TALLER REACT 1

Maria Jose Jimenez Diaz

1. Ejercicio:

2. Investiga la historia de React y menciona dos hitos importantes en su desarrollo.

¿Qué es React?

React es una biblioteca de JavaScript de código abierto, actualmente conocida en todo el mundo del desarrollo web y una de las referencias en el ámbito del front por sus características, pero, ¿Cómo ha llegado React a ser lo que es? veámoslo.

2007-2011: Los Primeros Pasos de React

La historia de React comenzó en Facebook a finales de la década de 2000. En 2007, Jordan Walke, un ingeniero de software en Facebook, comenzó a trabajar en una biblioteca llamada «FaxJS» para abordar los desafíos de la actualización eficiente de la interfaz de usuario en la plataforma. Más tarde, en 2011, Jordan presentó «FaxJS» a un pequeño equipo de ingenieros de Facebook, y esto marcó el comienzo de lo que eventualmente se convertiría en React.

Como dato curioso, mencionar que parte de la inspiración de Walke para la creación de esta librería nace del uso de XHP, una librería de componentes para PHP.

2013: React se Vuelve Open Source

En mayo de 2013, Facebook dio un paso audaz al abrir el código fuente de React y lo hizo de manera gratuita para la comunidad de desarrollo. Esto permitió que otros desarrolladores y empresas comenzaran a utilizar React en sus proyectos y contribuyeran a su desarrollo.

Esto sucedió en la JSConfUS de Mayo de 2013, puedes ver aquí la presentación.

2015: Introducción de React Native

Un gran hito en la historia de React fue la introducción de React Native en 2015. React Native permitió a los desarrolladores crear aplicaciones móviles nativas para iOS y Android utilizando la misma base de código de React. Esto revolucionó el desarrollo móvil al proporcionar una forma más eficiente de crear aplicaciones multiplataforma de alto rendimiento.

2016: Lanzamiento de React 15 y React Fiber

React 15 fue una actualización importante en la que se introdujo una nueva reconciliación llamada «React Fiber.» React Fiber permitió una actualización más

eficiente de la interfaz de usuario y un mejor manejo de la concurrencia, lo que llevó a una experiencia de usuario más suave en aplicaciones web.

2018: React 16 y Hooks

En octubre de 2018, React 16 introdujo una característica revolucionaria llamada «Hooks.» Los Hooks permitieron a los desarrolladores utilizar el estado y otras características de React en componentes funcionales, eliminando la necesidad de clases en gran medida y simplificando el código. Esto cambió la forma en que se escriben los componentes en React y se convirtió en una parte integral de la biblioteca.

2020 y Más Allá: React Continúa Evolucionando

React sigue evolucionando con cada nueva versión, introduciendo mejoras de rendimiento, nuevas características y herramientas para facilitar el desarrollo web moderno. La comunidad de React es activa y próspera, lo que ha llevado a una gran cantidad de bibliotecas y recursos relacionados con React, lo que hace que sea más fácil para los desarrolladores construir aplicaciones web y móviles poderosas y elegantes.

Hitos mas importantes:

- **Lanzamiento Inicial (2013)**: React fue lanzado por Facebook en marzo de 2013. Su enfoque innovador en la creación de interfaces de usuario, basado en un modelo de componentes y el uso de un Virtual DOM para mejorar el rendimiento, marcó un cambio significativo en la forma en que se desarrollan aplicaciones web. Este lanzamiento ayudó a resolver muchos problemas comunes de rendimiento y complejidad en aplicaciones web grandes.
- **React Hooks (2018)**: En febrero de 2018, React introdujo los "React Hooks" en la versión 16.8. Los Hooks permiten a los desarrolladores usar el estado y otras características de React sin necesidad de escribir una clase. Esto simplificó el desarrollo y mejoró la reutilización del código, permitiendo una mayor flexibilidad y una mejor organización en los componentes funcionales.

3. Escribe una breve explicación sobre por qué Facebook decidió crear React.

- **Origen y Problemas Iniciales**:
  - En 2010, los ingenieros de Facebook se encontraron con un problema complejo: la gestión de anuncios en la plataforma. La velocidad de actualización de datos y vistas en una aplicación gigante como Facebook era muy alta.
  - Modificar directamente el **DOM** (Document Object Model) de una página web (como lo hacía jQuery o Javascript puro) resultaba ineficiente y dificultaba el mantenimiento del código.
- **La Solución de Jordan Walke**:
  - Jordan Walke, un programador de Facebook, se propuso resolver estos problemas. Creó un prototipo llamado **FaxJS**, que más tarde evolucionaría en **React**.
  - React fue diseñada para manejar aplicaciones web de gran escala, como Facebook e Instagram. Su objetivo era mejorar la eficiencia y la organización del código.
- **Arquitectura Basada en Componentes**:
  - React propone una arquitectura basada en **componentes**. Estos componentes son piezas de código que combinan HTML, CSS y Javascript.
  - Cada componente contiene tanto la lógica como la presentación, lo que facilita su reutilización en diferentes partes de la aplicación

4. Menciona tres ventajas de usar React en el desarrollo de aplicaciones web.

- **Componentes Reutilizables**: React permite a los desarrolladores crear componentes modulares y reutilizables. Estos componentes encapsulan lógica y presentación, lo que facilita el mantenimiento y la escalabilidad del código. Una vez que un componente está creado, puede ser reutilizado en diferentes partes de la aplicación o en diferentes proyectos, lo que reduce el tiempo de desarrollo y mejora la consistencia.

- **Virtual DOM**: React utiliza un Virtual DOM (Document Object Model) para mejorar el rendimiento de las aplicaciones. En lugar de actualizar directamente el DOM real, React crea una copia virtual del DOM y realiza actualizaciones en esta copia. Luego, compara la versión actual del Virtual DOM con una versión anterior para determinar qué cambios son necesarios y actualiza solo los elementos que han cambiado. Esto reduce el costo de las actualizaciones y mejora la velocidad de la aplicación.
- **Ecosistema y Comunidad Activa**: React cuenta con un ecosistema robusto y una comunidad activa. Hay una gran cantidad de bibliotecas, herramientas y extensiones disponibles que se integran fácilmente con React, como Redux para la gestión del estado y React Router para la navegación. La comunidad de React también ofrece una abundante cantidad de recursos, tutoriales y soporte, lo que facilita la resolución de problemas y la adopción de mejores prácticas.

5. **Explica cómo el Virtual DOM mejora el rendimiento de una aplicación.**

- **Copia Virtual del DOM**: En lugar de manipular directamente el DOM real del navegador, React crea una representación en memoria del DOM llamada Virtual DOM. Esta copia es una estructura de datos ligera que refleja el estado actual de la interfaz de usuario.
- **Actualización Eficiente**: Cuando ocurre un cambio en la aplicación (por ejemplo, una actualización del estado o una interacción del usuario), React primero actualiza el Virtual DOM en lugar del DOM real. Luego, React compara el Virtual DOM actualizado con la versión anterior usando un algoritmo llamado "Reconciliation" o reconciliación. Este algoritmo identifica qué partes del DOM han cambiado.
- **Actualización Selectiva**: Basándose en la comparación entre el Virtual DOM anterior y el actualizado, React calcula la forma más eficiente de aplicar los cambios al DOM real. En lugar de actualizar el DOM completo, React solo realiza las modificaciones necesarias en el DOM real. Esto minimiza la cantidad de operaciones costosas que se realizan en el DOM, que es mucho más lento de manipular que las estructuras en memoria.
- **Minimización de Reflows y Repaints**: Manipular el DOM real implica que el navegador puede necesitar recalcular el diseño de la página (reflows) y volver a dibujar partes de la página (repaints). Estos procesos pueden ser costosos en términos de rendimiento. Al reducir

la cantidad de cambios directos en el DOM real, React ayuda a minimizar estos costos.

6. **Define qué es una Single Page Application (SPA).**

Una Single Page Application (SPA) es un tipo de aplicación web que carga una única página HTML y luego actualiza dinámicamente el contenido dentro de esa página a medida que el usuario interactúa con la aplicación. En lugar de cargar nuevas páginas desde el servidor en respuesta a las acciones del usuario, una SPA utiliza JavaScript para manipular el DOM y actualizar el contenido en la misma página.

Las SPA son populares debido a su capacidad para proporcionar una experiencia de usuario rápida y dinámica, similar a las aplicaciones de escritorio, sin la necesidad de recargar continuamente la página. Sin embargo, también presentan desafíos como la optimización del rendimiento inicial y el manejo del SEO (optimización en motores de búsqueda).

7. **Explica cómo React facilita la creación de una SPA. Proporciona un ejemplo de cómo un componente de React puede actualizar la interfaz sin recargar la página.**

React facilita la creación de una Single Page Application (SPA) mediante su enfoque basado en componentes y su gestión eficiente del estado. Aquí te explico cómo React ayuda en este proceso y te proporciono un ejemplo práctico:

Cómo React Facilita la Creación de una SPA

Componentes Reutilizables: React permite construir la interfaz de usuario usando componentes reutilizables. Estos componentes encapsulan su propio estado y lógica, lo que facilita la creación de una interfaz dinámica sin necesidad de recargar la página completa.

Virtual DOM: React usa un Virtual DOM para realizar actualizaciones eficientes. Cuando cambia el estado de un componente, React actualiza primero el Virtual DOM, compara los cambios con el DOM real y actualiza solo las partes necesarias del DOM real. Esto hace que las actualizaciones sean rápidas y fluidas, lo que es ideal para una SPA.

Enrutamiento del Lado del Cliente: Con bibliotecas como React Router, puedes manejar el enrutamiento en una SPA sin necesidad de hacer solicitudes completas al servidor para cada cambio de página. React Router permite definir rutas y cargar diferentes componentes según la URL, lo que simula una navegación entre páginas en una SPA.

Gestión del Estado: React y sus ecosistemas de herramientas (como Redux o Context API) ofrecen soluciones para gestionar el estado de la aplicación de manera efectiva. Esto facilita la sincronización de datos y la actualización de la interfaz en respuesta a eventos del usuario sin recargar la página.

Ejemplo de Actualización de la Interfaz

```jsx
import React, { useState } from 'react';

function App() {
  // Declaramos una variable de estado llamada 'count' y una función para actualizarla lla
  const [count, setCount] = useState(0);

  // Función para manejar el clic en el botón
  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>Contador: {count}</h1>
      {/* Al hacer clic en el botón, se llama a la función 'increment' */}
      <button onClick={increment}>Incrementar</button>
    </div>
  );
}

export default App;
```

Explicación del Ejemplo

Estado del Componente: Usamos el hook useState para crear una variable de estado count que mantiene el valor del contador. La función setCount se usa para actualizar el estado.

Actualización del Estado: Cuando el usuario hace clic en el botón, se llama a la función increment, que incrementa el valor del contador en uno usando setCount.

Renderizado Dinámico: React actualiza automáticamente el contenido de la interfaz de usuario (el valor del contador) en respuesta a los cambios en el estado. Esto se realiza sin necesidad de recargar la página completa.

10. Explica brevemente el propósito de las carpetas src y public en un proyecto React.

En un proyecto React, las carpetas src y public tienen propósitos distintos y juegan roles específicos en la estructura del proyecto:

1. Carpeta src

Propósito: La carpeta src (abreviatura de "source" o "fuente") contiene todo el código fuente de la aplicación React. Aquí es donde desarrollas y gestionas la lógica y la estructura de la aplicación.

Contenido Típico:

Componentes: Archivos JavaScript o TypeScript que definen los componentes de React.

Estilos: Archivos CSS, SCSS u otros estilos asociados a los componentes.

Servicios: Archivos que gestionan la lógica de negocio, como las llamadas a APIs.

Utilidades: Funciones y módulos auxiliares que pueden ser utilizados en diferentes partes de la aplicación.

Archivos de configuración: Archivos como index.js o App.js que configuran el punto de entrada de la aplicación y el enrutamiento.

11. Explica cómo JSX se diferencia del HTML tradicional.

JSX (JavaScript XML) es una sintaxis extendida para JavaScript que se utiliza en React para describir cómo debería lucir la interfaz de usuario. Aunque JSX se parece mucho a HTML, hay varias diferencias clave entre ellos:

Diferencias Clave entre JSX y HTML

Sintaxis de Atributos:

HTML: Los atributos se escriben en minúsculas y usan guiones para separar palabras. Por ejemplo, class y data-attribute.

JSX: Los atributos se escriben en camelCase y usan nombres diferentes para algunos atributos. Por ejemplo, en lugar de class, se usa className debido a que class es una palabra reservada en JavaScript. Los atributos como data-attribute se escriben como dataAttribute.

Ejemplo:

HTML:

JSX:

Expresiones JavaScript:

HTML: No permite incrustar directamente expresiones JavaScript. El HTML se utiliza para definir la estructura estática de la página.

JSX: Permite incrustar expresiones JavaScript dentro de llaves {}. Esto facilita la creación de contenido dinámico.

Ejemplo:

HTML: Hello, world!

JSX: Hello, {name}! , donde name es una variable de JavaScript.

Elementos Auto-Cerrables:

HTML: Algunos elementos, como <img>, <input>, y
, se pueden escribir como elementos auto-cerrables sin una etiqueta de cierre.

JSX: Todos los elementos deben cerrarse, incluso los auto-cerrables. Esto significa que se utiliza una barra inclinada / al final de la etiqueta.

Ejemplo:

HTML: <img src="image.jpg">

JSX:

JavaScript Dentro del JSX:

HTML: No se puede incluir directamente código JavaScript dentro de las etiquetas HTML.

JSX: Permite incluir código JavaScript dentro de las llaves {}. Esto es útil para calcular valores o llamar funciones.

Ejemplo:

HTML: No aplicable

JSX: <h1>{user.isLoggedIn ? 'Welcome back!' : 'Please sign in'}</h1>

Compilación:

HTML: Es un lenguaje que los navegadores entienden directamente.

JSX: No es entendido por los navegadores directamente y necesita ser compilado a JavaScript puro utilizando herramientas como Babel. Esta compilación convierte JSX en llamadas a React.createElement().

Ejemplo de Compilación:

JSX: <button onClick={handleClick}>Click me</button>

JavaScript compilado: React.createElement('button', { onClick: handleClick }, 'Click me')

12. Define los roles principales en un equipo SCRUM.

1. Product Owner (PO)

Responsabilidades Principales:

Visión del Producto: Define y comunica la visión del producto. Es responsable de asegurar que el producto entregado esté alineado con las necesidades y expectativas de los stakeholders.

Gestión del Product Backlog: Crea, prioriza y mantiene el Product Backlog, que es una lista priorizada de requisitos y características del producto.

Tomar Decisiones: Toma decisiones sobre las funcionalidades y el alcance del producto, y responde a preguntas y dudas del equipo sobre las prioridades y los requisitos.

Stakeholder Engagement: Actúa como el principal punto de contacto entre el equipo de desarrollo y los stakeholders, asegurando que las necesidades del cliente estén representadas.

2. Scrum Master

Responsabilidades Principales:

Facilitador: Facilita las ceremonias de SCRUM, como las reuniones diarias (Daily Stand-ups), las planificaciones de Sprint, las revisiones y las retrospectivas.

Eliminar Obstáculos: Identifica y elimina impedimentos que puedan estar bloqueando el progreso del equipo, ayudando a que el equipo trabaje de manera eficiente.

Coaching: Capacita al equipo y a la organización en la aplicación de SCRUM y en las prácticas ágiles, ayudando a mejorar los procesos y la colaboración.

Protector del Equipo: Protege al equipo de interrupciones externas y distracciones para que pueda concentrarse en el trabajo del Sprint.

3. Development Team (Equipo de Desarrollo)

Responsabilidades Principales:

Entregar Incrementos de Producto: Trabaja para entregar incrementos de producto potencialmente entregables al final de cada Sprint, asegurando que se cumplan los criterios de aceptación y calidad.

Autonomía y Autoorganización: Se autoorganiza y decide cómo abordar el trabajo en el Sprint. No está dirigido directamente por el Product Owner o el Scrum Master en el trabajo diario.

Colaboración: Colabora estrechamente con el Product Owner para comprender los requisitos y con el Scrum Master para superar impedimentos.

Responsabilidad Compartida: Todos los miembros del equipo comparten la responsabilidad de cumplir con el objetivo del Sprint y de entregar valor.

13. Explica qué es un sprint y cómo se planifica.

Un Sprint es un período de tiempo fijo en el que se realiza un trabajo específico para desarrollar una parte del producto en el marco de trabajo SCRUM. Cada Sprint tiene una duración predeterminada, que suele ser de 1 a 4 semanas. El objetivo del Sprint es entregar un incremento de producto potencialmente entregable, es decir, una versión del producto que pueda ser revisada y potencialmente puesta en producción.

Cómo se Planifica un Sprint

La planificación de un Sprint se realiza durante una ceremonia específica llamada **Sprint Planning** (Planificación del Sprint), que ocurre al comienzo de cada Sprint. La planificación se divide en dos partes principales:

1. **Definición del Objetivo del Sprint**:
    - **Product Owner**: Presenta el objetivo general del Sprint y la lista priorizada de elementos del Product Backlog que se desea abordar en el Sprint.

- **Equipo de Desarrollo**: Discute los elementos del Product Backlog propuestos y ayuda a definir un objetivo claro para el Sprint. El objetivo debe ser específico, alcanzable y alineado con la visión del producto.

2. **Planificación del Trabajo**:
    - **Descomposición de Tareas**: El Equipo de Desarrollo descompone los elementos seleccionados del Product Backlog en tareas más pequeñas y manejables. Esto ayuda a estimar el trabajo requerido y a entender cómo se abordará cada tarea.
    - **Asignación de Tareas**: Aunque el equipo es autoorganizado y decide cómo distribuir el trabajo, en esta fase se discuten las tareas y los miembros del equipo pueden asumir responsabilidades específicas para completar esas tareas.
    - **Estimación del Trabajo**: El equipo estima el esfuerzo necesario para completar cada tarea utilizando técnicas como estimaciones en puntos de historia, horas o días.

Pasos en la Planificación del Sprint

1. **Preparación**:
    - Antes de la reunión de planificación, el Product Owner debe tener el Product Backlog actualizado y priorizado.
    - El equipo debe estar preparado para discutir los requisitos y las prioridades.
2. **Sprint Planning Meeting**:
    - **Parte 1**: **¿Qué se va a hacer?** Se revisan los elementos del Product Backlog y se seleccionan los que se pueden completar durante el Sprint. Se define el objetivo del Sprint.
    - **Parte 2**: **¿Cómo se va a hacer?** El equipo descompone los elementos seleccionados en tareas específicas y detalla cómo se implementarán. Se planifica el trabajo diario y se asignan tareas si es necesario.
3. **Compromiso**:
    - El equipo se compromete a alcanzar el objetivo del Sprint y a completar las tareas planificadas. Se establece el alcance del trabajo basado en la capacidad del equipo y el tiempo disponible en el Sprint.

Resultados de la Planificación del Sprint

- **Sprint Backlog**: Un conjunto de elementos del Product Backlog seleccionados para el Sprint, junto con un plan detallado de cómo se llevará a cabo el trabajo.
- **Objetivo del Sprint**: Una declaración clara y concisa sobre lo que el equipo pretende lograr durante el Sprint.

REFERENCIAS:

https://tecnitium.com/cronologia-de-react-de-biblioteca-a-referencia/#:~:text=En%20resumen%2C%20React%20ha%20recorrido%20un%20largo%20camino,cambiantes%20de%20la%20industria%20del%20desarrollo%20de%20software.