Template Syntax

Vue.js template syntax allows you to bind the rendered DOM Vue instance's data.

Text

The most basic form of data binding is text interpolation using the "Mustache" syntax (double curly braces):
Example:

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

The mustache tag will be replaced with the value of the msg property on the corresponding data object. It will also be updated whenever the data object's msg property changes. Please check the following example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>VueJS</title>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</head>
<body>
<div id="app">
    <span>Message: {{ msg }}</span>
</div>
</body>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            msg: 'Hello Vue!'
        }
    })
</script>
</html>
```

Save the previous snippet in index.html file and open the file in your favourite browser. You will see something like following

```
Message: Hello Vue!
```

Open your browser console and write

```
app.msg="hello world"
```

The DOM will be updated automatically.

Lets say you want some DOM not to be updates. Do something like following

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

So, now if you do change the value form the console the DOM will not be updated.

The double mustaches interprets the data as plain text, not HTML. If you vueJs to interpret some html code, you have to use v-html directive.

```
<div id="app">
    <p>Using mustaches: {{ rawHtml }}</p>
    <p>Using v-html directive: <span v-html="rawHtml"></span></p>
</div>
......
 var app = new Vue({
        el: '#app',
        data: {
            rawHtml: '<span style="color: red">This should be red.</span>'
        }
 });
```

The contents of the span will be replaced with the value of the rawHtml property, interpreted as plain HTML - data bindings are ignored.

####Attributes
Mustaches cannot be used inside HTML attributes. Instead, use a v-bind directive:

```
<div id="app">
    <div v-bind:id="dynamicId" v-bind:title="dynamicId">Check my id and
title</div>
</div>
.......
<script>
    var app = new Vue({
        el: '#app',
        data: {
            dynamicId: 132232
        }
    })
</script>
```

In the case of boolean attributes, where their mere existence implies true, v-bind works a little differently. In this example:

```
<div id="app">
    <button v-bind:disabled="isButtonDisabled">Button</button>
</div>
......
<script>
    var app = new Vue({
        el: '#app',
        data: {
            isButtonDisabled: true
        }
    })
</script>
```

If isButtonDisabled has the value of null, undefined, or false, the disabled the button will not be rendered as disabled.

Using JavaScript Expressions

Vue.js actually supports the full power of JavaScript expressions inside all data bindings:

```
<div id="app">
    <p>{{ number + 1 }}</p>
    <p>{{ ok ? 'YES' : 'NO' }}</p>
    <p>{{ message.split('').reverse().join('') }}</p>
</div>
......
<script>
    var app = new Vue({
        el: '#app',
        data: {
            number: 10,
            ok:false,
            message:'hello world'
        }
    })
</script>
```

These expressions will be evaluated as JavaScript. One restriction is that each binding can only contain one single expression, so the following will NOT work:

```
<!-- this is a statement, not an expression: -->
{{ var a = 1 }}

<!-- flow control won't work either, use ternary expressions -->
{{ if (ok) { return message } }}
```

Directives

So far we have been using directives. For example: v-if, v-bind etc.

Directives are special attributes with the v- prefix.
Directive attribute values are expected to be a single JavaScript expression (with the exception of v-for)

Main purpose of directives is reactively manipulate the DOM when the value of it's expression changes.

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

Here, the v-if directive would remove/insert the <p> element based the value of the expression seen. Please follow the following snippet

```
<div id="app">
    <p v-if="seen">Now you see me</p>
</div>
......
<script>
    var app = new Vue({
        el: '#app',
        data: {
            seen: false
        }
    })
</script>
```

####Arguments
Some directives can take an “argument”, denoted by a colon after the directive name. For example, the v-bind directive is used to reactively update an HTML attribute: