Essentials

Getting Started
createRouter()
const routes = [{
path: 'value',
component: Comp...
<RouterLink>
<RouterView>
const router = useRouter()
router.push()
const route = useRoute()

Dynamic Routes
const routes = [
{ path: '/users/**:id**', ...
{ path: '/user/**:id**/posts/**:postId**', ...

<RouterLink :to=`/user/**${id}**`>
router.go(`/user/**${id}**`)
onBeforeRouteUpdate(async (to, from) => {
fetchUser(**route.params.id**)
})
watch(
() => **route.params.id**,
fetchData,
{immediate: true } // eager watcher
)

Routes' Matching Syntax
... path: '/:orderId(\\d+)'
... path: '/chapter*', ...
... path: '/chapter+', ...
... path: '/chapter?', ...

sensitive: true // case sensitive
strict: true. // ending / is strict

Named Routes
const routes = [{ name: 'value',
<RouterLink :to={ name: 'value' ...
router.push({ name: 'value', ...

Nested Routes
const routes = [{
path: '/user/:id',
component: Comp,
children: [{
path: 'profile',
component: Comp...

Programmatic Navigation
router.push({ path: '/users'})
router.replace({ name: 'users'}) // no history entry
router.go(1-)
router.back()
router.forward()

Named Views
<RouterView name="name1">
<RouterView name="name2">
const routes = [{
path: "value",
components: {
name1: Comp1,
name2: Comp2...

Redirect and Alias
redirect- URL changes to redirect, match new URL
const routes = [{
path: '/home', redirect: '/' }]

alias - URL changes to alias, matches original path
const routes = [{
path: '/', component: Home, alias: '/home' }]

Passing Props
// Using $route.params limits use of component
const routes = [{ path: '/user/:id', ... }]
{{ $route.params.id }}

// Setting param as a prop makes component more versatile
const routes = [{ path: '/user/:id', ... , **props: true** }]
defineProps({ id: String })]
{{ id }}

const routes = [{ path: '/user/:id', ... ,
props: { showUsername: false} }]
Joe

Active Links
RouterLink (in list) is active if:
link's to path matches current location
links params match current location params

2 classes:
router-link-active //
router-link-exact-active

<RouterLink
activeClass='...'
exactActiveClass='...'

Advanced

Navigation Guards
Functions run before or after route changes
May be defined globally, in a component, or per route in router def

global:
3. router.beforeEach((to, from, next) => { ... })
7. router.beforeResolve((to) => { ... })
9. router.afterEach((to, from, **failure**) => { ... }) // hook - not guard

per route:
// triggered when new route is matched
5. beforeEnter: (to, from) => { ... }
beforeEnter: [foo1, foo2]

in-component:
2. onBeforeRouteLeave(to, from) { ... } // e.g. confirm dialog
4. onBeforeRouteUpdate(to, from) { ... } // e.g. fetch if param changed

The Full Navigation Resolution Flow
1. Navigation triggered.
2. Call **beforeRouteLeave** 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 global **beforeResolve** guards.
8. Navigation is confirmed.

9. Call global **afterEach** hooks.
10. DOM updates triggered.

Route Meta Fields
const routes = [{ path: '/login', meta: { requiresAuth: true }, ...
route.meta //contains all meta fields from all matched routes

Data Fetching

After navigation:
watch(() => { route.params.id, fetchData, {immediate: true })

Before navigation
onBeforeRouteUpdate((to, from) => { ... }) // e.g. fetch if param changed

RouterView slots
// put a TemplateRef on a component placed in a router-view

```
<router-view v-slot="{ Component }">
  <component :is="Component" ref="mainContent" />
</router-view>
```

Transitions

Scroll Behavior

Lazy Loading Routes

Typed Routes

Extending RouterLink

Navigation Failures

Dynamic Routing