APLICANDO LO APRENDIDO 1

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?
3. Valores: ¿Qué pensamiento o estilo de programación consideraron mejor los creadores
4. Ejemplares: ¿Qué clase de problemas pueden resolverse más fácilmente en el lenguaje?

RESPUESTA

1) Las reglas escritas de JavaScript en base a la programación estructurada se centran en el uso de estructuras de control clásicas como condicionales y bucles, el uso de funciones que modularizan y reutilizan partes del código, para separar los procesos estructuradamente.

   Las variables creadas tienen un alcance o ámbito de vida dependiendo en qué parte del código sean declaradas (en una función, en un bloque, o globalmente). Además, cada sentencia en este paradigma ejecutará javascript en el orden en el que son escritas, salvo que se necesite bifurcar a una función y retomar el flujo una vez esa misma haya sido utilizada.

2) Los profesionales que trabajan con JavaScript suelen destacar las siguientes características como mejores frente a otros lenguajes:

- **Flexibilidad**: Se cree que JavaScript es más flexible y dinámico que otros lenguajes debido a su tipado débil y la capacidad de escribir código rápidamente sin una compilación estricta. Esta flexibilidad puede ser vista como una ventaja para desarrolladores que desean prototipar y modificar su código de manera ágil.
- **Portabilidad**: Otra creencia común es que JavaScript es más portable, ya que es el lenguaje principal de la web y puede ejecutarse en cualquier navegador sin necesidad de instalar un compilador o un entorno adicional.
- **Integración con la web**: Se considera que JavaScript es el mejor para programación estructurada en aplicaciones web, ya que fue diseñado con ese propósito y ofrece una integración nativa con tecnologías como HTML y CSS.

3) El estilo de programación considerado mejor por los programadores es dinámico, donde las variables pueden cambiar de tipo, las funciones pueden ser anónimas o creadas al vuelo, y los objetos pueden ser modificados en tiempo de ejecución.

4) JavaScript es excelente para resolver problemas relacionados con la creación de sitios web dinámicos e interactivos. La manipulación del DOM (Document Object Model) y la integración de eventos permiten una interactividad rica en aplicaciones web.

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?

RESPUESTA

1) JavaScript tiene una sintaxis bien definida basada en estándares internacionales, principalmente ECMAScript. La sintaxis sigue reglas claras para estructuras de control, definiciones de variables, funciones, y más. Aunque es flexible (por ejemplo, permite no declarar variables en ciertos contextos), esto sigue reglas específicas. Existe una documentación oficial proporcionada por organismos como Mozilla (MDN Web Docs) y el comité ECMAScript. Estos recursos ofrecen guías, especificaciones y ejemplos detallados sobre cómo usar el lenguaje.

2) Aunque JavaScript no es un lenguaje compilado en su forma tradicional, existen herramientas para verificar y validar el código.

3) Es confiable en el sentido de que es uno de los lenguajes más utilizados del mundo, respaldado por grandes comunidades y corporaciones. Sin embargo, debido a su tipado débil y su naturaleza interpretada, pueden surgir errores difíciles de rastrear en tiempos de ejecución si no se sigue una disciplina de desarrollo adecuada.

4) Tiene un nivel medio de ortogonalidad. Aunque los elementos del lenguaje están bien definidos y combinan adecuadamente (por ejemplo, funciones y objetos), algunas características como el tratamiento inconsistente de tipos (tipado dinámico) pueden generar comportamientos inesperados. Además, la sobrecarga implícita que ocurre al convertir tipos en operaciones puede reducir la ortogonalidad del lenguaje.

5) Consistencia: JavaScript es razonablemente consistente, aunque hay ciertas características que pueden confundir. Por ejemplo:

- El manejo de `null` y `undefined` puede ser inconsistente o difícil de entender, ya que ambos representan "ausencia de valor" pero tienen comportamientos distintos.
- El uso de `var`, `let`, y `const` para la declaración de variables puede introducir inconsistencias si no se usan correctamente, debido a diferencias en el alcance y el hoisting (Hoisting es un término para describir que las declaraciones de variables y funciones son desplazadas a la parte superior del scope más cercano, scope global o de función. Esto sucede solamente con las declaraciones y no con las asignaciones).

Uniformidad: Con el tiempo, JavaScript ha mejorado en términos de uniformidad gracias a las versiones más recientes de ECMAScript (por ejemplo, ES6). Estas actualizaciones introducen formas más consistentes de definir funciones (funciones de flecha), bloques de código y el manejo de clases.

6) JavaScript es altamente extensible. Se pueden crear librerías y módulos que extienden sus funcionalidades básicas. Además, el ecosistema de Node.js y los paquetes disponibles en npm permiten agregar nuevas capacidades fácilmente. Existen subconjuntos o variaciones de JavaScript. Un ejemplo es TypeScript, que es un superconjunto de JavaScript que añade tipado estático.

7) JavaScript es extremadamente transportable. Es compatible con cualquier navegador moderno sin importar el sistema operativo, lo que lo hace ideal para aplicaciones web. Además, con la aparición de entornos como **Node.js**, también puede ejecutarse en servidores, aumentando su portabilidad más allá del navegador.