Xtratego

ACADEMY

UTE EC


UTE
EC
tratego
ACADEMY

UTE EC | Xtratego ACADEMY

Modulo 2

CURSO

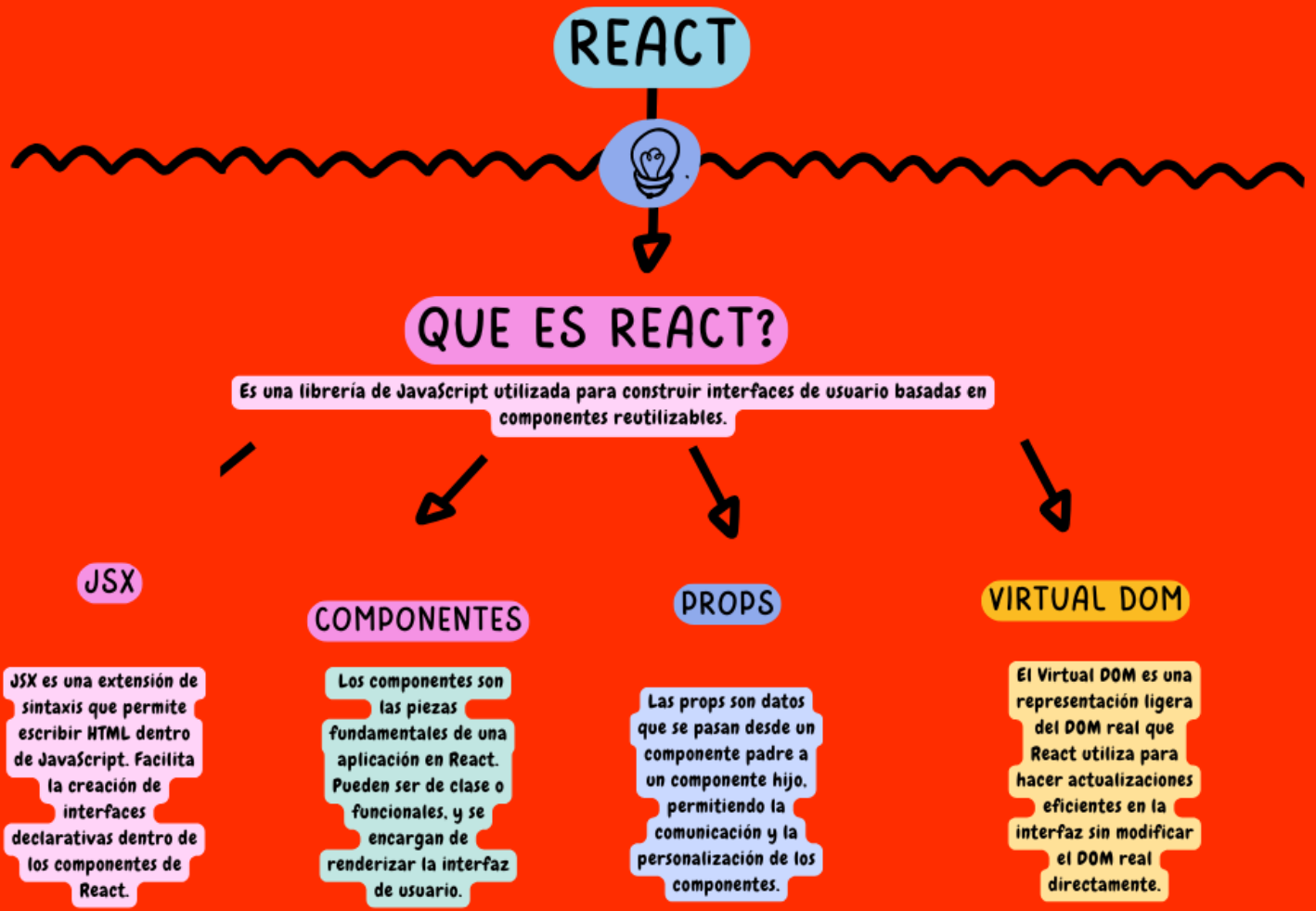
DESARROLLO FRONT END

xtratego.ec

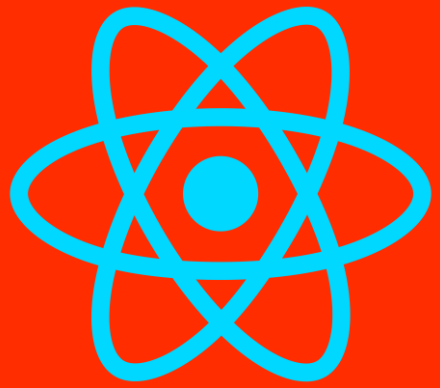
Unidad 1

Introducción a React.js

Introducción a React.js



Origen de React.js



Creación en Facebook en 2011 por Jordan Walke.

React fue diseñado para mejorar el rendimiento y la gestión de interfaces de usuario complejas en grandes aplicaciones

Ventajas de React

- **Modularidad:** Desarrollar aplicaciones utilizando componentes reutilizables.
- **Virtual DOM:** Mejora la eficiencia al actualizar solo los elementos necesarios en la interfaz.
- **Comunidad activa:** Ecosistema rico en herramientas y librerías.

UTE EC | Xtratego ACADEMY

Filosofía declarativa

React adopta un enfoque declarativo, lo que significa que los desarrolladores describen cómo debería verse la interfaz en un momento dado, y **React** se encarga de actualizar el DOM.

JSX

Sintaxis que permite escribir **HTML** dentro de **JavaScript**.

Aunque se parece al **HTML**, **JSX** permite el uso de lógica **JavaScript** dentro de las etiquetas

Virtual DOM

Es una representación ligera del **DOM** real. **React** actualiza de manera eficiente solo los componentes que han cambiado, lo que mejora el rendimiento en comparación con las manipulaciones directas del **DOM**

https://nodejs.org/es

Node.js es un entorno de ejecución de JavaScript del lado del servidor.

```
node --version
```

Ejemplo Node + Express

1

```
npm init
```

2

```
npm install express
```

4

```
node server.js
```

3

```
// Importar Express
const express = require('express');

// Crear una instancia de Express
const app = express();

// Configurar el puerto en el que escuchará el servidor
const port = 3000;

// Ruta básica de prueba
app.get('/', (req, res) => {
  res.send('¡Hola Mundo! Servidor Express en funcionamiento.');
});

// Iniciar el servidor
app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

Configuración inicial React

```
npx create-react-app my-app
```

Herramienta para generar rápidamente un proyecto con la estructura base de **React**, sin la necesidad de configurar manualmente **Webpack, Babel, etc.**

Archivos clave en un proyecto React

index.html: Página HTML donde se montará la aplicación React.

index.js: Punto de entrada principal, donde se renderiza el componente raíz (App.js).

App.js: Componente raíz de la aplicación.

package.json: Información sobre dependencias y scripts del proyecto.

Entorno de desarrollo

React Developer Tools: Extensión para depurar aplicaciones React.

Hot Reloading: Recarga automática de la aplicación tras cambios en el código.

UTE EC | Xtratego ACADEMY

Componentes en React

Un **componente** es una función o clase de **JavaScript** que acepta **props** como entrada y devuelve elementos **React** que describen cómo debe verse la interfaz.

Los componentes permiten dividir la **UI** en piezas independientes y reutilizables

Componentes funcionales

Los componentes funcionales son funciones de **JavaScript** que aceptan **props** como argumento y devuelven **JSX**

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

Componentes de clase

Son clases de JavaScript que extienden de React.Component y pueden tener estado interno.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hola, {this.props.name}</h1>;
  }
}
```

Aunque los componentes de clase eran comunes en versiones anteriores de React, hoy en día se recomienda usar componentes funcionales con hooks

Modularidad y reutilización

Los componentes permiten crear interfaces UI modulares y mantener una separación clara de responsabilidades.
Los componentes pueden anidarse y reutilizarse en diferentes partes de la aplicación

UTE EC | Xtratego ACADEMY

¿Qué son las props?:

Las props son el mecanismo mediante el cual los componentes reciben datos del componente padre. Son inmutables dentro del componente que las recibe.

```
function App() {
  return <Welcome name="Carlos" />;
}
```

Uso de Props en componentes

Se accede a las **props** en los componentes funcionales como argumentos **(props.name)** y en los de clase como **this.props.name**

```
function App() {
  return <Welcome name="Carlos" />;
}
```

Validación de Props

React proporciona la biblioteca **PropTypes** para validar las props que se pasan a los componentes y asegurar que sean del tipo correcto

```
import PropTypes from 'prop-types';

function Welcome(props) {
  return <h1>Hola, {props.name}</h1>;
}
Welcome.propTypes = {
  name: PropTypes.string.isRequired,
};
```

Ant Design (abreviado como **Antd**) es un **framework de componentes de interfaz de usuario (UI)** que se utiliza en el desarrollo de aplicaciones web con **React**

```
npm install antd
```

Ejemplo: modulo2/react-1

Desarrollar un componente que reciba una lista de nombres como props y los renderice dentro de un componente de lista.
Utilizar un componente User que reciba cada nombre individual como prop