**Corso React base**

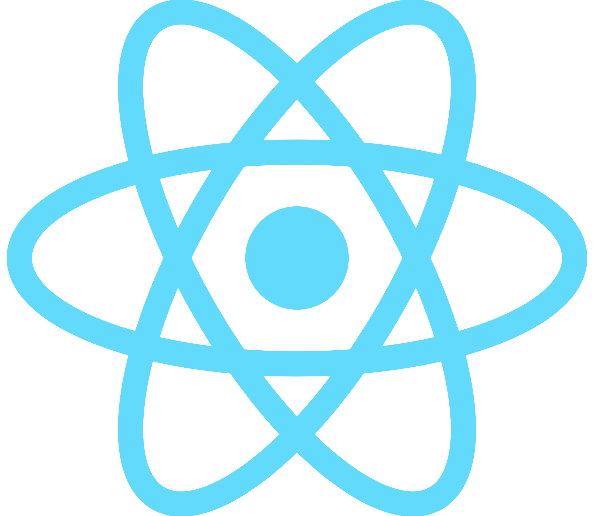


Figura 1 react\_logo.png

Link al corso: <https://www.youtube.com/watch?v=TCYnHcngPLc&list=PLP5MAKLy8lP9Ekc_hVSggV7ZkUzgW7IrH>

**Capitolo 1 – Introduzione**

Immagine che contiene nuvola, dipinto, paesaggio, arte

Descrizione generata automaticamente

**Cosa è?**

React è una **libreria** che serve a costruire applicazioni.

Essendo una libreria, e non un framework, può avere delle diramazioni che lo strutturino maggiormente, come Next.Js e Remix.Js.

**Capitolo 2 – Installazione**

Immagine che contiene Modello in scala, Set per costruzioni, Mattoncino giocattolo, Set di gioco

Descrizione generata automaticamente

1. Installazione Node.Js come environment
2. Installazione VSCode
3. Create React App **\***
4. React Vite

4a. Usiamo il commando **npm create vite@latest corso-react -- --template react** (<https://vitejs.dev/guide/>)

4b. Entriamo nella cartella (**cd corso-react**)

4c. Installiamo i pacchetti (**npm install**)

4d. Avviamo il progetto in development mode (**npm run dev**)

**\*** NOTA: Questo metodo di inizializzaione di un progetto in react non è deprecato, ma non è più consigliato per la creazione di nuove UI in React.

**Capitolo 3 – Struttura del Progetto**

Immagine che contiene testo, schermata, cielo

Descrizione generata automaticamente

**Struttura**:

* **node\_modules**: Pacchetti con le librerie node del progetto
* **public**: cartella con tutte le risorse pubbliche del sito
* **src**: codice effettivo de progetto
* **eslintrc.cjs**: strumento che forza determinate consuetudini di scrittura codice in fase di sviluppo
* **gitignore**: indici dei files da non caricare in remoto
* **index.html**: la pagina principale che fa da entry point
* **package.json**: file di configurazione che raccogli dipendenze (necessarie per il progetto) e devDipendenze (necessarie solo allo sviluppatore), script, ecc…
* **package-lock.json**: file di configurazione che raccoglie TUTTI gli indici delle dipendenze, mentre il package.json raccoglie solo le dipendenze principali usate da lo sviluppatore
* **reademe.md**: file descrittivo
* **vite.config.json**: file di configuazione di vite, il motore di build del progetto

**Capitolo 4 – Rudimenti di Sintassi**

Immagine che contiene testo, libreria, Pubblicazione, biblioteca

Descrizione generata automaticamente

Struttura di un applicativo React:

* **Entry point**: **index.html**, questo ha un elemento con **id=root** che contiene tutto il codice

che viene iniettato dallo **script main.jsx**

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

* **Inizializzazione contenuto**: **main.jsx**, modulo che inizializza l’applicazione importando le **main libraries**

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

La **struttura** di base di un **componente** consiste in una **funzione** che effettua il **return** di un blocco di **jsx**.



Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

**Capitolo 5 – Introduzione ai componenti**

Immagine che contiene schermata

Descrizione generata automaticamente

I componenti sono **funzioni** javascript che generano blocchi funzionali che generano **“html”**

Un componente viene generato a mano, la sua struttura di base è costituita da una funzione che contiene delle istruzioni in jsx (un js glorified) che restituiranno l’html voluto.

Tutti i componenti verrano incapsulati all’interno del componente patriarca denominato App.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 2 esempio di componente Navbar…

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 3 ...inserito all'interno del componente primigenio

**Capitolo 6 – JSX**

Immagine che contiene libro, schermata, Gioco per PC, interno

Descrizione generata automaticamente

**Regole base della sintassi JSX**:

1. Le parentesi servono quando ci sono più righe di codice
2. Ogni componente deve restituire un unico elemento
3. In considerazione della regola 2, ogni componente può essere attorniato non da un singolo div, ma dal **fragment**

Immagine che contiene schermata, Carattere, Elementi grafici, design

Descrizione generata automaticamente

1. Ogni tag element deve essere chiuso

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

1. Molti attributi diventano in **camelCase…**
2. … un’eccezione è **className**, essendo class una parola chiave di js
3. Una regola base è l’interpolazione di **valori** (primitive, oggetti o addirittura intere funzioni)

Immagine che contiene testo, Carattere, schermata, design

Descrizione generata automaticamente

1. Con le **doppie graffe** è possibile inserire anche **oggetti**, esempio per lo stile



**Capitolo 7 – Stile CSS**

Immagine che contiene dipinto, arte, Arte moderna, portafotografie

Descrizione generata automaticamente

**Importante**: ogni file css, per quanto correlato ad uno specifico componente, se questo è importato, sovrascriverà lo stile degli altri componenti, poiché sono **globali**.

Una libreria che evita questo problema e localizza lo stile allo specifico componente è [**styled-components**](https://styled-components.com/).

**Stilizzazione**:

* È possibile stilizzare **inline** un elemento html, oppure usare l’interpolazione a doppia graffa per importare un oggetto di stile.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 4 Nota: lo stile dovrà essere trattato come un oggetto

* È possibile usare una **classe**, come nel classico html, usando l’attributo **className**



* Essendo un oggetto lo stile può contenere tutta una serie di operazioni affinché questo sia reattivo

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

* È possibile inserire classi dinamicamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Figura 5 classi



Figura 6 jsx

**Capitolo 8 – Integrazione Tailwind CSS**

Immagine che contiene mammifero, cartone animato, cielo, dipinto

Descrizione generata automaticamente

* Installazione di tailwind css, autoprefixer e postcss cli

**npm install -D tailwindcss postcss autoprefixer**

* Inizializzaione config di tailwind

npx tailwindcss init -p

* Gestione tailwind.config.js

/\*\* @type {import('tailwindcss').Config} \*/

export default {

  content: ["./index.html", "./src/\*\*/\*.{js,ts,jsx,tsx}"],

  theme: {

    extend: {},

  },

  plugins: [],

};

* Import delle directives di tailwind css

@tailwind base;

@tailwind components;

@tailwind utilities;

* Avvio del servizio

**npm run dev**

* Esempio di uso in un h3



**Capitolo 9 – Props (proprietà dei componenti)**

Immagine che contiene disegno, arte, pianta, aria aperta

Descrizione generata automaticamente

**Cosa sono?**

Le **props** sono **attributi** specifici di React che possono essere usati sui componenti.

Questi permettono il passaggio dei dati da un componente ad un altro.

Avendo un componente **Card**

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Questo presenta un blocco html con un’immagine, un titolo e un paragrafo, affinchè questi siano dinamici e che i dati vengano inseriti dal componente padre, bisogna passare le props come argument della funzione componente, così da popolare l’html.

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamenteIl **componente** richiamato dentro il **padre** conterrà i valori da passare dentro le singole **card**.

Essendo le props un oggetto è possibile destrutturalo per inviarlo, così da rendere più efficiente il codice

Da

Card(props)

a

 Card({ title, img, description })

**Children**: è possibile invertire la situazione utilizzando la keyword **children**, questa permetteràdi passare il codice posto dentro il componente innestato direttamente dentro il componente figlio!

Componente **Padre** (App.jsx):

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Componente **Figlio** (Card.jsx):

Immagine che contiene schermata, testo, Carattere, numero

Descrizione generata automaticamente

**Capitolo 10 – Rendering condizionale**

Immagine che contiene Policromia, schermata, Elementi grafici, arte

Descrizione generata automaticamente

Il rendering condizionale permette di visualizzare, in base al soddisfacimento di condizioni, una parte di codice.

Es.

Vi sono delle card, una delle props “**isVisited**” è un booleano che indica se, la città nella card, sia stata o meno visitata

Componente Padre:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Componente Figlio:



Naturalmente è possibile definire a più livelli il rendering

Risultato:

Immagine che contiene testo, cielo, schermata, panorama

Descrizione generata automaticamente

Naturalmente gli approcci possibili…





…sono praticamente infiniti!

**AND**: operatore utilizzato per la medesima cosa, in base ad una condizione renderizza un contenuto



**Capitolo 11 – Rendering di liste**

Immagine che contiene testo, libro, forniture per ufficio, cancelleria

Descrizione generata automaticamente