Aplicando lo aprendido 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?

- Identificadores: nombre de variable, función, clase, etc. Debe comenzar con una letra, guion bajo o un signo de dólar (`$`). El lenguaje distingue entre mayúsculas y minúsculas.
- Palabras reservadas: `if, else, for, while, class, return, const, let, var, function`.
- Tipos de datos:
  - Primitivos: `number, string, boolean, undefined, null, bigint, symbol`.
  - Objetos: se restringe el uso a lo mínimo para mantener el paradigma estructurado.
- Operadores:
  - Aritméticos: `+, -, *, /, %`
  - Comparativos: `==, ===, !=, !==, <, >, <=, >=`
  - Lógicos: `&&, ||, !`
  - De asignación: `=, +=, -=`, etc.
- Control de flujos:
  - Los bloques de código se encierran entre `{ ... }`.
  - Las estructuras de control (`if, while, for, switch, try...catch`) deben seguir su sintaxis establecida.
- Funciones: bloques de código reutilizables definidos con `function nombre(parámetros){...}` .
- Variables: se declaran con `var` , `let` o `const`.
- Comentarios:
  - Una línea: `// comentario`
  - Multilínea: `/* comentario */`

2. Creencias de los profesionales: ¿Qué características particulares del lenguaje se cree que sean "mejores" que en otros lenguajes?

- Universalidad: JavaScript es el lenguaje nativo de los navegadores, lo que le da una posición única: cualquier dispositivo con navegador puede ejecutarlo sin instalación extra.
- Flexibilidad: No exige tipado estricto y permite modificar estructuras en tiempo de ejecución, lo cual facilita prototipar y adaptarse rápido.
- Facilidad: Es sencillo empezar a programar con él (solo se necesita un navegador), pero al mismo tiempo permite proyectos grandes y complejos.
- Multiparadigma: Soporta estilos imperativos, funcionales y orientados a objetos, lo que da libertad al programador.
- Integración inmediata: se cree mejor que otros porque se combina naturalmente con HTML y CSS para desarrollo frontEnd.
- Ejecución inmediata: al no requerir compilación, se pueden probar cambios rápidamente.

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 y responde:

1. ¿Tiene una sintaxis y una semántica bien definida? ¿Existe documentación oficial?

   JavaScript sí tiene una sintaxis y una semántica bien definida, porque todo el lenguaje está formalmente estandarizado en el documento ECMA-262, que describe ECMAScript, la especificación oficial del lenguaje.

   - Sintaxis: por ejemplo, que un bloque se encierre entre `{ }` o que un `if` se escriba como `if (condición) { ... }`.
   - Semántica: define el significado de esas estructuras, es decir, cómo las interpreta el motor de JavaScript.
2. ¿Es posible comprobar el código producido en ese lenguaje?

   Sí, es posible comprobar el código en JavaScript. El propio intérprete (como los motores de los navegadores o Node.js) valida la sintaxis y lanza errores cuando el código no respeta las reglas del lenguaje. Sin embargo, al ser un lenguaje dinámico, muchos errores se detectan recién en tiempo de ejecución (por ejemplo, variables no definidas o usos incorrectos de tipos).

3. ¿Es confiable?

   JavaScript es confiable como lenguaje porque está estandarizado y soportado globalmente, pero su naturaleza dinámica exige usar buenas prácticas y herramientas extra para aumentar la seguridad del código en proyectos grandes.

4. ¿Es ortogonal?

   JavaScript no es completamente ortogonal, ya que algunas de sus características interactúan de manera inesperada. Por ejemplo, la comparación `==` realiza coerción de tipos automática, mientras que `===` no, y las declaraciones `var`, `let` y `const` tienen reglas de alcance diferentes. Sin embargo, la mayoría de sus estructuras básicas, como las funciones, los bloques y las estructuras de control, se combinan de manera consistente y predecible.

5. ¿Cuáles son sus características de consistencia y uniformidad?
   - Consistencia:
     - Estructuras de control: `if`, `for`, `while`, `switch` funcionan de forma coherente.
     - Funciones y ámbito: las funciones tienen reglas claras de parámetros, retorno y alcance de variables según su tipo.
     - Tipos de datos primitivos: `number`, `string`, `boolean`, `null`, `undefined`, `bigint`, `symbol` operan de manera predecible en operaciones y comparaciones.
   - Uniformidad:
     - Sintaxis uniforme: declarar variables, funciones, objetos y clases sigue reglas definidas por ECMAScript.
     - Eventos y asincronía: callbacks, promesas y `async/await` siguen el mismo modelo basado en el *event loop*.
6. ¿Es extensible? ¿Hay subconjuntos de ese lenguaje?

   JavaScript es extensible porque se pueden agregar nuevas funcionalidades mediante librerías, frameworks y prototipos.

   - Subconjuntos:
     - TypeScript: un superconjunto que añade tipado estático y validaciones antes de ejecutar el código.
     - ECMAScript “estricto” (strict mode): un subconjunto de JavaScript que aplica reglas más estrictas para evitar errores comunes (`"use strict";`).
7. El código producido, ¿es transportable?

   El código JavaScript es altamente transportable, ya que puede ejecutarse en cualquier navegador moderno y en servidores mediante Node.js, lo que permite usar el mismo programa en distintos sistemas operativos sin cambios. Sin embargo, algunas funciones dependen del entorno: por ejemplo, la manipulación del DOM solo funciona en navegadores, y módulos de Node.js, como `fs`, solo funcionan en servidores. A pesar de estas limitaciones, JavaScript sigue siendo muy versátil y transportable entre diferentes plataformas y entornos.