Cheat Sheet for comprehensive Vue.js

Vue.js Basics

Installation

- **CDN**: `<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>`
- **NPM**: `npm install vue`
- **Vue CLI**: `npm install -g @vue/cli`

Creating a Vue Instance

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
});
```

Template Syntax

Interpolation

- **Text**: `{{ message }}`
- **Raw HTML**: `v-html="rawHtml"`
- **Attributes**: `v-bind:id="dynamicId"` (shorthand: `:id="dynamicId"`)

Directives

- **v-if**: `v-if="condition"`
- **v-for**: `v-for="item in items" :key="item.id"`
- **v-show**: `v-show="condition"`
- **v-bind**: `v-bind:class="{ active: isActive }"` (shorthand: `:class="{ active: isActive }"`)
- **v-on**: `v-on:click="handler"` (shorthand: `@click="handler"`)
- **v-model**: `v-model="message"`

Component Basics

Defining a Component

```
Vue.component('my-component', {
  template: '<div>{{ message }}</div>',
  data() {
    return {
      message: 'Hello from component!'
    };
  }
});
```

Using a Component

```
<my-component></my-component>
```

Props and Events

Props

- **Defining Props**: `props: ['title', 'likes']`
- **Using Props**: `<my-component title="Hello" :likes="42"></my-component>`

Events

- **Emitting Events**: `this.$emit('myEvent', data)`
- **Listening to Events**: `<my-component @myEvent="handler"></my-component>`

Lifecycle Hooks

- **beforeCreate**: Before the instance is initialized.
- **created**: After the instance is created.
- **beforeMount**: Before the instance is mounted to the DOM.
- **mounted**: After the instance is mounted.
- **beforeUpdate**: Before the instance is updated.
- **updated**: After the instance is updated.
- **beforeDestroy**: Before the instance is destroyed.
- **destroyed**: After the instance is destroyed.

Computed Properties and Watchers

Computed Properties

```
computed: {
  fullName() {
    return this.firstName + ' ' + this.lastName;
  }
}
```

Watchers

```
watch: {
  message(newVal, oldVal) {
    console.log(`Message changed from ${oldVal} to ${newVal}`);
  }
}
```

Vue Router

Installation

```
npm install vue-router
```

Basic Usage

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import Home from './components/Home.vue';
import About from './components/About.vue';

Vue.use(VueRouter);

const routes = [
  { path: '/', component: Home },
  { path: '/about', component: About }
];

const router = new VueRouter({
  routes
});

new Vue({
  router,
  render: h => h(App)
}).$mount('#app');
```

Vuex

Installation

```
npm install vuex
```

Basic Usage

```
import Vue from 'vue';
import Vuex from 'vuex';

Vue.use(Vuex);

const store = new Vuex.Store({
  state: {
    count: 0
  },
  mutations: {
    increment(state) {
      state.count++;
    }
  },
  actions: {
    incrementAsync({ commit }) {
      setTimeout(() => {
        commit('increment');
      }, 1000);
    }
  }
});

new Vue({
  store,
  render: h => h(App)
}).$mount('#app');
```

Tips and Tricks

Conditional Rendering

- **v-if vs v-show**: Use `v-if` for conditional rendering that may not need to be toggled frequently. Use `v-show` for elements that need to be toggled frequently.

Key Modifiers

- **Event Modifiers**: `@click.stop`, `@submit.prevent`, `@keyup.enter`

Dynamic Components

- **Keep-alive**: `<keep-alive><component :is="currentComponent"></component></keep-alive>`

Scoped Styles

- **Scoped CSS**: `<style scoped> .my-class { color: red; } </style>`

Advanced Features

Mixins

```
const myMixin = {
  created() {
    console.log('Mixin created');
  }
};

new Vue({
  mixins: [myMixin],
  created() {
    console.log('Component created');
  }
});
```

Custom Directives

```
Vue.directive('focus', {
  inserted(el) {
    el.focus();
  }
});
```

Plugins

```
const MyPlugin = {
  install(Vue, options) {
    Vue.mixin({
      created() {
        console.log('Plugin installed');
      }
    });
  }
};

Vue.use(MyPlugin);
```

Debugging and Tools

Vue Devtools

- **Browser Extension**: Install Vue Devtools for Chrome or Firefox.

- **Inspecting Components**: Use Vue Devtools to inspect component state, props, and events.

Error Handling

- **Global Error Handler**: `Vue.config.errorHandler = function (err, vm, info) { ... }`

Performance Optimization

Lazy Loading

```
const Home = () => import('./components/Home.vue');
const About = () => import('./components/About.vue');
```

Async Components

```
Vue.component('async-component', () => import('./AsyncComponent.vue'));
```

Optimizing Re-renders

- **Track by Key**: Use `:key` in `v-for` to optimize re-renders.

- **Computed Properties**: Use computed properties for complex logic to avoid unnecessary re-renders.

Conclusion

This cheat sheet covers the essential features, shortcuts, tips, and tricks for Vue.js. Use this as a quick reference to build efficient and maintainable Vue.js applications.

By Ahmed Baheeg Khorshid

ver 1.0