
ICARO
Nuestro compromiso es con el futuro.

React

Clase 2

React: Entorno Productivo

Conceptos básicos:

- **Intro (Vite)**
- **Instalación**
- **JSX**
- **Hooks (useState)**

React

Entorno Productivo: Intro

Entorno Productivo: Intro

Todo entorno de desarrollo de React consta de cuatro pilares:

Entorno Productivo: Intro

Todo entorno de desarrollo de React consta de cuatro pilares:

- Bundler
- Transpiler
- Live Module Reload
- Librerías de React

Entorno Productivo: Intro

Todo entorno de desarrollo de React consta de cuatro pilares:

- Bundler (empaquetador)
- Transpiler (transpilador)
- Live Module Reload (recarga modular)
- Librerías de React (react, react-dom)

Entorno Productivo: Intro

- *En este curso utilizaremos vite.*

Entorno Productivo: Intro

- *En este curso utilizaremos vite.*

- *Por medio de un único comando en la terminal Vite nos permite, de una manera sencilla, comenzar a desarrollar cualquier proyecto de React.*

Entorno Productivo: Intro

- *Hoy en día existen muchas formas de iniciar un proyecto de React. Recientemente CRA (create-react-app) dejó de ser la forma recomendada en la documentación oficial.*

Entorno Productivo: Intro

- *Hoy en día existen muchas formas de iniciar un proyecto de React. Recientemente CRA (create-react-app) dejó de ser la forma recomendada en la documentación oficial.*

- *Como alternativas existen Parcel, Next.js, Remix y Gatsby, sólo por nombrar algunas.*

Entorno Productivo: Intro

Vite

- ESBuild
- Babel
- Hot Module Replacement
- react y react-dom

Entorno Productivo: Intro

Vite

- ESBuild (bundler)
- Babel (transpiler)
- Hot Module Replacement (recarga modular)
- react y react-dom (librerías react)

Entorno Productivo: Intro

Pilares: ESBuild

Entorno Productivo: Intro

Pilares: ESBuild

ESBuild es el bundler utilizado por Vite, su principal ventaja es la gran velocidad que posee.

Webpack sigue siendo el más popular de los bundlers en la actualidad.

Los bundlers son vistos como “empaquetadores” que toman el código fuente (nuestro código) y lo optimiza para ser utilizado por distintos ambientes.

Entorno Productivo: Intro

Pilares: Babel

Entorno Productivo: Intro

Pilares: Babel

Babel es un transpilador de código (traductor, compilador), se encarga de convertir el código fuente en código entendible para el navegador.

**En React es vital para poder utilizar JSX.*

Entorno Productivo: Intro

Pilares: Hot Module Replacement

Entorno Productivo: Intro

Pilares: Hot Module Replacement

React utiliza el module-pattern (patrón modular). En el virtual DOM viven todos los estados de nuestra aplicación.

A la hora de escribir código en React el HMR aplica los cambios cuando guardamos sin necesidad de “bajar” y “levantar nuevamente” el proyecto.

Entorno Productivo: Intro

Pilares: react y react-dom

Entorno Productivo: Intro

Pilares: react y react-dom

Son las librerías oficiales de React. Tienen versiones idénticas y funcionan en conjunto.

Previamente habíamos utilizado la versión on-line de ellas.

Entorno Productivo: Instalación

Vite: comandos

npm:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  npm create vite@latest
```

yarn:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  yarn create vite
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...


success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
? Project name: » vite-project
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...


success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
? Project name: » ejemplo-react
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...


success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
√ Project name: » ejemplo-react
? Select a framework: » - Use arrow-keys. Return to submit.
>   Vanilla
    Vue
    React
    Preact
    Lit
    Svelte
    Others
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...

success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
√ Project name: » ejemplo-react
? Select a framework: » - Use arrow-keys. Return to submit.
    Vanilla
    Vue
>   React
    Preact
    Lit
    Svelte
    Others
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...

success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
√ Project name: » ejemplo-react
√ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
>   JavaScript
    TypeScript
    JavaScript + SWC
    TypeScript + SWC
```

Entorno Productivo: Instalación

Vite: terminal

```
[4/4] Building fresh packages...

success Installed "create-vite@4.2.0" with binaries:
      - create-vite
      - cva
√ Project name: » ejemplo-react
√ Select a framework: » React
√ Select a variant: » JavaScript
Scaffolding project in C:\Users\alumno\Documents\ejemplo-react...
Done. Now run:
  cd ejemplo-react
  yarn
  yarn dev
Done in 735.32s.
```

Entorno Productivo: Instalación

Vite: comandos

npm:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  cd ejemplo-react
$  npm install
$  npm run dev
```

yarn:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  cd ejemplo-react
$  yarn
$  yarn dev
```

Entorno Productivo: Instalación

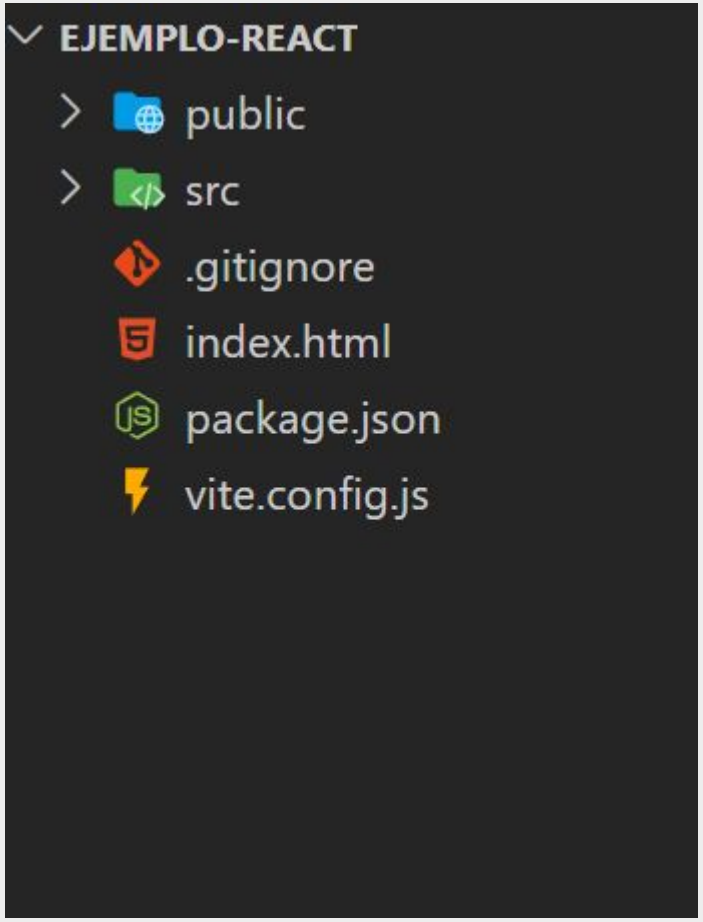
Vite: comandos

```
VITE v4.1.1  ready in 3680 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h to show help
```

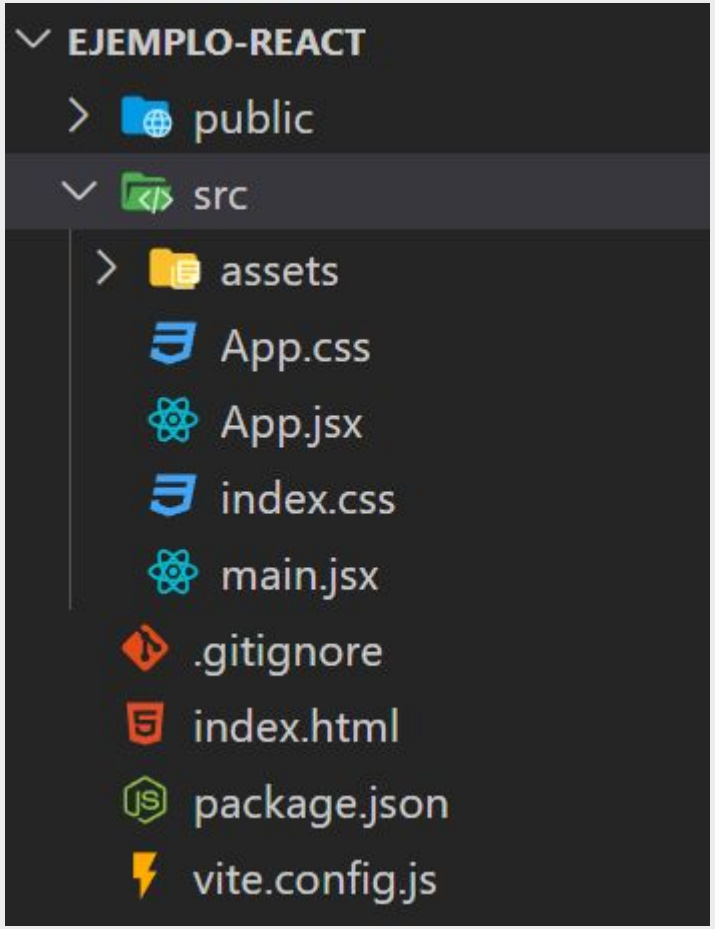
Entorno Productivo: Instalación

Vite: ejemplo carpetas



Entorno Productivo: Instalación

Vite: ejemplo carpetas



Entorno Productivo: Instalación

- *Hasta aquí hemos creado un proyecto desde cero.*

Entorno Productivo: Instalación

- *Hasta aquí hemos creado un proyecto desde cero.*

- *Dentro del archivo App.jsx podremos empezar a crear nuestros componentes*

Entorno Productivo: Instalación

- *Hasta aquí hemos creado un proyecto desde cero.*
- *Dentro del archivo App.jsx podremos empezar a crear nuestros componentes*
- *Se recomienda investigar “que hace cada cosa” dentro de ese archivo, pero luego borrar todo el contenido de la función App.*

React

Entorno Productivo: JSX

Entorno Productivo: JSX

- *JSX es una transpilación (traducción) realizada por Babel.*

Entorno Productivo: JSX

- *JSX es una transpilación (traducción) realizada por Babel.*

- *Esta transpilación nos permite escribir código de marcado (HTML) que será convertido a JavaScript.*

Entorno Productivo: JSX

- *JSX es una transpilación (traducción) realizada por Babel.*
- *Esta transpilación nos permite escribir código de marcado (HTML) que será convertido a JavaScript.*
- *Los archivos requieren ser renombrados con la extensión .jsx*

Entorno Productivo: JSX

No JSX

- Sintaxis de JavaScript
- No utiliza Babel
- Archivos con extensión .js
- Poca legibilidad

Entorno Productivo: JSX

No JSX: App.js

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     return React.createElement("div", {}, [
6.       React.createElement("h1", {}, "Hola mundo"),
7.     ])
8.  }
9.
10. export default App
```

Entorno Productivo: JSX

No JSX: App.js

```
1.  import React from "react" // Babel utilizará esta línea para transpilar con el método ".createElement()".
2.  import "./App.css"
3.
4.  function App() {
5.     return React.createElement("div", {}, [
6.       React.createElement("h1", {}, "Hola mundo"),
7.     ])
8.  }
9.
10. export default App
```

Entorno Productivo: JSX

No JSX: App.js

```
1.  import React from "react"
2.  import "./App.css" // Con esta importación podemos agregar el archivo CSS que necesitemos.
3.
4.  function App() {
5.     return React.createElement("div", {}, [
6.       React.createElement("h1", {}, "Hola mundo"),
7.     ])
8.  }
9.
10. export default App
```

Entorno Productivo: JSX

No JSX: App.js

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     return React.createElement("div", {}, [ // Creamos un elemento <div> con un hijo <h1>.
6.       React.createElement("h1", {}, "Hola mundo"),
7.    ])
8.  }
9.
10. export default App
```

Entorno Productivo: JSX

No JSX: App.js

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     return React.createElement("div", {}, [
6.       React.createElement("h1", {}, "Hola mundo"), // Creamos un elemento <h1> con texto como hijo.
7.    ])
8.  }
9.
10. export default App
```

Entorno Productivo: JSX

JSX

- Sintaxis Similar a HTML
- Utiliza Babel
- Archivos con extensión .jsx
- Mayor legibilidad

Entorno Productivo: JSX

Ejemplo JSX: App.jsx

```jsx
import React from "react"
import "./App.css"

function App() {
   return (
     <div>
       <h1>Hola Mundo</h1>
     </div>
   )
}

export default App
```

Entorno Productivo: JSX

Ejemplo JSX: App.jsx

```
1.  import React from "react" // Como habíamos dicho antes, Babel requiere obligatoriamente esta línea.
2.  import "./App.css"
3.
4.  function App() {
5.     return (
6.       <div>
7.         <h1>Hola Mundo</h1>
8.       </div>
9.     )
10. }
11.
12. export default App
```

Entorno Productivo: JSX

Ejemplo JSX: App.jsx

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     return (
6.       <div> // Creamos un elemento <div> tal y como en HTML
7.         <h1>Hola Mundo</h1>
8.       </div>
9.     )
10. }
11.
12. export default App
```

Entorno Productivo: JSX

Ejemplo JSX: App.jsx

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     return (
6.       <div>
7.         <h1>Hola Mundo</h1> // Creamos un elemento <h1> tal y como en HTML
8.       </div>
9.     )
10. }
11.
12. export default App
```

Entorno Productivo: JSX

- *En JSX podemos introducir código de JS utilizando llaves “{}”.*

Entorno Productivo: JSX

- *En JSX podemos introducir código de JS utilizando llaves “{}”.*

- *Todo lo que pongamos dentro de las llaves tiene ser algo que podamos poner después de un operador de asignación (igual ó “=“).*

Entorno Productivo: JSX

Ejemplo llaves

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     const nombre = "Juan"
6.
7.     return (
8.       <div>
9.         <h1>Hola {nombre}</h1>
10.      </div>
11.    )
12. }
13.
14. export default App
```

Entorno Productivo: JSX

Ejemplo llaves

```
1.  import React from "react"
2.  import "./App.css"
3.
4.  function App() {
5.     const nombre = "Juan"
6.
7.     return (
8.       <div>
9.         <h1>Hola {nombre}</h1> // Introducimos código de JS por medio de las llaves.
10.      </div>
11.    )
12. }
13.
14. export default App
```

React

Entorno Productivo: Hooks (useState)

Entorno Productivo: Hooks (useState)

- *Los hooks son una pieza fundamental de React.*

Entorno Productivo: Hooks (useState)

- *Los hooks son una pieza fundamental de React.*
- *El concepto está fuertemente inspirado en la programación funcional.*

Entorno Productivo: Hooks (useState)

- *Los hooks son una pieza fundamental de React.*
- *El concepto está fuertemente inspirado en la programación funcional.*
- *Solamente pueden ser utilizados en componentes funcionales.*

Entorno Productivo: Hooks (useState)

Veremos los hooks en profundidad en futuras clases, hoy simplemente nos limitaremos a la utilización de uno de ellos.

“useState()”

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react"
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("")
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1>
10.       <br />
11.
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react" // Importamos el hook "useState()".
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("")
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1>
10.       <br />
11.
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react"
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("") // En este caso utilizaremos "useState" para guardar un string.
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1>
10.       <br />
11.
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react"
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("")
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1> // Aquí utilizaremos llaves para poder insetar variables de JS.
10.       <br />
11.
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react"
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("")
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1>
10.       <br />
11.       // En react tenemos el atributo "onChange" el cual funciona exactamente igual al de vanilla JS.
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo: Hooks (useState)

useState

```
1.  import React, { useState } from "react"
2.  import "./App.css"
3.
4.  function App() {
5.    const [nombre, setNombre] = useState("")
6.
7.   return (
8.      <div>
9.        <h1>Hola {nombre}</h1>
10.       <br />
11.       // setNombre actualizará la variable nombre reactivamente (a medida que cambie)
12.       <input type="text" onChange={e => setNombre(e.target.value)} />
13.     </div>
14.   )
15. }
16. export default App
```

Entorno Productivo

Antes de ir al código…

A modo de práctica podemos recrear los ejemplos de la clase anterior.

¡Vamos al código!

Clase 2: Entorno Productivo

¡Muchas gracias!

ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar