Ejercicio 1

Parte A – Paradigma según Kuhn

1. Generalización simbólica (reglas del lenguaje)

- JavaScript permite definir instrucciones mediante **sentencias**, **expresiones** y **bloques** `{ ... }`.
- Las reglas básicas incluyen:
    - Variables: `var`, `let`, `const`
    - Estructuras de control: `if`, `else`, `switch`, `for`, `while`, `do-while`
    - Funciones como unidades de organización de código: `function nombre(params) { ... }`
- Se basa en **tipos dinámicos** y **coerción automática de tipos**.
- Todo código debe respetar la **sintaxis de bloques y terminación de instrucciones** (aunque el `;` es opcional en muchos casos).

> En otras palabras: las reglas son la sintaxis y la semántica definidas por el estándar ECMAScript.

2. Creencias de los profesionales (qué se considera “mejor”)

- JavaScript es **muy flexible y dinámico**, lo que permite escribir código rápido y adaptativo.
- Se valora su **portabilidad**: funciona en cualquier navegador moderno y en servidores con Node.js.
- Es **fácil de aprender** en comparación con lenguajes estructurados más estrictos como C.
- Tiene **una gran comunidad** y múltiples librerías que amplían sus capacidades.

- Su modelo de ejecución asíncrono (`callbacks`, `promises`, `async/await`) se considera muy útil, aunque aquí no usamos paradigma orientado a eventos.

Parte B – Elección de lenguaje

1. ¿Sintaxis y semántica bien definidas?

- Sí, la sintaxis y semántica están definidas por el estándar ECMAScript (ECMA-262).

2. ¿Es posible comprobar el código?

- Sí, mediante **interpretación** en navegadores o Node.js, y **herramientas de linting** (`eslint`) para detectar errores antes de ejecutar.

3. ¿Es confiable?

- Sí, el código es confiable dentro del contexto de su motor (V8, SpiderMonkey, etc.), aunque al ser dinámico puede generar errores en tiempo de ejecución si no se valida correctamente.

4. ¿Es ortogonal?

- Parcialmente. Las estructuras básicas (`if`, `for`, `while`) se combinan libremente, pero ciertas características del lenguaje (como coerción de tipos y hoisting) pueden romper la ortogonalidad estricta.

5. Consistencia y uniformidad

- La sintaxis es consistente y uniforme en la mayoría de los constructos básicos: bloques, funciones, operaciones aritméticas y lógicas.

- Algunos detalles históricos pueden generar inconsistencias (como `==` vs `===`).

6. Extensible / subconjuntos

- Sí, se puede extender mediante librerías y frameworks (p. ej., jQuery, Lodash).

- Existen subconjuntos o “dialectos” para propósitos específicos, como **TypeScript** o **ES5/ES6 limitado**.

7. Transportabilidad

- Muy alta. El mismo código JavaScript puede ejecutarse en **cualquier navegador moderno** y en **servidores con Node.js**, garantizando portabilidad multiplataforma.