SHIFTKEY LABS

A Javasctipt framework for Web App Development

Building Single Page Applications in VUE.js

by Dijana Kosmajac

Overview

- Introduction
  - What is VUE.js?
  - How it compares to Angular & React?
- MVVM design
- Components Architecture
  - Built-in directives
  - Event handling
  - Two-way binding
  - Computed properties
  - CSS styling, conditional rendering
- Reusable Components Basics
- Example app: Todo list

What is VUE.js?

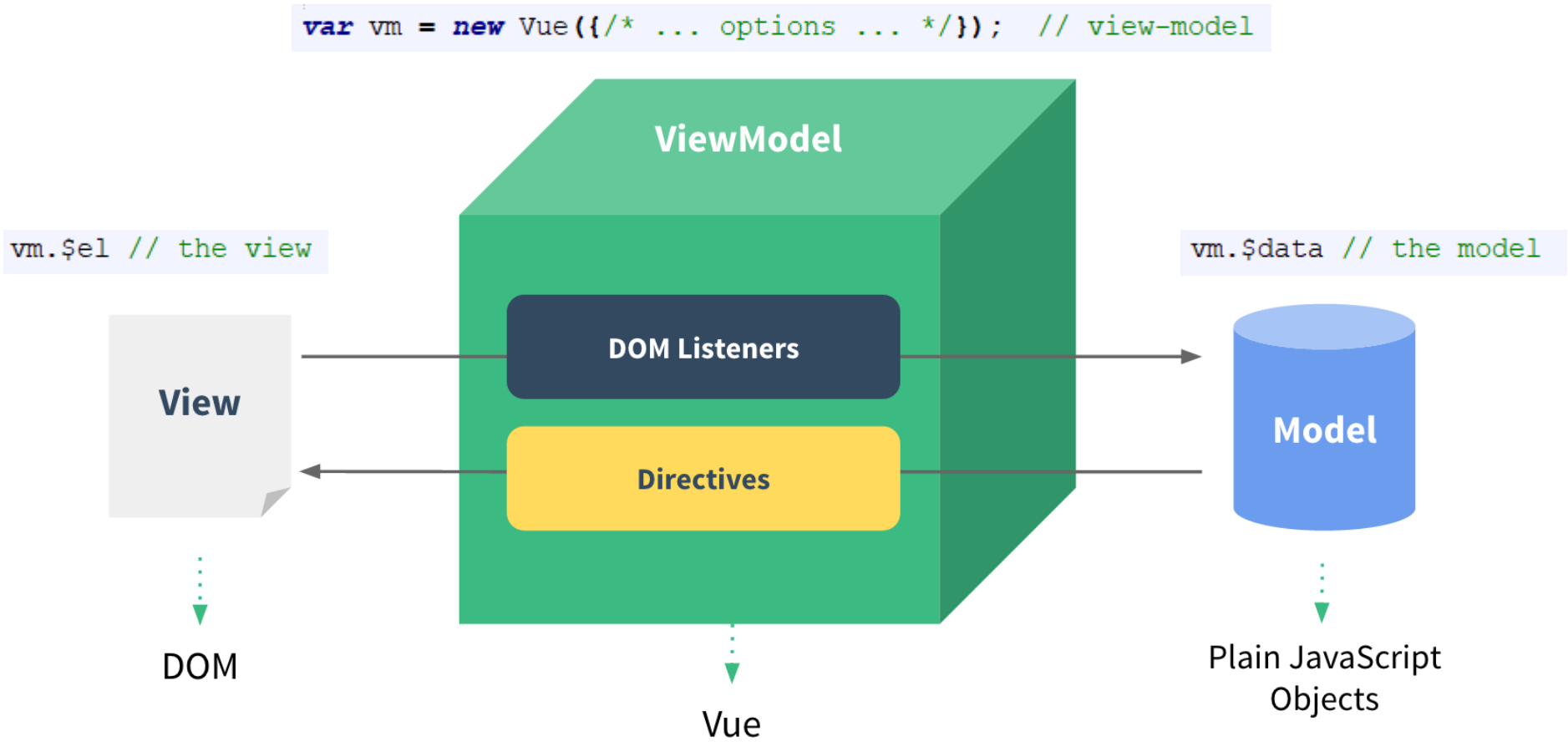
- A progressive framework for building interfaces
- MVVM design pattern with the focus on View-Model, connecting view and model with two-way reactive data binding
- Core values:

componentization | modularity | reactivity | simplicity | stability

- Created by:
  - an ex-engineer of Google, Evan You.
- Inspired by:
  - Angular, React, Knockout, Reactive, Rivet

• 2010 by Google	• 2014 by Evan You	• 2013 by Facebook
• Google, Wix, Weather.com	• Gitlab, Alibaba, 9gag	• Whatsapp, BBC, Twitter
• real DOM – slow	• Virtual DOM – fast	• Virtual DOM – fast
• Backed by Google .	• Backed by open-source community	• Backed by Facebook .
• Size: ~500KB	• Size: ~80KB	• Size: ~100KB
• Heavy-weight apps	• Light-weight apps	• Light-weight apps
• All-in-one	• Bare bones	• Bare bones

MVVM


var vm = new Vue({/* ... options ... */}); // view-model
ViewModel
vm.$el // the view
vm.$data // the model
DOM Listeners
Directives
View
Model
DOM
Vue
Plain JavaScript Objects

Example 1

HTML

```
<script src="https://vuejs.org/js/vue.js"></script>

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

JavaScript + No-Library (pure JS)

```
new Vue({
    el: '#app',
    data: {
      title: "Hi VueJS!"
    }
});
```

Example 2

HTML

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
  <p>
    {{ sayHi() }}
  </p>
</div>

```

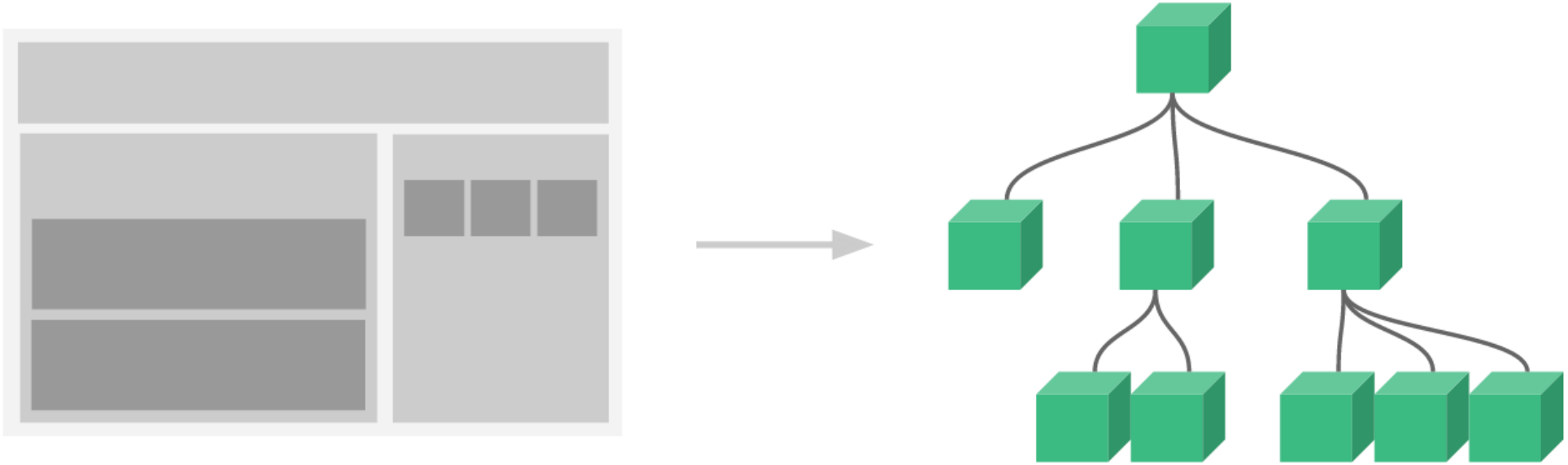
JavaScript + No-Library (pure JS)

```
new Vue({
  el: '#app',
  data: {
    title: "Hi VueJS!"
  },
  methods: {
    sayHi: function() {
      return "Hi All!";
    }
  }
});

```

Components architecture

- Self-contained components form a nested tree-like hierarchy that represents app interface



What is a component?

- A component is a Vue instance with predefined options
- Extends a html element with encapsulated reusable code
  - However: not Web components specification of custom elements
    - No polyfills
    - Have cross-component data flow
    - Custom event communication
- Can be registered globally or locally
- Data model must be a function to avoid sharing the same reference among components

Directives Basics

- A special token in the markup that tells the library to do something to a DOM element
- Similar to Angular, but much simpler
- Vue ships with a few built-in directives

Example 3

HTML

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
  <p>
    {{ title }}
  </p>
  <a v-bind:href="link">Dal CS</a>
</div>

```

JavaScript + No-Library (pure JS)

```
new Vue({
  el: '#app',
  data: {
    title: "Hi VueJS!",
    link: "https://www.dal.ca/faculty/computerscience.html"
  }
});

```

Event handling

- use the `v-on` directive to listen to DOM events and run some JavaScript when they’re triggered
- It is one of the built-in directives

Example 4

HTML

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
  <input type="text" v-on:input="changeTitle('ShiftKey Labs',$event)">
  <p>{{ title }}</p>
  <hr>
  <p class="yellow-square" v-on:mousemove="coordinates">
    {{x}}: {{y}}
  </p>
</div>
```

JavaScript + No-Library (pure JS)

```
new Vue({
    el: "#app",
    data: {
      title: "Hello VueJS!",
      x: 0,
      y: 0
    },
    methods: {
      changeTitle: function(name, event) {
        this.title = event.target.value + " " + name + "!";
      },
      coordinates: function(event) {
        this.x = event.clientX;
        this.y = event.clientY;
      }
    },
  });
```

Two-way binding

- use the `v-model` directive to create two-way data bindings on form input elements
- automatically picks the correct way to update the element based on the input type

Example 5

HTML

```
<script src="https://vuejs.org/js/vue.js"></script>

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

JavaScript + No-Library (pure JS)

```
new Vue({
  el: "#app",
  data: {
    title: "Hi Vuejs"
  }
})
```

Computed properties

- used to declaratively describe a value that depends on other values
- use the `computed` property:

```
new Vue({
    el: …,
    data: …,
    computed: …,
    methods: …,
});
```

- alternative `watch` property, but `computed` is recommended

Example 6

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<button v-on:click="counter++">Increase</button>
<button v-on:click="counter--">Decrease</button>
<button v-on:click="counter2++">Increase second</button>
  <p>C1 {{ counter }} | {{counter2}}</p>
  <p>Method {{ result() }} - Computed: {{ output }}</p>
</div>
```

```
new Vue({
  el: "#app",
  data: {
    counter: 0,
    counter2: 0
  },
  computed: {
    output: function() {
      console.log("computed");
      return this.counter > 3 ? ">3":"<3";
    }
  },
  methods: {
    result: function() {
      console.log("method");
      return this.counter > 3 ? ">3":"<3";
    }
  }
})
```

CSS styling – `v-bind:class` and `v-bind:style`

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<div class="democlass" v-on:click="setRed = !setRed" v-bind:class="{red: setRed}"></div>
<hr>
<div class="democlass" @click="setRed = !setRed" :class="{red: setRed}"></div>
<hr>
<div class="democlass" v-bind:style="{ 'background-color': 'green'}"></div>
</div>
```

```
new Vue({
   el: "#app",
   data: {
     setRed: false
   }
 })
```

```
.democlass {
   width: 100px;
   height: 100px;
   background-color: blue;
}

.red {
   background-color: red;
}

.green {
   background-color: green;
}
```

Conditional rendering – `v-if:` and `v-else`

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<h1 v-if="awesome" v-on:click="awesome = !awesome">Vue.js is awesome!</h1>
<h1 v-else  v-on:click="awesome = !awesome">React.js is better!</h1>
</div>
```

```
new Vue({
  el: "#app",
  data: {
    awesome: false
  }
})
```

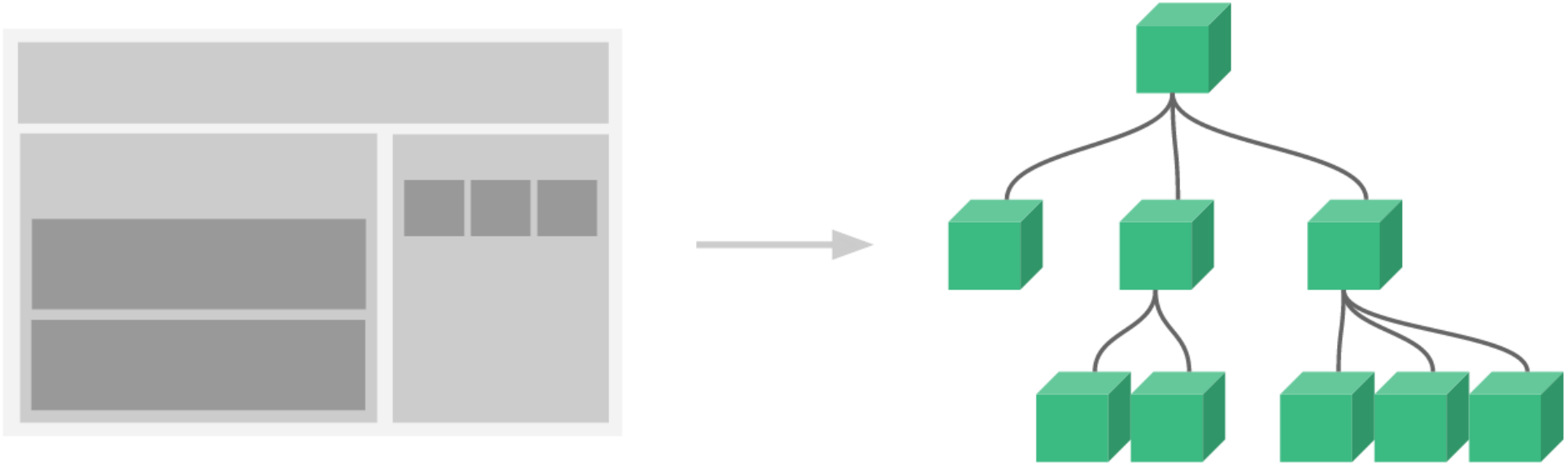
```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<template  v-if="awesome">
  <h1  v-on:click="awesome = !awesome">Vue.js is awesome!</h1>
  <p>
  Woohoo!
  </p>
</template>

<h1 v-else  v-on:click="awesome = !awesome">React.js is better!</h1>
</div>
```

Components architecture

- Self-contained components form a nested tree-like hierarchy that represents app interface



Component Basics

- reusable Vue instances with a name, example `<custom-button>`

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<custom-button></custom-button>
</div>
```

```
// Define a new component called custom-button
Vue.component('custom-button', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'
});
new Vue({
  el: "#app"
})
```

Passing properties to child components

- reusable Vue instances with a name, example `<custom-button>`

```
<script src="https://vuejs.org/js/vue.js"></script>

<div id="app">
<custom-button label="button"></custom-button>
</div>
```

```
// Define a new component called custom-button
Vue.component('custom-button', {
  props: ["label"],
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked {{ label }} {{ count }} tim
es.</button>'
});
new Vue({
  el: "#app"
})
```

Example: small todo list

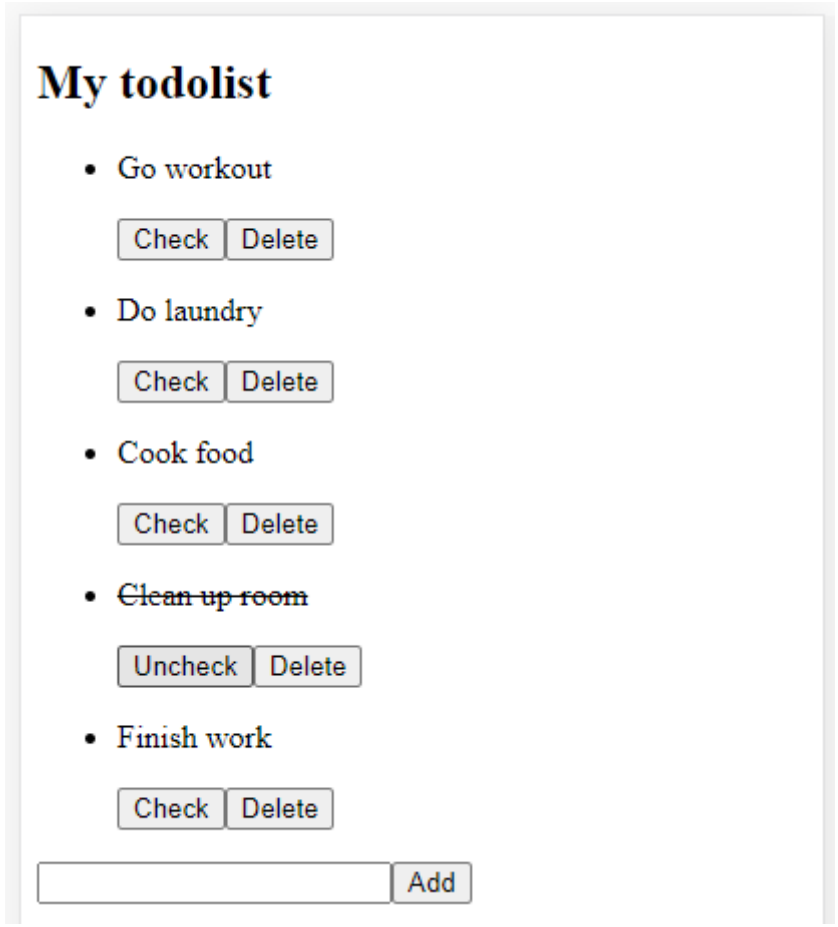
- Prerequisites:
  - Install node.js (platform-specific)
  - Install npm.js (comes with node.js)
  - Install vue.js (`npm install vue`)
  - Install vue-cli (`npm install -g @vue/cli`)

Example: small todo list

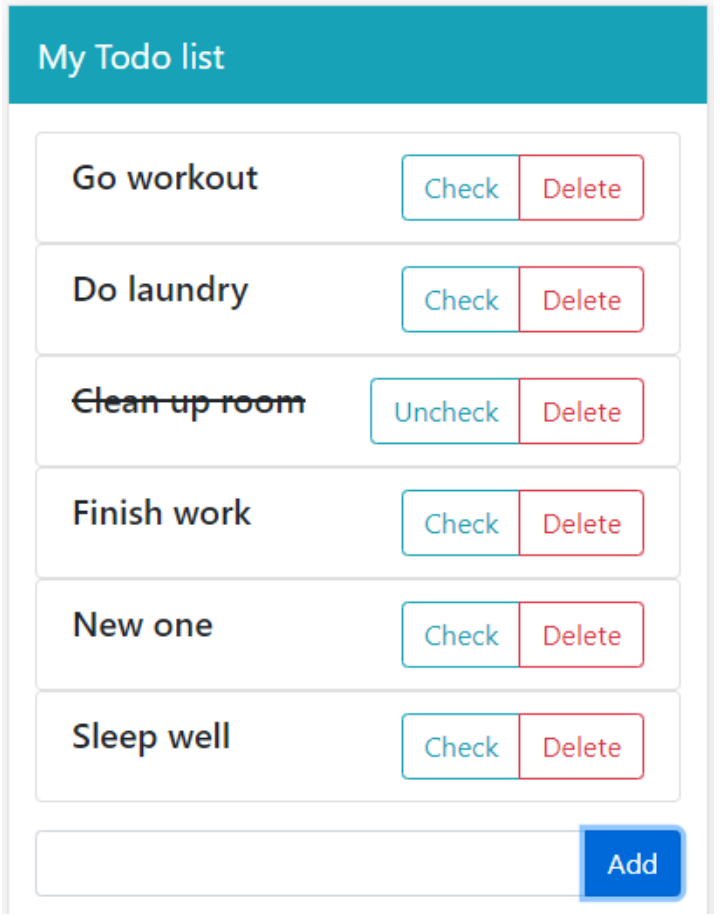
- In your shell create a new project with:
  ```
  vue create sk_todo
  ```
- Change working directory to sk_todo:
  ```
  cd sk_todo
  ```
- Install uuid (needed for autogeneration of object ids):
  ```
  npm install vue-uuid
  ```
- To try out the Hello World app, run:
  ```
  npm run serve
  ```

And that is it for now!

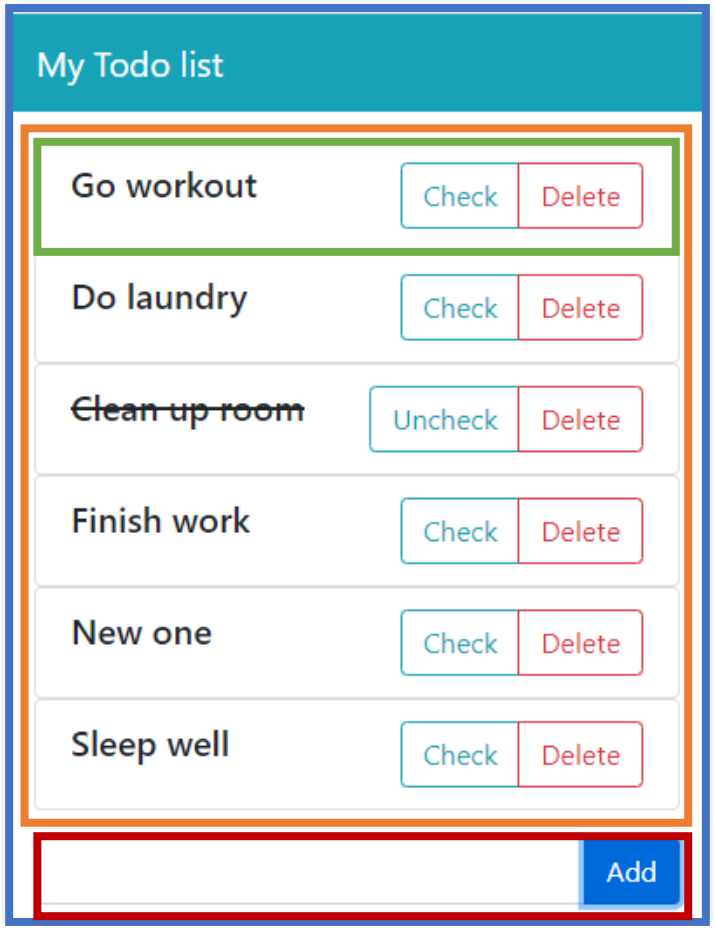
Example: small todo list



Bootstrap



Example: small todo list



App.js

Todos.js

Todo.js

AddTodo.js

```
\---src
    |   App.vue
    |   main.js
    |
    +---assets
    |       logo.png
    |
    \---components
            AddTodo.vue
            Todo.vue
            Todos.vue
```

Download the code

- https://github.com/dijana-sagit/sk_notes.git

Conclusions

- We only scratched the surface!
- Easier to learn if you are familiar with Angular, React or similar
- However, it has active dev community, and it is easy to find answers
- Things we didn’t talk about:
  - Component registration: local and global
  - Vue CLI
  - Real-world project organization
  - Routing
  - And much more!

References & useful resources

- Vue Guide: https://vuejs.org/v2/guide/
- Udemy courses (by Maximilian Schwarzmuller)
- https://www.tutorialspoint.com/vuejs/index.htm
- https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847

Thank you!