



Vue JS

<https://vuejs.org/>

Nasr Kassem

Email : nabelaziz@iti.gov.eg

Mobile : +201090290520

# What is Vue JS ?

- Vue (pronounced /vju:/, like **view**) is a **progressive framework** for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with [modern tooling](#) and [supporting libraries](#).

# What is Vue JS ?

- Vue.js is a framework that was created in 2014 by Evan You. More precisely, it's a **progressive** framework that allows us to build **user interfaces**. Progressive means Vue can be adopted incrementally, it's easy to integrate it in a project.
- If you already wrote some web applications with vanilla JavaScript (pure JS) or jQuery, things are going to be way different here.
- Vue makes it really easy to **bind data** to your HTML elements and is **reactive**. It's also a performant framework that is **lightweight (<80kb)**.
- Furthermore, Vue is **component-based**. It's easy to create components, compose them and reuse them.

# What is Vue JS ? Continue...

- **VueJS** is an open source progressive JavaScript framework used to develop interactive web interfaces. It is one of the famous frameworks used to simplify web development. VueJS focusses on the view layer. It can be easily integrated into big projects for front-end development without any issues.
- The installation for VueJS is very easy to start with. Any developer can easily understand and build interactive web interfaces in a matter of time. VueJS is created by Evan You, an ex-employee from Google. The first version of VueJS was released in Feb 2014. It recently has clocked to 64,828 stars on GitHub, making it very popular.

# Frontend Frameworks

FrontEnd  
Frameworks



<https://existek.com/blog/top-front-end-frameworks-2020/>  
<https://vuejs.org/v2/guide/comparison.html>

# Vue, React or Angular ?

- JavaScript frameworks. Choosing one is always a hard task. The goal here is not to say who is the best or who is the worst. Every framework has its pros and cons. I just want to let you know in which cases you may want to choose Vue over the other frameworks
- **Vue is a good fit for you if...**
  - ✓ You want to learn a framework that has the easiest learning curve
  - ✓ You want the most lightweight framework
  - ✓ You want to use it for a little or medium project
  - ✓ You work alone or in a small team

# Vue, React And Angular Comparison

VS vs React  
and Angular

	Angular	React	Vue
Initial release	2010	2013	2014
Official site	<a href="https://angular.io">angular.io</a>	<a href="https://reactjs.org">reactjs.org</a>	<a href="https://vuejs.org">vuejs.org</a>
Approx. size (KB)	500	100	80
Current version	7	16.6.3	2.17
Used by	Google, Wix	Facebook, Uber	Alibaba, GitLab

# Vue, React And Angular Comparison

VS vs React  
and Angular

Decision point	Angular 2	React	Vue.js
Stable	Yes	Yes	Yes
Backed by a strong community or some big players	Yes, huge community and Google is behind it	Yes, huge community and Facebook backs it	Not as huge but is big enough and is backed by Laravel and Alibaba
Good documentation	Yes	Yes	Yes
Easy to learn	Not with Typescript	Kind of	Yes
Integration with Bootstrap	Yes	Yes	Yes
Small	566K	139K	58.8 K
Allow us to reuse code	Yes	No, only CSS	Yes, HTML and CSS
Coding speed	Slow	Normal	Fast
Reactivity	Kind of	Yes	Yes
Component based	Yes	Yes	Yes

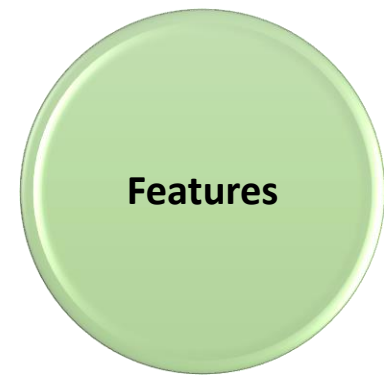


# Vue, React And Angular Comparison



**VS vs React  
and Angular**

<https://www.stefankrause.net/js-frameworks-benchmark7/table.html>



# Vue JS Features

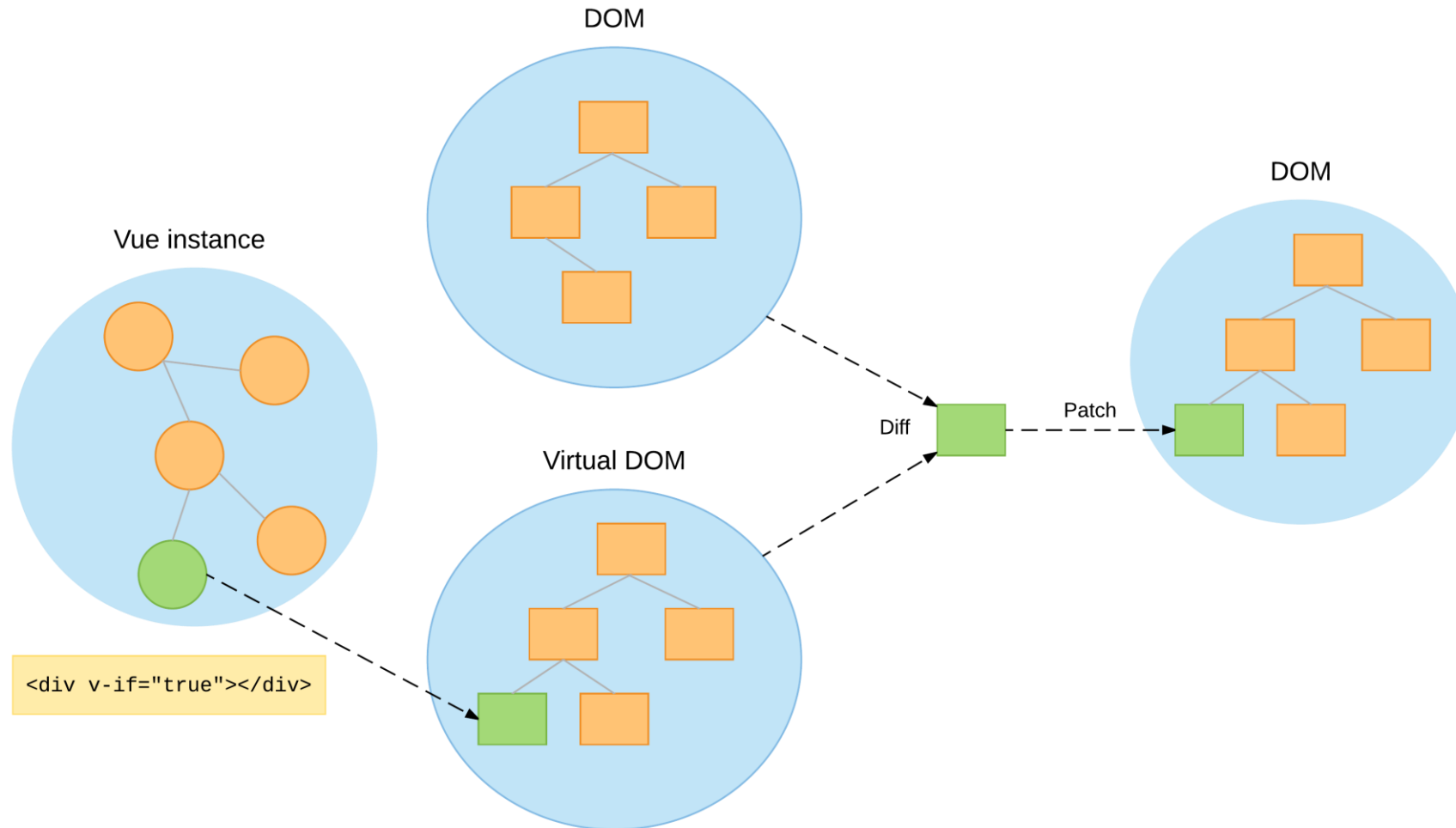
- Virtual DOM
- Data Binding
- Components
- Event Handling
- Animation/Transition
- Computed Properties
- Templates
- Directives
- Watchers
- Routing
- Lightweight
- Vue-CLI
- State Management
- Mixins
- And more ...



# Vue JS Features - Virtual DOM

- VueJS makes the use of virtual DOM, which is also used by other frameworks such as React, Ember, etc. The changes are not made to the DOM, instead a replica of the DOM is created which is present in the form of JavaScript data structures. Whenever any changes are to be made, they are made to the JavaScript data structures and the latter is compared with the original data structure. The final changes are then updated to the real DOM, which the user will see changing. This is good in terms of optimization, it is less expensive and the changes can be made at a faster rate.

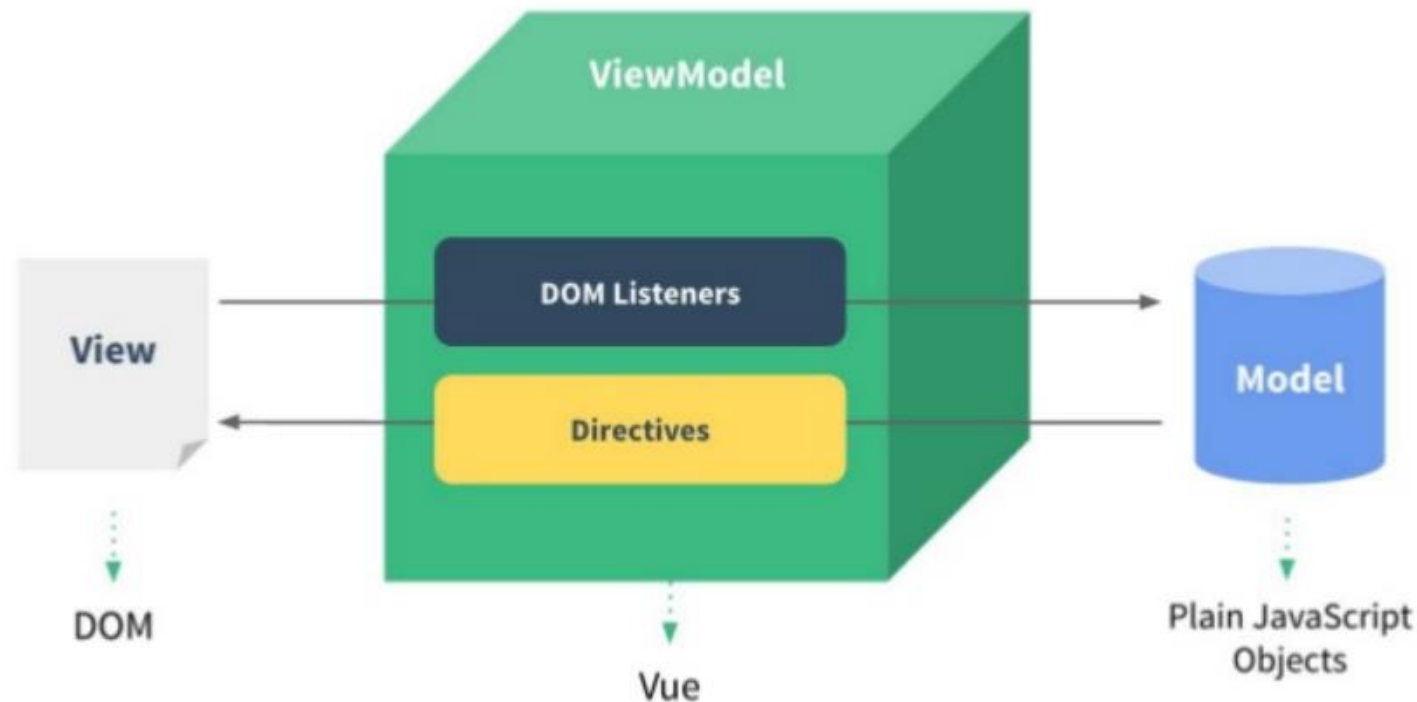
# Vue JS Features - Virtual DOM



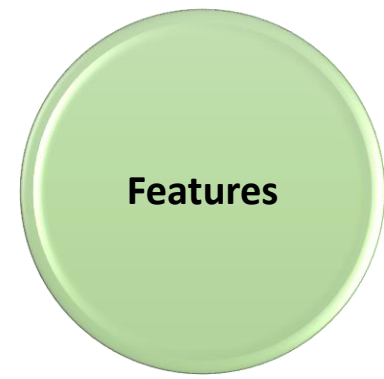
# Vue JS Features - Data Binding

- The data binding feature helps manipulate or assign values to HTML attributes, change the style, assign classes with the help of binding directive called **v-bind** available with VueJS.

**MVVM**



# Vue JS Features - Components



- Components are one of the important features of VueJS that helps create custom elements, which can be reused in HTML



# Vue JS Features - Event Handling

- **v-on** is the attribute added to the DOM elements to listen to the events in VueJS.



# Vue JS Features - Animation/Transition

- VueJS provides various ways to apply transition to HTML elements when they are added/updated or removed from the DOM. VueJS has a built-in transition component that needs to be wrapped around the element for transition effect. We can easily add third party animation libraries and also add more interactivity to the interface.



# Vue JS Features - **Computed Properties**

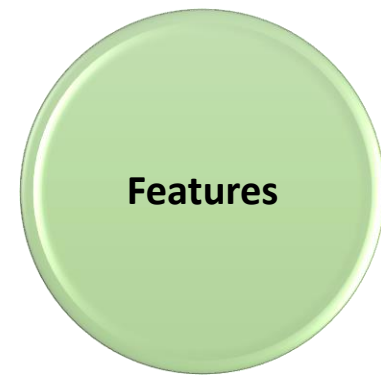


- This is one of the important features of VueJS. It helps to listen to the changes made to the UI elements and performs the necessary calculations. There is no need of additional coding for this.



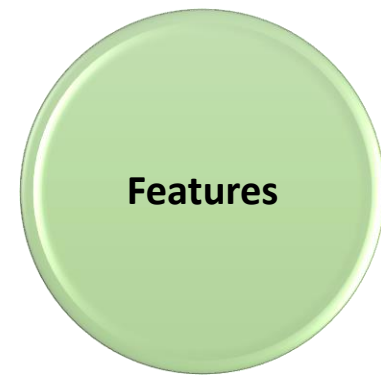
# Vue JS Features - Templates

- VueJS provides HTML-based templates that bind the DOM with the Vue instance data. Vue compiles the templates into virtual DOM Render functions. We can make use of the template of the render functions and to do so we have to replace the template with the render function.



# Vue JS Features - Directives

- VueJS has built-in directives such as v-if, v-else, v-show, v-on, v-bind, and v-model, which are used to perform various actions on the frontend.



# Vue JS Features - Watchers

- Watchers are applied to data that changes. For example, form input elements. Here, we don't have to add any additional events. Watcher takes care of handling any data changes making the code simple and fast.

# Vue JS Features - Routing

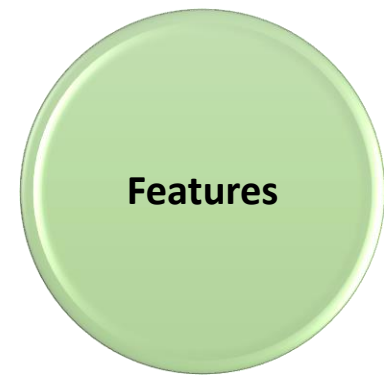


- Navigation between pages is performed with the help of vue-router.

# Vue JS Features - Lightweight



- VueJS script is very lightweight and the performance is also very fast.



# Vue JS Features - Vue-CLI

- VueJS can be installed at the command line using the vue-cli command line interface. It helps to build and compile the project easily using vue-cli.

# Environment Setup



- here are many ways to install VueJS
- Using the `<script>` tag directly in HTML file
- Using CDN
- Using NPM



# Prerequisites

- HTML
- CSS
- JavaScript



# Example

Example

```
<html>
  <head>
    <title>VueJs Introduction</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "intro" style = "text-align:center;">
      <h1>{{ message }}</h1>
    </div>
    <script type = "text/javascript">
      var vue_det = new Vue({
        el: '#intro',
        data: {
          message: 'My first VueJS Task'
        }
      });
    </script>
  </body>
</html>
```




# VueJS - Instances



- To start with VueJS, we need to create the instance of Vue, which is called the **root Vue Instance**.
- `var app = new Vue({`
- `// options`
- `})`

# VueJS – Instances , Continue ..



Instance

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "vue_det">
      <h1>Firstname : {{firstname}}</h1>
      <h1>Lastname : {{lastname}}</h1>
      <h1>{{mydetails()}}</h1>
    </div>
    <script type = "text/javascript" src = "js/vue_instance.js"></script>
  </body>
</html>
```

```
var vm = new Vue({
  el: '#vue_det',
  data: {
    firstname : "Ria",
    lastname  : "Singh",
    address   : "Mumbai"
  },
  methods: {
    mydetails : function() {
      return "I am " + this.firstname + " " + this.lastname;
    }
  }
})
```

# VueJS – Instances , Continue ..



- Log Vue Instance And Check Proxies And Reactivity
- \$options
- \$data



# Directive

- Directives are instruction for VueJS to do things in a certain way such as v-if, v-show, v-else, v-for, v-bind , v-model, v-on, etc..
- You can create your own directive

## Syntax

```
Vue.directive('nameofthedirective', {  
  bind(e1, binding, vnode) {  
  }  
})
```

```
<html>  
  <head>  
    <title>VueJs Instance</title>  
    <script type = "text/javascript" src = "vueobject.js"></script>  
  </head>  
  <body>  
    <div id = "databinding">  
      <div v-changestyle>VueJS Directive</div>  
    </div>  
    <script type = "text/javascript">  
      Vue.directive("changestyle",{  
        bind(e1) {  
          console.log(e1);  
          e1.style.color = "red";  
          e1.style.fontSize = "30px";  
        }  
      });  
      var vm = new Vue({  
        el: '#databinding',  
      });  
    </script>  
  </body>  
</html>
```

---

VueJS Directive



# Directive , Continue..

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <div v-changestyle = "{color:'green'}">VueJS Directive</div>
    </div>
    <script type = "text/javascript">
      Vue.directive("changestyle",{
        bind(e1, binding, vnode) {
          console.log(e1);
          console.log(binding.value.color);
          console.log(vnode);
          e1.style.color=binding.value.color;
          e1.style.fontSize = "30px";
        }
      });
      var vm = new Vue({
        el: '#databinding',
      });
    </script>
  </body>
</html>
```

binding.value here will  
return an object {color:'green'}

# Directive , Using Vue Data in Template

- ✓ Mustache , string Interpolation {{ }}
- ✓ Mustache not applicable with HTML Attributes !

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "vue_det">
      <h1>Firstname : {{firstname}}</h1>
      <h1>Lastname : {{lastname}}</h1>
      <h1>{{mydetails()}}</h1>
    </div>
    <script type = "text/javascript" src = "js/vue_instance.js"></script>
  </body>
</html>
```

```
var vm = new Vue({
  el: '#vue_det',
  data: {
    firstname : "Ria",
    lastname  : "Singh",
    address   : "Mumbai"
  },
  methods: {
    mydetails : function() {
      return "I am "+this.firstname +" "+ this.lastname;
    }
  }
})
```





# Directive , Built In Directives ...

- `v-html` XSS cross site scripting
- `v-text`
- `v-bind` [using shortcut `:`] => One Way Data Binding
- `V-model` => Two Way Data Binding
- `v-once`
- `v-if`
- `v-if-else`
- `v-else`
- `v-show`
- `v-on` [using shortcut `@`]
- `v-for` [arrays , range , object]

# Directive , Built In Directives ...

Directive

## EXPRESSIONS

```
<div id="app">
  <p>I have a {{ product }}</p>
  <p>{{ product + 's' }}</p>
  <p>{{ isWorking ? 'YES' : 'NO' }}</p>
  <p>{{ product.getSalePrice() }}</p>
</div>
```

## BINDING

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

shorthand

```
<a :href="url">...</a>
```

True or false will add or remove attribute:

```
<button :disabled="isButtonDisabled">...
```

If isActive is truthy, the class 'active' will appear:

```
<div :class="{ active: isActive }">...
```

Style color set to value of activeColor:

```
<div :style="{ color: activeColor }">
```

# Directive , Built In Directives ...

## Class binding



shapes:[

```
{type:'circle',isrounded:true,isanimated:true},  
{type:'rectangle',isrounded:false,isanimated:false},  
{type:'circle',isrounded:true,isanimated:false},  
{type:'triangle',isanimated:true,direction:'up'},  
{type:'rectangle',isrounded:false,isanimated:false},  
{type:'circle',isrounded:true,isanimated:false},  
{type:'triangle',isanimated:true,direction:'down'},
```

]

```
<div v-for="shape in shapes" v-  
bind:class="['shape',shape.type,shape.direction?shape.direction:'',shape.isanimated?'animation':'']"></div>
```

# Directive , Built In Directives ...

Directive

## DIRECTIVES

Element inserted/removed based on truthiness:

```
<p v-if="inStock">{{ product }}</p>
```

```
<p v-else-if="onSale">...</p>
<p v-else>...</p>
```

Toggles the display: none CSS property:

```
<p v-show="showProductDetails">...</p>
```

Two-way data binding:

```
<input v-model="firstName" >
```

`v-model.lazy`="..." Syncs input after change event

`v-model.number`="..." Always returns a number

`v-model.trim`="..." Strips whitespace

## LIST RENDERING

```
<li v-for="item in items" :key="item.id">
  {{ item }}
</li>
```

key always recommended

To access the position in the array:

```
<li v-for="(item, index) in items">...
```

To iterate through objects:

```
<li v-for="(value, key) in object">...
```

Using v-for with a component:

```
<cart-product v-for="item in products"
  :product="item" :key="item.id">
```

# Directive , Built In Directives ... V-for

```
<ul>
```

```
  <li v-for="(item,index) in items"> {{item.itemname}}
```

```
    <ol>
```

```
      <li v-for="(itemvalue,itemkey,itemindex) in item"> {{itemkey}} = {{itemvalue}}</li>
```

```
    </ol>
```

```
  </li>
```

```
</ul>
```

**\*\* LOOPING THROUGH NUMBER RANGE**

```
<div>
```

```
  <p v-for="(n,index) in 10">{{n}} at {{index}}</p>
```

```
</div>
```

**\*\* randomization for an array !!!**

# Directive , Built In Directives ...

## v-model.trim , v-model.lazy , v-model.number



### \*- Two Way Data Binding **v-model**

ex: `<input type="text" v-model:value="message"/>`

or `<input type="text" v-model="message"/>` [without :value]

### **Trim**

```
<textarea cols="30" rows="10" v-model.trim="message"></textarea>
```

```
<p style="border:2px solid black;white-space: pre;">{{message}}</p>
```

### **Number And Lazy**

```
<input type="number" value="5" v-model.number.lazy="id" />
```

**"this will not work for type number" ' if you want to check lazy with number change type to text'**

```
<input type="text" value="5" v-model.number.lazy="id" />
```

```
<p>{{id +20}}</p>
```

# Directive , Built In Directives ...

Directive

## ACTIONS / EVENTS

Calls addToCart method on component:

```
<button v-on:click="addToCart">...
```



shorthand

```
<button @click="addToCart">...
```

Arguments can be passed:

```
<button @click="addToCart(product)">...
```

To prevent default behavior (e.g. page reload):

```
<form @submit.prevent="addProduct">...
```

Only trigger once:

```
<img @mouseover.once="showImage">...
```

.stop

Stop all event propagation

.self

Only trigger if event.target is element itself

Keyboard entry example:

```
<input @keyup.enter="submit">
```

Call onCopy when control-c is pressed:

```
<input @keyup.ctrl.c="onCopy">
```

Key modifiers:

.tab	.up	.ctrl
.delete	.down	.alt
.esc	.left	.shift
.space	.right	.meta

Mouse modifiers:

.left	.right	.middle
-------	--------	---------

# Directive , Event Modifiers



- stop
- prevent
- once `<a v-on:click.once="doThis"></a>`
- self
- passive `<div v-on:scroll.passive="onScroll">...</div>` used to improve performance on mobile devices

<https://vuejs.org/v2/guide/events.html>



# Directive , Key Modifiers



Directive

When listening for keyboard events, we often need to check for specific keys. Vue allows adding key modifiers for `v-on` when listening for key events:

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

HTML

You can directly use any valid key names exposed via `KeyboardEvent.key` as modifiers by converting them to kebab-case.

```
<input v-on:keyup.page-down="onPageDown">
```

HTML

<https://vuejs.org/v2/guide/events.html>

# Directive , Key Codes



Using `keyCode` attributes is also permitted:

```
<input v-on:keyup.13="submit">
```

HTML

Vue provides aliases for the most commonly used key codes when necessary for legacy browser support:

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

<https://vuejs.org/v2/guide/events.html>

# Directive , Modifiers Keys




- ✓ .ctrl
- ✓ .alt
- ✓ .shift
- ✓ .meta On Macintosh keyboards, meta is the command key (⌘). On Windows keyboards, meta is the Windows key (⊞)

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

HTML

<https://vuejs.org/v2/guide/events.html>

# Directive , Exact Modifier



Directive

```
<!-- this will only fire when no system modifiers are pressed -->  
<button v-on:click.exact="onClick">A</button>
```

<https://vuejs.org/v2/guide/events.html>

# Instance Loading , **v-cloak**

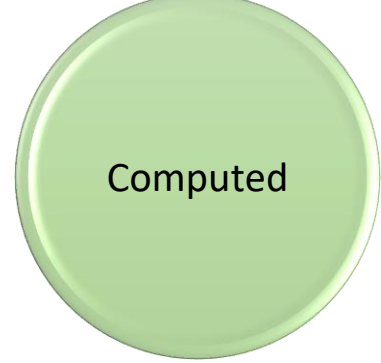


**hiding the elements until the Vue Instance is ready using (v-cloak) , you must set style for [v-cloak] {display:none;}**

# Methods



# Computed Properties



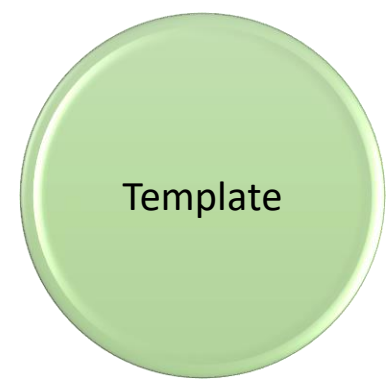
- Putting too much logic in your templates can make them bloated and hard to maintain and not declarative.
- Computed props are being **cached**.
- Dependencies will be **auto tracked**.

# Watchers

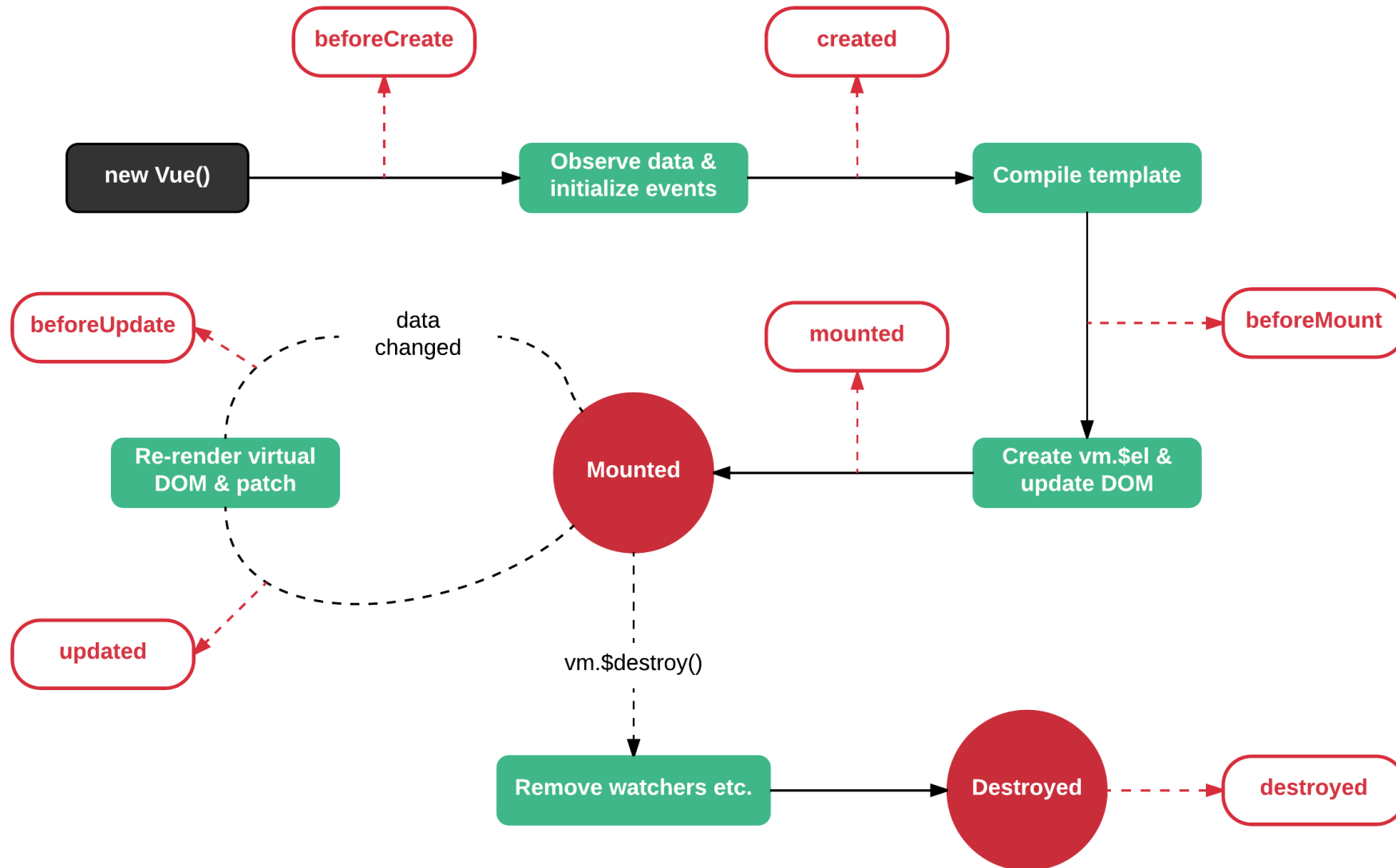




# Template



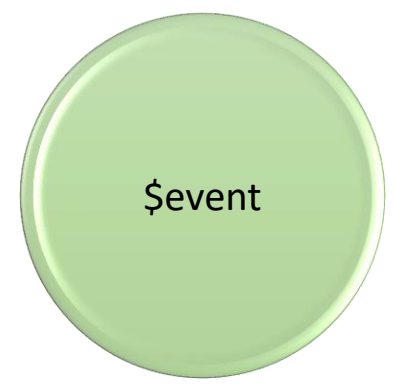
# Vue Instance – lifecycle hooks



\$destroy



\$event



\$refs

