Clase 2: Repaso ReactJS

Sitio: Centro de E-Learning - UTN.BA
Curso: Curso de Backend Developer - Turno Noche
Libro: Clase 2: Repaso ReactJS
Imprimido por: Nilo Crespi
Día: Friday, 23 de January de 2026, 10:13

Tabla de contenidos

1. Repaso ReactJS

En esta clase, daremos un paso más en nuestro aprendizaje de JavaScript explorando herramientas fundamentales para el desarrollo de aplicaciones modernas. Comenzaremos revisando qué son las librerías y cómo facilitan nuestro trabajo al proporcionar métodos y funcionalidades listas para usar.

Nos centraremos en **React**, una de las librerías más utilizadas para la construcción de interfaces interactivas, y en **Node.js**, el entorno que permite ejecutar JavaScript fuera del navegador. También veremos **NPM**, el gestor de paquetes de JavaScript, y cómo nos ayuda a instalar y gestionar dependencias en nuestros proyectos.

Además, aprenderemos a crear un proyecto con **Vite**, una herramienta que nos permite configurar aplicaciones de React de manera rápida y eficiente. Exploraremos la estructura de archivos de una aplicación en React y nos introduciremos en **JSX**, la sintaxis que facilita la integración de HTML dentro de JavaScript.

Al finalizar, comprenderemos las diferencias clave entre JSX y HTML, cómo aplicar estilos en JSX y cómo hacer referencia a variables dentro del código. ¡Manos a la obra para comenzar con la práctica!

2. Introducción

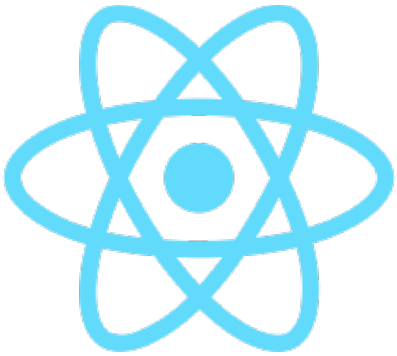
¿Qué es una librería?

Las librerías (o bibliotecas) JavaScript son un **conjunto de métodos** que facilitan el trabajo.

Algunos ejemplos: jQuery, Chart.js, etc.

¿Qué es React?

Es una **librería** Javascript de código abierto diseñada para crear **interfaces de usuario** con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre.



Concepto básicos

- **Usa un virtual DOM en vez del DOM real.**
- **Basado en componentes. Con estados y ciclos de vida.**
- **Se puede trabajar con React del lado del servidor con Node**
- **Se pueden crear aplicaciones móviles con React Native.**

3. NodeJS

Node.js es un entorno de ejecución de javascript que le **permite al código JS** ser ejecutado en nuestra computadora.

NPM

Ya que Node ser un sistema modular viene prácticamente "vacío". Por lo tanto, para utilizar una funcionalidad de alguna librería publicada, deberemos instalar módulos adicionales.
Esta operación se realiza de forma muy sencilla con la herramienta npm (Node Package Manager).

Entonces, NPM es un gestor de paquetes JavaScript, que nos va a permitir instalar y desinstalar paquetes, gestionar versiones y gestionar dependencias necesarias para ejecutar un proyecto desde la línea de comandos.

Instalación de Node.js

1. Vamos a ingresar a https://nodejs.org/en/
2. Descargamos la última versión de Node
3. Ejecutamos el archivo de instalación y seguimos los pasos

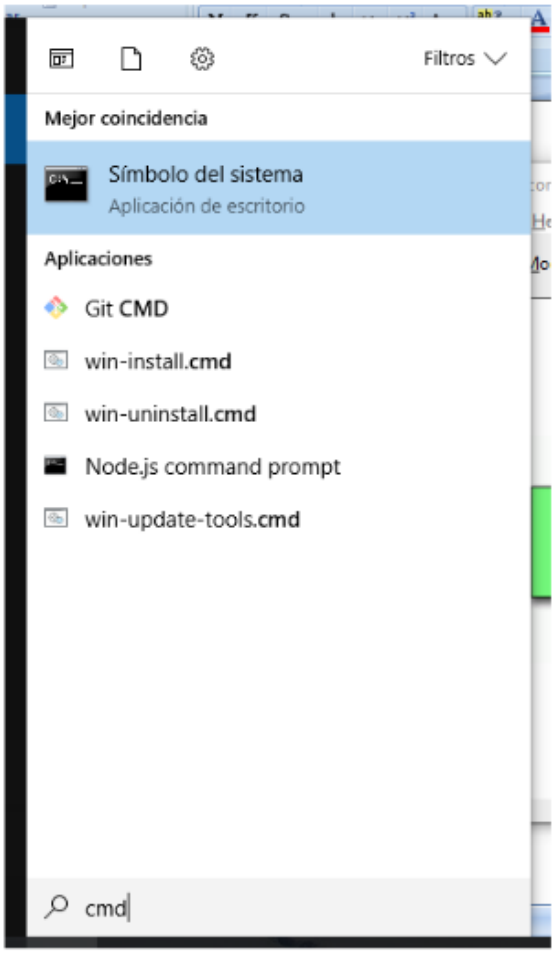
4. CLI

CLI

La interfaz de línea de comandos es un programa que nos permite dar instrucciones a algún programa informático por medio de una línea de texto simple.
En Windows podemos usar Command Prompt (símbolo del sistema) o Windows PowerShell.

Abrir la terminal / MacOS

Abrir la terminal / Windows


Filtros
Mejor coincidencia
Símbolo del sistema
Aplicación de escritorio
Aplicaciones
Git CMD
win-install.cmd
win-uninstall.cmd
Node.js command prompt
win-update-tools.cmd
cmd

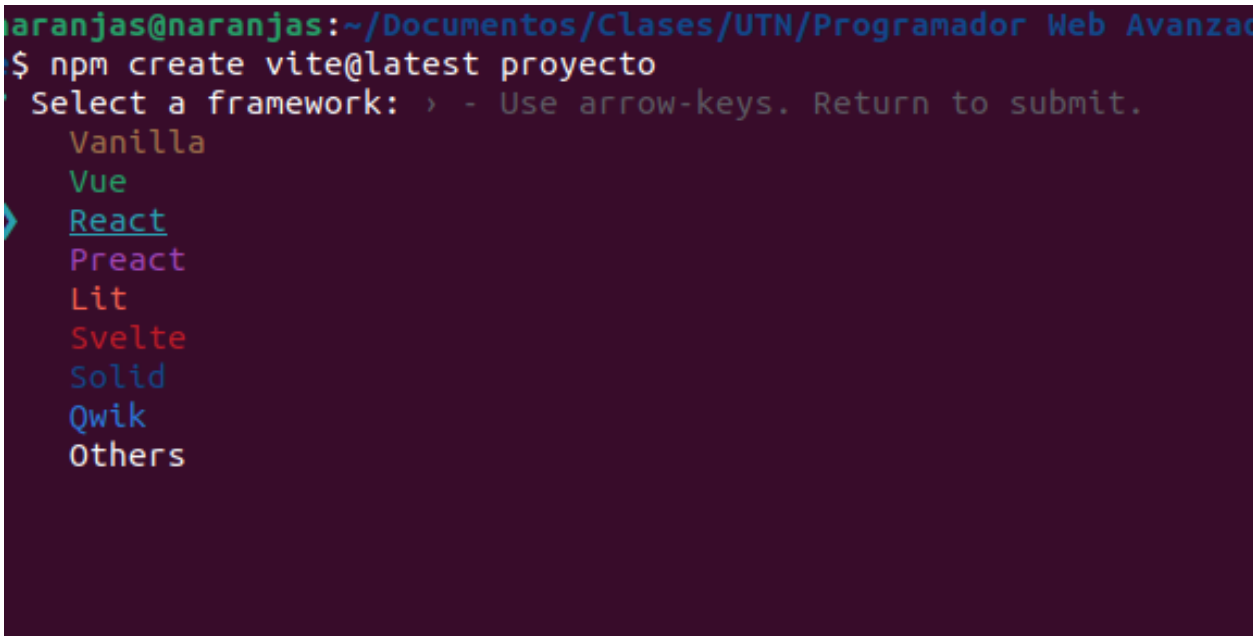
5. Crear Proyecto con vite

Instalación con vite

1. **Verifica que Node.js esté instalado: node -v**
2. **Crea un proyecto de React con Vite: npm create vite@latest mi-proyecto**

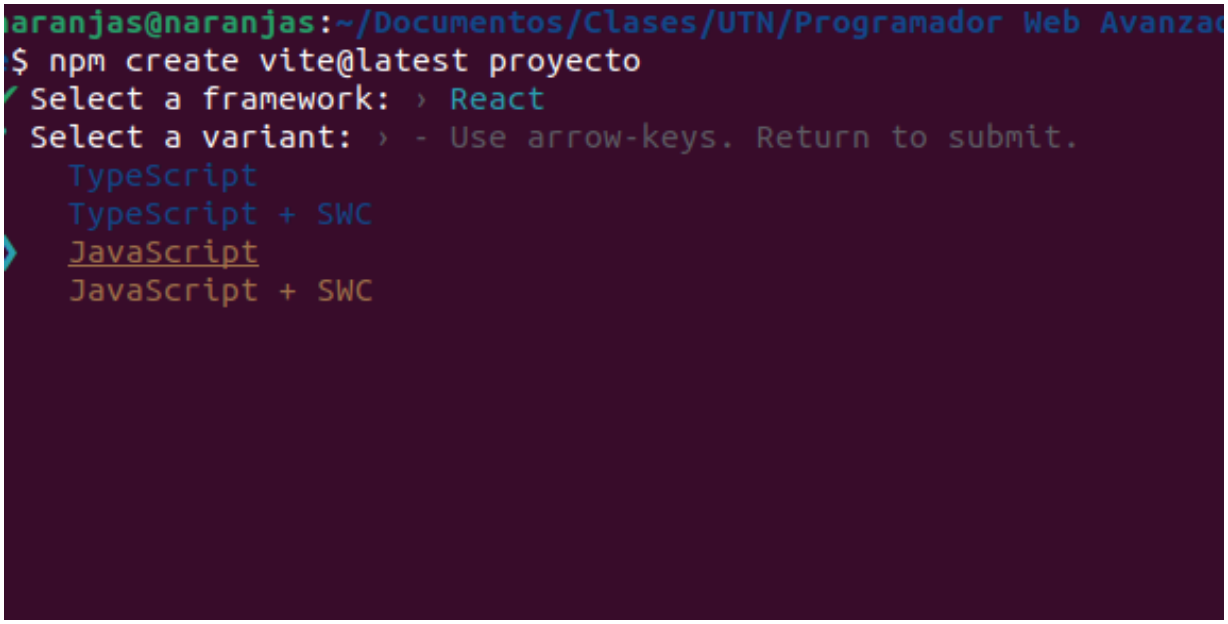
    cd mi-proyecto
    npm install //instalamos dependencias
    npm run dev // iniciamos servidor

Seleccionar React



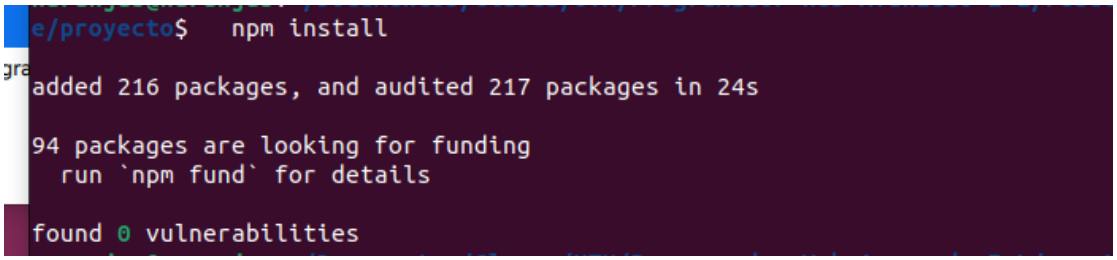
Seleccionar variante

En nuestro caso Javascript o Javascrip + SWC



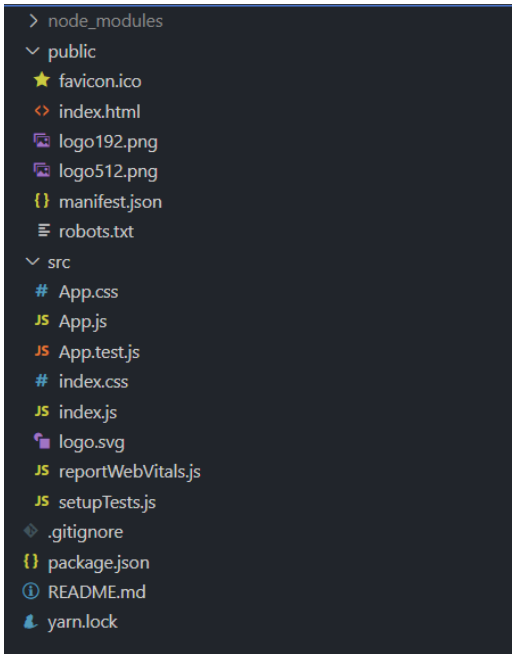
Ejecutar los comandos de dependencias e inicio del servidor

Listos para comenzar



Repaso por la app

Archivos



Vemos que la app se compone por varios archivos base…
Vamos a ir viendo en nuestra carpeta qué son éstos archivos y cuáles vamos a utilizar.

6. JSX

JSX frente a HTML

JSX (JavaScript XML) es una extensión a JS que nos permite escribir HTML dentro de nuestro código sin necesidad de representarlo como un string (encerrándolo entre comillas).

```
const element = <h1>Hello, world!</h1>;
```

- Es muy similar a HTML, lo que aporta facilidad de escritura y comprensión.

Entonces...

- **Se parece a HTML. No es ni una cadena de caracteres, ni HTML.**
- **Es una extensión que permite hacer llamadas a funciones y a construcción de objetos.**
- **JSX es una extensión de Javascript, no de React.**
- **Esto significa que no hay obligación de utilizarlo, pero es recomendado en la documentación oficial de React.**
- **Al final, JSX se transforma en código JavaScript.**

Esto nos da algunas ventajas como:

- **Ver errores en tiempo de compilación**
- **Asignar variables**
- **Utilizar métodos**

Algunas diferencias

- El estilo CSS en línea en JSX tiene una sintaxis diferente.
- El modo nombrar una clase en las etiquetas HTML es distinto.
- JSX puede hacer referencia a variables de JavaScript.

CSS en línea en JSX

HTML → style="background-color:cyan;color:blue"

JSX → style={{backgroundColor:'cyan',color:'blue'}}

1. En JSX, en lugar de envolver el estilo entre comillas dobles, usamos llaves dobles, style = {{…}}, la llave exterior indica que vamos a colocar una variable de JavaScript en esta y las segundas llaves indican un objeto JavaScript.
2. El nombre de la propiedad tiene un guión, tenemos que eliminar ese guión y cambiarlo al formato de camelCase comenzando con minúsculas, es decir, en lugar de background-color usamos backgroundColor.
3. El valor de la propiedad se ajustará como una cadena entre comillas simples o dobles.

4. Las diferentes propiedades se separan con una coma en lugar del punto y coma.

Nombre de clase en JSX

HTML → class="unameinput"

JSX → className="unameinput"

En JavaScript moderno, la palabra clave de clase está reservada para clases de JavaScript y para evitar esa colisión usamos className para dar un nombre de clase en etiquetas HTML en JSX en lugar de class.

className

```
class App extends React.Component{
    render(){
        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>Login</button>
            </div>
        );
    }
}
```

Referencia a las variables de JavaScript de JSX

JSX puede hacer referencia fácilmente a variables de JavaScript, es decir, podemos usar variables o funciones de JavaScript en lugar de texto sin formato colocándolas, las primeras llaves indican la variable de JavaScript.

<button>{buttonName}</button>

o

<button>{getButtonName()}</button>.

Si queremos reemplazar el texto del botón con la función JavaScript.

```
class App extends React.Component{
    render(){

        const buttonName="Log in";

        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>{buttonName}</button>
            </div>
        );
    }
}
```

Usando la variable JS en lugar de texto sin formato para el botón.

Si queremos reemplazar el texto del botón con la función JavaScript.

```
class App extends React.Component{
    render(){

        const getButtonname=()=>{
            return "Log in";
        }

        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>{getButtonname()}</button>
            </div>
        );
    }
}
```

Usando la función JS en lugar de texto sin formato para el botón.

7. Actividad Sugerida

Realizá la instalación de React con Vite, creá una carpeta específica para los componentes y exhibilos a través del archivo app.jsx.

Aplicá estilos personalizados a cada componente para mejorar su presentación. ¡Éxitos!