Full Stack Web Development

WS2019 - BSA5 Ausgewählte Kapitel

Alija Sabic
sabic@technikum-wien.at

Smart Homes and Assistive Technologies October 20, 2019

Outline

1. Vue

Section

Vue - JavaScript Frameworks

- AngularJS
- Angular
- React
- Vue.js
- Ember.js
- …

Alija Sabic

Web Development

Vue - Evan You [1, 2]

- Born and raised in Wuxi, PRC (无锡中国).
- Recruited by Google for his work, inspired by Chrome experiments, and joined the Five program during his study in the U.S.
- Started the work on Vue.js as a personal project during his work at Google Creative Labs in 2013.
- Works full-time on Vue since 2016.

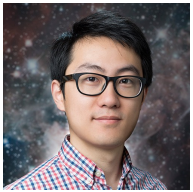

Figure 1: Evan You (尤雨溪)

Alija Sabic

Vue.js

- Client-side JavaScript framework
- Progressive framework
- Performant
- Lightweight (~33.30kB - `min+gzip`)
- Extensible
  - Vue Router
  - Vuex
  - Vue Server Renderer

Figure 2: Vue.js Logo [3]

Alija Sabic
Web Development

Vue.js - Features

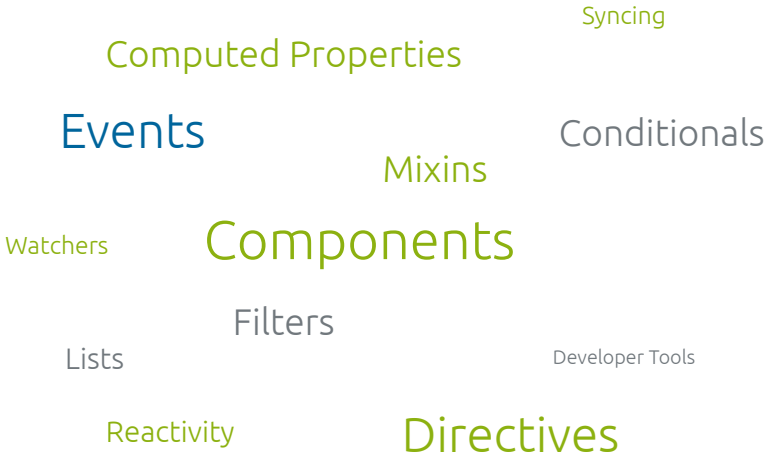

Figure 3: Vue.js Features

Alija Sabic
Web Development

Vue - Hello Vue

Vue applications can be created in different ways. To get started, we will employ a simpler setup and add the Vue script in the <head> section, using a <script> element.

```
<script src="https://vuejs.org/js/vue.js"></script>
```

Listing 1: Adding Vue to your page.

Alija Sabic
Web Development

Vue - Hello Vue

With the addition of this script, we can now access the class Vue in our scripts.

To create an application, we have to instantiate a object of this class. We have to pass a JSON object to the class constructor to configure the application.

```
<script>
  new Vue({
    el: "#app",
    data: {
      text: "Hello Vue!"
    }
  });
</script>
```

Listing 2: Creating a Vue application.

Vue uses the passed JSON object to setup the application.

Alija Sabic
Web Development

Vue - Hello Vue

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Vue Setup</title>
    <script src="https://vuejs.org/js/vue.js"></script>
  </head>
  <body>
    <div id="app">{{ text }}</div>
  </body>
  <script>
    new Vue({
      el: "#app",
      data: {
        text: "Hello Vue!"
      }
    });
  </script>
</html>
```

Listing 3: Minimal Example

Alija Sabic
Web Development

Vue - Hello Vue

We use the `el` property to specify what element should be used as the root element of our application. Vue interprets this parameter as a CSS selector to find the specified element(s) in the DOM of our page.

Vue then use the specified element(s) as a *template* to create a virtual DOM, which is then used internally to create a new (static) DOM. Subsequently, Vue replaces the specified element(s) with the newly created DOM tree.

We can use other properties in the JSON object to customize our application.

In this example, we use the `data` property to specify the data of our application. We can then access the data in the (virtual) DOM (*template syntax*).

We will explore the `data` and other properties in more detail, in the following sections.

Alija Sabic
Web Development

Vue - Hello Vue

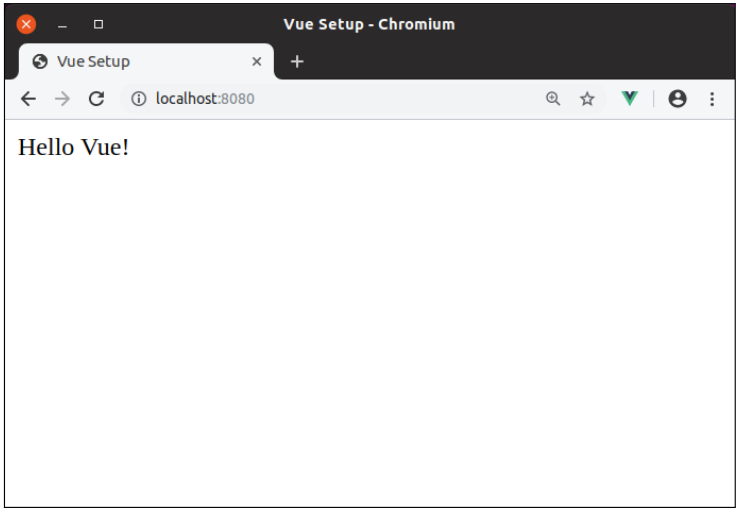

Figure 4: Minimal Example

Alija Sabic
Web Development

Vue - Hello Vue

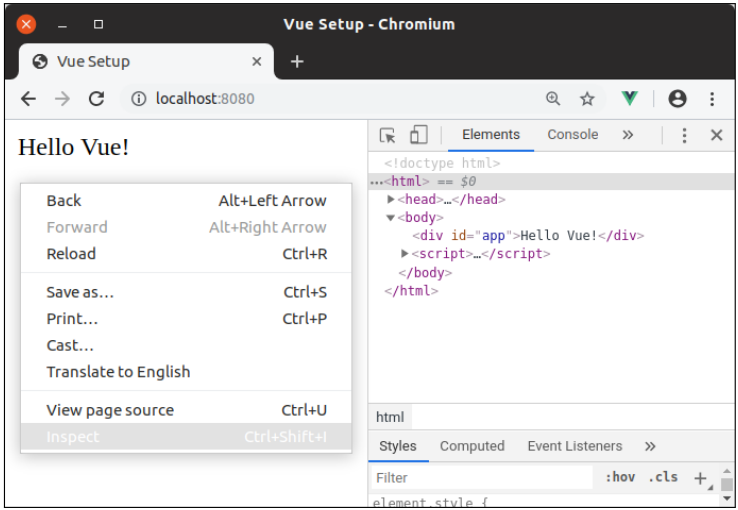

Figure 5: Chromium Developer Tools

Alija Sabic
Web Development

Vue - Browser Devtools

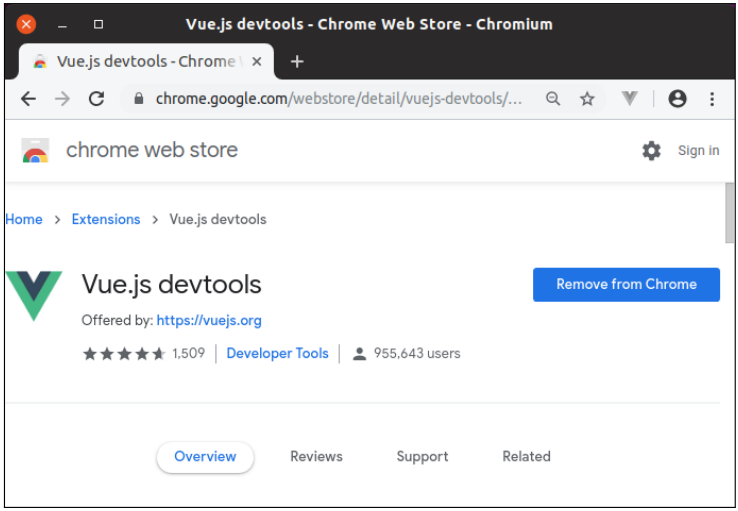

Figure 6: Chrome Vue.js Devtools

Alija Sabic
Web Development

Vue - Browser Devtools

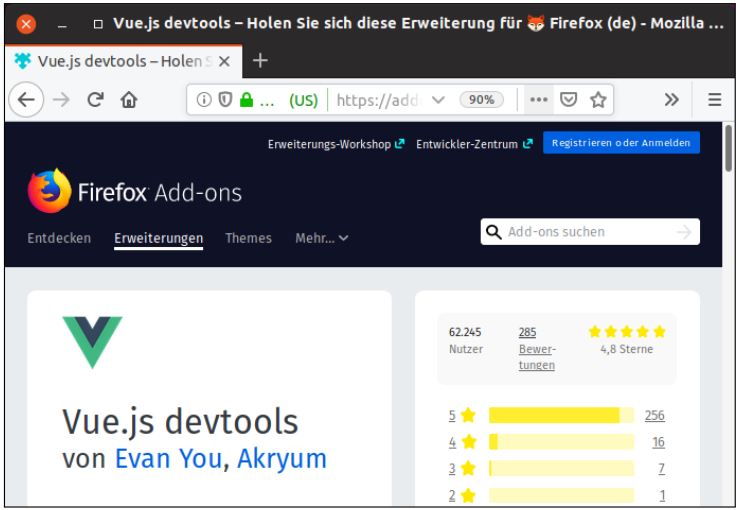

Figure 7: Firefox Vue.js Devtools

Alija Sabic
Web Development

Vue - Browser Devtools

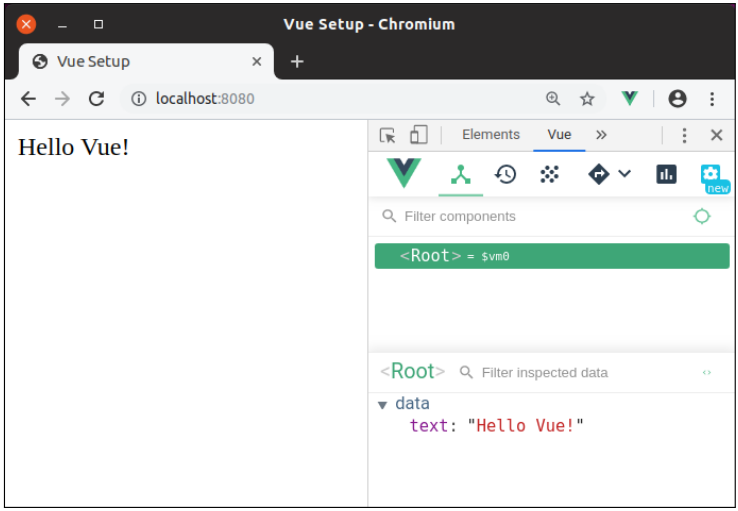

Figure 8: Chrome Vue.js Devtools - Tab

Alija Sabic
Web Development

Vue - HTTP Server

To test our application, we can install a simple HTTP server which hosts our files.

Run the following command(s) to install the npm package `http-server` on your machine.

```
$~ node -v
v10.16.3
$~ npm -v
6.9.0
$~ npm install -g http-server
+ http-server@0.11.1
added 26 packages from 28 contributors in 6.096s
```

Listing 4: Install npm module `http-server`

Alija Sabic
Web Development

Vue - HTTP Server

To host the files inside our folder `project`, inside our *home directory*, we have to run following command(s).

```
$~ ls
Desktop   Downloads Pictures Public    Videos
Documents Music     project  Templates
$~ http-server project/
Starting up http-server, serving project/
Available on:
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
$~/project http-server .
Starting up http-server, serving .
Available on:
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
```

Listing 5: Host directory with `http-server`

Alija Sabic
Web Development

Vue - Basics

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5      <title>Vue Setup</title>
6      <script src="https://vuejs.org/js/vue.js"></script>
7    </head>
8    <body>
9      <div id="app">
10       <p>{{ text }}</p>
11     </div>
12   </body>
13   <script>
14     new Vue({
15       el: "#app",
16       data: {
17         text: "Hello Vue!"
18       }
19     });
20   </script>
21 </html>
```

Listing 6: Vue Instance and DOM Template.

Alija Sabic
Web Development

Vue - Basics

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5      <title>Vue Setup</title>
6      <script src="https://vuejs.org/js/vue.js"></script>
7    </head>
8    <body>
9      <div id="app">
10       <p>{{ text }}</p>
11     </div>
12   </body>
13   <script>
14     new Vue({
15       el: "#app",
16       data: {
17         text: "Hello Vue!"
18       }
19     });
20   </script>
21 </html>
```

Template

Listing 6: Vue Instance and DOM Template.

Alija Sabic
Web Development

Vue - Basics

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5      <title>Vue Setup</title>
6      <script src="https://vuejs.org/js/vue.js"></script>
7    </head>
8    <body>
9      <div id="app">
10       <p>{{ text }}</p>
11     </div>
12   </body>
13   <script>
14     new Vue({
15       el: "#app",
16       data: {
17         text: "Hello Vue!"
18       }
19     });
20   </script>
21 </html>
```

Template

Instance

Listing 6: Vue Instance and DOM Template.

Alija Sabic
Web Development

Vue - Basics

With the `el` property of the configuration object, passed to the class constructor (`Vue( )`), we can use CSS selectors to specify what *templates* should be processed by the Vue *instance*.

Inside the *template* we can employ Vue's *template syntax* to reference `data` and `methods` of our Vue instance.

Alija Sabic
Web Development

Vue - Reactivity

Even though, the example is pretty simple and straightforward, Vue does a lot of work in the background. Vue's `data` property and the DOM are now linked together, which results in immediate updates of the DOM when changing the used properties.

To test the *reactivity* of our application, we can save a reference to our Vue instance and change the property manually using the browser's console.

```
13  <script>
14    const app = new Vue({
15      el: "#app",
16      data: {
17        text: "Hello Vue!"
18      }
19    });
20  </script>
```

Listing 7: Saving a reference to the Vue instance.

Alija Sabic
Web Development

Vue - Reactivity

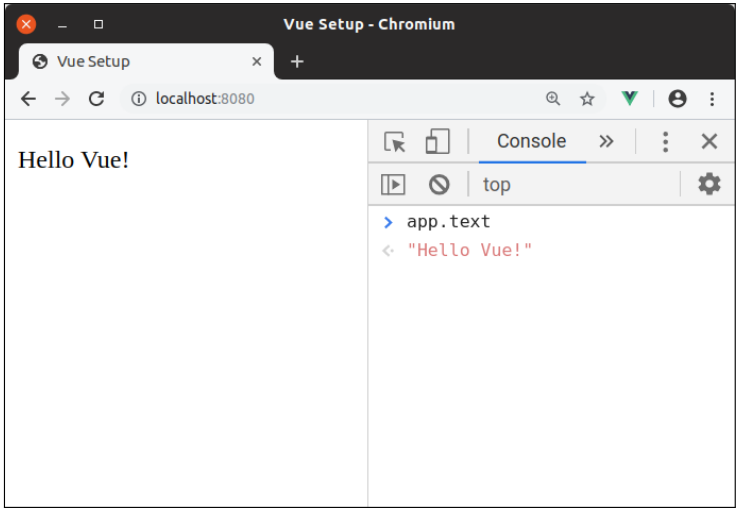

Figure 9: Changing properties of the Vue instance.

Alija Sabic
Web Development

Vue - Reactivity

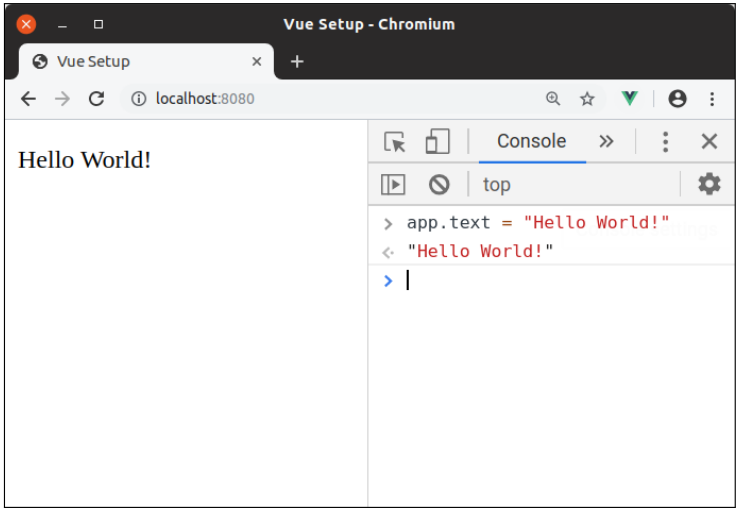

Figure 9: Changing properties of the Vue instance.

Alija Sabic
Web Development

Vue - Property data

The `data` property is used to store the data of our application. It will be used by Vue to create getters and setters from the properties inside of it.

```
14      data: {
15        text: "Hello Vue!"
16      },
```

Listing 8: The `data` property of the JSON (configuration) object.

Alija Sabic
Web Development

Vue - Property `data`

We can then access the data (e.g., `text`)

- Inside the template, using *Interpolation* or *String Interpolation*.
- Inside the Vue instance, using a (virtual) **`this`** reference.
- Using a reference to the Vue instance, either by
  - Vue's internal properties, that is
    - `app.$data.text` or
    - `app._data.text`
  - Generated getters and setters, that is `app.text`

Alija Sabic
Web Development

Vue - Property data

```
9   <div id="app">{{ text }}</div>
```

Listing 9: (String) Interpolation inside the template.

```
12  const app = new Vue({
13    el: "#app",
14    data: {
15      text: "Hello Vue!"
16    },
17    methods: {
18      hello() {
19        console.log(this.text);
20      }
21    }
22  });
```

Listing 10: Accessing data inside the Vue instance.

```
23  console.log(app.text); // Generated getters and setters
24  console.log(app.$data.text); // Vue's internal properties
```

Listing 11: Accessing the data with a reference to the Vue instance.

Alija Sabic
Web Development

Vue - Property data

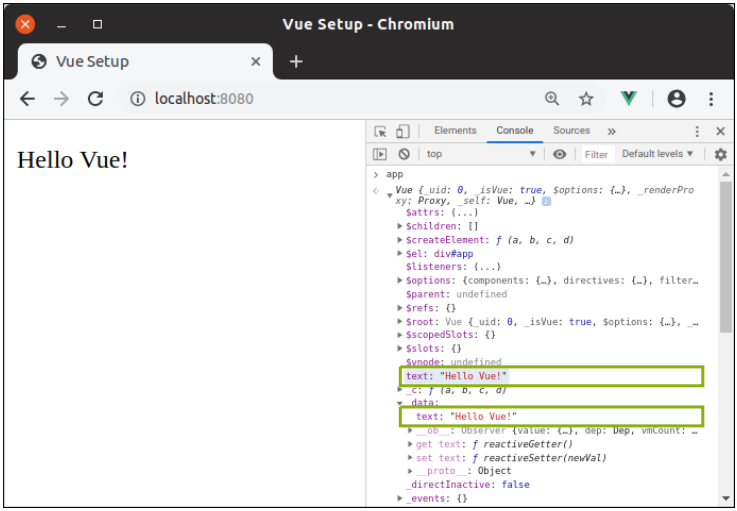

Figure 10: Getters and setters of the Vue instance.

Alija Sabic
Web Development

Vue - Property `methods`

The `methods` property is used to define methods to be mixed into the Vue instance.

All methods (except *arrow functions*) have their `this` context automatically bound to the Vue instance (by Vue).

```
18      methods: {
19        sayHello() {
20          return "Hello!";
21        },
22        sayText() {
23          // `this` is a reference to the Vue instance,
24          // not the JSON configuration object.
25          return this.text;
26        },
27        logText() {
28          console.log(this.sayHello());
29        }
30      }
```

Listing 12: The `methods` property of the JSON (configuration) object.

Alija Sabic
Web Development

Vue - Property `methods`

We can then access the methods (e.g., `sayHello`)

- Inside the template, using *Interpolation* or *String Interpolation*.
- Inside the Vue instance, using a (virtual) `this` reference.
- Using a reference to the Vue instance, either by
  - Vue's internal properties, that is
    - `app.$methods.sayHello()` or
    - `app._methods.sayHello()`
  - Generated methods, that is `app.sayHello()`

Alija Sabic
Web Development

Vue - Property methods

```
9     <div id="app">{{ sayHello() }}</div>
10    <div id="app">{{ sayText() }}</div>
```

Listing 13: (String) Interpolation inside the template.

```
32    console.log(app.sayHello());
33    console.log(app.sayText());
34    app.logHello();
```

Listing 14: Accessing the methods with a reference to the Vue instance.

Alija Sabic
Web Development

Vue - Property methods

```
13      const app = new Vue({
14        el: "#app",
15        data: {
16          text: "Hello Vue!"
17        },
18        methods: {
19          sayHello() {
20            return "Hello!";
21          },
22          sayText() {
23            // `this` is a reference to the Vue instance,
24            // not the JSON configuration object.
25            return this.text;
26          },
27          logText() {
28            console.log(this.sayHello());
29          }
30        }
31      });
```

Listing 15: Accessing methods inside the Vue instance.

Alija Sabic
Web Development

Vue - Property computed

The **computed** property is used to cache computed results of our application.

```
27        computed: {
28          reverseText: function() {
29            return this.text
30              .split("")
31              .reverse()
32              .join("");
33          }
34        },
```

Listing 16: The **computed** property of the JSON (configuration) object.

Alija Sabic
Web Development

Vue - Property computed

The in-template expressions we saw before, are very convenient.

However, they are meant for simple operations and should not be used to create complex expressions, as they bloat the code and are hard to maintain.

```
10      <p>
11        {{
12          text
13            .split("")
14            .reverse()
15            .join("")
16        }}
17      </p>
```

Listing 17: Using instance data for a complex expression.

Alija Sabic
Web Development

Vue - Property computed

To avoid issues resulting from more complex computation and expressions of our application data, we can declare a computed property (e.g. `reverseText`).

```
27      computed: {
28        reverseText: function() {
29          return this.text
30            .split("")
31            .reverse()
32            .join("");
33        }
34      },
```

Listing 18: Defining a computed property.

```
18      <p>{{ reverseText }}</p>
```

Listing 19: Using computed properties in the template.

Alija Sabic
Web Development

Vue - Property computed

Actually, we could achieve the same results with a function inside the `methods` property.

```
35        methods: {
36          reverseText() {
37            return this.text
38              .split("")
39              .reverse()
40              .join("");
41          }
42        }
```

Listing 20: Defining a method instead of a computed property.

The difference, however, is that computed properties are cached on their (reactive) dependencies. A computed property will only be re-evaluated when its dependencies have changed.

Alija Sabic
Web Development

Vue - Property computed

The following computed property will never update, because `Date.now()` is not a reactive dependency.

```
27        computed: {
28          now: function() {
29            return Date.now();
30          }
31        }
```

Listing 21: Computed property without reactive dependencies.

In comparsion, a method invocation will *always* run the function whenever a re-render happens. Computed properties allow us to cache the results of potentially expensive computations on our application data. Whenever the dependencies of a properties changes, Vue will re-evaluate all dependent properties.

We can access computed properties the same way we access our application data (cf. Page 24)

Alija Sabic
Web Development

Vue - Property computed

Computed properties are by default getter-only, but we can provide a setter in case we need one.

```
27      computed: {
28        fullName: {
29          get: function() {
30            // Template String, same as
31            // this.firstname + " " + this.lastName
32            return `${this.firstName} ${this.lastName}`;
33          },
34          set: function(newName) {
35            const names = newName.split(" ");
36            this.firstName = names[0];
37            this.lastName = names[names.length - 1];
38          }
39        }
40      }
```

Listing 22: Computed property getter and setter.

Alija Sabic
Web Development

Vue - Property watch

Instead of using computed properties to tell Vue to re-evaluate our data, we can use a more generic way to observe our data changes.

Consider following example.

```
14      data: {
15        firstName: "Foo",
16        lastName: "Bar",
17        fullName: "Foo Bar"
18      },
19      watch: {
20        firstName: function(val) {
21          this.fullName = val + " " + this.lastName;
22        },
23        lastName: function(val) {
24          this.fullName = this.firstName + " " + val;
25        }
26      }
```

Listing 23: The `watch` property of the JSON (configuration) object.

Alija Sabic
Web Development

Vue - Property watch

We can achieve the same result, with computed properties.

```
14        data: {
15          firstName: "Foo",
16          lastName: "Bar"
17        },
18        computed: {
19          fullName: function() {
20            return this.firstName + " " + this.lastName;
21          }
22        }
```

Listing 24: Same functionality with computed properties (cmp. Listing 23).

Compared to the previous example (Listing 23), using computed properties is less imperative and less repetitive.

Alija Sabic
Web Development

Vue - Property watch

While computed properties are more appropriate in most cases, there are times when a custom watcher is necessary.

For instance, you might want to delay the execution of an update for a specific time, due to an expensive computation.

```
8   <body>
9     <div id="app">
10      <p>
11        Ask a yes/no question:
12        <input v-model="question" />
13      </p>
14      <p>{{ answer }}</p>
15    </div>
16  </body>
17  <script src="https://.../axios.min.js"></script>
18  <script src="https://.../lodash.min.js"></script>
```

Listing 25: Advanced watcher example.

Alija Sabic
Web Development

Vue - Property watch

```
20      new Vue({
21        el: "#app",
22        data: {
23          question: "",
24          answer:
25            "I cannot give you an answer " +
26            "until you ask a question!"
27        },
28        watch: {
29          // whenever question changes, this function will run
30          question: function(newQuestion, oldQuestion) {
31            this.answer = "Waiting for you to stop typing...";
32            this.debouncedGetAnswer();
33          }
34        },
35        created: function() {
36          this.debouncedGetAnswer =
37            _.debounce(this.getAnswer, 500);
38        },
```

Listing 25: Advanced watcher example.

Alija Sabic
Web Development

Vue - Property watch

```
39        methods: {
40          getAnswer: function() {
41            if (this.question.indexOf("?") === -1) {
42              this.answer =
43                "Questions usually contain a question mark. ;-)";
44              return;
45            }
46            this.answer = "Thinking...";
47            var vm = this;
48            axios
49              .get("https://yesno.wtf/api")
50              .then(function(response) {
51                vm.answer =
52                  _.capitalize(response.data.answer);
53              })
54              .catch(function(error) {
55                vm.answer =
56                  "Error! Could not reach the API. " + error;
57              });
58          }
59        }
```

Listing 25: Advanced watcher example.

Alija Sabic
Web Development

Acronyms I

CSS	Cascading Stylesheets
DOM	Document Object Model
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
`npm`	Node Package Manager
PRC	People's Republic of China
U.S.	United States

Alija Sabic
Web Development

References I

[1] V. Cromwell, “Between the Wires: An interview with Vue.js creator Evan You,” https://dev.to/betweenthewires/evan-you, 11 2016.

[2] Evan You, “GitHub Customer Stories - Evan You,” https://github.com/customer-stories/yyx990803, 2019.

[3] ——, “The Progessive JavaScript Framework,” https://vuejs.org/, 2014.