



Tweaks útiles en VueJS

**Network Status Based UI / UX, Image
Optimization, Rendering Performance y
Animations**

Introducción

- Vue destaca por: Simplicidad y pocas restricciones + gran comunidad. (grande en cuánto a calidad que no a tamaño)
- Buena base con todo lo necesario (para montar un proyecto bastante completo) y muy ampliable con libertad.
- Cantidad de tweaks tanto de performance cómo visuales o de ahorro de recursos.

Algunos tweaks útiles (hay muchos más)

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- 4. One Line Animations

Índice

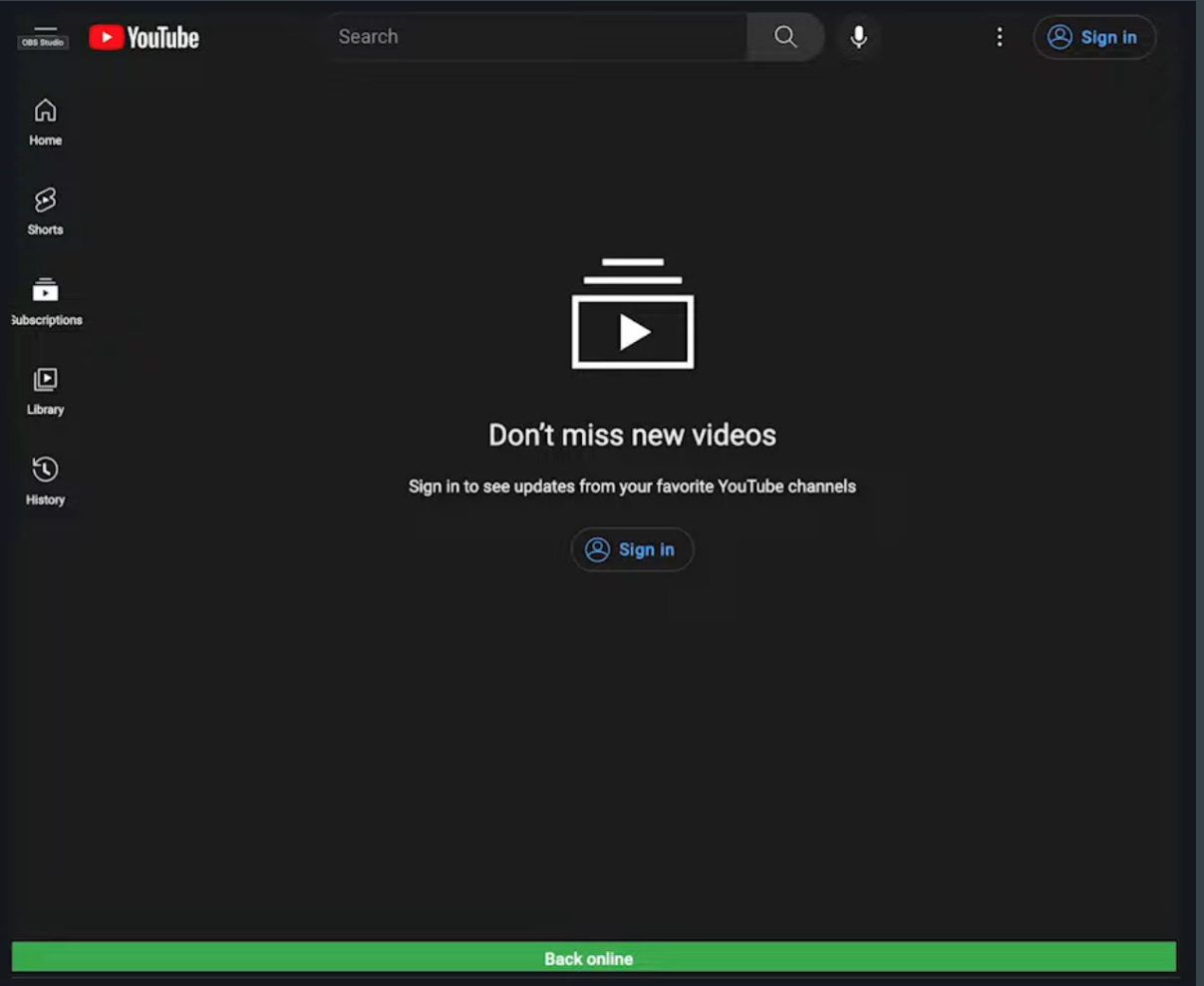
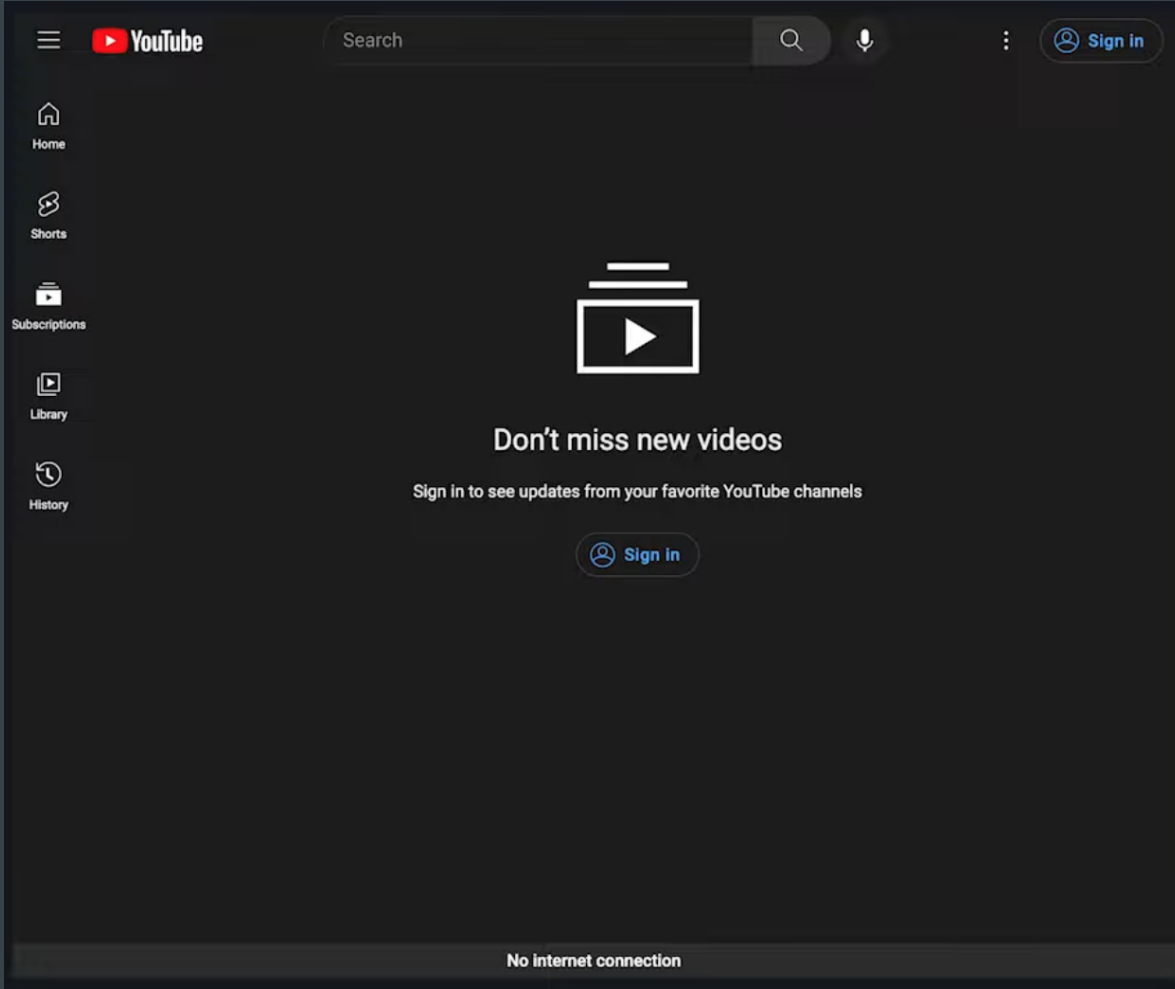
- **1. *Network Status Based UI***
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- 4. One Line Animations

1. Network Status Based UI / UX

UI/UX basada en el estado de la red

Network Status Based UI / UX

- Ideales en situaciones en las que tenemos una conexión a internet lenta / inestable (mercados emergentes).
- Útil para informar a los usuarios del estado de la conexión
- Muy utilizado en sitios de streaming (regulación del bitrate), o de edición de documentos onLine.
- Nota: Las funciones (useNetwork, useOnline) de **VueUse** ya implementan esto.
- Veremos cómo hacerlo VanillaJS cómo excepción



< Quality

1080p^{HD}

720p

480p

360p

240p

144p

✓ Auto

Ejemplo: Mostrar info conexión

- Nos apoyamos en window.navigator:

```
const navigator = window.navigator; // Info ab. user-agent

const isOnline = ref(navigator.onLine);

// Listen for events of the navigator interface to detect NW changes
window.addEventListener("online", () => {
    isOnline.value = true;
});
window.addEventListener("offline", () => {
    isOnline.value = false;
});
```

- Utilizamos isOnline en nuestra UI para adaptarla

Ejemplo: Enviar versión lightweight de recursos de nuestra app

- Nos apoyaremos en "Network Information API" (experimental), aporta información adicional sobre la red del usuario.
- Ejemplo de propiedades de la instancia:
 - `NetworkInformation.downlink`
 - `NetworkInformation.rtt`
 - `NetworkInformation.saveData`
 - `NetworkInformation.effectiveType`
 - etc
- Referencia: [MDN: NetworkInformation](#)

- `NetworkInformation.downlink` -> Estimación del ancho de banda en mbps

`NetworkInformation.downlink`

Read only

Returns the effective bandwidth estimate in megabits per second, rounded to the nearest multiple of 25 kilobits per seconds.

- `NetworkInformation.rtt` -> Round Trip Time (tiempo en ms que un paquete de datos tarda en volver a su emisor habiendo pasado por su destino)

`NetworkInformation.rtt`

Read only

Returns the estimated effective round-trip time of the current connection, rounded to the nearest multiple of 25 milliseconds.

- `NetworkInformation.saveData` -> Si el cliente tiene activada esta opción
- La opción `saveData` se activa a veces autom. al hacer wifi tether.

`NetworkInformation.saveData`

Read only

Returns `true` if the user has set a reduced data usage option on the user agent.

- `NetworkInformation.effectiveType` -> Tipo de red del cliente (estimado según otros valores)

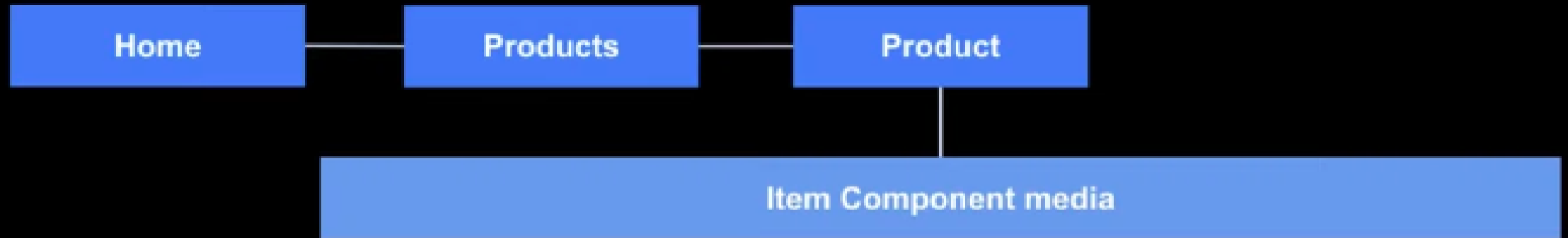
`NetworkInformation.effectiveType`

Read only

Returns the effective type of the connection meaning one of 'slow-2g', '2g', '3g', or '4g'. This value is determined using a combination of recently observed round-trip time and downlink values.

- Basándose en estos datos, se lleva a cabo el "Adaptive Loading", en el que no sólo se tiene en cuenta el tamaño de pantalla para decidir qué servir, sino que también se tiene en cuenta la red y Hardware del dispositivo del usuario.
- Se tienen en cuenta también preferencias del usuario, cómo por ejemplo el modo ahorro de datos, para servir assets más o menos grandes.

Adaptive Media Loading

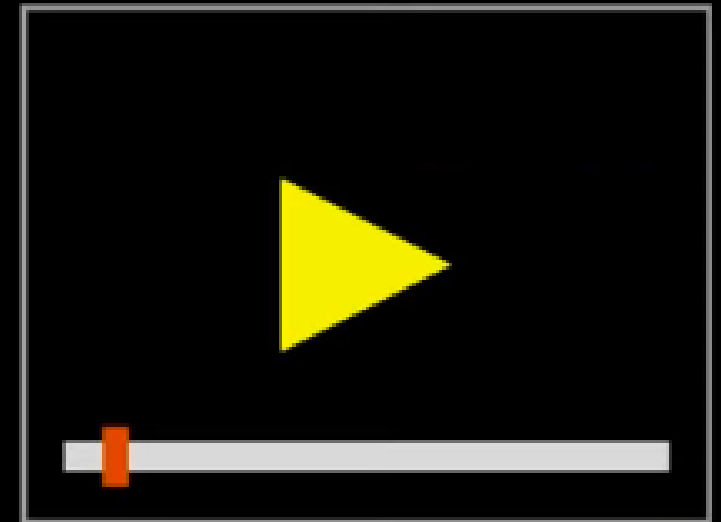


Connection:

Slow

Median

Fast



- Accediendo a esta información:

```
const navigator = window.navigator; // Info ab. user-agent
const connection = (navigator as any). connection

// Set default to 4g
const effectiveType = ref('4g')

updateNetworkInfo() {
    effectiveType.value = connection.effectiveType ?? 'unknown'
}

if (connection) { //the API is not available on every browser
    // Add eventListener for when connection changes
    connection.addEventListener('change', updateNetworkInfo)
    updateNetworkInfo() //Make a call to update the netw. info on the
}
```

- Utilizando la información para mostrar una imagen a completa resolución o una versión blurry de un thumbnail.

```

```

- Tener en cuenta que Safari o FFox aún no lo soportan

Índice

- 1. Network Status Based UI
- ***2. Image Optimization with Nuxt Image***
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- 4. One Line Animations

2. Image Optimization with Nuxt Image

Image Optimization with Nuxt Image

- El performance de nuestra web impacta todo (ux, ratios conversión, usabilidad, etc).
- Lo que más impacta al performance (+ tiempo) es la cantidad de datos que el usuario ha de descargar. (junto con el renderizado, que abordaremos más adelante y añade consumo cpu y ram)
- Hay varias maneras de reducir esta cantidad de datos, una de las más comunes siendo la optimización de imágenes.
- Ejemplo: Twitter tiene acceso a la versión HD de nuestra imagen de perfil, pero en la card del tweet, no tiene sentido. Trae una versión reducida.



Evan You @youyuxi · 26 ene.

Migrated Vue 3's entire test suite from Jest to Vitest in one day, 175 test files / 2647 test cases.

A couple of small issues, but I found workarounds - 99% of test code remained exactly the same, and almost everything in Jest has an equivalent.



17



51



917



99,7 mil



Mostrar más

Condiciones de
Política de con
Información d
© 2023 Twitter

Mensajes

QiHm_normal.jpg

{ } Editor de estilos



Rendimiento



Memoria



Almacenamiento



Accesibilidad

Aplicación



AdBlock

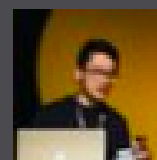


Filtrar elementos



```
<div class="css-1dbjc4n r-1niwhzg r-vvn4in  
r-u6sd8q r-4gszlv r-1p0dtai r...2tsx r-1d2f490  
r-u8s1d r-zchl nj r-ipm5af r-13qz1uu r-1wyyakw"  
style="background-image:  
url("https://pbs.twimg.com/profile_images  
/1206997998900850688  
/cTXTQiHm_normal.jpg");"></div> flex
```

```
 event
```



48 x 48

en línea

elemento :

```
background-image:  
url("https://pbs.twimg.com  
/profile_images/1206997998900850688  
/cTXTQiHm_normal.jpg");  
}
```

```
.r-1wyyakw :: {  
  z-index: -1;  
}
```

```
.r-4gszlv :: {  
  en línea
```

- El componente de Nuxt Image, nos permite interactuar con distintos servicios de compresión de imagen, especificando desde el tag en el template <nuxt-img>, el tamaño y formato que consideremos ideal para la imagen.

```
<template>  
  <nuxt-img src="/image.png" width="48" height="48" format="webp" />  
</template>
```

- La documentación es muy completa y destaca del componente, la cantidad de proveedores de compresión de imagen disponibles

Cloudflare

Cloudimage

Cloudinary

Contentful

Edgio (formerly Layer0)

Fastly

Glide

Gumlet

ImageEngine

ImageKit

Imgix

IPX

Layer0

Netlify

Prismic

Sanity

Storyblok

Strapi

Twicpics

Unsplash

Vercel

- La configuración del proveedor es muy sencilla.

```
image: {  
  provider: 'storyblok',  
  storyblok: {  
    baseUrl: 'https://a-us.storyblok.com'  
  }  
},
```

- Nuxt Image nos expone dos componentes

Nuxt Image	Nativo
<code><nuxt-img /></code>	<code></code>
<code><nuxt-picture /></code>	<code><picture /></code>

- Podemos además, especificar para algunos proveedores distintos tamaños para adoptar carácter responsive mediante "sizes"

```
<template>
  <nuxt-img
    src="/image.png"
    sizes="sm:100px md:300px lg:900px" <!-- Dynamic sizing depending on viewport real state-->
    format="webp"
  />
</template>
```

- Algunas otras acciones que se pueden tomar sobre las imágenes (depende del soporte del proveedor) son:
 - Grayscale
 - Blur
 - Smart cropping
 - Focal points
 - Image Rounding
 - Rotation
 - etc

- Estas opciones se pasan mediante la prop :modifiers

```
<template>
  <nuxt-img
    src="/image.png"
    sizes="sm:100px md:300px lg:900px"
    format="webp"
    :modifiers="{
      filters: {
        blur: 10,
        rotate: 90
      }
    }"
  />
</template>
```

Índice

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- **3. *Rendering performance***
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- 4. One Line Animations

3. Rendering Performance

Rendering Performance

- El performance de renderizado es una métrica muy importante para las aplicaciones web
- Por cada segundo que nuestra app tarda en renderizar, más probable que el usuario salga de la misma
- La aplicación ha de sentirse responsive ante el input de los usuarios
- Vue tiene por defecto un performance bastante bueno
- Aún así, hay maneras en las que podemos mejorar este performance

1. Virtual Lists

2. Eliminación de re-renderizaciones innecesarias

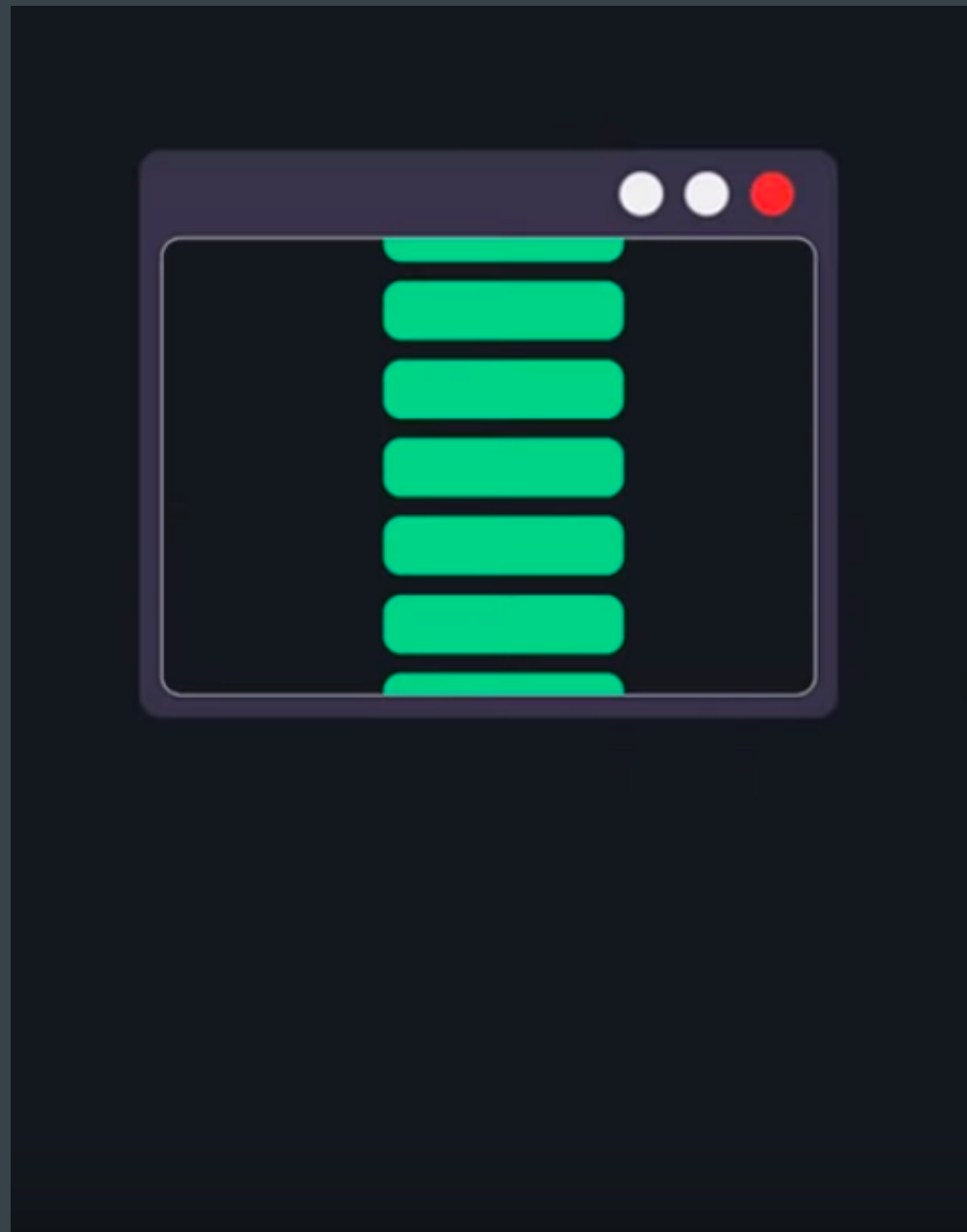
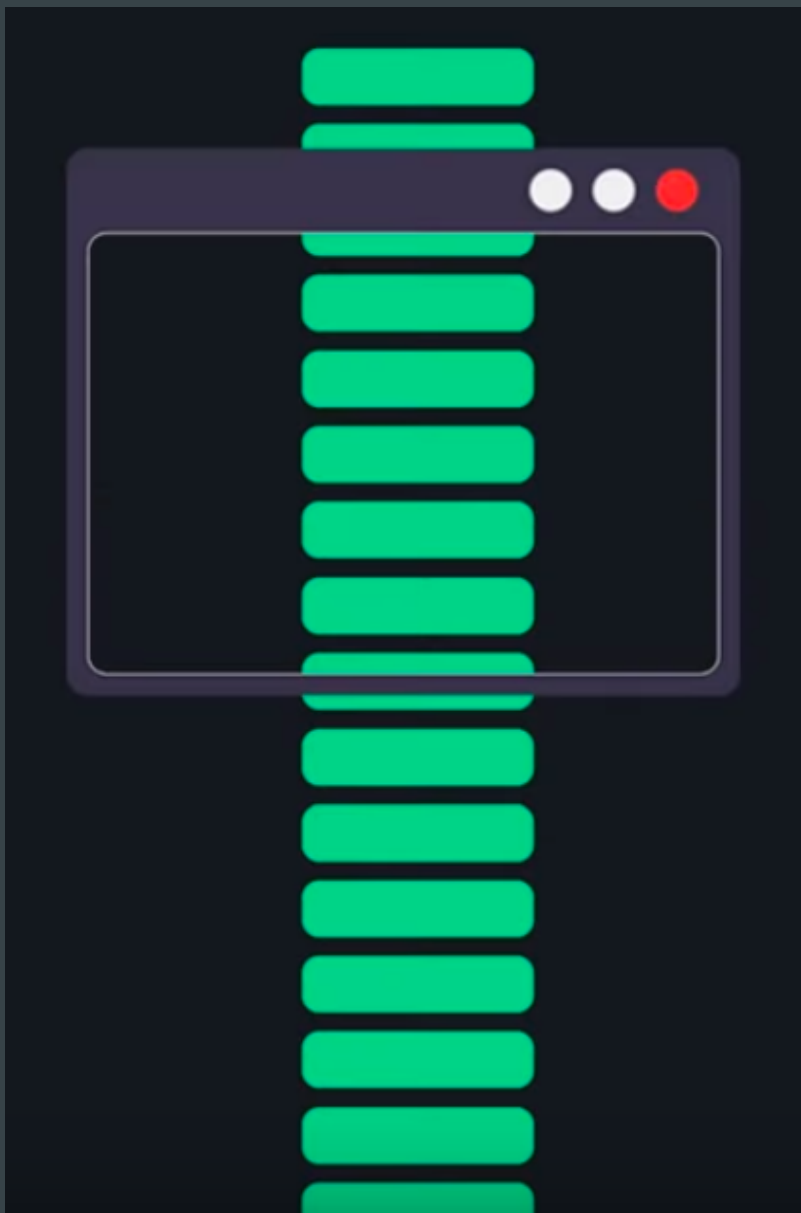
- v-once
- v-memo

Índice

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - ***3.1 Rendering Performance with Virtual Lists***
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- 4. One Line Animations

3.1 - Rendering Performance with Virtual Lists

- Si intentamos recorrer una lista enorme con v-for, tendremos un impacto en el performance de nuestra app. Tendremos tanto una primera carga lenta, cómo un scroll muy laggy.
- La solución a esta situación pasa por el ***virtual scrolling***, en concreto para listas, por el ***list virtualization***
- Renderizaremos divs que estén cerca del viewport actual
- Lo más común es combinar el virtual scrolling con el infinite scrolling (paginación +1 a la llegada al final de la lista actual)
- Visualización scroll v-for vs. virtual scroll



- Hay varias maneras de lograr el efecto deseado, pero en este caso vamos a ver la opción con el composable ***useVirtualList*** de VueUse
- Lo combinaríamos con ***useInfiniteScroll*** para lograr un efecto similar al que hacer twitter.
- El uso del composable ***useVirtualList*** nos ahorra realizar muchos cálculos y tiempo a la hora de pensar cómo hacer para que la zona a renderizar se corresponda correctamente con el scrollbar (vs el total de elementos de nuestra lista)
- Afortunadamente, este tipo de soluciones tan testadas y recomendadas nos permiten obtener ganancias de rendimiento de manera sencilla. Tendremos que instalar [VueUse](#)

```
<script setup>
import { ref } from 'vue'
import { useVirtualList } from '@vueuse/core'

// Data -> [50] of lorem ipsum
// We can use it as a ref or not. It's the same
const data = ref(Array.from(Array(50).keys(), () => 'Lorem Ipsum'))

// Properties returned by useVirtualList composable
// list -> items that should actually be shown
// containerProps -> ref for container element, inline styles, onScroll event
// wrapperProps -> styles for our wrapper (like height + margin)

const {list, containerProps, wrapperProps} = useVirtualList(data, {
  itemHeight: 96 // Used for calculations
  overscan: 5 // Numb of items to be rendered outside the viewport (up n down). Smoother scroll.
})
</script>
```

```
<template>
  <div v-bind="containerProps"> <!-- containerProps ok! -->
    <div v-bind="wrapperProps"> <!-- wrapperProps ok! -->
      <div
        v-for="{index, data} in list"
        :key="index"
        class="item-card-class" <!-- Height adds up to itemHeight! -->
      >
        <h2 class="h2-class">Item #{{index}}</h2>
        <p class="p-class">{{data}}</p>
      </div>
    </div>
  </div>
</template>
```

80px

16px

Item #0

Lorem ipsum

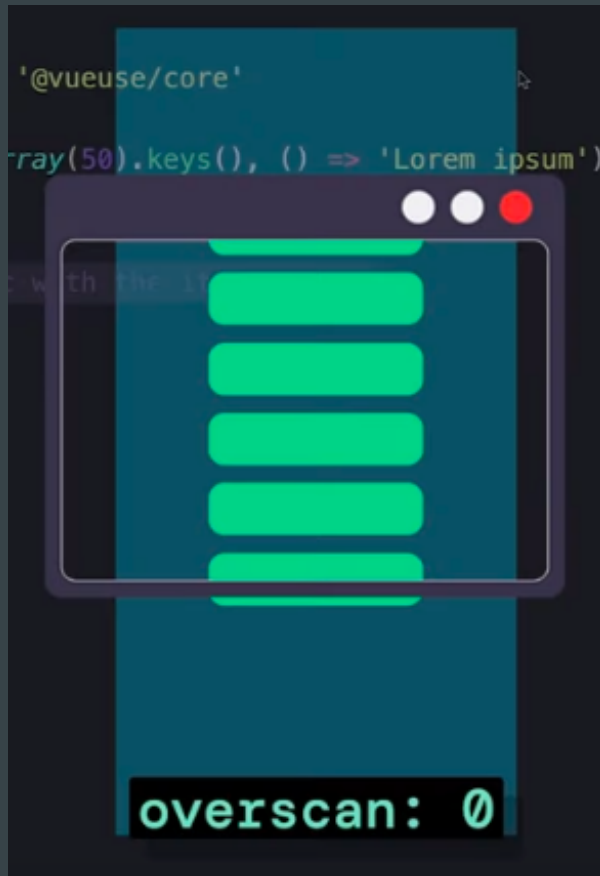
Item #1

Lorem ipsum

Item #2

Lorem ipsum

- Default overscan is 5



Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

```

Elements Console Sources Performance » 10
* <div style="width: 100%; height: 4800px; margin-top: 0px;" class="max-w-sm mx-auto">
  <!--|-->
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <div class="rounded-lg h-[80px] flex flex-col px-4 justify-center bg-neutral-800 mb-4 border-neutral-600"></div> flex
  <!--|--> == $0
</div>
</div>
<!--|-->
</astro-island>
</div>
</div>
<div style="position: static !important;"></div>

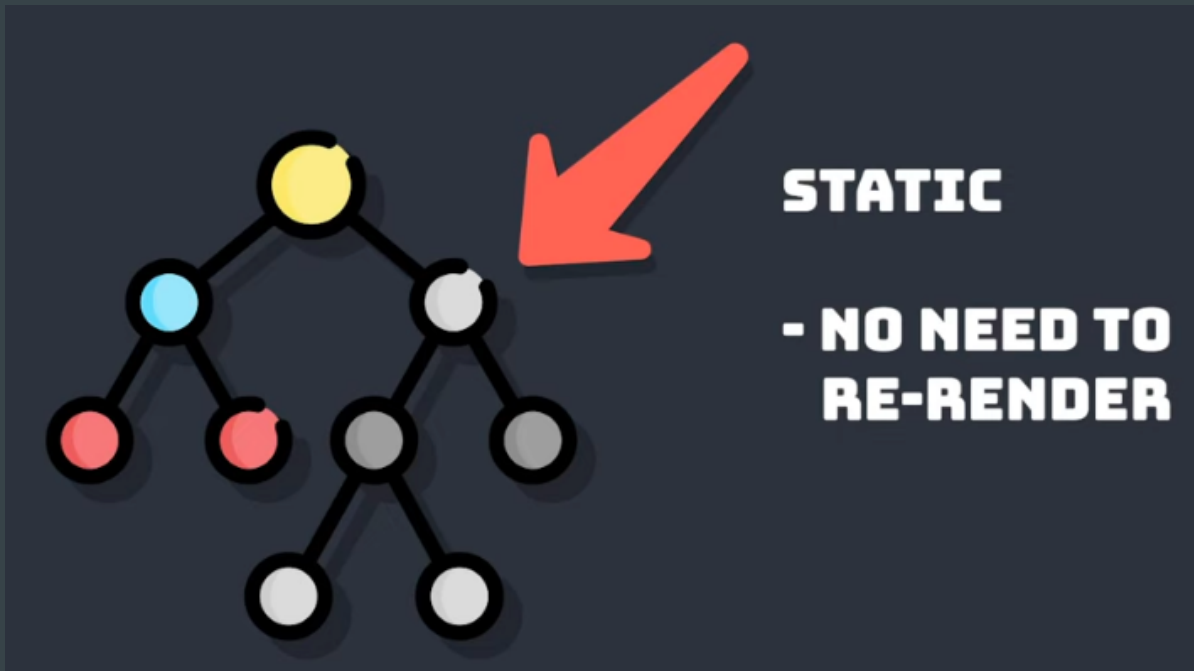
```

Índice

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
 - **3.2.1 - v-once**
 - 3.2.1 - v-memo
- 4. One Line Animations

3.2.1 - Rendering Performance with v-once

- Renderiza el elemento o componente una sola vez. Después de ese render inicial, ese elemento/componente y todos sus hijos, serán tratados como estáticos



- Para lograrlo, simplemente:

```
<!-- v-once in a single html element -->
```

```
<p v-once>{{myStaticMessage}}</p>
```

```
<!-- v-once in an element with children -->
```

```
<div v-once>
```

```
  <p>{{message}}</p>
```

```
</div>
```

```
<!-- v-once in a component -->
```

```
<my-component v-once />
```

```
<!-- v-once in a v-for directive -->
```

```
<ul>
```

```
  <li v-for="item in list" :key="item" v-once>{{item}}</li>
```

```
</ul>
```

- Consideraciones de v-once

- Tratar con cuidado esta directiva. Aunque cambien los datos reactivos, el componente no se re-renderizará
- La combinación de v-once con otras directivas puede tener efectos no-deseados, por la prioridad de este.

```
<!-- v-once with conditional components -->  
<p v-if="show" v-once>{{myStaticMessage}}</p>
```

- En el ejemplo, aunque cambie el show, el componente no reaccionará
- Utilizaremos v-once sólo cuándo sepamos que un componente no se ha de re-renderizar aunque los datos se actualizen.

Índice

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
 - 3.2.1 - v-once
 - **3.2.1 - v-memo**
- 4. One Line Animations

3.2.2 - Rendering Performance with v-memo

- Si queremos limitar cuándo un elemento se ha de re-renderizar, pero sin limitarlo sólo a la primera vez, utilizaremos v-memo
- v-memo memoriza un sub-arbol de hijos de un componente y almacena renderizaciones previas para mejorar el performance
- Lo podemos utilizar en cualquier elemento y acepta un array de dependencias. Limitará la re-renderización a cuándo alguna de esas dependencias cambien. Sin array lo hace "smart reactive"
- Importante: v-memo no funciona dentro de v-for, pero si al mismo nivel (en el mismo elemento que v-for si, pero no dentro)

- El `<p>` se re-renderizará reactivamente ante el cambio de `msg` o de `count`

```
<!-- v-memo and some dependencies -->  
<p v-memo="[msg, count]">{{msg}}, {{count}}</p>
```

- Si utilizamos `v-memo` pero pasándole un array de dependencias vacío, obtenemos el mismo resultado que utilizando `v-once`, ya que no hay nada que haga trigger del re-render. Si no pasamos array, lo hace de manera inteligente

```
<p v-memo="[]">  
  {{ msg }} | {{ count }}  
</p>
```

```
<p v-once>  
  {{ msg }} | {{ count }}  
</p>
```

- La documentación oficial considera que v-memo debería de ser utilizado sólo en micro-optimizaciones en escenarios de performance crítico (no utilizarlo by-default)
- El uso más popular es a la hora de renderizar listas con muchos registros (length > 1000)

Índice

- 1. Network Status Based UI
- 2. Image Optimization with Nuxt Image
- 3. Rendering performance
 - 3.1 Rendering Performance with Virtual Lists
 - 3.2 Rendering Performance with directives (avoid unnecessary re-render)
- **4. *One Line Animations***

4. One-Line Animations Auto Animate

One Line Animations


- Otra mejora que podemos llevar a cabo en nuestra aplicación es la del aspecto estético (con animaciones por ejemplo)
- Para ser óptimos, tanto con las animaciones (performance) cómo con el tiempo que nos lleva implementarlas, nos podemos servir de herramientas.
- Echaremos un ojo a [Auto Animate](#)
- Destaca por ser **zero** config y muy óptimo en cuanto a rendimiento. Además, funciona con cualquier app JavaScript (no está limitado a VueJS)
- Podemos importarla global o localmente. Expondrá directiva.

Global



```
import { createApp } from 'vue'
import { autoAnimatePlugin } from '@formkit/auto-animate/vue'
import App from './App.vue'

const app = createApp(App)
app.use(autoAnimatePlugin)
app.mount( '#app' )
```



Local



```
<script setup>
import { vAutoAnimate } from '@formkit/auto-animate'

</script>

<template>

</template>
```

- Esencialmente, `autoAnimate` es una función que acepta un elemento, para luego animar ese elemento padre y sus hijos inmediatos cuándo:
 - Elemento hijo se añade al DOM
 - Elemento hijo se elimina del DOM
 - Elemento hijo se mueve en el DOM



The diagram illustrates the `autoAnimate()` function and its target. On the left, the function `autoAnimate()` is shown in a dark box. A white curved arrow points from the opening parenthesis of the function to a `` element on the right. The `` element is followed by three `` elements, all stacked vertically in a dark box.

```
autoAnimate( )
```

```
<ul>  
<li/>  
<li/>  
<li/>
```


- Ejemplo en que contamos con un div "header", el cual al ser clickado nos "abre" / "muestra" un <p>. La diferencia entre nos "muestra" y nos "abre", nos la aporta auto animate, utilizando un par de líneas.

```
<script setup>
import { vAutoAnimate } from '@formkit/auto-animate' <!--import-->
import { ref } from 'vue'

const isOpen = ref(false)
</script>

<template>
  <div v-auto-animate class="p8 rounded-lg bg-neutral-900"> <!--apply-->
    <h2
      @click="isOpen = !isOpen" class="text-xl font-bold">
      Card Header
      <p v-if="isOpen" class="py-8">Card content</p>
    </h2>
  </div>
</template>
```

- Nota: Además de poder utilizar como directiva, contamos con un **composable**

```
<script setup>
import { vAutoAnimate } from '@formkit/auto-animate'
import { ref } from 'vue'
const isOpen = ref(false)
const [wrapperEl] = useAutoAnimate()                                //Composable
</script>

<template>
  <div ref="wrapperEl" class="p8 rounded-lg bg-neutral-900">
    <h2
      @click="isOpen = !isOpen" class="text-xl font-bold">
      Card Header
      <p v-if="isOpen" class="py-8">Card content</p>
    </h2>
  </div>
</template>
```

- Aunque simplemente funciona out-of-the-box, también podremos especificar opciones o incluso utilizar plugins.

```
<template>
  <div v-auto-animate="{duration: 1000, easing:'linear'}">
  </div>
</template>
```

- Para más información, consultar la [documentación](#)

Conclusión

Conclusión

- Vue nos da una base muy sólida para desarrollar proyectos bastante optimizados en cuanto a performance, a la vez que nos aporta la libertad necesaria para hacer lo que queramos
- Está muy bien contar con opciones de mejoras de rendimiento tanto nativas de JS, nativas de vue o de terceros.
- Es importante priorizar el performance de nuestra webapp para aportar una experiencia de usuario responsive y agradable.
- Estas son sólo algunos ejemplos, hay todo un mundo dedicado a este tipo de mejoras. Destaca el canal de youtube: [LearnVue](#)

Fin.