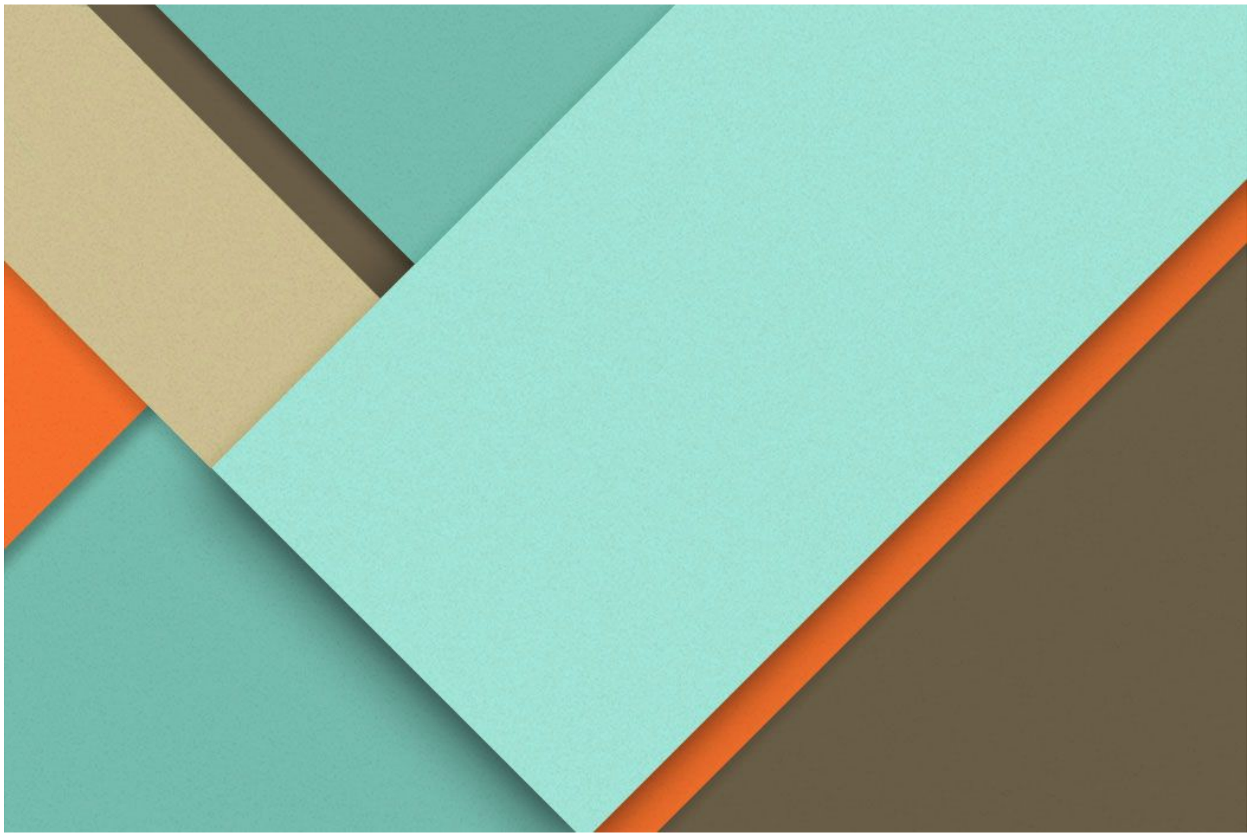


Quickly Quasar

Quasar, a Vue Widget Framework

Haris Hashim
© 2018 harishashim@gmail.com

Overview

A step by step training manual style instruction to accompany mentor and mentee discussion and learning session. This is a simple introduction to Quasar.

IMPORTANT This document does not include explanation that is done during one to one mentorship session.

Table of Contents

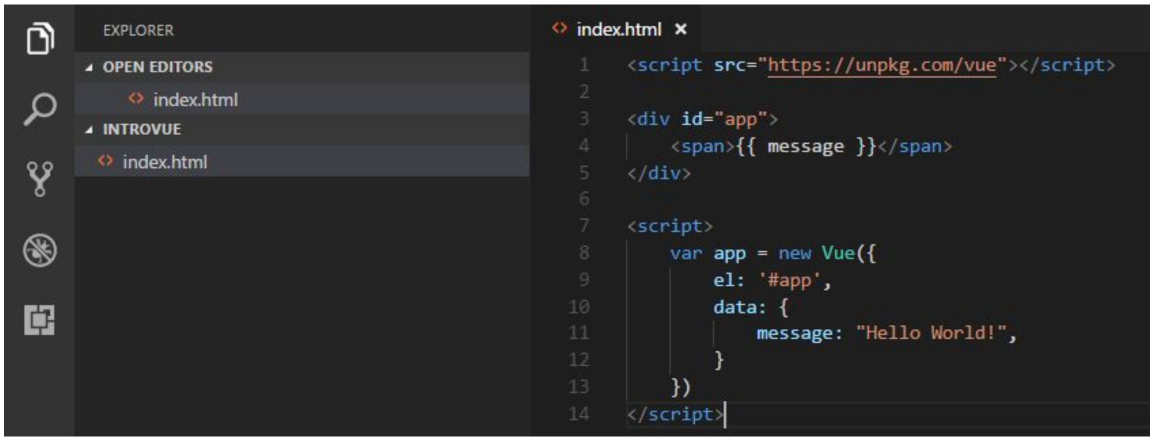
Part 0: Introduction to Vue

Prerequisite

1. VS Code installed
2. Chrome Browser installed
3. Internet connection

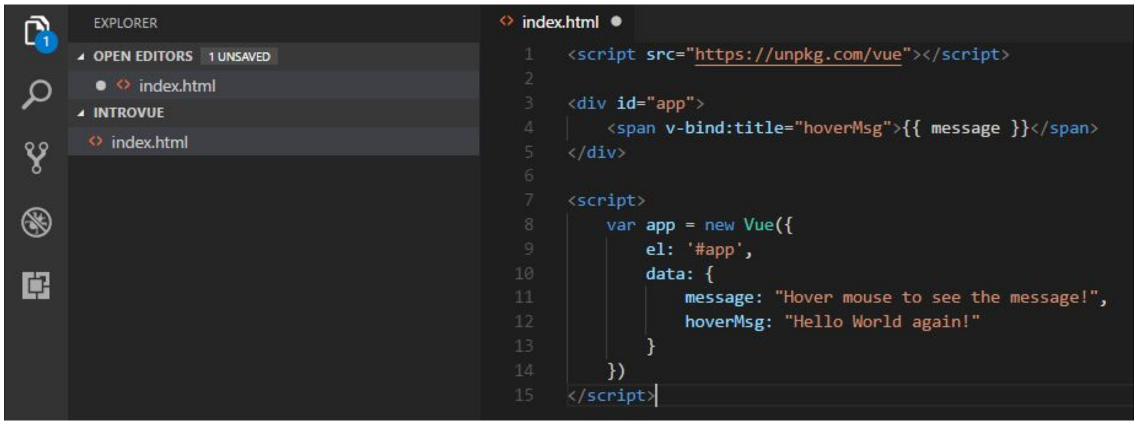
Steps

1. Create a root folder to contain all of our project. Let's name this folder QuicklyQuasar.
2. Create another folder inside QuicklyQuasar for this the tutorial and name it introvue.
3. Open introvue folder using VS Code and Create index.html file with the following contents:



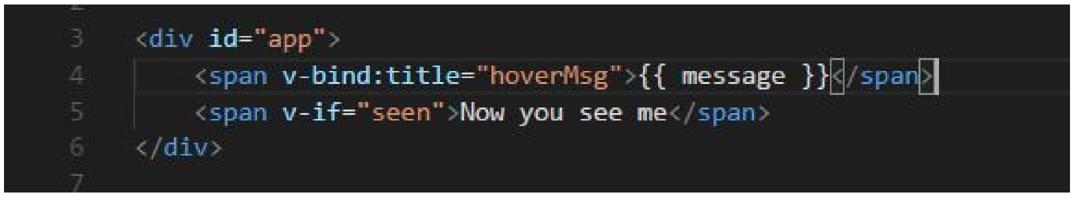
```
<script src="https://unpkg.com/vue"></script>

<div id="app">
    <span>{{ message }}</span>
</div>

<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: "Hello World!",
        }
    })
</script>
```

4. Save and open file in Chrome to see result. If the file is already opened, refresh using F5.
5. Open Chrome Dev Tool by pressing F12 key on the keyboard and open the Console tab so that you can type some JS code interactively.
6. Check the message variable value using app.message
7. Change message value by setting it to "Hi World!". Observe what is meant by reactive.
8. In above step text interpolation is used to reactively render variable value on a page. Another technique is bind element attribute. Do bellow code to implement this technique.



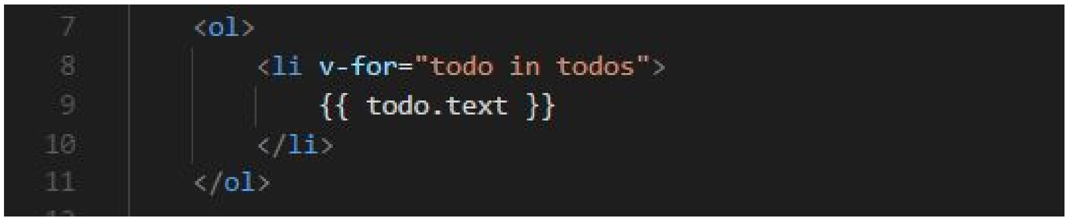
```
<script src="https://unpkg.com/vue"></script>

<div id="app">
    <span v-bind:title="hoverMsg">{{ message }}</span>
</div>

<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: "Hover mouse to see the message!",
            hoverMsg: "Hello World again!"
        }
    })
</script>
```

9. Refresh Chrome and repeat step to check and change hoverMsg variable value to “Hi World again!”
10. The usage of v-bind as attribute is called **directive**. This is special attribute provided by Vue. Let’s practice another directive called v-if directive.
11. Add bellow code after the message <span> element.



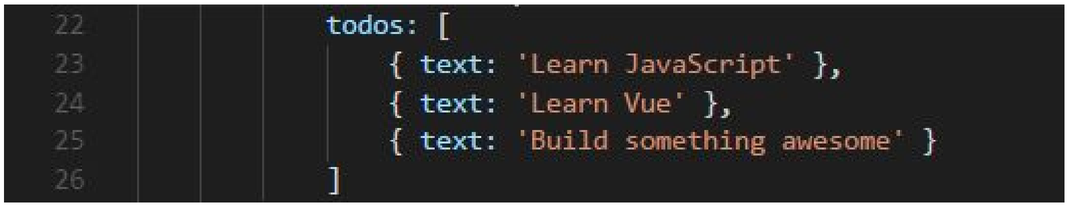
```
<div id="app">
    <span v-bind:title="hoverMsg">{{ message }}</span>
    <span v-if="seen">Now you see me</span>
</div>
```

12. To show your understanding, add another variable to data called seen and set the value to false.
13. Safe and refresh chrome, you will not see the new <span> element because seen is false. Change the value to true and observe what will happen.
14. And v-for directive to loop an element base on array variable.



```
    <ol>
        <li v-for="todo in todos">
            {{ todo.text }}
        </li>
    </ol>
```

15. With bellow array variable in data.



```
            todos: [
                { text: 'Learn JavaScript' },
                { text: 'Learn Vue' },
                { text: 'Build something awesome' }
            ]
```

16. Safe and refresh Chrome to see the to do list.
17. Add a new to do list item by running bellow code in Chrome Developer Tools.

```
app.todos.push({ text: 'New item' })
```

18. Observe Vue reactive nature when the new item is added immediately!

Exercise

1. This part of the training manual is a shorter version of **Introduction To Vue**. As a take home exercise, continue with the link to learn more. Instead of redoing the introduction as in the link, adapt it to all of the above step in introvue index.html file.

Part 1: Getting Started

Prerequisite

1. Node JS and NPM installed. Node version must be 8.9.0 or newer.

Steps

1. Open command prompt or console in QuicklyQuasar folder as created in part 0.
2. Install Quasar command line interface by executing in command prompt

   ```
   npm install -g quasar-cli
   ```

3. Install Vue command line interface by executing in command prompt

   ```
   npm install -g vue-cli
   ```

4. Install the default starter kit. This command will also create the quickly folder.

   ```
   quasar init quickly
   ```

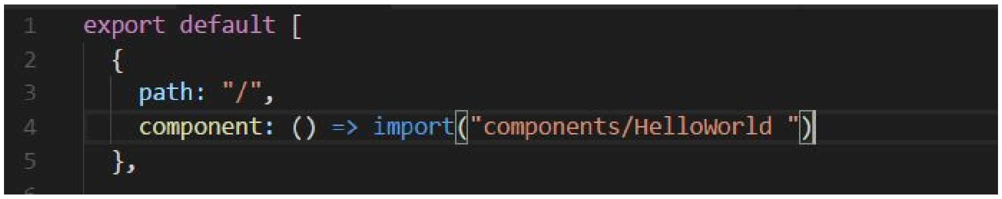
5. Changing directory to quickly folder generated above and execute.

   ```
   quasar dev
   ```

6. Open quickly folder as project workspace using VS Code and try to understand the codes.
   a. src/App.vue
   b. src/router/router.js
   c. src/layouts/default.vue
   d. src/pages/index.vue
   e. src/pages/404.vue
7. Rather than following above complex structure, we are going to do something simpler.
   a. Open another terminal (if needed) and create a new .vue file by doing

   ```
   quasar new component HelloWorld
   ```

   b. In router.js, change the route for path “/” to point to HelloWorld



```
export default [
  {
    path: "/",
    component: () => import("components/HelloWorld ")
  },
```

   c. Save file to see hot reload in action.
8. Change title in HelloWorld.vue from “My component” to “Hello World!”, save file to see hot reload in action again.

Part 2: Layout, Declaring Data, and Handling Click

Prerequisite

This part is continuing part 1. Code changes for layout and script is done in src/components/HelloWorld.vue file.

Configuring Eslint

Problem with eslint is that it give lots of error. This error does not confirm with formatting style executed by code formatter (when we use the ALT-SHIFT-F key). Paste bellow config in .eslintrc.js file in rules section.

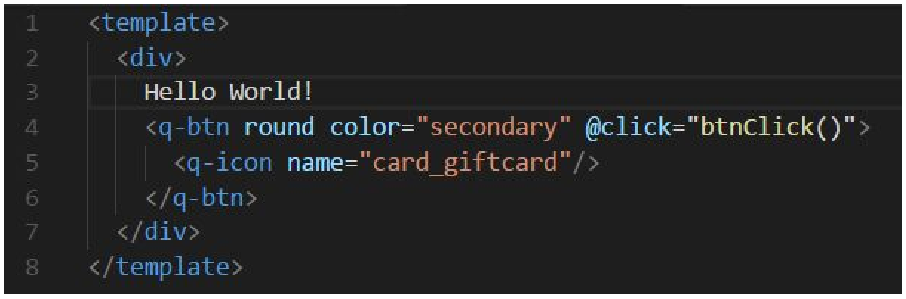
```
// Custom config
"space-before-function-paren": [0, "never"],
'semi': 0,
'brace-style': 0,
'quotes': 0,
'indent': 0,
'key-spacing': 0,
'no-tabs': 0,
'no-mixed-spaces-and-tabs': 0,
```

For every file with script tag, add following configuration exactly and after the opening script tag :

```
/* eslint brace-style: 0 */
```

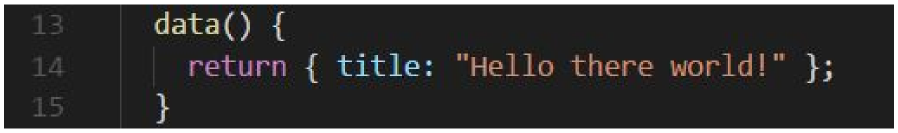
Steps

1. **Layout**. Change the layout to add a button.
    a. Put the button codes after “Hello World!” text:



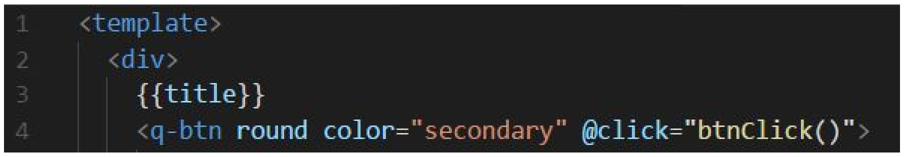
```
<template>
  <div>
    Hello World!
    <q-btn round color="secondary" @click="btnClick()">
      <q-icon name="card_giftcard"/>
    </q-btn>
  </div>
</template>
```

    b. Save and check in browser that button is shown after the title.
2. **Declaring Variable.** Rather than “hard coding” the title. We will use a variable.
    a. Add title variable to data() with previously specified value.



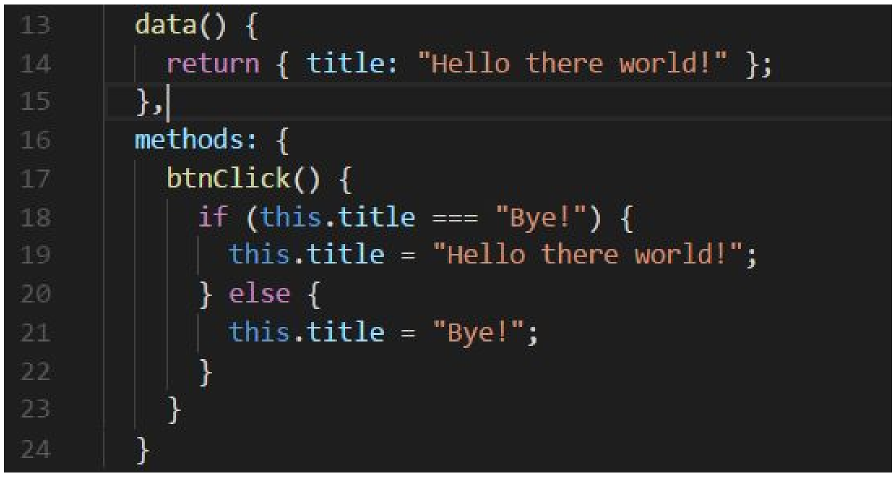
```
  data() {
    return { title: "Hello there world!" };
  }
```

    b. Replace “Hello World!” with title.



```
<template>
  <div>
    {{title}}
    <q-btn round color="secondary" @click="btnClick()">
```

3. **Handling Click**. Nothing will happen if button is clicked now. We need to code the handler.
    a. Add methods section and btnClick function to <script>. The code will toggle title to “Bye!” just so that we can see change happened when button is clicked.



```
  data() {
    return { title: "Hello there world!" };
  },
  methods: {
    btnClick() {
      if (this.title === "Bye!") {
        this.title = "Hello there world!";
      } else {
        this.title = "Bye!";
      }
    }
  }
```

    b. Save and test the button.