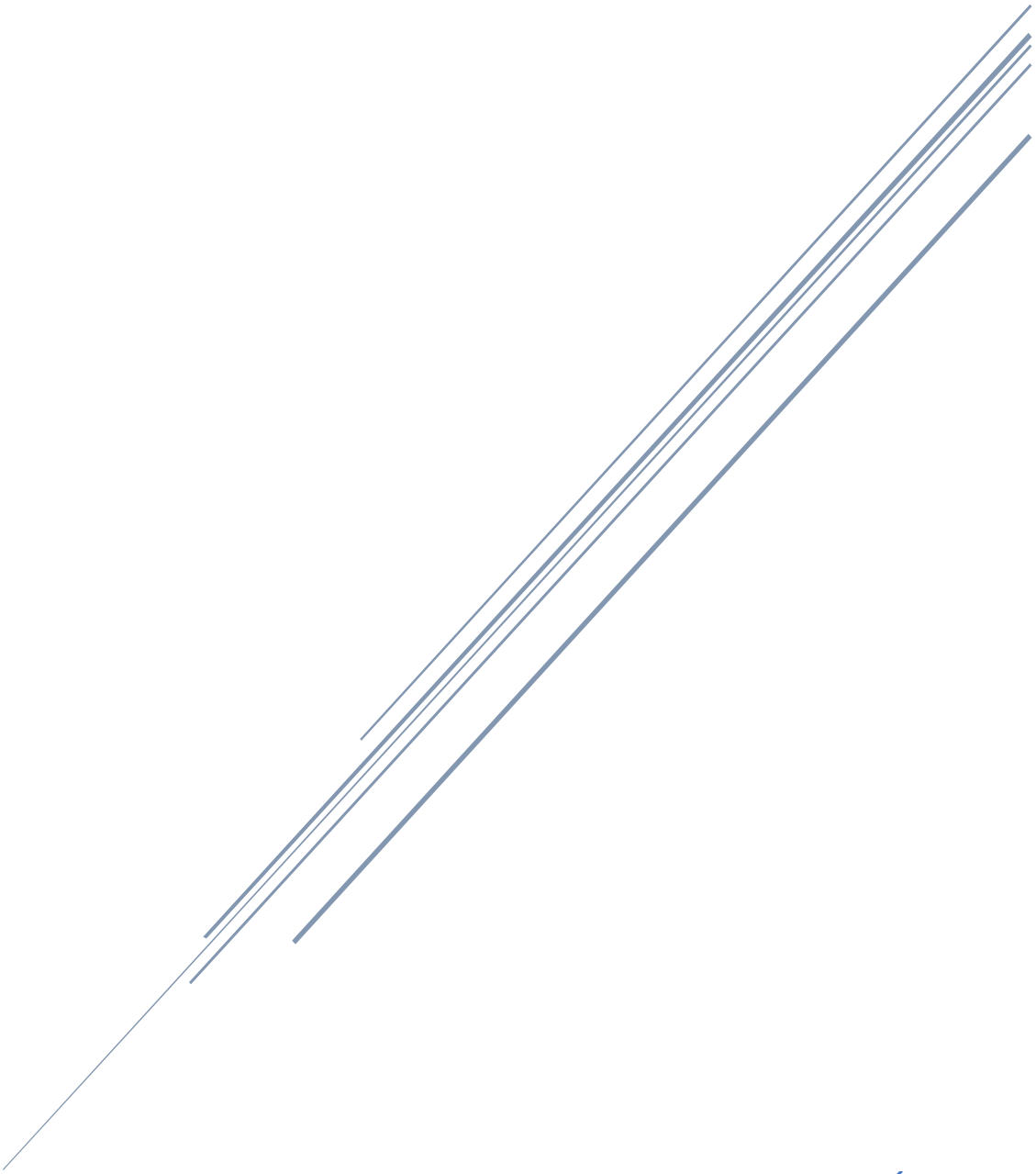
REACT

Ejercicios de Clase 24/02/2023



I.E.S. MAR DE CÁDIZ

2ª DESARROLLO DE APLICACIONES WEB

Índice

1. ¿Qué es React JS?

React JS es una biblioteca de JavaScript de código abierto para construir interfaces de usuario (UI) interactivas y reutilizables. Fue creado por Facebook y es ampliamente utilizado en el desarrollo web moderno. React JS utiliza una arquitectura de componentes para dividir la interfaz de usuario en piezas más pequeñas y manejables, lo que facilita el mantenimiento y la reutilización del código. Además, utiliza el concepto de "virtual DOM" para optimizar el rendimiento y mejorar la experiencia del usuario en aplicaciones de página única (SPA).

Fuentes:

https://www.nextu.com/blog/que-es-react-js-como-funciona-rc22/

https://es.wikipedia.org/wiki/React

https://reactjs.org/

https://kinsta.com/es/base-de-conocimiento/que-es-react-js/

2. ¿Por qué deberíamos utilizar ReactJS?

ReactJS es una librería de código abierto escrita en JavaScript que ofrece muchas ventajas. Algunas de ellas son:

- Facilita el proceso general de escritura de componentes.
- Aumenta la productividad y facilita un mayor rendimiento.
- Desarrollo rentable y entrega más rápida de proyectos.
- Amplia comunidad y soporte.
- Compuesto por componentes, lo que permite reutilización de código.
- Isomórfico, lo que significa que se puede ejecutar tanto en el servidor como en el cliente.

React es adecuada para muchos tipos de proyectos distintos y nos permite un desarrollo ágil, ordenado y con una arquitectura clara.

Fuentes:

https://sistemasgeniales.com/software/las-10-ventajas-principales-de-usar-react-js/

https://www.doonamis.es/en/para-que-sirve-react-js-beneficios-para-tus-apps/

https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html

https://es.quora.com/Cu%C3%A1ndo-y-por-qu%C3%A9-utilizar-React-JS

3. Explicar "DOM real" y "DOM virtual"

El DOM real es la estructura de objetos que representa el contenido de una página web y se puede acceder mediante JavaScript. El DOM virtual es una representación en memoria del DOM real, que se mantiene sincronizada con él mediante bibliotecas como ReactDOM. La ventaja del DOM virtual es que permite hacer comparaciones y cambios más eficientes que al DOM real. En React, el proceso de comparar el DOM virtual con el DOM real para actualizar la interfaz gráfica se llama reconciliación. Aunque ambos son conceptos diferentes, algunos autores los confunden.

Fuentes:

https://es.reactjs.org/docs/faq-internals.html

https://es.quora.com/Cu%C3%A1l-es-la-diferencia-entre-el-DOM-normal-y-un-DOM-virtual

https://www.codecademy.com/article/react-virtual-dom

https://www.geeksforgeeks.org/difference-between-virtual-dom-and-real-dom/

4. ¿Qué es JSX?

JSX es una extensión de sintaxis de JavaScript que permite mezclar JS y HTML/XML para definir la interfaz de usuario en React. JSX es una abstracción sobre la API React.createElement que permite expresar de forma aún más declarativa la definición de la UI que quieres renderizar.

Fuentes:

https://es.reactjs.org/docs/introducing-jsx.html

https://medium.com/@simonhoyos/qu%C3%A9-es-jsx-95006a2f94f9

https://carlosazaustre.es/jsx-para-novatos

5. ¿Qué son los "componentes"?

Los componentes son bloques de construcción fundamentales en ReactJS que permiten la creación de interfaces de usuario reutilizables y modulares. Cada componente encapsula una parte de la funcionalidad y la representación visual de la interfaz de usuario, lo que mejora la legibilidad del código y la eficiencia de la aplicación.

Fuentes:

https://keepcoding.io/blog/componentes-en-react/

https://www.digitalocean.com/community/tutorials/how-to-create-custom-components-in-react-es

https://platzi.com/blog/react-patrones-render/

6. ¿Cuáles son las etapas de vida de un componente?

En ReactJS, cada componente pasa por varias etapas de vida a medida que se crea, actualiza y se destruye. Las etapas de vida de un componente son las siguientes:

- Montaje (Mounting): En esta etapa, el componente se crea e inicializa. Se ejecutan los métodos constructor, getDerivedStateFromProps, render y componentDidMount.
- Actualización (Updating): En esta etapa, el componente se actualiza en respuesta a cambios en las propiedades (props) o estados (state). Se ejecutan los métodos getDerivedStateFromProps, shouldComponentUpdate, render, getSnapshotBeforeUpdate y componentDidUpdate.
- Desmontaje (Unmounting): En esta etapa, el componente se destruye y se elimina del DOM. Se ejecuta el método componentWillUnmount.
- Manejo de errores (Error Handling): En esta etapa, se manejan los errores que se producen durante el montaje, la actualización o el procesamiento de eventos. Se ejecutan los métodos getDerivedStateFromError y componentDidCatch.

Además de estas etapas principales, hay dos etapas adicionales que se aplican a los componentes que utilizan contextos en ReactJS:

- Lectura del contexto (Context Reading): En esta etapa, el componente lee datos de un contexto utilizando el método static contextType o el hook useContext.
- Actualización del contexto (Context Updating): En esta etapa, el componente se actualiza en respuesta a cambios en el contexto. Se ejecutan los métodos static getDerivedStateFromContext y shouldComponentUpdate.

Fuentes:

https://reactiveprogramming.io/blog/es/react/ciclo-de-vida-de-los-componentes

https://es.reactjs.org/docs/state-and-lifecycle.html

7. ¿Existe alguna diferencia entre un "componente" y un "elemento"?

Sí, existe una diferencia entre un componente y un elemento en React. Un elemento es un objeto plano que describe lo que se quiere mostrar en la pantalla, mientras que un componente es una función o clase que permite manipular las propiedades del objeto props y devolver un elemento React. En otras palabras, los componentes producen conjuntos de elementos basados en la lógica descrita en funciones o clases

Fuentes:

https://tutz.tv/react/element-vs-component

https://keepcoding.io/blog/diferencia-entre-elemento-y-componente-en-react/

8. ¿Los exploradores web pueden leer JSX?

No, los exploradores web no pueden leer JSX. Los exploradores están configurados para leer objetos JavaScript, pero un objeto JSX no es lo mismo que un objeto JavaScript. Para poder utilizar JSX en una aplicación web, se necesita utilizar herramientas como webpack y Babel para traducirlo a JavaScript legible por el navegador

Fuentes:

https://es.bitdegree.org/tutoriales/preguntas-entrevista-react-js/#heading-9

https://recluit.com/que-esperar-en-una-entrevista-de-react/

9. ¿Cuál es la diferencia entre ReactJS y React Native?

ReactJS es una biblioteca de JavaScript utilizada para crear interfaces de usuario en aplicaciones de software web, mientras que React Native es un marco nativo que le permite crear aplicaciones nativas reales que funcionan perfectamente en Android o iOS. React Native usa la misma API que React pero está diseñado para crear aplicaciones móviles nativas

Fuentes:

https://rootstack.com/en/blog/react-vs-react-native%20diferences

https://www.ranktracker.com/es/blog/reactjs-vs-react-native-which-is-the-best-option-to-consider-for-mobile-apps/

10. ¿Qué es "flux"?

Flux es un patrón de arquitectura de aplicaciones que se basa en un flujo unidireccional de datos. Los datos fluyen en una sola dirección, desde las vistas a los stores. Flux no es específico de React y se puede usar con cualquier librería de vistas.

Fuentes:

https://www.reactjs.wiki/que-es-flux

https://es.survivejs.com/react/implementing-kanban/react-and-flux/