Ejercicio 1

Parte A

Considera el lenguaje JavaScript acotado al paradigma de programación estructurada y analízalo en términos de los cuatro componentes de un paradigma mencionados por Kuhn.

1. Generalización simbólica: ¿Cuáles son las reglas escritas del lenguaje?
2. Creencias de los profesionales: ¿Qué características particulares del lenguaje se cree que sean "mejores" que en otros lenguajes?

1. **Generalización simbólica:** Son la reglas escritas que definen el lenguaje en este paradigma:
    a. Uso de estructuras de control clásicas: if-else, switch, for, while, do-while.
    b. Declaración de variables con var, let o const.
    c. Las sentencias deben terminar (opcionalmente) con “;”.
    d. Expresiones aritméticas, lógicas y relacionales.
    e. Estructuración del programa en bloques delimitados por { }.
    f. Ejecución secuencial como regla base: línea por línea, de arriba hacia abajo.
2. **Creencias de los profesionales:** Lo que se considera “mejor” de JavaScript:
    a. Simplicidad para ejecutar en navegadores SIN necesidad de compilación -> corre directamente en cualquier browser.
    b. Tipado dinámico -> flexibilidad para programar rápido sin declarar tipos de forma estricta.
    c. Portabilidad total: cualquier sistema con un navegador puede ejecutar el código.
    d. Rapidez para prototipar scripts sencillos en comparación con C o Java.
    e. Comunidad enorme, con abundancia de ejemplos, foros y documentación.

Parte B

Considera el lenguaje JavaScript acotado al paradigma de programación estructurada y analízalo en términos de los ejes propuestos para la elección de un lenguaje de programación (¿Cómo elegir un lenguaje?) y responde:

1. ¿Tiene una sintaxis y una semántica bien definida? ¿Existe documentación oficial?
2. ¿Es posible comprobar el código producido en ese lenguaje?
3. ¿Es confiable?
4. ¿Es ortogonal?
5. ¿Cuáles son sus características de consistencia y uniformidad?
6. ¿Es extensible? ¿Hay subconjuntos de ese lenguaje?
7. El código producido, ¿es transportable?

JavaScript acotado al paradigma de programación estructurada:

1. Sí, la sintaxis y semántica están estandarizadas por **ECMAScript (ES)**. La documentación oficial se encuentra en los estándares de ECMA y en recursos como MDN Web Docs.
2. Sí pero con algunas limitaciones, por ejemplo:
    a. Se ejecuta en tiempo de ejecución dentro de un intérprete (motor del navegador, cómo V8 en Chrome)
    b. La comprobación no es tan estricta como en lenguajes compilados, por lo que los errores pueden aparecer durante la ejecución.
    c. Herramientas como linters y entornos de prueba ayudan a verificar el código.
3. Relativamente, es confiable en el sentido de que está probado y funciona en millones de aplicaciones web. Sin embargo, su tipado dinámico puede causar errores inesperados (Ejemplos clásicos: [ ] + { } // "[object Object]"). Su confiabilidad depende mucho de las buenas prácticas y el uso de estándares modernos (ES6+).
4. Parcialmente, porque tiene ortogonalidad en cuanto a que estructuras básicas (condicionales, bucles, funciones) se pueden combinar de forma libre y consistente. PERO no es completamente ortogonal ya que ciertas operaciones funcionan distinto según el contexto (ejemplo: coerción implícita de tipos, `==` vs. `===`).
5. Es consistente en el uso de bloques { } y estructuras de control clásicas y uniformidad en la ejecución secuencial del código. Pero a su vez, tiene poca uniformidad en el manejo de tipos por la coerción automática. Las mejoras de consistencia se dieron a partir de ES6 con let, const, === y strict mode.
6. Sí, es extensible mediante librerías y frameworks. Y existen subconjuntos, como ES5, ES6; Subconjuntos de buenas prácticas como Strict Mode (“use strict”;) que acotan el comportamiento.
7. Sí, cualquier navegador moderno puede ejecutar el mismo código sin necesidad de modificadores. También es portable a entornos fuera del navegador (Node.js).