Se puede usar computed e importarlo para guardar valores ahí que se modifiquen se define como const valorComp = computed(()=>{ return valor1\*valor1})

Properties:

v-bind es igual a poner solo los dos puntos



A black background with orange and white text

Description automatically generated

O mejor así para evitar problemas, si es que esto es requerido.

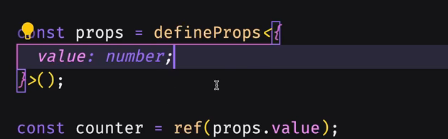
A screen shot of a computer

Description automatically generated

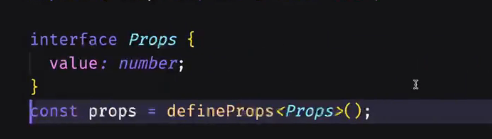
Para llamarlo con props.value



Para hacerlo con ts va a hacer obligatorio el valor :



Ooo con la interfaz



Para hacerlo opcional, lo que si se se hace opcional habría que manjearlo a los valores por si son undefined.   


defineProps() : Para recibir props del padre al hijo, no hace falta importar

defineEmits() : Para crear eventos que el hijo le emitirá al padre.

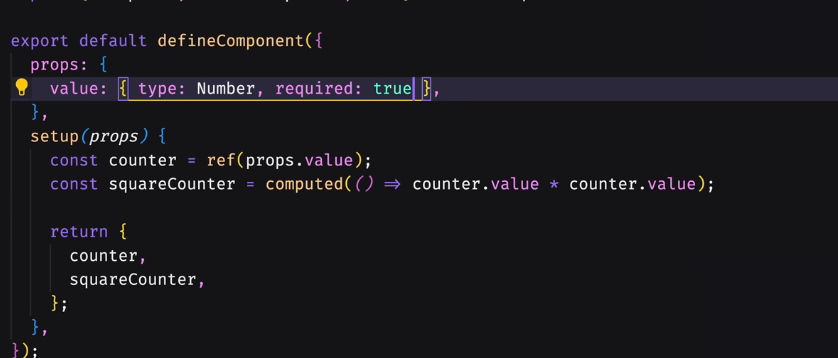
Las props en el html se pone sin props.text solo text

**Componente tradicional:**

En vez de usar script setup, usamos el setup abajo con defineComponent



Para mandar props notar que se envia al setup:



Separar archivos:

Llama directamente así al file de src

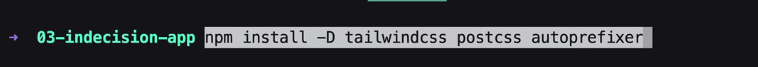


Y el ts es un ts común

A screen shot of a computer program

Description automatically generated

Tailwindcss es como una especie de Bootstrap

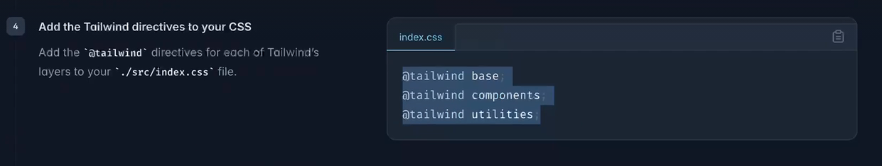




Bueno acá están los pasos

<https://tailwindcss.com/>

En este paso crear el styles.css e importarlo en el main.ts o js, eso va a hacer que se aplique globalmente



**Desplegar app:**

Npm run build

Creo 3 archivos



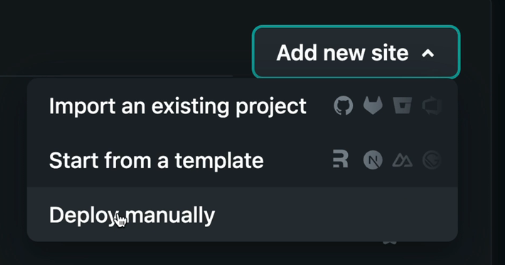
Y creo la carpeta de dist, esa no hay que subirla a github



Se puede subir con un tier gratis a netlify , al menos para lo que es front.

<https://www.netlify.com/>

deploy manually



Arrastramos dist ahí

Esto va a tomar el código y lo deploya

A screenshot of a computer

Description automatically generated

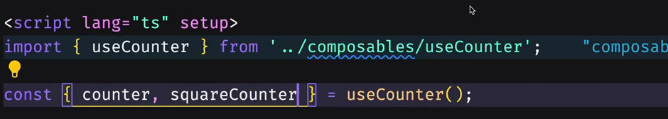
**Composable functions:**

La idea de los composables es reutilizar código.

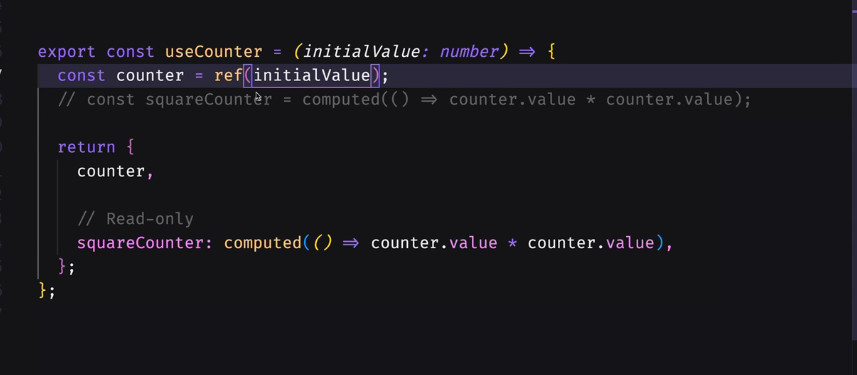
Es una función que se podría usar en varios lados



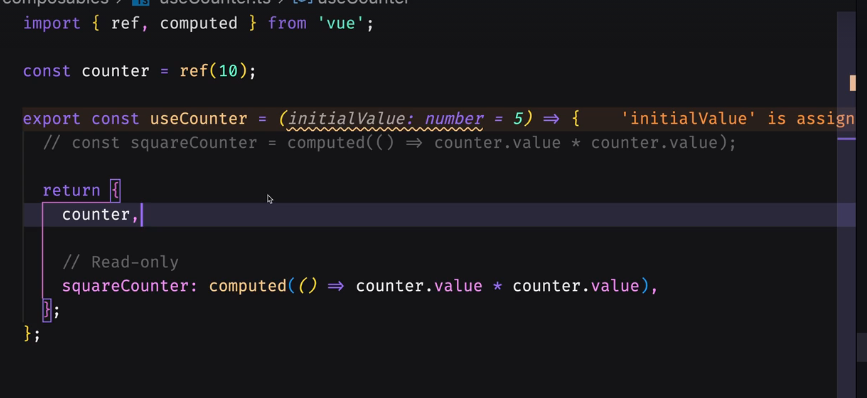
Y para usarlo sería algo así:



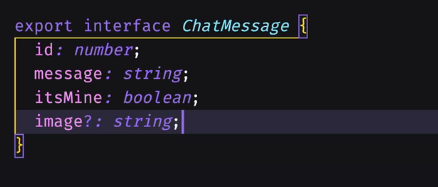
Pasar parámetros en el composable



Si definimos el counter por fuera del useCounter ese valor se va a compartir para todos los lugares donde se use el composable



También crea archivos de interface aparte de ts

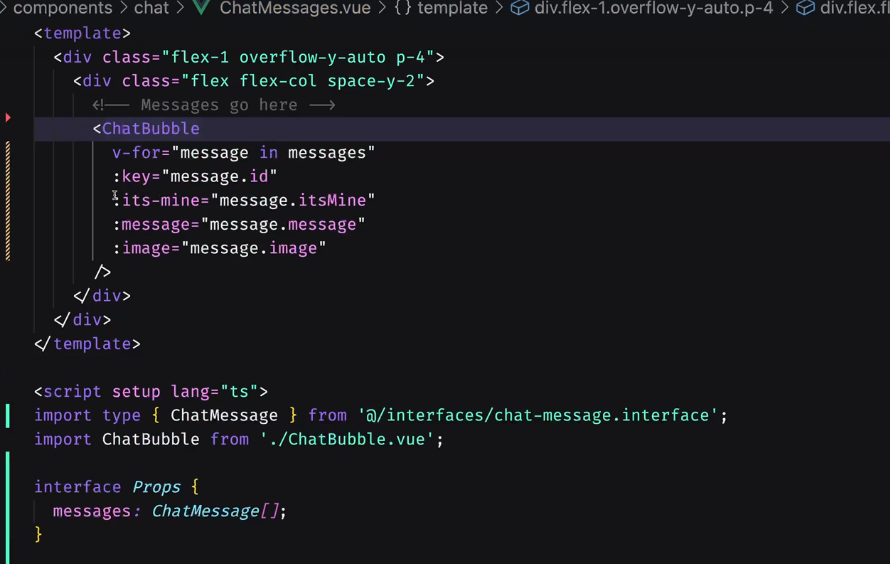


Después lo importa

A screen shot of a computer

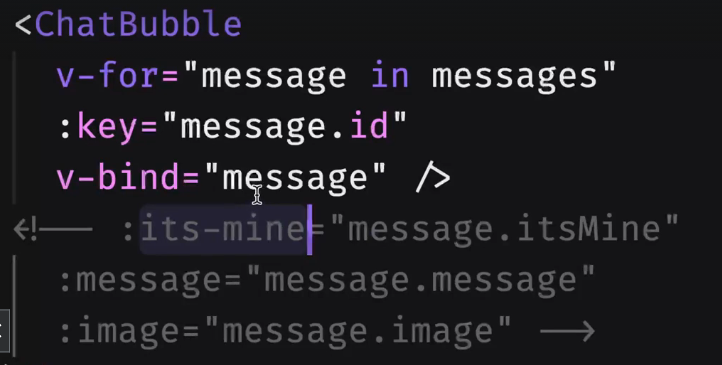
Description automatically generated

En ese ChatBubble estoy recibiendo esos datos de message del padre, se puede hacer de otra forma

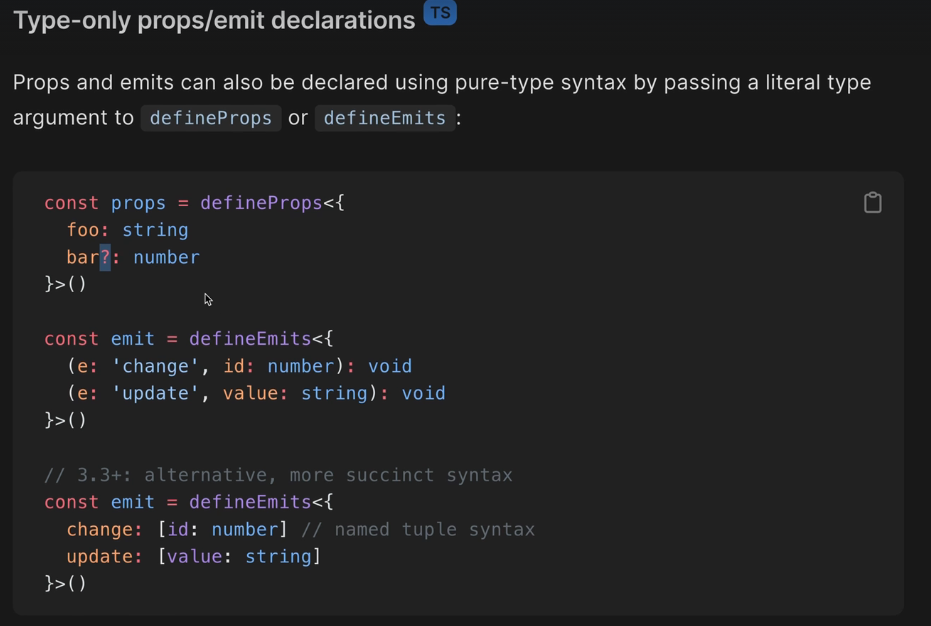


Se puede usar el v-bind,

Que terminaría haciendo lo mismo, va a pasar las props



defineEmits:



No hace falta importar defineEmits



En el padre se hace:

A purple and yellow text

Description automatically generated

Sepodría enviar también únicamente la función



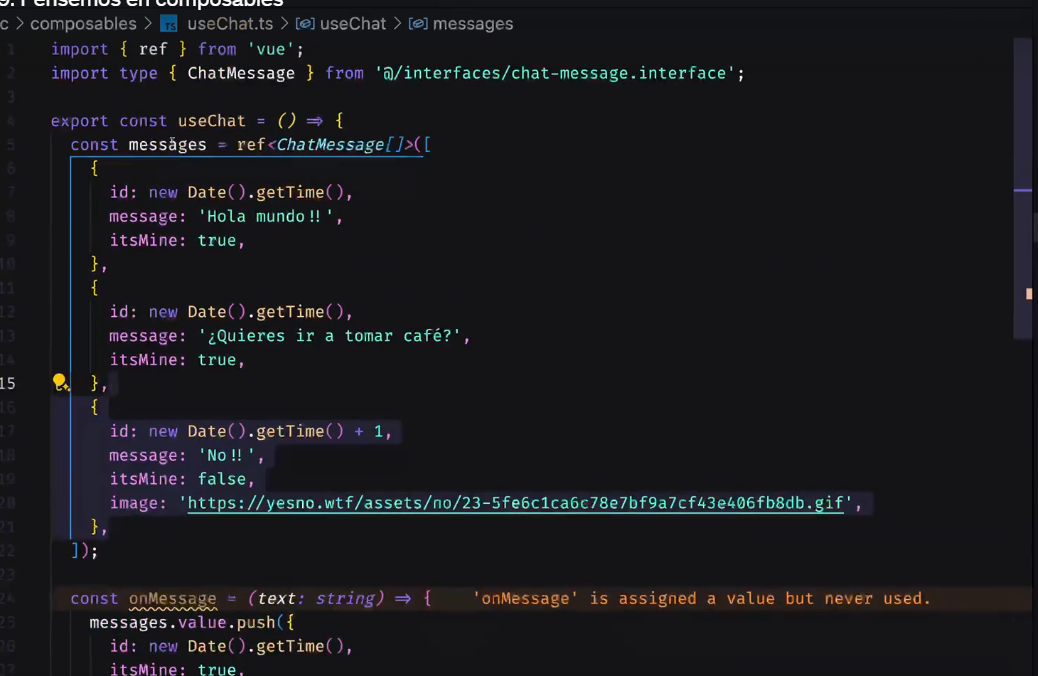
}A computer screen with text and symbols

Description automatically generated

Pensar en composables:

Crea uno nuevo:

Agarró las funciones que tenía en los componentes y lo puso en el composable:

  
  
Después retorna lo que haga falta

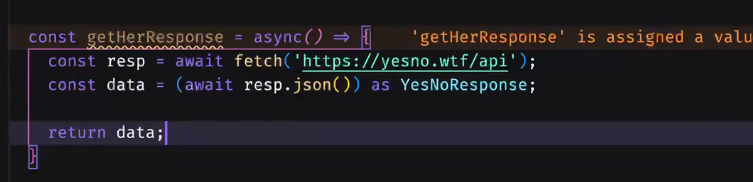
A computer screen shot of a program code

Description automatically generated

Donde necesitemos usar ese composable lo importamos y lo único que hacemos es desestructurar lo que nos haga falta:



Llamada a API, YesNoResponse es la interfaz que define el tipo de respuesta:



Promesa que demora en responder

A screen shot of a computer

Description automatically generated

Refs referencias con typescript:

Hablando de refs cuando pones en un elemento html ref=”chatRef”



Las refs tardan en poder usarse hasta que se monta todo el DOM,

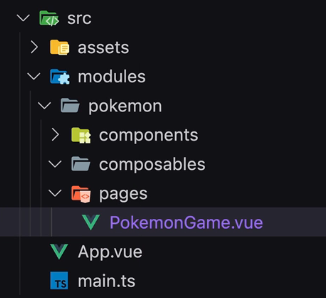
Watch:

Acá cada vez que cambie messages se va a ejecutar la función flecha



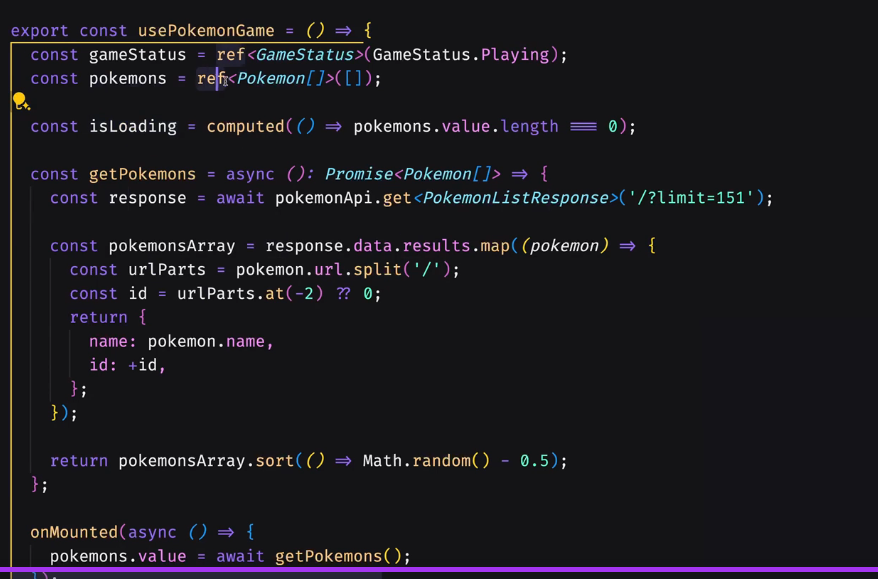
Estructura de archivos:



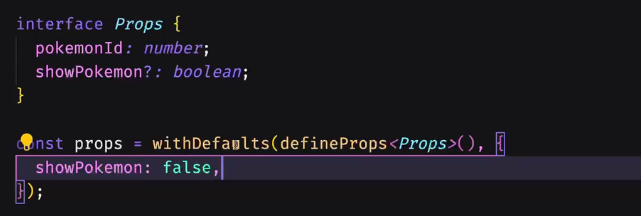


Loading:

Esta bueno como lo hiz, directamente creo una prop computada que esta verificando por el largo de un



Usar valores por defecto en props:



Con un dos puntos en el destructuring podes poner un alias:



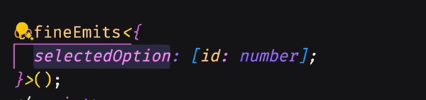
En este caso pokemonOptions le deja el alias de options

Las props desde el html se pueden llamar sin necesidad de hacer props. Creo, peeero acá las definió a las props pero no las guardó en ningún lado

A screen shot of a computer program

Description automatically generated

DefineEmits con typescript

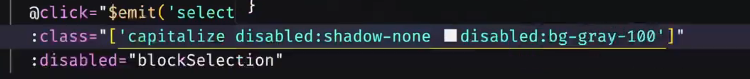




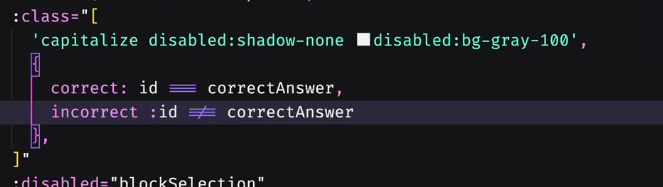
Solo con eso ya debería funcionar



Si tenemos una clase así va a tomar los estilos igual, el disabled:algo va a aplicar únicamente si la prop disabled es true



Entonces en este caso tenemos otras dos clases, una que es incorrect y otra que es correct, de esta forma las podemos mostrar dinámicamente



Alias para paths:

Si tenes algo así y se repite mucho laruta:   
  
En vite.config está la definición del @

A screen shot of a computer

Description automatically generated

Que apunta en este caso al src acá podemos agregar

A computer screen shot of a program

Description automatically generated

A black background with text on it

Description automatically generated

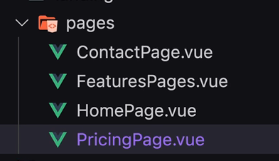
Si estamos con typescript hay que ir al tsconfig.app.ts y agregar esto:

A black background with blue text

Description automatically generated

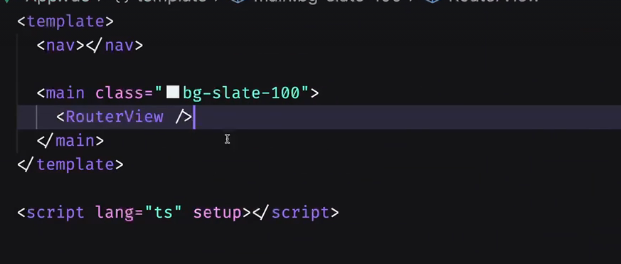
Router:

Empieza creando las pages:

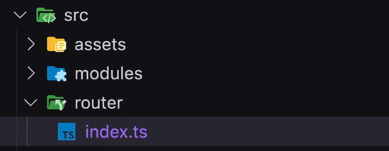


El RouterView viene del paquete que se instala de vue-router y también router-link

Entonces lo común es hacer en el app.vue:



Hay que crear el archivo del router



Ahí adentro:

History permite saber de donde venía el usuario ósea desde que url

A computer screen shot of a program code

Description automatically generated

Importamos y usamos el router en el main

