VUE3 CHEATSHEET FOR DEVELOPERS

Put together by your friends at learnvue.co

CREATING YOUR APP WITH VITE

Quick Vue3 development environment

```
npm init vite-app <project-name>
cd <project-name>
npm install
npm run dev
```

TEMPLATE SYNTAX

Text Interpolation Options

```
<span> {{ msg }} </span>
<span v-text='msg'></span>
```

Setting Inner HTML

```
<span v-html='rawHTML'></span>
```

Can use JS Expressions; NOT JS Statements

```
✔️ <span> {{ msg.reverse() }} </span>
❌ <span> {{ let msg = 'hi' }} </span>
```

DIRECTIVES

v-if	Puts el in DOM if true
v-else-if	Like a usual conditional
v-else	Like a usual conditional
v-show	Toggles display CSS value
v-text	Sets the inner text
v-html	Sets the inner HTML
v-for	Loop through an array/obj
v-on or @	Listens to DOM events
v-bind or :	Reactive updates attribute
v-model	Two way data binding
v-once	Sets val once; Never update

CONDITIONAL RENDERING

Add/Remove Element from DOM w/ Boolean

```
<div v-if='date == today'>...</div>
<div v-else-if='!done'>...</div>
<div v-else>...</div>
```

Toggles display CSS instead of editing DOM

```
<div v-show='date == today'>...</div>
```

HANDLING EVENTS

Capture and event and call a method

```
<div v-on:click='count'>Increase</div>
<!-- SHORTHAND -->
<div @click='count'>Increase</div>
```

Method is passed a Native DOM Cvent

```
const count = (event) => {
     console.log(event.target)
}
```

Event modifiers (usage: v-on:click.stop)

.stop	Stops event propagation
.once	Can only trigger event once
.prevent	Calls evt.preventDefault
.self	Don't send if target = child

LIST RENDERING

Basic Loop Over Array

```
<li v-for='item in items' :key='item'>
      {{ item }}
</li>
```

Loop and Track Index

```
<li v-for='(item, index) in items'>
      {{ index }} : {{ item }}
</li>
```

VUE3 CHEATSHEET FOR DEVELOPERS

Put together by your friends at learnvue.co

Loop Values in Object

```
<li v-for='obj in objects'>
      {{ obj }}
</li>
```

BINDING DATA

Simple Binding

```
<div v-bind:id='objectID'>...</div>
<!-- SHORTHAND -->
<div :id='objectID'>...</div>
```

Two way binding with data and input

```
<input v-model='email' />
```

Input Modifiers

`.lazy`	updates on change event
`.trim`	removes extra whitespace

Use Objects to Bind Class/Styles

```
<input :class='{error: hasError}' />
<input :style='{margin: space+"px"}' />
```

BIND DATA BETWEEN CHILD & PARENT

Use v-bind to pass data from parent to child and emit a custom event to send data back.

In Parent, Bind Data & Set Listener to Update

```
<custom :msg='s' @update='s = $event'/>
```

In Child, Send Back Using emit(event, data)

```
context.emit('update', 'hello world')
```

SLOTS

Slots allow for content injection from a parent component to a child component.

BASIC SLOTS

Child Component (MyButton.Vue)

```
<div>
   Hello World
   <slot></slot>
</div>
```

Parent Component

```
<my-button>
   This content will replace the slot
</my-button>
```

NAMED SLOTS

Useful when you have multiple slots. If unnamed, name is 'default'.

Child Component (MyButton.Vue)

```
<div>
   <slot name='top'></slot>
   <slot name='bottom'></slot>
</div>
```

Name Slots in the Parent Component

```
<my-button>
   <template v-slot:top> // ...
   </template>
   <template v-slot:bottom> // ...
   </template>
</my-button>
```

SCOPED SLOTS

Give parent component access to child data.

Child Component (MyButton.Vue)

```
<div>
   <slot v-bind:post='post'>
      {{ post.title }}
   </slot>
</div>
```

VUE3 CHEATSHEET FOR DEVELOPERS

Put together by your friends at learnvue.co

Parent Has Access to MyButton post data

```
<my-button>
   <template v-slot:default='slotData'>
      {{ post.author }}
   </template>
</my-button>
```

DYNAMIC COMPONENTS

Changes the rendered component - finds a registered component with the given name.

```
<component :is='componentName'/>
```

KEEP-ALIVE ELEMENTS

Stores a cached version of dynamic components when not visible. Avoids having to create a new component whenever toggled.

```
<keep-alive>
  <component :is='componentName'/>
</keep-alive>
```

COMPOSITION API

Everything returned by setup() is exposed to the template.

```
import { ref, reactive } from 'vue'
export default {
  setup(props, context) {
    const val = ref('example')
    const obj = reactive({ count: 0 })

    const evtHandler = () => {/*...*/}

    return {
       val, obj, evtHandler
    }
  }
}
```

SETUP() CONTEXT OBJECT PROPERTIES

attrs	Has component's attributes
slots	Has component's slots
emit	Function to emit events

VUEJS LIFECYCLE HOOKS

*beforeCreate	Use setup() instead
*created	Use setup() instead
onBeforeMount	Before mounting DOM
onMounted	DOM can be accessed
onBeforeUpdate	Reactive data changes
onUpdated	DOM has been updated
onBeforeUnmount	Component still complete
onUnmounted	Teardown complete

EXAMPLE LIFECYCLE HOOK CODE

```
import { onMounted } from 'vue'
// ...
setup() {
  onMounted(() => {
    console.log('component mounted!')
  }
}
```

VUE GLOBAL METHODS

mount()	Mount component to DOM
forceUpdate()	Force re-render
nextTick()	Runs func next update
destroy()	Destroy component/app

VUE3 CHEATSHEET FOR DEVELOPERS

Put together by your friends at learnvue.co

COMPUTED PROPERTIES

A computed property is a value that is calculated using one or more other properties.

```
setup() {
  const a = ref(1)
  const b = computed(() => a.value * 2)

  return { a, b }
}
```

WATCHEFFECT()

Listens to reactive dependencies and runs a method when one changes. Als runs on init.

```
 setup() {
   const site = ref('learnvue.co')

   watchEffect(() => {
     console.log(site.value)
   })

   return { site }
 }
```

TEMPLATE REFS

Give access to DOM elements.

```
// template
<div ref='example'> Example Div </div>
// script
setup() {
   const example = ref('learnvue.co')
   // wait for DOM to mount
   onMounted(() => {
     console.log(example.value)
   })

   return { example }
}
```

VUE OBJECT API OPTIONS

If you decide not to use the Composition API, your components will look similar to Vue2 with the Options API.

data()	Init reactive data
props	Data visible by parent
mixins	Declares mixins
components	Registers children
methods	Set of Vue methods
watchers	Watch values for change
computed	Cached reactive methods

TOP VUE LIBRARIES

vue-cli	Command Line Interface
vue-router	Handles Routing for SPAs
vuex	State Management Library

GREAT VUE UI RESOURCES

Vuetify	Bootstrap Vue	UIV
VueStrap	Vue Material	Mint UI
Element UI	Vuecidity	iView
Buefy	DeepReader	KeenUI
Quasar	AT UI	Vulma
Fish-UI	Muse UI	Vue Blu

CONTACT

For any corrections, comments, or concerns, just contact me at matt@learnvue.co

Hope this helped!