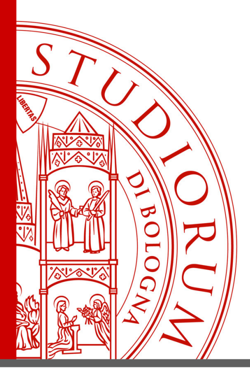


Framework di sviluppo front-end basati su componenti – parte I

Angelo Di Iorio

Università di Bologna

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Sistemi basati su componenti

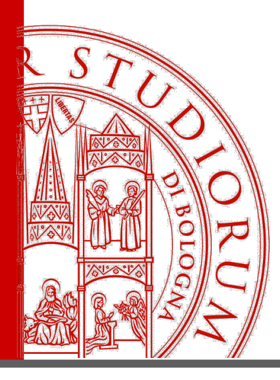
- *Angular*, *Vue* e *React* sono framework client-side **basati su componenti**
- L'applicazione è concepita come un insieme di **componenti indipendenti e riutilizzabili** che comunicano tra di loro
- Gli elementi di una **stessa componente** sono sviluppati in **maniera integrata** e divisa tra componente e componente
- Il framework fornisce meccanismi per **mettere in comunicazione le componenti**, per visualizzarle e interagirvi
- Un **componente** include quindi **frammenti HTML**, **regole CSS** e **codice Javascript** complessivamente **autosufficienti** e necessari per svolgere le funzioni di quel componente
- I componenti hanno struttura gerarchica e possono contenere quindi altri componenti. Esempi: *main area*, *sidebar*, *footer*, *barra di navigazione*, *maschera di ricerca*, *calendario*, *poll*, ecc.

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Framework component-based

- Questi sistemi non aggiungono funzionalità a Javascript ma aiutano gli sviluppatori ad usare le funzionalità di JS
- Dalla documentazione di Mozilla: *They don't bring brand-new powers to JavaScript; they give you easier access to JavaScript's powers so you can build for today's web*
- Ogni framework ha le sue specificità (e vincoli) ma molti tratti sono in comune
- Curiosità: uso dei framework in base allo score su GitHub e Stackoverflow: https://hotframeworks.com/

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



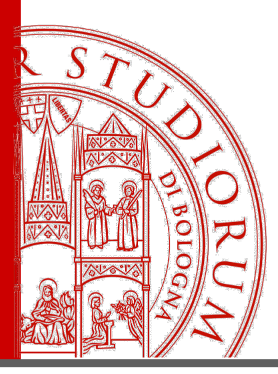
Template e approccio dichiarativo

- L'interfaccia dell'applicazione, o meglio di ogni componente dell'applicazione, è descritta attraverso un **template dichiarativo**
- Il template combina elementi HTML (e CSS) con istruzioni specifiche del framework per costruire l'interfaccia
- Il framework si occupa di **manipolare il DOM dietro le quinte** e produrre il risultato finale

```
<div id="welcome">
  <p v-if="sayWelcome">Welcome to Vue.js <b>{{name}}</b>! </p>
</div>
```

```
return (
<p>Ciao <b>{this.props.nome}</b>, benvenuto in React.js!</p>
 )
```

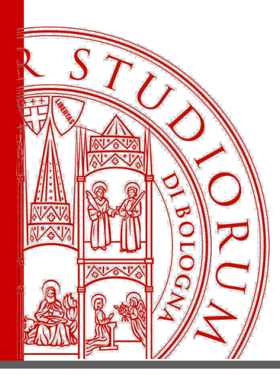

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Pattern MV* e binding

- Netta divisione tra i dati (model) e la loro visualizzazione (view) che viene modificata in seguito alla modifica sui dati
- I dati possono essere memorizzati nella pagina o nell'applicazione stessa o caricati via Ajax ed elaborati client-side
- Basati o ispirati ai pattern *MVC (Model View Controller)* o *MVVM (Model-View-Viewmodel)*
- **Binding (legame) monodirezionale**: la modifica ad una proprietà dell'applicazione (dati) implica una modifica dell'interfaccia (view)
- **Binding (legame) bidirezionale:** se un elemento dell'interfaccia è modificabile, allora cambiando il valore di quell'elemento cambia anche la corrispondente proprietà dell'applicazione

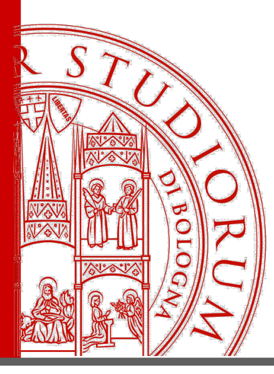
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Routing

- Molte pagine Web oggi sono costruite dinamicamente *client-side* e combinano dati ottenuti via Ajax
- La navigazione interna all'applicazione non richiede il caricamento di intere pagine (routing server-side) ma l'aggiornamento di parti di pagine
- Si parla di *single-page-applications* (SPAs) in cui i contenuti sono caricati appunto in un'unica pagina
- I framework forniscono meccanismi semplificati per supportare la navigazione e associare a specifici URI il caricamento di frammenti di pagina (componenti)

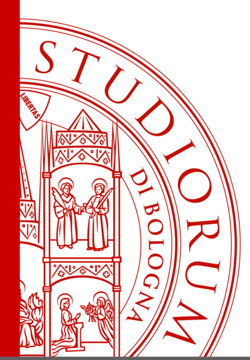
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



CLI tools

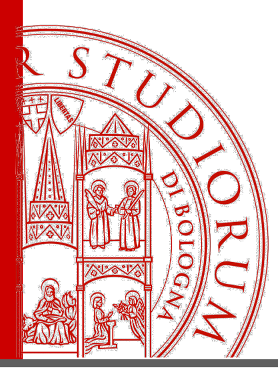
- I framework includono inoltre un vasto ecosistema di strumenti e script utili allo sviluppo:
  - Verifiche sul codice
  - Pre-processing (es. preprocessori CSS)
  - Server di test
  - Deploy ottimizzato per server in produzione
- Solitamente questi strumenti sono pacchetti Node.js installati e usati da linea di comando
- Specifici di ogni framework, si occupano di gestire configurazione e dipendenze
- Riprendo una frase di uno studente che ha riassunto benissimo*: "Il Web sta diventando command-line"*

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Introduzione a Vue.js

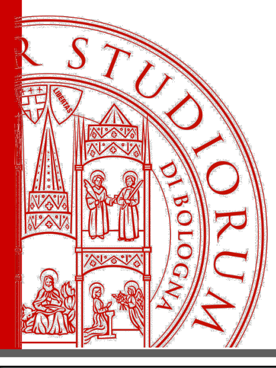
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Vue.js

- Home: https://vuejs.org/
- Creato nel 2014 da Evan You, uno degli sviluppatori di Angular. Meno complesso e ricco di Angular "by design"
- Struttura semplice, integrazione progressiva e bassa curva di apprendimento
- Si arricchisce di moduli separati e autonomi che possono essere aggiunti alle applicazioni, tra cui:
  - *Vue Router*: per fare *routing*
  - *Vuex:* per gestire lo stato di applicazioni più complesse e trasferire informazioni tra componenti
- Inoltre ***Vue CLI*** è un insieme di **tool da linea di comando** a supporto di sviluppo e test (maggiori dettagli in slide successive)

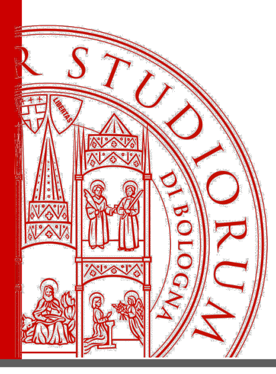
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Hello World

```
…
 <head>
    …
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"/>
   …
</head>
 <body>
    <div id="helloworld">
        <p>Welcome <b>{{ name }}</b>!!! </p>
    </div>
    …
    <script>
        const app = new Vue({
                        el: '#helloworld',
                        data: {name: 'Angelo'}
              })
        …
```

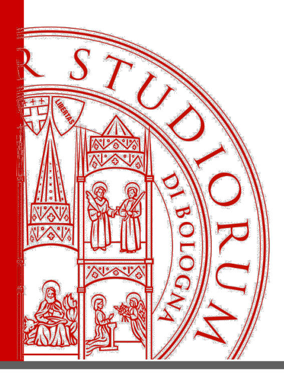
Welcome **Angelo**!!!



Istanza `Vue()`

- Per usare Vue si inizializza un'istanza `Vue(…)` a cui si passa un oggetto con le opzioni per configurare l'applicazione
- L'istanza Vue solitamente include altre componenti in una struttura gerarchica (che sono in realtà altre istanze Vue con un nome)
- Il framework **collega** il **DOM** con i **dati caricati,** attraverso un'operazione di **binding**: l'applicazione è ora **reattiva** e le modifiche alle proprietà sono riflesse nell'interfaccia
- Ispirato al pattern *MVVM (Model-View-Viewmodel)* associa un oggetto Vue (che memorizza le informazioni nel modello) ad una parte di sorgente HTML (vista) e li mantiene allineati

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Oggetti `el` e `data`

- Le opzioni sono passate tramite un oggetto che contiene diverse proprietà e metodi, tra cui:
  - `el`: indica il frammento di HTML su cui Vue ha il controllo; può essere un selettore CSS o un elemento HTML, ad esempio recuperato dal DOM
  - `data`: l'oggetto che l'istanza Vue sta osservando e le cui modifiche saranno propagate nell'interfaccia
    - oggetto JS che quindi può essere ulteriormente strutturato; <u>le sue proprietà possono far riferimento ad altri oggetti, array, etc.</u>
    - inoltre, poiché siamo in ambiente Javascript, si possono bloccare modifiche ad un oggetto attraverso la funzione `Object.freeze(obj)`

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Usiamo dati strutturati

```
corsiADI = [
        {nome : "Programmazione", semestre: 1},
        {nome : "Informatica", semestre: 1},
        {nome : "Tecnologie Web", semestre: 2}
        ]

 const listCorsi = new Vue({
        el: '#corsi',
        data: {
                corsi: corsiADI
                }
        })
```

```
<div id="corsi">
               {{corsi}}
</div>
```

[{ "nome": "Programmazione", "semestre": 1 }, { "nome": "Informatica", "semestre": 1 }, { "nome": "Tecnologie Web", "semestre": 2 }]

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Con un template HTML

```
corsiADI = [
        {nome : "Programmazione", semestre: 1},
        {nome : "Informatica", semestre: 1},
        {nome : "Tecnologie Web", semestre: 2}
        ]

 const listCorsi = new Vue({
        el: '#corsi',
        data: {
                corsi: corsiADI
              }
        })
```

Programmazione - 1° semestre

Informatica - 1° semestre

Tecnologie Web - 2° semestre

```
<div id="corsi">
        <p v-for="corso in corsi">{{corso.nome}} -
{{corso.semestre}}°  semestre</p>
</div>
```

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Miglioriamo la veste grafica

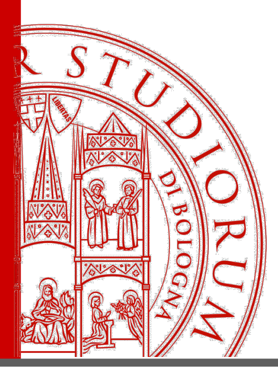
- Alla fine Vue produce uno o più frammenti HTML il cui stile può essere quindi definito in CSS (framework inclusi)

```
<div class="corso" v-for="corso in corsi">
      <div>{{corso.nome}}</div>
      <div>{{corso.semestre}}°  semestre</div>
</div>
```

```
div.corso {
         display: grid;
         grid-template-columns: 1fr 1fr 1fr
         width: 40vw;
         background-color: #d9ffb3;
         border-radius: 10px;
         border: 1px solid #4d9900;
         padding: 1vw;
         margin-bottom: 1vh;
      }
```

Programmazione	1° semestre
Informatica	1° semestre
Tecnologie Web	2° semestre

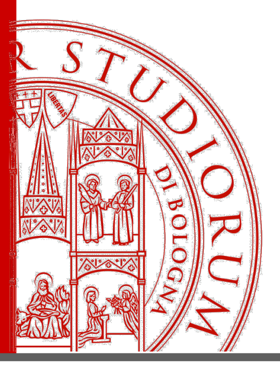
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Template e direttive (1)

- Il linguaggio di template di Vue è un'estensione della sintassi *Mustache* per interpolare variabili, ossia sostituirle con i valori nell'applicazione
- I simboli `{{` e `}}` sono usati per racchiudere l'espressione da interpolare
- E' possibile usare, oltre a variabili e proprietà, anche espressioni complesse e funzioni JS (ma sempre una singola espressione in ogni interpolazione)
- ```
  Welcome {{name.toLowerCase() + " - " + name[0].toUpperCase()}}!!! 
  ```
- Vue permette inoltre di aggiungere **direttive**, ossia attributi speciali che iniziano per **v-*** e che esprimono istruzioni per elaborare il DOM dopo aver valutato l'espressione che contengono

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI
  ```



Template e direttive (2)

v-if: il blocco è renderizzato solo se l'espressione è *truthy*

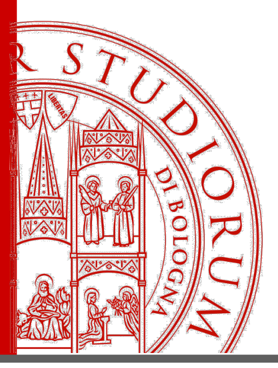
```
<div id="welcome">
        <p v-if="sayWelcome">Welcome <b>{{name}}</b>!!!
        </p>
</div>
```

v-bind: lega uno o più attributi a un elemento/dato

- Se nel binding è presente una proprietà dell'istanza Vue la modifica di questa proprietà modifica l'attributo
- Nell'esempio l'attributo `data-anno` assume il valore della proprietà `anno`

```
<div id="corsi" v-bind:data-anno="anno">
    A.A.: {{anno}}
</div>
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



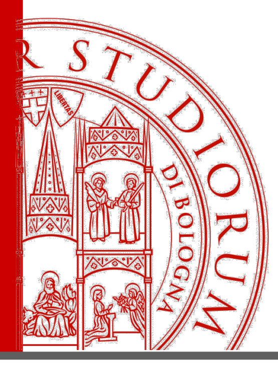
Template e direttive (3)

v-for: renderizza un elemento o un template più volte

- Si usa solitamente con vettori di oggetti
- Si definisce un alias usato nel blocco che sarà ripetuto
  - nell'esempio: ***corso*** è l'alias, ***corsi*** il vettore in input, i cui elementi sono oggetti con le proprietà ***nome*** e ***semestre***

```
<div v-for="corso in corsi">
      <div>{{corso.nome}}</div>
      <div>{{corso.semestre}}°  semestre</div>
</div>
```

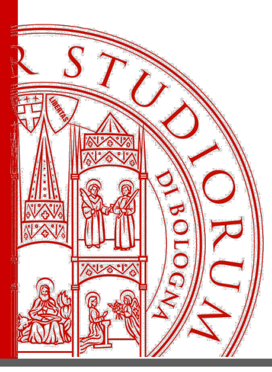

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Metodi e callback

- Negli esempi visti finora le opzioni passate all'istanza `Vue` contenevano solo proprietà ma è possibile anche aggiungere funzioni (metodi) :
  - da registrare come callback di eventi DOM (*click*, *mouseover*…)
  - da invocare all’interno della nostra istanza Vue, ad esempio per calcolare dati da visualizzare nell'interfaccia

```
const app = new Vue({
        el: '#welcome',
        data: {
                …
                alertCiao : function(){
                                alert('Ciao')
                        }
        …
```



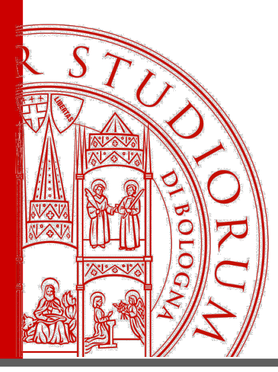
Template e direttive (4)

- La direttiva **v-on** permette di associare una funzione a un evento generato dall'interfaccia
- Sintassi: `v-on:<nome-evento>="<funzioneCallback>"`
- L'evento può essere predefinito (*click*, *mouseover*, ecc.) o personalizzato con un nome deciso dallo sviluppatore
- In questo secondo caso sarà necessario emettere l'evento in qualche punto dell'interfaccia (utile per far comunicare componenti, vedi prossime slide)

```
<button v-on:click="alertCiao">Ti saluto</button>
```

```
<div v-on:saluta="alertCiao">…</div>
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



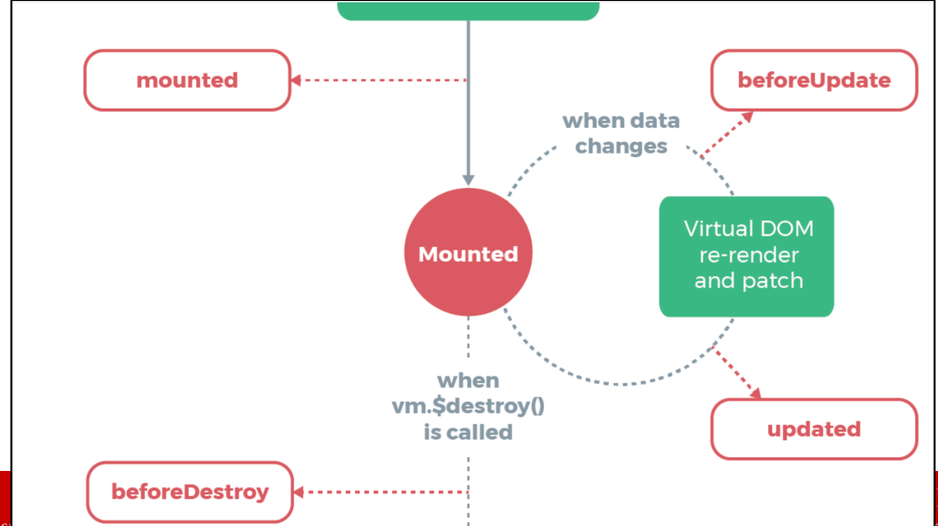
Ciclo di vita e *hooks*

- Oltre ad associare funzioni ad eventi sull'interfaccia, è possibile associarle anche a determinati momenti del ciclo di vita dell'istanza Vue
- Questi momenti sono pre-definiti e permettono allo sviluppatore di indicare azioni da compiere in corrispondenza
- Anche questo meccanismo è comune a molti framework, con nomi e dettagli del ciclo di vita diversi
- I principali in Vue:
  - *Created*: creazione dell'istanza, particolarmente utile per pre-caricare dati via Ajax
  - *Mounted*: l'istanza è associata a un elemento del DOM
  - *Updated*: le proprietà dell'istanza sono state modificate ed è necessario aggiornare l'interfaccia, o viceversa in caso di binding bidirezionale
  - *Destroyed:* istanza cancellata, ad esempio nel caso di direttiva `v-if` e componente rimosso dall'interfaccia

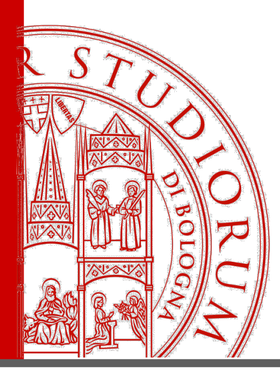
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

VirtualDOM: mount e update

- Vue (come React) utilizza un **VirtualDOM**, ossia una rappresentazione in memoria particolarmente efficiente che viene poi "mappata" sul **RenderedDOM** visibile all'utente
- L'operazione di *mount* esegue il collegamento tra l'istanza Vue (o una componente) e l'elemento del DOM
- I successivi cambiamenti sui dati saranno gestiti nel VirtualDOM e successivamente propagati nell'interfaccia



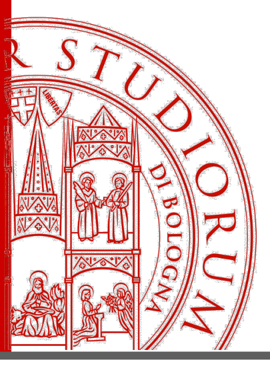
IL PRESENTE MATERIALE È RIS... ...O PER FINI NON ISTITUZIONALI



Richieste Ajax

- Apriamo una parentesi: siamo in ambiente JS, possiamo (dobbiamo) continuare a fare richieste asincrone per caricare dati via Ajax
- Possiamo usare *XMLHTTPRequest*, *fetch*, promesse, ecc.
- Una libreria molto usata è *axios*: https://axios-http.com/
- I framework si occupano di costruire e gestire l'interfaccia, il caricamento e l'elaborazione dati avviene "dietro le quinte"
- Quando i dati sono pronti e le proprietà dell'istanza modificate, Vue si occupa di aggiornare l'interfaccia di conseguenza

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Hook e richieste Ajax

```
const app = new Vue({
      el: '#welcome',
      data: {…},
      created: function() {
                    var myapp = this;
                    fetch("corsi.json")
                      .then(function(response) {
                          return response.json();
                          })
                      .then(function(data) {
                          myapp.corsi = data;
                    });
            },
      mounted: function() {
            alert("mounted!")
      }
…
```

Attenzione: necessario per accedere alle proprietà dell'istanza Vue nel .then

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

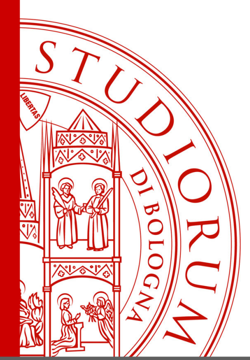
Binding bidirezionale e `v-model`

- La direttiva **v-model** permette di fare **binding bidirezionale**, ossia modificare i dati attraverso la view
- Particolarmente usato nella costruzione di form
- **Da notare**: diverso da *`v-bind`* che invece indica *binding unidirezionale*

```
<input v-model="nome" placeholder="Come ti chiami?">

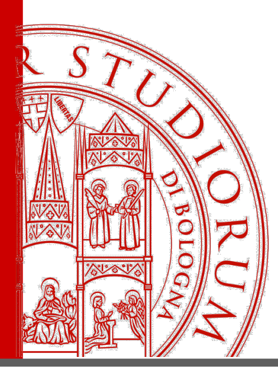
<p>Ciao <b>{{nome}}</b>, benvenuto in Vue.js!</p>
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Componenti

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Componenti

- Un'applicazione Vue è in realtà composta da diverse **componenti indipendenti** e **riutilizzabili**
- Un **componente** non è altro che un'**istanza Vue con un nome**. Di conseguenza accetta le stesse opzioni viste finora tra cui *data*, *methods*, *hooks*, ecc.
- La componente può essere inclusa in un'applicazione o un'altra componente con un **elemento *custom*** nel template:

```
<div id="welcome"><benvenuti></benvenuti></div>
```

- Due differenze da tenere in mente rispetto all'istanza Vue principale:
  - l'oggetto `el` non è previsto, visto che la componente è pensata per essere usata anche più volte e in punti diversi
  - `data` DEVE essere una funzione; questo per permettere ad ogni istanza della stessa componente di mantenere dati diversi

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Componenti: elementi principali

- Il metodo `Vue.component()` crea un componente, prendendo in input il *nome* e *un oggetto con le opzioni* (come per l'istanza principale)
- Tra queste opzioni `template` permette di specificare un frammento HTML da usare per **generare la vista**
- Il template può include altri componenti, che a loro volta ne possono includere altri in una struttura gerarchica
- **Attenzione**: per riusare un componente bisogna dichiararlo prima dell'istanza Vue in cui si usa (root o un altro componente)

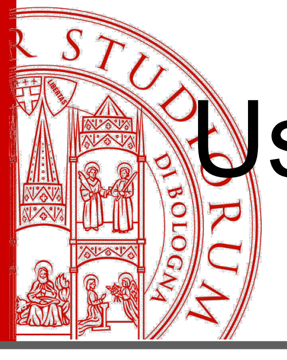
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

```
Vue.component('benvenuti', {
        template: `
        <h1>Benvenuti in Vue.js!</h1>
      `
      })
```

```
Vue.component('benvenuto', {
        props: ['nome'],
        template: `
        <p>Ciao <b>{{nome}}</b>, benvenuto in Vue.js!
             </p>
        `
      })
```

```
<div id="welcome">
    <benvenuti></benvenuti>
    <benvenuto nome="Angelo"></benvenuto>
    <benvenuto nome="Mario"></benvenuto>
</div>
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Usiamo i componenti nel nostro esempio

App principale

Componente **`<info-corsi>`**

Ciao **Angelo**, benvenuto in Vue.js!

Programmazione	Semestre: 1
Informatica	Semestre: 1
Tecnologie Web	Semestre: 2

Nota: avremmo potuto creare anche un ulteriore componente **`<info-corso>`**

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

```
const app = new Vue({
            el: '#welcome',
            data: {
                  nome: 'Angelo',
                  corsi: corsiADI,
...
```

Binding sull'attributo.
Viene passato il vettore `corsi` come parametro d_corsi
(si può usare anche lo stesso nome)

```
<div id="welcome">
  <p>Ciao <b>{{nome}}</b>, benvenuto in Vue.js
  <info-corsi :d_corsi="corsi"></info-corsi>
</div>
```

```
Vue.component('info-corsi', {
        props : ['d_corsi'],
        template: `
        <div class="infoCorsi">
        <div class="corso" v-for="corso in d_corsi">
          <div>{{corso.nome}}</div>
          <div>Semestre: {{corso.semestre}}</div>
       </div>
        </div>`
     }).
```

Dichirazione delle proprietà passate dalla componente padre (o root)

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

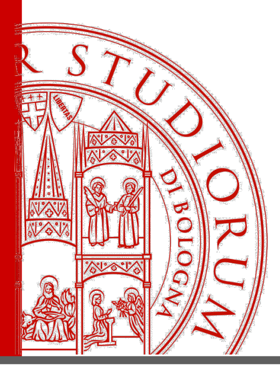
Passare valori ad un componente

- Per passare valori da una **componente padre** ad un **componente figlio** è necessario dichiarare le proprietà attraverso **props**
- Nel template le proprietà sono espresse come attributi dell'elemento *custom* che rappresenta il componente
- Il valore di questi attributi è legato ai dati tramite *binding*
- Le props saranno poi accessibili (ma non modificabili) dentro il componente

```
<info-corsi :d_corsi="corsi"></info-corsi>
```

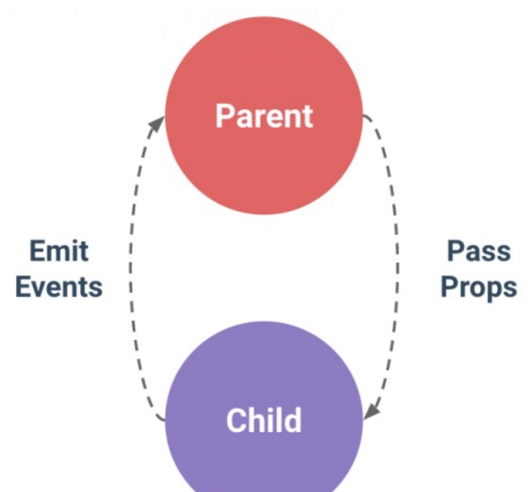
```
Vue.component('info-corsi', {
        props : ['d_corsi'],
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

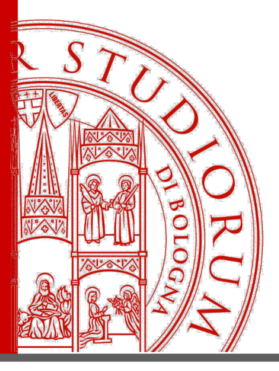


Props ed eventi

- Il flusso dei dati è monodirezionale **da padre a figlio:** le prop passate ad un componente possono essere modificate dall’esterno, ma non dal componente stesso
- Questo per isolare meglio le componenti ed evitare modifiche non desiderate
- Per comunicare dati nel verso opposto (da figlio a padre) si usano gli eventi:
  - Il componente padre registra una funzione di callback su un evento personalizzato
  - Il componente figlio emette quell'evento con `$.emit()`

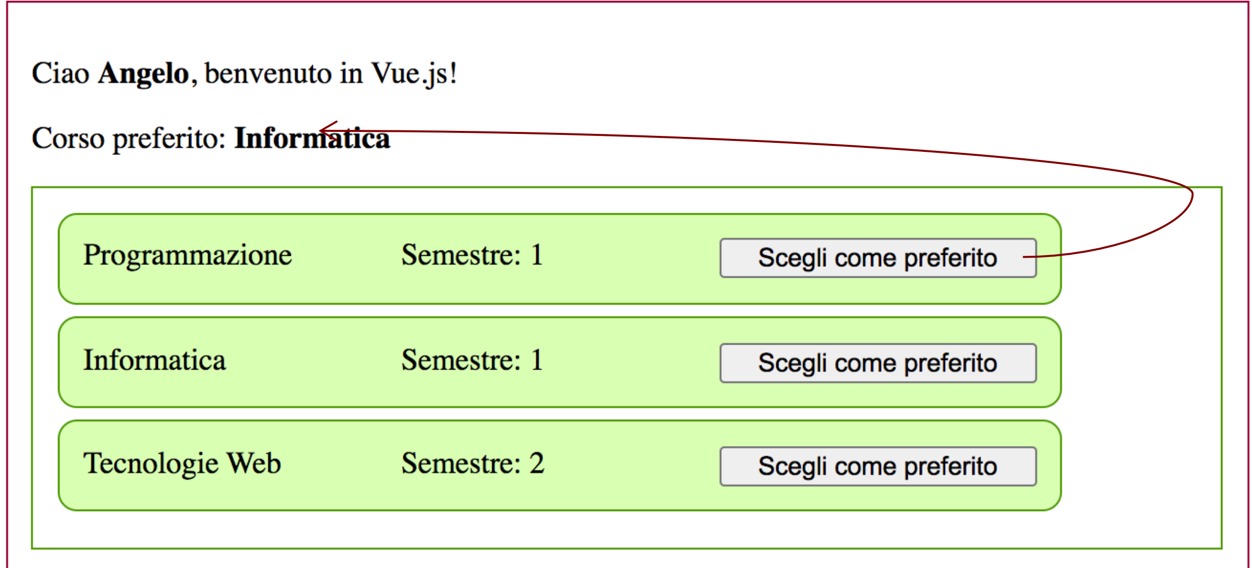


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Completiamo l'esempio

- Vogliamo mostrare nel componente padre il corso scelto con il bottone nel componente figlio
- Aggiungiamo una proprietà per memorizzare il corso scelto e facciamo comunicare le componenti dal basso verso l'altro tramite eventi (codice nella prossima slide)



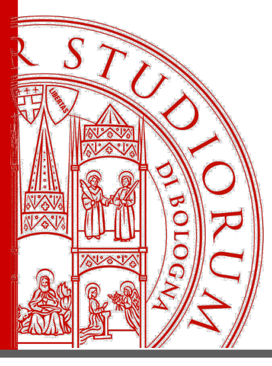
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

```
<p>Corso preferito: <b>{{preferito}}</b></p>
<info-corsi :d_corsi="corsi"
    v-on:preferito-evento="mostraPreferito" ></info-corsi>
```

```
const app = new Vue({
            el: '#welcome',
            data: {
                  nome: 'Angelo',
                  corsi: corsiADI,
                  preferito: undefined,
                  mostraPreferito : function(p){
                        app.preferito = p;
                        },
            …
```

```
<button v-on:click="$emit('preferito-evento', corso.nome)">
      Scegli come preferito</button>
```


IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Componenti e file `.vue`

- Vue permette anche di creare un unico file che include tutte le parti di uno stesso componente: template, stile, opzioni e comportamenti dinamici

Benvenuto.vue

```
<template>
  <div class="hello">
    <p>Ciao <b>{{nome}}</b>,
         benvenuto in
Vue.js!</p></div>
</template>

<script>
export default {
  name: 'Benvenuto',
  props: { nome: String }
}
</script>

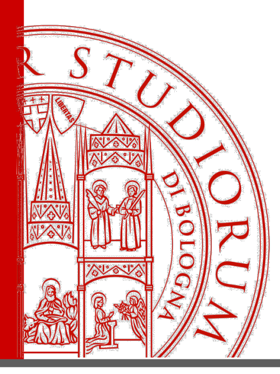
<style scoped>
       h1 {color:red;}
```

App.vue

```
<template>
  <div id="app">
    <Benvenuto nome="Angelo"/>
  </div>
</template>

<script>
import Benvenuto from
'Benvenuto.vue'

export default {
  name: 'App',
  components: {
    Benvenuto
  }
}
```



Conclusioni

- Vue è il più "leggero" tra i tre framework più usati oggi
- Meno usato di React.js ma in forte ascesa
- Il *core* si può estendere con moduli esterni, ad esempio per fare *routing* (*Vue Router*) o gestire lo stato globale delle applicazioni e la comunicazione e sincronizzazione tra molte componenti (*Vuex*)
- Molti dei concetti visti oggi sono comuni agli altri framework, anche se con sintassi e dettagli diversi

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI