

INTRODUCCIÓN AL DESARROLLO FRONTEND CON REACT 2023



SECRETARÍA DE
EXTENSIÓN
UNIVERSITARIA
UTN - FRC



*UTN
Facultad Regional Córdoba

Agencia
CÓRDOBA
JOVEN

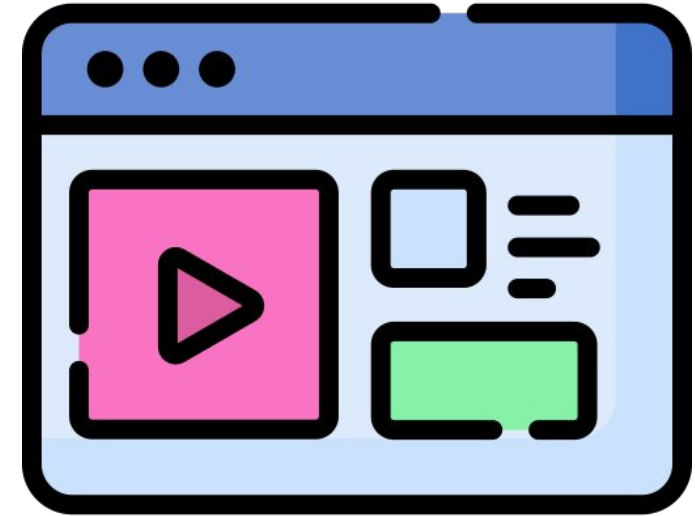


CÓRDOBA
entre todos

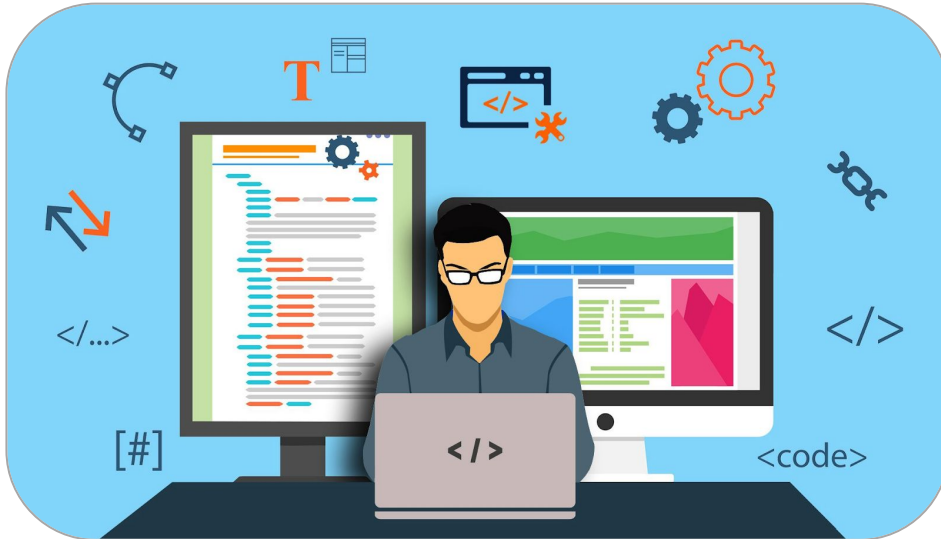
Clase 04 - Contenido

Fundamentos de React

- Introducción a React y su importancia en el desarrollo web.
- Conceptos básicos de React: componentes, props y estado.
- Instalación de React y creación de un proyecto React básico.

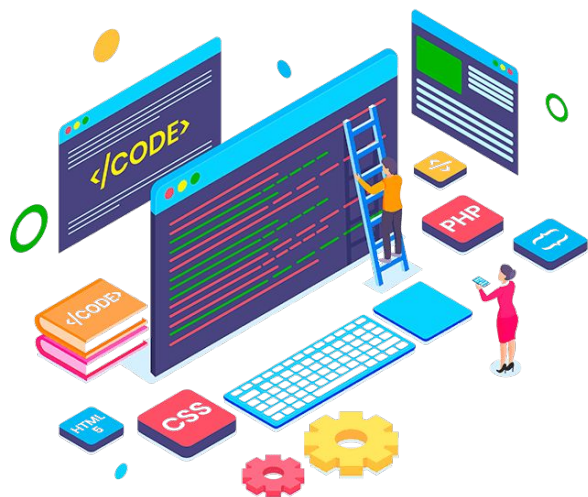


Objetivos de la Clase



- **Distinguir entre Frameworks:**
 - Ser capaz de diferenciar React de otros frameworks de frontend como Angular y Vue.
- **Comprender el Rol de React:**
 - Entender cómo React se posiciona en el ecosistema del desarrollo frontend.
- **Asimilar Conceptos Clave:**
 - Familiarizarse con los pilares de React: componentes, props y estado.
- **Manejar el Entorno de React:**
 - Aprender a instalar las herramientas necesarias y crear un proyecto React básico.
- **Identificar Herramientas y Librerías Auxiliares:**
 - Conocer bibliotecas y herramientas adicionales comunes en proyectos de React.

Conceptos Importantes para Frontend



- **Responsive Design:** Diseño web que se adapta a diferentes tamaños de pantalla para ofrecer una experiencia de usuario óptima.
- **Single-Page Application (SPA):** Aplicaciones web que cargan una sola página HTML y actualizan el contenido dinámicamente mediante JavaScript.
- **DOM (Document Object Model):** Representación estructurada de un documento HTML o XML, que permite la manipulación de elementos y atributos.
- **CSS Preprocessors:** Herramientas como SASS y LESS que extienden las capacidades de CSS para incluir variables, anidamiento y otras características.
- **Accessibility:** Diseño y desarrollo web que permite el acceso a la información y funcionalidad a todos los usuarios, incluidos aquellos con discapacidades.
- **AJAX (Asynchronous JavaScript and XML):** Técnica que permite la actualización dinámica de partes de una página web sin tener que recargar toda la página.
- **Framework/Librería:** Conjunto de herramientas y reglas que facilitan el desarrollo, como React, Angular o Vue.js.

¿Qué es un Framework de Frontend?

Definición:

- Un Framework de Frontend es una estructura pre-diseñada que ayuda a los desarrolladores a crear interfaces de usuario de manera eficiente y estandarizada.
- Los frameworks representan la arquitectura de software (el marco) de una aplicación y determinan fundamentalmente el proceso de desarrollo.



Características:

- Proporciona una arquitectura modular.
- Incluye herramientas y librerías pre-configuradas.
- Facilita la colaboración y mantenimiento del código.

Importancia:

- Acelera el proceso de desarrollo.
- Establece buenas prácticas y patrones de diseño.
- Permite la creación de aplicaciones más robustas y escalables.

Frameworks y Librerías Populares en Frontend

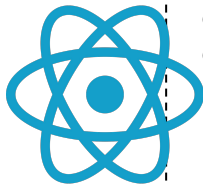
Angular

- **Desarrollado por:** Google
- **Arquitectura:** Modelo-Vista-Controlador (MVC)
- **Características Principales:**
 - Inyección de dependencias para mayor modularidad.
 - Angular CLI para una inicialización de proyecto más rápida.
 - Herramientas robustas para la manipulación de formularios y validación.
- **Casos de Uso Comunes:**
 - Aplicaciones empresariales
 - SPAs (Aplicaciones de Página Única)

Vue.js


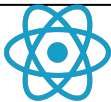

- **Desarrollado por:** Evan You y la comunidad
- **Arquitectura:** Orientado a componentes
- **Características Principales:**
 - Directivas para enriquecer el HTML.
 - Compatibilidad gradual para integración en proyectos existentes.
 - Vuex para manejo avanzado del estado.
- **Casos de Uso Comunes:**
 - Aplicaciones interactivas
 - Prototipos rápidos
 - Pequeños a medianos proyectos web

React

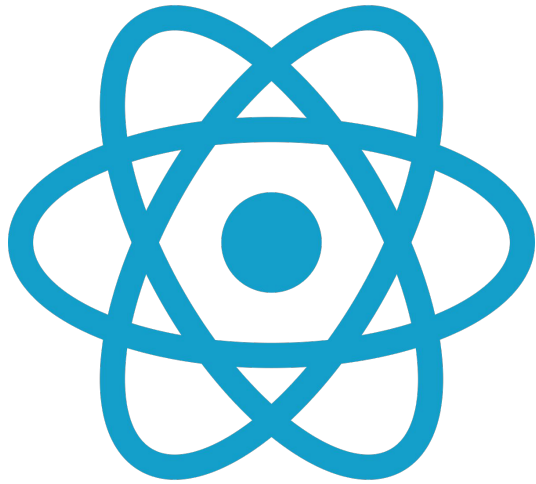


- **Desarrollado por:** Facebook
- **Arquitectura:** Orientado a componentes
- **Características Principales:**
 - Virtual DOM para actualizaciones de UI más eficientes.
 - Flexibilidad para integrar con diversas bibliotecas de terceros.
 - Hooks para manejo de estado y efectos secundarios.
- **Casos de Uso Comunes:**
 - Aplicaciones interactivas
 - Desarrollo rápido y ágil

Frameworks y Librerías Populares Comparativa

Características clave	 Angular	 React	 Vue.js
Facilidad de aprendizaje	Difícil	Moderado	Fácil
Tamaño de la comunidad	Grande	Muy grande	Creciente
Flexibilidad	Menor	Alta	Media
Rendimiento	Alto	Alto	Alto
Reusabilidad de Componentes	Sí	Sí	Sí
Soporte empresarial (proyectos a gran escala)	Amplio	Medio	Limitado
Actualizaciones y mantenimiento	Rápidas pero posiblemente disruptivas	Estables y menos disruptivas	Estables y menos disruptivas

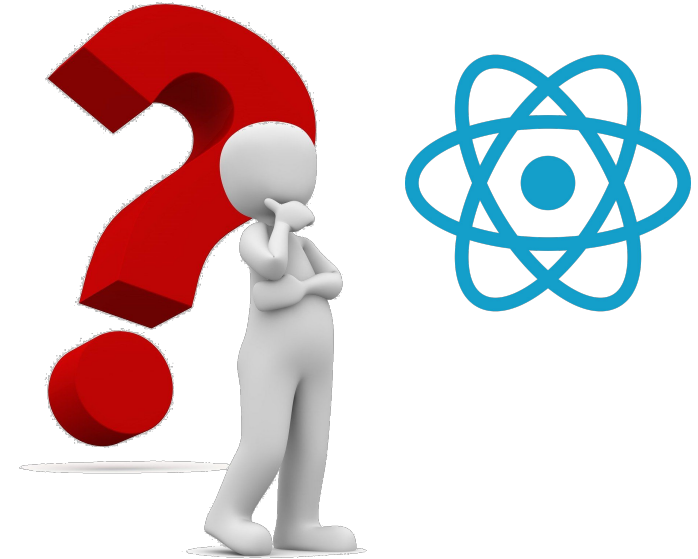
¿Qué es React?



- React es una biblioteca de JavaScript para construir interfaces de usuario.
- Desarrollada y mantenida por Facebook.
- Utiliza un DOM virtual para optimizar el rendimiento de renderizado.
- **Orientado a componentes:** Construye UI complejas a partir de pequeños y reutilizables componentes.
- **JSX:** Extensión de sintaxis que permite escribir HTML en JavaScript.
- **Unidireccional:** El flujo de datos se maneja de manera unidireccional, facilitando la previsibilidad y el debugging.
- **Ecosistema rico:** Amplia comunidad y cantidad de librerías y herramientas.

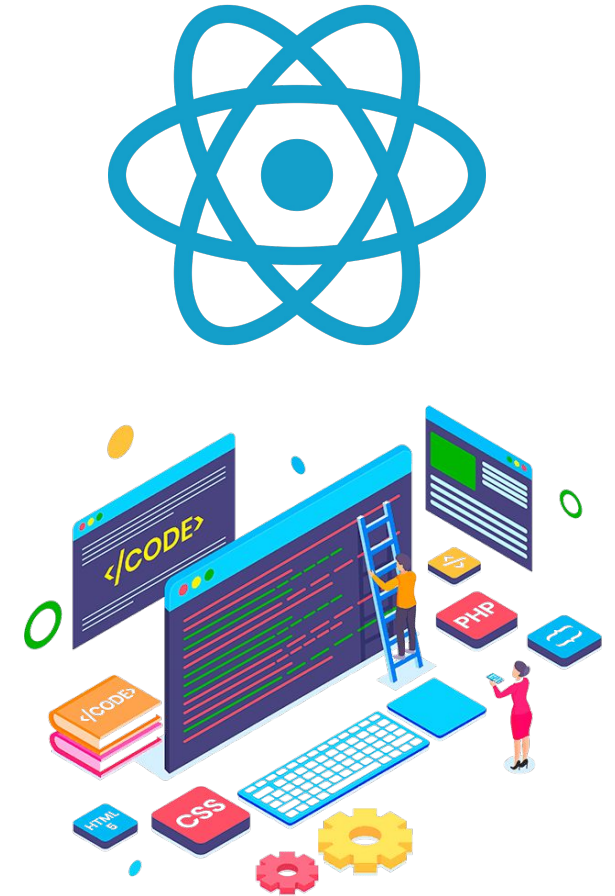
¿Por qué React?

- **Rendimiento Optimizado:** Utiliza un DOM virtual para minimizar las actualizaciones en el DOM real, mejorando así el rendimiento.
- **Componentes Reutilizables:** Permite crear componentes que pueden ser reutilizados, lo que reduce la redundancia del código.
- **Flexibilidad:** Ofrece gran libertad para elegir cómo quieres estructurar tu aplicación, sin forzar un paradigma específico.
- **Amplia Comunidad y Ecosistema:** Gran número de librerías y herramientas disponibles, además de una comunidad activa que contribuye con recursos, plugins y soluciones a problemas comunes.
- **Soporte a Largo Plazo:** Mantenido por Facebook y adoptado por muchas otras empresas, asegurando así un soporte y actualizaciones a largo plazo.



Casos de Uso de React

- **Aplicaciones Web de Una Sola Página (SPA):** Perfecto para crear aplicaciones que requieren una experiencia de usuario fluida sin recargar la página.
- **Aplicaciones Móviles:** Con React Native, puedes crear aplicaciones nativas para iOS y Android utilizando el mismo paradigma de componentes.
- **Aplicaciones Empresariales:** Ideal para construir aplicaciones escalables y mantenibles con requisitos complejos.
- **Blogs y Sitios Web Estáticos:** Puede ser utilizado en combinación con generadores de sitios estáticos para crear blogs y otros sitios web que no requieren una base de datos.
- **E-commerce:** Se usa en tiendas en línea para crear interfaces de usuario ricas y dinámicas que mejoran la experiencia de compra.
- **Dashboards y Herramientas de Análisis:** Para visualizar datos en tiempo real y crear paneles de control interactivos.
- **Realidad Virtual y 3D:** Con librerías como React VR, se pueden crear experiencias de realidad virtual.



Comparación detallada: React vs Angular

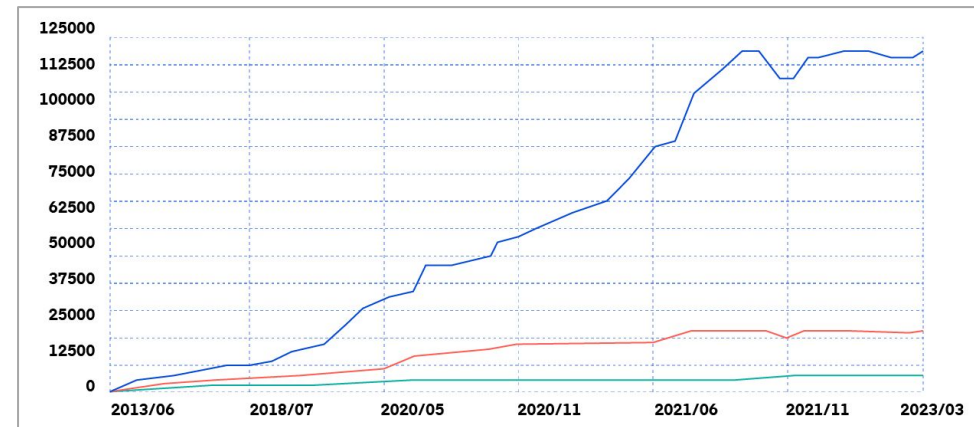
Angular:

- Angular ha crecido bastante desde principios de la década de 2010, originalmente conocido como AngularJS. A lo largo de los años,
- Angular maduró su tecnología y se hizo más popular hasta 2016, cuando se lanzó Angular 2. En ese momento, otras tecnologías estaban empezando a madurar y se estaban lanzando nuevos marcos a la comunidad.



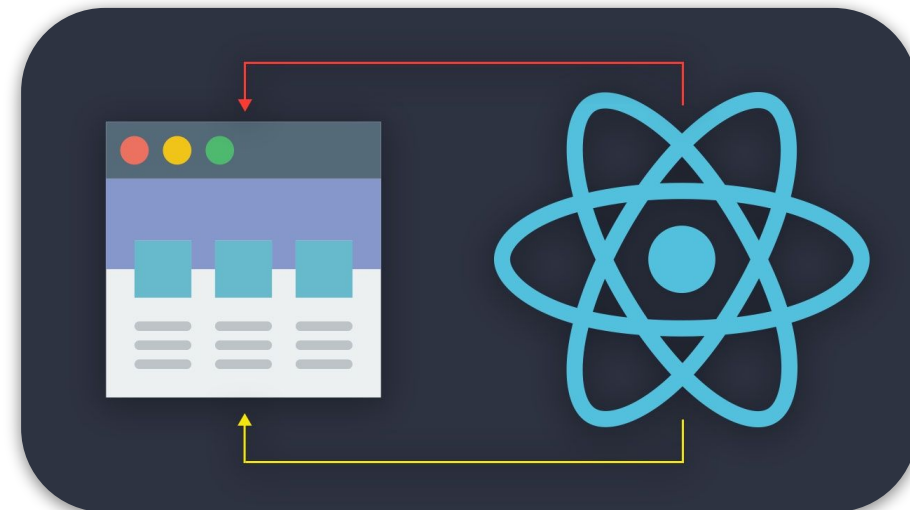
React:

- React se destaca por ser astronómicamente más popular que Angular con casi el doble de uso en los diferentes sitios.
- Probablemente esto se deba a la flexibilidad que ofrece debido a su enfoque minimalista.



Componentes en React

- Son las unidades básicas de construcción de la interfaz de usuario.
- Cada componente representa una parte de la UI que es independiente y reutilizable.
- Los componentes permiten dividir la interfaz de usuario en piezas aisladas que pueden gestionar su propio estado y ser compuestas para crear interfaces más complejas.
- Son esencialmente una función de JavaScript que toma "propiedades" (o "props") y devuelve algo que React puede renderizar en la pantalla, como un fragmento de HTML.
- Tipos de Componentes:
 - Clase (Class Components)
 - Función (Functional Components)



```
function Bienvenida(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

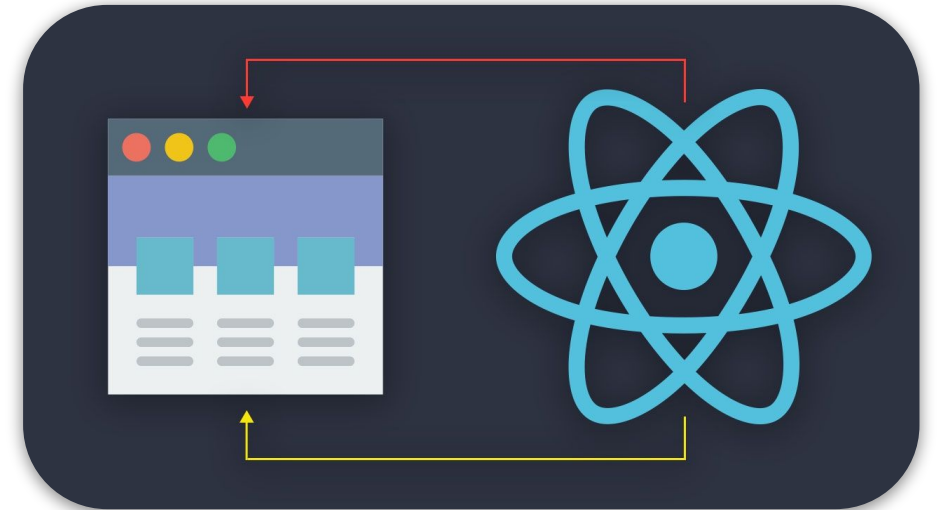
Props y Estado

Props: Son argumentos que se pasan a los componentes para configurar cómo deben aparecer o comportarse. Son inmutables, lo que significa que un componente no puede cambiar sus props.

```
<Bienvenida name="Sara" />
```

Estado: Es una forma de mantener y gestionar datos que pueden cambiar con el tiempo dentro de un componente. El estado se inicializa en el constructor y se modifica utilizando el método `setState`.

```
this.setState({ name: 'John' });
```



```
function Bienvenida(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Preparar entorno de Desarrollo: Node.js

Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

Descargar para macOS

18.16.0 LTS

Recomendado para la mayoría

20.2.0 Actual

Últimas características

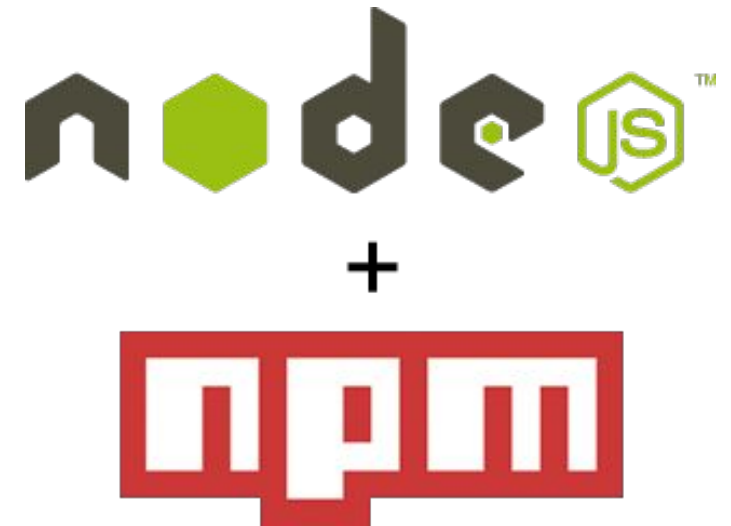
[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

[Otras Descargas](#) | [Cambios](#) | [Documentación de la API](#)

O eche un vistazo al [Programa de soporte a largo plazo \(LTS\)](#).

Instalar Nodejs

- Descargar archivo de instalación:
 - <https://nodejs.org/es>
- Seguir las instrucciones de instalación.



Uso de Componentes

Creando un Proyecto React desde cero vamos a ver en acción lo visto hasta el momento:

- Componentes
- Props.
- Estado.

Podemos crear un proyecto de React con los siguientes comandos donde **intro-react-app** es el nombre de la aplicación:

```
> npx create-react-app intro-react-app  
  
> cd intro-react-app  
  
> npm start
```



Uso de Componentes: Estructura

```
my-app
├── README.md
├── node_modules
├── package.json
├── package-lock.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```



Uso de Componentes: Repaso

- **Estructura de un Proyecto React:**
 - Exploración de la estructura de archivos generada por Create React App.
 - Carpetas clave como **src**, **public**, **node_modules**, entre otras.
- **Scripts en el Package.json:**
 - Revisión de los scripts preconfigurados como **start**, **build** y **test**.
 - Uso de **npm start** para ejecutar la aplicación en el servidor de desarrollo.
- **Componente Principal: App.js.**
 - Visualización y modificación del componente principal de la aplicación.
- **Prueba de la Aplicación:**
 - Abriendo el navegador y accediendo a la aplicación React localmente.
 - Comprobamos de que todo esté funcionando correctamente.
- **Entorno de Desarrollo en Vivo:**
 - Uso de **npm start** para habilitar actualizaciones en tiempo real.
 - Cómo se reflejan los cambios en el navegador sin necesidad de recargar la página.



Paso de datos entre componentes

```
function Usuario(props) {  
  return <h1>Hola, {props.nombre}, bienvenido/a a Desarrollo de Software</h1>;  
}  
  
function App() {  
  return (  
    <div>  
      <Usuario nombre="Sara" />  
    </div>  
  );  
}  
  
ReactDOM.render(<App />, document.getElementById('root'));
```

Virtual DOM vs. DOM Real

¿Qué es el DOM?

- El Document Object Model (DOM) es una representación estructurada de una página web en forma de árbol.
- Cada elemento HTML se representa como un nodo en el DOM.

¿Qué es el Virtual DOM?

- El Virtual DOM es una representación virtual de la estructura del DOM.
- React utiliza este Virtual DOM para realizar actualizaciones eficientes y minimizar la manipulación del DOM real.
- React actualiza el DOM real solo cuando es necesario, lo que mejora la eficiencia de las actualizaciones y reduce la carga en el navegador.
- Minimiza las manipulaciones en el DOM real, lo que reduce la sobrecarga y hace que las actualizaciones sean más rápidas.

Proceso de Actualización en React:

- Cuando un componente React se actualiza, React crea una nueva representación del Virtual DOM.
- Compara la nueva representación con la anterior para identificar los cambios.
- Luego, React actualiza sólo los elementos del DOM que han cambiado, en lugar de recargar toda la página.

Componentes Funcionales vs. Componentes de Clase en React

Componentes Funcionales

- Ventajas:
 - Sintaxis más concisa.
 - Uso de Hooks para gestionar el estado y efectos secundarios.
 - Mayor rendimiento (en general).
 - Facilita el uso de funciones puras.

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

function ContadorFuncional() {
  const [contador, setContador] = useState(0);

  const incrementar = () => {
    setContador(contador + 1);
  };

  return (
    <div>
      <p>Contador: {contador}</p>
      <button onClick={incrementar}>Incrementar</button>
    </div>
  );
}
```

Componentes de Clase

- Ventajas:
 - Manejo completo del ciclo de vida.
 - Uso de this.state y this.setState() para gestionar el estado local.
 - Compatibilidad con código heredado y bibliotecas de terceros.

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

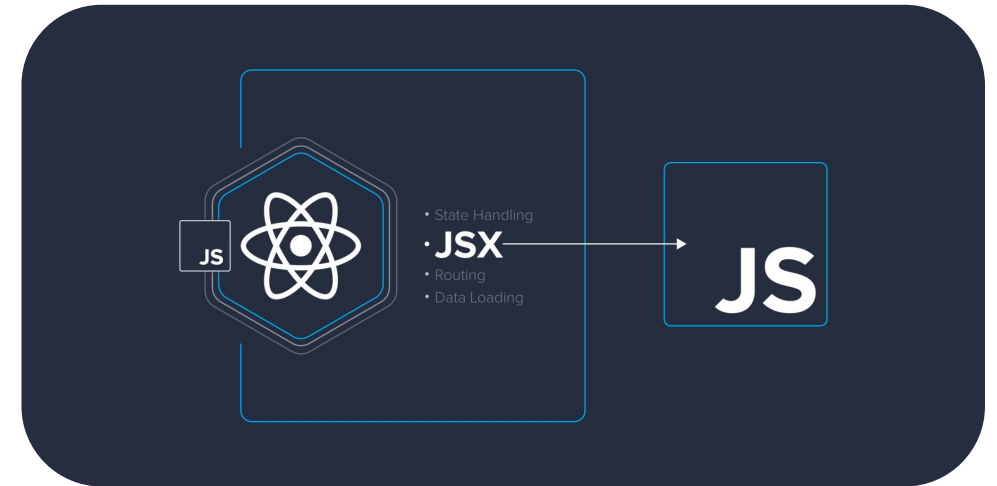
class ContadorClase extends Component {
  constructor(props) {
    super(props);
    this.state = { contador: 0 };
  }

  incrementar() {
    this.setState({ contador: this.state.contador + 1 });
  }

  render() {
    return (
      <div>
        <p>Contador: {this.state.contador}</p>
        <button onClick={() => this.incrementar()}>Incrementar</button>
      </div>
    );
  }
}
```

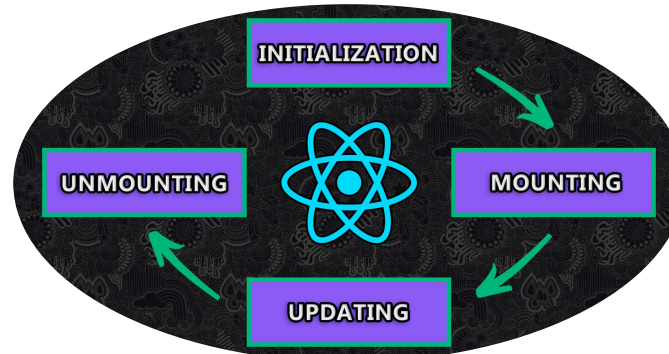
Introducción a JSX y su Papel en React

- JSX es una extensión de JavaScript utilizada en React para definir la estructura y el aspecto visual de los componentes. Comprender cómo funciona JSX es esencial para el desarrollo en React.
- Es una sintaxis de extensión de JavaScript que permite escribir código HTML dentro de JavaScript.
- **Ventajas de JSX:**
 - Facilita la visualización de la estructura del componente.
 - Permite la reutilización de componentes personalizados.
 - Integra el lenguaje de marcado HTML con JavaScript.



Estado y Ciclo de Vida de Componentes en React

- El estado y ciclo de vida de los componentes en React han evolucionado a lo largo del tiempo, con enfoques diferentes en componentes de clase y componentes funcionales. Entender estas conceptos es fundamental para el desarrollo en React.
- El ciclo de vida de un componente se basa en tres etapas:
 - **mounting (montaje):** En esta etapa, el componente se crea e inserta en el DOM. Aquí es donde el componente "nace" y se ejecutan algunas acciones iniciales.
 - **updating (actualización):** Esta etapa ocurre cada vez que el componente se actualiza debido a cambios en el estado o las props. Es como cambiar la decoración de una habitación de la casa.
 - **unmounting (desmontaje):** Esta etapa ocurre cuando el componente se elimina del DOM, es como demoler una casa.



Hooks en React

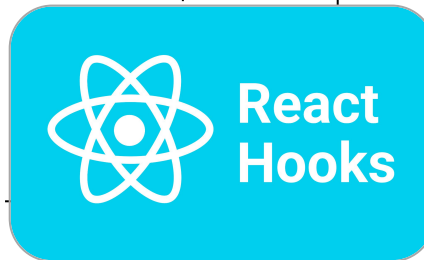
¿Qué son los Hooks?

- Los Hooks son una adición en React 16.8 que te permite utilizar el estado y otras características de React en **componentes funcionales**.

Ventajas de los Hooks:

- Simplifican la lógica del componente.
- Reutilización de la lógica del estado y los efectos secundarios.
- Reducción de la complejidad de los componentes.

- **useState:** Permite agregar estado a componentes funcionales. Se utiliza para almacenar y actualizar datos locales en el componente.
- **useEffect:** Permite realizar efectos secundarios en componentes funcionales. Se utiliza para gestionar ciclos de vida, realizar solicitudes a APIs, suscripciones a eventos y más.
- **useContext:** Facilita el acceso a datos globales compartidos, como temas o autenticación, en componentes anidados sin necesidad de pasar props manualmente.



Uso de Componentes: Más Ejercicios

- Ejemplo de cómo anidar componentes en React.
- Ejemplo práctico de manejo del estado en React.
- Ejemplo práctico del paso de props entre componentes.



Actividad 4: Paso a Paso App Encuestas en React

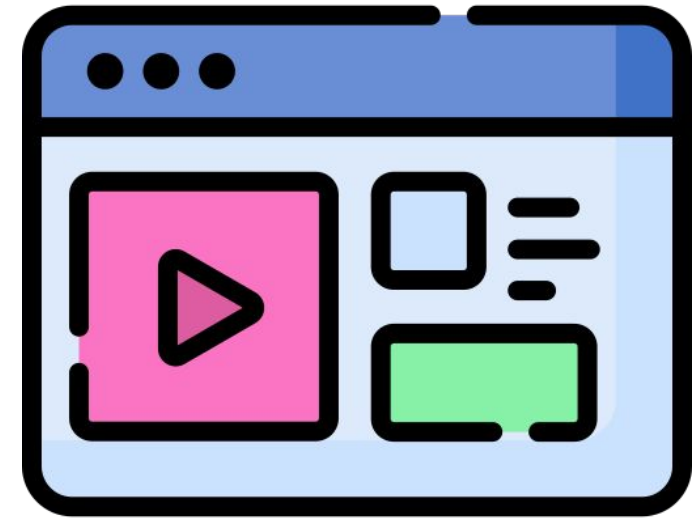
- Seguir las instrucciones de la actividad publicada en la UVE.



Próxima Clase

Desarrollo de aplicaciones web con React

- Creación de componentes React básicos.
- Uso de JSX en React.
- Trabajo con eventos en React.



MUCHAS TOTALES

ANDÉN
Centro de Innovación
y Emprendimientos Tecnológicos

SECRETARÍA DE
EXTENSIÓN
UNIVERSITARIA
UTN - FRC

SEU

UTN
Facultad Regional Córdoba

Agencia
**CÓRDOBA
JOVEN**



CÓRDOBA
entre todos