

TECNOLOGIE AVANZATE PER FRONTEND

React è una **libreria JS** per la creazione di interfacce utente (GUI) web. L'obiettivo è diventare la soluzione semplice, intuitiva e definitiva per sviluppatori front-end e app basate su HTML5.

Essendo una libreria JS viene eseguito all'interno del browser ne consegue che React.js NON è uno strumento per lo sviluppo lato back-end. Quindi React non interagisce con DB o qualsiasi altra sorgente di dati si trovi su back-end, tuttavia permette di invocare delle API lato server. Può interagire con diverse tecnologie back-end in Python/Flask, Ruby on Rails, Java/Sping, PHP ecc

L'approccio in sintesi

React si ispira alla metodologia di sviluppo delle GUI del tipo Single Page Application (SPA). Una **SPA** è un'applicazione web che interagisce col browser per modificare pagine web in modo dinamico dei in funzione dei dati che arrivano dal back-end, a differenza dell'approccio classico in cui il browser carica nuove pagine a seguito dell'interazione con l'utente. Si dice, infatti, che la SPA è un contenitore all'interno del quale la pagina evolve dinamicamente.

Lo sviluppo dell'applicazione avviene attraverso la scrittura di "componenti" i quali interagiscono con le API della libreria che manipolano, a loro volta, il DOM per la creazione di elementi di interfaccia utente.

React ha introdotto il concetto di Virtual DOM: al verificarsi di un evento, invece di modificare il DOM del browser, modifica una esatta copia del DOM (il virtual DOM) del browser e si trova in RAM. La modifica del virtual DOM è più leggera di quella del DOM browser, infatti lavorando su di esso React sarà in grado di inviare al DOM del browser solo le modifiche strettamente necessarie rendendo il processo di rendering della pagina più leggero, efficiente e veloce.

Ma quindi cosa succedeva con il DOM normale?

Se si modificasse un solo elemento del DOM (supponendo di averne tanti), con la tecnologia normale si andrebbe a riscrivere l'intero DOM

E React cosa fa?

React dice al browser di modificare solo le parti effettivamente modificate e possibilmente, prima di modificare il DOM del browser, accorpa un certo numero di modifiche fatte dall'utente e decide lui quando mandare le modifiche al browser

I vantaggi

I vantaggi per lo sviluppatori sono:

- L'approccio a componenti permette allo sviluppatore di costruire interfacce complesse attraverso la composizione di mattoncini
- Un altro vantaggio dell'approccio a componenti è la possibilità di riuso in modo semplice
- Lo sviluppatore definisce la logica dei componenti e la loro posizione all'interno della GUI. La gestione del virtual DOM, delle sue trasformazioni e della comunicazione con il DOM del browser è completamente a carico di React.js

Per l'utente finale invece l'impiego del virtual DOM alleggerisce il processo di rendering dell'interfaccia sul browser con conseguente aumento delle prestazioni percettibili dall'utilizzatore.

Un primo approccio con React può essere:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Primi passi con React</title>
    <!-- IMPORT DELLE LIBRERIE REACT -->
    <script src="https://unpkg.com/react@15/dist/react.js"></script>
    <script src="https://unpkg.com/react-dom@15/dist/react-dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.24/browser.js"></script>
  </head>
  <body>

    <!-- CREO IL CONTENITORE -->
    <div id="root"></div>

    <script type="text/babel">
      // CREAZIONE DI UN REACT ELEMENT
      const elem = <p>Hello <strong>React</strong>!</p>;

      // REINDIRIZZA IL TUTTO:
      //   - elem è cosa visualizzare
      //   - il document.getElementById(...) è dove visualizzarlo
      ReactDOM.render(elem, document.getElementById('root'));
    </script>
  </body>
</html>

```