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

Agustin nuñez pererira

Ejercicio 1

Parte A: Paradigma según Kuhn

1. Generalización simbólica

JavaScript estructurado define un conjunto de reglas sintácticas y léxicas: gramática ECMAScript, tipado dinámico, delimitadores (;, {}), estructuras de control clásicas (if, for, while), tipos primitivos (Number, String, Boolean) y funciones como bloques ejecutables. Estas reglas escritas dictan cómo escribir programas correctos y cómo el motor interpreta cada token y sentencia.

2. Creencias de los profesionales

La comunidad valora de JavaScript su flexibilidad (tipado débil, funciones de primera clase), su ubiquidad en el navegador y en servidores (Node.js), y la rapidez de iteración por su naturaleza interpretada. Se considera “mejor” en proyectos web por su ecosistema (NPM), programación asíncrona sencilla (async/await), y por permitir prototipado rápido sin compilación previa.

Parte B: Ejes para elegir JavaScript

1. Sintaxis y semántica

JavaScript tiene una sintaxis bien definida en la especificación ECMAScript y una semántica documentada línea a línea. Existe documentación oficial en nodejs.org y MDN (Mozilla Developer Network), que describen todas las construcciones, tipos y comportamientos del lenguaje.

2. Comprobación del código

Aunque es dinámico, JavaScript puede validarse en tiempo de compilación limitada con linters (ESLint) y tiparse estáticamente usando TypeScript o Flow. Además, frameworks de pruebas (Jest, Mocha) permiten verificar unidades de código y asegurar comportamientos esperados.

3. Confiabilidad

La madurez de motores como V8 (Chrome/node) y la amplia comunidad garantizan estabilidad y actualizaciones regulares. Los bugs críticos suelen parchearse rápidamente, y la compatibilidad con estándares ECMAScript se implementa en todas las plataformas principales.

4. Ortogonalidad

En su forma estructurada, JavaScript mantiene principios ortogonales: pocos elementos combinables de manera consistente (variables, funciones, bloques), sin interacciones inesperadas entre estructuras de control y tipos de datos.

5. Consistencia y uniformidad

La consistencia proviene de seguir la misma sintaxis para declarar funciones y gestionar bloques de código. La uniformidad se refuerza por la especificación ECMAScript que establece reglas claras para la resolución de alcance, coerción de tipos y orden de evaluación.

6. Extensibilidad y subconjuntos

JavaScript es altamente extensible mediante prototipos y permite crear DSLs (domain-specific languages) en tiempo de ejecución. Además, existen subconjuntos formales como TypeScript (superset tipado) o versiones reducidas (ES5 «strict mode»), que limitan el lenguaje para mayor seguridad.

7. Portabilidad

El código JavaScript es transportable: funciona en cualquier navegador moderno y en servidores con Node.js sin cambios. Herramientas como Babel permiten transpilar nuevas características a versiones antiguas, garantizando compatibilidad en entornos diversos.