Components

consider the following application: (no component)

```
<div id="app">

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

```
new Vue({
  el: '#app',
  data: {
    status: 'Critical'
  },
  template: '<p>Server Status: {{ status }}</p>'
})
```

With one Vue instance, we can only select the first DOM element as a placeholder.

If we want to use the instance in multiple place, we need to use **component**, a reusable pieces of Vue instance

Create a component and move 'data' property into the new 'my-component' component. using function to return 'data' to make sure that it does not interfere with the *real* data property in Vue instance.

```
Vue.component('my-component', {
  data: function() {
      status: 'Critical'
  },
```

```
  template: '<p>Server Status: {{ status }}</p>
              (<button @click="changeStatus"> Change
</button>)',
  methods: {
    changeStatus: function() {
      this.status = 'Normal';
    }
  }
}

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

Now update the HTML template by adding a placeholder for the new 'my-component'. We can have **multiple placeholders** but each of them will be tied to different **component objects**

```
<div id="app">
  <my-component></my-component>
  <hr>
  <my-component></my-component>
</div>
```

Registering Components

What if we want to have a 2nd app:

```
<div id="app">
  <my-component></my-component>
  <hr>
  <my-component></my-component>
</div>
```

```
<div id="app2">
  <my-cmp2></my-cmp2>
</div>
```

and create a 2nd Vue instance below the first one:

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

new Vue({
  el: '#app2'
})
```

The 2nd will use the same component because **Vue.component(...)** register a component **"globally"**.

We can store the component in a **variable** and add the **components** property in the Vue instance to tell which components are **registered locally**.

```
var cmp = {
  data: function() {
      return {
        status: 'Critical'
      }
  },
  template: ```<p>Server Status: {{ status }}</p>
              (<button @click="changeStatus"> Change
</button>```,
  methods: {
    changeStatus: function() {
      this.status = 'Normal';
```

```
    }
  }
}

new Vue({
  el: '#app',
  components: {
    'my-component': cmp
  }
})

new Vue({
        el: '#app2',
  components: {
        'my-cmp2': cmp
  }
})
```

Root component

Previously we have *main.js* and *App.vue* files as following:

```
\\ main.js file
import Vue from 'vue'
import App from './App.vue'

new Vue( {
  el: '#app',
  render: h => h(App)   // ES6 arrow function
 } )
```

The **render** property is an alternative to **template** but better. It's not restricted to 'string' template but also compiled version.

```
// App.vue file - Single File Template

// HTML template, can have vue interpolation and ...
<template>
  <h1> Hello World! <h1>
</template>

// behave like a Vue instance : new Vue( { } )
<script>
  export default {
  }
</script>

// CSS style
<style>

</style>
```

The **App.vue** is the ROOT component of the application.

Currently there is no component in the application, just a normal object. The **export default {...}** is considered as a normal object. It can store **data** only as a funciton.

```
<template>
  <h1> Hello World! <h1>
  <h2> Server Status: {{ status }} </h2>
</template>

// behave like a Vue instance : new Vue( { } )
<script>
  export default {
    data: function() {
      return {
        status: 'Critical'
```

```
      }
    }
  }
</script>
```

Create a new Component

We can create a new component in the 'src' folder. 'Home.vue' is created with the following code:

```
// Home.vue file
<template>
  <div>
    <p>Server Status: {{ status }}</p>
    <hr>
    <button @click="changeStatus2">Change</button>
  <div>
</template>

<script>
  export default {
    data: function() {
      return {
        status: 'Critical'
      }
    },
    methods: {
      changeStatus1 = function() {  // ES5 style

      },
      changeStatus2() {             // ES6 style
        this.status = 'Normal'
      }
    }
  }
</script>
```

Note that in the HTML template, there MUST be only one ROOT ELEMENT (no sibling). Therefore we need to wrap all DOM within single **'div'** tag.

To use the new **Home** object as a compoment, we can register it globally in the 'main.js' file.

```
\\ main.js file
import Vue from 'vue'
import App from './App.vue'
import Home from './Home.vue'

Vue.component('app-server-status', Home);

new Vue( {
  el: '#app',
  render: h => h(App)   // ES6 arrow function
 } )
```

Now we have to change the ROOT component, **App.vue**, by adding the **'app-server-status'** as a placeholder for the **Home** component.

```
// App.vue file

<template>
  <h1> Hello World! <h1>
  <app-server-status></app-server-status>
</template>

<script>

</script>
```

Using component locally

Supposed that we have multiple servers that we want to monitor their status. We can create a **ServerStatus** component and use it in the previously created **Home** component. We create the **ServerStatus** using the code from **Home**:

```
// ServerStatus.vue file
<template>
  <div>
    <p>Server Status: {{ status }}</p>
    <hr>
    <button @click="changeStatus2">Change</button>
  <div>
</template>

<script>
  export default {
    data: function() {
      return {
        status: 'Critical'
      }
    },
    methods: {
      changeStatus1 = function() {  // ES5 style

      },
      changeStatus2() {             // ES6 style
        this.status = 'Normal'
      }
    }
  }
</script>
```

Fisrt we need to change the placeholder tag for **Home** in the ROOT component from 'app-server-status' to 'app-servers'. We will now use the 'app-server-status' as a placeholder for the **ServerStatus** instead. Therefore we need to update 'main.js' as followed:

```
\\ main.js file
import Vue from 'vue'
import App from './App.vue'
import Home from './Home.vue'

Vue.component('app-servers', Home);

new Vue( {
  el: '#app',
  render: h => h(App)   // ES6 arrow function
 } )
```

```
// App.vue file

<template>
  <h1> Hello World! <h1>
  <app-servers></app-servers>
</template>

<script>

</script>
```

In the **Home** component, we need to locally register **ServerStatus**. We need to modify the **Home** as followed:

```
<template>
  <div>
    <app-server-status v-for="server in 5">
    </app-server-status>
  </div>
</template>
```

```
<script>
  import ServerStatus from './ServerStatus.vue';

  export default {
    components: {
      'app-server-status': ServerStatus
    }
  }
</script>
```

Project Folder Structure

By default all '.vue' files are stored in **src** folder. We can create subfolders to separatedly store these components grouped by *features* or *duty*.:

```
- main.js
- users/
-- account/
-- analytics/
- shop
-- main
-- checkout/
```

or

```
- main.js
- components/
-- shared/
--- Header.vue
--- Footer.vue
-- server/
--- Servers.vue
--- ServerDetail.vue
```

How to name your tag

We can use case-sensitive since javascript is case-sensitive and the template is compiled before deploy.

```
<appHeader></appHeader>
<app-header></app-header>   // this one is ok as well

<script>
  import Header from './Header.vue'

  export default {
    components: {
      appHeader: Header
    }
  }
</script>
```

but it is recommend tag with dash '-' becase it is resemble to HTML tag.

Scoping your component

By default any style defined in any component is applied globally. We can 'contain' the style to its component by adding **'scoped'** as attribute in **style** tag:

```
<style scoped>
  div {
    border: 1px solid red;
  }
</style>
```