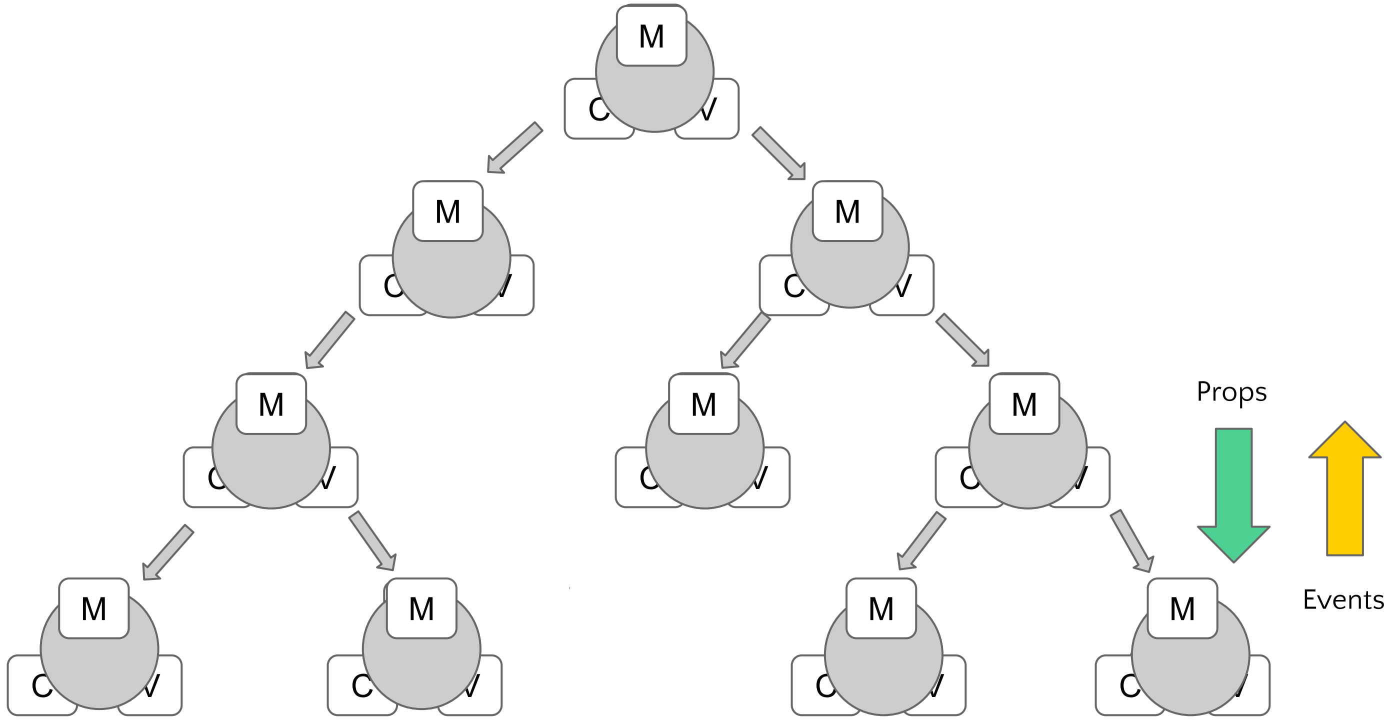
Curso de Vue.js - Clase 4

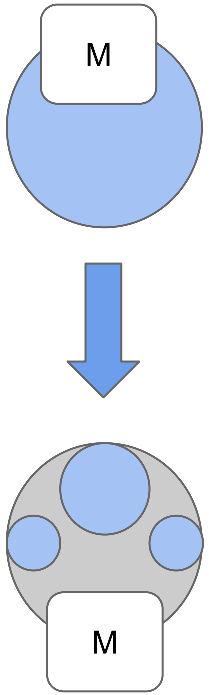
Vuenas tardes! 🖖

FICTIZIA

Componetización

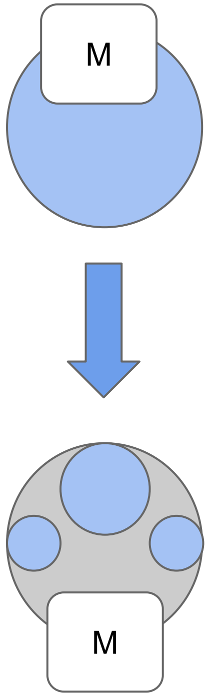


Slots



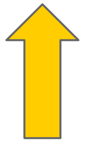
- <slot>
- Fallback content
- Named slots
- Scoped slots
- Destructuring slot props

Slots

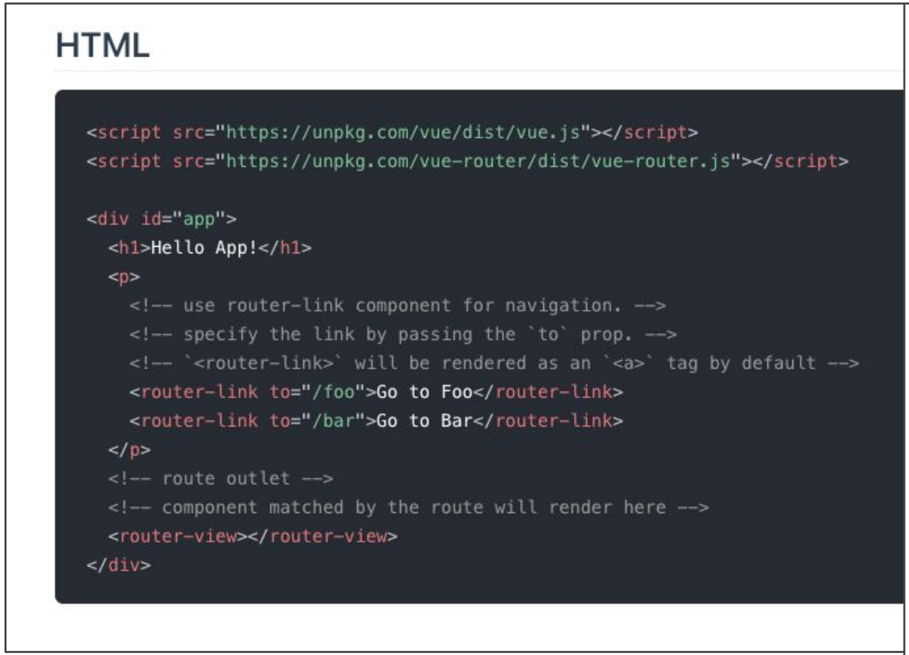


v-slot / #default

#default=”{ user }”



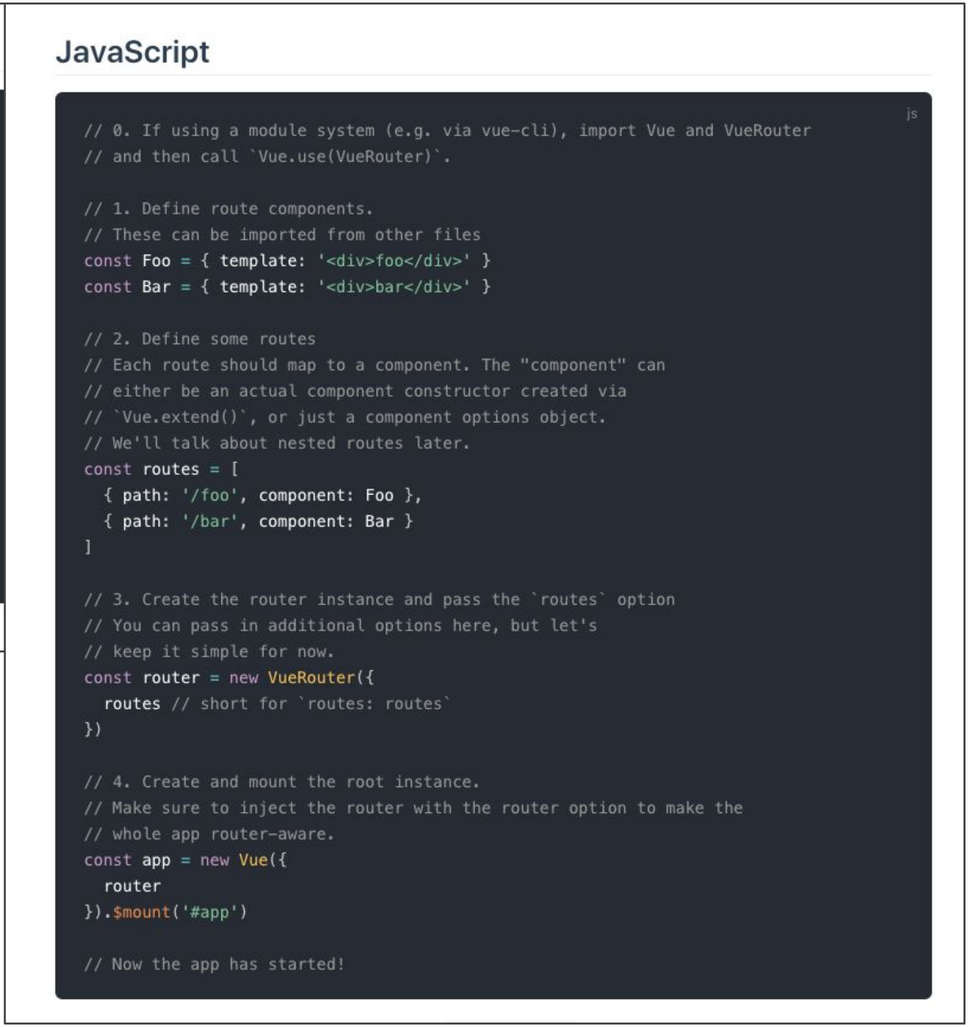
VUE ROUTER



HTML

```html
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>

<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- use router-link component for navigation. -->
    <!-- specify the link by passing the `to` prop. -->
    <!-- `<router-link>` will be rendered as an `<a>` tag by default -->
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <!-- route outlet -->
  <!-- component matched by the route will render here -->
  <router-view></router-view>
</div>
```



JavaScript

```js
// 0. If using a module system (e.g. via vue-cli), import Vue and VueRouter
// and then call `Vue.use(VueRouter)`.

// 1. Define route components.
// These can be imported from other files
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. Define some routes
// Each route should map to a component. The "component" can
// either be an actual component constructor created via
// `Vue.extend()`, or just a component options object.
// We'll talk about nested routes later.
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

// 3. Create the router instance and pass the `routes` option
// You can pass in additional options here, but let's
// keep it simple for now.
const router = new VueRouter({
  routes // short for `routes: routes`
})

// 4. Create and mount the root instance.
// Make sure to inject the router with the router option to make the
// whole app router-aware.
const app = new Vue({
  router
}).$mount('#app')

// Now the app has started!
```

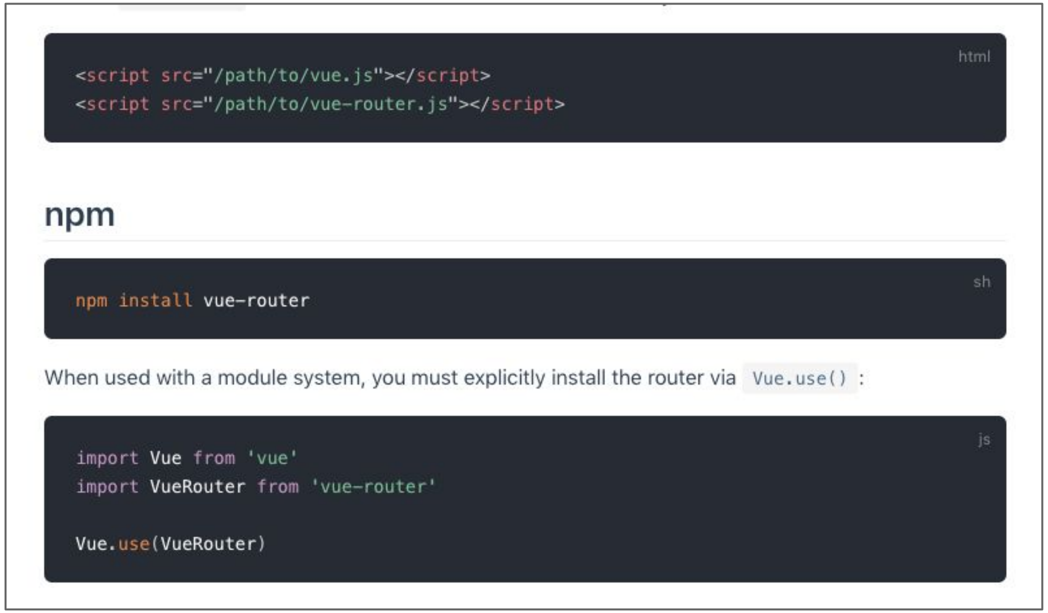
¿Por qué un router?

- Navegación adelante/atrás
- Refresh
- Compartir urls con vistas

Vue Router

- <router-view>
- <router-link to="">
- routes:[]
  - path
  - component

- npm install vue-router
- vue add router



html
```html
<script src="/path/to/vue.js"></script>
<script src="/path/to/vue-router.js"></script>
```

npm

sh
```sh
npm install vue-router
```

When used with a module system, you must explicitly install the router via `Vue.use()` :

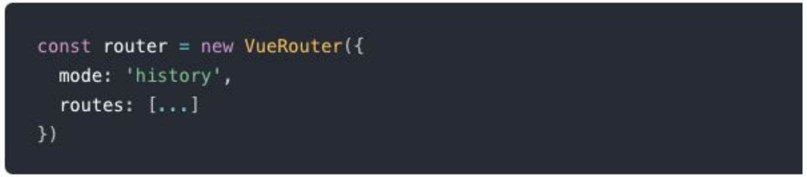
js
```js
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter)
```

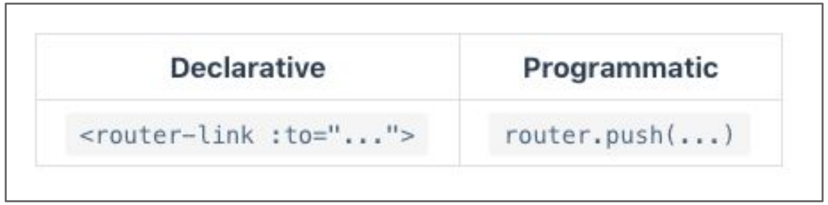
Programmatic Navigation

- HTML5 History Mode

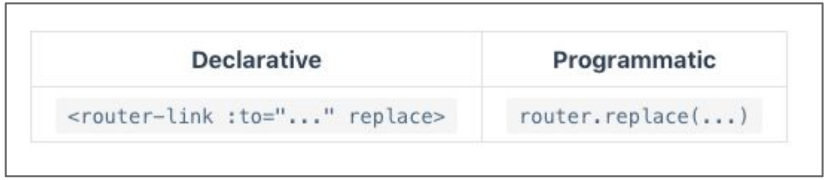
- router.push()
  - path
  - name
  - params
  - query



```
const router = new VueRouter({
  mode: 'history',
  routes: [...]
})
```



Declarative	Programmatic
`<router-link :to="...">`	`router.push(...)`

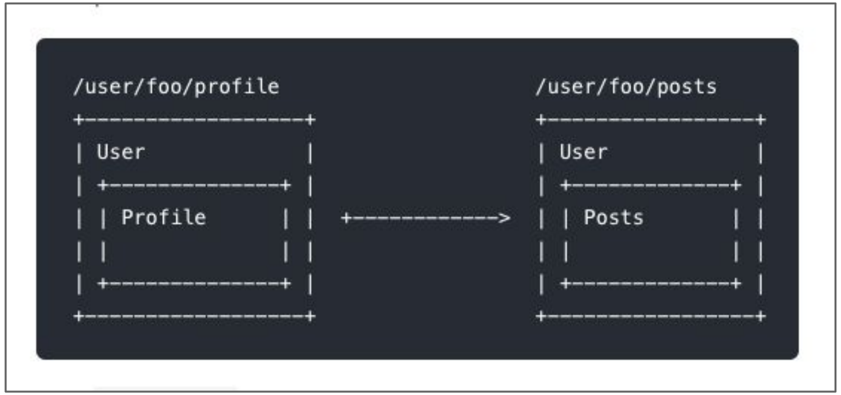


Declarative	Programmatic
`<router-link :to="..." replace>`	`router.replace(...)`

Dynamic Route Matching

- Ruta puede tener name
- Dynamic Route Matching
  - /pokemons/:id
- Nested routes
  - Children
- Named Views: components:{}

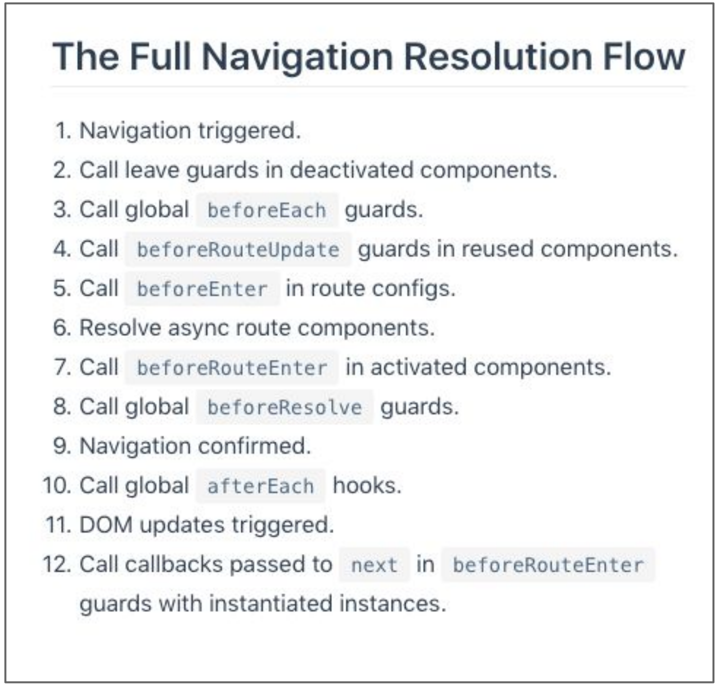
- React to params:
  - Watch: ‘$route’
  - beforeRouteUpdate



```
/user/foo/profile                     /user/foo/posts
+------------------+                  +------------------+
| User             |                  | User             |
| +--------------+ |                  | +--------------+ |
| | Profile      | |  +------------>  | | Posts        | |
| |              | |                  | |              | |
| +--------------+ |                  | +--------------+ |
+------------------+                  +------------------+
```

Navigation Guards

- Global before Guards
  - router.beforeEach((to, from, next) =>...
  - next(), next(false), next(‘route’)
- Per-Route Guard
  - beforeEnter: (to, from, next) =>...
- In-Component Guards
  - beforeRouteEnter
  - beforeRouteUpdate
  - beforeRouteLeave



The Full Navigation Resolution Flow

1. Navigation triggered.
2. Call leave guards in deactivated components.
3. Call global `beforeEach` guards.
4. Call `beforeRouteUpdate` guards in reused components.
5. Call `beforeEnter` in route configs.
6. Resolve async route components.
7. Call `beforeRouteEnter` in activated components.
8. Call global `beforeResolve` guards.
9. Navigation confirmed.
10. Call global `afterEach` hooks.
11. DOM updates triggered.
12. Call callbacks passed to `next` in `beforeRouteEnter` guards with instantiated instances.

Additional

- Params as Props
- Additional props
- Meta information

- Redirect & Aliases

- Transitions

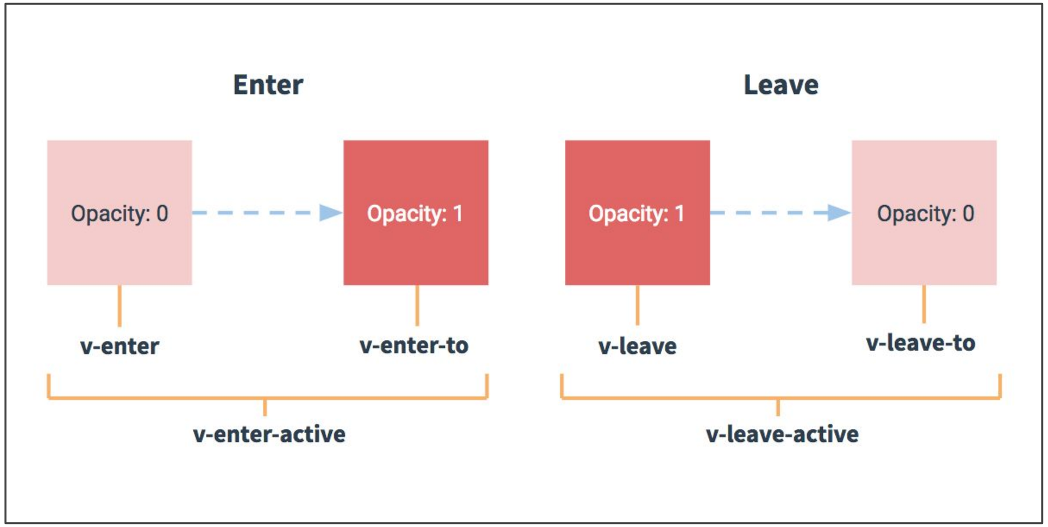
- scrollBehavior

Nos Vuemos! 🖖

Transitions

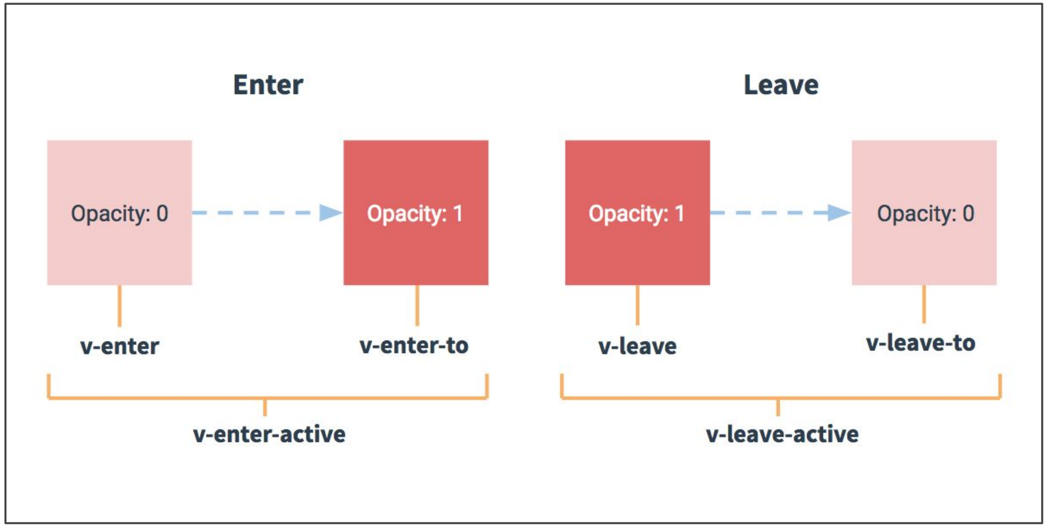
- v-enter
- v-enter-active
- v-enter-to

- v-leave
- v-leave-active
- v-leave-to



Transition Group

- tag
- v-move


Enter
Leave
Opacity: 0
Opacity: 1
Opacity: 1
Opacity: 0
v-enter
v-enter-to
v-leave
v-leave-to
v-enter-active
v-leave-active