

CORSO REACT

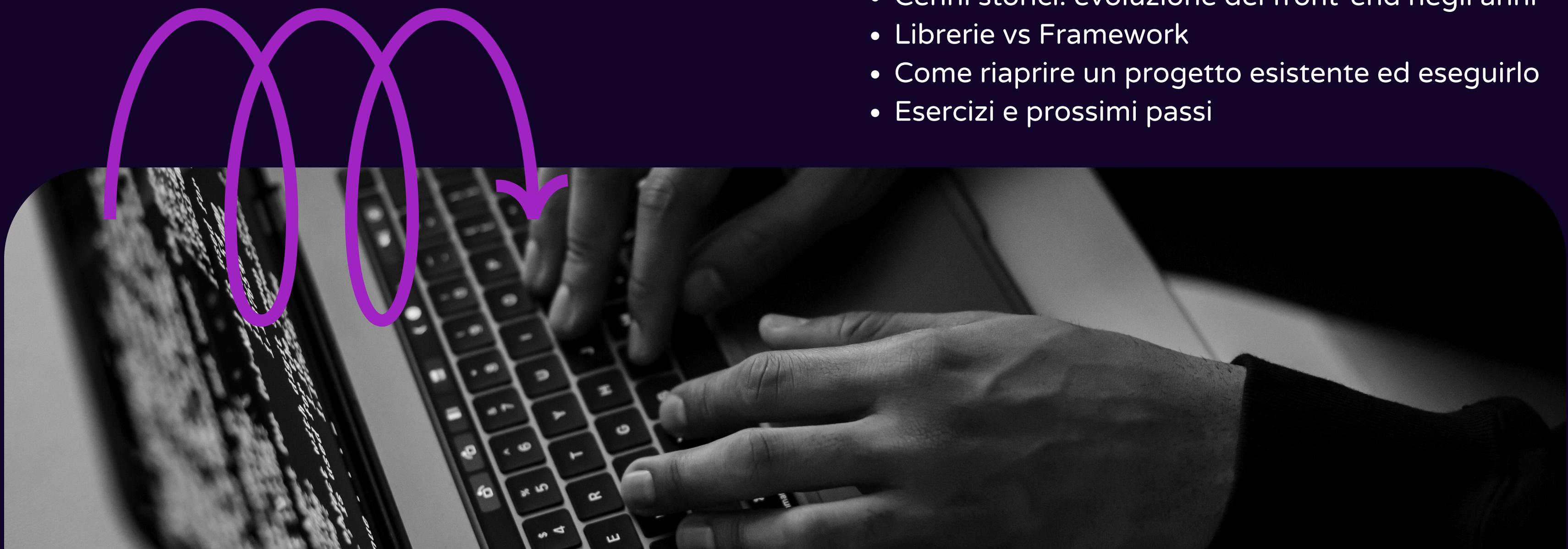
- 100 ORE -

PARTE 1 - INTRODUZIONE



Argomenti

- Introduzione: cos'è React js
- Pensare per componenti
- Introduzione a Vite js
- Creazione del primo progetto ed eseguirlo
- Analizziamo la struttura del progetto
- Angular, React e Vue a confronto
- Estensione .jsx
- Componenti: cosa sono e come crearli
- Fragment
- Cenni storici: evoluzione del front-end negli anni
- Librerie vs Framework
- Come riaprire un progetto esistente ed eseguirlo
- Esercizi e prossimi passi



materiale didattico

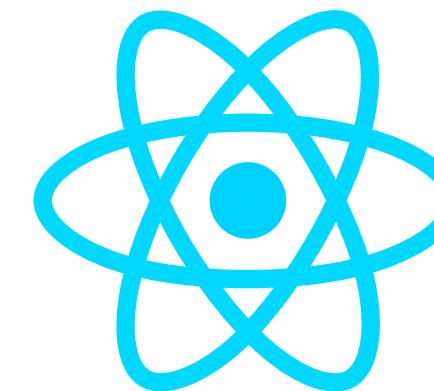


repository privata
con codice e slides:

https://github.com/itadevsupport/corso_react

Il materiale di questa lezione è nella cartella **lezione-1**

REACT è una libreria front-end, JavaScript



React

React.js è una libreria JavaScript open-source sviluppata da Facebook (ora Meta) dal 2013.

Permette la creazione interfacce utente (UI) interattive, dinamiche e reattive.

È basata su componenti riutilizzabili e permette la progettazione di applicazioni web modulari e scalabili.

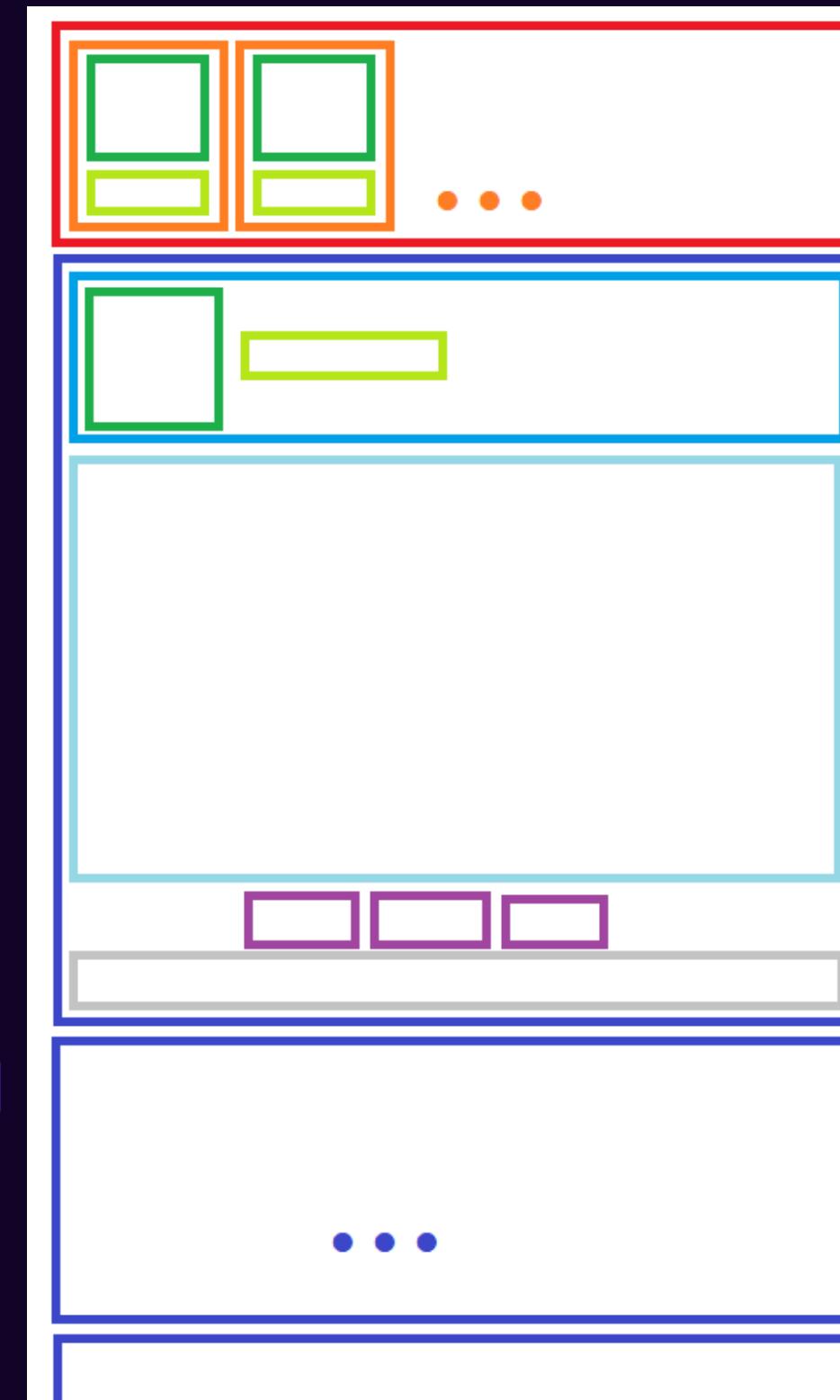
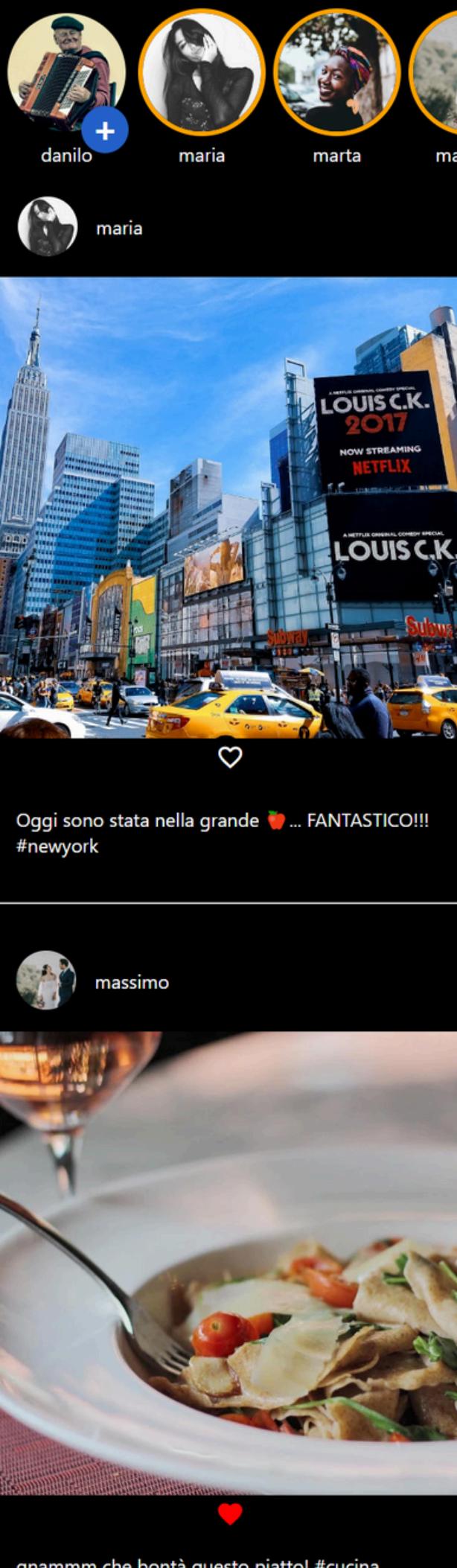
Grazie a React Native è inoltre possibile sviluppare applicazioni native per dispositivi mobile (iOS e Android).

Documentazione ufficiale di REACT: <https://react.dev/>

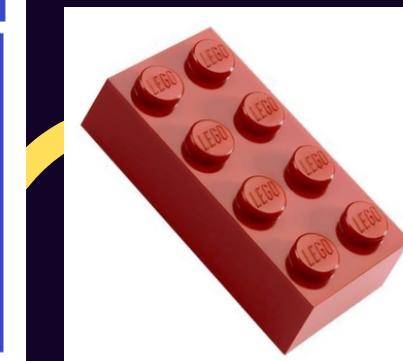


⚠️ **NON considerare la documentazione legacy (vecchie versioni): <https://it.legacy.reactjs.org/>** ❌

Pensare per componenti



Stories
User
UserImage
UserName
Post
UserInfo
PostImage
ActionButton
PostDescr



Non dobbiamo più guardare una pagina web pensandola composta da soli tag html con un loro stile e con degli eventi, ma pensiamo a componenti: porzioni di UI (user interface) composti anche da più tag html, che possono essere riutilizzati anche in altri punti del codice e possono essere resi dinamici (con proprietà distinte). Creeremo quindi vari componenti e li combineremo fra loro fino a realizzare l'interfaccia che vede l'utente.

il primo progetto in React

prerequisito I: Node js

1

Prerequisito: avere Node.js installato sul pc:

- aprire un terminale e verificare se node è installato con il comando: **node -v**
- Se compare il **numero di versione** possiamo procedere
- Se invece compare "il comando node non è riconosciuto..." dobbiamo installare node dal sito ufficiale e ripetiamo il punto 1.

<https://nodejs.org/en>



il primo progetto in React

(Vite js)

2

Creazione del progetto con Vite



Vite ci serve a creare la struttura di cartelle e file del nostro progetto React (che altrimenti dovremmo fare a mano).

Ci semplifica enormemente il lavoro!

In una nuova finestra di VSCode, trasciniamo la cartella in cui metteremo i nostri progetti di React.

Apriamo il terminale (CTRL + ò) e come indicato nel sito ufficiale di Vite <https://vite.dev/> digitiamo: **npm create vite@latest**

PS D:\LAVORO\ALTRO> **npm create vite@latest**

il primo progetto in React

(Vite js)



3

Seguendo le istruzioni successive andiamo a indicare il nome della cartella di progetto che vogliamo creare:

```
? Project name: » progetto-di-test-1|
```

poi indichiamo che tipo di progetto creare selezionando prima React e poi JavaScript come linguaggio per i componenti.

Attendiamo l'esecuzione e Vite creerà la struttura completa del progetto con file e sottocartelle che analizzeremo dopo nel dettaglio.

il primo progetto in React

(Vite js)



4

Seguiamo ancora le indicazioni e digitiamo: **cd** seguito dal nome del **progetto** indicato in precedenza per poter entrare dentro la cartella creata. Poi, sempre seguendo le indicazioni, digitiamo: **npm install** per installare tutti i moduli necessari e infine **npm run dev** per avviare l'applicazione.

```
cd progetto-di-test-1
npm install
npm run dev
```

il primo progetto in React

(Vite js)



5

Apriamo sul browser il link locale che ospita la nostra applicazione.

N.B: ad ogni salvataggio che faremo nel nostro progetto, la finestra del browser si aggiornerà automaticamente.

```
VITE v6.0.7 ready in 262 ms
→ Local: http://localhost:5175/
  Next update in 1ms
  Built to run on modern browsers
```

A screenshot of a terminal window showing the output of a Vite development server. The text "VITE v6.0.7 ready in 262 ms" is at the top. Below it, an arrow points to the URL "http://localhost:5175/" which is highlighted with a red box. The bottom of the terminal shows some smaller text related to browser compatibility and build status.

La struttura del progetto di React

index.html

Il file index.html ha solo un div e lo script main.jsx
Qui solitamente modifichiamo solo il tag title e carichiamo eventuali librerie css come Bootstrap o librerie per icone (es: google icons)

src/main.jsx

Script caricato nell'index.html
Si occupa di iniettare nell'unico div presente il componente principale dell'applicazione: App
Solitamente non ci servirà modificare questo file.

src/App.jsx

È il componente principale dell'applicazione.
Qui finalmente troviamo il codice html che vediamo a schermo.
Provando a togliere tutti i tag racchiusi da <> e sostituendoli da altri tag, vedremo le modifiche

index.css

Css globale nel quale inserire il codice degli stili che influenzano in modo globale l'applicazione, come ad esempio il body, i colori di base, il font-family, ecc

App.css

Per non mettere tutto lo stile css in un file in modo disordinato, ogni componente che creeremo, compreso App ha il suo css che useremo per dare stile agli elementi tipici di quel componente

Creiamo una cartella **components** nella quale inseriremo via via i vari componenti custom che creeremo.



Estensione .jsx e componenti

Il componente è una funzione JavaScript che ritorna del codice HTML con il relativo stile.



```
1  export default function Gatto() {  
2    return <p>io sono un gatto</p>;  
3  }
```



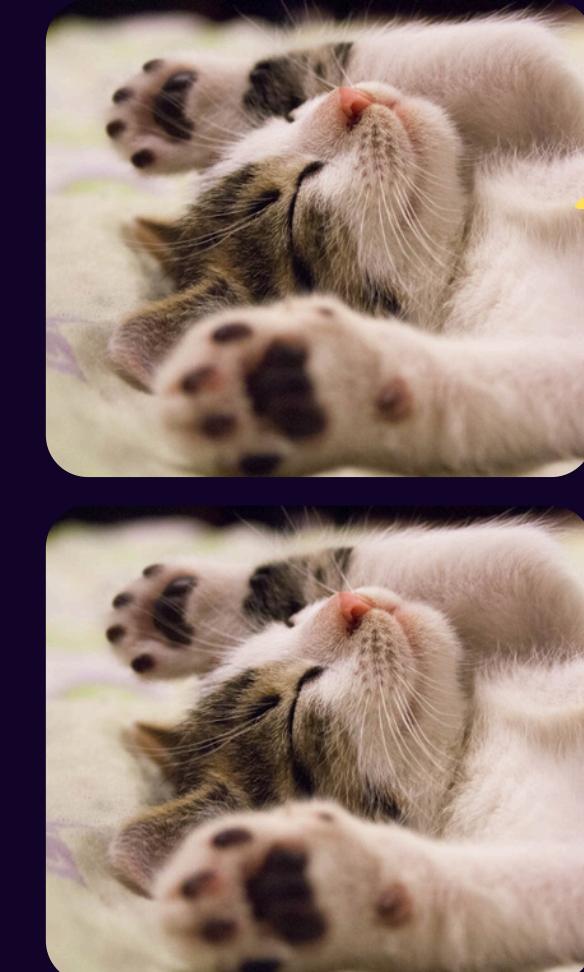
⚠ Come abbiamo fatto a inserire dei tag nel js???

Possiamo grazie all'estensione .jsx (che significa JavaScript XML) inserire direttamente i tag html nel codice JavaScript.

creiamo il nostro primo componente

poche semplici regole:

- 1- D'ora in poi creeremo i componenti all'interno della cartella components.
- 2- Per convenzione il file **.jsx** del componente inizia per lettera **Maiuscola**.
- 3- il componente sarà una funzione JavaScript con un **return** e del codice **HTML**. e il cui nome è identico a quello del file!
- 4- rendiamo esportabile il componente con **export default** davanti al nome della funzione



```
❖ Gatto.jsx •

1  export default function Gatto() {
2    |  return <p>io sono un gatto</p>;
3  }
```

Il componente potrà essere **riutilizzato** più volte quando servirà nella nostra applicazione: nel return del file App.jsx, importiamo il nostro componente Gatto:

```
import './App.css';
import Gatto from './components/Gatto';

function App() {
  return (
    <>
      <Gatto />
      <Gatto />
    </>
  );
}

export default App;
```

scrivendolo come se fosse un tag HTML ovvero racchiuso da **< e >**

⚠ Attenzione che venga inserita la riga di **import!**
Dupliciamolo e vedremo a schermo i due paragrafi che rappresentano le 2 copie del componente.

il fragment <> </>

un'altra semplice regola:

5- Dato che il return accetta solo un tag HTML, se ci serve inserire più tag nel nostro componente:

- dopo il return inseriamo le parentesi tonde () per ordinare il codice;
- al loro interno inseriamo un fragment ovvero un tag vuoto <> </> che servirà da contenitore per tutto il codice HTML del componente



```
App.jsx          Gatto.jsx X
1  export default function Gatto() {
2    return (
3      <>
4        <h1>hey miaooo</h1>
5        <p>io sono un gatto</p>
6        <hr />
7      </>
8    );
9 }
```

⚠ considerazione interessante ⚠

Il fragment <> </> a differenza di un'altro elemento contenitore come <div></div> (che avremmo potuto utilizzare), non appesantisce il browser dato che quando React renderizza la pagina, non inserirà un elemento div “inutile” nel DOM.

Nel nostro caso infatti, il div sarebbe inutile dato che l'elemento contenitore ci serve solo perché return vuole un solo elemento.

aprire un progetto creato in precedenza

Tutte le volte che dovremmo riaprire un progetto già creato ed eseguirlo, NON dobbiamo rieseguire Vite, ma solo posizionarci nella giusta cartella con il terminale ed eseguire l'applicazione:

1

Una volta aperto VSCode, mostriamo il terminale (CTRL + ò) e verifichiamo in quale cartella siamo. Dobbiamo spostarci nella cartella del progetto nella quale è presente il file index.html. Se ad esempio siamo sopra di un livello, digitiamo: `cd` seguito dal nome della sotto cartella.

2

Eseguire il progetto:
digitiamo `npm run dev` per avviare l'applicazione.

Verrà aperta la finestra del browser con l'applicazione in esecuzione.
Ad ogni modifica con salvataggio, il browser si aggiornerà con le modifiche.

cenni storici utili

evoluzione del front-end nei decenni



Sviluppo web semplice con pagine web statiche, js manipolava il DOM.
Disorganizzazione complessiva e cattiva gestione del codice con la crescita di complessità del progetto.

Nascono le prime librerie js:
2006: jQuery rivoluziona il front-end e per un decennio è la libreria incontrastata.
Con il suo motto “write less, do more” semplifica la manipolazione del DOM e AJAX.
Non risolve la gestione dello stato dei componenti.

L’era dei framework e delle SPA (single page application):
una sola pagina che non viene ricaricata, ma il cui contenuto viene aggiornato dinamicamente in base a cosa succede.
2010: con Angularjs di Google si può creare di SPA con MVC, ma è molto complesso...

2013: Facebook crea la libreria React che introduce i componenti e il virtual DOM.
2014: nasce Vue, un framework come Angular, ma flessibile come React.
2016: rinasce Angular
Oggi: si lavora con i componenti e nascono nuovi framework e strumenti (Vite, Next, ecc)

Librerie e framework

differenze, pro, contro

The diagram illustrates the differences between libraries and frameworks. At the top, two purple ovals represent 'librerie' (left) and 'framework' (right). Below them are the logos for React (blue atom icon), Angular (red hexagon with white 'A'), and Vue.js (green and white 'V'). The main content area is divided into four sections corresponding to these technologies, each with a title, a green checkmark icon, and a list of pros and cons.

librerie

framework

React PRO ✓

- libreria = progetto più leggero
- curva di apprendimento più rapida
- flessibilità, modularità
- interfacce e riutilizzabilità
- componenti
- facile integrazione
- reattività grazie al virtual DOM

React CONTRO ✗

- servono librerie aggiuntive (Redux, React Router) per aggiungere funzionalità

Angular PRO ✓

- struttura completa: non servono librerie aggiuntive
- Two-way data binding
- coerenza del progetto

Angular CONTRO ✗

- pesantezza del progetto
- curva di apprendimento ripida
- complesso su piccoli progetti
- meno flessibile

Vue.js PRO ✓

- unisce la semplicità di React con la completezza di Angular
- essendo framework non necessita di librerie aggiuntive
- curva di apprendimento rapida

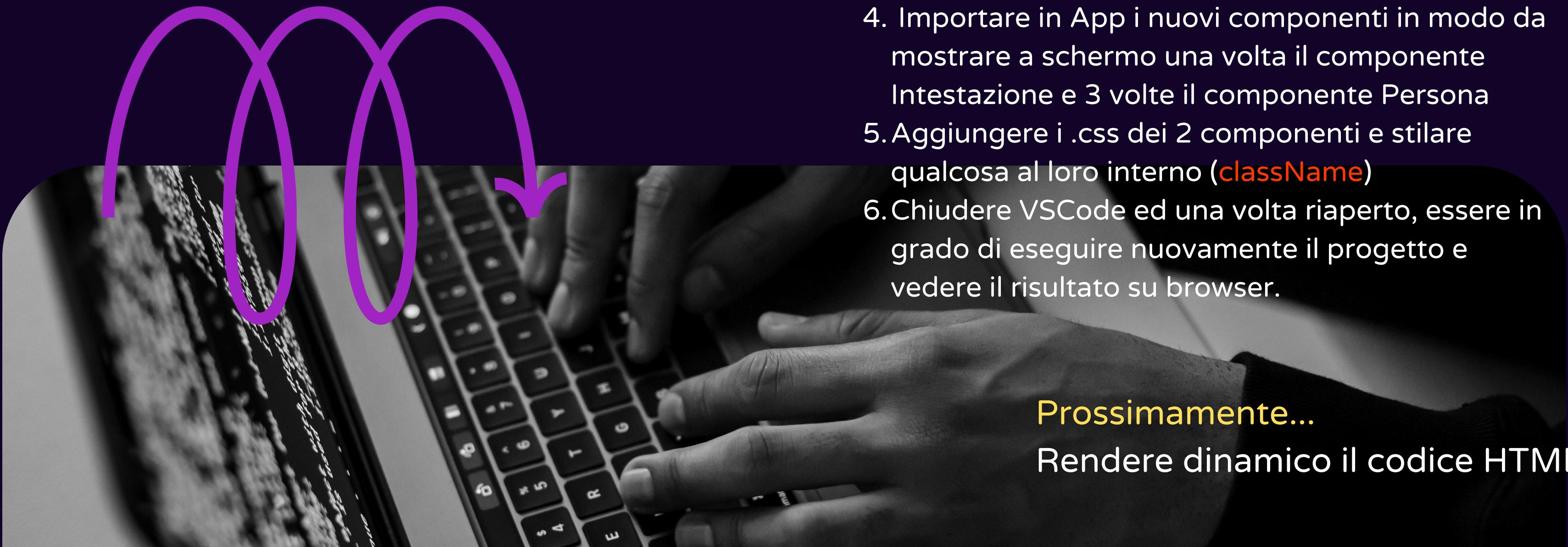
Vue.js CONTRO ✗

- in Italia a livello lavorativo non è ancora sufficientemente richiesto ed usato da grandi aziende, ma più da piccole realtà e startup

Esercizi e Prossimi passi

Esercizi per consolidare le competenze

1. Creare un nuovo progetto React js con il tool Vite
2. Ripulire il codice standard presente nella pagina (componente App)
3. Creare 2 nuovi componenti: Intestazione e Persona. Il primo conterrà alcuni tag html fra cui un h1; il secondo conterrà 2 paragrafi con il nome di una persona e un messaggio di saluto. (usare i fragmets)
4. Importare in App i nuovi componenti in modo da mostrare a schermo una volta il componente Intestazione e 3 volte il componente Persona
5. Aggiungere i .css dei 2 componenti e stilare qualcosa al loro interno (**className**)
6. Chiudere VSCode ed una volta riaperto, essere in grado di eseguire nuovamente il progetto e vedere il risultato su browser.



Prossimamente...

Rendere dinamico il codice HTML



GRAZIE
mILLE!

0%

100%

