**Rúbrica – Misión 2 (JavaScript y Git avanzado)**

**1. Estructura de carpetas y archivos**

* **Descripción:**  
  El proyecto debe tener una estructura clara, con archivos JavaScript externos dentro de una carpeta /js. El HTML debe estar en la raíz y organizado con los recursos correctamente vinculados (<script src="js/app.js"></script>).
* **Evidencia:**  
  Captura de pantalla del árbol de carpetas en VS Code mostrando la organización.

**2. Integración correcta de JavaScript**

* **Descripción:**  
  El archivo JavaScript debe estar enlazado desde el HTML, preferiblemente antes del cierre de la etiqueta </body> para cargarlo correctamente.
* **Evidencia:**  
  Captura de pantalla del HTML mostrando la línea <script src="js/app.js"></script>.

**3. Código limpio y comentado**

* **Descripción:**  
  El código JavaScript debe estar ordenado, con comentarios claros y buena indentación.  
  Se deben evitar variables innecesarias o código duplicado.
* **Evidencia:**  
  Captura de fragmentos del código mostrando comentarios y estilo limpio.

**4. Uso de Git para control de versiones**

* **Descripción:**  
  Se debe seguir usando Git como en la misión anterior, pero ahora con más commits reflejando el progreso del JavaScript.
* **Evidencia:**  
  Captura de historial de commits en GitHub, mostrando mensajes descriptivos.

**5. GitHub Pages actualizado**

* **Descripción:**  
  El proyecto debe estar desplegado en GitHub Pages para poder probar la funcionalidad de JavaScript en un entorno real.
* **Evidencia:**  
  Captura de la página funcionando en GitHub Pages junto con la URL pública. Adjuntar también captura del menú actualizado.

**6. Variables y tipos de datos**

* **Descripción:**  
  Uso correcto de var, let y const.  
  El código debe incluir ejemplos con diferentes tipos de datos: números, cadenas, booleanos, arreglos y objetos.
* **Evidencia:**  
  Captura de pantalla del código donde se ve la declaración y uso de estas variables.

**7. Operadores y condiciones**

* **Descripción:**  
  Uso de operadores aritméticos, de comparación y lógicos en ejemplos prácticos.  
  Implementación de estructuras de control como if, else if, else y/o switch.
* **Evidencia:**  
  Captura de código usando condiciones para ejecutar distintas acciones.

**8. Funciones en JavaScript**

* **Descripción:**  
  Uso de funciones tradicionales (function nombreFuncion()) y funciones flecha (const nombreFuncion = () => {}).  
  Cada función debe cumplir una tarea específica, por ejemplo: agregar o eliminar elementos.
* **Evidencia:**  
  Captura de código mostrando ambas formas de crear funciones.

**9. Manipulación del DOM**

* **Descripción:**  
  Demostrar que el estudiante puede seleccionar y modificar elementos del DOM con métodos como:
  + getElementById()
  + querySelector()
  + createElement()
  + appendChild()
  + remove()
* **Evidencia:**  
  Captura de código y de la página mostrando el cambio en tiempo real.

**10. Manejo de eventos**

* **Descripción:**  
  Uso de eventos como click, keydown o submit, implementados con addEventListener.  
  Ejemplo: un botón que agregue un elemento a la lista o un input que detecte la tecla Enter.
* **Evidencia:**  
  Captura del código y ejemplo funcionando en la página.

**11. Proyecto funcional: Lista de tareas (To-Do List)**

* **Descripción:**  
  Crear una aplicación básica que permita:
  + Agregar tareas desde un input.
  + Eliminar tareas con un botón (ícono de basura).
  + Limpiar el campo automáticamente después de agregar.
  + Soporte para agregar tarea con tecla Enter.
* **Evidencia:**  
  Captura del código y de la lista funcionando en el navegador.

**12. Uso de librerías externas**

* **Descripción:**  
  Integrar una librería externa como **SweetAlert** o **Font Awesome** para mejorar la experiencia visual (alertas o íconos).
* **Evidencia:**  
  Captura del HTML mostrando la importación de la librería y su uso en el proyecto.

**13. Proyecto funcional: Formulario**

* **Descripción:**  
  Crear un formulario que valide y capture los datos digitados en pantalla.
* **Evidencia:**  
  Captura del código y del form funcionando en el navegador, también captura del sweetalert con la información del usuario y también impresa en pantalla.

**14. Proyecto funcional: Validar y mostrar password versión 2.**

* **Descripción:**  
  Crear cajas de texto que valide los password y garantice que sean iguales y capture los datos digitados en pantalla.
* **Evidencia:**  
  Captura del código y del ejercicio funcionando en el navegador, también captura del sweetalert con las validaciones realizadas..