PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1) IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- **Denominación del Programa de Formación**: Tecnología en Análisis y Desarrollo de Software
- **Código del Programa de Formación:** 2824003
- **Nombre del Proyecto:** Construcción de software integrador de tecnologías orientadas a servicios
- **Fase del Proyecto:** Ejecución
- **Actividad de Proyecto:** Analizar los conocimientos de la programación asíncrona y bajo el lenguaje de programación JavaScript.
- **Competencia:** DESARROLLAR LA SOLUCIÓN DE SOFTWARE DE ACUERDO CON EL DISEÑO Y METODOLOGÍAS DE DESARROLLO
- **Resultados de Aprendizaje Alcanzar:** CREAR COMPONENTES FRONT-END DEL SOFTWARE DE ACUERDO CON LAS NECESIDADES DEL CLIENTE.
- **Duración de la Guía:** 50 Horas

2) PRESENTACIÓN

Aprendiz SENA:

Como cualquier otro lenguaje de programación, JavaScript tiene algunas características especiales: sintaxis, modelo de objetos, etc. Claramente, cualquier cosa que diferencia un lenguaje de otro. Además, descubrirás rápidamente que JavaScript es un lenguaje relativamente especial en su acercamiento a las cosas. Esta parte es esencial para cualquier principiante de programación e incluso para aquellos que ya conocen un lenguaje de programación debido a que las diferencias con otros lenguajes de programación son numerosas.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Claramente JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

3) FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

A. Actividades de reflexión inicial.

- Desarrollo de la actividad tipo investigación en la cual el aprendiz debe consultar en internet los siguientes interrogantes y dar un ejemplo de cada uno:
- Programación asíncrona:
    - Que son los temporizadores setimeout y setinterval **(de un ejemplo)**
    - Que es la asincronía y el evento loop **(de un ejemplo)**
    - Que son los callbacks **(de un ejemplo)**
    - Que son las promesas **(de un ejemplo)**
    - Que son las funciones asíncronas (async / await) **(de un ejemplo)**

Duración de la actividad 1 hora.

B. Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje.

Discusión en grupo.

- Se discutirá con el grupo el significado de los tipos de datos, estructuras de control objetos y funciones de lenguaje JavaScript.
- Se realizarán preguntas para definir el conocimiento previo y se despejarán las dudas que tengas respecto al diseño web.
- Se realizará una profundización de la investigación con diferentes ejemplos relacionados en la búsqueda y ejemplos propuestos en el momento de la socialización.

Duración de la actividad 4 hora.

C. Actividades de apropiación del conocimiento.

En esta sección trabajaremos una actividad

En este espacio de trabajo se busca identificar e iniciar el aprendizaje en los elementos fundamentales que componen el pensamiento lógico de la programación y su estructura, permitiendo plantear conceptos y luego incorporarlos en ejercicios prácticos, realizando los siguientes algoritmos:

Antes de iniciar a codificar una solución a los enunciados:

a) Cree un repositorio local
b) Cree un archivo index.html
c) Cree una lista de navegación con los ejercicios a solucionar (esta lista debe tener un diseño agradable, aplicar estilos css)
d) Agregue los cambios al staging área
e) Confirme los cambios, el nombre de la confirmación puede ser “Inicio del taller”
f) Cree una rama con el nombre desarrollo
g) Ubíquese en la nueva rama
h) Cree el primer archivo ejercicio1.html y su archivo ejercicio1.js (todos los ejercicios deben estar separados por archivos y se debe poder navegar por los diferentes vínculos del html)
i) Agregue los cambios al staging área
j) Confirme los cambios, el nombre de la confirmación puede ser “primer ejercicio”
k) Trabaje sobre la rama desarrollo todos los ejercicios
l) Al finalizar los ejercicios tome una captura de la terminal con el log de confirmaciones
m) Ubíquese en la rama main
n) Realice el merge de la rama desarrollo
o) Tome una captura de la terminal donde se logre evidenciar que elimino la rama desarrollo
p) Cree un repositorio remoto con el nombre introducción a javascript
q) Agregue el origen a su repositorio local
r) Realice un push a la rama main
s) Cree una página con la solución de los ejercicios
t) Descargue el repositorio y súbalo a territorium o en su defecto lo envía por correo al instructor (territorium podría estar fuera de servicio)
u) Envie el link para su validación

1. Escriba una función imprimirNumeros(desde, hasta) que genere un número cada segundo, comenzando desde desde y terminando con hasta.

   Haz dos variantes de la solución describiendo el paso a paso de cada una de ellas:

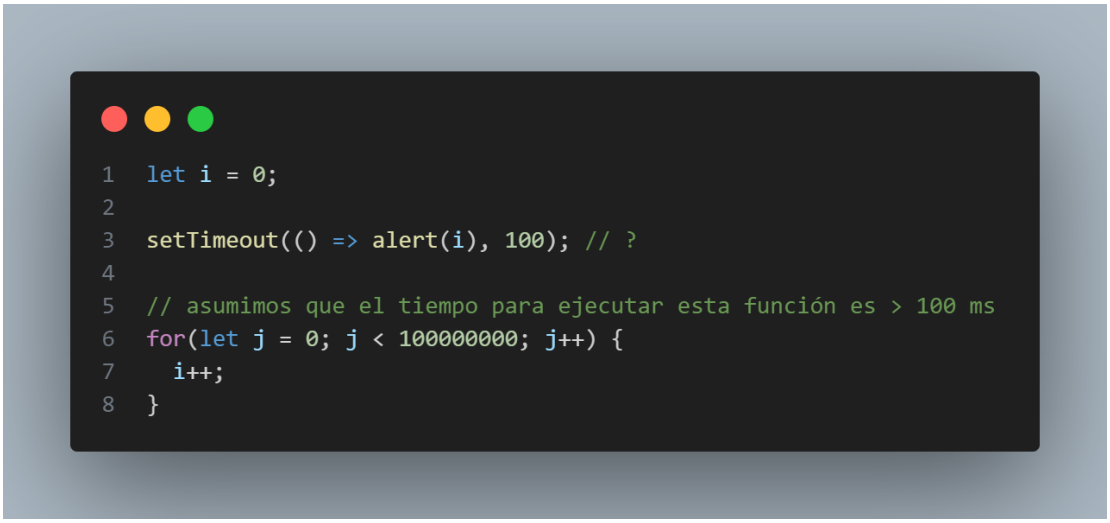
   - Usando setInterval.
   - Usando setTimeout anidado.

2. En el siguiente código hay una llamada programada setTimeout, luego se ejecuta un cálculo pesado que demora más de 100 ms en finalizar.

   ¿Cuándo se ejecutará la función programada y por qué se ejecutará?

   a) Después del bucle.
   b) Antes del bucle.
   c) Al comienzo del bucle.

   ¿Qué va a mostrar alert()?



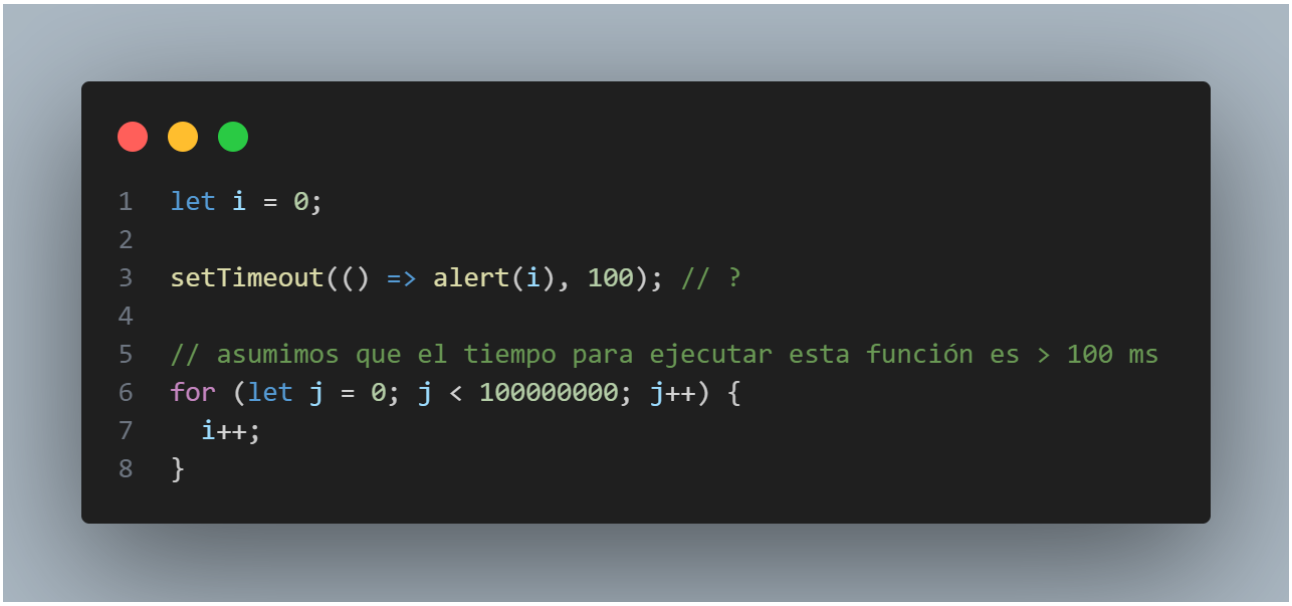
```
let i = 0;

setTimeout(() => alert(i), 100); // ?

// asumimos que el tiempo para ejecutar esta función es > 100 ms
for(let j = 0; j < 100000000; j++) {
  i++;
}
```

3. Crear una función **map** que acepte un array y un **callback** y que:

   - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
   - obtenga el resultado
   - lo pushee a un nuevo array
   - devuelva el array final con el resultado de cada una de las llamadas al **callback**.

4. Crear una función **filter** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - sí dicho **callback** devuelve true, pushea el elemento a un nuevo array
    - devuelva el array final con los elementos que pasaron el "filtro".
5. Crear una función **every** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva true si todas las llamadas al **callback** devolvieron true
6. Crear una función **some** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva true si al menos una de las llamadas al **callback** devolvió true
7. Crear una función **find** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva el elemento pasado como argumento del primer **callback** que devuelva true
    - sí ningún **callback** devuelve true, devuelva undefined
8. Crear una función **findIndex** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva el índice del elemento pasado como argumento del primer **callback** que devuelva true
    - sí ningún **callback** devuelve true, devuelva undefined
9. Crear una función **dropWhile** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva un array con los elementos a partir del primer **callback** que devolvió false
    - (Es decir, crea un nuevo array y va recorriendo elemento por elemento, mientras el **callback** de true, no los agrega, cuando el **callback** da false por primera vez agrega todos los elementos restantes a partir de dicho elemento inclusive)

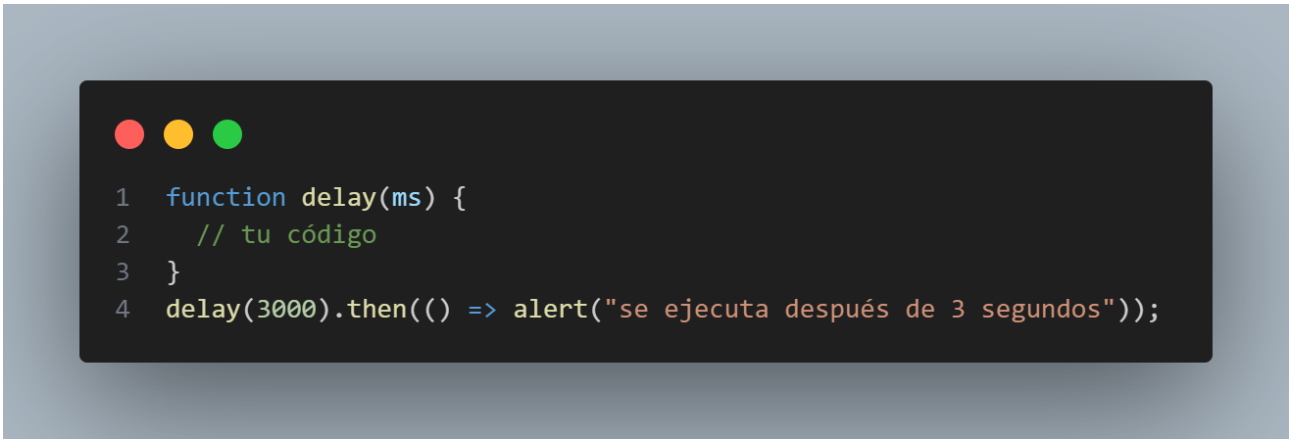
10. Crear una función **takeWhile** que acepte un array y un **callback** y que:
    - por cada elemento del array ejecute el **callback** pasándole dicho elemento como argumento
    - devuelva un array con los elementos hasta el primer **callback** que devolvió false
    - (Inverso del **dropWhile**)
11. ¿Cuál es el resultado del código a continuación?



```
let i = 0;

setTimeout(() => alert(i), 100); // ?

// asumimos que el tiempo para ejecutar esta función es > 100 ms
for (let j = 0; j < 100000000; j++) {
  i++;
}
```

12. La función incorporada setTimeout utiliza callbacks. Crea una alternativa basada en promesas.

    La función delay(ms) debería devolver una promesa. Esa promesa debería resolverse después de ms milisegundos, para que podamos agregarle. then, así:



```
function delay(ms) {
  // tu código
}
delay(3000).then(() => alert("se ejecuta después de 3 segundos"));
```

13. Crea una **Promise** que se resuelva después de 3 segundos y luego imprima "Promise resuelta" cuando se cumpla.
14. Define una función asincrónica que espere 1 segundo y luego devuelva una cadena que diga "Operación completada". Utiliza async/await.
15. Crea una Promise que se rechace después de 2 segundos y captura el error para imprimir "Error: Promise rechazada".
16. Crea tres Promises consecutivas, donde cada una se resuelva después de 1 segundo y devuelva un número diferente. Luego, encadena las tres Promises para sumar los resultados y mostrar el resultado final.
17. Realiza una llamada a un archivo local Muestra en formato json, luego muestre los datos obtenidos en la consola.
18. Después de realizar una llamada a un archivo local en formato json, utiliza el método then() para filtrar los datos y mostrar solo los elementos que cumplan ciertos criterios (por ejemplo, mostrar solo los nombres que comiencen con "A").
19. Crea una función asincrónica que realice una llamada a un archivo local en formato json y luego manipule los datos recibidos para mostrar información específica.
20. Crea tres Promises que se resuelvan después de diferentes intervalos de tiempo y luego utilice Promise.all() para mostrar un mensaje cuando todas se hayan resuelto.
21. Realiza tres Promises, algunas de las cuales se resuelvan y otras se rechacen. Utiliza Promise.allSettled() para obtener información sobre el estado de todas las Promises.
22. Crea un bucle que realice llamadas asincrónicas utilizando async/await para procesar una lista de elementos uno por uno.

23. Realiza una llamada a un archivo local en formato json y maneja posibles errores utilizando try/catch para mostrar un mensaje de error en caso de fallo.
24. Crear un objeto proxy usando la clase Proxy
25. Usar un proxy para la validación de los valores de propiedades
    - Cuando declaremos que el atributo es numérico solo perimimos que ingresen números
    - Cuando declaremos que el atributo es alfanumérico solo permitimos que ingresen letras
    - Cuando declaremos que el atributo es un email, solo permitimos que ingresen correos
    - Cuando declaremos que el atributo es de tipo date, solo permitimos que ingresen fechas
    - Para todos los atributos no se puede ingresar espacios en blanco tanto al inicio como al final de estos y se debe mostrar el error personalizado por consola

Toda la actividad se realizará con el acompañamiento del instructor, podrá realizar las preguntas requeridas al instructor para solucionar la actividad.

Duración de la actividad 8 horas.

D. Actividades de transferencia del conocimiento.

Esta actividad se validará como una actividad de evaluación del tema, para presentar esta actividad el aprendiz debe presentar la investigación junto con la solución del taller práctico después se entregarán 5 ejercicios para su solución de forma individual.

Esta actividad es individual donde se busca que el aprendiz sin apoyo del instructor mida su avance y apropiación del tema.

Duración de la actividad 13 horas.

4) ACTIVIDADES DE EVALUACIÓN

Tome como referencia la técnica e instrumentos de evaluación citados en la guía de Desarrollo Curricular

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
Evidencias de Conocimiento: **Evidencias de Desempeño:** • Evaluación(es). • Taller(es). **Evidencias de Producto:** • Código fuente • Repositorio remoto y local. • Link de publicación	Identifica y describe, en un sistema de información dado, los datos de entrada, procesamiento de los datos e información generada, según necesidades del cliente.	**Técnica:** Observación directa

5) GLOSARIO DE TÉRMINOS

- **JavaScript:** es un lenguaje de programación de alto nivel, interpretado y multiplataforma. Es uno de los tres pilares fundamentales de la web, junto con HTML (Hypertext Markup Language) y CSS (Cascading Style Sheets).
- **GIT:** sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente durante el desarrollo de software.
- **Compilación:** proceso de traducir el código fuente de un programa informático a un formato ejecutable que la computadora puede entender y ejecutar.
- **Repositorio:** Un repositorio en el contexto de la informática, especialmente en el desarrollo de software, se refiere a un lugar centralizado donde se almacenan y gestionan archivos y recursos relacionados con un proyecto específico.
- **Staging área:** El área de staging, también conocida como "área de preparación" o "índice", es una etapa intermedia en el proceso de trabajar con Git. Cuando realizas cambios en tus archivos de código fuente y estás listo para confirmar esos cambios en Git, primero los

"añades" al área de staging antes de "confirmar" definitivamente esos cambios en el historial de Git.

- **Commit:** En Git, un "commit" es una operación que registra los cambios realizados en los archivos de un repositorio en un punto específico de tiempo. Cada commit tiene un mensaje descriptivo que proporciona información sobre los cambios realizados en ese commit en particular. Los commits son la unidad básica de trabajo en Git y forman parte del historial de versiones del repositorio.
- **Branch:** En Git, una "branch" (rama) es una línea independiente de desarrollo que permite a los desarrolladores trabajar en características específicas del proyecto sin afectar el flujo de trabajo principal o "master". Cada rama en un repositorio de Git representa una versión separada del código, y los cambios realizados en una rama no afectan a otras ramas hasta que se fusionan.
- **Merge:** En Git, el comando merge se utiliza para combinar los cambios de una rama (branch) en otra. Esta operación se realiza típicamente cuando se quiere incorporar el trabajo realizado en una rama secundaria de vuelta a la rama principal (por lo general, master o main).
- **Deploy:** El término "deploy" (despliegue) en el contexto del desarrollo de software se refiere al proceso de implementar y poner en funcionamiento una aplicación o una actualización de software en un entorno de producción, donde los usuarios finales pueden interactuar con ella. Este proceso implica tomar el código fuente de la aplicación, compilarlo (si es necesario), configurar cualquier infraestructura requerida, y finalmente lanzar la aplicación para su uso público.

6) REFERENTES BILBIOGRÁFICOS

https://developer.mozilla.org/es/docs/Web/JavaScript

7) CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha

Autor (es)	**John Freddy Becerra Castellanos**	**Instructor**	**CIMI**	**24 – octubre - 2024**

8) CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	**Cargo**	**Dependencia**	**Fecha**	**Razón del Cambio**
Autor (es)					