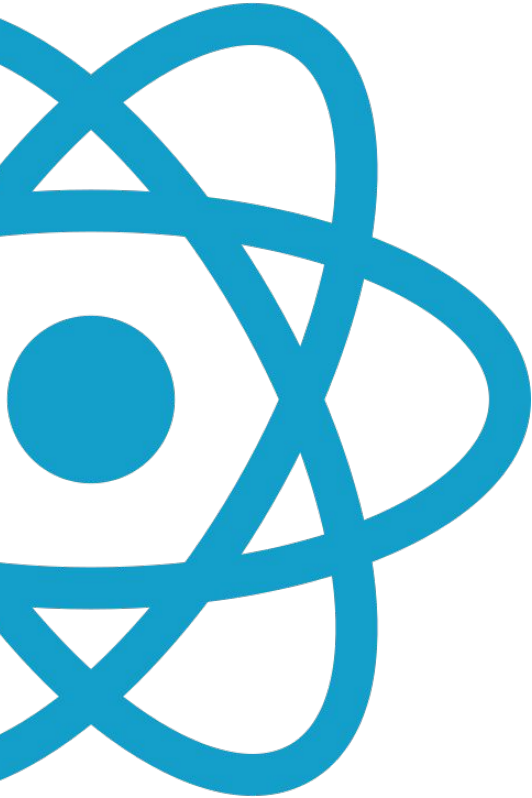


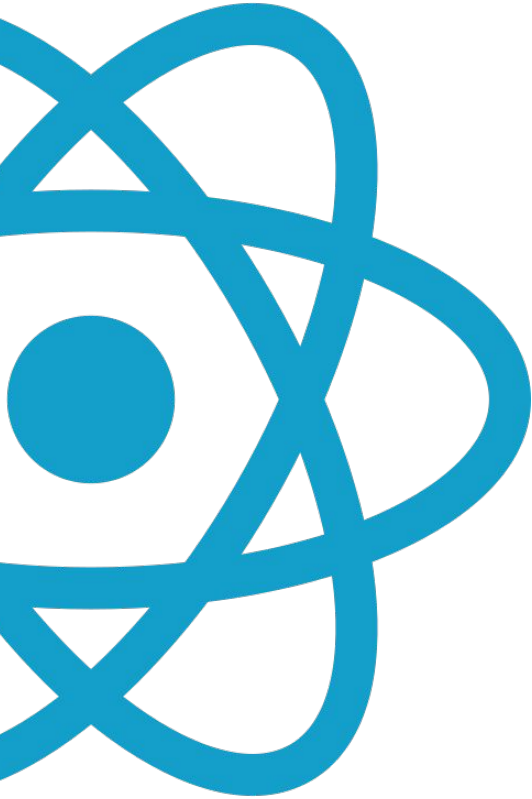
# ReactJS

Curso Programación Web / Diseño de medios Interactivos  
(2023)



## ¿Qué es ReactJS?

- Es una librería para crear aplicativos web. Estos aplicativos pueden ser sencillos o altamente robustos y React les provee un alto nivel de interactividad.
- Es una librería declarativa: *Siempre se describe el resultado final deseado en vez de mostrar todos los pasos o la secuencia/cadena de instrucciones para lograr el resultado como lo hace la programación imperativa.* Esto hace que sea fácil crear patrones de diseño y crear interfaces de usuario interactivas.
- Es una librería eficiente: *cuando hay un cambio que impacta en el algun elemento del DOM que hace que haya que volver a renderizarlo, React sólo hace el cambio en ese o esos elementos.*
- React trabaja de forma predecible porque la información fluye en una sola vía.
- React trabaja con Componentes que devuelven elementos HTML: *pequeñas piezas de código encapsulado que pueden tener estado o no. Constituyen pequeñas piezas sencillas y fáciles de mantener de uno o más bloques grandes y complejos. Da limpieza al código y lo hace fácil de leer.*



## ¿Qué es un componente en ReactJS?

- *Son un fragmento de la interfaz que cumple una única función.*
- *Son reutilizables ( principio DRY - Don't Repeat Yourself ).*
- *Son independientes, tanto de su contexto como del resto de componentes.*
- *Son autocontenidos, no filtran estilos o funcionalidad a otros componentes.*

## Componente

“

***Es un patrón visual repetido, que se puede resumir en un fragmento independiente de HTML, CSS y posiblemente JavaScript.***

”

- Nicole Sullivan (Product Manager for Frameworks, Google Chrome. Contributes to Chrome, Next.js, & React.)

## Creación de Componente Funcional

```
//Functional Component
export function Header(props) {
  const { title, bgColor } = props
  return (
    <header style={{ backgroundColor: bgColor }} >
      <h1>{title}</h1>
    </header>
  )
}
```

## Renderización de Componente Funcional

```
import { Header } from './components/Header'

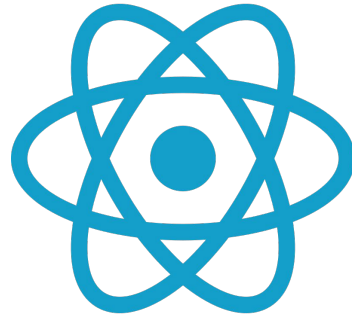
function App() {
  return (
    <main>
      { /* Component instance from components folder */ }
      <Header title='Hello World' bgColor='blue' />
    </main>
  )
}

export default App
```

# ¿Qué es JSX?

- JSX es la forma en como React nos permite implementar programación declarativa(etiquetas XML), programación imperativa y programación funcional(javascript) juntas en nuestro código.
- JSX = Javascript + XML
- **La extensión de un archivo con código JSX debe ser .jsx**

```
//Functional Component
export function Header(props) {
  const { title, bgColor } = props
  return (
    <header style={{ backgroundColor: bgColor }} >
      <h1>{title}</h1>
    </header>
  )
}
```



# Instalación de React en proyecto de Vite

## Instalación de react y react-dom por cli

- Crear proyecto de **vite@latest** con vanilla Javascript
- Instalar paquetes de npm en proyecto (**npm install** en directorio de proyecto)
- Instalar react y react-dom en proyecto (**npm install react react-dom**)

```
npm i react react-dom
```

- Cambiar extension del **main.js** principal creado por Vite a **main.tsx**
- Cambiar en el tag **<script>** del **index.html** del proyecto el **src="main.js"** a **src="main.jsx"**

```
<body>
  <div id="app"></div>
  <script type="module" src="/main.jsx"></script>
</body>
```



## Configuración del Nodo raíz "root" de React

- En **main.jsx** importar React desde la librería de **node\_modules "react"**

```
import React from 'react'
```

- Importar la función **createRoot** desde la librería de **node\_modules "react-dom/client"**

```
import { createRoot } from 'react-dom/client'
```

- Obtener el elemento del DOM con el **id="app"** creado por vite para usarlo como nodo de raíz de la aplicación. En elemento se renderizarán todos los componentes de la aplicación.

```
const rootElement = document.getElementById('app')
```

- Creamos la raíz de la app usando el elemento raíz del DOM con la función **createRoot()** pasándole como parámetro el elemento raíz que obtuvimos

```
const root = createRoot(rootElement)
```

- createRoot()** nos devuelve un objeto raíz, al cuál le podemos llamar el método **render()** que puede recibir como parámetro un nodo del DOM o un componente de React

```
root.render(<h1>Hello World</h1>)
```

```
import React from 'react'
import { createRoot } from 'react-dom/client'

const rootElement = document.getElementById('app')
const root = createRoot(rootElement)
root.render(<h1>Hola Mundo</h1>)
```


<https://react.dev/reference/react-dom/client/createRoot>

## Al Hacer code splitting

! Instalar el paquete de npm **@vitejs/plugin-react en cli** para evitar la constante importación de React en cada archivo y **configurar el vite.config.js** (ver documentación: <https://www.npmjs.com/package/@vitejs/plugin-react>)

```
npm i @vitejs/plugin-react
```

## Instalación de Extensiones útiles en VS Code




### Auto Close Tag

20ms

Automatically add HTML/XML close t...

Jun Han




### Auto Rename Tag

15ms

Auto rename paired HTML/XML tag

Jun Han




## ES7+ React/Redux/React-I

dsznajder | 8,720,821 | ★★★★★ (72)

Extensions for React, React-Native and Redux ...

Disable | Uninstall | ⚙️

This extension is enabled globally.



### Better Comments


v3.0.2

Aaron Bond [aaronbond.co.uk](https://aaronbond.co.uk) | 5,294,552 | ★

Improve your code commenting by annotating with alert

Disable | Uninstall | ⚙️

This extension is enabled globally.



### ESLint

v2.4.2

Microsoft [microsoft.com](https://microsoft.com) | 28,774,643 | ★★★

Integrates ESLint JavaScript into VS Code.

Disable | Uninstall | ⚙️

This extension is enabled globally.