**REACT**

React is a JavaScript library for building user interfaces.

React is used to build single-page applications.

React allows us to create reusable UI components.

The create-react-app tool is an officially supported way to create React applications.

Run this command to create a React application named my-react-app:

npx create-react-app my-react-app

create-react-app will set up everything you need to run a React application.

create-react-app will set up everything you need to run a React application.

**Note:** If you've previously installed create-react-app globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of create-react-app. To uninstall, run this command: npm uninstall -g create-react-app.

Run the React Application

Run this command to move to the my-react-app directory:

cd my-react-app

Run this command to execute the React application my-react-app:

npm start

A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.

**React is a tool for building UI (INTERFAZ DE USUARIO) components.**

How does React Work?

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React only changes what needs to be changed!

React finds out what changes have been made, and changes **only** what needs to be changed.

create-react-app includes built tools such as webpack, Babel, and ESLint.

To use React in production, you need npm which is included with [Node.js](https://nodejs.org/).

To get an overview of what React is, you can write React code directly in HTML.

But in order to use React in production, you need npm and [Node.js](https://nodejs.org/) installed.

The quickest way start learning React is to write React directly in your HTML files.But for production you will need to set up a **React environment**.

Setting up a React Environment

If you have npx and Node.js installed, you can create a React application by using create-react-app.

If you've previously installed create-react-app globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of create-react-app.

To uninstall, run this command: npm uninstall -g create-react-app.

Modify the React Application

So far so good, but how do I change the content?

Look in the my-react-app directory, and you will find a src folder. Inside the src folder there is a file called App.js, open it and try changing the HTML content and save the file.

ANTES DE CONTINUAR ES NECESARIO SABER:

Classes

ES6 introduced classes.

A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.

class Car {

constructor(name) {

this.brand = name;

}

}

Notice the case of the class name. We have begun the name, "Car", with an uppercase character. This is a standard naming convention for classes.

Now you can create objects using the Car class:

### **Example**

Create an object called "mycar" based on the Car class:

class Car {

constructor(name) {

this.brand = name;

}

}

const mycar = new Car("Ford");

**Note:** The constructor function is called automatically when the object is initialized.

## Method in Classes

You can add your own methods in a class:

### **Example**

Create a method named "present":

class Car {

constructor(name) {

this.brand = name;

}

present() {

return 'I have a ' + this.brand;

}

}

const mycar = new Car("Ford");

mycar.present();

As you can see in the example above, you call the method by referring to the object's method name followed by parentheses (parameters would go inside the parentheses).

Class Inheritance

To create a class inheritance, use the extends keyword.

A class created with a class inheritance inherits all the methods from another class:

### **Example**

Create a class named "Model" which will inherit the methods from the "Car" class:

class Car {

constructor(name) {

this.brand = name;

}

present() {

return 'I have a ' + this.brand;

}

}

class Model extends Car {

constructor(name, mod) {

super(name);

this.model = mod;

}

show() {

return this.present() + ', it is a ' + this.model

}

}

const mycar = new Model("Ford", "Mustang");

mycar.show();

The super() method refers to the parent class.

By calling the super() method in the constructor method, we call the parent's constructor method and get access to the parent's properties and methods.

# **React Render HTML**

React's goal is in many ways to render HTML in a web page.

React renders HTML to the web page by using a function called createRoot() and its method render().

The createRoot() function takes one argument, an HTML element.

The purpose of the function is to define the HTML element where a React component should be displayed.

The render() method is then called to define the React component that should be rendered.

But render where?

There is another folder in the root directory of your React project, named "public". In this folder, there is an index.html file.

You'll notice a single <div> in the body of this file. This is where our React application will be rendered.

### **Example**

Display a paragraph inside an element with the id of "root":

const container = document.getElementById('root');

const root = ReactDOM.createRoot(container);

root.render(<p>Hello</p>);

The result is displayed in the <div id="root"> element:

<body>

<div id="root"></div>

</body>

Note that the element id does not have to be called "root", but this is the standard convention.