Vuejs Class Notes

AltSchool Africa

Are you ready to learn Vue? Press `space` on your keyboard →

What is Vuejs?

Vue.js is a popular open-source JavaScript framework specifically desgined for building user interfaces (UIs) and single-page applicaions(SPAs).

Read more about Vuejs?

Getting Started

Create a starter project with just `index.html` or use the vite starter to scaffold a new project.

Getting Started using the Options API

Create a starter project with just `index.html` and use the CDN or use the vite starter or create-vue to scaffold a new project.

Table of contents

Class Note uses a vscode monaco editor and shiki to display types with twoslash for types annotation and errors for codeblocks

```
// More at https://shiki.style/packages/twoslash

import { computed, ref } from 'vue'

const count = ref(0);
const doubled = computed(() => count.value * 2);

doubled.value = 2
```

```
Cannot assign to 'value' because it is a read-only property.
```

Code Runners

Code Runners

Understanding Difference between Options API/Composition API

Options API uses data, method

```
<script setup>
  import {ref,createApp} from 'vue'

  const app = createApp(
    setup() {
      const message = ref('Hello vue!')
      const count = ref(0)

      return {
        message,
        count,
      }
    }
  )

  app.mount('#app')
</script>
```

Template Syntax

- Text Interpolation

```
<!-- #region snippet -->
<!-- Inside ./snippets/template-syntax/1.html -->
<p>{{ message }}</p>
<p>{{ count }}</p>
<!-- #endregion snippet -->
```

- Raw HTML

```
<!-- #region rawhtml -->
<!-- Inside ./snippets/template-syntax/1.html -->

<p v-html="message" ></p>

<!-- #endregion rawhtml -->
```

- Attribute Bindings using `v-bind:attributeName` or `:attributeName` - Using JavaScript Expressions - Directives

Attribute Bindings using `v-bind:attributeName` or `:attributeName`

```
<!-- using one object variable for multiple binding -->
<script setup>
  const multipleBinding = {
    id: 'unique-id',
    class: 'counter-button',
  };
</script>

<template>
  <div v-bind="multipleBinding">
    I am a div using multiple dynamic binding
  </div>
</template>
```

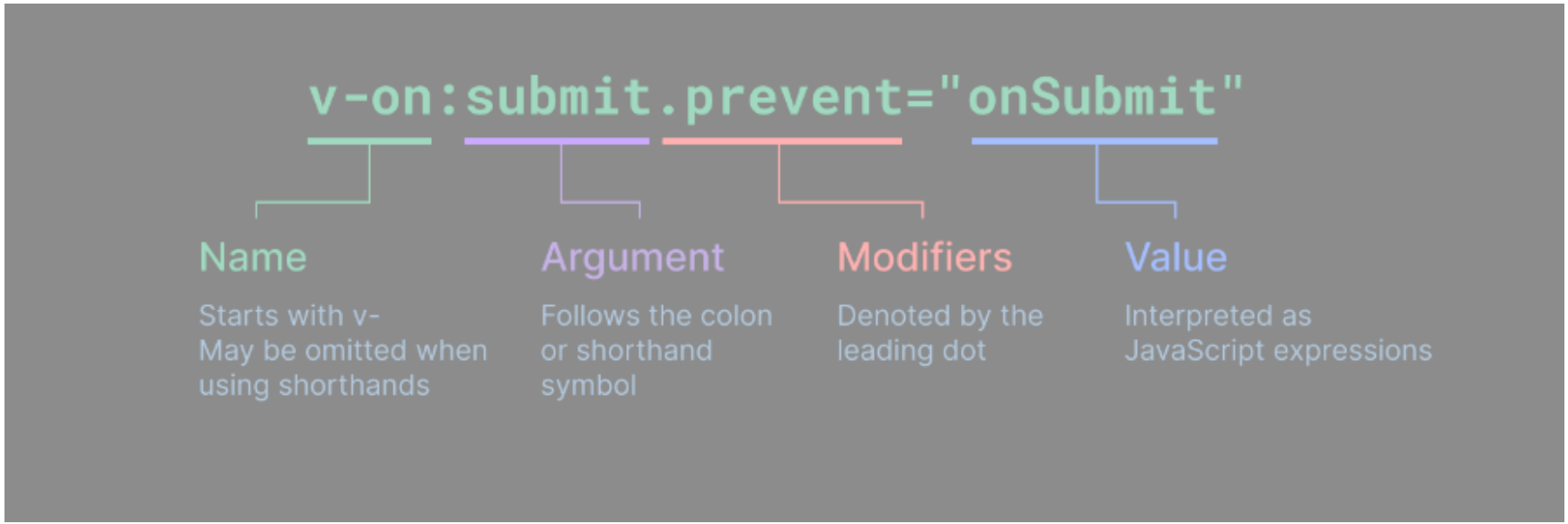
JavaScript Expressions in Expression Slots

```
<!-- #region snippet -->
<!-- Inside ./snippets/vue/expressions/1.vue -->
<script setup>
const message = 'Hello, Vue 3! ';
const count = 0;

// format date
function formatDate(date) {
  return date.toISOString();
}

</script>

<template>
  <h1>{{ message + 'AltSchool' }}</h1>
  <h3>{{ message.toUpperCase() }}</h3>
  <!-- we are allowed to do any JavaScript Expression -->
  <h2>{{message.repeat(3)}}</h2>
  <h2>{{message + 'AltSchool'.toUpperCase()}}</h2>
  <div>Cart: {{ count > 1 ? 'items' : 'item' }}</div>
  <!-- we are allowed to use function -->
  <div>Current time: {{ formatDate(new Date()) }}</div>
</template>
<!-- #endregion snippet -->
```

Directives v-*

```vue
<!-- #region snippet -->
<!-- Inside ./snippets/vue/directives/1.vue -->
<template>
  <span v-text="msg"></span> <span>{{msg}}</span>
  <div v-html="html"></div>

  <!--conditional rendering(v-show, v-if, v-else, v-else-if)-->
  <h1 v-show="ok">Hello!</h1> <!-- display: none -->
  <div v-if="type === 'A'"> <!-- can be used on template -->
  A
  </div>
  <div v-else-if="type === 'B'">
    B
  </div>
  <div v-else-if="type === 'C'">
    C
  </div>
  <div v-else>
    Not A/B/C
  </div>

  <!--List rendering(v-for)-->
  <div v-for="(item, index) in items"></div>
  <div v-for="(value, key) in object"></div>
```

Directives v-* Image



Directives using `v-name:argument.modifiers="value"`

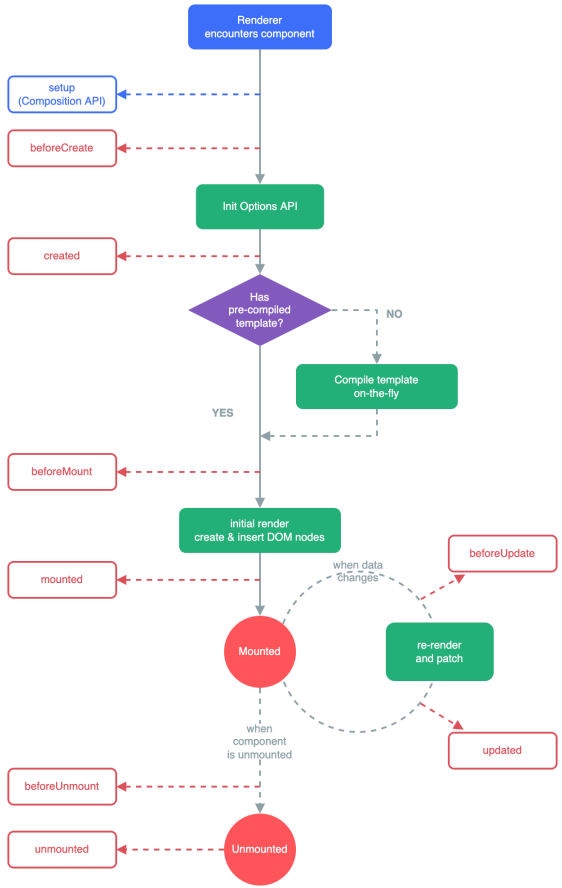
```
<!--v-cloak is used to hide raw templates until the component is ready-->
<div v-cloak>
  {{ message }}
</div>

<style scoped>
  [v-cloak] {
  display: none;
}
</style>
```

Vue Reactivity with `ref` and `reactive`

```
import { reactive, ref } from 'vue'

const state = reactive({
  apiData: null,
});

const stateRef = ref(null)

// it is recommended to use ref() as the primary API for declaring reactive state due to reactive limitations.
// Read more on ref and reactive https://vuejs.org/guide/essentials/reactivity-fundamentals.html#declaring-reactive-stat
```

Methods

LifeCycle hooks



LifeCycle hooks

- `onUpdated`
- `onUnmounted`

LifeCycle hooks `onMounted`

```
<script setup>
import { ref, onMounted } from 'vue'

const el = ref()

onMounted(() => {
  el.value // <div>
})
</script>

<template>
  <div ref="el"></div>
</template>
```

LifeCycle hooks `onUpdated`

```
<script setup>
import { ref, onUpdated } from 'vue'

const count = ref(0)

onUpdated(() => {
  // text content should be the same as current `count.value`
  console.log(document.getElementById('count').textContent);
});
</script>

<template>
  <button id="count" @click="count++">{{ count }}</button>
</template>
```

LifeCycle hooks `onUnmounted`

```
<script setup>
import { onMounted, onUnmounted } from 'vue'

let intervalId
onMounted(() => {
  intervalId = setInterval(() => {
    // ...
  })
})

onUnmounted(() => clearInterval(intervalId))
</script>
```

Read More

Conditional Rendering

In Vue.js, conditional rendering allows you to dynamically control which parts of your component's template are displayed based on certain conditions. This is a powerful feature that helps create interactive and responsive user interfaces.

We have:

- `v-if`
- `v-show`
- `v-else`
- `v-else-if`

V-if

V3 students, click me

Computed Properties

A computed property is used to declaratively describe a value that depends on other values.
Computed properties save you time and make your code cleaner by automatically reflecting changes in your data.

First Name: Enter your first name Last Name: Enter your last name

Full Name:

Check the ComputedProperties.vue component for the code.

Watchers

Hello, AltSchool

TODO:

Components

- 0 Jane Doe 0 + set computed

+Default Click Me

Component Details

- Global using `.js` and Local Registration using `.vue`
- Props
- Event
- V-model in Parent Child component(works in only input, textarea and select)
- Slots
- Provide / Inject

Read More about Dynamic Component and Async Component

TODO: create slide content for each of the sub topic on Components

Example of V-model in Parent and Child Component

Hello World!

My input Hello World!

Please make a change in the input notice what happens

Data Fetching using LifeCycle Hook `onMounted`

```
import { computed, reactive, onMounted, ref } from 'vue'

const state = reactive({
  apiData: null,
});

const stateRef = ref(null)

onMounted(
  () => {
    fetch('https://api.github.com/users/Oluwasetemi')
      .then((res) => res.json())
      .then((data) => {
        console.log(data)
        // set value to reactive state | ref
        state.apiData = data
        stateRef.value = data
      });
  }
)
```

Data Fetching using `onMounted` and `watchEffect`

```
import { watchEffect, reactive, onMounted, ref } from 'vue'

const username = ref(null)
const state = reactive({
  apiData: null,
});

onMounted(() => {
  watchEffect(
    () => {
      fetch(`https://api.github.com/users/${username}`)
        .then((res) => res.json())
        .then((data) => {
          console.log(data)
          // set value to reactive state | ref
          state.apiData = data
        });
    }
  )
})
```

Routing

```
import { createApp } from 'vue'

import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')
```

State Management

In Vue js, there are numerous ways of managing states. Although Vuejs shares data between multiple components through the use of props. The issue with this approach is that props are unidirectional. We can only pass data or props from parent to child to grandchild, but not the other way round. Even though Custom events try to fix this, it still has its own limitations as we can not pass data to other components with either of two, and this is where the following state management come to place

- Props
- Event Bus
- Simple State Management with Reactivity API
- Vuex or Pinia

1. Emitting and Listening to Event from Child to Parent

```
<script setup>
  const emit = defineEmits(['inFocus', 'submit'])

  function buttonClick() {
    emit('submit')
  }
</script>
```

Continuation: State Management - Composition API

2. Simple State Management with Reactivity API

```
<!-- ComponentB.vue -->
<script setup>
import { store } from './store.js'
</script>

<template>From B: {{ store.count }}</template>
```

Continuation: State Management - VUEX

3. Setting Up State Management Using Vuex

```
<script>
  import Vuex from 'vuex';
  export default new Vuex.Store({
    state: {
      itemCount: 0
    },
    mutations: {
      addItem(state) {
        state.itemCount++;
      },
      removeItem(state){
        if (state.itemCount > 0) {
          state.itemCount--;
        }
      }
    }
  })
</script>
```

Continuation: State Management- Pinia

```
<script>
  import {useCounterStore} from '@state/counter.js'

  const store = useCounterStore()

  console.log(store.count);
  store.increaseCount()
</script>
```

Todo App Store

```
<script setup>

export const useTodoListStore = defineStore('todoList', {
  state: () => ({
    todoList: [],
    id: 0
  }),
  actions: {
    addTodo(item) {
      this.todoList.push({ item, id: this.id++, completed: false })
    },
    deleteTodo(itemId) {
      this.todoList = this.todoList.filter((object) => {
        return object.id !== itemId
      })
    },
    toggleCompleted(idToFind) {
      const todo = this.todoList.find((obj) => obj.id === idToFind)
      if (todo) {
        todo.completed = !todo.completed
      }
    }
  }
```

Repo List Used For Teaching

- Vue AltSchool v3 teaching
- Pinia Setup

Using TypeScript With Vuejs

layout: two-cols

Left

This shows on the left

layout: two-cols

Left

This shows on the left.

Using TypeScript With Vuejs

- Typing Component Props
- Typing Component Emits
- Typing ref()
- Typing reactive()
- Typing computed()
- Typing Event Handlers
- Typing Provide / Inject
- Typing Template Refs
- Typing Component Template Refs

Contributors - Thank you all

- Chidinma Nwosu
- Kofoworola Shonuyi
- Segun Olawale & Abosede Racheal