

VUE.JS

*dr Michał Kuciapski*  
Higher Banking School

# AGENDA

1. Front-end CSS frameworks trend
2. How to choose the right framework?
3. Front-end JS frameworks

## Vue.js

1. What is Vue.js?
2. Comparison with other libraries/frameworks
3. Installing Vue.js
4. Dev Tools
5. Rendering
6. Lifecycle diagram

### Referenced materials (selected):

1. <https://vuejs.org/v2/guide>
2. Andrea Passaglia, Vue.js 2 Cookbook, Packt Publishing 2017
3. Murat Dogan, Vue.js – Getting Started
4. Julio Bitencourt, Vue.js for beginners

# AGENDA

## Vue.js

7. Application instance
8. Binding
9. Interpolations
10. Directives
11. Arguments
12. Modifiers
13. Shorthands
14. Computed properties
15. Binding HTML Classes
16. Binding Inline Styles
17. List Rendering
18. v-for
19. Event Handling
20. Components
21. SFC Concept
22. Router
23. Vuex

# FRONT-END CSS FRAMEWORKS TREND

● Bootstrap  
Search term

● Foundation  
Search term

● Material UI  
Search term

● Semantic-UI  
Search term

● Materialize  
Search term

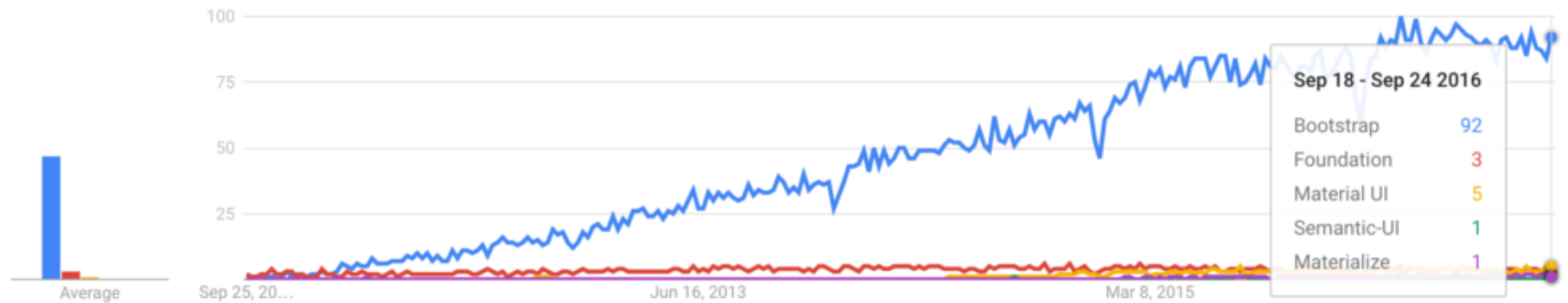
Worldwide ▼

Past 5 years ▼

Web Design & Development ▼

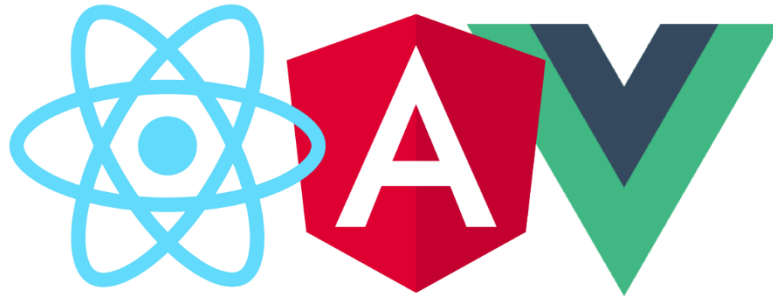
Web Search ▼

Interest over time ?



# HOW TO CHOOSE THE RIGHT FRAMEWORK?

1. How mature are the frameworks / libraries?
2. Are the frameworks likely to be around for a while?
3. How extensive and helpful are their corresponding communities?
4. How easy is it to find developers for each of the frameworks?
5. What are the basic programming concepts of the frameworks?
6. How easy is it to use the frameworks for small or large applications?
7. What does the learning curve look like for each framework?
8. What kind of performance can you expect from the frameworks?
9. Where can you have a closer look under the hood?
10. How can you start developing with the chosen framework?



# FRONT-END JS FRAMEWORKS

## Frameworks:

- AngularJS - <https://angular.io>
- Angular2/4 - <https://angularjs.org>
- ReactJS - <https://facebook.github.io/react>
- Vue.js - <https://vuejs.org>
- Ember.js - <https://www.emberjs.com>
- Meteor.js - <https://www.meteor.com>

## Popularity:

Parametr/Framework	AngularJS	ReactJS	Vue.js
Stars (tousands)	56,1	69,2	57,1
Commits (tousands)	8,5	8,7	2
Contributors (tousands)	1,6	1,0	0,1

# FRONT-END JS FRAMEWORKS

## Efficiency

Parameter/Frameworks	angular v1.6.3-keyed	react v15.5.4-keyed	vue v2.3.3-keyed
<b>create rows</b> Duration for creating 1000 rows after the page loaded.	251.848.00 (1.82)	188.9310.93 (1.36)	166.688.63 (1.20)
<b>replace all rows</b> Duration for updating all 1000 rows of the table (with 5 warmup iterations).	278.2616.67 (1.88)	201.036.40 (1.36)	168.494.98 (1.14)
<b>partial update</b> Time to update the text of every 10th row (with 5 warmup iterations).	12.551.95 (1.00)	16.482.30 (1.03)	17.332.90 (1.08)
<b>select row</b> Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	8.083.62 (1.00)	8.763.37 (1.00)	9.311.66 (1.00)

# FRONT-END JS FRAMEWORKS

## Efficiency

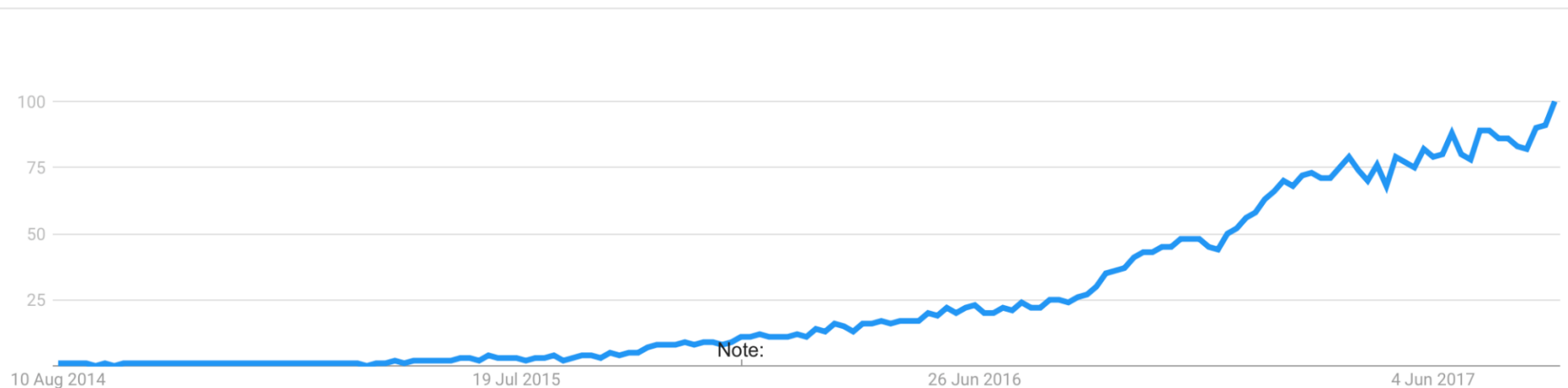
Parameter/Frameworks	angular	react v15.5.4-keyed	vue v2.3.3-keyed
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	14.681.48 (1.00)	14.660.90 (1.00)	18.291.52 (1.14)
remove row Duration to remove a row. (with 5 warmup iterations).	47.382.43 (1.12)	47.223.20 (1.12)	52.632.69 (1.24)
create many rows Duration to create 10,000 rows	3108.702162.20 (2.34)	1852.3629.03 (1.39)	1587.5233.89 (1.19)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	454.7942.09 (1.58)	345.6210.40 (1.20)	399.4610.98 (1.39)
clear rows Duration to clear the table filled with 10.000 rows.	817.5937.16 (4.68)	398.388.25 (2.28)	254.515.03 (1.46)
startup time Time for loading, parsing and starting up	118.125.14 (2.91)	69.982.85 (1.73)	56.552.48 (1.39)
slowdown geometric mean	<b>1.69</b>	<b>1.30</b>	<b>1.22</b>



# VUE - WHAT IS VUE.JS?

1. Vue (pronounced /vju:/, like view) is a progressive framework for building user interfaces.<sup>1</sup>
2. It was first released in February 2014 by ex-Google-employee Evan You
3. Only thing you need to know is HTML, CSS and JavaScript
4. It's been popular since its last version 2.0
5. Vue is used by Alibaba, Baidu, Expedia, Nintendo, GitLab — a list of smaller projects can be found on [madewithvuejs.com](https://madewithvuejs.com).

Interest over time 



# VUE - WHAT IS VUE.JS?



## Simple

Write some HTML, grab some JSON, create a Vue instance, that's it.



## Reactive

Expressions & computed properties with transparent dependency tracking.



## Components

Compose your application with decoupled, reusable components.



## Compact

~24kb min+gzip, no dependency.



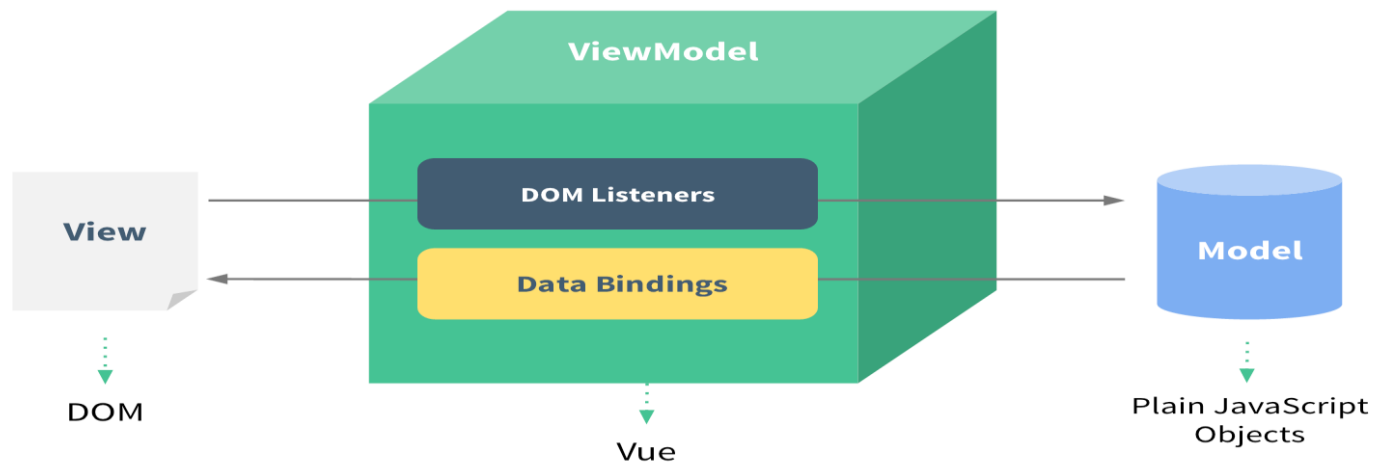
## Fast

Precise and efficient async batch DOM updates.

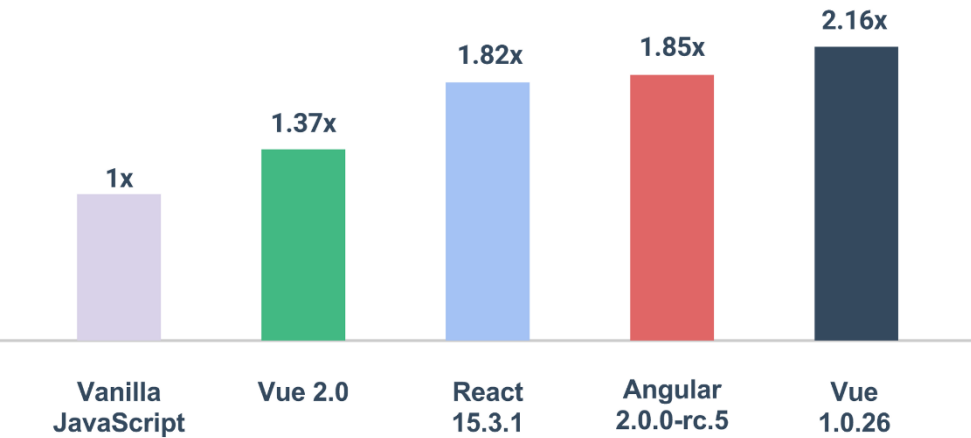


## Package Ready

Install via NPM or Bower - leverage your favorite eco system!



# VUE - COMPARISON WITH OTHER LIBRARIES/FRAMEWORKS



Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
ready memory Memory usage after page load.	4.8 ± 0.0 (1.3)	4.9 ± 0.1 (1.3)	3.8 ± 0.0 (1.0)
run memory Memory usage after adding 1000 rows.	10.9 ± 0.1 (1.4)	10.8 ± 0.1 (1.4)	7.5 ± 0.1 (1.0)

[3rd party benchmark](#)

facebook / react

Watch

4,944

★ Star

75,379

🔗 Fork

14,225

angular / angular

Watch

2,621

★ Star

27,647

🔗 Fork

6,922

vuejs / vue

Watch

3,718

★ Star

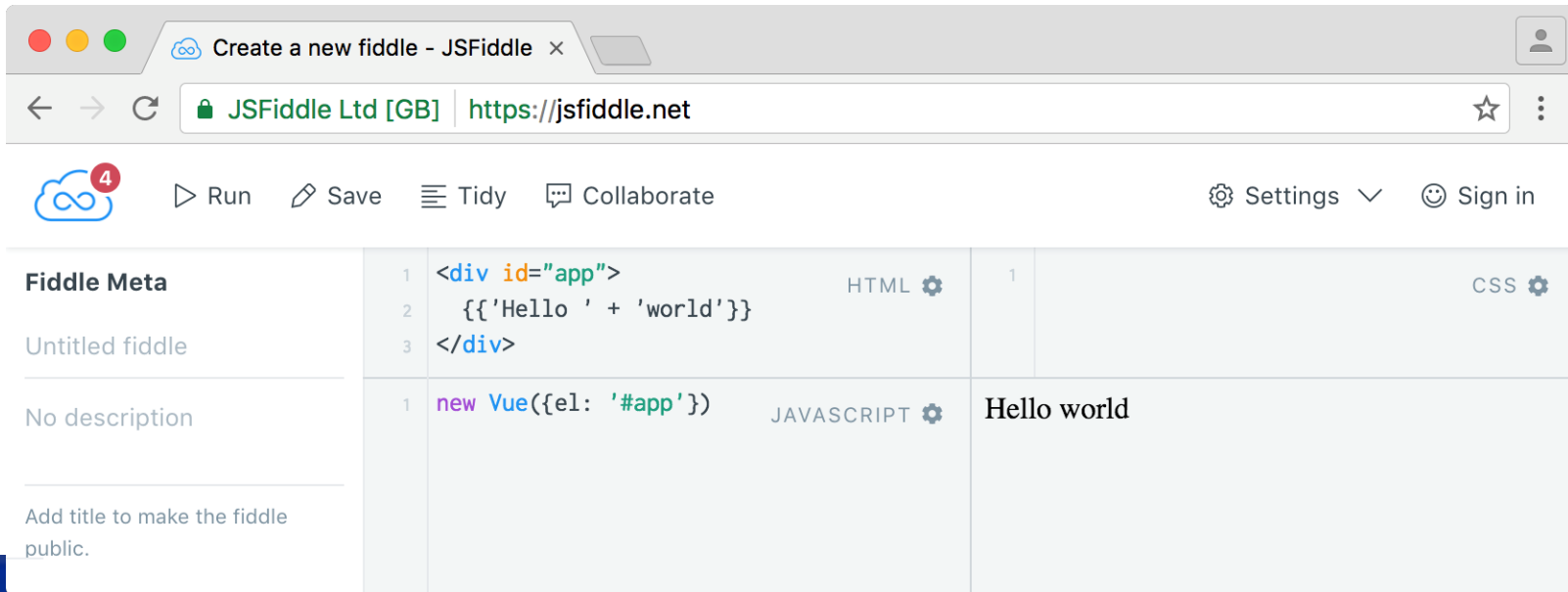
66,272

🔗 Fork

9,535

# VUE - INSTALLING VUE.JS

- You can install core library via CDN (Content Delivery Network)
- If you want to always follow up the latest version of Vue, use unpkg:
  - <https://unpkg.com/vue> - Always redirect to the latest version of Vue!
  - [https://unpkg.com/vue@\[version\]/dist/vue.min.js](https://unpkg.com/vue@[version]/dist/vue.min.js) - to specific version.
- You can install core library using Bower or npm
- Open <https://jsfiddle.net> , choose the additional JS Framework; Vue.js



The screenshot shows a web browser window with the address bar displaying "https://jsfiddle.net". The page title is "Create a new fiddle - JSFiddle". The main content area is divided into three panels: "Fiddle Meta", "HTML", and "CSS". The "Fiddle Meta" panel on the left shows "Untitled fiddle" and "No description". The "HTML" panel in the center contains the following code:

```
1 <div id="app">
2   {{ 'Hello ' + 'world' }}
3 </div>
```

The "CSS" panel on the right is empty. Below the HTML panel, the "JAVASCRIPT" panel contains the following code:

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

The output of the fiddle is displayed in the bottom right corner, showing "Hello world".

# ZADANIE – REALIZACJA WSPÓLNA

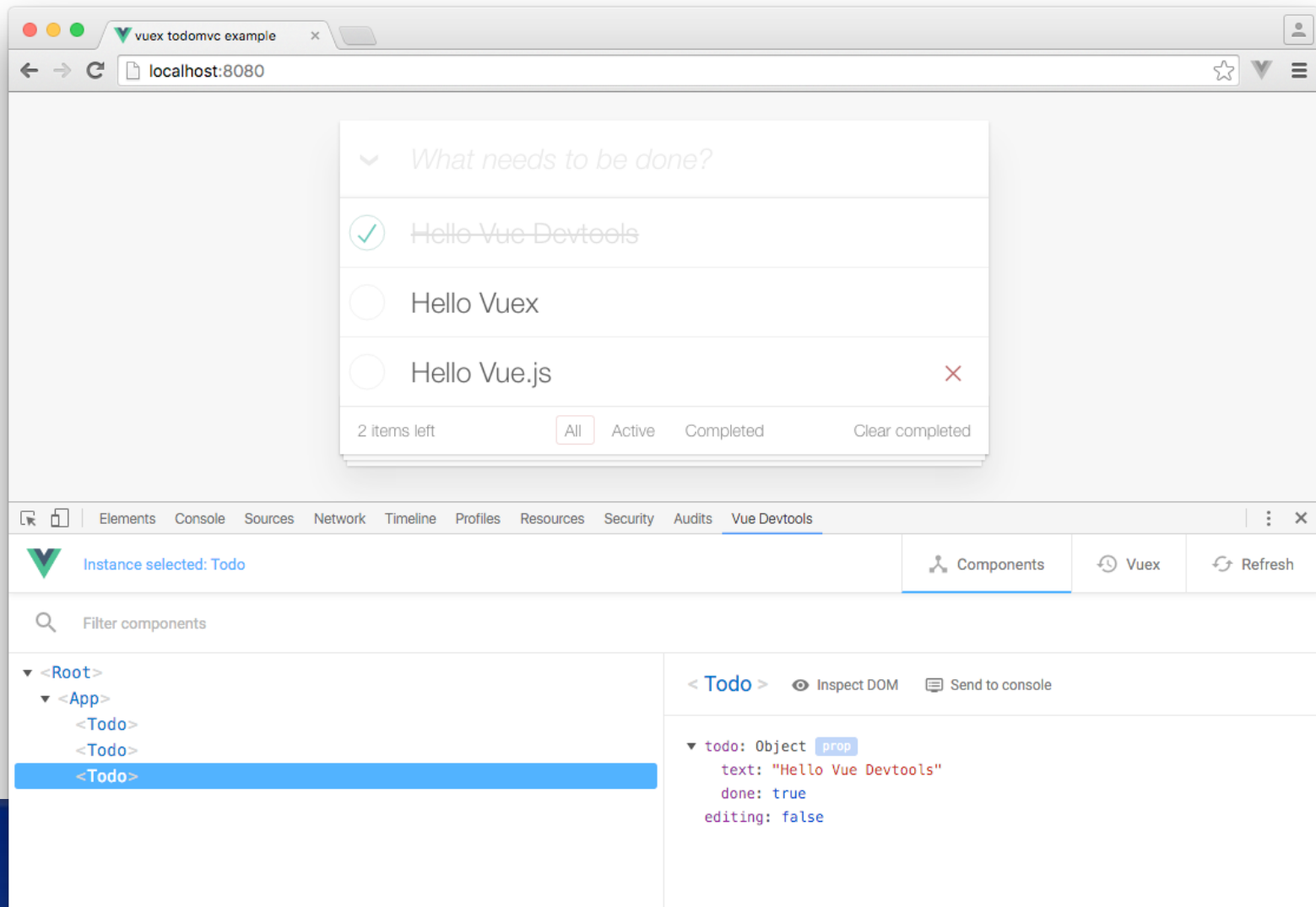
1. **Otwórz** witrynę startową projektu i **zaimportuj** framework Vue.js (

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.5.13/dist/vue.js">  
</script> <script src="https://cdn.jsdelivr.net/npm/vue"> </script>  
)
```

Projekt służący zarządzaniu wpisami odnośnie posiadanych kompetencji, np. Ionic, Vue.js, wraz z możliwościami społecznego dzielenia się informacjami.

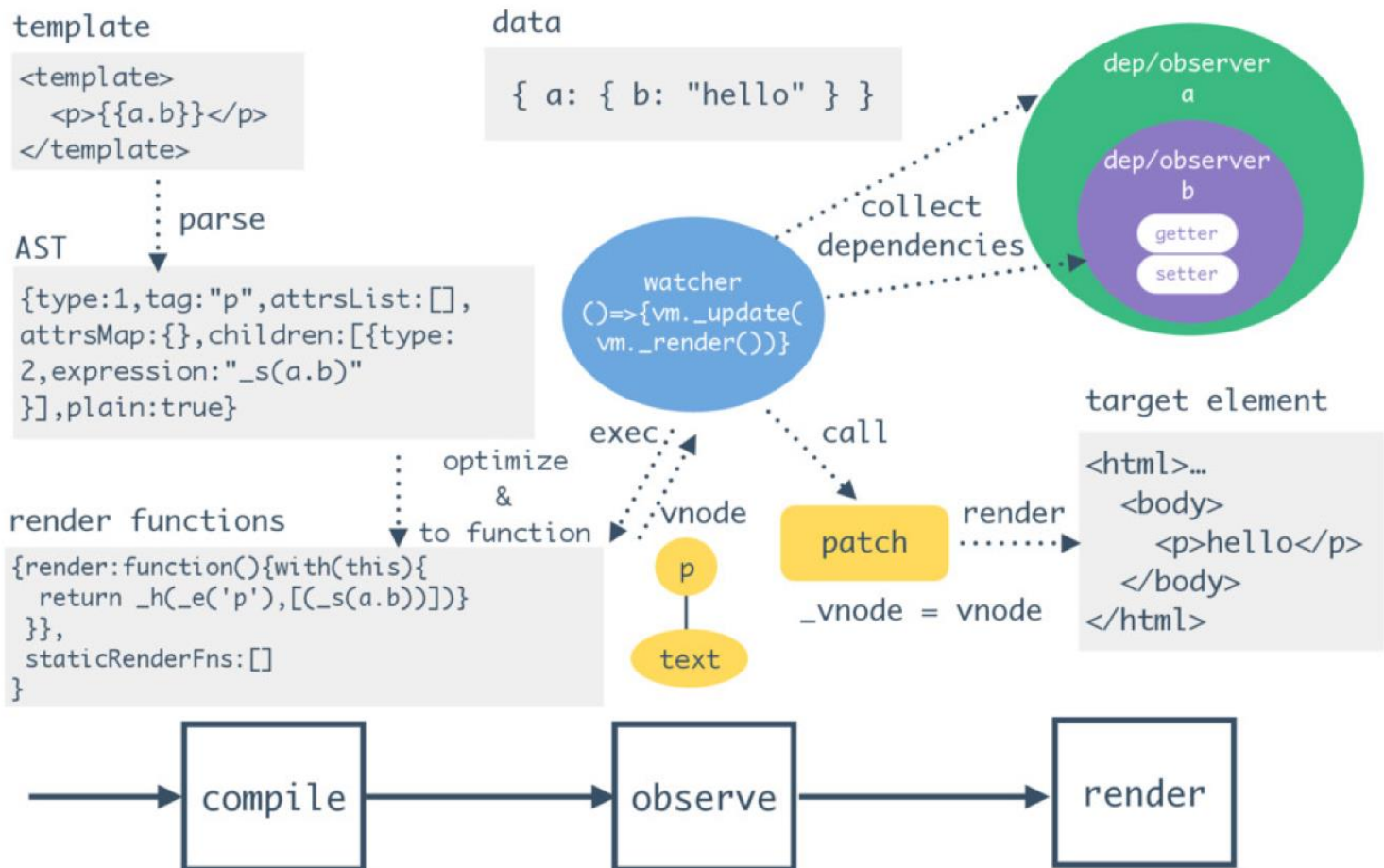
# VUE - DEV TOOLS

- <https://github.com/vuejs/vue-devtools>

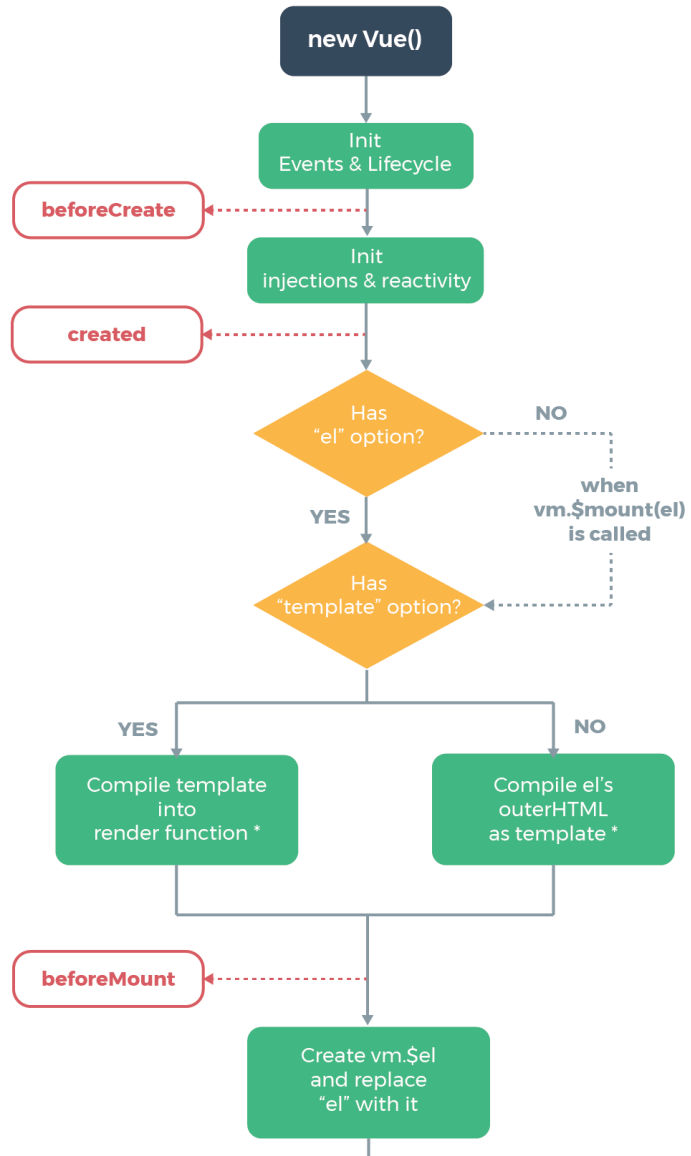


# VUE - RENDERING

## 2.0 rendering flow



# VUE – LIFECYCLE DIAGRAM



```
new Vue({
  data: {
    a: 1
  },
  created: function () {
    // `this` points to the vm instance
    console.log('a is: ' + this.a)
  }
})
// => "a is: 1"
```



# VUE — APPLICATION INSTANCE

- **new Vue({el:'#app'})** will instantiate a new Vue instance.
- Everything that we will write inside the **<div> with the ID as app** will be under the scope of Vue.
- placing the **#app on the body or html tag will throw an error.**

```
<!DOCTYPE html>
<html>
  <body>
    <div id="app"></div>
  </body>
</html>
```

- App is **referenced** in JavaScript with **el** property.

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Vue.JS</title>

  <style></style>
</head>

<body onload="init()">

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

  <script src="https://cdn.jsdelivr.net/vue/latest/vue.js">
  <script type="text/javascript">
function init() {
  new Vue({
    el: '#app',
    data: {
      message: 'Hello Vue.js!'
    }
  });
}
</script>

</body></html>
```

## ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html utwórz **instancję aplikacji Vue**.
2. Jako dane wstaw **tablicę** – plik: /js/competences.js.

# VUE - BINDING

```
<div id="app">
<h1>{{message}}</h1>
</div>
new Vue({
  el: '#app',
  data: {
    message: 'Hello World'
  }
});
```

One-way

```
<div id="app">
<h1>{{message}}</h1>
<input type="text" v-model="message">
</div>
new Vue({
  el: '#app',
  data: {
    message: 'Hello World'
  }
});
```

Two-way

```
<span v-once>This will never change: {{ msg }}</span>
```

# VUE - BINDING

## Attributes

```
<div v-bind:id="dynamicId"></div>
```

```
<button v-bind:disabled="isButtonDisabled">Button</button>
```

# VUE - INTERPOLATIONS

- Simple
- Expressions

```
<span>Message: {{ msg }}</span>
```

```
{{ number + 1 }}
```

```
{{ ok ? 'YES' : 'NO' }}
```

```
{{ message.split('').reverse().join('') }}
```

```
<div v-bind:id="'list-' + id"></div>
```

# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html przygotuj kafelkę (ang. tile) w oparciu o **Bootstrap Cards** (<https://getbootstrap.com/docs/4.0/components/card/>) wyświetlającą kolejno dane jednej z kompetencji, której poziom (skala od 0 – 5, gdzie 0 to minimum) określiłbyś na najwyższy z wszystkich:
  - Nazwę kompetencji
  - Rysunek (podpowiedź: expressions v-bind:src=„...“)
  - Poziom poznania
  - Opis



# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie `competences.html` dla przygotowanej kafelki (ang. tile) wyświetlającej kolejno dane jednej z kompetencji zdefiniuj, aby tytuł kompetencji był z dużych liter – zastosuj **Expressions**.

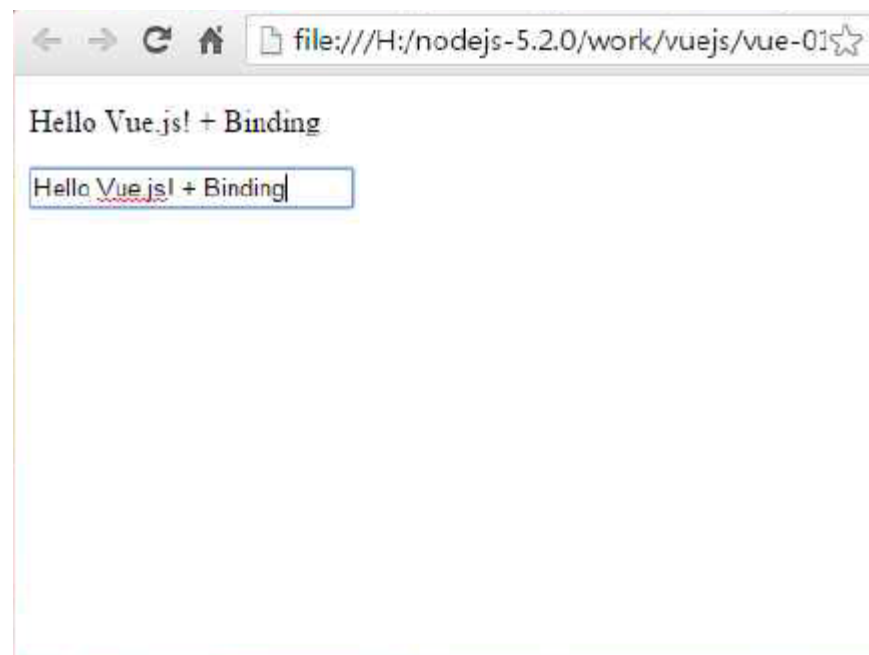


```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Vue.JS</title>
  <style></style>
</head>

<body onload="init()">

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

  <script src="https://cdn.jsdelivr.net/vue/latest/vue.js">
  <script type="text/javascript">
function init() {
  new Vue({
    el: '#app',
    data: {
      message: 'Hello Vue.js!'
    }
  });
}
</script>
</body></html>
```



# VUE - BINDING HTML CLASSES

## Object Syntax

```
<div v-bind:class="{ active: isActive }"></div>
```

Active class will be included if the data property **isActive** is **true**.

## Array Syntax

```
<div v-bind:class="[activeClass, errorClass]"></div>
```

```
data: {  
  activeClass: 'active',  
  errorClass: 'text-danger'  
}
```

Classes included accordingly to **data's** properties.

```
<div v-bind:class="[isActive ? activeClass : '', errorClass]"></div>
```

# VUE - BINDING INLINE STYLES

- When you use a CSS property that requires vendor prefixes in v-bind:style, will **automatically detect and add appropriate prefixes**.
- It looks almost like CSS, except it's a JavaScript object. You can use either **camelCase** or **kebab-case** (use quotes with kebab-case) for the CSS

```
<div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
```

```
data: {  
  activeColor: 'red',  
  fontSize: 30  
}
```

- It is often a good idea to **bind to a style object directly** so that the template is cleaner.

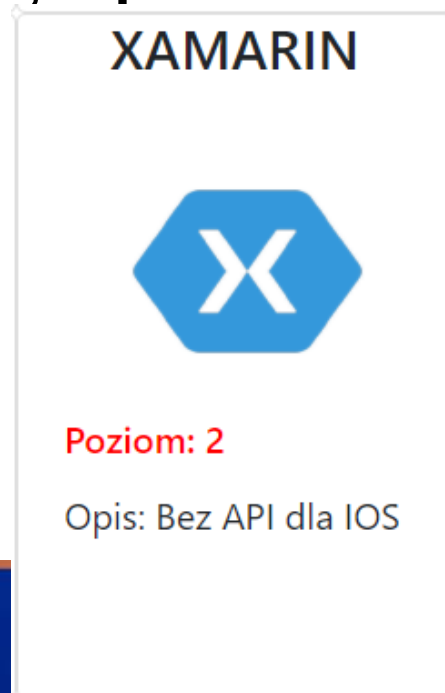
```
<div v-bind:style="styleObject"></div>
```

```
data: {  
  styleObject: {  
    color: 'red',  
    fontSize: '13px'  
  }  
}
```

```
v-bind:style="{color: ocena < 3 ? 'red' : 'black'}">
```

# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html dla przygotowanej kafelki (ang. tile) wyświetlającej kolejno dane jednej z kompetencji zdefiniuj, aby poziom kompetencji był kolorem **czerwonym**, jeśli ma wartość mniejszą niż 2 - zastosuj **Expressions** dla **atrybutu style**.
2. Na stronie competences.html dla przygotowanej kafelki (ang. tile) wyświetlającej kolejno dane jednej z kompetencji zdefiniuj, aby poziom kompetencji był kolorem **zielonym**, jeśli ma wartość większą niż 3 - zastosuj **Expressions** dla **atrybutu class**.



## VUE - ARGUMENTS

- “argument”, denoted by a **colon after the directive name**.

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

*href attribute* to the value of the expression *url*.

# VUE - EVENT HANDLING

- **v-on directive to listen to DOM events** and run some JavaScript when they're triggered.

```
<div id="example-1">
  <button v-on:click="counter += 1">Add 1</button>
  <p>The button above has been clicked {{ counter }} times.</p>
</div>
```

```
var example1 = new Vue({
  el: '#example-1',
  data: {
    counter: 0
  }
})
```

- v-on can also **accept the name of a method** you'd like to call.

```
<div id="example-2">
  <!-- `greet` is the name of a method defined below -->
  <button v-on:click="greet">Greet</button>
</div>
```

```
var example2 = new Vue({
  el: '#example-2',
  data: {
    name: 'Vue.js'
  },
  // define methods under the `methods` object
  methods: {
    greet: function (event) {
      // `this` inside methods points to the Vue instance
      alert('Hello ' + this.name + '!')
      // `event` is the native DOM event
      if (event) {
        alert(event.target.tagName)
      }
    }
  }
})
```

```
// you can invoke methods in JavaScript too
example2.greet() // => 'Hello Vue.js!'
```

```

<div id="app">
  <md-toolbar> <h1 class="md-title">Learning Vue.JS</h1> </md-toolbar>

  <div class="main-content">
    <md-input-container>
      <label>Enter Todo</label>
      <md-input v-model="newTodo"></md-input>
    </md-input-container>
    <md-button class="md-raised md-primary" v-on:click="addTodo">Add Todo</md-button>

    <ul>
      <li v-for="(todo, index) in todos">
        <md-button class="md-icon-button md-warn" v-on:click="removeTodo(index)"><md-icon>remov
        {{ todo.text }}
      </li>
    </ul>

  </div></div>

```

# VUE - SHORTHANDS

- The v- prefix serves as a visual cue for identifying Vue-specific attributes in your templates - **verbose for some frequently used directives.**

## # v-bind Shorthand

```
<!-- full syntax -->  
<a v-bind:href="url"></a>  
  
<!-- shorthand -->  
<a :href="url"></a>
```

## # v-on Shorthand

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



# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html dodaj interfejs dla wyszukiwania kompetencji oparte o pole tekstowe. Zdefiniuj zdarzenie wprowadzania znaku w polu tekstowym powodujące wywołanie metody **searchCompetence**, która (rozwiązanie tymczasowe) powoduje wypisanie w oknie (alert) **podanego tekstu wyszukiwania**. Ponadto zdefiniuj binding (v-model) powodujący, że po **wpisaniu tekstu** w pole wyszukiwania pojawia się on w **etykiecie** (rozwiązanie tymczasowe).

Podaj nazwę kompetencji

...

(<https://getbootstrap.com/docs/4.0/components/input-group>)

# VUE - MODIFIERS

- Modifiers are **special postfixes denoted by a dot**, which indicate that a **directive should be bound in some special way**.

```
<form v-on:submit.prevent="onSubmit"></form>
```

The .prevent modifier tells the v-on directive to call event.preventDefault() on the triggered event:

# VUE - EVENT MODIFIERS

- It is a very common need to call `event.preventDefault()` or `event.stopPropagation()` **inside event handlers**.
- It would be better if the **methods can be purely about data logic** rather than having to deal with DOM event details.
- Recall that **modifiers are directive postfixes** denoted by a dot:

- `.stop`
- `.prevent`
- `.capture`
- `.self`
- `.once`

```
<!-- the click event's propagation will be stopped -->
```

```
<a v-on:click.stop="doThis"></a>
```

```
<!-- the submit event will no longer reload the page -->
```

```
<form v-on:submit.prevent="onSubmit"></form>
```

```
<!-- modifiers can be chained -->
```

```
<a v-on:click.stop.prevent="doThat"></a>
```

```
<!-- just the modifier -->
```

```
<form v-on:submit.prevent></form>
```

```
<!-- use capture mode when adding the event listener -->
```

```
<!-- i.e. an event targeting an inner element is handled here before being handled by the outer element -->
```

```
<div v-on:click.capture="doThis">...</div>
```

```
<!-- only trigger handler if event.target is the element itself -->
```

```
<!-- i.e. not from a child element -->
```

```
<div v-on:click.self="doThat">...</div>
```

# VUE - KEY MODIFIERS

- Vue also allows adding **key modifiers** for v-on when listening for **key events**.

- **key modifier aliases:**

- .enter
- .tab
- .delete (captures both “Delete” and “Backspace” keys)
- .esc
- .space
- .up
- .down
- .left
- .right

```
<!-- only call vm.submit() when the keyCode is 13 -->  
<input v-on:keyup.13="submit">
```

```
<!-- same as above -->  
<input v-on:keyup.enter="submit">
```

```
<!-- also works for shorthand -->  
<input @keyup.enter="submit">
```

- You can also define **custom key modifier aliases** via the global config.keyCodes

```
// enable v-on:keyup.f1  
Vue.config.keyCodes.f1 = 112
```

```
<input v-on:keyup.f1="submit">
```

# VUE - KEY MODIFIERS - MODIFIER KEYS

- It is possible to use the following modifiers to trigger mouse or keyboard event listeners only when the corresponding **modifier key** is pressed:
  - .ctrl
  - .alt
  - .shift
  - .meta - Macintosh keyboards (⌘), on Windows keyboard (⌞)

```
<!-- Alt + C -->  
<input @keyup.alt.67="clear">  
  
<!-- Ctrl + Click -->  
<div @click.ctrl="doSomething">Do something</div>
```

# VUE - MOUSE BUTTON MODIFIERS

- These modifiers **restrict the handler to events triggered by a specific mouse button.**
- New in **2.2.0+**:
  - .left
  - .right
  - .middle

# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html dodaj przycisk (**submit**) o tytule „Zapisz tekst zapytania” pozwalający (funkcjonalność implementowana później) zapisać treść zapytania poprzez przesłanie jej jako AJAX do serwera.
2. Zdefiniuj, aby kliknięcie na przycisk (submit) nie powołało przesłania danych na serwer, ale wywołanie metody **saveQuery**, która (rozwiązanie tymczasowe) jako okno (alert) wyświetla treść zapytania.

# VUE - DIRECTIVES

- Directives are **special attributes** with the **v- prefix**.
- Directive attribute values are expected to be a single JavaScript expression (with the exception for v-for).
- A directive's job is to reactively apply side effects to the DOM when the value of its expression changes.

## v-if

```
<p v-if="seen">Now you see me</p>
```

```
<h1 v-if="ok">Yes</h1>  
<h1 v-else>No</h1>
```

## v-show

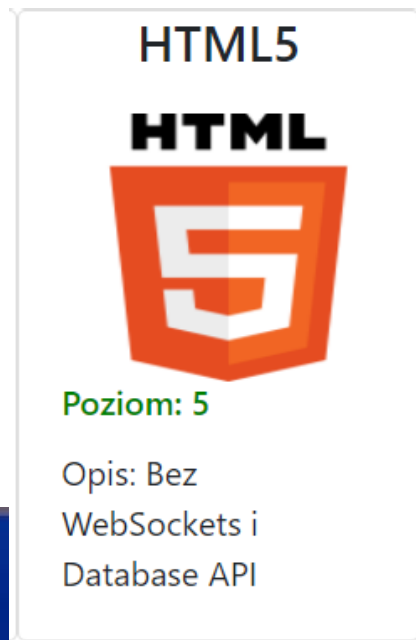
```
<h1 v-show="ok">Hello!</h1>
```

v-if has higher toggle costs while v-show has higher initial render costs.



# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html napisz kod, aby wyświetlane były tylko kafelki, gdzie poziom kompetencji jest **większy od 0** – v-if/v-show.
2. Na stronie competences.html napisz kod, aby **opis kompetencji był wyświetlany tylko jeśli istnieje** – v-if.
3. [Opcjonalne] Na stronie competences.html napisz kod, aby **poziomy kompetencji wyświetlane** były nie jako liczby (0-5), ale **tekst** (brak, podstawowy, itd.) – v-if.



# VUE - LIST RENDERING

- We can use the **v-for** directive to render a list of items based on an array.

```
<ul id="example-1">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>
```

```
var example1 = new Vue({
  el: '#example-1',
  data: {
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

- Foo
- Bar

# VUE - LIST RENDERING

- Inside v-for blocks we have **full access to parent scope** properties.
- v-for also supports an optional second argument for the **index of the current item**.

```
var example2 = new Vue({
  el: '#example-2',
  data: {
    parentMessage: 'Parent',
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

- Parent - 0 - Foo
- Parent - 1 - Bar

```
<ul id="example-2">
  <li v-for="(item, index) in items">
    {{ parentMessage }} - {{ index }} - {{ item.message }}
  </li>
</ul>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8"><title>Vue.JS</title><style></style>
</head>
<body onload="init()">
  <div id="app">
    <ul>
      <li v-for="todo in todos">
        {{ todo.text }}
      </li>
    </ul>
  </div>

  <script src="https://cdn.jsdelivr.net/vue/latest/vue.js">
  <script type="text/javascript">
function init() {
  new Vue({
    el: '#app',
    data: {
      todos: [
        { text: 'Learn JavaScript' },
        { text: 'Learn Vue.js' },
        { text: 'Build Something Awesome' }
      ]
    }
  });
}
</script>
</body></html>
```

file:///H:/nodejs-5.2.0/work/vuejs/vue-02

- Learn JavaScript
- Learn Vue.js
- Build Something Awesome

# VUE — V-FOR

- We can also use v-for to iterate through **the properties of an object**.

```
<ul id="v-for-object" class="demo">
  <li v-for="value in object">
    {{ value }}
  </li>
</ul>
```

```
new Vue({
  el: '#v-for-object',
  data: {
    object: {
      firstName: 'John',
      lastName: 'Doe',
      age: 30
    }
  }
})
```

- John
- Doe
- 30

- v-for can also **take an integer** - in this case it will repeat the template that many times.

```
<div>
  <span v-for="n in 10">{{ n }} </span>
</div>
```

1 2 3 4 5 6 7 8 9 10

- When they exist on the same node, **v-for has a higher priority than v-if**.

```
<li v-for="todo in todos" v-if="!todo.isComplete">
  {{ todo }}
</li>
```

```

<!DOCTYPE html>
<html lang="en">
<head><meta charset="utf-8"><title>Vue.JS</title><style></style></head>
<body onload="init()">
  <div id="app">
    <input v-model="newTodo" v-on:keyup.enter="addTodo">
    <ul>
      <li v-for="todo in todos">
        <span>{{ todo.text }}</span>
        <button v-on:click="removeTodo($index)">X</button>
      </li>
    </ul>
  </div>

```

```

<script src="https://cdn.jsdelivr.net/vue/latest/vue.js">
<script type="text/javascript">
function init() {

```

```

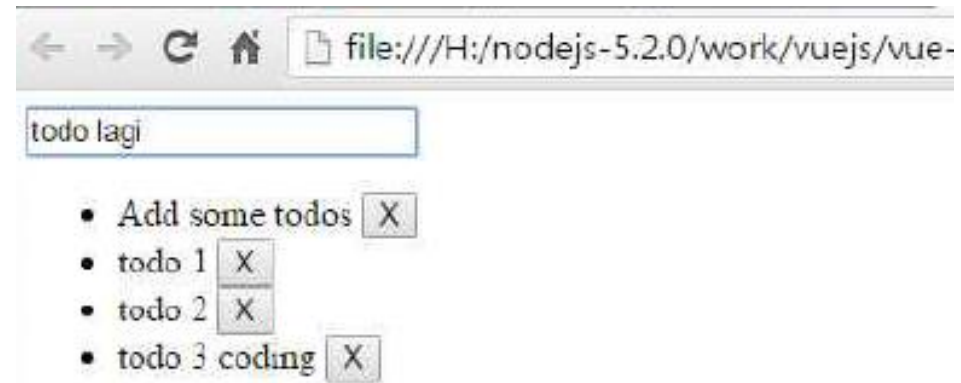
  new Vue({
    el: '#app',
    data: {
      newTodo: '',
      todos: [ { text: 'Add some todos' } ]
    },
    methods: {
      addTodo: function () {
        var text = this.newTodo.trim()
        if (text) { this.todos.push({ text: text }); this.newTodo = '' }
      },
      removeTodo: function (index) { this.todos.splice(index, 1) }
    }
  });

```

```

</script></body></html>

```



# ZADANIE – REALIZACJA V

1. Na stronie competences.html wszystkich kompetencji – **v-for**

(układ jako flex: <https://www.w3schools.com>)

## BOOTSTRAP



Poziom: 4

Opis: Wersja 3

## ANGULARJS



ANGULARJS

Poziom: 3

## ANGULAR



Poziom: 4

## IONIC



Poziom: 4

Opis: Wersja 2

## PHONEGAP



PhoneGap

Poziom: 1

Opis: Tylko absolutne podstawy

## XAMARIN



Poziom: 2

Opis: Bez API dla IOS

## HTML5



Poziom: 5

Opis: Bez WebSockets i Database API

## CSS3



Poziom: 4

## WEBAPI



Poziom: 5

# VUE - COMPUTED PROPERTIES

- **Putting too much logic** into your templates can make them bloated and hard to maintain.

```
<div id="example">
  {{ message.split('').reverse().join('') }}
</div>
```

```
<div id="example">
  <p>Original message: "{{ message }}"</p>
  <p>Computed reversed message: "{{ reversedMessage }}"</p>
</div>
```

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
})
```

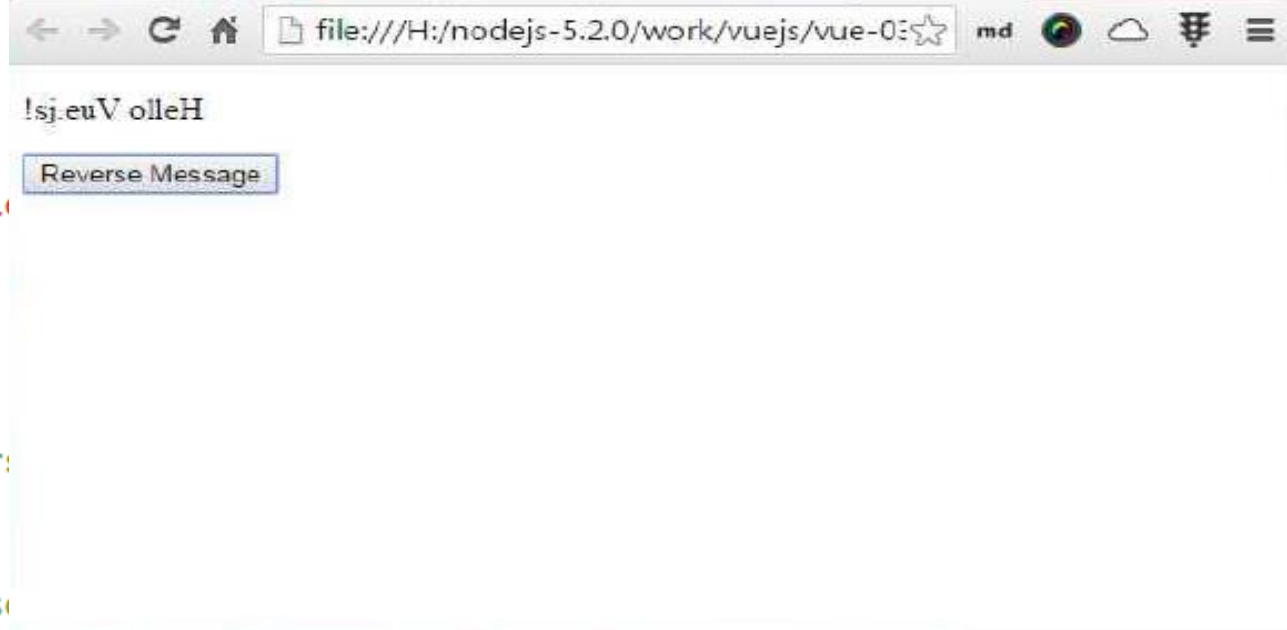


# VUE - COMPUTED PROPERTIES

- Computed properties **are cached based on their dependencies**. A computed property will only re-evaluate when some of its dependencies have changed.

```
<p>Reversed message: "{{ reverseMessage() }}"</p>
```

```
// in component
methods: {
  reverseMessage: function () {
    return this.message.split('').reverse().join('')
  }
}
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8"><title>
</head>
<body onload="init()">
  <div id="app">
    <p>{{ message }}</p>
    <button v-on:click="reverseMessage">Reverse Message</button>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.min.js"></script>
  <script type="text/javascript">
function init() {
  new Vue({
    el: '#app',
    data: {
      message: 'Hello Vue.js!'
    },
    methods: {
      reverseMessage: function () {
        this.message = this.message.split('').reverse().join('');
      }
    }
  });
}
  </script>
</body></html>
```

```
<div id="example">
  <p>Original message: "{{ message }}"</p>
  <p>Computed reversed message: "{{ reversedMessage }}"</p>
</div>
```

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
})
```

Original message: "Hello"

Computed reversed message: "olleH"

# VUE - COMPUTED PROPERTIES - EXAMPLE

```
<!-- 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',
});
```

# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie competences.html w oparciu o interfejs dla wyszukiwania kompetencji zapewnij wyszukiwanie kompetencji po ich nazwie (występowanie tekstu) - odwołania nie do bazowej listy kompetencji, ale przefiltrowanej listy kompetencji (**computed property**). Filtrowanie - [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter).

Podaj nazwę kompetencji

angular

ANGULARJS



ANGULARJS

Poziom: 3

ANGULAR



Poziom: 4

# ZADANIE – REALIZACJA WSPÓLNA

1. [Opcjonalne] Zdefiniuj wyszukiwanie po **wielu słowach** oddzielanych spacją - np. „html css” zwróci kafelki HTML5 i CSS3.
2. [Opcjonalne] Zdefiniuj listę wyboru **poziomu kompetencji** (0-5) powodującą wyświetlenie wyłącznie kompetencji o wskazanym poziomie – w połączeniu z wyszukiwaniem kompetencji.
3. [Opcjonalne] Napisz kod powodujący, że lista kompetencji jest posortowana po nazwach – zdarzenie **created** ([https://www.w3schools.com/jsref/jsref\\_sort.asp](https://www.w3schools.com/jsref/jsref_sort.asp)).

```
new Vue({  
  data: {  
    a: 1  
  },  
  created: function () {
```

# VUE - COMPONENTS

- Help to **extend basic HTML** elements to encapsulate **reusable code**. At a high level, components are custom elements that Vue's compiler attaches behavior to.
- All Vue components are also Vue instances, and so accept the same options object (except for a few root-specific options) and provide the same **lifecycle hooks**.

```
<div id="example">  
  <my-component></my-component>  
</div>
```

```
// register  
Vue.component('my-component', {  
  template: '<div>A custom component!</div>  
'})  
  
// create a root instance  
new Vue({  
  el: '#example'  
})
```



```
<div id="example">  
  <div>A custom component!</div>  
</div>
```

# VUE – COMPONENTS


```
<script>
  var vm1 = new Vue({
    el: '#appPlan',
    components: {
      'row-category':
      {
        props: ['category', 'columns'],
        template: '<tr v-on:dblclick="$emit(\'show-cate
          category[\'Kategoria\'].length == 1 || catego
          \''] == \'C.4\', summaryRowLevel2: category[\'
          \'C.3\' && category[\'Kategoria\'] != \'C.4\'
          5 || category[\'Kategoria\'].length == 6, su
          category[\'Kategoria\'] == \'NAKŁADY\' }" cla
          Object.keys(category)" v-if="key.substr(key.1
          bind:class="{planRok: columns[index].Display.
          \'">{{category[key]}}</p><p v-else>{{getNume
          methods:
          {
            getNumberAsString: function (value)...
```



# VUE - COMPONENTS

```
var data = { counter: 0 }
```

```
Vue.component('simple-counter', {  
  template: '<button v-on:click="counter += 1">{{ counter }}</button>',  
  // data is technically a function, so Vue won't  
  // complain, but we return the same object  
  // reference for each component instance  
  data: function () {  
    return data  
  }  
})  
  
new Vue({  
  el: '#example-2'  
})
```



```
data: function () {  
  return {  
    counter: 0  
  }  
}
```

```
<div id="counter-event-example">
  <p>{{ total }}</p>
  <button-counter v-on:increment="incrementTotal"></button-counter>
  <button-counter v-on:increment="incrementTotal"></button-counter>
</div>
```

```
Vue.component('button-counter', {
  template: '<button v-on:click="incrementCounter">{{ counter }}</button>',
  data: function () {
    return {
      counter: 0
    }
  },
  methods: {
    incrementCounter: function () {
      this.counter += 1
      this.$emit('increment')
    }
  },
})

new Vue({
  el: '#counter-event-example',
  data: {
    total: 0
  },
  methods: {
    incrementTotal: function () {
      this.total += 1
    }
  }
})
```

# VUE - COMPONENTS

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

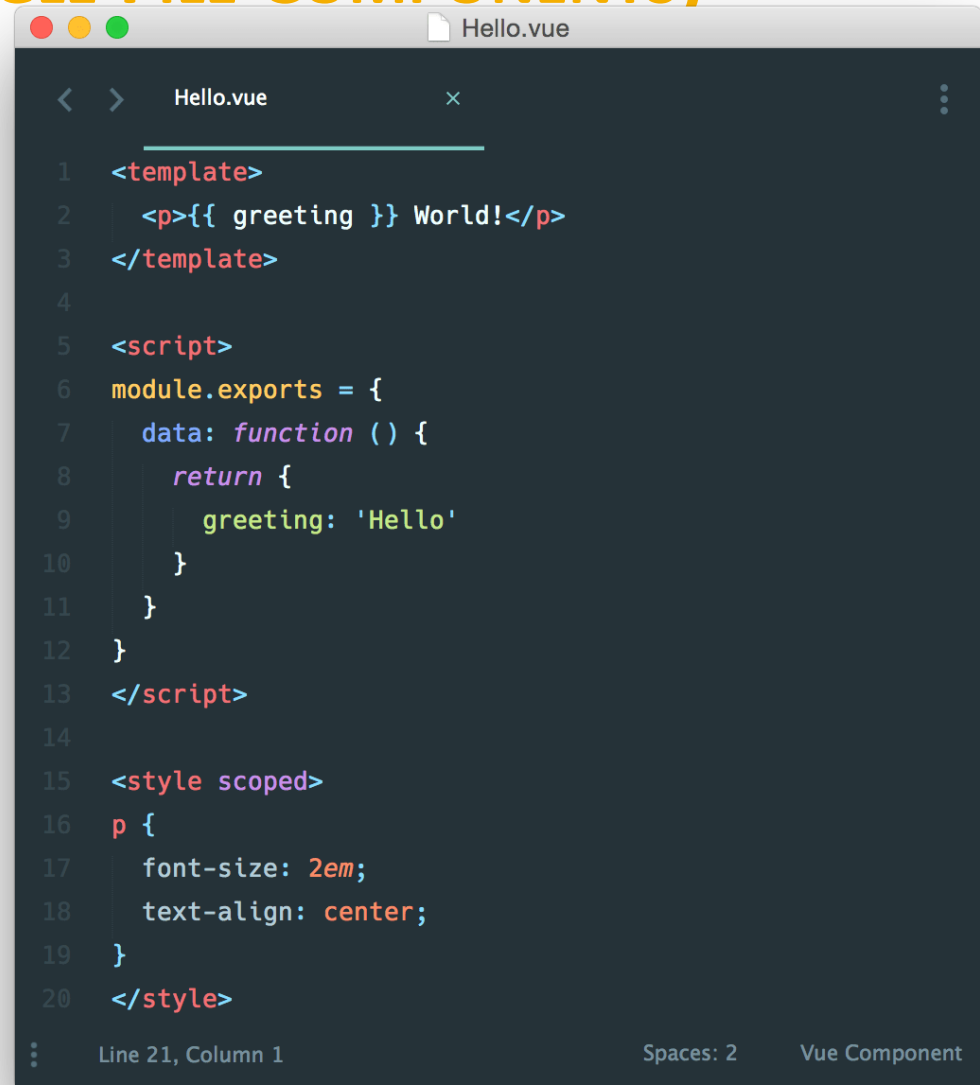
<!-- The template -->
<template id="counter-template">
  <h1>{{ header }} {{ count }}</h1>
  <button @click="count += 1">Click me!</button>
</template>

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

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

# VUE - SFC CONCEPT (SINGLE FILE COMPONENTS)

- CommonJS modules
- Component-scoped CSS
- You can specify language attributes to write more specific code.



```
1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6 module.exports = {
7   data: function () {
8     return {
9       greeting: 'Hello'
10    }
11  }
12 }
13 </script>
14
15 <style scoped>
16 p {
17   font-size: 2em;
18   text-align: center;
19 }
20 </style>
```

Line 21, Column 1      Spaces: 2      Vue Component

# ZADANIE – REALIZACJA WSPÓLNA

1. Na stronie `competences.html` napisz, aby kafelka z danymi kompetencji była **komponentem**.
2. [Opcjonalne] Dla komponentu napisz kod, aby po **najechnaniu** na niego była wypisywana (alert, okno modalne, inne rozwiązanie) lista kompetencji, które są na tym samym poziomie **umiejętności**.

## VUE - RUTER

- Creating a SPA with Vue.js + vue-router:
- Not only client-side routing (hash / history API), but also module based URL mapping:
  - nested routes and sub components
  - async load
  - others.
- The following hooks are available:
  - data, activate, deactivate, canActivate, canDeactivate, canReuse
  - onEnter, onLeave, onChange in Vue-Router since 0.8.0

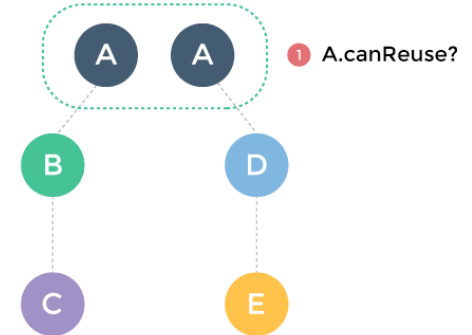
# VUE - RUTER

## Nested Routes and Sub Components & Hooks

Example URL change: /a/b/c -> /a/d/e

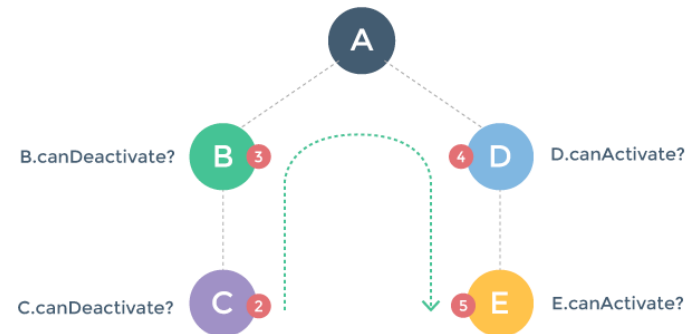
### 1. Reusability phase

Check reusability



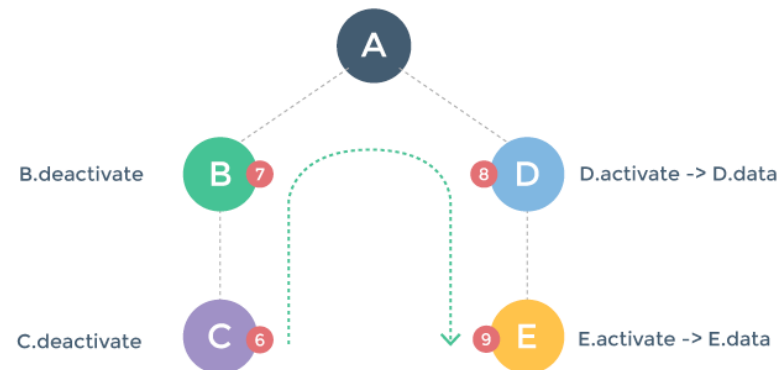
### 2. Validation phase

Check if all current components can be activated / deactivated



### 3. Activation phase

Deactivate current components and activate new components with data hook



# VUE - RUTER

## Code Example

### main.vue

```
router.map({
  '/about': {
    component: require('./components/about.vue')
  },
  '/user/:userId': {
    component:
require('./components/user/index.vue'),
    subRoutes: {
      'profile/:something': {
        component:
require('./components/user/profile.vue')
      }
    },
    '*': {
      component: require('./components/not-found.vue')
    }
  }
})
```

### app.vue

```
<template>
  <div>
    <p v-show="authenticating" style="color:red">Authenticating...</p>
    <h1>App Header</h1>
    <a v-link="{ path: '/about' }">about</a>
    <a v-link="{ path: '/user/1234/profile/what' }">user</a>
    <a v-link="{ path: '/mypage' }">mypage</a>
    <router-view class="view" transition="test" transition-mode="out-in" keep-
alive></router-view>
  </div>
</template>
```



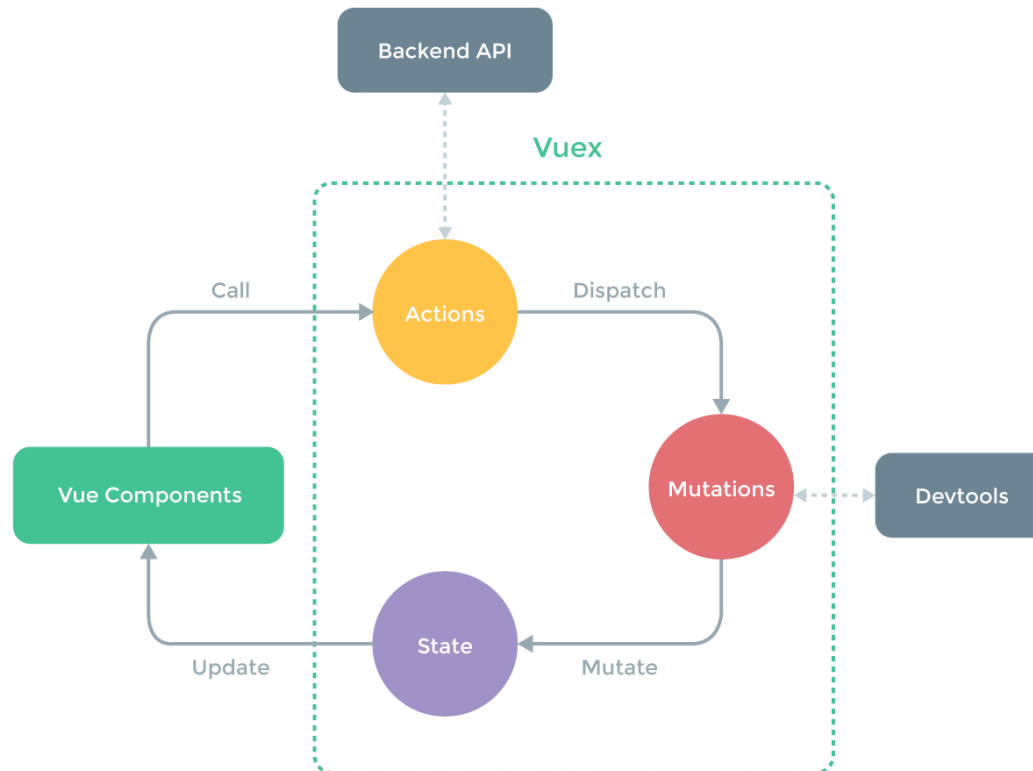
### components/user/index.vue

```
<template>
  <div>
    <h2>User yo</h2>
    <router-view></router-view>
  </div>
</template>
```



# VUEX

1. Official **redux** inspired flux framework for Vue.js
2. Vuex is more fit for Vue.js with efficient **reactive system** such as data reactive rendering and components systems.



# VUEX - FEATURES

- Store
  - basically a container that holds your application reactive states
  - only way to change a store's state is by explicitly dispatching mutations
- Mutation
  - sync (can not be async)
  - split into modules with corresponding slice of the state
- Action
  - dispatch mutations
  - can be async

## VUEX - ADVANTAGES

- simple unidirectional flow (less side effects)
- easy undo/redo - time travel
- hot reloading
- easy test

# VUEX - FLOW

Increment  
Component

A user clicks the “Increment +1” button. This calls a special function on the view model *incrementCounter* which is called an **action**.

↓  
Calls

Action

The action is a function which determines what updates must be applied on a **store**. The action **cannot update the state** directly. Instead it has to *dispatch* a *mutation* of type *INCREMENT*.

↓  
Dispatches

Mutation

A mutation is a **command** to update the store in a very specific way. A mutation has a **type** which is a string and optionally some arguments. In this example, we will define the *INCREMENT* mutation which will actually increment the counter value.

↓  
Updates

Store / State

The store is a single object containing the current **state** of the application. Every component can read from the store via **getters**.

↓  
Notifies

Getter

A **getter** is a function which extracts a part of the state which is useful to the component. Getters are similar to computed functions and have reactive properties.

↓  
Reflects new Value

Display Component

The display component uses a getter to find the value of interest - which is the current count.

## VUEX – MUTATION EXAMPLE

- Mutation (a name and a handler) **mutates states** with synchronous functions:
  - Vuex store's state is made reactive by Vue;
  - when we mutate the state, Vue components observing the state will update automatically.

```
import Vuex from 'vuex'
```

```
const store = new Vuex.Store({  
  state: {  
    count: 1  
  },  
  mutations: {  
    INCREMENT (state) {  
      // mutate state  
      state.count++  
    }  
  }  
})
```

Must be called by dispatch in Action

```
store.dispatch('INCREMENT')
```

# VUEX - GETTER

A getter provides **accessing way to store data from components**:

- Components are not allowed to directly mutate a state.
- It allows us to reuse computed methods (e.g. totals, averages) through a getter.

```
- Getter Example
// getters.js
export function filteredMessages (state) {
  return state.messages.filter(message => message.threadID === state.currentThreadID)
}
```

```
- Component
<script>
export default {
  vuex: {
    getters: {
      filteredMessages
    }
  }
}
</script>
```

```
<template>
  <div>
    <ul>
      <li v-for="msg in filteredMessages">{{msg.title}}</li>
    </ul>
  </div>
</template>
```

## VUEX – FORM HANDLING

- v-model directly rewrites store data. This is not the Vuex way (throws an error in strict mode).
- The Vuex way with v-model:
  - Set a getter value to form value
  - Dispatch action by UI event (e.g. onclick)

Template:

```
<input :value="message" @input="updateMessage">
```

JS:

```
actions: {  
  updateMessage: ({ dispatch }, e) => {  
    dispatch('UPDATE_MESSAGE', e.target.value)  
  }  
}
```