

JEMORE
TOGETHER FOR SUCCESS

Formazione Frontend

Obiettivi della Formazione

Overview

Oggi ci concentreremo sul Frontend.

Contenuto:

- *Imparare i concetti fondamentali di Next.*
- *Configurare e utilizzare Next.js per sviluppare siti web dinamici.*
- *Creare componenti riutilizzabili e strutturare un'applicazione.*
- *Gestire il routing tra pagine e comprendere il layout globale.*
- *Utilizzare props e state per gestire dati dinamici.*



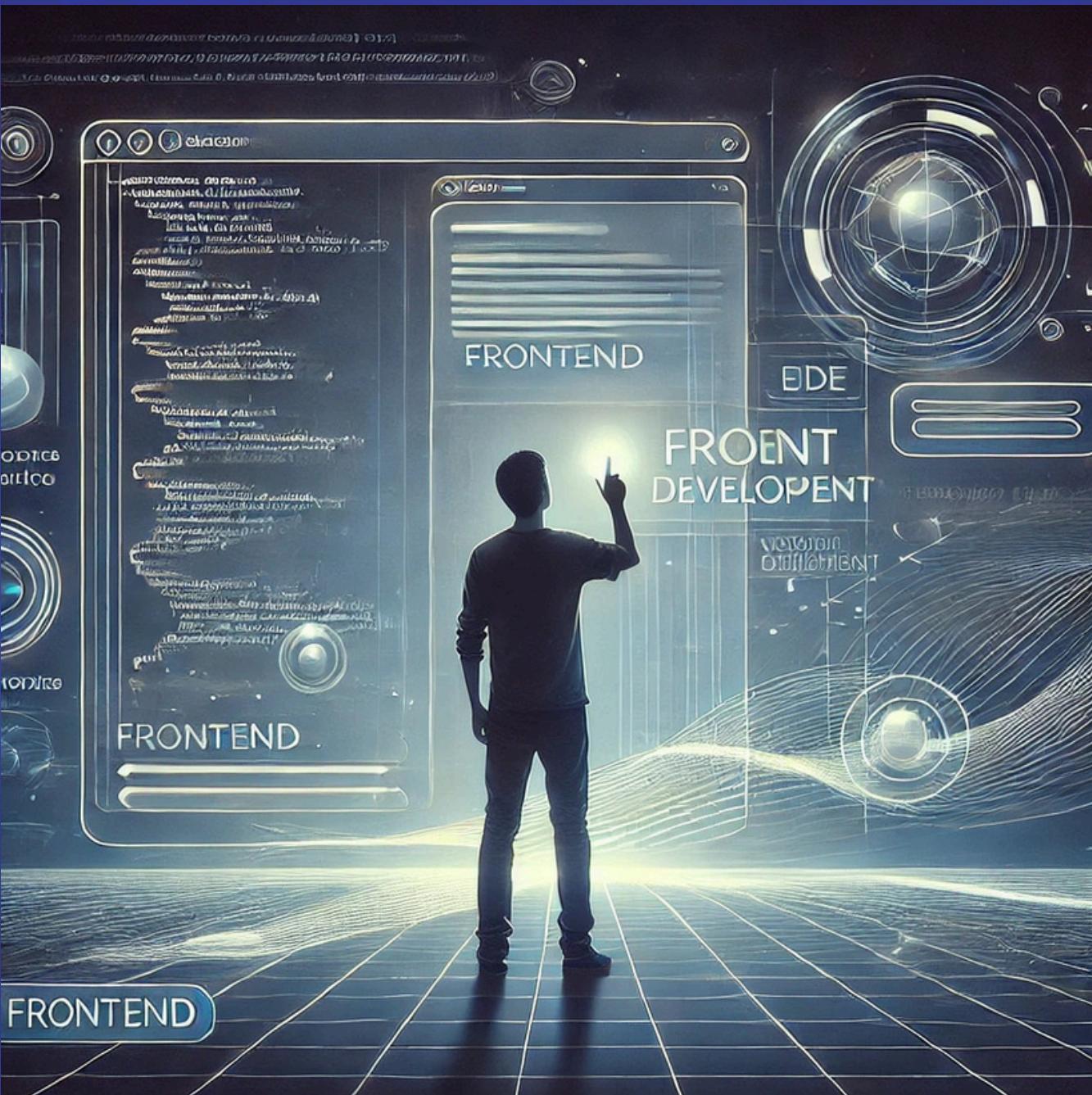
Cos'è il frontend

Le Basi

Il frontend è la parte visibile e interattiva di un'applicazione web o di un sito internet. È ciò che gli utenti vedono e con cui interagiscono direttamente quando visitano una pagina web. Il frontend si occupa di rendere l'esperienza utente fluida, intuitiva e accattivante.

Un frontend developer si occupa di:

- **Creare interfacce utente responsive e accessibili.**
- **Ottimizzare le prestazioni delle pagine web.**
- **Garantire compatibilità cross-browser.**
- **Collaborare con i backend developer per integrare dati dinamici.**



Cos'è Next

Le Basi

Next.js è un framework open-source basato su React che semplifica la creazione di applicazioni web moderne e performanti. Offre strumenti avanzati per gestire il routing, il rendering, l'ottimizzazione delle prestazioni e molto altro

Perché Usare Next.js?

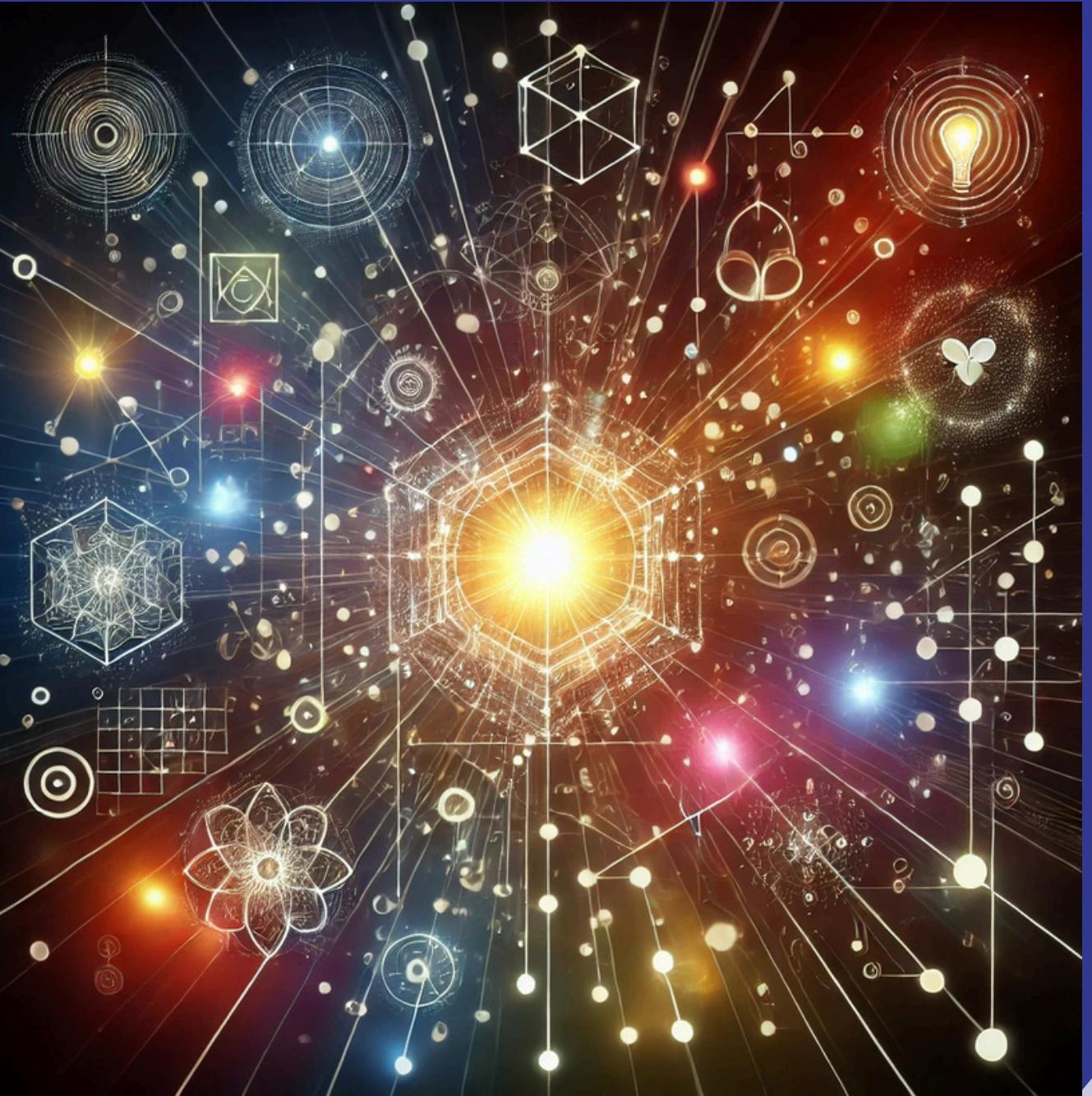
- *Perfetto per applicazioni che richiedono prestazioni elevate e SEO ottimizzato.*
- *Ideale per siti statici, blog, e-commerce e dashboard.*
- *Riduce la complessità dello sviluppo grazie a funzionalità preconfigurate.*



Concetti importanti

Prima di mettere le mani sul codice e costruire qualcosa è importante capire dei concetti importanti:

- *Componenti*
- *useState*
- *useEffect*
- *routing*
- *layout*



Componenti

Componenti

Una componente è un blocco riutilizzabile di codice che rappresenta una parte dell'interfaccia utente (UI).

```
export default function Home() {  
  return (  
    <div>  
      <Header />  
      <Hero  
        title="Benvenuto nel mio sito!"  
        subtitle="Scopri i nostri fantastici prodotti!">  
      </Hero>  
      <SubHero  
        title="Perché sceglierci?"  
        content="Offriamo soluzioni innovative e personalizzate.">  
      </SubHero>  
      <Footer />  
    </div>  
  );  
}
```

Immagina di costruire un sito web come se fosse fatto con i Lego 

Ogni componente è un pezzo: bottone, barra di ricerca, prodotto, footer, ecc.

Li combini e componi per creare l'intera pagina.



JEMORE

Props

Componenti

- **Dati passati da un componente padre a un componente figlio.**
- **Sono immutabili all'interno del componente figlio.**

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

useState: Gestione dello Stato Locale

Cos'è e sintassi

Cos'è useState?

- è un hook di React che consente ai componenti funzionali di gestire lo stato locale.
- Permette di dichiarare una variabile di stato e una funzione per aggiornarla.
- Senza uno stato, i componenti funzionali sarebbero "statici" e non potrebbero reagire a interazioni dell'utente o a cambiamenti dinamici

```
const [state, setState] = useState(initialValue);
```

- **state:** La variabile che contiene il valore corrente dello stato.
- **setState:** Una funzione per aggiornare lo stato.
- **initialValue:** Il valore iniziale dello stato.

Esempio

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

export default function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Contatore: {count}</p>
      <button onClick={() =>
        setCount(count + 1)}>Incrementa</button>
    </div>
  );
}
```

useEffect: Effetti Collaterali nei Componenti

Cos'è

Cos'è useEffect?

- Hook di React per eseguire effetti collaterali (side effects) in un componente funzionale.
- Viene eseguito dopo il rendering iniziale e/o quando cambiano determinate dipendenze.

Gli effetti collaterali sono operazioni che non riguardano direttamente il rendering, come:

- Chiamate API.
- Manipolazione del DOM.
- Sottoscrizioni a eventi.
- Timer o intervalli.

useEffect: Effetti Collaterali nei Componenti

Sintassi

```
useEffect(() => {  
    // Codice da eseguire  
}, [dependencies]);
```

- **primo argomento :** Una funzione contenente il codice da eseguire.
- **Secondo argomento (opzionale) :** Un array di dipendenze che determina quando l'effetto deve essere eseguito

Esempio

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

export default function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Hai cliccato ${count} volte`;
  }, [count]);

  return (
    <div>
      <p>Hai cliccato {count} volte</p>
      <button onClick={() => setCount(count + 1)}>
        clicca qui
      </button>
    </div>
  );
}
```

Esempi Pratici di useEffect

Esempio 1: Aggiornare il Titolo della Pagina

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

function PageTitle() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Hai cliccato ${count} volte`;
  }, [count]);

  return (
    <div>
      <p>Hai cliccato {count} volte</p>
      <button onClick={() => setCount(count + 1)}>clicca qui</button>
    </div>
  );
}
```

Esempio 2: Chiamata API

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

function FetchData() {
  const [data, setData] = useState(null);

  useEffect(() => {
    fetch('<https://api.example.com/data>')
      .then((response) => response.json())
      .then((result) => setData(result));
  }, []);

  return (
    <div>
      {data ? <pre>{JSON.stringify(data, null, 2)}</pre> : <p>Caricamento...</p>}
    </div>
  );
}
```

Tutto chiaro???

Speriamo!!!

**Iniziamo a scrivere
qualcosa**

Inizializzazione Progetto

npx create-next-app@latest

Componente Header

Il componente Header rappresenta la parte superiore del sito, solitamente contenente il logo e i link di navigazione

```
export default function Header() {  
  return (  
    <header>  
      <h1>Benvenuti nel mio sito</h1>  
    </header>  
  );  
}
```

Componente Footer

Il componente Footer rappresenta la parte inferiore del sito, solitamente contenente informazioni aggiuntive o copyright.

```
export default function Footer() {  
  return (  
    <footer>  
      <p>© 2023 My Website</p>  
    </footer>  
  );  
}
```

Componente Hero

Il componente Hero è una sezione prominente che attira l'attenzione dell'utente.

```
export default function Hero() {  
  return (  
    <section>  
      <h2>Benvenuti nel nostro eroe!</h2>  
    </section>  
  );  
}
```

Componente SubHero

Il componente SubHero è una sezione secondaria che fornisce ulteriori dettagli o informazioni.

```
export default function SubHero() {  
  return (  
    <section>  
      <h3>Un sottotitolo importante</h3>  
    </section>  
  );  
}
```

**Creiamo qualche
pagina in più !!!**

Routing in Next.js 13 (File-based Routing)

Cos'è

- In Next.js 13, il routing è basato sulla struttura della cartella /app (non più /pages).
- Ogni file nella cartella /app rappresenta una route.
 - app/home/page.js → /home
 - app/about/page.js → /about

```
app/  
  layout.js      // Layout globale  
  page.js        // Pagina principale ()  
  about/  
    page.js       // Pagina About (/about)
```

Che rottura dover
riattaccare header e
footer in ogni
componente... di certo ci
sarà un modo migliore

Layout Comune per Tutte le Pagine

Scrivere

Cos'è un Layout? Un layout è una struttura condivisa tra più pagine, come un header o un footer.

Come si usa? In Next.js 13, il file layout.js nella cartella /app definisce un layout globale.

```
// app/layout.js
import Header from '...';
import Footer from '...';

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <Header />
        {children}
        <Footer />
      </body>
    </html>
  );
}
```

**Aggiungiamo una
componente un po' più
tricky**

BlogList

Con dati Mockati

```
import React from 'react';
import BlogListProps from './index.types';

const BlogList: React.FC<BlogListProps> = ({ posts }) => {
  return (
    <section>
      <h3>Ultimi articoli del blog</h3>
      <ul>
        {posts.map((post) => (
          <li key={post.id}>
            <h4>{post.title}</h4>
            <p>
              <strong>Autore:</strong> {post.author} |{post.date}
              <strong>Data:</strong> {post.date}
            </p>
            <p>{post.content}</p>
          </li>
        ))}
      </ul>
    </section>
  );
}
```

Conclusioni

Finalmente

- **Hai imparato i fondamenti di React e Next.js.**
- **Hai creato componenti, gestito il routing e implementato props e state.**
- **Continua a praticare e approfondisci ulteriormente con argomenti avanzati come SSR, SSG e API Routes.**





JEMORE
TOGETHER FOR SUCCESS



www.jemore.it
E SIA LODATA FOGGIA

Valutazione

