# **Ejercicio 1**

### **Considera el lenguaje TypeScript acotado al paradigma de programación estructurada y analízalo en términos de** [**los cuatro componentes de un paradigma**](https://www.notion.so/f36d432c55274b93913dc289446f424d?pvs=21) **mencionados por Kuhn.**

### **1. Generalizaciones simbólicas**

**¿Cuáles son las reglas escritas del lenguaje TypeScript dentro de la programación estructurada?**

* **Sintaxis y semántica bien definida:** TypeScript extiende JavaScript agregando un sistema de tipos estáticos opcional, pero sigue respetando las estructuras propias del paradigma estructurado: secuencia, selección (condicionales) y repetición (bucles).
* **Control explícito del flujo:** El orden de ejecución se establece por estructuras como condicionales y bucles, no por invocaciones automáticas ni por delegación como en otros paradigmas (por ejemplo, el funcional).
* **Tipado estático opcional:** Aunque opcional, el tipado se convierte en parte de las reglas formales cuando se adopta TypeScript, permitiendo mayor control en tiempo de desarrollo.
* **Modularidad mediante funciones:** Las funciones se utilizan para dividir el programa en partes reutilizables y comprensibles.

### **2. Creencias de los profesionales**

**¿Qué se valora como "mejor" en TypeScript desde la comunidad que lo usa bajo un enfoque estructurado?**

* **Mayor seguridad y claridad que JavaScript:** Muchos desarrolladores creen que el tipado estático ayuda a evitar errores comunes y mejora la legibilidad.
* **Compatibilidad total con JavaScript:** Se valora la capacidad de usar todo el ecosistema JS sin perder las ventajas del control estructurado y del tipado.
* **Productividad mejorada en proyectos grandes:** Se cree que el uso de TypeScript facilita el mantenimiento de proyectos complejos gracias a sus herramientas de verificación en tiempo de compilación.
* **Buena integración con herramientas modernas:** La comunidad valora cómo TypeScript mejora el trabajo con editores, entornos de desarrollo y linters, sin sacrificar la estructura clásica del código imperativo.

### **3. Valores del grupo**

**¿Qué se considera importante en el uso del lenguaje en este paradigma?**

* **Claridad en el flujo del programa:** Se valora una estructura secuencial comprensible, evitando la ambigüedad y favoreciendo el paso a paso del procesamiento.
* **Control total sobre el estado:** Se prioriza el manejo explícito del estado y las variables, en contraposición al enfoque más abstracto de otros paradigmas como el funcional.
* **Eficiencia en la ejecución:** Aunque TypeScript se transpila a JavaScript, la estructuración del código puede contribuir a un mejor rendimiento lógico y mantenibilidad.
* **Facilidad para depurar y mantener:** La comunidad estructurada tiende a valorar que el código sea predecible y fácil de seguir.

### **4. Ejemplares**

**¿Qué tipos de problemas se resuelven y cómo se abordan en este paradigma con TypeScript?**

* **Lógica de negocio lineal:** Casos donde el flujo del programa puede representarse como una serie de pasos ordenados, como formularios, validaciones, flujos de usuario.
* **Aplicaciones web con estructura clara:** Uso en scripts front-end donde cada acción sigue una lógica estructurada con condicionales y bucles.
* **Automatización y scripts pequeños:** Donde se requiere control total sobre el orden y condiciones del procesamiento.
* **Programas educativos o introductorios:** Donde la claridad estructural es ideal para enseñar los fundamentos de programación.