Vue.js è framework JavaScript progressivo per costruire interfacce utente.

Con "Progressivo" si intende che è possibile usarlo in modo graduale: iniziare integrandolo in una singola pagina HTML, oppure costruire una SPA (Single Page Application) completa con routing, state management e build tool avanzati.

Cenni storici

Vue è stato creato nel **2014** da **Evan You**, ex sviluppatore di Google.
L’idea nacque dal desiderio di prendere le cose migliori di Angular e renderle più leggere, semplici e flessibili.

Da allora, Vue è cresciuto costantemente grazie a una community attiva e a una documentazione chiara e accessibile.
Oggi è uno dei framework più amati e usati al mondo, insieme a React e Angular.

Viene scelto da molti sviluppatori anche per la sua **curva di apprendimento** dolce.

Perché scegliere Vue?

- Facile da imparare
- Sintassi pulita e intuitiva
- Reattività automatica.
- Componente-based
- Ecosistema completo: routing, gestione dello stato, strumenti di sviluppo e altro.

Vue.js vs Angular e React

Caratteristica	Vue.js	React	Angular
Tipo di tecnologia	Framework progressivo	Libreria per le interfacce UI	Framework completo
Paradigma	Declarativo, basato su componenti	Declarativo, basato su componenti	Declarativo, architettura MVC-ish
Linguaggio principale	JavaScript (con supporto TypeScript)	JavaScript (con supporto TypeScript)	TypeScript (obbligatorio)
Sintassi JSX/Template	Template HTML + direttive	JSX (HTML in JS)	Template HTML con sintassi estesa
Curva di apprendimento	Facile	Media	Ripida
Gestione dello stato	Vuex / Pinia	Redux / Zustand / altri	Integrata (RxJS + Services)
Routing	Ufficiale (Vue Router)	Esterno (React Router)	Integrato (Angular Router)
Tooling	Vite, Vue CLI	Vite, CRA, Next.js	CLI ufficiale molto completa
Comunità e diffusione	Grande e in crescita	Molto ampia	Ampia ma più corporate

Create App

Per avviare un’app Vue, usiamo la funzione createApp().
È il punto di partenza dove dichiariamo i dati, i metodi e le funzionalità della nostra app.

Il suo primo argomento è un oggetto, ed suo interno potrai inserire metodi e proprietà specifici per sfruttare al meglio le funzionalità di Vue(Data(), methods, computed ecc…)

```
Vue.createApp({
    data() {
        return {
            //variabili da usare nell'interfaccia
        }
    },
    methods: {
        //metodi da usare nell'interfaccia
    }
}).mount('#app')
```

Data

La funzione data() restituisce un oggetto con le variabili che vogliamo usare nel template.

Vue rende queste variabili reattive, cioè ogni volta che cambiano, l’interfaccia si aggiorna automaticamente.

```
Vue.createApp({
    data() {
        return {
            //variabili da usare nell'interfaccia
        }
    },
    methods: {
        //metodi da usare nell'interfaccia
    }
}).mount('#app')
```

Methods

Dentro methods definiamo le funzioni che possono essere chiamate dagli eventi del template (es. @click).

Ogni metodo ha accesso ai dati restituiti da data() tramite this.

```
Vue.createApp({
    data() {
        return {
            //variabili da usare nell'interfaccia
        }
    },
    methods: {
        //metodi da usare nell'interfaccia
    }
}).mount('#app')
```

v-on:event

Quando vogliamo intercettare un’azione dell’utente, come un click su un pulsante o la pressione di un tasto, in Vue usiamo la direttiva v-on.

v-on serve per collegare gli eventi del browser ai metodi della nostra app.
In pratica, dice a Vue: “Quando succede questo evento, esegui questa funzione.”

Con v-on possiamo rispondere a:

- click del mouse
- cambiamenti in un input
- invii di un form
- movimenti del mouse
- pressione dei tasti

e molti altri eventi standard del DOM