



Vue.js

Un'alternativa a ...



Walter Tommasi




<https://github.com/jtommy>



Progressive Framework



DOM Driven vs Data Driven



```
<div id="app">
  <form onsubmit="event.preventDefault()">
    <input type="text" id="text">
    <button type="submit" id="add">Add</button>
  </form>
  <ul id="list">
  </ul>
  <div id="content">
    Count: <span id="count">0</span>
  </div>
</div>
```

```
var input = $('#text')
var btnAdd = $('#add')
var list = $('#list')
var count = $('#count')
var index = 1
// btnAdd click listener
btnAdd.on('click', function() {
  // new btn (remove) + click listener
  var btn = $('<button>').text('x')
  btn.on('click', function(event) {
    // remove row
    var el = $(this).parent('li')
    var id = el.attr('data-id')
    $(el).remove()
    // refresh count
    count.text(list.find('li').length)
  })
  // append todo text
  $('<li>').attr('data-id', index).text(input.val()).append(btn).appendTo(list)
  // refresh count
  count.text(list.find('li').length)
  index++
  // reset input
  input.val('')
})
```



Perchè Vue.js ?

- Versatilità
- Performance
- Semplicità / Accessibilità



Rendering Dichiarativo

- Template / Render function
- Reattività
- DOM Virtuale

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

<script>
var app = new Vue({
  el: '#app',
  data: function() {
    return {
      message: 'Not a Bug Conf'
    }
  }
})
</script>
```



Direttive

- **condizionali e cicli**
 - v-if, v-else-if, v-else, v-show
 - v-for
- **binding e eventi**
 - v-bind:[attr] = :[attr]
 - es: v-bind:class, v-bind:style, v-bind:disabled
 - v-on:[event] = @[event]
 - es: v-on:click, v-on:click.prevent, v-on:keyup.enter, ...
- **two-way binding**
 - v-model = v-bind:value + v-on:input
 - v-model.lazy = v-bind:value + v-on:change
- **...**
 - v-html
 - v-[custom-name]

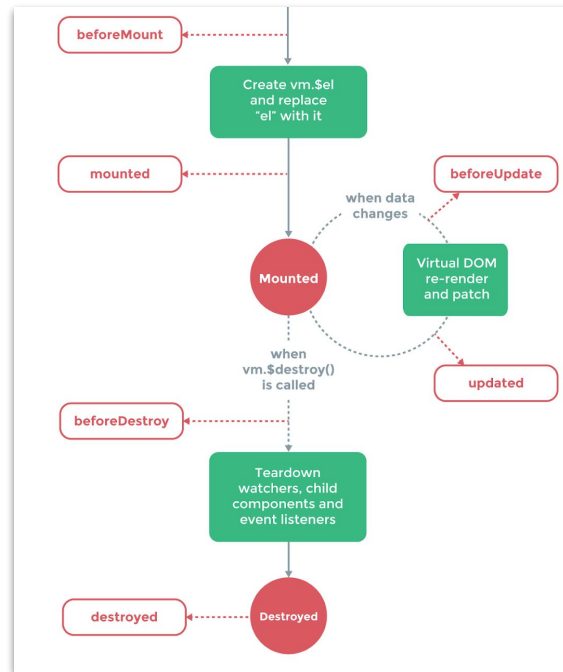
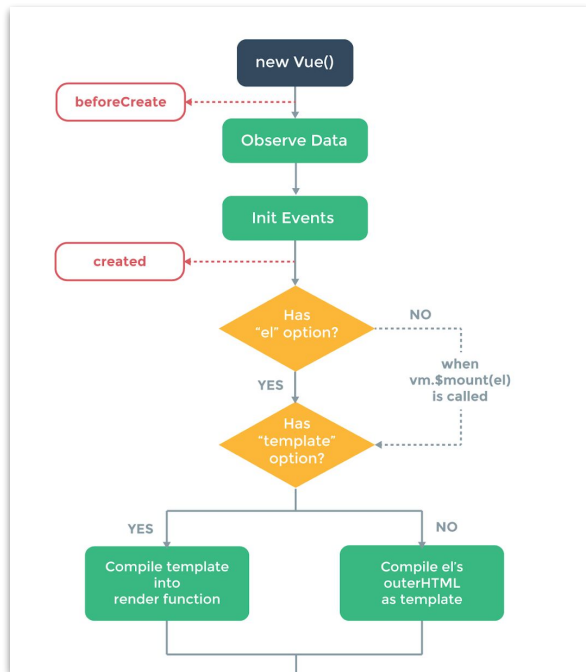


```
<div id="app">
  <form @submit.prevent>
    <input type="text" v-model="text">
    <button type="submit" @click="add">Add</button>
  </form>
  <ul>
    <li v-for="(item, index) in items" :key="index"
      :style="{ 'background-color' : (index % 2 ? 'grey' : 'white')}">
      {{item}}
      <button type="button" @click="remove(index)">X</button>
    </li>
  </ul>
  <div>
    Count: {{count}}
  </div>
</div>
```

```
new Vue({
  el: '#app',
  data: function() {
    return {
      items: [],
      text: ''
    }
  },
  computed: {
    count: function() {
      return this.items.length;
    }
  },
  methods: {
    add: function() {
      this.items.push(this.text)
      this.text = ''
    },
    remove: function(index) {
      this.items.splice(index, 1)
    }
  }
})
```



Ciclo di vita

- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- updated
- beforeDestroy
- destroyed





Componenti




```
var MyComponent = {
  template: `
    <div>
      <button type="button" @click="toggle"> {{btnLabel}} </button>
      <p v-show="show"> {{message}} </p>
    </div>
  `,
  data: function() {
    return {
      show: true,
      message: 'Hello Vue.js'
    }
  },
  computed: {
    btnLabel: function() {
      return this.show ? 'Hide' : 'Show'
    }
  },
  methods: {
    toggle: function() {
      this.show = !this.show
    }
  }
}

Vue.component('my-component', MyComponent)
```



Comunicazione tra componenti

- props
- eventi
- slots



Props + Eventi

```
<form>
  <input type="text" v-model="name" />
  <my-button
    type="submit"
    label="Click Me!"
    @click="onClick">
  </my-button>
</form>
```


```
Vue.component('my-button', {
  props: {
    label: {
      type: String,
      required: true
    },
    type: {
      type: String,
      default: 'button',
      validator: function (value) {
        return ['button', 'submit', 'reset'].indexOf(value) !== -1
      }
    }
  },
  methods: {
    click(event) {
      this.$emit('click', event)
    }
  },
  template: '<button :type="type" @click="click"> {{ label }} </button>'
})
```



Slot

```
<form>
  <input type="text" v-model="name" />
  <my-button
    type="submit"
    @click="onClick">
    Click Me!
  </my-button>
</form>
```

```
Vue.component('my-button', {
  props: {
    type: {
      type: String,
      default: 'button',
      validator: function (value) {
        return ['button', 'submit', 'reset'].indexOf(value) !== -1
      }
    }
  },
  methods: {
    click(event) {
      this.$emit('click', event)
    }
  },
  template: '<button :type="type" @click="click"> <slot /> </button>'
})
```



Single File Component

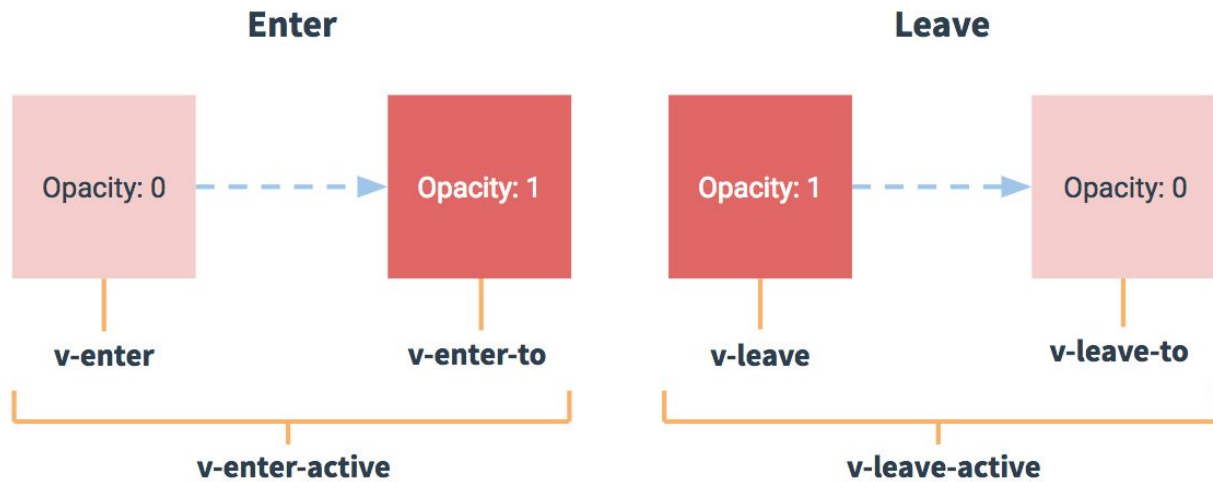
```
<template>
  <button @click="click" :class="classType">
    <slot />
  </button>
</template>

<script>
export default {
  props: {
    type: {
      type: String,
      default: 'button',
      validator: function (value) {
        return ['button', 'submit', 'reset'].indexOf(value) !== -1
      }
    }
  },
  computed: {
    classType() {
      if (this.type === 'submit') {
        return 'submit-btn'
      } else if (this.type === 'reset') {
        return 'reset-btn'
      }
      return 'default-btn'
    }
  },
  methods: {
    click(event) {
      this.$emit('click', event)
    }
  }
}
</script>

<style scoped>
  .default-btn {
    background-color: gray;
  }
  .submit-btn {
    background-color: orange;
  }
  .reset-btn {
    background-color: green;
  }
</style>
```



Animazioni / Transizioni





```
<div id="app">
  <form @submit.prevent>
    <input type="text" v-model="text">
    <button type="submit" @click="add">Add</button>
  </form>
  <ul>
    <transition-group name="fade">
      <li v-for="(item, index) in items" :key="index"
        :style="{ 'background-color' : (index % 2 ? 'grey' : 'white')}">
        {{item}}
        <button type="button" @click="remove(index)">X</button>
      </li>
    </transition-group>
  </ul>
  <button type="button" @click="showCount = !showCount"> Toggle </button>
  <transition name="fade">
    <div v-show="showCount">
      Count: {{count}}
    </div>
  </transition>
</div>
```

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .3s;
}
.fade-enter, .fade-leave-to {
  opacity: 0;
}
```



Ecosistema

- vue-router -> routing lato client
- vuex -> state management (flux/redux)
- vue-cli -> scaffolding tool (... in arrivo la versione 3)
- vue-test-utils -> libreria per unit test abbinabile con Jest, Mocha ecc.
- nuxt.js -> server side rendering (ssr)



Buefy

```
const buefy = vue + bulma
```



<https://buefy.github.io>



Grazie!