TECNOLÓGICO NACIONAL DE MÉXICO®

React: introducción.

Instituto Tecnológico de Tepic
Asignatura: Interfaces web.
Docente: Nava Hernandez Irving Yair

Integrantes del equipo:
Martinez Velazquez Pedro
Perez Renteria Carlos Alberto
Santos Molla Guillermo Hilario
Frias Bogarin Rodolfo

Introducción.

En la actualidad, el desarrollo de interfaces web dinámicas y responsivas se ha convertido en una necesidad clave en la industria del software. Los usuarios esperan aplicaciones rápidas, intuitivas y visualmente atractivas, lo que ha llevado a los desarrolladores a buscar herramientas que faciliten la creación de interfaces ricas sin sacrificar rendimiento ni escalabilidad.

Una de las tecnologías más utilizadas para este propósito es **React**, una biblioteca de JavaScript desarrollada por Meta (antes Facebook), que ha revolucionado la forma en que se construyen aplicaciones web. Su enfoque basado en componentes, su rendimiento optimizado mediante el uso de un DOM virtual, y su ecosistema en constante crecimiento la han posicionado como una de las herramientas más importantes en el desarrollo frontend moderno.

Desarrollo:

¿Qué es React?

React es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario, especialmente en aplicaciones web de una sola página (Single Page Applications - SPA), donde se requiere una experiencia de usuario dinámica e interactiva. Fue desarrollada por Jordan Walke, un ingeniero de software de Facebook, y lanzada públicamente en 2013. Su creación surgió como una solución a los desafíos de mantenimiento y rendimiento que enfrentaban en la construcción de aplicaciones como Facebook y posteriormente Instagram.

A diferencia de otros frameworks como Angular o Vue, React no es un framework completo, sino una librería enfocada únicamente en la capa de vista (View) del patrón arquitectónico MVC (Modelo - Vista - Controlador). Esto le permite integrarse fácilmente con otras bibliotecas o frameworks y otorga flexibilidad al desarrollador para organizar el resto de la arquitectura como desee.

Características de React

React destaca por una serie de características que lo convierten en una herramienta poderosa y eficiente para el desarrollo de interfaces modernas. A continuación, se detallan sus principales características:

1. Basado en componentes

React está diseñado bajo una arquitectura de componentes reutilizables, donde cada componente representa una parte independiente de la interfaz de usuario, como un botón, una barra de navegación o una tarjeta de producto. Estos componentes pueden ser anidados, combinados y reutilizados en diferentes partes de la aplicación, promoviendo un desarrollo más ordenado, mantenible y escalable.

2. DOM virtual (Virtual DOM)

Una de las innovaciones más importantes de React es el uso del DOM virtual, una representación ligera del DOM real que se mantiene en memoria. Cuando se producen cambios en los datos o en el estado de un componente, React actualiza primero el Virtual DOM, lo compara con su versión anterior y, finalmente, actualiza solo las partes modificadas en el DOM real. Esto mejora considerablemente el rendimiento, reduciendo la cantidad de manipulaciones directas del DOM, que suelen ser costosas en términos de recursos.

3. Unidireccionalidad de datos (One-Way Data Binding)

React utiliza un flujo de datos en una sola dirección. Los datos fluyen de los componentes padres hacia los hijos a través de props (propiedades), lo que permite un mejor control y previsibilidad del comportamiento de la interfaz. Este enfoque facilita la depuración y evita efectos colaterales indeseados al manipular datos.

4. JSX (JavaScript XML)

JSX es una extensión de la sintaxis de JavaScript que permite escribir estructuras HTML directamente en el código JavaScript. Aunque es opcional, JSX es ampliamente usado en React por hacer el código más legible y mantener juntos la lógica de comportamiento y la estructura de la interfaz. Por ejemplo:

```
const Saludo = <h1>Hola, mundo</h1>;
```

5. React Hooks

A partir de React 16.8, se introdujeron los Hooks, una característica que permite a los componentes funcionales utilizar estado (useState) y otras funcionalidades como efectos (useEffect) sin necesidad de escribir componentes de clase. Esto simplifica el código y permite una mejor organización de la lógica.

6. Actualizaciones eficientes

Gracias al Virtual DOM y al algoritmo de comparación llamado reconciliación, React realiza actualizaciones eficientes del DOM. Esto asegura un rendimiento óptimo incluso en aplicaciones con muchas interacciones o con interfaces dinámicas complejas.

7. Gran ecosistema y comunidad

React cuenta con una comunidad activa y un ecosistema rico en herramientas, bibliotecas y extensiones. Entre las más populares están:

- **React Router**: Para el manejo de rutas.
- **Redux / Context API**: Para la gestión de estado global.
- **Next.js**: Framework que extiende a React para aplicaciones más completas con renderizado del lado del servidor (SSR).

8. Compatibilidad con desarrollo móvil (React Native)

React también puede utilizarse para desarrollar aplicaciones móviles nativas mediante React Native, permitiendo reutilizar parte del código y lógica de negocio entre aplicaciones web y móviles.

¿Qué son los componentes en React?

Un componente en React.js, es básicamente una pieza de código encapsulada que combina tanto la lógica de negocio como la interfaz de usuario en una sola unidad reutilizable. La esencia fundamental detrás de los componentes es descomponer una aplicación en partes más pequeñas y manejables, simplificando así la construcción y el mantenimiento del código.

En React, hay dos tipos principales de componentes: los componentes funcionales y los componentes de clase. En este artículo, nos centraremos principalmente en los componentes funcionales. La razón de esto es que los componentes funcionales son actualmente la forma más común y popular de escribir componentes en React.

Componentes Funcionales

Un componente funcional en React es, en términos sencillos, es un bloque de construcción básico creado mediante una función de JavaScript. Esta función tiene la tarea de proporcionar elementos JSX, que son las instrucciones para que React construya la interfaz de usuario. Este tipo de componente es fácil de entender y es una forma común de definir partes reutilizables en React.

La importancia de los componentes funcionales ha crecido aún más con la introducción de los Hooks en React 16.8. Antes de los Hooks, los componentes funcionales tenían limitaciones en comparación con los componentes de clase, pero ahora, gracias a los Hooks, pueden manejar estados, efectos secundarios y otras características de manera eficiente.

Para crear un componente funcional en React, solo necesitas definir una función que devuelva los elementos JSX que deseas mostrar en tu aplicación.

```
import React from "react";
export function FunctionComponent() {
  return <p> Hello function component! </p>;
}
```

Componentes de clase (Class Components)

Son componentes definidos como clases que extienden de React.Component. Pueden manejar estado y métodos de ciclo de vida (como componentDidMount, componentDidUpdate, etc.)

ejemplo:

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

class Saludo extends Component {

  render() {

    return <h1>Hola, {this.props.nombre}</h1>;

  }

}
```

Estructura típica de un componente funcional moderno

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



function MiComponente() {

  const [dato, setDato] = useState('valor inicial');



  useEffect(() => {

    console.log('Componente montado o actualizado');

  }, [dato]);



  return (

    <div>

      <h2>{dato}</h2>

      <button onClick={() => setDato('Nuevo valor')}>Cambiar dato</button>

    </div>

  );

}
```

¿Cómo funciona React?

React funciona creando una representación virtual del DOM (Virtual DOM). Cuando se produce un cambio en el estado de un componente, React calcula las diferencias entre el DOM actual y el nuevo estado deseado, y actualiza únicamente los elementos modificados. Esto mejora el rendimiento, especialmente en aplicaciones grandes y complejas.

El flujo básico de React es:

1. El usuario interactúa con la UI.
2. Se actualiza el estado del componente.
3. React vuelve a renderizar el componente afectado.
4. Se actualiza el DOM real mediante el Virtual DOM.

¿Por qué usar React?

1. Alta eficiencia y rendimiento

Gracias a su DOM virtual, React realiza actualizaciones del DOM de manera eficiente, renderizando solo los componentes que realmente han cambiado. Este enfoque minimiza las operaciones costosas sobre el DOM real, lo que se traduce en mayor velocidad y mejor rendimiento, especialmente en aplicaciones interactivas o con muchos datos dinámicos.

2. Componentes reutilizables y modulares

La arquitectura basada en componentes permite crear elementos de interfaz reutilizables en distintas partes de la aplicación o incluso en otros proyectos. Esta modularidad reduce la duplicación de código, facilita el mantenimiento y acelera el desarrollo.

3. Curva de aprendizaje razonable

Aunque React introduce conceptos nuevos como JSX o Hooks, su enfoque basado en JavaScript puro y su simplicidad comparado con frameworks más completos como Angular lo hacen accesible para desarrolladores con conocimientos básicos de HTML, CSS y JavaScript.

4. Gran ecosistema y comunidad activa

React tiene una comunidad masiva y activa. Esto significa:

- Hay una enorme cantidad de recursos, tutoriales, cursos y documentación.
- Puedes encontrar soluciones a problemas comunes fácilmente en foros como Stack Overflow o GitHub.
- Hay una gran variedad de librerías complementarias como React Router, Redux, Formik, etc.

Además, al ser mantenido por Meta (Facebook), se actualiza con frecuencia y tiene un respaldo corporativo sólido.

5. Compatibilidad con aplicaciones móviles (React Native)

React permite extender la lógica de desarrollo al entorno móvil gracias a React Native, una tecnología basada en React que permite construir aplicaciones móviles nativas (iOS y Android) utilizando JavaScript. Esto ofrece la posibilidad de compartir parte del código entre la app web y la app móvil, reduciendo tiempos y costos de desarrollo.

6. Facilidad de integración

React puede integrarse fácilmente en proyectos ya existentes. No es necesario reescribir toda una aplicación para comenzar a usar React: se puede aplicar por partes, lo que facilita la migración desde otras tecnologías de forma progresiva.

7. Herramientas de desarrollo avanzadas

React cuenta con herramientas muy útiles como las React Developer Tools, extensiones de navegador que permiten inspeccionar los componentes, ver el estado, las props y el árbol de renderizado. Esto ayuda a depurar y optimizar aplicaciones de forma más sencilla.

8. Orientado al futuro

React adopta prácticas modernas del desarrollo web, como el uso de componentes declarativos, el manejo eficiente del estado, y una estructura pensada para escalar. Su

enfoque hace que las aplicaciones sean más predecibles y más fáciles de testear, y se alinea bien con los estándares de desarrollo profesional actuales.

9. Gran demanda laboral

Dado que muchas empresas grandes y startups utilizan React (Facebook, Instagram, Netflix, Airbnb, Uber, entre otras), el conocimiento en React aumenta las oportunidades de empleo y es una habilidad altamente valorada en el mercado.

Conclusión:

React se ha consolidado como una de las herramientas más potentes y populares para el desarrollo de interfaces de usuario modernas, gracias a su enfoque declarativo, su arquitectura basada en componentes y su excelente rendimiento mediante el DOM virtual. Estas características, sumadas a su comunidad activa, su amplio ecosistema de librerías y su compatibilidad con el desarrollo móvil mediante React Native, lo convierten en una opción atractiva tanto para desarrolladores principiantes como para profesionales.

Adoptar React permite construir aplicaciones más organizadas, reutilizables y fáciles de mantener, lo cual es esencial en entornos de desarrollo ágil y proyectos de gran escala. Además, su curva de aprendizaje accesible y su integración progresiva facilitan la transición desde otros enfoques tradicionales del desarrollo web.

En resumen, React no solo representa una herramienta para crear interfaces eficientes, sino también una filosofía moderna de desarrollo que promueve la modularidad, la productividad y la escalabilidad. Aprender y dominar React es, sin duda, una inversión valiosa para cualquier desarrollador web en la actualidad.

Referencias Bibliográficas.

React. (s.f.). *Built-in React Hooks*. Recuperado el 4 de mayo de 2025, de https://react.dev/reference/react/hooksFreeCodeCamp+3React+3arXiv+3

React. (s.f.). *Virtual DOM and Internals*. Recuperado el 4 de mayo de 2025, de https://legacy.reactjs.org/docs/faq-internals.htmlReact

React. (s.f.). *Presentando Hooks*. Recuperado el 4 de mayo de 2025, de https://es.legacy.reactjs.org/docs/hooks-intro.htmlGeeksforGeeks+5es.legacy.reactjs.org+5React+5

Ikechukwu, V. (2020, 25 de febrero). *Learn React Hooks – A Beginner's Guide*. freeCodeCamp. Recuperado de https://www.freecodecamp.org/news/the-beginners-guide-to-react-hooksFreeCodeCamp

Abba, I. (2020, 10 de febrero). *React Hooks Tutorial – useState, useEffect, and How to Create Custom Hooks*. freeCodeCamp. Recuperado de https://www.freecodecamp.org/news/introduction-to-react-hooksFreeCodeCamp+1FreeCodeCamp+1

GeeksforGeeks. (2024, 12 de diciembre). *ReactJS Virtual DOM*. Recuperado de https://www.geeksforgeeks.org/reactjs-virtual-domGeeksforGeeks+1GeeksforGeeks+1

GeeksforGeeks. (2024, 28 de abril). *Introduction to React Hooks*. Recuperado de https://www.geeksforgeeks.org/introduction-to-react-hooksGeeksforGeeks

Nyamador, D. (2020, 23 de octubre). *React's Virtual DOM Explained*. Pluralsight. Recuperado de https://www.pluralsight.com/resources/blog/guides/reacts-virtual-dom-explainedPluralsight

Mascetti, S., Ducci, M., Cantù, N., Pecis, P., & Ahmetovic, D. (2020). *Developing Accessible Mobile Applications with Cross-Platform Development Frameworks*. arXiv. https://arxiv.org/abs/2005.06875