TALENTO TECH VALLE **ACTIVIDAD: NODE.JS** **103 – 104** **CLASE 16**	

1. Inicializar el proyecto con `npm init`

Este comando configura el archivo `package.json`, que es esencial para cualquier proyecto en Node.js.

Opción 1: Proceso interactivo

Ejecuta:

```bash
npm init
```

- Te hará preguntas como:
  - Nombre del proyecto.
  - Versión inicial (por defecto es `1.0.0`).
  - Descripción del proyecto.
  - Entrada principal (por defecto `index.js`).
  - Autor.
  - Licencia (por defecto `ISC`).
- Al completar las preguntas, generará un archivo `package.json` con los datos que proporcionaste.

Opción 2: Inicialización rápida

Si no quieres responder preguntas, usa la bandera `-y` para aceptar los valores predeterminados:

```bash
npm init -y
```

Esto generará un archivo `package.json` con valores básicos.

---

2. Crear el archivo principal (por defecto `index.js`)

1. Dentro del directorio del proyecto, crea el archivo principal:

```bash
touch index.js
```

   O bien, usa un editor de texto (como VS Code) para crear el archivo.

2. Abre `index.js` e incluye un simple mensaje para probar:

```javascript
console.log('¡Hola, mundo desde Node.js!');
```

---

3. Ejecutar el archivo con Node.js

Para verificar que tu proyecto funciona:

1. En la terminal, ejecuta el archivo principal:

```bash
node index.js
```

2. Deberías ver en la consola:

```
¡Hola, mundo desde Node.js!
```

---

4. Agregar dependencias al proyecto

Si necesitas librerías o módulos adicionales:

- Para instalar una dependencia, usa:

```bash
npm install nombre-de-la-libreria
```

Esto agrega la biblioteca al proyecto y la registra en el archivo `package.json`.

Por ejemplo, si quieres instalar `express`:

```bash
npm install express
```

Esto crea una carpeta `node_modules` y actualiza el archivo `package.json` con una entrada en `"dependencies"`.

---

5. Ignorar carpetas innecesarias

Crea un archivo `.gitignore` si usas Git, para ignorar carpetas como `node_modules y .evn` al hacer un commit:

TALENTO TECH VALLE **ACTIVIDAD: NODE.JS** **103 – 104** **CLASE 16**	

---

6. Ejecutar scripts personalizados

En el archivo `package.json`, puedes definir scripts personalizados. Por ejemplo:

```json
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

- Luego, ejecuta los scripts con:

```bash
npm run start
npm run dev
```

---

7. Instalar herramientas útiles para desarrollo

Algunas herramientas comunes:

- **Nodemon:** Reinicia automáticamente tu aplicación cuando detecta cambios en los archivos.

```bash
npm install --save-dev nodemon
```

Luego, puedes usarlo en tus scripts:

```json
"scripts": {
  "dev": "nodemon index.js"
}
```

Estructura final del proyecto

Después de estos pasos, tu proyecto debería tener esta estructura básica:

```go
nombre-del-proyecto/
├── node_modules/
├── package.json
├── package-lock.json
├── .gitignore
└── index.js
```