

TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE:

- Transiciones.
- Rutas lazy loading.
- Transiciones a librería externa.

TRANSICIONES:

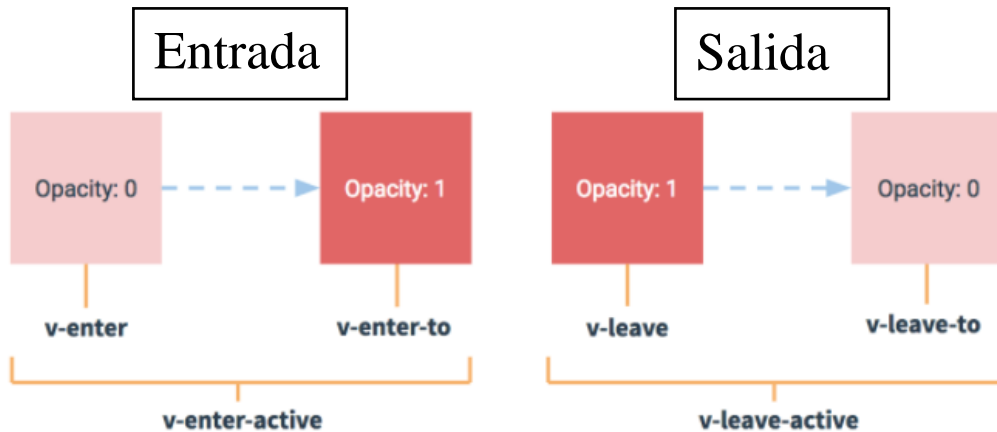
Vue ofrece una variedad de formas para aplicar efectos de transición cuando los elementos son insertados, actualizados. o eliminados del DOM.

Para realizar esta tarea, nos provee clases para aplicarlas.

CLASES DE TRANSICIONES EN VUE:

Hay 6 clases aplicadas para transiciones de entrada/salida.

1. **v-enter:** estado inicial para entrada. Aplicada antes que el elemento sea insertado, y se elimina después de un frame.
2. **v-enter-active:** estado activo, y de finalización para entrada. Aplicada antes que el elemento sea insertado, y se elimina cuando la animación/transición finaliza.
3. **v-enter-to:** estado final para entrada. Agregado un frame, después que el elemento es insertado (Al mismo tiempo se elimina v-enter).
4. **v-leave:** estado inicial para salida. Aplicada cuando la transición de salida es activa, y se elimina después de un frame.
5. **v-leave-active:** estado activo, y de finalización para salida. Aplicada cuando la transición de salida es activada, y se elimina cuando la animación/transición finaliza.
6. **v-leave-to:** estado final para salida. Agregado un frame después que la transición de salida ha sido gatillada (al mismo tiempo, v-leave es removido), y es removido cuando la transición/animación finalizó.



RUTAS LAZY LOADING:

Al construir nuestra aplicación con un empaquetador, el código Javascript generado puede volverse un poco largo, y esto afectará cuando la página cargue por primera vez, ya que toda nuestra aplicación estará compactada en un solo archivo.

Esto podría ser más eficiente si fuese posible dividir cada ruta en un archivo separado, y solo ser cargado cuando la ruta específica sea visitada.



Cuando instalamos nuestro proyecto con vue/cli, incluyendo router, nos añade 2 vistas de ejemplo, una de éstas muestra el uso de Lazy Loading.

```

1  {
2    path: '/about',
3    name: 'About',
4    // route level code-splitting
5    // this generates a separate chunk (about.[hash].js) for this
6    route
7    // which is lazy-loaded when the route is visited.
8    component: () => import(/* webpackChunkName: "about" */
9    '../views/About.vue')
10 }
  
```

Si nos damos cuenta, la forma de importar el componente es diferente, pues al hacer el import, éste agrega un "webpackChunkName": "about".

Cuando visitamos la ruta about, se inserta un archivo llamado about.js, aliviando el tiempo de carga adicional.

Name	Status	Type	Initiator	Size	Time	Waterfall	
 about.js	200	script	app.js:919	(pref...	6 ms		

AGRUPANDO CHUNKS:

Como vimos, el “webpackChunkName” nos sirve para indicar el nombre del archivo, que será generado cuando se visite dicha ruta.

Pero también, allí podemos definir un “webpackChunkName” para más de una ruta.

```
const Foo = () => import(/* webpackChunkName: "group-foo" */ './Foo.vue')
const Bar = () => import(/* webpackChunkName: "group-foo" */ './Bar.vue')
const Baz = () => import(/* webpackChunkName: "group-foo" */ './Baz.vue')
```