

VUEJS 2 PROGRESSIVE FRAMEWORK

BASICS, DIRECTIVES & INSTANCES

Created by @Manz (<https://twitter.com/Manz>)<https://lenguajejs.com/>

d Directives	<code><elem v-directive></elem></code>
BASIC DIRECTIVES	
v-text = <code>s</code> set.textContent (text) {v} use expression or data SHORTHAND v-html = <code>s</code> set.innerHTML (raw HTML) xss v-pre skip render <code>{vars}</code> (inside) v-cloak hide element until vue is ready v-once only render <code>{vars}</code> first time	
CONTROL DIRECTIVES	
v-show = <code>b</code> css-hide if false (always exists) v-if = <code>b</code> render content if true, else none v-else-if = <code>b</code> chained v-if condition 2.1.0+ v-else otherwise, render this content	
LOOPS DIRECTIVES	
v-for = <code>i in elem</code> iterate elem (for...in) v-for = <code>(v, i) in a</code> v=value, i=index v-for = <code>(v, k, i) in o</code> v=value, k=key, i=index v-for = <code>i in n</code> iterate from 1 to num v-for = <code>i in s</code> iter. from first to last char :key = <code>n s</code> hint for identify elements	
BINDING DIRECTIVES	
MODEL FORM BINDING (2-WAY)	
v-model = <code>s</code> bind on input/select/textarea v-model.lazy = <code>s</code> listen "change" event v-model.number = <code>n</code> cast to number v-model.trim = <code>s</code> strip left/right spaces	
HTML ATTRIBUTE BINDING	
v-bind = <code>o s</code> bind key:val on element attrs v-bind:attr = <code>s</code> bind "s" on attr (ex: alt) :attr = <code>s</code> binding var "s" on attr SHORTHAND :attr = <code>b</code> only show attr if true (ex: disabled) :attr.prop = <code>s</code> binding as DOM property :attr.camel = <code>s</code> kebab-c → camelCase 2.1.0+ :attr.sync = <code>s</code> expand into v-on handler 2.3.0+	
CLASS BINDING	INLINE EQUIVALENT
:class = <code>s</code> add string to class = <code>"string"</code> :class = <code>[a s]</code> add all to class = <code>"[s1, s2]"</code> :class = <code>[o b]</code> add key if true = <code>"[key:b]"</code>	
STYLE BINDING	INLINE EQUIVALENT
:style = <code>s</code> apply string styles = <code>"prop: val"</code> :style = <code>[o s]</code> apply k:v styles = <code>"{prop: val}"</code> :style = <code>[a o s]</code> apply objects = <code>"[o1, o2]"</code>	
EVENTS DIRECTIVES	
v-on = <code>o f</code> add n-events (key:func) 2.4.0+ v-on:event = <code>f</code> add event listener @event = <code>f</code> event listener SHORTHAND @event = <code>f(param, \$event)</code> inline @event.stop = <code>f</code> call .stopPropagation() @event.prevent = <code>f</code> call .preventDefault() @event.capture = <code>f</code> capture mode listen @event.self = <code>f</code> trigger only from this @event.once = <code>f</code> trigger at most once @event.keyCode = <code>f</code> trigger on key @event.sysKey = <code>f</code> trigger on key 2.1.0+ @event.native = <code>f</code> listen native event @event.button = <code>f</code> only mouse button 2.2.0+ @event.passive = <code>f</code> add passive event 2.3.0+	

{} Template Syntax	Templates
BASIC (MUSTACHE SYNTAX)	
<code>{{ var }}</code> show data (props, data...) <code>{{ bool ? 'Yes' : 'No' }}</code> ternary condition <code>{{ var + 's' }}</code> JS short expression	
ADVANCED (MUSTACHE SYNTAX)	
<code>{{ computed }}</code> complex expression <code>{{ var filter }}</code> apply filter over var	
f Custom Filters	CALLBACK VALUES <code>f(v)</code> prev. value <code>Vue.filter(s name, f filter_fn)</code>
i Vue Instance	vm = ViewModel
PROPERTIES	
o <code>vm.\$data</code> data vue observing o <code>vm.\$props</code> component props 2.2.0+	
READ-ONLY PROPERTIES	
o <code>vm.\$el</code> root DOM element o <code>vm.\$options</code> instance custom options v <code>vm.\$parent</code> parent instance v <code>vm.\$root</code> root instance (or itself) a <code>vm.\$children</code> direct child instances o <code>vm.\$slots</code> distribution slots o <code>vm.\$scopedSlots</code> scoped slots 2.1.0+ o <code>vm.\$refs</code> elems registered w/ref attrs b <code>vm.\$isServer</code> server-side instance SSR o <code>vm.\$attrs</code> parent-scope attrs bindings o <code>vm.\$listeners</code> parent-scope v-on ev. l.	
METHODS	
f <code>vm.\$watch(f exp, f fn, o options)</code> * <code>vm.\$set(o a tgt, s n key, * v)</code> * <code>vm.\$delete(o a tgt, s n key)</code>	
METHODS (CUSTOM EVENTS)	
vm.\$on(s a ev, f fn) listen c. ev. vm.\$once(s ev, f fn) only once vm.\$off(s a ev, f fn) remove c. ev. vm.\$emit(s name, a args) trigger	
METHODS (LIFECYCLE)	
v <code>vm.\$mount(e s el)</code> manual mount vm.\$forceUpdate() force to re-render vm.\$nextTick(f fn) next DOM update vm.\$destroy() destroy vue instance	
e Custom Events	Communication
EMIT CUSTOM EVENT FROM CHILD TO PARENT	
emits custom event w/data to parent: <code>this.\$emit('myevent-kebab-case')</code>	
CAPTURE CUSTOM EVENT ON PARENT	
capture custom event from child: <code>@myevent="methodHandler"</code>	
KEYCODE ALIAS enter up tab down delete left esc right space custom	SYSKEY ctrl alt shift meta BUTTON left middle right WATCH OPTIONS exact bdeep nested bimmediate

s Special Tags & Attrs	<code><elem attr></elem></code>
SPECIAL ATTRIBUTES	
ref = <code>s</code> register elem or child component slot = <code>s</code> name of component slot slot-scope = <code>s</code> scoped slot (expr) 2.5.0+ key = <code>n s</code> hint for vDOM identify node	
SPECIAL TAGS	
<component> meta-component (dynamic) is = <code>s o</code> used in dynamic components inline-template include templates inline <keep-alive> cache comp. w/o destroy include = <code>s</code> only cache match names 2.1.0+ exclude = <code>s</code> not cache match names 2.1.0+ max = <code>n</code> maximum comp. to cache 2.5.0+ <slot> serve as content distribution slot name = <code>s</code> name slot	
t Vue Transitions	Transition & Animations
SINGLE TRANSITIONS	
<transition> transition effect for elem/comp. name = <code>s</code> auto expand name (def: "v") appear apply transition on init render css = <code>b</code> apply css transition classes type = <code>s</code> set "transition" or "animation" mode = <code>s</code> set timing "out-in" or "in-out"	
CLASSES FOR APPLY TRANSITION CLASS	
enter *-class *-to-class *-active-class leave *-class *-to-class *-active-class appear *-class *-to-class *-active-class	
EVENTS	
enter before-* after-* *-cancelled leave before-* after-* *-cancelled appear before-* after-* *-cancelled	
MULTIPLE TRANSITIONS	
<transition-group> trans. effect for multiple tag= <code>s</code> convert to html tag move-class = <code>s</code> overwrite during trans. same trans. props (except mode) & events	
d Custom Directives	Create your v-some!
GENERAL SYNTAX (GLOBAL)	
Vue.directive(...) ① <code>(s name, o f hooks)</code> global register ② <code>(s name, f bind_and_update_fn)</code> SHORTHAND	
HOOK METHODS	
.bind : <code>f</code> when bound to element (only once) .inserted : <code>f</code> when inserted into parent node .update : <code>f</code> after content has updated .componentUpdated : <code>f</code> same, and children .unbind : <code>f</code> when unbound from elem (once)	
HOOK METHODS ARGUMENTS	
e el when bound to element (only once)	
HOOK METHODS ARGUMENTS (READ-ONLY)	
o binding data object with... s .name directive name, w/o v- prefix * .value directive value, computed * .oldValue prev value (update hooks) s .expression value as string s .arg argument (ex: v-sample:foo) o modifiers mods list (ex:v-sample.foo)	
e vnode virtual node produced by vue e oldVnode prev virtual node (update hooks)	

n number	c component	a array	o object
s string	d date	s symbol	undefined
b boolean	r regexp	t datatype	e HTMLElement
b def: true	f function	* any	v Vue instance

VUEJS 2 PROGRESSIVE FRAMEWORK

COMPONENTS & SINGLE FILE COMPONENTS

Created by @Manz (<https://twitter.com/Manz>)<https://lenguajejs.com/>

c Vue Components Create your component

GENERAL SYNTAX (GLOBAL)

`Vue.component(...)`
 ① `($id, o def)` register a global component
 ② `($id)` retrieve a registered component

DEFINITION / NAME

`.name: s` set/change name to component

DEFINITION / DOM OPTIONS

`.el: s` e existing DOM element to mount on
`.template: s` string template as markup
`.render: ce` alternative with render functions
`.renderError: ce` alt. output for errors 2.2.0+

DEFINITION / DATA OPTIONS

`.props: a s o` attrs exposed from parent
`.propsData: o` pass props to an instance
`.data: f` data object (reactive). Return a `o`
`.computed: o f` cached computed data
`.methods: o f` methods object data
`.watch: o f` observe for changes (old, new)

DEFINITION / LIFECYCLE HOOKS

`.beforeCreate: f` before data observation
`.created: f` after instance is created
`.beforeMount: f` before mount step
`.mounted: f` after mount step
`.beforeUpdate: f` before DOM patched
`.updated: f` after vDOM re-render & patch
`.activated: f` kept-alive c. activated
`.deactivated: f` kept-alive c. deactivated
`.beforeDestroy: f` before instance destroy
`.destroyed: f` after instance destroyed
`.errorCaptured: f` (err, vm, info) 2.5.0+

DEFINITION / ASSETS

`.directives: o` name-directive hash
`.filters: o` name-filter hash
`.components: o` name-component hash

DEFINITION / COMPOSITION

`.parent: v` set parent-child relationship
`.mixins: a o` merge objects (priority: own)
`.extends: o` extends from other component
`.provide: o` data inject into descendants 2.2.0+
`.inject: a o` retrieve data from ancestor 2.2.0+

DEFINITION / MISC

`.delimiters: a s` set delimiters `['{{', '}}']`
`.functional: b` set stateless & instanceless
`.model: o s` custom prop/event v-model 2.2.0+
`.inheritAttrs: b` set parent-scope inherit 2.4.0+
`.comments: b` preserve comments 2.4.0+

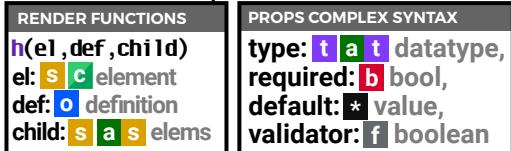
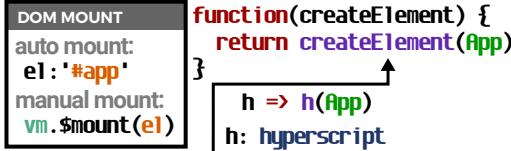
x Reuse Components Mixins & Extends

REUSABILITY & COMPOSITION (GLOBAL)

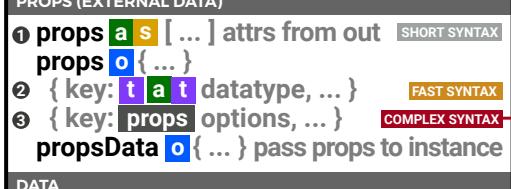
`Vue.extend(o def)` create "subclass"
`Vue.mixin(o mixin)` apply & merge mixin

INITIAL SNIPPET FOR MIXINS

```
import { m1 } from './mixins/m1.js';
export default {
  mixin: [m1]
  export const = {...}
```



o Data Options Main data options



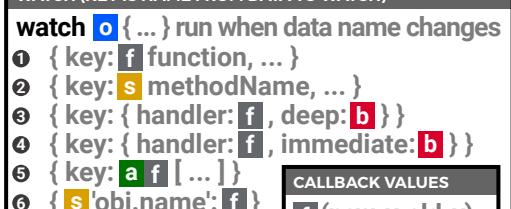
COMPUTED DATA (CACHED)



METHODS (FUNCTIONS)



WATCH (KEY IS NAME FROM DATA TO WATCH)



EXAMPLE SNIPPET FOR WATCH

```
① a: (v,old) => {...};
② b: 'fnName',
③ c: { handler: (v,old) => {...}, deep: b }
④ 'd.a': (v,old) => {...}
```

A Global API Other main options

ASYNC UPDATE QUEUE

`Vue.nextTick(f fn, o ctx)` defer exec func. to next DOM update cycle (wait changes)
 ① `.nextTick(f fn)` normal use
 ② `.nextTick().then(f fn)` w/promise 2.1.0+

SET/UNSET REACTIVE DATA

`Vue.set(o a tgt, s n key, * v)` add data
`Vue.delete(o a tgt, s n key)` del data

OTHERS

`Vue.use(o f plugin)` install Vue plugin

`Vue.compile(s tpl)` compile to render func.

`s` `Vue.version` return Vue version string

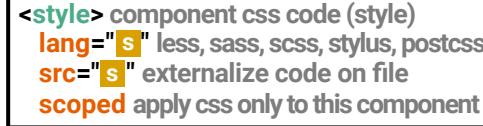
v Vue Create new Vue instance



s Single File Components .vue files



STYLES (CSS CODE)



SCRIPT (JAVASCRIPT CODE)



INITIAL SNIPPET FOR SFC COMPONENT

```
import File from './File';
① export default { ... }
② export default Vue.extend({
  name: 'file',
  components: {
    'file-component': File
})
```

s Structure Vue File Structure

INITIAL FILE & FOLDER STRUCTURE

<code>src</code>	// source files
<code>assets</code>	// dynamic assets
<code>components</code>	// ui components
<code>mixins</code>	// reused defs
<code>router</code>	// routing data
<code>store</code>	// vuex, flux, ...
<code>translations</code>	// i18n translations
<code>utils</code>	// helpers
<code>views</code>	// view components
<code>main.js</code>	// app entry file
<code>App.vue</code>	// main app component

c Vue.config Configuration API

PROPERTIES

`.optionMergeStrategies o f` (par, child, vm)
`.ignoredElements a s` customElems
`.keyCode o n` define keycode aliases

ERRORS/WARNINGS HANDLER

`.ErrorHandler f` (err, vm, info 2.2.0+)
`.warnHandler f` (msg, vm, trace) 2.4.0+

FLAGS

`.silent b` supress all vue logs/warnings
`.devtools b` allow vue devtools inspect
`.performance b` monitoring vue perf.
`.productionTip b` show tips on startup

