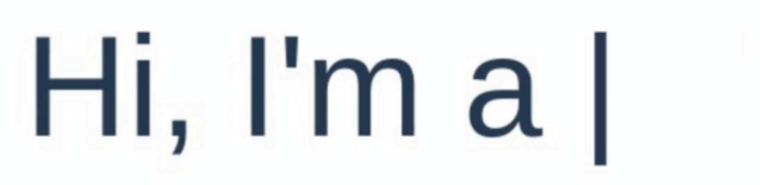
Gayathri R
Posted on Apr 23

How to add a Typewriter animation in VueJs

#javascript #webdev #vue #tutorial

In this blog, I'm going to explain to you how to make a typing animation in VueJs.

Here's a preview:



Let's get started...

Create a new VueJs project.

```
$ vue create animations-vuejs
```

```
Vue CLI v5.0.4
? Please pick a preset: Default ([Vue 3] babel, eslint)


Vue CLI v5.0.4
✨  Creating project in /home/user/tutorials/animations-vuejs.
🗃  Initializing git repository...
⚙️  Installing CLI plugins. This might take a while...


added 839 packages, and audited 840 packages in 35s

84 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
🚀  Invoking generators...
📦  Installing additional dependencies...


added 97 packages, and audited 937 packages in 9s

94 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project animations-vuejs.
👉  Get started with the following commands:

 $ cd animations-vuejs
 $ npm run serve
```

Template

The template is quite simple. To create the typewriter effect, you need an element for static text and an element for changing text. This component contains three span tags encapsulated in a div.

```
<template>
  <div class="container">
    <h1>
```

```
      Hi, I'm a
      <span class="typed-text">{{ typeValue }}</span>
      <span class="blinking-cursor">|</span>
      <span class="cursor" :class="{ typing: typeStatus }">&nbsp;</span>
    </h1>
  </div>
</template>
```

Styles

I've used my own custom styling for displaying the contents & cursor blinking. Here is the code,

```
<!-- Add "scoped" attribute to limit CSS to this component only -->
<style lang="scss" scoped>
.container {
  width: 100%;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}
h1 {
  font-size: 6rem;
  font-weight: normal;
  span.typed-text {
    color: #d2b94b;
  }
}

// Cursor blinking CSS Starts...
.blinking-cursor {
  font-size: 6rem;
  color: #2c3e50;
  -webkit-animation: 1s blink step-end infinite;
  -moz-animation: 1s blink step-end infinite;
  -ms-animation: 1s blink step-end infinite;
  -o-animation: 1s blink step-end infinite;
  animation: 1s blink step-end infinite;
}
@keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-moz-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-webkit-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-ms-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-o-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
// Cursor blinking CSS Ends...
</style>
```

> **Info** Using SCSS inside the *VueJs (Default ([Vue 3] babel, eslint))* components you may face ***'sass-loader'*** issues. Please refer to this blog for fixing the problem.

Script

Script

- **Data**

This component mainly contain 5 values: typeValue, displayTextArray, typingSpeed, erasingSpeed, and newTextDelay. The **typeValue** field is the static text. The **displayTextArray** field is an array of changing words. The **typingSpeed** field is the typing speed, and the **erasingSpeed** field is the delete speed. The **newTextDelay** field is the delay time seconds to start printing the next word.

```
data: () => {
    return {
      typeValue: "",
      typeStatus: false,
      displayTextArray: ["YouTuber", "Developer", "Blogger", "Designer", "F
      typingSpeed: 100,
      erasingSpeed: 100,
      newTextDelay: 2000,
      displayTextArrayIndex: 0,
      charIndex: 0,
    };
  },
```

Methods

- **typeText() & eraseText()**

These methods contain all the logic to determine which word is being typed, whether to type or delete, or to change to the next word based on the 'typeStatus' field. Take a look below.

```
methods: {
    typeText() {
      if (this.charIndex < this.displayTextArray[this.displayTextArrayInde
        if (!this.typeStatus) this.typeStatus = true;
        this.typeValue += this.displayTextArray[this.displayTextArrayIndex
          this.charIndex
        );
        this.charIndex += 1;
        setTimeout(this.typeText, this.typingSpeed);
      } else {
        this.typeStatus = false;
        setTimeout(this.eraseText, this.newTextDelay);
      }
    },
    eraseText() {
      if (this.charIndex > 0) {
        if (!this.typeStatus) this.typeStatus = true;
        this.typeValue = this.displayTextArray[this.displayTextArrayIndex]
          0,
          this.charIndex - 1
        );
        this.charIndex -= 1;
        setTimeout(this.eraseText, this.erasingSpeed);
      } else {
        this.typeStatus = false;
        this.displayTextArrayIndex += 1;
        if (this.displayTextArrayIndex >= this.displayTextArray.length)
          this.displayTextArrayIndex = 0;
        setTimeout(this.typeText, this.typingSpeed + 1000);
      }
    },
  },
```

Created Lifecycle

When the component is loaded, it calls the typeText() method to begin the typing sequence.

```
created() {
   setTimeout(this.typeText, this.newTextDelay + 200);
},
```

Here's the final code:

```
<template>
  <div class="container">
    <h1>
      Hi, I'm a
      <span class="typed-text">{{ typeValue }}</span>
      <span class="blinking-cursor">|</span>
      <span class="cursor" :class="{ typing: typeStatus }">&nbsp;</span>
    </h1>
  </div>
</template>

<script>
export default {
  name: "typeWiriter",
  data: () => {
    return {
      typeValue: "",
      typeStatus: false,
      displayTextArray: ["YouTuber", "Developer", "Blogger", "Designer", "F
      typingSpeed: 100,
      erasingSpeed: 100,
      newTextDelay: 2000,
      displayTextArrayIndex: 0,
      charIndex: 0,
    };
  },
  props: {},
  created() {
    setTimeout(this.typeText, this.newTextDelay + 200);
  },
  methods: {
    typeText() {
      if (this.charIndex < this.displayTextArray[this.displayTextArrayInde
        if (!this.typeStatus) this.typeStatus = true;
        this.typeValue += this.displayTextArray[this.displayTextArrayIndex
          this.charIndex
        );
        this.charIndex += 1;
        setTimeout(this.typeText, this.typingSpeed);
      } else {
        this.typeStatus = false;
        setTimeout(this.eraseText, this.newTextDelay);
      }
    },
    eraseText() {
      if (this.charIndex > 0) {
        if (!this.typeStatus) this.typeStatus = true;
        this.typeValue = this.displayTextArray[this.displayTextArrayIndex].
          0,
          this.charIndex - 1
        );
        this.charIndex -= 1;
        setTimeout(this.eraseText, this.erasingSpeed);
      } else {
        this.typeStatus = false;
        this.displayTextArrayIndex += 1;
        if (this.displayTextArrayIndex >= this.displayTextArray.length)
          this.displayTextArrayIndex = 0;
        setTimeout(this.typeText, this.typingSpeed + 1000);
      }
    },
  },
};
</script>
```

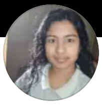
```
<!-- Add "scoped" attribute to limit CSS to this component only -->
<style lang="scss" scoped>
.container {
  width: 100%;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}
h1 {
  font-size: 6rem;
  font-weight: normal;
  span.typed-text {
    color: #d2b94b;
  }
}

// Cursor blinking CSS Starts...
.blinking-cursor {
  font-size: 6rem;
  color: #2c3e50;
  -webkit-animation: 1s blink step-end infinite;
  -moz-animation: 1s blink step-end infinite;
  -ms-animation: 1s blink step-end infinite;
  -o-animation: 1s blink step-end infinite;
  animation: 1s blink step-end infinite;
}
@keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-moz-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-webkit-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-ms-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
@-o-keyframes blink {
  from,
  to {
    color: transparent;
  }
  50% {
    color: #2c3e50;
  }
}
// Cursor blinking CSS Ends...
</style>
```

Do you like this solution? Don’t forget to star the repo on GitHub. Stars keep me motivated and are highly appreciated.

Code reviews welcome. Let me know if I can do something better.

Discussion (0)

Code of Conduct • Report abuse



Gayathri R

UI/UX Developer & DevOps Evangelist @ gopaddle.io | Co-Organizer @ KCD Chennai | Tech Blogger | Speaker

LOCATION

LOCATION
Palakkad, Kerala, India

EDUCATION
Master of Computer Application

WORK
UI/UX Developer @ gopaddle.io

JOINED
Mar 12, 2022

More from Gayathri R

ReactJs QR Code Generator
#webdev #javascript #beginners #react

How to resolve CORS issue in VueJs
#webdev #beginners #vue #javascript

How to resolve "multi-word vue/multi-word-component-names" issue in VueJs 3 default option.
#beginners #javascript #webdev #vue