Vue.js for beginners

@juliobitencourt

JavaScript everywhere.

METEOR
Rendr
TypeScript
browserify
express
ember
flow
RIOT
Gulp

Which path to choose?

Julio Bitencourt

Web Developer since 2000
@juliobitencourt

Why Vue.js?

You know. One more JavaScript Framework.

Reactive Components for Modern Web Interfaces

- Reactivity
- Components
- Modularity

and more

```
$ npm install vue
$ bower install vue
```

or just CDN

```
<script type="text/javascript" src="http://cdnjs.cloudflare.com/ajax/libs/vue/1.0.12/vue.js"></script>


<div id="app">

</div>


new Vue({
      el: '#app',
});
```

Databinding

forget $('#mydiv p').text('foo')…

Binding

```
<div id="app">
	<h1>{{message}}</h1>
</div>
```

```
new Vue({
	el: '#app',
	data: {
		message: 'Hello World'
	}
});
```

Two-way binding

```
<div id="app">
	<h1>{{message}}</h1>
	<input type="text" v-model="message">
</div>

new Vue({
	el: '#app',
	data: {
		message: 'Hello World'
	}
});
```

Lists

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

new Vue({
	el: '#app',
	data: {
		todos: [
			{ text: 'Learn JavaScript' },
			{ text: 'Learn Vue.js' },
			{ text: 'Build Something Awesome' }
		]
	}
})
```

Interpolations

One-time interpolation

```
<span>This will never change: {{* msg }}</span>
```

Raw HTML

```
<span>{{{ raw_html }}}</span>
```

HTML Attributes

```
<div id="item-{{ id }}"></div>
```

JavaScript Expressions

```
{{ number + 1 }}
{{ ok ? 'YES' : 'NO' }}
{{ message.split('').reverse().join('') }}
```

Directives

v- syntax

Special attributes with the *v-* syntax.

A directive’s job is to reactively apply special behavior to the DOM when the value of its expression changes.

You could pass an expression or a filter

Directives

```
<p v-show="greeting">Hello!</p>

<a v-bind:href="url"></a>

<!-- full syntax -->
<button v-bind:disabled="someDynamicCondition">Button</button>

<!-- shorthand -->
<button :disabled="someDynamicCondition">Button</button>
```

DOM events

```
<!-- full syntax -->
<a v-on:click="doSomething"></a>

<!-- shorthand -->
<a @click="doSomething"></a>


<form action="done.html" v-on:submit.prevent="myMethod">
        <button type="submit">Click here!</button>
</form>


<!-- shorthand -->
<form action="done.html" @submit.prevent="myMethod">
```

```
new Vue({
        el: '#app',
        methods: {
                myMethod: function() {
                        console.log("Hi there!");
                }
        }
});
```

Conditional Rendering

```
<p v-if="votes > 1000">Popular!</p>

<a v-else="doSomething">Please vote</a>


<p v-show="votes > 1000">Popular!</p>

<a v-else="doSomething">Please vote</a>
```

Components

Just don’t repeat yourself

Create reusable components

```
<!-- Your App -->
<div id="app">
	<counter header="Up"></counter>
	<counter header="Down"></counter>
</div>

<!-- The template -->
<template id="counter-template">
	<h1>{{ header }} {{ count }}</h1>
	<button @click="count += 1">Click me!</button>
</template>


Vue.component('counter', {
	template: '#counter-template',
	data: function() {
		return { count: 0 }
	}
});

new Vue({
	el: '#app',
});
```

Computed Properties

Encapsulate complex logic

```
<!-- The template -->
<template id="counter-template">
        <h1>{{ header }} {{ count }} {{ status }}</h1>
        <button @click="count += 1">Click me!</button>
</template>
```

```
Vue.component('counter', {
        template: '#counter-template',
        data: function() {
                return { count: 0 }
        },
      computed: {
          status: function() {
                return this.count > 10 ? 'Good!' : 'Normal';
        }
      }
});

new Vue({
        el: '#app',
});
```

Where to go now?

Beyond the basics

- Routing - vue-router
- Ajax - vue-resource
- Modules handler (browserify, webpack, duo.js)

Check out the docs

Thanks ;)

@juliobitencourt