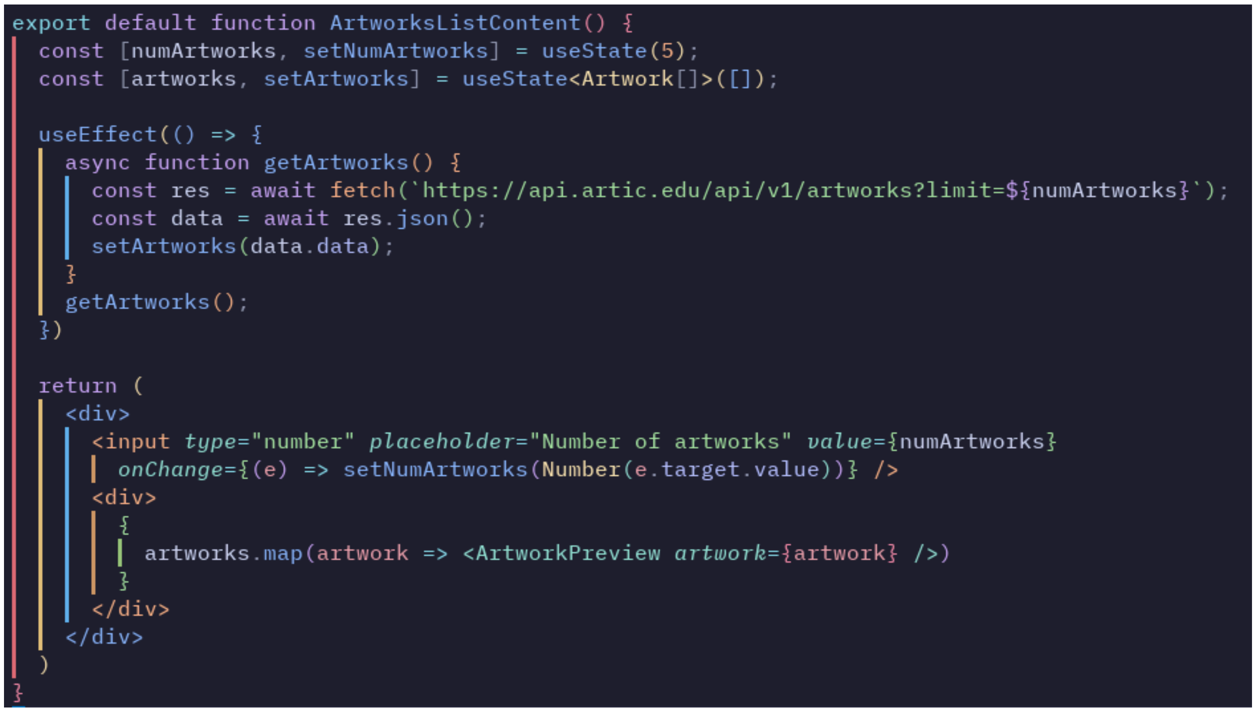
```
export type Thumbnail = {
        lqip: string;
        width: number;
        height: number;
        alt_text: string;
};

export type Artwork = {
        id: number;
        title: string;
        thumbnail: Thumbnail;
        place_of_origin: string;
        medium_display: string;
        is_public_domain: boolean;
        is_on_view: boolean;
        image_id: string;
};
```

2. Install the **styled-components** package so that we can use css within our React files. Use the command **npm install styled-components**
3. Import the **Artwork** type and create another component in the **artworks-list-content.tsx** file to display a preview of each artwork.
    a. This component could be put in its own file. However, since it will only be used here, it can be thought of as a child component of **ArtworksListContent** so we will put it in the same file



```
import { Artwork } from "../types";
import { styled } from "styled-components";

const ArtworkPreviewDiv = styled.div`
  margin: 3rem;
  padding: 1rem;
  width: 20rem;
  background-color: lightblue;
`;

const ArtworkPreview = ({artwork}: {artwork: Artwork}) => {
  return (
    <ArtworkPreviewDiv>
      <h3>{artwork.title}</h3>
      <p>{artwork.place_of_origin}</p>
      <p>{artwork.medium_display}</p>
    </ArtworkPreviewDiv>
  )
}
```

4. Edit **ArtworksListContent** to fetch and display artwork data from the API
   a. First, we create a variable that will contain a list of artworks
   b. Use the useEffect hook to fetch data from the API again when **numArtworks** changes. This hook is also imported from **react**, like the **useState** hook. BE SURE TO IMPORT IT!!!
   c. We then use the map array method and pass in each artwork to an anonymous function that returns a preview component with that piece's data
   d. When calling the API, we pass in the value of **numArtworks** via string interpolation as the query parameter called **limit**. This defines the maximum number of items for us to receive. This works because it is a parameter specific in the API's code. See their documentation here: https://api.artic.edu/docs/#endpoints
      i. Here is the content within the backticks: **https://api.artic.edu/api/v1/artworks?limit=${numArtworks}**
      ii. To fetch data from the API, we declare an asynchronous function that awaits the API, then turns the result into JSON, and, finally, sets the new value of the **artworks** variable. This function is then called.



```
export default function ArtworksListContent() {
  const [numArtworks, setNumArtworks] = useState(5);
  const [artworks, setArtworks] = useState<Artwork[]>([]);

  useEffect(() => {
    async function getArtworks() {
      const res = await fetch(`https://api.artic.edu/api/v1/artworks?limit=${numArtworks}`);
      const data = await res.json();
      setArtworks(data.data);
    }
    getArtworks();
  })

  return (
    <div>
      <input type="number" placeholder="Number of artworks" value={numArtworks}
        onChange={(e) => setNumArtworks(Number(e.target.value))} />
      <div>
        {
          artworks.map(artwork => <ArtworkPreview artwork={artwork} />)
        }
      </div>
    </div>
  )
}
```

Optional Additional Work:
If you have finished early, here is some additional work you may want to complete to improve the user experience. If you have any other ideas, feel free to try those as well.

1. Further style the **ArtworkPreview** component

2. Handle errors that may occur when fetching data from the API
3. Get rid of extraneous artwork data
    a. This is best practice, but we neglected this step for the sake of simplicity
    b. Hint: This could involve mapping the data array from the API. However, a better way may involve reading the API documentation and only requesting the required fields
4. Add filtering options to only show pieces that meet certain criteria