VueJS - компоненти

Компоненти Vue є однією з важливих функцій VueJS, яка створює власні елементи, які можна повторно використовувати в HTML.

Давайте працювати з прикладом і створити компонент, який дасть краще розуміння того, як компоненти працюють з VueJS.

Приклад

```
<html>
   <head>
      <title>VueJs Instance</title>
      <script type = "text/javascript" src = "js/vue.js"></script>
   </head>
   <body>
      <div id = "component_test">
         <testcomponent></testcomponent>
      </div>
      <div id = "component_test1">
         <testcomponent></testcomponent>
      </div>
      <script type = "text/javascript" src = "js/vue_component.js"></script>
   </body>
</html>
```

vue_component.js

```
Vue.component('testcomponent',{
   template : '<div><h1>This is coming from component</h1></div>'
});
var vm = new Vue({
   el: '#component_test'
});
var vm1 = new Vue({
   el: '#component_test1'
});
```

У файлі .html ми створили дві розділи з id **component_test** і **component_test1** . У файлах **.js,** показаних вище, створюються два екземпляри Vue з ідентифікаторами div. Ми створили загальний компонент для використання як з екземплярами перегляду.

Для створення компонента наступним є синтаксис.

```
Vue.component('nameofthecomponent',{ // options});
```

Після створення компонента ім'я компонента стає спеціальним елементом, і те ж саме можна використовувати в створеному елементі екземпляра Vue, тобто в розділі div з ids **component_test** і **component_test1** .

У файлі **.js** ми використовували тестовий компонент як ім'я компонента, а таке ж ім'я використовується як спеціальний елемент усередині розділу divs.

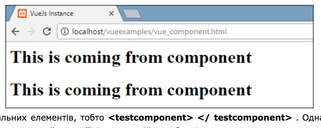
Приклад

```
<div id = "component_test">
   <testcomponent></testcomponent>
</div>
<div id = "component_test1">
   <testcomponent></testcomponent>
</div>
```

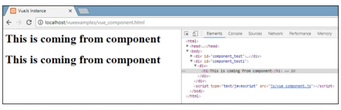
У компоненті, створеному у файлі **.js** , ми додали шаблон, до якого ми призначили код HTML. Це спосіб **реєстрації глобального компонента** , який може бути частиною будь-якої інсталяції vue, як показано в наступному сценарії.

```
Vue.component('testcomponent',{
   template : '<div><h1>This is coming from component</h1></div>'
});
```

Після виконання, це буде відображено і в браузері.



Для компонентів надається тег спеціальних елементів, тобто **<testcomponent> </ testcomponent>** . Однак, коли ми перевіряємо те ж саме в браузері, ми не помітимо спеціальний тег у звичайному HTML, присутній у шаблоні, як показано на наступному скріншоті.



Ми також безпосередньо зробили компоненти частиною екземпляру vue, як показано в наступному сценарії.

```
var vm = new Vue({
   el: '#component_test',
   components:{
      'testcomponent': {
         template : '<div><h1>This is coming from component</h1></div>'
      }
   }
});
```

Це називається **локальною реєстрацією,** і компоненти будуть частиною лише створеної instance vue.

Поки що ми бачили основний компонент із основними параметрами. Тепер давайте додамо деякі інші параметри, такі як дані та методи до нього. Так само, як приміряють Vue має дані та методи, компонент також поділяє те ж саме. Отже, ми розширимо код, який ми вже бачили за допомогою даних та методів.

Приклад

```
<html>
   <head>
      <title>VueJs Instance</title>
      <script type = "text/javascript" src = "js/vue.js"></script>
   </head>
   <body>
      <div id = "component_test">
         <testcomponent></testcomponent>
      </div>
      <div id = "component_test1">
         <testcomponent></testcomponent>
      </div>
      <script type = "text/javascript" src = "js/vue_component.js"></script>
   </body>
</html>
```

vue_component.js

```
Vue.component('testcomponent',{
   template : '<div v-on:mouseover = "changename()" v-on:mouseout = "originalname();"><h1>Custom Component created by <span id = "name">{{name}}</span></h1></div>',
   data: function() {
      return {
         name : "Ria"
      }
   },
   methods:{
      changename : function() {
         this.name = "Ben";
      },
      originalname: function() {
         this.name = "Ria";
      }
   }
});
var vm = new Vue({
   el: '#component_test'
});
var vm1 = new Vue({
   el: '#component_test1'
});
```

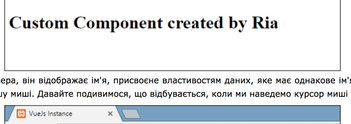
У файлі **.js** вище ми додали дані, тобто функцію, яка повертає об'єкт. Об'єкт має властивість імені, яка призначається значенням 'Ria'. Це використовується у наступному шаблоні.

```
template : '<div v-on:mouseover = "changename()" v-on:mouseout = "originalname();"><h1>Custom Component created by <span id = "name">{{name}}</span></h1></div>',
```

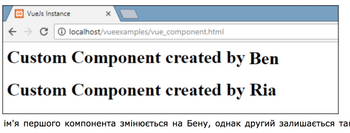
Незважаючи на наявність даних у функції компонентів, ми можемо використовувати його властивості так само, як ми використовуємо з прямим екземпляром Vue. Крім того, додається два способи, ім'я змін та оригінальне ім'я. У змінному імені ми змінюємо властивість імені, і в оригінальній назві ми скидаємо його назад до початкового імені.

Ми також додали два події на div, overlay і mouseout. Деталі подій будуть розглянуті в розділі "Події". Таким чином, для наведення курсору миші використовується метод зміни змінних і виклики клавіш mouseout з методом **originalname** .

Відображення такого ж виглядає в наступному браузері.



Як видно з наведеного вище браузера, він відображає ім'я, присвоєне властивостям даних, яке має однакове ім'я. Ми також призначили подію на клавіатурі на диску, а також клавішу миші. Давайте подивимося, що відбувається, коли ми наведемо курсор миші та клацніть мишею



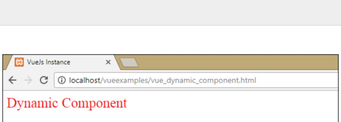
Наведення курсора ми бачимо, що ім'я першого компонента змінюється на Бену, однак другий залишається так, як і є. Це тому, що компонент даних є функцією, і він повертає об'єкт. Таким чином, коли він змінюється в одному місці, той же не перезаписується в інших випадках.

Динамічні компоненти створюються за допомогою ключового слова **<component> </ component>,** і він пов'язаний ізДинамічні компоненти використанням властивості, як показано в наступному прикладі.

Приклад

```
<html>
   <head>
      <title>VueJs Instance</title>
      <script type = "text/javascript" src = "js/vue.js"></script>
   </head>
   <body>
      <div id = "databinding">
         <component v-bind:is = "view"></component>
      </div>
      <script type = "text/javascript">
         var vm = new Vue({
            el: '#databinding',
            data: {
               view: 'component1'
            },
            components: {
               'component1': {
                  template: '<div><span style = "font-size:25;color:red;">Dynamic Component</span></div>'
               }
            }
         });
      </script>
   </body>
</html>
```

Вихідні дані



Динамічний компонент створюється за допомогою наступного синтаксису.

```
<component v-bind:is = "view"></component>
```

Він має v-bind: is = "view", і йому присвоєно значення перегляду. Вигляд визначається в екземплярі Vue наступним чином.

```
var vm = new Vue({
   el: '#databinding',
   data: {
      view: 'component1'
   },
   components: {
      'component1': {
         template: '<div><span style = "font-size:25;color:red;">Dynamic Component</span></div>'
      }
   }
});
```

При виконанні шаблон **Dynamic Component** відображається у браузері.