

The European



Delphi Conference

26 maggio 2017 - ROMA



Applicazioni Web ultra-performanti con Vue.js e Delphi

Marco Breveglieri

The European Delphi Conference



Marco Breveglieri

Software & Web Developer @ABLS Team

Blogger (www.compilaquindiva.com)

Host @Delphi Podcast (www.delhipodcast.com)

and sushi eater! 🍣



Chi sono



The European Delphi Conference



Cos'è Vue.js?



Che cos'è Vue.js?



Vue.js è un ***progressive framework*** per la costruzione di interfacce utente



Che cos'è Vue.js?

Comparazione con altri
celebri framework
JavaScript

- Meglio chiamarla libreria, non framework
- Ha le prestazioni di React
- Usa convenzioni già viste in AngularJS (e altre librerie MVVM)



Cos'è Vue.js?

Incremental adoption

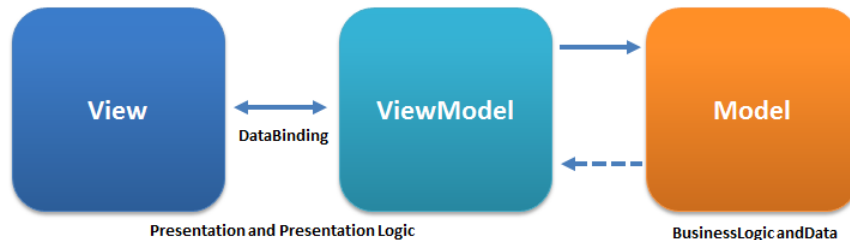
- Può essere gradualmente integrato in un'applicazione Web esistente
- Curva di apprendimento morbida



Cos'è Vue.js?

**Focalizzato sulla
«View»**

La libreria si concentra sulla gestione delle "viste" per generare e aggiornare la UI.



Cos'è Vue.js?

**Perfetto anche per SPA
sophisticate**

- Supporta i moderni tool di sviluppo JavaScript
 - Node, WebPack, Browserify, ES2016, ...
 - Approccio component-based
 - Disponibile *vue-cli*
- Numerosi componenti ed estensioni di terze parti



The European Delphi Conference



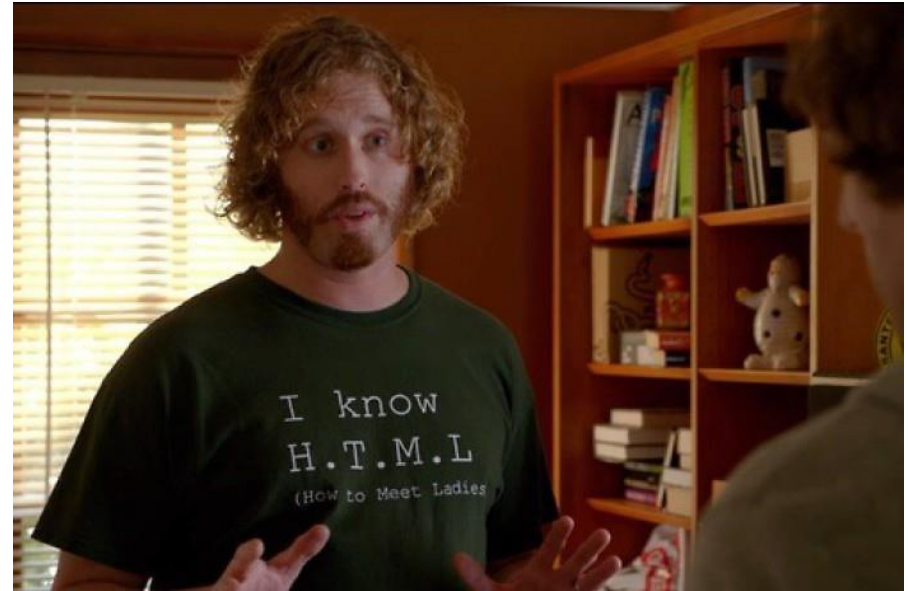
Perché Vue.js?



Perché Vue.js?

Accessibile

Ti basta conoscere
HTML, CSS e JavaScript.



Perché Vue.js?

Versatile

Può essere usato in progetti sia piccoli che di grandi dimensioni.



Perché Vue.js?

Performante

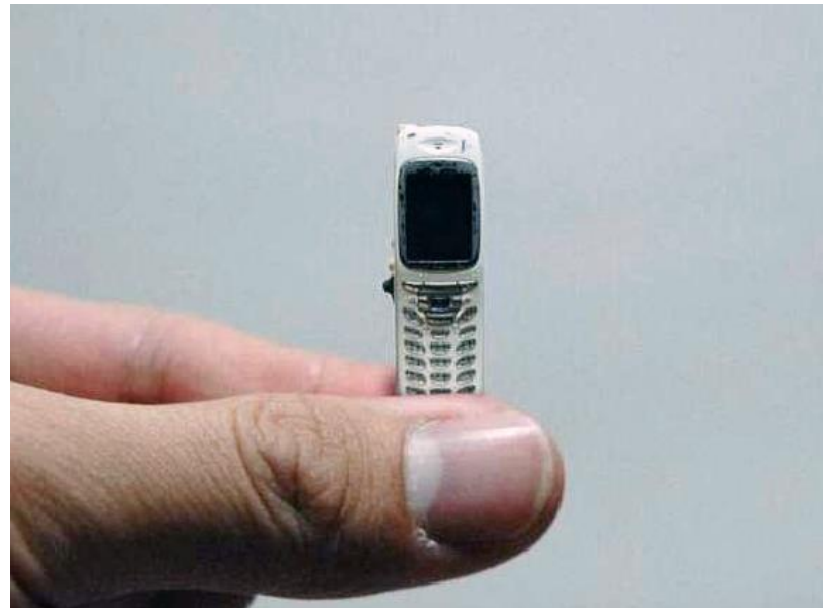
Estremamente veloce
grazie al proprio
Virtual DOM.



Perché Vue.js?

Compatto

Il runtime è molto piccolo: solo ~19Kb.



The European Delphi Conference



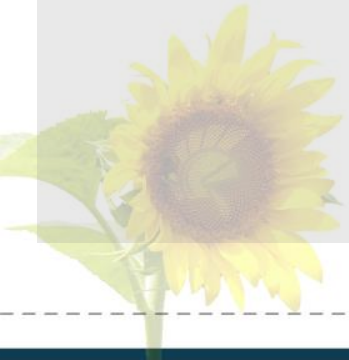
Primi passi



Primi passi

Importare lo script di Vue nella pagina:

```
<script src="https://unpkg.com/vue"></script>
```



Primi passi

Creare il markup dell'applicazione:

```
<div id="app">  
  <p>{{ message }}</p>  
</div>
```



Primi passi

Costruire un'istanza dell'oggetto Vue:

```
<script>  
  var app = new Vue({  
    el: "#app",  
    data: {  
      message: "Hello Vue!"  
    }  
  });  
</script>
```



The European Delphi Conference



DEMO



DEMO

Declarative Rendering

```
<div id="app">
  <p>{{ message }}</p>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "Hello Vue!"
    }
  });
</script>
```



DEMO

Attribute Binding

```
<div id="app">
  <p v-bind:title="message">
    Sposta qui il mouse per vedere il messaggio.
  </p>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "Hello Vue!"
    }
  });
</script>
```



DEMO

Conditionals

```
<div id="app">  
  <p v-if="visible">Ora mi vedi!</p>  
</div>
```

```
<script>  
  var app = new Vue({  
    el: "#app",  
    data: {  
      visible: true  
    }  
  });  
</script>
```



DEMO

Loops

```
<div id="app">
  <ol>
    <li v-for="todo in todoList">
      {{ todo.text }}
    </li>
  </ol>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      todoList: [
        { text: "Imparare Vue" },
        { text: "Integrarlo con Delphi" },
        { text: "Costruire applicazioni spaziali!" }
      ]
    }
  });
</script>
```



DEMO

Events

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse</button>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "REDRUM"
    },
    methods: {
      reverseMessage: function () {
        this.message = this.message.split("").reverse().join("");
      }
    }
  });
</script>
```




DEMO

Two-Way Binding

```
<div id="app">
  <p>{{ message }}</p>
  <input v-model="message"/>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "REDRUM"
    }
  });
</script>
```



DEMO

Filters

```
<div id="app">
  <p>{{ message | reverse | toUpper }}</p>
  <input v-model="message"/>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "redrum"
    },
    filters: {
      reverse: function (value) {
        return value.split("").reverse().join("");
      },
      toUpper: function (value) {
        return value.toUpperCase();
      },
    }
  });
</script>
```



DEMO

Computed Properties

```
<div id="app">
  <p>{{ reversedMessage }}</p>
  <input v-model="message"/>
</div>

<script>
  var app = new Vue({
    el: "#app",
    data: {
      message: "REDRUM"
    },
    computed: {
      reversedMessage: function () {
        return this.message.split("").reverse().join("");
      }
    }
  });
</script>
```



The European Delphi Conference



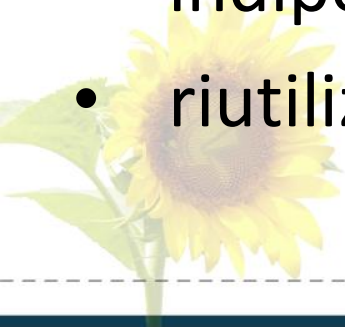
Componenti!



Componenti

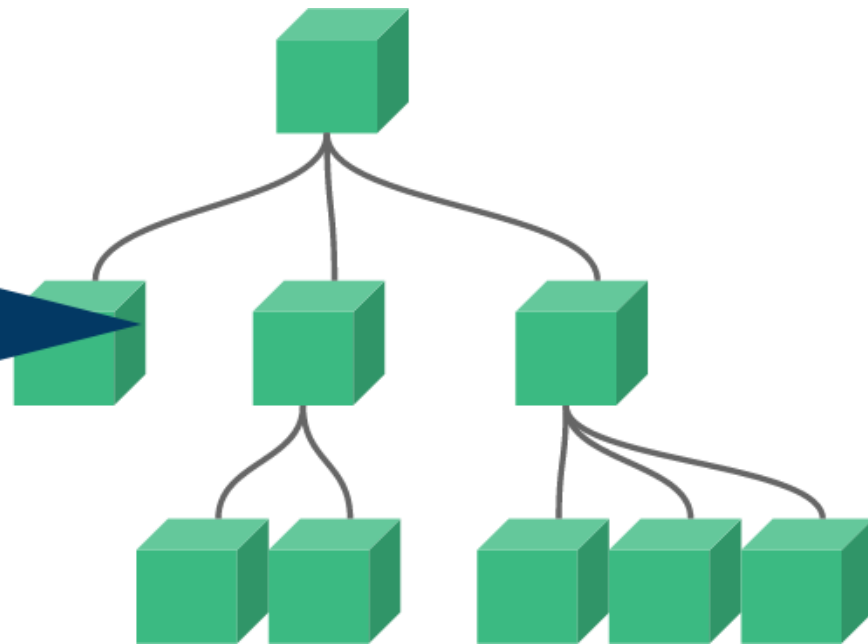
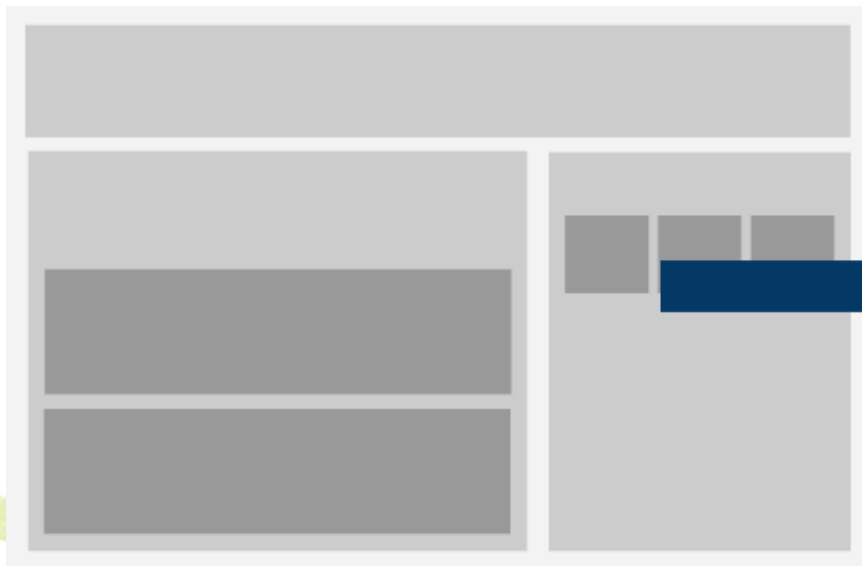
Ogni interfaccia utente può essere scomposta in **componenti**, ossia parti

- più piccole
- incapsulate
- indipendenti
- riutilizzabili



Componenti

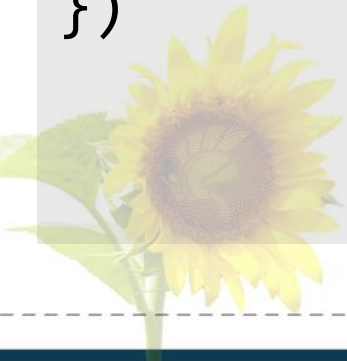
UI Decomposition



Componenti

Creare un componente è molto semplice...

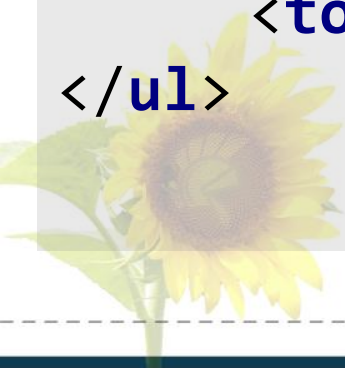
```
Vue.component('todo-item', {  
  template: '<li>Cosa da fare</li>'  
})
```



Componenti

...e anche utilizzarlo. 😊


```
<ul>  
  <todo-item></todo-item>  
  <todo-item></todo-item>  
  <todo-item></todo-item>  
</ul>
```



Componenti

Aggiungiamo proprietà per renderlo "configurabile"...

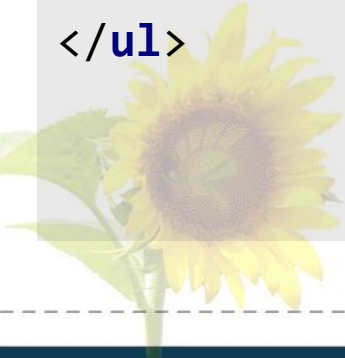
```
Vue.component('todo-item', {  
  props: ['task'],  
  template: '<li>{{ task.text }}</li>'  
})
```



Componenti

...e successivamente **valorizziamo la proprietà.**

```
<ul>  
  <todo-item v-bind:task="{ text: 'Task 1' }"></todo-item>  
  <todo-item v-bind:task="{ text: 'Task 2' }"></todo-item>  
  <todo-item v-bind:task="{ text: 'Task 3' }"></todo-item>  
</ul>
```

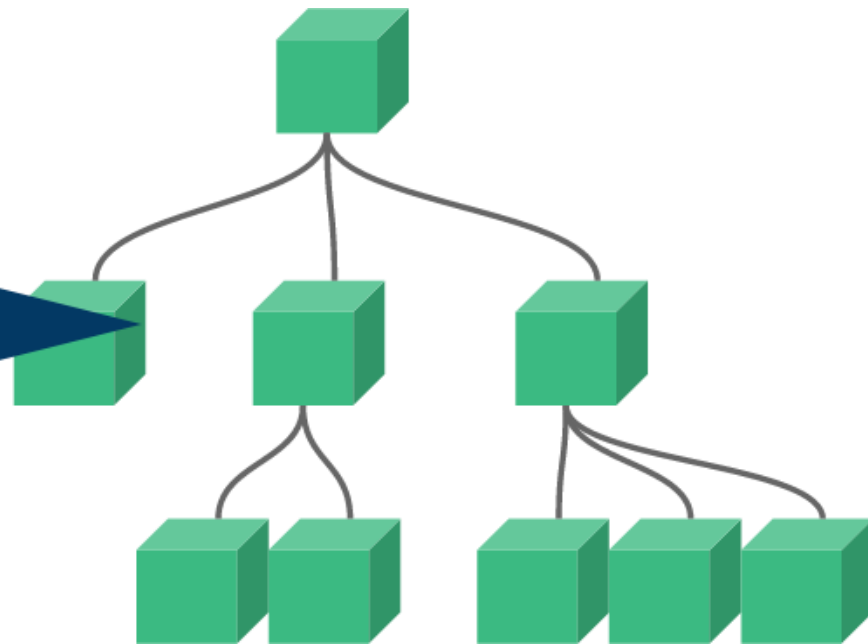
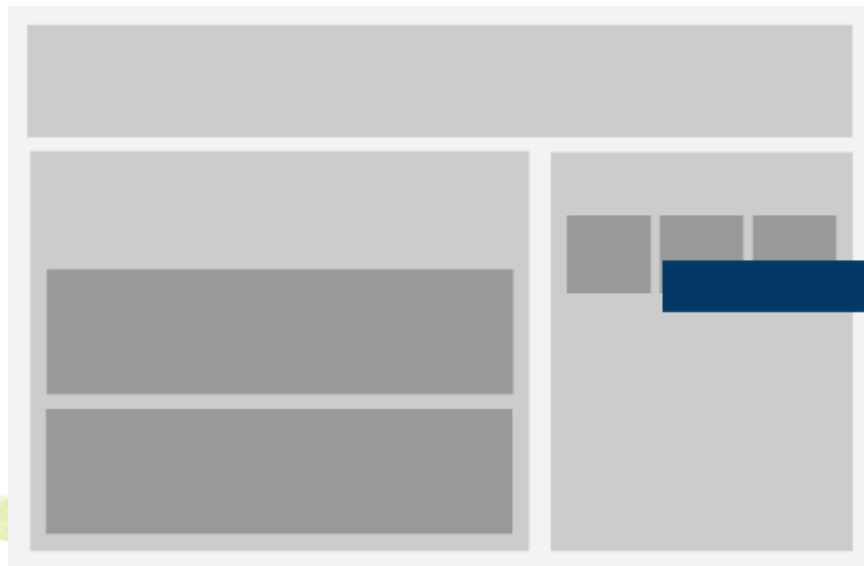


DEMO



Componenti

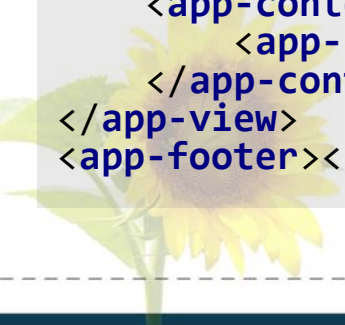
Ricordate lo schema iniziale?



Componenti

Possibile struttura di un'applicazione:

```
<app-title></app-title>
<app-nav>
  <app-menuitem></app-menuitem>
  <app-menuitem></app-menuitem>
  <app-menuitem></app-menuitem>
</app-nav>
<app-view>
  <app-sidebar>
    <app-contextmenu></app-contextmenu>
  </app-sidebar>
  <app-content>
    <app-content-title></app-content-title>
  </app-content>
</app-view>
<app-footer></app-footer>
```



The European Delphi Conference



Dietro le quinte



Dietro le quinte

Le istanze Vue sono dotate di **estensioni**.

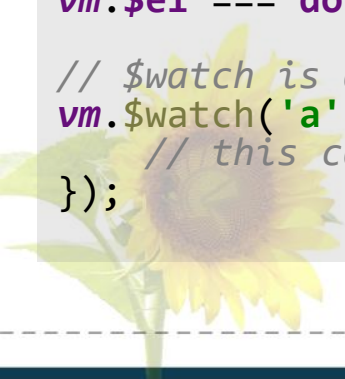
```
var data = { a: 1 };

var vm = new Vue({
  el: '#example',
  data: data
});

vm.$data === data; // -> true

vm.$el === document.getElementById('example') ; // -> true

// $watch is an instance method
vm.$watch('a', function (newVal, oldVal) {
  // this callback will be called when `vm.a` changes
});
```



Dietro le quinte

Gestione degli eventi
del ciclo di vita

- `created`
- `mounted`
- `updated`
- `destroyed`





**KEEP
CALM
AND
ASK
QUESTIONS**



Thanks!

