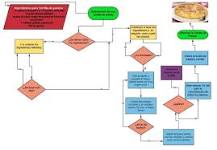


**Taller de Nivelación Curso Desarrollo Web Frontend  
Por:  
Mónica Y. Rincón Rincón  
MÓDULO SOBRE HTML, CSS Y RESPONSIVE DESIGN  
Preguntas teóricas**

1. *¿Qué es la lógica en el contexto de la programación? explicar por qué es importante en el desarrollo web Frontend.*

La lógica en programación es la *organización* y *planificación* de instrucciones en un algoritmo (conjunto de instrucciones finitas), con el objetivo de ser implementado en un programa o software. La lógica de programación consiste en la organización y planificación coherente de las instrucciones necesarias para ejecutar con éxito un programa.

La lógica de la programación es la organización coherente de las instrucciones del programa para que su objetivo sea alcanzado. En la lógica de programación es *donde aplicamos todos los conceptos de algoritmos*, la definición paso a paso y trasladamos toda la lógica del algoritmo desarrollado a un lenguaje de programación.



1. *Definir el concepto de “algoritmo” y proporcionar un ejemplo sencillo de un algoritmo relacionado con la lógica de programación.*

Un *algoritmo*es un conjunto finito de instrucciones que, si se siguen rigurosamente, llevan a cabo una tarea específica. Un algoritmo es la descripción de cómo hacer la lógica en programación. Son la secuencia de pasos con la intención de lograr un objetivo.

Un algoritmo tiene las siguientes características:

* *Entrada*: Entran cero o más valores.
* *Salida*: Por lo menos produce un valor.
* *Definido*: Cada instrucción es clara y no ambigua.
* *Finito*: Si realizamos el seguimiento al algoritmo, entonces, en todos los casos, el algoritmo termina en un número finito de pasos.
* *Efectivo*: Cada instrucción deberá ser tan básica que puede llevarse a cabo, en principio, por una persona, sólo usando lápiz y papel. No sólo la instrucción deberá ser definida sino también viable o factible.

1. *¿Qué son estructuras de control en la programación?, ¿Cuáles son los tipos de estructuras de control y las estructuras más comunes de cada tipo?, Describir al menos dos tipos de estructura de control, explicar por qué son importantes y proporcionar ejemplos de cada uno y de cómo se utilizan en el desarrollo web Frontend.*

*Las estructuras de control son* el conjunto de reglas que permiten controlar el flujo de ejecución de las instrucciones de un algoritmo o de un programa. Las estructuras de control son *importante*s porque nos dan el poder de **alterar, controlar o modificar el orden o el flujo en el que se ejecutan las instrucciones de un software como nosotros queramos**. Gracias a las estructuras de control podemos abstraer algoritmos o secuencias de instrucciones en un software para lograr su objetivo.

*Los tipos de estructura de control son tres*: Secuenciales, Selectivas, Iterativas.  
*Selectivas:*

Las estructuras *selectivas* nos sirven cuando necesitamos que se evalúe el valor de alguna variable o de alguna condición para decidir qué instrucciones ejecutar a continuación.

Las estructuras más comunes de las estructuras selectivas son:

*Selectivas simples*: Evalúan un valor o una condición y determinan las instrucciones a ejecutar en caso de cumplirse la condición.

#### ***Selectivas dobles*:** Evalúan un valor o una condición, determinan las instrucciones a ejecutar en caso de cumplirse la condición y también las instrucciones a ejecutar en caso de no cumplirse.

*Selectivas múltiples o anidadas:* Permiten combinar selectivas simples y dobles para crear estructuras y condiciones más complejas cuando el algoritmo en cuestión lo necesite.

### *Iterativas: Iterativas, de iteración, de repetición o repetitivas:* Este tipo de estructuras de control nos sirven cuando necesitamos que se ejecute un conjunto específico de instrucciones en diversas ocasiones. La cantidad de veces que se repite dicho bloque de acciones puede ser estático o puede depender del valor de alguna variable o de alguna condición. *Iterativas con cantidad fija de iteraciones:* Se utilizan cuando *a priori*se conoce la cantidad de ocasiones que debe repetirse un bloque de instrucciones. Normalmente, usan una variable de iteración o índice para contar la cantidad de repeticiones que se han realizado. *Iterativas con cantidad variable de iteraciones:* Se utilizan cuando la cantidad de ocasiones que debe repetirse un bloque de instrucciones está determinado por una condición. Por lo regular, existen dos variantes: repetir un bloque de instrucciones mientras se cumpla una condición o repartirlo hasta que se cumpla una condición.

1. *Describir cómo se declaraban las variables y constantes en JavaScript antes de la introducción de ECMAScript 6 (ES6). Explicar cómo ES6 mejoró la declaración de variables y constantes, y mencionar los problemas que esta mejora resuelve en el desarrollo web Frontend.*

Antes de (ES6), las variables se declaraban con la palabra *var.* Después de (ES6) las variables se declaran con la palabra *let* y las constantes se declaran con la palabra *const.*

*5. ¿Cómo se declaran las funciones en JavaScript y cuál es la diferencia entre una declaración de función, una expresión de función y una función de flecha (arrow function)? Proporcionar ejemplos de cada una*.

*Las*[*funciones en JavaScript*](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Functions) son una manera de escribir una acción de la que obtendremos un resultado según ciertos parámetros. Como su nombre lo indica**, las funciones nos permiten obtener un resultado en función de lo que definamos. Una función tiene una tarea en específica y es reutilizable. La función permite reutilizar código sin duplicar código. Es un bloque de código que permite hacer una tarea específica. *Para declarar una función*, usamos la palabra reservada *function,* y al lado de la palabra reservada va el nombre de la función*.* Una función siempre va a llevar paréntesis () y llaves {}. Dentro de las llaves va el *cuerpo* de la función. *Invocación de funciones*: Para que una función se invoque, hay que llamar el nombre de la función y abrir y cerrar paréntesis. La invocación de la función se hace por fuera de las llaves. Hay dos tipos de funciones: Declarativas y expresivas. *Declarar una función:* Es usada con la palabra reservada function. Para declarar una función se debe de usar la siguiente sintaxis**

**Function () { Consol.log()**

**}**

**Para ejecutar una función, después de la función, indico el nombre de la función, abro y cierro paréntesis. En la función declarativa, no es necesario declarar una función antes de llamar o ejecutar la función.**

*Expresar una función:* Una función expresiva no tiene nombre. No es necesario poner la palabra function, son conocidas también como funciones anónimas. Con la función expresiva, primero necesitamos hacer la función para después llamarla. Las funciones anónimas se tienen que almacenar en una variable para poder ser llamadas después. La función anónima no goza del concepto de hoisting, es decir, si yo llamo a la función antes de haberla declarado habrá un error de inicialización. Aparte de la función, a la función se le pueden pasar argumentos o parámetros. Los argumentos o parámetros van dentro de los paréntesis. Y cuando se llame la función, también se debe de llamar el argumento.

*Función flecha (arrow function):* Tiene la siguiente sintaxis. () => {} La primera característica de la función flecha es que es una función anónima, es decir, para poderla declarar, necesito almacenarla en una variable para poderla llamar después. La segunda característica es que no necesitan de la palabra reservada function. La tercera característica es que no es necesario escribir la palabra reservada *retorn.* La cuarta característicaes que si la función tiene solo un parámetro podemos omitir los paréntesis y si sólo es una línea de código, podemos omitir las llaves. Esto nos permite declara una función de una manera más simplificada, es decir, nos permite optimizar el código. Si hay más de un parámetro, sí será necesario usar los paréntesis. Cuando la función no tiene parámetros, los paréntesis vacíos serán obligatorios.Si después de la flecha se necesita especificar un objeto (el cual debe de estar entre llaves), es necesario envolverlo dentro de paréntesis, esto con el objetivo de no confundir la estructura de una función flecha con las llaves del objeto.

*6. ¿Por qué es necesario el uso de funciones en el desarrollo web Frontend? Enumerar al menos tres razones fundamentales y proporcionar ejemplos de situaciones en las que las funciones son esenciales. Además, mencionar la ventaja de las funciones flecha en el contexto de estas razones.*

Es necesario ya que nos permite construir un código reutilizable; Nos permitirá construir un código que dará solución a un problema que se repita sin necesidad de invocar nuevamente el código ya que ejecutará instrucciones de la manera en que nosotros indiquemos previamente.

*7. ¿Cuál es la diferencia entre parámetro y argumento?*

Los parámetros sirven para pasar valores que serían utilizados por el código de una función. Los argumentos son los valores que le envío a los parámetros.

*8. Definir el concepto de Callback y proporcionar un ejemplo práctico.*

Un callback es una función que se puede pasar a otra como argumento y se ejecuta después de que se haya completado cualquier operación.

*9. ¿Qué es el hoisting en JavaScript y cómo afecta a las variables y funciones? Proporcionar ejemplos de hoisting en declaraciones de variables y funciones.*

El hoisting es la elevación de funciones que hace javascript al momento de ejecutar el código. Coge la función, la llama, y luego define la función más abajo. Cuando el programa inicia, lo que hace es mover la declaración de las variables al comienzo, al principio del scop del bloque. El hoisting es el comportamiento de javascript de mover todas las declaraciones al inicio de cada scope*.(alcance y ámbito de una variable, dónde y cuándo puedo utilizar una variable o puedo acceder a ella.) el scope es global, local, bloque.* ¿cómo afecta a las variables y funciones? Permite que se llamen las variables y las funciones antes de definir la función

*10. Definir brevemente el concepto de objeto en JavaScript y cuál es la visión general sobre este concepto. Indicar también, cómo se declaran estas estructuras de datos.*

Un objeto es un elemento que creamos que tiene diferentes propiedades, parámetros valores o atributos. Un objeto es la colección de unas propiedades. Para declarar un objeto, primero se debe declarar una variable, le asigno un nombre a la variable, abro y cierro llaves, y dentro de las llaves están los datos o la información del objeto (atributos). Dentro del objeto pueden ir números, strings, o variables verdaderas o falsas.Atributo: son variables que tiene dentro un objeto.

*11. ¿Qué son propiedades? Y ¿ cuál es la diferencia entre una propiedad y un método en un objeto?*

Las propiedades son los elementos o las características de un objeto. La diferencia entre una propiedad, un método y un objeto es que las propiedades definen las características de un objeto. Una propiedad es una asociación entre un nombre y un valor. Un método es una función puesta en un objeto.

*12. Explicar las dos formas de acceder a una propiedad de objetos e indicar las situaciones en que conviene usar una manera sobre la otra.*

Para acceder a una propiedad, abro corchetes dentro del objeto, y en el corchete se escribe la propiedad a la que quiero acceder. También podemos acceder a través del punto, dentro del objeto se pone un punto, y se indica a la propiedad que se quiere acceder.

*13. ¿Existe alguna forma de recorrer las propiedades de un objeto? En caso negativo, explicar por qué no es posible. En caso positivo, proporcionar un ejemplo. Mencionar una situación en la cuál sea muy útil recorrer las propiedades de un objeto.*

*14. ¿Por qué son útiles los objetos en la programación web y qué tipos de datos pueden almacenar?*

Porque enviar un solo objeto (dato) que contiene varia información, es mucho más eficiente que enviar varios elementos (o datos) de manera individual. Los objetos permiten que la información sea más fácil de procesar.

*15. ¿Qué es un array en JavaScript y por qué son esenciales?*

Un Array es un tipo de dato. Se identifica con los corchetes []. En los arrays se agrupan valores que representan un conjunto de datos. Un Array en JavaScript *es****una colección o agrupación de elementos en una misma variable, cada uno de ellos ubicado con referencia a la posición que ocupa dentro del mismo. Es un contenedor que nos permite tener varios datos almacenados al mismo tiempo.*** *¿*Por qué son esenciales los array? Porque permite organizar y acceder a datos.Los*arreglos* permiten *organizar las variables de entrada* y procesarlas o modificarlas  dentro de un estructura.Entonces, el hecho de modificar datos almacenados dentro de una estructura en filas y columnas *permite aplicar mayor complejidad y lógica de programación*, aparte de satisfacer la necesidad de organizar datos aunque sea de manera temporal.

*16. ¿Cómo acceder a un elemento dentro de un array? Explicar con un ejemplo.*

Para poder acceder a uno de estos elementos, debemos escribir el nombre de nuestro array, y luego insertar entre corchetes la posición del elemento al que queremos acceder. Los elementos en los array inicializan en cero.

**MÓDULO SOBRE HTML, CSS Y RESPONSIVE DESIGN**

**Preguntas teóricas**

*1. ¿Qué significa HTML y cuál es su función en el desarrollo web?*

Es un lenguaje de marcado (lenguaje de marcado de hipertexto) que permite crear un formato al momento de desarrollar o la elaborar una página web. ¿cuál es su función en el desarrollo web?

Describir la estructura básica de una página y organizar la forma en la que se mostrará el contenido de la misma.

*2. ¿Cuál es la estructura básica de un documento HTML?* Describir las etiquetas esenciales.

Las etiquetas son las que se usan para darle formato al texto y luego de ser procesadas, le indican al computador lo que debe de hacer. Las estructuras básicas son:

<!DOCTYPE **html>** Apertura etiqueta html.

<html Lang =” en” Le indica al navegador que el documento está basado en el estándar html

*Lenguaje html*

<head> Representa la raíz de un doc. html

<meta … /> todos los elementos de html deben

<meta … /> de estar aquí.

<link …... />

</head>

<body> Alberga todo lo que se quiere mostrar

<body> en la página web

**<html/>** Cierre etiqueta html

*3. ¿Qué es CSS y cuál es su propósito en el desarrollo web?*

Es otro lenguaje (hoja de estilo en cascada) de un navegador web. Su propósito es permitir de forma automática y masiva darle diseño, estilo, y personalizar una página web para definir el aspecto visual de una página web.

*4. ¿qué son selectores CSS?*

Es una referencia que permite asignar las propiedades (los estilos) a los elementos (etiquetas) de html.

*¿cuáles son los principales tipos de selectores?*

Universal, de tipo o etiqueta, selector descendente, de clase, de ID.

*¿por qué es importante entender la especificidad en el contexto de las hojas de estilo en cascada (CSS)?*

Porque nos permite conocer cuáles son los valores de una propiedad CSS más relevantes para un elemento, y, por ende, cuál valor será aplicado. *Describir al menos tres tipos de selectores CSS y cómo la especificidad afecta a la aplicación de estilos en un proyecto de desarrollo web Frontend*.

Selector universal: Con la referencia del asterisco (\*), se otorga el mismo estilo a todo el documento.

Selector por id: Asignando un atributo a la etiqueta, el selector por id permite dar estilos a una etiqueta específica. Los id no se repiten en el código.

Selector por clase: Es igual que el selector por id, pero se diferencia porque las clases si se pueden repetir en el código.

- *Proporcionar ejemplos de situaciones en las que se utiliza la especificidad de selectores para resolver conflictos de estilos.* Hay conflicto de estilos cuando una etiqueta contiene a otra. Para resolver los conflictos de estilo acudimos a la jerarquía de la especificidad, esta es: Id, clase, y etiqueta.

*5. Explicar las diferencias entre los estilos en línea (inline), estilos internos (embedded) y estilos externos (external) en CSS. Indicar cuál de los tres estilos es el recomendado usar y por qué.*

En línea: Permite poner los estilos dentro de las etiquetas del html.

Internos: Permite poner los estilos, pero dentro de la etiqueta <head> del html, anclando dentro de la etiqueta style, seguido del atributo type, el css. Dentro de la etiqueta se definen los distintos estilos.

Externos: Permite separa los estilos de la estructura html indicando en la extensión del nuevo archivo que es un documento css. En el html se debe enlazar los estilos del css a través de la etiqueta link.

Es más recomendable usar estilos externos porque puedo vincular el css a varias páginas y dar estilo a todas en la misma hoja de estilos. También es mejor al momento de dar estilos a sitios web amplios, grandes.

*6. ¿Qué es flexbox y cómo se utiliza para el diseño de páginas web?*

Es una forma de maquetar un sitio web que permite colocar cajas de forma flexible. Para usar flexbox, primero se debe de tener un *contenedo*r (elemento padre) que albergue todos los elementos (elementos hijos) que son las cajas flexibles que se pueden adaptar a varias posiciones.

*7. Explicar cómo se emplean las propiedades flexbox y explicar la función de las principales propiedades en la creación de diseños flexibles.*

Las propiedades de Flexblox se emplean manejando las posiciones de los elementos(layout) en una sola dimensión a la vez (como fila o como columna) ya que Flexbox maneja dos ejes (el principal y el cruzado).

*explicar la función de las principales propiedades en la creación de diseños flexibles.*

La principal propiedad de flexbox es flex-direction. Esta a su vez tiene 4 valores principales. El valor row, o row- reverser, correrá a lo largo de la fila según la dirección que le indiquemos (marge inicial y margen final). El valor columna o columna-reverse correrá a lo largo de la columna según la dirección que se le indique (desde el borde superior hasta el borde inferior).

*8. ¿Qué es CSS Grid Layout?*

Es una propiedad de CSS que contiene funciones de diseño.

*en qué se diferencia de flexbox?*

A diferencia de flexbox, el css grid permite dividir la página en varias áreas o regiones prmitiendo pisicionar elementos en filas y columnas al mismo tiempo.

*9. Proporcionar un ejemplo de cómo crear una cuadrícula sencilla con CSS Grid.*

*<div class= contenedor principal>*

*<div class= contenedor 1> </div>*

*<div class= contenedor 2> </div>*

*<div class= contenedor 3> </div>*

*<div class= contenedor 4> </div>*

*</div****>***

*.contenedor-principal{*

*display:grid*

*grid-template-columns: auto auto*

*}*

*10. ¿Qué significa el diseño responsivo en el contexto del desarrollo web?*

Es crear un sitio web usando html y CSS de forma que se adapte bien en todos los formatos de un dispositivo (celular, Tablet, escritorio)

*11. Enumerar al menos tres técnicas o estrategias utilizadas para lograr el diseño responsivo en una página web.*

1. Añadir al head del documento html un meta de viewport.

<meta name=”viewport”>

2.En las imágenes, usar un withe del 100%, de esta forma el tamaño de la imagen se irá adaptando al tamaño de la pantalla.

3. Media queries: @media. Permite definir estilos completamente diferentes para diferentes tamaños del navegador.

**MÓDULO SOBRE DOM E INTERACCIÓN CON EL DOM**

**Preguntas teóricas**

*1. ¿Qué es el DOM (Modelo de Objeto de Documento) en el contexto de la programación web?*

El dom es la representación del html (la cual está formada por múltiples etiquetas html) *en una estructura de datos de árbol conformada por nodos,* el cual se puede crear o modificar para desarrollar lo que necesitemos.

HTML

<body>

<head>

<tittle>

<footer>

<main>

<header>

*2. ¿Cuál es la diferencia entre el DOM y el HTML en una página web?*

El DOM es una API definida para representar e interactuar con cualquier documento html. Esta API permite modificar el documento a través de lenguajes de programación como javascript aportando dinamismo y una mejor experiencia a los usuarios. Se trata de un esquema en forma de árbol de nodos, en el que cada elemento es un nodo que a su vez puede anidar otros nodos. Se diferencia del html porque éste solo “demarca”, crea las etiquetas para que a través del DOM se pueda interactuar con ellas. A través del objeto document podemos acceder al DOM, que representa la página que ha cargado el navegador. Con el método querySelector podemos buscar un elemento en el DOM a través de un selector.Con el método createElement, creamos un elemento que se guardará en una variable.

DocumentElement representa la etiqueta del html en el DOM.

*3. ¿Por qué es importante entender y manipular el DOM en el desarrollo web Frontend?*

Porque permite crear páginas web dinámicas que dan la oportunidad a los usuarios de interactuar con la misma sin necesidad de recargar la página manualmente brindando una mejor experiencia de usuario.

*4 ¿Qué son los eventos del DOM y cuál es su función en una página web?*

Son las ejecuciones de código. Su función es que cada vez que el usuario hace ciertas acciones en el sitio web, se ejecutan esas acciones (ejemeplo, el click)

*5. Proporcionar ejemplos de eventos prácticos y comunes, como “click”, “submit”y “load o DOMContentLoad”.*

*6. ¿Por qué es importante manejar eventos en la interacción usuario-web y cómo se añaden controladores de eventos a los elementos del DOM?*

Porque permite controlar las acciones del usuario y definir ciertos comportamientos y el comportamiento del documento, que suceda en cierto momento o cuando se cumplan algunas condiciones. Los controladores de eventos se añaden a través de *las funciones* que se ejecutan en un evento (manejadores de eventos). Cuando una función se convierte en un manejador de evento, es decir, se ejecuta en un evento determinado, nosotros podemos acceder a un objeto que es el evento en sí con la palabra reservada evento, esto es lo que denominados manejadores de eventos.

*7. Describir al menos tres métodos para seleccionar elementos del DOM en JavaScript.*

*8. ¿Cómo se crea un nuevo elemento HTML y se agrega al DOM utilizando JavaScript?*

Con el método. createElement especificando el elemento a crear. Después de creador, se agrega al DOM con el método. getElementbyid.

*9. ¿cuál es la importancia de la manipulación del DOM en la creación de aplicaciones web dinámicas e interactivas?*

*10. Explicar brevemente los conceptos “eventbubbling”y “eventdelegation”en el contexto de eventos del DOM.*

*Eventbubbling:* Es un concepto que se refiere al orden de difusión de eventos en el “árbol” jerárquido de las etiquetas HTML (DOM). Todo el evento viaja por todo el “árbol” DOM hacia arriba.

*Eventdelegation:* La delegación de eventos permite detectar un evento, capturarlo y propagarlo siempre que haya varios elementos manejados de manera similar. En vez de asignar un manejador a cada uno, se pone un único manejador.

*11. ¿Por qué son relevantes los conceptos “eventbubbling”y “eventdelegation”enla gestión de eventos en páginas web con múltiples elementos interactivos?* Porque facilita el desarrollo web y la eficiencia al momento de crear una página.

**MÓDULO SOBRE COMUNICACIÓNCON EL SERVIDOR (STORAGE, PROMESAS ASINCRONÍASY PETICIONES HTTPS)**

**Preguntas teóricas**

*1. Definir brevemente el concepto de localStorage y sessionStorage.*

ObjetoStorage: Es una API de almacenamiento web que permite almacenar datos de manera local en el navegador, sin necesidad de realizar alguna conexión a una base de datos. El localStorage y el SesssionStorage son propiedades que acceden al objetoStorage, tiene la funcionalidad y la posibilidad de almacenar datos de manera local

*2. Describir las diferencias claves entre localStorage y sessionStorage.*

La diferencia entre cada uno es que el localSotorage almacena la información de manera indefinidao hasta que se decida limpiar los datos del navegador de manera intencional. El sessionStorage almacena datos mientras la pestaña del navegador donde se esté utilizando siga abierta, una vez que se cierre, se elimina la información de manera automática. El localSotage solamente almacena strings.

*3. ¿Por qué son útiles almacenar datos en el navegador y cuáles su límite de capacidad?*

Es útil porque permite identificar y recordar a un usuario. Las páginas web puedan guardar configuraciones, preferencias, evitar tener que iniciar sesión cada vez que se cierra una pestaña permitiendo tener una mejor experiencia de usuario. El límite de capacidad de este almacenamiento de datos viene dado por el navegador y actualmente se encuentra en 5 MB por dominio en Chrome, Firefox y 10 MB en Internet Explorer.

*4. ¿Qué son las promesas en JavaScript y para qué se utilizan en el desarrollo web?*

Una promesa es un mecanismo que tiene javascript que permite controlar el asincronismo. Si realizamos una petición a una base de datos y la respuesta a esa petición toma un tiempo, estamos frente a una situación de asincronismo. Las promesas se utilizan en el desarrollo web para esto, para esperar la respuesta de una petición dirigida a una base de datos.

*5. Explica el concepto de asincronía en programación y cómo las promesas ayudan a manejar operaciones asincrónicas.*

La asincronía plantea cuándo tendrá lugar una respuesta solicitada por ejemplo, a una base de datos. En la asincronía, la respuesta sucede en el futuro. La operación se ejecuta pero sin saber cuándo vendrá la respuesta.

*6. Proporciona un ejemplo de cómo se utiliza una promesa para realizar una operación asincrónica, como una solicitud de red.*

*7. ¿Qué es JSON Server y cómo se utiliza en el desarrollo web?*

Es un servidor local a través de la cual podemos crear nuestro propio endpoint (API). Se puede acceder a los endpoint a través de los métodos GET/POST/PUT/DELETE/OPTIONS.

*8. ¿Por qué es útil simular una API REST falsa con JSON Server en el desarrollo frontend?*

Porque facilita la creación de servicios web como base de datos. Permite hacer pruebas de manera local para poder hacer la demostración de un proyecto.

*9. ¿Cuáles son las diferencias claves entre los métodos del objeto promesa .then() .catch() y las palabras claves async/await?*

*Son dos formas distintas de gestionar promesas en javascript, .then y .catch fueron las primeras formas de gestionar las promesas. .then(), que recibe una función*callback*y retorna un "objeto-promesa".****No es un retorno de los datos, es la***promesa***del retorno de esos datos.*** *Entonces, podemos escribir el código de lo que sucederá a continuación con los datos recibidos por*Promise*, y JavaScript esperará a que*Promise*se resuelva sin pausar el flujo del programa. El resultado puede o no estar listo todavía, y no hay forma de obtener el valor de una Promesa de forma síncrona; Solo es posible pedirle a Promise que ejecute una función*callback*cuando el resultado esté disponible, ya sea lo que se solicitó (los datos de la API, por ejemplo) o un mensaje de error si algo salió mal con la solicitud (el servidor puede estar fuera de servicio, por ejemplo). Además del método .then() que recibe el objeto-Promise para ser resuelto, el método .catch() retorna en caso de rechazo de la Promesa.*

*10. Proporciona un ejemplo de cómo configurar una API falsa con JSON Server y realizar solicitudes (GET, POST, PUT, PATCH y DELETE)a través de ella.*

*11. Describe las diferencias entre Fetch y Axios como métodos para realizar solicitudes HTTP en JavaScript.*

*12. ¿Por qué es importante considerar las peticiones HTTP en aplicaciones web modernas?*

*13. Proporciona ejemplos de cómo se utilizan Fetch y Axios para realizar solicitudes GET y POST*.

*14. Explica la importancia del manejo de errores al trabajar con promesas en el desarrollo web.*

*15. Describe cómo se manejan los errores en las promesas, incluyendo el uso de catch.*

*16. ¿Cuáles son las diferencias claves entre los métodos del objetopromesa .then().catch()y la estructura try/catch?*

*17. Proporciona un ejemplo de cómo se puede manejar un error en una promesa al realizar una solicitud de red.*

**MÓDULO SOBRE REACT JS  
Preguntas teóricas**

1. *Explicar brevemente el concepto de ReactJS y sus principales características.*

Es una de las principales *librerías* de JavaScript de código abierto. Se utiliza para construir *interfaz* de usuario virtual (UI) basada en *componentes* y para crear aplicaciones web del lado del cliente y del lado del servidor.El componente es interactivo y reutilizable, es decir, no es necesario repetir el código, solo se tiene que importar la lógica del componente en la parte del código que se necesite o donde se vaya a utilizar.

*Características:*

1. El estilo de React es declarativo, no imperativo.
2. Tiene un flujo de datos, un patrón de funcionamiento unidireccional
3. Tiene un DOM virtual
4. Es isomorfo
5. Tiene sintaxis propia llamada JSX
6. Permite crear componentes de diversas maneras
7. Implementa un ciclo de vida para los componentes
8. *Definir qué es un componente en ReactJS y mencionar los tipos de componentes que existen.*

Los componentes son funciones que reciben el objeto de propiedades *props* y devuelven elementos. Los componentes permiten separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada.

*Tipos de componentes que existen:* Componentes de tipo función y componentes de tipo clase

1. *¿Qué es el Virtual DOM y cuál es su función en ReactJS?*

Es un DOM virtual en el que trabaja *React*. Es una representación del DOM guardada en la memoria del navegador. El virtual DOM es un falso DOM, es una copia que permite hacer cálculos pesados sin afectar el rendimiento del verdadero DOM. Su función es comprobar si algún componente de *React* sufre algún cambio para así estar actualizando dicho componente. La actualización solo se ejecuta en los datos de los componentes “ hijos” nunca en los datos de los componentes “padres”, por eso funciona mucho más rápido que el DOM real.

1. *¿Qué es JSX en ReactJS y porqué es importante?*

JSX es una sintaxis de javascript usada por *React.* Es importante porque permite escribir la lógica de JavaScript dentro de la estructura HTML y los componentes de *React.*

1. *¿Qué es un Hook en ReactJS y cuál es su propósito?*

Un hook es la forma de controlar un estado en los componentes funcionales. Su propósito es permitir desarrollar componentes funcionales con estado y ciclo de vida, por lo tanto, prescindir de un componente de clase.

1. *Mencionar al menos tres tipos de Hooks en ReactJS y explicar brevemente para qué se utilizan:*

*useContext:* Permite acceder al contexto de un componente. Acepta un contexto y devuelve el valor actual del contexto.

***useCallback:*** Permite crear una función que solo se recrea cuando cambian las dependencias especificadas. Acepta dos argumentos: una función y una matriz de dependencias. Esto es útil para evitar recrear funciones innecesariamente en componentes hijos, lo que puede mejorar el rendimiento. ***useRef:*** Este hook permite crear una referencia a un elemento del DOM. Acepta un valor inicial y devuelve un objeto de referencia que puede ser utilizado para acceder al elemento del DOM.

1. *¿Cuáles son las reglas de uso para los Hooks en ReactJS?*
2. Solo se pueden llamar *en el nivel superior*. No se pueden llamar dentro de loops, condiciones o funciones anidadas.
3. Solo se pueden llamar ***desde componentes de función de React***.
4. *Explicar qué es React Router DOM versión 6, para qué se utiliza y cuáles son sus principales componentes y Hooks.*

*Reac*t Router DOM versión 6 es una librería. Se utiliza para agregar *enrutamiento* a una aplicación web de *React* de manera dinámica y sencilla.

*Sus principales componentes son:*

1. BrowserRouter
2. Link y NavLink
3. Navigate
4. Route

*Sus principales hooks son:*

1.useParams

2. useLocation

3. useNavigate

4. useSearchParams

1. *Explicar cómo se realiza la navegación entre diferentes páginas utilizando React Router DOM.*

-Instalar librería *React* router dom

-Inicializar la aplicación con el comando npm run star

-Dentro de la carpeta src, se crea otra carpeta llamada pages

-En la carpeta pages, se crean los *componentes*

-En el archivo index,jsx, se importa el browser router de la librería *React* browser rout

-Para activar las rutas, se envuelven los componentes dentro del componente browser router

1. *¿Cómo se definen rutas en React Router DOM?*

En el componente *routes* se declaran las rutas *React* que queramos usando el componente Route para cada ruta. Casa ruta declarada debe tener dos partes: un *path (aquí se define la ruta o la URL)* y un element (*aquí se define qué componente se va a renderizar*)

1. *Describir cómo crear un proyecto ReactJS con Vite:*

1.instalar node.js

2.con node.js instalado se tiene acceso al comando *npm create vite*@latest especificando que se quiere un proyecto vite en su última versión.

3.se ejecuta el comando *npm create vite*@latest

4.Después de ejecutar el comando, nos pregunta si se quiere instalar la biblioteca

5.se instala la biblioteca

6.se dan los datos del proyecto (nombre, herramienta (react)

7.vit recomienda entrar al proyecto, hacer la instalación de los paquetes y correr el servidor.

8.se ingresa a la carpeta del proyecto

9.estando en la carpeta del proyecto, se abre el proyecto en *vite* para trabajar ahí directamente con el comando *(code .)*

10.se abre una terminal en *vite* y se ejecuta el comando *npm install*

11.para ver lo que se está desarrollando en tiempo real, se usa el servidor web que incluye *vite* que es *npm run dev*

12.El comando anterior ejecuta un servidor. Y así se a creado un proyecto *react* con *vite*

1. *Describir cómo desplegar un JSON server en cualquier Hosting free o servicio en la nube gratuito de su elección*
2. *Describir cómo desplegar una aplicación en cualquier Hosting de su elección.*