**Analizando Código Vue 3 – Viniendo de ReactJs**

**Composition - API**

**Estudiante**

**David Medina**

**Unidad De Formación**

**Proyecto de Software 3**

**Docente**

**Adrián Izquierdo**

**Instituto Universitario del putumayo**

**Utp – Mocoa**

**04 / 09 / 2025**

Teniendo en cuenta que vengo trabajando con React desde hace mucho tiempo decidí hacer una pequeña comparación entre algunos elementos de ReactJs y Vue 3 que tienen en común, elementos que son la base fundamental de cada uno de ellos.

Iniciando por:

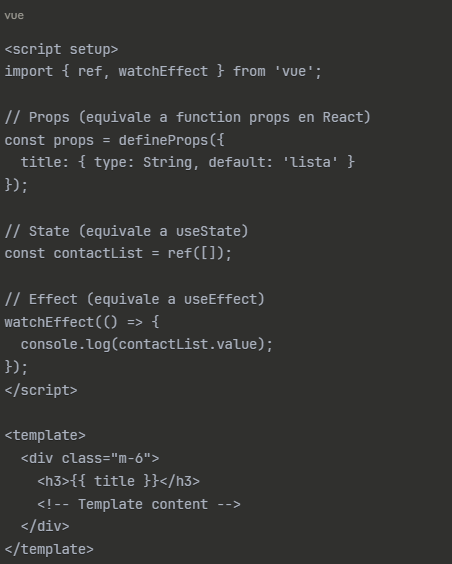
**Estructura de Componentes**

* **Componente Básico**

**React:**

****

**Vue 3:**

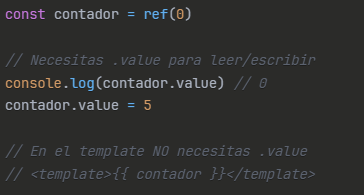


Como podemos observar la forma en que usamos props en nuestros componentes es totalmente diferente mientras que en React las obtenemos directamente en los paréntesis de la función, en Vue 3 usamos una palabra reservada para dar a definir los props de nuestro componente. Es decir, lo que recibimos.

La forma en que definimos estados de variables es distinta. Mientras que en React usamos el useState para definir cualquier tipo de variable en Vue es distinto, ya que aca para variables primitivas es decir por lo general variables que contienen un solo dato, cosa que si se reasigna se reasigna entera. A eso se le llaman variables primitivas y se definen de la siguiente manera:

* **Variables Primitivas**

Como podemos observar solo contiene un solo dato por lo que si reasignamos en cualquier lugar del código se reasigna entera la variable de la siguiente manera:



Como vemos se reasigna con un .value una sintaxis bastante parecida a cuando manipulamos el DOM de HTML desde JS.

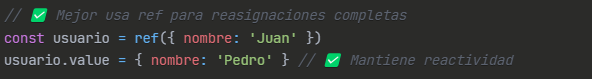
Cabe destacar que para poder leer o Reescribir esa variable se tiene que si o si poner .Value dentro de lo que cabe de la lógica del archivo de Vue.

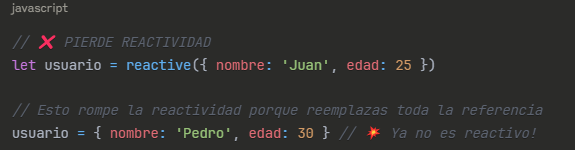
Ya en el Témplate para poder usarlo lo podemos usar sin él. Value

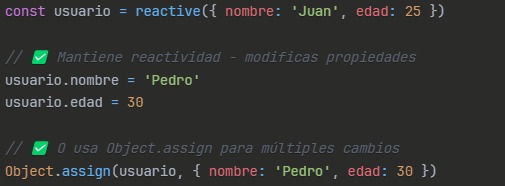
* **Para Objetos o Arrays**

De igual manera para estados de objetos o arrays se suele usar Reactive y no REF

Aunque depende de como se vaya a reasignar la variable mas adelante. Si se reasigna completa lo mejor es usar REF ya que mantendríamos la reactividad que es que REF se usa cuando se reasigna toda la variable por completo.

Si se piensa reasignar la variable, pero no entera, lo adecuado es usar Reactive para mantener la reactividad.





También podemos observar que el cuerpo del componente de React a diferencia del de Vue esta dentro de una función, cosa que en Vue no pasa así, aca usamos etiqueta <script> para poder definir el cuerpo de la lógica del componente y ya luego si Témplate para hacer la función de JSX de React.JS



**Custom Hooks vs Composables**

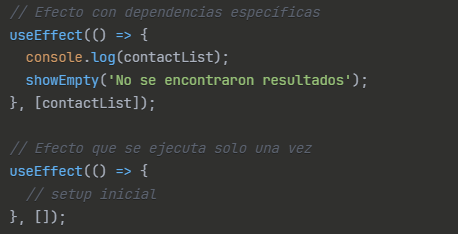
En React usamos Custom Hooks de la siguiente manera para aplicar la lógica del negocio.

En Vue tenemos Composables que son básicamente componentes apartes que almacenan cierta lógica del negocio que se podrá reutilizar en otros componentes, tal como hacemos en React con los Custom Hooks.

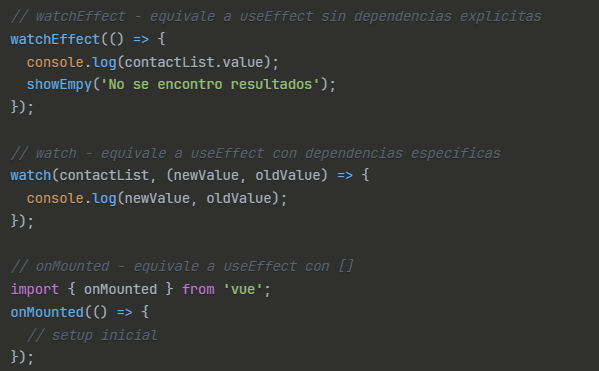
Como podemos observar así lucen los Composables de Vue son similares a los Custom Hooks de React, sin embargo, hay una gran diferencia. Y es que el estado de la lista se encuentra global en el caso de Vue, ya que se encuentra afuera de la función como tal.

**Efectos – UseEffect vs WatchEffect**

Mientras que en ReactJs usamos UseEffect para actualizar dependencias de manera dinámica en vue3 usamos WatchEffect que este hace lo mismo, pero detecta las dependencias de manera automática, siendo mucho más eficiente.

**React**

**Vue**

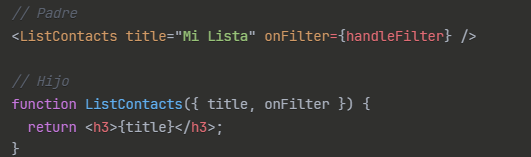


Como podemos observar WatchEffect detecta las dependencias automáticamente, sin embargo, también existen otras palabras reservadas como Watch que equivale a un UseEffect con dependencias especificas de React. O OnMounted que básicamente es como hacer un Setup inicial con React.

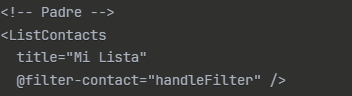
**Comunicación de Componentes**

La comunicación entre componentes también es algo que cambia en cierto modo puesto que en React para comunicarnos de **Padre – Hijo** usábamos directamente Props en la llamada del componente.

De la siguiente manera:

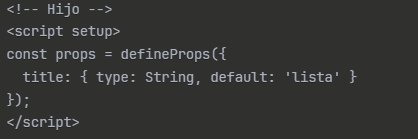


Como podemos ver el Hijo recibe esos datos por medio de las props que contiene la función propia del componente.

En Vue 3 funciona distinto pues mientras que el padre envía la información al hijo de la misma forma por medio de props en el llamado a el componente hijo.

El hijo no recibe esa información de la misma forma que en React, ya que como no maneja como tal una función.

Estos datos se reciben bajo la definición de esos props en el componente hijo, con la palabra reservada asignada para definir props. De la siguiente manera:



**Comunicación Hijo – Padre**

Mientas que React es mas directo a la hora de la comunicación de Hijo – Padre Vue es más basado en eventos.

Es decir, en React tenemos la siguiente forma de comunicarnos de hijo a padre:

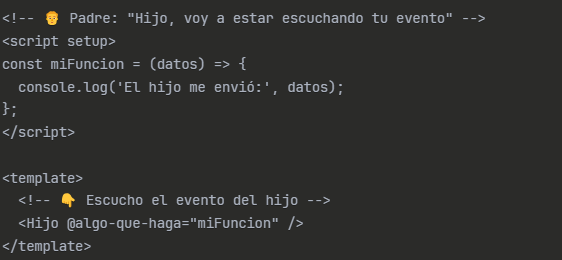
Como podemos observar el padre le está dando a el hijo una función propia de él, por medio de props. Ahora el hijo recibe esta función por props y el decide cuando la usa.

Siguiendo esta analogía en ese caso de comunicación:



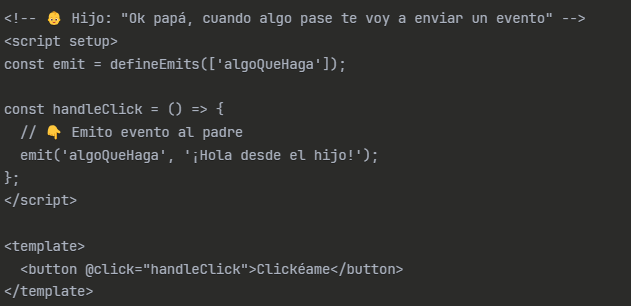
En el caso de Vue 3 como bien dije se enfoca mas a eventos por lo tanto el padre no le da a disposición como tal una función directamente, si no que esta preparado para escuchar eventos del hijo para él así poder reaccionar.

De la siguiente manera.



En este caso podemos ver como el padre crea una función en la cual espera recibir datos del hijo (por decirlo así el canal por el cual escuchara los eventos). Luego en los props del componente del hijo se prepara con esa función par poder escuchar los posibles eventos que emita el hijo.

**Por el lado del hijo tendríamos lo siguiente:**

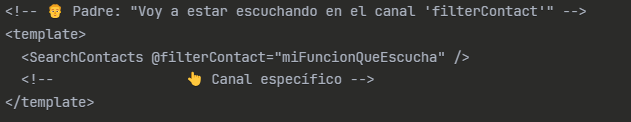
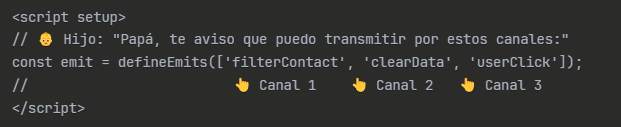
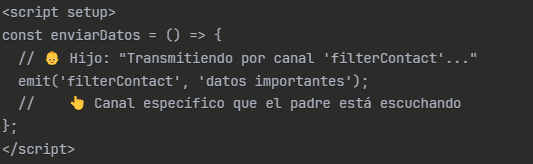


Como observamos el hijo lo primero que hace es definir el canal o los canales por los cuales más adelante podrá enviar información. En este caso el padre esta escuchando por el canal [algoQueHaga], por ende, el hijo tiene que definir ese canal para mas adelante poder enviar información.

Una vez definido el hijo ya puede emitir eventos al padre con información a través de ese canal dedicado entre los dos.

Por ello es que luego llama al canal que había definido arriba y seguido a el pone el mensaje que quiere que escuche el padre.

Un ejemplo más entendible seria este:

* Padre define el canal por el cual escuchara
* Hijo define los canales por los cuales él puede transmitir
* Hijo transmitiendo por el canal que escucha el papa

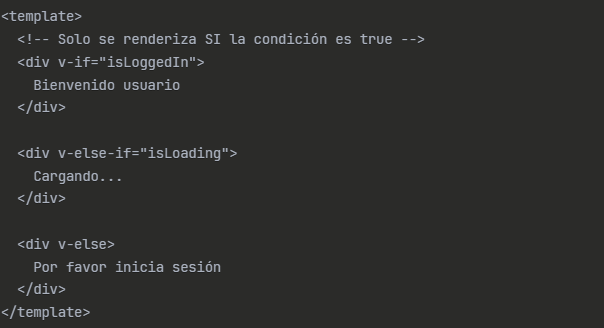
**Características Propias de Vue**

* **Directivas Vue**

Las directivas son atributos que empiezan con V- que le dan instrucciones al HTML sobre cómo debe comportarse dinámicamente.

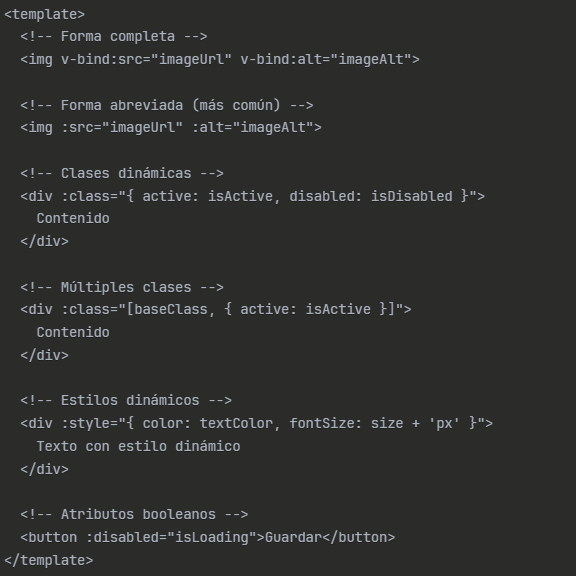
Entre las directivas principales tenemos las siguientes:

* + **V-if / V-else / V-else-if**

Básicamente funcionan como condicionales en los que v-if renderiza lo que contiene el div si la variable a la que es igual existe, de lo contrario entra en juego el v-else-if que iguala a otra variable y renderiza algo diferente y por último si ninguna de las dos pasas, pues directamente se iría por el V-else que es la última opción.

Así que si, funciona tal cual como los condicionales de JS

* + **Atributos dinámicos**



Como podemos observar también podemos poner atributos dinámicos a través de dos formas la forma completa:

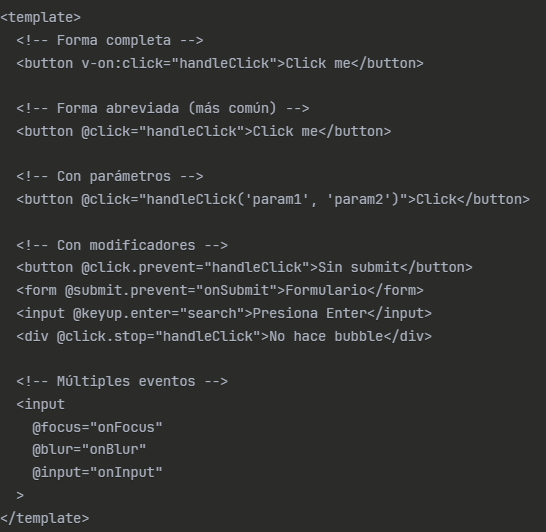


O de la forma abreviada que es la mas general ya que pues se evita escribir demás

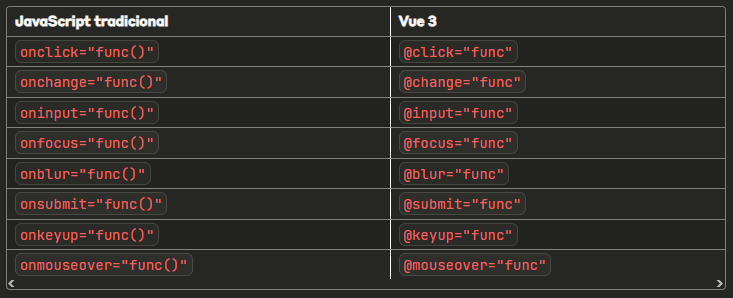


* + **Event Listeners**

Básicamente también tenemos en Vue palabras reservadas para poder escuchar eventos del DOM como el OnClick de toda la vida de JS y demás eventos que se puedan dar, en este caso la forma en la que lo hace Vue es la siguiente:



Basicamente se usa un @con una abreviatura del evento en el caso de OnClick seria @click. Aquí tenemos la tabla mas desglosada de los eventos y la sintaxis que hay tanto JS tradicional y Vue 3



**Enlace Git Hub**

**https://github.com/neunapp/CursoVue3.git**