REACT RIFERIMENTO RAPIDO

Componenti, hook, stato, effetti, pattern

Basi JSX

Espressioni e Attributi

```
const name = "Alice";
const el = <h1>Hello, {name}!</h1>;
const img = <img src={url} alt="photo" />;
```

Regole JSX

`{expression}`	Incorpora qualsiasi espressione JS
`className`	Usa al posto di `class`
`htmlFor`	Usa al posto di `for`
`style={{color: 'red'}}`	Stili inline come oggetto
`<Component />`	Tag self-closing obbligatori
`<> ... </>`	Fragment (nessun nodo DOM aggiuntivo)

Componenti

Componenti Funzionali

```
function Greeting({ name }) {
  return <h1>Hello, {name}!</h1>;
}

// Variante arrow function
const Greeting = ({ name }) => (
  <h1>Hello, {name}!</h1>
);
```

Props

```
function Card({ title, children }) {
  return (
    <div className="card">
      <h2>{title}</h2>
      {children}
    </div>
  );
}

<Card title="Benvenuto">
  <p>Contenuto qui</p>
</Card>
```

Props con Default

```
function Button({ label = "Clicca", onClick }) {
  return <button onClick={onClick}>{label}</button>;
}
```

Stato (useState)

Stato di Base

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);
  return (
    <button onClick={() => setCount(count + 1)}>
      Contatore: {count}
    </button>
  );
}
```

Aggiornamenti Funzionali

```
setCount(prev => prev + 1); // usa stato precedente
setItems(prev => [...prev, newItem]); // array
setUser(prev => ({...prev, name: "Bob"})); // oggetti
```

Regole dello Stato

`Aggiornamenti immutabili`	Non mutare mai lo stato direttamente
`Batching asincrono`	Gli aggiornamenti possono essere raggruppati
`Forma funzionale`	Usa `prev =>` quando dipende dallo stato precedente

Effetti (useEffect)

Pattern degli Effetti

```
import { useEffect } from "react";

// Esegui ad ogni render
useEffect(() => { /* ... */ });

// Esegui una sola volta al mount
useEffect(() => { /* ... */ }, []);

// Esegui quando le deps cambiano
useEffect(() => { /* ... */ }, [count]);

// Pulizia all'unmount
useEffect(() => {
  const id = setInterval(tick, 1000);
  return () => clearInterval(id);
}, []);
```

Liste e Chiavi

```
function TodoList({ items }) {
  return (
    <ul>
      {items.map(item => (
        <li key={item.id}>{item.text}</li>
      ))}
    </ul>
  );
}
```

Le chiavi devono essere ID stabili e unici — evitare l'indice dell'array come chiave

Gestione degli Eventi

Eventi

```
<button onClick={handleClick}>Clicca</button>
<button onClick={() => handleDelete(id)}>Elimina</button>
<input onChange={(e) => setVal(e.target.value)} />
<form onSubmit={(e) => {
  e.preventDefault();
  handleSubmit();
}}>
```

Eventi Comuni

`onClick`	Clic del mouse
`onChange`	Cambio valore dell'input
`onSubmit`	Invio del form
`onKeyDown`	Pressione tasto
`onFocus / onBlur`	Focus acquisito / perso
`onMouseEnter`	Il mouse entra nell'elemento

Rendering Condizionale

```
// Ternario
{isLoggedIn ? <Dashboard /> : <Login />}

// AND logico (short-circuit)
{hasError && <ErrorBanner />}

// Return anticipato
function Page({ user }) {
  if (!user) return <Login />;
  return <Dashboard user={user} />;
}
```

Hook

useRef

```
const inputRef = useRef(null);
// Accesso: inputRef.current.focus();
<input ref={inputRef} />
```

useRef persiste i valori tra i render senza innescare un re-render

useMemo e useCallback

```
// Memoizza computazione costosa
const sorted = useMemo(
  () => items.sort(compareFn),
  [items]
);

// Memoizza callback
const handleClick = useCallback(
  () => setCount(c => c + 1),
  []
);
```

useContext

```
import { useContext } from "react";
const value = useContext(MyContext);
```

Hook Personalizzati

```
function useLocalStorage(key, initial) {
  const [value, setValue] = useState(() => {
    const saved = localStorage.getItem(key);
    return saved ? JSON.parse(saved) : initial;
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);

  return [value, setValue];
}

// Utilizzo
const [name, setName] = useLocalStorage("name", "");
```

Gli hook personalizzati devono iniziare con 'use'

Context API

Crea e Fornisce

```
import { createContext, useContext } from "react";

const ThemeCtx = createContext("light");

function App() {
  return (
    <ThemeCtx.Provider value="dark">
      <Page />
    </ThemeCtx.Provider>
  );
}
```

Consumo

```
function Page() {
  const theme = useContext(ThemeCtx); // "dark"
  return <div className={theme}>...</div>;
}
```