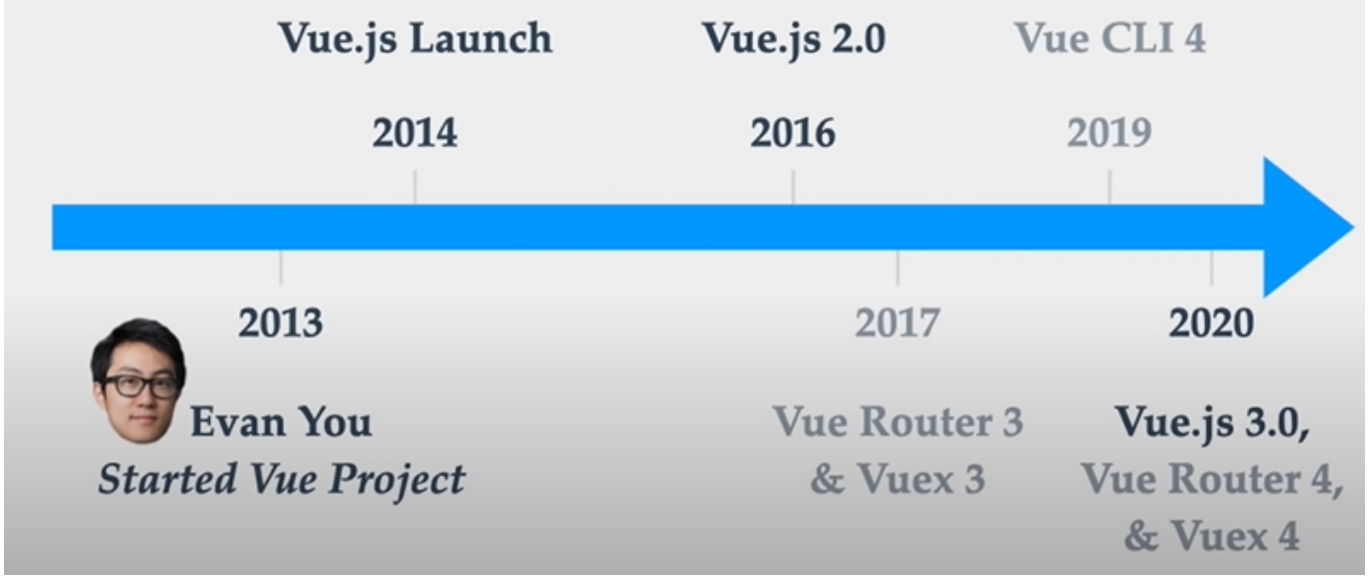
Features & Benefits

- Virtual DOM
- Lightweight (10k gzip)
- Progressive
- Vue Ecosystem
- Flexible

1. Virtual Dom ist synchronisierte Version des HTML DOMs und ist gespeichert in dem Arbeitsspeicher und ermöglicht effizienz
2. Vue ist leichtfertig
3. Progressiv -> man kann wenig vue importieren oder es als ganzes Framework nutzen

History of Vue.js



Basic usage ⌨ (0:12:17)

1. Two-Way Databinding Options API -> data: function() { greeting: "hello"}
   -> `<div>{{greeting}}</div>`

Vue JS Directives ⌨ (0:19:48)

1. v-if /else/else-if -> conditional rendering -> inserting and excluding Element of Dom
2. v-show -> changing the display property so it is more efficient

Events and Methods ⌨ (0:29:20)

1. v-on:"event" -> click or other events -> @click="greeting(variableInData)"
2. eventModifer -> keyup.modifier/enter -> when pressed enter it fires
3. @click.prevent.stop -> prevent submit by clicking -> chainable

Components ⌨ (0:39:27)

1. Normale HTML Tag with custom name placed in Dom
2. defined in vue app between the app init and app.mount
3. funtion app.component("nameofCustomComponent", Options API)

4.
```
<script src="https://unpkg.com/vue@next"></script>
<script>
  let app = Vue.createApp({
    data: function () {
      return {
        greeting: "Hello Vude3",
        isVisible: false,
      };
    },
    methods: {
      toggelBox() {
        this.isVisible = !this.isVisible
      },
    }
  });
  app.component('login-form', {
      template:  `
          <form @submit.prevent="handlSubmit">
              <h1> {{ title}} </h1>
              <input type="email" v-model="email" />
              <input type="password" v-model="password" />
              <button>Log in </button>
          </form>
      `,
      data() {
          return {
              title: 'Login Form',
              email: '',
              password: ''
          }
      },
      methods: {
          handlSubmit() {
              console.log(this.email, this.password)
          }
      }
  } )
  app.mount("#app");
</script>
</body>
```

Component Props ⌨ (0:57:30)

1. There are parent components and child components -> possibility to exchange data between them -> after template definition of ComponentHTML place

"components[]" array with other Custom-Components

2. Data declared at one child component can be used in the parent component throug (v-bind:"property") that takes HTML tag properties and bind them to javascript -> `<parentForm> <childInput v-bind:label="vueProp">` -> parent data containing the vueProp: "labelName" -> in ChildComponent declare
props: ['label'] to set the parent Label in the input that container {{label}}
3. v-bind:label -> :label
4. all v-model define for the same variable an modelValue that can be passed to childs in props
5. below props there could be a computed object get() and set() the childComponent v-model="inputValue"
6. there is the possibiliy to trigger functions with this.
$emit("functionname") -> this$.emit('update:modelValue', value)

```
app.component('login-form', {
        template: `
        <form @submit.prevent="handlSubmit">
            <h1> {{ title}} </h1>
            <custom-input v-model="email" :label="emailLabel" />
            <custom-input v-model="password" :label="passwordLabel" />
            <button>Log in </button>
        </form>
    `,
        components: ['custom-input'],
        data() {
            return {
                title: 'Login Form',
                email: '',
                password: '',
                emailLabel: 'Email',
                passwordLabel: 'Password',
            }
        },
        methods: {
            handlSubmit() {
                console.log(this.email, this.password)
            }
        }
    })
    app.component('custom-input', {
        template: `
        <label>
            {{label}}
            <input type="text" v-model="inputValue">
        </label>
    `,
        props: ['label', 'modelValue'],
        computed: {
            inputValue: {
                get() {
                    return this.modelValue;
                },
                set(value) {
                    this.$emit('update:modelValue', value)
                }
            }

        },
        data() {
            return {
                inputVlue: '',
            }
        }
    })
```

Loops ⌨ (1:06:09)

1. `<p v-for="{str,i} in inputs" :key="i"> {{str}}</p>`

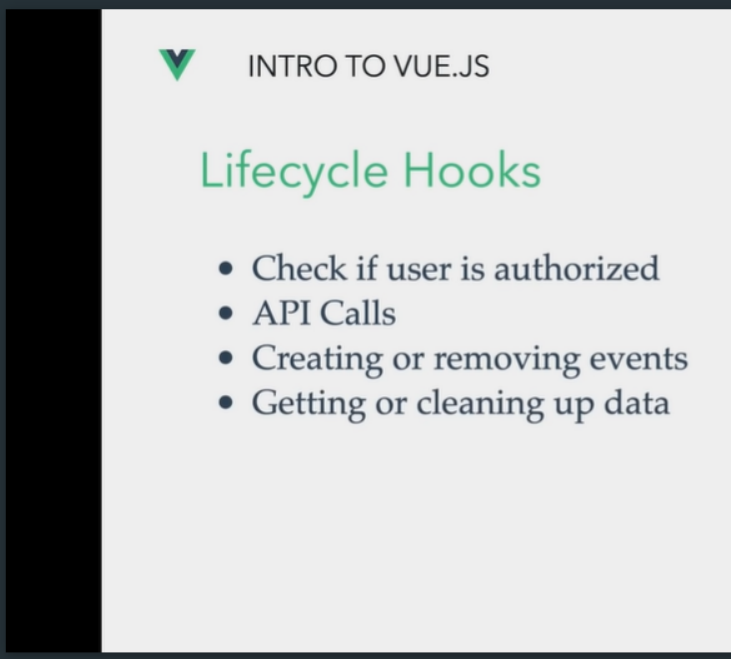
2.

```
app.component('login-form', {
          template: `
          <form @submit.prevent="handlSubmit">
              <h1> {{ title}} </h1>
              <custom-input
              v-for="(input,i) in inputs"
              :key="i"
              v-model="input.value"
              :label="input.label"
              :type="input.type" />
              <button>Log in </button>
          </form>
      `,
          components: ['custom-input'],
          data() {
              return {
                  title: 'Login Form',
                  inputs: [
                      {
                          label: 'Email',
                          value: '',
                          type: 'email'
                      },
                      {
                          label: 'Password',
                          value: '',
                          type: 'password'
                      }
                  ],
                  email: '',
                  password: '',
                  emailLabel: 'Email',
                  passwordLabel: 'Password',
              }
          },
          methods: {
              handlSubmit() {
                  console.log(this.inputs[0].value, this.inputs[1].value)
              }
          }
      })
      app.component('custom-input', {
          template: `
          <label>
              {{label}}
              <input :type="type" v-model="inputValue">
          </label>
      `,
          props: ['label','type', 'modelValue'],
```

Lifecycle Hooks Adding Vue.js to a static site ⌨ (1:14:30

1. Components or html elemts entering and leaving the dom is their lifecycle
   a. beforeCreated, created, beforeMounted, mounted, beforeUpdated, updated, beforeUnmounted, unmounted -> used often for API Calls
   b.

```
<script src="https://unpkg.com/vue@next"></script>
<script>
  let app = Vue.createApp({
    data: function () {
      return {
        isVisible: false,
      };
    },
    methods: {
      toggelBox() {
        this.isVisible = !this.isVisible
      }
    },
    updated() {

    },
  });
  app.component('test-box', {
      template:`
          <div class="box"></div>
      `,
      created() {

      },
      mounted() {

      },
      unmounted() {

      },

  })
  app.mount("#app");
```



App Demo ⌨ (1:26:45)
Adding Items to Cart ⌨ (1:43:22)
Dynamic Content ⌨ (2:15:54)
Reuseable Components Single Page Application with Vue CLI ⌨ (2:26:20)
Vue CLI ⌨ (2:32:48)
Vue Folder Structure ⌨ (2:43:58)
Top Nav ⌨ (2:48:45)
Styling with SASS ⌨ (3:00:25)
Application Pages ⌨ (3:06:07)
Sidebar ⌨ (3:20:46)
Adding Items to Cart ⌨ (3:24:46)
Finishing Up (Card component)