Step-by-Step Exercise Lab

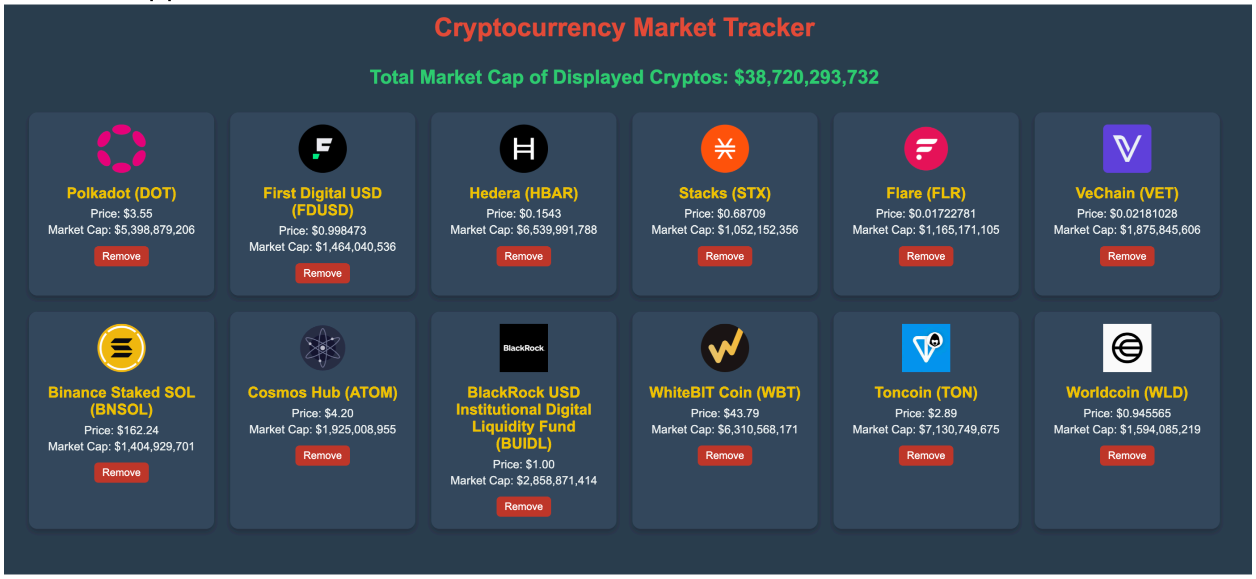
Build a Cryptocurrency Market Tracker (using CDNs)

This exercise will guide you through creating a React web application from scratch using Content Delivery Networks (CDNs). You'll learn to fetch data from an API, display it, perform calculations, and add interactivity, all within one HTML file.
Goal: Create an app that fetches a list of cryptocurrencies, displays a random selection, calculates their total market capitalization, and allows you to remove items from the list.
Prerequisites:
A text editor (like VS Code, Notepad++, Sublime Text).
A web browser (Chrome, Firefox, Safari).
Basic understanding of HTML structure.
Basic JavaScript concepts (variables, functions, arrays).

You will build a React App to render some data from the following API:
https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_cap_desc&per_page=100&page=1&sparkline=false

You can check the contents of the API response before you start creating your app.

Out final App will look like to:



Cryptocurrency Market Tracker

Total Market Cap of Displayed Cryptos: $38,720,293,732

Polkadot (DOT)
Price: $3.55
Market Cap: $5,398,879,206

First Digital USD (FDUSD)
Price: $0.998473
Market Cap: $1,464,040,536

Hedera (HBAR)
Price: $0.1543
Market Cap: $6,539,991,788

Stacks (STX)
Price: $0.68709
Market Cap: $1,052,152,356

Flare (FLR)
Price: $0.01722781
Market Cap: $1,165,171,105

VeChain (VET)
Price: $0.02181028
Market Cap: $1,875,845,606

Binance Staked SOL (BNSOL)
Price: $162.24
Market Cap: $1,404,929,701

Cosmos Hub (ATOM)
Price: $4.20
Market Cap: $1,925,008,955

BlackRock USD Institutional Digital Liquidity Fund (BUIDL)
Price: $1.00
Market Cap: $2,858,871,414

WhiteBIT Coin (WBT)
Price: $43.79
Market Cap: $6,310,568,171

Toncoin (TON)
Price: $2.89
Market Cap: $7,130,749,675

Worldcoin (WLD)
Price: $0.945565
Market Cap: $1,594,085,219

Step 1: Create Your Basic HTML File

You'll start with the barebones HTML structure.
Create a New File: In an empty folder on your computer, create a new file and name it index.html.
Add Basic HTML Boilerplate: Copy and paste the following code into your **index.html** file.

Step 2: Include React, ReactDOM, and Babel from CDNs

To use React directly in your HTML, you need to include the React library, ReactDOM (which connects React to the DOM), and Babel (to translate modern JavaScript and JSX into browser-understandable code).

1. **Add Script Tags:** Place the following <script> tags **inside the <head> section** of your index.html, just before the closing </head> tag.

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Scoreboard</title>
  <link rel="stylesheet" href="css/style.css" />
</head>

<body>
  <div id="root"></div>

  <script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
  <script type="text/babel" src="./js/app.js"></script>
</body>

</html>
```

Step 3: Create Your First React Component (App)

You'll now define your main React component and tell ReactDOM to render it into the #root div.

1. **Add React Script Block:** Add a new <script> tag **just before the closing </body> tag**. This script tag **must** have type="text/babel" so Babel knows to process its content.

```
// Destructure useState and useEffect from the global React object
 const {useState, useEffect} = React;

 // Define your main App component
 const App = () => {
     // For now, let's just show a simple message
     return (
 <div>
     <h1>Hello, Crypto Tracker!</h1>
 </div>
 );
 };

 // Tell ReactDOM to render your App component into the div with id="root"
 const root = ReactDOM.createRoot(document.getElementById('root'));
 root.render(<App />);
```

Test It: Save and refresh your browser. You should now see "Hello, Crypto Tracker!" on your page.

Step 4: Add the shuffleArray Helper Function

You'll need a way to randomly select cryptocurrencies. This shuffleArray function will be used for that.

1. **Define shuffleArray:** Place the shuffleArray function **inside your <script type="text/babel"> block**, but **outside** the App component definition (e.g., just above const App = () => { ... }).

```
// Destructure useState and useEffect from the global React object
const { useState, useEffect } = React;

// Step 4: Define shuffleArray helper function
function shuffleArray(array, n = 12) {
    let newArray = [...array]; // Create a shallow copy to avoid mutating original
    let currentIndex = newArray.length;
    let randomIndex;

    while (currentIndex !== 0) {
        randomIndex = Math.floor(Math.random() * currentIndex);
        currentIndex--;
        [newArray[currentIndex], newArray[randomIndex]] = [
            newArray[randomIndex],
            newArray[currentIndex],
        ];
    }
    const numToSelect = Math.min(n, newArray.length);
    return newArray.slice(0, numToSelect);
}
```

2. **Test It:** Save and refresh. No visual change, but the helper function is now available.

Step 5: Fetch Data from CoinGecko API

Now, let's get the actual crypto data!

1. **Add State for Data:** Inside the App component, define a state variable to hold your cryptocurrency list.

```
const App = () => {

    const [cryptoList, setCryptoList] = useState([]); // Step 5: State for fetched
crypto data
```

2. **Add useEffect for Fetching:** Still inside the App component, add the useEffect hook to fetch data when the component mounts.

```
useEffect(() => {
        console.log('Step 5: Fetching crypto data!');

fetch('https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_cap
_desc&per_page=100&page=1&sparkline=false')
            .then(res => {
                if (!res.ok) { // Check if the network request was successful
                    throw new Error(`HTTP error! status: ${res.status}`);
                }
                return res.json(); // Parse the JSON response
            })
            .then(data => {
                // For now, let's just log the raw data to see if it works
                console.log('Raw API data:', data);
                // We'll process and set cryptoList in the next step
            })
            .catch(err => console.error('Error fetching crypto:', err)); // Catch any
errors during fetch
    }, []); // Empty dependency array: runs only once after initial render
```

3. **Test It:** Save and refresh your browser. Open your browser's developer console (usually F12 or right-click -> Inspect -> Console). You should see "Step 5: Fetching crypto data!" and then a large array of cryptocurrency objects logged to the console.

Step 6: Process and Select Random Cryptocurrencies

Now, refine the fetched data and pick your random selection.

1. **Map and Select:** Modify the useEffect's second .then() block to map the data and use shuffleArray.

```
const App = () => {
    const [cryptoList, setCryptoList] = useState([]);

    useEffect(() => {
        console.log('Step 6: Processing and selecting crypto data!');

fetch('https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_
cap_desc&per_page=100&page=1&sparkline=false')
            .then(res => {
                if (!res.ok) {
                    throw new Error(`HTTP error! status: ${res.status}`);
                }
                return res.json();
            })
            .then(data => {
                const mappedData = data.map(crypto => ({
                    id: crypto.id, // Unique ID for React key
                    name: crypto.name,
                    symbol: crypto.symbol,
                    currentPrice: crypto.current_price,
                    marketCap: crypto.market_cap,
                    image: crypto.image // URL to the crypto's logo
                }));

                const selectedCryptos = shuffleArray(mappedData, 12); // Select 12
random unique cryptos
                setCryptoList(selectedCryptos); // Update state with the selected
list
                console.log('Selected cryptos:', selectedCryptos); // Log to
confirm
            })
            .catch(err => console.error('Error fetching crypto:', err));
    }, []);

    return (
        <div>
            <h1>Crypto Market Tracker</h1>
            {/* Step 6: Display a loading message or placeholder for now */}
            {cryptoList.length === 0 ? (
                <p>Loading cryptocurrencies...</p>
            ) : (
                <p>Cryptos loaded! Ready to display {cryptoList.length} items.</p>
            )}
        </div>
    );
};
```

2. **Test It:** Save and refresh. Check your console. You should now see "Selected cryptos:" followed by an array of 12 (or fewer if the API returned less) cleaned-up crypto objects. The page itself will show "Cryptos loaded! Ready to display X items."

Step 7: Create the CryptoCard Component

This component will be responsible for displaying individual cryptocurrency details.

1. **Define CryptoCard:** Place this code **inside your <script type="text/babel"> block**, but **outside** the App component definition (e.g., just above const App = () => { ... }).

```
// Step 7: Define the CryptoCard component
function CryptoCard({ id, name, symbol, price, marketCap, image, onRemove }) {
    return (
        <div className="crypto-card"> {/* We'll add CSS for this class later */}
            <img src={image} alt={`${name} logo`} className="crypto-card-img" />
            <h3>{name} ({symbol.toUpperCase()})</h3>
            <p>Price: ${price.toLocaleString(undefined, { minimumFractionDigits: 2,
maximumFractionDigits: 8 })}</p>
            <p>Market Cap: ${marketCap.toLocaleString()}</p>
            <button className="remove-button" onClick={() =>
onRemove(id)}>Remove</button>
        </div>
    );
}
```

Test It: Save and refresh. No visual change yet, as CryptoCard isn't being used.

Step 8: Render the List of Crypto Cards

Now, let's make the cards appear on the page!

1. **Map cryptoList to CryptoCards:** In App's return statement, modify the content to map over cryptoList and render CryptoCard components.
2. **Define removeCrypto function:** This function will be passed down to CryptoCard to handle removing items.

```
    const removeCrypto = (id) => {
        // Filter out the removed crypto
        const updatedList = cryptoList.filter(crypto => crypto.id !== id);
        setCryptoList(updatedList);
        // We'll update total market cap based on this new list in a later step
    };

    return (
        <div>
            <h1>Cryptocurrency Market Tracker</h1>
            {/* Step 8: Render the list of CryptoCards */}
            <div className="crypto-list"> {/* We'll add CSS for this class later */}
                {cryptoList.length > 0 ? (
                    cryptoList.map(crypto => (
                        <CryptoCard
                            key={crypto.id} // Essential for React lists
                            id={crypto.id}
                            name={crypto.name}
                            symbol={crypto.symbol}
                            price={crypto.currentPrice}
                            marketCap={crypto.marketCap}
                            image={crypto.image}
                            onRemove={removeCrypto} // Pass the removal function
                        />
                    ))
                ) : (
                    <p className="no-data-message">Loading cryptocurrencies...</p>
                )}
            </div>
        </div>
    );
```

Test It: Save and refresh. You should now see 12 (or fewer) crypto cards displayed vertically on your page. If you click a "Remove" button, the card should disappear.

Step 9: Calculate and Display Total Market Capitalization

Now, let's add the core calculation feature.

1. **Add totalMarketCap State:** Inside the App component, define a new state variable.

```
    const [totalMarketCap, setTotalMarketCap] = useState(0); // Step 9: State for
total market cap
```

2. **Calculate on Initial Load:** In useEffect, after selectedCryptos are set, calculate their total market cap and update the state.

```
.then(data => {
                const mappedData = data.map(crypto => ({
                    id: crypto.id, // Unique ID for React key
                    name: crypto.name,
                    symbol: crypto.symbol,
                    currentPrice: crypto.current_price,
                    marketCap: crypto.market_cap,
                    image: crypto.image // URL to the crypto's logo
                }));

                const selectedCryptos = shuffleArray(mappedData, 12); // Select 12
random unique cryptos
                setCryptoList(selectedCryptos); // Update state with the selected list
                console.log('Selected cryptos:', selectedCryptos); // Log to confirm

                // Step 9: Calculate total market cap for the displayed list on
initial load
                const sumMarketCap = selectedCryptos.reduce((acc, crypto) => {
                    return acc + crypto.marketCap;
                }, 0);
                setTotalMarketCap(sumMarketCap);
            })
            .catch(err => console.error('Error fetching crypto:', err));
    }, []);
```

3. **Recalculate on Removal:** In removeCrypto, after filtering the list, recalculate the sum based on the *new* updatedList.

```
    const removeCrypto = (id) => {
        const updatedList = cryptoList.filter(crypto => crypto.id !== id);
        setCryptoList(updatedList);

        // Step 9: Recalculate total market cap with the updated list
        const newTotal = updatedList.reduce((acc, crypto) => acc + crypto.marketCap,
0);
        setTotalMarketCap(newTotal);
    };
```

4. **Display the Total:** In App's return, add a heading to display the totalMarketCap.

```
    return (
        <div>
            <h1>Cryptocurrency Market Tracker</h1>
            {/* Step 8: Render the list of CryptoCards */}
            {/* Step 9: Display total market cap */}
            <h2>Total Market Cap of Displayed Cryptos:
${totalMarketCap.toLocaleString()}</h2>
            <div className="crypto-list"> {/* We'll add CSS for this class later */}
                {cryptoList.length > 0 ? (
                    cryptoList.map(crypto => (
                        <CryptoCard
                            key={crypto.id} // Essential for React lists
                            id={crypto.id}
                            name={crypto.name}
                            symbol={crypto.symbol}
                            price={crypto.currentPrice}
                            marketCap={crypto.marketCap}
                            image={crypto.image}
                            onRemove={removeCrypto} // Pass the removal function
                        />
                    ))
                ) : (
                    <p className="no-data-message">Loading cryptocurrencies...</p>
                )}
            </div>
        </div>
    );
```

5. **Test It:** Save and refresh. You should now see the "Total Market Cap" at the top. Click "Remove" on a card, and the total should update dynamically.

Step 10: Basic Styling

Let's make the app look better by adding some CSS.

1. **Add <style> Block:** Place the following CSS rules **inside the <head> section** of your index.html, after the <title> tag and before any <script> tags for CDNs.

```
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #2c3e50; /* Dark background */
            color: #ecf0f1; /* Light text */
            text-align: center;
        }
        h1 {
            color: #e74c3c; /* Red accent */
            margin-bottom: 30px;
        }
        h2 {
            color: #2ecc71; /* Green accent */
            margin-top: 20px;
            margin-bottom: 30px;
        }
        .crypto-list {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            gap: 20px; /* Space between cards */
        }
        .crypto-card {
            background-color: #34495e; /* Darker card background */
            border-radius: 10px;
            padding: 15px;
            width: 200px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            transition: transform 0.2s ease-in-out;
            cursor: pointer; /* Indicate it's clickable */
            display: flex;
            flex-direction: column;
            align-items: center;
            text-align: center;
        }
        .crypto-card:hover {
            transform: translateY(-5px);
        }
        .crypto-card-img {
            width: 60px;
            height: 60px;
            margin-bottom: 10px;
        }
        .crypto-card h3 {
            margin: 5px 0;
            color: #f1c40f; /* Yellow accent for name */
        }
        .crypto-card p {
            margin: 2px 0;
            font-size: 0.9em;
        }
        .crypto-card .remove-button {
            background-color: #c0392b; /* Dark red for remove button */
            color: white;
            border: none;
            border-radius: 5px;
            padding: 5px 10px;
            margin-top: 10px;
            cursor: pointer;
            font-size: 0.8em;
        }
        .crypto-card .remove-button:hover {
            background-color: #e74c3c;
        }
        .no-data-message {
            margin-top: 50px;
            font-style: italic;
            color: #bdc3c7;
        }
```

Test It: Save and refresh. Your app should now have a much better visual presentation!

Final Review: Complete index.html Code

For your convenience, here's the complete index.html (with all the code in the .html) file with all the steps combined. You can use this as a reference or to check your work.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Crypto Market Tracker</title>
    <script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #2c3e50;
            /* Dark background */
            color: #ecf0f1;
            /* Light text */
            text-align: center;
        }

        h1 {
            color: #e74c3c;
            /* Red accent */
            margin-bottom: 30px;
        }

        h2 {
            color: #2ecc71;
            /* Green accent */
            margin-top: 20px;
            margin-bottom: 30px;
        }

        .crypto-list {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            gap: 20px;
            /* Space between cards */
        }

        .crypto-card {
            background-color: #34495e;
            /* Darker card background */
            border-radius: 10px;
            padding: 15px;
            width: 200px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            transition: transform 0.2s ease-in-out;
            cursor: pointer;
            /* Indicate it's clickable */
            display: flex;
            flex-direction: column;
            align-items: center;
            text-align: center;
        }

        .crypto-card:hover {
            transform: translateY(-5px);
        }

        .crypto-card-img {
            width: 60px;
            height: 60px;
            margin-bottom: 10px;
        }

        .crypto-card h3 {
            margin: 5px 0;
            color: #f1c40f;
            /* Yellow accent for name */
        }

        .crypto-card p {
            margin: 2px 0;
            font-size: 0.9em;
        }

        .crypto-card .remove-button {
            background-color: #c0392b;
            /* Dark red for remove button */
            color: white;
            border: none;
            border-radius: 5px;
            padding: 5px 10px;
            margin-top: 10px;
```

```
            cursor: pointer;
            font-size: 0.8em;
        }

        .crypto-card .remove-button:hover {
            background-color: #e74c3c;
        }

        .no-data-message {
            margin-top: 50px;
            font-style: italic;
            color: #bdc3c7;
        }
    </style>
</head>

<body>
    <div id="root"></div>
    <script type="text/babel">
        const { useState, useEffect } = React;

        // Step 4: Define shuffleArray helper function
        function shuffleArray(array, n = 12) {
            let newArray = [...array];
            let currentIndex = newArray.length;
            let randomIndex;

            while (currentIndex !== 0) {
                randomIndex = Math.floor(Math.random() * currentIndex);
                currentIndex--;
                [newArray[currentIndex], newArray[randomIndex]] = [
                    newArray[randomIndex],
                    newArray[currentIndex],
                ];
            }
            const numToSelect = Math.min(n, newArray.length);
            return newArray.slice(0, numToSelect);
        }

        // Step 7: Define the CryptoCard component
        function CryptoCard({ id, name, symbol, price, marketCap, image, onRemove }) {
            return (
                <div className="crypto-card">
                    <img src={image} alt={`${name} logo`} className="crypto-card-img"
/>
                    <h3>{name} ({symbol.toUpperCase()})</h3>
                    <p>Price: ${price.toLocaleString(undefined, {
minimumFractionDigits: 2, maximumFractionDigits: 8 })}</p>
```

```
                    <p>Market Cap: ${marketCap.toLocaleString()}</p>
                    <button className="remove-button" onClick={() =>
onRemove(id)}>Remove</button>
                </div>
            );
        }

        // Step 3: Define the App component
        const App = () => {
            // Step 5: State for fetched crypto data
            const [cryptoList, setCryptoList] = useState([]);
            // Step 9: State for total market cap
            const [totalMarketCap, setTotalMarketCap] = useState(0);

            // Step 5: useEffect for fetching data
            useEffect(() => {
                console.log('Step 5 & 6: Fetching and processing crypto data!');

fetch('https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_cap
_desc&per_page=100&page=1&sparkline=false')
                    .then(res => {
                        if (!res.ok) {
                            throw new Error(`HTTP error! status: ${res.status}`);
                        }
                        return res.json();
                    })
                    .then(data => {
                        // Step 6: Map and select random cryptos
                        const mappedData = data.map(crypto => ({
                            id: crypto.id,
                            name: crypto.name,
                            symbol: crypto.symbol,
                            currentPrice: crypto.current_price,
                            marketCap: crypto.market_cap,
                            image: crypto.image
                        }));

                        const selectedCryptos = shuffleArray(mappedData, 12);
                        setCryptoList(selectedCryptos);

                        // Step 9: Calculate total market cap for the displayed list
on initial load
                        const sumMarketCap = selectedCryptos.reduce((acc, crypto) => {
                            return acc + crypto.marketCap;
                        }, 0);
                        setTotalMarketCap(sumMarketCap);
                    })
                    .catch(err => console.error('Error fetching crypto:', err));
            }, []);

            // Step 8: Define removeCrypto function
            const removeCrypto = (id) => {
                const updatedList = cryptoList.filter(crypto => crypto.id !== id);
                setCryptoList(updatedList);

                // Step 9: Recalculate total market cap with the updated list
                const newTotal = updatedList.reduce((acc, crypto) => acc +
crypto.marketCap, 0);
                setTotalMarketCap(newTotal);
            };
            return (
                <div>
                    <h1>Cryptocurrency Market Tracker</h1>
                    {/* Step 9: Display total market cap */}
                    <h2>Total Market Cap of Displayed Cryptos:
${totalMarketCap.toLocaleString()}</h2>
                    <div className="crypto-list">
                        {/* Step 8: Render the list of CryptoCards */}
                        {cryptoList.length > 0 ? (
                            cryptoList.map(crypto => (
                                <CryptoCard
                                    key={crypto.id}
                                    id={crypto.id}
                                    name={crypto.name}
                                    symbol={crypto.symbol}
                                    price={crypto.currentPrice}
                                    marketCap={crypto.marketCap}
                                    image={crypto.image}
                                    onRemove={removeCrypto}
                                />
                            ))
                        ) : (
                            <p className="no-data-message">Loading
cryptocurrencies...</p>
                        )}
                    </div>
                </div>
            );
        };

        // Step 3: Mount your React App to the DOM
        const root = ReactDOM.createRoot(document.getElementById('root'));
        root.render(<App />);
    </script>
</body>
</html>
```