**Section 5 – Vue Behind The Scenes**

Chapter 59 – An Introduction To Vue’s Reactivity

1. The data you define is something Vue keeps track of, Vue will take the data and merge all your properties into a global behind the scenes managed object, also your methods. Vue will turn them into data object and into a reactive data object.

Chapter 60 – Vue Reactivity: A Deep Dive

1. Example 1 using JavaScript

let message = 'Hello!';

let longMessage = message + ' World!';

console.log(longMessage);

message = 'Hello!!!';

console.log(longMessage);

// Result

Hello! World!

Hello! World!

1. Example 2 using Vue

const data = {

  message: 'Hello!'

};

const handler = {

  set(target, key, value) {

    console.log(target);

    console.log(key);

    console.log(value);

  }

};

const proxy = new Proxy(data, handler);

proxy.message = 'Hello!!!';

// Result

{message: "Hello!"}

message

Hello!!!

1. Using Vue

const data = {

  message: 'Hello!',

  longMessage: 'Hello! World!'

};

const handler = {

  set(target, key, value) {

    if (key === 'message') {

      target.longMessage = value + ' World!';

    }

    target.message = value;

  }

};

const proxy = new Proxy(data, handler);

proxy.message = 'Hello!!!';

console.log(proxy.longMessage);

// Result

Hello!!! World!

Chapter 61 – One App VS Multiple Apps

1. We may have two or more Apps in one project, such as App and App2. App and App2 will not connect each other, all the data, properties, and methods will be different.

<section id="app2">

      <!-- This will not work, because it is not connected to id="app"

      <p>{{ message }}</p>  -->

      <p>{{ favoriteMeal }}</p>

</section>

Chapter 62 – Understanding Templates

1. We can make a template in Vue (App), the HTML code will be like below.

<section id="app2">

</section>

The Vue code like below.

const app2 = Vue.createApp({

  template: `

    <p>{{ favoriteMeal }}</p>

  `,

  data() {

    return {

      favoriteMeal: 'Pizza!'

    };

  }

});

app2.mount('#app2');

Template will give a result like a constructor, so if you make a new App2 section, the template will be printed.

Chapter 63 – Working With Refs

1. Refs is one of Vue feature that allows you to retrieve values from DOM elements when you need them, instead of all the time. The implementation for HTML Code like below.

<input type="text" ref="userText">

And the Vue Code like below.

methods: {

    saveInput(event) {

      this.currentUserInput = event.target.value;

    },

    setText() {

      // this.message = this.currentUserInput;

      // $ sign is used to define Vue variable

      this.message = this.$refs.userText

      // Will print the HTML code, such as <input type="text">

      console.log(this.$refs.userText);

      // Will print the input directory

      console.dir(this.$refs.userText);

      // Will print the inputted value

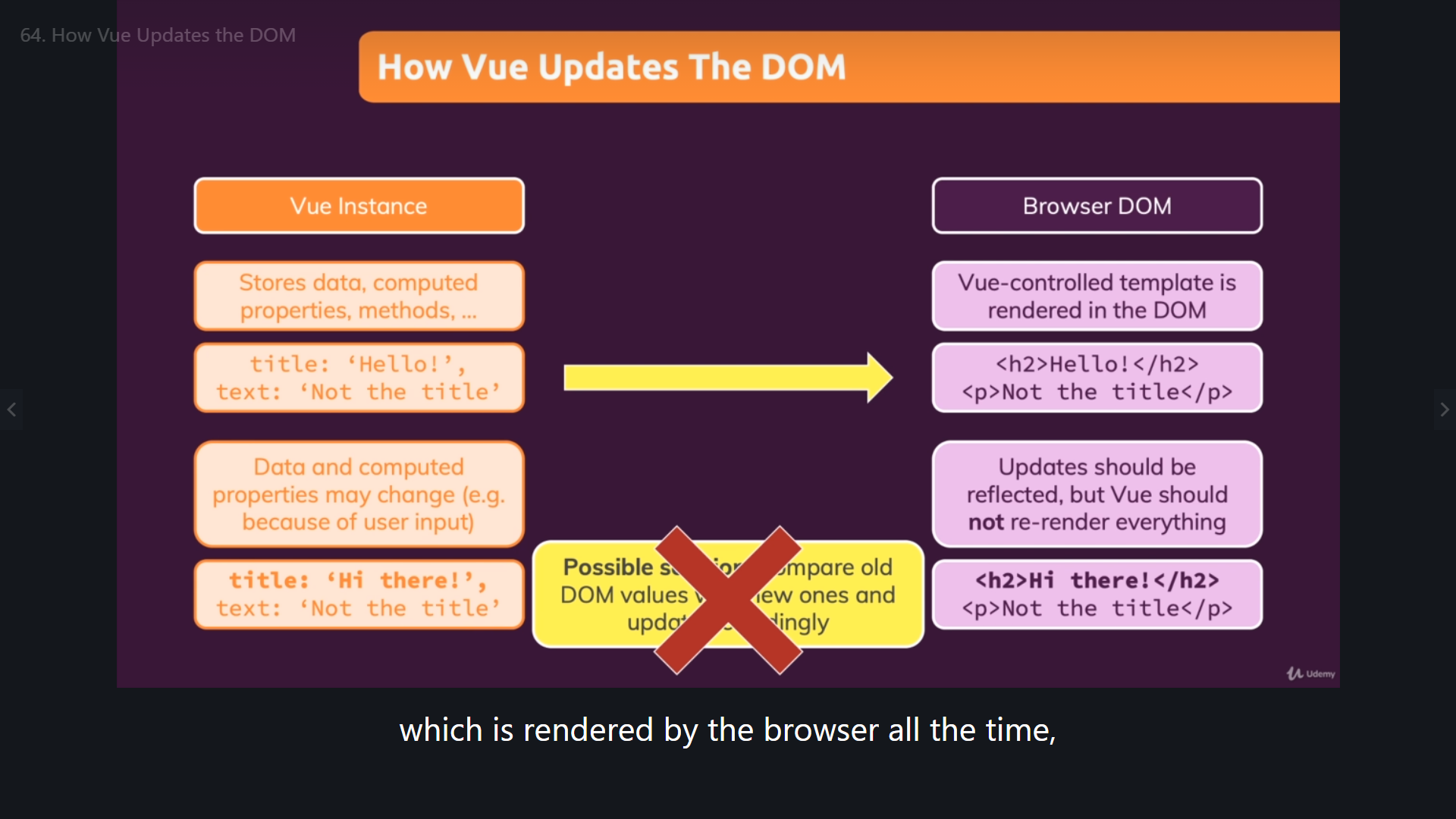
      this.message = this.$refs.userText.value;

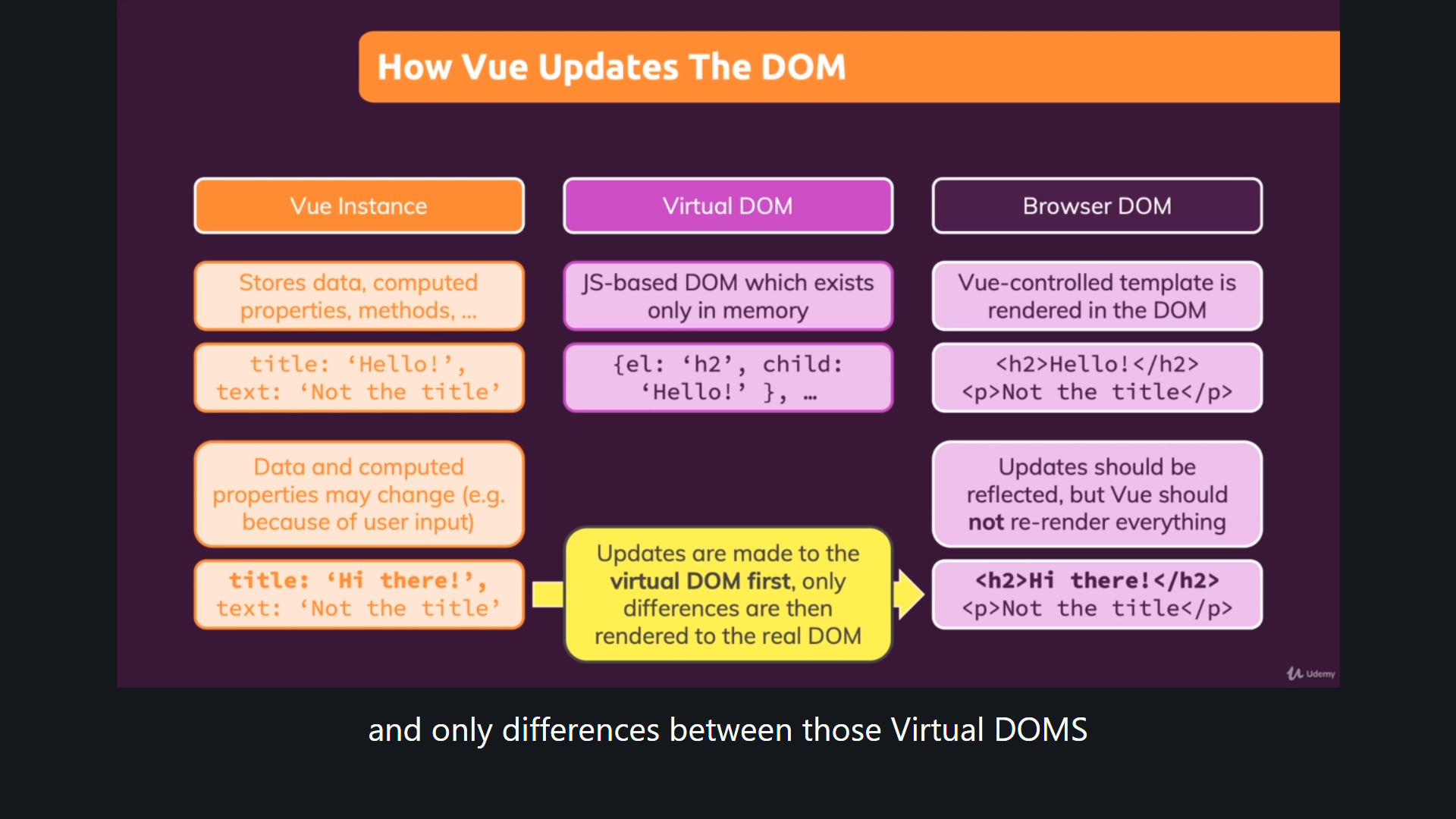
    },

},

Chapter 64 – How Vue Updates The DOM

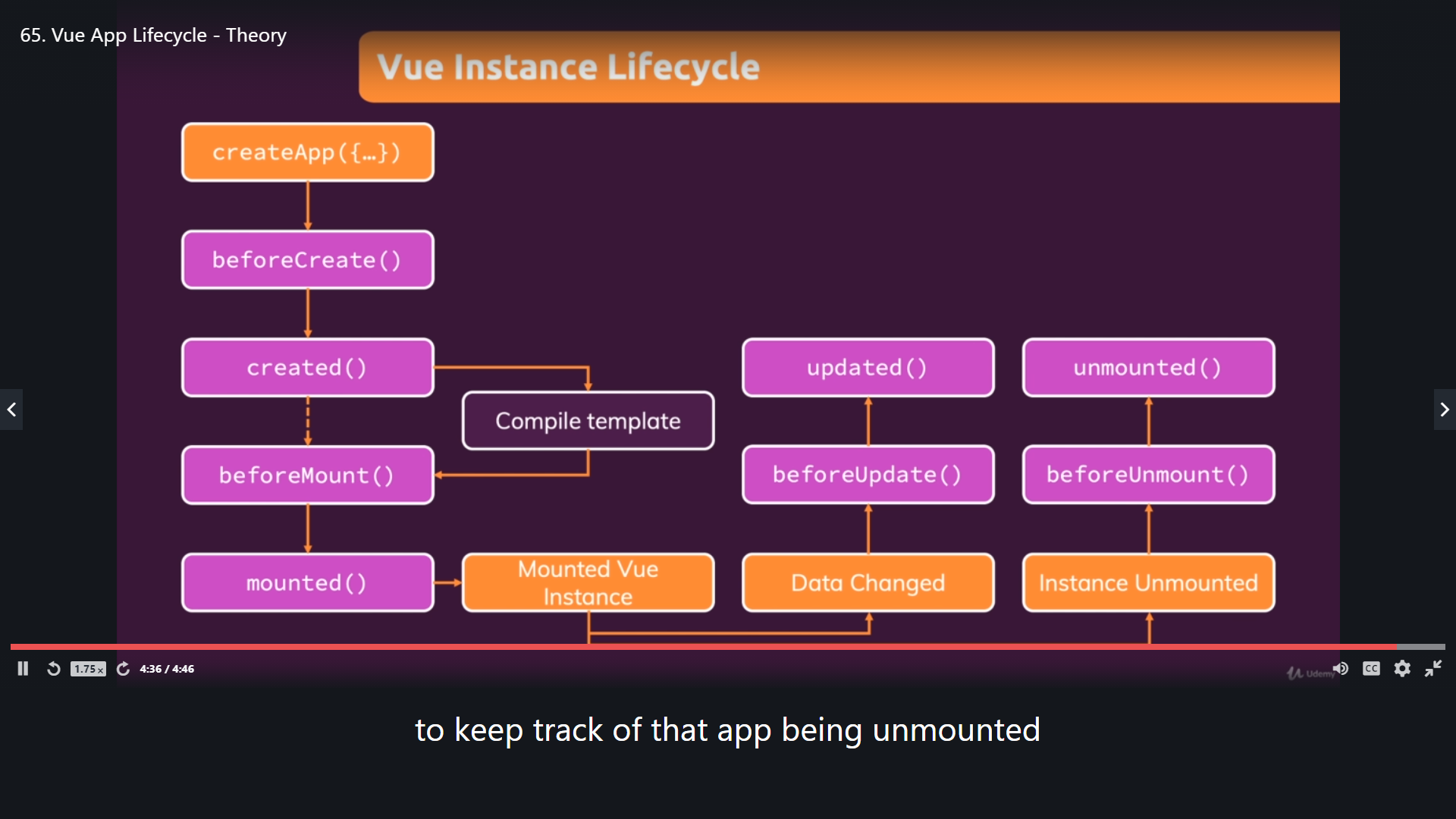
1. The concept like below.





Chapter 65 – Vue App Lifecycle Theory

1. Create App can be reflected in methods, add to your Vue app configuration object tor un code you might need to run at this point of time.



Chapter 66 – Vue App Lifecycle Practice

1. Checking BeforeCreate() in Vue App Lifecycle

methods: {

    saveInput(event) {

      this.currentUserInput = event.target.value;

    },

    setText() {

      // Will print the inputted value

      this.message = this.$refs.userText.value;

    },

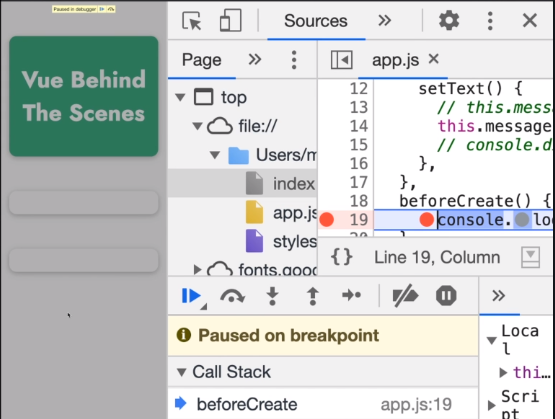
    beforeCreate() {

      console.log('beforeCreate()');

    }

}

The result will be like below.

In this situation, you will not see anything in your window.

1. Checking Create() in Vue App Lifecycle

beforeCreate() {

      console.log('beforeCreate()');

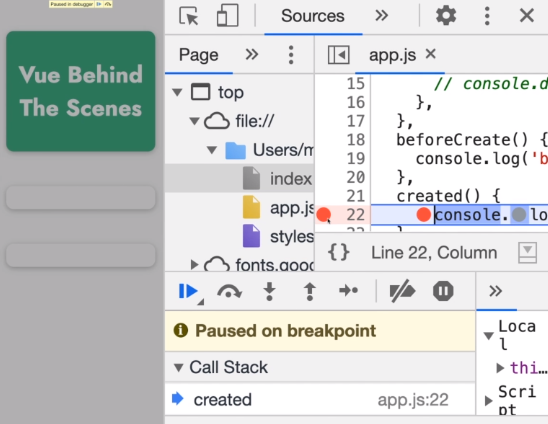
    },

    created() {

      console.log('created()');

    }

The result will be like below.



In this situation, you also will not see anything in your window.

1. Checking BeforeMount() and Mounted() in Vue App Lifecycle

beforeCreate() {

      console.log('beforeCreate()');

    },

    created() {

      console.log('created()');

    },

    beforeMount() {

      console.log('beforeMount()');

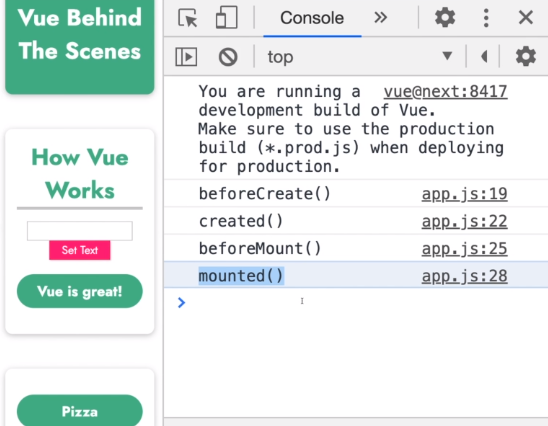
    },

    mounted() {

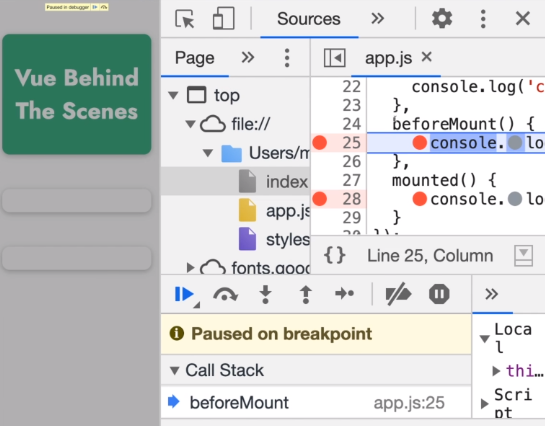
      console.log('mounted()');

    }

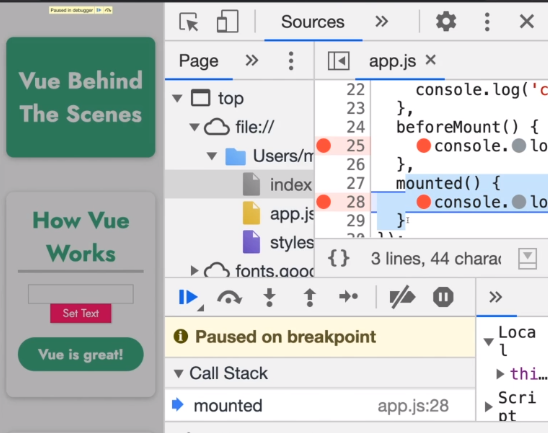
The result will be like below.



BeforeMount() Window Result



Mounted() Window Result



1. How about Data Changed ?

beforeCreate() {

      console.log('beforeCreate()');

    },

    created() {

      console.log('created()');

    },

    beforeMount() {

      console.log('beforeMount()');

    },

    mounted() {

      console.log('mounted()');

    },

    beforeUpdate() {

      console.log('beforeUpdate()');

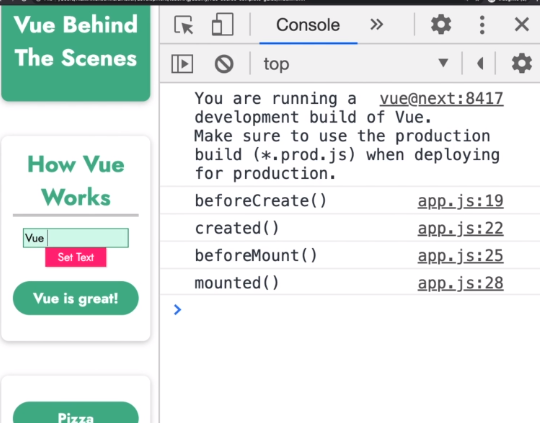
    },

    update() {

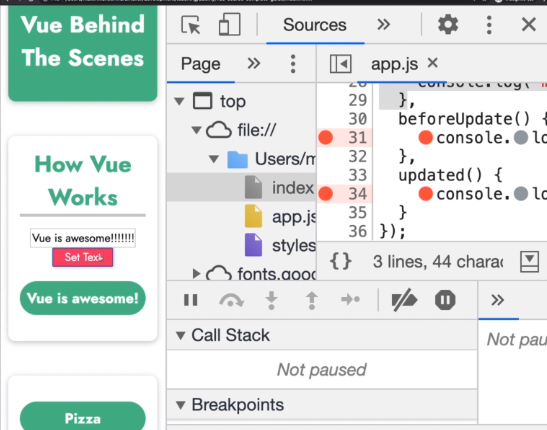
      console.log('updated()');

    }

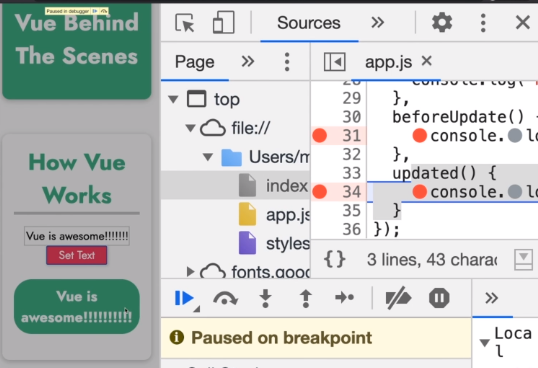
The result will be like below.



BeforeUpdate() Window Result



Updated() Window Result



1. How about Instance Unmounted ?

beforeCreate() {

      console.log('beforeCreate()');

    },

    created() {

      console.log('created()');

    },

    beforeMount() {

      console.log('beforeMount()');

    },

    mounted() {

      console.log('mounted()');

    },

    beforeUpdate() {

      console.log('beforeUpdate()');

    },

    updated() {

      console.log('updated()');

    },

    beforeUnmount() {

      console.log('beforeUnmount()');

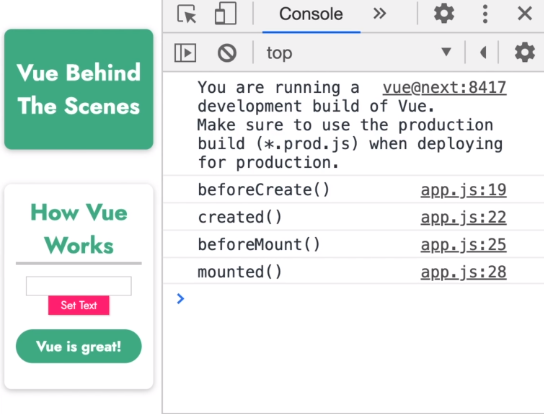
    },

    unmounted() {

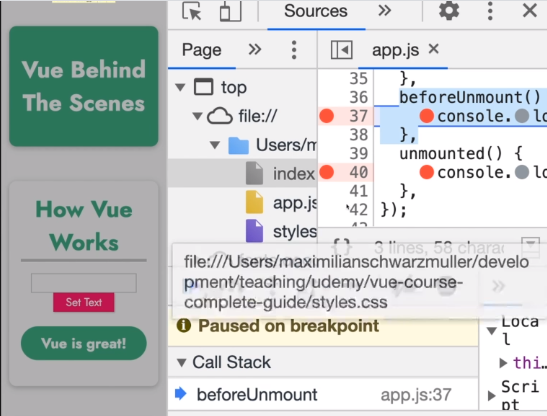
      console.log('unmounted()');

    }

The result will be like below.



BeforeUnmount() Window Result



Unmounted() Window Result

