**SECCIÓN 2**

* ¿Qué es React?
* Conceptos generales
* Babel
* JSX

**SECCIÓN 3**

[**https://developer.mozilla.org/es/**](https://developer.mozilla.org/es/)

* Generar la base sobre JavaScript
* Constantes y variables Let
* Template String
* Objetos literales
* Arreglos
* Desestructruación \* (sumamente importante)
* Promesas
* Fetch API
* Ternarios
* Async – Await

**SECCIÓN 4**

* Nuestra primera aplicación - Hola Mundo
* Exposiciones sobre los componentes
* Creación de componentes (Functional Components)
* Propiedades - Props
* Impresiones en el HTML
* PropTypes
* DefaultProps
* Introducción general a los Hooks
* useState

**SECCIÓN 5**

[**https://jestjs.io/**](https://jestjs.io/)

* Introducción a las pruebas
* AAA
  + Arrange - Arreglar
  + Act - Actuar
  + Assert - Afirmar
* Primeras pruebas
* Jest
* Expect
* toBe
* Enzyme
* Comandos útiles en la terminal para pruebas
* Revisar elementos renderizados en el componente
* Simular eventos

Jest es compatible con react testing library

npm install --save-dev jest

package.json añadir línea test

  "scripts": {

    "dev": "vite",

    "build": "vite build",

    "preview": "vite preview",

    "test": "jest --watchAll"

  },

Ejecutar

Npm run test

1er. Test en demo.test.js

Documentación expect

<https://jestjs.io/docs/expect>

instalar los tipos para que vscode nos de la ayuda

npm i @types/jest

si queremos la última versión añadir @types/jest@latest

al usar babel añadir

**sing Babel**

To use [Babel](https://babeljs.io/), install required dependencies:

* **npm**
* **Yarn**
* **pnpm**

npm install --save-dev babel-jest @babel/core @babel/preset-env

Configure Babel to target your current version of Node by creating a babel.config.js file in the root of your project:

babel.config.js

module.exports = {  
 presets: [['@babel/preset-env', {targets: {node: 'current'}}]],  
}

el las fuentes de herrera está

module.exports = {

    presets: [

        [ '@babel/preset-env', { targets: { esmodules: true } } ],

        [ '@babel/preset-react', { runtime: 'automatic' } ],

    ],

};

En el vídeo 12 usa este api para los tests async

<https://developers.giphy.com/>

en vídeo 14 también comenta que es buena la librearía de test **react testing library**

*que utiliza para testear renderizados y eventos de los componentes complementa jest con:*

<https://testing-library.com/docs/>

es buena para manejar el DOM virtual

En el propio jest lo recomiendan para el DOM:

<https://jestjs.io/docs/tutorial-react>

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se instala con:

npm install --save-dev @testing-library/react

él tiene en el archive jest.config.js, a mi creo que me funciona sin él

*// import '@testing-library/jest-dom';*

 "jest-environment-jsdom": "^28.0.2",

También hay que poner en el bael.config.js

module.exports = {

    presets: [

        [ '@babel/preset-env', { targets: { esmodules: true } } ],

        [ '@babel/preset-react', { runtime: 'automatic' } ],

    ],

};

Esto si parece que es obligatorio, si lo quito falla

Viene en el paquete:  "@babel/preset-react": "^7.16.7",

**SECCIÓN 6**

* Custom Hooks
* Fetch hacia un API
* Comunicación entre componentes
* Clases de CSS
* Animaciones
* Enviar métodos como argumentos
* Crear listados
* keys
* Giphy

Esta es una aplicación pequeña pero muy ilustrativa que explica cómo utilizar React + customHooks para poder resolver necesidades en específico que podremos reutilizar después.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

Usa el API

<https://developers.giphy.com/>

Explicar estructura de carpetas del proyecto

Y como ver la APP como un árbol de componentes

estructura de archivos (descatalogado)

<https://es.legacy.reactjs.org/docs/faq-structure.html>

<https://hackernoon.com/structuring-projects-and-naming-components-in-react-1261b6e18d76>

css del proyecto:

<https://gist.github.com/Klerith/e4fca6ac127ccb9abd1e8ad77fcd52f1>

descripción del api de gifs

<https://developers.giphy.com/docs/api/endpoint#trending>

stric mode en vídeo 13, duplica llamadas. También se explica useEffect

<https://legacy.reactjs.org/docs/strict-mode.html>

Vídeo 20, archivos de barril o índice

Un **archivo de barril** (en inglés: **barrel file**) es un archivo que **reexporta** varios módulos desde un solo lugar, facilitando y simplificando las importaciones en tu proyecto.

**📦 ¿Para qué sirve un archivo de barril?**

Permite que otros módulos importen desde **un solo archivo central**, en lugar de importar cada cosa individualmente desde múltiples archivos.

**🔧 Ejemplo práctico en JavaScript/TypeScript**

Supongamos que tienes una carpeta utils/ con varios archivos:

utils/

├── formatDate.js

├── calculateTax.js

└── validateEmail.js

**🧾 Sin archivo de barril:**

import formatDate from './utils/formatDate';

import calculateTax from './utils/calculateTax';

import validateEmail from './utils/validateEmail';

**🛢️ Con archivo de barril (utils/index.js):**

// utils/index.js

export { default as formatDate } from './formatDate';

export { default as calculateTax } from './calculateTax';

export { default as validateEmail } from './validateEmail';

Ahora puedes importar todo desde un solo lugar:

import { formatDate, calculateTax, validateEmail } from './utils';

**✅ Ventajas de usar archivos de barril**

* **Código más limpio** y ordenado.
* **Centraliza los imports/exports**.
* Facilita el **refactoring** (si cambias la estructura de archivos, solo cambias el barril).
* Muy útil en estructuras grandes (como componentes en React, servicios, utils, etc.).

**🧠 ¿Cuándo usarlo?**

* Cuando tienes muchos módulos en una carpeta y necesitas importarlos frecuentemente juntos.
* Especialmente útil en proyectos **modulares o con muchos componentes**.

**SECCIÓN 7**

**nuevas características gemini code:**

[Gemini Code Assist release notes  |  Google for Developers](https://developers.google.com/gemini-code-assist/resources/release-notes)

* Aprender cómo realizar backups a repositorios de Git
* Subir nuestro repositorio a GitHub
* Uso de Github Pages
* Desplegar nuestra aplicación de React
* Generar build de producción de nuestra aplicación

Github Pages:

Ir a **settings** (en el menú superior), y luego en el menú lateral a **Pages**

**SECCIÓN 13**

* Subir código a nuestro repositorio
* Tener un listado de customHooks y código que podemos reutilizar
* Una idea para mantener tu repositorio ordenado con ejemplos