VUEJS

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

VueJS is a progressive JavaScript framework used to develop interactive web interfaces. Focus is more on the view part, which is the front end. It is very easy to integrate with other projects and libraries.

The installation of VueJS is fairly simple, and beginners can easily understand and start building their own user interfaces. The content is divided into various chapters that contain related topics with simple and useful examples.

Audience

This tutorial is designed for software programmers who want to learn the basics of VueJS and its programming concepts in a simple and easy manner. This tutorial will give the readers enough understanding on the various functionalities of VueJS from where they can take themselves to the next level.

Prerequisites

Before proceeding with this tutorial, readers should have a basic understanding of HTML, CSS, and JavaScript.

Copyright &Disclaimer

Table of Contents

1. VueJS – Overview

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.

Features

Following are the features available with VueJS.

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.

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.

Components

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

Event Handling

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

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.

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.

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.

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.

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.

Routing

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

Lightweight

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

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.

Comparison with Other Frameworks

Now let us compare VueJS with other frameworks such as React, Angular, Ember, Knockout, and Polymer.

VueJS v/s React

Virtual DOM

Virtual DOM is a virtual representation of the DOM tree. With virtual DOM, a JavaScript object is created which is the same as the real DOM. Any time a change needs to be made to the

DOM, a new JavaScript object is created and the changes are made. Later, both the JavaScript objects are compared and the final changes are updated in the real DOM.

VueJS and React both use virtual DOM, which makes it faster.

Template v/s JSX

VueJS uses html, js and css separately. It is very easy for a beginner to understand and adopt the VueJS style. The template based approach for VueJS is very easy.

React uses jsx approach. Everything is JavaScript for ReactJS. HTML and CSS are all part of JavaScript.

Installation Tools
React uses **create react app** and VueJS uses **vue-cli /CDN/npm**. Both are very easy to use and the project is set up with all the basic requirements. React needs webpack for the build, whereas VueJS does not. We can start with VueJS coding anywhere in jsfiddle or codepen using the cdn library.

Popularity
React is popular than VueJS. The job opportunity with React is more than VueJS. There is a big name behind React i.e. Facebook which makes it more popular. Since, React uses the core concept of JavaScript, it uses the best practice of JavaScript. One who works with React will definitely be a very good with all the JavaScript concepts.

VueJS is a developing framework. Presently, the job opportunities with VueJS are less in comparison to React. According to a survey, many people are adapting to VueJS, which can make it more popular in comparison to React and Angular. There is a good community working on the different features of VueJS. The vue-router is maintained by this community with regular updates.

VueJS has taken the good parts from Angular and React and has built a powerful library. VueJS is much faster in comparison to React/Angular because of its lightweight library.

VueJS v/s Angular

Similarities
VueJS has a lot of similarities with Angular. Directives such as v-if, v-for are almost similar to ngIf, ngFor of Angular. They both have a command line interface for project installation and to build it. VueJS uses Vue-cli and Angular uses angular-cli. Both offer two-way data binding, server side rendering, etc.

Complexity
Vuejs is very easy to learn and start with. As discussed earlier, a beginner can take the CDN library of VueJS and get started in codepen and jsfiddle.

For Angular, we need to go through a series of steps for installation and it is little difficult for beginners to get started with Angular. It uses TypeScript for coding which is difficult for people coming from core JavaScript background. However, it is easier to learn for users belonging to Java and C# background.

Performance
To decide the performance, it is up to the users. VueJS file size is much lighter than Angular. A comparison of the framework performance is provided in the following link http://stefankrause.net/js-frameworks-benchmark4/webdriver-ts/table.html

Popularity
At present, Angular is more popular than VueJS. A lot of organizations use Angular, making it very popular. Job opportunities are also more for candidates experienced in Angular. However, VueJS is taking up the place in the market and can be considered as a good competitor for Angular and React.

Dependencies
Angular provides a lot of built-in features. We have to import the required modules and get started with it, for example, @angular/animations, @angular/form.

VueJS does not have all the built-in features as Angular and needs to depend on third party libraries to work on it.

Flexibility
VueJS can be easily merged with any other big project without any issues. Angular will not be that easy to start working with any other existing project.

Backward Compatibility
We had AngularJS, Angular2 and now Angular4. AngularJS and Angular2 have vast difference. Project application developed in AngularJS cannot be converted to Angular2 because of the core differences.

The recent version of VueJS is 2.0 and it is good with backward compatibility. It provides good documentation, which is very easy to understand.

Typescript
Angular uses TypeScript for its coding. Users need to have knowledge of Typescript to get started with Angular. However, we can start with VueJS coding anywhere in jsfiddle or codepen using the cdn library. We can work with standard JavaScript, which is very easy to start with.

VueJS v/s Ember

Similarities
Ember provides Ember command line tool, i.e. ember-cli for easy installation and compiling for Ember projects.

VueJS has also a command line tool vue-cli to start and build projects.

They both have features such as router, template, and components which makes them very rich as the UI framework.

Performance
VueJS has better performance in comparison to Ember. Ember has added a glimmer rendering engine with the aim of improving the re-render performance, which is a similar concept as

VueJS and React using virtual DOM. However, VueJS has a better performance when compared to Ember.

VueJS v/s Knockout

Knockout provides a good browser support. It is supported on the lower version of the IE whereas VueJS is not supported on IE8 and below. Knockout development has slowed down over time. There is not much popularity for the same in recent times.

On the other hand, VueJS has started gaining popularity with the Vue team providing regular updates.

VueJS v/s Polymer

Polymer library has been developed by Google. It is used in many Google projects such as Google I/O, Google Earth, Google Play Music, etc. It offers data binding and computed properties similar to VueJS.

Polymer custom element definition comprises plain JavaScript/CSS, element properties, lifecycle callbacks, and JavaScript methods. In comparison, VueJS allows to easily use JavaScript/html and CSS.

Polymer uses web component features and requires polyfills for browsers, which does not support these features. VueJS does not have such dependencies and works fine in all browsers from IE9+.

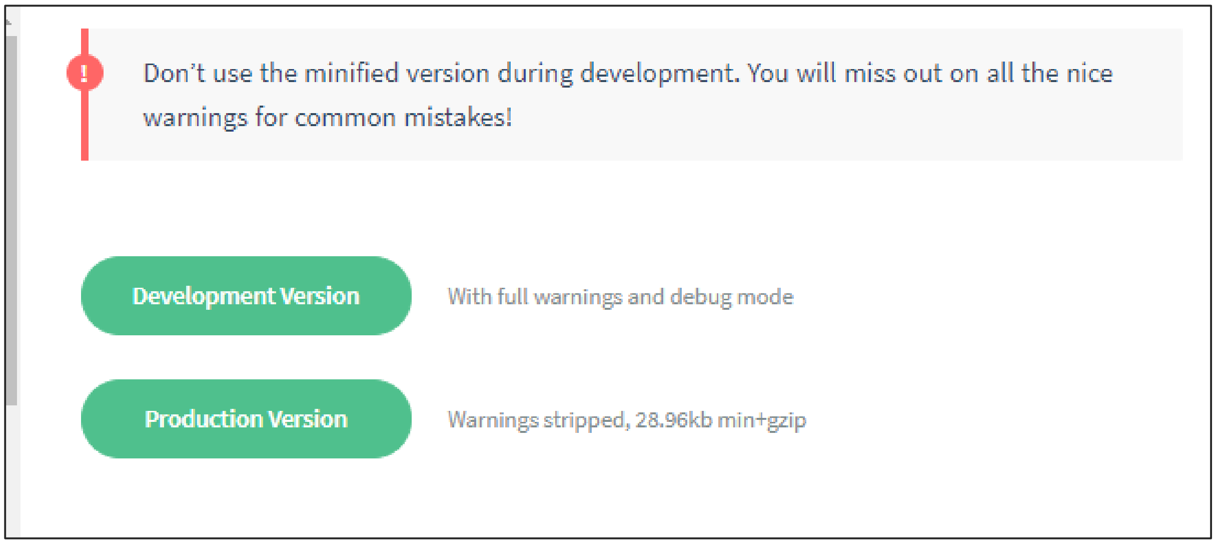
2. VueJS – Environment Setup

There are many ways to install VueJS. Some of the ways on how to carry out the installation are discussed ahead.

Using the <script> tag directly in HTML file

```
<html>
<head>
<script type="text/javascript" src="vue.min.js"></script>
</head>
<body>
</body>
</html>
```

Go to the home site **https://vuejs.org/v2/guide/installation.html** of VueJS and download the vue.js as per need. There are two versions for use - production version and development version. The development version is not minimized, whereas the production version is minimized as shown in the following screenshot. Development version will help with the warnings and debug mode during the development of the project.



Don't use the minified version during development. You will miss out on all the nice warnings for common mistakes!

Development Version
With full warnings and debug mode
Production Version
Warnings stripped, 28.96kb min+gzip

Using CDN

We can also start using VueJS file from the CDN library. The link https://unpkg.com/vue will give the latest version of VueJS. VueJS is also available on jsDelivr (https://cdn.jsdelivr.net/npm/vue/dist/vue.js) and cdnjs (https://cdnjs.cloudflare.com/ajax/libs/vue/2.4.0/vue.js).

We can host the files at our end, if required and get started with VueJS development.

Using NPM

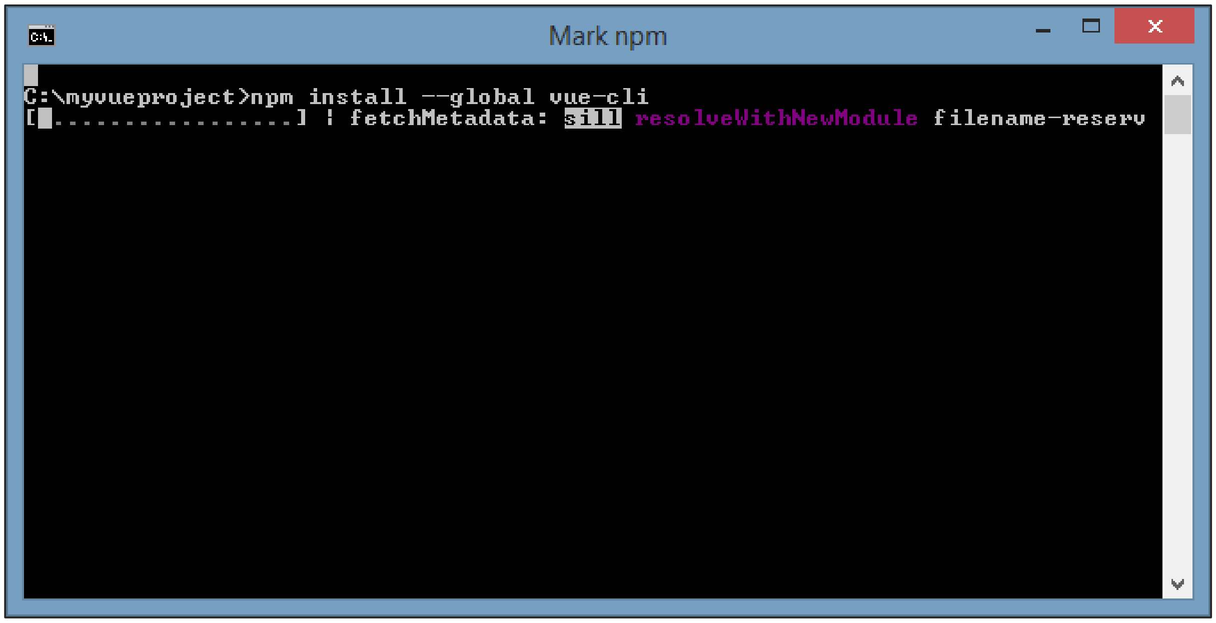
For large scale applications with VueJS, it is recommended to install using the npm package. It comes with Browserify and Webpack along with other necessary tools, which help with the development. Following is the command to install using npm.

```
npm  install vue
```

Using CLI Command Line

VueJS also provides CLI to install the vue and get started with the server activation. To install using CLI, we need to have CLI installed which is done using the following command.

```
npm install --global vue-cli
```



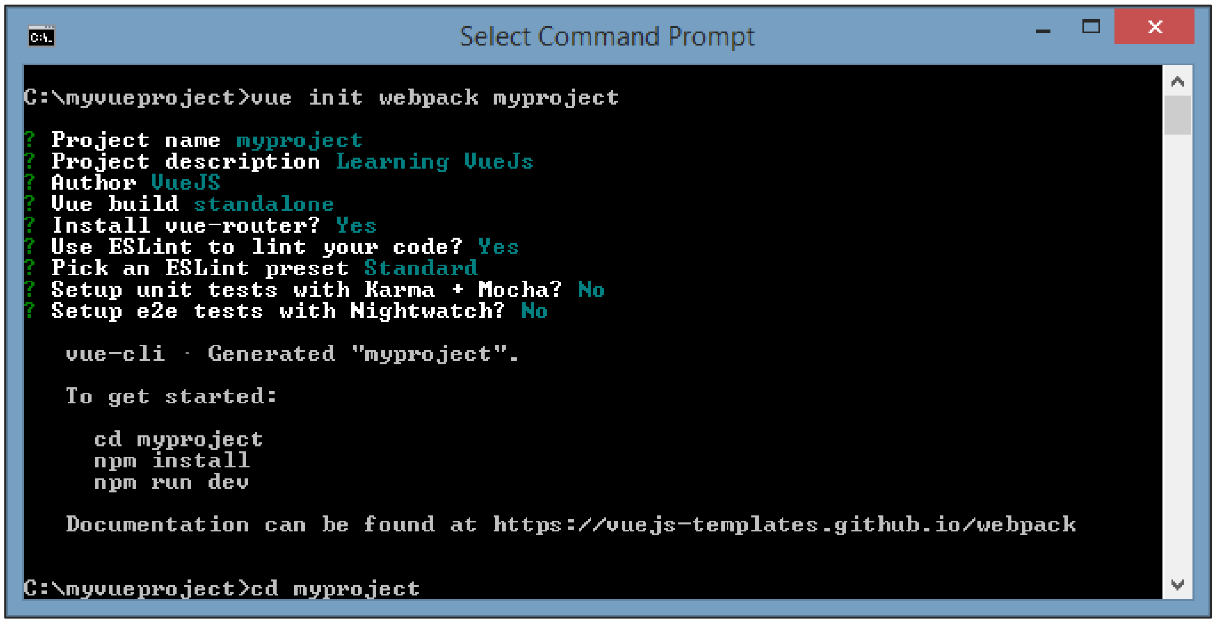
```
C:\myvueproject>npm install --global vue-cli
[█.................] | fetchMetadata: sill resolveWithNewModule filename-reserv
```

Once done, it shows the CLI version for VueJS. It takes a few minutes for the installation.

```
+ vue-cli@2.8.2

added 965 packages in 355.414s
```

Following is the command to create the project using Webpack.

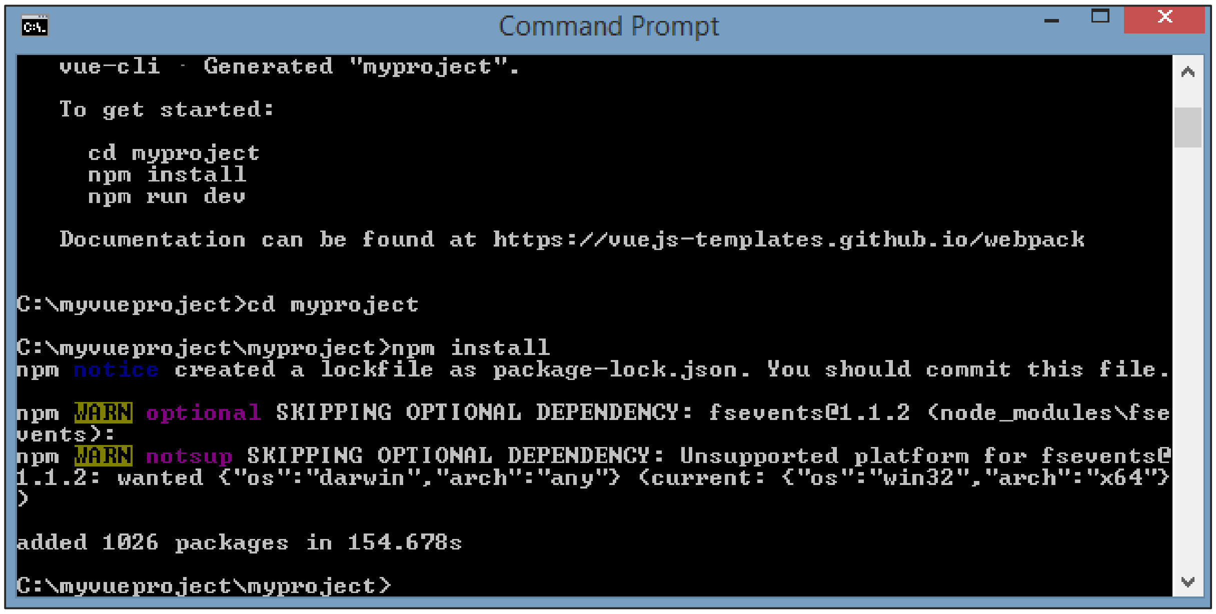
```
vue init webpack myproject
```



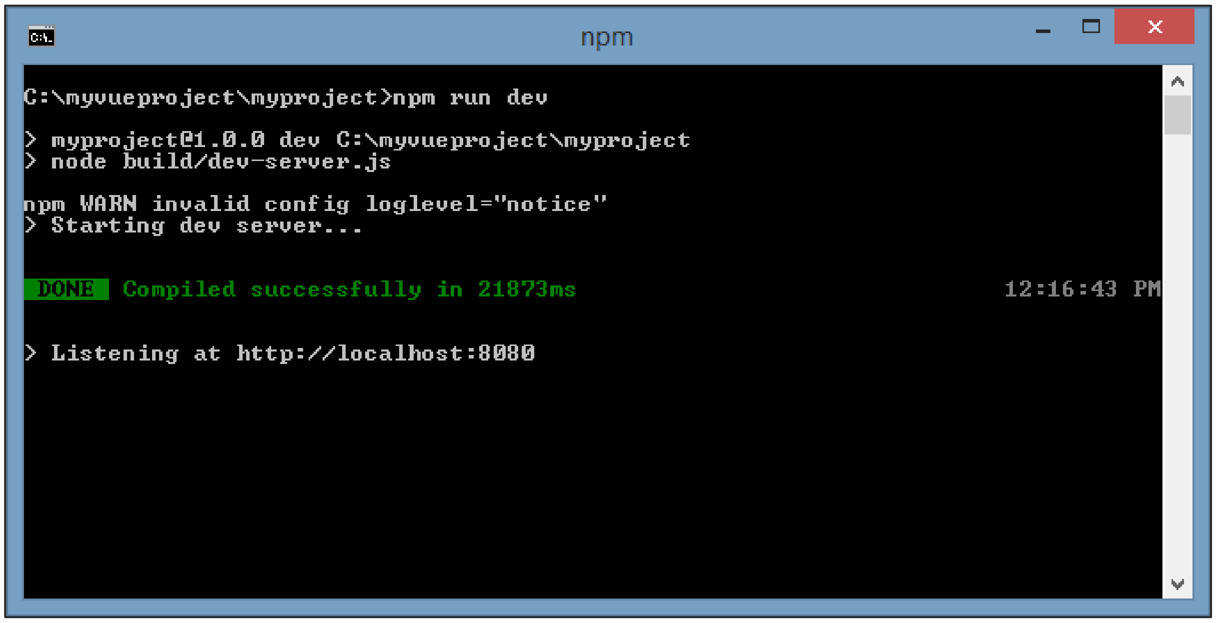
```
C:\myvueproject>vue init webpack myproject

? Project name myproject
? Project description Learning VueJs
? Author VueJS
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Setup unit tests with Karma + Mocha? No
? Setup e2e tests with Nightwatch? No

   vue-cli · Generated "myproject".

   To get started:

     cd myproject
     npm install
     npm run dev

   Documentation can be found at https://vuejs-templates.github.io/webpack


C:\myvueproject>cd myproject
```

To get started, use the following command.

```
cd myproject


npm install


npm run dev
```

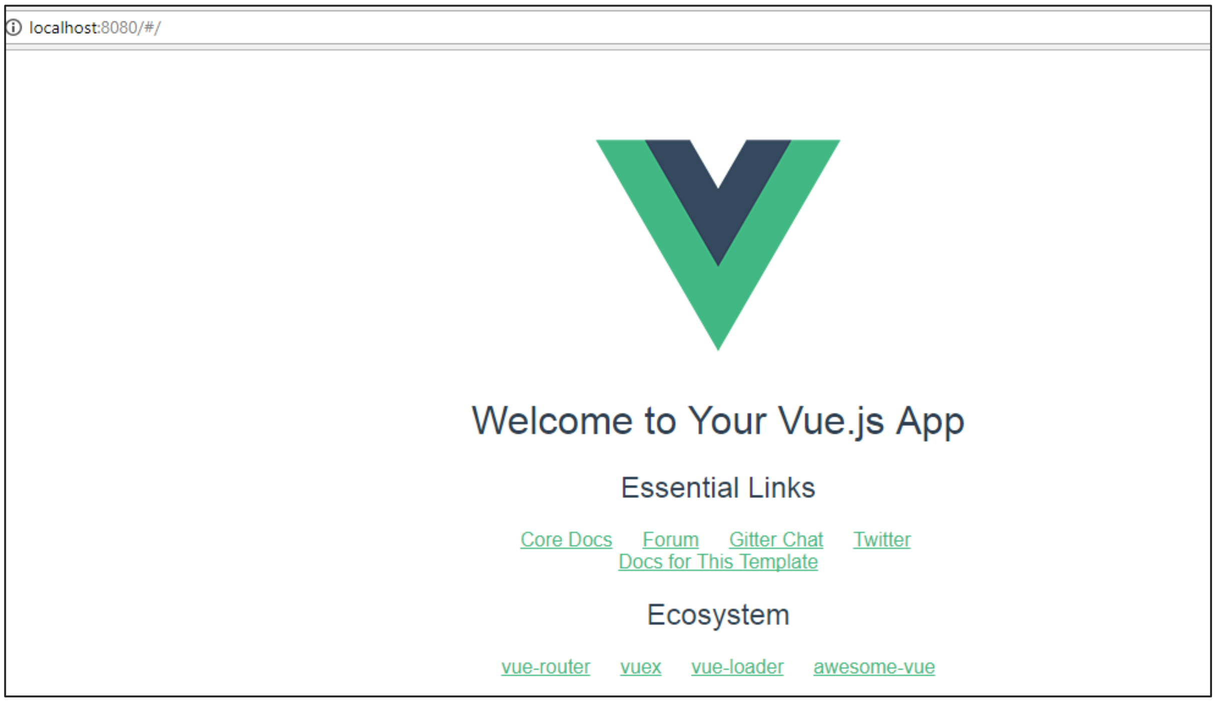


```
Command Prompt
   vue-cli · Generated "myproject".

   To get started:

     cd myproject
     npm install
     npm run dev

   Documentation can be found at https://vuejs-templates.github.io/webpack


C:\myvueproject>cd myproject

C:\myvueproject\myproject>npm install
npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.2 (node_modules\fse
vents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
1.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"}
)

added 1026 packages in 154.678s

C:\myvueproject\myproject>
```



```
npm
C:\myvueproject\myproject>npm run dev

> myproject@1.0.0 dev C:\myvueproject\myproject
> node build/dev-server.js

npm WARN invalid config loglevel="notice"
> Starting dev server...


 DONE  Compiled successfully in 21873ms                             12:16:43 PM


> Listening at http://localhost:8080
```

Once we execute npm run dev, it starts the server and provides the url for display to be seen in the browser which is as shown in the following screenshot.


localhost:8080/#/

Welcome to Your Vue.js App

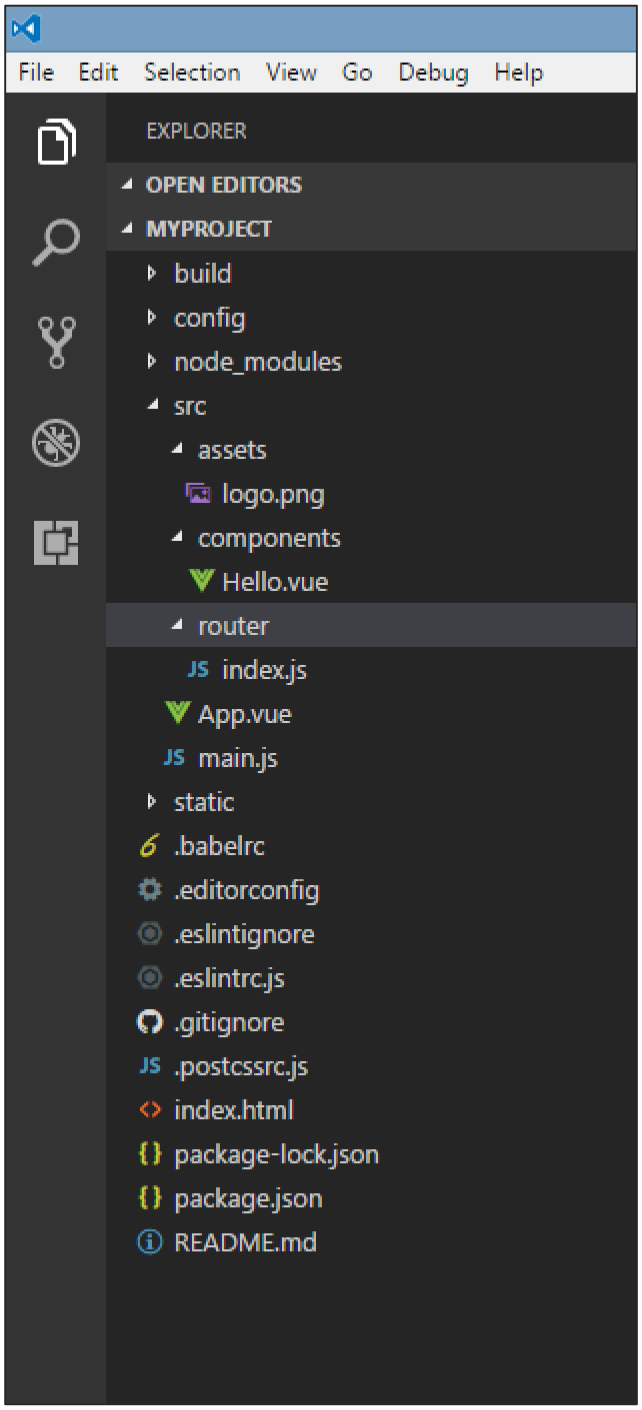
Essential Links

Core Docs Forum Gitter Chat Twitter
Docs for This Template

Ecosystem

vue-router vuex vue-loader awesome-vue

The project structure using CLI looks like the following.



```
build
config
node_modules
src
  assets
    logo.png
  components
    Hello.vue
  router
    index.js
  App.vue
  main.js
static
.babelrc
.editorconfig
.eslintignore
.eslintrc.js
.gitignore
.postcssrc.js
index.html
package-lock.json
package.json
README.md
```

3. VueJS – Introduction

Vue is a JavaScript framework for building user interfaces. Its core part is focused mainly on the view layer and it is very easy to understand. The version of Vue that we are going to use in this tutorial is 2.0.

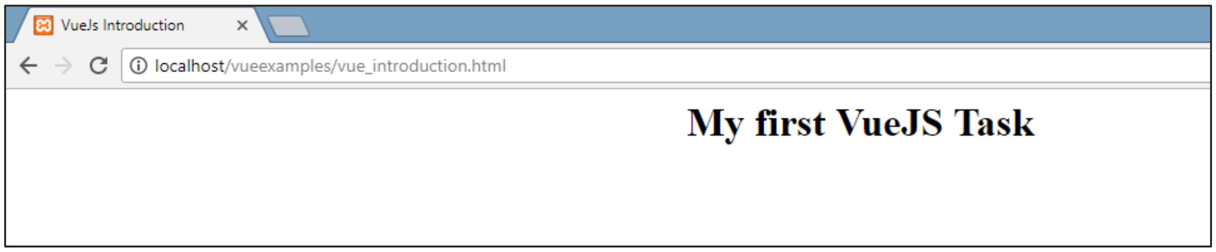
As Vue is basically built for frontend development, we are going to deal with lot of HTML, JavaScript and CSS files in the upcoming chapters. To understand the details, let us start with a simple example.

In this example, we are going to use the development verison of vuejs.

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>
```

Output



My first VueJS Task

This is the first app we have created using VueJS. As seen in the above code, we have included vue.js at the start of the .html file.

```
<script type="text/javascript" src="js/vue.js"></script>
```

There is a div which is added in the body that prints **"My first VueJS Task"** in the browser.

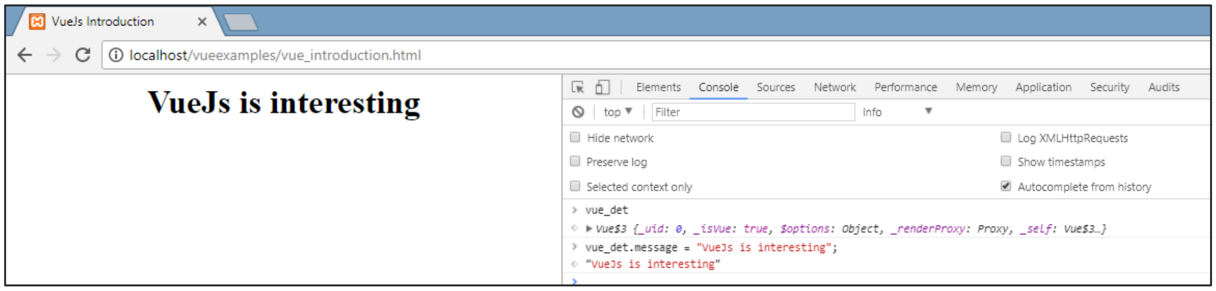
```
  <div id="intro" style="text-align:center;">
         <h1>{{ message }}</h1>
  </div>
```

We have also added a message in a interpolation, i.e. **{{}}**. This interacts with VueJS and prints the data in the browser. To get the value of the message in the DOM, we are creating an instance of vuejs as follows:

```
var vue_det = new Vue({
        el: '#intro',
        data: {
            message: 'My first VueJS Task'
        }
 })
```

In the above code snippet, we are calling Vue instance, which takes the id of the DOM element i.e. e1:'#intro', it is the id of the div. There is data with the message which is assigned the value **'My first VueJS Task'.** VueJS interacts with DOM and changes the value in the DOM {{message}} with **'My first VueJS Task'.**

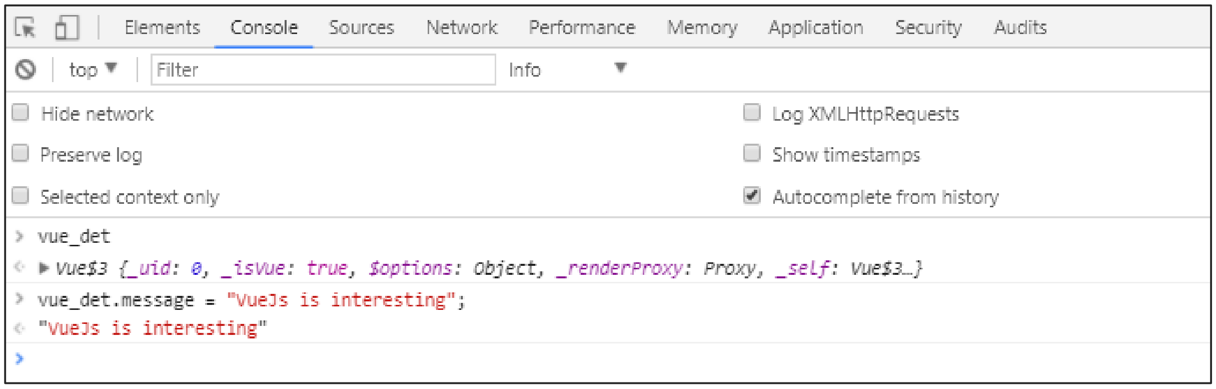
If we happen to change the value of the message in the console, the same will be reflected in the browser. For example:



VueJs is interesting

```
> vue_det
< ▶ Vue$3 {_uid: 0, _isVue: true, $options: Object, _renderProxy: Proxy, _self: Vue$3…}
> vue_det.message = "VueJs is interesting";
< "VueJs is interesting"
```

Console Details



Hide network

Preserve log

Selected context only

Log XMLHttpRequests

Show timestamps

✓ Autocomplete from history

```
> vue_det
< ▶ Vue$3 {_uid: 0, _isVue: true, $options: Object, _renderProxy: Proxy, _self: Vue$3…}
> vue_det.message = "VueJs is interesting";
< "VueJs is interesting"
>
```

In the above console, we have printed the vue_det object, which is an instance of Vue. We are updating the message with **"VueJs is interesting"** and the same is changed in the browser immediately as seen in the above screenshot.

This is just a basic example showing the linking of VueJS with DOM, and how we can manipulate it. In the next few chapters, we will learn about directives, components, conditional loops, etc.

4. VueJS – Instances

To start with VueJS, we need to create the instance of Vue, which is called the **root Vue Instance.**

Syntax

```
var app = new Vue({
  // options
})
```

Let us look at an example to understand what needs to be part of the Vue constructor.

```
<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>
```

vue_instance.js

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

For Vue, there is a parameter called **e1**. It takes the id of the DOM element. In the above example, we have the id **#vue_det**. It is the id of the div element, which is present in .html.

```
          <div id="vue_det"></div>
```

Now, whatever we are going to do will affect the div element and nothing outside it.

Next, we have defined the data object. It has value firstname, lastname, and address.

The same is assigned inside the div. For example,

```
 <div id="vue_det">
        <h1>Firstname : {{firstname}}</h1>
        <h1>Lastname : {{lastname}}</h1>
 </div>
```

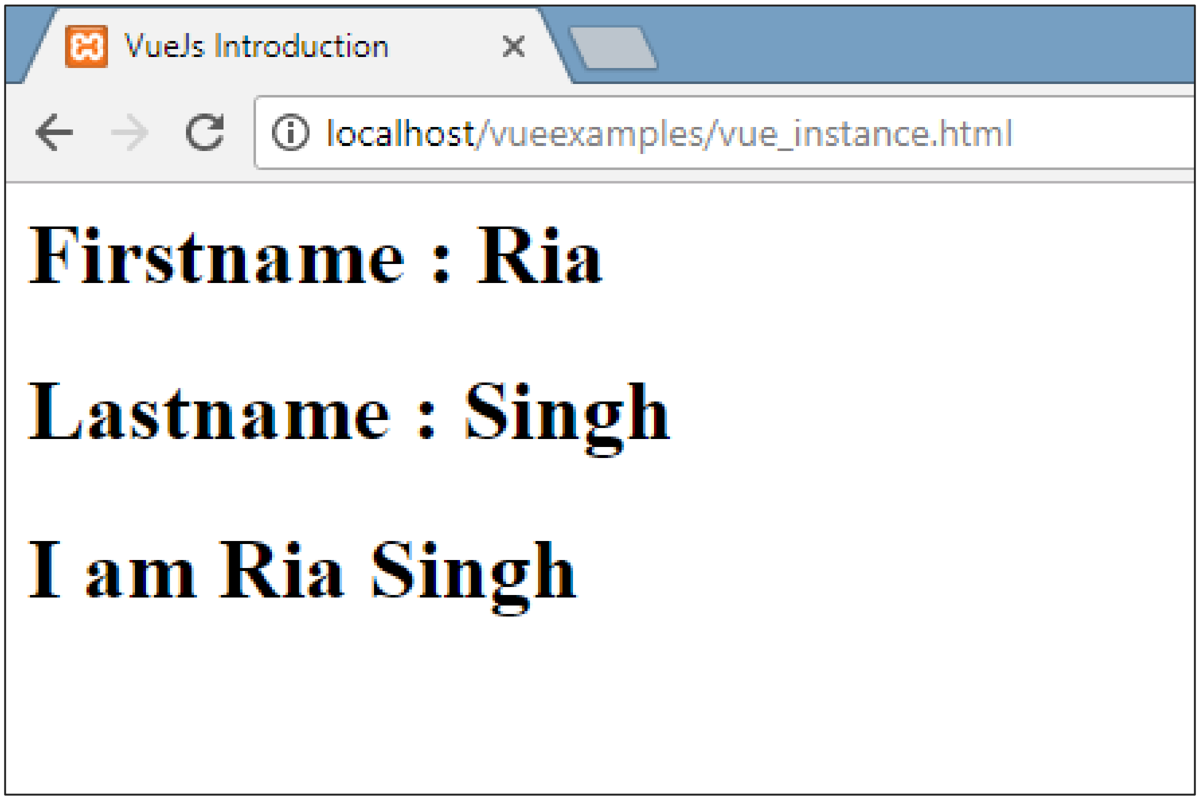
The Firstname : {{firstname}} value will be replaced inside the interpolation, i.e. {{}} with the value assigned in the data object, i.e. Ria. The same goes for last name.

Next, we have methods where we have defined a function mydetails and a returning value. It is assigned inside the div as

```
<h1>{{mydetails()}}</h1>
```

Hence, inside {{} } the function mydetails is called. The value returned in the Vue instance will be printed inside {{}}. Check the output for reference.

Output



```
Firstname : Ria

Lastname : Singh

I am Ria Singh
```

Now, we need to pass options to the Vue constructor which is mainly data, template, element to mount on, methods, callbacks, etc.

Let us take a look at the options to be passed to the Vue.

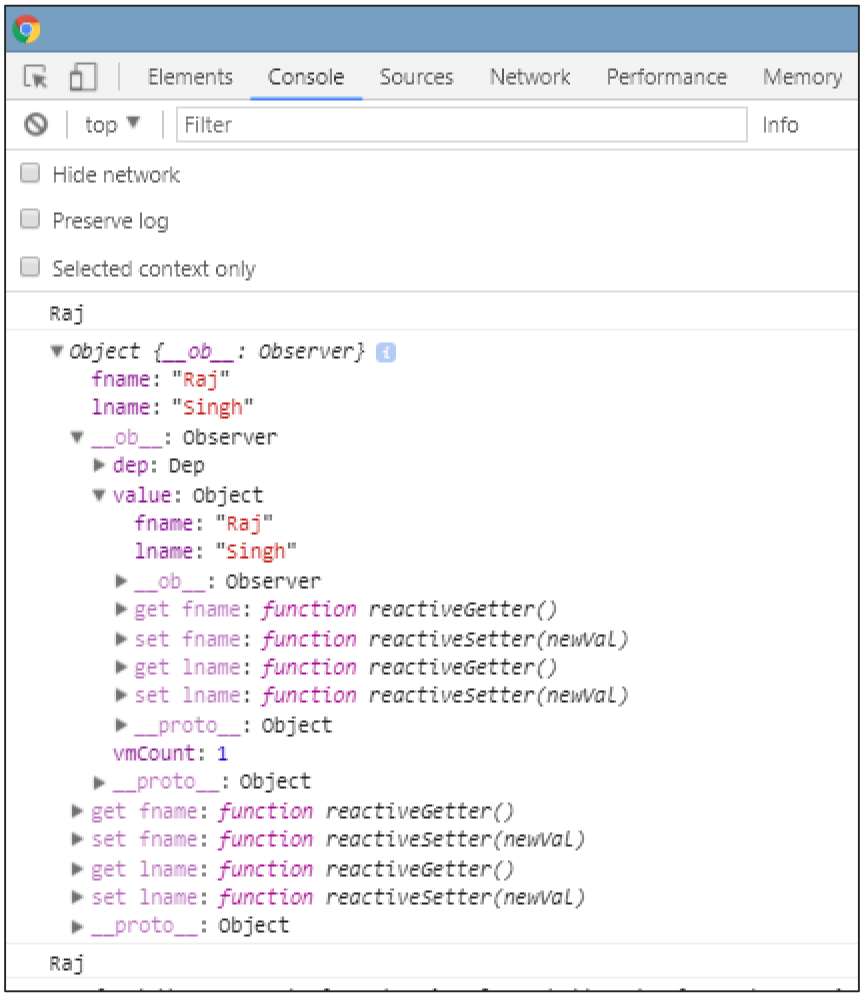
#data: This type of data can be an object or a function. Vue converts its properties to getters/setters to make it reactive.

Let's take a look at how the data is passed in the options.

Example

```
<html>
    <head>
        <title>VueJs Introduction</title>
        <script type="text/javascript" src="js/vue.js"></script>
    </head>
    <body>
        <script type="text/javascript">
            var _obj = { fname: "Raj", lname: "Singh"}
            // direct instance creation
            var vm = new Vue({
                data: _obj
            });
            console.log(vm.fname);
            console.log(vm.$data);
             console.log(vm.$data.fname);
        </script>
    </body>
</html>
```

Output



Elements Console Sources Network Performance Memory

top ▼ Filter Info

Hide network

Preserve log

Selected context only

```
Raj
▼Object {__ob__: Observer}
   fname: "Raj"
   lname: "Singh"
  ▼__ob__: Observer
   ▶dep: Dep
   ▼value: Object
      fname: "Raj"
      lname: "Singh"
     ▶__ob__: Observer
     ▶get fname: function reactiveGetter()
     ▶set fname: function reactiveSetter(newVal)
     ▶get lname: function reactiveGetter()
     ▶set lname: function reactiveSetter(newVal)
     ▶__proto__: Object
    vmCount: 1
   ▶__proto__: Object
  ▶get fname: function reactiveGetter()
  ▶set fname: function reactiveSetter(newVal)
  ▶get lname: function reactiveGetter()
  ▶set lname: function reactiveSetter(newVal)
  ▶__proto__: Object
Raj
```

console.log(vm.fname); // prints Raj

console.log(vm.$data); prints the full object as shown above

console.log(vm.$data.fname); // prints Raj

If there is a component, the data object has to be referred from a function as shown in the following code.

```
<html>
    <head>
        <title>VueJs Introduction</title>
        <script type="text/javascript" src="js/vue.js"></script>
    </head>
    <body>
        <script type="text/javascript">
            var _obj = { fname: "Raj", lname: "Singh"};
            // direct instance creation
            var vm = new Vue({
                data: _obj
            });
            console.log(vm.fname);
            console.log(vm.$data);
            console.log(vm.$data.fname);
            // must use function when in Vue.extend()
            var Component = Vue.extend({
                data: function () {
                    return _obj
                }
            });
            var myComponentInstance = new Component();
            console.log(myComponentInstance.lname);
            console.log(myComponentInstance.$data);
        </script>
    </body>
</html>
```

In case of a component, the data is a function, which is used with Vue.extend as shown above. The data is a function. For example,

```
                    data: function () {
                        return _obj
                    }
```

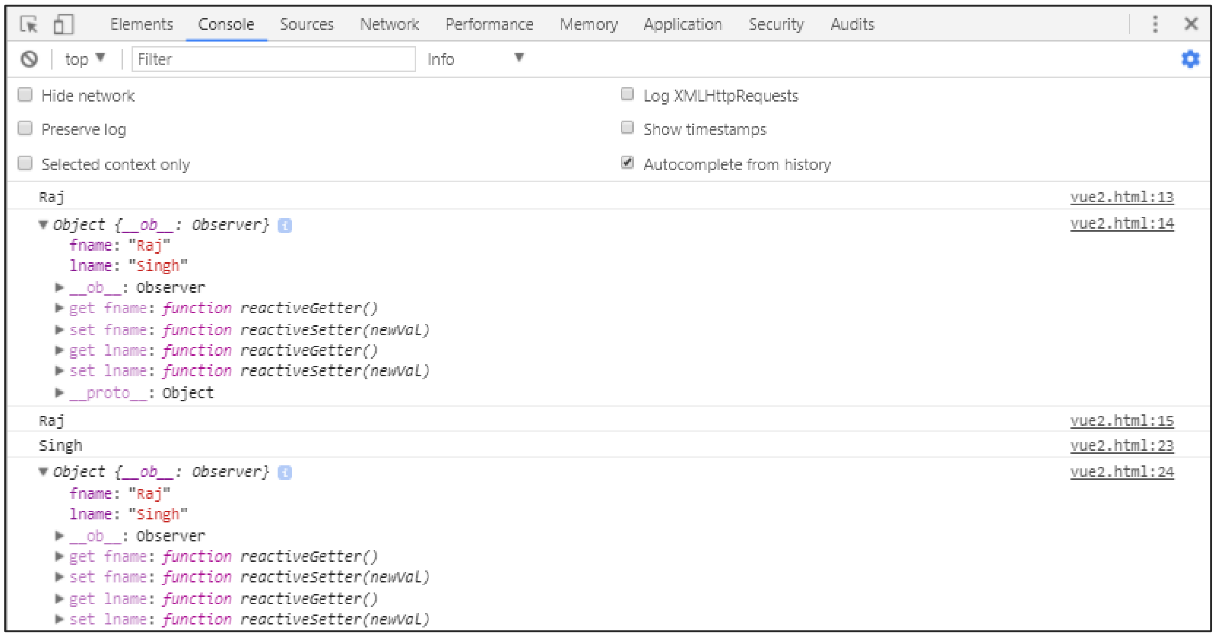
To refer to the data from the component, we need to create an instance of it. For example,

```
                    var myComponentInstance = new Component();
```

To fetch the details from the data, we need to do the same as we did with the parent component above. For example,

```
console.log(myComponentInstance.lname);
console.log(myComponentInstance.$data);
```

Following are the details displayed in the browser.



```
Raj                                                          vue2.html:13
▼Object {__ob__: Observer}                                   vue2.html:14
    fname: "Raj"
    lname: "Singh"
  ▶__ob__: Observer
  ▶get fname: function reactiveGetter()
  ▶set fname: function reactiveSetter(newVal)
  ▶get lname: function reactiveGetter()
  ▶set lname: function reactiveSetter(newVal)
  ▶__proto__: Object
Raj                                                          vue2.html:15
Singh                                                        vue2.html:23
▼Object {__ob__: Observer}                                   vue2.html:24
    fname: "Raj"
    lname: "Singh"
  ▶__ob__: Observer
  ▶get fname: function reactiveGetter()
  ▶set fname: function reactiveSetter(newVal)
  ▶get lname: function reactiveGetter()
  ▶set lname: function reactiveSetter(newVal)
```

Props: Type for props is an array of string or object. It takes an array-based or object-based syntax. They are said to be attributes used to accept data from the parent component.

Example 1

```
Vue.component('props-demo-simple', {
  props: ['size', 'myMessage']
})
```

Example 2

```
Vue.component('props-demo-advanced', {
  props: {
    // just type check
    height: Number,
    // type check plus other validations
    age: {
      type: Number,
      default: 0,
      required: true,
      validator: function (value) {
        return value >= 0
      }
    }
  }
})
```

propsData: This is used for unit testing.

Type: array of string. For example, { [key: string]: any }. It needs to be passed during the creation of Vue instance.

Example

```
var Comp = Vue.extend({
  props: ['msg'],
  template: '<div>{{ msg }}</div>'
})
var vm = new Comp({
  propsData: {
    msg: 'hello'
  }
})
```

Computed: Type: { [key: string]: Function | { get: Function, set: Function } }

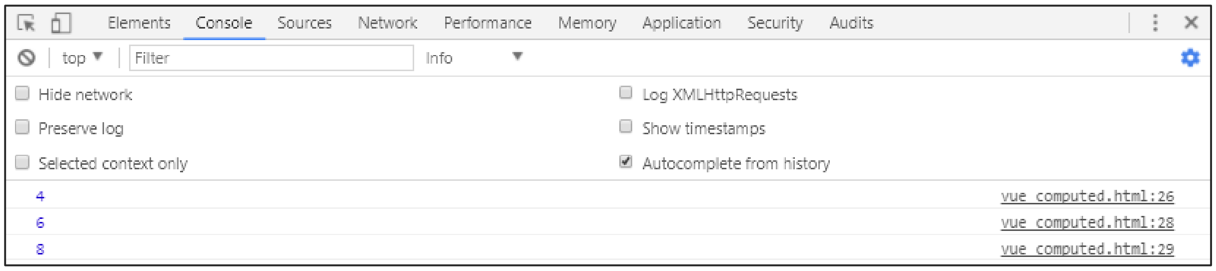
Example

```
<html>
    <head>
        <title>VueJs Introduction</title>
        <script type="text/javascript" src="js/vue.js"></script>
    </head>
    <body>
        <script type="text/javascript">
           var vm = new Vue({
                data: { a: 2 },
                computed: {
                    // get only, just need a function
                    aSum: function () {
                        return this.a + 2;
                    },
                    // both get and set
                    aSquare: {
                        get: function () {
                            return this.a*this.a;
                        },
                        set: function (v) {
                            this.a = v*2;
                        }
                    }
                }
            })
            console.log(vm.aSquare);  // -> 4
            vm.aSquare = 3;
            console.log(vm.a);       // -> 6
            console.log(vm.aSum); // -> 8
        </script>
    </body>
</html>
```

Computed has two functions **aSum** and **aSquare**.

Function aSum just returns **this.a+2**. Function aSquare again two functions **get** and **set**.

Variable vm is an instance of Vue and it calls aSquare and aSum. Also vm.aSquare = 3 calls the set function from aSquare and vm.aSquare calls the get function. We can check the output in the browser which looks like the following screenshot.

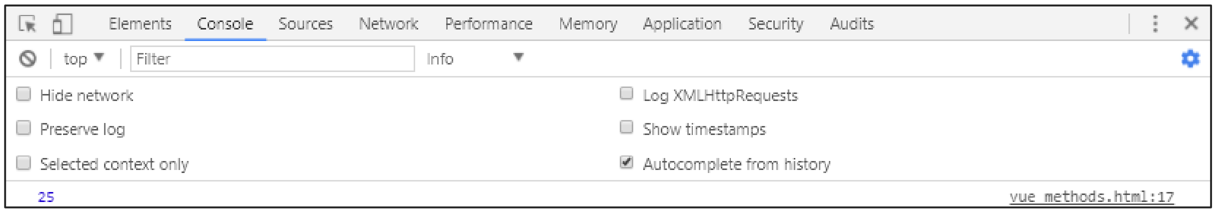


```
4                                                    vue_computed.html:26
6                                                    vue_computed.html:28
8                                                    vue_computed.html:29
```

Methods: Methods are to be included with the Vue instance as shown in the following code. We can access the function using the Vue object.

```
<html>
    <head>
        <title>VueJs Introduction</title>
        <script type="text/javascript" src="js/vue.js"></script>
    </head>
    <body>
        <script type="text/javascript">
            var vm = new Vue({
                data: { a: 5 },
                methods: {
                    asquare: function () {
                        this.a *= this.a;
                    }
                }
            })
            vm.asquare();
            console.log(vm.a); // 25
        </script>
    </body>
</html>
```

Methods are part of the Vue constructor. Let us make a call to the method using the Vue object **vm.asquare ()**, the value of the property **a** is updated in the **asquare** function. The value of a is changed from 1 to 25, and the same is seen reflected in the following browser console.



```
25                                                                vue_methods.html:17
```