

Profesor: Johel Adolfo Vargas Sandoval

VLA – Full Stack Front End Developer

Observaciones:

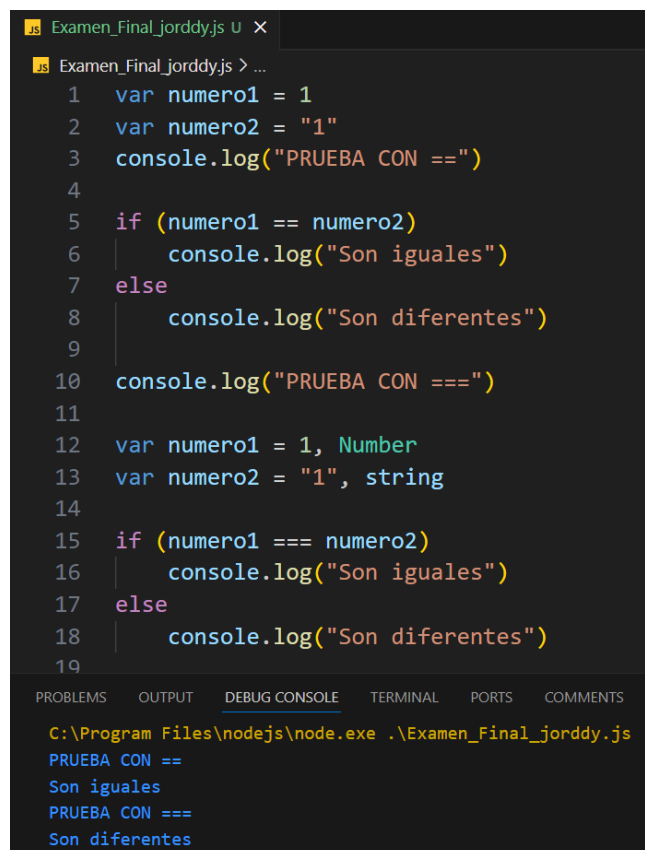
- La ejecución de la actividad es de forma individual.
- Debe presentar un documento PDF con las respuestas y las debidas evidencias de código.

Punto de ejecución:

Solución estudiante: Jorddy Castro Araya

1. ¿Cuál es la diferencia entre `=` y `===`? Argumente su respuesta y presente un ejercicio de ambos casos.

R/ La diferencia es que la segunda es una comparación estricta, además de comparar el valor, también compara el tipo de dato, es decir, si es int/boolean/string.



```
Examen_Final_jorddy.js
1  var numero1 = 1
2  var numero2 = "1"
3  console.log("PRUEBA CON ==")
4
5  if (numero1 == numero2)
6  |   console.log("Son iguales")
7  else
8  |   console.log("Son diferentes")
9
10 console.log("PRUEBA CON ===")
11
12 var numero1 = 1, Number
13 var numero2 = "1", string
14
15 if (numero1 === numero2)
16 |   console.log("Son iguales")
17 else
18 |   console.log("Son diferentes")
19

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
C:\Program Files\nodejs\node.exe .\Examen_Final_jorddy.js
PRUEBA CON ==
Son iguales
PRUEBA CON ===
Son diferentes
```

2. ¿Cuáles son las diferencias entre las variables `var`, `let` y `const`? Presente también un ejercicio que demuestre la falencia de `var`.

R/ Var la puedes sobre-escribir dentro de la función, con let no puedes esto, en el siguiente ejemplo si cambias todos los "var" por "let", te dará un error debido a la línea 4 donde se reescribe el parámetro ingresado para la función, esto se puede ver como una ventaja para evitar replicar variables y perder su valor dentro del código.

```
pregunta2.js U x
pregunta2.js > ...
1  var a = 10;
2  function falenciaVar(b) {
3      var a = 2;
4      var b = 20;
5      var c = b + 1;
6      var d = a;
7      if (c==21) {console.log(true);} //c es igual a 21 porque el parametro inicial a(10) se sobre escribe en linea 4
8      else console.log(false)
9      if (d == 2) {
10         console.log(false); //d es igual a 2 porque tenemos una nueva variable a dentro de la funcion
11         console.log(a);
12         console.log(b);
13         console.log(c);
14         console.log(d);
15     }
16     return(a,b,c,d)
17 }
18 console.log(a) //el valor de a se mantiene en 10 fuera de la funcion
19 falenciaVar(a)
```

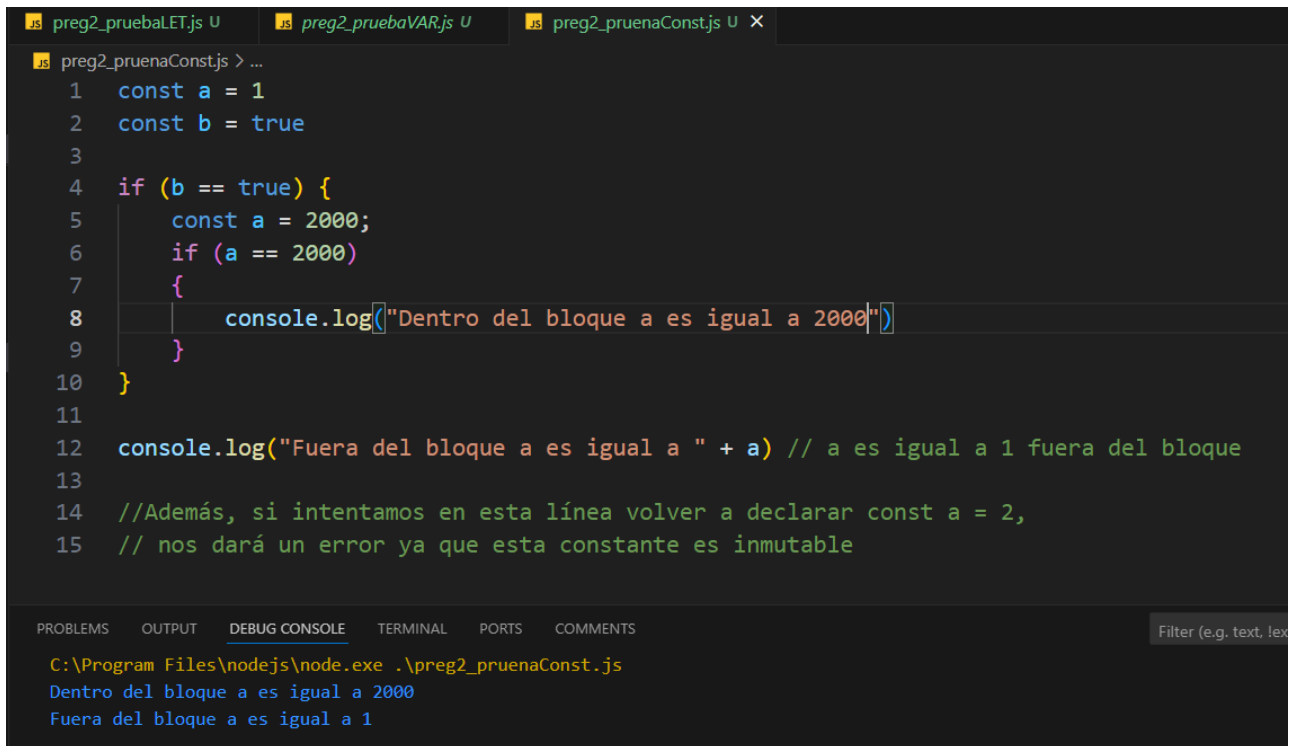
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Filter (e.g., text, lexclude, \escape)

```
C:\Program Files\nodejs\node.exe .\pregunta2.js
10
true
false
2
20
21
2
```

Error que nos muestra si cambiamos var por let:

SyntaxError: Identifier 'b' has already been declared

Mientras que si utilizamos const nos sucede lo mismo, no podemos volver a declararla dentro de la función, pero además, de manera global una vez la inicializas no puedes sobre escribirla, a diferencia de las 2 anteriores, vemos el comportamiento de const en el siguiente ejemplo:



```
1  const a = 1
2  const b = true
3
4  if (b == true) {
5      const a = 2000;
6      if (a == 2000)
7      {
8          console.log("Dentro del bloque a es igual a 2000")
9      }
10 }
11
12 console.log("Fuera del bloque a es igual a " + a) // a es igual a 1 fuera del bloque
13
14 //Además, si intentamos en esta línea volver a declarar const a = 2,
15 // nos dará un error ya que esta constante es inmutable
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Filter (e.g. text, lex)

C:\Program Files\nodejs\node.exe .\preg2_pruenaConst.js

Dentro del bloque a es igual a 2000

Fuera del bloque a es igual a 1

3. Por qué se considera que JQuery no es un lenguaje de programación.

R/ JQuery es una biblioteca de JavaScript. No es independiente, no tiene su propia estructura ni tiene la capacidad de crear programas independientes, si no que se usa a través de un file.js para su uso, es decir, existe dentro de JavaScript.

4. ¿Para qué sirve el document ready en JQuery?

R/ El método `$(document).ready()` sirve para asegurarnos que el código que queremos correr dentro de una función, se ejecute únicamente cuando el DOM (Document Object Model) esté completamente listo, así nos garantizamos evitar problemas/errores como por ejemplo que el script dentro de la función sea leído antes de que el HTML esté completamente disponible.

5. ¿Para qué sirve el append child en javascript? Por favor presente un ejercicio utilizando una etiqueta de select y que sea llenado por un vector de script con nombres de países.

R/ El método `appendChild()` sirve para agregar un nodo hijo a un nodo ya existente, es decir a un nodo padre. En el siguiente ejemplo agregamos `<opciones>` a un `<select>` previamente estructurado como un array de datos:

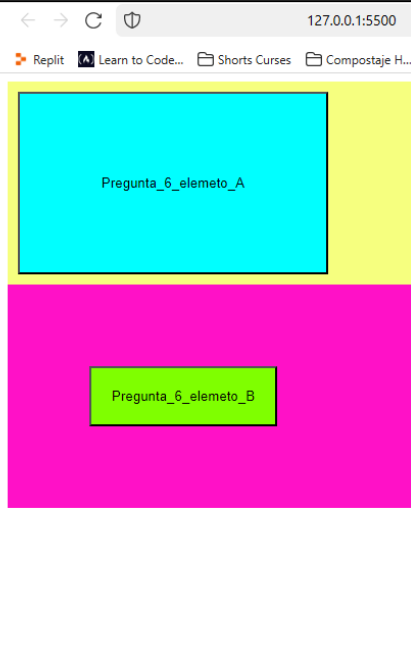
```
pregunta_5.html U X
pregunta_5.html > html
2 <html lang="es">
9 <body>

13 <script>
14   const paises = ['Argentina', 'Brasil', 'Chile', 'México', 'España', 'Colombia'];
15   const selectElement = document.getElementById('paises');
16
17   paises.forEach(function(pais) {
18     const opcion = document.createElement('opcion');
19     opcion.value = pais;
20     opcion.textContent = pais;
21     selectElement.appendChild(opcion); //Acá utilizamos el método para agregar opciones al vector paises
22   });
23 </script>
24 </body>
25
26 </html>
```

6.Cuál es la diferencia entre padding y margin en una implementación de css, realizar un ejercicio de esta diferencia.

R/ Ambas son parte del “Style” de una etiqueta. La diferencia está en Padding se refiere a la distancia entre el centro de la etiqueta y su borde, es decir su relleno, y margin a la distancia entre el borde de la etiqueta y otras etiquetas o elementos adyacentes. A continuación, un ejemplo donde le aplicamos un mayor padding al elemento A y un mayor margin al elemento B:

```
pregunta_6.html U X
pregunta_6.html > ...
1 <div style="background-color: rgb(246, 255, 127);">
2 <button
3   style="padding: 80px;
4   margin: 10px;
5   background-color: aqua;">
6   Pregunta_6_elemento_A</button>
7 </div>
8
9 <div style="background-color: rgb(255, 17, 199);">
10 <button
11   style="padding: 20px;
12   margin: 80px;
13   background-color: chartreuse;">
14   Pregunta_6_elemento_B</button>
15 </div>
16
```



7.Cuál es la diferencia entre em y rem en una ejecución de css.

R/ La diferencia es que em tiene un efecto cascada, ya que el tamaño va a ser relativo a su contenedor. Es decir si tenemos una etiqueta anidada dentro de un div, el valor em va a ser relativo al valor establecido en el contenedor div. Por otra parte, rem mantiene los valores relativos al tamaño de la fuente raíz html, esto hace que los tamaños sean consistentes.

8. En html, ¿Cuál es la diferencia entre head y body?

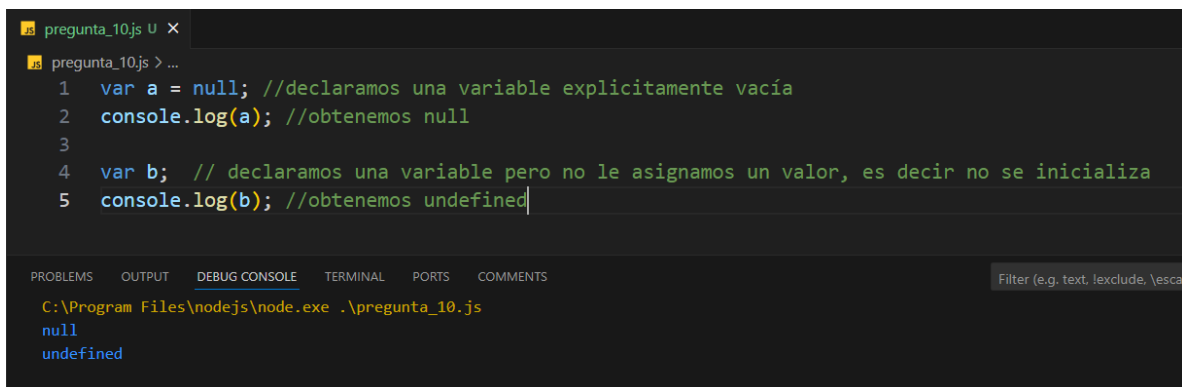
/R En la sección <head> se encuentra información importante para el buen funcionamiento del documento HTML que no se muestra en la página web, como por ejemplo metadatos, título de la página, enlaces externos. En <body> se muestran todos los elementos visibles en la página web con los cuales el usuario final interactúa, como por ejemplo texto, imágenes, enlaces, formularios, etc.

9. ¿Por qué se colocan las referencias de javascript se colocan en el body y no en head?

R/ Por un tema de rendimiento y visualización, ya que cuando se carga la página se lee el documento en orden, y debido a que se lee primero el head antes que el body. Por lo tanto si colocamos las referencias en el head, el navegador tardaría un tiempo cargando estas referencias, antes de leer el contenido visual, es decir, el renderizado de la página quedaría al final del proceso y la experiencia del usuario no sería la mejor.

10. En javascript, ¿Cuál es la diferencia entre null y undefined? Presente un ejercicio con esta situación.

R/ null es una asignación explícita vacía o de valor nulo y undefined es cuando se declara una variable pero no se le asigna un valor. A continuación un ejercicio con esta situación:



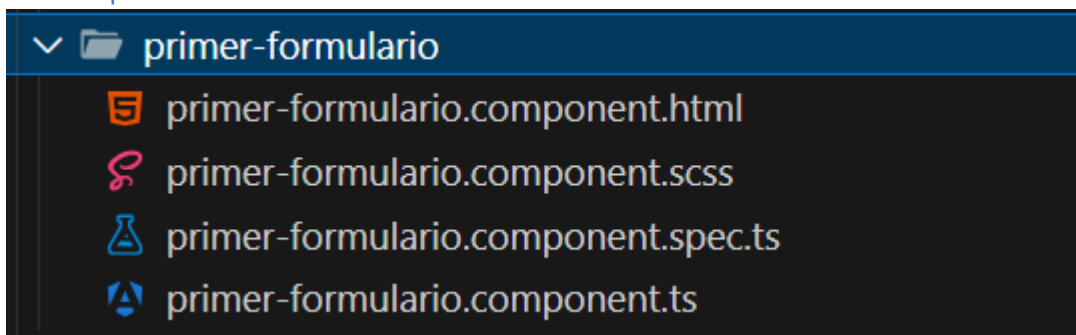
```
pregunta_10.js U X
pregunta_10.js > ...
1 var a = null; //declaramos una variable explícitamente vacía
2 console.log(a); //obtenemos null
3
4 var b; // declaramos una variable pero no le asignamos un valor, es decir no se inicializa
5 console.log(b); //obtenemos undefined
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Filter (e.g. text, lexclude, \escap

```
C:\Program Files\nodejs\node.exe .\pregunta_10.js
null
undefined
```

11. ¿Cuáles son los 4 archivos que se generan al momento de crear un componente en angular? Muestre cuáles son y explique.

R/ A continuación se muestran los 4 archivos que se generan al momento de crear un componente llamado primer-formulario:



- El archivo .html define la estructura de la vista del componente.

- El archivo .scss define los estilos del componente.
- El archivo .spec.ts configura un entorno de pruebas de lógica y comportamiento del componente.
- El archivo .ts define la lógica y comportamiento del componente en TypeScript.

12. En Angular, implemente un ejercicio en el que se utilicen las directivas ngFor y ngIf.

R/

- ngFor para recorrer un vector llamado países (el cual se debe declarar previamente en data), mostrando cada elemento país.

```
<ul>
  <li *ngFor="let pais of paises">{{ pais }}</li>
</ul>
```

- ngIf para habilitar un mensaje en una etiqueta de , en este caso se indica que si “mostrarMensajeError” es verdadero, se debe mostrar el mensajeCorreoPerona.

```
<span *ngIf="mostrarMensajeError" style="color: red;">{{mensajeCorreoPersona}}</span>
```

13. ¿Qué es TypeScript? ¿Cuál es la diferencia con javascript? Presente un ejemplo de

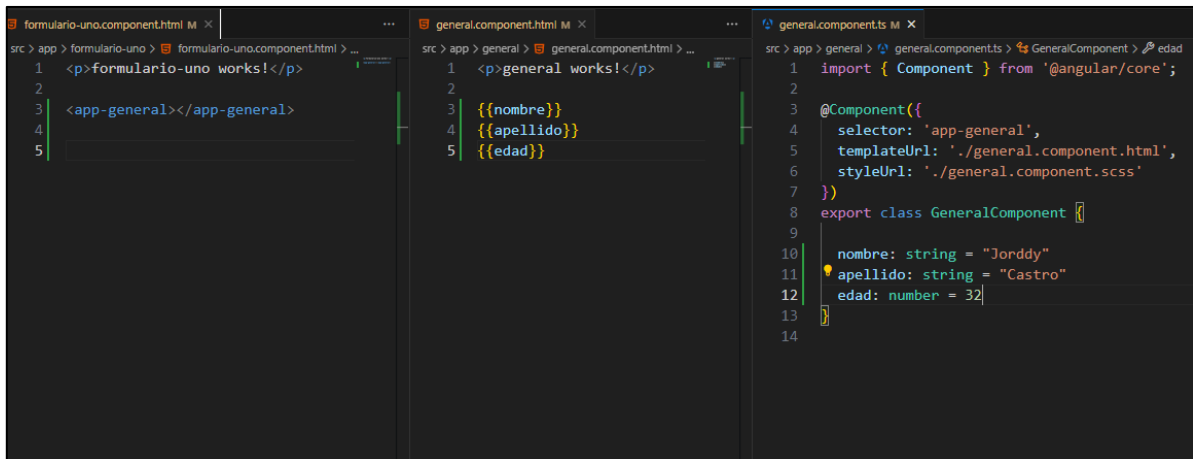
tipificación de datos.

/R TypeScript es un lenguaje de programación. Se dice que es un javascript con “hormonas” debido a que este lenguaje requiere estrictamente definir el tipo de datos (tipificación) de las variables desde su creación. Un ejemplo de la diferencia al crear una variable en ambos lenguajes:

```
let a: string = "jorddy" //así se declara en typeScript
let b = "jorddy " //así se declara en JavaScript
```

14. ¿Que son los Props en un componente? Implemente el ejercicio usando Angular.

R/ Los props son parámetros o propiedades que permiten pasar datos o valores de un componente padre a un componente hijo. Dentro del componente hijo son inmutables, no puedes modificarlos, únicamente son de lectura para renderizar. En el siguiente ejercicio se incorpora el componente hijo(general-component) dentro del componente padre (formulario-uno-component):

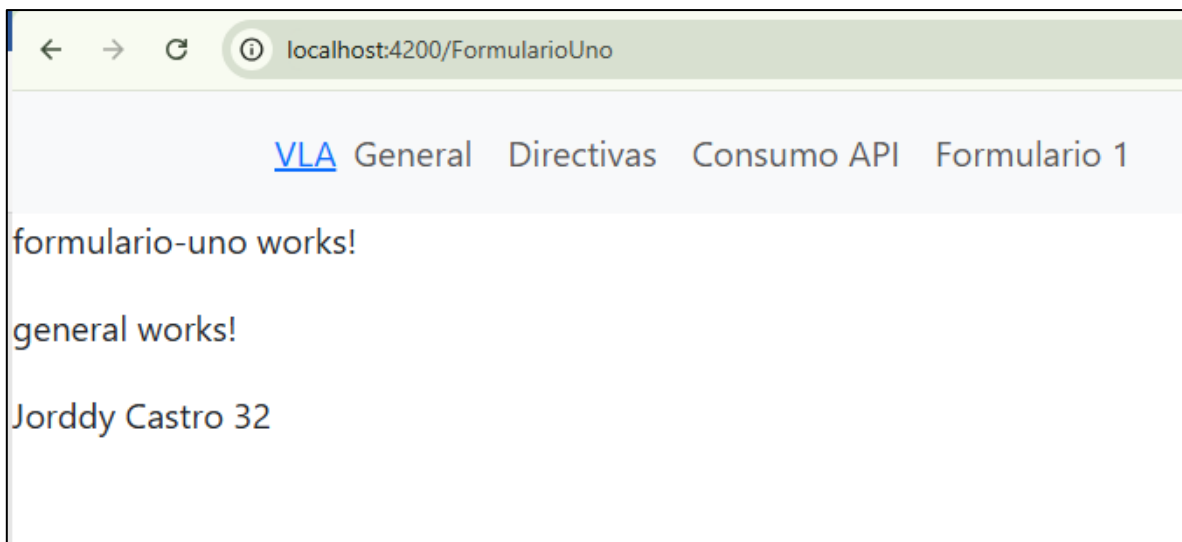


```
formulario-uno.component.html
1 <p>formulario-uno works!</p>
2
3 <app-general></app-general>
4
5

general.component.html
1 <p>general works!</p>
2
3 {{nombre}}
4 {{apellido}}
5 {{edad}}

general.component.ts
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-general',
5   templateUrl: './general.component.html',
6   styleUrls: ['./general.component.scss']
7 })
8 export class GeneralComponent {
9
10  nombre: string = "Jorddy"
11  apellido: string = "Castro"
12  edad: number = 32
13 }
14
```

De esta manera obtenemos un componente dentro de otro:



15. En react, cuál es la diferencia entre un componente de clase y un componente funcional.

R/ Entre ambos hay diferencias de sintaxis/estructura, estado y ciclo de vida:

Sintaxis:

- Componente clase: se utiliza una clase de JavaScript React.component.
- Componente funcional: es una función de JavaScript que devuelve un JSX.

Comportamiento:

- Componente clase: se maneja a través de this.setState().
- Componente funcional: Se utilizan los Hooks que permiten usar el estado a través de useState y useEffect.

Ciclo de vida:

-Componente clase: tienen acceso a métodos que permiten ejecutar código en momentos específicos del ciclo de vida del componente.

-Componente funcional: No tienen métodos de ciclo de vida, en estos se utiliza el hook `useEffect` para emular estos comportamientos de los métodos de ciclo de vida.

16. Qué es un hook en React.

R/ Hook se traduce como “engancher”, esto permite funcionalidades adicionales de estado, efectos secundarios sobre los componentes funcionales. Estos son:

- `useState`
- `useEffect`
- `useContext`
- `useReducer`
- `useRef`

17. Cuando uno usa el hook `use State`, para qué sirve el `set` en la asignación de una variable.

R/ Sirve para establecer el valor del estado y actualizarlo de manera controlada.

18. ¿Cómo se implementa la suscripción a eventos en React? Proporciona un ejemplo utilizando un botón y también un input.

R/ La suscripción a eventos en REACT se implementa a través de elementos JSX. Para un botón utilizamos `onClick` y para un input `onChange`, es importante notar que estos eventos deben estar en camelCase.

Ejemplo botón `onClick` utilizando un ejemplo con un contador:

```
18 import React, { useState } from 'react';
19
20 function MiComponente() {
21
22   const [contador, setContador] = useState(0);
23   const manejarClick = () => {setContador(contador + 1)};
24
25   return (
26     <div>
27       <h1>Contador: {contador}</h1>
28       <button onClick={manejarClick}>Incrementar contador</button>
29     </div>
30   );
31 }
32
```

Ejemplo input `onChange` utilizando un input:


```

18 import React, { useState } from 'react';
19
20 function MiPrimerComponente() {
21   const [texto, setTexto] = useState('');
22   const manejarCambioInput = (event) => { setTexto(event.target.value);};
23
24   return (
25     <div>
26       <h2>Texto ingresado: {texto}</h2>
27       <input type="text" value={texto} onChange={manejarCambioInput} placeholder="Escribe algo..." />
28     </div>
29   );
30 }

```

19. En Vue, cuál es la diferencia entre v-bind y v-model. Realice un ejercicio y capture la evidencia.

R/ Ambas son directivas de Vue. v-bind se utiliza para enlazar atributos HTML a expresiones, permitiendo que el atributo sea dinámico y cambie según el estado al que está vinculado. Mientras que v-Model permite una vinculación de manera bidireccional entre el estado y los elementos de entrada, como por ejemplo en un input donde el usuario introduzca un valor a través de la interface.

Ejemplo v-bind:

En el siguiente código se muestra en la línea 13 la inserción de una imagen, para esto utilizamos el v-bind para vincular esta imagen a una variable de tipo string llamada "imagenXbox" declarada en la línea 143. Por lo tanto si esta variable cambia, también cambia la imagen que se muestra debido a la línea 13, ya que están directamente vinculadas, sin embargo este vínculo es en una sola dirección.

13	
143	imagenXbox: "../assets/img/xbox.png" as string

Ejemplo v-model:

En el siguiente código se muestra en la línea 6 el uso del v-model en un input.

```

1 <template>
2   <div class="container">
3     <div class="contenedorSolicitud">
4       <h1>{{userName_input}} Submit your request now!</h1>
5     <div>
6       <input v-model.lazy="userName_input" class="form-control" placeholder="Your Name" :style="{backgroundColor:userColor}">

```

Notamos como al introducir el nombre en 'Your Name' (que es asignado al v-model "userName_input") también se agrega el nombre en la parte superior, ya que el v-model de la línea 6 en el código Vue cambia.

Submit your request now!	Jorddy Submit your request now!	Adolfo Submit your request now!
<input type="text" value="Your Name"/>	<input type="text" value="Jorddy"/>	<input type="text" value="Adolfo"/>
<input type="text" value="Phone Number"/>	<input type="text" value="Phone Number"/>	<input type="text" value="Phone Number"/>
<input type="text" value="Email"/>	<input type="text" value="Email"/>	<input type="text" value="Email"/>
<input type="text" value="Select your favorite consola"/>	<input type="text" value="Select your favorite consola"/>	<input type="text" value="Select your favorite consola"/>
<input type="text" value="Additional Comment"/>	<input type="text" value="Additional Comment"/>	<input type="text" value="Additional Comment"/>

20. Describe el uso de las computed properties en Vue.js. Proporcione un ejemplo de este tema en Vue.

R/ Son propiedades reactivas, esto quiere decir que si el data o las props se modifican, estas propiedades se vuelven a calcular de manera automática, lo que mejora el rendimiento.