Università degli Studi di Palermo

Edgemony

TECNOLOGIE FRONT END

REACT.JS

CASIMIRO (CAS) P. CIANCIMINO

CHE COS'È...

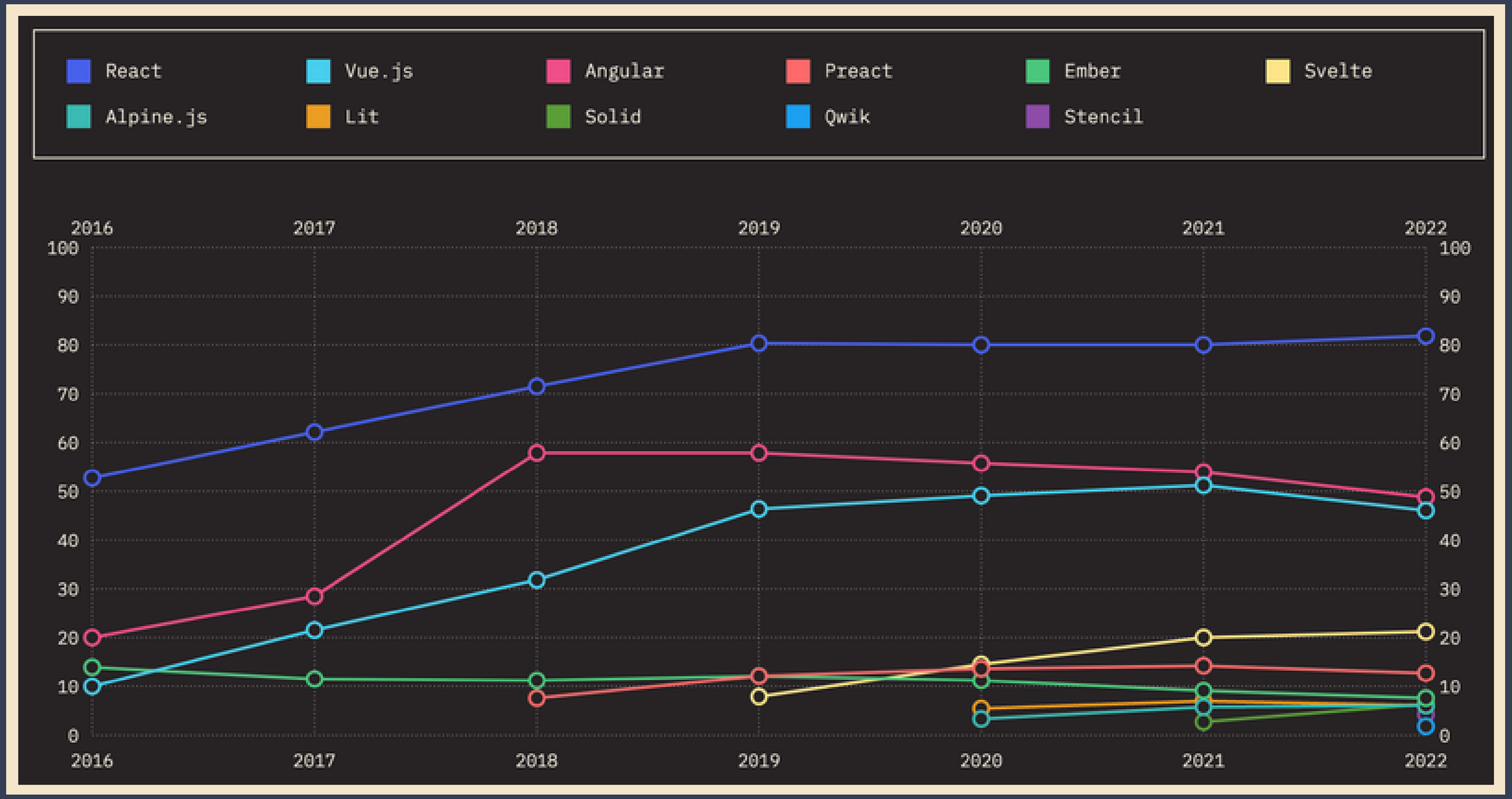
IL FRONT-END DEVELOPMENT?

Il front-end development web è come la **scenografia** di un teatro: **è ciò che gli spettatori vedono e interagiscono**, come l'**aspetto estetico** e le **funzioni** visibili di un sito web, mentre il **back-end** è il regista e gli attori dietro le quinte.

TECNOLOGIE

Ce n'è per tutti i gusti. Il mondo del web front-end è in continua espansione. Frameworks, librerie, tools: sono tutti accomunati in qualche modo eppure sembrano così diversi...

VANILLA JS	ANGULAR	REACT
VUE.JS	PREACT	EMBER
SVELTE	ALPINE.JS	LIT
SOLID	QWIK	STENCIL



https://2022.stateofjs.com/en-US/usage

I > COMPETITORS

ANGULAR

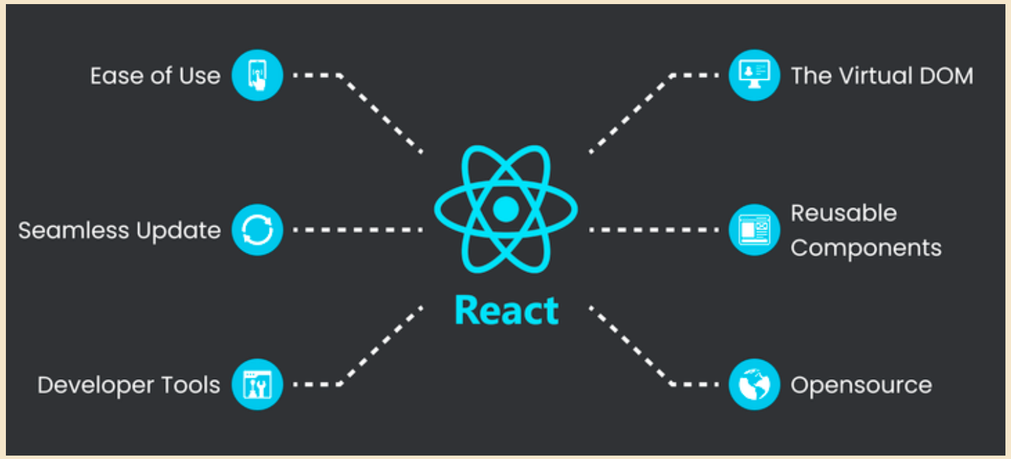
Per progetti aziendali di **grandi dimensioni**, con una struttura rigida e una curva di apprendimento più ripida. Il suo sistema di moduli e la **dependency injection** facilitano la creazione di applicazioni scalabili e modulari. By **Google**

REACT

Libreria flessibile e adatta a progetti di varie dimensioni, **buona curva di apprendimento**. **leggera**, **veloce** e altamente **personalizzabile** grazie al suo vasto ecosistema di pacchetti e librerie di terze parti. By **Meta**

VUE.JS

Adatto a progetti di piccole e medie dimensioni*, estendibile per adattarsi a esigenze più complesse. Vue offre un sistema di **componenti** simile a React, ma con sintassi semplice e curva di apprendimento più veloce. By **Evan You**



REACT.JS

The library for web and native user interfaces

https://react.dev

+19,331,904

Downloads settimanali

https://www.npmjs.com/package/react

- **Componenti riutilizzabili** => interfaccia modulare
- **Programmazione dichiarativa** => maggiore leggibilità del codice
- **Virtual DOM** => aggiornamento efficiente dell'interfaccia utente
- **Gestione dello stato** => reattività semplificate
- **Multipiattaforma** => inclusi web, mobile e desktop
- **Vasto ecosistema e comunità attiva**
- **Riduzione dei costi di sviluppo**
- **Velocità e performance ottimizzate**
- **Adattabilità a progetti di diverse dimensioni e complessità**
- **Velocità e performance ottimizzate**

LE CARATTERISTICHE

GLI STRUMENTI FONDAMENTALI

I COMPONENTI

Blocchi di **codice riutilizzabile**, rappresentano parti dell'**interfaccia utente di un'applicazione**. Essi permettono di suddividere l'interfaccia in pezzi indipendenti e gestibili. Ogni componente ha il proprio stato e le proprie proprietà (**props**), che consentono di **personalizzarne** il comportamento e l'aspetto.

GLI HOOKS

Una funzionalità introdotta in **React 16.8** che permette di utilizzare lo stato e altre funzionalità di React all'interno dei **componenti funzionali**.
Più comuni sono **useState**, per gestire lo stato locale del componente, e **useEffect**, per gestire gli effetti collaterali, come le chiamate API o le sottoscrizioni.

IL DIETRO LE QUINTE

I CONCETTI FONDAMENTALI

ONE DIRECTION DATA FLOW

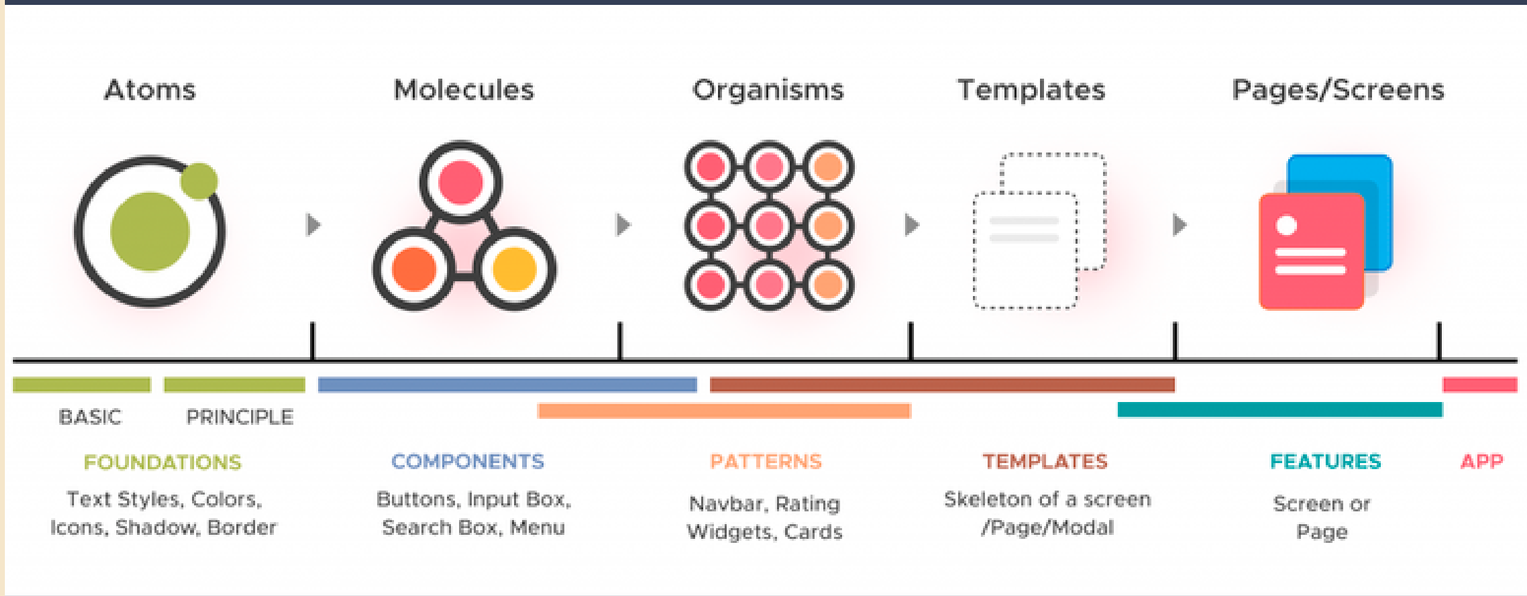
Stabilisce che i dati all'interno dell'applicazione **scorrono in una sola direzione**, dai componenti "**genitore**" ai componenti "**figlio**". I componenti genitore passano i dati ai figli attraverso le proprietà (**props**)

VIRTUAL DOM

Rappresentazione **leggera e in-memory** del DOM reale del browser. Quando lo stato di un componente cambia, React crea una nuova versione del Virtual DOM e la confronta con la versione precedente, utilizzando un algoritmo di differenziazione. In seguito, React **applica solo le modifiche minime** necessarie al DOM reale.

PENSARE IN REACT

Il design atomico in React è come costruire con i **LEGO**: gli **atomi** sono i mattoncini di base, le **molecole** sono piccoli gruppi di atomi uniti, gli **organismi** sono sezioni più complesse formate da molte molecole, i **template** sono le istruzioni per assemblare gli organismi e gli elementi di **layout** sono le base sulla quale poggia l'intera costruzione. Unendo questi elementi insieme, è possibile creare interfacce utente complesse e strutturate in modo modulare e riutilizzabile.



Atomi: come pulsanti, input o etichette.

Molecole: come moduli di input o liste di elementi.

Organismi: come intestazioni, barre laterali o aree di contenuto.

Template: definiscono la struttura di layout e il flusso dell'interfaccia.

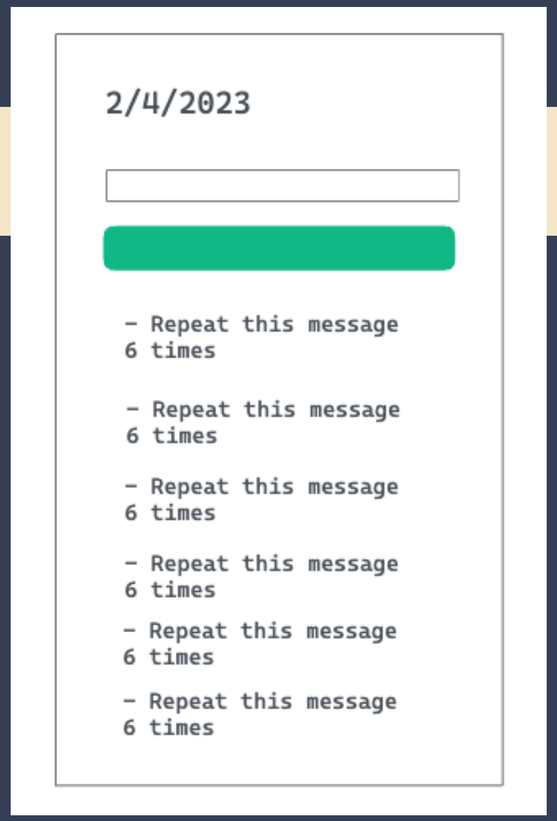
Layout: rappresentano l'interfaccia utente finale, unendo template e data per creare pagine complete e funzionali.

COME SI COSTRUISCE

UNA SEMPLICE TODO-APP

COSA CI SERVE?

- Un gestore di pacchetti, per Node e co. (**Vite.js**)
- Componenti separati
- Gli Hooks fondamentali (**useState** - **useEffect**)
- Gestire la chiamata esterna (*https://dummyjson.com/todos*)
- Un poco di colore e stile (buon vecchio caro **CSS**)



Definizione del piano d'attacco!

- Visualizzare la data giornaliera
- Catturare le todos presente in un server
- Possibilità di aggiungere nuove todos
- Possibilità di rimuovere vecchie todos
- ...

Università degli Studi di Palermo

Edgemony

GRAZIE

ADESSO PROGRAMMIAMO!

Per qualunque informazione non esitate a contattarmi, mi trovate su **LinkedIn** oppure scrivetemi in privato a **cp.ciancimino@gmail.com**