Index.jsx.

```
import React, { useState } from 'react';
import AddressForm from '../components/AddressForm';
import * as Web3 from '@solana/web3.js';
import './index.css';
```

} Import statements

solana JS SDK.

```
const Home = () => {
  const [balance, setBalance] = useState(0)
  const [address, setAddress] = useState('')
  const [isExecutable, setIsExecutable] = useState(false);

  const addressSubmittedHandler = (address) => {
    try {
      setAddress(address) — 1
      const key = new Web3.PublicKey(address) — 2
      const connection = new Web3.Connection(Web3.clusterApiUrl('devnet')) — 3
      connection.getBalance(key).then(balance => { — 4
        setBalance(balance / Web3.LAMPORTS_PER_SOL)
      })
      connection.getAccountInfo(key).then(info => { — 5
        setIsExecutable(info?.executable ?? false);
      })
    } catch (error) { — 6
      setAddress('')
      setBalance(0)
      alert(error)
    }
  }
```

} useState Hook declaration.

1. function first set the address state variable to entered value.
2. Publickey of entered address is created
3. new connection is made using the cluster API for devnet.

④ getBalance method is called on the connection, passing in the publicKey, to retrieve the balance of the account in lamports.

- The value is divided by number of lamports per SOL and stored in the balance state variable using getBalance method.

⑤ The getAccountInfo method is called on the connection, passing in the publicKey, to retrieve info about the account. The "executable" field is then extracted from returned info object and stored in the isExecutable state variable

⑥ If any errors then, Balance, address set (0, '') respectively and alert (error) is displayed.

```
return (
  <>
    <div className="container">

      <header className="header">
        <p className="title">
          Enter Your Solana Address
        </p>
        <AddressForm handler={addressSubmittedHandler} />
        <p className="address">{`Address: ${address}`}</p>
        <p className="balance">{`Balance: ${balance} SOL`}</p>
        <p className="executable">{`Is it executable? ${isExecutable ? 'Yes' : 'No'}`}</p>
      </header>
    </div>
  </>
)
}

export default Home;
```

Address Form component called.

```
import React, { useState } from 'react';
import './AddressForm.css';

function AddressForm(props) {
  const [values, setValues] = useState({
    address: '',
  });

  const handleSubmit = (e) => {
    e.preventDefault();
    props.handler(values.address)
  };

  const handleAddressInputChange = (event) => {
    event.persist();
    setValues((values) => ({
      ...values,
      address: event.target.value,
    }));
  };
```

import.

initial value of address empty string

spread operator

```
  return (
    <div className="address-form">
      <form onSubmit={handleSubmit} className="form">
        <input
          id="public-key"
          className="input"
          type="text"
          placeholder="Public Address"
          name="firstName"
          value={values.address}
          onChange={handleAddressInputChange}
        />
        <br />
        <button type="submit" className="button">
          Check SOL Balance
        </button>
      </form>
    </div>
  );
}

export default AddressForm;
```

trigger on submit.

triggers and updates the values.