React Equivalent

Same App, Different Approach

React vs Electron

React	Electron
Web Browser	Desktop App
Single Page App	Native Window
npm start (dev server)	electron .
Build → Deploy to server	Build → Distribute .exe/.app

React Project Structure

```
react/
├── src/
│   ├── App.jsx          # Main calculator component
│   ├── App.css          # Styling
│   └── main.jsx         # Entry point
├── index.html           # HTML template
└── package.json         # Dependencies
```

1. package.json

```
{
  "name": "simple-calculator",
  "version": "1.0.0",
  "description": "Simple React Calculator",
  "scripts": {
    "start": "vite",
    "build": "vite build"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.0.0",
    "vite": "^4.3.0"
  }
}
```

2. index.html

This is an example using Vite.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Simple Calculator</title>
</head>
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

3. src/main.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import './App.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

4. src/Appx.js

Business Logic (calulate) & UI

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

function App() {
  const [num1, setNum1] = useState('')
  const [num2, setNum2] = useState('')

  const calculate = () => {
    ...
    setResult(res)
  }

  return (
    <div className="calculator">
      <h1>Simple Calculator</h1>
      <div className="input-group">
        <select value={operation} onChange={(e) => setOperation(e.target.value)}>
          <option value="+">+ (Add)</option>
        </select>
      </div>
      <button onClick={calculate}>Calculate</button>
      ...
    </div>
  );
}

export default App;
```

Key Differences

Electron Approach

```
// Direct DOM manipulation
document.getElementById('btn')
  .addEventListener('click', () => {
    alert('Clicked!');
  });
```

React Approach

```
// State management
const [message, setMessage] = useState('');
<button onClick={() => setMessage('Clicked!')}>
```

Running React App

```
# Install dependencies
npm install

# Start development server
npm start
# Opens http://localhost:3000

# Build for production
npm run build
# Creates optimized files in build/
```

React Benefits

1. **Component-based**
    - Reusable UI pieces
    - Better organization
2. **State Management**
    - Reactive updates
    - No manual DOM work
3. **Modern Tooling**
    - Hot reload
    - JSX syntax

Limitations of React

- Runs in browser only
- No file system access
- No native menus
- Can't minimize to tray
- No offline functionality*

*Without extra work