**Aplicando lo aprendido 1**

**Paradigma de la programación.**

**Ejercicio 1**

**Parte A**

1. Generalización simbólica: ¿Cuáles son las reglas escritas del lenguaje?  
   Las reglas escritas del lenguaje se encuentran en la especificación ECMAScript, que define: sintaxis, semántica, estructura de control, nomas de tipado y manejo de errores.  
    Es decir su generalización simbólica esta basada por las normas de ECMAScript y su adaptación por los navegadores y entornos como Node.js.
2. Creencias de los profesionales: ¿Qué características particulares del lenguaje se cree que sean "mejores" que en otros lenguajes?  
    Los programadores suene considerar que JavaScript tiene ventajas respecto a otros lenguajes en que:  
    Es universal: corre en todos los navegadores sin instalación extra.  
    Es flexible y expresivo: permite escribir de forma sencilla desde scripts pequeños hasta aplicaciones completas.  
    Prototipado rápido: Es muy rápido para desarrollar y probar ideas.  
    Comunidad enorme y ecosistema rico: biblioteca y frameworks (React, vue, Angular) que potencian el desarrollo.  
   Ejecución en cliente y servidor (gracias a Node.js), lo que permite usar un mismo lenguaje en todo el stack.

**Parte B**

1. ¿Tiene una sintaxis y una semántica bien definida? ¿Existe documentación oficial?  
    Si, ECMAScript define formalmente la sintaxis y semántica del lenguaje. Y la documentación oficial disponible está en ECMA-262
2. ¿Es posible comprobar el código producido en ese lenguaje?  
    Si, puede ejecutarse en cualquier navegador, en Node.js o en otros entornos de prueba.
3. ¿Es confiable?  
   Es confiable una parte porque su comportamiento esta estandarizado y es predecible según ECMAScript pero, aun así, el tipado dinámico puede provocar errores en tiempo de ejecución si no se validan bien los datos.
4. ¿Es ortogonal?  
   No es completamente ortogonal, porque, aunque permite muchas combinaciones de características, hay excepciones y comportamientos inesperados que rompen independencia de los componentes.
5. ¿Cuáles son sus características de consistencia y uniformidad?  
    En JavaScript, la consistencia y uniformidad son moderadas:  
    Consistente en sus estructuras de control (if, for, while) y en la forma en las que funciones pueden devolver cualquier valor.  
    Poco uniforma por tener varias formas de declarar variables (Var, let, const), dos tipos de comparación (==, ===) y coerción de tipos que produce resultados distintos según el contexto.
6. ¿Es extensible? ¿Hay subconjuntos de ese lenguaje?  
    Si, es extensible gracias a sus bibliotecas, frameworks y posibilidad de usar APIs externas.  
   Existe “JavaScript estricto” que es un subconjunto con reglas mas estrictas para evitar errores comunes.
7. El código producido, ¿es transportable?  
   Si, es portable, cualquier navegador moderno o motor compatible con ECMAScript puede ejecutar el mismo código. Tambien puede correr en el servidor (Node.js) o incluso en dispositivo loT.

Glosario.

ECMAScript: Es el lenguaje base y las reglas que siguen los navegadores y motores (como V8 de Chrome) para ejecutar JavaScript.

Node.js: Entorno de ejecución que permite correr JavaScript fuera del navegadore, por ejemplo, como lo hacemos en la materia.

Frameworks: Herramientas y estructuras predefinidas que facilitan el desarrollo de aplicaciones, ofreciendo patrones y funcionalidades ya listas para usar.  
  
 React: Librería para crear interfaces de usuario basada en componentes.  
  
 Vue: Framework presivo para desarrollar interfaces interactivas.  
  
 Angular: Framework completo para construir aplicaciones web escalables.  
  
 Stack: Conjunto de tecnologías, lenguajes y herramientas que se usan juntas para desarrollar una aplicación completa.  
  
 ECMA-262: Es la especificación oficial de ECMAScript, que define como debe comportarse JavaScript, incluyendo su sintaxis, tipo de datos y reglas de ejecución.  
  
 Coercion**:** significa convertir un valor de un tipo a otro de forma automática o implícita.

API (Application Programming Interface - Interfaz de Programación de Aplicaciones): Una API es un conjunto de reglas y funcionamiento que permite que dos programas o sistemas se comuniquen entre si.

IoT (Internet of Things - Internet de las Cosas): El loT es el concepto de conectar objetos físicos a internet para que puedan enviar y recibir datos.