INTRODUCCIÓN AL DESARROLLO FRONTEND

CON REACT

2023

ANDÉN Centro de Innovación y Emprendimientos Tecnológicos

SECRETARÍA DE EXTENSIÓN UNIVERSITARIA UTN - FRC SEU

UTN Facultad Regional Córdoba

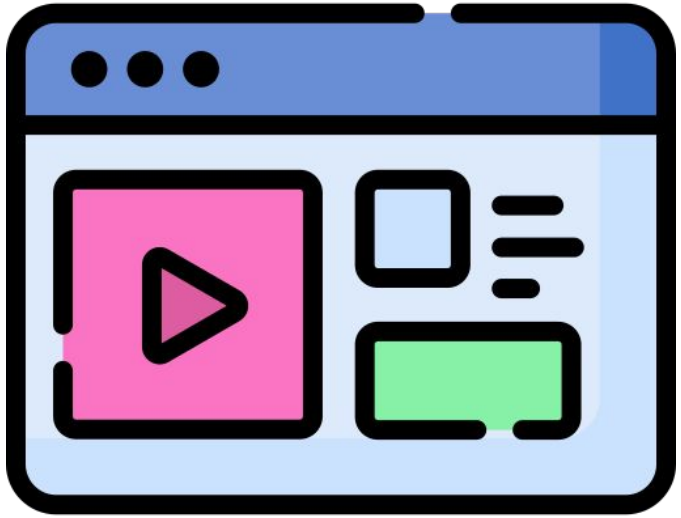
Agencia CÓRDOBA JOVEN

CÓRDOBA entre todos

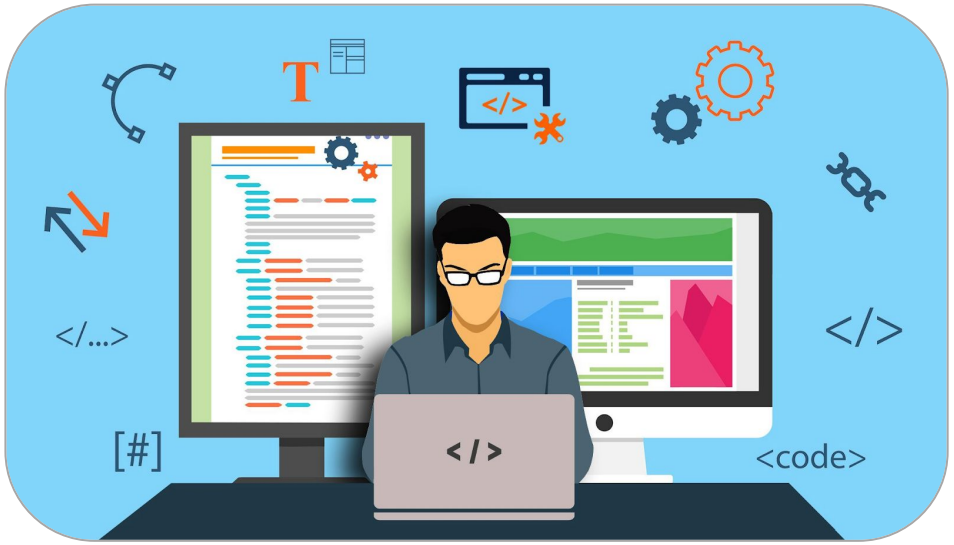
Clase 05 - Contenido

Desarrollo de aplicaciones web con React

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

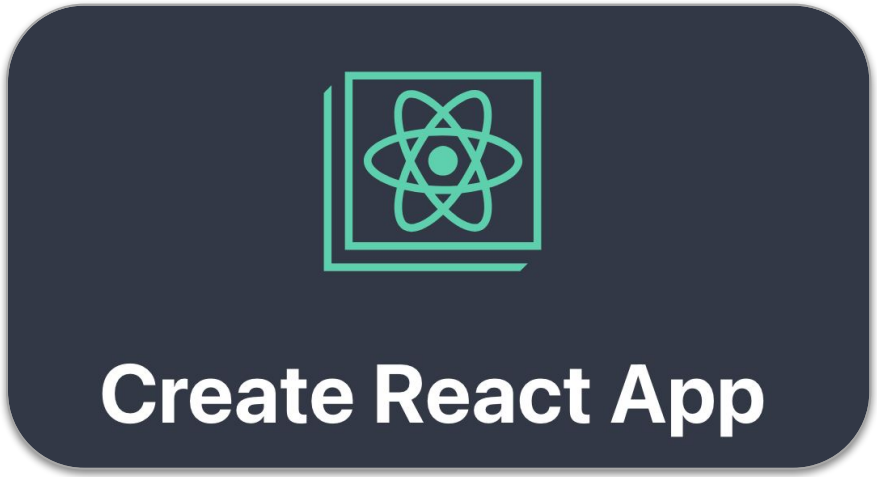


Objetivos de la Clase



- **Entender los Fundamentos de React y JSX:** Al final de la clase, los estudiantes tendrán una comprensión sólida de qué es React, cómo se estructura un proyecto y la sintaxis de JSX, incluyendo sus diferencias y ventajas respecto a HTML.
- **Desarrollar Componentes React:** Los alumnos aprenderán cómo crear tanto componentes funcionales como componentes de clase en React. Entenderán cómo usar 'props' para pasar datos entre componentes.
- **Manejar el Estado y Eventos en React:** Los estudiantes serán capaces de usar el hook **useState** para manejar el estado en componentes funcionales y cómo manejar eventos del usuario, como clics y cambios en campos de entrada.
- **Implementar Enrutamiento con React Router:** Los alumnos aprenderán los fundamentos del enrutamiento en una aplicación de React, incluyendo cómo configurar rutas y navegar entre diferentes componentes.

Creación de Proyectos con Create React App



- **¿Qué es Create React App?**
  - Una herramienta de línea de comandos para inicializar proyectos de React con una buena configuración predeterminada.
- **Pre-requisitos**
  - Tener Node.js y npm instalados.
- **Instalación de Create React App**
  - No es necesario instalarlo globalmente.
  - `npx create-react-app my-app`
    - Creará un nuevo proyecto llamado "my-app".
- **Iniciando el Proyecto**
  - `cd my-app`
    - Navegar hasta el directorio del proyecto.
  - `npm start`
    - Iniciar la aplicación en modo de desarrollo.
- **Acceso al Proyecto**
  - Abrir un navegador y navegar a `http://localhost:3000/`
  - Verás la página de bienvenida de React.
- **Conclusión**
  - Ahora tienes un proyecto de React listo para ser desarrollado.

Estructura del Proyecto

- **src/ y public/**
  - src/
    - Donde reside el código fuente de la aplicación.
  - public/
    - Archivos estáticos como HTML, imágenes, etc.
- **Componentes y Contenedores**
  - src/components/
    - Carpeta para los componentes reutilizables.
  - src/containers/
    - Carpeta para los componentes de nivel superior o contenedores.
- **Otros Archivos Importantes**
  - src/index.js
    - Punto de entrada de la aplicación.
  - src/App.js
    - Componente principal de la aplicación.
- **Archivos de Configuración**
  - package.json
    - Contiene metadatos y dependencias del proyecto.
  - .env
    - Variables de entorno.
- **Recursos y Assets**
  - public/favicon.ico
    - Icono de la página web.
  - public/index.html
    - Plantilla HTML principal.

```
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
```

¿Qué es JSX?

1. **Definición de JSX**
   a. JSX => JavaScript XML.
   b. Permite escribir HTML en JavaScript y viceversa.
2. **¿Por qué JSX?**
   a. Simplifica la creación de componentes React.
   b. Facilita la lectura y mantenimiento del código.
3. **Sintaxis Básica**

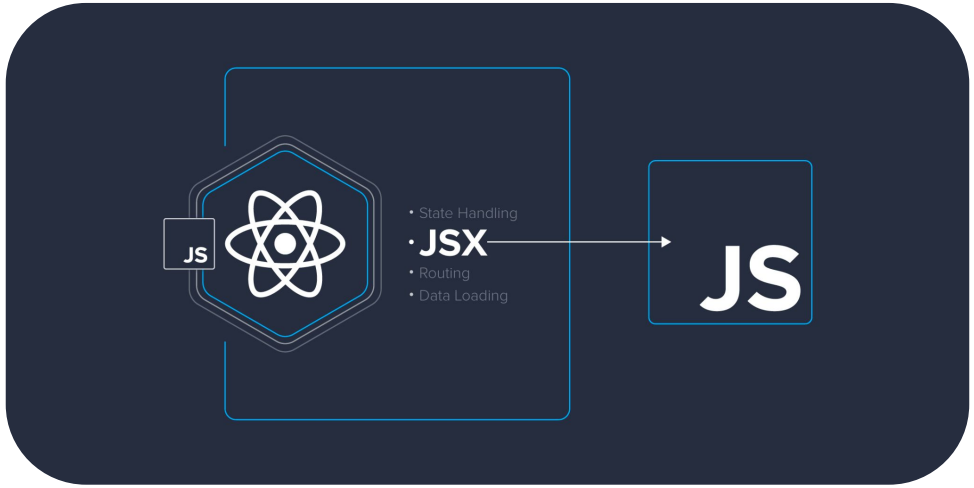
```
const elemento = <h1>Hola, mundo!</h1>;
```

4. **JSX vs HTML**
   a. Etiquetas auto-cerradas: `<img src="image.jpg" />`
   b. Nombre de las props: `className`, `onClick`, etc.
5. **Inserción de Expresiones:** Utilizar llaves para insertar expresiones.

```
nombre = 'Juan';
const elemento = <h1>Hola, {nombre}</h1>;
```

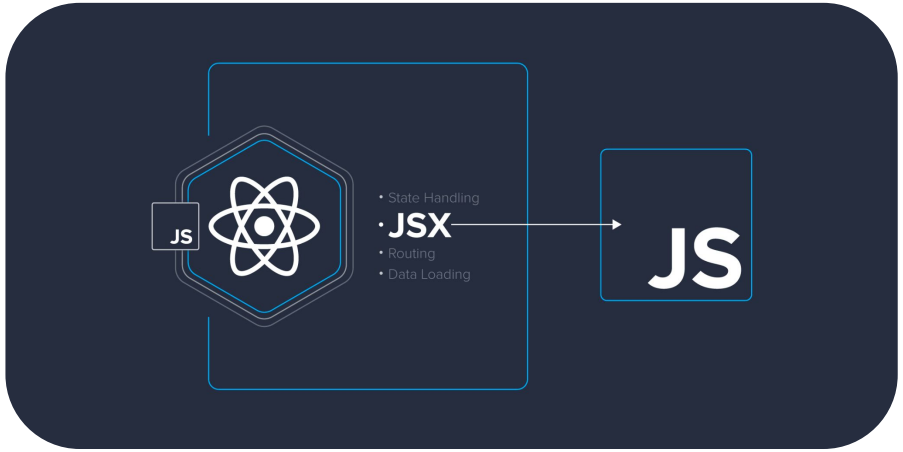
6. **Atributos y Props:** Usar className en lugar de class.

```
const elemento = <div
className="mi-clase">Contenido</div>;
```



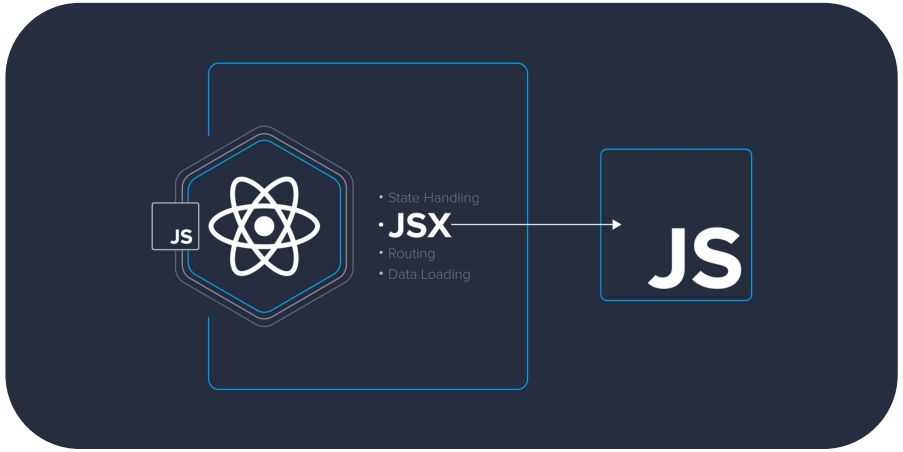
JSX vs HTML

- **Diferencias de Sintaxis**
  - Auto-cierre de etiquetas:
    - HTML: `<img src="imagen.jpg">`
    - JSX: `<img src="imagen.jpg" />`
  - Nombres de atributos:
    - HTML: `class`, `for`
    - JSX: `className`, `htmlFor`
- **Uso de JavaScript en JSX**
  - JSX permite incorporar lógica de JavaScript directamente.
  - `const elemento = <h1>{'Hola ' + nombre}</h1>;`
- **Componentización**
  - JSX facilita la creación y uso de componentes React.
  - `const Saludo = () => <h1>Hola, mundo!</h1>;`
- **Ventajas de JSX**
  - Mayor flexibilidad.
  - Fácil de leer y escribir.
  - Facilita el desarrollo de componentes reutilizables.
- **Desventajas de JSX**
  - Requiere compilación (Babel).
  - Curva de aprendizaje inicial.

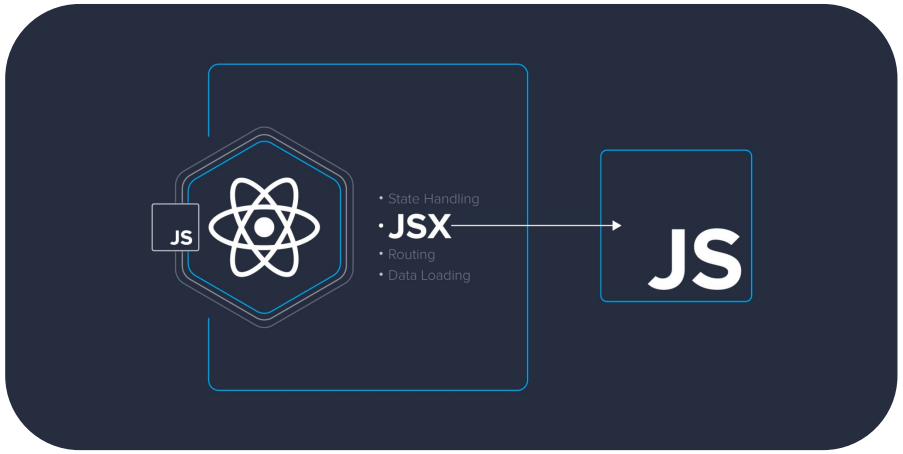


Anidamiento en JSX

- **Definición de Anidamiento en JSX**
  - Colocar un elemento JSX dentro de otro elemento JSX.
- **Sintaxis Básica:**
  - Ejemplo de anidamiento simple:
    - const anidado = <h1>Título</h1> Texto ;
- **Claves para el Anidamiento**
  - Mantener un elemento raíz:
    - const correcto = <Elemento1 /><Elemento2 /> ;
  - Evitar múltiples elementos sin raíz:
    - const incorrecto = <Elemento1 /><Elemento2 />;
- **Uso de Arrays y map()**
  - Generar listas mediante el método map().
    - const lista = [1, 2, 3].map(num => <li key={num}>{num});
- **Fragmentos de React**
  - Utilizar Fragment para evitar elementos raíz adicionales.
    - <> <Elemento1 /> <Elemento2 /> </>;
- **Anidamiento de Componentes**
  - Cómo anidar componentes personalizados.
    - const Layout = () => <Header /><Content /><Footer /> ;



Expresiones en JSX: Uso de variables y operaciones



- **Introducción a las Expresiones en JSX**: Las expresiones en JSX se escriben entre llaves `{}`.
- **Variables en JSX**: Insertar el valor de una variable dentro de JSX.

```
const nombre = 'Juan';
const saludo = <h1>Hola, {nombre}</h1>;
```

- **Operaciones Matemáticas**: Realizar operaciones matemáticas básicas.

```
const resultado = <h1>{10 + 20}</h1>;
```

- **Operaciones Condicionales**
  a. Operador ternario:
     i. `const elemento = <h1>{condicion ? 'Verdadero' : 'Falso'}</h1>;`
  b. `&&` para renderizado condicional:
     i. `{condicion && <Componente />}`
- **Uso de Funciones**: Invocar funciones dentro de JSX.

```
const obtenerSaludo = () => 'Hola, mundo';
const elemento = <h1>{obtenerSaludo()}</h1>;
```

- **Listas y Map**: Uso de `map()` para renderizar listas.

```
const numeros = [1, 2, 3];
const lista = <ul>{numeros.map(num => <li key={num}>{num}</li>)}</ul>;
```

JSX

```
const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Ejemplo de código sin JSX

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Estilos CSS en un componente en React

```jsx
function MiComponente() {
  return (
    <div className="mi-clase-css">
      Este es un componente con clase CSS
    </div>
  );
}
```

```css
.mi-clase-css {
  background-color: red;
  color: white;
}
```

Estilos CSS en un componente en React

```jsx
<div className={isActive ? "active-class" : "inactive-class"}>
  Este elemento es {isActive ? "activo" : "inactivo"}
</div>
```

```css
.active-class {
  color: green;
  font-weight: bold;
}

.inactive-class {
  color: gray;
  font-weight: normal;
}
```

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.

```jsx
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>
  );
}
```

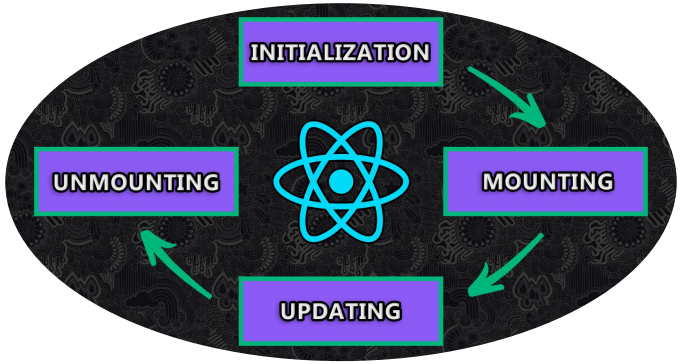
Componentes de Clase

- 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.

```jsx
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>
    );
  }
}
```

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.



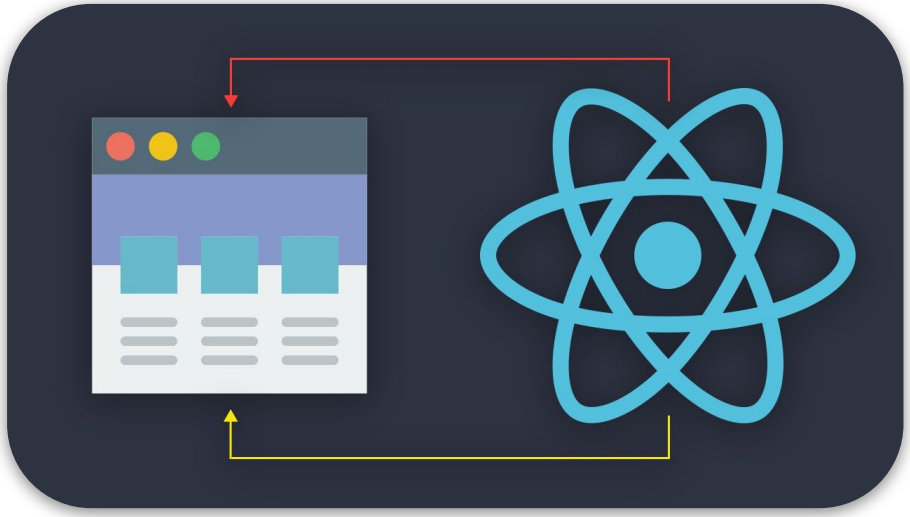
Props y Estado

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' });
```



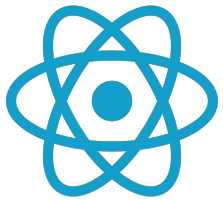
```
function Bienvenida(props) {
     return <h1>Hello, {props.name}</h1>;
}
```

Props: Paso de datos entre componentes

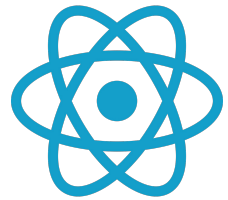
```
// Componente Usuario
function Usuario(props) {
    return <h1>Hola, {props.nombre}, bienvenido/a a Desarrollo de Software</h1>;
}


//Componente Principal: App
function App() {
    return (
        <div>
            <Usuario nombre="Sara" />
        </div>
        );
}
```

Estado: Uso de useState en Componentes Funcionales

- Estado se refiere a la data que puede cambiar durante el ciclo de vida de un componente.
- **¿Qué es useState?:** useState es un Hook de React que permite añadir estado a componentes funcionales.
- **Sintaxis Básica de useState:**
  `const [estado, setEstado] = useState(valorInicial);`
  - estado: Valor actual del estado.
  - setEstado: Función para actualizar el estado.
  - valorInicial: Valor inicial del estado.
- **Ventajas de useState**
  - Facilita la gestión del estado en componentes funcionales.
  - Mejora la legibilidad y mantenibilidad del código.



- **Ejemplo de Contador**
  `const incrementar = () => setContador(contador + 1);`
- **Renderizado Condicionado con Estado**
  `return mostrar ? <Componente /> : null;`
- **Actualización de Estado con Valores Previos**
  `setContador(prevContador => prevContador + 1);`
- **Múltiples Estados**
  `const [edad, setEdad] = useState(0);`
-



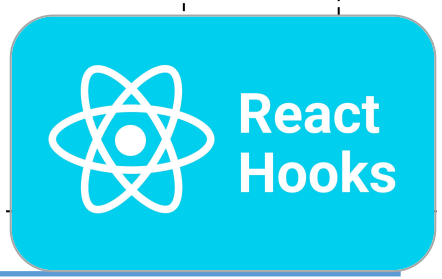
Hooks en React

¿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.

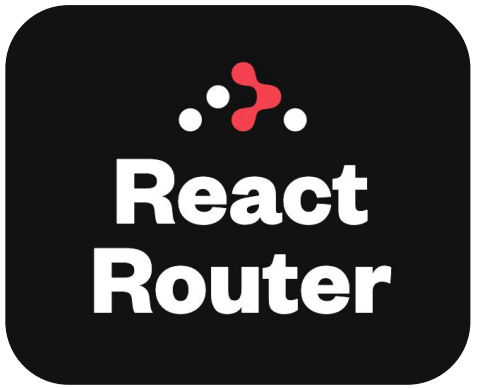
Rutas y navegación

- Los ruteos del lado del cliente (browser) permiten que la aplicación actualice la URL sin necesidad de hacer un request o pedido al servidor, sino que inmediatamente renderiza el componente de la interfaz del usuario.
- El ruteo del lado del cliente se habilita en React, agregando una librería de ruteo, y luego crear un "Router" y enlazando las páginas con 'Link' y 'Form'.

```
npm install react-router-dom
```

Ventajas

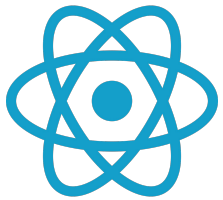
- Permite experiencias de usuario **más rápidas** porque el navegador no necesita solicitar un documento completamente nuevo o volver a evaluar los activos.
- Permite experiencias de usuario **más dinámicas** porque es posible aplicar animaciones.



Enrutamiento en React

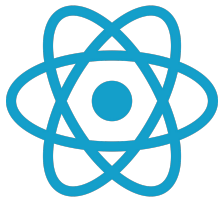
- **Introducción al Enrutamiento**
  - El enrutamiento permite navegar entre diferentes componentes o páginas en una aplicación de React.
- **React Router**
  - La librería más popular para manejar enrutamiento en React.
- **Instalación de React Router**
  - `npm install react-router-dom`
  - Cómo instalar React Router en tu proyecto.
- **Componentes Básicos**
  - `BrowserRouter`, `Route`, `Link`, `Switch`

```
<BrowserRouter>
       <Link to="/">Inicio</Link>
       <Switch>
         <Route path="/" component={InicioComponente} />
       </Switch>
     </BrowserRouter>
```



- **Ventajas del Enrutamiento en React**
  - Permite crear aplicaciones de una sola página (SPA).
  - Mejora la experiencia de usuario al evitar recargas de página.
- **Hooks en React Router**
  - `useParams`: Captura parámetros dinámicos en la URL.
  - `useLocation`: Proporciona información sobre la ubicación actual (ruta).
  - `useHistory:` Proporciona acceso al historial de navegación.

```
const { id } = useParams();
const location = useLocation();
const history = useHistory();
```



Agregar una Ruta

```
function App() {
  return (
    <BrowserRouter>
      <Router>
        <div className="App">
          <Routes>
            <Route path="/" element={Home} />
          </Routes>
        </div>
      </Router>
    </BrowserRouter>
  );
}

export default App;
```

Agregar una Ruta

```
import { BrowserRouter, Route, Routes, Navigate  } from "react-router-dom";
```

```
function App() {
  return (
    <BrowserRouter>
      <Router>
        <div className="App">
          <Routes>
            <Route path="/" element={Home} />
          </Routes>
        </div>
      </Router>
    </BrowserRouter>
  );
}

export default App;
```

- Con Routes, React Router recorre cada ruta en el orden en que aparecen dentro del componente y compara la URL actual con la propiedad "path" de cada ruta. **La primera ruta que coincida con la URL se renderizará**, y las demás rutas no se renderizarán.
- **Routes utiliza una sintaxis de árbol de rutas anidadas.** Esto significa que podemos definir rutas anidadas dentro de otras rutas, lo que hace que sea más fácil y flexible definir rutas complejas en la aplicación.

Agregar una Ruta

```jsx
// Inicio.js
export default function Inicio() {
  return <h2>Esta es la página de inicio</h2>;
}

// Acerca.js
export default function Acerca() {
  return <h2>Esta es la página de Acerca</h2>;
}

// Detalle.js
export default function Detalle({ id }) {
  return <h2>Detalle del elemento con ID: {id}</h2>;
}

// Error404.js
export default function Error404() {
  return <h2>La página que buscas no existe (Error 404)</h2>;
}
```

```jsx
import { BrowserRouter, Route, Routes, Switch } from "react-router-dom";
import Acerca from "./components/Acerca";
import Inicio from "./components/Inicio";
import Detalle from "./components/Detalle";
import Error404 from "./components/Error404"

function App() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Inicio />} index />
          <Route path="/acerca" element={<Acerca />} />
          <Route path="/detalle/:id" element={<Detalle />} />
          <Route path="*" element={<Error404 />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Hook useParams

```jsx
// Inicio.js
export default function Inicio() {
  return <h2>Esta es la página de inicio</h2>;
}

// Acerca.js
export default function Acerca() {
  return <h2>Esta es la página de Acerca</h2>;
}

// Detalle.js
export default function Detalle({ id }) {
  return <h2>Detalle del elemento con ID: {id}</h2>;
}

// Error404.js
export default function Error404() {
  return <h2>La página que buscas no existe (Error 404)</h2>;
}
```

```jsx
import { BrowserRouter, Route, Routes, Switch } from "react-router-dom";
import Acerca from "./components/Acerca";
import Inicio from "./components/Inicio";
import Detalle from "./components/Detalle";
import Error404 from "./components/Error404"

function App() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Inicio />} index />
          <Route path="/acerca" element={<Acerca />} />
          <Route path="/detalle/:id" element={<Detalle />} />
          <Route path="*" element={<Error404 />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Hook useParams

Ruta: /productos/:categoria/:id

```jsx
// Producto.js
import { useParams } from 'react-router-dom';

export default function Producto() {
  const { categoria, id } = useParams();

  return (
    <div>
      <h2>Producto</h2>
      <p>Categoría: {categoria}</p>
      <p>ID: {id}</p>
    </div>
  );
}
```

UseNavigate

```
import { useNavigate } from 'react-router-dom';
const navigate = useNavigate();

 <button onClick={()=>navigate("contactos")}>Contactos</button>
 <button onClick={()=>navigate("acercade")}>Acerca De</button>
```

Formularios

- Para trabajar con React se recomienda el uso de la librería React Hook Form
- Permite:
    - Validar campos de entrada.
    - Rendimiento, ya que solo renderiza el input que cambio.
    - Facilita el manejo de errores al mostrar mensajes de una manera sencilla.
    - Integración con bibliotecas de diseño como Bootstrap y Material-UI.

react-hook-form: Instalación

- npm install react-hook-form

Ver más info en:

https://react-hook-form.com/get-started#Registerfields

Componentes principales de react-hook form

- **useForm**: inicializar la lógica de un formulario

```
function MiFormulario() {
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm();
```

- **register**: para registrar campos de entrada.
- **handleSubmit**: maneja la lógica de envío de datos del formulario.
- **formState**: información de estado, tales como errores de validación.

Validaciones de react-hook-form

Reglas de validación soportadas:

- **required**
- **min**
- **max**
- **minLength**
- **maxLength**
- **pattern**
- **validate**

```
import React from "react";
import { useForm } from "react-hook-form";
function MiFormulario() {
  const {register, handleSubmit, formState: { errors }} = useForm();
  const onSubmit = (data) => {    console.log("Datos del formulario:", data);  };
  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <label htmlFor="name">Nombre:</label>
      <input id="name" type="text"
      {...register("name", { required: "El campo nombre es requerido",
            minLength: { value: 8, message: 'El campo debe tener al menos 8 caracteres'  }})}

      />
      {errors.name && <p>{errors.name.message}</p>}
      <button type="submit">Enviar</button>
    </form>
  );
}
export default MiFormulario;
```

Bueno, Vamo a Codea!!!!

Enunciado del Ejemplo: Mini Catálogo de Libros con React

Descripción
En este ejemplo, los estudiantes desarrollarán un mini catálogo de libros usando React. La aplicación permitirá a los usuarios ver una lista de libros y obtener más información al hacer clic en un título. Los libros tendrán una categoría asignada (e.g., "Ciencia Ficción", "Historia").

Funcionalidades

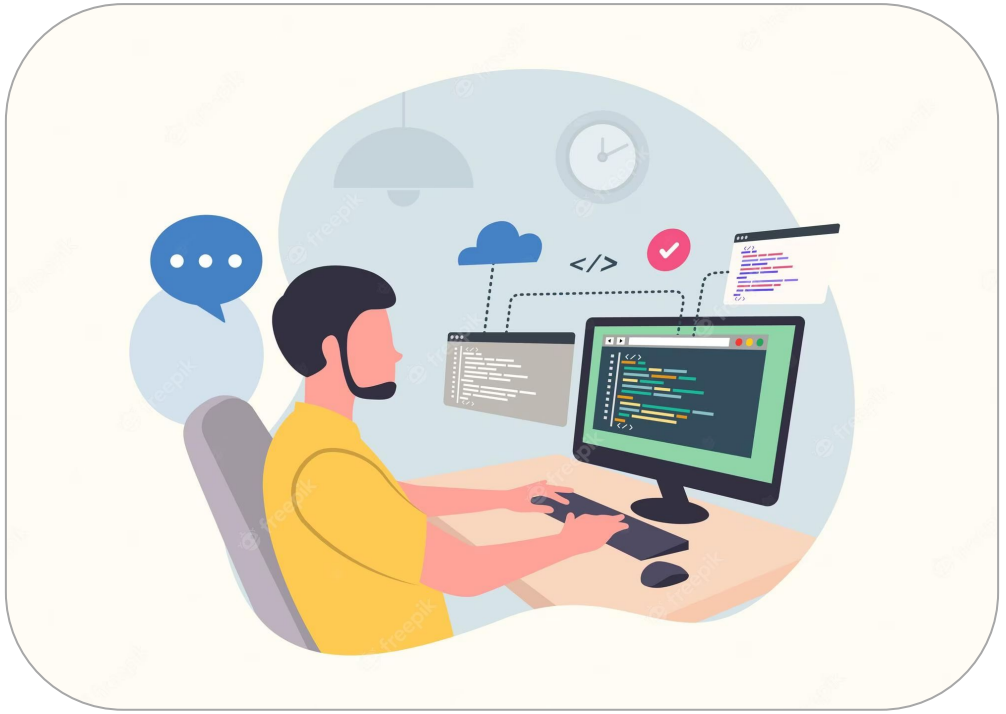
- Mostrar una lista de libros con título y autor.
- Filtrar los libros por categoría mediante un menú desplegable.
- Hacer clic en un libro para ver más detalles como sinopsis y precio.

Conceptos a Aplicar

- Creación del primer proyecto con Create React App.
- Estructura de un proyecto React.
- Uso de JSX para crear la UI.
- Creación y uso de componentes funcionales.
- Uso de **useState** para manejar el estado de la aplicación.
- Implementación de enrutamiento para navegar a la página de detalles del libro.
- Manejo de eventos como cambios en el menú desplegable y clics en los títulos de los libros.

Actividad 5: Paso a Paso App Encuestas en React

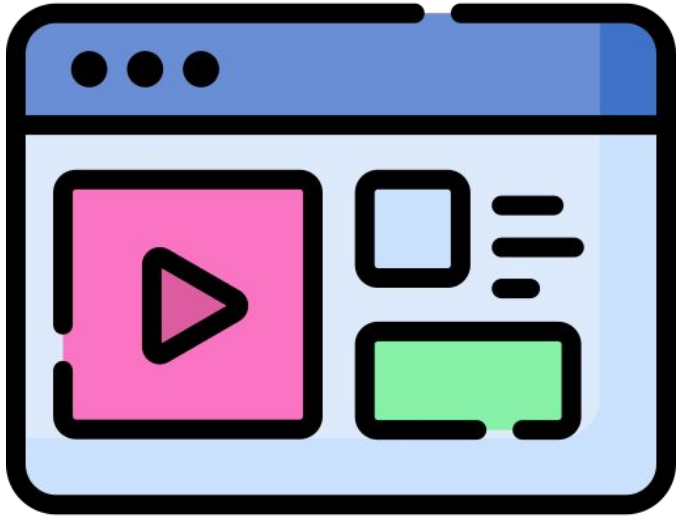
- Seguir las instrucciones de la actividad publicada en la UVE.



Próxima Clase

Generación de código fuente asistida por IA con ChatGPT

- ¿Qué es ChatGPT y cómo funciona?
- Uso de ChatGPT como herramienta de generación de código.
- Integración de ChatGPT en una aplicación 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