Vue.js Cheat Sheet

A concise reference for Vue.js, covering essential concepts, syntax, directives, and components.

Core Concepts & Syntax

Instance Creation

`new Vue({ options })`	Creates a new Vue instance. **Options:** • `el`: DOM element to mount to. • `data`: Data object. • `methods`: Methods object. • `computed`: Computed properties object. • `watch`: Watchers object. • `mounted`: Lifecycle hook (called after mounting).
Example	`new Vue({ el: '#app', data: { message: 'Hello Vue!' } })`

Data Binding

`{{ message }}`	Text interpolation (double curly braces).
`v-model`	Two-way data binding (forms).
`v-bind:attribute` or `:attribute`	Dynamically bind an attribute to an expression.
Example of v-model	`<input type="text" v-model="message"> <p>{{ message }}</p>`
Example of v-bind	`<img :src="imageURL">`

Directives

`v-if`, `v-else-if`, `v-else` - Conditional rendering.
`v-show` - Conditional display (toggles `display` CSS property).
`v-for` - Loop through arrays or objects.
`v-on:event` or `@event` - Listen to DOM events.

Components

Component Definition

`Vue.component('component-name', { options })`	Registers a global component. **Options:** • `template`: HTML template for the component. • `props`: List of props the component accepts. • `data`: (Function) Data for the component (must be a function that returns an object). • `methods`: Methods object. • `computed`: Computed properties object. • `mounted`: Lifecycle hook (called after mounting).
Example	`Vue.component('my-component', { template: '<div>A custom component!</div>' })`
Local Registration	Components can also be registered locally within another component's `components` option.

Props

`props: ['propName']`	Declares props that the component accepts.
Prop Types	`props: { propName: String, propNumber: Number, propBoolean: Boolean, propArray: Array, propObject: Object }`
Prop Validation	`props: { propName: { type: String, required: true, default: 'Default Value', validator: function (value) { return value.length > 0 } } }`

Emitting Events

`this.$emit('event-name', data)`	Emits a custom event from the component.

Example	`// In component method: this.$emit('custom-event', this.message)`
	
	`// In parent template: <my-component @custom-event="handleEvent"></my-component>`

Computed Properties & Watchers

Computed Properties

Define	`computed: { fullName: function () { return this.firstName + ' ' + this.lastName } }`
Usage	`{{ fullName }}` (in template)
Getter/Setter	`computed: { fullName: { get: function () { return this.firstName + ' ' + this.lastName }, set: function (newValue) { // ... } } }`

Watchers

Basic Watcher	`watch: { message: function (newValue, oldValue) { console.log('message changed', newValue, oldValue) } }`
Options	`watch: { message: { handler: function (newValue, oldValue) {}, deep: true, // for watching objects immediate: true // trigger immediately on creation } }`
When to use	Use watchers for asynchronous or expensive operations in response to data changes. Computed properties are better for synchronous value derivations.

Lifecycle Hooks

`beforeCreate`	Called before the instance is created.
`created`	Called after the instance is created.
`beforeMount`	Called before the element is mounted.
`mounted`	Called after the element is mounted.
`beforeUpdate`	Called before the data is updated.
`updated`	Called after the data is updated.
`beforeDestroy`	Called before the instance is destroyed.
`destroyed`	Called after the instance is destroyed.

Vue Router

Installation

```
npm install vue-router
```

Basic Configuration

Import	`import Vue from 'vue'`
	`import VueRouter from 'vue-router'`
	
	`Vue.use(VueRouter)`

Routes	`const routes = [ { path: '/home', component: HomeComponent }, { path: '/about', component: AboutComponent } ]`
	
	`const router = new VueRouter({ routes // short for `routes: routes` })`
	
	`new Vue({ router }).$mount('#app')`

Router Links

```
<router-link to="/home">Go to Home</router-link>
```

```
<router-view></router-view>
```

Dynamic Route Matching

Define	`const routes = [ { path: '/user/:id', component: User } ]`
Access	`// Inside User component: this.$route.params.id`