Guida rapida a React

Stefanoo94

Contents

Guida rapida a React

Versione: 1.0
Autore: Stefanoo94

Scopo

Questa guida fornisce i passaggi pratici per iniziare con React, organizzarne un progetto, usare hook principali, routing, gestione dello stato, testing, build e deploy. Contiene esempi pratici e comandi da eseguire.

Prerequisiti

- Node.js LTS (consigliato) e npm o yarn
- Git
- Familiarità base con JavaScript (ES6+), HTML e CSS

---

1. Creare un nuovo progetto (consigliato: Vite)

Vite è veloce e moderno:

```
# con npm
npm create vite@latest nome-app -- --template react

# oppure con yarn
yarn create vite nome-app --template react
```

Alternativa (Create React App):

```
npx create-react-app nome-app
```

Entra nella cartella e installa dipendenze:

```
cd nome-app
npm install
npm run dev
```

---

2. Struttura consigliata

Una struttura semplice e scalabile:

```
/src
  /assets
  /components
    Header.jsx
    Footer.jsx
  /pages
    Home.jsx
    About.jsx
    Dashboard.jsx
  /hooks
    useFetch.js
  /contexts
    AuthContext.jsx
  /services
    api.js
  App.jsx
  index.jsx
/public
package.json
```

---

3. Componenti e JSX

Esempio componente funzionale:

```
// src/components/Counter.jsx
import { useState } from 'react';

export default function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(c => c + 1)}>Incrementa</button>
```

```
    </div>
  );
}
```

---

4. Hook principali

- useState: stato locale
- useEffect: effetti collaterali (fetch, subscribe)
- useRef: riferimento persistente
- useContext: stato condiviso semplice
- useReducer: stato complesso

Esempio useEffect + fetch:

```
import { useState, useEffect } from 'react';

function UsersList() {
  const [users, setUsers] = useState([]);
  useEffect(() => {
    let mounted = true;
    fetch('/api/users')
      .then(r => r.json())
      .then(data => { if (mounted) setUsers(data); });
    return () => { mounted = false; };
  }, []);
  return <ul>{users.map(u => <li key={u.id}>{u.name}</li>)}</ul>;
}
```

---

5. Routing (react-router)

Installazione:

```
npm install react-router-dom
```

Uso base:

```
// src/App.jsx
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './pages/Home';
import About from './pages/About';

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home/>} />
        <Route path="/about" element={<About/>} />
      </Routes>
    </BrowserRouter>
  );
}
```

---

6. Gestione dello stato

- Per app semplici: Context + useReducer
- Per stati condivisi estesi: Redux Toolkit, Zustand, MobX
- Esempio minimo Context:

```
// src/contexts/AuthContext.jsx
import { createContext, useContext, useState } from 'react';
const AuthContext = createContext();
export const useAuth = () => useContext(AuthContext);
export function AuthProvider({ children }) {
  const [user, setUser] = useState(null);
  return <AuthContext.Provider value={{ user, setUser }}>{children}</AuthContext.Provider>;
}
```

---

7. Styling

Opzioni comuni: - CSS Modules - Styled Components / Emotion - Tailwind CSS (utility-first) - Global CSS e BEM per progetti semplici

Esempio con CSS Module:

```
// Button.module.css
.primary { background: #06f; color: white; padding: 8px 12px; border-radius: 4px; }

// Button.jsx
import styles from './Button.module.css';
export default function Button({ children }) { return <button className={styles.primary}>{children}</bu
```

---

8. Form e validazione

- Usa React Hook Form + Yup per validazioni moderne:

```
npm install react-hook-form yup @hookform/resolvers
```

Esempio:

```
import { useForm } from 'react-hook-form';
import { yupResolver } from '@hookform/resolvers/yup';
import * as yup from 'yup';

const schema = yup.object({ name: yup.string().required() });

function MyForm() {
  const { register, handleSubmit, formState: { errors } } = useForm({ resolver: yupResolver(schema) });
  return (
    <form onSubmit={handleSubmit(data => console.log(data))}>
      <input {...register('name')} />
      {errors.name && <p>Nome richiesto</p>}
      <button type="submit">Invia</button>
    </form>
  );
}
```

---

9. Comunicazione con API

- Usa fetch o axios:

```
npm install axios
```

Esempio service:

```
// src/services/api.js
import axios from 'axios';
export const api = axios.create({ baseURL: process.env.REACT_APP_API_URL || '/api' });
```

Usa interceptors per token e gestione errori.

---

10. Testing

- React Testing Library + Jest (Vite crea già setup)

```
npm install --save-dev @testing-library/react @testing-library/jest-dom
# Esempio test
```

Esempio test:

```
import { render, screen } from '@testing-library/react';
import Counter from '../components/Counter';
test('mostra contatore iniziale', () => {
  render(<Counter />);
  expect(screen.getByText(/Count:/i)).toBeInTheDocument();
});
```

---

11. Lint, formatting e commit hooks

- ESLint + Prettier:

```
npm install --save-dev eslint prettier eslint-config-prettier eslint-plugin-react
npx eslint --init
```

- Husky + lint-staged per pre-commit:

```
npm install --save-dev husky lint-staged
npx husky install
# aggiungi in package.json: "lint-staged": { "*.js": "eslint --fix" }
```

---

12. Build e deploy

- Build:

```
npm run build
```

- Deploy semplici:
  - Vercel: collegare repo e deploy automatico
  - Netlify: collegare repo o upload build

  - GitHub Pages (con gh-pages) per SPA
- Esempio GitHub Pages:

```
npm install --save-dev gh-pages
# package.json
# "homepage": "https://tuo-utente.github.io/nome-repo"
# scripts: "predeploy": "npm run build", "deploy": "gh-pages -d build"
```

---

13. Esempio minimo completo (App.jsx)

```
import React from 'react';
import Counter from './components/Counter';

export default function App() {
  return (
    <div>
      <h1>Benvenuto in React</h1>
      <Counter />
    </div>
  );
}
```

---

14. Checklist rapida prima del deploy

- ☐ Variabili d'ambiente configurate (REACT_APP_*)
- ☐ Errori ESLint risolti
- ☐ Test principali passati
- ☐ Build creata e verificata localmente
- ☐ File sensibili non committati (.env escluso)

Risorse utili

- React docs: https://reactjs.org
- Vite: https://vitejs.dev
- React Router: https://reactrouter.com
- Redux Toolkit: https://redux-toolkit.js.org
- React Testing Library: https://testing-library.com/docs/react-testing-library/intro

---

Come estendere questa guida

Vuoi che aggiunga sezioni più avanzate: SSR (Next.js), performance tuning, Pattern avanzati (Compound Components, Render Props), esempi di architettura enterprise o workflow CI/CD con GitHub Actions? Dimmi quale preferisci ed estendo la guida con esempi e file di configurazione.