Use ECharts with *TypeScript* in React

1. Install ECharts & TypeScript wrapper

```
npm install echarts echarts-for-react
npm install --save-dev @types/echarts
```

Note: `@types/echarts` might not be necessary, as `echarts` has built-in typings.

2. Example `LiveChart.tsx`

```
// src/components/LiveChart.tsx
import React, { useState, useEffect } from 'react';
import ReactECharts from 'echarts-for-react';
// Optional
import { ECOption } from './types/echarts-types';
const LiveChart: React.FC = () => {
  const [data, setData] = useState<number[]>([10, 22, 33,
44, 55, 66]);

  useEffect(() => {
    const interval = setInterval(() => {
      setData(prev => [...prev.slice(1),
Math.floor(Math.random() * 100)]);
    }, 2000);
    return () => clearInterval(interval);
  }, []);

  const option: ECOption = {
    title: { text: 'Live Updating Chart (TS)' },
    tooltip: { trigger: 'axis' },
    xAxis: { type: 'category', data: ['A', 'B', 'C', 'D',
'E', 'F'] },
    yAxis: { type: 'value' },
    series: [
      {
        name: 'Value',
        type: 'line',
        smooth: true,
```

```
        data: data,
        animationDurationUpdate: 1000,
        animationEasingUpdate: 'cubicInOut',
      },
    ],
  };

  return <ReactECharts option={option} />;
};

export default LiveChart;
```

Create a Type Alias for ECharts Option

You can create this helper for autocompletion:

```
// src/types/echarts-types.ts
import type { EChartsOption } from 'echarts';
export type ECOption = EChartsOption;
```

Then import `ECOption` in your component like this:

```
import { ECOption } from '../types/echarts-types';
```

3. Use in `App.tsx`

```
// src/App.tsx
import React from 'react';
import LiveChart from './components/LiveChart';

const App: React.FC = () => (
  <div style={{ width: '80%', margin: '50px auto' }}>
    <LiveChart />
  </div>
);

export default App;
```

Works with `.tsx` out of the box

- Fully typed

- Compatible with React 18+
- Works with Vite, CRA, Next.js, etc.

Option 1: Fetch Real Data from an API

Let's say you're fetching from `/data.json` or a real API endpoint.

1. Add a JSON file (optional)

Create `public/data.json`:

```
[35, 48, 59, 28, 62, 43]
```

2. Update `LiveChart.jsx`

```
// src/components/LiveChart.jsx
import React, { useState, useEffect } from 'react';
import ReactECharts from 'echarts-for-react';

const LiveChart = () => {
  const [data, setData] = useState([]);

  useEffect(() => {
    // Fetch data on mount
    fetch('/data.json')
      .then(res => res.json())
      .then(json => setData(json))
      .catch(err => console.error('Failed to fetch data:',
err));

    // Optional: simulate updates every 2 seconds
    const interval = setInterval(() => {
      setData(prev => [...prev.slice(1),
Math.floor(Math.random() * 100)]);
    }, 2000);

    return () => clearInterval(interval);
  }, []);
```

```
  const option = {
    title: { text: 'Live Chart with Real Data' },
    tooltip: {},
    xAxis: {
      type: 'category',
      data: ['A', 'B', 'C', 'D', 'E', 'F'], // update to
match data source
    },
    yAxis: { type: 'value' },
    series: [
      {
        name: 'Value',
        type: 'line',
        smooth: true,
        data: data,
        animationDurationUpdate: 1000,
        animationEasingUpdate: 'cubicInOut',
      },
    ],
  };

  return <ReactECharts option={option} />;
};

export default LiveChart;
```

Replace `/data.json` with a Real API

If you have a real API:

```
fetch('https://api.example.com/data')
```

The API should return something like:

```
[10, 30, 25, 60, 48, 39]
```

About the JSON Data:

The data structure depends on what kind of chart you're building in ECharts.

1. Simple One-Dimensional Array

Works perfectly for **line**, **bar**, or **area** charts with **a single series** and **static x-axis categories**.

```
[10, 20, 30, 40, 50]
```
You'd pair that with:

```
xAxis: { data: ['A', 'B', 'C', 'D', 'E'] },
series: [{ type: 'line', data }]
```

2. Array of Objects

Use this if you want:

- Named data points
- Multi-series support
- Pie charts or scatter plots
- Timestamps or labels embedded with values

Example (for Pie or Label-Driven Line Chart):

```
[
  { "name": "Apples", "value": 35 },
  { "name": "Oranges", "value": 65 }
]
```

Usage:

```
series: [{
  type: 'pie',
  data: fetchedData, // directly pass this array of objects
}]
```

Or for line/bar:

```
[
  { "label": "Mon", "value": 120 },
  { "label": "Tue", "value": 200 },
  { "label": "Wed", "value": 150 }
]
```

Then extract in React:

```
const labels = data.map(d => d.label);
const values = data.map(d => d.value);

xAxis: { data: labels },
series: [{ type: 'bar', data: values }]
```

3. Multi-Series Chart

For stacked/multi-series charts:

```
{
  "x": ["Q1", "Q2", "Q3", "Q4"],
  "series": {
    "Product A": [120, 132, 101, 134],
    "Product B": [220, 182, 191, 234]
  }
}
```

You can dynamically build this:

```
const option = {
  xAxis: { data: json.x },
  series: Object.entries(json.series).map(([name, data]) =>
({
    name,
    type: 'bar',
    data
  }))
}
```

JSON Structure	Chart Type
`[10, 20, 30]`	Single series line/bar chart
`[{"name":"A", "value": 10}]`	Pie chart, labeled data
`[{"label":"Mon","value":120}]`	Bar/line chart with dynamic labels

`{ x: [...], series: {name: [...]}}`	Multi-series bar/line (stacked)

Labels

You can generate dynamic x-axis labels:

```
const labels = Array.from({ length: data.length }, (_, i)
=> `T${i}`);
```

And use:

```
xAxis: {
  type: 'category',
  data: labels,
}
```

Labels for Different Charts

1. Bar / Line / Area Charts

Shows values at data points:

```
series: [{
  type: 'bar',
  data: [10, 20, 30],
  label: {
    show: true,
    position: 'top',  // could be 'inside', 'left',
'right', etc.
    formatter: '{c}', // {c} is the current data value
  }
}]
```

2. Pie Charts

Shows category name + value/percentage inside each slice:

```
series: [{
  type: 'pie',
  data: [
    { name: 'Apples', value: 30 },
    { name: 'Oranges', value: 70 }
  ],
  label: {
    show: true,
    formatter: '{b}: {d}%'  // {b} = name, {d} = percentage
  }
}]
```

Label Formatter Tokens

Token	Meaning
{a}	Series name
{b}	Data name (e.g., x-axis or pie label)
{c}	Data value
{d}	Percentage (for pie charts)

Can also use a **custom function**:

```
formatter: function(params) {
  return `Value: ${params.value}`;
}
```

echarts-for-react

```
//This will render bars with numeric labels on top of them.
const option = {
  xAxis: {
    type: 'category',
    data: ['Mon', 'Tue', 'Wed'],
  },
  yAxis: {
    type: 'value',
  },
```

```
  series: [
    {
      type: 'bar',
      data: [120, 200, 150],
      label: {
        show: true,
        position: 'top',
        formatter: '{c}',
      },
    },
  ],
};
```

Styling Options

You can also customize label font, color, and alignment:

```
label: {
  show: true,
  position: 'inside',
  color: '#fff',
  fontSize: 14,
  fontWeight: 'bold',
  rotate: 45
}
```

Use	Property
Show label	`label.show: true`
Label position	`label.position: 'top'`, `'inside'`, etc.
Label content	`label.formatter: '{c}'` or custom function
Label style	`color`, `fontSize`, `rotate`, etc.