1. 🛠️ Procedimiento de Instalación de Node.js

Node.js es el entorno de ejecución necesario para ejecutar JavaScript fuera del navegador (y, por lo tanto, es esencial para Playwright).

A. Descarga del Instalador

1. Ve al sitio web oficial de **Node.js** (nodejs.org).
2. Verás dos opciones principales para la descarga:
   - **LTS (Long Term Support):** Es la versión **recomendada** para la mayoría de los usuarios y entornos de producción, ya que es la más estable y recibe soporte a largo plazo.
   - **Current:** Incluye las últimas características, pero puede ser menos estable.
3. Descarga el instalador que corresponda a tu sistema operativo (Windows, macOS, o Linux).

B. Ejecución del Instalador

- **Windows/macOS:** Ejecuta el archivo descargado y sigue las instrucciones del asistente. Generalmente, puedes aceptar todas las opciones predeterminadas (Next, Next, Install).El instalador incluye automáticamente el administrador de paquetes **npm (Node Package Manager)**.
- **Linux (Opcional):** Para usuarios avanzados, se recomienda usar un gestor de versiones como **nvm (Node Version Manager)** para instalar y cambiar fácilmente entre diferentes versiones de Node.js.

C. Verificación de la Instalación

Abre tu terminal o línea de comandos (Command Prompt, PowerShell, o Terminal) y ejecuta los siguientes comandos:

- Para verificar la versión de Node.js:
  Bash
  `node -v`

- Para verificar la versión de npm:
  Bash
  `npm -v`

Si ambos comandos muestran un número de versión, la instalación fue exitosa.

---

2. 📁 Creación de un Proyecto Node.js

Crear un proyecto implica establecer una carpeta y un archivo de configuración para gestionar las dependencias y la información del proyecto.

A. Inicializar el Proyecto

1. Crea una nueva carpeta para tu proyecto y navega hacia ella en la terminal:
   Bash
   ```
   mkdir mi-proyecto-playwright
   cd mi-proyecto-playwright
   ```

2. Ejecuta el comando de inicialización de npm:
   Bash
   ```
   npm init -y
   ```

   El *flag* -y acepta todas las opciones predeterminadas. Esto crea el archivo **package.json** en la raíz de tu proyecto.

B. El Archivo package.json

Este archivo es el **manifiesto** de tu proyecto. Contiene metadatos (nombre, versión, descripción) y una lista de todas las **dependencias** (librerías externas como Playwright) que tu proyecto necesita para funcionar.

C. Estructura Básica del Proyecto

Crea un archivo principal (ej. index.js o main.js) donde escribirás el código.

---

3. 🧩 Requisitos para la Definición e Importación de Módulos

Para estructurar tu código de manera organizada y reutilizable, usarás el sistema de módulos de JavaScript. Hay dos principales en Node.js, y es importante saber cuál estás usando.

A. CommonJS (CJS)

Este fue el sistema original de módulos de Node.js.

Acción	Sintaxis (CommonJS)	Requisitos en package.json
Definición/Exportación	Utiliza module.exports para exportar variables, funciones, o clases.	Ninguno. Es el comportamiento predeterminado si type no está definido.
Importación/Uso	Utiliza la función require() para cargar el módulo.	

Ejemplo (CommonJS)

Archivo: utilidad.js (Exportación)

JavaScript

```
function multiplicar(a, b) {
  return a * b;
}

// Exportar la función
module.exports = {
  multiplicar
};
```

Archivo: index.js (Importación)

JavaScript

```
// Importar la función usando require()
const { multiplicar } = require('./utilidad');

console.log(multiplicar(5, 4)); // Salida: 20
```

B. ES Modules (ESM)

Este es el estándar moderno de JavaScript (introducido en ES6) y el preferido para proyectos nuevos, especialmente para herramientas como Playwright.

Acción	**Sintaxis (ES Modules)**	**Requisitos en package.json**
Definición/Exportación	Utiliza la palabra clave export.	Debes añadir "type": "module", al archivo package.json.
Importación/Uso	Utiliza la palabra clave import.	

Requisito Clave para ES Modules

Asegúrate de que tu archivo package.json contenga esta línea para habilitar la sintaxis moderna de import/export:

JSON

```
{
  "name": "mi-proyecto",
  "version": "1.0.0",
  "type": "module",  // <-- ¡Este es el requisito!
  ...
}
```

Ejemplo (ES Modules)

Archivo: utilidad.js (Exportación)

JavaScript

```
// Exportar la función
export function dividir(a, b) {
  return a / b;
}
```

Archivo: index.js (Importación)

JavaScript

```
// Importar la función usando import
import { dividir } from './utilidad.js'; // Nota: se recomienda incluir la extensión .js

console.log(dividir(10, 2)); // Salida: 5
```