EXERCISES QUE TRABAJAREMOS EN LA CUE:

- EXERCISE 1: WATCH PROPERTY.
- EXERCISE 2: WATCH (NEWVALUE, OLDVALUE).
- EXERCISE 3: SLOTS.
- EXERCISE 4: SLOT CON NOMBRE.

EXERCISE 1: WATCH PROPERTY

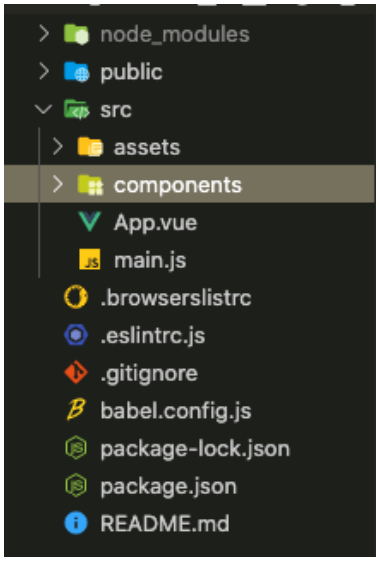
Comenzaremos creando un proyecto con vue/cli. Para ello, ingresamos al terminal, buscamos un directorio donde deseemos trabajar, en este caso hemos elegido escritorio (Desktop), y escribimos: “vue create nombre_proyecto”. Se le puso cue7, pero puede llamarse como quieras.

Desvincular componente HelloWorld.vue

Al crear un proyecto con vue/cli, por defecto éste nos agrega un componente de ejemplo, llamado: HelloWorld.vue, que se encuentra dentro de la carpeta “components”.

1. Eliminar HelloWorld.vue, desde la carpeta components.
2. Borrar referencias en archivo App.vue, del componente recién eliminado.

Ahora, la estructura de nuestro proyecto debería quedar como la siguiente imagen:



El archivo App.vue, es donde debemos borrar la referencia al componente HelloWorld.vue, quedando de esta forma:

```
<template>
  <div id="app">

  </div>
</template>

<script>

export default {
  name: 'App',
  components: {

  }
}
</script>

<style>
</style>
```

Creando Componente:

Vamos a crear un componente en dentro de la carpeta components, al cual le llamaremos: MyComponent.vue.

```
<template>

</template>

<script>
export default {
  name:"MyComponent",
  data:function(){
    return{

    }
  },
  watch:{

  }
}
</script>
<style>

</style>
```

Para comenzar, agregaremos el dato **title** al objeto data, para luego observar sus cambios con un watcher.

```
data:function(){
    return{
      title:"Estudiando Watch properties",
    }
  },
```

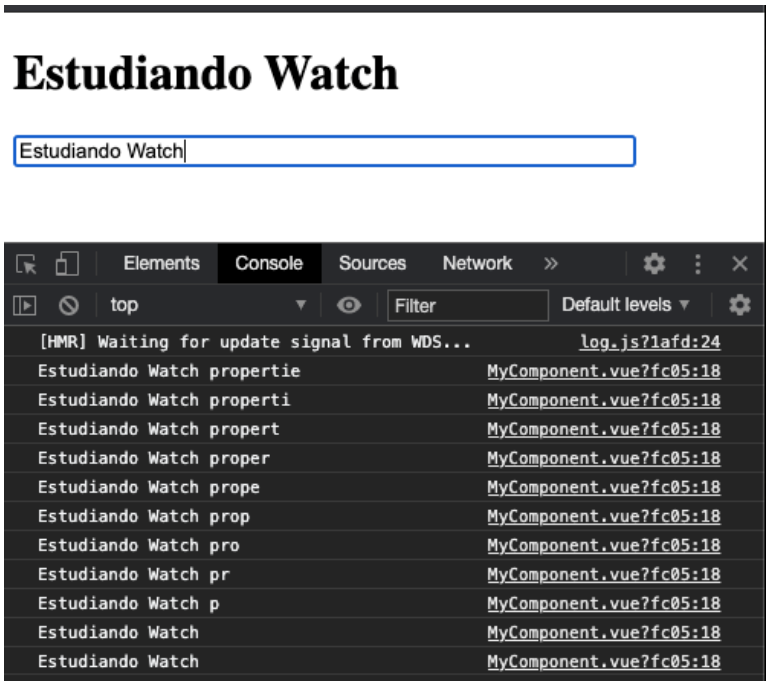
Ahora, procederemos a crear el watch. Debemos entender que el watcher debe llamarse igual que la propiedad observada, en este caso: title.

```
watch:{
    title(value){
      console.log(value)
    },
  }
```

Una vez teniendo el watch, procederemos a crear un pequeño template, para poder enlazar el dato title a un input, y así poder ver los cambios que se generan.

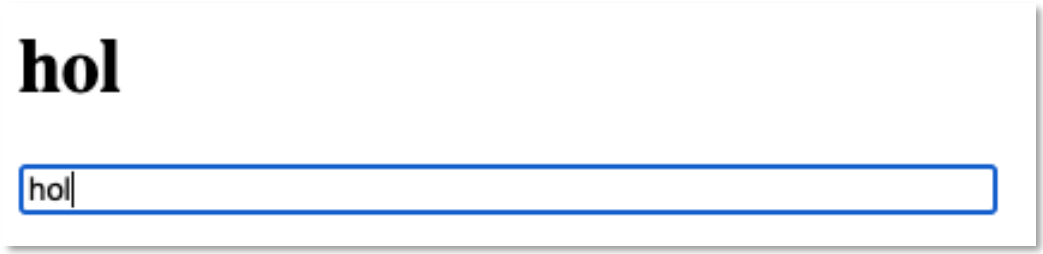
```
<template>
  <div>
    <h1>{{title}}</h1>
    <input type="text" v-model="title">
  </div>
</template>
```

Si borramos algún dato, o agregamos un caracter en el input, el watcher title se ejecutará.



Si lo notamos, la propiedad watcher nos puede servir para gatillar métodos, cuando pase algún cambio en el dato. Por ejemplo: vamos a crear un condicional, que al detectar la palabra “hola”, escriba automáticamente otro mensaje.

```
watch:{
    title(value){
      if(value == "hola"){
        this.title = "este es un nuevo mensaje"
      }
      console.log(value)
    },
  }
```



Ahora, en el script agregaremos otro dato llamado counter, y lo inicializaremos en 0.

```
data:function(){
    return{
      title:"Estudiando Watch properties",
      counter:0,
    }
  },
```

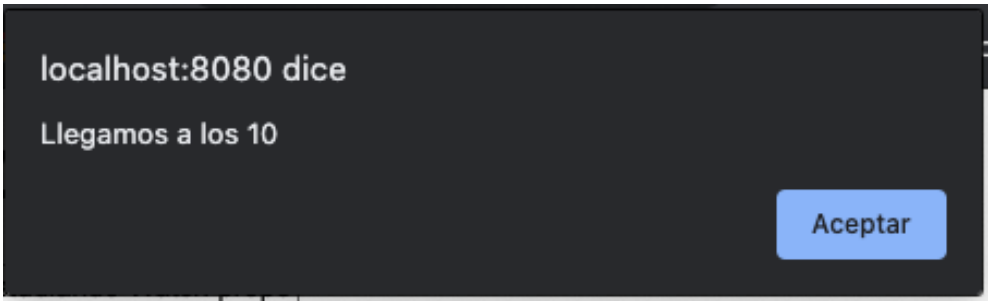
Creamos ahora un watcher, que gatillará una alerta cuando éste llegue a 10.

```
watch:{
    title(value){
      if(value=="hola"){
        this.title="Este es un nuevo mensaje"
      }
      console.log(value)
    },
    counter(value){
      if(value == 10){
        alert("Llegamos a los 10")
      }
    }
  }
```

En la sección de template, agregaremos un H1, el cual mostrará el valor actual de counter, y un botón que nos ayudará a sumar 1 por cada clic.

```
<template>
  <div>
    <h1>{{title}}</h1>
    <input type="text" v-model="title">
    <h1>Contador: {{counter}}</h1>
    <button @click="counter++">Sumar</button>
  </div>
</template>
```

Si hacemos clic en el botón sumar, al llegar a 10, recibiremos una alerta.



Algo importante, no olvidar referenciar el componente creado a App.vue.

```
<template>
  <div id="app">
    <MyComponent></MyComponent>
  </div>
</template>

<script>
import MyComponent from '@/components/MyComponent.vue'
export default {
  name: 'App',
  components: {
    MyComponent
  }
}
</script>

<style>

</style>
```

EXERCISE 2: WATCH (NEWVALUE, OLDVALUE)

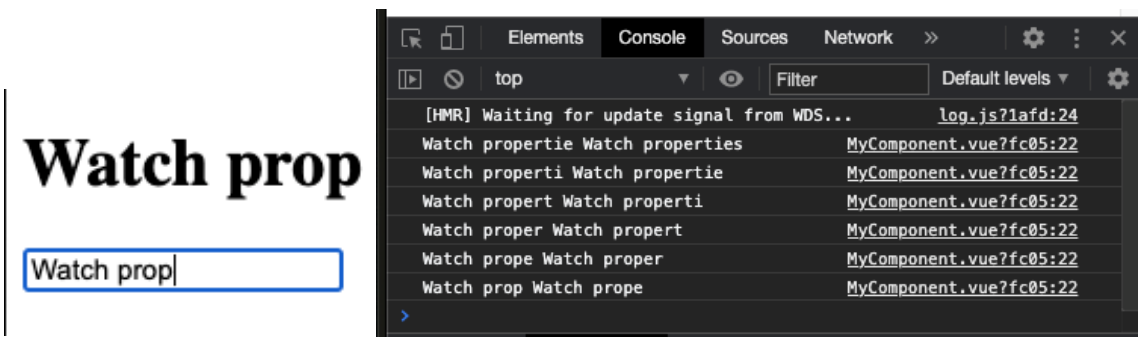
Para continuar, vamos a revisar una propiedad de los watchers, que nos permite obtener el valor actual, y el valor anterior de la propiedad observada. Utilizaremos el ejemplo del ejercicio 1, y solo agregaremos un argumento más a nuestro watcher title.

Watcher con 2 argumentos:

```
watch:{
    title(newValue,oldValue){

      console.log(newValue,oldValue)
    },
```

Si observamos la consola; por ejemplo, cuando borramos elementos del input, veremos que obtendremos dos valores, estos son los que imprime **console.log**.

Si comenzamos a borrar caracteres, veremos que obtenemos el valor nuevo y el anterior, de nuestra propiedad observada.



El uso que le daremos dependerá de nuestros requerimientos. Lo importante es saber que la podemos utilizar cuando necesitemos.

MyComponent.vue debe ser importado a App.vue.

```
<template>
  <div>
    <h1>{{title}}</h1>
    <input type="text" v-model="title">
  </div>
</template>
<script>
export default {
  name:"MyComponent",
  data:function(){
    return{
      title:"Watch properties",
    }
  },
  watch:{
    title(newValue,oldValue){
      console.log(newValue,oldValue)
    },
  }
}
</script>
<style></style>
```

App.vue.

```
<template>
  <div id="app">
    <MyComponent></MyComponent>
  </div>
</template>
<script>
import MyComponent from '@/components/MyComponent.vue'
export default {
  name: 'App',
  components: {
    MyComponent,
  }
}
</script>
<style></style>
```

EXERCISE 3: SLOTS

En algunas ocasiones, es necesario dejar un espacio (slot) dentro de nuestro componente, para que, desde afuera de él, se pueda inyectar algún contenido; ya sean elementos HTML, o algún otro componente.

Para ello, debemos utilizar la etiqueta `<slot></slot>` dentro de nuestro componente. Éste se llamará: MyFomr.vue, y estará en la capeta components.

Creando Componente:

En primer lugar, crearemos un elemento raíz, que en este caso será un `<div>`, luego asignaremos un título `<h1>` y párrafo `<p>`, que estarán dentro de nuestro componente. Finalmente, haremos un div, y dentro de él un slot. De esta forma, podremos agregar información de manera externa al componente, sin tener que usar props

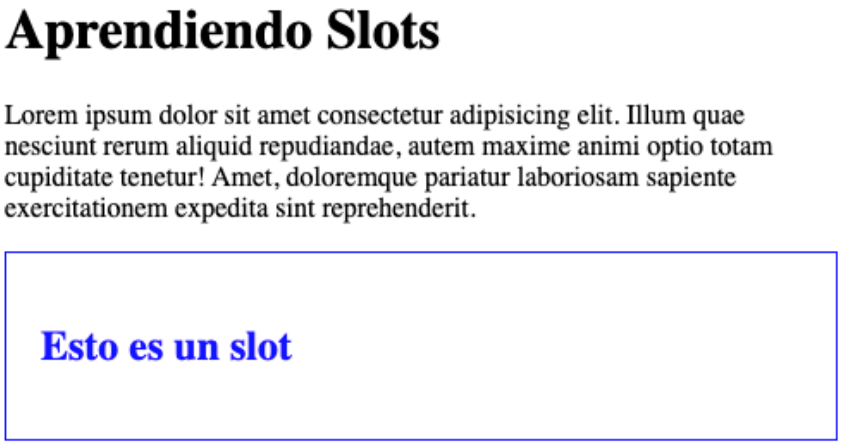
```
<template>
  <div>
    <h1>Aprendiendo Slots</h1>
    <p>
 Lorem ipsum dolor sit amet consectetur adipisicing elit. Illum quae nesciunt rerum aliquid repudiandae, autem maxime animi optio totam cupiditate tenetur! Amet, doloremque pariatur laboriosam sapiente exercitationem expedita sint reprehenderit.
    </p>
    <div id="subtitle">
      <slot></slot>
    </div>
  </div>
</template>

<script>
export default {
  name: "MyFormComponent",
};
</script>
<style scoped>
#subtitle {
  color: blue;
  padding: 20px;
  border: 1px solid blue;
}
</style>
```

Incluyendo MyForm.vue en App.vue

```
<template>
  <div id="app">
    <MyForm>
      <h2>Esto es un slot</h2>
    </MyForm>
  </div>
</template>

<script>
import MyForm from '@/components/MyForm.vue'
export default {
  name: 'App',
  components: {
    MyForm
  }
}
</script>

<style>

</style>
```

Como se puede observar, la forma de agregar nuevo contenido será dentro de la etiqueta del componente al que le agregamos el slot.

```
<MyForm>
   <h2>Esto es un slot</h2>
</MyForm>
```



El slot anterior tiene color azul, ya que dentro del componente MyForm.vue, le agregamos un estilo al div que contiene al slot.

EXERCISE 4: SLOT CON NOMBRE

Cuando necesitamos posicionar más de un slot dentro de nuestro componente, obligatoriamente debemos darle un nombre, para poder definir a que slot inyectar la información.

Para ello, solo debemos agregar el atributo name dentro de la etiqueta slot, por ejemplo:

```
<slot name="header"></slot>
```

```
<slot name="footer"></slot>
```

De esta forma, al agregar información externa, podremos definir a cuál de los 2 slots queremos añadir el nuevo contenido.

Para el ejemplo, utilizaremos el component MyForm.vue, y agregaremos algunos slots con nombres.

```
<template>
  <div>
    <h1>Aprendiendo Slots</h1>
    <slot name="header"></slot>
    <p>
      Lorem ipsum dolor sit amet consectetur
adipisicing elit.Illum quaenesciunt rerum aliquid repudiandae,
autem maxime animi optio totamcupiditate tenetur! Amet,
doloremque pariatur laboriosam sapiente
      exercitationem expedita sint reprehenderit.
    </p>
    <div id="subtitle">
      <slot name="subtitle"></slot>
    </div>
    <slot name="footer"></slot>
  </div>
</template>
<script>
export default {
```

Desde Componente Externo:

Desde donde invocamos a nuestro componente, podemos asignar el slot usando la etiqueta template, y dentro de ésta emplearemos, por ejemplo, la directiva: v-slort:nombre_slot.

```
<template v-slot:header>
   <h1>header slot</h1>
</template>
```

De esta forma podemos posicionar el elemento en el slot indicado, no importando el orden en que éste es declarado, pues se está apuntando por el nombre.

```
<template>
  <div id="app">
    <MyForm>
      <template v-slot:header>
        <h1>header slot</h1>
      </template>
      <template v-slot:subtitle>
        <h2>Subtitle slot</h2>
      </template>
      <template v-slot:footer><h3>Footer</h3></template>
    </MyForm>
  </div>
</template>

<script>
import MyForm from '@/components/MyForm.vue'
export default {
  name: 'App',
  components: {
    MyForm
  }
}
</script>

<style>

</style>
```

De esta forma, podemos asignar un contenido distinto a cada slot.